1001551229

K-means clustering algorithm performed on Iris data.

First data preprocessing, of reading the data from iris.txt and then extracting the class label column from the whole data and X\_train has the all the features excluding class label column.

```
4
       # Fetching Iris Data
5
       data = pand.read_csv('iris.txt')
6
7
       # Extracting Class label from the data
       col = data.columns[data.columns.str.startswith('I')]
8
9
       # Seperating classlabel and data
.0
.1
       classlabel = data[col]
.2
       features = data.drop(data.columns[4], axis=1)
.3
.4
       X train = features.as matrix()
```

I have used k= 3 because we can evaluate, as there are 3 labels (0 Iris-setosa, 1 Iris-versicolor, 2 Iris-virginica) in the dataset. This can be done intuitively. Also, the data is uniformly distributed over all the clusters. A good clustering with smaller k such has 3 can have a lower Sum of Squared Error (SSE) than a poor clustering with higher k.

The below function will calculate the initial centroids to start our algorithm.

```
# Fuction for Calculating Initial Centriods

| def initialCentriod(k=3):
| np.random.seed(0)
| cen = np.random.random((k,4))
| return cen
| initialcen = initialCentriod()*5
```

Function to calculate the Euclidean distance from centroid to each data point so as to form a neighborhood which easily identifies which data point belongs to which cluster.

```
# function to calculate the distance between centriod and data points
from math import sqrt

def pairwisedistance(datapoint,cent):
    return sqrt(np.sum((datapoint-cent)*(datapoint-cent)))
```

KmeansClustering is the function which performs the kmeans clustering algorithm (mentioned below) using 2 parameters the training data and number of iterations.

- 1: Select K points as the initial centroids.
- 2: repeat
- 3: Form K clusters by assigning all points to the closest centroid.
- 4: Recompute the centroid of each cluster.
- 5: **until** The centroids don't change

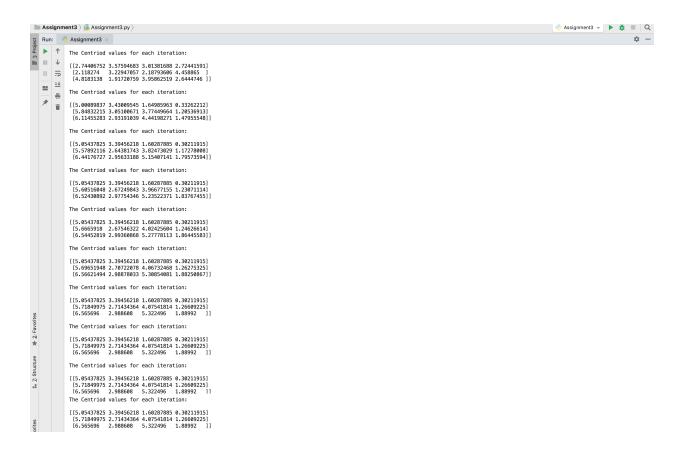
The Complexity of the above algorithm is O(n \* K \* I \* d) where n=number of points(), K= number of clusters, I = number of iterations, d = number of attributes.

```
# Function for calculating Kmeans along with training data and iterations
        def kmeansClustering(X_train,iterations):
            size = X_train.shape[0]
            centriod = initialCentriod(3)*5
            kvalue = centriod.shape[0]
            distance = np.zeros([size, kvalue])
35
            classAssign = np.zeros([size, ])
            cenn = centriod
36
            temp = np.zeros([1, 4])
            for t in range(iterations):
                 # print(centriod)
                 for r in range(0, size):
                     for c in range(0, kvalue):
    distance[r][c] = pairwisedistance(X_train[r], centriod[c])
43
14
                classAssign = (np.argmin(distance, axis=1)).reshape((-1,))
45
                cenn = np.concatenate((cenn, centriod))
print("\nThe Centriod values for each iteration:\n")
46
                print(centriod)
                 for c in range(0, kvalue):
                     temp = np.zeros([1, 4])
                     count = 0
                     for r in range(0, size):
                         temp = temp + (0.98) * (classAssign[r] == c) * X_train[r] + (0.02) * (classAssign[r] != c) * X_train[r]
                         count = count + (0.98) * (classAssign[r] == c) + (0.02) * (classAssign[r] != c)
                     centriod[c] = (temp.reshape((-1,))) / (count)
            return centriod, classAssign, cenn
```

The kmeansClustering results in centroids computed in each iterations along with the assignments of the data points to each cluster.

```
centroid ,classAssign,cenn = kmeansClustering(X_train,iterations=10)
print("\nKmeans Result Assignments:\n")
print(classAssign)
```

10 Iterations centroids are shown below:



## The result of our algorithm is:

## References:

http://madhugnadig.com/articles/machine-learning/2017/03/04/implementing-k-means-clustering-from-scratch-in-python.html

http://benalexkeen.com/k-means-clustering-in-python/

https://www.kaggle.com/andyxie/k-means-clustering-implementation-in-python

GitHub & Wikipedia