students_groups_clustering_project

December 15, 2025

[ ]:
```
# Project: Segmenting Students Based on Their Social Network Profiles
# this is a unsupervised learning(clustering) project.
# we are going to tell the model what groups to create.
# buisness question: can we feagure out what student groups we have similar␣
↪online behave
# and interests.
# project objective: the is to group students meaningful cluster based on:
    # profile keywords
    # interest and behaviour patterns
```

[ ]:
```
# steps:
    # import libraries and load data
    # exploration the dataset (EDA)
    # cleaning the data(if applicable)
    # select relevent numeric columns to be used in the model (feature␣
↪engineering)
    # scaling the data (if values are on different ranges)
    # apply k means clustering
    # use elbow method to decide the optimal number of clusters
    # apply clusters with that value of k
    # optionaly use PCA to visualize clusters
    # feature weightage in each PC1, PC2
    # interpret each cluster i buisness decision
```

[34]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# load the data

df = pd.read_csv("Clustering_Marketing.csv")
df.head(10)
```

[34]:
```
   gradyear gender     age  NumberOffriends  basketball  football  soccer  \
0      2007    NaN     NaN                0           0         0       0
1      2007      F   17.41               49           0         0       1
2      2007      F  17.511               41           0         0       0
3      2006      F     NaN               36           0         0       0
4      2008      F  16.657                1           0         0       0
5      2008      M  18.034               32           0         5       0
6      2006      M   18.53               18           0         0       0
7      2006      F     NaN                0           0         0       0
8      2006      F  19.168                0           0         0       0
9      2007    NaN     NaN               21           0         0       0

   softball  volleyball  swimming  …  blonde  mall  shopping  clothes  \
0         0           0         0  …       0     0         0        0
1         0           0         1  …       0     0         0        0
2         0           0         0  …       0     1         0        0
3         0           0         0  …       0     0         0        0
4         0           0         1  …       0     0         0        3
5         0           0         0  …       0     0         0        0
6         0           0         0  …       0     0         0        1
7         0           0         0  …       0     0         0        0
8         0           0         1  …       0     0         0        0
9         0           0         0  …       0     0         1        1

   hollister  abercrombie  die  death  drunk  drugs
0          0            0    0      0      0      0
1          0            0    0      0      1      0
2          0            0    0      0      1      1
3          0            0    0      0      0      0
4          0            0    0      0      0      0
5          0            0    0      0      0      0
6          0            0    0      0      0      0
7          0            0    0      0      0      0
8          0            0    0      0      0      0
9          0            0    0      0      0      0

[10 rows x 40 columns]
```

[35]:
```python
# exploratory data analysis

print("Initial dataset info: ")
df.info()
```

```
Initial dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 40 columns):
 #   Column           Non-Null Count  Dtype
```

```
---  ------          --------------  -----
 0   gradyear        15000 non-null  int64
 1   gender          13663 non-null  object
 2   age             12504 non-null  object
 3   NumberOffriends 15000 non-null  int64
 4   basketball      15000 non-null  int64
 5   football        15000 non-null  int64
 6   soccer          15000 non-null  int64
 7   softball        15000 non-null  int64
 8   volleyball      15000 non-null  int64
 9   swimming        15000 non-null  int64
 10  cheerleading    15000 non-null  int64
 11  baseball        15000 non-null  int64
 12  tennis          15000 non-null  int64
 13  sports          15000 non-null  int64
 14  cute            15000 non-null  int64
 15  sex             15000 non-null  int64
 16  sexy            15000 non-null  int64
 17  hot             15000 non-null  int64
 18  kissed          15000 non-null  int64
 19  dance           15000 non-null  int64
 20  band            15000 non-null  int64
 21  marching        15000 non-null  int64
 22  music           15000 non-null  int64
 23  rock            15000 non-null  int64
 24  god             15000 non-null  int64
 25  church          15000 non-null  int64
 26  jesus           15000 non-null  int64
 27  bible           15000 non-null  int64
 28  hair            15000 non-null  int64
 29  dress           15000 non-null  int64
 30  blonde          15000 non-null  int64
 31  mall            15000 non-null  int64
 32  shopping        15000 non-null  int64
 33  clothes         15000 non-null  int64
 34  hollister       15000 non-null  int64
 35  abercrombie     15000 non-null  int64
 36  die             15000 non-null  int64
 37  death           15000 non-null  int64
 38  drunk           15000 non-null  int64
 39  drugs           15000 non-null  int64
dtypes: int64(38), object(2)
memory usage: 4.6+ MB
```

[36]:

```
df.describe()
```

[36]:

	gradyear	NumberOffriends	basketball	football	soccer	\
count	15000.000000	15000.000000	15000.000000	15000.000000	15000.00000	
mean	2007.496933	29.834533	0.267000	0.255467	0.22200	
std	1.116516	35.386649	0.788851	0.702260	0.92042	
min	2006.000000	0.000000	0.000000	0.000000	0.00000	
25%	2006.000000	3.000000	0.000000	0.000000	0.00000	
50%	2008.000000	20.000000	0.000000	0.000000	0.00000	
75%	2008.000000	44.000000	0.000000	0.000000	0.00000	
max	2009.000000	605.000000	22.000000	9.000000	22.00000	

	softball	volleyball	swimming	cheerleading	baseball	\
count	15000.000000	15000.000000	15000.000000	15000.000000	15000.000000	
mean	0.159667	0.142933	0.135000	0.105133	0.104133	
std	0.737344	0.638747	0.548691	0.502491	0.519205	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	0.000000	0.000000	0.000000	0.000000	
max	17.000000	14.000000	31.000000	8.000000	14.000000	

	…	blonde	mall	shopping	clothes	\
count	…	15000.000000	15000.000000	15000.000000	15000.000000	
mean	…	0.112867	0.259467	0.357267	0.150267	
std	…	2.708619	0.704398	0.728512	0.478716	
min	…	0.000000	0.000000	0.000000	0.000000	
25%	…	0.000000	0.000000	0.000000	0.000000	
50%	…	0.000000	0.000000	0.000000	0.000000	
75%	…	0.000000	0.000000	1.000000	0.000000	
max	…	327.000000	12.000000	11.000000	8.000000	

	hollister	abercrombie	die	death	drunk	\
count	15000.000000	15000.000000	15000.000000	15000.000000	15000.000000	
mean	0.071800	0.051467	0.185867	0.118067	0.091733	
std	0.356258	0.280755	0.609928	0.455200	0.420631	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	0.000000	0.000000	0.000000	0.000000	
max	8.000000	8.000000	16.000000	14.000000	8.000000	

	drugs
count	15000.000000
mean	0.061067
std	0.349112
min	0.000000
25%	0.000000
50%	0.000000

```
75%          0.000000
max         16.000000

[8 rows x 38 columns]
```

[37]:
```
# fix each column object to numeric

df["age"] = pd.to_numeric(df["age"], errors="coerce")

# fill missing age values with mean age

df["age"] = df["age"].fillna(df["age"].mean())
```

[38]:
```
# convertage to integer

df["age"] = df["age"].astype(int)
```

[39]:
```
# fix gender column ( categorical plus missing values )

# check for missing values
df["gender"].value_counts(dropna=False)
```

[39]:
```
gender
F      11057
M       2606
NaN     1337
Name: count, dtype: int64
```

[40]:
```
# fill missing values with the mode

df["gender"] = df["gender"].fillna(df["gender"].mode()[0])
```

[41]:
```
df.isnull().sum()
```

[41]:
```
gradyear           0
gender             0
age                0
NumberOffriends    0
basketball         0
football           0
soccer             0
softball           0
volleyball         0
swimming           0
cheerleading       0
baseball           0
tennis             0
```

```
sports           0
cute             0
sex              0
sexy             0
hot              0
kissed           0
dance            0
band             0
marching         0
music            0
rock             0
god              0
church           0
jesus            0
bible            0
hair             0
dress            0
blonde           0
mall             0
shopping         0
clothes          0
hollister        0
abercrombie      0
die              0
death            0
drunk            0
drugs            0
dtype: int64
```

[42]:
```python
# encode gender : male = 0, female:1

df["gender"] = df["gender"].map({"M":0, "F":1})

df["gender"].value_counts()
```

[42]:
```
gender
1    12394
0     2606
Name: count, dtype: int64
```

[43]:
```python
# select feature for scaling

features = df.columns      #all columns are goint to be used as features in the
 ↪clustering

x = df[features]
```

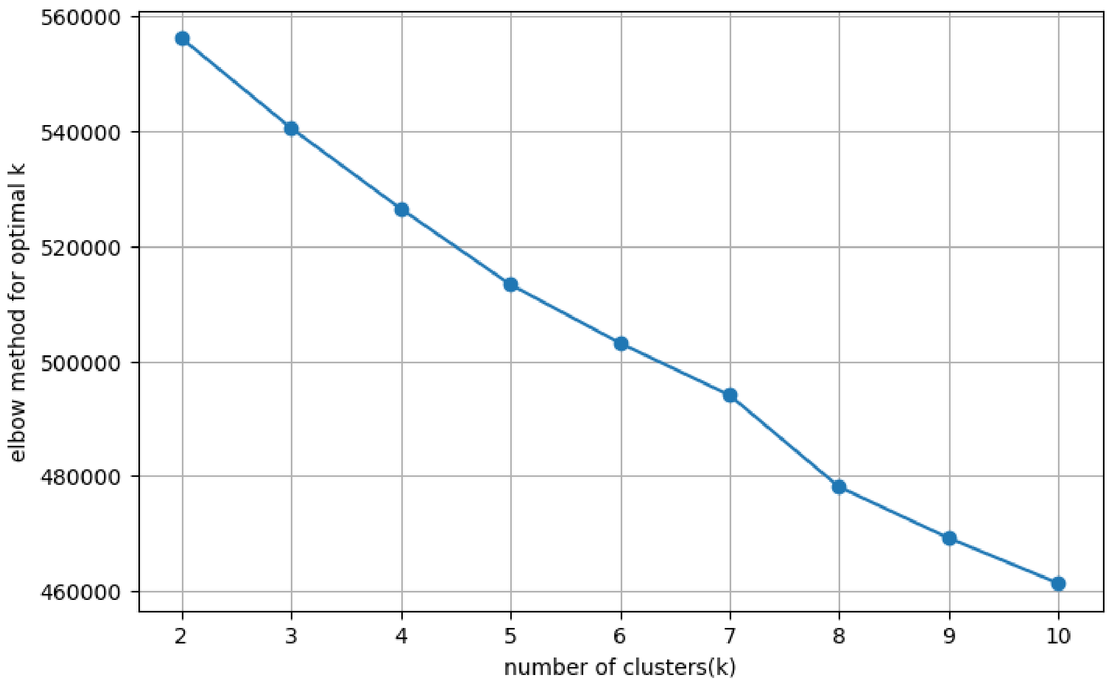
[44]:

```python
# features scaling (to make sure all the features are in the same range)

scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)

print("\n Data scaling done")
```

```
 Data scaling done
```

[30]:

```python
# elbow methos to find optimal k

wcss = []

for k in range(2, 11):
    clusters = KMeans(n_clusters=k, random_state = 42)
    clusters.fit(x_scaled)
    wcss.append(clusters.inertia_)

plt.figure(figsize =(8,5))
plt.plot(range(2,11), wcss, marker = "o")
plt.xlabel("number of clusters(k)")
plt.ylabel("elbow method for optimal k")
plt.grid(True)
plt.show()
```



[45]:
```
# train the final k mean (k = 3)

clusters = KMeans(n_clusters = 3, random_state = 42)

df["Cluster"] = clusters.fit_predict(x_scaled)

df.head(10)
```

[45]:
```
   gradyear  gender  age  NumberOffriends  basketball  football  soccer  \
0      2007       1   17                0           0         0       0
1      2007       1   17               49           0         0       1
2      2007       1   17               41           0         0       0
3      2006       1   17               36           0         0       0
4      2008       1   16                1           0         0       0
5      2008       0   18               32           0         5       0
6      2006       0   18               18           0         0       0
7      2006       1   17                0           0         0       0
8      2006       1   19                0           0         0       0
9      2007       1   17               21           0         0       0

   softball  volleyball  swimming  …  mall  shopping  clothes  hollister  \
0         0           0         0  …     0         0        0          0
1         0           0         1  …     0         0        0          0
2         0           0         0  …     1         0        0          0
3         0           0         0  …     0         0        0          0
4         0           0         1  …     0         0        3          0
5         0           0         0  …     0         0        0          0
6         0           0         0  …     0         0        1          0
7         0           0         0  …     0         0        0          0
8         0           0         1  …     0         0        0          0
9         0           0         0  …     0         1        1          0

   abercrombie  die  death  drunk  drugs  Cluster
0            0    0      0      0      0        2
1            0    0      0      1      0        2
2            0    0      0      1      1        0
3            0    0      0      0      0        2
4            0    0      0      0      0        1
5            0    0      0      0      0        2
6            0    0      0      0      0        2
7            0    0      0      0      0        2
8            0    0      0      0      0        2
9            0    0      0      0      0        2

[10 rows x 41 columns]
```

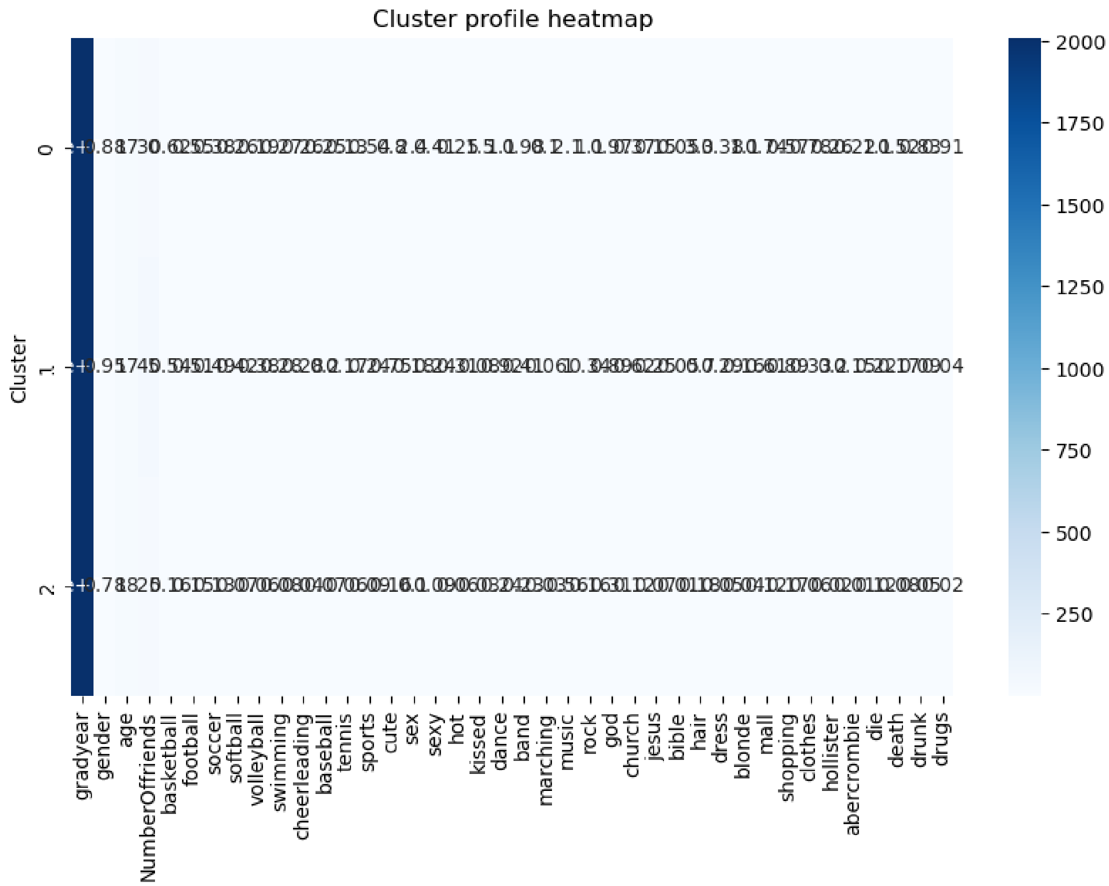
[47]:
```
df["Cluster"].value_counts()
```

[47]:
```
Cluster
2    10863
1     3533
0      604
Name: count, dtype: int64
```

[49]:
```python
# cluster profiling

cluster_profile = df.groupby("Cluster").mean().round(2)

print("\n cluster profile: ")
print(cluster_profile)
```

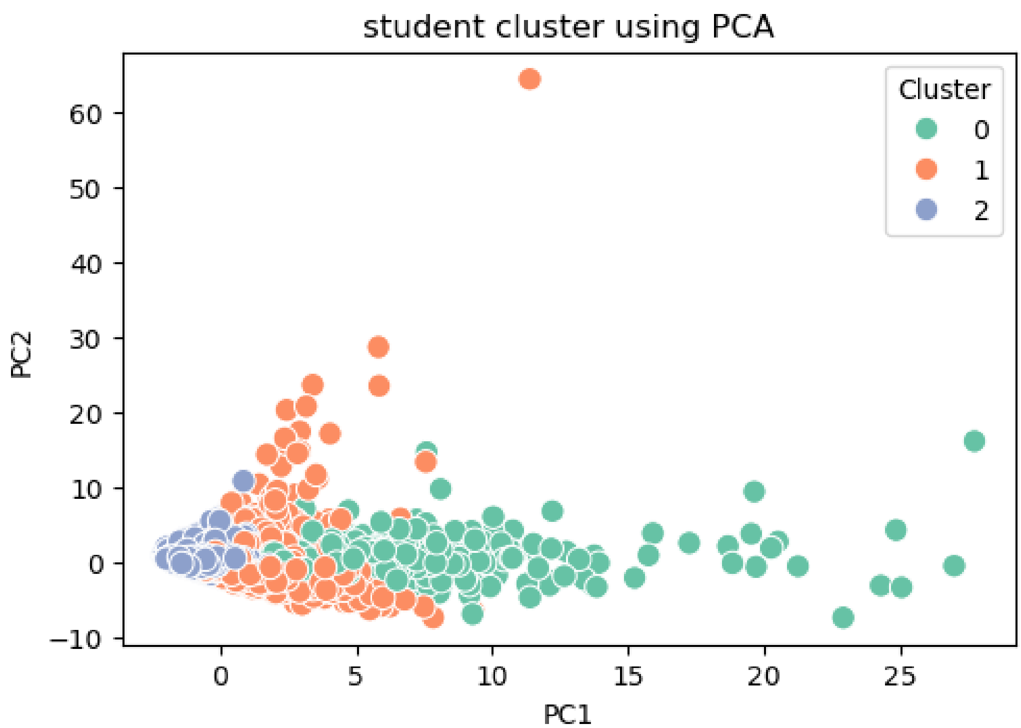
```
 cluster profile:
         gradyear  gender    age  NumberOffriends  basketball  football  \
Cluster
0         2007.64    0.88  17.17            30.47        0.62      0.55
1         2007.80    0.95  16.83            45.40        0.54      0.51
2         2007.39    0.78  17.59            24.74        0.16      0.15

         soccer  softball  volleyball  swimming  …  blonde  mall  shopping  \
Cluster                                          …
0          0.38      0.26        0.19      0.27  …    1.10  0.74      0.57
1          0.49      0.42        0.38      0.28  …    0.16  0.61      0.89
2          0.13      0.07        0.06      0.08  …    0.04  0.12      0.17

         clothes  hollister  abercrombie   die  death  drunk  drugs
Cluster
0           0.78       0.26         0.22  1.15   0.52   0.83   0.91
1           0.33       0.20         0.15  0.22   0.17   0.09   0.04
2           0.06       0.02         0.01  0.12   0.08   0.05   0.02

[3 rows x 40 columns]
```

[50]:
```python
# heatmap for cluster interpretation

plt.figure(figsize=(10,6))
sns.heatmap(cluster_profile, annot=True, cmap="Blues")
plt.title("Cluster profile heatmap")
plt.show()
```




```
[56]: # PCA

from sklearn.decomposition import PCA

pca = PCA(n_components = 2, random_state=42)
pca_components = pca.fit_transform(x_scaled)

df["PC1"] = pca_components[:,0]
df["PC2"] = pca_components[:,1]
```

```
[58]: # cluster scatter plot

plt.figure(figsize=(6,4))
sns.scatterplot(x="PC1", y="PC2", data=df, hue="Cluster", palette = "Set2",␣
 ↪s=80)
plt.title("student cluster using PCA")
plt.show()
```



[60]:
```
# PCA loading
loadings = pd.DataFrame(pca.components_.T, columns=["PC1_Loading",␣
 ↪"PC2_Loading"],index=x.columns)
print("\n PCA Loadings:")
print(loadings.sort_values("PC1_Loading", ascending=False))
```

```
 PCA Loadings:
                 PC1_Loading  PC2_Loading
hair                0.354240    -0.031398
kissed              0.271048     0.017065
clothes             0.243650    -0.069296
music               0.243065     0.056458
cute                0.212697    -0.113617
sex                 0.211653     0.077365
drugs               0.210783     0.060319
rock                0.209763     0.056976
die                 0.205658     0.116437
shopping            0.189704    -0.202601
mall                0.186774    -0.145188
drunk               0.179327     0.053652
abercrombie         0.177930    -0.201746
sports              0.166879    -0.011217
hollister           0.162856    -0.213837
```

```
dress               0.160581    -0.051493
death               0.157719     0.129744
dance               0.156162    -0.057945
basketball          0.142203    -0.041466
god                 0.138168     0.452918
football            0.134018    -0.007072
sexy                0.132376    -0.025298
church              0.119531     0.175640
band                0.117378     0.153170
hot                 0.111366    -0.102637
swimming            0.094189    -0.051586
cheerleading        0.092703    -0.103323
blonde              0.091906     0.066744
gender              0.091287    -0.187506
NumberOffriends     0.084543    -0.051955
baseball            0.076688     0.027079
volleyball          0.075063    -0.093942
soccer              0.074119    -0.030439
softball            0.073512    -0.064422
gradyear            0.070842    -0.112303
jesus               0.069986     0.461508
bible               0.060816     0.435937
tennis              0.049839     0.013654
marching            0.045574     0.125134
age                -0.022015     0.053242
```

[ ]:
```
# Cluster 0 : Appearence & Social Media Focused Students

# Behaviour Pattern

# High mentions of : cute, hot, clothes, mall, shopping
# Moderate sports
# High online presence or footprints
# These students are :
# 1) Image Concious
# 2) Fashion & Lifestyle driven
# 3) Social Media expressive

# Business Insights :
#1) Target for fashion brands
#2) Influencer campaigns
#3) Lifestyle Marketing
#4) Youth- Oriented advertising

# clsuter 1 (Orange):
# Risk prone / Emotionally Expressive Students
# Higher mention of : drugs, drinks, die, death
```

```
# lower sports & social activity
# Emotional expression visible

# These students may:
#1) Be under stress
#2) Show risk-taking behaviour
#3) Need attention or guidance

# Business/ Social Insights :
#1) Important for counselling programs
# 2) Awarness campaigns
#3) Mental Health Initiatives

# Cluster 2: Sports & Socially Active Students
#1) High Sports mentions (basketball, football, soccer)
#2) Active social life ( dance band, music)
#3) High number of friends

# Interpretation :

# These students are :
#1. Energetic
#2. Socially Active
#3. Team- Oriented
#4. Participate in outdoor & group activities

# Business Insights
#1. Ideal for sports events, college tournaments
#2. Good target for group activities, clubs/communities, college fests.
```