CS60050 (Machine Learning) Project 3 (dated 15th April, 2023)

Project Code: PLHC-AS

Premier League Player Rating using Single Linkage Agglomerative (Bottom-Up) Clustering Technique

Roll: 19EC39034
Name: Shaswata Dutta

Few important notes:

- For this assignment, since we have been asked to use Cosine Similarity as the metric for clustering, we are computing the equivalent distance between two entries x and y as:

$$d = 1 - sim(x, y) = 1 - \left(\frac{x^T y}{\|x\|_2 \|y\|_2}\right),$$

  which clearly lies in the range [0, 2]. We could have also used $\sqrt{2d}$ as the equivalent distance, but the former distance measure (i.e., d) provided higher values of Silhouette Coefficients compared to the latter.

- Before performing the clustering, we have imputed the data with Linear Regression to take care of the missing values. After that, we have normalized each row (i.e., sample or record) before performing any sort of clustering.
- Functions used:
  - **lin_reg()**: For performing linear regression on a column to impute missing values
  - **mean_imput()**: For performing mean imputation on a particular column to replace the missing values with the mean of the remaining values in that column.
  - **kmeans_cluster()**: performs clustering on the passed **data_** based on the total number of iterations (**Tmax**), total number of clusters to form (**k**) and set of randomly chosen seeds (**randk_**).
  - **silhouette_score()**: returns the Silhouette Score of each sample (record in the data) as an array. We use *numpy.mean* on the array to get the Silhouette Coefficient for that particular clustering.
  - **calsingledist()**: Computes the minimum possible "distance" (which equals 1 – cosine_similarity() as defined above) between any point in one cluster and any other point in the other cluster. The output is passed to the *bottom_up_single_linkage()* function.
  - **bottom_up_single_linkage()**: Performs Bottom-Up Single Linkage clustering on the given dataset.
  - **jaccard_score():** Computes the Jaccard Score and corresponding mapping between two clusterings (one obtained by k-Means, the other obtained by Hierarchical clustering).

Step-1 and Step-2:

We have implemented Lloyd's algorithm for Step 1.

Upon performing Step 1 and Step 2 for k = 3, considering Regression Imputation and normalization of the data for each row, we have the following output for a randomly selected set of k starting seeds (clearly, the indices of the seeds will range in [0, 1, …, 142]):

```
Starting indices: [ 60 105 118]
Time elapsed for kmeans clustering:  0.2406805999999051

Length of cluster 0 =  61
Length of cluster 1 =  68
Length of cluster 2 =  14


Time elapsed for computing Silhouette Coefficient:  0.4322847389994422
Silhouette Coefficient =  0.7433014220409784
Time elapsed for saving as a text file ('kmeans_k=3.txt'):  0.0014584000000468222
Total time for this cell:  0.6787716509998063
```

We have saved the cluster information in a file *kmeans_k=3.txt* for the first running of k=3 (i.e., only Step 1 and Step 2). The times taken are all in seconds. The above cell takes 0.73 seconds approximately to run. However, depending upon the seeds, it may take upto (approximately) 0.9 seconds to run as well.

Clearly, the k-Means algorithm takes roughly 0.24 seconds, and the Silhouette coefficient calculation takes approximately 0.43 seconds. However, the maximum time both of them can take is always under 0.6 seconds.

Step-3:

For deciding the optimal number of clusters, we have performed **20 experiments for each value of k among {3, 4, 5, 6}**. We have computed an average Silhouette Coefficient (SC) over the 20 experiments for each k, and using this average SC we have determined the optimal value of k for clustering. **Running this code cell takes approximately 85 seconds** (since there are 80 experiments being performed, each having one call for *kmeans_cluster()* function and one call for *silhouette_score()* function).

```
Time elapsed for 25 experiments for k = 3 :  17.92366932799996
Mean Silhouette Coefficient (SC) for k = 3 over 25 runs:  0.6672782988299617
Time elapsed for 25 experiments for k = 4 :  20.10370742499981
Mean Silhouette Coefficient (SC) for k = 4 over 25 runs:  0.6297204526544806
Time elapsed for 25 experiments for k = 5 :  22.169331929000236
Mean Silhouette Coefficient (SC) for k = 5 over 25 runs:  0.6314927049577947
Time elapsed for 25 experiments for k = 6 :  23.741074625999772
Mean Silhouette Coefficient (SC) for k = 6 over 25 runs:  0.6179084279443592
Optimal k = 3
```

Clearly, the optimal value of k thus obtained is k = 3. However, at certain times (although rarely), k = 4 was obtained as optimal k.

Hence, we shall rename the previously saved file *kmeans_k=3.txt* as *kmeans.txt* and go to Step 4.

Step – 4:

We have built a function *bottom_up_single_linkage()* to perform the hierarchical agglomerative clustering. The splitting and time taken (in seconds) is as follows:

```
Sample split for k = 3 : 130 6 7

29.25832234599966
```

The sample split depicts the sizes of the clusters thus formed. The time taken is approx. 29.26 seconds.

The Jaccard Similarity score was also computed. The details are:

```
Cluster Map:  {0: 0, 1: 1, 2: 2}
Jaccard Score recorded:  {0: 0.4580152671755725, 1: 0.0, 2: 0.5}
0.0033222020000163088
```

The time taken for computation is 0.0033 seconds.

Thus, the scores can be represented as:

Mapping (k-means cluster to Hierarchical Cluster)	Jaccard Score
0 -> 0	0.458
1 -> 1	0.0
2 -> 2	0.538

Clearly, the Jaccard Score is not that good as expected. Also, it depends upon the initially chosen seeds for the k-Means clustering algorithm. For some seeds, all the scores are positive (ex: [0.43, 0.12, 0.13] was one of several outputs for the Jaccard score map), however for some others, one of the scores is zero.

The times taken to run the remaining cells in the notebook have been mentioned in the outputs of the corresponding cells. The total time taken to run all the code cells in the notebook is **approximately 116 seconds (in a reasonable PC configuration)**. The major contribution is from the **Hierarchical clustering algorithm in Step 4 (29.26 seconds)** and **obtaining the best value of k in Step-3 (approx. 86 seconds)**.