02-K Means Clustering Project

July 16, 2021

___ # K Means Clustering Project

For this project we will attempt to use KMeans Clustering to cluster Universities into to two groups, Private and Public.

It is **very important to note, we actually have the labels for this data set, but we will NOT use them for the KMeans clustering algorithm, since that is an unsupervised learning algorithm.**

When using the Kmeans algorithm under normal circumstances, it is because you don't have labels. In this case we will use the labels to try to get an idea of how well the algorithm performed, but you won't usually do this for Kmeans, so the classification report and confusion matrix at the end of this project, don't truly make sense in a real world setting!. ___

0.1 The Data

We will use a data frame with 777 observations on the following 18 variables. * Private A factor with levels No and Yes indicating private or public university * Apps Number of applications received * Accept Number of applications accepted * Enroll Number of new students enrolled * Top10perc Pct. new students from top 10% of H.S. class * Top25perc Pct. new students from top 25% of H.S. class * F.Undergrad Number of fulltime undergraduates * P.Undergrad Number of parttime undergraduates * Outstate Out-of-state tuition * Room.Board Room and board costs * Books Estimated book costs * Personal Estimated personal spending * PhD Pct. of faculty with Ph.D.'s * Terminal Pct. of faculty with terminal degree * S.F.Ratio Student/faculty ratio * perc.alumni Pct. alumni who donate * Expend Instructional expenditure per student * Grad.Rate Graduation rate

0.2 Import Libraries

** Import the libraries you usually use for data analysis.**

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

0.3 Get the Data

** Read in the College_Data file using read_csv. Figure out how to set the first column as the index.**

[2]:
```
cllg = pd.read_csv('College_Data')
```

Check the head of the data

[3]:
```
cllg.head()
```

[3]:

	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

[ ]:
```
```

** Check the info() and describe() methods on the data.**

[4]:
```
cllg.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
```

[ ]:

[5]:
```
cllg.describe()
```

[5]:

	Apps	Accept	Enroll	Top10perc	Top25perc
count	777.000000	777.000000	777.000000	777.000000	777.000000
mean	3001.638353	2018.804376	779.972973	27.558559	55.796654
std	3870.201484	2451.113971	929.176190	17.640364	19.804778
min	81.000000	72.000000	35.000000	1.000000	9.000000
25%	776.000000	604.000000	242.000000	15.000000	41.000000
50%	1558.000000	1110.000000	434.000000	23.000000	54.000000
75%	3624.000000	2424.000000	902.000000	35.000000	69.000000
max	48094.000000	26330.000000	6392.000000	96.000000	100.000000

	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books
count	777.000000	777.000000	777.000000	777.000000	777.000000
mean	3699.907336	855.298584	10440.669241	4357.526384	549.380952
std	4850.420531	1522.431887	4023.016484	1096.696416	165.105360
min	139.000000	1.000000	2340.000000	1780.000000	96.000000
25%	992.000000	95.000000	7320.000000	3597.000000	470.000000
50%	1707.000000	353.000000	9990.000000	4200.000000	500.000000
75%	4005.000000	967.000000	12925.000000	5050.000000	600.000000
max	31643.000000	21836.000000	21700.000000	8124.000000	2340.000000

	Personal	PhD	Terminal	S.F.Ratio	perc.alumni
count	777.000000	777.000000	777.000000	777.000000	777.000000
mean	1340.642214	72.660232	79.702703	14.089704	22.743887
std	677.071454	16.328155	14.722359	3.958349	12.391801
min	250.000000	8.000000	24.000000	2.500000	0.000000
25%	850.000000	62.000000	71.000000	11.500000	13.000000
50%	1200.000000	75.000000	82.000000	13.600000	21.000000

```
75%     1700.000000   85.000000   92.000000   16.500000    31.000000
max     6800.000000  103.000000  100.000000   39.800000    64.000000
```

```
              Expend  Grad.Rate
count     777.000000  777.00000
mean     9660.171171   65.46332
std      5221.768440   17.17771
min      3186.000000   10.00000
25%      6751.000000   53.00000
50%      8377.000000   65.00000
75%     10830.000000   78.00000
max     56233.000000  118.00000
```

[ ]:

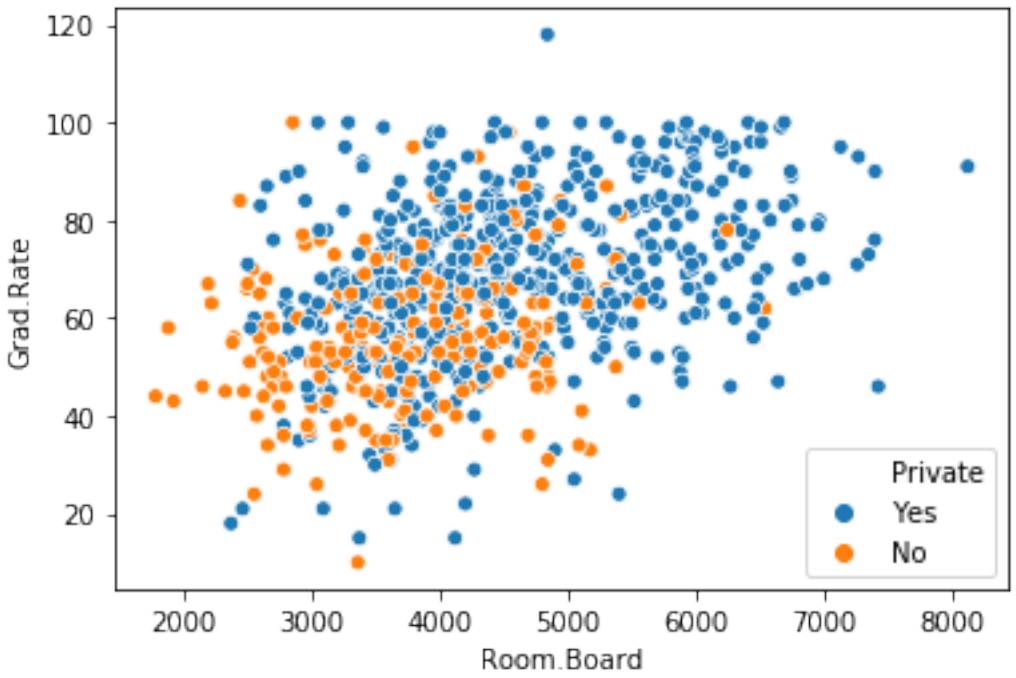
0.4 EDA

It's time to create some data visualizations!

** Create a scatterplot of Grad.Rate versus Room.Board where the points are colored by the Private column. **

[6]:
```
sns.scatterplot(x=cllg['Room.Board'],y=cllg['Grad.Rate'],hue=cllg['Private'])
```

[6]: <matplotlib.axes._subplots.AxesSubplot at 0x16a2f2b1348>

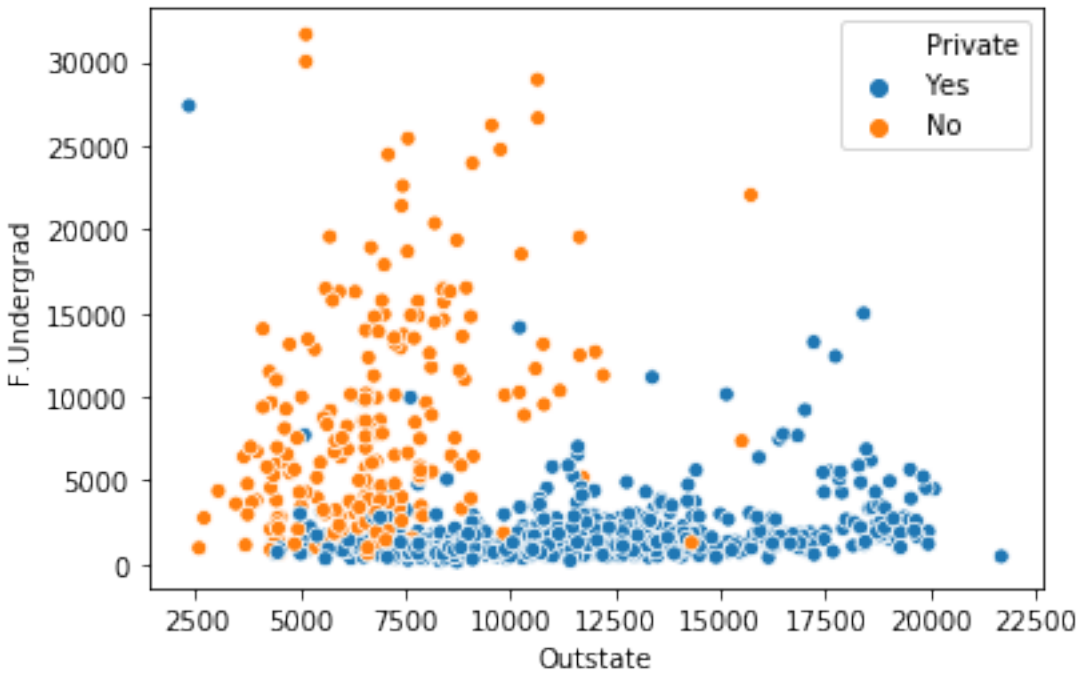


[ ]:

Create a scatterplot of F.Undergrad versus Outstate where the points are colored by the Private column.

[7]:
```
sns.scatterplot(x=cllg['Outstate'],y=cllg['F.Undergrad'],hue=cllg['Private'])
```

[7]: <matplotlib.axes._subplots.AxesSubplot at 0x16a2fa4eb08>

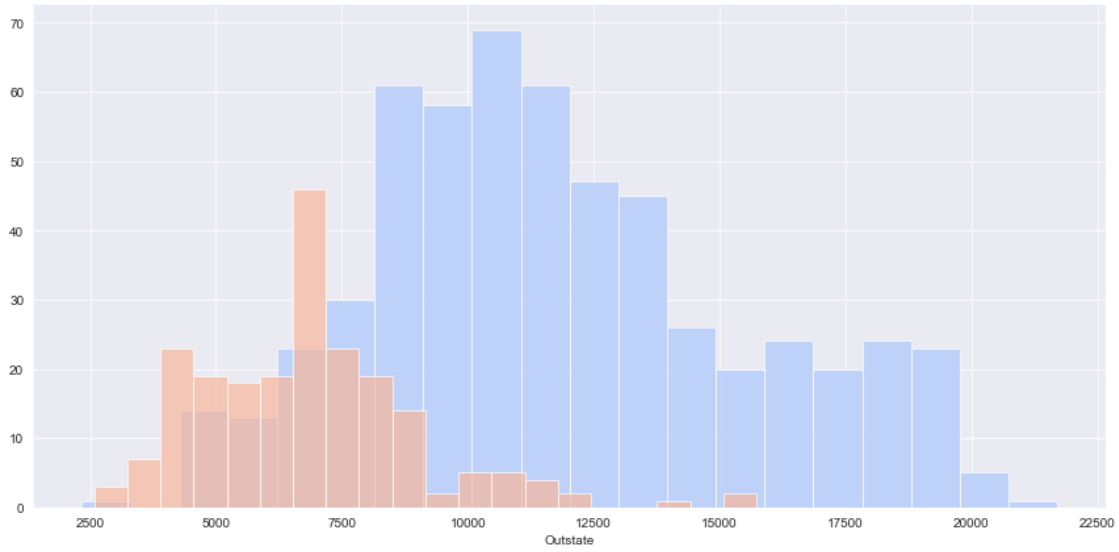


[ ]:

** Create a stacked histogram showing Out of State Tuition based on the Private column. Try doing this using sns.FacetGrid. If that is too tricky, see if you can do it just by using two instances of pandas.plot(kind='hist'). **

[8]:
```
sns.set_style('darkgrid')
g = sns.FacetGrid(cllg,hue="Private",palette='coolwarm',height=6,aspect=2)
g.map(plt.hist,'Outstate',bins=20,alpha=0.7)
```

[8]: <seaborn.axisgrid.FacetGrid at 0x16a2fad9488>

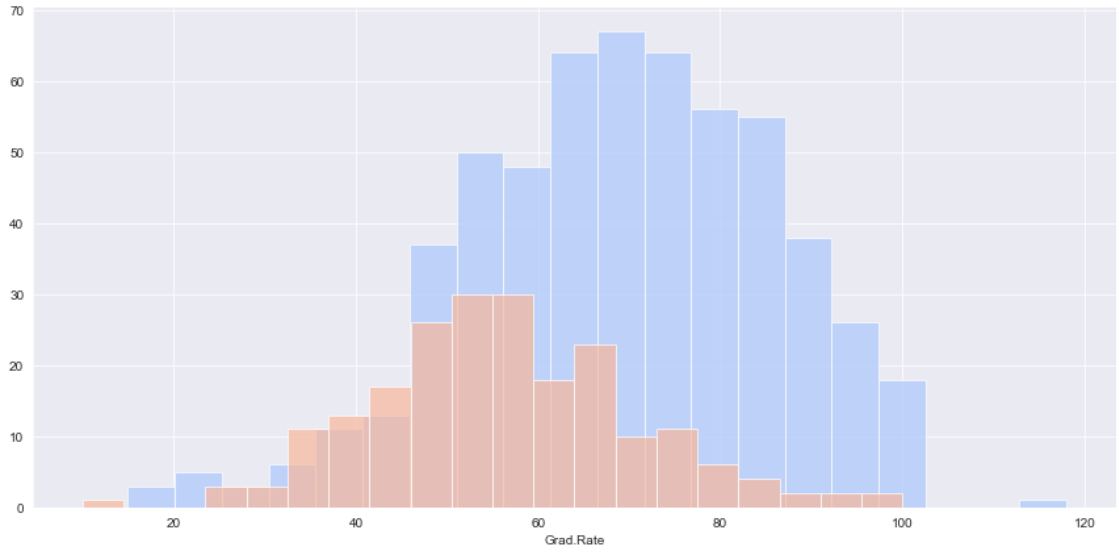


[ ]:

Create a similar histogram for the Grad.Rate column.

[9]:
```
sns.set_style('darkgrid')
g = sns.FacetGrid(cllg,hue="Private",palette='coolwarm',height=6,aspect=2)
g.map(plt.hist,'Grad.Rate',bins=20,alpha=0.7)
```

[9]: <seaborn.axisgrid.FacetGrid at 0x16a2fd5f888>



** Notice how there seems to be a private school with a graduation rate of higher than 100%.What

is the name of that school?**

[10]:
```
cllg[cllg['Grad.Rate']>100]
```

[10]:
```
            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
```

[ ]:
```

```

** Set that school's graduation rate to 100 so it makes sense. You may get a warning not an error) when doing this operation, so use dataframe operations or just re-do the histogram visualization to make sure it actually went through.**

[11]:
```
cllg['Grad.Rate'][95]=100
```

```
C:\Users\krish\anaconda3\lib\site-packages\ipykernel_launcher.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
  """Entry point for launching an IPython kernel.
```

[12]:
```
cllg[cllg['Grad.Rate']>100]
```

[12]:
```
Empty DataFrame
Columns: [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]
Index: []
```

[ ]:
```

```

0.5 K Means Cluster Creation

Now it is time to create the Cluster labels!

** Import KMeans from SciKit Learn.**

[13]:
```
from sklearn.cluster import KMeans
```

** Create an instance of a K Means model with 2 clusters.**

[14]:
```
kmeans = KMeans(n_clusters=2)
```

[15]:
```
cllg.drop(['Unnamed: 0'],axis=1,inplace=True)
```

[ ]:
```

```

Fit the model to all the data except for the Private label.

[16]:
```
kmeans.fit(cllg.drop('Private',axis=1))
```

[16]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

[ ]:
```

```

** What are the cluster center vectors?**

[17]:
```
kmeans.cluster_centers_
```

[17]:
```
array([[1.81323468e+03, 1.28716592e+03, 4.91044843e+02, 2.53094170e+01,
        5.34708520e+01, 2.18854858e+03, 5.95458894e+02, 1.03957085e+04,
        4.31136472e+03, 5.41982063e+02, 1.28033632e+03, 7.04424514e+01,
        7.78251121e+01, 1.40997010e+01, 2.31748879e+01, 8.93204634e+03,
        6.50926756e+01],
       [1.03631389e+04, 6.55089815e+03, 2.56972222e+03, 4.14907407e+01,
        7.02037037e+01, 1.30619352e+04, 2.46486111e+03, 1.07191759e+04,
        4.64347222e+03, 5.95212963e+02, 1.71420370e+03, 8.63981481e+01,
        9.13333333e+01, 1.40277778e+01, 2.00740741e+01, 1.41705000e+04,
        6.75925926e+01]])
```

[ ]:
```

```

0.6 Evaluation

There is no perfect way to evaluate clustering if you don't have the labels, however since this is just an exercise, we do have the labels, so we take advantage of this to evaluate our clusters, keep in mind, you usually won't have this luxury in the real world.

** Create a new column for df called 'Cluster', which is a 1 for a Private school, and a 0 for a public school.**

[18]:
```
def Cluster(x):
    if x=='Yes':
        return(1)
    else:
        return(0)
```

[19]:
```
type1 = cllg['Private']
```

[20]:
```
type2 = pd.get_dummies(type1,drop_first=True)
```

[21]:
```
kmeans.labels_
```

[21]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0])
```

[22]:
```
type3 = cllg['Private'].apply(Cluster)
```

[23]:
```
cllg['Private'] = type3
```

[24]:
```
cllg.head()
```

[24]:

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	\
0	1	1660	1232	721	23	52	2885	
1	1	2186	1924	512	16	29	2683	
2	1	1428	1097	336	22	50	1036	
3	1	417	349	137	60	89	510	
4	1	193	146	55	16	44	249	

	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD	Terminal	\
0	537	7440	3300	450	2200	70	78	
1	1227	12280	6450	750	1500	29	30	
2	99	11250	3750	400	1165	53	66	
3	63	12960	5450	450	875	92	97	
4	869	7560	4120	800	1500	76	72	

	S.F.Ratio	perc.alumni	Expend	Grad.Rate
0	18.1	12	7041	60
1	12.2	16	10527	56
2	12.9	30	8735	54
3	7.7	37	19016	59
4	11.9	2	10922	15

** Create a confusion matrix and classification report to see how well the Kmeans clustering worked without being given any labels.**

[25]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

[26]:
```
print(classification_report(cllg['Private'],kmeans.labels_))
```

```
              precision    recall  f1-score   support

           0       0.21      0.65      0.31       212
           1       0.31      0.06      0.10       565

    accuracy                           0.22       777
   macro avg       0.26      0.36      0.21       777
weighted avg       0.29      0.22      0.16       777
```

[27]:
```
print(confusion_matrix(cllg['Private'],kmeans.labels_))
```

```
[[138  74]
 [531  34]]
```

[ ]:
```

```

Not so bad considering the algorithm is purely using the features to cluster the universities into 2

distinct groups! Hopefully you can begin to see how K Means is useful for clustering un-labeled data!

0.7 Great Job!