Final project for COGS118A
Fall 2020

Due on December 15, 2020 11:59PM CA time
No late submission is possible without extremely unusual circumstances.

Read this paper first: https://www.cs.cornell.edu/~caruana/ctp/ct.papers/caruana.icml06.pdf

Your project will be to replicate a part of the analysis done in the paper by Caruana & Niculescu-Mizil (hereafter referred to as CNM06).

You will write a report with >1,000 words (excluding references & appendices). The main sections are: a) abstract, b) introduction, c) method, d) experiment, e) conclusion, and f) references.

You can follow the paper format from leading machine learning journals such as Journal of Machine Learning Research (https://www.jmlr.org/format/authors-guide.html) or IEEE Trans. on Pattern Analysis and Machine Intelligence (http://www.computer.org/web/tpami), or leading conferences like NeurIPS (the conference formerly known as NIPS; https://papers.nips.cc/) and ICML (https://icml.cc/Conferences/2020/StyleAuthorInstructions).

Experiments & results:

From CNM06, pick 3 of the datasets (available from UCI machine learning repository) and pick 3 of the algorithms (different kernels of SVM are not 2 different classifiers, pick truly different ones). For each classifier and dataset combo, you will need to do 3 trials (CNM06 does 5; we will make it easier for you). That's 3x3x3 = 27 total trials.

Each trial you will follow the procedure laid out in CNM06: randomly choose 5000 data samples for 5 fold cross-validation to select the hyperparameters via a gridsearch. In CNM06 section 2.1 they lay out the hyperparameter values they used in their search for each algorithm; use those for your search too.

Each algorithm has a different number of hyperparameter settings to try, so the exact number of train/validate cycles you do will depend on the algorithm you choose. For example CNM06 say about Logistic Regression: "train both unregularized and regularized models, varying the ridge (regularization) parameter by factors of 10 from 10^-8 to 10^4." That would be 14 total settings (including one where regularization = 0) to try, yielding 14 hyperparameter settings * 5 folds 378 total train/validate cycles just for a single trial of Logistic Regression. At the end of those 378 train/validate cycles, you will select the hyperparameters settings that did best for the

mean over all 5 folds of that setting. Then you will train the model one more time on all 5000 training data samples, and measure model performance on the test set (all the data in the dataset other than the 5000 random samples).

Model performance will be measured by a single performance metric; Here again we make it easier for you than CNM06's 8 different performance metrics. Let's default to accuracy, but if you think that a particular dataset would benefit from a different performance metric or metrics that is your call. The obvious case would be a dataset with very imbalanced classes which would require something like F1 or AUC to better understand performance.

Your main results will be something similar to tables 2 & 3 in CNM06:

- a table of mean (across 3 trials) test set performance for each algorithm/dataset combo[1]. This table should be annotated with uncorrected 2 sample t-tests to compare across algorithms. The difference between your table and CNM06 is that you are using only a single performance metric; therefore you do not have to normalize & calibrate your performance metrics to compare across them as in CNM06.[2]
- a table of mean (across 3 trials x 3 datasets) test set performance for each algorithm. This table should be annotated with uncorrected 2 sample t-tests to compare across algorithms. The difference between your table and CNM06 is that you are using only a single performance metric; therefore you do not have to normalize & calibrate your performance metrics to compare across them as in CNM06.[3]

Secondary results you should report

- A main matter table showing mean *training* set performance for the optimal hyperparameters on each of the dataset/algorithm combos and a discussion of the difference between each algorithms' training and test set performance
- An appendix table with raw test set scores, not just the mean scores of the table
- An appendix table with the p-values of the comparisons across algorithms in the different main matter tables

Secondary results you may wish to report (extra credit land):

- An analysis of the time complexity of the algorithms you tried
- A learning curve per algorithm/dataset combo: comparing test set performance for the best hyperparameter choice as you vary the number of training samples or a given dataset
- A heatmap-style plot of the validation performance vs hyperparameter setting for your algorithms

[1] Obviously standard deviation is almost laughable at 3 trials per setup; but it's the thought that counts I guess?

[2] This could change if you wanted to try multiple error metrics. Please discuss this with me if you're headed this way.

[3] This could change if you wanted to try multiple error metrics. Please discuss this with me if you're headed this way.

Grading and extra credit:

You will turn in your report and code in a single PDF. The code is in an appendix. You should be prepared to provide your code in a working format (.ipynb, .py, .cpp, whatever) to the grader upon request via email.

The project is marked out of 100 points:

- 50 are based on the technical aspects of what you implement being done correctly
- 30 points for the write up (clarity & correctness of each section; we will only be marking down for English mistakes if they rise to the level of making it hard to understand)
- 10 points for the code quality and legibility/commenting
- 10 points for hardness of the undertaking, aesthetics, and other quality issues.

If you feel that your work deserves bonus points due to reasons such as:

- novel ideas and algorithms or state-of-the-art results,
- large efforts in your own data collection/preparation
- doing more than the 27 trials/1 metric required for the main analysis
- doing lots of secondary results

then please create a "Bonus Points" section to describe why you deserve bonus points. I'm not setting an upper limit on bonus points, but FYI if someone did a couple of extra algorithms/metrics/secondary analyses well I would probably grant 5-10 points of extra credit.

Implementation and advice:

You may use what you'd like, but I highly recommend scikit-learn as it implements all these algorithms plus many important helper functions for cross-validation, scoring metrics, etc.

As you decide which algorithms to use, do some time testing on a single train/validate cycle. If it takes 5 minutes per train/validate cycle and there's > 1,000 cycles to do to get all the hyperparameters done… well you get the problem that 3.5 days of computation for a single algorithm/dataset combo could land you in.

Note that you might not get exactly the same algorithm ranking results as CNM06 did. The test set performances and optimal parameters vary due to the particular ML libraries you are using as well as random sampling of the data. But the overall differences should be reasonable and interpretable and certain overall trends should be the same.

If you have a question or a problem seek advice from the instruction team ASAP!!! There's not much time left for going down the wrong path.

This project is based on previous projects in this course, but this quarter is different than other ones as we deal with pandemic. If we need to adjust expectations as we go we will. But to do that I need communication from you!