```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


#https://medium.com/@dilekamadushan/introduction-to-k-means-
clustering-7c0ebc997e00
import pandas as pd
xls_file = 'cluster.xlsx'
data1 = pd.read_excel(xls_file)
data_main=data1.drop(['Column1'],axis=1)


# In[2]:


print(data_main)


# data['Column1'].values

# In[3]:


listdata1=[]
listdata1=data_main.values.tolist()
print(len(listdata1))

animal_list=[]
for i in range(50):
    animal_list.append(listdata1[i])

fruit_list=[]
for j in range(269,329):
    fruit_list.append(listdata1[j])

country_list=[]
for k in range(211,269):
    country_list.append(listdata1[k])

vege_list=[]
for l in range(50,211):
    vege_list.append(listdata1[l])


# In[4]:




# In[5]:
```

```
# In[ ]:




#

# In[6]:







# In[7]:




# In[8]:




# In[44]:


from scipy.spatial import distance
import random
# min_list=[]



#initial centers

import numpy as np
def initial_centers(k):
    centers=[]
    for i in range(k):
        r=random.randint(0,329)
        centers.append(listdata1[r])
    return centers




#calculating euclidean distance

def cal_euclidean_distance(list1,list2,k):
    appended_list=[]
```

```
    for i in range(len(list2)):
        l=[]
        appended_list.append(l)
#     print(len(list1))
#     print(len(list2))



    for i in list1:
        min_list=[]
        for j in list2:
#             print(len(j))
            d=distance.euclidean(j,i)
            min_list.append(d)
        min_val=999999
        for x in min_list:
            if(x<min_val):
                min_val=x
        min_index=min_list.index(min_val)
        appended_list[min_index].append(i)

    return appended_list



recall_e=[]
precision_e=[]
cluster_no_e=[]
f_score_e=[]



def next_centroid(list_app):
    new_center=[]
    for i in list_app:
        temp_clus=[]
        length=len(i)
        for k in range(300):
            temp3=[]
            for j in i:
#                 print(type(j))
                temp3.append(j[k])
#             print(len(temp3))
            sum1=sum(temp3)
            if(length!=0):
                sum_final=sum1/length
                temp_clus.append(sum_final)
        new_center.append(temp_clus)
    return new_center


def comb(n):
    return (n*(n-1))/2



def compare_centroids(list1,list2):
```

```
    return (list1)==(list2)


# print(len(listdata1[0]))

def count(clusture_list):
    c_ani=0
    c_fruit=0
    c_veg=0
    c_country=0
    list1=[]
    for g in clusture_list:
        if g in animal_list:
            c_ani+=1
        elif g in fruit_list:
            c_fruit+=1
        elif g in vege_list:
            c_veg+=1
        else:
            c_country+=1
    list1.append(c_ani)
    list1.append(c_country)
    list1.append(c_fruit)
    list1.append(c_veg)
    return list1


#main function
for cluster in range(1,11):
    list_distance_euc=[]
    cfm=[]
#     for i in range(cluster):
#         l=[]
#         cfm.append(l)

    old_clist=initial_centers(cluster)
#     print(len(old_clist))
    list_distance_euc=
cal_euclidean_distance(listdata1,old_clist,cluster)
#     print(len(list_distance_euc[0][6]))
    new_clist=next_centroid(list_distance_euc)
#     print(len(new_clist))
    while(compare_centroids(new_clist,old_clist)!=True):
        old_clist=new_clist
        list_distance_euc=
cal_euclidean_distance(listdata1,old_clist,cluster)
        new_clist=next_centroid(list_distance_euc)
    total_p=0
    for i in list_distance_euc:
        list1=count(i)
        cfm.append(list1)
        t=comb(len(i))
        total_p+=t
    tps=0
    fn=0
#     print(cfm)
    for i in cfm:
```

```
        mul=1
        for k in range(4):
            tps+=comb(i[k])
            mul*=(i[k])
        fn+=mul
    recall=tps/(tps+fn)
    recall_e.append(recall)
    pres=tps/total_p
    precision_e.append(pres)
    cluster_no_e.append(cluster)
    f_score=(2*(recall*pres))/(recall+pres)
    f_score_e.append(f_score)
#     print(tps)
#     print(fn)

print("number of clusters: " ,cluster_no_e)
print("recall for euclidean distance: " ,recall_e)
print("prescision for euclidean distance: ",precision_e)
print("f-score for euclidean distance: ", f_score_e)




#for cluster=4
# cluster=4
# old_clist=initial_centers(cluster)
# list_distance_euc=
cal_euclidean_distance(listdata1,old_clist,cluster)
# new_clist=next_centroid(list_distance_euc)
# while(compare_centroids(new_clist,old_clist)!=True):
#     old_clist=new_clist
#     list_distance_euc=
cal_euclidean_distance(listdata1,old_clist,cluster)
#     new_clist=next_centroid(list_distance_euc)

# print("Number of cluster is 4")
# print("Number points in first cluster is ",
len(list_distance_euc[0]))
# print("Number points in second cluster is ",
len(list_distance_euc[1]))
# print("Number points in third cluster is ",
len(list_distance_euc[2]))
# print("Number points in fourth cluster is ",
len(list_distance_euc[3]))


# In[13]:


#https://www.geeksforgeeks.org/graph-plotting-in-python-set-1/
import matplotlib.pyplot as plt
plt.plot(cluster_no_e, recall_e,color="red", label = "recall")
plt.plot(cluster_no_e, precision_e, color="blue",label = "precision")
plt.plot(cluster_no_e, f_score_e,color="green" label = "f-score")
plt.xlabel('Number of clusters')
plt.ylabel('Recall, prescision and f-score for euclidean')
plt.legend()
```

```
plt.title('Graph for Euclidean Distance')
plt.show()


# In[19]:


recall_m=[]
precision_m=[]
cluster_no_m=[]
f_score_m=[]
def cal_manhattan_distance(list1,list2,k):
    appended_list=[]
    for i in range(len(list2)):
        l=[]
        appended_list.append(l)
#     print(len(list1))
#     print(len(list2))
    for i in list1:
        min_list=[]
        for j in list2:
#             print(len(j))
            d=distance.cityblock(j,i)
            min_list.append(d)
        min_val=999999
        for x in min_list:
            if(x<min_val):
                min_val=x
        min_index=min_list.index(min_val)
        appended_list[min_index].append(i)

    return appended_list




for cluster in range(1,11):
    list_distance_manhattan=[]
    old_clist=initial_centers(cluster)
#     print(len(old_clist))
    list_distance_manhattan=
cal_manhattan_distance(listdata1,old_clist,cluster)
#     print(len(list_distance_euc[0][6]))
    new_clist=next_centroid(list_distance_manhattan)
#     print(len(new_clist))
    while(compare_centroids(new_clist,old_clist)!=True):
        old_clist=new_clist
        list_distance_manhattan=
cal_manhattan_distance(listdata1,old_clist,cluster)
        new_clist=next_centroid(list_distance_manhattan)
    total_p=0
    for i in list_distance_euc:
        list1=count(i)
        cfm.append(list1)
        t=comb(len(i))
        total_p+=t
```

```
    tps=0
    fn=0
#     print(cfm)
    for i in cfm:
        mul=1
        for k in range(4):
            tps+=comb(i[k])
            mul*=(i[k])
        fn+=mul
    recall=tps/(tps+fn)
    recall_m.append(recall)
    pres=tps/total_p
    precision_m.append(pres)
    cluster_no_m.append(cluster)
    f_score=(2*(recall*pres))/(recall+pres)
    f_score_m.append(f_score)
#     print(tps)
#     print(fn)

print("number of clusters: " ,cluster_no_m)
print("recall for manhattan distance: " ,recall_m)
print("prescision for manhattan distance: ",precision_m)
print("f-score for manhattan distance: ", f_score_m)



# In[29]:


#https://www.geeksforgeeks.org/graph-plotting-in-python-set-1/
import matplotlib.pyplot as plt
plt.plot(cluster_no_m, recall_m,color="red", label = "recall")
plt.plot(cluster_no_m, precision_m,color="blue", label = "precision")
plt.plot(cluster_no_m, f_score_m,color="green" label = "f-score")
plt.xlabel('Number of clusters')
plt.ylabel('Recall, prescision and f-score for manhattan')
plt.legend()
plt.title('Graph for Manhattan Distance')
plt.show()


# In[28]:



recall_c=[]
precision_c=[]
cluster_no_c=[]
f_score_c=[]


def cal_cosine_distance(list1,list2,k):
    appended_list=[]
    for i in range(len(list2)):
        l=[]
        appended_list.append(l)
#     print(len(list1))
```

```
#     print(len(list2))
    for i in list1:
        min_list=[]
        for j in list2:
#             print(len(j))
            d=distance.cosine(j,i)
            min_list.append(d)
        min_val=999999
        for x in min_list:
            if(x<min_val):
                min_val=x
        min_index=min_list.index(min_val)
        appended_list[min_index].append(i)

    return appended_list



for cluster in range(1,11):
    list_distance_cosine=[]
    old_clist=initial_centers(cluster)
#     print(len(old_clist))
    list_distance_cosine=
cal_cosine_distance(listdata1,old_clist,cluster)
#     print(len(list_distance_euc[0][6]))
    new_clist=next_centroid(list_distance_cosine)
#     print(len(new_clist))
    while(compare_centroids(new_clist,old_clist)!=True):
        old_clist=new_clist
        list_distance_cosine=
cal_cosine_distance(listdata1,old_clist,cluster)
        new_clist=next_centroid(list_distance_cosine)
    total_p=0
    for i in list_distance_euc:
        list1=count(i)
        cfm.append(list1)
        t=comb(len(i))
        total_p+=t
    tps=0
    fn=0
#     print(cfm)
    for i in cfm:
        mul=1
        for k in range(4):
            tps+=comb(i[k])
            mul*=(i[k])
        fn+=mul
    recall=tps/(tps+fn)
    recall_c.append(recall)
    pres=tps/total_p
    precision_c.append(pres)
    cluster_no_c.append(cluster)
    f_score=(2*(recall*pres))/(recall+pres)
    f_score_c.append(f_score)
#     print(tps)
#     print(fn)
```

```
print("number of clusters: " ,cluster_no_c)
print("recall for cosine similarity: " ,recall_c)
print("prescision for cosine similarity: ",precision_c)
print("f-score for cosine similarity: ", f_score_c)



# In[30]:


#https://www.geeksforgeeks.org/graph-plotting-in-python-set-1/
import matplotlib.pyplot as plt
plt.plot(cluster_no_c, recall_c,color="red", label = "recall")
plt.plot(cluster_no_c, precision_c,color="blue", label = "precision")
plt.plot(cluster_no_c, f_score_c, color="green",label = "f-score")
plt.xlabel('Number of clusters')
plt.ylabel('Recall, prescision and f-score for cosine')
plt.legend()
plt.title('Graph for Cosine Similarity')
plt.show()
```