K-Means Analysis

August 15, 2024

0.0.1 Executive Summary:

The goal of this project is to use un-supervised learning method to understand the differences among universities.

0.0.2 Method:

Seaborn package will be utilized to analyze the data, kmeans will be the un-supervised learning method.

[1]:
```
#import necessary packages
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from ydata_profiling import ProfileReport
import matplotlib.pyplot as plt
```

0.0.3 Step 0: import data and have a glimpse

[2]:
```
university_data = pd.read_csv("University_Data.csv")
```

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

[3]:
```
                University_Name Private    Apps  Accept  Enroll  Top10perc  \
0  Abilene Christian University     Yes  1660.0  1232.0     721       23.0
1            Adelphi University     Yes  2186.0  1924.0     512       16.0
2                Adrian College     Yes  1428.0  1097.0     336       22.0
3           Agnes Scott College     Yes   417.0   349.0     137       60.0
4     Alaska Pacific University     Yes   193.0   146.0      55       16.0

   Top25perc  F.Undergrad  P.Undergrad  Outstate  Room.Board  Books  Personal  \
0       52.0         2885          537    7440.0        3300    450    2200.0
1       29.0         2683         1227   12280.0        6450    750    1500.0
2       50.0         1036           99   11250.0        3750    400    1165.0
3       89.0          510           63   12960.0        5450    450     875.0
4       44.0          249          869    7560.0        4120    800    1500.0
```

```
   PhD  Terminal  S.F.Ratio  perc.alumni   Expend  Grad.Rate
0   70      78.0       18.1           12   7041.0       60.0
1   29      30.0       12.2           16  10527.0       56.0
2   53      66.0       12.9           30   8735.0       54.0
3   92      97.0        7.7           37  19016.0       59.0
4   76      72.0       11.9            2  10922.0       15.0
```

0.0.4 Meaning of each column:

University_Name: the name of the university
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

0.0.5 Step 1: profiling the data and see how much missing values in the dataset

[4]:
```
data_profile = ProfileReport(university_data, title = "University Data")
```

[5]:
```
data_profile.to_notebook_iframe()
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

<IPython.core.display.HTML object>
```

From the data profiling results, we see that there are missing values. We need to impute these missing values.

0.0.6 Step 2: Impute missing values by mean of each associated column.

The reason of imputing by mean value is that mean value will not skew the data distribution.

[6]:
```
X_numerical = university_data[['Apps', 'Accept', 'Enroll', 'Top10perc',
       'Top25perc', 'F.Undergrad', 'P.Undergrad', 'Outstate', 'Room.Board',
       'Books', 'Personal', 'PhD', 'Terminal', 'S.F.Ratio', 'perc.alumni',
       'Expend', 'Grad.Rate']]
```

[7]:
```
X_numerical.mean()
```

[7]:
```
Apps            2996.800258
Accept          2013.808786
Enroll           779.972973
Top10perc         27.546392
Top25perc         55.835917
F.Undergrad     3699.907336
P.Undergrad      855.298584
Outstate       10434.002581
Room.Board      4357.526384
Books            549.380952
Personal        1342.543928
PhD               72.660232
Terminal          79.734194
S.F.Ratio         14.084238
perc.alumni       22.743887
Expend          9655.750323
Grad.Rate         65.498708
dtype: float64
```

[8]:
```
# fill in the NA values

X_numerical = X_numerical.fillna(X_numerical.mean())
```

[9]:
```
#profiling again to check if there are any missing values after imputing

data_profile_2 = ProfileReport(X_numerical, title = "University Data")
```

[10]:
```
data_profile_2.to_notebook_iframe()
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

<IPython.core.display.HTML object>
```

There is no missing values anymore.

0.0.7 Step 3: normalize the data before using Kmeans.

The purpose of normalization are to: 1. Equal Weighting of Features 2. Prevention of Bias 3. Improved Convergence 4. Have meaningful Centroids

[11]:
```
X_numerical.std()
```

[11]:
```
Apps           3867.851104
Accept         2446.269714
Enroll          929.176190
Top10perc        17.637104
Top25perc        19.768381
F.Undergrad    4850.420531
P.Undergrad    1522.431887
Outstate       4018.131624
Room.Board     1096.696416
Books           165.105360
Personal        676.252557
PhD              16.328155
Terminal         14.704356
S.F.Ratio         3.953451
perc.alumni      12.391801
Expend         5212.088635
Grad.Rate        17.164212
dtype: float64
```

[12]:
```
normalized_data = (X_numerical - X_numerical.mean())/X_numerical.std()
```

0.0.8 Step 4: Start K-Means clustering and use Silhouette Score method to optimize the number of clusters

[13]:
```
range_n_clusters = [3, 4, 5, 6, 8, 9, 10, 11, 12]

for n_clusters in range_n_clusters:

    # Create a subplot with 1 row and 2 columns
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(18, 7)

    # The 1st subplot is the silhouette plot
    # The silhouette coefficient can range from -1, 1 but in this example all
    # lie within [-0.1, 1]
    ax1.set_xlim([-0.1, 1])
    # The (n_clusters+1)*10 is for inserting blank space between silhouette
    # plots of individual clusters, to demarcate them clearly.
    ax1.set_ylim([0, len(normalized_data) + (n_clusters + 1) * 10])
```

```
    # Initialize the clusterer with n_clusters value and a random generator
    # seed of 10 for reproducibility.
    clusterer = KMeans(n_clusters=n_clusters, random_state=10)
    cluster_labels = clusterer.fit_predict(normalized_data)

    # The silhouette_score gives the average value for all the samples.
    # This gives a perspective into the density and separation of the formed
    # clusters
    silhouette_avg = silhouette_score(normalized_data, cluster_labels)
    print(
        "For n_clusters =",
        n_clusters,
        "The average silhouette_score is :",
        silhouette_avg,
    )
```

```
For n_clusters = 3 The average silhouette_score is : 0.24095380083274182
For n_clusters = 4 The average silhouette_score is : 0.18240161738073857
For n_clusters = 5 The average silhouette_score is : 0.17359010325035037
For n_clusters = 6 The average silhouette_score is : 0.15972590024991634
For n_clusters = 8 The average silhouette_score is : 0.13589602395878855
For n_clusters = 9 The average silhouette_score is : 0.13870816466011637
For n_clusters = 10 The average silhouette_score is : 0.11722824669562198
For n_clusters = 11 The average silhouette_score is : 0.10673227365323146
For n_clusters = 12 The average silhouette_score is : 0.10641530904410898
```

```
/var/folders/w1/501_x3g12xxdmvpx2kvt2plc0000gn/T/ipykernel_96470/1358523001.py:3
3: UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown
  plt.show()
```

From the above k-means analysis and silhouette score, 3 clusters gives the best score.

0.0.9 Step 5: Using 3 clusters to re-cluster the data and assign labels to the original data

[14]:
```
clusterer = KMeans(n_clusters=3, random_state=10)
cluster_labels = clusterer.fit_predict(normalized_data)
```

[15]:
```
university_data['cluster_label'] = list(cluster_labels)
```

[16]:
```
university_data
```

[16]:
```
                    University_Name Private    Apps  Accept  Enroll  \
0      Abilene Christian University     Yes  1660.0  1232.0     721
1                Adelphi University     Yes  2186.0  1924.0     512
2                    Adrian College     Yes  1428.0  1097.0     336
```

```
3              Agnes Scott College     Yes    417.0   349.0     137
4         Alaska Pacific University     Yes    193.0   146.0      55
..                              ...     ...      ...     ...     ...
772         Worcester State College      No   2197.0  1515.0     543
773               Xavier University     Yes   1959.0  1805.0     695
774  Xavier University of Louisiana     Yes   2097.0  1915.0     695
775                 Yale University     Yes  10705.0  2453.0    1317
776    York College of Pennsylvania     Yes   2989.0  1855.0     691

     Top10perc  Top25perc  F.Undergrad  P.Undergrad  Outstate  Room.Board  \
0         23.0       52.0         2885          537    7440.0        3300
1         16.0       29.0         2683         1227   12280.0        6450
2         22.0       50.0         1036           99   11250.0        3750
3         60.0       89.0          510           63   12960.0        5450
4         16.0       44.0          249          869    7560.0        4120
..         ...        ...          ...          ...       ...         ...
772        4.0       26.0         3089         2029    6797.0        3900
773       24.0       47.0         2849         1107   11520.0        4960
774       34.0       61.0         2793          166    6900.0        4200
775       95.0       99.0         5217           83   19840.0        6510
776       28.0       63.0         2988         1726    4990.0        3560

     Books  Personal  PhD  Terminal  S.F.Ratio  perc.alumni   Expend  \
0      450    2200.0   70      78.0       18.1           12   7041.0
1      750    1500.0   29      30.0       12.2           16  10527.0
2      400    1165.0   53      66.0       12.9           30   8735.0
3      450     875.0   92      97.0        7.7           37  19016.0
4      800    1500.0   76      72.0       11.9            2  10922.0
..     ...       ...  ...       ...        ...          ...
772    500    1200.0   60       NaN       21.0           14   4469.0
773    600    1250.0   73      75.0       13.3           31   9189.0
774    617     781.0   67      75.0       14.4           20   8323.0
775    630    2115.0   96      96.0        5.8           49  40386.0
776    500    1250.0   75      75.0       18.1           28   4509.0

     Grad.Rate  cluster_label
0         60.0              0
1         56.0              0
2         54.0              0
3         59.0              1
4         15.0              0
..         ...            ...
772       40.0              0
773       83.0              0
774       49.0              0
775       99.0              1
776       99.0              0
```

```
[777 rows x 20 columns]
```

[17]:
```
university_data.columns
```

[17]:
```
Index(['University_Name', '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', 'cluster_label'],
      dtype='object')
```

0.0.10 Step 6: Create Heatmap by with the mean values for each cluster

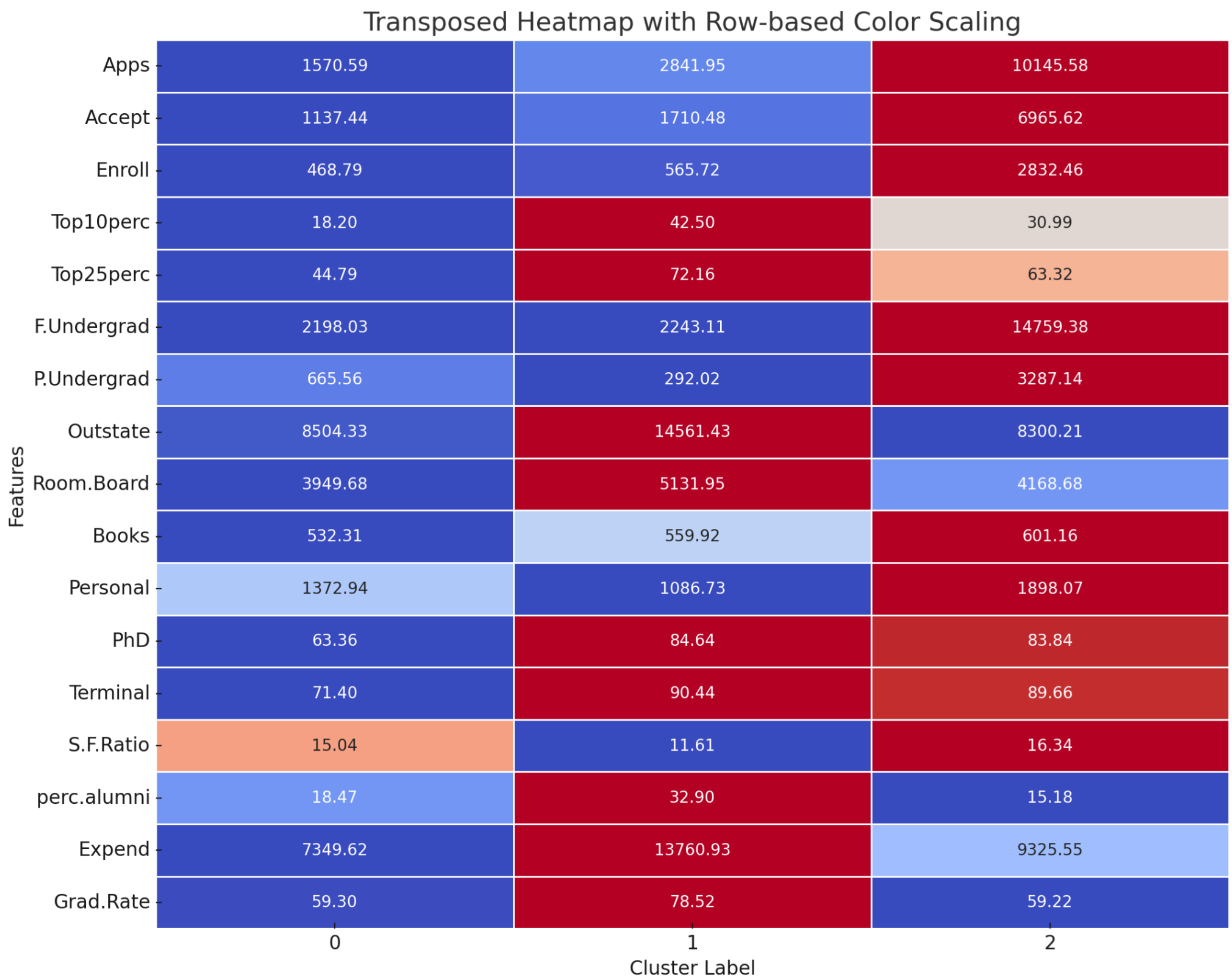
[43]:
```
# Normalizing the data by row for heatmap color scaling
clustered_university_data = np.asarray(university_data.
 ↪groupby('cluster_label')[['Apps', 'Accept', 'Enroll', 'Top10perc',
       'Top25perc', 'F.Undergrad', 'P.Undergrad', 'Outstate', 'Room.Board',
       'Books', 'Personal', 'PhD', 'Terminal', 'S.F.Ratio', 'perc.alumni',
       'Expend', 'Grad.Rate']].mean())

norm_clustered_data = (clustered_university_data.T - clustered_university_data.
 ↪T.min(axis=1)[:, np.newaxis]) / (
        clustered_university_data.T.max(axis=1) - clustered_university_data.T.
 ↪min(axis=1))[:, np.newaxis]

# Plotting the heatmap with the new color scheme
plt.figure(figsize=(12, 10))
sns.heatmap(norm_clustered_data, annot=clustered_university_data.T,␣
 ↪cmap='coolwarm', fmt=".2f", cbar=False,
            vmin=0, vmax=1, linewidths=.5)

plt.title('Transposed Heatmap with Row-based Color Scaling')
plt.xlabel('Cluster Label')
plt.ylabel('Features')
plt.show()
```

```
/var/folders/w1/501_x3g12xxdmvpx2kvt2plc0000gn/T/ipykernel_96470/3552015984.py:1
8: UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown
  plt.show()
```

[43]: <IPython.core.display.Image object>

0.1 Cluster 0:

Higher Values:


Transposed Heatmap with Row-based Color Scaling
Apps
Accept
Enroll
Top10perc
Top25perc
F.Undergrad
P.Undergrad
Outstate
Room.Board
Books
Personal
PhD
Terminal
S.F.Ratio
perc.alumni
Expend
Grad.Rate
Features
0
1
2
Cluster Label
1570.59
2841.95
10145.58
1137.44
1710.48
6965.62
468.79
565.72
2832.46
18.20
42.50
30.99
44.79
72.16
63.32
2198.03
2243.11
14759.38
665.56
292.02
3287.14
8504.33
14561.43
8300.21
3949.68
5131.95
4168.68
532.31
559.92
601.16
1372.94
1086.73
1898.07
63.36
84.64
83.84
71.40
90.44
89.66
15.04
11.61
16.34
18.47
32.90
15.18
7349.62
13760.93
9325.55
59.30
78.52
59.22

Applications: This cluster has the highest average number of applications, indicating that universities in this cluster tend to attract a large number of applicants.

Expenditure: The spending per student is relatively high, which may correlate with more resources or facilities available to students. Room & Board, Books, Personal Expenses: Costs associated with attending these universities (room and board, books, personal expenses) are generally high, indicating potentially higher living costs or more expensive university options. #### Lower Values: ##### Graduation Rate: The graduation rate is lower compared to the other clusters, suggesting that students in this cluster might face more challenges in completing their programs. ##### Student-Faculty Ratio: This cluster has a relatively higher student-to-faculty ratio, which might imply larger class sizes or fewer faculty resources available to students.

0.2 Cluster 1:

Higher Values:

Top 10% & Top 25% of High School Class: Universities in this cluster have the highest percentage of students coming from the top 10% and top 25% of their high school classes, indicating that these institutions are more selective or attract higher-achieving students.

PhD & Terminal Degrees: This cluster shows the highest percentage of faculty with PhD or terminal degrees, which often correlates with higher academic prestige or research focus.

Graduation Rate: The graduation rate is relatively high, suggesting that students at these universities are more likely to complete their degrees.

Lower Values:

Applications, Acceptances, and Enrollments: This cluster has lower values in terms of applications, acceptances, and enrollments, which might suggest smaller or more specialized institutions.

Expenditure: Spending per student is lower, potentially indicating fewer resources or less emphasis on physical infrastructure.

0.3 Cluster 2:

Higher Values:

Student-Faculty Ratio: This cluster has the lowest student-to-faculty ratio, indicating smaller class sizes and potentially more personalized attention for students.

Percentage of Alumni Donations: This cluster has the highest percentage of alumni donations, which could reflect strong alumni engagement and satisfaction with their education.

Lower Values:

Top 10% & Top 25% of High School Class: The percentage of students from the top 10% and top 25% of their high school classes is lower in this cluster, suggesting a less selective admission process.

Graduation Rate: The graduation rate is relatively low, which might indicate challenges in student retention or program completion.

0.4 Final Summary:

Cluster 0 appears to represent larger, potentially less selective universities with higher operating costs but lower student success metrics like graduation rates.
Cluster 1 seems to consist of more selective, academically prestigious institutions that attract high-achieving students but might be smaller in size.
Cluster 2 might represent smaller, community-focused universities with strong student-faculty engagement and active alumni networks, though they may face challenges in attracting high-performing students and achieving high graduation rates.