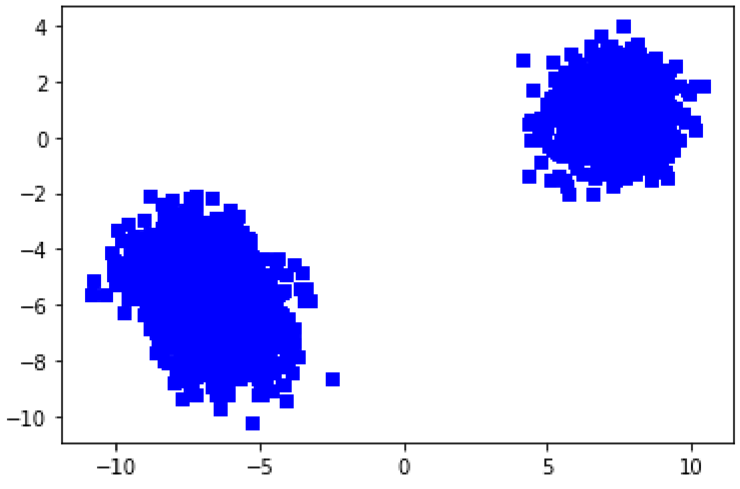
```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt


from google.colab import drive

drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive
```

```
main_directory  = '/content/gdrive/MyDrive/Pattern Lab/Assignment4/data_k_mean.txt'
train = pd.read_csv(main_directory, sep=' ' , header = None)
train1 = train.to_numpy()
print(train1)
```

```
[[-7.87157 -4.86573]
 [-4.76661 -6.87944]
 [-6.67986 -5.8308 ]
 ...
 [ 6.91832 -0.32132]
 [-8.23828 -4.00405]
 [-5.75112 -5.99531]]
```

```
#plotting all data points
plt.scatter(train[0], train[1], c = 'blue', marker = 's')
plt.show()
```



```
#taking k as input
k = int(input("Enter the value of k : "))
```

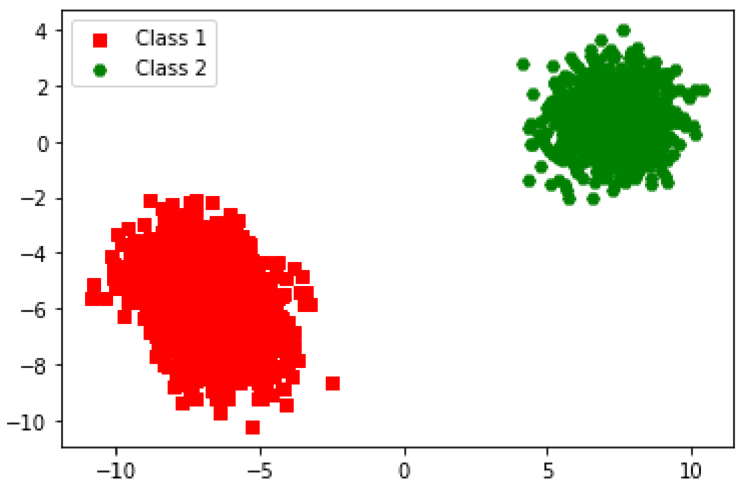
```
Enter the value of k : 2
```

```
#random centroids for 1st iteration
np.random.seed(seed=72)
random_numbers = np.random.randint(low=0, high=len(train1), size=(k))
```

```
centroids = [train1[random_numbers[i]] for i in range(k)]
print(centroids)
```

```
[array([ 6.80375, -0.13017]), array([6.68468, 0.85224])]
```

```
distance = [] #to store the distance from point to classes
index_clusters = [-1 for i in range(len(train1))] #to store class corresponding to index
count = 0 #to count the iteration number
clusters = {} #to store class numbers as keys and data points as values
for x in range(500):
    count = x
    # flag to keep track whether change occurs or not
    flag = 0
    for y in range(k):
        clusters[y] = []
    # iterate through each data points
    for i in range(len(train1)):
        distance = []
        for j in range(k):
            dist = np.sqrt(pow(abs((train1[i][0] - centroids[j][0])), 2) + pow(abs((train1
            distance.append(dist)
        index = distance.index(min(distance))
        # check whether the change occurs or not
        if index_clusters[i] != index:
            flag = 1
            index_clusters[i] = index
        clusters[index].append(train1[i])
    # if change occurs
    if flag == 0:
        break
    # calculating new centroids
    centroids = [np.mean(np.asarray(clusters[z]), axis=0) for z in range(k)]


x1 = np.asarray(clusters[0])[:, 0]
y1 = np.asarray(clusters[0])[:, 1]

x2 = np.asarray(clusters[1])[:, 0]
y2 = np.asarray(clusters[1])[:, 1]


# plotting classified data points of two classes with different colored marker
plt.scatter(x1, y1, c = 'red', marker = 's', label = 'Class 1')
plt.scatter(x2, y2, c = 'green', marker = 'H', label = 'Class 2')
plt.legend(loc = 'best')
plt.show()
```



✓ 0s completed at 7:32 PM