ANLY 555: Data Science Python Toolbox
Deliverable 4: Transaction Data Set, ROC, and Decision Tree

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 OR TransactionDataSet
5. ARM (greedy tree search ARM) OR hmmClassifier (dynamic programming) 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. QualDataSet

b. ClassifierAlgorithm
    i. simplekNNClassifier
    ii. decisionTreeClassifier
c. Experiment

** may be implemented in final week for extra credit

2. Member Methods for each Super Class (subclasses will have more specified members as well)
    a. DataSet
        i. __init__(self, filename)
        ii. __readFromCSV(self, filename)
        iii. __load(self, filename)
        iv. clean(self)
        v. explore(self)
    b. ClassifierAlgorithm
        i. __init__(self)
        ii. train(self)
        iii. test(self)
    c. Experiment
        i. runCrossVal(self, k)
        ii. score(self)
        iii. __confusionMatrix(self)

Details for Deliverable #4. Transaction Data Set, ROC for Experiment Class, and Decision Tree

There are 4 main components for your deliverable this week.

1. Using Python, you will implement a ROC method for the Experiment class that will produce a Receiver Operating Characteristic Curve based on the current experiment. ***And then you will implement 1 of the following (both not required!!)***. You will also implement a decisionTreeClassifier Class which is a subclass to Classifier AND a subclass to Tree (You will implement ABC Tree). OR you will implement TransactionDataSet a subset to the DataSet Class. **If both decisionTreeClassifier and TransactionDataSet are implemented, there will be up to a 25 point bonus!!**
    a. **Sorting (and/or heaps) for improved Efficiency.** Experiment ROC method; see supplemental journal article as reference
        i. For 2 class problems, the ROC method will produce a ROC plot which contains a ROC curve for each algorithm (overlaid if more than one algorithm is run in the experiment). If there are more than 2 classes, the ROC method will compute multiple (one versus all) curves. The figure will include a legend when multiple curves are created.
    b. **Multiple Inheritance.** decisionTreeClassifier use the supplemental journal article as reference. It will classify qualitative data or quantitative data sets.
        i. decisionTreeClassifier will inherit from two superclasses: Classifier and Tree. Read about multiple inheritance from these supplemental docs:
            1. 9.5.1. Multiple Inheritance. https://docs.python.org/3/tutorial/classes.html
            2. https://realpython.com/lessons/multiple-inheritance-python/

ii. The train method for decisionTreeClassifier will have input parameters trainingData and true labels. Both training and testing methods will follow specifications detailed in the supplemental decision tree paper.
iii. The test method will have parameters testData and will return labels for the test data as specified in the in the supplemental decision tree paper.
iv. The Tree ABC will have a member method __str__() that will allow the Tree to be printed to the console as an ascii string that can be subsequently rendered by tool: http://mshang.ca/syntree/ .

c. **Subset Generation.** TransactionDataSet a subclass of DataSet. Like all DataSets, TransactionDataSet will have the following attributes:
   i. Attributes
      1. __init__(self, filename)
      2. __readFromCSV(self, filename)
      3. __load(self, filename)
      4. clean(self)
      5. explore(self)
   ii. Notably and the crux of this implementation will be the implementation of ARM and specifically the apriori algorithm (from scratch). This is a good example of curtailed subset generation. Specifically, the explore(self) method will call member method __ARM__(supportThreshold = .25) and will compute the Support, Confidence, and Lift for all Rules above supportThreshold. The top 10 rules along with these three measures will be displayed to the console. To help with data organization, a Rule class will be implemented. See supplemental paper for details related to the Apriori algorithm and measures: Confidence, Lift, and Support.

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. Both decisionTreeClassifier train and test methods
   b. Experiment ROC 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. 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.

Approved libraries:

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

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>P4. For example, I would create a folder named jeremyBoltonP4 which contained all files. I would then zip this folder creating file jeremyBoltonP4.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.