day-6-630

February 19, 2024

[1]:
```
# unsupervised learning algorithms
# k means clustering
# k means clustering is an unsupervised learning algorithm that will attempt to␣
↪group similar clusters together from your data
# It is mainly used in clustering similar documents
# clustering customers based on similar features
```

[2]:
```
from sklearn.datasets import make_blobs
# used to generate synthetic dataset
# for clustering and classification tasks
# this function will create clusters of data points with
# gaussiam distribution
```

[3]:
```
# creating random dataset
data = make_blobs(n_samples=200,n_features=2,centers=3,cluster_std=5.
↪6,random_state=101)
# n_samples = total no of points equally divided among clusters.
# n_features = It indicates the no of features(columns)
# centers = It determines no of clusters to be generated
# cluster_std = It sets the standards deviation of the clusters , high value␣
↪makes the clusters to spread out
data
```

[3]:
```
(array([[ -5.24981605,  -4.90754635],
        [-11.62061369,   3.19390403],
        [ -1.08265038,  -1.82412171],
        [ -7.41638   ,   9.71532652],
        [-16.74681638, -15.71394126],
        [ -3.99146763,   2.03202524],
        [  9.34655477,   8.52472571],
        [ -5.23430305,   2.51542897],
        [-15.19614755, -15.10875788],
        [  1.38466629,  -2.83633178],
        [ -2.98477661,   5.864642  ],
        [  3.97423306,  -0.37482931],
        [  3.92208342,  -3.65520771],
        [  8.97021315,  10.85677608],
```

```
[  6.75283137,  10.58839336],
[  3.90517983,   3.25937101],
[ -9.93058674,  -2.65759501],
[  5.71213823, -10.57743709],
[-24.75768521,   1.95678094],
[ -7.50669603,  -5.47042511],
[  6.80118781,   5.61107482],
[ -4.938591  , -15.90777285],
[ -5.10748325,  -4.92195377],
[  2.83599543,   1.00386038],
[  0.3807653 ,  -2.67450522],
[  3.96319993,   3.42341868],
[-10.97376961,   5.55803526],
[-14.21773092,   2.40620013],
[-11.02795336,   5.13745236],
[  3.61608096,  -1.34759405],
[ -5.53399286,   8.65283337],
[ -5.30107418,  -2.74067061],
[ -4.8569233 ,  11.84539091],
[-10.99577267,  -0.61838855],
[-10.58295234,   7.39677816],
[ 13.77988294,  -0.67404326],
[ -0.69598527,  12.17248613],
[ -3.52719179,  12.28354958],
[ -7.22713521, -10.10040402],
[  7.40730314,   4.81358077],
[-14.87031384, -15.09463097],
[-18.10867365,  -4.13215051],
[  6.07348381,   0.53640347],
[-14.89247183, -12.47673022],
[  1.97809804,   1.96236487],
[ -1.78053203,   2.70323526],
[  1.23394721,   6.07831595],
[ 10.93082725,  -1.60065119],
[  4.36438311,  -5.43590767],
[  2.60960459,   3.02606826],
[ -0.54996606,  -1.36348297],
[ -7.43014736,  -6.03274459],
[-14.93406105,   5.92714249],
[ 12.1601229 ,   7.65616321],
[ 14.92138932,   5.2410015 ],
[ -0.58222196,   4.13388067],
[  2.94498958,   2.69820205],
[  4.41161975,   9.28778447],
[ -2.38410135,  -7.70155751],
[-12.68066018,   0.20098592],
[ -8.43366456, -10.07967714],
```

```
[-14.62846253,  -1.82690505],
[-10.16918984, -14.04608059],
[ -4.45194252, -16.86942771],
[  9.32593978,  -4.83462408],
[-16.34995533, -11.04076346],
[  2.68687897,   3.29981864],
[ -4.37469066,  -9.58185815],
[-17.22302221,  -5.74364264],
[-12.49517084,  -3.78789593],
[ -0.32595802,  12.06317859],
[ -8.10333994,  -5.44519625],
[-10.58935781,   6.4991462 ],
[  4.41050545,  -3.41995937],
[  3.89364577,   5.44477282],
[ -4.98339583,   1.0448202 ],
[ 10.37684638,   8.5854883 ],
[ -6.8437892 ,  -0.68125892],
[ -4.42125855,   4.80675769],
[  4.05936075,  -1.60076217],
[  1.93493197, -12.42917384],
[ -3.89338754,  -3.69210359],
[ -2.70074891, -11.16416059],
[  3.28932812,  -1.88505125],
[ 12.39511264,  -2.00409786],
[  0.15347386,  12.27697268],
[-13.51552696, -10.60218146],
[ -9.07041593,   3.2836011 ],
[  4.71454244,  -2.55678919],
[  3.61431741,  -7.66619609],
[ -6.02091464,  -4.96531814],
[  5.42499657,   9.64552807],
[  4.14274459,   7.21005216],
[ -3.08177872,  -3.61172116],
[ -5.89825096,  14.23819369],
[  8.8604806 ,  14.23310739],
[ 15.38218867,   5.04373595],
[  5.00255096,   7.54824385],
[  5.88638389,   8.08989931],
[  0.56014823,  -0.88855852],
[  1.15132247,  -1.27155728],
[ -1.86250456,   7.17179344],
[ -1.4946331 ,  16.28725818],
[  2.5259099 ,   2.65369957],
[ -5.20743544, -10.89299448],
[  4.99402708,  10.14242201],
[ -9.62333264,  -2.7144797 ],
[-17.26093767,  -7.79770689],
```

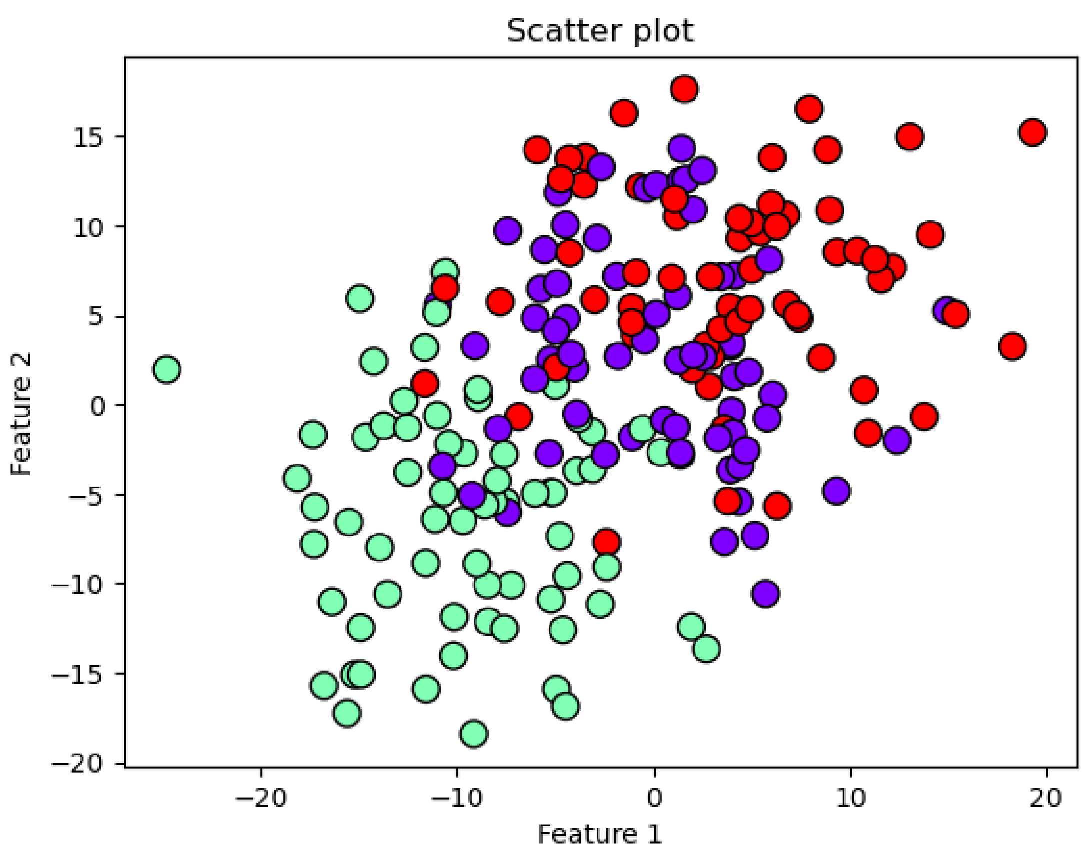
```
[  1.39642076,  12.49439275],
[  5.98802375,  11.20185662],
[ -0.9489662 ,   3.91186589],
[ -8.5700148 ,  -5.63079271],
[ -8.97398467,  -8.89463713],
[  1.20865205,  10.51883034],
[ -3.46985401,  13.82763337],
[  1.58951527,  17.62828863],
[ 11.58579716,   7.01937378],
[ -6.02140355,   4.83202802],
[ 19.30159377,  15.21093982],
[  1.66148352,  12.59460454],
[-10.66185214,  -4.92991414],
[  2.4897423 ,  13.08385551],
[-13.71972389,  -1.18069152],
[  6.30073365,  -5.67813819],
[  2.02289907,  10.89820012],
[  5.16729587,  -7.32786437],
[ -7.78881273,   5.744719  ],
[ 11.27565909,   8.11816138],
[  8.54195454,   2.60818957],
[ -0.86913535,   7.36274223],
[ -0.42713548,   3.60030766],
[ -3.10241438,  -1.54190753],
[-11.55909508, -15.89949828],
[ -4.28430809,  13.73659972],
[ -8.39183314, -12.13606931],
[  3.4188036 ,   4.21882032],
[-13.92643723,  -7.98905142],
[ -4.46112527,  10.04849802],
[-10.4052321 ,  -2.20078447],
[-11.10175564,  -6.40637461],
[ -7.59458492,  -2.79244983],
[ -1.10096395,   5.43355988],
[  6.27786082,   9.95680828],
[  2.69584278, -13.66611186],
[  3.45708123,   7.1522893 ],
[  1.43410687,  14.31008042],
[ 18.27248416,   3.24764627],
[ -1.09755957,   4.58636425],
[ -4.70029128,  12.60473604],
[  7.31828097,   5.01116956],
[ -9.22610866,  -5.07771699],
[-11.61866126,   1.16108127],
[-10.71861603,  -3.44120446],
[ 10.72202398,   0.80719218],
[-12.51542109,  -1.294115  ],
```

```
       [ -4.5943801 , -12.5984737 ],
       [-15.48477782,  -6.57155466],
       [ -2.85608307,   9.29444593],
       [ -2.37505481,  -9.07779765],
       [  1.26263852,   2.44656215],
       [  1.37586988,  -2.69058109],
       [ -4.93185963,   2.07972363],
       [  4.12216919,   1.55954243],
       [ 13.05200063,  14.9667644 ],
       [ -2.45580997,  -2.80943859],
       [ -4.25177743,   8.47923093],
       [  4.34719254,   4.61440727],
       [  0.15112935,   5.0523764 ],
       [  1.08050443,  11.49448065],
       [  4.85112567,   1.82192592],
       [  6.04304587,  13.80824563],
       [  0.9340018 ,   7.07984909],
       [ -4.9551033 ,   4.12796096],
       [  2.91444877,   7.15689858],
       [ -4.75813015,  -7.35598513],
       [  4.90525112,   5.33108154],
       [-10.13264796, -11.86486037],
       [ -3.80780104,  -0.79083806],
       [  7.93769593,  16.52341902],
       [  2.03950902,   2.77514637],
       [  5.79200276,  -0.76078498],
       [  3.78124833,  -5.3983936 ],
       [ -5.72266764,   6.46677918],
       [-15.5775964 , -17.25441627],
       [ -2.65386239,  13.27262508],
       [ -7.577668  , -12.52264068],
       [ -8.90619876,   0.37898376],
       [ -3.90184935,  -0.52659188],
       [ -9.13001131, -18.41420022],
       [ -7.93975751,  -4.25911797],
       [  4.37275142,  10.40593894],
       [ 14.09335575,   9.49053914],
       [-11.57676101,  -8.86175039],
       [ -4.89815586,   6.76166819],
       [-17.32161928,  -1.69248758],
       [ -8.9226025 ,   0.83203907],
       [ -9.68630376,  -6.50000933],
       [ -6.03723717,   1.41540402],
       [ -4.17807591,   2.83322077],
       [ -7.89010596,  -1.35358227]]),
array([2, 1, 0, 0, 1, 0, 2, 0, 1, 0, 2, 0, 0, 2, 2, 0, 1, 0, 1, 1, 2, 1,
       1, 2, 1, 0, 0, 1, 1, 2, 0, 0, 0, 1, 1, 2, 2, 2, 1, 2, 1, 1, 0, 1,
```

```
2, 0, 0, 2, 0, 2, 1, 0, 1, 2, 0, 0, 2, 2, 2, 1, 1, 1, 1, 1, 0, 1,
2, 1, 1, 1, 0, 1, 2, 0, 2, 1, 2, 2, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0,
0, 0, 1, 2, 0, 1, 2, 2, 2, 2, 0, 0, 0, 0, 2, 0, 1, 2, 1, 1, 0, 2,
2, 1, 1, 2, 2, 2, 2, 0, 2, 0, 1, 0, 1, 2, 0, 0, 2, 2, 2, 2, 0, 1,
1, 2, 1, 2, 1, 0, 1, 1, 1, 2, 2, 1, 0, 0, 2, 2, 2, 2, 0, 2, 0, 2,
1, 1, 1, 0, 1, 0, 0, 2, 0, 2, 0, 2, 2, 0, 2, 0, 2, 2, 0, 2, 1, 2,
1, 1, 2, 0, 0, 2, 0, 1, 0, 1, 1, 0, 1, 1, 2, 2, 1, 0, 1, 1, 1, 0,
0, 0]))
```

[4]:
```python
import matplotlib.pyplot as plt
x , y = data
plt.scatter(x[:, 0],x[:,1],c=y, cmap = 'rainbow',edgecolor='black',s=100)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Scatter plot')
```

[4]:
```
Text(0.5, 1.0, 'Scatter plot')
```



[5]:
```python
data[0].shape
```

[5]: (200, 2)

[6]:
```
from sklearn.cluster import KMeans
```

[7]:
```
kmeans = KMeans(n_clusters=4)
kmeans.fit(data[0])
```

```
D:\anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning:
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the
value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
D:\anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:
KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.
  warnings.warn(
```

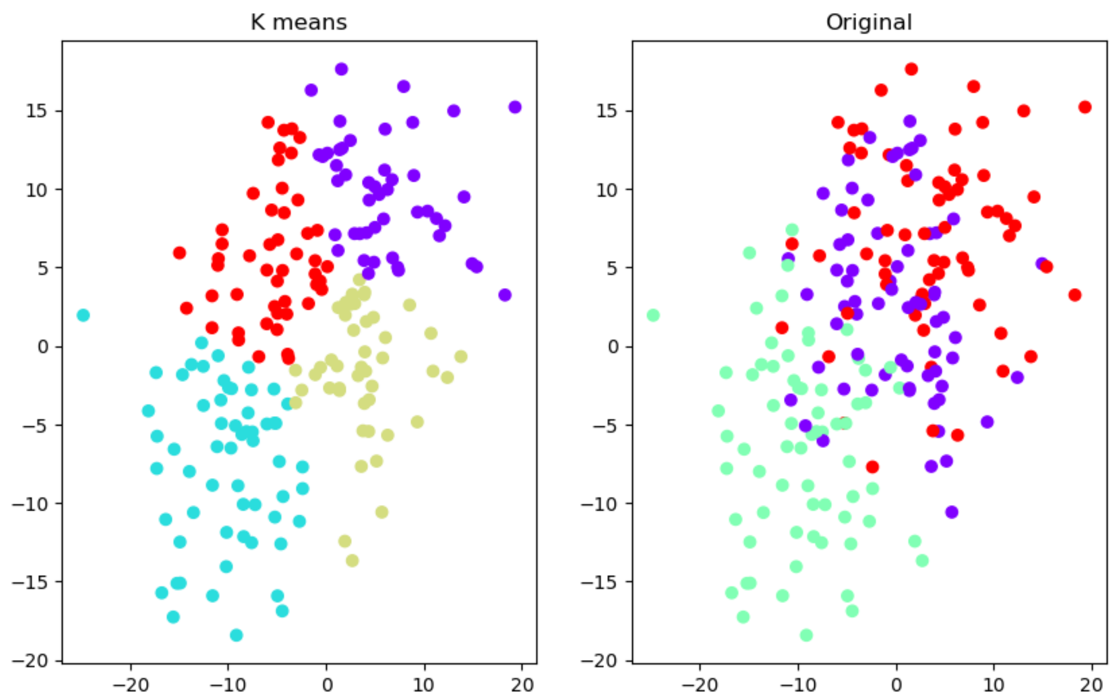
[7]: KMeans(n_clusters=4)

[8]:
```
kmeans.cluster_centers_
```

[8]:
```
array([[  6.13969746,   9.71742729],
       [-10.15568421,  -7.59115373],
       [  3.75744583,  -1.6588095 ],
       [ -5.53321259,   5.67200087]])
```

[9]:
```
fig, (ax1,ax2) = plt.subplots(1,2,figsize=(10,6))
ax1.set_title('K means')
ax1.scatter(data[0][:,0],data[0][:,1],c=kmeans.labels_,cmap='rainbow')
ax2.set_title('Original')
ax2.scatter(data[0][:,0],data[0][:,1],c=data[1],cmap='rainbow')
```

[9]: <matplotlib.collections.PathCollection at 0x2cf016bcd10>



[10]:
```
# Project-4
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("College_Data")
df
```

[10]:
```
                         Unnamed: 0 Private   Apps  Accept  Enroll  Top10perc  \
0      Abilene Christian University     Yes   1660    1232     721         23
1                Adelphi University     Yes   2186    1924     512         16
2                    Adrian College     Yes   1428    1097     336         22
3               Agnes Scott College     Yes    417     349     137         60
4         Alaska Pacific University     Yes    193     146      55         16
..                              …       …     …       …       …          …
772          Worcester State College      No   2197    1515     543          4
773               Xavier University     Yes   1959    1805     695         24
774  Xavier University of Louisiana     Yes   2097    1915     695         34
775                  Yale University     Yes  10705    2453    1317         95
776    York College of Pennsylvania     Yes   2989    1855     691         28

     Top25perc  F.Undergrad  P.Undergrad  Outstate  Room.Board  Books  \
0           52         2885          537      7440        3300    450
1           29         2683         1227     12280        6450    750
```

```
2           50        1036         99     11250       3750     400
3           89         510         63     12960       5450     450
4           44         249        869      7560       4120     800
..         …          …           …       …         …      …
772         26        3089       2029      6797       3900     500
773         47        2849       1107     11520       4960     600
774         61        2793        166      6900       4200     617
775         99        5217         83     19840       6510     630
776         63        2988       1726      4990       3560     500

     Personal  PhD  Terminal  S.F.Ratio  perc.alumni  Expend  Grad.Rate
0        2200   70        78       18.1           12    7041         60
1        1500   29        30       12.2           16   10527         56
2        1165   53        66       12.9           30    8735         54
3         875   92        97        7.7           37   19016         59
4        1500   76        72       11.9            2   10922         15
..        …   …        …        …          …      …        …
772      1200   60        60       21.0           14    4469         40
773      1250   73        75       13.3           31    9189         83
774       781   67        75       14.4           20    8323         49
775      2115   96        96        5.8           49   40386         99
776      1250   75        75       18.1           28    4509         99

[777 rows x 19 columns]
```

[11]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 777 entries, 0 to 776
Data columns (total 19 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   777 non-null    object
 1   Private      777 non-null    object
 2   Apps         777 non-null    int64
 3   Accept       777 non-null    int64
 4   Enroll       777 non-null    int64
 5   Top10perc    777 non-null    int64
 6   Top25perc    777 non-null    int64
 7   F.Undergrad  777 non-null    int64
 8   P.Undergrad  777 non-null    int64
 9   Outstate     777 non-null    int64
 10  Room.Board   777 non-null    int64
 11  Books        777 non-null    int64
 12  Personal     777 non-null    int64
 13  PhD          777 non-null    int64
 14  Terminal     777 non-null    int64
```

```
 15  S.F.Ratio    777 non-null    float64
 16  perc.alumni  777 non-null    int64
 17  Expend       777 non-null    int64
 18  Grad.Rate    777 non-null    int64
dtypes: float64(1), int64(16), object(2)
memory usage: 115.5+ KB
```

[12]:
```python
df.isna().sum()
```

[12]:
```
Unnamed: 0     0
Private        0
Apps           0
Accept         0
Enroll         0
Top10perc      0
Top25perc      0
F.Undergrad    0
P.Undergrad    0
Outstate       0
Room.Board     0
Books          0
Personal       0
PhD            0
Terminal       0
S.F.Ratio      0
perc.alumni    0
Expend         0
Grad.Rate      0
dtype: int64
```

[13]:
```python
df.duplicated()
```

[13]:
```
0      False
1      False
2      False
3      False
4      False
       …
772    False
773    False
774    False
775    False
776    False
Length: 777, dtype: bool
```

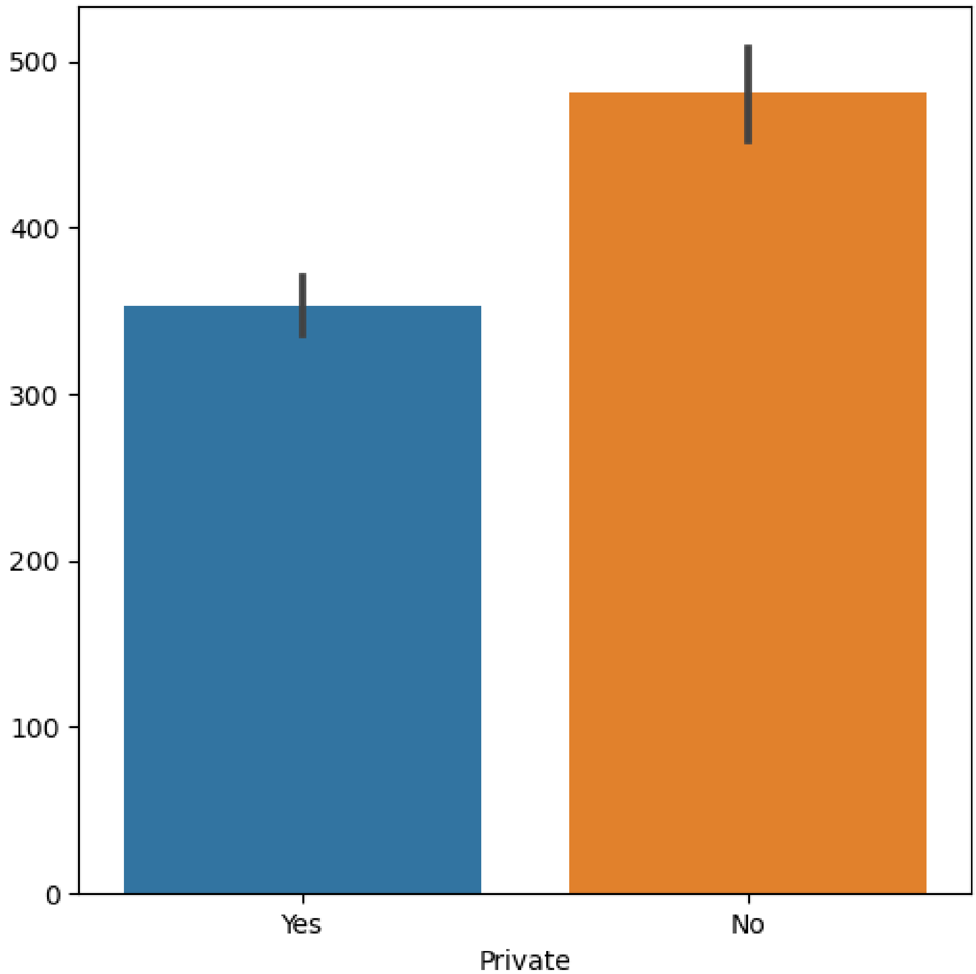
[14]:
```python
if not df[df.duplicated()].empty:
    print(df[df.duplicated()])
```

```
else:
    print("No duplicated datas")
```

```
No duplicated datas
```

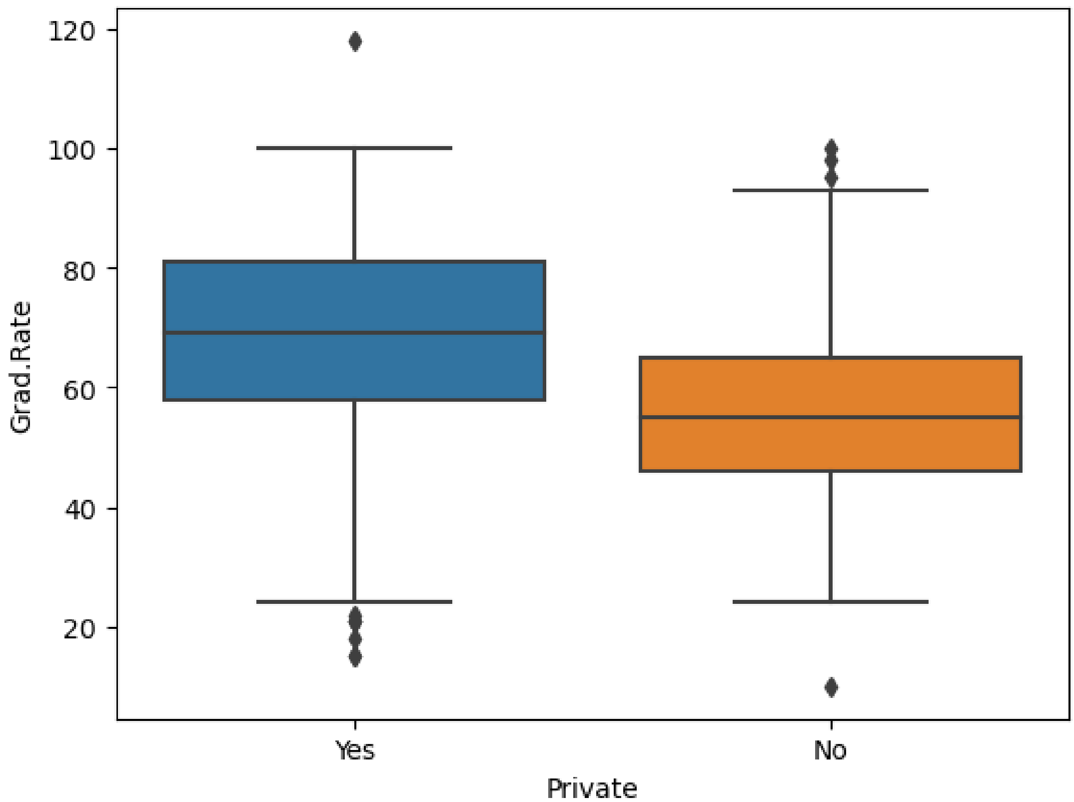
[15]:
```
plt.figure(figsize=(6,6))
sns.barplot(x=df['Private'], y=df.index)
```

[15]: `<Axes: xlabel='Private'>`



[16]:
```
sns.boxplot(x='Private',y='Grad.Rate',data=df)
```

[16]: `<Axes: xlabel='Private', ylabel='Grad.Rate'>`



[17]:
```
plt.savefig("Comparision.png")
```

```
<Figure size 640x480 with 0 Axes>
```

[18]:
```
(df['Grad.Rate']/df['Grad.Rate'].sum())*100
```

[18]:
```
0      0.117959
1      0.110095
2      0.106163
3      0.115993
4      0.029490
         …
772    0.078640
773    0.163177
774    0.096333
775    0.194633
776    0.194633
Name: Grad.Rate, Length: 777, dtype: float64
```

[19]:
```
(df['Grad.Rate'] > 100).sum()
```

[19]: 1

[20]:
```
df[(df['Grad.Rate'] > 100)]
```

[20]:
```
           Unnamed: 0 Private  Apps  Accept  Enroll  Top10perc  Top25perc  \
95  Cazenovia College     Yes  3847    3433     527          9         35

    F.Undergrad  P.Undergrad  Outstate  Room.Board  Books  Personal  PhD  \
95         1010           12      9384        4840    600       500   22

    Terminal  S.F.Ratio  perc.alumni  Expend  Grad.Rate
95        47       14.3           20    7697        118
```

[21]:
```
df.columns
```

[21]:
```
Index(['Unnamed: 0', 'Private', 'Apps', 'Accept', 'Enroll', 'Top10perc',
       'Top25perc', 'F.Undergrad', 'P.Undergrad', 'Outstate', 'Room.Board',
       'Books', 'Personal', 'PhD', 'Terminal', 'S.F.Ratio', 'perc.alumni',
       'Expend', 'Grad.Rate'],
      dtype='object')
```

[22]:
```
df['Grad.Rate'][95] = 100
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_11844\148951594.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Grad.Rate'][95] = 100
```

[23]:
```
(df['Grad.Rate'] > 100).sum()
```

[23]: 0

[24]:
```
df.loc[95]
```

[24]:
```
Unnamed: 0     Cazenovia College
Private                      Yes
Apps                        3847
Accept                      3433
Enroll                       527
Top10perc                      9
Top25perc                     35
F.Undergrad                 1010
P.Undergrad                   12
Outstate                    9384
```

```
Room.Board                 4840
Books                       600
Personal                    500
PhD                          22
Terminal                     47
S.F.Ratio                  14.3
perc.alumni                  20
Expend                     7697
Grad.Rate                   100
Name: 95, dtype: object
```

[25]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2)
df = df.drop('Private',axis=1)
```

[26]:
```
df
```

[26]:
```
                        Unnamed: 0   Apps  Accept  Enroll  Top10perc  \
0     Abilene Christian University   1660    1232     721         23
1               Adelphi University   2186    1924     512         16
2                   Adrian College   1428    1097     336         22
3                Agnes Scott College    417     349     137         60
4          Alaska Pacific University    193     146      55         16
..                             ...    ...     ...     ...        ...
772        Worcester State College   2197    1515     543          4
773              Xavier University   1959    1805     695         24
774  Xavier University of Louisiana   2097    1915     695         34
775                Yale University  10705    2453    1317         95
776    York College of Pennsylvania   2989    1855     691         28

     Top25perc  F.Undergrad  P.Undergrad  Outstate  Room.Board  Books  \
0           52         2885          537      7440        3300    450
1           29         2683         1227     12280        6450    750
2           50         1036           99     11250        3750    400
3           89          510           63     12960        5450    450
4           44          249          869      7560        4120    800
..         ...          ...          ...       ...         ...    ...
772         26         3089         2029      6797        3900    500
773         47         2849         1107     11520        4960    600
774         61         2793          166      6900        4200    617
775         99         5217           83     19840        6510    630
776         63         2988         1726      4990        3560    500

     Personal  PhD  Terminal  S.F.Ratio  perc.alumni  Expend  Grad.Rate
0        2200   70        78       18.1           12    7041         60
1        1500   29        30       12.2           16   10527         56
2        1165   53        66       12.9           30    8735         54
```

```
3         875   92         97         7.7          37   19016           59
4        1500   76         72        11.9           2   10922           15
..        ...  ...        ...         ...         ...     ...
772      1200   60         60        21.0          14    4469           40
773      1250   73         75        13.3          31    9189           83
774       781   67         75        14.4          20    8323           49
775      2115   96         96         5.8          49   40386           99
776      1250   75         75        18.1          28    4509           99

[777 rows x 18 columns]
```

[27]:
```python
df.columns
```

[27]:
```
Index(['Unnamed: 0', 'Apps', 'Accept', 'Enroll', 'Top10perc', 'Top25perc',
       'F.Undergrad', 'P.Undergrad', 'Outstate', 'Room.Board', 'Books',
       'Personal', 'PhD', 'Terminal', 'S.F.Ratio', 'perc.alumni', 'Expend',
       'Grad.Rate'],
      dtype='object')
```

[28]:
```python
kmans = KMeans(n_clusters=7)
features = df.iloc[:,2:]
features
```

[28]:
```
     Accept  Enroll  Top10perc  Top25perc  F.Undergrad  P.Undergrad  Outstate  \
0      1232     721         23         52         2885          537      7440
1      1924     512         16         29         2683         1227     12280
2      1097     336         22         50         1036           99     11250
3       349     137         60         89          510           63     12960
4       146      55         16         44          249          869      7560
..      ...     ...        ...        ...          ...          ...       ...
772    1515     543          4         26         3089         2029      6797
773    1805     695         24         47         2849         1107     11520
774    1915     695         34         61         2793          166      6900
775    2453    1317         95         99         5217           83     19840
776    1855     691         28         63         2988         1726      4990

     Room.Board  Books  Personal  PhD  Terminal  S.F.Ratio  perc.alumni  \
0          3300    450      2200   70        78       18.1           12
1          6450    750      1500   29        30       12.2           16
2          3750    400      1165   53        66       12.9           30
3          5450    450       875   92        97        7.7           37
4          4120    800      1500   76        72       11.9            2
..          ...    ...       ...  ...       ...        ...          ...
772        3900    500      1200   60        60       21.0           14
773        4960    600      1250   73        75       13.3           31
774        4200    617       781   67        75       14.4           20
775        6510    630      2115   96        96        5.8           49
```

```
776        3560     500       1250    75         75        18.1            28

     Expend  Grad.Rate
0      7041         60
1     10527         56
2      8735         54
3     19016         59
4     10922         15
..      …          …
772    4469         40
773    9189         83
774    8323         49
775   40386         99
776    4509         99

[777 rows x 16 columns]
```

[29]:
```python
# convert all columns data type to string to apply standardScalar()
features.columns = features.columns.astype(str)
```

[30]:
```python
from sklearn.preprocessing import StandardScaler
# StandardScaler
# It is a preprocessing class that is used to standardize or normalize the␣
↪features of the dataset
# It scales each feature in such a way that it has a mean of 0 and std of 1
```

[31]:
```python
scaler = StandardScaler()
scaled_featured = scaler.fit_transform(features)
scaled_featured
```

[31]:
```
array([[-0.32120545, -0.0635089 , -0.2585828 , …, -0.86757419,
        -0.50191008, -0.31799293],
       [-0.03870299, -0.28858421, -0.6556556 , …, -0.5445722 ,
         0.16610985, -0.55180463],
       [-0.37631793, -0.47812132, -0.31530749, …,  0.58593475,
        -0.17728996, -0.66871048],
       …,
       [-0.04237716, -0.0915087 ,  0.36538874, …, -0.22157022,
        -0.25624125, -0.96097509],
       [ 0.17725626,  0.57833266,  3.82559456, …,  2.12019418,
         5.88797079,  1.96167109],
       [-0.06687159, -0.09581636,  0.02504063, …,  0.42443375,
        -0.98711561,  1.96167109]])
```

[32]:
```python
scaled_featured.shape
```

[32]:
```
(777, 16)
```

[33]:
```
kmeans = KMeans(n_clusters=2)
```

[34]:
```
df['Cluster'] = kmeans.fit_predict(scaled_featured)
df
```

```
D:\anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning:
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the
value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

[34]:
```
                         Unnamed: 0   Apps  Accept  Enroll  Top10perc  \
0      Abilene Christian University   1660    1232     721         23
1                Adelphi University   2186    1924     512         16
2                    Adrian College   1428    1097     336         22
3               Agnes Scott College    417     349     137         60
4         Alaska Pacific University    193     146      55         16
..                              ...    ...     ...     ...        ...
772         Worcester State College   2197    1515     543          4
773               Xavier University   1959    1805     695         24
774  Xavier University of Louisiana   2097    1915     695         34
775                 Yale University  10705    2453    1317         95
776    York College of Pennsylvania   2989    1855     691         28

     Top25perc  F.Undergrad  P.Undergrad  Outstate  Room.Board  Books  \
0           52         2885          537      7440        3300    450
1           29         2683         1227     12280        6450    750
2           50         1036           99     11250        3750    400
3           89          510           63     12960        5450    450
4           44          249          869      7560        4120    800
..         ...          ...          ...       ...         ...    ...
772         26         3089         2029      6797        3900    500
773         47         2849         1107     11520        4960    600
774         61         2793          166      6900        4200    617
775         99         5217           83     19840        6510    630
776         63         2988         1726      4990        3560    500

     Personal  PhD  Terminal  S.F.Ratio  perc.alumni  Expend  Grad.Rate  \
0        2200   70        78       18.1           12    7041         60
1        1500   29        30       12.2           16   10527         56
2        1165   53        66       12.9           30    8735         54
3         875   92        97        7.7           37   19016         59
4        1500   76        72       11.9            2   10922         15
..        ...  ...       ...        ...          ...     ...        ...
772      1200   60        60       21.0           14    4469         40
773      1250   73        75       13.3           31    9189         83
774       781   67        75       14.4           20    8323         49
775      2115   96        96        5.8           49   40386         99
```

```
776       1250    75         75        18.1           28     4509          99

     Cluster
0          1
1          1
2          1
3          0
4          1
..       …
772        1
773        1
774        1
775        0
776        1

[777 rows x 19 columns]
```

[35]:
```python
kmeans
```

[35]:
```
KMeans(n_clusters=2)
```

[36]:
```python
from sklearn.metrics import confusion_matrix,accuracy_score
print(confusion_matrix(df['Cluster'], kmeans.labels_))
```

```
[[279   0]
 [  0 498]]
```

[37]:
```python
kmeans.labels_
```

[37]:
```
array([1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1,
       1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1,
       0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
       0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0,
```

```
       1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1,
       1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0,
       0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0,
       1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 0, 1])
```

[38]:
```
print(accuracy_score(kmeans.labels_,df['Cluster']))
```

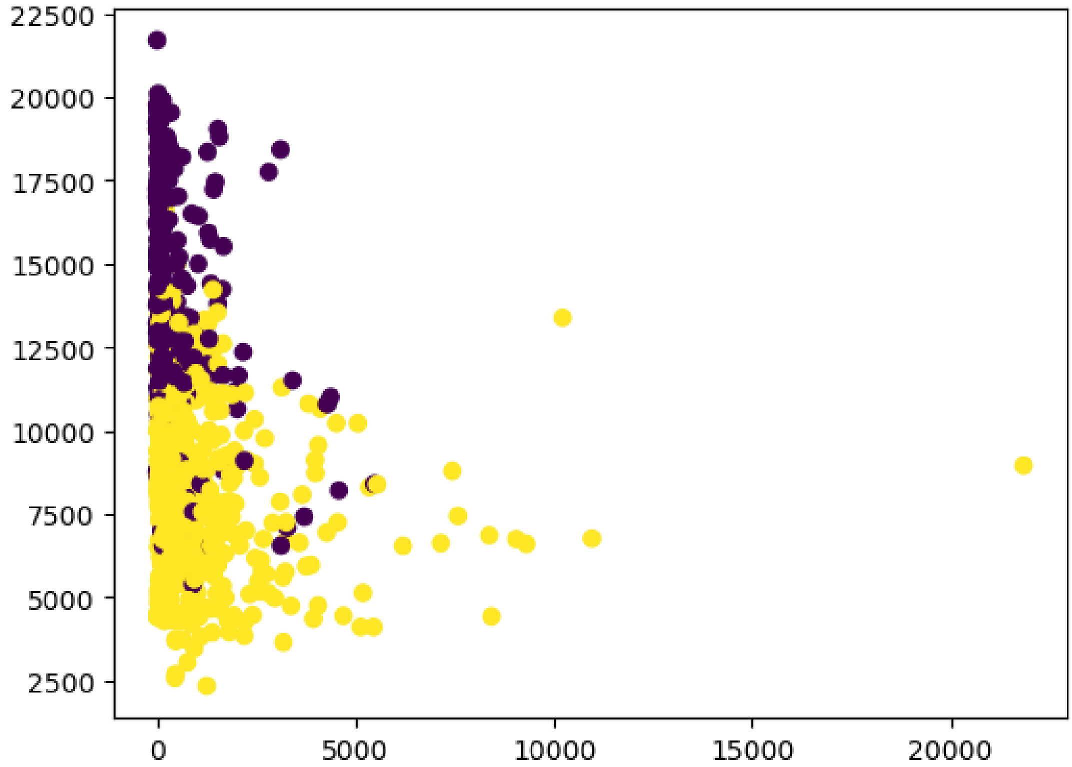
```
1.0
```

[39]:
```
features.columns
```

[39]:
```
Index(['Accept', 'Enroll', 'Top10perc', 'Top25perc', 'F.Undergrad',
       'P.Undergrad', 'Outstate', 'Room.Board', 'Books', 'Personal', 'PhD',
       'Terminal', 'S.F.Ratio', 'perc.alumni', 'Expend', 'Grad.Rate'],
      dtype='object')
```

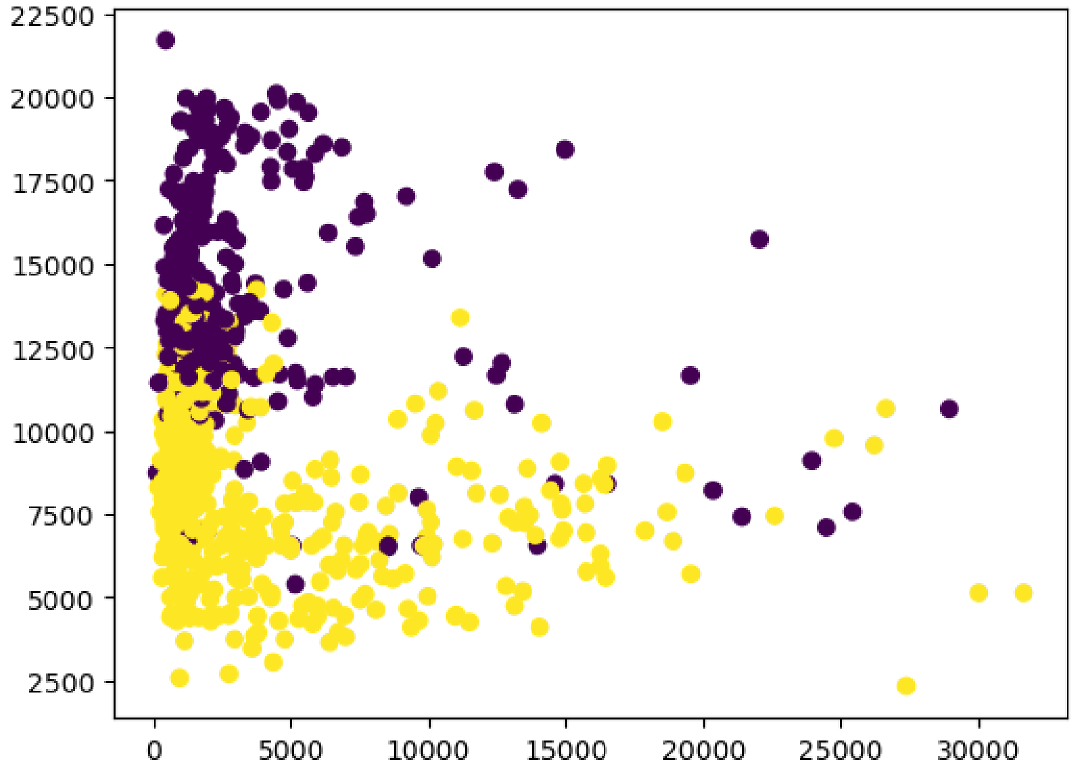
[40]:
```
# To draw a scattered plot -----> PhD
plt.scatter(features['P.Undergrad'],features['Outstate'],c = kmeans.labels_)
```

[40]:
```
<matplotlib.collections.PathCollection at 0x2cf0479c710>
```



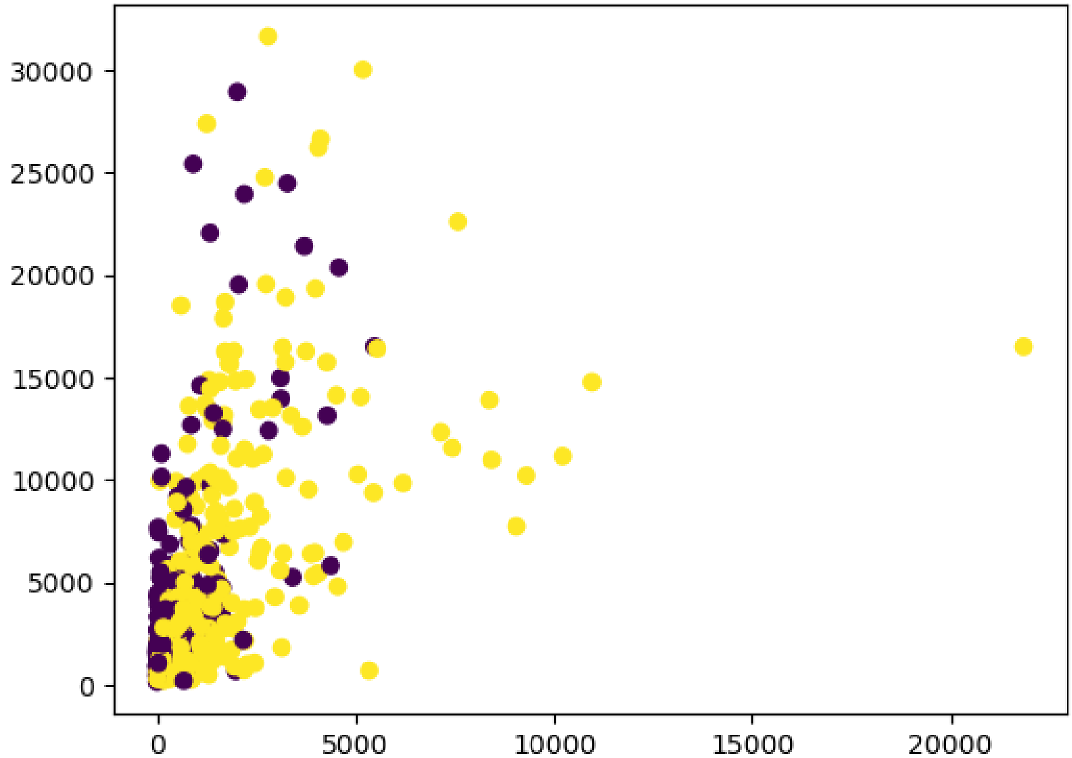

```
[41]: plt.scatter(features['F.Undergrad'],features['Outstate'],c = kmeans.labels_)
```

[41]: <matplotlib.collections.PathCollection at 0x2cf047e9790>



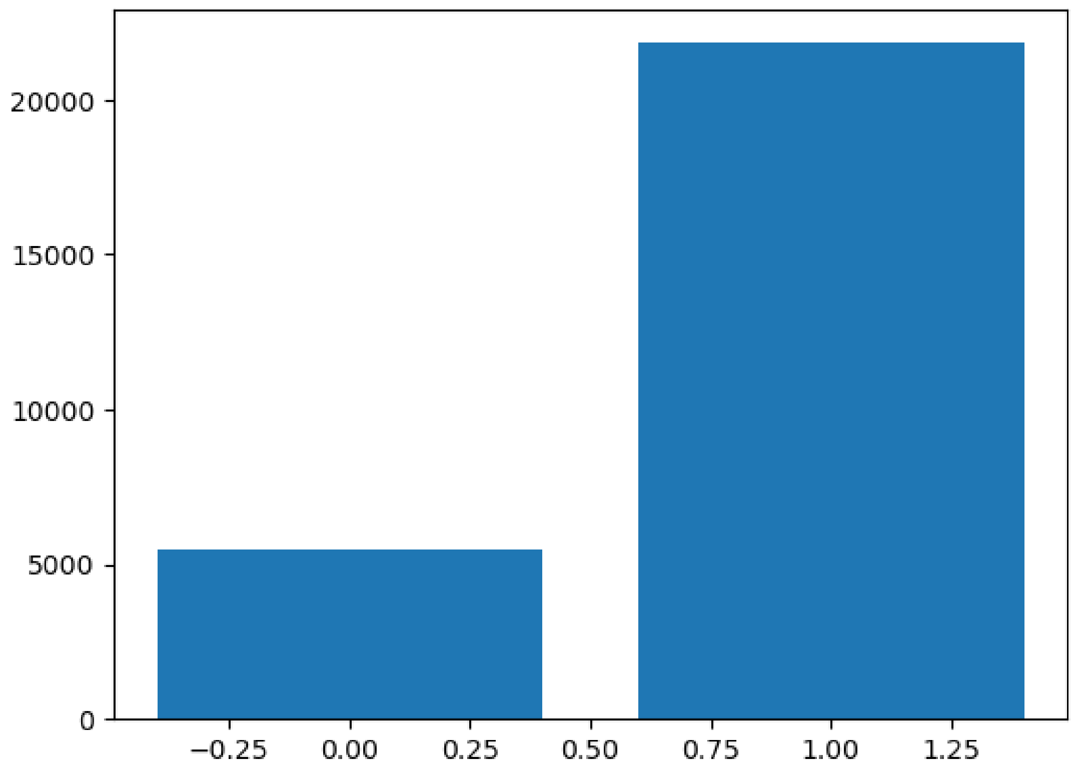
[42]:
```
plt.scatter(features['P.Undergrad'],features['F.Undergrad'],c = kmeans.labels_)
```

[42]: <matplotlib.collections.PathCollection at 0x2cf04506510>



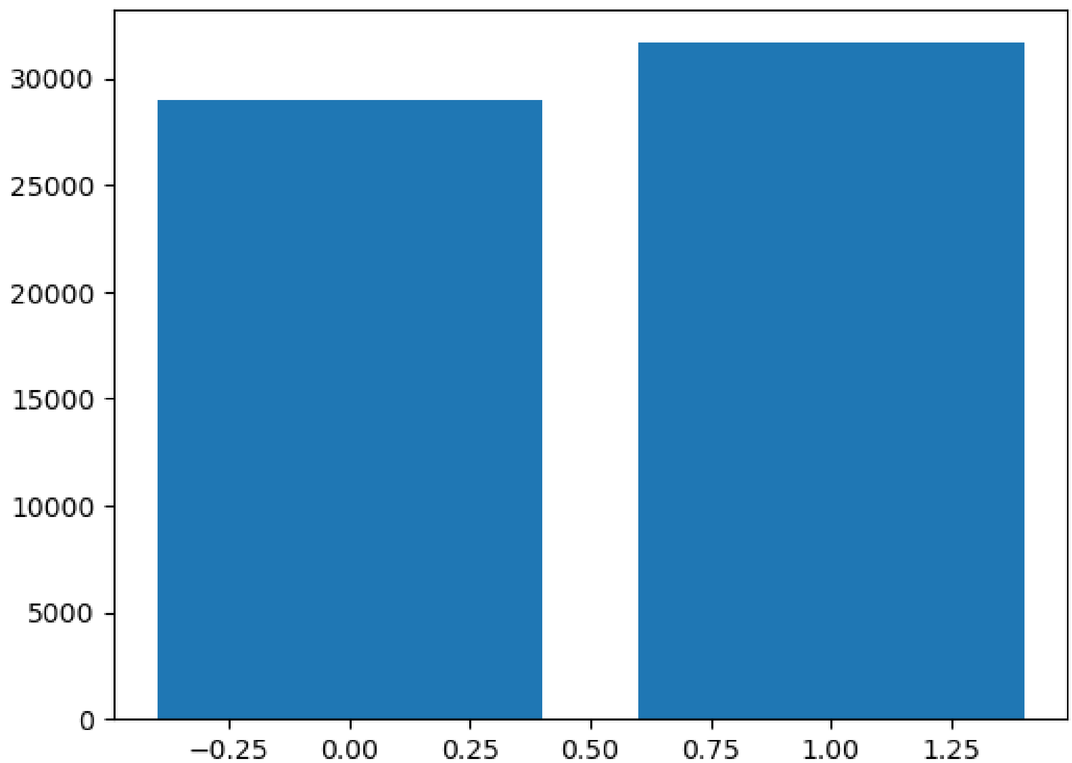
[43]:
```
plt.bar(kmeans.labels_,features['P.Undergrad'])
```

[43]:
```
<BarContainer object of 777 artists>
```



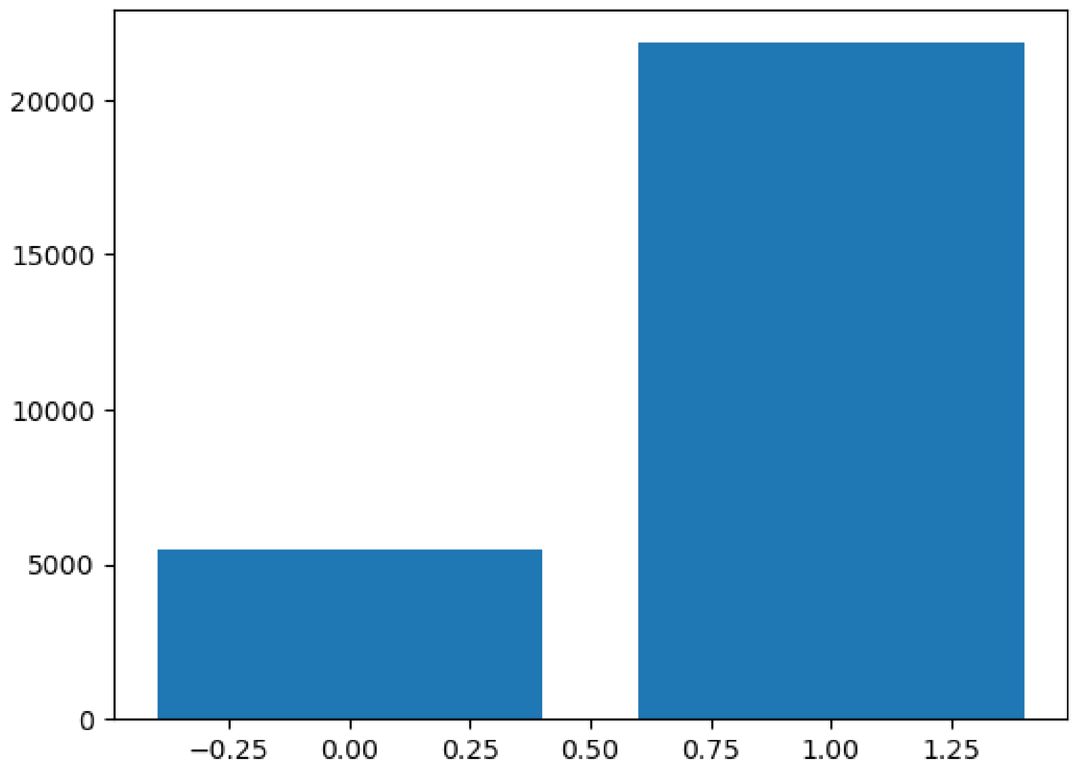
[44]:
```
plt.bar(kmeans.labels_,features['F.Undergrad'])
```

[44]: <BarContainer object of 777 artists>



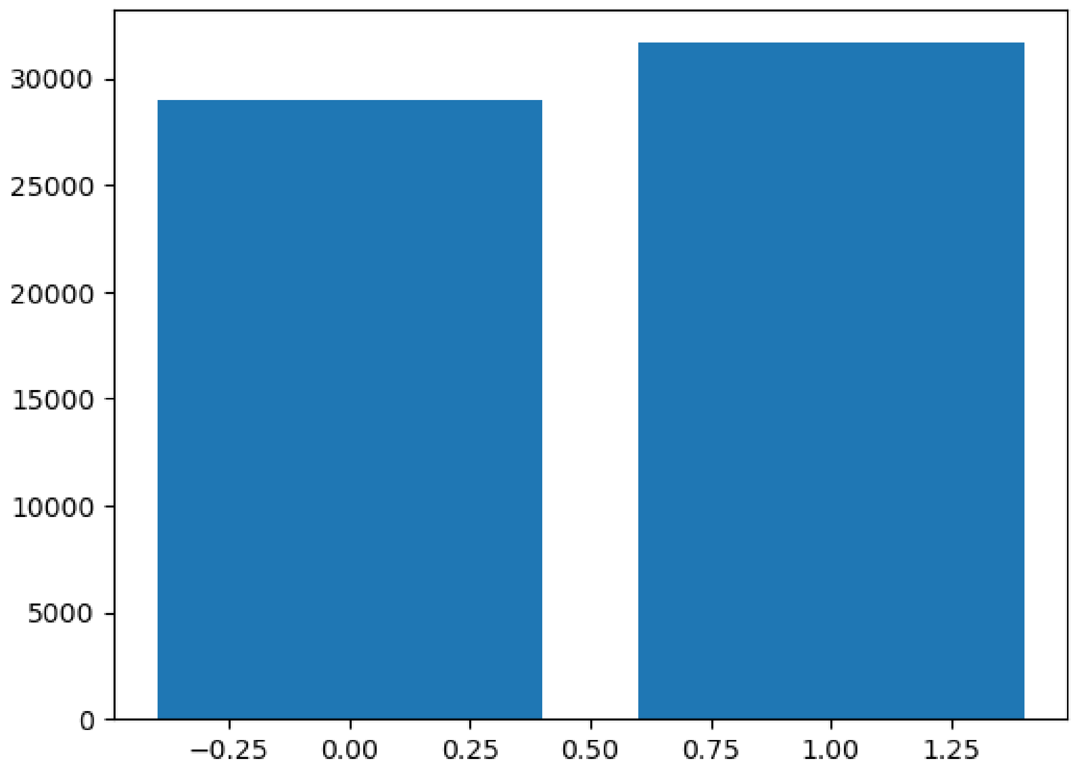
[45]:
```
plt.bar(df['Cluster'],features['P.Undergrad'])
```

[45]: <BarContainer object of 777 artists>



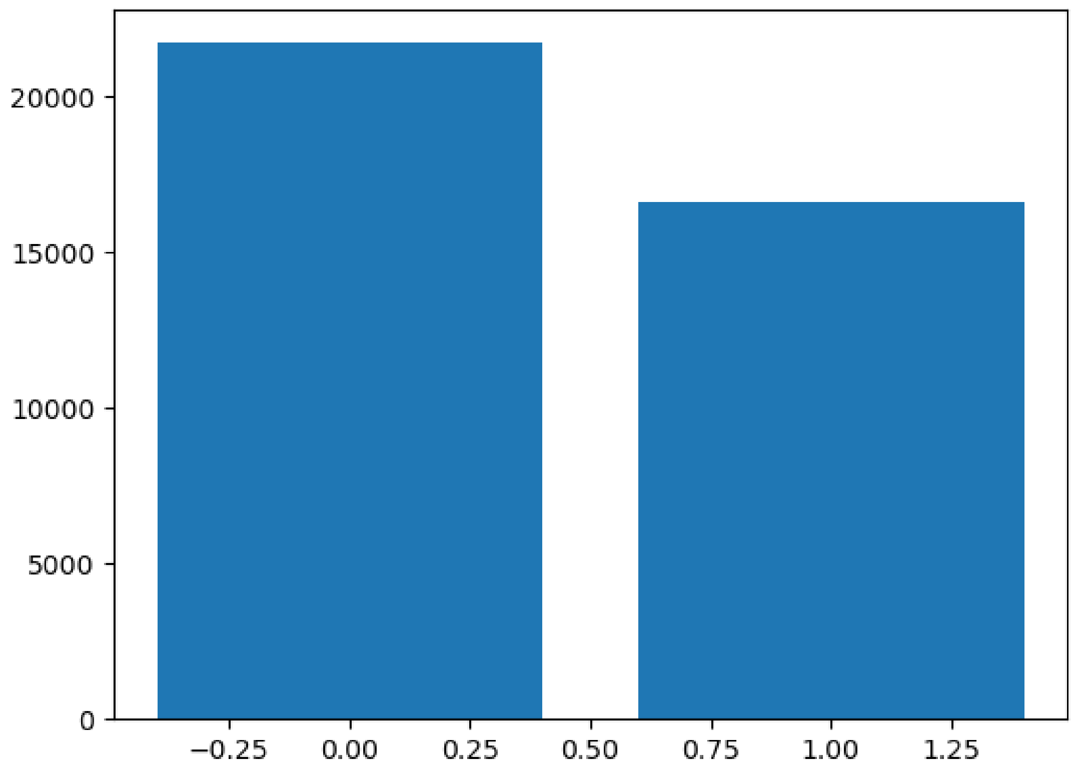
[46]:
```
plt.bar(df['Cluster'],features['F.Undergrad'])
```

[46]: <BarContainer object of 777 artists>



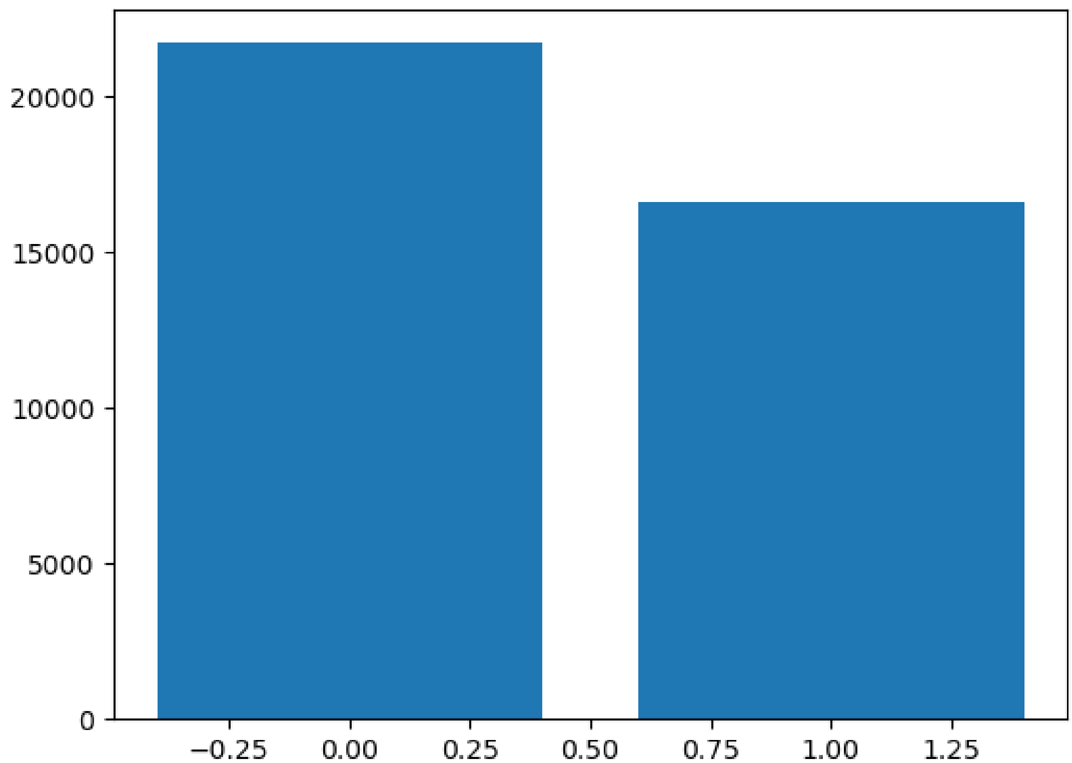
[47]:
```
plt.bar(df['Cluster'],features['Outstate'])
```

[47]: <BarContainer object of 777 artists>



[48]:
```
plt.bar(kmeans.labels_,features['Outstate'])
```

[48]:
```
<BarContainer object of 777 artists>
```



[49]:
```
# KNN is used for classification and regression
# k means is used for clusturing problems
# KNN is supervised algorithm
# k means is unsupervised algorithm
# To training KNN, we need a statement with all the data points having class␣
↪labels
# for training K Means, we need any such information
# we use KNN to predict the class labels or new points
# we use K Means to find patterns in a given dataset by grouping datainputs␣
↪into clusters
```

[50]:
```
# KNN
import pandas as pd
import numpy as np
```

[51]:
```
df = pd.read_csv('Classified Data')
df
```

[51]:
```
   Unnamed: 0       WTT       PTI       EQW       SBI       LQE       QWG  \
0           0  0.913917  1.162073  0.567946  0.755464  0.780862  0.352608
1           1  0.635632  1.003722  0.535342  0.825645  0.924109  0.648450
2           2  0.721360  1.201493  0.921990  0.855595  1.526629  0.720781
```

```
3            3  1.234204  1.386726  0.653046  0.825624  1.142504  0.875128
4            4  1.279491  0.949750  0.627280  0.668976  1.232537  0.703727
..         …        …         …         …         …         …         …
995        995  1.010953  1.034006  0.853116  0.622460  1.036610  0.586240
996        996  0.575529  0.955786  0.941835  0.792882  1.414277  1.269540
997        997  1.135470  0.982462  0.781905  0.916738  0.901031  0.884738
998        998  1.084894  0.861769  0.407158  0.665696  1.608612  0.943859
999        999  0.837460  0.961184  0.417006  0.799784  0.934399  0.424762

          FDJ       PJF       HQE       NXJ  TARGET CLASS
0    0.759697  0.643798  0.879422  1.231409             1
1    0.675334  1.013546  0.621552  1.492702             0
2    1.626351  1.154483  0.957877  1.285597             0
3    1.409708  1.380003  1.522692  1.153093             1
4    1.115596  0.646691  1.463812  1.419167             1
..        …         …         …         …           …
995  0.746811  0.319752  1.117340  1.348517             1
996  1.055928  0.713193  0.958684  1.663489             0
997  0.386802  0.389584  0.919191  1.385504             1
998  0.855806  1.061338  1.277456  1.188063             1
999  0.778234  0.907962  1.257190  1.364837             1

[1000 rows x 12 columns]
```

[52]:
```python
df = pd.read_csv('Classified Data',index_col = 0)
df
```

[52]:
```
          WTT       PTI       EQW       SBI       LQE       QWG       FDJ  \
0    0.913917  1.162073  0.567946  0.755464  0.780862  0.352608  0.759697
1    0.635632  1.003722  0.535342  0.825645  0.924109  0.648450  0.675334
2    0.721360  1.201493  0.921990  0.855595  1.526629  0.720781  1.626351
3    1.234204  1.386726  0.653046  0.825624  1.142504  0.875128  1.409708
4    1.279491  0.949750  0.627280  0.668976  1.232537  0.703727  1.115596
..        …         …         …         …         …         …         …
995  1.010953  1.034006  0.853116  0.622460  1.036610  0.586240  0.746811
996  0.575529  0.955786  0.941835  0.792882  1.414277  1.269540  1.055928
997  1.135470  0.982462  0.781905  0.916738  0.901031  0.884738  0.386802
998  1.084894  0.861769  0.407158  0.665696  1.608612  0.943859  0.855806
999  0.837460  0.961184  0.417006  0.799784  0.934399  0.424762  0.778234

          PJF       HQE       NXJ  TARGET CLASS
0    0.643798  0.879422  1.231409             1
1    1.013546  0.621552  1.492702             0
2    1.154483  0.957877  1.285597             0
3    1.380003  1.522692  1.153093             1
4    0.646691  1.463812  1.419167             1
..        …         …         …           …
```

```
995  0.319752  1.117340  1.348517            1
996  0.713193  0.958684  1.663489            0
997  0.389584  0.919191  1.385504            1
998  1.061338  1.277456  1.188063            1
999  0.907962  1.257190  1.364837            1

[1000 rows x 11 columns]
```

[53]:
```python
# Apply standard scaler
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
```

[54]:
```python
scalar.fit(df.drop('TARGET CLASS',axis=1))
```

[54]:
```
StandardScaler()
```

[55]:
```python
scalar_standard = scalar.transform(df.drop('TARGET CLASS',axis=1))
scalar_standard
```

[55]:
```
array([[-0.12354188,  0.18590747, -0.91343069, …, -1.48236813,
        -0.9497194 , -0.64331425],
       [-1.08483602, -0.43034845, -1.02531333, …, -0.20224031,
        -1.82805088,  0.63675862],
       [-0.78870217,  0.33931821,  0.30151137, …,  0.28570652,
        -0.68249379, -0.37784986],
       …,
       [ 0.64177714, -0.51308341, -0.17920486, …, -2.36249443,
        -0.81426092,  0.11159651],
       [ 0.46707241, -0.98278576, -1.46519359, …, -0.03677699,
         0.40602453, -0.85567   ],
       [-0.38765353, -0.59589427, -1.4313981 , …, -0.56778932,
         0.3369971 ,  0.01034996]])
```

[56]:
```python
df_feat = pd.DataFrame(scalar_standard)
df_feat.head()
```

[56]:
```
          0         1         2         3         4         5         6  \
0 -0.123542  0.185907 -0.913431  0.319629 -1.033637 -2.308375 -0.798951
1 -1.084836 -0.430348 -1.025313  0.625388 -0.444847 -1.152706 -1.129797
2 -0.788702  0.339318  0.301511  0.755873  2.031693 -0.870156  2.599818
3  0.982841  1.060193 -0.621399  0.625299  0.452820 -0.267220  1.750208
4  1.139275 -0.640392 -0.709819 -0.057175  0.822886 -0.936773  0.596782

          7         8         9
0 -1.482368 -0.949719 -0.643314
1 -0.202240 -1.828051  0.636759
2  0.285707 -0.682494 -0.377850
```

```
3  1.066491  1.241325 -1.026987
4 -1.472352  1.040772  0.276510
```

[57]:
```python
# Example of standard scalar
data = np.array([[0,0],[0,1],[1,0],[1,1]])
data
```

[57]:
```
array([[0, 0],
       [0, 1],
       [1, 0],
       [1, 1]])
```

[58]:
```python
scl = StandardScaler()
```

[59]:
```python
scl_data = scl.fit_transform(data)
```

[60]:
```python
data
```

[60]:
```
array([[0, 0],
       [0, 1],
       [1, 0],
       [1, 1]])
```

[61]:
```python
scl_data
```

[61]:
```
array([[-1., -1.],
       [-1.,  1.],
       [ 1., -1.],
       [ 1.,  1.]])
```

[62]:
```python
scl_data.mean()
```

[62]:
```
0.0
```

[63]:
```python
scl_data.std()
```

[63]:
```
1.0
```

[64]:
```python
df
```

[64]:
```
         WTT       PTI       EQW       SBI       LQE       QWG       FDJ  \
0   0.913917  1.162073  0.567946  0.755464  0.780862  0.352608  0.759697
1   0.635632  1.003722  0.535342  0.825645  0.924109  0.648450  0.675334
2   0.721360  1.201493  0.921990  0.855595  1.526629  0.720781  1.626351
3   1.234204  1.386726  0.653046  0.825624  1.142504  0.875128  1.409708
4   1.279491  0.949750  0.627280  0.668976  1.232537  0.703727  1.115596
..       …         …         …         …         …         …         …
```

```
995  1.010953  1.034006  0.853116  0.622460  1.036610  0.586240  0.746811
996  0.575529  0.955786  0.941835  0.792882  1.414277  1.269540  1.055928
997  1.135470  0.982462  0.781905  0.916738  0.901031  0.884738  0.386802
998  1.084894  0.861769  0.407158  0.665696  1.608612  0.943859  0.855806
999  0.837460  0.961184  0.417006  0.799784  0.934399  0.424762  0.778234

          PJF       HQE       NXJ  TARGET CLASS
0    0.643798  0.879422  1.231409             1
1    1.013546  0.621552  1.492702             0
2    1.154483  0.957877  1.285597             0
3    1.380003  1.522692  1.153093             1
4    0.646691  1.463812  1.419167             1
..        …         …         …           …
995  0.319752  1.117340  1.348517             1
996  0.713193  0.958684  1.663489             0
997  0.389584  0.919191  1.385504             1
998  1.061338  1.277456  1.188063             1
999  0.907962  1.257190  1.364837             1

[1000 rows x 11 columns]
```

[65]:
```python
df_feat.head()
```

[65]:
```
          0         1         2         3         4         5         6  \
0 -0.123542  0.185907 -0.913431  0.319629 -1.033637 -2.308375 -0.798951
1 -1.084836 -0.430348 -1.025313  0.625388 -0.444847 -1.152706 -1.129797
2 -0.788702  0.339318  0.301511  0.755873  2.031693 -0.870156  2.599818
3  0.982841  1.060193 -0.621399  0.625299  0.452820 -0.267220  1.750208
4  1.139275 -0.640392 -0.709819 -0.057175  0.822886 -0.936773  0.596782

          7         8         9
0 -1.482368 -0.949719 -0.643314
1 -0.202240 -1.828051  0.636759
2  0.285707 -0.682494 -0.377850
3  1.066491  1.241325 -1.026987
4 -1.472352  1.040772  0.276510
```

[66]:
```python
df_feat = pd.DataFrame(scalar_standard,columns=df.columns[:-1])
df_feat
```

[66]:
```
          WTT       PTI       EQW       SBI       LQE       QWG       FDJ  \
0   -0.123542  0.185907 -0.913431  0.319629 -1.033637 -2.308375 -0.798951
1   -1.084836 -0.430348 -1.025313  0.625388 -0.444847 -1.152706 -1.129797
2   -0.788702  0.339318  0.301511  0.755873  2.031693 -0.870156  2.599818
3    0.982841  1.060193 -0.621399  0.625299  0.452820 -0.267220  1.750208
4    1.139275 -0.640392 -0.709819 -0.057175  0.822886 -0.936773  0.596782
..        …         …         …         …         …         …         …
```

```
995  0.211653 -0.312490  0.065163 -0.259834  0.017567 -1.395721 -0.849486
996 -1.292453 -0.616901  0.369613  0.482648  1.569891  1.273495  0.362784
997  0.641777 -0.513083 -0.179205  1.022255 -0.539703 -0.229680 -2.261339
998  0.467072 -0.982786 -1.465194 -0.071465  2.368666  0.001269 -0.422041
999 -0.387654 -0.595894 -1.431398  0.512722 -0.402552 -2.026512 -0.726253

          PJF       HQE       NXJ
0   -1.482368 -0.949719 -0.643314
1   -0.202240 -1.828051  0.636759
2    0.285707 -0.682494 -0.377850
3    1.066491  1.241325 -1.026987
4   -1.472352  1.040772  0.276510
..        …         …         …
995 -2.604264 -0.139347 -0.069602
996 -1.242110 -0.679746  1.473448
997 -2.362494 -0.814261  0.111597
998 -0.036777  0.406025 -0.855670
999 -0.567789  0.336997  0.010350

[1000 rows x 10 columns]
```

[67]:
```
df_feat.isnull().sum()
```

[67]:
```
WTT    0
PTI    0
EQW    0
SBI    0
LQE    0
QWG    0
FDJ    0
PJF    0
HQE    0
NXJ    0
dtype: int64
```

[68]:
```
df_feat.isna().sum()
```

[68]:
```
WTT    0
PTI    0
EQW    0
SBI    0
LQE    0
QWG    0
FDJ    0
PJF    0
HQE    0
NXJ    0
```

```
dtype: int64
```

[69]:
```
from sklearn.model_selection import train_test_split
x = df_feat
y = df['TARGET CLASS']
```

[70]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.
 ↪3,random_state=101)
x_train.shape
```

[70]:
```
(700, 10)
```

[71]:
```
x_train.shape
```

[71]:
```
(700, 10)
```

[72]:
```
x_test.shape
```

[72]:
```
(300, 10)
```

[73]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[74]:
```
knn = KNeighborsClassifier(n_neighbors=13)
knn
```

[74]:
```
KNeighborsClassifier(n_neighbors=13)
```

[75]:
```
# To train the model
knn.fit(x_train,y_train)
```

[75]:
```
KNeighborsClassifier(n_neighbors=13)
```

[76]:
```
pred = knn.predict(x_test)
pred
```

[76]:
```
array([0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1,
       1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0,
       1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1,
```

```
        1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0,
        0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0], dtype=int64)
```

[77]:
```python
from sklearn.metrics import accuracy_score
accuracy_score = accuracy_score(pred,y_test)
accuracy_score
```

[77]:
```
0.95
```

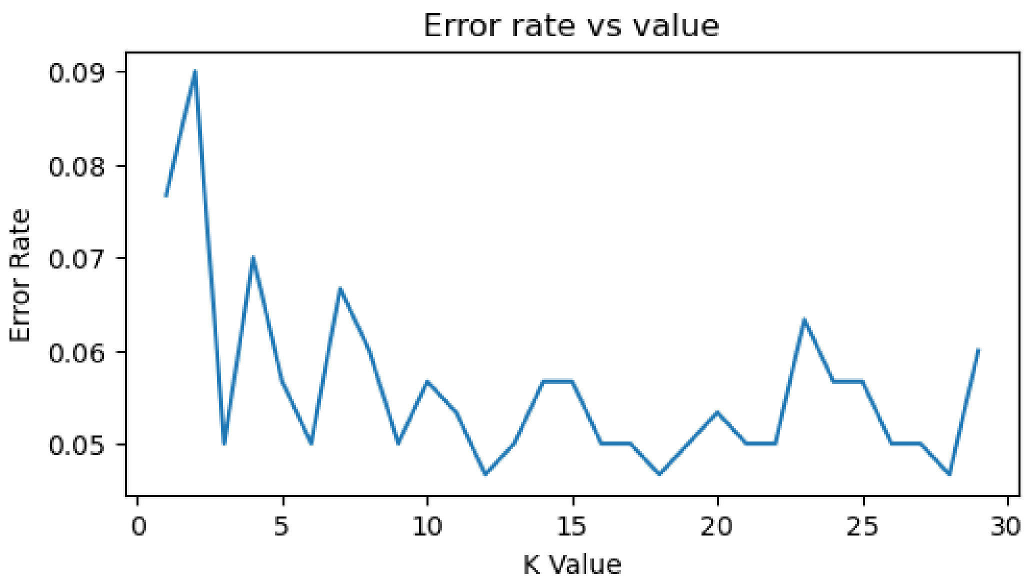
[78]:
```python
error_rate = []
for sal in range(1,30):
    knn = KNeighborsClassifier(n_neighbors=sal)
    knn.fit(x_train,y_train)
    pred_i = knn.predict(x_test)
    error_rate.append(np.mean(pred_i != y_test))
```

[79]:
```python
error_rate
```

[79]:
```
[0.07666666666666666,
 0.09,
 0.05,
 0.07,
 0.056666666666666664,
 0.05,
 0.06666666666666667,
 0.06,
 0.05,
 0.056666666666666664,
 0.05333333333333334,
 0.04666666666666667,
 0.05,
 0.056666666666666664,
 0.056666666666666664,
 0.05,
 0.05,
 0.04666666666666667,
 0.05,
 0.05333333333333334,
 0.05,
 0.05,
 0.06333333333333334,
 0.056666666666666664,
 0.056666666666666664,
 0.05,
 0.05,
 0.04666666666666667,
 0.06]
```

[80]:
```python
import matplotlib.pyplot as plt
plt.figure(figsize=(6,3))
plt.plot(range(1,30),error_rate)
plt.title("Error rate vs value")
plt.xlabel("K Value")
plt.ylabel("Error Rate")
```

[80]: Text(0, 0.5, 'Error Rate')



[81]:
```python
df = pd.read_csv("cancerKNNAlgorithmDataset.csv")
df
```

[81]:
```
           id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0      842302         M        17.99         10.38          122.80     1001.0
1      842517         M        20.57         17.77          132.90     1326.0
2    84300903         M        19.69         21.25          130.00     1203.0
3    84348301         M        11.42         20.38           77.58      386.1
4    84358402         M        20.29         14.34          135.10     1297.0
..        …       …          …           …             …          …
564    926424         M        21.56         22.39          142.00     1479.0
565    926682         M        20.13         28.25          131.20     1261.0
566    926954         M        16.60         28.08          108.30      858.1
567    927241         M        20.60         29.33          140.10     1265.0
568     92751         B         7.76         24.54           47.92      181.0

     smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0            0.11840           0.27760         0.30010              0.14710
```

```
1            0.08474           0.07864          0.08690                 0.07017
2            0.10960           0.15990          0.19740                 0.12790
3            0.14250           0.28390          0.24140                 0.10520
4            0.10030           0.13280          0.19800                 0.10430
..               …                 …                …                     …
564          0.11100           0.11590          0.24390                 0.13890
565          0.09780           0.10340          0.14400                 0.09791
566          0.08455           0.10230          0.09251                 0.05302
567          0.11780           0.27700          0.35140                 0.15200
568          0.05263           0.04362          0.00000                 0.00000

     …  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
0    …          17.33           184.60      2019.0           0.16220
1    …          23.41           158.80      1956.0           0.12380
2    …          25.53           152.50      1709.0           0.14440
3    …          26.50            98.87       567.7           0.20980
4    …          16.67           152.20      1575.0           0.13740
..   …            …                …           …                …
564  …          26.40           166.10      2027.0           0.14100
565  …          38.25           155.00      1731.0           0.11660
566  …          34.12           126.70      1124.0           0.11390
567  …          39.42           184.60      1821.0           0.16500
568  …          30.37            59.16       268.6           0.08996

     compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
0              0.66560           0.7119                0.2654          0.4601
1              0.18660           0.2416                0.1860          0.2750
2              0.42450           0.4504                0.2430          0.3613
3              0.86630           0.6869                0.2575          0.6638
4              0.20500           0.4000                0.1625          0.2364
..                 …                …                     …               …
564            0.21130           0.4107                0.2216          0.2060
565            0.19220           0.3215                0.1628          0.2572
566            0.30940           0.3403                0.1418          0.2218
567            0.86810           0.9387                0.2650          0.4087
568            0.06444           0.0000                0.0000          0.2871

     fractal_dimension_worst  Unnamed: 32
0                    0.11890          NaN
1                    0.08902          NaN
2                    0.08758          NaN
3                    0.17300          NaN
4                    0.07678          NaN
..                       …            …
564                  0.07115          NaN
565                  0.06637          NaN
566                  0.07820          NaN
```

```
567                    0.12400           NaN
568                    0.07039           NaN

[569 rows x 33 columns]
```

[ ]: