CSC4/522: Homework 2-3R. Released: 2/19/18; Due: 3/14/18
Instructor: Ranga Raju Vatsavai

How the code works (read this before you start writing your code):

1. **your_unity_id.R -** Contains Functions for
    a. loading data for kmeans and knn (code already provided)
    b. Plotting outcomes of clustering (code already provided)
    c. bisecting kmeans (you need to implement),
    d. knn (you need to implement)
    e. comparison between bisecting kmeans and vanilla kmeans (you need to implement)
    f. accuracy measures calculation for KNN (you need to implement)

    You will rename this file (for eg. if your unity id is kgadira, you will rename this to kgadira.R).
2. **main.R** - where you will call the functions to generate results/plots - code is already provided for you. DONOT change any part of this file, except for the value of the variables your_unity_id, clustering_k and classification_k. DONOT submit this file. TA will use his own version of this file.

Instructions:

- Ensure that you name the variables and functions as instructed. If your code doesn't follow the naming convention/function format (function name, argument types, argument order) specified, it will not be graded.
- Before asking questions about the implementation, refer to the FAQ posted under each question.
- You will submit a zipped file containing:
    - A PDF of the plots and explanations requested in Q1 (named your_unity_id.pdf)
    - Your solution for your_unity_id.R renamed using your unity id as specified above.

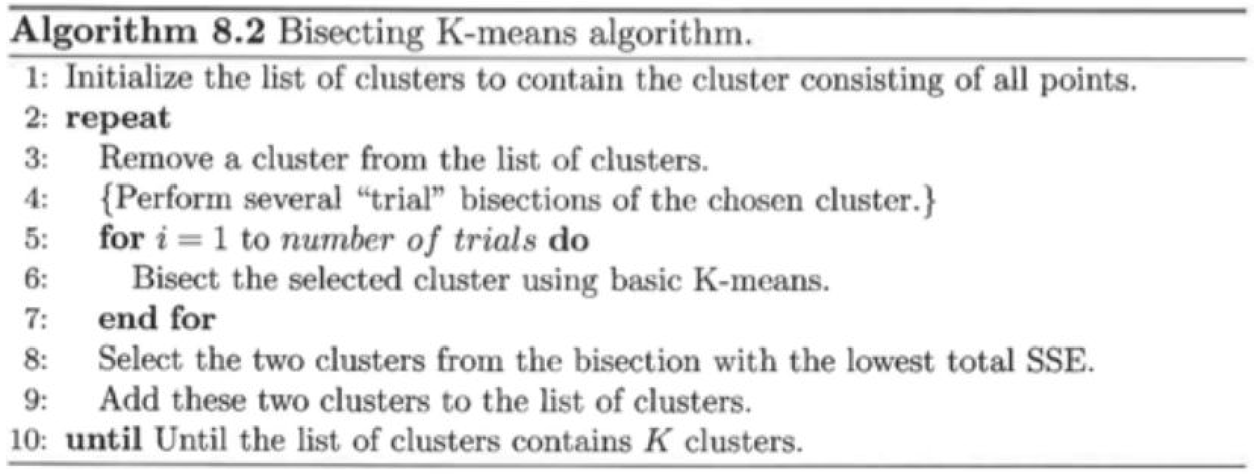
Implementing Supervised and Unsupervised Learning Algorithms:

For this project, you will be implementing the following two methods on the popular iris dataset:

1) bisecting K-Means.
2) K-Nearest Neighbors

Q1) (15 points) Bisecting KMeans

The bisecting KMeans algorithm is as shown below (source: Textbook):



Algorithm 8.2 Bisecting K-means algorithm.

```
1: Initialize the list of clusters to contain the cluster consisting of all points.
2: repeat
3:    Remove a cluster from the list of clusters.
4:    {Perform several "trial" bisections of the chosen cluster.}
5:    for $i = 1$ to $number\ of\ trials$ do
6:       Bisect the selected cluster using basic K-means.
7:    end for
8:    Select the two clusters from the bisection with the lowest total SSE.
9:    Add these two clusters to the list of clusters.
10: until Until the list of clusters contains $K$ clusters.
```

a) (15 points) implement the bisecting kmeans method. Write your implementation in '**bkm'** function provided in the **your_unity_id.R** file.

i) Input:

1) data.df: input data frame using the load_data_bkm() function from utils.R
2) iter.max: Max. number of iterations (trials) for kmeans, as per Algorithm 8.2. You are allowed to use the pre-defined kmeans() function.
3) k: Number of clusters to find in bisecting k-means

ii) Returns:

1) Your output/function return value will be a list containing 2 elements
    a) first element of the list is a vector containing cluster assignments (i.e., values 1 to k assigned to each data point)
    b) second element of the list is a vector containing SSE of each cluster

FAQ:

1) Which cluster to split in each iteration?
   A: When identifying which cluster to split, choose the one with maximum SSE
2) How do I select centers for KMeans?
   A: When performing kmeans, pick two random points the cluster with largest SSE as centers.
3) What is the terminating condition for bisecting kmeans?
   A: Stop when **k** clusters have been found
4) Which KMeans algorithm am I supposed to use?
   A: Use the "Lloyd" algorithm
5) Do I need to implement SSE calculation?
   A: No need, explore the KMeans function and what its output is

b) (5 points) Comparison with kmeans: Write your implementation in '**kmeans_comparison**' function provided in the **your_unity_id.R** file.
implementation:
Now use the same dataset from (a), and using points with ID values (15,16 and 19) as initial centers, identify the 3 clusters (use default settings for everything else) and do the following:

a) Run kmeans clustering using the centers mentioned above
b) Plot the clustering outcome of KMeans (use Sepal.Length as x-axis, and Petal.Length as y-axis)
c) Compare the the clusters you found here with the clusters you found in (a). Did you notice any difference (visually)? Compare in terms of size and shape of the clusters and overall SSE. Submit your findings (including the plots for Bisecting KMeans and KMeans) in the PDF.

Function arguments:
i) Input:

1) data.df: Dataset used for kmeans/bisecting kmeans clustering
2) result: Variable of type list, obtained on running bkm() function

k : k value for k-means
km_centers : id's of rows for kmeans

Returns:
None

Q2) (15 points) K Nearest Neighbors Classification

i) (15 points) Implementing KNN: my_knn() function implementation:
In this question, you will implement a simple kNN classifier which uses euclidean distance.
Write your code in the function my_knn() with:
i) Input:

1) train: training data frame (containing columns: "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width")
2) test: test data frame (having same columns as train)
3) cl: class labels for training data
4) k: 'k' value in KNN

ii) Returns:

1) A vector of class labels. return variable should be of type factor.

FAQ:

1. Can i use the predefined dist() (or any other predefined function) to calculate distances?
   A: No. Implement a function to calculate euclidean distance
2. Can I use the predefined knn() function?
   A: No. Implement KNN

ii) (5 points) my_knn_accuracy() function implementation:
In this question, you will write code to calculate the following accuracy measures:
 (i) overall accuracy,
(ii) precision for the class 'setosa',
(iii) recall for the class 'setosa',
(iv) F-measure for the class 'setosa'
Please note, you are not allowed to use any pre-defined methods or packages to calculate these values. Please implement the formulae. For generating the confusion matrix, you can use the table() function.
Format for my_knn_accuracy():
i) Input:
1) test_cl: actual class labels for test data
2) knn_result: predicted class labels for test data

ii) Returns:
1) A vector of size 4 in the following order:
 (overall accuracy, precision for the class 'setosa', recall for the class 'setosa', F-measure for the class 'setosa')