K Means Clustering Project

Cluster Universities into to two groups, Private and Public.

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.

About 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

In [1]:
```
### Import Libs
```

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
### Load Data set
```

In [4]:
```
Data_org = pd.read_csv('data/College_Data', index_col =0)
```

In [5]:
```
print(Data_org.shape)
Data_org.head(4)
```

```
(777, 18)
```

Out[5]:

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outs
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
Agnes Scott College	Yes	417	349	137	60	89	510	63	1

In [ ]:
```
```

In [6]:
```
Data_org.describe()
```

Out[6]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Under
count	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.00
mean	3001.638353	2018.804376	779.972973	27.558559	55.796654	3699.907336	855.29
std	3870.201484	2451.113971	929.176190	17.640364	19.804778	4850.420531	1522.43
min	81.000000	72.000000	35.000000	1.000000	9.000000	139.000000	1.00
25%	776.000000	604.000000	242.000000	15.000000	41.000000	992.000000	95.00
50%	1558.000000	1110.000000	434.000000	23.000000	54.000000	1707.000000	353.00
75%	3624.000000	2424.000000	902.000000	35.000000	69.000000	4005.000000	967.00
max	48094.000000	26330.000000	6392.000000	96.000000	100.000000	31643.000000	21836.00

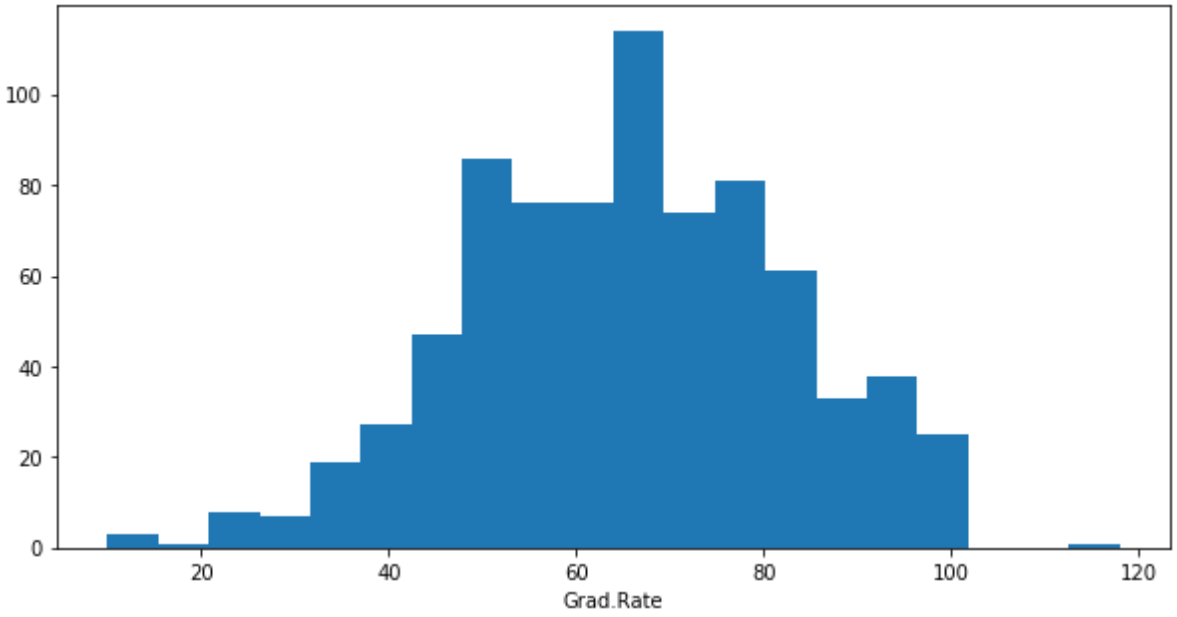
Note that in "Grad.Rate" column, graduation rate of higher than 100%. max value of "Grad.Rate = 118

In the follwoing we replace that with 100%

In [7]:
```
# make a copy of data
Data = Data_org.copy()
```

In [8]:
```
plt.figure(figsize=(10,5))
plt.hist(Data_org['Grad.Rate'], bins=20)
plt.xlabel('Grad.Rate')
```

Out[8]:
```
Text(0.5, 0, 'Grad.Rate')
```



In [9]:
```
Data_org[Data_org['Grad.Rate']>100]
```

Out[9]:

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

In [10]:
```
# replace 'Grad.Rate'>100 by 100
Index = Data_org[Data_org['Grad.Rate']>100].index
print(Index)
Data.loc[Index, 'Grad.Rate'] = 100
```

```
Index(['Cazenovia College'], dtype='object')
```

In [11]:
```
Data[Data['Grad.Rate'] > 100]['Grad.Rate'].any()
```

Out[11]:
```
False
```

Make data reday

In [12]:
```
X = Data.drop(columns=['Private'], axis=1)
# Note: we will note use y in algorithm
y = Data['Private']
```

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

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

In [15]:
```
kmeans.fit(X)
```

Out[15]:
```
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)
```

In [16]:
```
#cluster center vectors
print(X.shape)
print(kmeans.cluster_centers_.shape)
kmeans.cluster_centers_
```

```
(777, 17)
(2, 17)
```

Out[16]:
```
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]])
```

In [51]:
```
kmeans.labels_[0:50]
```

Out[51]:
```
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])
```

In [52]:
```
#Sum of squared distances(SSE) of samples to their closest cluster center.
kmeans.inertia_
```

Out[52]:
```
48356200684.31276
```

Evaluation

Since we have the label of the data, we can use it just to compare

In [18]:
```
# To convert Label {Yes, No} into numerical value
def convertor(cluster):
    if cluster == 'Yes':
        return 1
    else:
        return 0
```

In [19]:
```
y_value = y.apply(convertor)
y_value[0:10]
```

Out[19]:
```
Abilene Christian University    1
Adelphi University              1
Adrian College                  1
Agnes Scott College             1
Alaska Pacific University       1
Albertson College               1
Albertus Magnus College         1
Albion College                  1
Albright College                1
Alderson-Broaddus College       1
Name: Private, dtype: int64
```

In [20]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [21]:
```
print(confusion_matrix(y_value, kmeans.labels_))
print(classification_report(y_value,kmeans.labels_))
```

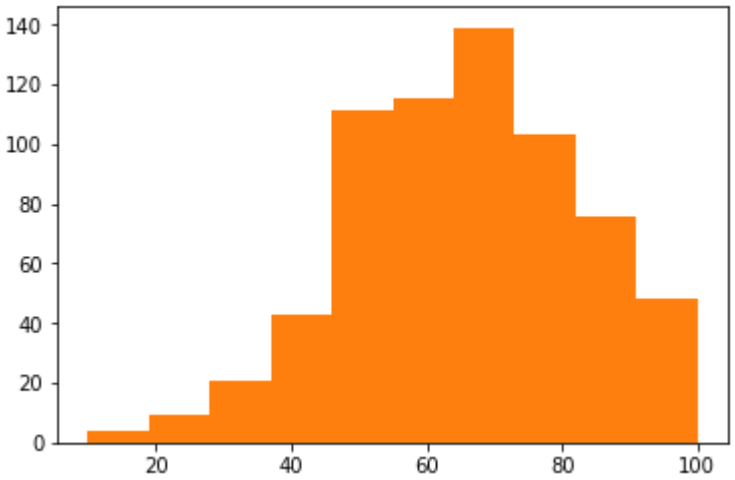
```
[[138  74]
 [531  34]]
              precision    recall  f1-score   support

           0       0.21      0.65      0.31       212
           1       0.31      0.06      0.10       565

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

Some visualization on Clusters

In [41]:
```
centroids = kmeans.cluster_centers_
```

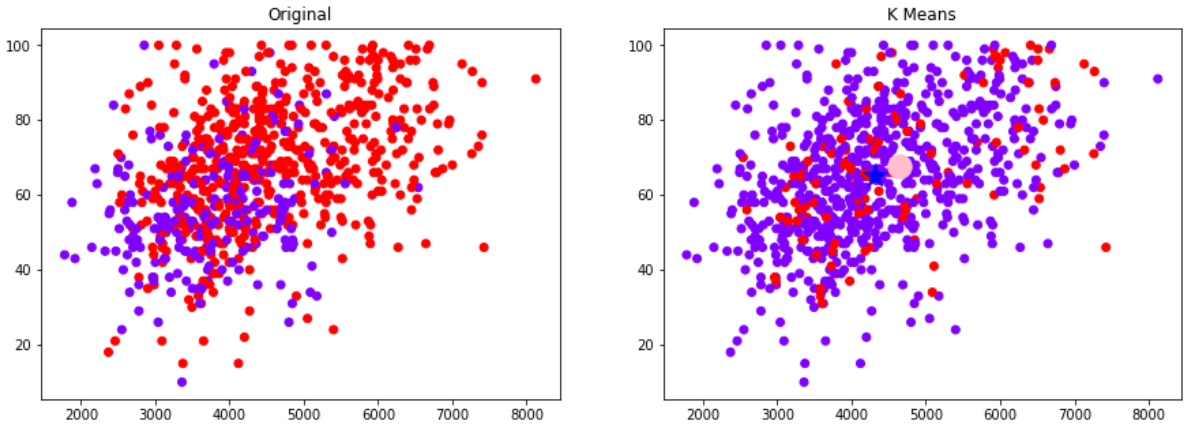
In [22]:
```
plt.hist(X[kmeans.labels_ == 1]['Grad.Rate'])
plt.hist(X[kmeans.labels_ == 0]['Grad.Rate'])
```

Out[22]:
```
(array([  4.,   9.,  21.,  43., 111., 115., 139., 103.,  76.,  48.]),
 array([ 10.,  19.,  28.,  37.,  46.,  55.,  64.,  73.,  82.,  91., 100.]),
 <a list of 10 Patch objects>)
```



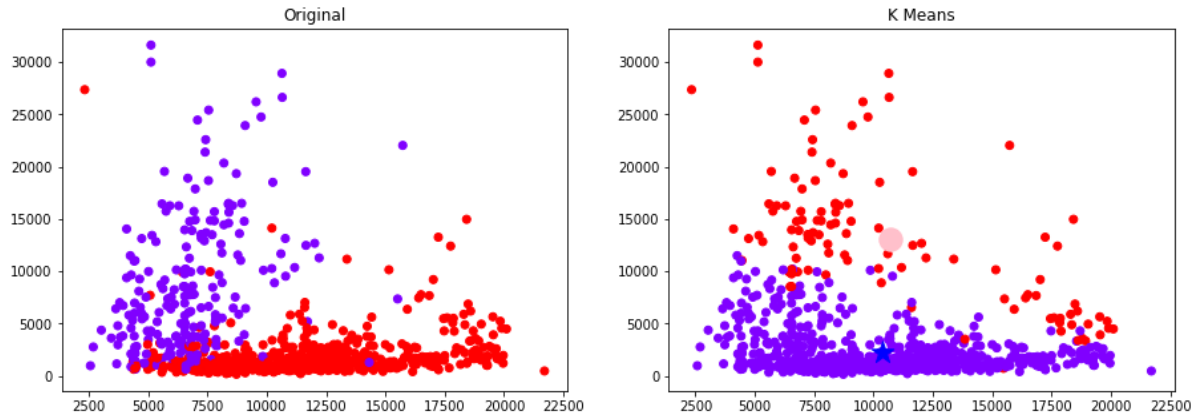
In [43]:
```
f, ax = plt.subplots(1, 2, figsize=(15,5))

ax[0].scatter(x=Data['Room.Board'], y=Data['Grad.Rate'], c=y_value, cmap = 'ra
inbow')
ax[1].scatter(x=Data['Room.Board'], y=Data['Grad.Rate'], c=kmeans.labels_, cma
p = 'rainbow')

ax[0].set_title("Original")
ax[1].set_title('K Means')

ax[1].scatter(centroids[0, 8], centroids[0, 16], marker='*', s=300,
            c='blue', label='centroid')
ax[1].scatter(centroids[1, 8], centroids[1, 16], marker='o', s=300,
            c='pink', label='centroid')
```

Out[43]:
```
<matplotlib.collections.PathCollection at 0x24a311f07f0>
```



In [40]:
```
f, ax = plt.subplots(1, 2, figsize=(15,5))

ax[0].scatter(x=Data['Outstate'], y=Data['F.Undergrad'], c=y_value, cmap = 'ra
inbow')
ax[1].scatter(x=Data['Outstate'], y=Data['F.Undergrad'], c=kmeans.labels_, cma
p = 'rainbow', )

ax[0].set_title("Original")
ax[1].set_title('K Means')



ax[1].scatter(centroids[0, 7], centroids[0, 5], marker='*', s=300,
            c='blue', label='centroid')
ax[1].scatter(centroids[1, 7], centroids[1, 5], marker='o', s=300,
            c='pink', label='centroid')

#plt.legend()
```

Out[40]: <matplotlib.collections.PathCollection at 0x24a314f89e8>



In [44]:
```
X.head(2)
```

Out[44]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Ro
Abilene Christian University	1660	1232	721	23	52	2885	537	7440	
Adelphi University	2186	1924	512	16	29	2683	1227	12280	

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Calculate Silhouette Coefficient

Silhouette analysis can be used to determine the degree of separation between clusters.

For each sample

- Compute the average distance from all data points in the same cluster (a_i).
- Compute the average distance from all data points in the closest cluster (b_i).
- Compute the coefficient for the i-th example:

$$s_i = \frac{b_i - a_i}{max(a_i, b_i)}$$

The coefficient can take values in the interval [-1, 1].

- If it is 0 –> the sample is very close to the neighboring clusters.
- It it is 1 –> the sample is far away from the neighboring clusters.
- It it is -1 –> the sample is assigned to the wrong clusters.

Therefore, we want the coefficients to be as big as possible and close to 1 to have a good clusters.

In [48]:
```
from sklearn.metrics import silhouette_score
silhouette_score(X,kmeans.labels_)
```

Out[48]:
```
0.5599267973651544
```

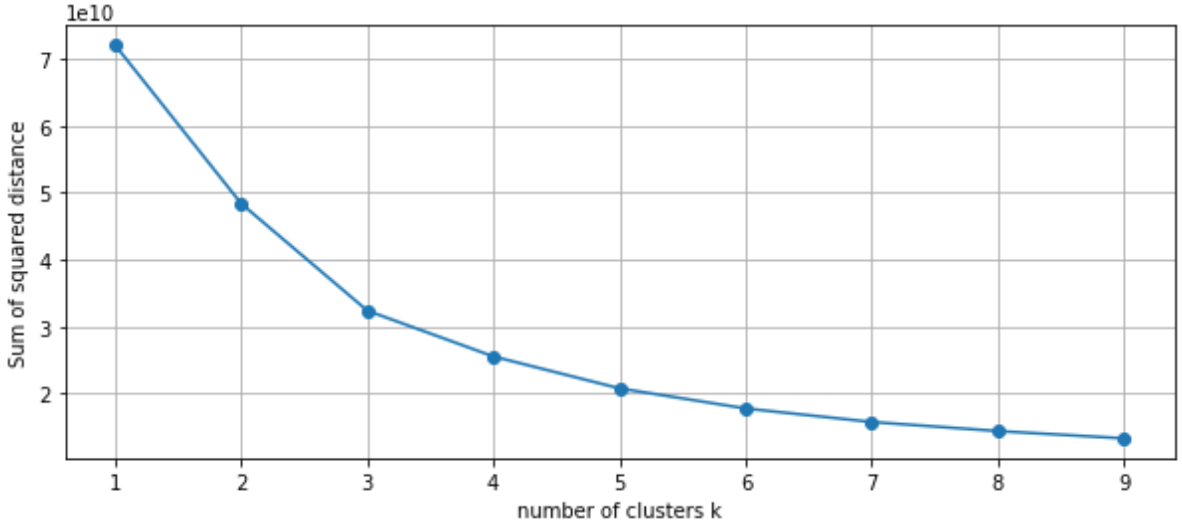
In [ ]:
```

```

Elbow Method to find best K value

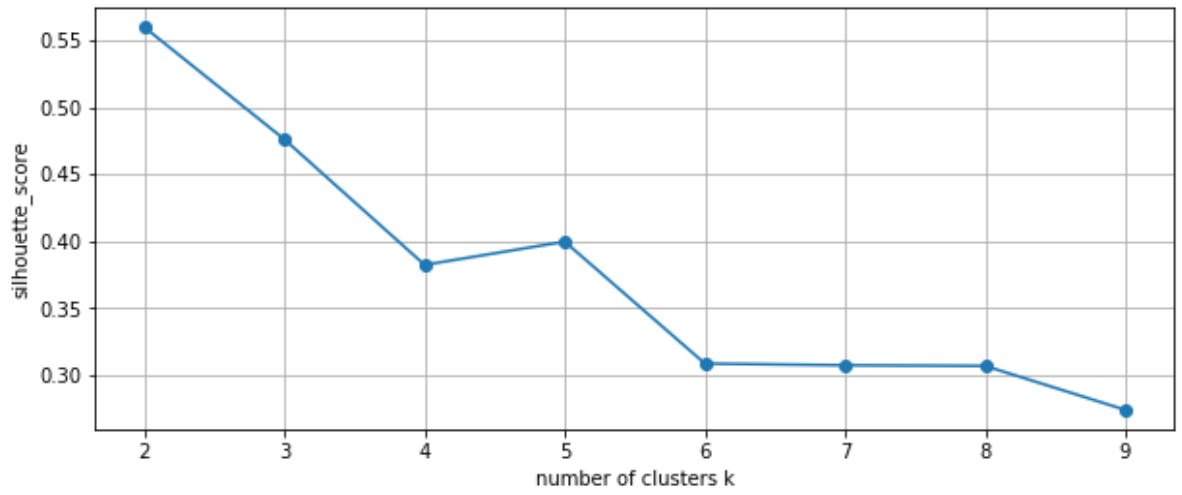
In [81]:
```
SSE =[]
silh_score =[]
list_k = list(range(1,10))
print(list_k)
for k in list_k:
    model = KMeans(n_clusters=k)
    model.fit(X)
    SSE.append(model.inertia_)
    if k>1:
        silh_score.append(silhouette_score(X, model.labels_))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [83]:
```
## plot SSE against K
plt.figure(figsize=(10,4))
plt.plot(list_k, SSE, marker ='o')
plt.xlabel('number of clusters k')
plt.ylabel('Sum of squared distance')
plt.grid(True)
```



In [86]:
```
## plot silhouette_score against K
plt.figure(figsize=(10,4))
plt.plot(list_k[1:], silh_score, marker ='o')
plt.xlabel('number of clusters k')
plt.ylabel('silhouette_score')
plt.grid(True)
```



we want the coefficients to be as big as possible and close to 1 to have a good clusters.