Unit 2 assignments – You will turn in a set of separate fully debugged python/R/julia scripts representing each part of the assignment.

Appropriate datasets for machine learning tasks can be found at https://www.kaggle.com/

IMPORTANT – Sample code for the class demos can be found at this GitHub website/repo which is also linked from my personal webpage

https://people.rit.edu/gabsbi/

https://github.com/gbabbitt/course-code-repo

1. Start a new script with any pre-classified training data set you choose and use all of the following simple machine learning methods below to classify a new dataset. You may split a single data set into training and testing data. Use a colored scatterplot and table to report correct and incorrect assignments. The methods you can demonstrate include K nearest neighbors = knn() function, naïve Bayes =naiveBayes() function from e1071 library, linear discriminant analysis = lda() function from the MASS library or quadratic discriminant analysis = qda() function from MASS library. Optional exercise: employ an unsupervised clustering using expectation-maximization (EM clustering) using the model-based clustering package (mclust).

   Random subsetting of a single data set can be achieved with the functions below

   https://www.statmethods.net/management/subset.html

   https://www.rdocumentation.org/packages/base/versions/3.4.3/topics/sample

   (optional) Perform 5-fold cross validation by random subsetting your data into 5 groups (NOTE: you can probably find functions for this in R libraries and python modules without coding it yourself)

The general procedure is as follows:

A. Shuffle the dataset randomly.
B. Split the dataset into k groups
C. For each unique group:
   a. Take the group as a hold out or test data set
   b. Take the remaining groups as a training data set
   c. Run the method on the training set and evaluate it on the test set
   d. Retain the evaluation score and discard the model
D. Summarize the skill of the model using the sample of model evaluation scores

2. Use e1071, and kernlab libraries (or their python/Julia equivalents) to conduct and compare support vector machine methods for classification. You can use any data except the IRIS example I posted previously. Try some different kernel functions (e.g. linear, polynomial and radial basis function) and compare classification accuracy. Did the classification turn out better than with simpler methods in question 1? Give percentages of correct classification (confusion

matrix) and/or cross-validation to support your answer. Show a color-mapped scatterplot as well. Note – ksvm is automatically parameter tuned while svm is not.

3. Repeat your classification you did with the svm with some artificial neural network machine learning methods utilizing a simple neuralnet() function from the neuralnet R package (or their python/Julia equivalent). Does it perform better? Compare and contrast. Find a very large dataset and build a deep learning network using the keras and tensorflow packages in R. Deploy it on various hardware (your laptop, the sporc cluster and GPU if possible…this is optional) and compare performance and speed. Specific to your data set, Produce a plot showing performance as a function of number of layers in the model and speed as a function of number of computing cores used.

4. Similar to previous scripts build a classifier based upon a random forest using randomForest() function from the randomForest library, and an boosted gradient (adaboost algorithm) using the ada() function from the ada library. Compare and contrast. You may also use python or Julia equivalents here.