ANLY 555: Data Science Python Toolbox

Deliverable 2: DataSets

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 ClassifierAlgorithm Class, simpleKNNClassifier, and Experiment Class
4. Implement ROC and kdTreeKNNClassifier
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. kdTreeKNNClassifier
      iii. hmmClassifier **

iv. graphkNNClassifier **

c. Experiment

** may be implemented in 5th deliverable for extra credit

2. Member Methods for each Super and Sub Class (subclasses will have more specified members as well to be added later). Each subclass will inherit superclass constructor. All other member methods will be overridden unless design deviation is well-justified.
    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 #2. DataSets class and subclasses.

d. DataSet Subclasses
    i. TimeSeriesDataSet
    ii. TextDataSet
    iii. QuantDataSet
    iv. QualDataSet
e. DataSet Members (to be overridden)
    i. __init__(self, filename)
    ii. __readFromCSV(self, filename)
    iii. __load(self, filename)
    iv. clean(self)
    v. explore(self)

There are 3 main components for your deliverable this week.

1. Using Python, you will implement the DataSet Class (as an ABC) with at least the noted 5 member methods. You will also need to have member attributes to help store data and possibly state information.

The member methods will likely behave differently for different types of data (which is why we are using this hierarchical OOP design).

a. The load function will prompt the user to enter the name of a file – assumedly which stores a data set to load.
    i. You may also interactively ask the user for info, such as type of data set, eg quantitative.
b. The clean method should "clean" data according to category of data as follows:
    i. Quant data should fill in missing values with the mean
    ii. Qual data should fill in missing values with the median or mode
    iii. Time series information should run a median filter with optional parameters which determine the filter size.
    iv. Text data should remove stop words (and feel free to stem and / or lemmatize).
c. The explore method should create at least 2 visualizations of the data. Note you will need to create appropriate visualizations given the category of data.
d. If any of the above methods encounters a runtime error (e.g a file loading error), an exception should be thrown. A message describing the exception should be included.

2. Using python you will implement a demo/test script that tests the functionality of your code. You will test to the full functionality of the DataSet subclasses. You will test all constructors and methods on each subclass.
    a. DataSets for testing. Feel free to choose data sets from each category for testing. As I am sure you have observed, the format of data sets is not really standardized, and thus building a "load" function that will work for all data sets is not practical. So feel free to test on a data set of your choice or use the following:

        Quant Data: https://www.kaggle.com/crawford/weekly-sales-transactions

        Qual Data: https://www.kaggle.com/c/kaggle-survey-2019/data (multiple choice)

        Text Data: https://www.kaggle.com/omkarsabnis/yelp-reviews-dataset

        TimeSeries: https://www.kaggle.com/shayanfazeli/heartbeat

    b. 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.
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.

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

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>P2`. For example, I would create a folder named jeremyBoltonP2 which contained all files. I would then

`zip this folder creating file jeremyBoltonP2.zip . I would then submit this zip file.` Late submissions will be penalized heavily – see rubric for details. 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.

Programming Skills

The programming skills required to complete this assignment include:

- Multiple File Organization
- UML Design
- OOP Design
- Inheritance
- Polymorphism

Grade Rubric (Knowing what constitutes each category is part of the assignment.)

Correct Multi-File Organization:	25
Correct Makefile:	15
Good Coding Practices:	20
Implementation Follows Specs:	20
Implemenation is Correct:	20
TOTAL:	100