CE705: Introduction to Programming in Python

Set by:
Due Date:
Assignment maximum mark: 100
Proportion to final module mark: 50%

SUBMISSION REQUIREMENTS:

Students are required to submit **ONE** single .py file containing the code for this assignment no later than 12:00 (UK time) on the . The standard lateness penalty will be applied to late work. Do NOT include any project file (i.e. no files other than the .py).

FEEDBACK FROM THIS ASSIGNMENT

Individual feedback will be provided within 4 weeks of the due date.

F.A.Q.

1) Can I submit anything other than a working .py file?
Absolutely not. You must submit a .py file that works “out-of-the-box” in Idle. If you submit anything else (e.g. ipynb, pyw, pyi, etc.) this will be an automatic fail. You must submit a .py file that works in Idle.

2) Can I import modules?
You can import any modules that come with Python (e.g. math, os, etc). You cannot use any module that requires extra installation (e.g. Pandas). The only exception to this rule is NumPy.

3) Can I make a small change in the return type of a function or method?
No. If a function or method is supposed to return a number, say 5, and you return ‘5’, [5] or anything other than just 5, you will lose all marks related to this function or method.

4) Can I make a small change in the data type of a parameter?
No. Such changes will lead to you losing all marks related to the function or method in question.

5) Can I add or remove a parameter?
No. Such changes will lead to you losing all marks related to the function or method in question.

6) Can I make a minor change in the name of a function or method?
No. Such changes will lead to you losing all marks related to the function or method in question. Please note that Python is case-sensitive. For instance, the name run_test is not the same thing as Run_Test.

7) Can I implement extra functions or methods to make my code easier/cleaner?
Yes. Please note you must implement all the functions and methods described in the assignment brief. If you’d like to implement more, you are welcome to do so.

8) Can I implement the algorithm in this assignment in any other way than what the assignment brief describes?
No. In this assignment we are trying to measure your ability to code a programme following a specification. Hence, you must follow this specification.

9) Does my code need to work only for the data set provided?
No, it should work for any data set. In other words, do not hardcode values such as the number of rows, the number of columns, etc.

10) Why am I not allowed to make changes?
Large pieces of software (e.g. Windows) are not written by a single programmer, but by many. All programmers will be working on different parts of the software, but all of these parts are likely to interact in some way. The programme specification makes sure everybody knows what each function expects to receive and what each function should return. If one programmer unilaterally decides to make a small change that goes against the specification… then the software will not work as expected.

Assignment: identifying groups of similar wines

A sommelier is a trained professional who spends his or her day tasting different wines, and identifying similarities (or sometimes dissimilarities) between these. Given this is clearly an *exhausting* task, you have been hired to develop a software capable of grouping similar wines together. Your software will load a data set containing information about each wine (Alcohol content, alkalinity of ash, Proanthocyanins, colour intensity, etc) and identify which wines are similar.

Luckily, your employer has already identified a suitable algorithm and designed the software for you. All you are required to do is to write the actual source code (with comments).

Technical details:
You'll be using different data structures to accomplish the below. Your assignment must contain the code for the functions and methods below. If you wish you can write more functions and methods, but those described below must be present.

1) Class: matrix
You will code a class called matrix, which will have an attribute called **array_2d**. This attribute is supposed to be a NumPy array containing numbers in two dimensions. The class matrix must have the following methods:
(in these, the parameters are in addition to `self`)

load_from_csv
This method should have one parameter, a file name (including, if necessary, its path and extension). This method should read this CSV file and load its data to the array_2d of matrix. Each row in this file should be a row in array_2d. Notice that in CSV files a comma separates columns (CSV = comma separated values).
You should also write code so that
`m = matrix('validfilename.csv')`
Creates a matrix m with the data in the file above in array_2d.

standardise
This method should have no parameters. It should standardise the array_2d in the matrix calling this method. For details on how to standardise a matrix, read the appendix.

get_distance
This method should have three parameters, two matrices (let us call them other_matrix and weights) and a number (let us call it beta). If the matrix calling this method and the matrix weights have only one row, this function should return a matrix containing the weighted Euclidean distance between the row in the matrix calling this method and each of the rows in other_matrix. For details about how to calculate this distance, read the appendix.
To be clear: if other_matrix has n rows, the matrix returned in this method will have n rows and 1 column.

get_count_frequency
This method should have no parametes, and it should work if the array_2d of the matrix calling this method has only one column. This method should return a dictionary mapping each element of the array_2d to the number of times this element appears in array_2d.

2) Functions
The code should also have the functions (i.e. not methods, so not part of the class matrix) below. No code should be outside any function or method in this assignment.

get_initial_weights
This function should have one parameter, an integer m. This function should return a matrix with 1 row and m columns containing random values, each between zero and one. The sum of these m values should be equal to one.

get_centroids
This function should have three parameters: (i) a matrix containing the data, (iii) the matrix **S**, (iii) the value of K. This function should implement the Step 9 of the algorithm described in the appendix. It should return a matrix containing K rows and the same number of columns as the matrix containing the data.

get_groups
This function should have three parameters: a matrix containing the data, and the number of groups to be created (K), and a number beta (for the distance calculation). This function follows the algorithm described in the appendix. It should return a matrix **S** (defined in the appendix). This function should use the other functions you wrote as much as possible. Do not keep repeating code you already wrote.

get_new_weights
This function takes three parameters: a matrix containing the data, a matrix containing the centroids, and a matrix **S** (see the algorithm in the Appendix). This function should return a new matrix weights with 1 row and as many columns as the matrix containing the data (and the matrix containing the centroids). Follow Step 10 of the algorithm in the Appendix.

run_test
Your code must contain the function below (do not change anything)

```
def run_test():
   m = matrix('Data.csv')
   for k in range(2,5):
        for beta in range(11,25):
             S = get_groups(m, k, beta/10)
             print(str(k)+'-'+str(beta)+'='+str(S.get_count_frequency()))
```

The aim of this function is just to run a series of tests. By consequence, here (and only here) you can use hard-coded values for the strings containing the filenames of data and values for K.

More details

You will implement a data-driven algorithm that creates groups of entities (here, an entity is a wine, described as a row in our data matrix) that are similar. If two entities are assigned to the same group by the algorithm, it means they are similar. This will create groups of similar wines. Your software just needs the number of groups the user wants to partition the data into, the data itself, and a numeric value for Beta.

The number of partitions (K) is clearly a positive integer. Your software should only allow values in the interval [2, n-1], where n is the number of rows in the data. This way you'll avoid trivial partitions. You can test values of Beta that are higher than 1.

Your software should follow the algorithm described in the appendix and generate a matrix **S** indicating to which group (1, 2, …, K) each entity (wine, a row in the data matrix) has been assigned to. Clearly **S** will have n elements.

You can find more information online if you search for clustering, or k-means (this is not the algorithm we are implementing, but it is similar).

For the brave:
You are implementing this algorithm: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1407871
If you cannot understand the paper in the link, do NOT panic. You do NOT need to understand the paper above in order to do this assignment. You CAN get an excellent mark in this assignment without even opening this link. If you are not strong in maths (multivariable calculus, in particular) do NOT open the link.

Appendix

Data standardization

Let D be a data matrix, so that D_{ij} is the value of D at row i and column j. You can standardize D by following the equation below.

$$D'_{ij} = \frac{D_{ij} - \overline{D_j}}{max(D_j) - min\,(D_j)}$$

where $\overline{D_j}$ is the average of column j, $max(D_j)$ is the highest value in column j, and $min\,(D_j)$ is the lowest value in column j. D'_{ij} is the standardized version of D_{ij} – the algorithm below should **only** be applied to D'_{ij} (i.e. do not apply the algorithm below to D_{ij}).

Basic notation for the algorithm
n = number of rows of in the data matrix
m = number of columns in the data matrix
K = number of clusters (notice that k is not the same thing as K)
Beta = exponent used in the distance calculation

Clustering algorithm

1. Set a positive value for *K,* and a positive value for Beta.
2. Initialise a matrix called weights with 1 row and m columns. Each value in this matrix should be between zero and one, and the sum of all values in weights should be equal to one.
3. Create an empty matrix called centroids.
4. Create a matrix called **S** with n rows and 1 column, initialise all of its elements to zero.
5. Select *K* **different** rows from the data matrix at random.
6. For each of the selected rows
    a. Copy its values to the matrix centroids.
    (at the end of step 6, centroids should have K rows and m columns)
7. For each row i in the data matrix
    a. Calculate the weighted Euclidean distance between data row D'_i and each row in centroids (use weights and Beta in this calculation, as per equation in the Appendix).
    b. Set S_i to be equal to the index of the row in centroids that is the nearest to the row D'_i. For instance, if the nearest row in centroids is row 3, then assign the number 3 to row i in **S**.
8. If the previous step does not change **S**, stop.
9. For each *k* = 1, 2, …, *K*
    a. Update the *k* row in centroids. Each element *j* of this row should be equal to the mean of the column D'_j but only taking into consideration those rows whose value in **S** is equal to k (i.e. those who have been assigned to cluster *k*).
10. For each v = 1, 2, …, m
    a. Update the entry v of the matrix weights (see Appendix).
11. Go to Step 7.

Weighted Euclidean distance
There are different weighted Euclidean distances, in this assignment you must follow the below. The distance between a vector **a** and a vector **b**, using the weights in a vector **w** (all three vectors with size m), with a value for Beta β is given by:

$$d = \sum_{i=1}^{m} w_i^{\beta}(a_i - b_i)^2$$

Calculating weights

Weights (w in the below) is a vector (a matrix with 1 row and m columns in your implementation). To calculate the entry j of this vector (i.e. w_j, that is, row 1 column j of the matrix weights) we first need to calculate the dispersion of the column j in the data matrix:

$$\Delta_j = \sum_{k=1}^{K} \sum_{i=1}^{n} u_{ik} \left(D'_{ij} - c_{kj} \right)^2$$

where,

n is the number of rows in the data matrix

m is the number of columns in the data matrix (used below)

K is the number of clusters (remember k is not the same thing as K)

c is the matrix centroids

u_{ik} is equal to one if $S_i = k$, and zero otherwise.

If the value $\Delta_j = 0$ then $w_j = 0$, otherwise:

$$w_j = \frac{1}{\sum_{t=1}^{m} \left[\frac{\Delta_j}{\Delta_t} \right]^{\frac{1}{\beta - 1}}}$$

Marking Scheme

Characteristics of an excellent project (70% or more):

- Excellent code documentation
- Excellent use of Python's native methods and code standards
- Excellent use of relevant data types
- Follows carefully the specification provided
- Implements the described run_test, which shows the expected results.
- Excellent code optimisation in terms of memory, speed and readability
- Generally, an excellent solution, carefully worked out;

Characteristics of a good project (60%):

- Good code documentation
- Good use of Python's code standards
- Good use of relevant data types
- Follows the specification provided, with no major deviations.
- Implements the described run_test, which shows the expected results.
- Good code optimisation in terms of memory, speed and readability
- Generally a good solution, which delivers what the final user would expect.

Characteristics of a fair project (50% or less):

- No meaningful code documentation
- Code tends to be more verbose than actually needed or at times difficult to read
- No real thought on the relevance of data types
- Does not follow the specification provided (this alone will indicate a fail).
- It contains code that is outside of a function or method
- It is not a .py file that runs in Idle (this alone will indicate a fail)
- It keeps printing things on the screen (run_test is an exception to this).
- Does not implement run_test as described, or this does not show the expected results.
- A solution that only seems to deliver what the final user would expect.

Please note:

- You must submit only one file
- You must follow the instructions for each function and method in terms of parameters and returns
- You should document your code.
- It is a good idea to test each function/method at a time, and only afterwards test the whole project