Assignment2

April 9, 2020

[3]:
```
import pandas as pd
from sklearn.impute import KNNImputer
from sklearn.decomposition import PCA
import numpy as np
import matplotlib.pylab as plt
from adjustText import adjust_text
from pandas.plotting import scatter_matrix
import matplotlib.pylab as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

data = pd.read_csv('/Users/nivethida/Downloads/universities_pca_2.csv')
```

[21]:
```
#Sub Category 1

# No Missing records
# All entries are float
from sklearn.preprocessing import MinMaxScaler, StandardScaler, RobustScaler,␣
 ↪MaxAbsScaler
import pandas as pd

data = pd.read_csv('/Users/nivethida/Downloads/universities_pca_2.csv')

scaler = StandardScaler()
data = data.iloc[:, 1:]

# Using standard scalar
scaler = StandardScaler()
standardized_data = norm_df = pd.DataFrame(scaler.fit_transform(data),␣
 ↪index=data.index, columns=data.columns)
print("StandardScaler: \n", standardized_data.head(5), "\n")

# Using minmax
scaler = MinMaxScaler()
standardized_data2 = pd.DataFrame(scaler.fit_transform(data), index=data.index,␣
 ↪columns=data.columns)
print("MinMaxScaler: \n",standardized_data2, "\n")
```

```
# using robust scalar
transformer = RobustScaler().fit(data[1:])
normalized_data = transformer.transform(data[1:])
print("Using robust scalar: \n",normalized_data, "\n")

# Using MaxAbs scalar
scalar =   MaxAbsScaler().fit(data[1:])
normalized_data = transformer.transform(data[1:])
print("Using maxabs scalar: \n",normalized_data, "\n")

# I have decided to used min max scalar for the rest of the assignment.
```

```
StandardScaler:
    # appli. rec'd  # appl. accepted  # new stud. enrolled  \
0       -0.726085         -0.766447             -0.793414
1       -0.737636         -0.778042             -0.756242
2       -0.575673         -0.589724             -0.539768
3       -0.624090         -0.616912             -0.714697
4        0.311318         -0.225084             -0.487290

   % new stud. from top 10%  % new stud. from top 25%  # FT undergrad  \
0                 -0.650759                 -0.573903       -0.710495
1                 -1.300829                 -1.558991       -0.658397
2                  2.112035                  1.593292       -0.468871
3                 -0.109035                 -0.426140       -0.648535
4                  0.107655                  0.214168       -0.569208

   # PT undergrad  in-state tuition  out-of-state tuition      room     board  \
0        0.046333         -0.335086             -0.700046 -0.843743  0.667644
1        0.680985         -1.390805             -1.241943  0.411116  0.226150
2       -0.382380          0.408890              0.251873 -0.240175  0.544026
3       -0.434836         -0.240728             -0.579315 -1.180618  0.738283
4       -0.439369         -0.678766             -1.139785 -1.118857 -1.027693

   add. fees  estim. book costs  estim. personal $  % fac. w/PHD  \
0  -0.700526           1.541090           0.276102      0.167704
1  -0.970586          -0.299262          -0.220137     -2.054792
2  -0.728658          -0.912713          -0.604796      0.047569
3  -0.784920          -0.299262          -0.311163     -0.613173
4   0.109652           2.767992           0.129286     -1.033645

   stud./fac. ratio  Graduation rate
0         -0.529598        -2.789257
1         -1.145818        -1.465311
2          0.009594         0.355113
3         -0.657977        -1.189489
```

```
4          0.394732        -1.079161
```

MinMaxScaler:

	# appli. rec'd	# appl. accepted	# new stud. enrolled	\
0	0.002416	0.003236	0.004399	
1	0.001437	0.002132	0.009741	
2	0.015161	0.020062	0.040848	
3	0.011059	0.017473	0.015711	
4	0.090322	0.054779	0.048390	
..	…	…	…	
466	0.117708	0.169744	0.303221	
467	0.001562	0.002551	0.007541	
468	0.010517	0.016636	0.022781	
469	0.086407	0.137234	0.311076	
470	0.040652	0.055388	0.164336	

	% new stud. from top 10%	% new stud. from top 25%	# FT undergrad	\
0	0.157895	0.384615	0.000000	
1	0.031579	0.164835	0.007740	
2	0.694737	0.868132	0.035899	
3	0.263158	0.417582	0.009206	
4	0.305263	0.560440	0.020991	
..	…	…	…	
466	0.200000	0.472527	0.263331	
467	0.168421	0.351648	0.001051	
468	0.210526	0.384615	0.017519	
469	0.136842	0.560440	0.237434	
470	0.231579	0.406593	0.232083	

	# PT undergrad	in-state tuition	out-of-state tuition	room	\
0	0.039753	0.356659	0.341940	0.234674	
1	0.084635	0.058178	0.219458	0.448755	
2	0.009434	0.567002	0.557095	0.337644	
3	0.005725	0.383337	0.369228	0.177203	
4	0.005404	0.259491	0.242548	0.187739	
..	…	…	…	…	
466	0.057843	0.082085	0.312343	0.221504	
467	0.000962	0.435666	0.422754	0.277778	
468	0.003526	0.505849	0.494542	0.223659	
469	0.107076	0.053971	0.212531	0.332854	
470	0.068102	0.066694	0.259446	0.196839	

	board	add. fees	estim. book costs	estim. personal $	% fac. w/PHD	\
0	0.491022	0.037071	0.315556	0.190840	0.715789	
1	0.428678	0.007414	0.182222	0.139237	0.326316	
2	0.473566	0.033982	0.137778	0.099237	0.694737	
3	0.500998	0.027804	0.182222	0.129771	0.578947	
4	0.251621	0.126043	0.404444	0.175573	0.505263	

```
..        …        …                 …                 …             …
466  0.151870   0.327464          0.084000          0.154198      0.757895
467  0.341397   0.024714          0.182222          0.175573      0.421053
468  0.317955   0.025332          0.253333          0.236641      0.336842
469  0.361347   0.098857          0.271111          0.200000      0.726316
470  0.356359   0.089589          0.226667          0.190840      0.873684

     stud./fac. ratio  Graduation rate
0            0.347490         0.000000
1            0.254826         0.233010
2            0.428571         0.553398
3            0.328185         0.281553
4            0.486486         0.300971
..                …               …
466          0.633205         0.446602
467          0.216216         0.339806
468          0.332046         0.359223
469          0.528958         0.339806
470          0.471042         0.291262

[471 rows x 17 columns]

Using robust scalar:
 [[-0.49164016 -0.60966237 -0.5600316  … -1.54166667 -0.76328502
  -1.03846154]
 [-0.27664954 -0.35108427 -0.24723539 … -0.08333333  0.10628019
   0.23076923]
 [-0.34091836 -0.38841614 -0.5        … -0.54166667 -0.39613527
  -0.84615385]
 …
 [-0.34940054 -0.4004941  -0.42890995 … -1.5        -0.37681159
  -0.53846154]
 [ 0.83940951  1.33873181  2.4699842  …  0.04166667  0.60869565
  -0.61538462]
 [ 0.12266536  0.15838595  0.99447077 …  0.625       0.31884058
  -0.80769231]]

Using maxabs scalar:
 [[-0.49164016 -0.60966237 -0.5600316  … -1.54166667 -0.76328502
  -1.03846154]
 [-0.27664954 -0.35108427 -0.24723539 … -0.08333333  0.10628019
   0.23076923]
 [-0.34091836 -0.38841614 -0.5        … -0.54166667 -0.39613527
  -0.84615385]
 …
 [-0.34940054 -0.4004941  -0.42890995 … -1.5        -0.37681159
  -0.53846154]
 [ 0.83940951  1.33873181  2.4699842  …  0.04166667  0.60869565
```

```
 -0.61538462]
[ 0.12266536  0.15838595  0.99447077 …  0.625       0.31884058
 -0.80769231]]
```

[3]:
```
#Pre analysis before PCA
import pandas as pd
data = pd.read_csv('/Users/nivethida/Downloads/universities_pca_2.csv')
data.info() # View data

# No Missing records
# All entries are float
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 471 entries, 0 to 470
Data columns (total 18 columns):
College Name                471 non-null object
# appli. rec'd              471 non-null int64
# appl. accepted            471 non-null int64
# new stud. enrolled        471 non-null int64
% new stud. from top 10%    471 non-null int64
% new stud. from top 25%    471 non-null int64
# FT undergrad              471 non-null int64
# PT undergrad              471 non-null int64
in-state tuition            471 non-null int64
out-of-state tuition        471 non-null int64
room                        471 non-null int64
board                       471 non-null int64
add. fees                   471 non-null int64
estim. book costs           471 non-null int64
estim. personal $           471 non-null int64
% fac. w/PHD                471 non-null int64
stud./fac. ratio            471 non-null float64
Graduation rate             471 non-null int64
dtypes: float64(1), int64(16), object(1)
memory usage: 66.4+ KB
```

[18]:
```
# Pre analysis before PCA
# LoOking for any instances having same value?
data.describe()
# In-state and Out-of-state have same max tution
# No zero varience
# No zero s
```

[18]:

	# appli. rec'd	# appl. accepted	# new stud. enrolled	\
count	471.000000	471.000000	471.000000	
mean	3147.303609	2062.955414	780.704883	

```
std       4073.138136       2503.752754            915.633300
min         77.000000         61.000000             27.000000
25%        802.000000        635.500000            264.000000
50%       1646.000000       1227.000000            443.000000
75%       3862.000000       2456.000000            896.500000
max      48094.000000      26330.000000           6392.000000

       % new stud. from top 10%  % new stud. from top 25%  # FT undergrad  \
count                471.000000                471.000000      471.000000
mean                  28.012739                 55.651805     3562.938429
std                   18.479196                 20.324333     4669.226389
min                    1.000000                  9.000000      249.000000
25%                   15.000000                 40.000000     1018.000000
50%                   23.000000                 54.000000     1715.000000
75%                   36.000000                 69.000000     4055.500000
max                   96.000000                100.000000    31643.000000

       # PT undergrad  in-state tuition  out-of-state tuition        room  \
count      471.000000        471.000000            471.000000   471.00000
mean       797.454352       9406.634820          10575.161359  2221.10828
std       1545.796419       5516.794516           4311.672049   713.18811
min          1.000000        608.000000           1044.000000   640.00000
25%         81.500000       3650.500000           7290.000000  1740.00000
50%        299.000000       9858.000000          10100.000000  2090.00000
75%        869.000000      13246.000000          13286.000000  2663.00000
max      21836.000000      20100.000000          20100.000000  4816.00000

             board    add. fees  estim. book costs  estim. personal $  \
count   471.000000   471.000000         471.000000         471.000000
mean   2121.940552   379.021231         548.783439        1311.940552
std     566.861037   355.855253         163.185575         681.847238
min     531.000000    10.000000          90.000000         250.000000
25%    1750.000000   137.500000         500.000000         850.000000
50%    2082.000000   280.000000         500.000000        1200.000000
75%    2420.000000   486.000000         600.000000        1600.000000
max    4541.000000  3247.000000        2340.000000        6800.000000

       % fac. w/PHD  stud./fac. ratio  Graduation rate
count    471.000000        471.000000       471.000000
mean      73.208068         13.962633        65.562633
std       16.665649          3.898855        18.146912
min        8.000000          2.900000        15.000000
25%       63.000000         11.300000        53.000000
50%       76.000000         13.400000        66.000000
75%       87.000000         16.450000        79.000000
max      103.000000         28.800000       118.000000
```

[ ]:
```
# Pre analysis before PCA
# Do we have any feature that is highly corelated ?
data.corr()
#application accecpted and applica received
# new stu enrolled and ft undegrad
# ft undergrad and new stu enrolled
#New stu from to 10 and 20
#out of state and instate tuition
```

[9]:
```
# Sub category 2.1
# Explained variance by component
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA  #For PCA

scaler = MinMaxScaler() #Am using min max scaler
pcs = PCA(whiten=True)
scaledData = pd.DataFrame(pcs.fit_transform(scaler.fit_transform(data.iloc[:,1:
 ↪])),
                          columns=['PC{}'.format(i) for i in range(1, len(data.
 ↪iloc[:,1:].columns) + 1)])
scaledData.head()
```

[9]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	\
0	-0.694017	-0.712218	-0.199021	2.427105	-0.248957	-2.109476	1.382989	
1	-1.345484	-1.127077	0.580702	1.697747	-1.523022	0.855264	-0.670533	
2	0.834128	0.339727	-2.058898	0.288680	-0.413592	0.537881	1.326476	
3	-0.440296	-0.712704	-0.758674	1.192295	0.340365	-0.578288	0.351317	
4	-0.762631	-0.110402	-1.642377	0.858098	-0.486873	0.628348	0.420154	

	PC8	PC9	PC10	PC11	PC12	PC13	PC14	\
0	1.394826	0.418490	-0.184980	1.685107	-0.505497	-1.100050	-0.747420	
1	1.047350	-0.511319	-2.560828	-0.557086	0.294791	0.721178	1.714918	
2	0.381057	0.418668	-0.618879	-1.283485	1.426036	0.423111	0.020694	
3	1.528346	0.582275	-1.294826	-0.178288	0.420163	-0.807991	-0.248523	
4	0.042938	0.551891	0.681923	2.849480	-0.001616	-0.684945	-0.898671	

	PC15	PC16
0	0.493196	-0.116278
1	0.400643	-0.447853
2	0.617220	0.581347
3	0.704053	0.521778
4	1.302816	0.131740

[12]:
```
summary = pd.DataFrame({'Explained Variance': pcs.explained_variance_,␣
 ↪#explained variance
```

```
                          'Explained Variance Ratio': pcs.
↪explained_variance_ratio_,  #explained variance %
                          'Cumulative Proportion': np.cumsum(pcs.
↪explained_variance_ratio_)}) #cumulative %
summary = summary.transpose()
summary.columns = scaledData.columns
summary
```

[12]:

	PC1	PC2	PC3	PC4	PC5	\
Explained Variance	0.219108	0.079210	0.032307	0.018368	0.017194	
Explained Variance Ratio	0.500544	0.180951	0.073804	0.041962	0.039280	
Cumulative Proportion	0.500544	0.681495	0.755299	0.797261	0.836541	

	PC6	PC7	PC8	PC9	PC10	\
Explained Variance	0.015844	0.012831	0.010840	0.009752	0.007538	
Explained Variance Ratio	0.036196	0.029311	0.024763	0.022279	0.017220	
Cumulative Proportion	0.872736	0.902047	0.926811	0.949089	0.966309	

	PC11	PC12	PC13	PC14	PC15	\
Explained Variance	0.004507	0.003695	0.002749	0.002020	0.001268	
Explained Variance Ratio	0.010297	0.008442	0.006280	0.004613	0.002897	
Cumulative Proportion	0.976606	0.985048	0.991327	0.995941	0.998837	

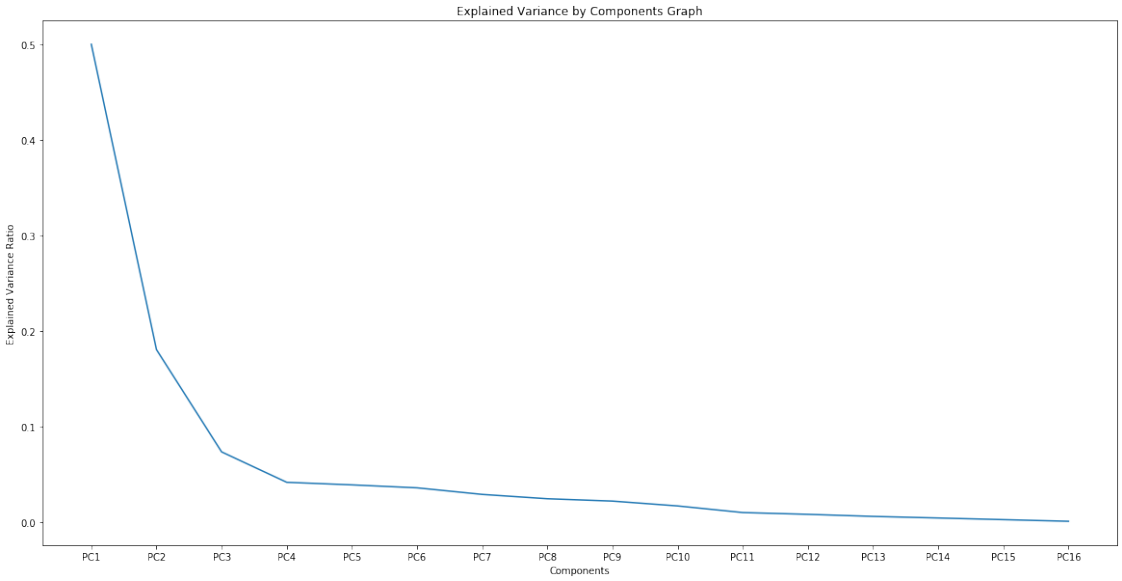
	PC16
Explained Variance	0.000509
Explained Variance Ratio	0.001163
Cumulative Proportion	1.000000

[24]:
```
# scree plot

plt.figure(figsize=(20,10))
plt.plot(summary[1:2].transpose())
plt.xlabel('Components')
plt.ylabel('Explained Variance Ratio')
plt.title('Explained Variance by Components Graph')

# I plan to use first three components as the inflection point of the␣
↪inflection plot is at 4 also when I look in to
# the cumulative proportion PC4 have 79 % of necessary details.
```

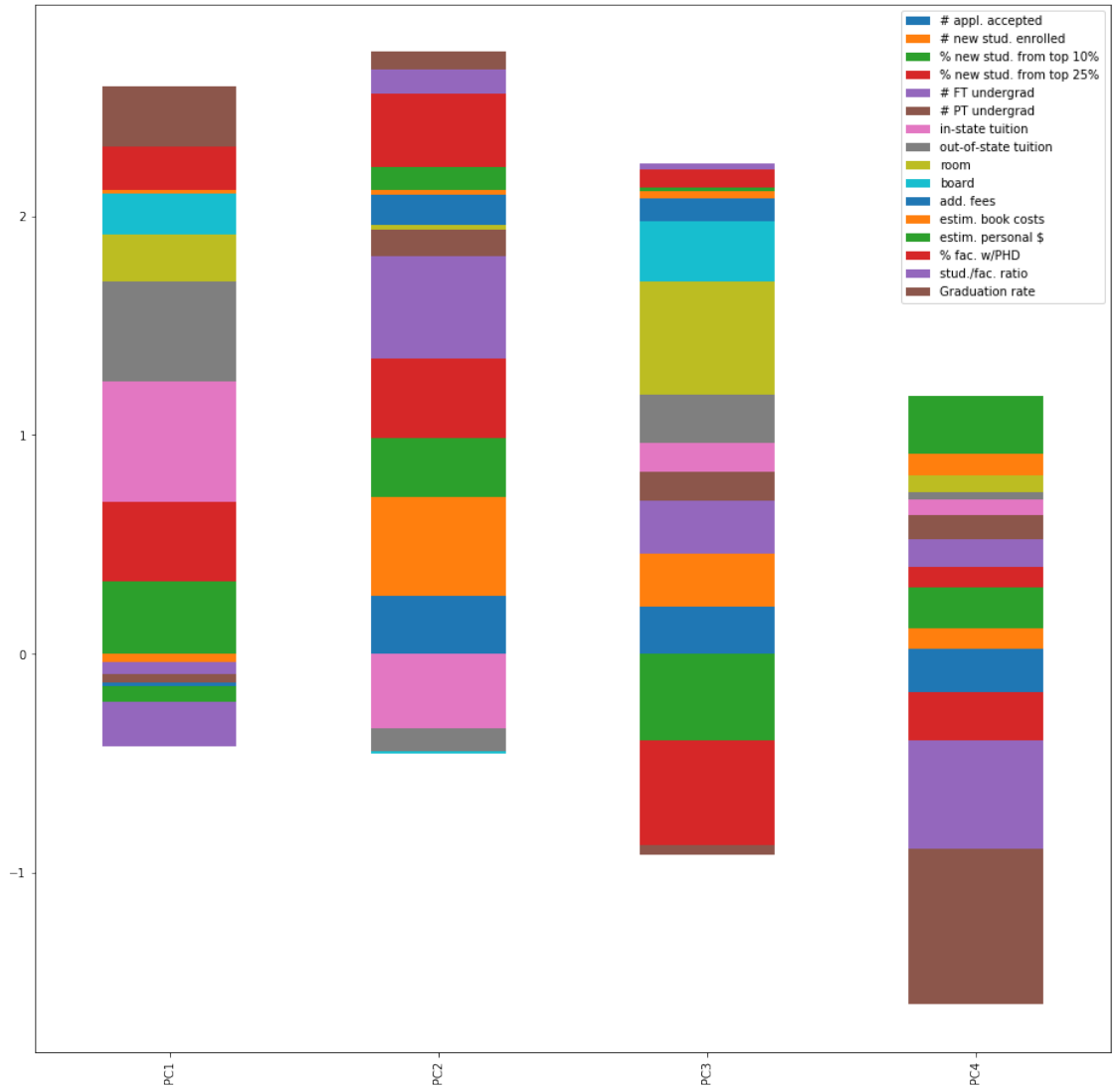
[24]:
```
Text(0.5, 1.0, 'Explained Variance by Components Graph')
```



[26]:
```
# Sub category 2.2
# Analysis of the first 4 components
import numpy as np
summary = pd.DataFrame({'Explained Variance': pcs.explained_variance_,␣
 ↪#explained variance
                         'Explained Variance Ratio': pcs.
 ↪explained_variance_ratio_,  #explained variance %
                         'Cumulative Proportion': np.cumsum(pcs.
 ↪explained_variance_ratio_)}) #cumulative %
summary = summary.transpose()
summary.columns = scaledData.columns
pcsComponents_df = pd.DataFrame(pcs.components_.transpose(), columns=summary.
 ↪columns,
                                 index=data.iloc[:,1:].columns)
pcsComponents_df[['PC1','PC2','PC3','PC4']].transpose().plot(kind='bar',␣
 ↪stacked=True,figsize=(16, 16))

# In the bar chart below we can see that
# PC1 is mostly represented by Pink, Red and Grey ie in state tuition,new stu␣
 ↪from top 25 and out of state tuition
# PC2 is mostly represented by Red, Purple and Orange ie new stu from top 25,␣
 ↪stud/ fac ratio and new stu enrolled
# PC2 is mostly represented by Red and Lime green ie new stu from top 25 and␣
 ↪room
# PC4 is mostly represented by Brown and Purple ie Graduation rate and stud/␣
 ↪fac ratio
```

[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2528e250>



[25]:
```
# Subcategory 2.3
# Scatter plot the first 2 components

from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA  #principal components Analysis

scaler = MinMaxScaler() #min max scaler
pcs = PCA(whiten=True)
scaledData = pd.DataFrame(pcs.fit_transform(scaler.fit_transform(data.iloc[:,1:
 ↪])),
```

```
                    columns=['PC{}'.format(i) for i in range(1, len(data.
 ↪iloc[:,1:].columns) + 1)])
scaledData.plot.scatter(x='PC1', y='PC2',figsize=(16, 16))
# We see natural clusting of points.
# Avoided the named points for the plot as it looks cluttered with 400+ records.
```

[25]: <matplotlib.axes._subplots.AxesSubplot at 0x1a22eb8d50>