Aditya Das
1001675762
axd5763

Machine Learning Project 3

Language used: Python (ver:3)
Name of the file: kmeans_axd5763.py
Libraries used: os, copy, pandas, math, matplotlib, numpy
To run file: python3 kmeans_axd5763.py

Requirements:

1. Reading from the iris dataset csv and randomly selecting k clusters

The project has included the csv file required to run the program. Formally, the objective is to find

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

Where X is a set of observations (x1, x2, ..., xn), with each observation being a d-dimensional real vector, S is the number of clusters k (<=n) **S** = $\{S_1, S_2, ..., S_k\}$ so as to minimize the within cluster sum of squares.

Here we select k random centroids, as the initial centroids. The decision to select the value for k depends upon the unique number of nominal classes in the species attribute.

2.Setting the initial closest centroids to each of the datapoints

Next, we assign the closes centroids to each datapoint in the dataset. This is accomplished by executing a Frobenius L2 norm on the dimensional values of the datapoints and figuring out, which centroid each datapoint is closest to. We then set a color for the designated cluster for the same.

3.Setting an iteration limit and norm limit on the loop condition

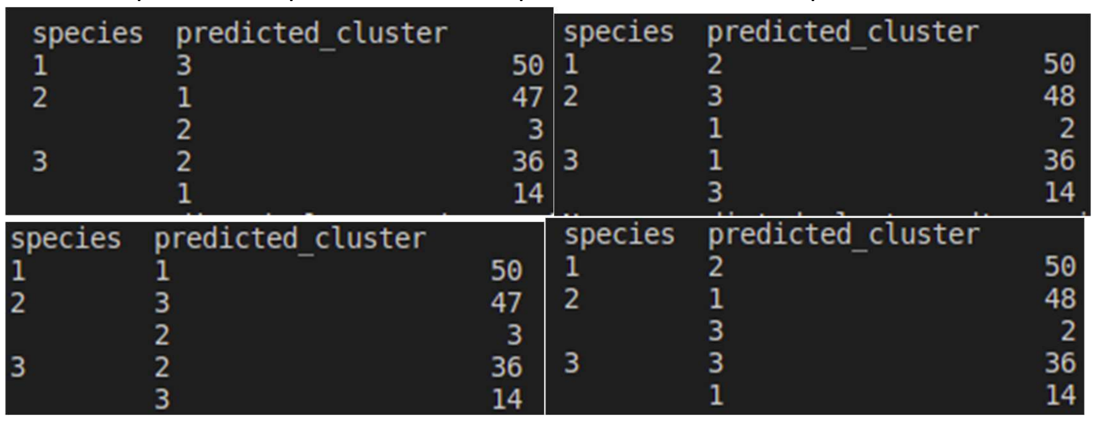
Kmeans by default is an unending loop unless we stop it at an iteration limit or when the L2 distance between the old centroids and the new centroids doesn't change or is less than a particular value.

Here we set the number of iterations at 10000 and have a limit that If the difference between the old centroids and the new centroids is less than 0.001, we exit the loop.

The datapoints are assigned a new centroid every loop if the distance to the created centroids are lesser than the older ones. Even if the centroids are selected at random, they usually converge to the true centroid values.

4.Issues while assigning the clusters to the nominal values.

A common issue faced while assigning the clusters is that the cluster name is not equal to the value of the nominal attribute “species”, thus leading to erroneous values if we simply check if the cluster value is equal to the species nominal replaced value. An example of the above can be seen below



```
species  predicted_cluster
1        3                    50
2        1                    47
         2                     3
3        2                    36
         1                    14
```

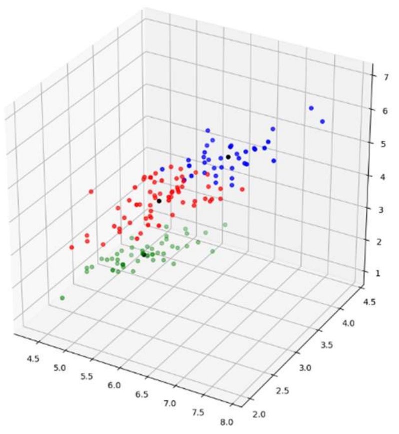
```
species  predicted_cluster
1        2                    50
2        3                    48
         1                     2
3        1                    36
         3                    14
```

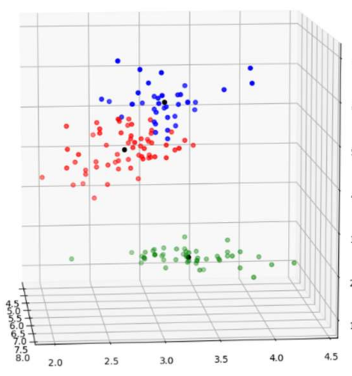
```
species  predicted_cluster
1        1                    50
2        3                    47
         2                     3
3        2                    36
         3                    14
```

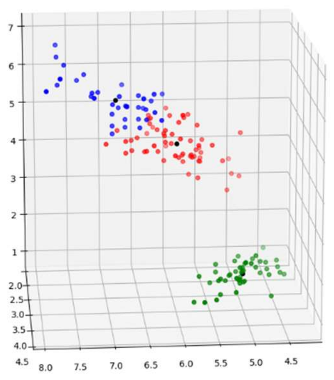
```
species  predicted_cluster
1        2                    50
2        1                    48
         3                     2
3        3                    36
         1                    14
```

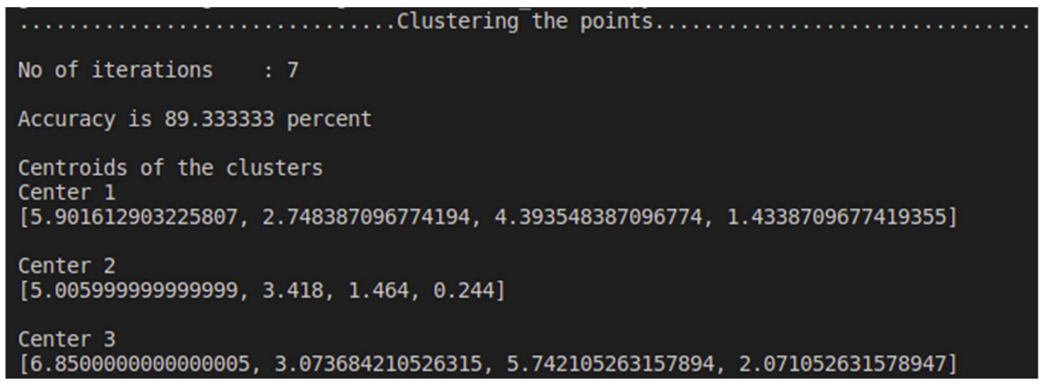
5.Results

The figures below show the results. There is a small variation between results because of the random nature of the selection of the centroids





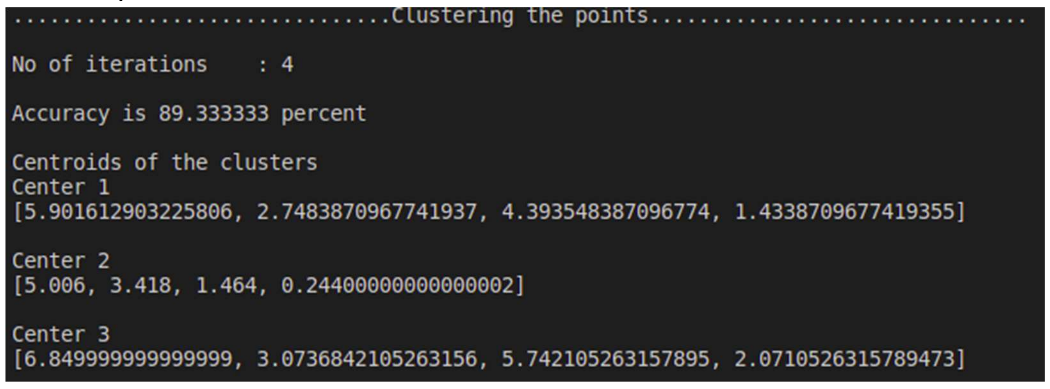




```
..............................Clustering the points..............................

No of iterations    : 7

Accuracy is 89.333333 percent

Centroids of the clusters
Center 1
[5.901612903225807, 2.748387096774194, 4.393548387096774, 1.4338709677419355]

Center 2
[5.005999999999999, 3.418, 1.464, 0.244]

Center 3
[6.8500000000000005, 3.073684210526315, 5.742105263157894, 2.071052631578947]
```

Iterations : 7

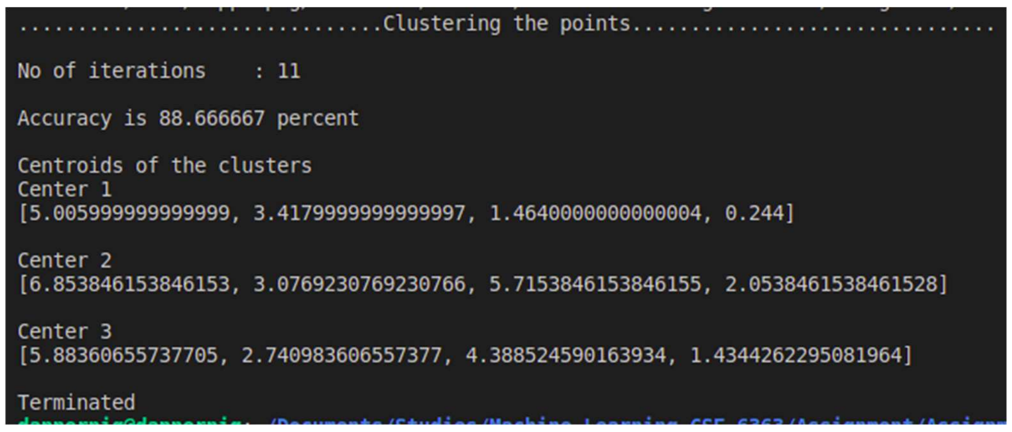
Accuracy 89.33



```
..............................Clustering the points..............................

No of iterations    : 4

Accuracy is 89.333333 percent

Centroids of the clusters
Center 1
[5.901612903225806, 2.7483870967741937, 4.393548387096774, 1.4338709677419355]

Center 2
[5.006, 3.418, 1.464, 0.24400000000000002]

Center 3
[6.849999999999999, 3.0736842105263156, 5.742105263157895, 2.0710526315789473]
```

Iterations : 4

Accuracy 89.33



```
..............................Clustering the points..............................

No of iterations    : 11

Accuracy is 88.666667 percent

Centroids of the clusters
Center 1
[5.005999999999999, 3.4179999999999997, 1.4640000000000004, 0.244]

Center 2
[6.853846153846153, 3.0769230769230766, 5.7153846153846155, 2.0538461538461528]

Center 3
[5.88360655737705, 2.740983606557377, 4.388524590163934, 1.4344262295081964]

Terminated
```

Iterations : 11

Accuracy 88.667

Methods:

There are 7 methods in the python file.

1. **def read_replace()**:

This function is used to read the iris.csv value required. This replaces the nominal values present for the 'species' attribute to ordinal values.

parameters returned: data (a pandas dataframe that has the values of the dataset with the respective attribute names)

2. **def set_random_centers(data)**:

This function selects k random centroids for the initial assignment of the centroid values. The value of k depends on the number of unique nominal values of the attribute 'species'

parameters accepted: data a pandas dataframe that has the values of the dataset with the respective attribute names)
parameters returned: centers(returns a dictionary of the randomly selected centroid values)

3. **def center_assignment (df,centers):**

This function assigns the closest centers to each datapoint. It does a Frobenius L2 norm, selects the closest centroid and assigns the value.

parameters accepted: data(a pandas dataframe that has the values of the dataset with the respective attribute names)
centers(returns a dictionary of the randomly selected centroid values)
parameters returned: df(a pandas dataframe that has the values of the dataset with the respective attribute names)

4. **def kmeans(_data, center):**

This function calls the center_assignment() function to assign the centers initially. Then updates the centers using the update_center() function and then starts the while loop to continuously calculate the new distances and assign them to the datapoints.

parameters accepted: data(a pandas dataframe that has the values of the dataset with the respective attribute names)
centers(returns a dictionary of the randomly selected centroid values)
parameters returned: df(a pandas dataframe that has the values of the dataset with the respective attribute names)
centers(returns a dictionary of the randomly selected centroid values)

5. **def update_center(center,data):**

This function updates the centers.
parameters accepted: data(a pandas dataframe that has the values of the dataset with the respective attribute names)
centers(returns a dictionary of the randomly selected centroid values)
parameters returned: centers(returns a dictionary of the randomly selected centroid values)

6. **def plot_cluster(data,center):**

This function plots the datapoints and the centroid values on a 3d plot.

parameters accepted: data(a pandas dataframe that has the values of the dataset with the respective attribute names)
centers(returns a dictionary of the randomly selected centroid values)

7. **def main():**

Main function. Displays the accuracy and the centroid values.