Final project for COGS118A Winter 2021

Due on March 17, 2021 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). The undertaking is to investigate a few ML algorithms and determine if any of them are better than others at binary classification tasks.

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) discussion, and f) at least 3 references. Follow the paper format from a leading machine learning journal (e.g., Journal of Machine Learning Research https://www.jmlr.org/format/authors-guide.html) or conference (e.g., NeurIPS click for LaTeX template).

Experiments & results:

You will use at least 3 algorithms (truly different algos, not say kernel variations of the same algo) on at least 4 different datasets. For each classifier and dataset combo, you will need to do 5 trials. That's 3x4x5 = 60 total trials.

Each trial you will need to do a hyperparameter search to find the best settings to use for the algorithm. NB: this means you will likely be using different settings for each trial! There are many ways to do hyperparameter search which we will cover in class. If you're unsure what method to use, just follow the procedure laid out in CNM06: randomly choose 5000 data samples for your training set with replacement. Do 5-fold cross-validation on the training set to select the hyperparameters via a systematic gridsearch of the parameter space. In CNM06 section 2.1 they lay out the hyperparameter values they used in their search for each algorithm; you may use those for your search too.

The exact number of train/validate cycles you do will depend on the hyperparameter search. For example if you are following the CNM06 method to investigate Logistic Regression it says: "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 which is 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).

To sum up: if you are doing the CNM06 logistic regression method you will be training 60*378=22,680 times. That will probably take somewhere between 10^1 and 10^2 minutes because logistic regression is *O*(n_samples * m_variables) time to train. **But for slower algorithms this can take hours or even days if you are exploring a large number of hyperparameters. So give yourself enough time to do this project, don't wait until the week before its due!** Also it can be worthwhile to time a single training/validation cycle and multiply to predict how long it will take for a given algorithm to complete. Of course some hyperparameter settings can be slower to converge than others, so even this prediction may not be accurate ¯_(ツ)_/¯

Model performance will be measured using at least 3 performance metrics that are in the range [0,1]. By default those metrics will be accuracy, F1 score, and AUC. However, depending on the datasets and scenarios you take on it could be appropriate to try other metrics. For instance for imbalanced datasets it has been argued that Matthew's Correlation Coefficient is a better choice than F1 (see this paper). That's not to say that you should be using MCC; it's to show you that particular needs given your task should affect your choice of metrics. And that a top class report (especially one from a large group) would take on some research in this respect.

NB: DON'T run each trial 3 times to measure 3 metrics. You can either (a) run each trial once, save the predictions and true labels in arrays, and calculate 3 metrics after completing the trial; or (b) use sklearn's convenience functions to calculate multiple metrics during the hyperparameter search.

Your main results will be something similar to Tables 1-3 in CNM06:

- Table 1 explaining the datasets
- Table 2 giving the mean test set performance across trials for each algorithm/dataset combo, with separate columns for each metric. There will also be a column for the mean performance across metrics. This table should be annotated as in CNM06 using boldface to show the best performing algorithm in each column and * to denote a non-significant difference between best algo and the others[1].
- Table 3 giving the mean test set performance across trials for each algorithm/metric combo. Annotated as Table 2 to show best/non-significant differences from best.

Secondary results you should report

- An appendix 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

[1] Null hypothesis statistical testing is not going well powered at 5 trials. But it's the thought that counts I guess?

- 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
- Mean (over trials) ROC and PR curves for algorithm/dataset

Grading and extra credit:

A minimal implementation of this project (appropriate for a single person project aiming for somewhere between a B and an A-) is to closely follow CNM06. Pick algorithms and datasets that are present in the paper and reproduce those results. Someone shooting for an individual project that goes into A+ territory needs to add

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 correctness of your results and the clarity of their presentation in text, tables and graphics.
- 25 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)
- 25 points for the code correctness, quality and legibility/commenting

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 lots more trials/datasets/algorithms than required
- 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.

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