CS273P Project 2

Machine Learning & Data Mining: Spring 2018

Due: Thursday June 14, 2018

In this project you will work with a 1994 Census bureau dataset of population survey by Ronny Kohavi and Barry Becker[1]. Here the goal is to predict whether the income of a person exceeds \$50K/yr based on census data.

Project Details.

This course project will consist of groups of three students working together, not necessarily the same groups as in project1. We will use a class Kaggle competition `https://www.kaggle.com/c/uci-s2018-cs273p-2` to manage performance evaluation.

Step 1: Form teams.

Please form teams of 3 students who share your interests and with whom you will directly collaborate. Piazza can help with locating other students in need of joining at team.

There is no barrier to collaboration on this project, so feel free to talk to other teams about what they are doing, how they are doing it, and how well it works (and you can see this last point on the leaderboard as well). A significant component of good performance can boil down to feature design and choices, so you may want to talk to other teams about their choices and how it affects performance. You are also free to use online code, or other sources. However, please make sure that your own entries are developed by your team members – the purpose of this project is to give you practical experience, so it will not be helpful if you just follow instructions from a friend.

Step 2: Download the data.

Download the following files from the Kaggle website -

```
X_train.txt - training data feature values
Y_train.txt - target values for training data
X_test.txt - test data feature values
```

You will train your models using X_train and Y_train, make predictions $\hat{Y}$ using X_test, and upload your predictions $\hat{Y}$ to Kaggle. Kaggle will then score your predictions, and report your performance on a subset of X_test, used for placing your team on the current leaderboard (the "leaderboard" data). After the competition, the score on the remainder of the data (X_test) will be used to determine your final standing; this ensures that your scores are not affected by overfitting to the leaderboard data.

Your submission has to be a CSV file containing two columns separated by a comma. The first column should be the instance number, followed by the second column that is the soft prediction value for this instance. Further, the first line of the file should be "ID,Prob1", i.e. the name of the two columns.

[1] Ron Kohavi, "Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid", Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996.

Scoring.

Scoring of predictions is done using AUC, the area under the ROC (receiver-operator characteristic) curve. This gives an average of your learner's performance at various levels of sensitivity to positive data. This means that you will likely do better if, instead of simply predicting the target class, you also include your confidence level of that class value, so that the ROC curve can be evaluated at different levels of specificity. To do so, you can report your confidence in class +1 (as a real number); the predictions will then be sorted in order of confidence, and the ROC curve evaluated.

Step 3: Choose techniques to explore.

You are required to employ at least two (2) different types of learners. Suggestions include

- **K-Nearest Neighbors**.
- **Logistic regression**.
- **SVM**.
- **Neural networks**.
- **Random forests**.
- **Boosted learners**.
- **Other**. You may pick any other algorithm/method of your choosing.

You may use any platform, environment, language, library or code you choose, as long as you list any any third party software/library/tools you used in your report. Or you may implement the algorithms yourself (although I caution against it).

However, a key point is that you must *explore* your approach(es); so you must do more than simply download a publicly available package and run it with default settings, or trying a few values for regularization or other basic parameter settings. You must at least explore the method fully enough to understand how changes might affect its performance, verify that your findings make sense, and then use your findings to optimize your performance; ideally, you should repeat this process several times, exploring several different ideas. In your report, you should describe why you decided to explore this aspect, what you expected to find, and how your findings matched (or didn't match) your expectations.

Step 4: Build your learners, Evaluate, Repeat.

Using the techniques you chose to focus on, construct predictive models for your target(s). For each learner, you should do enough work to make sure that it achieves "reasonable" performance. Then, take your best learned models, and combine them using a blending or stacking technique. This could be done via a simple average/vote, or a weighted vote based on another learning algorithm. Feel free to experiment and see what performance gains are possible.

Be aware of the positive and negative aspects of the learners we have discussed. For example, nearest neighbor methods can be very powerful, but can also be very slow for large data sets; a similar statement applies to dual-form SVMs. For such learners, dealing with the large data set may be a significant issue – perhaps you could reduce the data in some way without sacrificing performance? On the other hand, linear methods are very fast, but may not have enough model

complexity to provide a good fit. For such learners, you may need to try to generate better features, etc.

Output your predictions and evaluate them on Kaggle:

```
fh = open('predictions.csv','w')    # open file for upload
fh.write('ID,Prob1\n')         # output header line
for i,yi in enumerate(Yte):
  fh.write('{},{}\n'.format(i,yi))  # output each prediction
fh.close()                          # close the file
```

You can check the leaderboard to see your "test" performance.

NOTE: You're limited to a few (≈ 2) submissions per day to avoid over-loading their servers and to enforce good practices. Thus, you should not try to upload every possible model with every possible parameter setting – use validation data, or cross-validation, to assess *which* models are worth uploading, and just use the uploads to verify your performance on those.

Step 5: Write it up

Your team will produce a single write-up document (one report per team), approximately **6 pages** long, describing the problem you chose to tackle and the methods you used to address it, including which model(s) you tried, how you trained them, how you selected any parameters they might require, and how they performed in on the test data. Consider including tables of performance of different approaches, or plots of performance used to perform model selection (i.e., parameters that control complexity).

Within your document, please try to describe to the best of your ability who was responsible for which aspects (which learners, etc.), and how the team as a whole put the ideas together.

You are free to collaborate with other teams, *including* sharing ideas and even code, but please document where your predictions came from. (This also relaxes the proscription from posting code or asking for code help on Piazza, at least for project purposes.) For example, for any code you use, please say in your report who wrote the code and how it was applied (who determined the parameter settings and how, etc.) Collaboration is particularly true for learning ensembles of predictors: your teams may each supply a set of predictors, and then collaborate to learn an ensemble from the set.

Requirements / Grading

I am looking for several elements to be present in any good project. These are:

(a) Exploration of techniques/elements on which we did not spend significant time in class. For example, using neural networks, or random forests are great ideas; but if you do this, you should explore in some depth the various options available to you for parameterizing the model, controlling complexity, etc. (This should involve more than simply varying a parameter and showing a plot of results.) Other options might include feature design, or optimizing your models to deal with special aspects of the data (e.g.possible outlier data; etc.). Your report should describe what aspects you chose to focus on.

(b) Performance validation. You should practice good form and use validation or cross-validation to assess your models' performance, do model selection, combine models, etc. You should *not* simply upload hundreds of different predictors to the website to see how they do. Think of the website as "test" performance – in practice, you would only be able to measure this once you go live.

(c) Adaptation to under- and over-fitting. Machine learning is not very "one size fits all" – it is impossible to know for sure what model to choose, what features to give it, or how to set the parameters until you see how it does on the data. Therefore, much of machine learning revolves around assessing performance (e.g., is my poor performance due to underfitting, or overfitting?) and deciding how to modify your techniques in response. Your report should describe how, during your process, you decided how to adapt your models and why.

Your project grade will be based on the quality of your written report, and groups whose final prediction accuracy is mediocre may still receive a high grade, if their work and results are described and analyzed carefully. But, some additional bonus points will also be given to the teams at the top of the leaderboard.