Problem Set 01
Data Mining and Machine Learning – Spring 2016
Due date: 2016-02-19 13:00

All assignments must be uploaded to the assignments tab in ClassesV2 (notice that this is **not** the dropbox) by the date and time specified. Make sure that you follow the instructions exactly as described. This is a large class with a limited number of TA's and we will be grading parts of the assignment using an automated grading engine; you will lose points for things such as not naming files correctly. You may discuss problem sets with others, but must write up your own solutions. This means that you should have no need to look at other's final written solutions.

You need to turn in all of your solutions as a zip compressed file, named `netid_pset02.zip`, with your actual netid filled in in all lower case letters. This archive should contain the following files:

- `pset02.R` or `pset02.py`
- `pset02.pdf`

The pdf file should contain any written solutions; the contents of the script are described below.

Implementation

You will implement a function to fit an additive model using the backfitting algorithm described in class. For the smoother, you will use knn with $k = 20$. Unlike in problem set 1, your solution should be written as an executable script that can be run as either:

```
> Rscript pset02.R train.csv test.csv
```

Or,

```
> python3 pset02.R train.csv test.csv
```

The script should load the training data and testing data given as the first two arguments to the script, fit a model on the training data, predict the values on the test data, and save the results as a file `results.csv` in the working directory. The training and testing data are comma separated numeric matrices. The training data has one additional column, the first of which is the response and remainder are the predictor variables. The test set has only the predictor variables. Rather than setting up a complex stopping criteria, simply run the iteration 25 times. You can use a pre-built knn regression function; I have made suggestions for what would work best in the starter code for both Python and R.

You should write the code such that you iteratively learn the values of $\widehat{g}_j$ just on the training set. In order to fit the learned model on the test set, use linear interpolation between the fitted values of each $\widehat{g}_j$ (this will be much easier than the direct alternative).

A script with these working functions should be uploaded as described above (i.e., named `pset02.R` or `pset02.py` and placed in a zip archive). There are sample data files `train.csv`, `test.csv`, and `results.csv` on the class website that you can use to check your code (your results may differ slightly, but should be fairly highly correlated). Note that this time we will specifically be using the Anaconda distribution of Python 3.5.[1]

Data files

The data referenced in the rest of this problem set is a subset of the Longitudinal Employer-Household Dynamics data collection by the US Census Bureau. You can download the files directly from here:

```
http://lehd.ces.census.gov/data/lodes/LODES7/ct/rac/ct_rac_S000_JT00_
2013.csv.gz
http://lehd.ces.census.gov/data/lodes/LODES7/ny/rac/ny_rac_S000_JT00_
2013.csv.gz
http://lehd.ces.census.gov/data/lodes/LODES7/mt/rac/mt_rac_S000_JT00_
2013.csv.gz
http://lehd.ces.census.gov/data/lodes/LODES7/ca/rac/ca_rac_S000_JT00_
2013.csv.gz
```

Note that changing the two letter state code will access additional states. Documentation for the files can be found here:

```
http://lehd.ces.census.gov/data/lodes/LODES7/LODESTechDoc7.0.pdf
```

Note that we are only using the residential files (RAC) for this problem set.

Questions

Save your answers from this section as `pset02.pdf`. You will fit several different models, but do not need to submit your code or predictions; we will just we grading your responses.

In these questions you will fit a random forest estimator to the LEHD data from the state of Connecticut (ct). The response will be the proportion of jobs that earn more than $3333 per month:

$$y = \frac{\text{CE03}}{\text{C000}}$$

For predictor variables, use the other count variables in the dataset divided again by C0000 (other than CE01, CE02 and CE03). Using this data fit a random forest estimator to the dataset. You can mostly start with the default settings in R or Python, though you will probably want to set

[1] `https://www.continuum.io/downloads`

verbose (`do.trace` in R) to True and start with around 25-50 trees. In Python you will also want to set `max_features` to around 10, rather than the default. Play with the number of trees a bit, and any other features you would like until you think you have a reasonably good model (no need to go too crazy here tuning it). Fit predictions to the other three states and then answer the following questions.

1. Describe how you decided the ultimate number of trees to use.

2. Compare the 'out-of-bag' mean squared error in CT to the errors of NY, MT, and CA. Describe what patterns or surprising features arise.

3. Now fit a linear regression using the same variables on the CT data and predict the results from the other three states. How do these mean squared errors compare to the random forest model? Note: The raw data has perfect multicolinearity because each set of variables add up to one; either drop the last count from each set, or, in R, just ignore the warnings as this is what will be done automatically for you.

4. The field `h_geocode` in the datasets gives a unique identifier for the census block from which the data are taken. The first two digits give the state FIPS code and the next 3 digits give the county FIPS code. You can easily find sources that give a mapping from county FIPS codes to county names. What are the three worst counties in terms of squared error under the random forest model in New York and California? Explain any patterns you see and (if applicable) suggest a possible solution.

5. Calculate the variable importance scores for the random forest model. Describe the most important variables; does it make sense that these would help identify areas with a high proportion of high income earners?