DATA SCIENCE

K-Means Clustering

BY :

NADHIFA SOFIA

(19/448721/PPA/05804)

MASTER OF COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE AND ELECTRONICS

FACULTY OF MATHEMATICS AND NATURAL SCIENCES

UNIVERSITAS GADJAH MADA

2020

K Means Clustering is an unsupervised learning algorithm that will attempt to group similar clusters together in your data. By way of example on Figure 1. The overall goal is to divide data into distinct groups such that observations within each group are similar. Typical clustering problem look like;

- Cluster similar documents
- Cluster customers based on features
- Market segmentation
- Identify similar physical groups

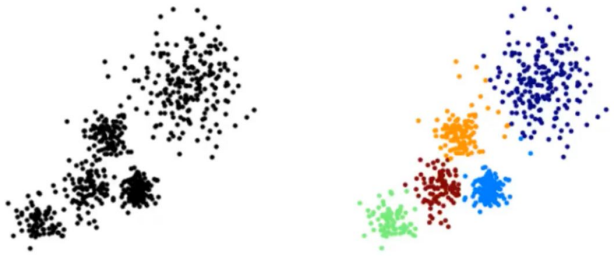


Figure 1. Before and after clustering.

The K Means Algorithm

- Choose a number of clusters ‘K’
- Randomly assign each point to a cluster
- Until clusters stop changing, repeat the following;
    - For each cluster, compute the cluster centroid by taking the mean vector of points in the cluster
    - Assign each data point to the cluster for which the centroid is the closest

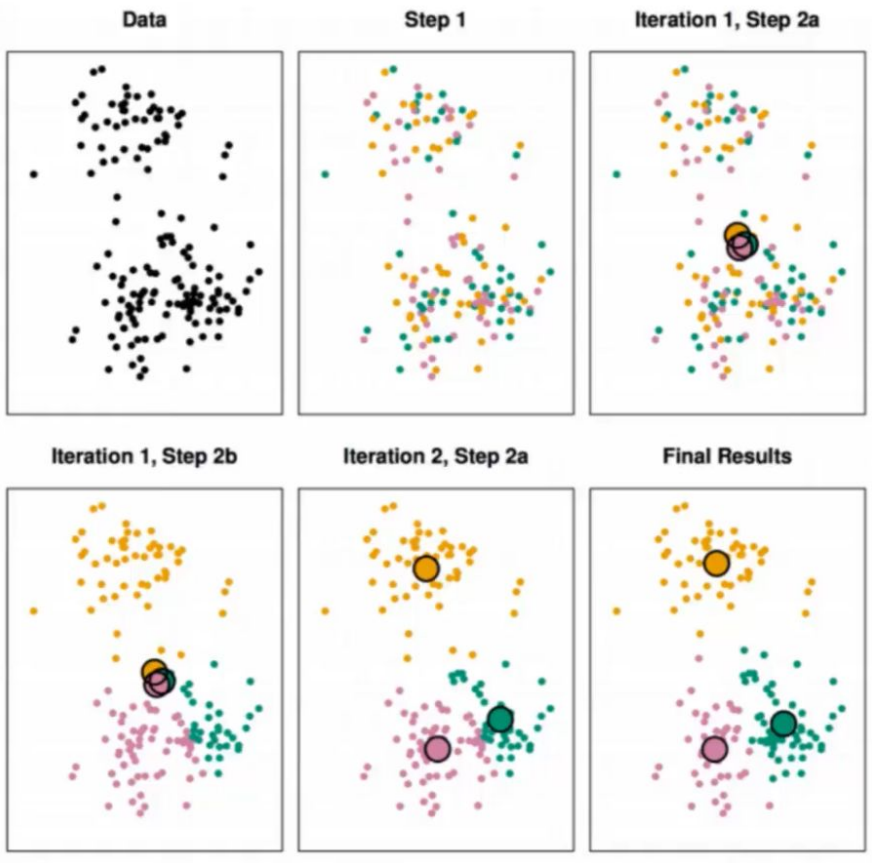


Figure 2. How KMeans work

Choosing a K Value

- There is no easy answer to choose a 'best' K value
- One way is the elbow method

First of all, compute the Sum of Squared Error (SSE) for some values of k (for example: 2, 4, 6, 8, etc). The SSE is defined as the sum of the squared distance between each of the clusters and its centroid like in figure 3.

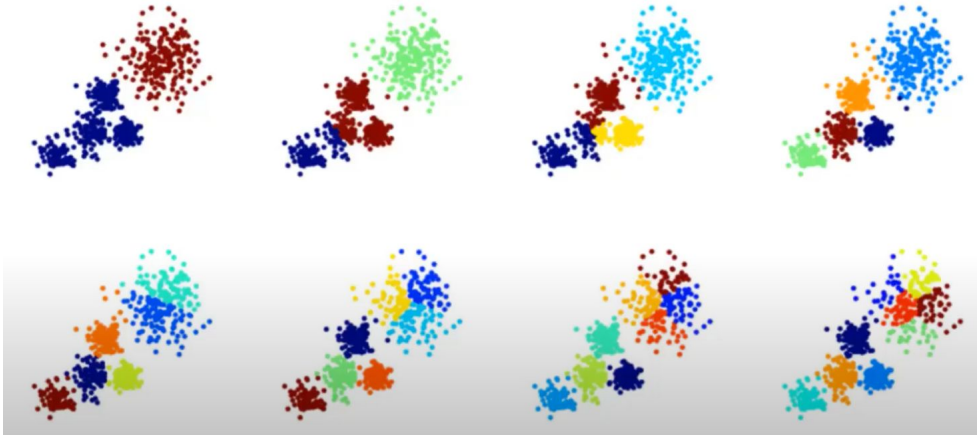


Figure 3. Choosing a K value

If you plot k against the SSE, you will see that the error decreases as k gets larger; this is because when the number of clusters increases, they should be smaller, so distortion is also smaller. The idea of the elbow method is to choose the k at which the SSE decreases abruptly.

This produces an 'elbow effect' in the graph, as you can see in the following picture;

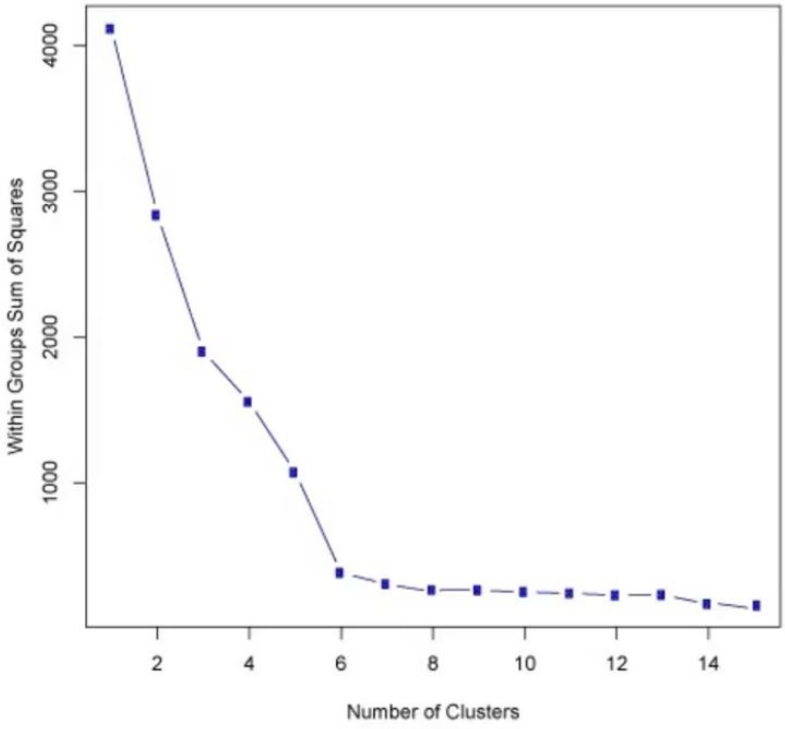


Figure 4. Example of elbow effect

K-means is a simple unsupervised machine learning algorithm that groups a dataset into a user-specified number (*k*) of clusters. The algorithm is somewhat naive--it clusters the data into *k* clusters, even if *k* is not the right number of clusters to use. Therefore, when using k-means clustering, users need some way to determine whether they are using the right number of clusters. The elbow method runs k-means clustering on the dataset for a range of values for k (say from 1-10) and then for each value of k computes an average score for all clusters. By default, the distortion score is computed, the sum of square distances from each point to its assigned center.

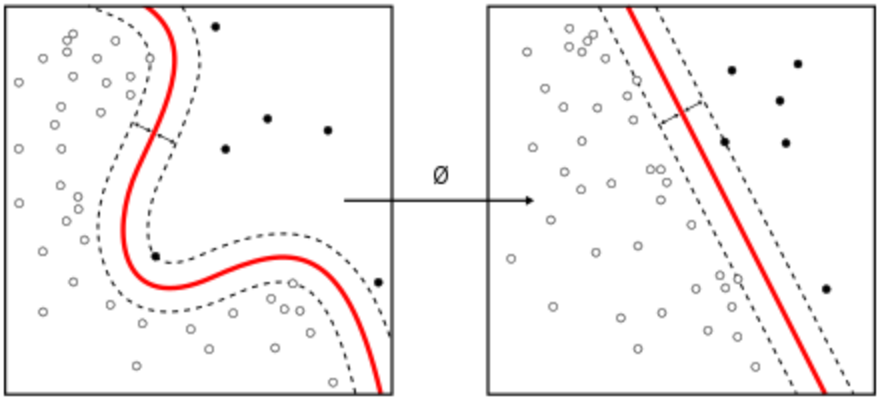


Figure 5. KMeans clustering

IMPLEMENTATION

Full version : https://github.com/dhifaaans/kmeans

'Buatlah cluster optimal dari dataset terlampir, dengan deskripsi seperti dalam video project description di masteri [minggu ke-8] (https://drive.google.com/file/d/1yFiIz_T1tblDkG1h0PJIoCZ0ICFKLANw/view)'.

We will use a dataframe with 777 observations on the following 8 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 : Percentage new students from top 10% of H.S. Class
- Top25perc : Percentage 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 : Percentage of faculty with Ph.D.'s
- Terminal : Percentage 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 pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams["patch.force_edgecolor"] = True

%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 [2]:
```
df = pd.read_csv('College_Data1.csv', index_col=0)
```

Check the head of the data

In [3]:
```
df.head()
```

Out[3]:

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

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 777 entries, Abilene Christian University to York College of
Pennsylvania
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
```

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

```
In [5]: df.describe()
```

Out[5]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Unde
count	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.0
mean	3001.638353	2018.804376	779.972973	27.558559	55.796654	3699.907336	855.2
std	3870.201484	2451.113971	929.176190	17.640364	19.804778	4850.420531	1522.4
min	81.000000	72.000000	35.000000	1.000000	9.000000	139.000000	1.0
25%	776.000000	604.000000	242.000000	15.000000	41.000000	992.000000	95.0
50%	1558.000000	1110.000000	434.000000	23.000000	54.000000	1707.000000	353.0
75%	3624.000000	2424.000000	902.000000	35.000000	69.000000	4005.000000	967.0
max	48094.000000	26330.000000	6392.000000	96.000000	100.000000	31643.000000	21836.0

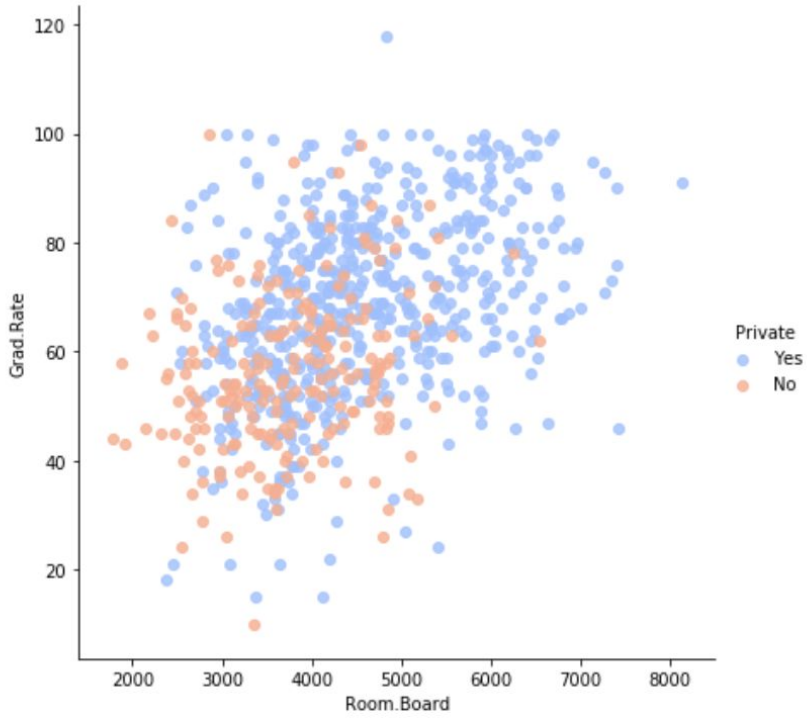
Exploratory Data Analysis

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

In [6]:
```
sns.lmplot(x='Room.Board',
           y='Grad.Rate',
           data=df,
           hue='Private',
           palette='coolwarm',
           fit_reg=False,
           size=6,
           aspect=1)
```

```
/Users/nadhifasofia/anaconda3/lib/python3.7/site-packages/seaborn/regression.py:574: UserWarning: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

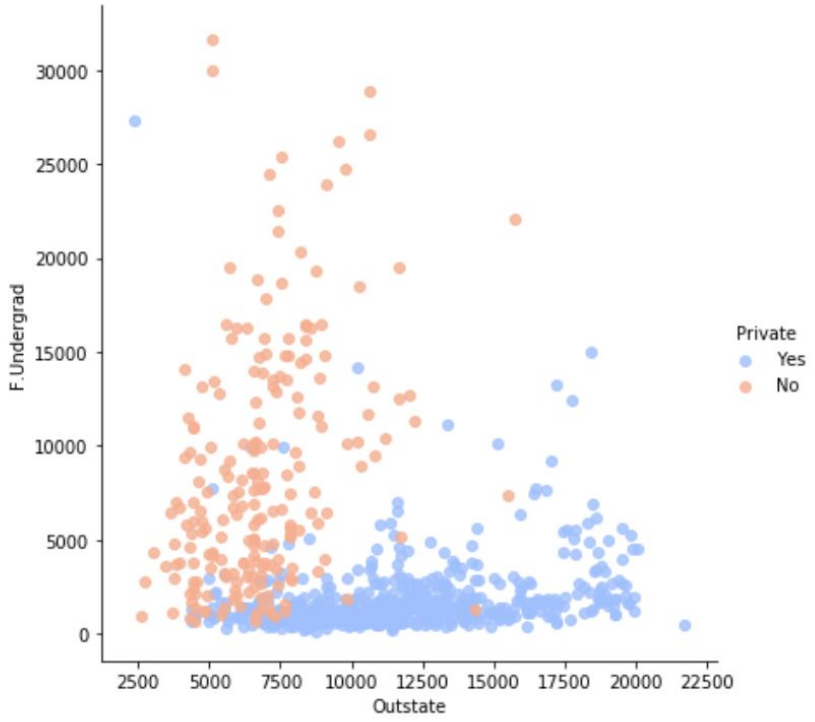
Out[6]:
```
<seaborn.axisgrid.FacetGrid at 0x1a2341ec18>
```



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

In [7]:
```
sns.lmplot(x='Outstate',
           y='F.Undergrad',
           data=df,
           hue='Private',
           palette='coolwarm',
           fit_reg=False,
           size=6,
           aspect=1)
```

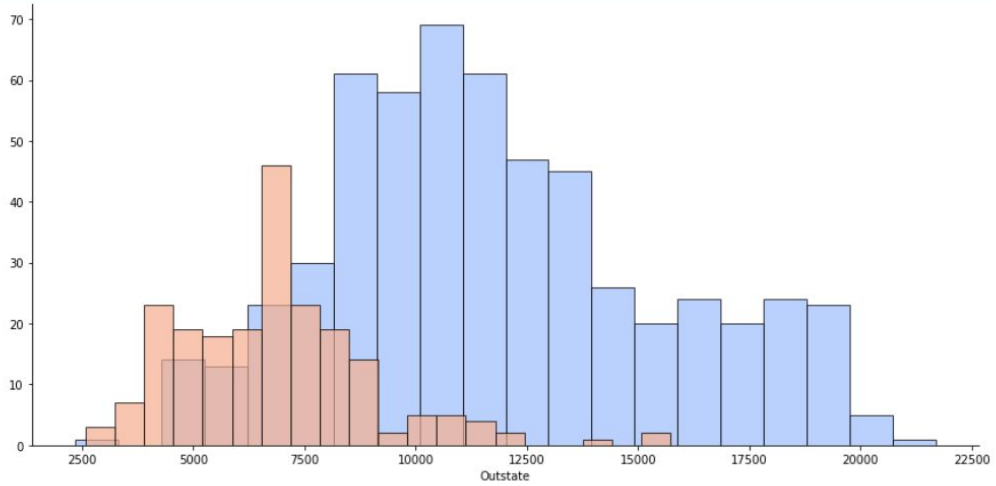
Out[7]: <seaborn.axisgrid.FacetGrid at 0x1a23e3d160>



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 we can do it just by using two instances of pandas.plot(kind='hist').

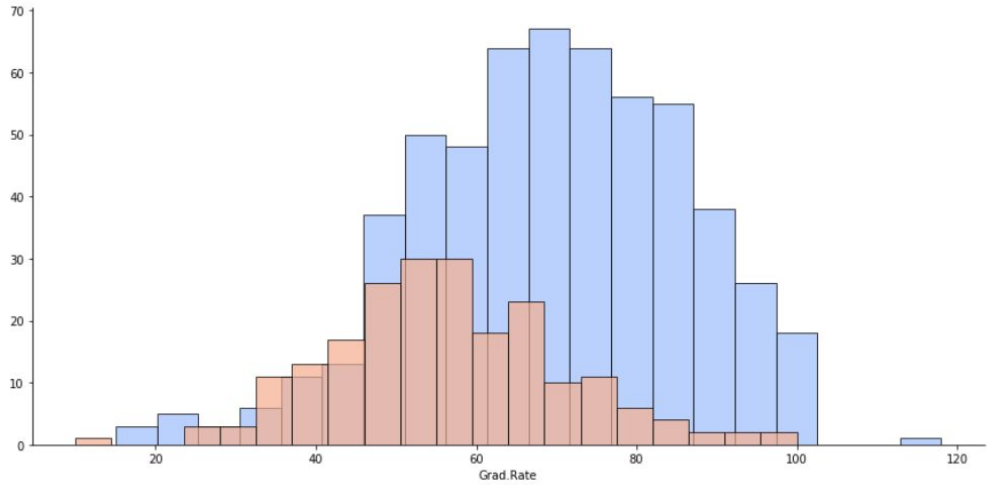
In [8]:
```
g = sns.FacetGrid(df, hue='Private', palette='coolwarm', size=6, aspe
g = g.map(plt.hist,'Outstate',bins=20,alpha=0.7)
```

```
/Users/nadhifasofia/anaconda3/lib/python3.7/site-packages/seaborn/a
xisgrid.py:243: UserWarning: The `size` parameter has been renamed
to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```



Create a similar histogram for the Grad.Rate column.

In [9]:
```
h = sns.FacetGrid(df, hue='Private', palette='coolwarm', size=6, aspe
h = h.map(plt.hist,'Grad.Rate',bins=20,alpha=0.7)
```



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

Out[10]:

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

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 [11]:
```
df['Grad.Rate']['Cazenovia College'] = 100
```

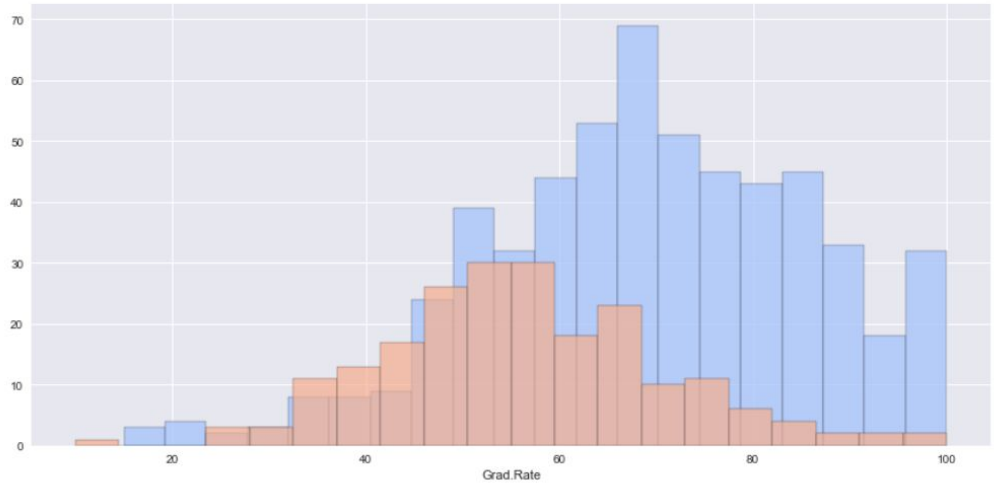
```
/Users/nadhifasofia/anaconda3/lib/python3.7/site-packages/ipykernel
_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy
  """Entry point for launching an IPython kernel.
```

```
In [12]: df[df['Grad.Rate'] > 100]
```

Out[12]:

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

```
In [13]: h = sns.FacetGrid(df,
                           hue='Private',
                           palette='coolwarm',
                           size=6,
                           aspect=2)
         h = h.map(plt.hist,'Grad.Rate',bins=20,alpha=0.7)
```



KMeans Cluster Creation

K Means Cluster Creation

Now it is time to create the Cluster labels!

Import KMeans from SciKit Learn.

```
In [14]: from sklearn.cluster import KMeans
```

Create an instance of a K Means model with 2 clusters; private and public

```
In [15]: km = KMeans(n_clusters = 2, random_state=90)
```

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

```
In [16]: km.fit(df.drop('Private', axis=1))

Out[16]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
             n_clusters=2, n_init=10, n_jobs=1, precompute_distances='auto',
             random_state=90, tol=0.0001, verbose=0)
```

What are the cluster center vectors?

```
In [17]: km.cluster_centers_

Out[17]: array([[  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],
                [  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]])
```

Evaluation

Since I have the label from the dataset as Mr. Seagate gave me through https://simaster.ugm.ac.id/elearning/mhs_materi/detail/1bMOdF6cQH9Ffi-kwxSN-5kf6TOBBMFeZXlt9kt-vII=/CJ78DlyuTjQjdhkUwDVWlFEZuDpGlk4h-Y3fbSsoY_4=, so I use it to test how KMeans performed by using confusion matrix.

Evaluation

Since I have the label from dataset as Mr. Seagate gave me thru simaster.ugm.ac.id, so I use it to test how KMeans performed by using confusion matrix.

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

In [18]:
```
def converter(prvt):
    if prvt == 'Yes':
        return 1
    else:
        return 0
```

In [19]:
```
df['Cluster'] = df['Private'].apply(converter)
```

In [20]:
```
df.head(3)
```

Out[20]:

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Out
Abilene Christian University	Yes	1660	1232	721	23	52	2885	537	
Adelphi University	Yes	2186	1924	512	16	29	2683	1227	1
Adrian College	Yes	1428	1097	336	22	50	1036	99	1

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

In [21]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [22]:
```
print(confusion_matrix(df['Cluster'],km.labels_))
print(classification_report(df['Cluster'],km.labels_))
```

```
[[ 74 138]
 [ 34 531]]
             precision    recall  f1-score   support

          0       0.69      0.35      0.46       212
          1       0.79      0.94      0.86       565

avg / total       0.76      0.78      0.75       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!
```