CMPSC497 Fall 2020 Programming Assignment 3.

Assigned: Friday, November 20, 2020

Due: Friday, December 11, 2020 (by 5pm, submit a package of codes in .ipynb and a report in .pdf via Canvas)

Maximum: 100 point

Note: This assignment is to be done by an individual student, no team work allowed.

In this assignment, you will implement the k-Means algorithm and use your implementation, along with the k-Means available from sklearn to cluster the *Credit Card Clients Dataset [1]* and perform cluster analysis.

The clustering will be based on three sets of attributes/features: i) Pay Amount, ii) Bill Amount, and iii) Remaining Balance, which is a new set of features derived from Pay Amount and Bill Amount. For implementation, a template for k-Means is given for your reference. After running k-Means on the dataset to generate clusters, you need to perform a cluster analysis on the original dataset to discuss the results and report your findings.

Dataset and new features

- *Credit Card Clients Dataset [1]*. We define the Remaining Balance as Bill Amount – Pay Amount. Please calculate this new feature and add them (6 features/columns) to the dataset. Experimentally, you will use Remaining Balance, Bill Amount, and Pay Amount (18 features in total).

Implementation and Experiments

k-Means Algorithm

k-Means is a clustering algorithm widely used for explorative analysis of unknown (unlabeled) data. The goal is to find k clusters in the data, where the number of clusters k is a parameter to be specified. The algorithm works iteratively to assign each data point to one of the k clusters based on the considered features/attributes of the data points. The results of the k-Means clustering algorithm are (1) the *centroids* of the k clusters, which can be used to denote (label) their corresponding clusters; (2) the clustering label (or cluster number) for each data point which is assigned to the corresponding cluster.

Data Preprocessing

Before you run k-Means, please perform normalization on the attributes used for clustering. As 18 attributes are used for clustering (which is quite many), please apply Principle Component Analysis from sklearn to reduce the dimensionality to 3 (i.e., n_components=3). The dataset to run k-Means therefore has 24000 rows and 3 columns.

k-Means Implementation (your own)

In this assignment, please follow the instructions below, step by step, to implement your own k-Means in python. A function template is provided below, with codes for key functions, i.e., *rand_center(data, k), update_centroids(data, centroids, k=5)* and *converged(centroids1, centroids2)*. Your task is to implement these functions in Jupyter Notebook:

```
def rand_center(data,k):
    """
    >>> Function you need to write
    >>> Select "k" random points from "data" as the initial centroids.
    """
    pass

def converged(centroids1, centroids2):
    """
    >>> Function you need to write
    >>> check whether centroids1==centroids2
    >>> add proper code to handle infinite loop if it never converges
    """
    pass

def update_centroids(data, centroids, k):
    """
    >>> Function you need to write
    >>> Assign each data point to its nearest centroid based on the Euclidean dis
tance
    >>> Update the cluster centroid to the mean of all the points assigned to tha
t cluster
    """
    pass
```

```
def kmeans(data,k=5):
    """
    >>> Main function of your k-Means implementation
    """
    # step 1:
    centroids = rand_center(data,k)
    converge = False
    while not converge:
        old_centroids = np.copy(centroids)
        # step 2 & 3; labels can be an array of labels for all the data points
        centroids, label = update_centroids(data, old_centroids)
        # step 4
        converge = converged(old_centroids, centroids)
    print(">>> final centroids")
    print(centroids)
    return centroids, label
```

Step 1. Let C = {c_1, c_2, $c_{3, \ldots}$ c_k} denote the set of centroids. Randomly pick k data points from the 24000 client data points as the initial cluster centroids.

Step 2. Assign each data point *x* to its nearest cluster, based on its Euclidean distances to the centroids. Effectively, this is to assign the cluster number/label *i* to *x* in accordance with the following:

$$\arg\min_{c_i \in C} dist(x, c_i)$$

where dist(·) denotes the Euclidean distance.

Step 3. Update each cluster centroid to the mean of all the points assigned to the corresponding cluster.

$$c_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$$

where S_i is the set of points assigned to *i*-th cluster.

Step 4. Repeat Step 2 and Step 3 until the centroids converges. (Note: add proper code to handle infinite loop if it never converges.)

Evaluation Metrics

As discussed in classes, *SSE* is used to decide the best clustering. As part of the assignment, you need to implement *SSE(centroids, data).* Feel free to change the parameters list of SSE as you see appropriate, e.g., including *label*. Note that K-Means should be run on the dataset multiple times to obtain a set of clusterings, from which you will apply *SSE* to decide the best clustering for cluster analysis.

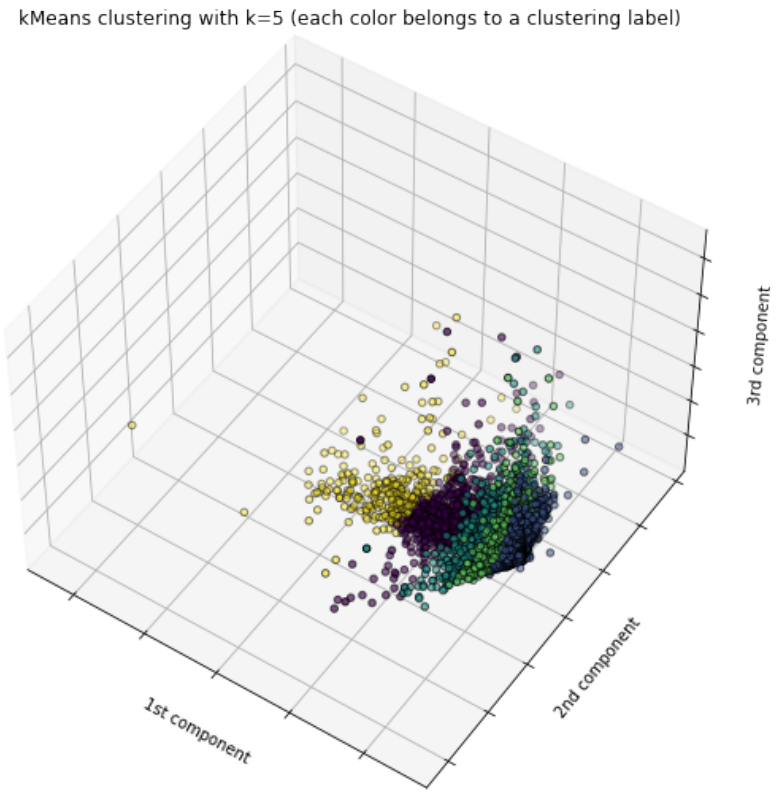
You are required to use SSE as the default metric. On other hand, if you find other useful metrics good for decide the best clustering (in different aspects), please feel free to use them for comparison with the clustering obtained via SSE.

Cluster Analysis

After the clustering is decided, an important task is to perform data analysis on data objects in each cluster. The goal is to characterize and understand better the clients in different clusters. In addition to *default status* (i.e., the class label used in the classification task), the analysis may be carried out on all other attributes/features to characterize the clusters. You may examine the attributes of the medoid (i.e., the representative client) in each cluster or calculate the statistics of user attributes in each cluster for comparison. For example, you may calculate the Impurity (GINI Index) of the customers' default status (and gender, martial status, etc) or calculate the average payment (and age, credit, etc) for each cluster. You may also analyze the distributions of various attribute values to see whether you can capture a pattern. There is various interesting information about customers in different clusters that you can explore (i.e., the more the better) and make comparison. Please show your findings in the follow-up tasks/discussions.

Results and Follow-up Discussion

1. Given k = 5, run your own k-Means multiple times (at least 20 times) to find the best clustering based on SSE (sum of squared errors) and compare it with the result obtained from k-Means in sklearn (with n_clusters=5). Show both clustering results using matplotlib. For the plot, you can refer to the sample code from sklearn here. An example figure with k=5 is shown below.



2. Between the best clusterings you obtained via scikit-learn and your own k-Means implementation, which clustering implementation is better? Use the results obtained in (1) to explain why it is important to choose proper initial centroids. Please note that your implementation uses random initialization while sklearn uses a default initialization method from k-Means++.
3. Please report your findings in cluster analysis. For example, based on the best clustering you obtained, compare clusters in terms of (i) Impurity (Gini Index) of default status and (ii) percentage of defaults. Which measure can reveal more interesting information under the context of this application? Note that simply completing the comparison mentioned in the above example is insufficient. This task will be graded based on both of the effort you made on the analysis and the findings. Please make as many analyses as you can and report your findings
4. You are asked to perform k-Means with k=5 in (1). However, k is a parameter that can be further explored. By trying different k, you may get totally different clustering results. Please run k-Means with k in different ranges (e.g., from 2, 3, … to 10 or from 5, 10, 15, …, to 50). Please keep track of SSE for each k (e.g., a line graph) and decide the best k. You should present the best k and state your reasoning. Finally, please report your findings in cluster analysis for this newly obtained clustering in comparison with the result obtained in (3).
5. In the preprocessing step, the data points are normalized. Is it really better than clustering without preprocessing with normalization? Please use data without

normalization to find the best clustering. Compared with the normalized data, present the plots to make comparison. Report your finding in the comparison. (Note: you are not required to perform clustering analysis in this task but you are welcome to do so.)

6. **Bonus (up to 20%):** you may repeat the above (or part of the above tasks) by using your choice of attributes for clustering. State attributes you propose to use (with justification/explanation) and report your findings/results in comparison with the previously reported result. For example, having credit limit and remaining balance together as a combination of features can result in a different clusterings. Please perform clustering analysis on the new clustering and compare with what you have previously. Please report your findings and explain different clustering results.

Deliverables

- Please submit the report (.pdf) and the complete Jupyter Notebook (.ipynb) on Canvas.
- The report should contain the experimental results for various tasks in the assignment.

Packages

- sklearn (http://scikit-learn.org/). A machine learning framework in Python
- matplotlib (https://matplotlib.org/). Website provides tutorials on how to plot bar chart and histogram in Python.
- NumPy (http://scikit-learn.org/). A fundamental package for scientific computing in Python.
- pandas (https://pandas.pydata.org/). A framework for easy-to-use data analysis and manipulation in Python

Reference

- [1] I-Cheng Yeh, Che-hui Lien, The comparison of data mining techniques for the predictive accuracy of probability of default card clients, Expert Systems with Applications, Volume 36 Issue 2 Part 1, 2009, Pages 2473-2480
- https://www.sciencedirect.com/science/article/pii/S0957417407006719 (accessible on PSU VPN or PSU access)