

**Course:**

Intro to Data Science – DS-GA-1001 /

Data Mining for Business Analytics - INFO-GB.3336.11

Fall 2014

**Instructor:**

Brian Dalessandro

**Homework 4 – Due 11/12/2014 at 5 pm**

**(Submissions must be a WORD or PDF document)**

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Grade Total: \_\_\_\_\_\_\_\_\_\_ out of 20**

In this assignment we will be looking at data generated by particle physicists to test whether machine learning can help classify whether certain particle decay experiments identify the presence of a Higgs Boson. One does not need to know anything about particle physics to do well here, but if you are curious, full feature and data descriptions can be found here:

https://www.kaggle.com/c/higgs-boson/data

http://higgsml.lal.in2p3.fr/files/2014/04/documentation\_v1.8.pdf

The goal of this assignment is to learn to use cross-validation for model selection as well as bootstrapping for error estimation. We’ll also use learning curve analysis to understand how well different algorithms make use of limited data. For more documentation on cross-validation with Python, you can consult the following:

http://scikit-learn.org/stable/modules/cross\_validation.html#cross-validation

**Part 1 – Data Preparation**

Create a data prep function that does the following:

def cleanBosonData(infile\_name):

1. Single input is a file name string
2. Reads data (the data is comma separated, has a row header and the first column EventID is the index) into a dataFrame
3. Cleans the data
   1. Convert Label to numeric (choose the minority class to be = 1)
      1. Create a feature Y with numeric label
      2. Drop the feature "Label"
   2. If a feature has missing values (-999):
      1. Create a dummy var for the missing value
         1. Call the variable "orig\_var\_name"+"mv" where "orig\_var\_name" is the name of the actual var with a missing value
         2. Give this new variable a 1 if the orig var is missing
      2. Replace the missing value with the average of the feature (make sure to compute the mean on records where the value isn't missing, and use pandas.replace())
4. returns the cleaned dataset

***Copy and paste your function here (as a guide, this function can easily be done in less than 15 lines)***

**Part 2 – Basic Evaluations**

In this part you will build an out-of-the box logistic regression model and support vector machine. You will then plot ROC for the LR and SVM model.

Download the following data from NYUClasses > Resources > Data, and build the following datasets in Python:

train = cleanBosonData(../boson\_training\_cut\_2000.csv)

test = cleanBosonData(../boson\_testing\_cut.csv)

On the training set, build the following models:

1. A logistic regression using sklearn.linear\_model (use linear\_model.LogisticRegression(C=1e30)).
2. An SVM using sklearn.svm.svc (specify that kernel=’linear’)

For each model above, plot the ROC curve of both models on the same plot. Make sure to use the test set for computing and plotting. In the legend, also print out the Area Under the ROC (AUC) for reference.

* ***Copy and paste the plot here.***
* ***Which of the two models is generally better at ranking the test set?***
* ***Are there any classification thresholds where the model identified above as ‘better’ would underperform the other in a classification metric (such as TPR)?***

**Part 3 – Model Selection with Cross-Validation**

We think we might be able to improve the performance of the SVM if we perform a grid search on the hyper-parameter C. Because we only have 1000 instances, we will have to use cross-validation to find the optimal C.

Write a cross-validation function that does the following:

**def xValSVM(dataset, label\_name, k, cs):**

1. Inputs a dataset, a label name, # of splits/folds, a sequence of values for c
2. Performs two loops
   1. Outer Loop – for each f in range(k):
      1. Splits the data into train & validate according to x-validation logic
   2. Inner Loop – for each c in cs:
      1. Trains an SVM on training split with C=c, kernel=’linear’
      2. Computes AUC-ck on validation data
      3. Stores AUC-ck in a dictionary of values
3. Returns a dictionary, where each key-value pair is: c:[auc-c1,auc-c2,..auc-ck]

Using the function written above, do the following:

1. Generate a sequence of 15 c values in the interval [10-9, 101] (i.e., do all powers of 10 from -9 to 1).
2. Call aucs = xValSVM(train, ‘Y’, 10, cs)
3. For each c in cs, get mean(AUC) and StdErr(AUC)
4. Compute the value for max(meanAUC-StdErr(AUC)) across all values of c.
5. Generate a plot with the following:
   1. Log10(c) on the x-axis
   2. 1 series with mean(AUC) for each c
   3. 1 series with mean(AUC)-stderr(AUC) for each c (use ‘k+’ as color pattern)
   4. 1 series with mean(AUC)+stderr(AUC) for each c (use ‘k--‘ as color pattern)
   5. a reference line for max(AUC-StdErr(AUC)) (use ‘r’ as color pattern)

*Word of caution: this x-validation step can easily take an hour or more. When testing your code, use a smaller k and a smaller range of cs (such as k=3 and len(cs)=2). If you believe your code works, run the entire grid search and go watch some TV for an hour.*

* ***Copy and paste the plot from 3.4 here***
* ***Did the model parameters selected beat the out-of-the-box model for SVM?***

**Part 4 – Learning Curve Analysis with Bootstrapping**

In this HW we are trying to find the best linear model to predict if a record represents the Higgs Boson. One of the drivers of the performance of a model is the sample size of the training set. As a data scientist, sometimes you have to decide if you have enough data or if you should invest in more. We can use learning curve analysis to determine if we have reached a performance plateau. This will inform us on whether or not we should invest in more data (in this case it would be by running more experiments).

Given a training set of size N, we test the performance of a model trained on a subsample of size N\_i, where N\_i<=N. We can plot how performance grows as we move N\_i from 0 to N.

Because of the inherent randomness of subsamples of size N\_i, we should expect that any single sample of size N\_i might not be representative of an algorithm’s performance at a given training set size. To quantify this variance and get a better generalization, we will also use bootstrap analysis. In bootstrap analysis, we pull multiple samples of size N\_i, build a model, evaluate on a test set, and then take an average and standard error of the results.

Create a bootstrap function that can do the following:

def modBootstrapper(train, test, nruns, sampsize, lr, c):

1. Takes as input:
   1. A master training file (train)
   2. A master testing file (test)
   3. Number of bootstrap iterations (nruns)
   4. Size of a bootstrap sample (sampsize)
   5. An indicator variable to specific LR or SVM (lr=1)
   6. A c option (only applicable to SVM)
2. Runs a loop with bn iterations, within each loop:
   1. Sample bs instances from train, with replacement
   2. Fit either an SVM or LR (depending on options specified). For SVM, use the value of C identified using the 1 standard error method from part 3.
   3. Computes AUC on test data using predictions from model in 2.b
   4. Stores the AUC in a list
3. Returns the mean(AUC) and Standard Error(mean(AUC)) across all bootstrap samples

***Copy and paste your function here (as a guide, this function can easily be done in less than 20 lines)***

For both LR and SVM, run 20 bootstrap samples for each samplesize in the following list: samplesizes = [50, 100, 200, 500, 1000, 1500, 2000].

(Note, this might take 10-15 mins…feel free to go grab a drink or watch Youtube while this runs).

Generate a plot with the following:

* 1. Log2(samplesize) on the x-axis
  2. 2 sets of results lines, one for LR and one for SVM, the set should include
     1. 1 series with mean(AUC) for each sampsize (use ‘g’ for svm, ‘r’ for lr)
     2. 1 series with mean(AUC)-stderr(AUC) for each c (use ‘+’ as color pattern, ‘g’,’r’ for SVM, LR respectively)
     3. 1 series with mean(AUC)+stderr(AUC) for each c (use ‘--‘ as color pattern ‘g’,’r’ for SVM, LR respectively)

***Copy and paste the plot here, then answer the following questions:***

***Which of the two algorithms are more suitable for smaller sample sizes, given the set of features? If it costs twice the investment to run enough experiments to double the data, do you think it is a worthy investment?***

***Bonus Question (4 extra credit points):***

***If the “optimal” SVM chosen via cross-validation did not out-perform the out-of-the-box solution, using what you’ve observed with the learning curves, why do you think cross-validation chose the wrong optimal value of C (Hint: refer to ESL figure 7.8)? Is there a reason why cross-validation might be biased? If so, in what direction is it biased?***