Update

January 6, 2021

Contents

1 Task 1

1.1 Description

Perform the same comparison as you have done for AL and non-AL by including the other down-sampling or up-sampling methods you have suggested, so we can see whether AL would also perform better than other sampling technologies.

1.2 Solution

I have chosen four sampling strategies from imbalanced-learn and RandomForestClassifier:

- ADASYN
- SMOTE
- CondensedNearestNeighbor
- InstanceHardnessThreshold

I compared every sampling strategy from the list above with uncertainty sampling from modAL (run 10 fold cross-validation and calculated t-test stats for AUC lower boundary (AUC LB), AUC, AUC upper boundary (AUC UB), Accuracy, F1, MCC).

All performance metrics for models trained using uncertainty sampling are significantly higher than for models trained using SMOTE. Please see Figure 1 for radar charts with visualization (also, see table "t-test stats.csv" in "./results/SMOTE")

All performance metrics for models trained using uncertainty sampling are significantly higher than for models trained using ADASYN except for F1 (0.533 ADASYN vs. 0.566 uncertainty sampling, but the difference is not statistically significant). Please see Figure 1 for radar charts with visualization (also, see table "t-test stats.csv" in "./results/ADASYN")

All performance metrics for models trained using uncertainty_sampling are significantly higher than for models trained using CondensedNearestNeighbour except for F1 (0.567 CondensedNearestNeighbour vs. 0.565 uncertainty sampling). Please see Figure 1 for radar charts with visualization (also, see table "t-test_stats.csv" in "./results/CondensedNearestNeighbour")

Only Accuracy and MCC for models trained using uncertainty_sampling are significantly higher than for models trained using InstanceHardnessThreshold. Please see Figure 1 for radar charts with visualization (also, see table "t-test_stats.csv" in "./results/InstanceHardnessThreshold")

2 Task 2

2.1 Description

Include other ML methods and other AL sampling methods

2.2 Solution

I wrote in one of the previous email that uncertainty sampling could work only with ensemble models, of course it is not correct, it works with any probabilistic model, as we just need to find samples nearest to the decision boundary. I chose following models/sampling strategies:

- ExtraTreesClassifier with uncertainty sampling and without sampling
- GaussianNB with uncertainty sampling and without sampling

- LGBM with uncertainty sampling and without sampling
- RandomForestClassifier with vote entropy sampling by Committee (3 learners) and without sampling (for some reasons it took an eternity to run sampling by Committee, so I have not continued further experiments)

All performance metrics for models trained using uncertainty sampling with ExtraTreesClassifier are significantly higher than for models trained on all data without sampling. Please see Figure 2 for radar charts with visualization (also, see table "t-test stats.csv" in "./results/ExtraTreesClassifier")

Only Accuracy and MCC for models trained using uncertainty sampling with GaussianNB are statistically significantly higher than for models trained on all data without sampling. Please see Figure 2 for radar charts with visualization (also, see table "t-test stats.csv" in "./results/GaussianNB"). Mean values of F1, AUC LB, AUC and AUC UB are also higher for AL, but results are not statistically significant.

MCC, F1, Accuracy and AUC for models trained using uncertainty sampling with LGBM are statistically significantly higher than for models trained on all data without sampling. Please see Figure 2 for radar charts with visualization (also, see table "t-test stats.csv" in "./results/LGBM"). For some reasons I failed to calculate AUC LB and AUC UB for AL-models.

All performance metrics except for Accuracy for models trained using vote entropy sampling by Committee with RandomForestClassifier are significantly higher than for models trained on all data without sampling. Please see Figure 2 for radar charts with visualization (also, see table "t-test stats.csv" in "./results/Committee_vote_entropy_sampling")

3 Task 3

3.1 Description

Try pipeline on the different datasets for SCAMs

3.2 Solution

- Small Shoichet dataset balanced by positive data from large Shoichet dataset (SCAMS_balanced_with_positive)
- Small Shoichet dataset concatenated with 780 random samples from large Shoichet dataset, so it is imbalanced (SCAMS_added_positives_653_1043, imbalance ratio = 1.59)
- Balanced b-Lactamase database by Tropsha.
- Balanced Cruzain database by Tropsha

AL models trained on balanced and imbalanced Shoichet datasets demonstrated better performance that non-AL based based on all metrics, please see Figure 3 for radar charts with visualization.

You could find the results in the folders "./results/SCAMS_balanced_with_positive" and "./results/SCAMS_added_positives_653_1043".

I have not run pipeline on datasets by Tropha, since they are large and it would take an eternity. I should think about code optimization, batch selection and stopping criteria, but suppose even after optimization it could take considerable amount of time. I have read some of your papers [1, 2] and saw other articles[3, 4] that outlined stopping criteria, but my accuracy curve looked chaotic (I have not checked MCC curve) and did not demonstrate exponential decay. I would be happy to hear further directions on this.

4 Task 4

4.1 Description

Revisit the scaffold-based train/test splitting to generate more difficult machine learning problems.

4.2 Solution

I decided to implement RDKit butina clustering algorithm[5], since I need scikit-learn 0.23 in environment to run imbalanced-learn, but current deepchem version is incompatible with scikit-learn 0.23 (see. issue here, Bharath kindly informed me that I can use nightly build, but I have already implemented functions based on butina clustering). The current version of butina split include in the test set only molecules which are "lonely" in their clusters, so we preclude the case when training set include many molecules that are very similar to the ones in the test set (I used function written by Patrick Walters to generate clusters, please, see butina_cluster in utilities.py and wrote a function split_with_butina in main.py). I run 10-fold cross-validation with butina splitter and compared AL-non-AL models performance using paired t-test (see Fig. 4). We observe that model trained using AL strategy perform better based on all metrics. Also, see table "t-test_stats.csv" in "./results/Butina". Also, I adapted ScaffoldSplitter code from deepchem and run pipepline using ScaffoldSplitter. AL based models perform better (see Fig. 4).

5 Task 5

5.1 Description

Identify the fraction of data at which the active learning sampling is most predictive.

5.2 Solution

I decided to run pipeline 100 times and save percentage of data used to reach the maximum MCC value. Thereby I can make a reliable estimation (see Fig. 5). The ultimate goal is to have a pipeline which reaches in 95% of cases the maximum performance. As you can see from the figure, the estimated the 95 percentile value is 88%. Therefore, I implemented a naïve stopping criteria: 88 % of data.

6 Task 6

6.1 Description

Add parallelization.

6.2 Solution

When I tried to change code structure and add Parallel from joblib in cycle I run into issues ("Could not pickle the task to send it to the workers"). Also, I found some advice here, but they did not help me.

After, I tried to use threading (I know that due to GIL problem in general multithreading is not very effective in Python, but RF (the standard model in our pipeline) is rather numpy intensive (runs C code), so I I thought it could be of help). Multithreading improved time performance, but model performance dropped.

Eventually, I decided just add 'n_jobs=-1' in model initialization, so it runs faster now. For instance, it took 22 min to run the standard pipeline, even taking into account that my laptop has broken down, so I am using another one with much less computational powers (I do not have the exact estimation, but it took approximately an hour before on my personal laptop).

If I am not mistaken, the time complexity to build RF model is linearithmic $O(n \log n)$, where n is the number of samples. In the case of AL-pipeline it is $O(n \log n)$! (I am not sure that there is such notation, maybe just factorial, $O(n!)$). It takes 22 minutes to run the pipeline with ≈ 1000 samples. Tropha's beta-lactamase dataset consists of ≈ 60000 samples. So the estimated time to run the pipeline (if my calculations are correct) with Tropha's beta-lactamase dataset on my current laptop is 60!*22 min.

7 Task 7

7.1 Description

Add bath mode.

7.2 Solution

I added uncertainty batch sampling to pipeline and tried to run it with different batch sizes (I observed anticorrelation between batch size and model performance, so I stick with a bath size 3). Even though the AL-strategy increases the mean values of all metrics, it only statistically significant for F1 and Accuracy. I have not plotted a radar chart for this run (as my laptop is broken, I do not have access to the class which was built on matplotlib, but I currently added function to plot radar chart with a package plotly, but I did not add a new plot to the update, as it has different style).

References

[1] Daniel Reker, Petra Schneider, Gisbert Schneider, and JB Brown. Active learning for computational chemogenomics. *Future medicinal chemistry*, 9(4):381–402, 2017.

[2] Christin Rakers, Daniel Reker, and Brown JB. Small random forest models for effective chemogenomic active learning. *Journal of Computer Aided Chemistry*, 18:124–142, 2017.

[3] Jingbo Zhu, Huizhen Wang, Eduard Hovy, and Matthew Ma. Confidence-based stopping criteria for active learning for data annotation. *ACM Transactions on Speech and Language Processing (TSLP)*, 6(3):1–24, 2010.

[4] Seyda Ertekin, Jian Huang, Leon Bottou, and Lee Giles. Learning on the border: active learning in imbalanced data classification. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 127–136, 2007.

[5] Darko Butina. Unsupervised data base clustering based on daylight's fingerprint and tanimoto similarity: A fast and automated way to cluster small and large data sets. *Journal of Chemical Information and Computer Sciences*, 39(4):747–750, 1999.

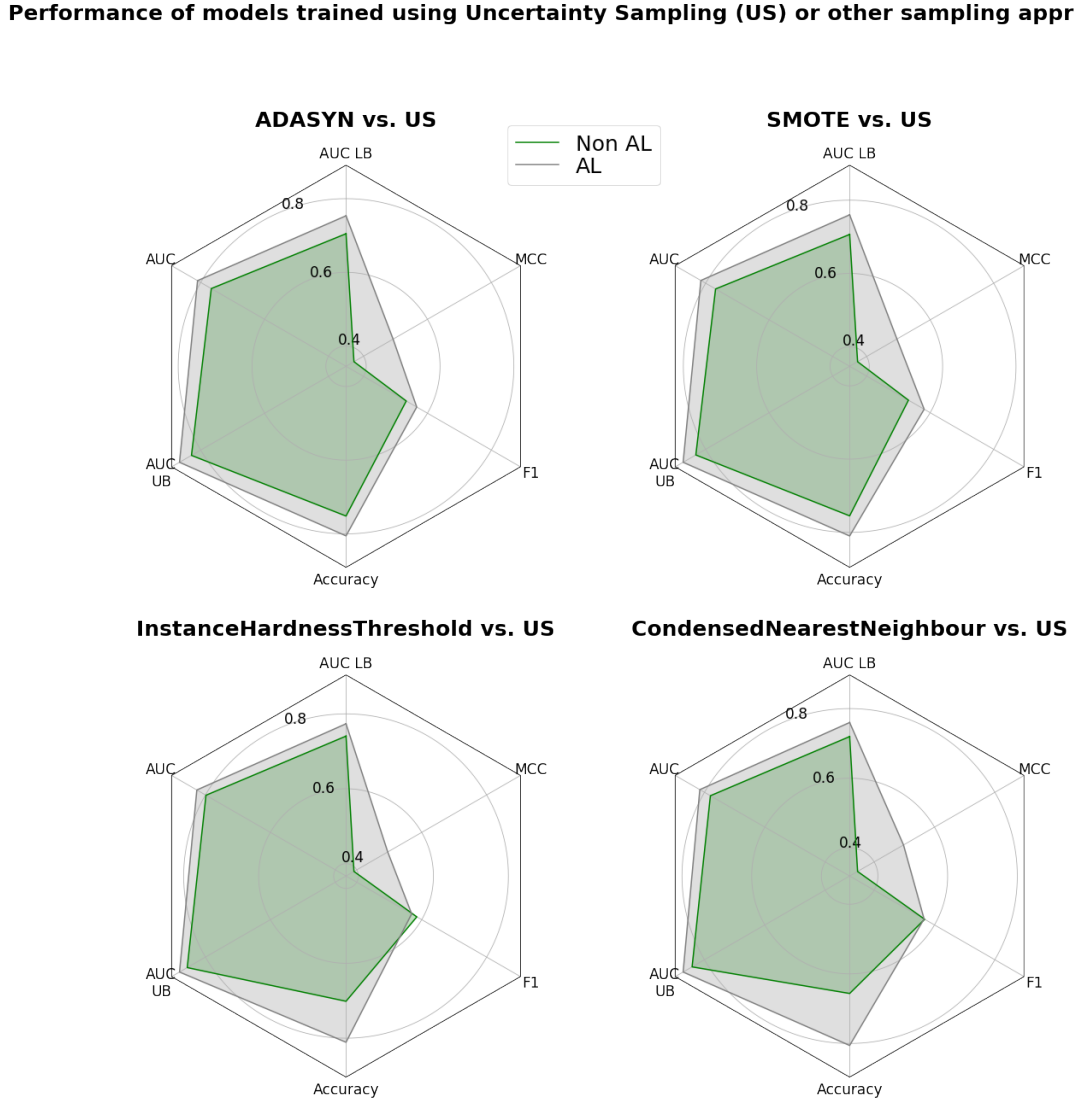

Figure 1: Performance of models trained using Uncertainty Sampling (US) or other sampling approaches

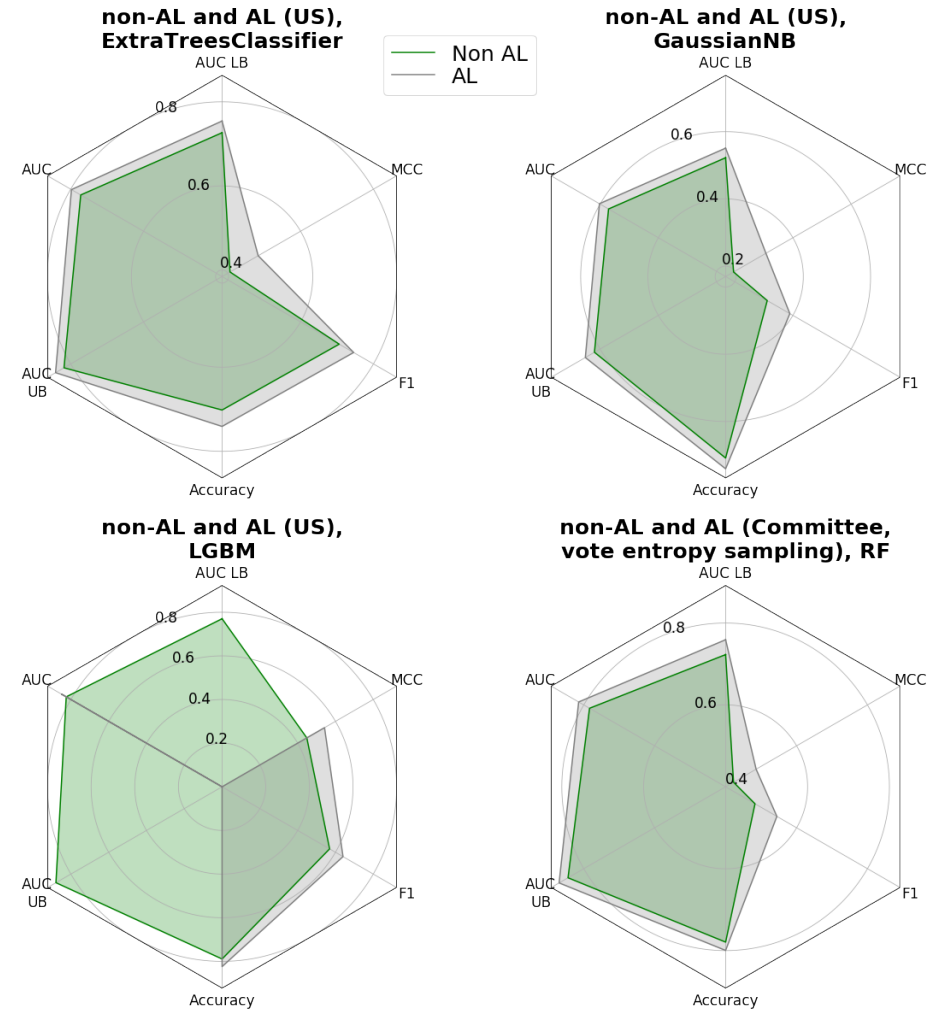

Figure 2: Performance of models trained based on AL (different sampling methods) and non-AL sampling strategy using different ML models

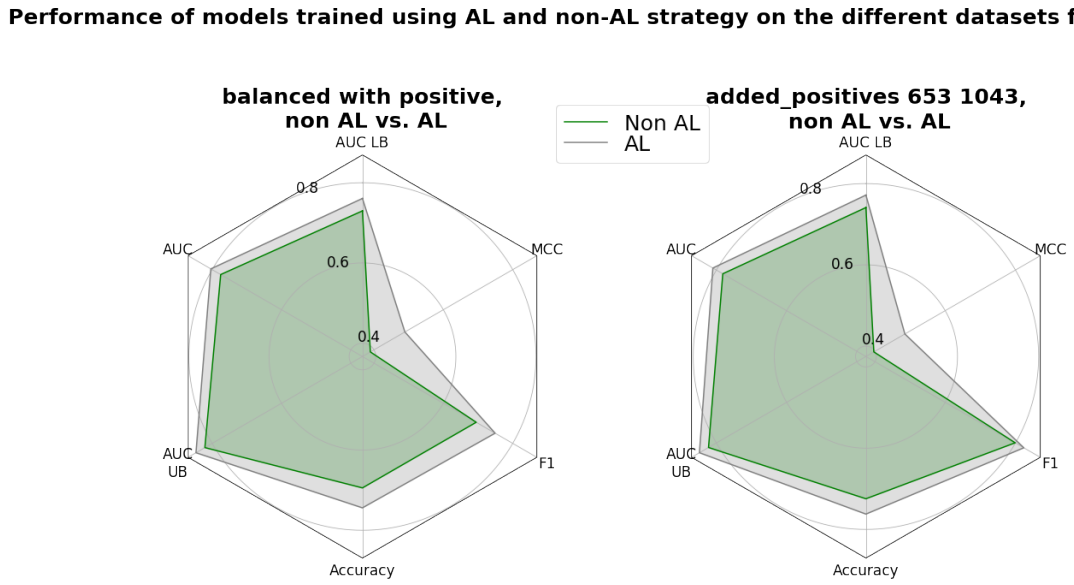

Figure 3: Performance of models trained using AL and non-AL strategy on the different datasets for SCAMs

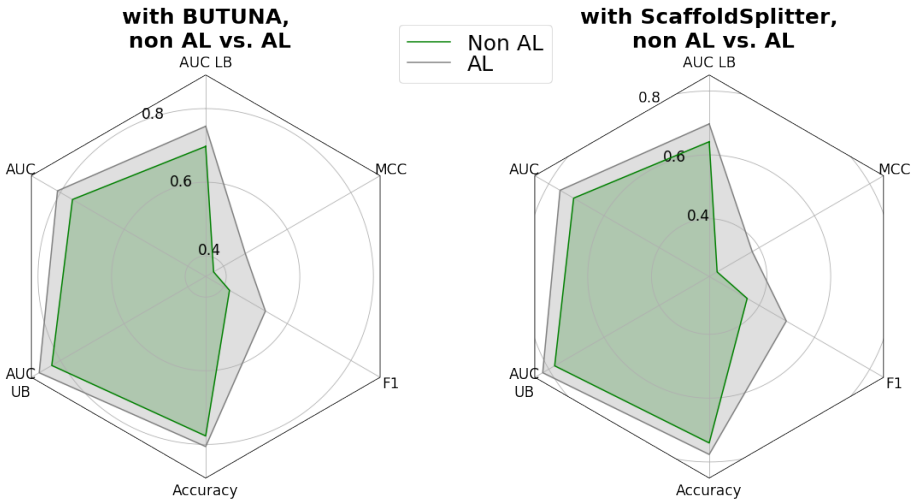

Figure 4: Performance of models trained using AL and non-AL strategy with different splitters

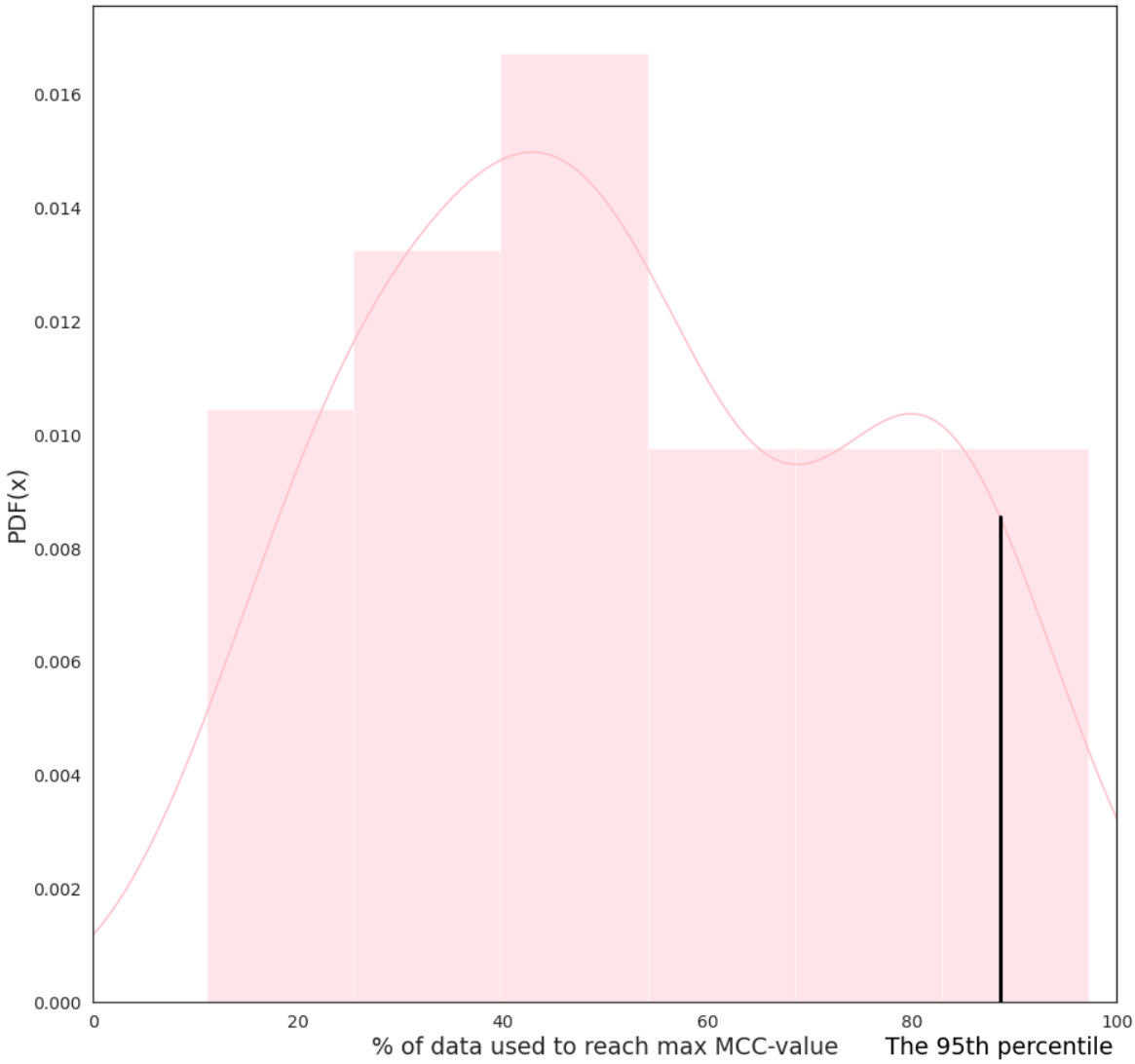

Figure 5: Distribution of % of data used to reach the max MCC value