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

# In[1]:


import pandas as pd
xls_file = 'cluster.xlsx'
data1 = pd.read_excel(xls_file)


# In[5]:


#normalization
import pandas as pd
#https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.normalize.htm
l
#https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.Normalizer.ht
ml
from sklearn import preprocessing
normalized_data=data1.drop(['Column1'],axis=1)
normal=preprocessing.normalize(normalized_data,norm='l2')

normal_df=pd.DataFrame(normal)
normal_df.head()


# In[3]:


norm_list=[]

norm_list=normal_df.values.tolist()
print(len(norm_list))


# In[20]:


a_list=[]
for i in range(50):
    a_list.append(norm_list[i])


c_list=[]
for k in range(211,269):
    c_list.append(norm_list[k])





from scipy.spatial import distance
```

```
import random
# min_list=[]


f_list=[]
for j in range(269,329):
    f_list.append(norm_list[j])
v_list=[]
for l in range(50,211):
    v_list.append(norm_list[l])
#initial centers

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


    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_en=[]
precision_en=[]
cluster_no_en=[]
f_score_en=[]
```

```
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
    tem=[]
    for g in clusture_list:
        if g in a_list:
            c_ani+=1
        elif g in f_list:
            c_fruit+=1
        elif g in v_list:
            c_veg+=1
        else:
            c_country+=1
    tem.append(c_ani)
    tem.append(c_country)
    tem.append(c_fruit)
    tem.append(c_veg)
    return tem


#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)

    list_distance_euc=
cal_euclidean_distance(norm_list,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(norm_list,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])

    for i in range(len(cfm)):

        for j in range(i+1,len(cfm)):
            for k in range(4):
                mul=cfm[i][k]+cfm[j][k]
                fn=fn+mul



    recall=tps/(tps+fn)
    recall_en.append(recall)
    pres=tps/total_p
    precision_en.append(pres)
    cluster_no_en.append(cluster)
    f_score=(2*(recall*pres))/(recall+pres)
    f_score_en.append(f_score)


print("number of clusters: " ,cluster_no_en)
print("recall for  euclidean distance: " ,recall_en)
print("prescision for  euclidean distance: ",precision_en)
print("f-score for  euclidean distance: ", f_score_en)




# In[21]:
```

```
#https://www.geeksforgeeks.org/graph-plotting-in-python-set-1/
import matplotlib.pyplot as plt
plt.plot(cluster_no_en, recall_en,color="red", label = "recall")
plt.plot(cluster_no_en, precision_en, color="blue",label =
"precision")
plt.plot(cluster_no_en, f_score_en,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 normalized Euclidean Distance')
plt.show()


# In[ ]:
```