

**Course:**

Intro to Data Science – DS-GA-1001 /

Data Mining for Business Analytics - INFO-GB.3336.11

Fall 2014

**Instructor:**

Brian Dalessandro

**Homework 4 – Solution**

**Part 1 – Data Preparation 5 Points**

***Copy and paste your function here (as a guide, this function can easily be done in less than 15 lines) (5 points)***

*Note for grading – it is not necessary to grade this line by line. The main criterion is that it is relatively concise and that it works. Also, it is not required that the students scale the data. Here are some guides for grading*

* *If the function is more than 20 lines, deduct 1 point.*
* *If the function does not take in a filename as an input, deduct 1 point*
* *If there is no function, but a sequence of lines/statements that load data, deduct 3 points*

def cleanBosonData(infile, scale):

dat = pd.read\_csv(infile, header = 0, sep = ',', index\_col = 0)

Y = (dat['Label'] == 's')\*1

dat = dat.drop('Label', 1)

flds = dat.columns.values

mv = -999.0

#Loop through the fields and replace mv with avg and create dummy vars

for f in flds:

#Don't do this for Y or if no missing vals

if (sum((dat[f]==mv))>0 and f !='Y'):

mu\_f = dat[f][(dat[f] != mv)].mean()

dat[f+"mv"] = (dat[f] != mv)\*1

dat[f] = dat[f].replace(to\_replace=mv, value=mu\_f)

if (scale==1):

flds2 = dat.columns

scl = preprocessing.StandardScaler()

sdat = pd.DataFrame(scl.fit\_transform(dat), columns = flds2)

sdat['Y'] = Y.values

return sdat

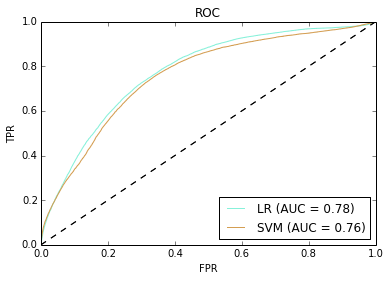
else:

dat['Y'] = Y

return dat

**Part 2 – Basic Evaluations 3 points**

* ***Copy and paste the plot here. (1 points)***

****

* ***Which of the two models is generally better at ranking the test set? (1 point)***

The out of the box Logistic Regression is better than the out of the box SVM.

* ***Are there any classification thresholds where the model identified above as ‘better’ would underperform the other in a classification metric (such as TPR)? (1 points)***

*Note to grader – the essence of this question is knowing how to interpret ROC curves that cross. Look for this in their answer and make sure it is consistent with the above ROC plots.*

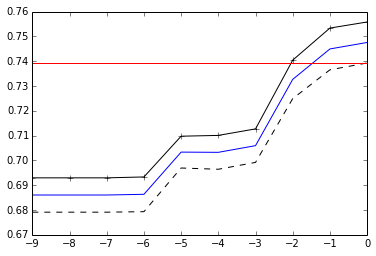
Despite the fact that the two lines seem to be identical for low FPR, it does not look like the two ROC lines ever cross, so there is no threshold where the LR underperforms the SVM.

**Part 3 – Model Selection with Cross-Validation 4 Points**

* ***Copy and paste the plot from 3.4 here (3 points)***

Note to grader, use the following for deducting points

* If there is no 1 std-error rule line (red), deduct 1 point
* If there are no std-error bounds, deduct 1 point
* If std-error bounds are much bigger than below, chances are they used std-dev of k folds and not std-error, deduct 1 point
* If x-axis contains lower than range [-9, 0], deduct 1 point
* If y-axis (AUC) results don’t reach at least 0.74, deduct 1 point

******

* ***Did the model parameters selected beat the out-of-the-box model for SVM? (1 points)***

In my analysis, the choices tested above did not beat the out-of-the box model. We can see that the k-fold averages look like they start to plateau around Log10(C)=0. The out-of-the-box implementation of SVM in Sklearn uses log10(C)=1. It thus turns out that the default choice happens to be a good one.

**Part 4 – Learning Curve Analysis with Bootstrapping (8 Points)**

***Copy and paste your function here (as a guide, this function can easily be done in less than 20 lines) (4 points)***

Notes for grading

* If the function is more than 20 lines (not including comments), deduct 1 point
* If the function definition isn’t exactly as below, deduct 1 point
* If they don’t do this as a function, but do it as lines of code, deduct 2 points

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

'''

Samples with replacement, runs multiple train/eval attempts

returns mean and stdev of AUC

'''

lab = 'Y'

auc\_res = []

for i in range(nruns):

train\_samp = train.iloc[np.random.randint(0, len(train), size=sampsize)]

if (lr == 1):

lr\_i = linear\_model.LogisticRegression(C=1e30)

lr\_i.fit(train\_samp.drop(lab,1),train\_samp[lab])

p = lr\_i.predict\_proba(test.drop(lab,1))[:,1]

else:

svm\_i = svm.SVC(kernel='linear', C=10\*\*best\_c\_ser)

svm\_i.fit(train\_samp.drop(lab,1),train\_samp[lab])

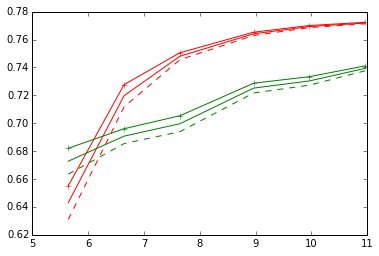
p = svm\_i.decision\_function(test.drop(lab,1))

auc\_res.append(roc\_auc\_score(test[lab], p))

return [mean(auc\_res),math.sqrt(var(auc\_res)/nruns)]

***Copy and paste the plot here, then answer the following questions: (2 points)***

Note – the LR is the one in red and x-axis is log(SampleSize)

******

***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? (2 points)***

The SVM does better for the absolute minimum sample size tested here, but that advantage seems like outlier behavior. Overall, the LR does substantially better for most small sample sizes. So in general, I would choose LR over SVM given the current sample size (or maybe less).

For Logistic Regression, we can see that performance reaches a plateau point and doubling the data will likely not add large value to prediction performance.

The SVM shows a near linear trade-off between sample size and AUC. It is likely that we can get more performance out of the SVM as we increase the data. We can not conclude though based on this evidence that doubling the data will double performance.

Extra Credit Note (1 Point) – If they mention the following add a point

We can not really answer this question precisely without knowing the economic value of increased AUC. It is unlikely that doubling the data will double the performance using either algorithm, but evaluating the investment depends on how much 1 or 2 points of additional AUC is worth.

***Bonus Question 1 (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?***

*The following statement is worth 1 point if answered correctly.*

Cross validation is negatively biased

*If they can identify the reason, it is worth 3 points*

The learning curves in figure 7.8 show that performance of a classifier generally decreases as the training data sample size decreases. The exact trade-off depends on the data and the classifier. Cross-validation only uses (k-1)/k% of the data for training. If the learning curve hasn’t reached its plateau at (k-1)/k% of the data, then the cross-validation procedure will be negatively biased due to the smaller sample size. We can see in the SVM learning curve that the curve is nearly linear at all sample size points, meaning that it hasn’t reached its performance plateau. Thus, we’d expect cross-validation in this case to be negatively biased, which has the potential to result in sub-optimal decision making.

***Bonus Question 2 (3 extra credit points)***

***If you can compare SVM with normalized data against SVM with unnormalized data for the values of C you explore, I'll throw in 3 extra credit points.***

*If they include a cross-validation plot that has normalized vs. unnormalized SVM then give them 3 points.*