K Means Clustering Project

Dishant Fapot

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!.

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

Import Libraries

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

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Get the Data

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

In [7]:

```
df = pd.read_csv('College_Data',index_col=0)
```

Check the head of the data

In [8]:

```
df.head()
```

Out[8]:

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outsta
Abilene Christian University	Yes	1660	1232	721	23	52	2885	537	74
Adelphi University	Yes	2186	1924	512	16	29	2683	1227	122
Adrian College	Yes	1428	1097	336	22	50	1036	99	112
Agnes Scott College	Yes	417	349	137	60	89	510	63	129
Alaska Pacific University	Yes	193	146	55	16	44	249	869	75

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

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 777 entries, Abilene Christian University to York College of Pennsylvani
a
Data columns (total 18 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Private      777 non-null    object
 1   Apps         777 non-null    int64
 2   Accept       777 non-null    int64
 3   Enroll       777 non-null    int64
 4   Top10perc    777 non-null    int64
 5   Top25perc    777 non-null    int64
 6   F.Undergrad  777 non-null    int64
 7   P.Undergrad  777 non-null    int64
 8   Outstate     777 non-null    int64
 9   Room.Board   777 non-null    int64
 10  Books        777 non-null    int64
 11  Personal     777 non-null    int64
 12  PhD          777 non-null    int64
 13  Terminal     777 non-null    int64
 14  S.F.Ratio    777 non-null    float64
 15  perc.alumni  777 non-null    int64
 16  Expend       777 non-null    int64
 17  Grad.Rate    777 non-null    int64
dtypes: float64(1), int64(16), object(1)
memory usage: 115.3+ KB
```

In [13]:
```
df.describe()
```

Out[13]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergra
count	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.0000
mean	3001.638353	2018.804376	779.972973	27.558559	55.796654	3699.907336	855.2985
std	3870.201484	2451.113971	929.176190	17.640364	19.804778	4850.420531	1522.4318
min	81.000000	72.000000	35.000000	1.000000	9.000000	139.000000	1.0000
25%	776.000000	604.000000	242.000000	15.000000	41.000000	992.000000	95.0000
50%	1558.000000	1110.000000	434.000000	23.000000	54.000000	1707.000000	353.0000
75%	3624.000000	2424.000000	902.000000	35.000000	69.000000	4005.000000	967.0000
max	48094.000000	26330.000000	6392.000000	96.000000	100.000000	31643.000000	21836.0000

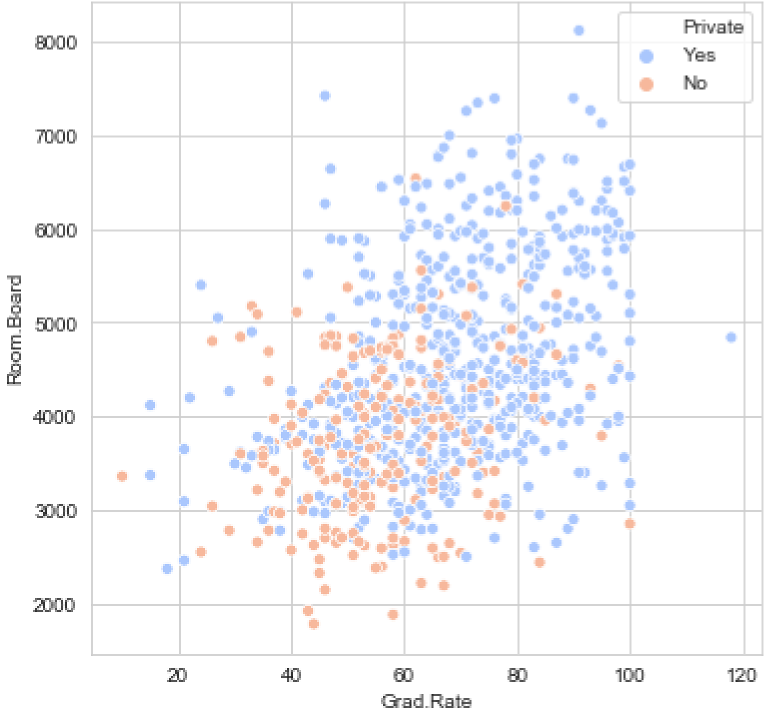
EDA

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. **

```
In [22]: plt.figure(figsize=[6,6])
         sns.set_style('whitegrid')
         sns.scatterplot(x='Grad.Rate',y='Room.Board',hue='Private',data=df,palette='coolw
```

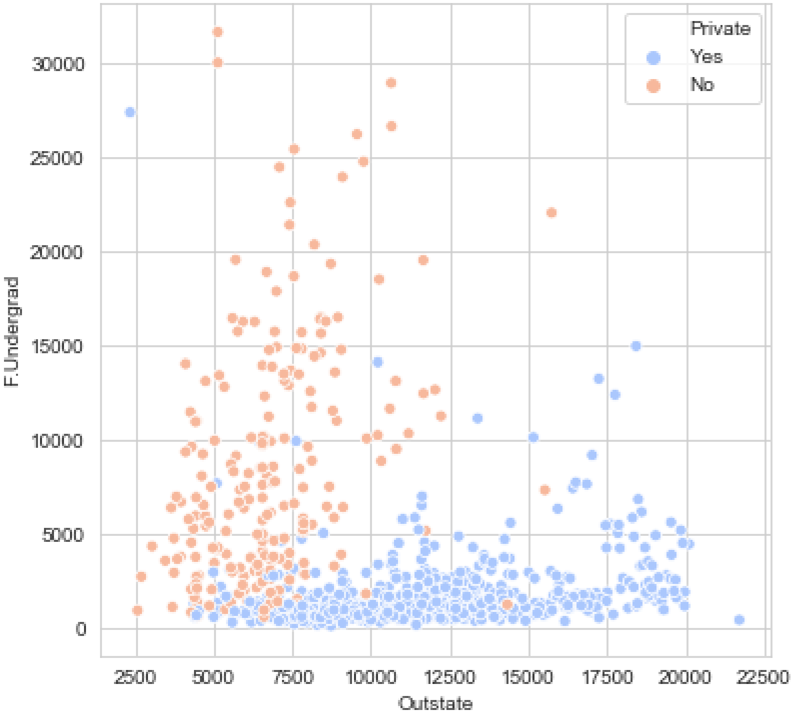
```
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x1e2e9a61ac8>
```



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

In [24]:
```
plt.figure(figsize=[6,6])
sns.set_style('whitegrid')
sns.scatterplot(x='Outstate',y='F.Undergrad',hue='Private',data=df,palette='coolw
```

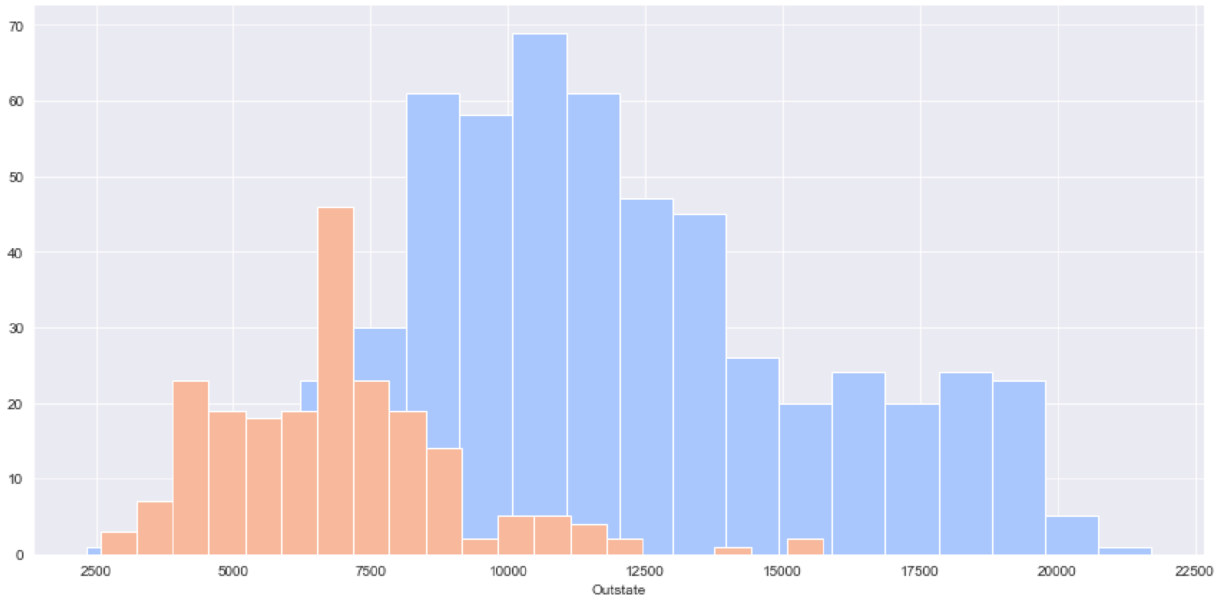
Out[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e2ea0665c8>
```



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

In [30]:

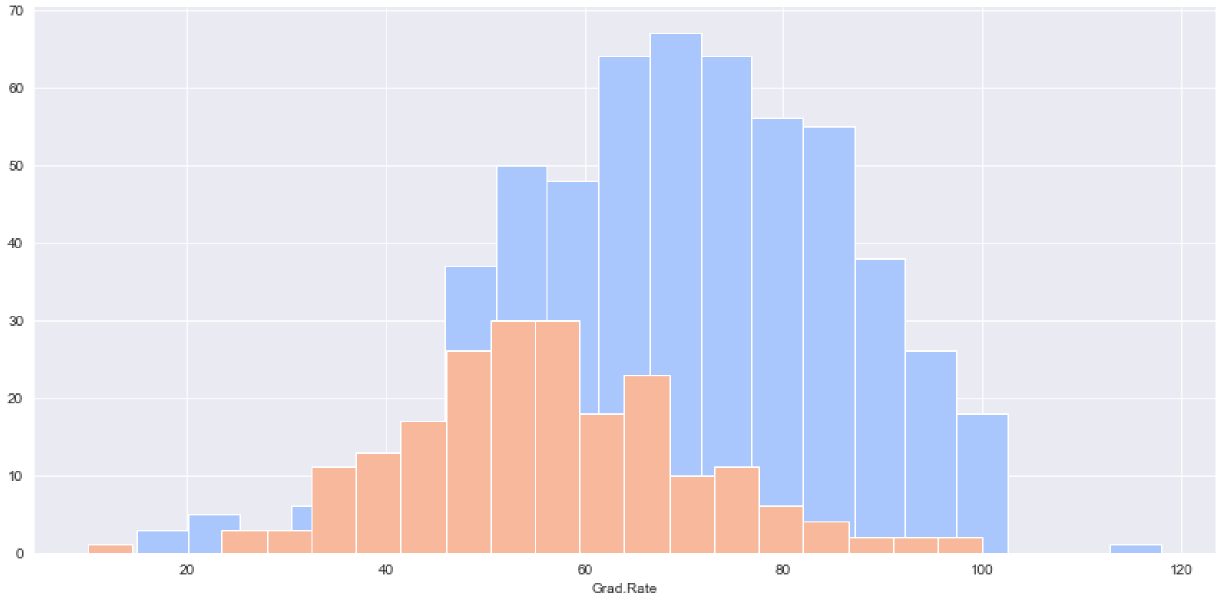
```
sns.set_style('darkgrid')
g=sns.FacetGrid(df,hue="Private",palette='coolwarm',height=6,aspect=2)
g=g.map(plt.hist,'Outstate',bins=20)
```



Create a similar histogram for the Grad.Rate column.

In [44]:

```
sns.set_style('darkgrid')
g=sns.FacetGrid(df,hue="Private",palette='coolwarm',height=6,aspect=2)
g=g.map(plt.hist,'Grad.Rate',bins=20)
```



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

In [49]:
```python
df[df['Grad.Rate']>100]
```

Out[49]:

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outsta
Cazenovia College	Yes	3847	3433	527	9	35	1010	12	93

** 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.**

In [54]:
```python
df['Grad.Rate']['Cazenovia College'] = 100
df[df['Grad.Rate']>100]
```

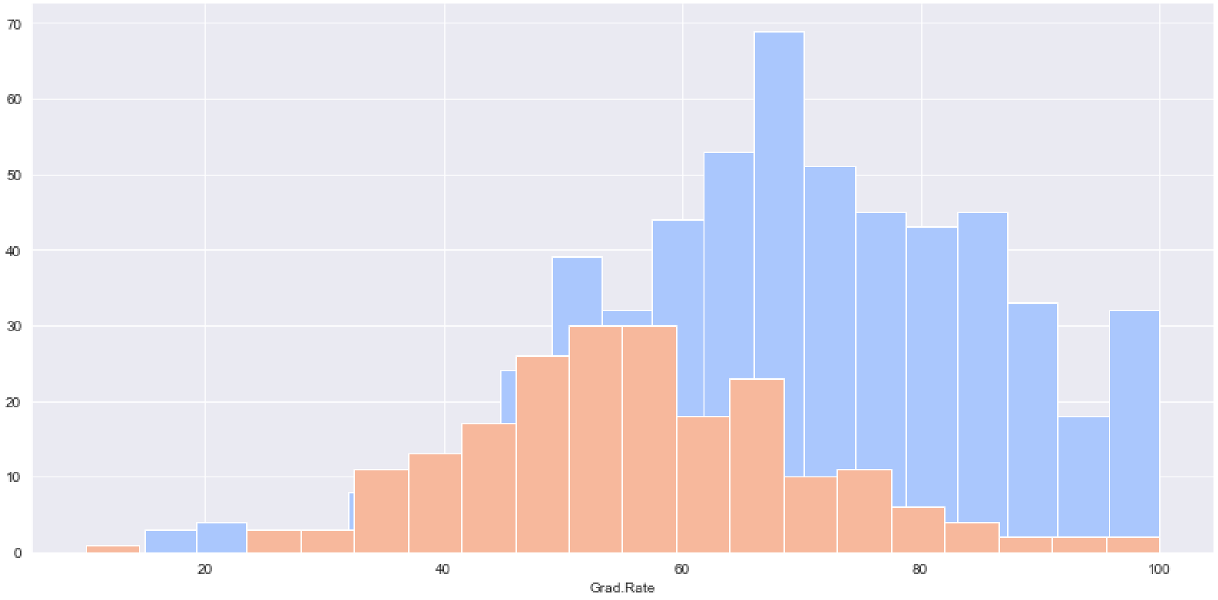
```
C:\Users\user\anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  """Entry point for launching an IPython kernel.
```

Out[54]:

Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.

In [55]:
```python
sns.set_style('darkgrid')
g=sns.FacetGrid(df,hue="Private",palette='coolwarm',height=6,aspect=2)
g=g.map(plt.hist,'Grad.Rate',bins=20)
```



K Means Cluster Creation

Now it is time to create the Cluster labels!

** Import KMeans from SciKit Learn.**

In [57]:

```
from sklearn.cluster import KMeans
```

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

In [58]:

```
kmeans = KMeans(n_clusters=2)
```

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

In [62]:

```
kmeans.fit(df.drop('Private',axis=1))
```

Out[62]:

```
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?**

In [63]:

```
kmeans.cluster_centers_
```

Out[63]:

```
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]])
```

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.**

In [64]:

```
def converter(cluster):
    if cluster=='Yes':
        return 1
    else:
        return 0
```

In [65]:
```
df['Cluster'] = df['Private'].apply(converter)
```

In [67]:
```
df.head(2)
```

Out[67]:

Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD
23	52	2885	537	7440	3300	450	2200	70
16	29	2683	1227	12280	6450	750	1500	29

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

In [68]:
```
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(df['Cluster'],kmeans.labels_))
print(classification_report(df['Cluster'],kmeans.labels_))
```

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

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!

Great Job!