FINAL PROJECT

I590 – PYTHON

TABLE OF CONTENTS

OBJECTIVE

Implement “k-means” algorithm for Wisconsin Breast Cancer data using Python.

PROBLEM STATEMENT

Breast cancer is a rising issue among women. A cancer’s stage is a crucial factor in deciding what treatment options to recommend, and in determining the patient’s prognosis. Today, in the United States, approximately one in eight women over their lifetime has a risk of developing breast cancer. An analysis of the most recent data has shown that the survival rate is 88% after 5 years of diagnosis and 80% after 10 years of diagnosis. With early detection and treatment, it is possible that this type of cancer will go into remission. In such a case, the worse fear of a cancer patient is the recurrence of the cancer.

Now it is time for you to address this issue and implement one of the most popular data mining technique, k-means clustering on famous Wisconsin Breast Cancer Data. At the end of this project, you are going to classify benign and malign cells in two different groups.

DATASET DESCRIPTION

Wolberg’s breast cancer data can be found here. Samples arrive periodically as Dr. Wolberg reports his clinical cases. There are total 11 columns or features in this dataset.

Column/Attribute Information		Column/Attribute name
1. Sample code number:	id number	ID
2. Clump Thickness:	1 - 10	A2
3. Uniformity of Cell Size:	1 - 10	A3
4. Uniformity of Cell Shape:	1 - 10	A4
5. Marginal Adhesion:	1 - 10	A5
6. Single Epithelial Cell Size:	1 - 10	A6
7. Bare Nuclei:	1 - 10	A7
8. Bland Chromatin:	1 - 10	A8
9. Normal Nuclei:	1 - 10	A9
10. Mitoses:	1 - 10	A10
11. Class:	2 for benign, 4 for malignant	Class

PHASE 1 (30%)

This Project has been designed to be completed in three weeks. Each week you need to deliver different phases of the project.

During the first week, you need to complete following data analysis tasks.

- Get yourself comfortable with k-means algorithm
- Download the data and load it in Python
- Add Headers (Scn A2 A3 A4 A5 A6 A7 A8 A9 A10 CLASS)
- Impute missing values
- Plot basic graphs
- Compute data statistics

DELIVERABLES

a. Download Breast cancer data from UCI machine learning repository.
Link:

https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data

b. Load dataset into Python.

Hint: you may use *pandas* library to load dataset. You may want to use option `na_values="?"` to replace missing values entered to column A7 as "?" with `NaN`.

c. Impute missing value to column A7. The missing values will be either "?" or NaN. You may replace missing values using techniques like mean, median, mode imputation or any other methods of your choice.

Example: If you want to impute missing values by 'mean' imputation method, you first need to compute the mean of column 'A7' without considering '?' data. Once you compute the mean, replace '?' value with computed mean value.

d. Plot histograms for attributes A2 to A10 (nine histograms).

Hint: A histogram is a kind of bar plot that gives a discretized display of value frequency. The data points are split into discrete, evenly spaced bins, and the number of data points in each bin is plotted. Use *Matplotlib* and "*hist*" method on the Series to plot a histogram.

For example, if *s* is a *Series*, the following statement will plot a histogram with 16 blue bins and opacity 0.5:

```
heights.hist(bins=16, color = "b", alpha = 0.5)
```

Play with bin size parameter to make histogram more appealing.

e. Find the mean, median, standard deviation and variance of each of the attributes A2 to A10. So you will have total of nine mean, median, standard deviation and variance values.

SUBMISSION GUIDELINES

- Create a public github with at least three folders Phase 1, Phase 2, Phase 3 and subfolders (e.g., Data, Code) + readme
- Write your programs in Python 3, and use NumPy, pandas and matplotlib for the computation and plotting.
- Include a header to program file with program description, date and your name. Your code should be properly formatted, structured, indented and commented. Make sure that your program compiles.
- No constraint on how many functions and python scripts you write to perform this task. However, you should have an executable python script named 'main.py' that itself will be able to run your entire code by directly executing it.
- Implement your code using Jupyter notebook - Ideally you will use Markdown for sections, description of your steps and Code lines for 1. importing, 2. cleaning, 3. stats. Make sure to display all your graphs in jupyter.
- Please submit a link to your github with your code, dataset, and notebook, readme.md.
- Describe each team member individual contribution for phase I in'readme.md' file.

(See next page for Phase 2)

Working with Matplot in Jupyter:
First code cell (add all your library ihere if you use):

```
%matplotlib inline
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
```

Second code cell - code for importing dataset
Next cell - see example for histogram data:

```
x = np.random.normal(size = 1000)
plt.hist(x, normed=True, bins=30)
plt.ylabel('Probability')
plt.xlabel('Weight')
#Display histogram
plot.show()
```

PHASE 2 (40%)

In the second phase you will implement k-means algorithm:

- Revise k-means algorithm before writing code.
- Write code for 'Initialization' step
- Write code for 'Assignment' step
- Write code for 'Recalculation'

DELIVERABLES

In the dataset, 'id' is the first and 'class' the last column. We do not consider these two columns for k-means computation. However, you will need these two columns for printing results and of course in phase 3 to calculate error. Use columns A2 to A10 for k-means computation.

a. Use Dataset with imputed missing values from phase 1.

b. Write code for 'Initialization' step for *K = 2*.

   Steps:

   Choose any two points randomly from dataset as the initial means. Since you only consider column A2 to A10, a mean is nothing but a nine dimensional vector. Give first mean variable name μ_2 and second mean variable name μ_4. The two means represent two clusters.

   Example:

   Let's say randomly selected datapoints are 6 and 246 as two initial means. So values of μ_2 and μ_4 are

$$\mu_2 = (8, 10, 10, 8, 7, 10, 9, 7, 1)\ ;\ \ \mu_4 = (5, 1, 1, 2, 2, 2, 3, 1, 1)$$

   Note: You need to select two means randomly. So everytime the code runs, two different means are selected. You may use 'numpy random' library for selecting random points.

c. Write code for 'Assignment' step

   You have defined two means in the previous step. Now, for each one of 699 datapoints compute euclidian distance from the two means.

   For each datapoint you will have two distances. Assign a datapoint to cluster 2 if its distance from μ_2 is closer than its distance from μ_4, assign the datapoint to cluster 4 otherwise.

   At the end of this step, you have each point assigned to one of the two clusters.

Example:

Let's take datapoint at row number 375 and compute its distance from both means.

d375 = (3, 1, 2, 1, 2, 1, 2, 1, 1)

$$d(375, \mu_2) = \sqrt{(3-8)^2 + (1-10)^2 + (2-10)^2 \dots\dots + (1-1)^2} = 20.25$$
$$d(375, \mu_4) = \sqrt{(3-5)^2 + (1-1)^2 + (2-1)^2 \dots\dots + (1-1)^2} = 2.83$$

Since $d(375, \mu_4) < d(375, \mu_2)$, point at row number 375 will be assigned to cluster 4.

d. Write code for 'Recalculation' step

So far, you have got every datapoint assigned to one of the two clusters. Next, you will update the means.

Example:

Let's say after performing step b, you have 300 datapoints assigned to cluster 2 and remaining 399 datapoints assigned to cluster 4. Update μ_2 by computing the mean from cluster 2 datapoints and update μ_4 by computing the mean from cluster 4 datapoints.

e. Iterate steps *c* and *d* untill any one of the following conditions is true:

1. All the datapoints do not change their cluster assignment compared to the previous iteration.
2. Steps *c* and *d* iterated 50 times.

In step c, use μ_2 and μ_4 values you computed in step d of the previous iteration.

At the end step e, you will have final values of means and information about which datapoints are assigned to which cluster. Print this result in console.

Your console output may look like this:

```
-------------------------Final mean---------------------------------
mu_2:  [3.0472103004291844, 1.3025751072961373, 1.446351931330472, 1.3433476394849786, 2.0879828326180259, 1.3800011310866602, 2.1051502145922747, 1.2618025751072961, 1.109442060085837]
mu_4:  [7.1587982832618025, 6.7982832618025748, 6.7296137339055795, 5.733905579399142, 5.4721030042918457, 7.8739655269921256, 6.1030042918454939, 6.0772532188841204, 2.5493562231759657]


-------------------------Cluster assignment---------------------------------
         ID  Class Predicted_Class
0   1000025      2               2
1   1002945      2               4
2   1015425      2               2
3   1016277      2               4
4   1017023      2               2
5   1017122      4               4
6   1018099      2               2
7   1018561      2               2
8   1033078      2               2
9   1033078      2               2
10  1035283      2               2
11  1036172      2               2
12  1041801      4               2
13  1043999      2               2
14  1044572      4               4
15  1047630      4               2
16  1048672      2               2
17  1049815      2               2
18  1050670      4               4
19  1050718      2               2
20  1054590      4               4
```

Note: Your output may be different than the above. Include the first 20 datapoints (rows) in your report.

This version of k-means algorithm may suffer from poor initialization. Therefore, you may see different answers or swapped cluster assignments. We recommend running program multiple times and submitting the best result.

SUBMISSION GUIDELINES

- Write your programs in Python 3, and use NumPy, pandas and matplotlib for the computation and plotting.
- Include a header to program file with program description, date and your name. Your code should be properly formatted, structured, indented and commented. Make sure that your program compiles.
- No constraint on how many functions and python scripts you write to perform this task. However, you should have an executable python script named ‘main.py’ that itself will be able to run your entire code by directly executing it.
- Create a second Jupyter notebook file that imports dataset, create a function for k-means algorithm and output the first 20 datapoints (rows) in your report.
- Please submit a link to Phase II github folder zip file with your code, notebook and readme.
- Describe each team member individual contribution for phase II in ‘readme.md’ file.

(See next page for Phase 3)

PHASE 3 (30%)

During the third week, you will analyze the quality of the clustering.

- Write a code to calculate the individual and total error rate of your 2 clusters.
- Submit final report

DELIVERABLES

Upon stopping your K means algorithm, you will have two clusters - one which contains malign cells (cluster = 4) and the other containing benign cells (cluster = 2). But there are chances that a malign cell is being clustered into a benign cluster and vice versa. To check how well your clustering worked, you need to calculate the error rate for each of your cluster.

a. Write code for 'ErrorRate' for K value 2.

Steps:
Your ErrorRate function will take two input arguments. First argument is cluster you got from your k-means algorithm, and second argument is the actual value of cluster for a particular datapoint. Actual cluster value of any datapoint is nothing but the corresponding 'class' column value.

For example: let's say the following data points are found to be part of cluster 2 i.e μ_2 after you run your k-means algorithm:

1017023,4,1,1,3,2,1,3,1,1,2	These two points belong to benign cells (cluster =2) since their class is 2. Here, you got cluster 2 and actual cluster is also 2. So No error for these two datapoints.
1018561,2,1,2,1,2,1,3,1,1,2	
1054593,10,5,5,3,6,7,7,10,1,4	This point is identified as malign cell (cluster = 4) since its class is 4. Here, you got cluster 2 and actual cluster is 4. So it is an error for this datapoint.

Now you have to calculate error rate using the following formula for each clusters:

1. For μ_2:

$$error\ B = \frac{total\ number\ of\ datapoints\ \ with\ Predicted\ class = 4\ coressponding\ to\ Actual\ class = 2}{total\ number\ of\ datapoints\ with\ Predicted\ class = 2}$$

2. For μ_4:

$$error\ M = \frac{total\ number\ of\ datapoints\ \ with\ Predicted\ class = 2\ coressponding\ to\ Actual\ class = 4}{total\ number\ of\ datapoints\ with\ Predicted\ class = 4}$$

3. Total errror rate

$$\text{Total } error\ rate = \frac{total\ number\ of\ datapoints\ \ with\ Predicted\ class\ \neq\ Actual\ class}{total\ number\ of\ datapoints}$$

Print above results in console.

b. Prepare final report that incorporates all the results and your comments for Phases 1 to 3.

SUBMISSION GUIDELINES

- Write your programs in Python 3, and use NumPy, pandas and matplotlib for the computation and plotting.
- Include a header to program file with program description, date and your name. Your code should be properly formatted, structured, indented and commented. Make sure that your program compiles.
- Follow the same guidelines for submission on github for Phase III folder
- Use Jupyter notebook to write a final report - summary of three phases
- Provide each team member contribution in 'readme.md'.