A APPENDIX

A.1 GLOSSARY OF ACRONYMS

AL active learning
ALP active learning pipeline
AUBC area under the budget curve
DS dataset
ML machine learning
QS query strategy
SOTA state-of-the-art
DNN deep neural network
ETC extremely randomized trees
GBDT gradient-boosted decision tree
k-NN k-nearest neighbor
LR logistic regression
MLP multi-layer perceptron
NB naïve Bayes
PFN prior-fitted network
RF random forest
SVM support vector machine
XGB XGBoost
AAL adaptive active learning
ALBL active learning by learning
BALD Bayesian active learning by disagreement
CER combined error reduction
CLUE clustering uncertainty-weighted embeddings
CluMS cluster margin
CoreSet CoreSet
DWUS density weighted uncertainty sampling
EER expected error reduction
EMC expected model change
ES entropy sampling
EU epistemic uncertainty sampling
EVR expected variance reduction
FALCUN fast active learning by contrastive uncertainty
FIVR Fisher information variance reduction
GRAPH graph density
HIER hierarchical sampling
LC least-confident sampling
k-means k-means sampling
MarginDensity pre-clustering and margin sampling
MaxEnt maximum entropy
MaxER maximum error reduction
MinMS minimum margin sampling
MLI minimum loss increase
MMC maximum model change
MS margin sampling
PowBALD power-set BALD
PowMS power-set margin sampling
QBC query-by-committee
QBC VR QBC VR
QUIRE querying informative and representative examples
Rand random sampling
TypClu typical clustering
VR variance reduction

A.2 Comparison to existing benchmarks for tabular data

In the following, we present an extensive table which compares `ALPBench` with existing active learning benchmarks. The QS and learning algorithms are ordered by their year of appearance. In Table 3, we additionally present a detailed version of Table 2 in the main paper, which shows which exact QS and learners were implemented in the benchmarks.

Query Strategy	Year	Yang et al. (2018)	Zhan et al. (2021)	Bahri et al. (2022a)	Lu et al. (2023)	ALPBench
ES Shannon (1948)	1948	✓	✓	✓	✓	✓
QBC Seung et al. (1992)	1992	✗	✓	✗	✓	✓
VR Cohn (1993)	1993	✗	✓	✗	✓	✓
LC Lewis and Gale (1994)	1994	✗	✓	✓	✓	✓
FIVR Zhang (2000)	2000	✓	✗	✗	✗	✗
MS Scheffer et al. (2001)	2001	✗	✓	✓	✓	✓
EER Roy and McCallum (2001)	2001	✓	✓	✗	✓	✓
MaxER Guo and Greiner (2007)	2007	✓	✗	✗	✗	✗
CER Guo and Schuurmans (2007b)	2007b	✓	✗	✗	✗	✗
EVR Schein and Ungar (2007)	2007	✓	✗	✗	✗	✗
EMC Settles et al. (2007)	2007	✗	✓	✗	✗	✗
MLI Hoi et al. (2008)	2008	✓	✗	✗	✗	✗
BALD Houlsby et al. (2011)	2011	✗	✗	✓	✗	✓
MMC Cai et al. (2017)	2017	✓	✗	✗	✗	✗
MaxEnt Gal et al. (2017)	2017	✗	✗	✓	✗	✓
QBC VR Beluch et al. (2018)	2018	✗	✗	✓	✗	✓
EU Nguyen et al. (2019)	2019	✗	✗	✗	✗	✓
PowMS Kirsch et al. (2021)	2021	✗	✗	✓	✗	✓
MinMS Jiang and Gupta (2021)	2021	✗	✗	✓	✗	✓
k-means Kang et al. (2004)	2004	✗	✓	✗	✗	✓
HIER Dasgupta and Hsu (2008)	2008	✗	✓	✗	✓	✗
CoreSet Sener and Savarese (2018)	2018	✗	✗	✓	✓	✓
TypClu Hacohen et al. (2022)	2022	✗	✗	✓	✗	✓
MarginDensity Nguyen and Smeulders (2004)	2004	✗	✗	✓	✗	✗
DWUS Settles and Craven (2008)	2008	✗	✓	✗	✓	✗
QUIRE Huang et al. (2010)	2010	✗	✓	✗	✓	✗
GRAPH Ebert et al. (2012)	2012	✗	✓	✗	✓	✗
AAL Li and Guo (2013)	2013	✓	✗	✗	✗	✗
ALBL Hsu and Lin (2015)	2015	✗	✓	✗	✓	✗
CluMS Citovsky et al. (2021)	2021	✗	✗	✓	✗	✓
CLUE Prabhu et al. (2021)	2021	✗	✗	✗	✗	✓
FALCUN Gilhuber et al. (2024)	2024	✗	✗	✗	✗	✓

Learning Algorithm	Year	Yang et al. (2018)	Zhan et al. (2021)	Bahri et al. (2022a)	Lu et al. (2023)	ALPBench
LR Berkson (1944)	1944	✓	✗	✗	✗	✓
k-NN Fix and Hodges (1952)	1952	✗	✗	✗	✗	✓
MLP Werbos (1974)	1974	✗	✗	✓	✗	✓
NB Kononenko (1990)	1990	✗	✗	✗	✗	✓
SVM Boser et al. (1992)	1992	✗	✓	✗	✓	✓
RF Breiman (2001)	2001	✗	✗	✗	✗	✓
ETC Geurts et al. (2006)	2006	✗	✗	✗	✗	✓
XGB Chen and Guestrin (2016)	2016	✗	✗	✗	✗	✓
Catboost Dorogush et al. (2018)	2018	✗	✗	✗	✗	✓
TabNet Arik and Pfister (2021)	2021	✗	✗	✗	✗	✓
TabPFN Hollmann et al. (2023)	2023	✗	✗	✗	✗	✓

Table 3: Comparison of the scopes of `ALPBench` and previous benchmarks for tabular data.

		Yang et al. (2018)	Zhan et al. (2021)	Bahri et al. (2022a)	Lu et al. (2023)	**Ours**
QS	Info.	ES, MaxER, MMC, FIVR, EER, CER, EVR, MLI	ES, QBC, VR, LC, MS, EER, EVR	ES, LC, MS, BALD, MaxEnt, QBC VR, PowMS, MinMS, PowBALD	ES, QBC, VR, LC, MS, EER	ES, QBC, VR, LC, MS, EER, BALD, MaxEnt, QBC VR, EU, PowMS, MinMS, PowBALD
	Repr.	-	k-means, HIER	CoreSet, TypClu	HIER, CoreSet	k-means, CoreSet, TypClu
	Hybr.	AAL	DWUS, QUIRE, GRAPH, ALBL	MarginDensity, CluMS	DWUS, QUIRE, GRAPH, ALBL	CluMS, CLUE, FALCUN
Learner	Base	LR	SVM	-	SVM	k-NN, SVM, RF, LR, NB, ETC
	GBDT	-	-	-	-	CatBoost, XGB
	DNN	-	-	MLP	-	MLP, TabNet
	PFN	-	-	-	-	TabPFN
ALP	$\sum$	9	13	12	12	**209**
DS	Binary	44	35	35	26	48
	Multi	-	9	34	-	38
	$\sum$	44	44	69	26	**86**
AL Setting		1	1	3	1	5
Metrics		Accuracy	Accuracy	Accuracy	Accuracy	Accuracy, AUC, F1, Prec, Recall, Logloss

A.3 Experiments

In this section, we elaborate in more detail on the experiments that were conducted within our evaluation study.

Datasets. From the 90 datasets from the OpenML-CC18 Bischl et al. (2019) and the TabZilla Benchmark Suite McElfresh et al. (2023) we filtered and excluded the datasets with OpenML IDs 1567, 1169, 41147, and 1493. The first three were filtered out for all settings because they consist of more than 300,000 data points, which would result in a large amount of computing time for the non-info. based QSs. The last dataset with OpenML ID 1493 was filtered out since it consists of 100 classes, which would result in a huge amount of the per iteration budget $\mathcal{R}$, limiting the number of iterations to a high degree. Further, for the large setting, we wanted to guarantee that at least 10 iterations can be performed until all instances from $\mathcal{D}_U$ are queried. This led to the removal of OpenML IDs 11, 12, 14, 16, 18, 22, 25, 51, 54, 188, 307, 458, 469, 1468, 1501, 40966, and 40979 for this setting. For the preprocessing steps, we proceed as follows. Categorical features are one-hot encoded and missing values are imputed by the mean or mode of the corresponding feature.

Active Learning Setting. As mentioned, we investigate a small and a large setting. Explicitly, the small and large settings are specified by $|\mathcal{D}_L^0| = R = 5 \cdot |\mathcal{Y}|$ and $|\mathcal{D}_L^0| = R = 20 \cdot |\mathcal{Y}|$, respectively, for the given dataset and a total amount of 20 iterations or until all instances from the unlabeled pool $\mathcal{D}_U$ are queried. We choose the factor 5 for the small setting, since then $\mathcal{R}$ matches the one in the (static) small setting in Bahri et al. (2022a). For the large setting, we should have chosen a factor of 100 to be again consistent with Bahri et al. (2022a). However, this seemed unrealistic to us for real-world applications. For some (imbalanced) datasets, it may happen that not every class is at least once represented in $\mathcal{D}_L^0$. In these cases, we additionally randomly sample one instance from $\mathcal{D}_U$ per missing class and add them with their corresponding label to $\mathcal{D}_L^0$. We run each ALP ten times with different seeds, where the seed defines the $\frac{2}{3}/\frac{1}{3}$-split of the total dataset $\mathcal{D}$ into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ as well as the split of $\mathcal{D}_{\text{train}}$ into $\mathcal{D}_L$ and $\mathcal{D}_U$. Needless to say, the datasets we consider are originally (fully-)labeled datasets. Tailored to the AL setting, we discard the labels for the instances in $\mathcal{D}_U$ and assure that only the oracle $\mathcal{O}$ can access them.

Configuration of Learning Algorithms. In general, we do not perform any hyperparameter optimization (HPO) but rather stick to the default parameters. To contain computational costs, we limit the training time of the learning algorithms. For XGB and Catboost, we reduce the training time by setting the tree method to *hist* and limiting the amount of iterations, respectively. For Catboost and for TabNet, we implement a timeout of three minutes per iteration for the same purpose. This of course may decrease the performance of the learning algorithms and poses a limitation to the generalizability of our empirical study. Further, TabPFN (Hollmann et al., 2023) can so far only be fitted on a maximum amount of 1,000 instances. Therefore, we uniformly sample 1,000 instances from the current dataset to be fitted on, in case this constraint is violated, similar to McElfresh et al. (2023). For TabPFN and TabNet we modify the implementation for the representation-based and hybrid approaches. Concretely, we extract the output of the encoder from the TabPFN and the activations of the

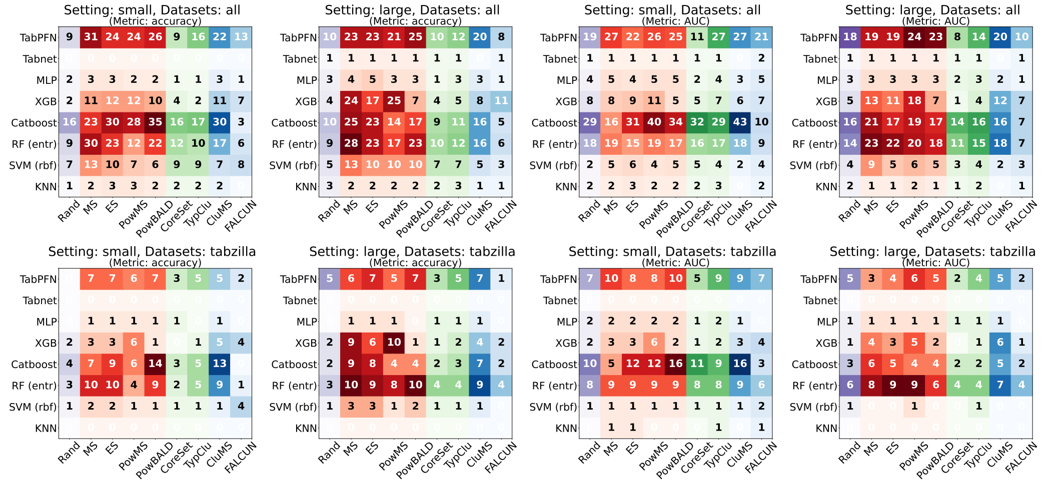

Figure 5: Heatmaps for all ALPs within our evaluation study using AUBC **(accuracy)** as performance measure (first and second column) and AUBC **(AUC)** (third and fourth), separately for **all** (first row) datasets and for the **TabZilla** (second row) datasets. Information-based, representation-based, and hybrid QSs are colored in red, green, and blue, respectively, and random sampling is in purple.

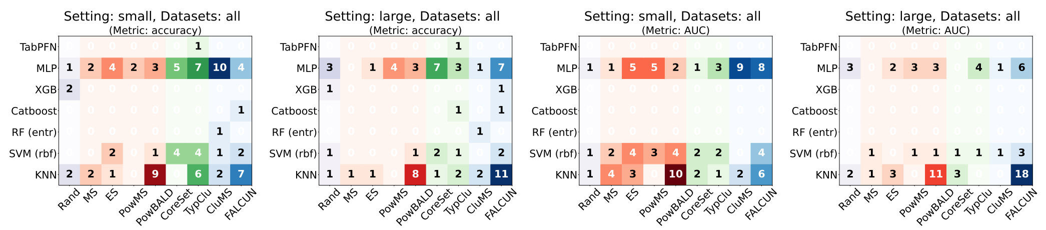

Figure 6: Lose-Heatmaps for all ALPs within our evaluation study using AUBC **(accuracy)** as performance measure (first and second subfigure) and AUBC **(AUC)** (third and fourth) on all datasets **without** statistical significance. The color-coding is consistent with Figure 5.

penultimate layer from TabNet to compute the representativeness of each instance based on its embedding. The exact details can obviously be looked up in our implementation.

Implementation. All experiments were conducted with 2 CPU cores and 8GiB RAM or 16GiB for the small and large settings, respectively, to resemble end-user environments. The HPC nodes for the computations are equipped with two AMD Milan 7763 and 256GiB main memory in total. Runs exceeding these limits have been canceled by the workload manager.

A.4 Results

This section contains more experimental results, comprising more heatmaps and win-matrices distinguishing between binary and multi-class datasets, small and large settings and different metrics. We also present more budget curves for other datasets and learners.

Precisely, we first present heatmaps where we - similar to the main paper - distinguish between small and large settings as well as both metrics AUBC (accuracy) and AUBC (AUC). However, we now compute heatmaps for all datasets (binary and multi-class combined) and for all datasets from the TabZilla Benchmark Suite McElfresh et al. (2023), cf. Figure 5 the first and the second row, respectively. The latter one is a selection of particulary hard or difficult datasets, so we suppose them to be hard for active learning as well.

The main trend of the results of all datasets looks quite similar to the binary datasets in the main paper: Most winning pipelines constitute of TabPFN, Catboost, XGB or RF as learner and information-based QS. However, CluMS is also part of many winning pipelines, especially in the small setting and Rand is quite competitive when considering AUC. For the TabZilla datasets, TabPFN and XGB appear to be not that strong. The QS k-NN and Tabnet (almost) never constitute a winning pipeline and CluMS again is competitive regarding both metrics, especially in the small setting.

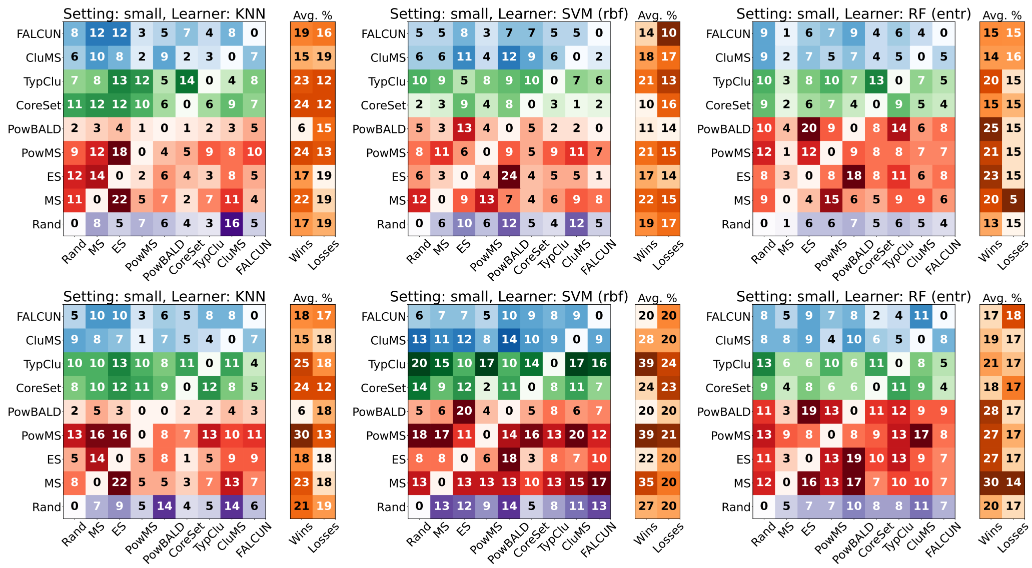

Figure 7: Win-Matrices for k-NN, SVM and RF for the **small** setting on **multi-class** datasets using AUBC **(accuracy)** as performance measure (first row) and AUBC **(AUC)** (second row).

To investigate which ALPs perform particularly poorly, we present *Lose-Heatmaps* in Figure 6, where the losing pipeline replaces ALP_d. Hereby, we do not separate between binary and multi-class datasets and further exclude TabNet as it did not perform at all in our investigated setting. We neglect statistical significance, which may seem an unusual perspective, but it helps to reveal insights into which ALPs exhibit the lowest performance for each dataset. In this figure, we find that it is more important to choose a strong learner than selecting a suitable QS. Concretely, one should avoid MLP or k-NN, and ALPs combining k-NN with PowBALD or MLP with FALCUN or CluMS proved disadvantageous. It might happen, that your learner is not strong, because you maybe want to use a very simple, interpretable model or the data is extremely difficult to learn. In this case it might not be a good idea to rely on any probabilistic estimates but rather choose Rand, as it rarely constitutes to loosing pipelines for k-NN and MLP.

In Figure 7 we present win-matrices for the learners k-NN, SVM and RF considering the small setting and evaluating on multi-class datasets. Hereby, we distinguish again between the metrics AUBC (accuracy) and AUBC (AUC). If the metric is chosen as accuracy, we make the following observations. For the k-NN the representation-based and hybrid approaches are very competitive with the information-based strategies. This effect decreases, when SVM is chosen and for the RF the information-based strategies are dominant with MS being extremely robust. In contrast to the RF, Rand is not a too bad choice for k-NN and SVM. Regarding the AUC, TypClu is quite strong for the SVM. For the RF, the information-based strategies are outperforming other QS and in particular MS is strong. Again, we see that the performance of all QSs depend on the chosen learning algorithm.

Further, we present budget curves comparing a subset of 5 different QS for enhanced visual clarity. Precisely, we chose Rand, two representatives for the information-based strategies (MS and power-set BALD (PowBALD)), and one representative for each remaining group, namely CoreSet (CoreSet) and CluMS.

For the large setting, we present budget curves for the datasets with OpenML ID 3 and 1043 in Figure 8. For both datasets, MS is a strong competitor, however CluMS seems to be very strong in the first few iterations. Rand is outperformed by all other strategies, except for the XGB on the first dataset. If the learner achieves high accuracy (as XGB and Catboost do), its probability estimates seem to be reliable and hence information-based strategies are very strong. For the dataset with ID 1043, we observe that CoreSet is initially also quite competitive. If initially the learner has not yet learned too much about the data distribution and achieves also not too good test performance (less than 0.8 accuracy), it might be advantageous to sample representative instances.

In Figure 9, we present budget curves for the datasets with OpenML ID 11 and 51, which both are included in the TabZilla benchmark suite. For the first dataset, one can see that the budget curves for the strong learners RF and TabPFN look quite smooth, especially for TabPFN and also achieve quite high accuracy. The simpler learners k-NN and MLP are struggling more and k-NN even drops in performance in the second half of the active learning procedure. The suitability of different QS again, is quite dependent on the learner: Whereas for the MLP and TabPFN the information-based strategies MS and PowBALD are outperforming the rest, they are the worst when considering k-NN and RF as learners. Regarding the dataset with ID 51, all learners have a hard time learning the data distribution, as the budget curve is very noisy and also the increases in accuracy are very marginal, except for the MLP. One can deduce, that this dataset definitely is hard for active learning.

In Figure 10, we consider the small setting and present budget curves for the dataset with OpenML ID 334. Overall, the budget curves are much less unstable, compared to the large setting. This is expected, as we start with a very small initial labeled dataset, which makes it really hard to learn the data distribution. The performance of the different QS differs quite a lot for different learners. CoreSet is very strong if the learning algorithms is chosen to be k-NN or TabPFN, whereas for both other learners, the information-based strategies are quite strong. The pipelines consisting of TabPFN as learning algorithm achieve all a much higher accuracy than the pipelines constituted of the other learners. This highlights the importance of choosing an appropriate learning algorithm for the given dataset.

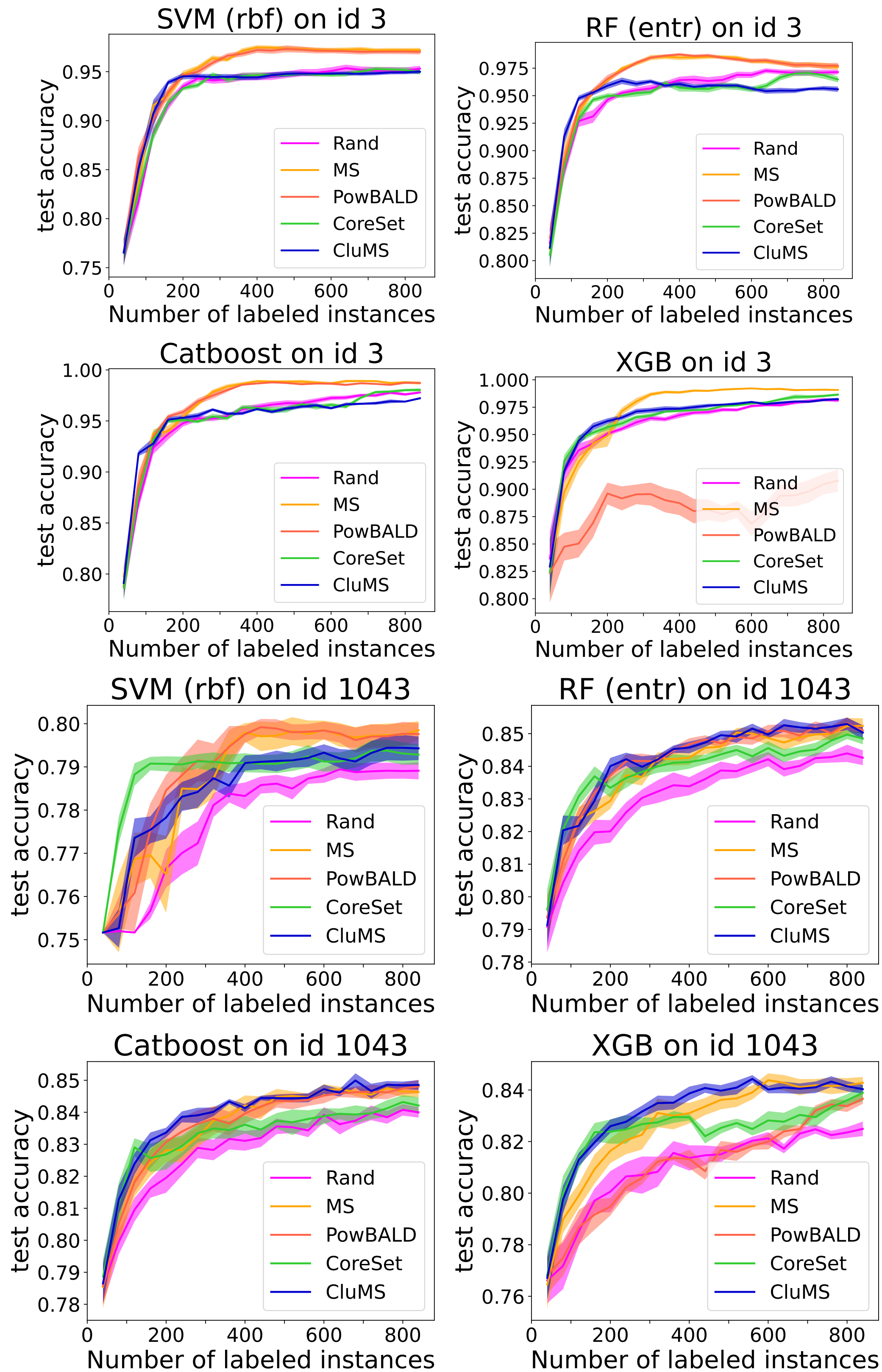

Figure 8: Budget curves for different ALPs on the dataset with OpenML ID 3 and 1043, considering the **large** setting.

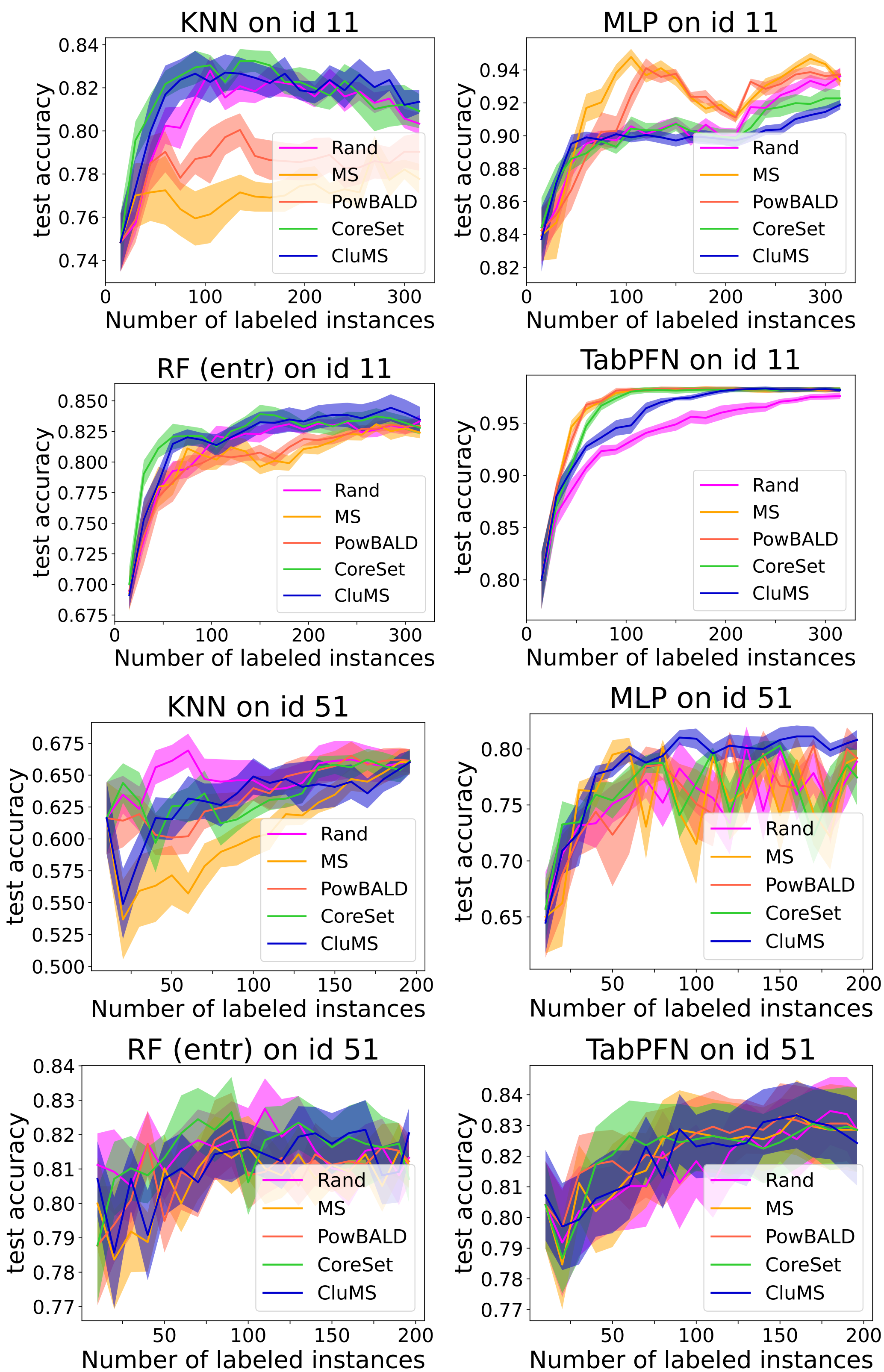

Figure 9: Budget curves for different ALPs on the dataset with OpenML ID 11 and 51, considering the **small** setting.

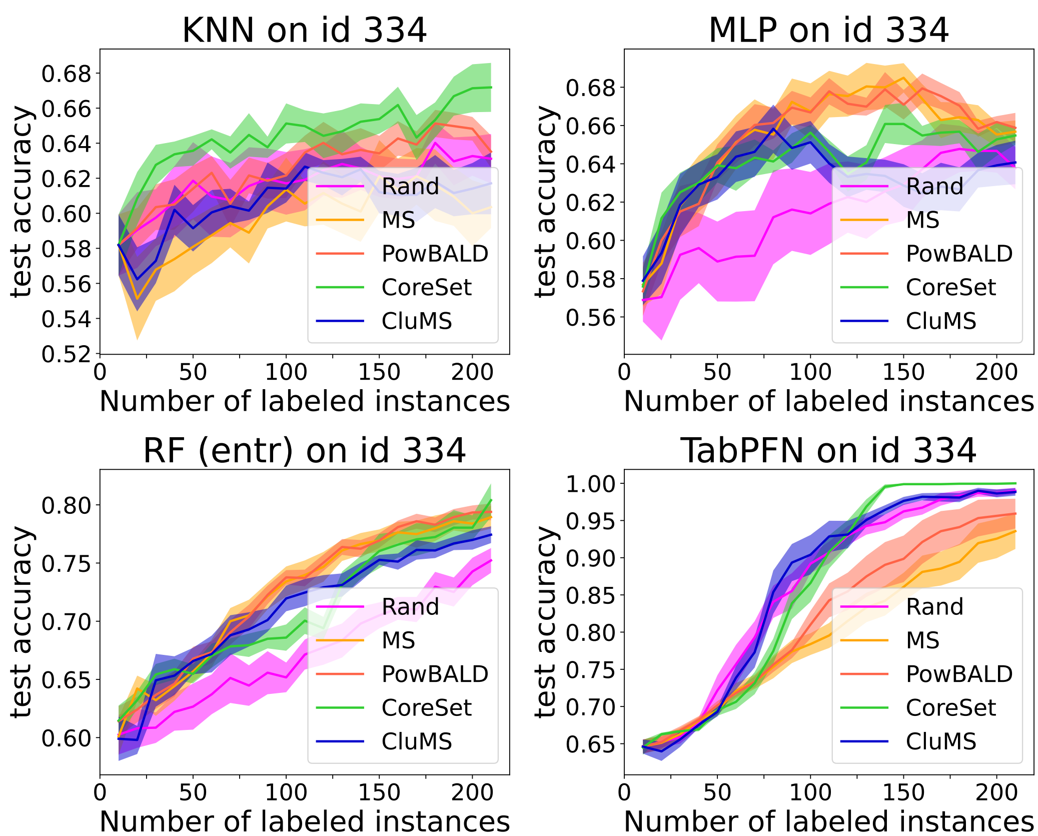

Figure 10: Budget curves for different ALPs on the dataset with OpenML ID 334, considering the **small** setting.