ANLY 555: Data Science Python Toolbox
Deliverable 5: Advanced Topics

Technical Resources:

See canvas.

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, kdTreekNNClassifier subclass and Experiment Class
4. Implement
    a. ROC method for Experiment Class,
    b. simpleTreeClassifier
5. Advanced Topics:
    a. HeterogeneousDataSet
    b. lshKNNclassifier OR kdTreeKNNClassifier

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
        v. TransactionDataSet
        vi. HeterogeneousDataSet
    b. ClassifierAlgorithm
        i. simplekNNClassifier
        ii. decisionTreeClassifier
        iii. kdTreeKNNclassifier

iv. lshKNNclassifier

c. Experiment

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)
        iv. ROC

Details for Deliverable #5. Advanced Topics.

There are 3 main components for your deliverable this week.

1. Using Python, you need only implement one of the following options: a, b, or c. Given the differing degrees of difficulty, scoring will be as follows: Option a alone, the max possible score is 90/100. Implementing option b or c alone, the max possible score will be 100 / 100. Implementing any two, the max possible score will be 125 / 100.
    a. HeterogenousData via Composition of Objects. HeterogeneousDataSet Class have member attribute which is a list of DataSets. It will likely need other attributes to help maintain state.
        i. Just like other DataSets, it will have the following member methods / functionality
            1. Initializing and Loading the data
                a. **__init__**(self, filename)
                b. **__load**(self, filename)

            You can consider 2 use cases here (Pick one).

                1) You will build HeterogeneousDataSets from a collection of other data sets. For example you can instantiate some quantData and qualData and then instantiate the HeterogeneousDataSet using a constructor: HeterogeneousDataSet(list(quantData1, qualData1))
                2) You can prompt the user to help guide the initialization of the data, reading the data from specific files. Etc.
            2. **clean**(self)

a. This will simply call the clean methods from each of the individual datasets

3. **explore**(self)
    a. This will simply call the explore methods from each of the individual datasets

ii. You will add the following member **select(self, …),** which will select one of the constituent data sets. (Use case. You may wish to train or test on only a selected constituent dataset.)

b. **kdTreeKNNclassifier** use the supplemental journal article as reference.
    i. The train method for kdTreeKNNClassifier will have input parameters trainingData and true labels. Both training and testing methods fill follow specifications detailed in the kd-tree knn paper.
    ii. The test method will have parameters testData and k, and will find the k closest training samples and will compute and return labels for the test data as specified in the in the kd-tree knn paper.
    iii. 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.
        1. Both kdTreeKNNClassifier train and test methods
        2. Compare to simpleKNNClassifier methods previously analyzed.

c. **lshKNNclassifier**. LSH (Locality Sensitive Hashing -- *try Algorithm 2 from paper*). Build a hash table to find nearest neighbors and implement as a kNN classifier.
    i. 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.
        1. Both **lshKNNclassifier** train and test methods
        2. Compare time complexity AND accuracy to simpleKNNClassifier method previously analyzed.

2. 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.
3. 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.

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>P5`. For example, I would create a folder named `jeremyBoltonP5` which contained all files. I would then zip this folder creating file `jeremyBoltonP5.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.