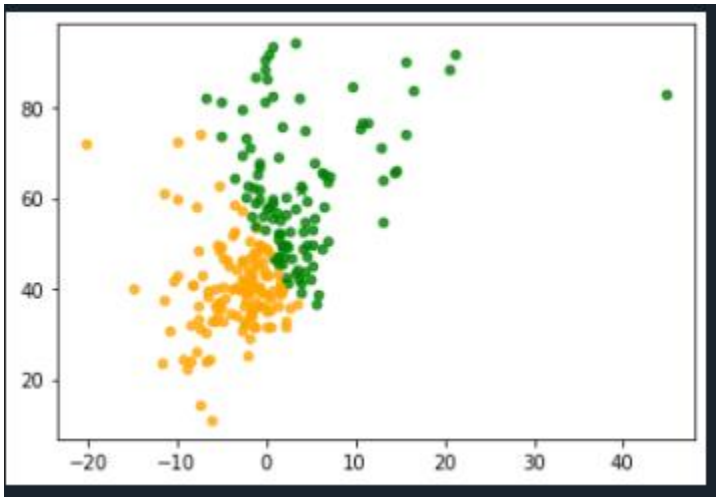
Tarea 4: Implementación del Algoritmo Jerárquico de Aglomeración por centroides

Se utilizo el algoritmo JA middle-L, donde se generar centroides en posiciones random para posteriormente calcular la distancia q estos tienen con los puntos que necesitamos agrupar. El algoritmo esta diseñado para ir ajustando el centroide de acuerdo a la media de las distancias que tienen con los puntos, y así hasta agrupar de forma perfecta los datos.

En la siguiente grafica se muestra los datos agrupados obtenidos de un dataset de league of legends, usando el Score que tiene cada campeón vs. La tendencia a usarlo




```
In [74]: runfile('C:/Users/waldo/Documents/Programas/PROGRA_CD/
untitled0.py', wdir='C:/Users/waldo/Documents/Programas/
PROGRA_CD')
CENTROIDES:

[[ 24.   5.]
 [  3.   4.]
 [ 72. -20.]
 [ 21.   9.]]
C:\Users\waldo\Documents\Programas\PROGRA_CD\untitled0.py:74:
RuntimeWarning: invalid value encountered in double_scalars
  centroides[j,0]/=lenc[j]
C:\Users\waldo\Documents\Programas\PROGRA_CD\untitled0.py:75:
RuntimeWarning: invalid value encountered in double_scalars
  centroides[j,1]/=lenc[j]
```

Código

```
# -*- coding: utf-8 -*-
"""
Created on Wed Dec 21 21:21:28 2022

@author: waldo
"""

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


dataset = pd.read_csv('C:/Users/waldo/Documents/Programas/PROGRA_CD/dataset/stats/League of Legends Champion Stats 12.1.csv', sep=';')

"""
El DATASET ESTA FORMADO CONTIENE
 Name || Class || Role || Score || Trend || Win % || Role % || Pick % || Ban % || KDA
"""

#data = dataset[['Score','Trend']]
score=dataset['Score']
trend = dataset['Trend']
rol = dataset['Role']
KDA = dataset['KDA']

np.random.seed(3)
size = 232
```

```
k = 4
#patrones = np.random.rand(size,2)
patrones = np.zeros((size,2))
centroides = np.zeros((k,2))
distancias = np.zeros((size,k))

pertenencias = np.zeros(size)
bandera =1
colores=['blue','orange','purple','g','r','k','c']

def distancia(x1,x2,y1,y2):
  return pow((x2-x1)**2+(y2-y1)**2,0.5)

for i in range(size):
    patrones[i,0] = trend[i]
    patrones[i,1] = score[i]

for i in range(0,k):
    centroides[i,0] = np.random.randint(0,100)
    centroides[i,1] = np.random.randint(-20,10)



print('CENTROIDES: \n')
print(centroides)

while(bandera == 1):
  bandera = 0
  for i in range(0,k):
    for j in range(0,size):
```

```
    distancias[j,i] = distancia(patrones[j,0],centroides[i,0],patrones[j,1],centroides[i,1])


 centroides = np.zeros((k,2))
 for j in range(0,size):
  for i in range(0,k):
   if(min(distancias[j])==distancias[j,i]):
    if(pertenencias[j]!=i):
     banderas=1
    pertenencias[j]=i
 lenc=np.zeros(k)
 for i in range(0,size):
  for j in range(0,k):
   if(pertenencias[i]==j):
    centroides[j,0]+=patrones[i,0]
    centroides[j,1]+=patrones[i,1]
    lenc[j] += 1
 for j in range(0,k):
  centroides[j,0]/=lenc[j]
  centroides[j,1]/=lenc[j]



for i in range(0,size):
 plt.scatter(patrones[i,0],patrones[i,1],marker='o',color =
colores[int(pertenencias[i])],alpha=0.8,s=22)


#for i in range(0,size):
 #plt.text(patrones[i,0],patrones[i,1],i)


for i in range(0,k):
```

```
  plt.scatter(centroides[i,0],centroides[i,1],marker='*',color=colores[i],alpha=0.8,s=49)


plt.show()
```