ANLY 555: Data Science Python Toolbox
Deliverable 3: Classifier and Experiment

Technical Resources:

- Commenting Standards:
  - https://www.python.org/dev/peps/pep-0257/#what-is-a-docstring
  - https://www.python.org/dev/peps/pep-0257/
- Documentation using Doxygen
  - http://www.doxygen.nl/
    - http://www.doxygen.nl/manual/
    - http://www.doxygen.nl/manual/docblocks.html#pythonblocks

Background

Throughout this course you will be designing and implementing a Data Science Toolbox using Python. There will be 5 major deliverables and required, supporting discussion posts. The first deliverable will be focused on designing the class hierarchy, building the basic coding infrastructure, and beginning the documentation process.

Overview of Deliverables

1. Design
2. Implement DataSets Class and subclasses
3. Implement ClassifierAlgorithms Class, simplekNNClassifier subclass and Experiment Class
4. Implement
   a. ROC method for Experiment Class,
   b. decisionTreeClassifier
5. Advanced Algorithms and final package

Software Design Requirements Overview

The toolbox will be implemented using OOP practices and will take advantage of inheritance and polymorphism. Specifically, the toolbox will consist of 3 main classes some of which have subclasses and member methods as noted below. You will also submit a demo script for each submission that tests the capabilities of your newly created toolbox.

1. Class Hierarchy
   a. DataSet
      i. TimeSeriesDataSet
      ii. TextDataSet
      iii. QuantDataSet
      iv. QualQuanDataSet

b. ClassifierAlgorithm
    i. simplekNNClassifier
c. Experiment

2. Member Methods for each Super Class (subclasses will have more specified members as well)
    a. DataSet
        i. __init__
        ii. _readFromCSV(…)
        iii. load(…)
        iv. clean(…)
        v. explore(…)
    b. Classifier
        i. __init__
        ii. train(…)
        iii. test(…)
    c. Experiment
        i. runCrossVal(…)
        ii. score(…)
        iii. __confusionMatrix(…)
3. Demo

Details for Deliverable #3. Classifer class and simplekNNClassifier subclass.

There are 4 main components for your deliverable this week. Focus on not only correctness, but also focus on efficiency of your algorithmic solution. Consider both time and space efficiency.

1. Using Python, you will implement the Classifier Class (as an ABC … note this was largely completed in previous deliverable) , simplekNNClassifier subclass, and Experiment Class. Implement methods train and test for the Classifier class and subclass. Implement runCrossVal, score, and confusionMatrix for the Experiment Class. You will also need to have member attributes to help store data and possibly state information.
    a. simplekNNClassifier
        i. The train method for simplekNNClassifier will have input parameters trainingData and true labels. It will simply store the data and labels member attributes (simple!!).
        ii. The test method will have parameters testData and k, and will find the k closest training samples and return the mode of the k labels associated from the k closest training samples. The predicted labels will be stored in a member attribute and will also be returned.
    b. Experiment
        i. The Experiment Constructor will take as input one data set , labels, and a list of classifiers. Each will be stored in a member attribute.
            1. Assume that each classifier will be applied to the data set, and if they are not applicable, one of the intrinsic exceptions to the classifier will be thrown.

ii. crossValidation method will take as input kFolds. This method will perform k-fold crossvalidation, and for each fold will train all classifiers (on the training folds), and test all classifiers on the testing folds.
    1. Predicted labels will be stored in a matrix numSamples x numClassifiers
iii. The score method will compute the accuracy of each classifier and present the result as a table.
iv. The confusionMatrix method will compute and display a confusion matrix for each classifier.

2. Perform formal computational Complexity Analysis on the following methods. Include a space count S(n) and step count T(n) function (where n is the size of the input) as well as a tight-fit upperbound using Big-O notation. Assume worst case and justify your analyses.
    a. simpleKNNClassifier test method
    b. Experiment score method
    c. Experiment confusion matrix method

3. Using python you will implement a demo script that tests the functionality of your code. You will test to the full functionality of the new code submitted for this deliverable. You will test all constructors and methods.
    a. Your demo test script must run without error. Be sure to include all data files and supporting files in the zip submission. Also be sure that all paths are relative and will work from the zip folder (once unzipped) as a base directory.
4. Using Doxygen (or another UML-like documentation tool), UPDATE your documentation which illustratively describes the class hierarchy, member attributes, and member methods. The description should include structural and functional details.

Constraints

All coding must be done by you – this means you must implement knn, score, and ROC "from scratch". For example, you must compute the distances used in knn from scratch (you cannot simply call dist() or math.dist(), instead you must code this yourself). You may not import any libraries / modules EXCEPT for those listed below, and you cannot repurpose any other code for this submission. The libraries below may be used to help with loading the data and visualizing the data only. If you wish to use another library, please ask for approval.

<u>Approved libraries to aid with reading in data, use of data structures, io, and visualizing data:</u>

- csv
- nltk
- numpy
- matplotlib.pyplot
- os
- wordcloud

Academic Integrity

This is an individual project and all work must be your own. Refer to the guidelines specified in the *Academic Honesty* section of this course syllabus or contact me if you have any questions.

Include the following comments at the start of your source code files:

```
/*
 * <FileName>.<file extension>
 *
 *  ANLY 5555 <term year>
 *  Project <>
 *
 *  Due on: <Due Date>
 *  Author: <your name>
 *
 *
 *  In accordance with the class policies and Georgetown's
 *  Honor Code, I certify that, with the exception of the
 *  class resources and those items noted below, I have neither
 *  given nor received any assistance on this project other than
 *  the TAs, *professor, textbook and teammates.
 *
 *  References not otherwise commented within the program source code.
 *  Note that you should not mention any help from the TAs, the professor,
 *  or any code taken from the class textbooks.
 */
```

These comments must appear **exactly** as shown above. The only difference will be values that you replace where there are "place holders" within angle brackets such as `<netID>` which should be replaced by your own `netID`.

Submission Details

Upload (as instructed by your professor) a zip folder containing ALL files (`.py, .pdf, and/or .html files`). Use the following folder name: `<firstname><Lastname>P3. For example, I would create a folder named jeremyBoltonP3 which contained all files. I would then zip this folder creating file jeremyBoltonP3.zip . I would then submit this zip file.` Late submissions will be penalized heavily. If you are late you may turn in the project to receive feedback but the grade may be zero. In general, requests for extensions will not be considered.