CSE 231 Spring 2009

Programming Project 08

This assignment is worth 50 points (5.0% of the course grade) and must be **completed and turned in before 11:59 on Monday, March 23rd .**

Assignment Overview

This assignment will give you more experience on the use of:

1. Functions
2. Dictionaries, Lists and Sets

The goal of this project is to gain more practice in the use of functions and dictionaries. This project offers a lot of opportunity to modularize your code into functions. In general, any time you find yourself copying and pasting your code, you should probably place the copied code into a separate function and then call that function.

Problem Statement

Given a data file containing 30,162 records with values describing attributes of individuals, including whether or not her/his annual income exceeds $50,000, develop a simple rule-based classifier (more on what a classifier is later) that can be used to predict the class (<=50K or >50K) of a set of unknown records.

Background

Making predictions is hard and for as long as we've had computers, we've used them to try and make better predictions. In this project, we'll be writing a small program to predict whether or not an individual's annual income exceeds $50,000 when given values for 14 other attributes for this individual.

Each record is described by values for 15 attributes:

	Attributes	Range of Values
1	Age	Continuous
2	Work-class	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked
3	Fnlwgt	Continuous
4	Education	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool
5	Education-num	Continuous
6	Marital-status	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse
7	occupation	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces
8	Relationship	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
9	Race	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
10	Sex	Female, Male
11	Capital-gain	Continuous

12	Capital-loss	Continuous
13	Hours-per-week	Continuous
14	Native-country	Nominal
15	Class	>50K, <=50K

Some of these values are categorical and some are continuous. The attribute “Fnlwgt” and “Native-country” are not of interest to us so we can ignore them. The attribute “Education-num” is simply a numeric mapping of “Education”, so for calculation convenience, we will ignore the attribute “Education”. Thus, we will use attributes 1, 2, 5-13 to predict the value of the “class” attribute.

There are five tasks associated with this project.

1. **Create a training set.**
   A training set is a set of records where we know the value of the class. We’ll use the training set in the next step to build up a model that can be used to make predictions. We provide a function for you to use for this step.

2. **Train a simple classifier.**
   A classifier is a model of the problem such that when we’re given a new record we can compare the new record to the model in order to predict the class of the new record. We use the training set to build up this model. Our model is very simple. We record “averages” for the various attributes for those records that make more than 50K as well as averages for those that make less than 50K. We use this model to predict whether a new record (a set of attributes we have not used in our calculations of the model) makes more or less than 50K.

   After processing the training set, we end up with two models: a “>50K model” and a “<=50K model.” The “averages” are calculated as follows. If the attribute is continuous (a numeric value that varies within a range), we calculate the average value; if it is categorical (a numeric value that comes from only a discrete set of values), we calculate the ratio for each category.
3. **An example of the discrete attributes**.
   Calculating averages for attributes such as “Hours per week” is simple, just add up all the attributes and divide by the number. The discrete attributes are a little more interesting. Imagine that we have 10 records, all of which are examples of the >50K examples. If the “Relationship” attribute for 2 of these 10 records is “Wife”, 3 is “Own-child”, 2 is “Husband”, 1 is “Not-in-family”, 1 is “Other-relative”, and 1 is “Unmarried”, then the “Relationship” attribute in the >50K model would be as follows:

Relationship	Wife: 0.2 Own-child:0.3 Husband:0.2 Not-in-family:0.1 Other-relative:0.1 Unmarried:0.1

The <= 50 model is created similarly, but using the <= 50 records instead of the >50 records.

Now we will have two models that will be used to predict the “Class” attribute for records in the test set.

There are many different methods in the areas of Artificial Intelligence and Machine Learning that have been used by computer programmers to make predictions. Most of these methods rely heavily on statistics-based methods that use computers to crunch a lot of numbers. We're more interested in developing our programming skills than delving deep into statistics so we are going to use a very simple method to make predictions. That is to say, our classifier is probably not statistically sound but it serves as a good programming exercise as well as a good introduction to the problem of predicting classes.

(Furthermore, in the real world, we commonly face lots of issues that crop up with missing data, noisy data, or other problems. We don't face any of these issues in this assignment. It is safe to assume that all of the data is there and correct.)

4. **Create a test set.**
   A test set is a set of records where (1) the records were not used to train the classifier and (2) we know the value of the class.

   Since the classifier hasn't seen these records yet but we know the true class, we can test out the classifier's accuracy on these new records. The classifier will predict the class of the record which we can then compare to the actual class. This step is done for you.

5. **Apply the classifier to the test set.**
   For each record in the test set, if the majority of the new record's attributes are closer to >50K model, then the new record is predicted to be >50K, otherwise it is predicted to be <=50K. For the categorical attributes of the new record, we will compare the ratios of the category in two models. For example, if the "Relationship" attribute of the new record is "Wife", the ratio of "Wife" in the >50K model is 0.2, and the ratio of "Wife" in <=50K model record is 0.1, then we will say the "Relationship" attribute of the new record is closer to the >50K model.

6. **Report accuracy of classifier.**
   For each record in the test set, compare the predicted class to the actual class and then print out the accuracy of the classifier, indicating both the number correct and the total number and also the percentage correct.

Project Description / Specification

Finish implementing project08.py to complete the five tasks listed above.

Functions have been provided for you to create both the training and test sets. You'll need you to finish the functions to train the classifier, apply the classifier to the test set, and report the accuracy of the classifier. Stub functions have been started for each of these tasks.

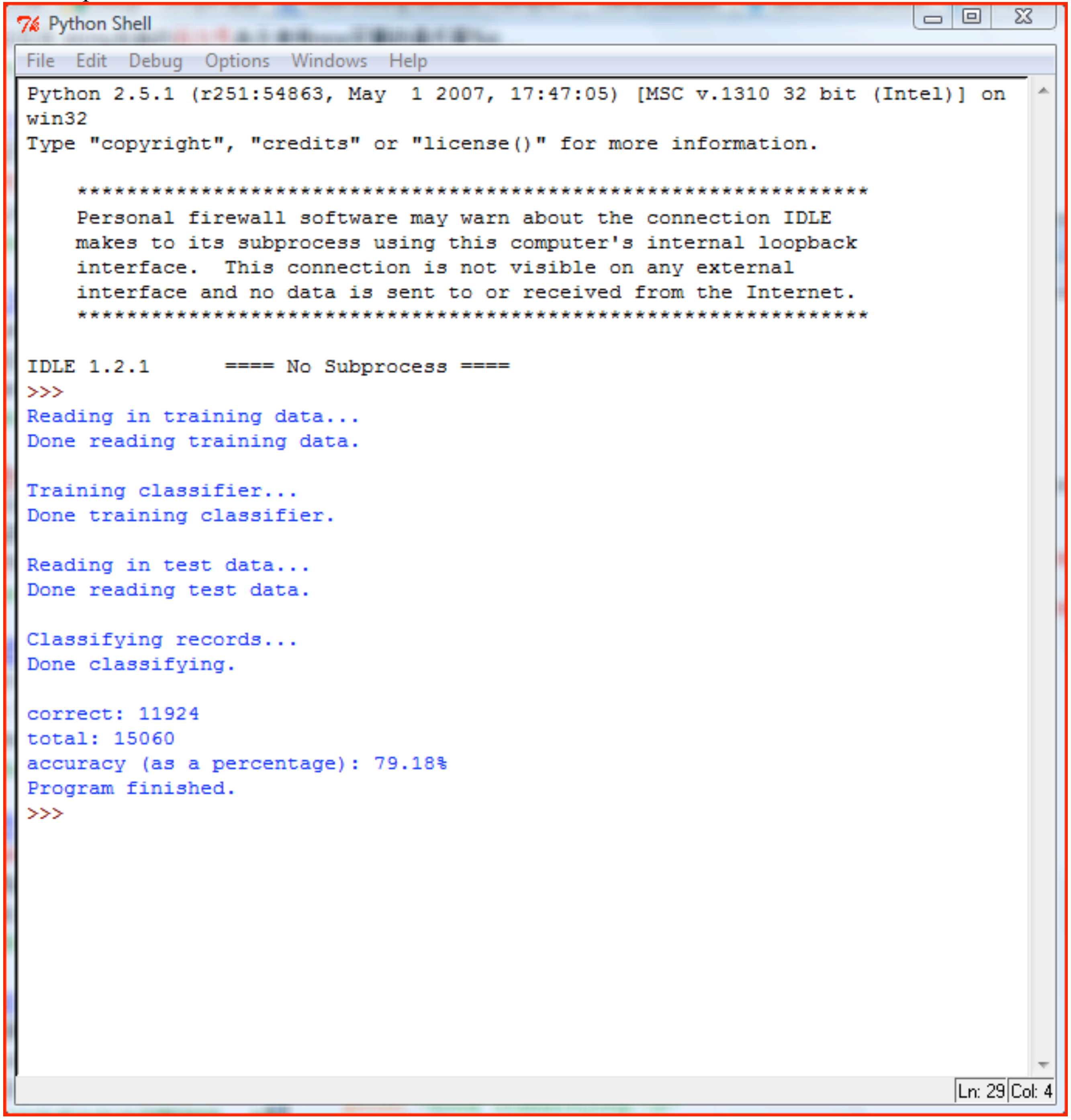
In addition to completing the three tasks, you must implement at least two non-trivial 'helper' functions that will be used to help complete the above tasks. (Hint: this should be especially useful when training the classifier to do something that's repeated several times such as initializing a dictionary for attributes or for calculating an average.) By non-trivial function, we mean that the helper functions should complete some meaningful task that requires passing of parameters and the use of return values. (Your helper functions should not just print one line out.)

Implementation Checklist:

1. Finish implementing the function "trainClassifier"

2. Implement your first helper function
3. Implement your second helper function
4. Finish implementing the function “classifyTestRecords”
5. Finish implementing the function “reportAccuracy”

The output should look like this:



```
Python 2.5.1 (r251:54863, May  1 2007, 17:47:05) [MSC v.1310 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 1.2.1      ==== No Subprocess ====
>>> 
Reading in training data...
Done reading training data.

Training classifier...
Done training classifier.

Reading in test data...
Done reading test data.

Classifying records...
Done classifying.

correct: 11924
total: 15060
accuracy (as a percentage): 79.18%
Program finished.
>>> 
```

Deliverables

proj08.py – your source code solution (remember to include your section, the date, project number and comments in your program).

1. Please be sure to use the specified file name, i.e. “proj08.py”
2. Save a copy of your file in your CSE account disk space (H drive on CSE computers).
3. You will electronically submit a copy of the file using the “handin” program: http://www.cse.msu.edu/handin/webclient

List of Files to Download

template.py
annual-income-training.data
annual-income-test.data

Notes and Hints:

The provided code is marked with ‘TODO’ where you need to modify the provided program.

The provided code has a lot of comments describing pseudo-code that should help you implement each function.

Start by familiarizing yourself with code and comments that are provided. It should run without any errors. (Don’t be confused by the print statements, the provided program only creates the training and test sets – the provided program doesn’t actually do anything with them other than read in the two files and store their contents in some data structures.)

The tasks have to be completed in order. You obviously can’t use a classifier before you’ve trained it.

Don’t try to tackle this project all at once. Complete one function (or part of a function) and test it out. Continuously save, run, and test your code as you’re working. The provided program doesn’t have any code for testing whether or not a function is correct. If you’re working on a function it would be helpful to add print statements so you can watch what your program is doing. You can comment these out later once you’re sure the code works.