Due: Apr. 10

Econ 5970 - Spring 2018 **Problem Set 10** **beginning of class**

This problem set will give you practice in using cross-validation to tune classification models using four of the Five Tribes of Machine Learning: trees, neural networks, naive Bayes, and k-nearest neighbor / support vector machines.

As with the previous problem sets, you will submit this problem set by pushing the document to *your* (private) fork of the class repository. You will put this and all other problem sets in the path `/DScourseS18/ProblemSets/PS10/` and name the file `PS10_LastName.*`. Your OSCER home directory and GitHub repository should be perfectly in sync, such that I should be able to find these materials by looking in either place. Your directory should contain at least three files:

- `PS10_LastName.R` (you can also do this in Python or Julia if you prefer, but I think it will be much more difficult to use either of those alternatives for this problem set)
- `PS10_LastName.tex`
- `PS10_LastName.pdf`

1. Type `git pull origin master` from your OSCER `DScourseS18` folder to make sure your OSCER folder is synchronized with your GitHub repository.

2. Synchronize your fork with the class repository by doing a `git fetch upstream` and then merging the resulting branch. (`git merge upstream/master -m "commit message"`)

3. Install the following machine learning packages if you haven't already:
   - `rpart`
   - `e1071`
   - `kknn`
   - `nnet`

4. Start your code file by importing the starter code I have provided you at `https://github.com/tyleransom/DScourseS18/blob/master/MachineLearning/PS10starter.R`. This starter code reads in the adult income data from the UC Irvine datasets repository. The goal of this problem set is to compare the 5 Tribes in terms of their ability to classify whether someone is high-income or not. Thus, the target variable for this exercise will be `income$high.earner`.

5. Using the `mlr` library, create the following objects for your cross validation exercise:
   - The classification task
   - The **3-fold** cross-validation strategy

- The tuning strategy (e.g. `random` with **10** guesses)
- Each of the six "learners" (algorithms):

  1. Trees: `classif.rpart`
  2. Logistic regression: `classif.glmnet`
  3. Neural network: `classif.nnet`
  4. Naive Bayes: `classif.naiveBayes`
  5. kNN: `classif.kknn`
  6. SVM: `classif.svm`

  For each algorithm, add `predict.type = "response"` as an additional argument to the `makeLearner()` function.

6. Now set up the hyperparameters of each algorithm that will need to be cross validated. If you aren't sure what to call them, please refer to the individual package documenation.

   - Tree model
     - `minsplit`, which is an integer ranging from 10 to 50 (governs minimum sample size for making a split)
     - `minbucket`, which is an integer ranging from 5 to 50 (governs minimum sample size in a leaf)
     - `cp`, which is a real number ranging from 0.001 to 0.2 (governs complexity of the tree)
   - Logit model (this is the elastic net model from last problem set, so the two parameters are λ and α. For this problem set, let $\lambda \in [0, 3]$ and $\alpha \in [0, 1]$.
   - Neural network model
     - `size`, which is an integer ranging from 1 to 10 (governs number of units in hidden layer)
     - `decay`, which is a real number ranging from 0.1 to 0.5 (acts like λ in the elastic net model)
     - `maxit`, which is an integer ranging from 1000 to 1000 (i.e. this should be fixed at 1,000 ... it governs the number of iterations the neural network takes when figuring out convergence)
   - Naive Bayes
     - There's nothing to regularize here, so you don't need to do any cross-validation or tuning for this model

- kNN
  - `k`, which is an integer ranging from 1 to 30 (governs the number of "neighbors" to consider)
- SVM
  - `kernel`, which is a string value equal to `"radial"` (program it as "discrete" in `mlr`'s interface)
  - `cost`, which is a real number ("discrete" in `mlr`'s interface) in the set $\{2^{-2}, 2^{-1}, 2^{0}, 2^{1}, 2^{2}, 2^{10}\}$ (governs how soft the margin of classification is)
  - `gamma`, which is also a real number in the set $\{2^{-2}, 2^{-1}, 2^{0}, 2^{1}, 2^{2}, 2^{10}\}$ (governs the shape [variance] of the Gaussian kernel)

7. Now tune the models. Use the F1 score and the G-measure as criteria for tuning (these are `f1` and `gmean`, respectively). Remember that you don't need to tune the Naive Bayes model.

8. Once tuned, apply the optimal tuning parameters to each of the algorithms (again, you don't need to do this for Naive Bayes since there was no tuning done previously). Then train the models, generate predictions, and assess performance.

9. As a **table** in your .tex file, report the optimal values of the tuning parameters for each of the algorithms. How does each algorithm's out-of-sample performance compare with each of the other algorithms?

10. Compile your .tex file, download the PDF and .tex file, and transfer it to your cloned repository on OSCER using your SFTP client of choice (or via `scp` from your laptop terminal). You may also copy and paste your .tex file from your browser directly into your terminal via `nano` if you prefer, but you will need to use SFTP or `scp` to transer the PDF.[1]

11. You should turn in the following files: .tex, .pdf, and any additional scripts (e.g. .R, .py, or .jl) required to reproduce your work. Make sure that these files each have the correct naming convention (see top of this problem set for directions) and are located in the correct directory (i.e. `~/DScourseS18/ProblemSets/PS10`).

12. Synchronize your local git repository (in your OSCER home directory) with your GitHub fork by using the commands in Problem Set 2 (i.e. `git add`, `git commit`

[1] If you want to try out something different, you can compile your .tex file on OSCER by typing `pdflatex myfile.tex` at the command prompt of the appropriate directory. This will create the PDF directly on OSCER, removing the requirement to use SFTP or `scp` to move the file over.

`-m "message"`, and `git push origin master`). Once you have done this, issue a `git pull` from the location of your other local git repository (e.g. on your personal computer). Verify that the PS10 files appear in the appropriate place in your other local repository.