import numpy as np # linear algebra  
import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)  
import matplotlib.pyplot as plt # Data Visualization   
import seaborn as sns # Python library for Vidualization  
# Comment this if the data visualisations doesn't work on your side  
%matplotlib inline  
  
plt.style.use('bmh')  
from sklearn.preprocessing import StandardScaler  
from sklearn.cluster import KMeans  
from sklearn.metrics import confusion\_matrix, classification\_report

# Import the dataset  
dataset = pd.read\_csv('college.csv')  
dataset.head()

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   
  
 Top25perc F.Undergrad P.Undergrad Outstate Room.Board Books Personal \  
0 52 2885 537 7440 3300 450 2200   
1 29 2683 1227 12280 6450 750 1500   
2 50 1036 99 11250 3750 400 1165   
3 89 510 63 12960 5450 450 875   
4 44 249 869 7560 4120 800 1500   
  
 PhD Terminal S.F.Ratio perc.alumni Expend Grad.Rate   
0 70 78 18.1 12 7041 60   
1 29 30 12.2 16 10527 56   
2 53 66 12.9 30 8735 54   
3 92 97 7.7 37 19016 59   
4 76 72 11.9 2 10922 15

# total rows and colums in the dataset  
dataset.shape

(777, 19)

dataset.rename(columns= {"Unnamed: 0":"college\_name"}, inplace = True)

dataset.info() # there are no missing values

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 777 entries, 0 to 776  
Data columns (total 19 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 college\_name 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

pd.set\_option('display.float', '{:.2f}'.format)  
pd.set\_option('display.max\_rows', None)  
pd.set\_option('display.max\_columns', None)  
pd.set\_option('display.width', None)  
pd.set\_option('display.max\_colwidth', None)  
  
dataset.describe().to\_csv("dataset\_describe.csv")  
dataset.describe()

Apps Accept Enroll Top10perc Top25perc F.Undergrad \  
count 777.00 777.00 777.00 777.00 777.00 777.00   
mean 3001.64 2018.80 779.97 27.56 55.80 3699.91   
std 3870.20 2451.11 929.18 17.64 19.80 4850.42   
min 81.00 72.00 35.00 1.00 9.00 139.00   
25% 776.00 604.00 242.00 15.00 41.00 992.00   
50% 1558.00 1110.00 434.00 23.00 54.00 1707.00   
75% 3624.00 2424.00 902.00 35.00 69.00 4005.00   
max 48094.00 26330.00 6392.00 96.00 100.00 31643.00   
  
 P.Undergrad Outstate Room.Board Books Personal PhD Terminal \  
count 777.00 777.00 777.00 777.00 777.00 777.00 777.00   
mean 855.30 10440.67 4357.53 549.38 1340.64 72.66 79.70   
std 1522.43 4023.02 1096.70 165.11 677.07 16.33 14.72   
min 1.00 2340.00 1780.00 96.00 250.00 8.00 24.00   
25% 95.00 7320.00 3597.00 470.00 850.00 62.00 71.00   
50% 353.00 9990.00 4200.00 500.00 1200.00 75.00 82.00   
75% 967.00 12925.00 5050.00 600.00 1700.00 85.00 92.00   
max 21836.00 21700.00 8124.00 2340.00 6800.00 103.00 100.00   
  
 S.F.Ratio perc.alumni Expend Grad.Rate   
count 777.00 777.00 777.00 777.00   
mean 14.09 22.74 9660.17 65.46   
std 3.96 12.39 5221.77 17.18   
min 2.50 0.00 3186.00 10.00   
25% 11.50 13.00 6751.00 53.00   
50% 13.60 21.00 8377.00 65.00   
75% 16.50 31.00 10830.00 78.00   
max 39.80 64.00 56233.00 118.00

dataset = dataset[dataset["Grad.Rate"]<= 100]

dataset.describe().to\_csv("dataset\_describe\_updated.csv")  
dataset.describe()

Apps Accept Enroll Top10perc Top25perc F.Undergrad \  
count 776.00 776.00 776.00 776.00 776.00 776.00   
mean 3000.55 2016.98 780.30 27.58 55.82 3703.37   
std 3872.58 2452.17 929.73 17.64 19.80 4852.59   
min 81.00 72.00 35.00 1.00 9.00 139.00   
25% 776.00 603.25 242.00 15.00 41.00 991.00   
50% 1557.50 1109.50 434.00 23.00 54.00 1707.50   
75% 3603.00 2407.50 902.25 35.00 69.00 4030.25   
max 48094.00 26330.00 6392.00 96.00 100.00 31643.00   
  
 P.Undergrad Outstate Room.Board Books Personal PhD Terminal \  
count 776.00 776.00 776.00 776.00 776.00 776.00 776.00   
mean 856.39 10442.03 4356.90 549.32 1341.73 72.73 79.74   
std 1523.11 4025.43 1097.27 165.20 676.83 16.24 14.68   
min 1.00 2340.00 1780.00 96.00 250.00 8.00 24.00   
25% 95.00 7305.00 3595.75 469.50 865.00 62.00 71.00   
50% 354.00 9990.00 4197.50 500.00 1200.00 75.00 82.00   
75% 967.25 12931.25 5050.00 600.00 1700.00 85.00 92.00   
max 21836.00 21700.00 8124.00 2340.00 6800.00 103.00 100.00   
  
 S.F.Ratio perc.alumni Expend Grad.Rate   
count 776.00 776.00 776.00 776.00   
mean 14.09 22.75 9662.70 65.40   
std 3.96 12.40 5224.66 17.08   
min 2.50 0.00 3186.00 10.00   
25% 11.50 13.00 6749.25 53.00   
50% 13.60 21.00 8392.50 65.00   
75% 16.50 31.00 10838.50 78.00   
max 39.80 64.00 56233.00 100.00

dataset["Private"].value\_counts()

Yes 564  
No 212  
Name: Private, dtype: int64

# Missing values computation  
dataset.isnull().sum()

college\_name 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

## EDA

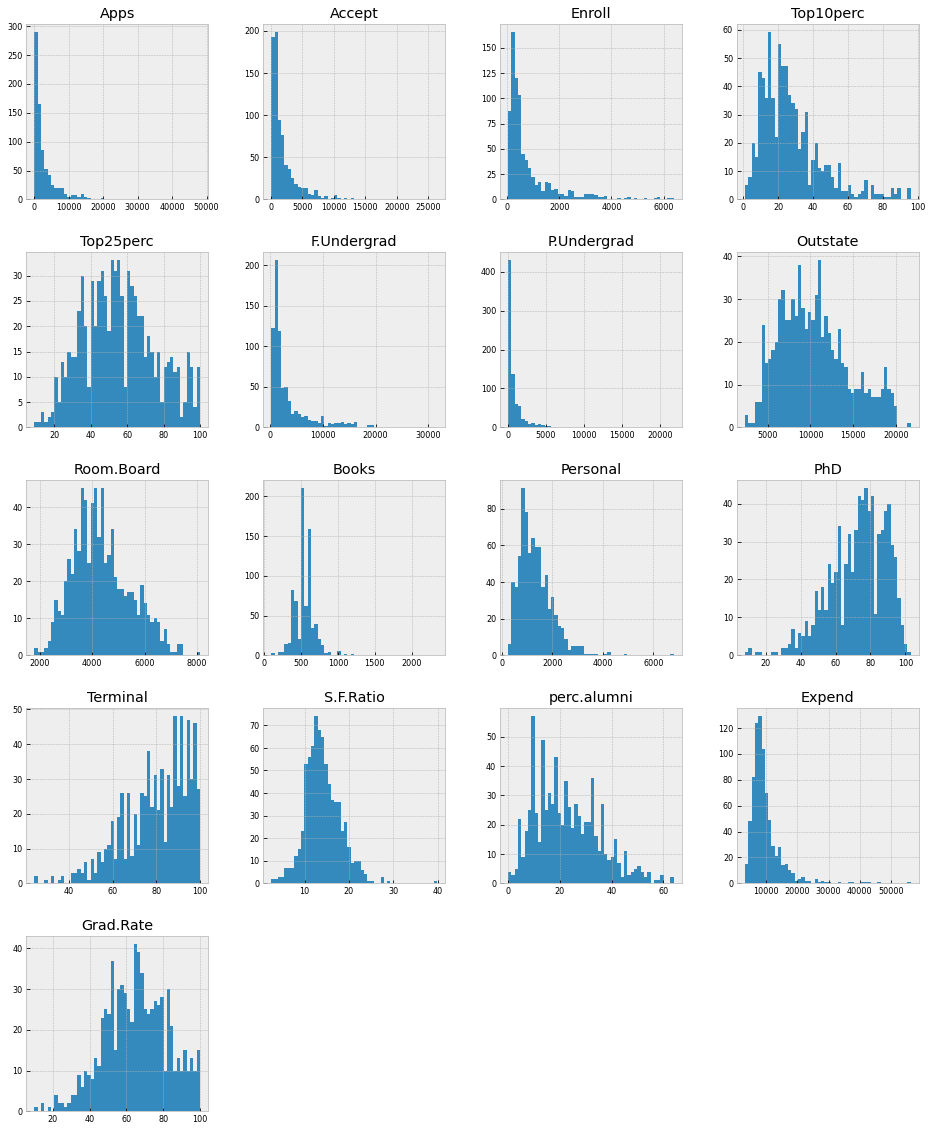
dataset.groupby(["Private"], as\_index = False).agg(app\_total=pd.NamedAgg(column="Apps", aggfunc="sum"),\  
 acpt\_total=pd.NamedAgg(column="Accept", aggfunc="sum"),\  
 enl\_total=pd.NamedAgg(column="Enroll", aggfunc="sum"),\  
 outstate\_total=pd.NamedAgg(column="Outstate", aggfunc="sum"),\  
 PhD\_total=pd.NamedAgg(column="PhD", aggfunc="sum"),\  
 avg\_expend=pd.NamedAgg(column="Expend", aggfunc="mean"),\  
 avg\_grad\_rate=pd.NamedAgg(column="Grad.Rate", aggfunc="mean"))

Private app\_total acpt\_total enl\_total outstate\_total PhD\_total \  
0 No 1214743 830889 347865 1444443 16289   
1 Yes 1113683 734289 257647 6658573 40146   
  
 avg\_expend avg\_grad\_rate   
0 7458.32 56.04   
1 10491.30 68.91

df\_num = dataset.select\_dtypes(include = ['float64', 'int64'])  
df\_num.head()

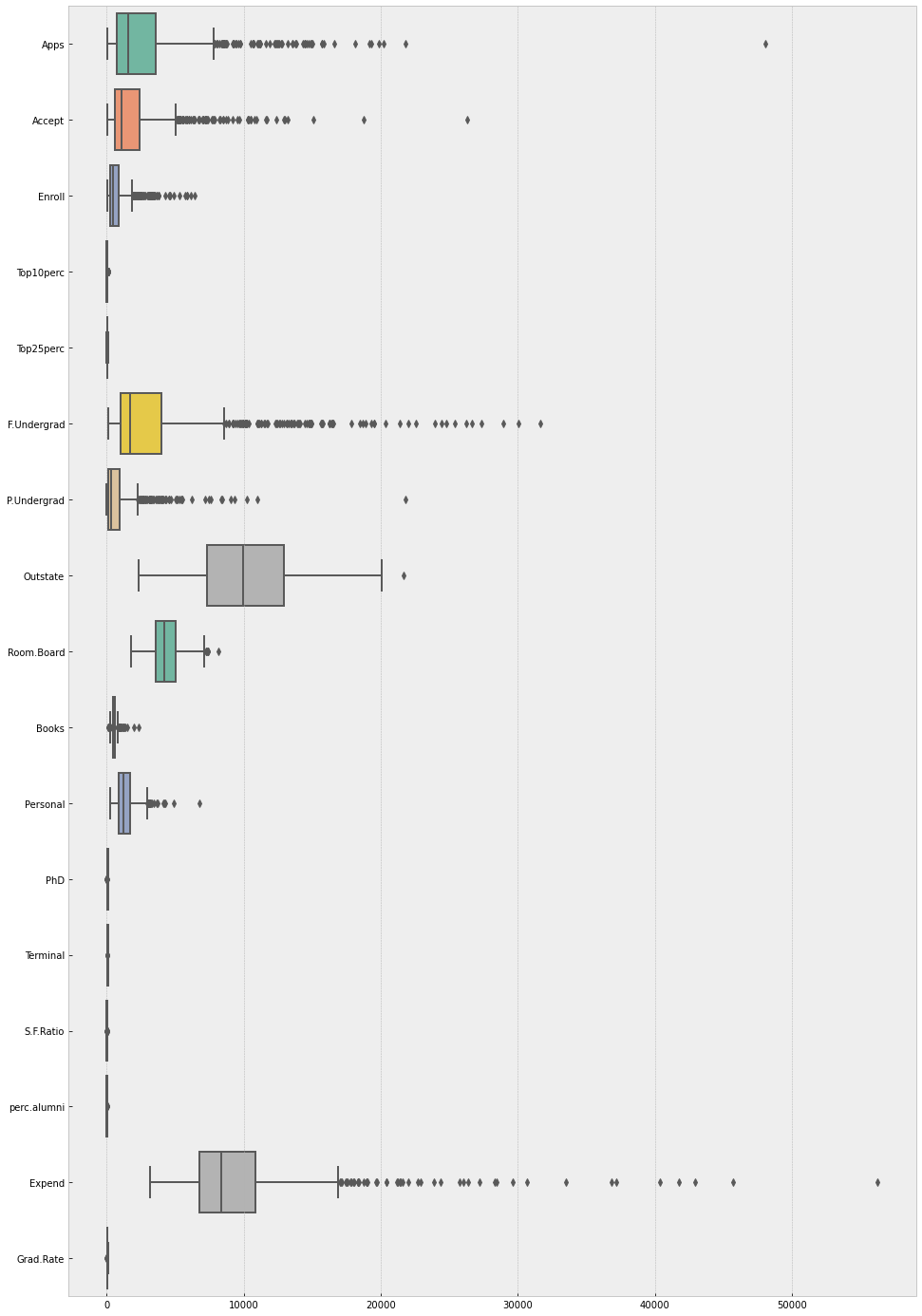
Apps Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad \  
0 1660 1232 721 23 52 2885 537   
1 2186 1924 512 16 29 2683 1227   
2 1428 1097 336 22 50 1036 99   
3 417 349 137 60 89 510 63   
4 193 146 55 16 44 249 869   
  
 Outstate Room.Board Books Personal PhD Terminal S.F.Ratio \  
0 7440 3300 450 2200 70 78 18.10   
1 12280 6450 750 1500 29 30 12.20   
2 11250 3750 400 1165 53 66 12.90   
3 12960 5450 450 875 92 97 7.70   
4 7560 4120 800 1500 76 72 11.90   
  
 perc.alumni Expend Grad.Rate   
0 12 7041 60   
1 16 10527 56   
2 30 8735 54   
3 37 19016 59   
4 2 10922 15

df\_num.hist(figsize=(16, 20), bins=50, xlabelsize=8, ylabelsize=8); # ; avoid having the matplotlib verbose informations



plt.figure(figsize=(16, 25))  
sns.boxplot(data=df\_num, orient="h", palette="Set2")

<AxesSubplot:>



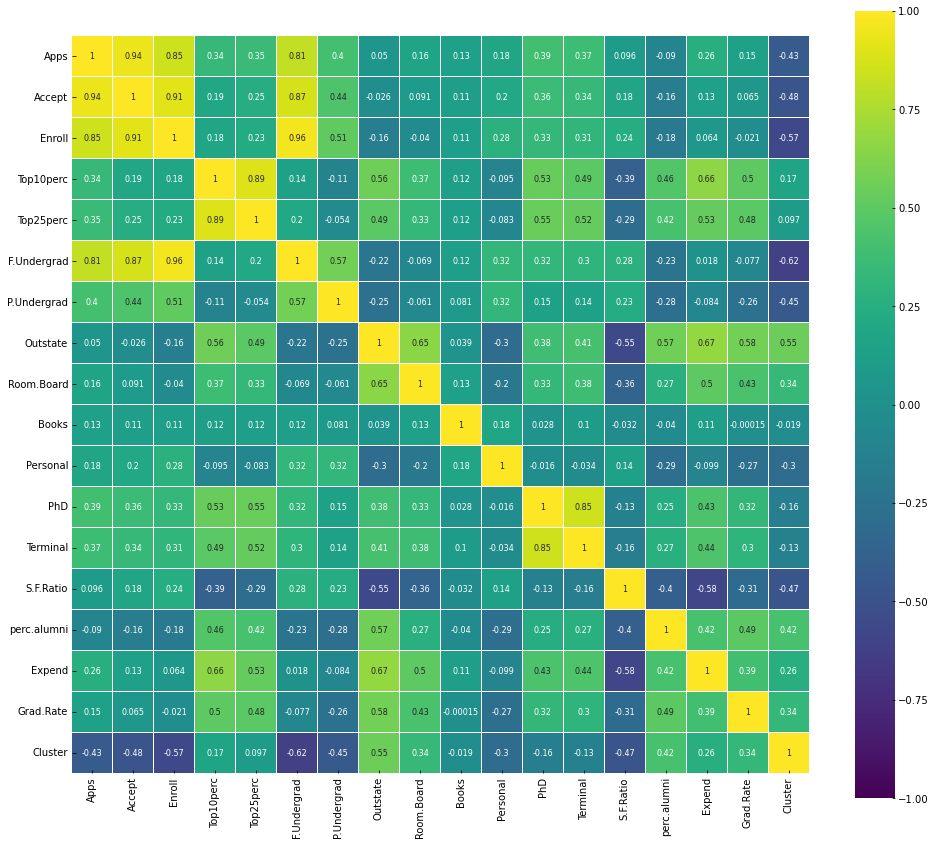
# Category encoding  
# dataset["Private\_code"] = dataset["Private"].astype('category')  
# dataset["Private\_code"] = dataset.Private\_code.cat.codes  
# dataset[["Private","Private\_code"]].drop\_duplicates().reset\_index(drop= True)

# adding a binary coded column for cross evaluation  
dataset['Cluster'] = dataset['Private'].apply(lambda x: 1 if x == 'Yes' else 0)

df\_num = dataset.select\_dtypes(include = ['float64', 'int64'])  
corr\_df = df\_num.drop('Cluster', axis = 1).corr()  
corr\_df

Apps Accept Enroll Top10perc Top25perc F.Undergrad \  
Apps 1.00 0.94 0.85 0.34 0.35 0.81   
Accept 0.94 1.00 0.91 0.19 0.25 0.87   
Enroll 0.85 0.91 1.00 0.18 0.23 0.96   
Top10perc 0.34 0.19 0.18 1.00 0.89 0.14   
Top25perc 0.35 0.25 0.23 0.89 1.00 0.20   
F.Undergrad 0.81 0.87 0.96 0.14 0.20 1.00   
P.Undergrad 0.40 0.44 0.51 -0.11 -0.05 0.57   
Outstate 0.05 -0.03 -0.16 0.56 0.49 -0.22   
Room.Board 0.16 0.09 -0.04 0.37 0.33 -0.07   
Books 0.13 0.11 0.11 0.12 0.12 0.12   
Personal 0.18 0.20 0.28 -0.10 -0.08 0.32   
PhD 0.39 0.36 0.33 0.53 0.55 0.32   
Terminal 0.37 0.34 0.31 0.49 0.52 0.30   
S.F.Ratio 0.10 0.18 0.24 -0.39 -0.29 0.28   
perc.alumni -0.09 -0.16 -0.18 0.46 0.42 -0.23   
Expend 0.26 0.13 0.06 0.66 0.53 0.02   
Grad.Rate 0.15 0.07 -0.02 0.50 0.48 -0.08   
  
 P.Undergrad Outstate Room.Board Books Personal PhD \  
Apps 0.40 0.05 0.16 0.13 0.18 0.39   
Accept 0.44 -0.03 0.09 0.11 0.20 0.36   
Enroll 0.51 -0.16 -0.04 0.11 0.28 0.33   
Top10perc -0.11 0.56 0.37 0.12 -0.10 0.53   
Top25perc -0.05 0.49 0.33 0.12 -0.08 0.55   
F.Undergrad 0.57 -0.22 -0.07 0.12 0.32 0.32   
P.Undergrad 1.00 -0.25 -0.06 0.08 0.32 0.15   
Outstate -0.25 1.00 0.65 0.04 -0.30 0.38   
Room.Board -0.06 0.65 1.00 0.13 -0.20 0.33   
Books 0.08 0.04 0.13 1.00 0.18 0.03   
Personal 0.32 -0.30 -0.20 0.18 1.00 -0.02   
PhD 0.15 0.38 0.33 0.03 -0.02 1.00   
Terminal 0.14 0.41 0.38 0.10 -0.03 0.85   
S.F.Ratio 0.23 -0.55 -0.36 -0.03 0.14 -0.13   
perc.alumni -0.28 0.57 0.27 -0.04 -0.29 0.25   
Expend -0.08 0.67 0.50 0.11 -0.10 0.43   
Grad.Rate -0.26 0.58 0.43 -0.00 -0.27 0.32   
  
 Terminal S.F.Ratio perc.alumni Expend Grad.Rate   
Apps 0.37 0.10 -0.09 0.26 0.15   
Accept 0.34 0.18 -0.16 0.13 0.07   
Enroll 0.31 0.24 -0.18 0.06 -0.02   
Top10perc 0.49 -0.39 0.46 0.66 0.50   
Top25perc 0.52 -0.29 0.42 0.53 0.48   
F.Undergrad 0.30 0.28 -0.23 0.02 -0.08   
P.Undergrad 0.14 0.23 -0.28 -0.08 -0.26   
Outstate 0.41 -0.55 0.57 0.67 0.58   
Room.Board 0.38 -0.36 0.27 0.50 0.43   
Books 0.10 -0.03 -0.04 0.11 -0.00   
Personal -0.03 0.14 -0.29 -0.10 -0.27   
PhD 0.85 -0.13 0.25 0.43 0.32   
Terminal 1.00 -0.16 0.27 0.44 0.30   
S.F.Ratio -0.16 1.00 -0.40 -0.58 -0.31   
perc.alumni 0.27 -0.40 1.00 0.42 0.49   
Expend 0.44 -0.58 0.42 1.00 0.39   
Grad.Rate 0.30 -0.31 0.49 0.39 1.00

# correlation heatmap  
corr = dataset.corr()  
plt.figure(figsize=(14, 12))  
  
sns.heatmap(corr  
# [(corr >= 0.5) | (corr <= -0.4)]  
 ,   
 cmap='viridis', vmax=1.0, vmin=-1.0, linewidths=0.1,  
 annot=True, annot\_kws={"size": 8}, square=True);  
plt.tight\_layout()  
plt.savefig('corr\_heatmap\_all.png', facecolor='w')



abs(corr\_df)

Apps Accept Enroll Top10perc Top25perc F.Undergrad \  
Apps 1.00 0.94 0.85 0.34 0.35 0.81   
Accept 0.94 1.00 0.91 0.19 0.25 0.87   
Enroll 0.85 0.91 1.00 0.18 0.23 0.96   
Top10perc 0.34 0.19 0.18 1.00 0.89 0.14   
Top25perc 0.35 0.25 0.23 0.89 1.00 0.20   
F.Undergrad 0.81 0.87 0.96 0.14 0.20 1.00   
P.Undergrad 0.40 0.44 0.51 0.11 0.05 0.57   
Outstate 0.05 0.03 0.16 0.56 0.49 0.22   
Room.Board 0.16 0.09 0.04 0.37 0.33 0.07   
Books 0.13 0.11 0.11 0.12 0.12 0.12   
Personal 0.18 0.20 0.28 0.10 0.08 0.32   
PhD 0.39 0.36 0.33 0.53 0.55 0.32   
Terminal 0.37 0.34 0.31 0.49 0.52 0.30   
S.F.Ratio 0.10 0.18 0.24 0.39 0.29 0.28   
perc.alumni 0.09 0.16 0.18 0.46 0.42 0.23   
Expend 0.26 0.13 0.06 0.66 0.53 0.02   
Grad.Rate 0.15 0.07 0.02 0.50 0.48 0.08   
  
 P.Undergrad Outstate Room.Board Books Personal PhD \  
Apps 0.40 0.05 0.16 0.13 0.18 0.39   
Accept 0.44 0.03 0.09 0.11 0.20 0.36   
Enroll 0.51 0.16 0.04 0.11 0.28 0.33   
Top10perc 0.11 0.56 0.37 0.12 0.10 0.53   
Top25perc 0.05 0.49 0.33 0.12 0.08 0.55   
F.Undergrad 0.57 0.22 0.07 0.12 0.32 0.32   
P.Undergrad 1.00 0.25 0.06 0.08 0.32 0.15   
Outstate 0.25 1.00 0.65 0.04 0.30 0.38   
Room.Board 0.06 0.65 1.00 0.13 0.20 0.33   
Books 0.08 0.04 0.13 1.00 0.18 0.03   
Personal 0.32 0.30 0.20 0.18 1.00 0.02   
PhD 0.15 0.38 0.33 0.03 0.02 1.00   
Terminal 0.14 0.41 0.38 0.10 0.03 0.85   
S.F.Ratio 0.23 0.55 0.36 0.03 0.14 0.13   
perc.alumni 0.28 0.57 0.27 0.04 0.29 0.25   
Expend 0.08 0.67 0.50 0.11 0.10 0.43   
Grad.Rate 0.26 0.58 0.43 0.00 0.27 0.32   
  
 Terminal S.F.Ratio perc.alumni Expend Grad.Rate   
Apps 0.37 0.10 0.09 0.26 0.15   
Accept 0.34 0.18 0.16 0.13 0.07   
Enroll 0.31 0.24 0.18 0.06 0.02   
Top10perc 0.49 0.39 0.46 0.66 0.50   
Top25perc 0.52 0.29 0.42 0.53 0.48   
F.Undergrad 0.30 0.28 0.23 0.02 0.08   
P.Undergrad 0.14 0.23 0.28 0.08 0.26   
Outstate 0.41 0.55 0.57 0.67 0.58   
Room.Board 0.38 0.36 0.27 0.50 0.43   
Books 0.10 0.03 0.04 0.11 0.00   
Personal 0.03 0.14 0.29 0.10 0.27   
PhD 0.85 0.13 0.25 0.43 0.32   
Terminal 1.00 0.16 0.27 0.44 0.30   
S.F.Ratio 0.16 1.00 0.40 0.58 0.31   
perc.alumni 0.27 0.40 1.00 0.42 0.49   
Expend 0.44 0.58 0.42 1.00 0.39   
Grad.Rate 0.30 0.31 0.49 0.39 1.00

abs(corr\_df).min(axis=1)

Apps 0.05  
Accept 0.03  
Enroll 0.02  
Top10perc 0.10  
Top25perc 0.05  
F.Undergrad 0.02  
P.Undergrad 0.05  
Outstate 0.03  
Room.Board 0.04  
Books 0.00  
Personal 0.02  
PhD 0.02  
Terminal 0.03  
S.F.Ratio 0.03  
perc.alumni 0.04  
Expend 0.02  
Grad.Rate 0.00  
dtype: float64

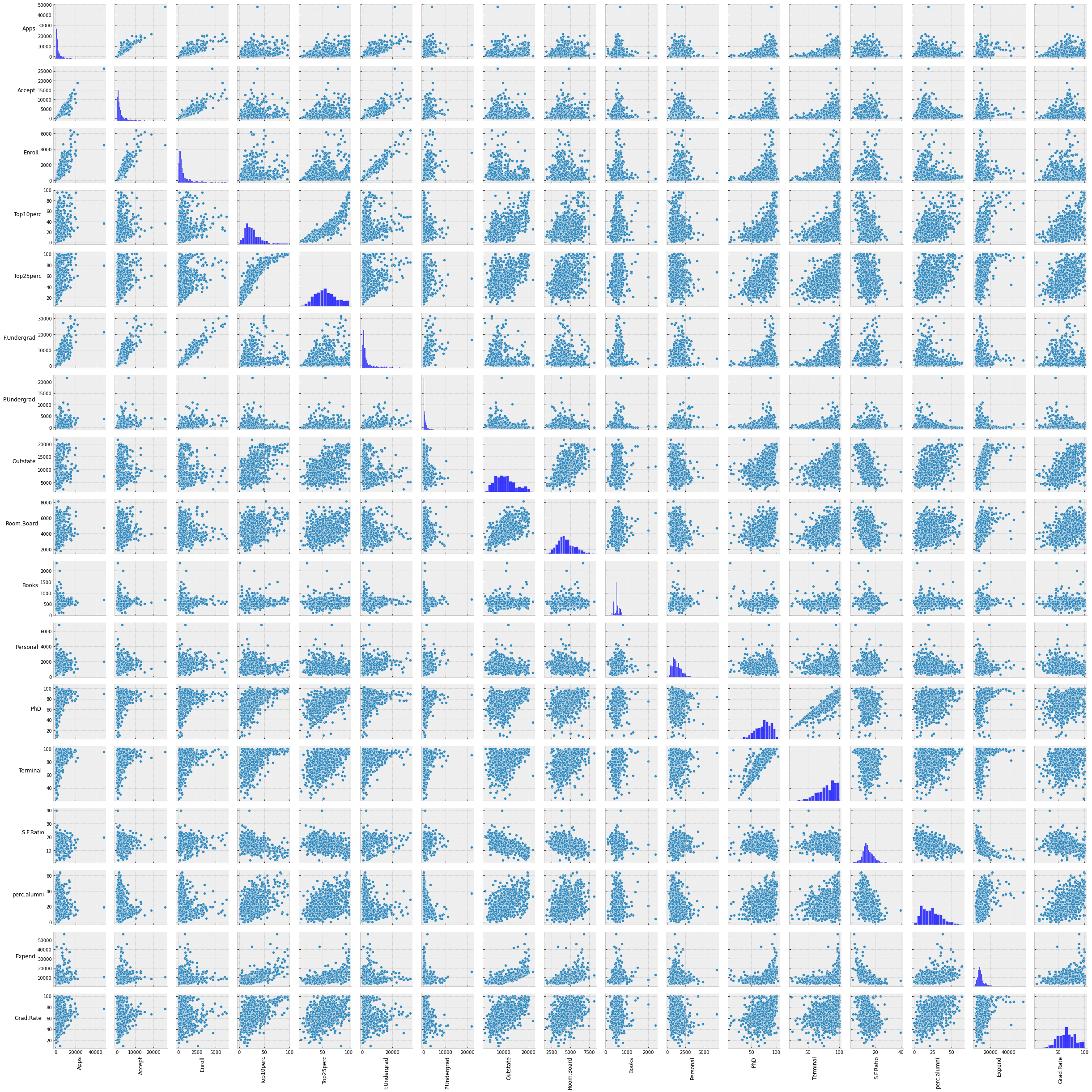
abs(corr\_df).min(axis=0)

Apps 0.05  
Accept 0.03  
Enroll 0.02  
Top10perc 0.10  
Top25perc 0.05  
F.Undergrad 0.02  
P.Undergrad 0.05  
Outstate 0.03  
Room.Board 0.04  
Books 0.00  
Personal 0.02  
PhD 0.02  
Terminal 0.03  
S.F.Ratio 0.03  
perc.alumni 0.04  
Expend 0.02  
Grad.Rate 0.00  
dtype: float64

abs(corr\_df).idxmin()

Apps Outstate  
Accept Outstate  
Enroll Grad.Rate  
Top10perc Personal  
Top25perc P.Undergrad  
F.Undergrad Expend  
P.Undergrad Top25perc  
Outstate Accept  
Room.Board Enroll  
Books Grad.Rate  
Personal PhD  
PhD Personal  
Terminal Personal  
S.F.Ratio Books  
perc.alumni Books  
Expend F.Undergrad  
Grad.Rate Books  
dtype: object

# Create the default pairplot using seaborn  
g =sns.pairplot(dataset.drop('Cluster', axis=1), height =2)  
for ax in g.axes.flat:  
 # rotate x axis labels  
 ax.set\_xlabel(ax.get\_xlabel(), rotation = 90)  
 # rotate y axis labels  
 ax.set\_ylabel(ax.get\_ylabel(), rotation = 0)  
 # set y labels alignment  
 ax.yaxis.get\_label().set\_horizontalalignment('right')  
plt.tight\_layout()  
plt.savefig('pairplot\_all.png', facecolor='w')  
plt.show()



# dataframe creation for separate modeling techniques  
dataset1 = dataset.copy()  
dataset2 = dataset.copy()  
dataset3 = dataset.copy()  
dataset4 = dataset.copy()

# Modeling using all the variables and elbow method

X = dataset.iloc[:, 2:19].values  
X

array([[1.6600e+03, 1.2320e+03, 7.2100e+02, ..., 1.2000e+01, 7.0410e+03,  
 6.0000e+01],  
 [2.1860e+03, 1.9240e+03, 5.1200e+02, ..., 1.6000e+01, 1.0527e+04,  
 5.6000e+01],  
 [1.4280e+03, 1.0970e+03, 3.3600e+02, ..., 3.0000e+01, 8.7350e+03,  
 5.4000e+01],  
 ...,  
 [2.0970e+03, 1.9150e+03, 6.9500e+02, ..., 2.0000e+01, 8.3230e+03,  
 4.9000e+01],  
 [1.0705e+04, 2.4530e+03, 1.3170e+03, ..., 4.9000e+01, 4.0386e+04,  
 9.9000e+01],  
 [2.9890e+03, 1.8550e+03, 6.9100e+02, ..., 2.8000e+01, 4.5090e+03,  
 9.9000e+01]])

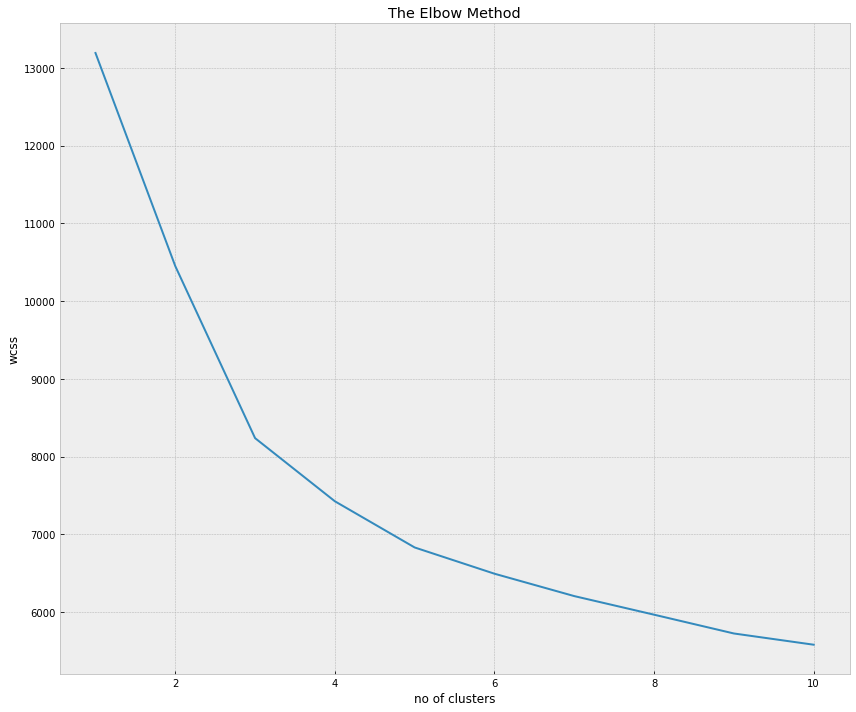
X\_scaler = StandardScaler().fit(X)  
X\_scaled = X\_scaler.transform(X)  
X\_scaled

array([[-3.46387711e-01, -3.20323986e-01, -6.38219163e-02, ...,  
 -8.67328505e-01, -5.02117275e-01, -3.16018711e-01],  
 [-2.10473291e-01, -3.79427213e-02, -2.88763103e-01, ...,  
 -5.44524227e-01, 1.65533536e-01, -5.50296733e-01],  
 [-4.06334756e-01, -3.75412817e-01, -4.78187261e-01, ...,  
 5.85290747e-01, -1.77676519e-01, -6.67435744e-01],  
 ...,  
 [-2.33470218e-01, -4.16153100e-02, -9.18050305e-02, ...,  
 -2.21719949e-01, -2.56584188e-01, -9.60283272e-01],  
 [ 1.99077186e+00, 1.77923882e-01, 5.77637163e-01, ...,  
 2.11861107e+00, 5.88423278e+00, 1.96819201e+00],  
 [-2.98416623e-03, -6.60992347e-02, -9.61101250e-02, ...,  
 4.23888608e-01, -9.87054697e-01, 1.96819201e+00]])

# Building the Model  
# KMeans Algorithm to decide the optimum cluster number , KMeans++ using Elbow Mmethod  
# to figure out K for KMeans, I will use ELBOW Method on KMEANS++ Calculation  
wcss = []  
  
# We always assume the max number of cluster would be 10  
# You can judge the number of clusters by doing averaging  
for i in range(1,11):  
 kmeans = KMeans(n\_clusters= i, init='k-means++', random\_state=0)  
 kmeans.fit(X\_scaled)  
 wcss.append(kmeans.inertia\_)  
  
 #inertia\_ is the formula used to segregate the data points into clusters

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\cluster\\_kmeans.py:881: 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=4.  
 warnings.warn(

# Visualizing the ELBOW method to get the optimal value of K   
plt.figure(figsize=(12, 10))  
plt.plot(range(1,11), wcss)  
plt.title('The Elbow Method')  
plt.xlabel('no of clusters')  
plt.ylabel('wcss')  
plt.tight\_layout()  
plt.savefig('elbow\_all.png', facecolor='w')  
plt.show()

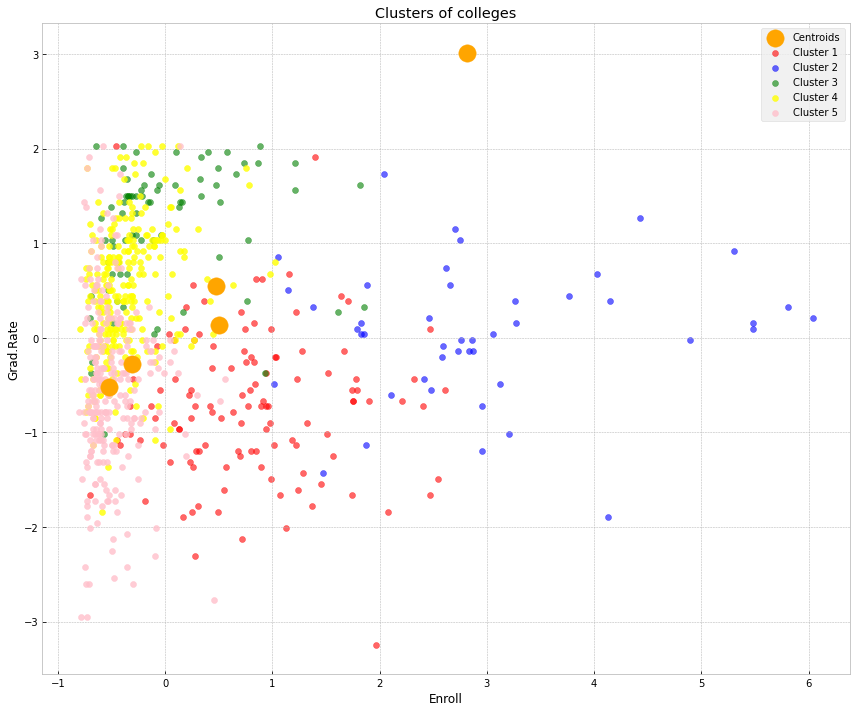


# unsupervised learning  
kmeansmodel\_unsup = KMeans(n\_clusters= 5, random\_state=0)  
y\_kmeans\_unsup= kmeansmodel\_unsup.fit\_predict(X\_scaled)

dataset1.iloc[:, 2:19].columns

Index(['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')

#Visualizing all the clusters   
plt.figure(figsize=(12, 10))  
ax = plt.axes()  
ax.set\_facecolor("white")  
plt.scatter(kmeansmodel\_unsup.cluster\_centers\_[:, 0], kmeansmodel\_unsup.cluster\_centers\_[:, 1], s = 300, c = 'orange', zorder = 5, \  
 label = 'Centroids', alpha = 1)  
plt.scatter(X\_scaled[y\_kmeans\_unsup == 0, 2], X\_scaled[y\_kmeans\_unsup == 0, 16] , c = 'red', label = 'Cluster 1', zorder = 0,\  
 alpha = .6)  
plt.scatter(X\_scaled[y\_kmeans\_unsup == 1, 2], X\_scaled[y\_kmeans\_unsup == 1, 16], c = 'blue', label = 'Cluster 2', zorder = 1,\  
 alpha = .6)  
plt.scatter(X\_scaled[y\_kmeans\_unsup == 2, 2], X\_scaled[y\_kmeans\_unsup == 2, 16], c = 'green', label = 'Cluster 3', zorder = 2,\  
 alpha = .6)  
plt.scatter(X\_scaled[y\_kmeans\_unsup == 3, 2], X\_scaled[y\_kmeans\_unsup == 3, 16], c = 'yellow', label = 'Cluster 4', zorder = 3,\  
 alpha = .8)  
plt.scatter(X\_scaled[y\_kmeans\_unsup == 4, 2], X\_scaled[y\_kmeans\_unsup == 4, 16], c = 'pink', label = 'Cluster 5', zorder = 4,\  
 alpha = .8)  
  
plt.title('Clusters of colleges')  
plt.xlabel('Enroll')  
plt.ylabel('Grad.Rate')  
plt.legend()  
plt.tight\_layout()  
plt.savefig('clusters\_all.png', facecolor='w')  
plt.show()



dataset1["Kmean\_cluster"] = y\_kmeans\_unsup  
cluster\_count\_df = dataset1["Kmean\_cluster"].value\_counts().to\_frame()  
cluster\_count\_df.reset\_index(inplace= True)  
cluster\_count\_df.rename(columns = {"index" : "Kmean\_cluster", "Kmean\_cluster" : "count"}, inplace = True)  
cluster\_count\_df.sort\_values("Kmean\_cluster", ignore\_index=True, inplace = True)  
cluster\_count\_df

Kmean\_cluster count  
0 0 120  
1 1 45  
2 2 78  
3 3 264  
4 4 269

# Getting counts  
dataset1.groupby(["Private", "Cluster", "Kmean\_cluster"], as\_index = False).agg(\  
 count=pd.NamedAgg(column="college\_name", aggfunc="count"))

Private Cluster Kmean\_cluster count  
0 No 0 0 111  
1 No 0 1 39  
2 No 0 2 2  
3 No 0 3 14  
4 No 0 4 46  
5 Yes 1 0 9  
6 Yes 1 1 6  
7 Yes 1 2 76  
8 Yes 1 3 250  
9 Yes 1 4 223

## Feature selection for the model - 2 features elbow method

X = dataset[['Enroll', 'Grad.Rate']].values  
X

array([[ 721, 60],  
 [ 512, 56],  
 [ 336, 54],  
 ...,  
 [ 695, 49],  
 [1317, 99],  
 [ 691, 99]], dtype=int64)

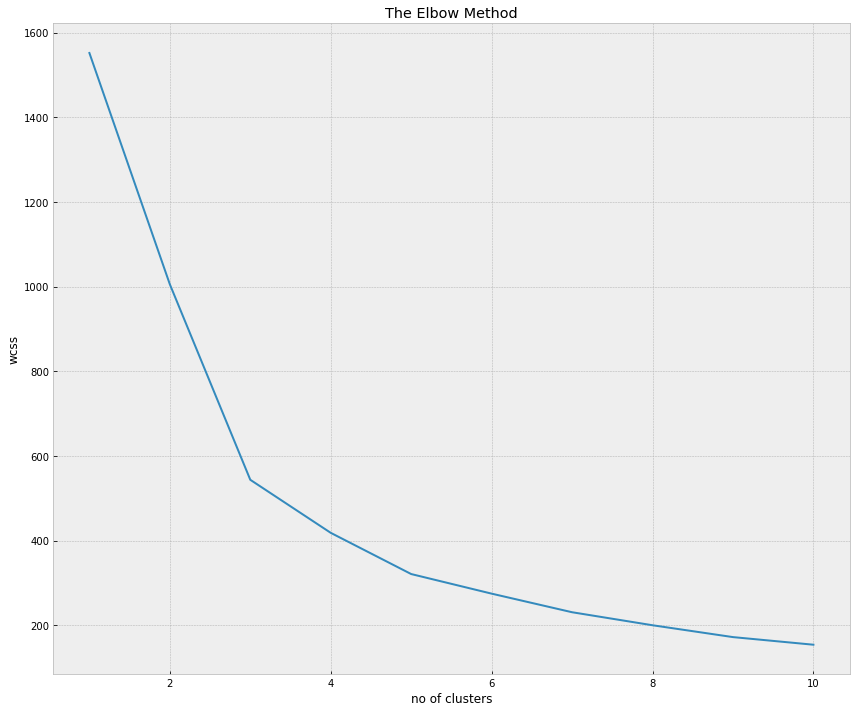
X\_scaler = StandardScaler().fit(X)  
X\_scaled = X\_scaler.transform(X)  
X\_scaled

array([[-0.06382192, -0.31601871],  
 [-0.2887631 , -0.55029673],  
 [-0.47818726, -0.66743574],  
 ...,  
 [-0.09180503, -0.96028327],  
 [ 0.57763716, 1.96819201],  
 [-0.09611012, 1.96819201]])

# Building the Model  
# KMeans Algorithm to decide the optimum cluster number , KMeans++ using Elbow Mmethod  
# to figure out K for KMeans, I will use ELBOW Method on KMEANS++ Calculation  
wcss = []  
  
# We always assume the max number of cluster would be 10  
# You can judge the number of clusters by doing averaging  
for i in range(1,11):  
 kmeans = KMeans(n\_clusters= i, init='k-means++', random\_state=0)  
 kmeans.fit(X\_scaled)  
 wcss.append(kmeans.inertia\_)  
  
 #inertia\_ is the formula used to segregate the data points into clusters

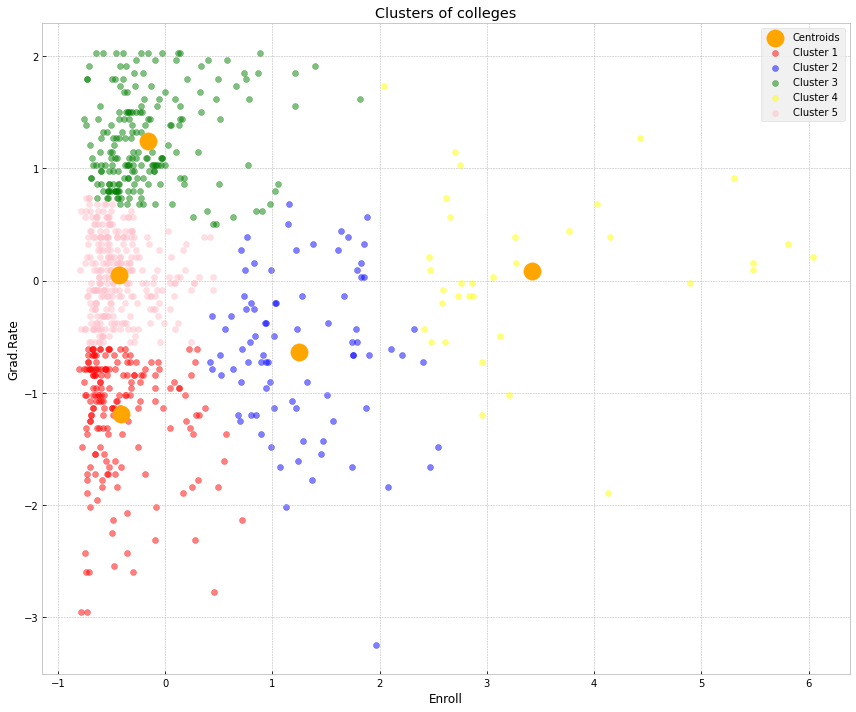
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\cluster\\_kmeans.py:881: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP\_NUM\_THREADS=4.  
 warnings.warn(

# Visualizing the ELBOW method to get the optimal value of K   
plt.figure(figsize=(12, 10))  
plt.plot(range(1,11), wcss)  
plt.title('The Elbow Method')  
plt.xlabel('no of clusters')  
plt.ylabel('wcss')  
plt.tight\_layout()  
plt.savefig('elbow\_two.png', facecolor='w')  
plt.show()



# unsupervised learning  
kmeansmodel\_unsup = KMeans(n\_clusters= 5, random\_state=0)  
y\_kmeans\_unsup= kmeansmodel\_unsup.fit\_predict(X\_scaled)

#Visualizing all the clusters   
plt.figure(figsize=(12, 10))  
ax = plt.axes()  
ax.set\_facecolor("white")  
plt.scatter(kmeansmodel\_unsup.cluster\_centers\_[:, 0], kmeansmodel\_unsup.cluster\_centers\_[:, 1], s = 300, c = 'orange', zorder = 5, \  
 label = 'Centroids', alpha = 1)  
plt.scatter(X\_scaled[y\_kmeans\_unsup == 0, 0], X\_scaled[y\_kmeans\_unsup == 0, 1] , c = 'red', label = 'Cluster 1', zorder = 0,\  
 alpha = .5)  
plt.scatter(X\_scaled[y\_kmeans\_unsup == 1, 0], X\_scaled[y\_kmeans\_unsup == 1, 1], c = 'blue', label = 'Cluster 2', zorder = 1,\  
 alpha = .5)  
plt.scatter(X\_scaled[y\_kmeans\_unsup == 2, 0], X\_scaled[y\_kmeans\_unsup == 2, 1], c = 'green', label = 'Cluster 3', zorder = 2,\  
 alpha = .5)  
plt.scatter(X\_scaled[y\_kmeans\_unsup == 3, 0], X\_scaled[y\_kmeans\_unsup == 3, 1], c = 'yellow', label = 'Cluster 4', zorder = 3,\  
 alpha = .5)  
plt.scatter(X\_scaled[y\_kmeans\_unsup == 4, 0], X\_scaled[y\_kmeans\_unsup == 4, 1], c = 'pink', label = 'Cluster 5', zorder = 4,\  
 alpha = .5)  
  
plt.title('Clusters of colleges')  
plt.xlabel('Enroll')  
plt.ylabel('Grad.Rate')  
plt.legend()  
plt.tight\_layout()  
plt.savefig('clusters\_2f\_4c.png', facecolor='w')  
plt.show()



# Getting counts  
dataset2["Kmean\_cluster"] = y\_kmeans\_unsup  
dataset2.groupby(["Private", "Cluster", "Kmean\_cluster"], as\_index = False).agg(\  
 count=pd.NamedAgg(column="college\_name", aggfunc="count"))

Private Cluster Kmean\_cluster count  
0 No 0 0 62  
1 No 0 1 74  
2 No 0 2 14  
3 No 0 3 32  
4 No 0 4 30  
5 Yes 1 0 117  
6 Yes 1 1 11  
7 Yes 1 2 188  
8 Yes 1 3 3  
9 Yes 1 4 245

cluster\_count\_df = dataset2["Kmean\_cluster"].value\_counts().to\_frame()  
cluster\_count\_df.reset\_index(inplace= True)  
cluster\_count\_df.rename(columns = {"index" : "Kmean\_cluster", "Kmean\_cluster" : "count"}, inplace = True)  
cluster\_count\_df.sort\_values("Kmean\_cluster", ignore\_index=True, inplace = True)  
cluster\_count\_df

Kmean\_cluster count  
0 0 179  
1 1 85  
2 2 202  
3 3 35  
4 4 275

# Modeling using all the variables and 2 clusters

X = dataset.iloc[:, 2:19].values  
X

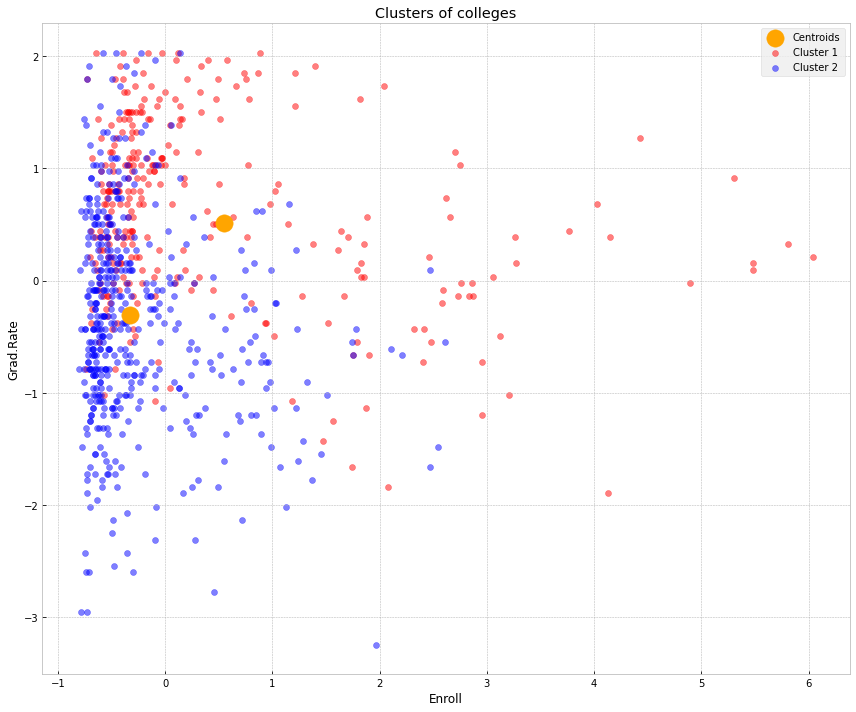
array([[1.6600e+03, 1.2320e+03, 7.2100e+02, ..., 1.2000e+01, 7.0410e+03,  
 6.0000e+01],  
 [2.1860e+03, 1.9240e+03, 5.1200e+02, ..., 1.6000e+01, 1.0527e+04,  
 5.6000e+01],  
 [1.4280e+03, 1.0970e+03, 3.3600e+02, ..., 3.0000e+01, 8.7350e+03,  
 5.4000e+01],  
 ...,  
 [2.0970e+03, 1.9150e+03, 6.9500e+02, ..., 2.0000e+01, 8.3230e+03,  
 4.9000e+01],  
 [1.0705e+04, 2.4530e+03, 1.3170e+03, ..., 4.9000e+01, 4.0386e+04,  
 9.9000e+01],  
 [2.9890e+03, 1.8550e+03, 6.9100e+02, ..., 2.8000e+01, 4.5090e+03,  
 9.9000e+01]])

X\_scaler = StandardScaler().fit(X)  
X\_scaled = X\_scaler.transform(X)  
X\_scaled

array([[-3.46387711e-01, -3.20323986e-01, -6.38219163e-02, ...,  
 -8.67328505e-01, -5.02117275e-01, -3.16018711e-01],  
 [-2.10473291e-01, -3.79427213e-02, -2.88763103e-01, ...,  
 -5.44524227e-01, 1.65533536e-01, -5.50296733e-01],  
 [-4.06334756e-01, -3.75412817e-01, -4.78187261e-01, ...,  
 5.85290747e-01, -1.77676519e-01, -6.67435744e-01],  
 ...,  
 [-2.33470218e-01, -4.16153100e-02, -9.18050305e-02, ...,  
 -2.21719949e-01, -2.56584188e-01, -9.60283272e-01],  
 [ 1.99077186e+00, 1.77923882e-01, 5.77637163e-01, ...,  
 2.11861107e+00, 5.88423278e+00, 1.96819201e+00],  
 [-2.98416623e-03, -6.60992347e-02, -9.61101250e-02, ...,  
 4.23888608e-01, -9.87054697e-01, 1.96819201e+00]])

# unsupervised learning  
kmeansmodel\_unsup = KMeans(n\_clusters= 2, random\_state=31)  
y\_kmeans\_unsup= kmeansmodel\_unsup.fit\_predict(X\_scaled)

#Visualizing all the clusters   
plt.figure(figsize=(12, 10))  
ax = plt.axes()  
ax.set\_facecolor("white")  
plt.scatter(kmeansmodel\_unsup.cluster\_centers\_[:, 0], kmeansmodel\_unsup.cluster\_centers\_[:, 1], s = 300, c = 'orange', zorder = 4, \  
 label = 'Centroids', alpha = 1)  
plt.scatter(X\_scaled[y\_kmeans\_unsup == 0, 2], X\_scaled[y\_kmeans\_unsup == 0, 16] , c = 'red', label = 'Cluster 1', zorder = 0,\  
 alpha = .5)  
plt.scatter(X\_scaled[y\_kmeans\_unsup == 1, 2], X\_scaled[y\_kmeans\_unsup == 1, 16], c = 'blue', label = 'Cluster 2', zorder = 1,\  
 alpha = .5)  
plt.title('Clusters of colleges')  
plt.xlabel('Enroll')  
plt.ylabel('Grad.Rate')  
plt.legend()  
plt.tight\_layout()  
plt.savefig('clusters\_all\_2c.png', facecolor='w')  
plt.show()



## Model Evaluation for all features with 2 clusters

# Getting counts  
dataset3["Kmean\_cluster"] = y\_kmeans\_unsup  
dataset3.groupby(["Private", "Cluster", "Kmean\_cluster"], as\_index = False).agg(\  
 count=pd.NamedAgg(column="college\_name", aggfunc="count"))

Private Cluster Kmean\_cluster count  
0 No 0 0 66  
1 No 0 1 146  
2 Yes 1 0 225  
3 Yes 1 1 339

cluster\_count\_df = dataset3["Kmean\_cluster"].value\_counts().to\_frame()  
cluster\_count\_df.reset\_index(inplace= True)  
cluster\_count\_df.rename(columns = {"index" : "Kmean\_cluster", "Kmean\_cluster" : "count"}, inplace = True)  
cluster\_count\_df.sort\_values("Kmean\_cluster", ignore\_index=True, inplace = True)  
cluster\_count\_df

Kmean\_cluster count  
0 0 291  
1 1 485

print(confusion\_matrix(dataset3['Cluster'], y\_kmeans\_unsup))

[[ 66 146]  
 [225 339]]

print(classification\_report(dataset3['Cluster'], y\_kmeans\_unsup))

precision recall f1-score support  
  
 0 0.23 0.31 0.26 212  
 1 0.70 0.60 0.65 564  
  
 accuracy 0.52 776  
 macro avg 0.46 0.46 0.45 776  
weighted avg 0.57 0.52 0.54 776

## Feature selection for the model - 2 features 2 clusters

X = dataset[['Enroll', 'Grad.Rate']].values  
X

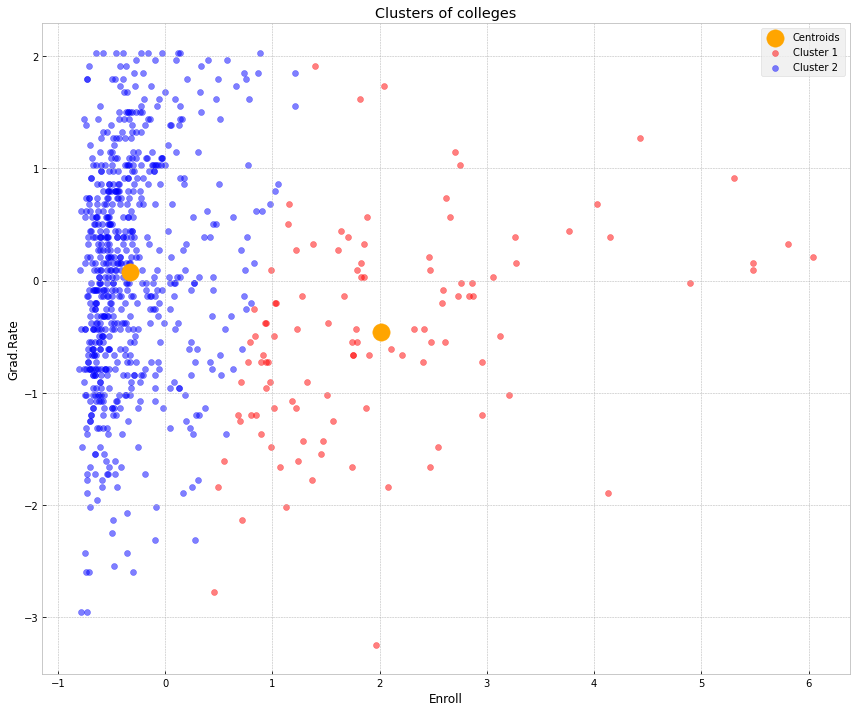
array([[ 721, 60],  
 [ 512, 56],  
 [ 336, 54],  
 ...,  
 [ 695, 49],  
 [1317, 99],  
 [ 691, 99]], dtype=int64)

X\_scaler = StandardScaler().fit(X)  
X\_scaled = X\_scaler.transform(X)  
X\_scaled

array([[-0.06382192, -0.31601871],  
 [-0.2887631 , -0.55029673],  
 [-0.47818726, -0.66743574],  
 ...,  
 [-0.09180503, -0.96028327],  
 [ 0.57763716, 1.96819201],  
 [-0.09611012, 1.96819201]])

# unsupervised learning  
kmeansmodel\_unsup = KMeans(n\_clusters= 2, random\_state=31)  
y\_kmeans\_unsup= kmeansmodel\_unsup.fit\_predict(X\_scaled)

#Visualizing all the clusters   
plt.figure(figsize=(12, 10))  
ax = plt.axes()  
ax.set\_facecolor("white")  
plt.scatter(kmeansmodel\_unsup.cluster\_centers\_[:, 0], kmeansmodel\_unsup.cluster\_centers\_[:, 1], s = 300, c = 'orange', zorder = 4, \  
 label = 'Centroids', alpha = 1)  
plt.scatter(X\_scaled[y\_kmeans\_unsup == 0, 0], X\_scaled[y\_kmeans\_unsup == 0, 1] , c = 'red', label = 'Cluster 1', zorder = 0,\  
 alpha = .5)  
plt.scatter(X\_scaled[y\_kmeans\_unsup == 1, 0], X\_scaled[y\_kmeans\_unsup == 1, 1], c = 'blue', label = 'Cluster 2', zorder = 1,\  
 alpha = .5)  
plt.title('Clusters of colleges')  
plt.xlabel('Enroll')  
plt.ylabel('Grad.Rate')  
plt.legend()  
plt.tight\_layout()  
plt.savefig('clusters\_2f\_2c.png', facecolor='w')  
plt.show()



## Model Evaluation for 2 features with 2 clusters

# Getting counts  
dataset4["Kmean\_cluster"] = y\_kmeans\_unsup  
dataset4.groupby(["Private", "Cluster", "Kmean\_cluster"], as\_index = False).agg(\  
 count=pd.NamedAgg(column="college\_name", aggfunc="count"))

Private Cluster Kmean\_cluster count  
0 No 0 0 98  
1 No 0 1 114  
2 Yes 1 0 12  
3 Yes 1 1 552

cluster\_count\_df = dataset4["Kmean\_cluster"].value\_counts().to\_frame()  
cluster\_count\_df.reset\_index(inplace= True)  
cluster\_count\_df.rename(columns = {"index" : "Kmean\_cluster", "Kmean\_cluster" : "count"}, inplace = True)  
cluster\_count\_df.sort\_values("Kmean\_cluster", ignore\_index=True, inplace = True)  
cluster\_count\_df

Kmean\_cluster count  
0 0 110  
1 1 666

print(confusion\_matrix(dataset4['Cluster'], y\_kmeans\_unsup))

[[ 98 114]  
 [ 12 552]]

print(classification\_report(dataset4['Cluster'], y\_kmeans\_unsup))

precision recall f1-score support  
  
 0 0.89 0.46 0.61 212  
 1 0.83 0.98 0.90 564  
  
 accuracy 0.84 776  
 macro avg 0.86 0.72 0.75 776  
weighted avg 0.85 0.84 0.82 776

import pandas\_profiling as pp  
# Perform data profiling  
profile\_report = pp.ProfileReport(dataset)  
profile\_report.to\_file('college\_eda.html')  
profile\_report

{"model\_id":"a7f71332a91248468c5c951051c04942","version\_major":2,"version\_minor":0}

{"model\_id":"935825c7403e40c0a7d51db4d71112cb","version\_major":2,"version\_minor":0}

{"model\_id":"c0122ead00854aee98d2784bd5e8b4fe","version\_major":2,"version\_minor":0}

{"model\_id":"10ff0f48916a4838b2532a10fe0469ef","version\_major":2,"version\_minor":0}

<IPython.core.display.HTML object>