9/13/2017

CSE 5243
Instructor: Jason Van Hulse
Homework 2

Due Date: 9/27/2017 5:30pm

In this lab, you will build upon the programs that you have written for homework #1. You will write a program to implement a k-Nearest Neighbor (kNN) classification algorithm and test this method on the Income dataset.

You will be provided with a test dataset, **income_te**. The program that you write should create the following output:

> An *mx4* data frame or table, where *m* is the number of examples in the *income_te* dataset. The *Actual Class* is the class of the record that is listed in the Test file, *Predicted Class* is the prediction that is made based on the kNN algorithm and training dataset, and *Posterior Probability* is the probability that the record belongs to the predicted class, based on the kNN algorithm.

Transaction ID	Actual Class	Predicted Class	Posterior Probability
1	<=50k	<=50k	0.550
2	<=50k	>50k	0.731

(this example is illustrative).

You will also compare your kNN implementation to any other off-the-shelf kNN implementation (e.g., the kNN classifier in R).

Report:
In addition to turning in the program, you should create a report (maximum of approximately 9 pages for individuals, 14 pages for teams of 2) which describes your program and the design decisions and analyzes the performance of your kNN classification algorithms.

Section 1: Describe the approach you took to train and test the model, and where applicable the rationale for the approach. Detailing what you did is very important even if it did not work. Describe any difficulties you may have encountered and any assumptions you are making.

Section 2: Evaluate the performance of the classifiers. You can vary the two parameters of your kNN algorithm: the number of neighbors *k* and the proximity.

A. Produce confusion matrices, classification and error rates for the test dataset for a few different values of *k*.
B. Compute True Positive, False Positive, True negative and False negative rates, Recall, Precision and F-Measure, as well as the ROC curve.
C. Provide detailed analysis of the results. What trends did you observe? Provide graphs and/or statistics to back up and support your observations.
D. You implemented 2 different proximities in homework #1. If you change the proximity, do the prediction results change significantly?
E. When you vary the number of nearest neighbors *k,* how does the performance on test data change? What value of *k* do you recommend for these datasets?
F. Did you need to make any changes from Homework #1 because your algorithm was not working well on the test data?

Some additional items to consider if working in a team of two:

G. The above analysis should be much more extensive. Come up with additional, creative ways to analyze the results.
H. For the Income data, analyze the results by looking at different levels of some of the independent variables. For example, did your kNN algorithm perform similarly for each value of *workclass* or *relationship?* If not, any reasons why you see these results?
I. For the income dataset, analyze what happens when you reduce the size of the training dataset. You can do this by randomly selecting 50% of the examples, or 25% of the examples, and using only these to classify instances in the Test dataset. What impact does size of the training dataset have on performance?

J. Try implementing a weighted posterior when determining the predicted class (i.e., adjust the voting algorithm to include the distance to each nearest neighbor somehow). How does this weighted posterior perform compared to the un-weighted version?

K. You can compute the classification performance measures on the training dataset as well. In other words, run the kNN classifier on the training dataset itself, and compute classification performance. How does the accuracy on the training dataset compare to the accuracy on the test dataset?

Section 3: Compare your kNN classifier to any off-the-shelf implementation of the kNN classifier. How did your algorithm compare? You do not need to turn in this code, just briefly describe the implementation you used and the parameter settings selected. Focus on comparing the classification performance of the off-the-shelf implementation to your program.

What you need to turn in:

1) Code
2) Readme - contains all the important information about the directory, including how to run the program and how to view the resulting output.
3) Report
   - (maximum of approximately 9 pages for individuals, 14 pages for teams of 2)
   - The report should be well-written. Please proof-read and remove spelling and grammar errors and typos. *Writing and presentation will be part of your grade for this assignment.*
   - *Please hand in a hard-copy of the written report in class on the due date.*

You do *not* need to turn in the output datasets, rather these will be obtained by running your code.

Note: It is expected for you to code these from scratch, and not to use existing functions.The only built in *mathematical* or *statistical* functions you should use are mean, median, standard deviation, minimum and maximum. *Please feel free to modify your programs from Homework #1 within this assignment.*

How to hand in your work:

Please choose one of the programming languages from: JAVA, Python, R. All the related files except for the data will be tarred in a **single** *.zip file or *.tgz file, and submitted via Carmen. Please use this naming convention: ”Project2_Surname_DotNumber.zip" or “Project2_Surname_DotNumber.tgz.” The submitted file should be less than 5MB.

On Linux System (Mac OS, Ubuntu, RedHat, etc.)

[Source Code, BashScript, Readme.txt, Report] should be submitted. Do not submit raw dataset.

The program should be able to run on a standard Linux system. Readme.txt tells me where to put input data (raw dataset) and where to find output data and how to interpret the output.

When I type command "bash BashScript", the output would be generated.

On Windows system

[Source Code, Readme.txt, Report] should be submitted. Do not submit raw dataset.

The program should be able to run on a Win-7 system. Readme.txt tells me where to put input data (raw dataset) and where to find output data and how to interpret the output. Readme also tells me how to compile and run the program.

If you use JAVA, please make sure your program can get compiled and run on Eclipse.