Man's Event Segmentation with K-Means

We want to apply event segmentation on this data. Especially, we want to find out charateristics of 2019 event for Men's data. As the dataset is very huge and we have no idea what segmentation we can do on the data. So we tried to find out "is there any actual charcteristics"? This notebook will walk through you in this findings.

In [1]:

```
import random
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
%matplotlib inline
import pandas as pd
```

Load Data From CSV File

Before you can work with the data, you must load the data from csv file.

In [2]:

```
men_df = pd.read_csv("2020-Mens-Data/MEvents2019.csv")
men_df.head()
```

Out[2]:

	EventID	Season	DayNum	WTeamID	LTeamID	WFinalScore	LFinalScore	WCurrentScore	LCurrentScore	ElapsedSeconds	Event
0	10442717	2019	1	1104	1380	82	62	0	0	18	
1	10442718	2019	1	1104	1380	82	62	0	0	18	
2	10442719	2019	1	1104	1380	82	62	0	0	42	
3	10442720	2019	1	1104	1380	82	62	0	0	58	
4	10442721	2019	1	1104	1380	82	62	0	0	63	

Pre-processing

As you can see, 'EventType','EventSubType' in this dataset is a categorical variable. k-means algorithm isn't directly applicable to categorical variables because Euclidean distance function isn't really meaningful for discrete variables. So, lets drop this feature and run clustering. We also drop some other columns which is just provided to link with different tables.

In [3]:

```
df = men_df.drop(['EventID','Season','EventTeamID','EventPlayerID','EventType','EventSubType','X','Y','Area'], axis=1)
df.sample(5)
```

Out[3]:

	DayNum	WTeamID	LTeamID	WFinalScore	LFinalScore	WCurrentScore	LCurrentScore	ElapsedSeconds
2127132	107	1250	1119	91	81	0	0	186
461394	20	1359	1354	77	60	0	0	465
1637248	83	1229	1232	76	62	0	0	749
730586	34	1310	1306	71	67	0	0	1869
2158257	108	1347	1457	87	81	0	0	1884

Normalizing over the standard deviation

Now let's normalize the dataset. But why do we need normalization in the first place? Normalization is a statistical method that helps mathematical-based algorithms to interpret features with different magnitudes and distributions equally. We use standardScaler() to normalize our dataset.

In [4]:

```
from sklearn.preprocessing import StandardScaler
X = df.values[:,1:]
X = np.nan_to_num(X)
Clus_dataSet = StandardScaler().fit_transform(X)
Clus_dataSet
```

Out[4]:

```
array([[-1.7701715 ,  0.91736145,  0.33695799, ..., -0.57617143,
        -0.57001357, -1.77446613],
       [-1.7701715 ,  0.91736145,  0.33695799, ..., -0.57617143,
        -0.57001357, -1.77446613],
       [-1.7701715 ,  0.91736145,  0.33695799, ..., -0.57617143,
        -0.57001357, -1.74035831],
       ...,
       [ 1.4560847 , -1.55653162, -1.42409656, ..., -0.14668164,
        -0.00991078, -1.20173897],
       [ 1.4560847 , -1.55653162, -1.42409656, ..., -0.57617143,
        -0.57001357, -1.20173897],
       [ 1.4560847 , -1.55653162, -1.42409656, ..., -0.14668164,
         0.09192609, -1.17189462]])
```

Modeling¶

In our example (if we didn't have access to the k-means algorithm), it would be the same as guessing that each group would have certain "DayNum WTeamID LTeamID WFinalScore LFinalScore WCurrentScore LCurrentScore ElapsedSeconds" with multiple tests and experiments. However, using the K-means clustering we can do all this process much easier.

Lets apply k-means on our dataset, and take look at cluster labels.

In [5]:

```
clusterNum = 3
k_means = KMeans(init = "k-means++", n_clusters = clusterNum, n_init = 12)
k_means.fit(X)
labels = k_means.labels_
print(labels)
```

```
[0 0 0 ... 0 0 0]
```

In [6]:

```
df["Clus_km"] = labels
df.head(5)
```

Out[6]:

	DayNum	WTeamID	LTeamID	WFinalScore	LFinalScore	WCurrentScore	LCurrentScore	ElapsedSeconds	Clus_km
0	1	1104	1380	82	62	0	0	18	0
1	1	1104	1380	82	62	0	0	18	0
2	1	1104	1380	82	62	0	0	42	0
3	1	1104	1380	82	62	0	0	58	0
4	1	1104	1380	82	62	0	0	63	0

Insights

We assign the labels to each row in dataframe.

We assign the labels to each row in dataframe.

We can easily check the centroid values by averaging the features in each cluster.

In [7]:

```
df.groupby('Clus_km').mean()
```

Out[7]:

Clus_km	DayNum	WTeamID	LTeamID	WFinalScore	LFinalScore	WCurrentScore	LCurrentScore	ElapsedSeconds
0	68.057752	1287.084363	1283.554588	78.290492	66.070635	4.477857	3.681597	459.825306
1	68.707233	1287.477902	1283.470052	78.479740	66.631771	22.219841	18.720124	2070.261017
2	68.218379	1287.207873	1283.745084	78.320774	66.063018	13.469246	11.109970	1262.186387

From our analysis, we can see 'WCurrentScore', 'LCurrentScore' and 'ElapsedSeconds' has significant contribution in the cluster.

In [ ]:

```
area = np.pi * ( X[:, 1])**2
plt.scatter(X[:, 0], X[:, 3], s=area, c=labels.astype(np.float), alpha=0.5)
plt.xlabel('WCurrentScore', fontsize=18)
plt.ylabel('ElapsedSeconds', fontsize=16)

plt.show()
```

In [ ]:

```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(1, figsize=(8, 6))
plt.clf()
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)

plt.cla()
# plt.ylabel('Age', fontsize=18)
# plt.xlabel('Income', fontsize=16)
# plt.zlabel('Education', fontsize=16)
ax.set_xlabel('WCurrentScore')
ax.set_ylabel('LCurrentScore')
ax.set_zlabel('ElapsedSeconds')

ax.scatter(X[:, 1], X[:, 0], X[:, 3], c= labels.astype(np.float))
```

I was trying to visualize the clusters but because of the data size or scaling it didn't run properly.