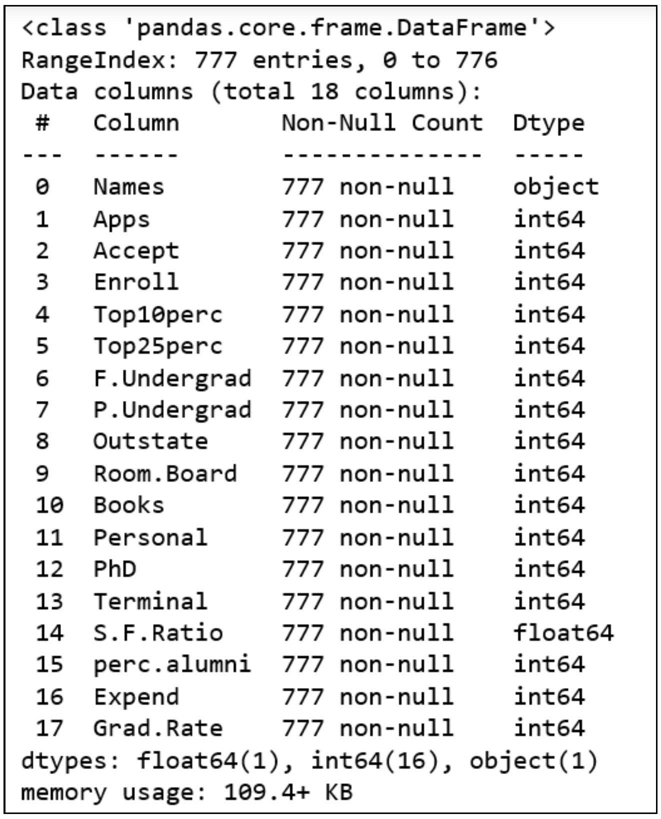
***Problem 2***

***The dataset Education - Post 12th Standard.csv is a dataset that contains the names of various colleges. This particular case study is based on various parameters of various institutions. You are expected to do Principal Component Analysis for this case study according to the instructions given in the following rubric. The data dictionary of the 'Education - Post 12th Standard.csv' can be found in the following file: Data Dictionary.xlsx.***

**2.1) Perform Exploratory Data Analysis [both univariate and multivariate analysis to be performed]. The inferences drawn from this should be properly documented.**

The dataset has (777, 18) Rows and Columns

It has 18 Columns, Names column is of type Object we can remove this for Analysis.

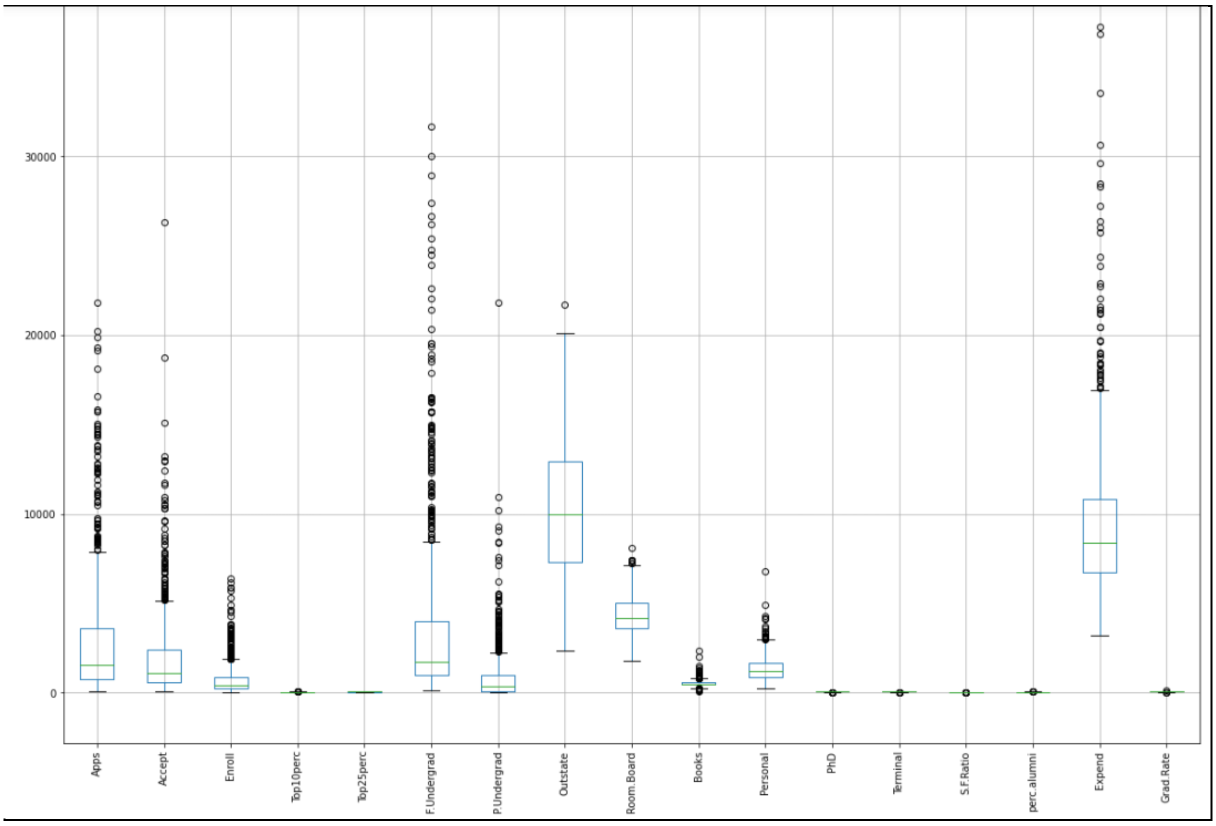


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 777 entries, 0 to 776
Data columns (total 18 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Names        777 non-null    object
 1   Apps         777 non-null    int64
 2   Accept       777 non-null    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: 109.4+ KB
```

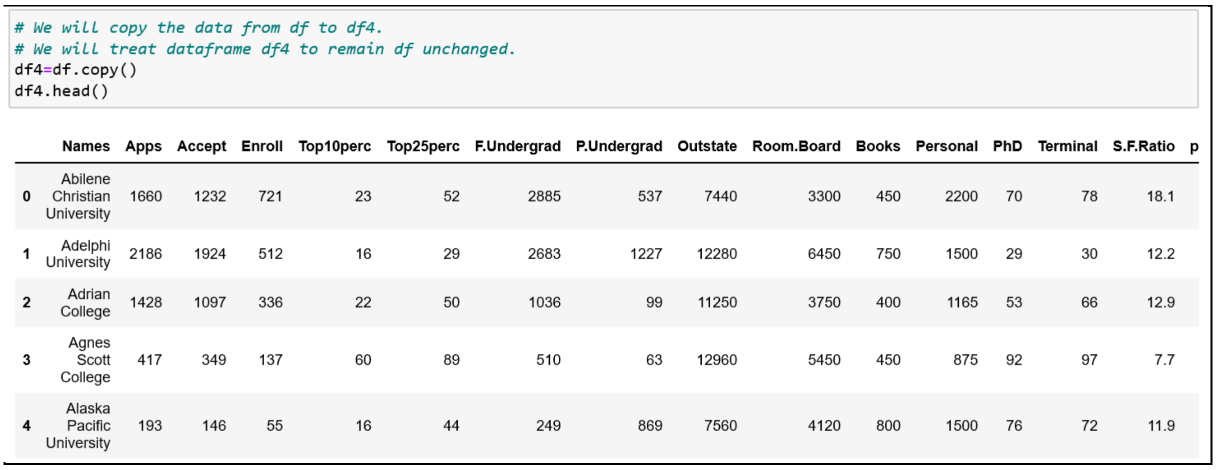
No Null Values

```
Names          0
Apps           0
Accept         0
Enroll         0
Top10perc      0
Top25perc      0
F.Undergrad    0
P.Undergrad    0
Outstate       0
Room.Board     0
Books          0
Personal       0
PhD            0
Terminal       0
S.F.Ratio      0
perc.alumni    0
Expend         0
Grad.Rate      0
dtype: int64
```

Check for Outlier:



Almost Every variable has outliers.



```python
# We will copy the data from df to df4.
# We will treat dataframe df4 to remain df unchanged.
df4=df.copy()
df4.head()
```

| | Names | Apps | Accept | Enroll | Top10perc | Top25perc | F.Undergrad | P.Undergrad | Outstate | Room.Board | Books | Personal | PhD | Terminal | S.F.Ratio | p |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Abilene Christian University | 1660 | 1232 | 721 | 23 | 52 | 2885 | 537 | 7440 | 3300 | 450 | 2200 | 70 | 78 | 18.1 | |
| **1** | Adelphi University | 2186 | 1924 | 512 | 16 | 29 | 2683 | 1227 | 12280 | 6450 | 750 | 1500 | 29 | 30 | 12.2 | |
| **2** | Adrian College | 1428 | 1097 | 336 | 22 | 50 | 1036 | 99 | 11250 | 3750 | 400 | 1165 | 53 | 66 | 12.9 | |
| **3** | Agnes Scott College | 417 | 349 | 137 | 60 | 89 | 510 | 63 | 12960 | 5450 | 450 | 875 | 92 | 97 | 7.7 | |
| **4** | Alaska Pacific University | 193 | 146 | 55 | 16 | 44 | 249 | 869 | 7560 | 4120 | 800 | 1500 | 76 | 72 | 11.9 | |

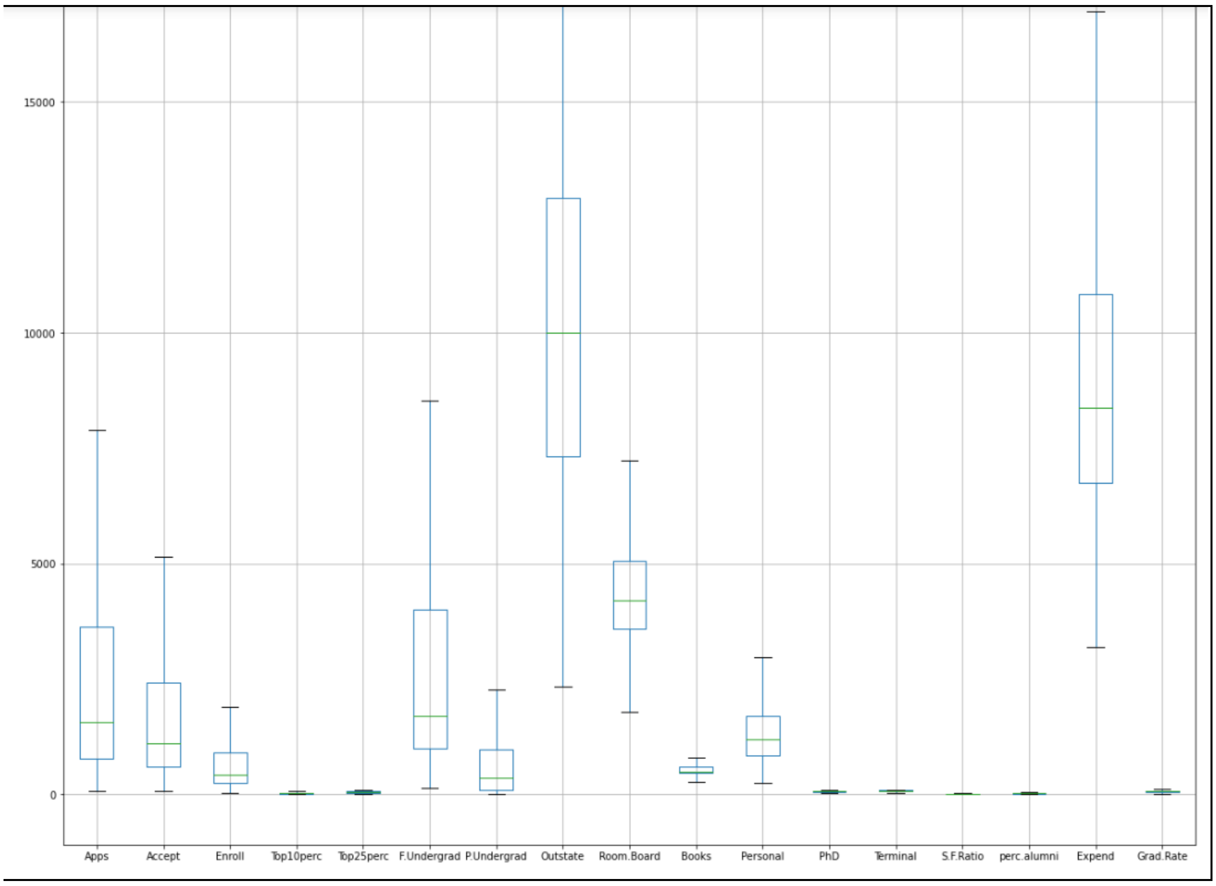
We will Define function to Remove Outliers, In this we are calculating Q1,Q3,Inter Quartile range

This function is Returning lower_range and Upper range values.

Outlier Treatment by using user defined Function "remove_outlier".

In lrapps we are saving result of lower range and in urapps we are saving result of upper range

After Treatment The boxplot will look like


15000
10000
5000
0
Apps
Accept
Enroll
Top10perc
Top25perc
F.Undergrad
P.Undergrad
Outstate
Room.Board
Books
Personal
PhD
Terminal
S.F.Ratio
perc.alumni
Expend
Grad.Rate

## ***2.1) Perform Exploratory Data Analysis [both univariate and multivariate analysis to be performed]. The inferences drawn from this should be properly documented.***

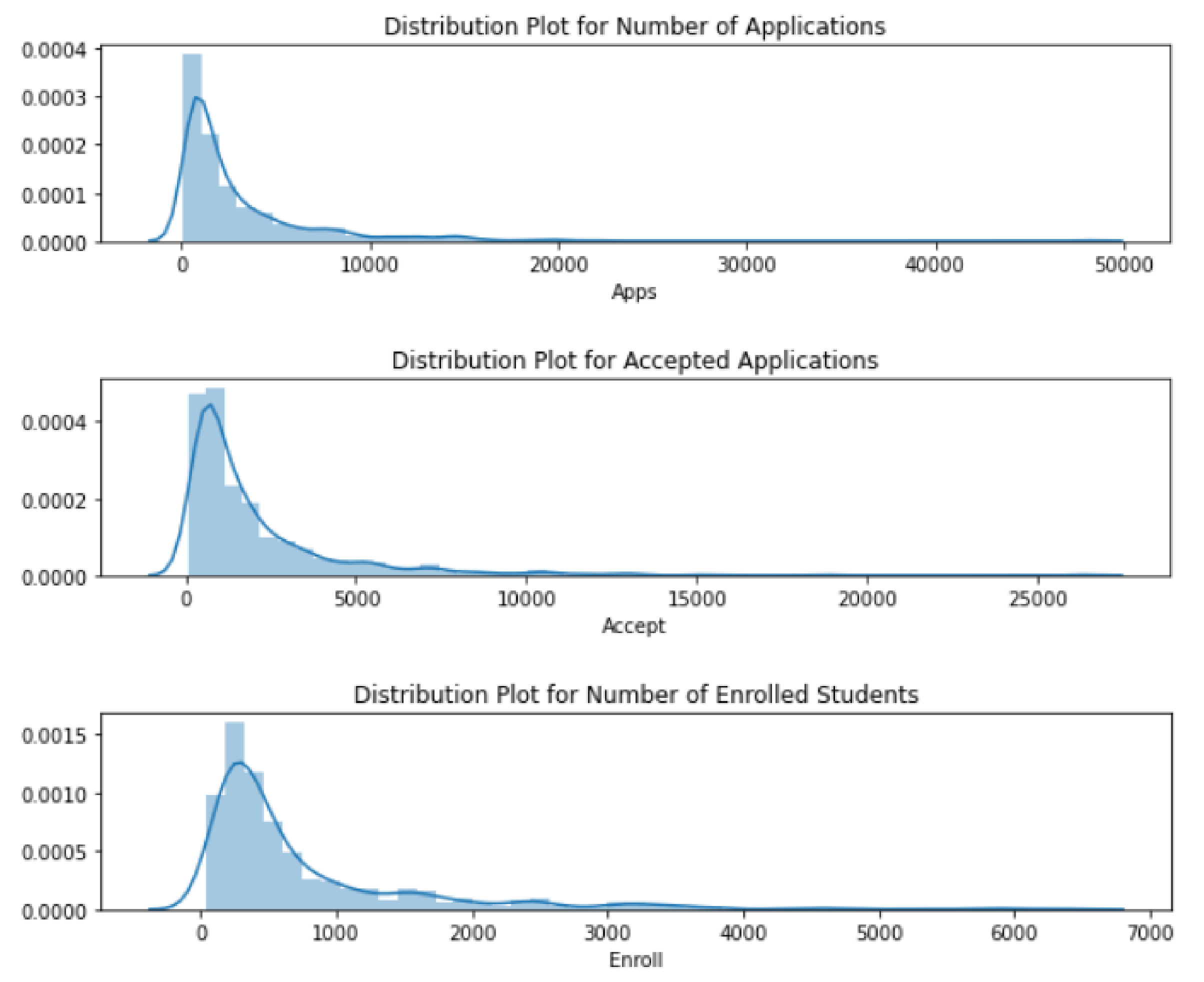
Lets plot a distplot to see how the data is distributed.

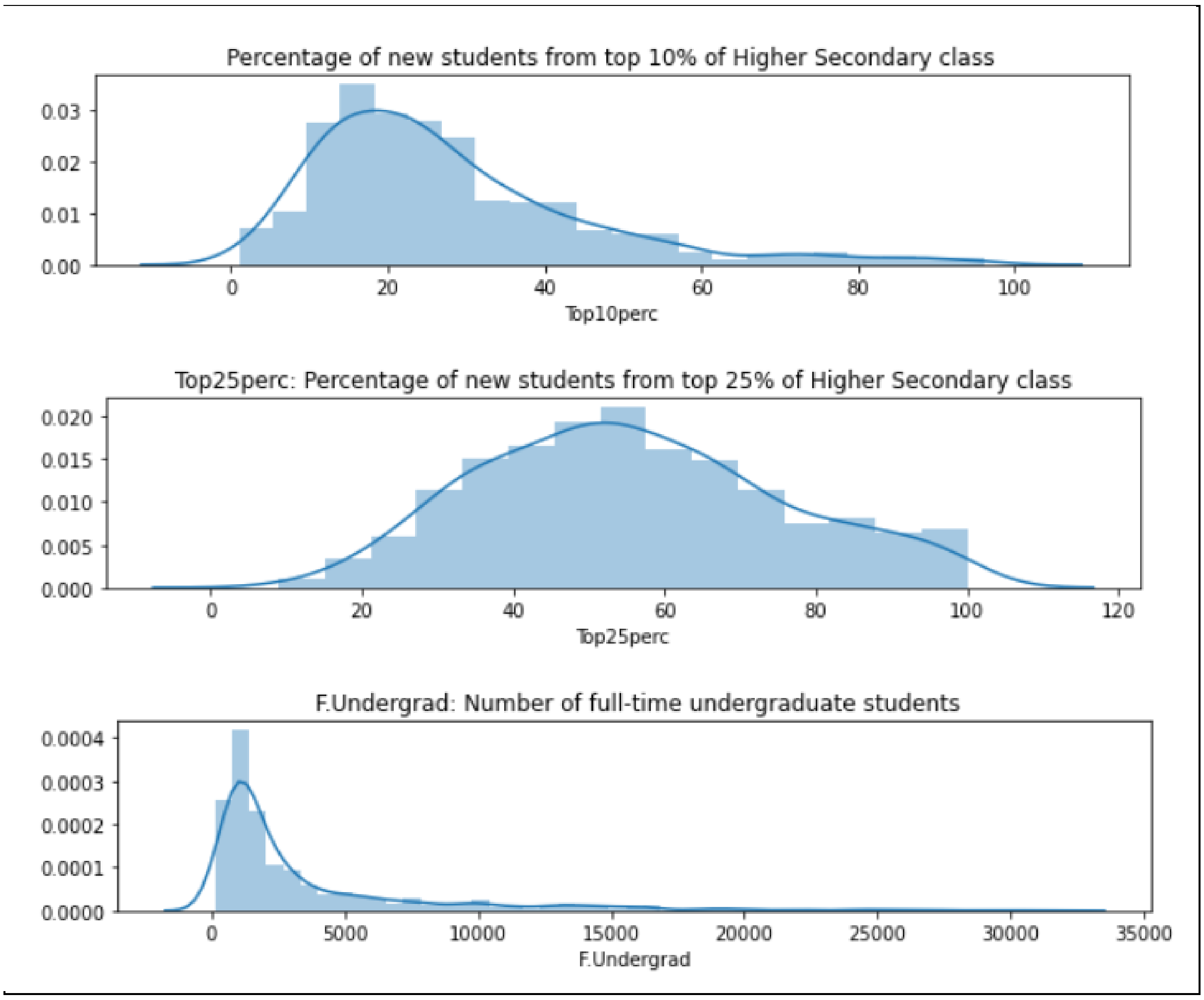
Creating Univariate Distribution plot to analyse each variable thouroughly, we will get outliers then

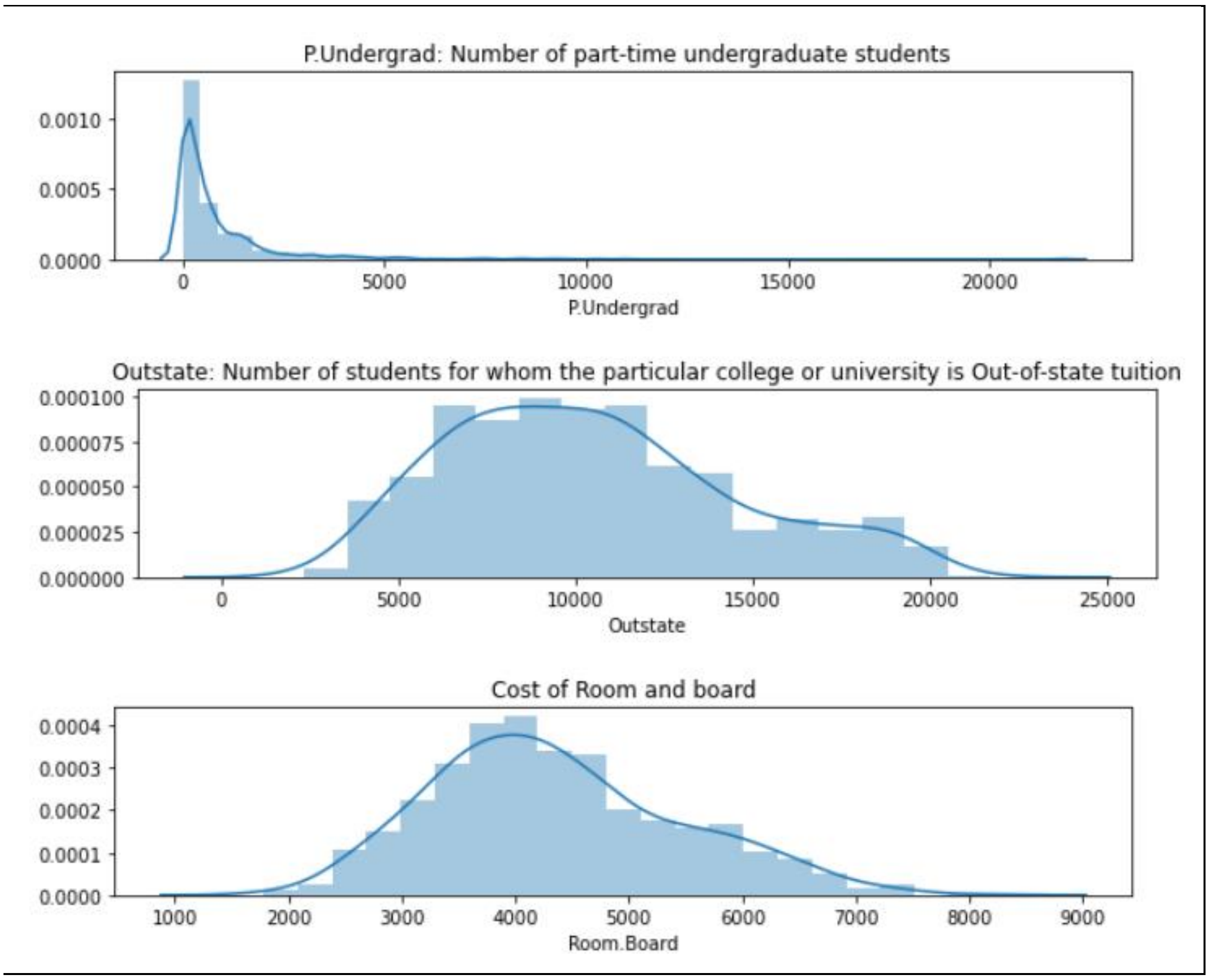
Shape of plot, How is the distribution.

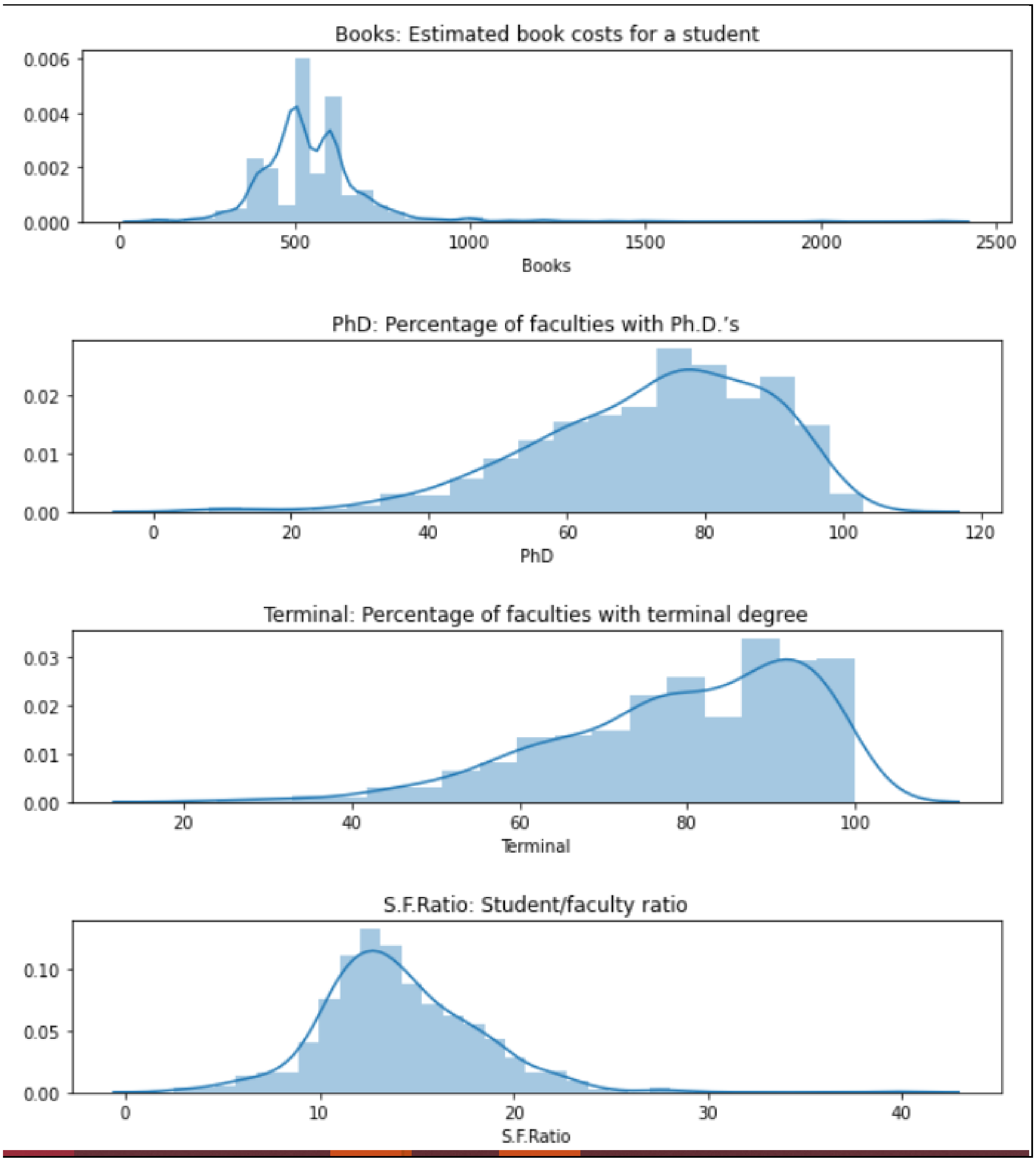
by plotting pairplot,Heatmap we can see correlation between two different variables.

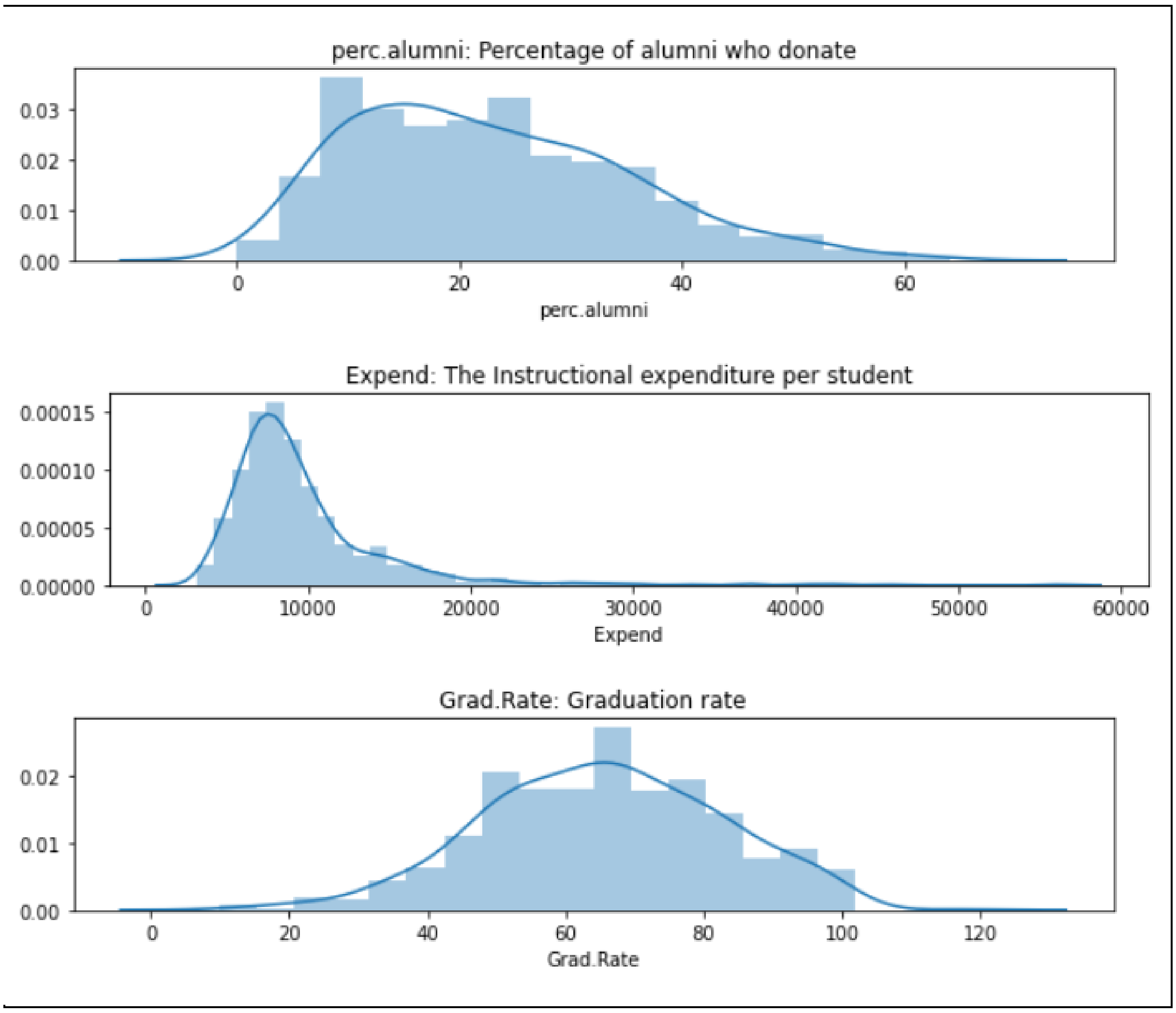
We will get the initial idea of distribution here.




Percentage of new students from top 10% of Higher Secondary class
0.00
0.01
0.02
0.03
0
20
40
60
80
100
Top10perc
Top25perc: Percentage of new students from top 25% of Higher Secondary class
0.000
0.005
0.010
0.015
0.020
0
20
40
60
80
100
120
Top25perc
F.Undergrad: Number of full-time undergraduate students
0.0000
0.0001
0.0002
0.0003
0.0004
0
5000
10000
15000
20000
25000
30000
35000
F.Undergrad

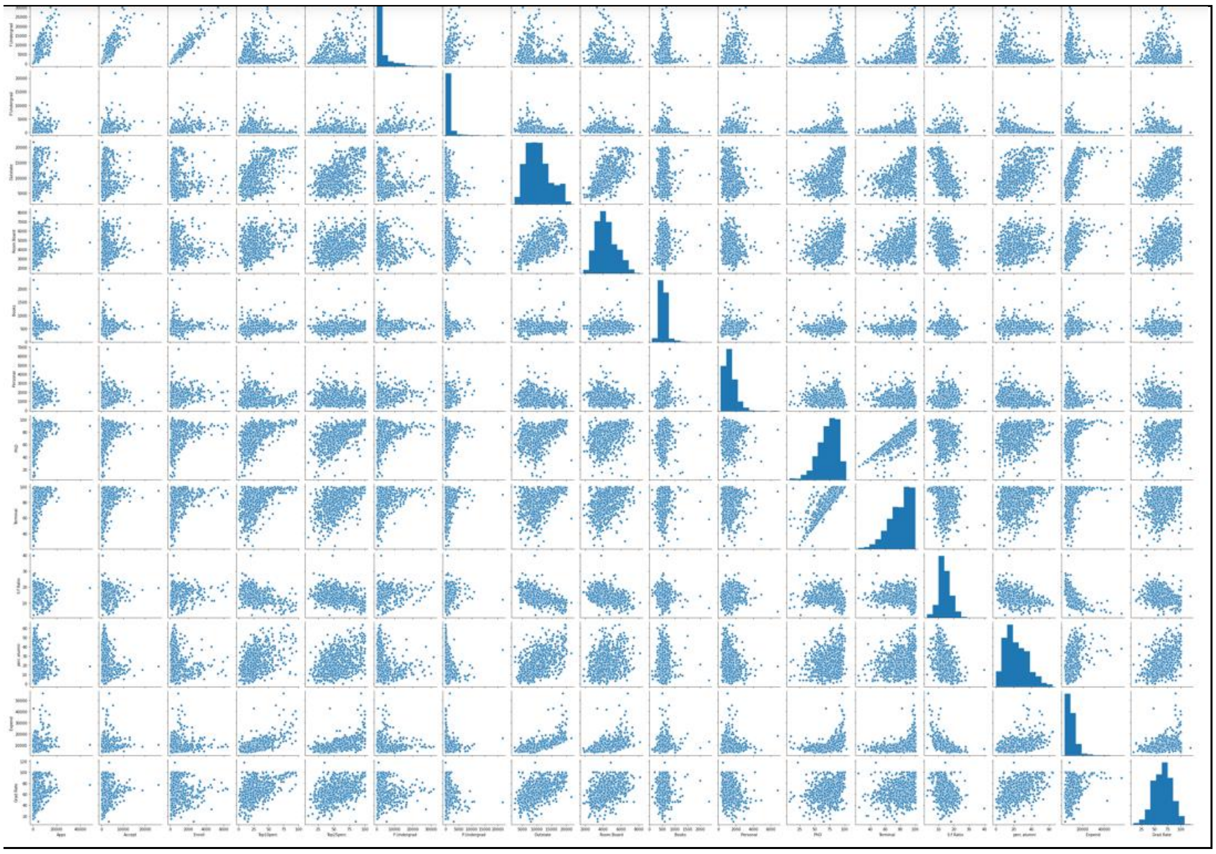

P.Undergrad: Number of part-time undergraduate students
0.0010
0.0005
0.0000
0
5000
10000
15000
20000
P.Undergrad
Outstate: Number of students for whom the particular college or university is Out-of-state tuition
0.000100
0.000075
0.000050
0.000025
0.000000
0
5000
10000
15000
20000
25000
Outstate
Cost of Room and board
0.0004
0.0003
0.0002
0.0001
0.0000
1000
2000
3000
4000
5000
6000
7000
8000
9000
Room.Board


Books: Estimated book costs for a student
0.006
0.004
0.002
0.000
0
500
1000
1500
2000
2500
Books
PhD: Percentage of faculties with Ph.D.'s
0.02
0.01
0.00
0
20
40
60
80
100
120
PhD
Terminal: Percentage of faculties with terminal degree
0.03
0.02
0.01
0.00
20
40
60
80
100
Terminal
S.F.Ratio: Student/faculty ratio
0.10
0.05
0.00
0
10
20
30
40
S.F.Ratio

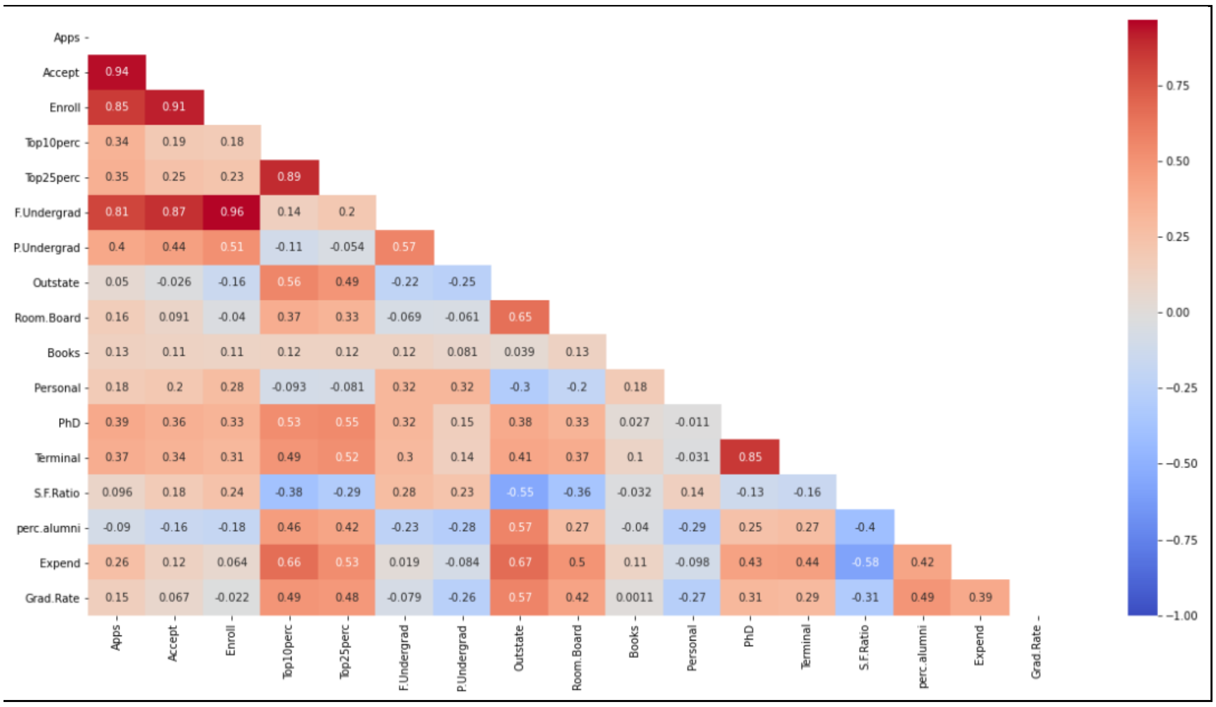

perc.alumni: Percentage of alumni who donate
0.03
0.02
0.01
0.00
0
20
40
60
perc.alumni
Expend: The Instructional expenditure per student
0.00015
0.00010
0.00005
0.00000
0
10000
20000
30000
40000
50000
60000
Expend
Grad.Rate: Graduation rate
0.02
0.01
0.00
0
20
40
60
80
100
120
Grad.Rate

***Creating Bivariate Distribution  plot***

*Plot Pairplot*



*Plot Heatmap, to see Correlation between variables  Lets plot correlation Heatmap to get better visualization of behaviour of Multiple variables amongst each other*



We can group all the distributions into Left Skewed,Right Skewed and Normal Distribution as follows by looking at Distribution plots.

-->Left Skewed: Personal
PhD
Terminal

--> Right Skewed: Apps
Accept
Enroll
Top10perc
F.Undergrad
P.Undergrad
Books
Expend
S.F.Ratio

-->Normal Distribution Top25perc
Outstate
Room.Board
perc.alumni
Grad.Rate

Some Insights that can be drawn from Above Heatmap.

-->From the above heatmap,we can see Strong Positive Correlation between Acceptance and Number of Applications,Enrollment and No of Applications,No.of Full time Under graduates and No of Applications,

-->Enrollment and No of Acceptance,No.of Full time Under graduates and No of Acceptance, No.of part time Under graduates and No of Acceptance, Acceptance is directly proportional to PhD candidates.

-->It seems Obvious that Student to Faculty ratio is inversely related to Instructional Expenditure, Expenditure is increasing when S/F ratio is decreased

-->If top 10% students and top 25% students count is High in college it seems that Graduation Rate is High, PhD rate is High. It is possible that These top students might end up by doing PhD It is interesting if we deeper dive into relation of These students with PhD graduates.

-->Due to higher percentage of outstate student, Cost of Room and boarding is also High or vice versa.

-->College having Faculties with terminal degrees, seems to have higher Instructional expenditure of students

-->Outstate students scored higher percentage in case of top 10 percent and 25 percent students

***2.2) Scale the variables and write the inference for using the type of scaling function for this case study.***

Often the variables of the data set are of different scales i.e. one variable is in Currency and other is Headcount. in our data set Room and Boarding Fees is having values in thousands and Top 10 percent,25 Percent just two digits. Since the data in these variables are of different scales, it is tough to compare these variables.

Feature scaling (also known as data normalization) is the method used to standardize the range of features of data. Since, the range of values of data may vary widely, it becomes a necessary step in data preprocessing to draw some fruitful Insights.

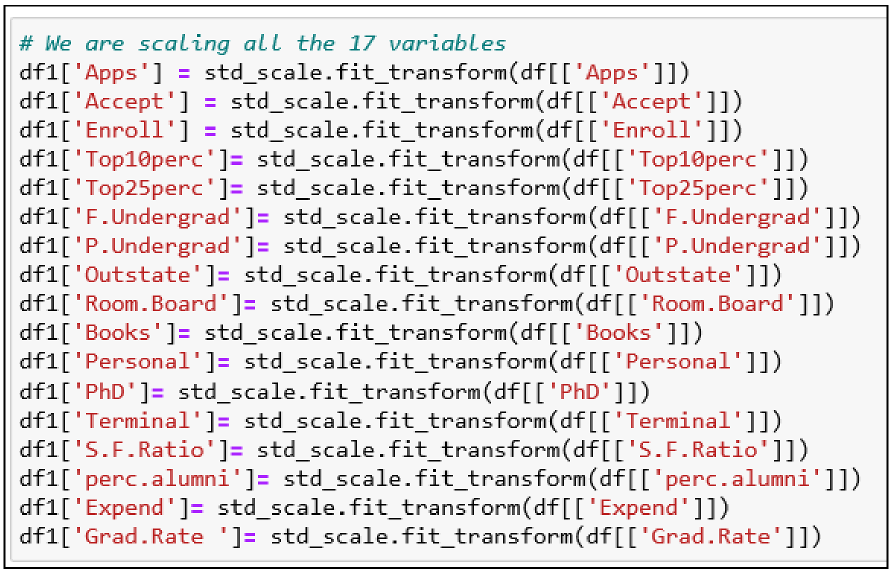
In this method, we convert variables with different scales of measurements into a single scale.

StandardScaler function normalizes the data using the formula (x-mean)/standard deviation.

We will be doing this only for the numerical variables and there are 17 numerical variables and one is nominal variable

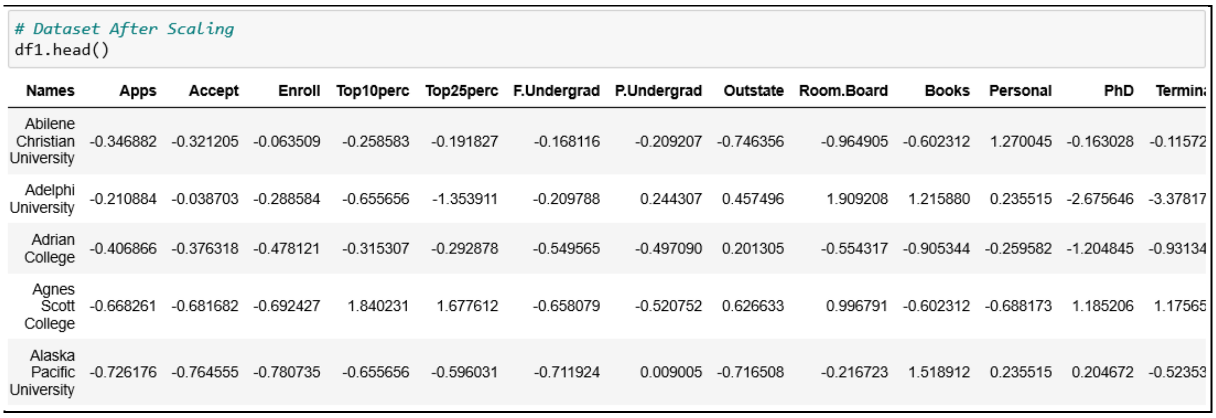
StandardScaler() function Scales the data. and it is taken from sklearn.preprocessing library, Essentially returns the z-scores of every attribute

Scaling all the 17 variables



```
# We are scaling all the 17 variables
df1['Apps'] = std_scale.fit_transform(df[['Apps']])
df1['Accept'] = std_scale.fit_transform(df[['Accept']])
df1['Enroll'] = std_scale.fit_transform(df[['Enroll']])
df1['Top10perc']= std_scale.fit_transform(df[['Top10perc']])
df1['Top25perc']= std_scale.fit_transform(df[['Top25perc']])
df1['F.Undergrad']= std_scale.fit_transform(df[['F.Undergrad']])
df1['P.Undergrad']= std_scale.fit_transform(df[['P.Undergrad']])
df1['Outstate']= std_scale.fit_transform(df[['Outstate']])
df1['Room.Board']= std_scale.fit_transform(df[['Room.Board']])
df1['Books']= std_scale.fit_transform(df[['Books']])
df1['Personal']= std_scale.fit_transform(df[['Personal']])
df1['PhD']= std_scale.fit_transform(df[['PhD']])
df1['Terminal']= std_scale.fit_transform(df[['Terminal']])
df1['S.F.Ratio']= std_scale.fit_transform(df[['S.F.Ratio']])
df1['perc.alumni']= std_scale.fit_transform(df[['perc.alumni']])
df1['Expend']= std_scale.fit_transform(df[['Expend']])
df1['Grad.Rate ']= std_scale.fit_transform(df[['Grad.Rate']])
```

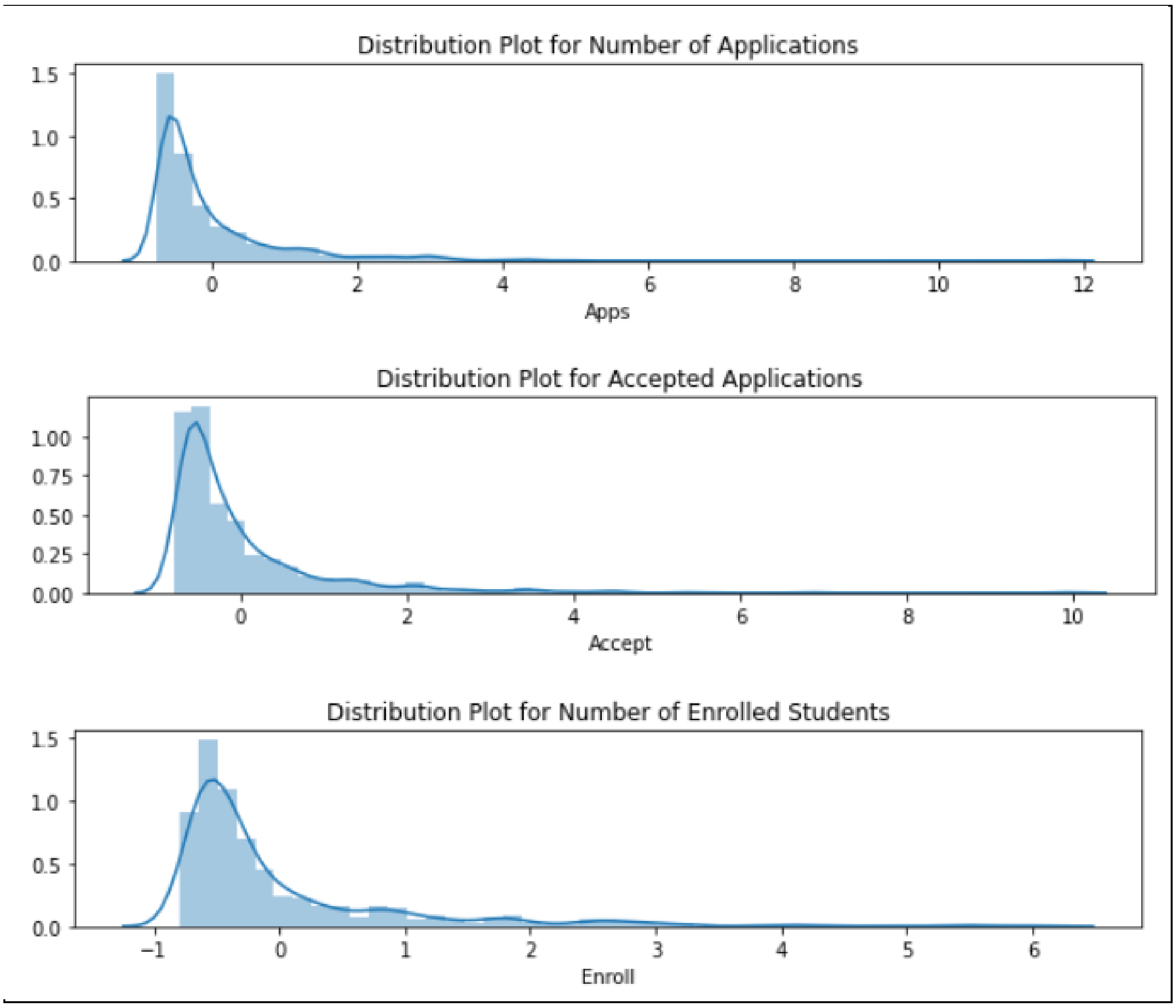
We will find all the Scaled values in dataframe 1, That is df1

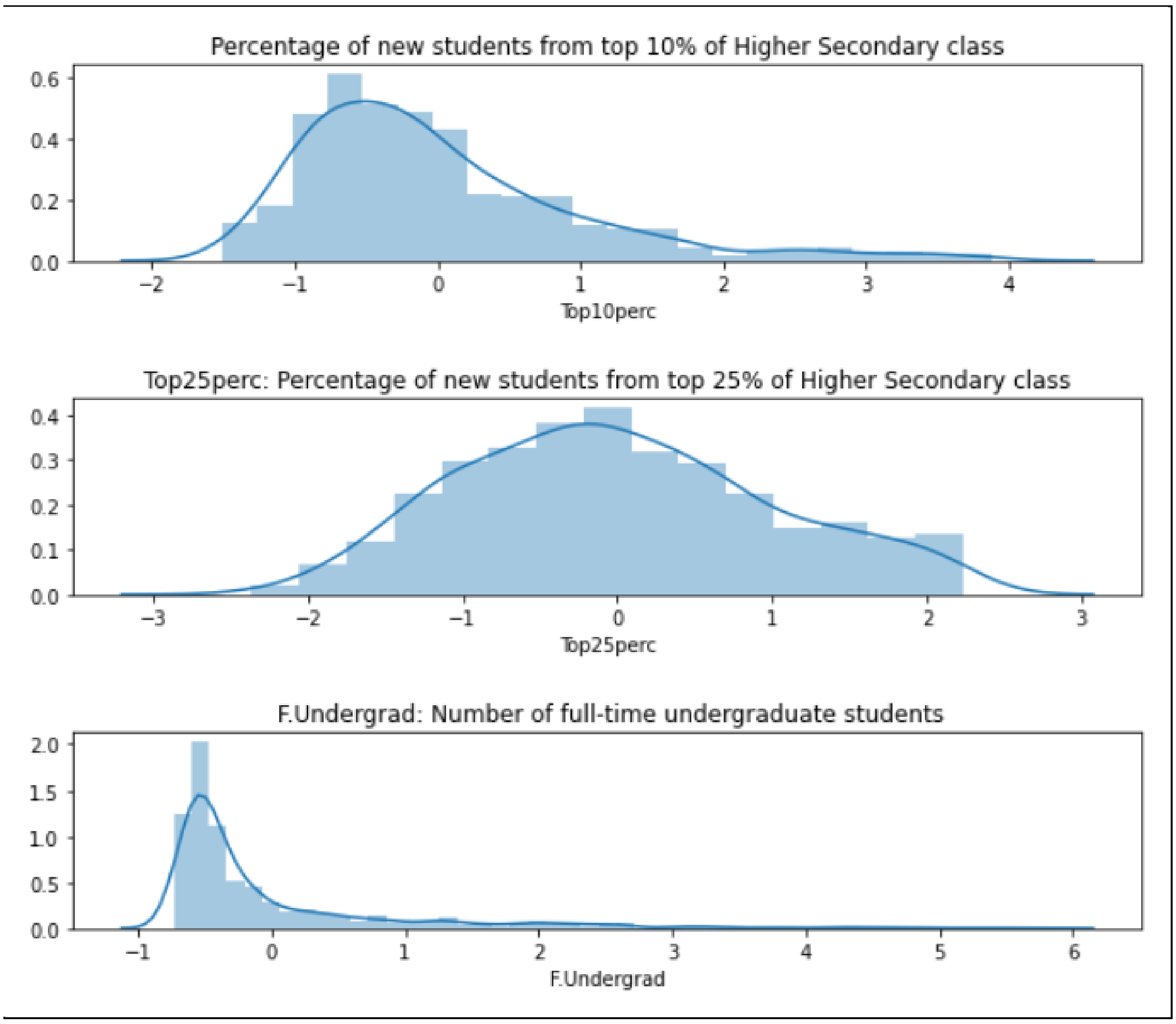


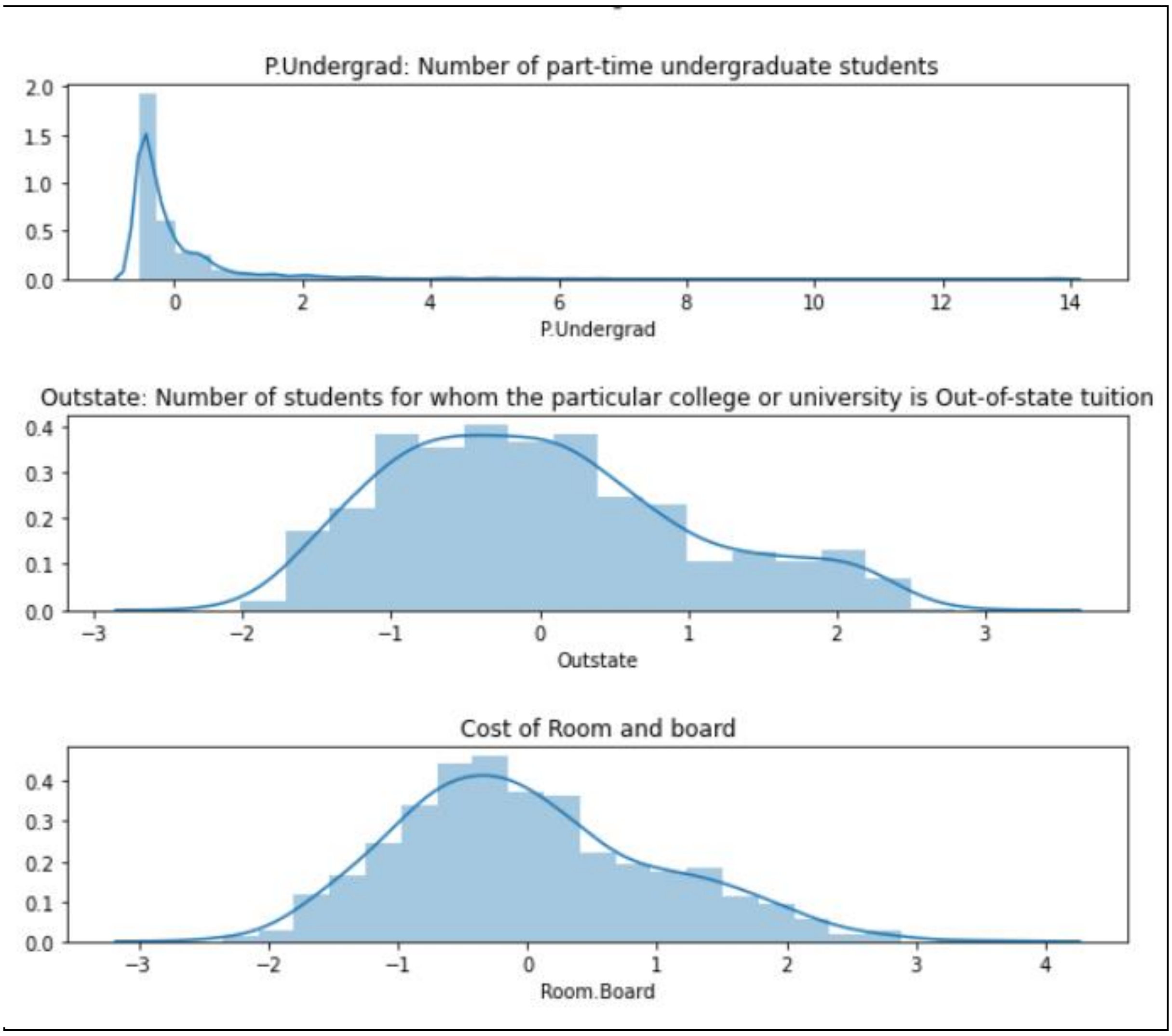
```
# Dataset After Scaling
df1.head()
```

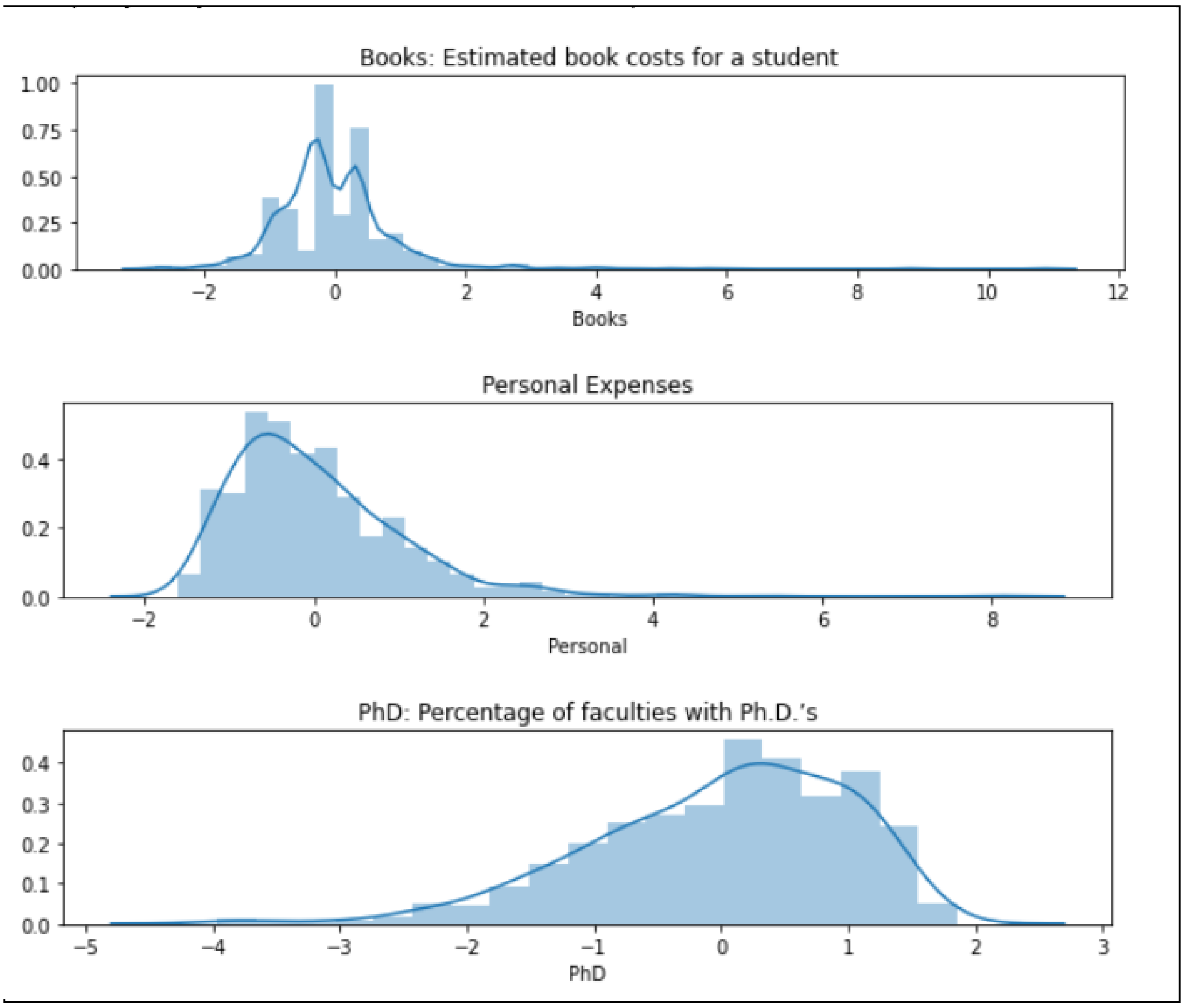
| Names | Apps | Accept | Enroll | Top10perc | Top25perc | F.Undergrad | P.Undergrad | Outstate | Room.Board | Books | Personal | PhD | Termin |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Abilene Christian University | -0.346882 | -0.321205 | -0.063509 | -0.258583 | -0.191827 | -0.168116 | -0.209207 | -0.746356 | -0.964905 | -0.602312 | 1.270045 | -0.163028 | -0.11572 |
| Adelphi University | -0.210884 | -0.038703 | -0.288584 | -0.655656 | -1.353911 | -0.209788 | 0.244307 | 0.457496 | 1.909208 | 1.215880 | 0.235515 | -2.675646 | -3.37817 |
| Adrian College | -0.406866 | -0.376318 | -0.478121 | -0.315307 | -0.292878 | -0.549565 | -0.497090 | 0.201305 | -0.554317 | -0.905344 | -0.259582 | -1.204845 | -0.93134 |
| Agnes Scott College | -0.668261 | -0.681682 | -0.692427 | 1.840231 | 1.677612 | -0.658079 | -0.520752 | 0.626633 | 0.996791 | -0.602312 | -0.688173 | 1.185206 | 1.17565 |
| Alaska Pacific University | -0.726176 | -0.764555 | -0.780735 | -0.655656 | -0.596031 | -0.711924 | 0.009005 | -0.716508 | -0.216723 | 1.518912 | 0.235515 | 0.204672 | -0.52353 |

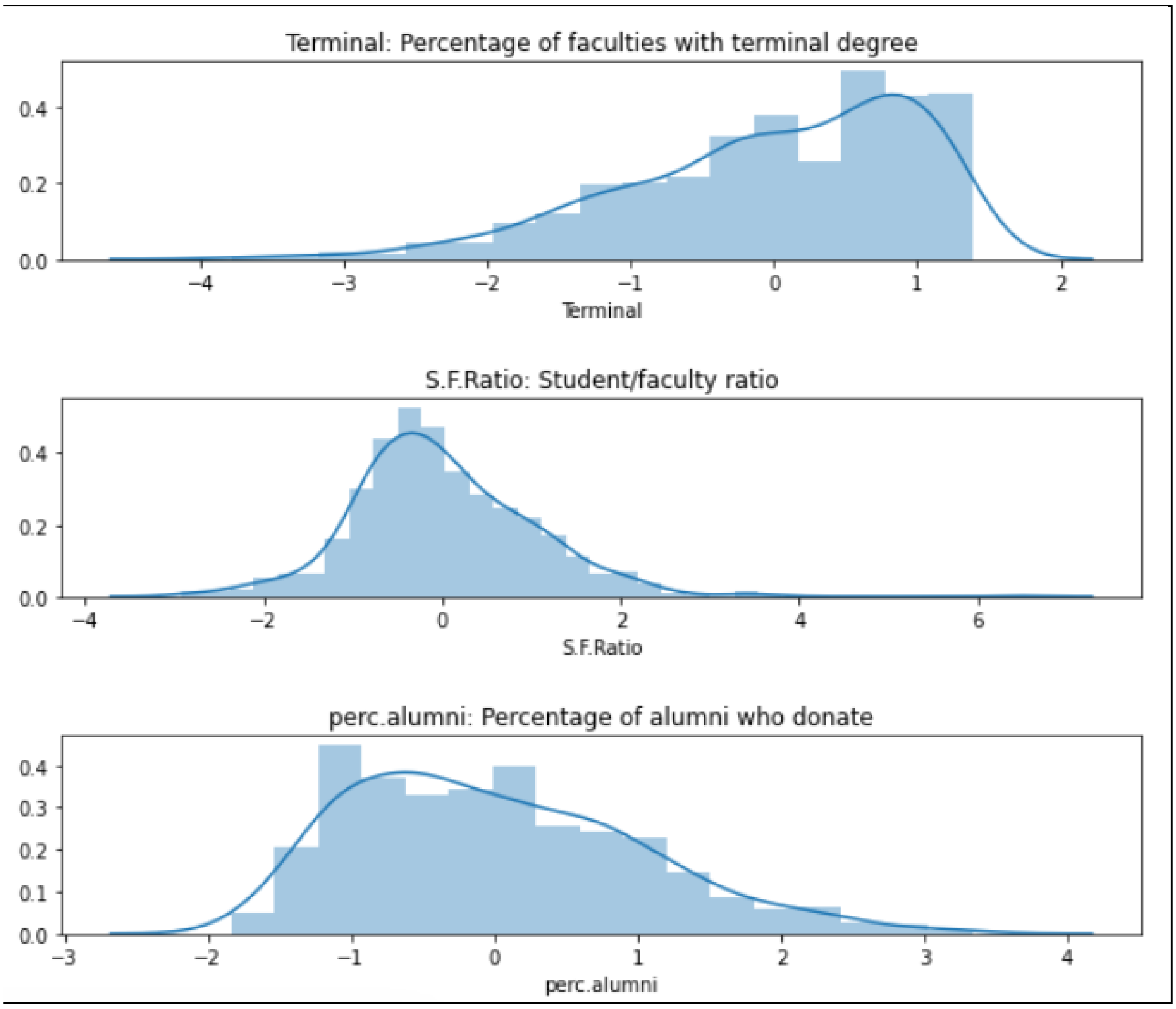
Plot After Scaling,Creating Univariate Distribution  plot

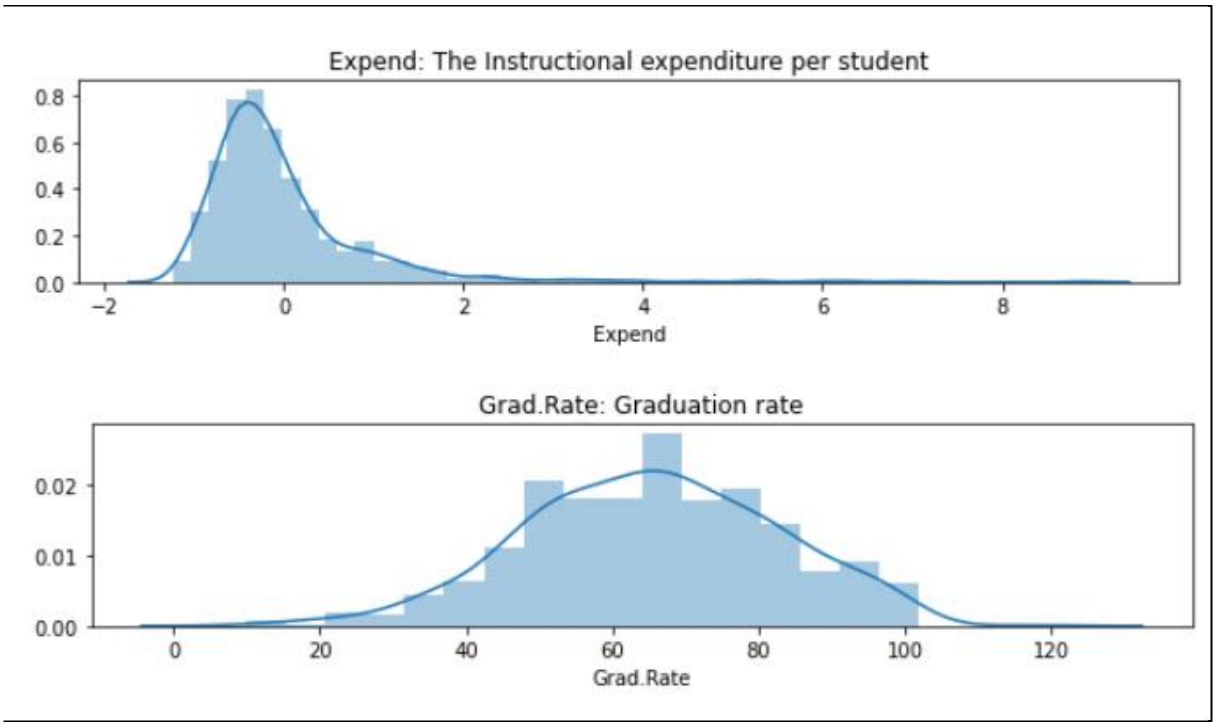



Percentage of new students from top 10% of Higher Secondary class
Top10perc
Top25perc: Percentage of new students from top 25% of Higher Secondary class
Top25perc
F.Undergrad: Number of full-time undergraduate students
F.Undergrad


P.Undergrad: Number of part-time undergraduate students
P.Undergrad
Outstate: Number of students for whom the particular college or university is Out-of-state tuition
Outstate
Cost of Room and board
Room.Board


Books: Estimated book costs for a student
Books
Personal Expenses
Personal
PhD: Percentage of faculties with Ph.D.'s
PhD


Terminal: Percentage of faculties with terminal degree
Terminal
S.F.Ratio: Student/faculty ratio
S.F.Ratio
perc.alumni: Percentage of alumni who donate
perc.alumni



So from Above Graphs we can make Inferences, And we can divide variables into Three Category as similar after Scaling.

Distributon is Right Skew: Apps Accept
Enroll
Top10perc
F.Undergrad
P.Undergrad
Books
Expend
S.F.Ratio

Distributon is Left Skew: Personal
PhD
Terminal

Distributon is Normal: Top25perc
Outstate
Room.Board perc.alumni
Grad.Rate

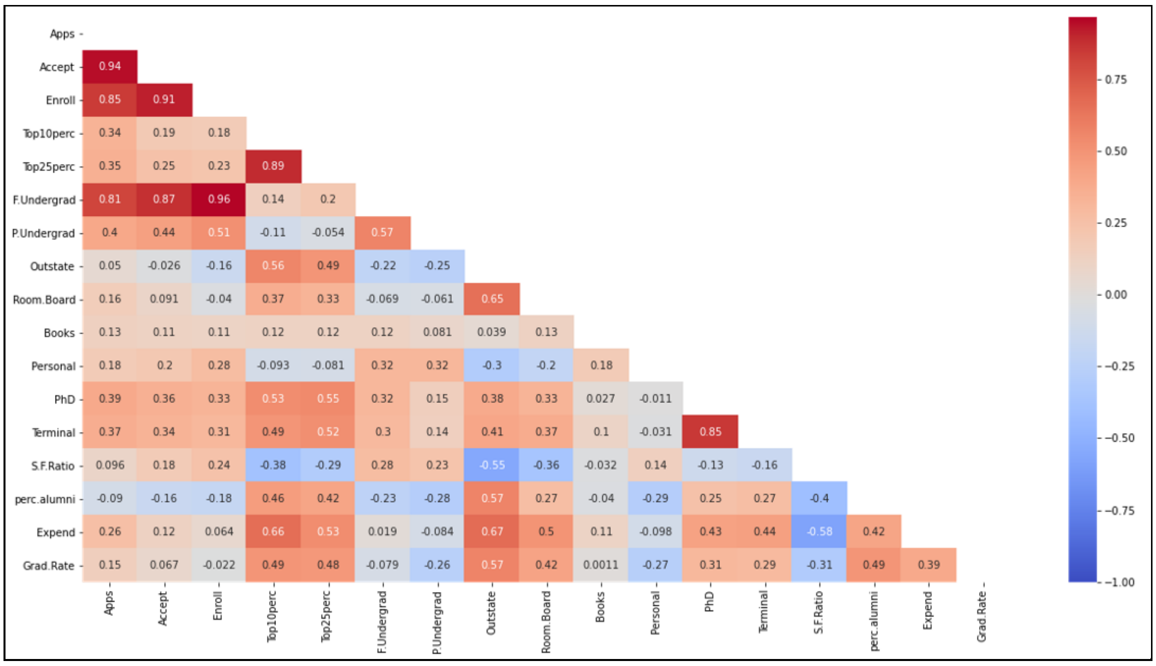
-->We can see the Distribution is same before and after the Scaling. Scaling only changes scale, by calculating Z scores, The shape of graph will remain same.

-->The skewed data needs Outlier Remove treatment. that we will do in subsequent steps.

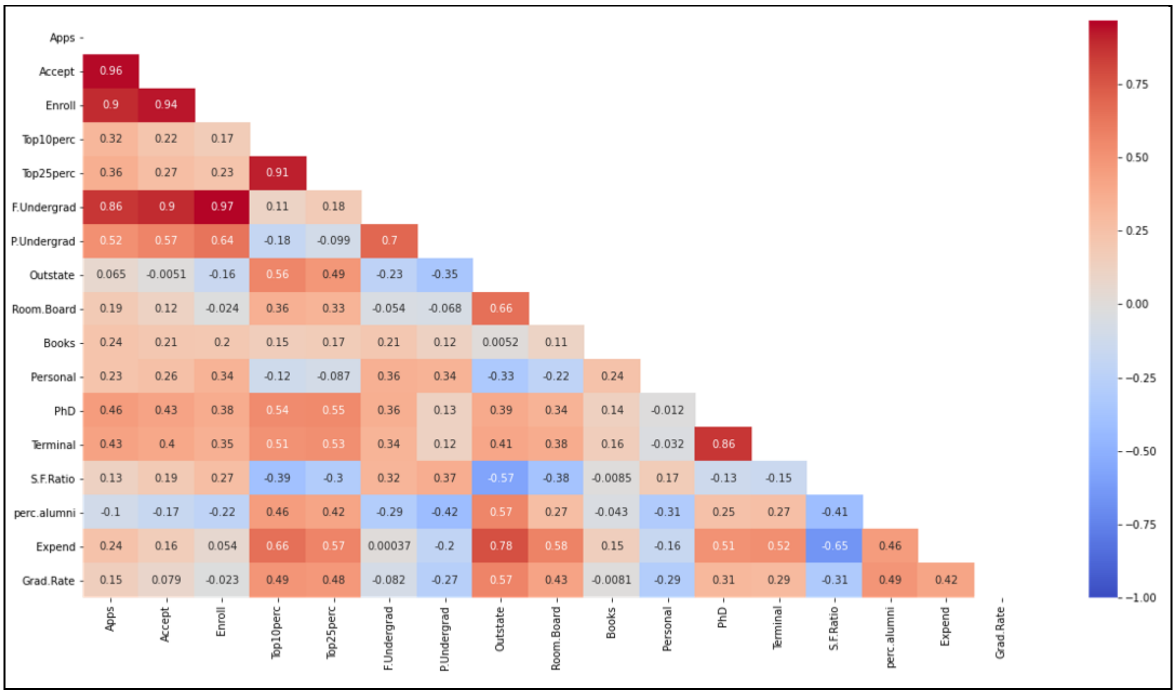
-->Still we have not done Outlier treatment, There are High number of Outliers that Resulted into Skewness in plots.We will do outlier treatment in next steps.

***2.3) Comment on the comparison between covariance and the correlation matrix.***

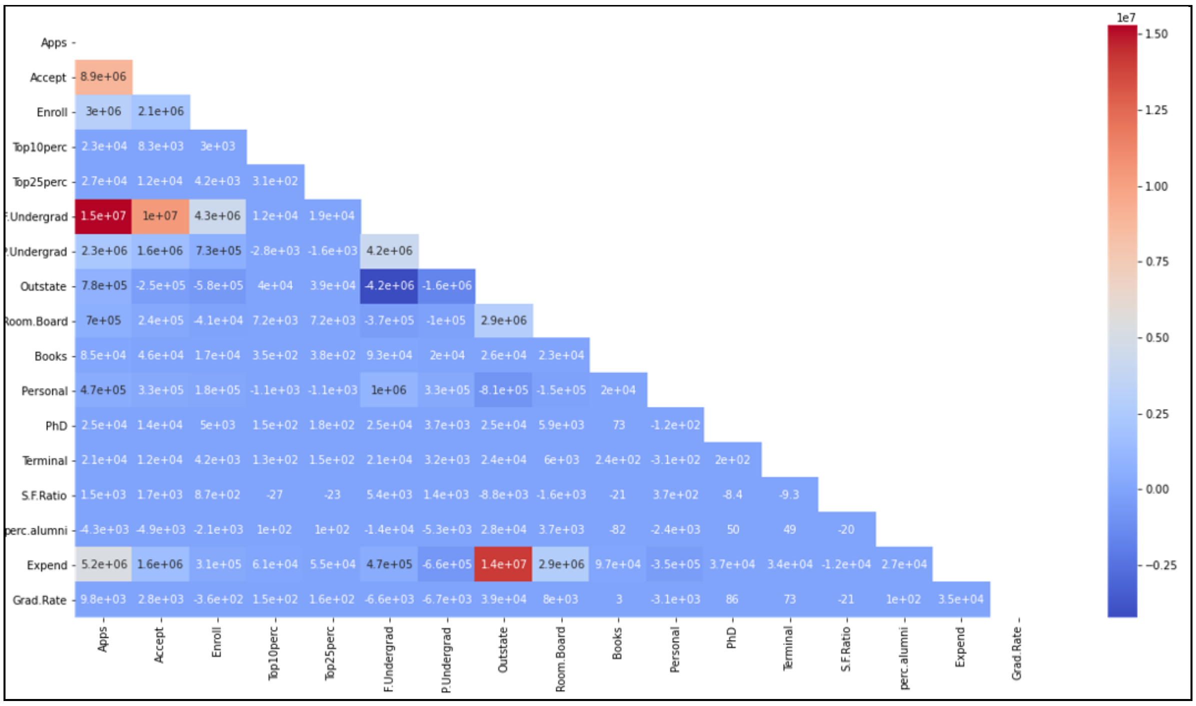
*Plot Correlation matrix (dataframe df, Where Outlier treatment is not done)*



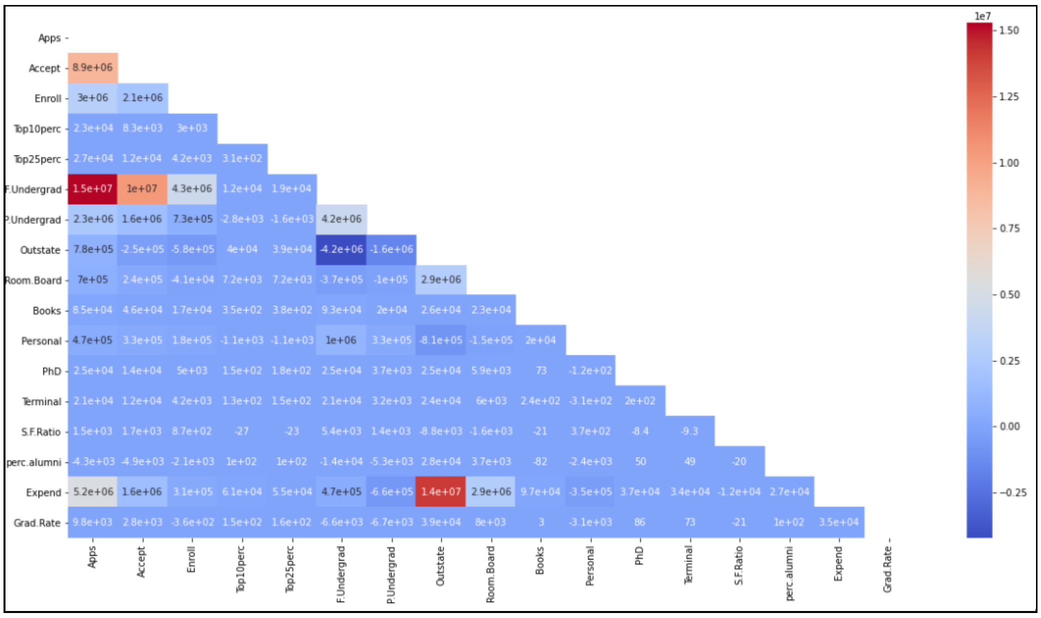
*Plot Correlation matrix, Correlation matrix for Data Without Outliers(Outlier treatment is done on DataFrame df4)*



Plot Covarience matrix (dataframe df, Where Outlier treatment is not done)



Plot Covarience matrix,Covarience matrix for Data Without Outliers



If we Comapare Two Correlation matrix, One with Outliers and other without outliers, We can say that these both Correlation matrix have little changes and are almost equivalent.

Same goes with Covariance Matrix,One with Outliers and other without outliers, We can say that these both covariance matrix have no changes and are equivalent.

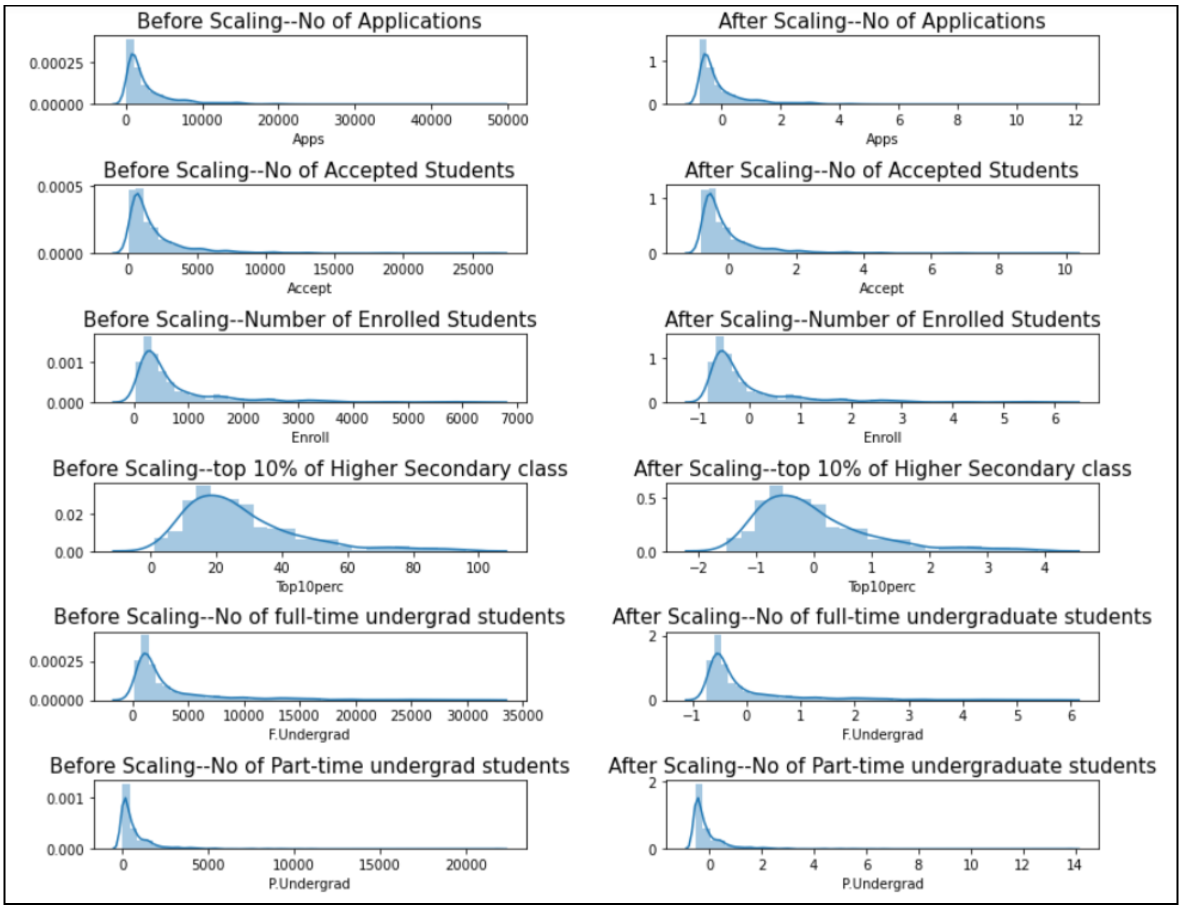
Covarience tells us, is there any dependence between two variables,metrics such as "To what extent variables change together". It does not tell us about dependency between variables.

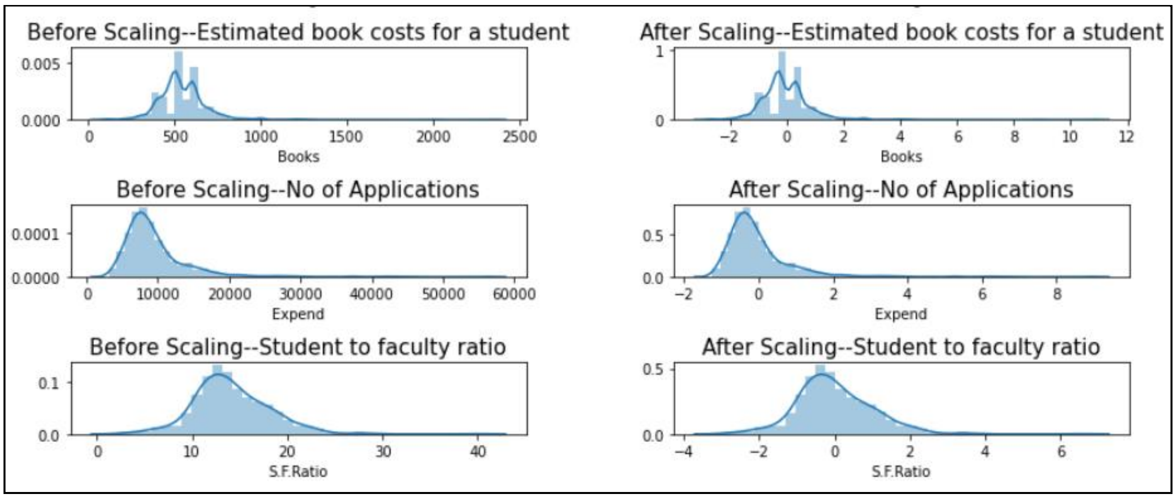
while Correlation gives us Strength and Direction as well. That is One variable is Increasing then other is Increasing or decreasing.additionally it tell us the strength of bond between two variables.

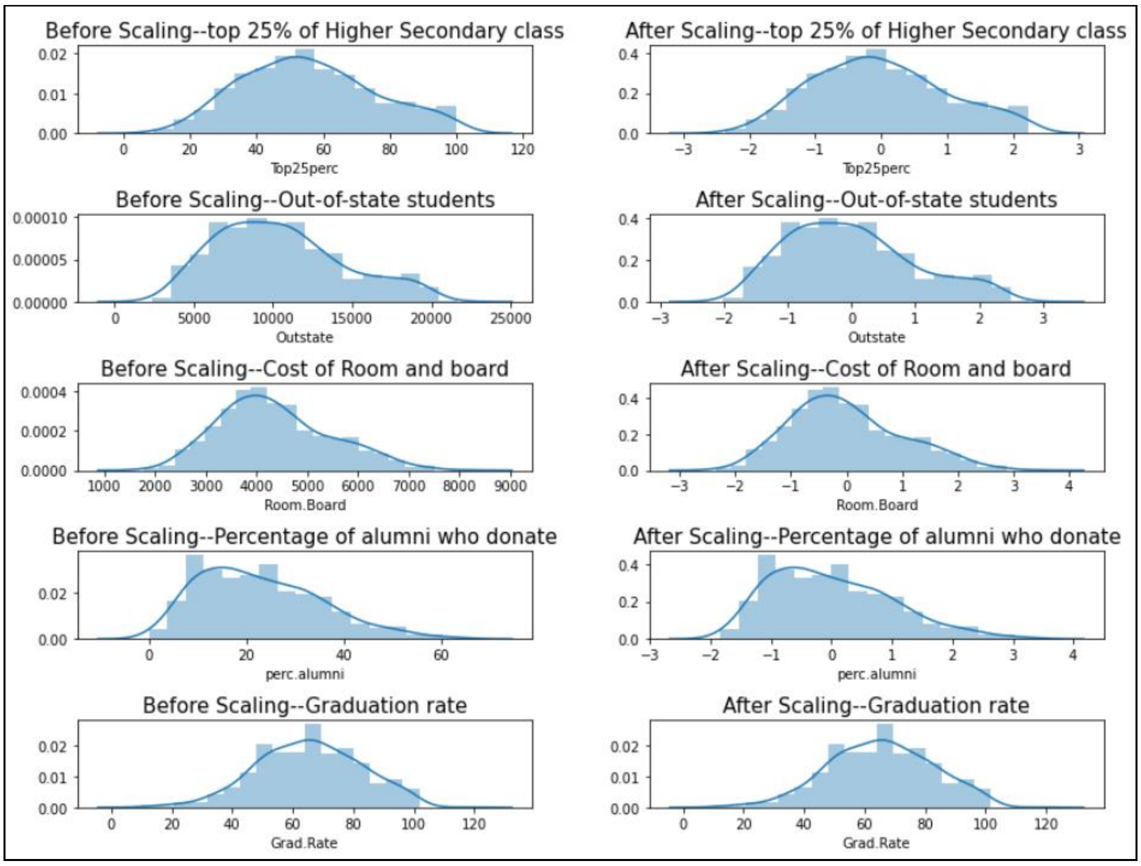
Covariance and Correlation have same analogy as Variance and Std deviation. We have to divide Covariance by Std deviation,then we get Correlation.

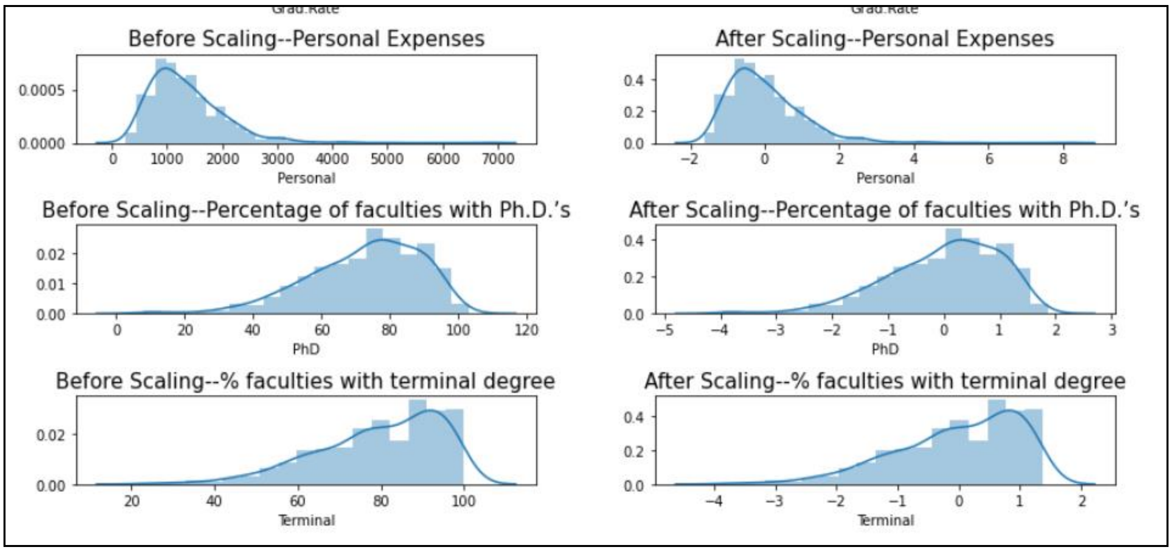
***2.4) Check the dataset for outliers before and after scaling. Draw your inferences from this exercise.***

We will Approach this Question by two plots, One is distribution plot (to see the distriution before and after scaling) and Boxplot(Which will give us idea about outliers)

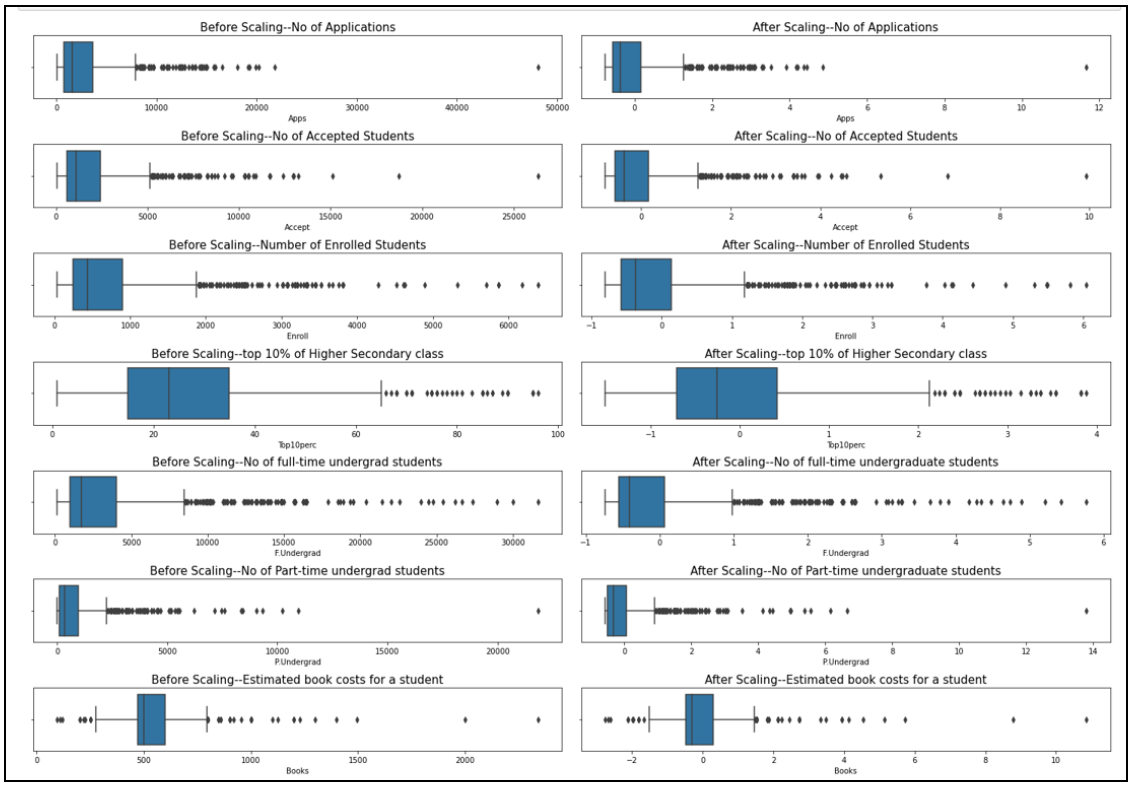


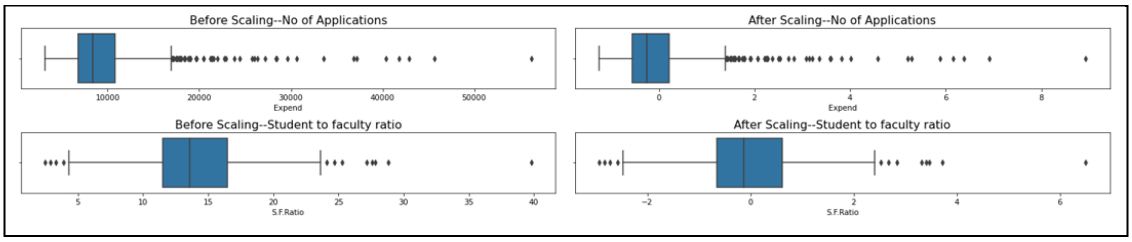


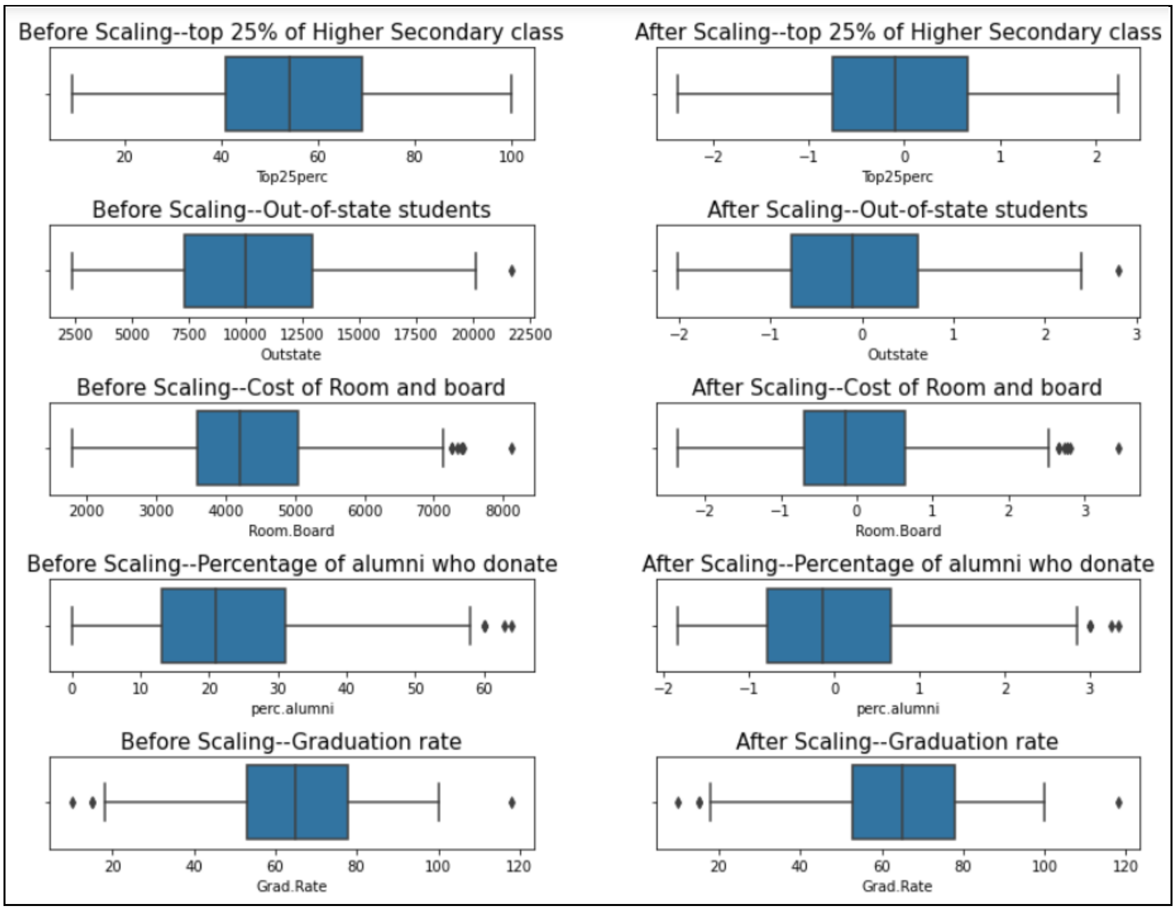

Before Scaling--top 25% of Higher Secondary class
Top25perc
After Scaling--top 25% of Higher Secondary class
Top25perc
Before Scaling--Out-of-state students
Outstate
After Scaling--Out-of-state students
Outstate
Before Scaling--Cost of Room and board
Room.Board
After Scaling--Cost of Room and board
Room.Board
Before Scaling--Percentage of alumni who donate
perc.alumni
After Scaling--Percentage of alumni who donate
perc.alumni
Before Scaling--Graduation rate
Grad.Rate
After Scaling--Graduation rate
Grad.Rate

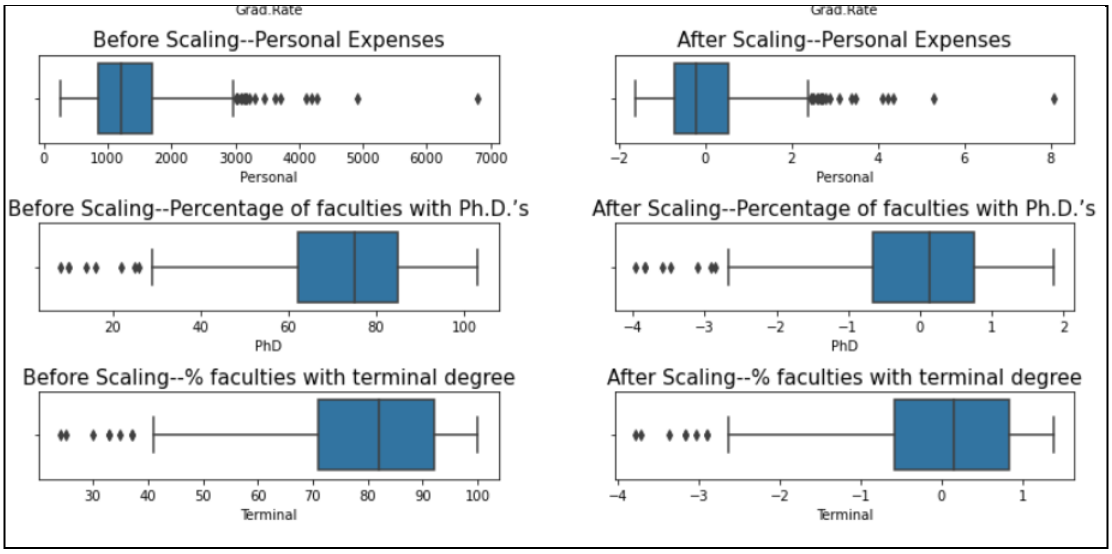

Grad.Rate
Grad.Rate
Before Scaling--Personal Expenses
Personal
After Scaling--Personal Expenses
Personal
Before Scaling--Percentage of faculties with Ph.D.'s
PhD
After Scaling--Percentage of faculties with Ph.D.'s
PhD
Before Scaling--% faculties with terminal degree
Terminal
After Scaling--% faculties with terminal degree
Terminal

Boxplots:


Before Scaling--No of Applications
Apps
After Scaling--No of Applications
Apps
Before Scaling--No of Accepted Students
Accept
After Scaling--No of Accepted Students
Accept
Before Scaling--Number of Enrolled Students
Enroll
After Scaling--Number of Enrolled Students
Enroll
Before Scaling--top 10% of Higher Secondary class
Top10perc
After Scaling--top 10% of Higher Secondary class
Top10perc
Before Scaling--No of full-time undergrad students
F.Undergrad
After Scaling--No of full-time undergraduate students
F.Undergrad
Before Scaling--No of Part-time undergrad students
P.Undergrad
After Scaling--No of Part-time undergraduate students
P.Undergrad
Before Scaling--Estimated book costs for a student
Books
After Scaling--Estimated book costs for a student
Books


Before Scaling--No of Applications
Expend
After Scaling--No of Applications
Expend
Before Scaling--Student to faculty ratio
S.F.Ratio
After Scaling--Student to faculty ratio
S.F.Ratio


Before Scaling--top 25% of Higher Secondary class
After Scaling--top 25% of Higher Secondary class
Top25perc
Before Scaling--Out-of-state students
After Scaling--Out-of-state students
Outstate
Before Scaling--Cost of Room and board
After Scaling--Cost of Room and board
Room.Board
Before Scaling--Percentage of alumni who donate
After Scaling--Percentage of alumni who donate
perc.alumni
Before Scaling--Graduation rate
After Scaling--Graduation rate
Grad.Rate


Grad.Rate
Before Scaling--Personal Expenses
After Scaling--Personal Expenses
Personal
Before Scaling--Percentage of faculties with Ph.D.'s
After Scaling--Percentage of faculties with Ph.D.'s
PhD
Before Scaling--% faculties with terminal degree
After Scaling--% faculties with terminal degree
Terminal

So, as you can see the after scaling also, the outliers remain outliers. Only the range is changed.

By scaling your variables, you can help compare different variables on equal footing.

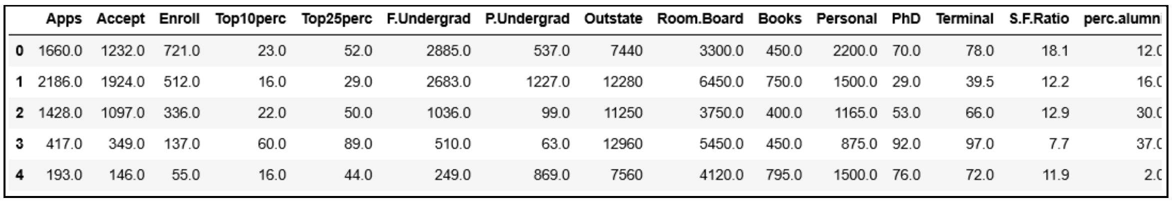
Scaling will change the range only, and in all the cases outliers will remain outliers after normalisation.

All distplot are also same, Note :we are just scaling variables not Normalizing. Hence shape will be same after and before Scaling.

***2.5) Build the covariance matrix, eigenvalues, and eigenvector.***

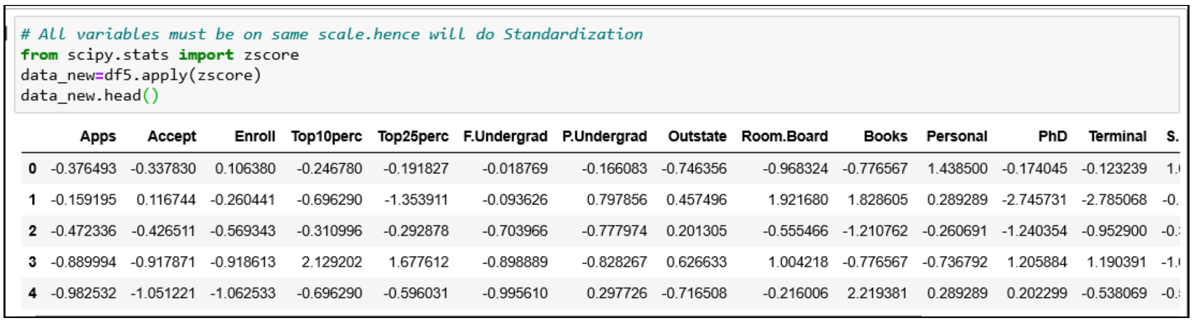
*We will first take copy of original data and remove column Names, as it has String value not continous numerical values*

*Remove Names Column and Copy it to New dataframe.No changes in df and df1*



| | Apps | Accept | Enroll | Top10perc | Top25perc | F.Undergrad | P.Undergrad | Outstate | Room.Board | Books | Personal | PhD | Terminal | S.F.Ratio | perc.alumn |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1660.0 | 1232.0 | 721.0 | 23.0 | 52.0 | 2885.0 | 537.0 | 7440 | 3300.0 | 450.0 | 2200.0 | 70.0 | 78.0 | 18.1 | 12. |
| **1** | 2186.0 | 1924.0 | 512.0 | 16.0 | 29.0 | 2683.0 | 1227.0 | 12280 | 6450.0 | 750.0 | 1500.0 | 29.0 | 39.5 | 12.2 | 16. |
| **2** | 1428.0 | 1097.0 | 336.0 | 22.0 | 50.0 | 1036.0 | 99.0 | 11250 | 3750.0 | 400.0 | 1165.0 | 53.0 | 66.0 | 12.9 | 30. |
| **3** | 417.0 | 349.0 | 137.0 | 60.0 | 89.0 | 510.0 | 63.0 | 12960 | 5450.0 | 450.0 | 875.0 | 92.0 | 97.0 | 7.7 | 37. |
| **4** | 193.0 | 146.0 | 55.0 | 16.0 | 44.0 | 249.0 | 869.0 | 7560 | 4120.0 | 795.0 | 1500.0 | 76.0 | 72.0 | 11.9 | 2. |

All variables must be on same scale.hence will do Standardization

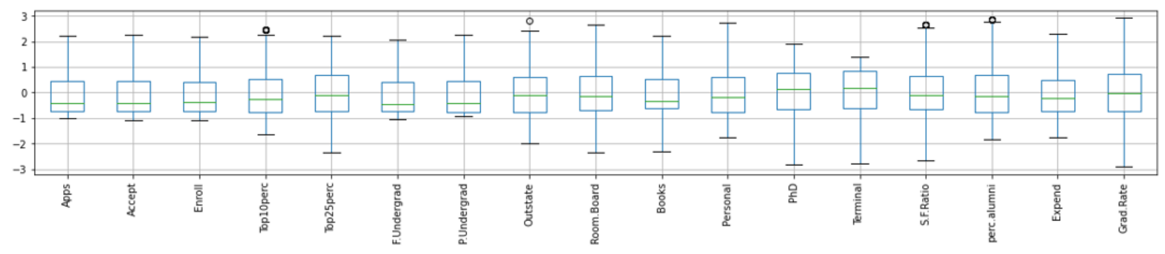


```
# All variables must be on same scale.hence will do Standardization
from scipy.stats import zscore
data_new=df5.apply(zscore)
data_new.head()
```

| | Apps | Accept | Enroll | Top10perc | Top25perc | F.Undergrad | P.Undergrad | Outstate | Room.Board | Books | Personal | PhD | Terminal | S. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | -0.376493 | -0.337830 | 0.106380 | -0.246780 | -0.191827 | -0.018769 | -0.166083 | -0.746356 | -0.968324 | -0.776567 | 1.438500 | -0.174045 | -0.123239 | 1. |
| **1** | -0.159195 | 0.116744 | -0.260441 | -0.696290 | -1.353911 | -0.093626 | 0.797856 | 0.457496 | 1.921680 | 1.828605 | 0.289289 | -2.745731 | -2.785068 | -0. |
| **2** | -0.472336 | -0.426511 | -0.569343 | -0.310996 | -0.292878 | -0.703966 | -0.777974 | 0.201305 | -0.555466 | -1.210762 | -0.260691 | -1.240354 | -0.952900 | -0. |
| **3** | -0.889994 | -0.917871 | -0.918613 | 2.129202 | 1.677612 | -0.898889 | -0.828267 | 0.626633 | 1.004218 | -0.776567 | -0.736792 | 1.205884 | 1.190391 | -1. |
| **4** | -0.982532 | -1.051221 | -1.062533 | -0.696290 | -0.596031 | -0.995610 | 0.297726 | -0.716508 | -0.216006 | 2.219381 | 0.289289 | 0.202299 | -0.538069 | -0. |

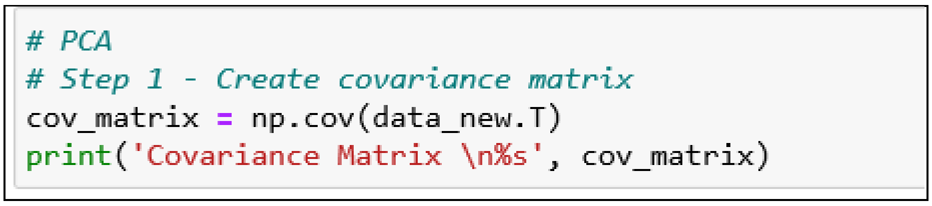
Plot Boxplot to check data has no outliers

```
data_new.boxplot(figsize=(20,3))
plt.xticks(rotation=90)
plt.show()
```

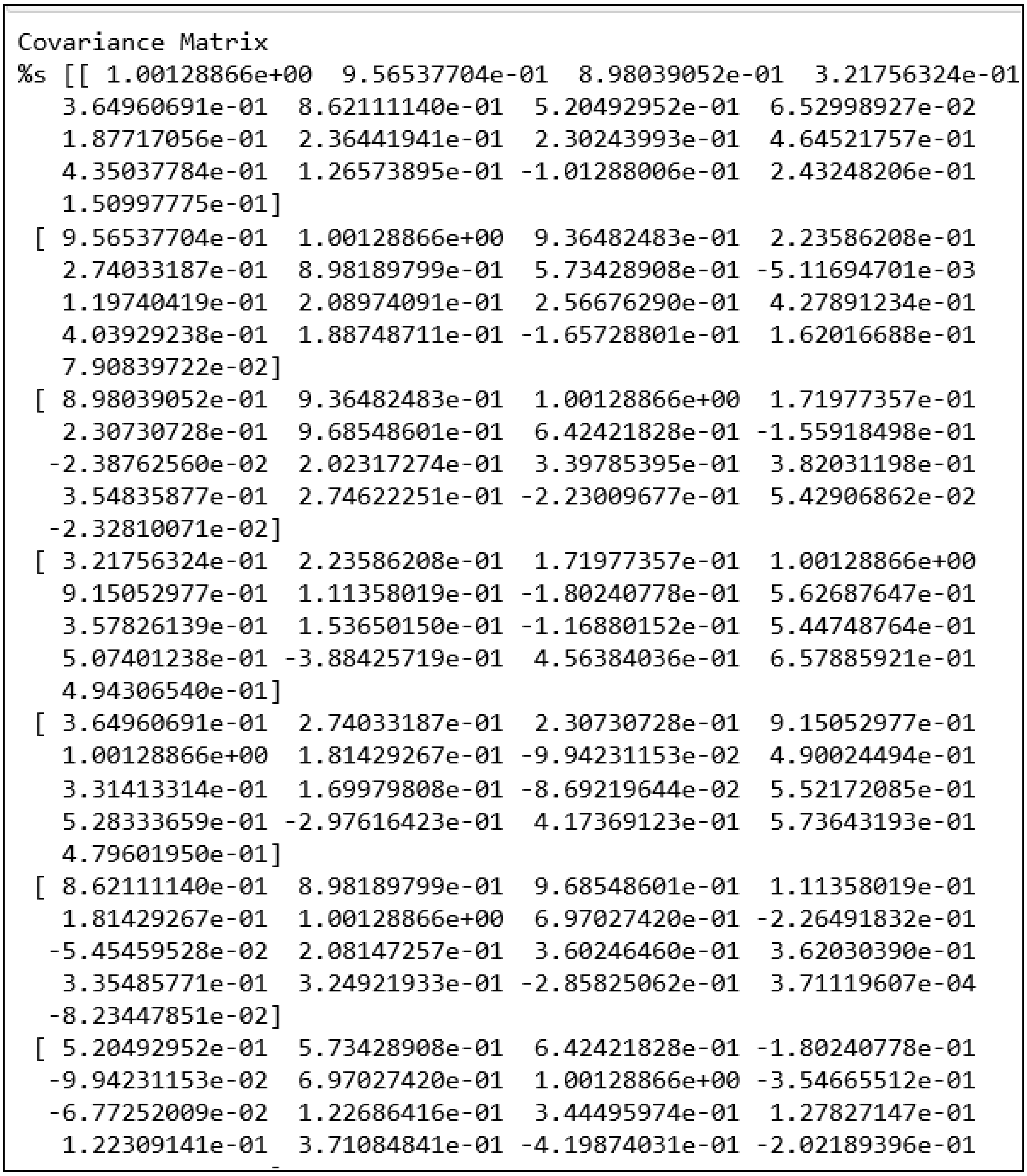


**Step 1: PCA-*Covariance matrix***

*Np.cov will give us covariance*

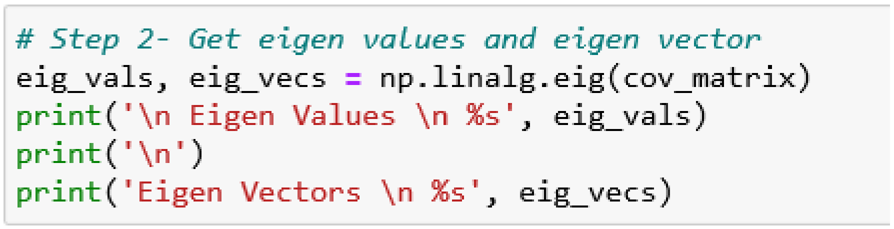


```
# PCA
# Step 1 - Create covariance matrix
cov_matrix = np.cov(data_new.T)
print('Covariance Matrix \n%s', cov_matrix)
```

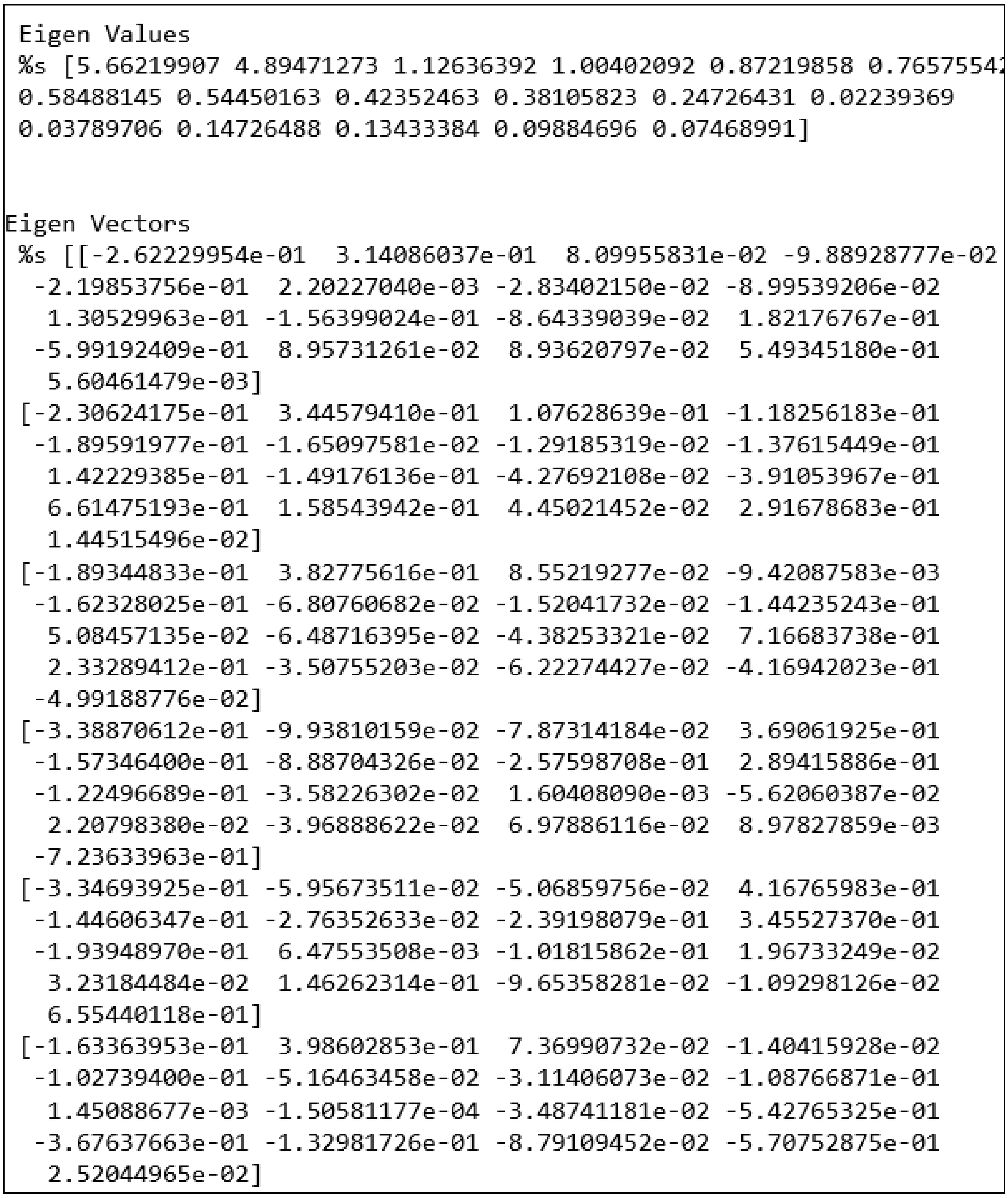


```
Covariance Matrix
%s [[ 1.00128866e+00  9.56537704e-01  8.98039052e-01  3.21756324e-01
   3.64960691e-01  8.62111140e-01  5.20492952e-01  6.52998927e-02
   1.87717056e-01  2.36441941e-01  2.30243993e-01  4.64521757e-01
   4.35037784e-01  1.26573895e-01 -1.01288006e-01  2.43248206e-01
   1.50997775e-01]
 [ 9.56537704e-01  1.00128866e+00  9.36482483e-01  2.23586208e-01
   2.74033187e-01  8.98189799e-01  5.73428908e-01 -5.11694701e-03
   1.19740419e-01  2.08974091e-01  2.56676290e-01  4.27891234e-01
   4.03929238e-01  1.88748711e-01 -1.65728801e-01  1.62016688e-01
   7.90839722e-02]
 [ 8.98039052e-01  9.36482483e-01  1.00128866e+00  1.71977357e-01
   2.30730728e-01  9.68548601e-01  6.42421828e-01 -1.55918498e-01
  -2.38762560e-02  2.02317274e-01  3.39785395e-01  3.82031198e-01
   3.54835877e-01  2.74622251e-01 -2.23009677e-01  5.42906862e-02
  -2.32810071e-02]
 [ 3.21756324e-01  2.23586208e-01  1.71977357e-01  1.00128866e+00
   9.15052977e-01  1.11358019e-01 -1.80240778e-01  5.62687647e-01
   3.57826139e-01  1.53650150e-01 -1.16880152e-01  5.44748764e-01
   5.07401238e-01 -3.88425719e-01  4.56384036e-01  6.57885921e-01
   4.94306540e-01]
 [ 3.64960691e-01  2.74033187e-01  2.30730728e-01  9.15052977e-01
   1.00128866e+00  1.81429267e-01 -9.94231153e-02  4.90024494e-01
   3.31413314e-01  1.69979808e-01 -8.69219644e-02  5.52172085e-01
   5.28333659e-01 -2.97616423e-01  4.17369123e-01  5.73643193e-01
   4.79601950e-01]
 [ 8.62111140e-01  8.98189799e-01  9.68548601e-01  1.11358019e-01
   1.81429267e-01  1.00128866e+00  6.97027420e-01 -2.26491832e-01
  -5.45459528e-02  2.08147257e-01  3.60246460e-01  3.62030390e-01
   3.35485771e-01  3.24921933e-01 -2.85825062e-01  3.71119607e-04
  -8.23447851e-02]
 [ 5.20492952e-01  5.73428908e-01  6.42421828e-01 -1.80240778e-01
  -9.94231153e-02  6.97027420e-01  1.00128866e+00 -3.54665512e-01
  -6.77252009e-02  1.22686416e-01  3.44495974e-01  1.27827147e-01
   1.22309141e-01  3.71084841e-01 -4.19874031e-01 -2.02189396e-01
  2.65499420e-01]
 [ 6.52998927e-02 -5.11694701e-03 -1.55918498e-01  5.62687647e-01
   4.90024494e-01 -2.26491832e-01 -3.54665512e-01  1.00128866e+00
   6.56093211e-01  5.17686720e-03 -3.26082557e-01  3.91418707e-01
   4.12807102e-01 -5.74358839e-01  5.66379983e-01  7.76324230e-01
   5.72937965e-01]
 [ 1.87717056e-01  1.19740419e-01 -2.38762560e-02  3.57826139e-01
   3.31413314e-01 -5.45459528e-02 -6.77252009e-02  6.56093211e-01
   1.00128866e+00  1.09064551e-01 -2.19837042e-01  3.41908577e-01
   3.79759015e-01 -3.76915472e-01  2.72743761e-01  5.81370284e-01
   4.26338910e-01]
 [ 2.36441941e-01  2.08974091e-01  2.02317274e-01  1.53650150e-01
   1.69979808e-01  2.08147257e-01  1.22686416e-01  5.17686720e-03
   1.09064551e-01  1.00128866e+00  2.40172145e-01  1.36566243e-01
   1.59523091e-01 -8.54689129e-03 -4.28870629e-02  1.50176551e-01
  -8.06107505e-03]
 [ 2.30243993e-01  2.56676290e-01  3.39785395e-01 -1.16880152e-01
  -8.69219644e-02  3.60246460e-01  3.44495974e-01 -3.26082557e-01
  -2.19837042e-01  2.40172145e-01  1.00128866e+00 -1.16986124e-02
  -3.20117803e-02  1.74136664e-01 -3.06146886e-01 -1.63481407e-01
  -2.91268705e-01]
 [ 4.64521757e-01  4.27891234e-01  3.82031198e-01  5.44748764e-01
   5.52172085e-01  3.62030390e-01  1.27827147e-01  3.91418707e-01
   3.41908577e-01  1.36566243e-01 -1.16986124e-02  1.00128866e+00
   8.64040263e-01 -1.29556494e-01  2.49197779e-01  5.11186852e-01
   3.10418895e-01]
 [ 4.35037784e-01  4.03929238e-01  3.54835877e-01  5.07401238e-01
   5.28333659e-01  3.35485771e-01  1.22309141e-01  4.12807102e-01
   3.79759015e-01  1.59523091e-01 -3.20117803e-02  8.64040263e-01
   1.00128866e+00 -1.51187934e-01  2.66375402e-01  5.24743500e-01
   2.93180212e-01]
  [illegible]]
[ 1.26573895e-01  1.88748711e-01  2.74622251e-01 -3.88425719e-01
 -2.97616423e-01  3.24921933e-01  3.71084841e-01 -5.74358839e-01
 -3.76915472e-01 -8.54689129e-03  1.74136664e-01 -1.29556494e-01
 -1.51187934e-01  1.00128866e+00 -4.12632056e-01 -6.55219504e-01
 -3.08922187e-01]
[-1.01288006e-01 -1.65728801e-01 -2.23009677e-01  4.56384036e-01
  4.17369123e-01 -2.85825062e-01 -4.19874031e-01  5.66379983e-01
  2.72743761e-01 -4.28870629e-02 -3.06146886e-01  2.49197779e-01
  2.66375402e-01 -4.12632056e-01  1.00128866e+00  4.63518674e-01
  4.92040760e-01]
[ 2.43248206e-01  1.62016688e-01  5.42906862e-02  6.57885921e-01
  5.73643193e-01  3.71119607e-04 -2.02189396e-01  7.76324230e-01
  5.81370284e-01  1.50176551e-01 -1.63481407e-01  5.11186852e-01
  5.24743500e-01 -6.55219504e-01  4.63518674e-01  1.00128866e+00
  4.15826026e-01]
[ 1.50997775e-01  7.90839722e-02 -2.32810071e-02  4.94306540e-01
  4.79601950e-01 -8.23447851e-02 -2.65499420e-01  5.72937965e-01
  4.26338910e-01 -8.06107505e-03 -2.91268705e-01  3.10418895e-01
  2.93180212e-01 -3.08922187e-01  4.92040760e-01  4.15826026e-01
  1.00128866e+00]]
```

***Step 2: Eigen Values and Eigen Vectors***



```
# Step 2- Get eigen values and eigen vector
eig_vals, eig_vecs = np.linalg.eig(cov_matrix)
print('\n Eigen Values \n %s', eig_vals)
print('\n')
print('Eigen Vectors \n %s', eig_vecs)
```



```
 Eigen Values
 %s [5.66219907 4.89471273 1.12636392 1.00402092 0.87219858 0.7657554
 0.58488145 0.54450163 0.42352463 0.38105823 0.24726431 0.02239369
 0.03789706 0.14726488 0.13433384 0.09884696 0.07468991]


Eigen Vectors
 %s [[-2.62229954e-01  3.14086037e-01  8.09955831e-02 -9.88928777e-02
  -2.19853756e-01  2.20227040e-03 -2.83402150e-02 -8.99539206e-02
   1.30529963e-01 -1.56399024e-01 -8.64339039e-02  1.82176767e-01
  -5.99192409e-01  8.95731261e-02  8.93620797e-02  5.49345180e-01
   5.60461479e-03]
 [-2.30624175e-01  3.44579410e-01  1.07628639e-01 -1.18256183e-01
  -1.89591977e-01 -1.65097581e-02 -1.29185319e-02 -1.37615449e-01
   1.42229385e-01 -1.49176136e-01 -4.27692108e-02 -3.91053967e-01
   6.61475193e-01  1.58543942e-01  4.45021452e-02  2.91678683e-01
   1.44515496e-02]
 [-1.89344833e-01  3.82775616e-01  8.55219277e-02 -9.42087583e-03
  -1.62328025e-01 -6.80760682e-02 -1.52041732e-02 -1.44235243e-01
   5.08457135e-02 -6.48716395e-02 -4.38253321e-02  7.16683738e-01
   2.33289412e-01 -3.50755203e-02 -6.22274427e-02 -4.16942023e-01
  -4.99188776e-02]
 [-3.38870612e-01 -9.93810159e-02 -7.87314184e-02  3.69061925e-01
  -1.57346400e-01 -8.88704326e-02 -2.57598708e-01  2.89415886e-01
  -1.22496689e-01 -3.58226302e-02  1.60408090e-03 -5.62060387e-02
   2.20798380e-02 -3.96888622e-02  6.97886116e-02  8.97827859e-03
  -7.23633963e-01]
 [-3.34693925e-01 -5.95673511e-02 -5.06859756e-02  4.16765983e-01
  -1.44606347e-01 -2.76352633e-02 -2.39198079e-01  3.45527370e-01
  -1.93948970e-01  6.47553508e-03 -1.01815862e-01  1.96733249e-02
   3.23184484e-02  1.46262314e-01 -9.65358281e-02 -1.09298126e-02
   6.55440118e-01]
 [-1.63363953e-01  3.98602853e-01  7.36990732e-02 -1.40415928e-02
  -1.02739400e-01 -5.16463458e-02 -3.11406073e-02 -1.08766871e-01
   1.45088677e-03 -1.50581177e-04 -3.48741181e-02 -5.42765325e-01
  -3.67637663e-01 -1.32981726e-01 -8.79109452e-02 -5.70752875e-01
   2.52044965e-02]
 [-2.25370199e-02  3.57544493e-01  4.02991311e-02 -2.25334018e-01
   9.57648415e-02 -2.45333768e-02 -9.99602946e-03  1.23887243e-01
  -6.34603597e-01  5.46413915e-01  2.52353663e-01  2.95032320e-02
   2.62574146e-02  4.97741097e-02  4.48330712e-02  1.46416174e-01
  -3.97051558e-02]
 [-2.83418894e-01 -2.51926173e-01  1.47793128e-02 -2.63127474e-01
  -3.74030661e-02 -2.04238675e-02  9.42393435e-02  1.09944753e-02
  -8.70300720e-03 -2.32377635e-01  5.93993894e-01  1.05340125e-03
  -8.12623645e-02  5.59505595e-01  6.92980972e-02 -2.11124503e-01
  -1.88520595e-03]
 [-2.44165100e-01 -1.31948412e-01 -2.12471630e-02 -5.80771998e-01
   6.94353667e-02  2.37310219e-01  9.45330546e-02  3.89816941e-01
  -2.20477471e-01 -2.54708532e-01 -4.75559133e-01  9.85317812e-03
   2.67153412e-02 -1.06827479e-01  1.72539410e-02 -1.01163660e-01
  -2.82201807e-02]
 [-9.67336197e-02  9.39488008e-02 -6.97109228e-01  3.63052887e-02
  -3.53925445e-02  6.38604128e-01 -1.11136983e-01 -2.39880475e-01
   2.10349187e-02  9.11818746e-02  4.35903625e-02  4.36176092e-03
   1.04812743e-02  5.13778037e-02  3.55930858e-02 -2.85899031e-02
  -8.08775742e-03]
 [ 3.51980721e-02  2.32449626e-01 -5.30972095e-01  1.15165232e-01
   3.72672043e-04 -3.81497039e-01  6.39263507e-01  2.77411175e-01
   1.73317147e-02 -1.27800615e-01  1.52639049e-02 -1.08719247e-02
   4.53381131e-03  9.53529169e-03 -1.17755963e-02  3.37965245e-02
   1.42538056e-03]
 [-3.26417621e-01  5.50923798e-02  8.11568293e-02  1.47493434e-01
   5.50740871e-01  3.31663813e-03  8.92642452e-02 -3.42192998e-02
   1.66553904e-01  1.00896901e-01 -3.91625008e-02  1.33154500e-02
   1.24535635e-02 -7.38415939e-02  7.02378177e-01 -6.42349369e-02
   8.31249410e-02]
 [-3.23125866e-01  4.29817641e-02  5.90116499e-02  8.92316091e-02
   5.90384989e-01  3.53819563e-02  9.17842698e-02 -9.02524016e-02
   1.12652610e-01  8.59246464e-02 -8.42765501e-02  7.38073579e-03
  -1.79263411e-02  1.66315802e-01 -6.61893940e-01  9.87799111e-02
  -1.13346355e-01]
  1.1594[illegible]e-01]
[ 1.63117512e-01  2.59836893e-01  2.74182670e-01  2.59459470e-01
  1.42671769e-01  4.68715236e-01  1.52657742e-01  2.42697067e-01
 -1.53949926e-01 -4.70803066e-01  3.62555960e-01  8.85700652e-03
  1.83281207e-02 -2.40168599e-01 -4.85696576e-02  6.20843553e-02
  3.86298044e-03]
[-1.86577501e-01 -2.57127529e-01  1.03778841e-01  2.23891521e-01
 -1.28301294e-01  1.25611171e-02  3.91648861e-01 -5.65908139e-01
 -5.39234325e-01 -1.47213285e-01 -1.74291024e-01 -2.40556269e-02
 -9.94066346e-05 -4.90339178e-02  3.56767931e-02  2.80213390e-02
 -7.29014869e-03]
[-3.28948915e-01 -1.60079675e-01 -1.84231441e-01 -2.13763195e-01
  2.25379163e-02 -2.31552651e-01 -1.50372492e-01 -1.18830581e-01
  2.42191160e-02 -8.05077510e-02  3.92738917e-01  1.05552769e-02
  5.60077020e-02 -6.90530124e-01 -1.28890989e-01  1.28701212e-01
  1.45275815e-01]
[-2.38796220e-01 -1.67560132e-01  2.45356453e-01  3.60906371e-02
 -3.56800027e-01  3.13588215e-01  4.68649048e-01  1.80744837e-01
  3.16040253e-01  4.88148722e-01  8.76873675e-02 -2.51301375e-03
  1.48067785e-02 -1.58936554e-01 -6.35994331e-02 -7.18789854e-03
 -3.24971944e-03]]
```

## *2.6) Write the explicit form of the first PC (in terms of Eigen Vectors).*

PCA technique is drawing straight, explanatory lines through data.Which covers Maximum variance.

Each straight line represents a "principal component," or a relationship between an independent and dependent variable. While there are as many principal components as there are dimensions in the data, PCA's role is to prioritize them.

The first principal component bisects a scatterplot with a straight line in a way that explains the most variance; that is, it follows the longest dimension of the data. (This happens to coincide with the least error, as expressed by the red lines…)

Second component tries to fix the error of First Component, then Third willl try to fix errors caused by First and second Principal Component.It just goes on.

Eigen value is giving Strength/Magnitude of Line while Eigen vector gives us Direction. Eigen vector Represents Principal Component.

So General form of Conversion is

Current value of Covariance Matrix * Corresponding Eigen Vector Matrix = Eigen Valus * Target matrix produced by Mapping small projections on New line where most of variables Covered.

we can write Below equation based on this,

Covariance Matrix(A) * Eigen Vector of Matrix(u) = Scaled or Transformed version of Matrix(Which has same direction of Unscaled Matrix)

We can Write Above Equation in Simple Format as

***A * u = Lambda * u*** ¶

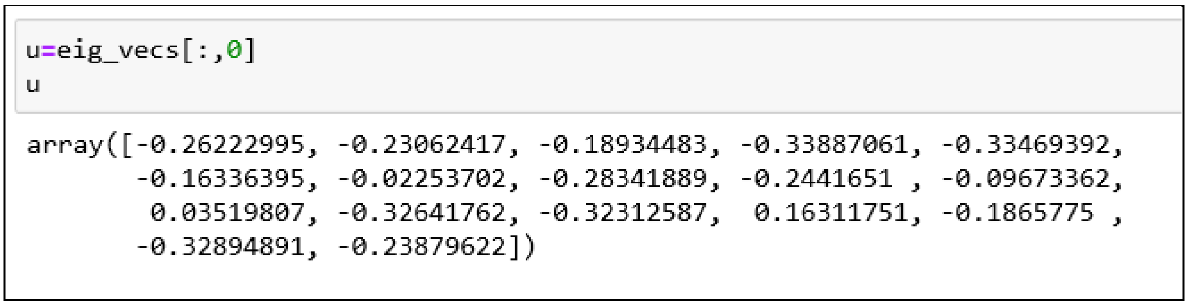
This 'Lambda' is scaled Factor Which is Nothing But Eigen value.

Basically lambda is Scaled multiplier of Matrix A by using Eigen Vector v, The Scale is Changed For the First Principal Component, [5.66219907] is Eigen Value and Eigen Vector is V[:,0]

In below module we can get Highest Cumulative variance as 33%, which shows First PC is Covering 33% data variance.

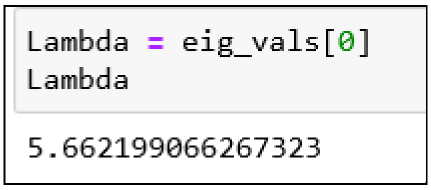
For First Principal Component let's find out Eigen Vector and its Eigen Value form that is lambda * Eigen Vector

For First Principal Component let's find out Eigen Vector and its Eigen Value form that is lambda * Eigen Vector



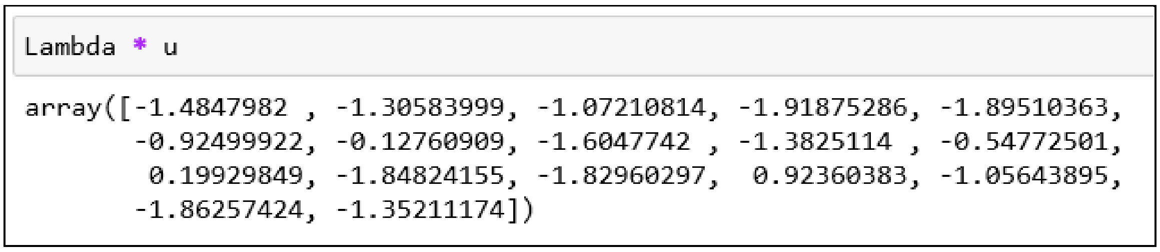
```
u=eig_vecs[:,0]
u
```

```
array([-0.26222995, -0.23062417, -0.18934483, -0.33887061, -0.33469392,
       -0.16336395, -0.02253702, -0.28341889, -0.2441651 , -0.09673362,
        0.03519807, -0.32641762, -0.32312587,  0.16311751, -0.1865775 ,
       -0.32894891, -0.23879622])
```



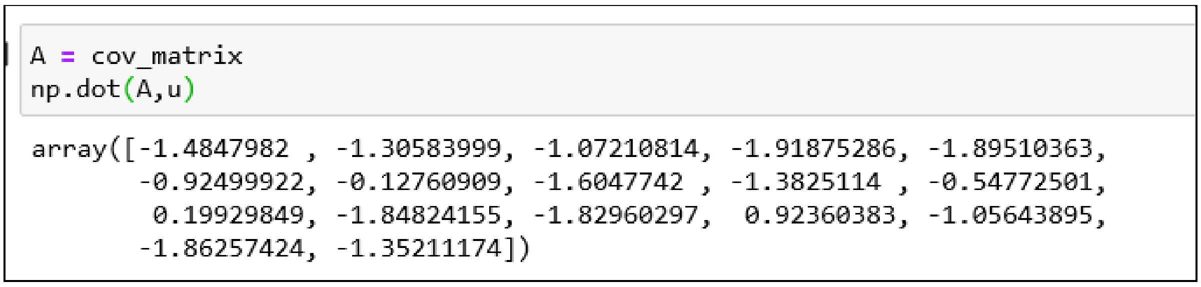
```
Lambda = eig_vals[0]
Lambda
```

```
5.662199066267323
```



```
Lambda * u
```

```
array([-1.4847982 , -1.30583999, -1.07210814, -1.91875286, -1.89510363,
       -0.92499922, -0.12760909, -1.6047742 , -1.3825114 , -0.54772501,
        0.19929849, -1.84824155, -1.82960297,  0.92360383, -1.05643895,
       -1.86257424, -1.35211174])
```



```
A = cov_matrix
np.dot(A,u)
```

```
array([-1.4847982 , -1.30583999, -1.07210814, -1.91875286, -1.89510363,
       -0.92499922, -0.12760909, -1.6047742 , -1.3825114 , -0.54772501,
        0.19929849, -1.84824155, -1.82960297,  0.92360383, -1.05643895,
       -1.86257424, -1.35211174])
```

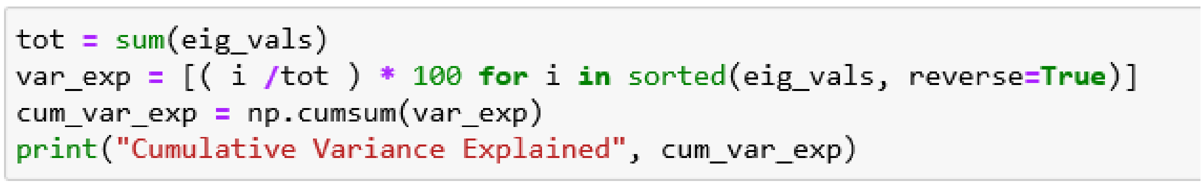
Above is the First Principal Component which is Scaled Version of Matrix applied to Eigen vector. We can see transformation as

***A.u = lambda * u***

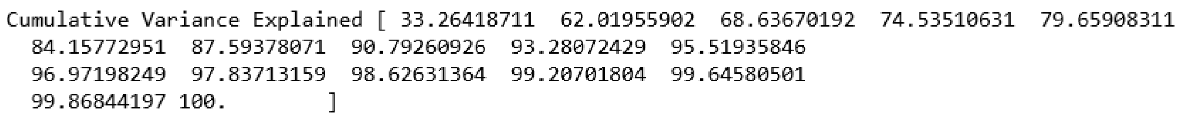
***2.7) Discuss the cumulative values of the eigenvalues. How does it help you to decide on the optimum number of principal components? What do the eigenvectors indicate? Perform PCA and export the data of the Principal Component scores into a data frame***

The eigenvectors with the lowest eigenvalues bear the least information about the distribution of the data; those are the ones can be dropped.

Calculate Cumulative variance:



```python
tot = sum(eig_vals)
var_exp = [( i /tot ) * 100 for i in sorted(eig_vals, reverse=True)]
cum_var_exp = np.cumsum(var_exp)
print("Cumulative Variance Explained", cum_var_exp)
```



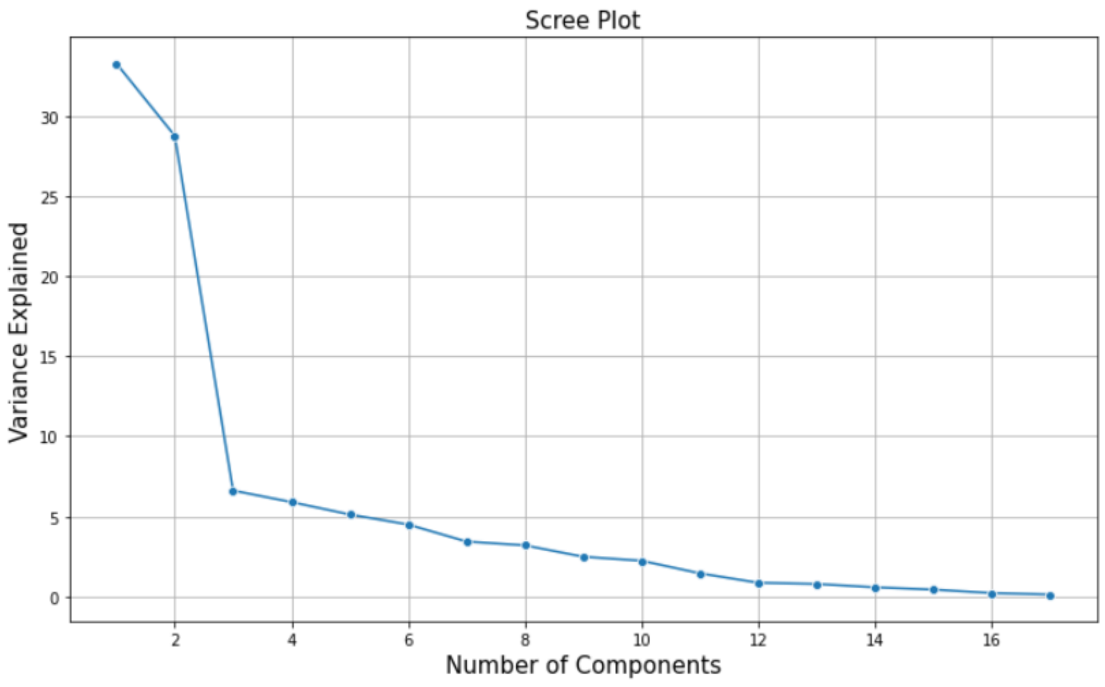
```
Cumulative Variance Explained [ 33.26418711  62.01955902  68.63670192  74.53510631  79.65908311
  84.15772951  87.59378071  90.79260926  93.28072429  95.51935846
  96.97198249  97.83713159  98.62631364  99.20701804  99.64580501
  99.86844197 100.        ]
```

The Above shown cumulative values help us to Select optimum number of Principal components. The First Principal component covers 33.26% variance, If we select first Two as Principal components variance will be covered as 62%, If we select first 3 then 68.63% variance will be covered. Here we will select 5 pricipal components, so that it will cover 80% data.

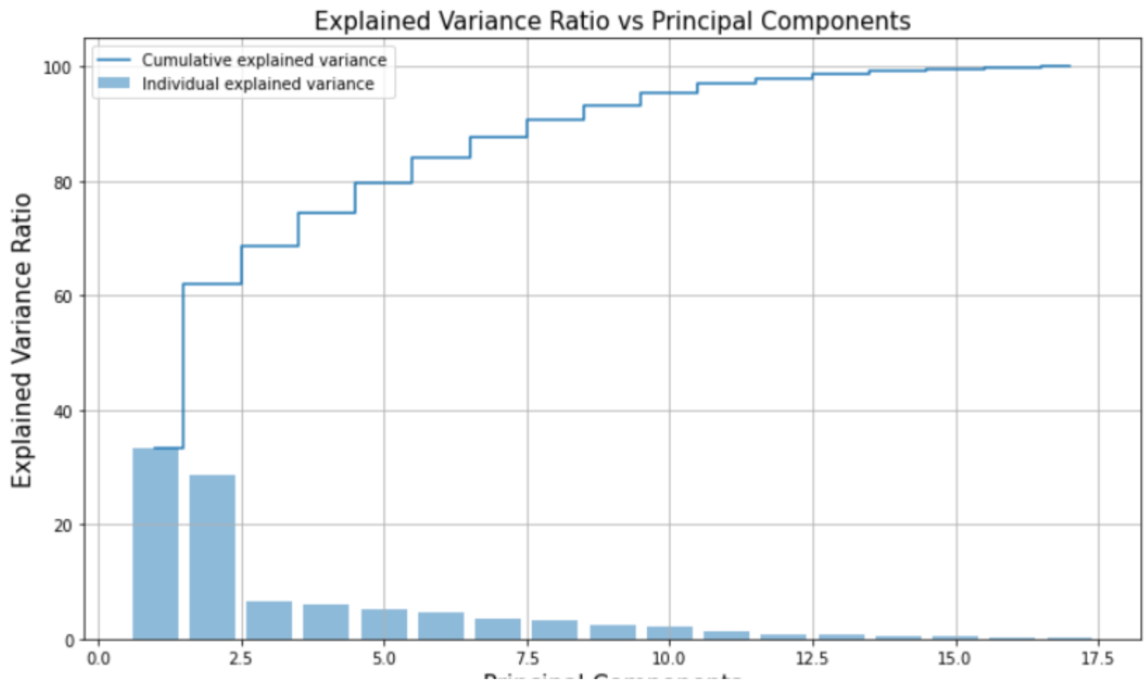
***Data of the Principal Component scores into a data frame.***

| | Variance covered |
|---|---|
| **0** | 33.264187 |
| **1** | 62.019559 |
| **2** | 68.636702 |
| **3** | 74.535106 |
| **4** | 79.659083 |
| **5** | 84.157730 |
| **6** | 87.593781 |
| **7** | 90.792609 |
| **8** | 93.280724 |
| **9** | 95.519358 |
| **10** | 96.971982 |
| **11** | 97.837132 |
| **12** | 98.626314 |
| **13** | 99.207018 |
| **14** | 99.645805 |
| **15** | 99.868442 |
| **16** | 100.000000 |

```
## Plotting the scree plot
plt.figure(figsize=(12,7))
sns.lineplot(y=var_exp,x=range(1,len(var_exp)+1),marker='o')
plt.xlabel('Number of Components',fontsize=15)
plt.ylabel('Variance Explained',fontsize=15)
plt.title('Scree Plot',fontsize=15)
plt.grid()
plt.show()
```

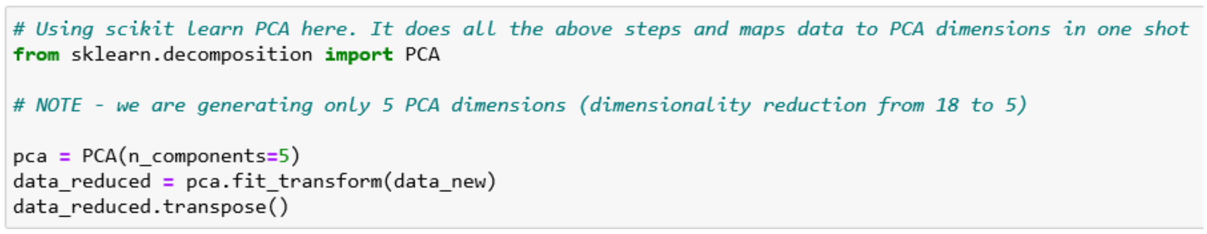


```python
# Plotting
plt.figure(figsize=(12 ,7))
plt.bar(range(1, eig_vals.size + 1), var_exp, alpha = 0.5, align = 'center', label = 'Individual explained variance')
plt.step(range(1, eig_vals.size + 1), cum_var_exp, where='mid', label = 'Cumulative explained variance')
plt.ylabel('Explained Variance Ratio',fontsize=15)
plt.xlabel('Principal Components',fontsize=15)
plt.title('Explained Variance Ratio vs Principal Components',fontsize=15)
plt.legend(loc = 'best')
plt.grid()
plt.show()
```



## *PCAMethod 2*

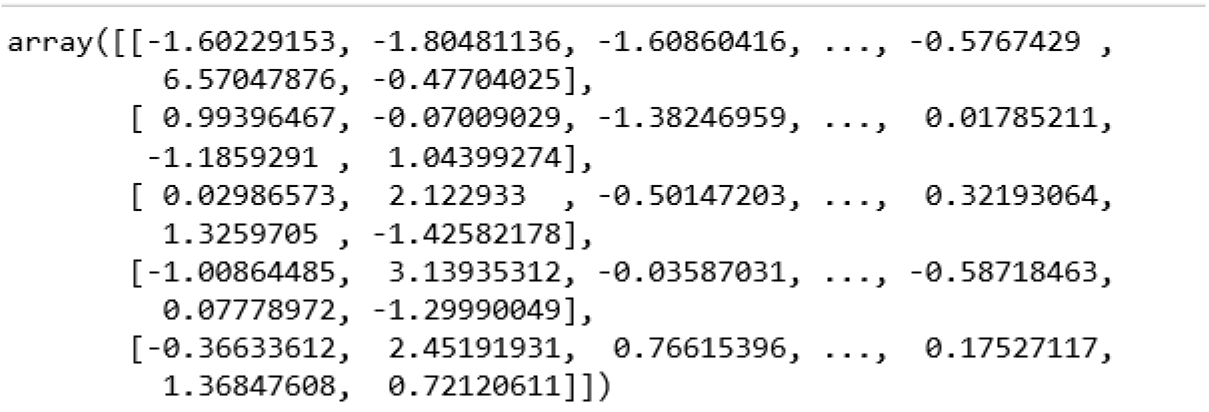
Use PCA from sklearn's decomposition class and find Principal Components



```python
# Using scikit learn PCA here. It does all the above steps and maps data to PCA dimensions in one shot
from sklearn.decomposition import PCA

# NOTE - we are generating only 5 PCA dimensions (dimensionality reduction from 18 to 5)

pca = PCA(n_components=5)
data_reduced = pca.fit_transform(data_new)
data_reduced.transpose()
```



```
array([[-1.60229153, -1.80481136, -1.60860416, ..., -0.5767429 ,
         6.57047876, -0.47704025],
       [ 0.99396467, -0.07009029, -1.38246959, ...,  0.01785211,
        -1.1859291 ,  1.04399274],
       [ 0.02986573,  2.122933  , -0.50147203, ...,  0.32193064,
         1.3259705 , -1.42582178],
       [-1.00864485,  3.13935312, -0.03587031, ..., -0.58718463,
         0.07778972, -1.29990049],
       [-0.36633612,  2.45191931,  0.76615396, ...,  0.17527117,
         1.36847608,  0.72120611]])
```

**Pca_components:**

```
pca.components_
```

```
array([[ 2.62229954e-01,  2.30624175e-01,  1.89344833e-01,
         3.38870612e-01,  3.34693925e-01,  1.63363953e-01,
         2.25370199e-02,  2.83418894e-01,  2.44165100e-01,
         9.67336197e-02, -3.51980721e-02,  3.26417621e-01,
         3.23125866e-01, -1.63117512e-01,  1.86577501e-01,
         3.28948915e-01,  2.38796220e-01],
       [ 3.14086037e-01,  3.44579410e-01,  3.82775616e-01,
        -9.93810159e-02, -5.95673511e-02,  3.98602853e-01,
         3.57544493e-01, -2.51926173e-01, -1.31948412e-01,
         9.39488008e-02,  2.32449626e-01,  5.50923798e-02,
         4.29817641e-02,  2.59836893e-01, -2.57127529e-01,
        -1.60079675e-01, -1.67560132e-01],
       [-8.09955686e-02, -1.07628653e-01, -8.55219347e-02,
         7.87314185e-02,  5.06859744e-02, -7.36990642e-02,
        -4.02991316e-02, -1.47793111e-02,  2.12471622e-02,
         6.97109227e-01,  5.30972095e-01, -8.11568301e-02,
        -5.90116490e-02, -2.74182670e-01, -1.03778841e-01,
         1.84231439e-01, -2.45356454e-01],
       [ 9.88929492e-02,  1.18256107e-01,  9.42085070e-03,
        -3.69061925e-01, -4.16765989e-01,  1.40416295e-02,
         2.25334015e-01,  2.63127482e-01,  5.80771995e-01,
        -3.63052900e-02, -1.15165232e-01, -1.47493438e-01,
        -8.92316044e-02, -2.59459472e-01, -2.23891521e-01,
         2.13763189e-01, -3.60906387e-02],
       [ 2.19854056e-01,  1.89591660e-01,  1.62327913e-01,
         1.57346399e-01,  1.44606323e-01,  1.02739559e-01,
        -9.57648506e-02,  3.74031007e-02, -6.94353814e-02,
         3.53925387e-02, -3.72673475e-04, -5.50740888e-01,
        -5.90384969e-01, -1.42671776e-01,  1.28301294e-01,
        -2.25379421e-02,  3.56800020e-01]])
```

```
pca.explained_variance_ratio_
```

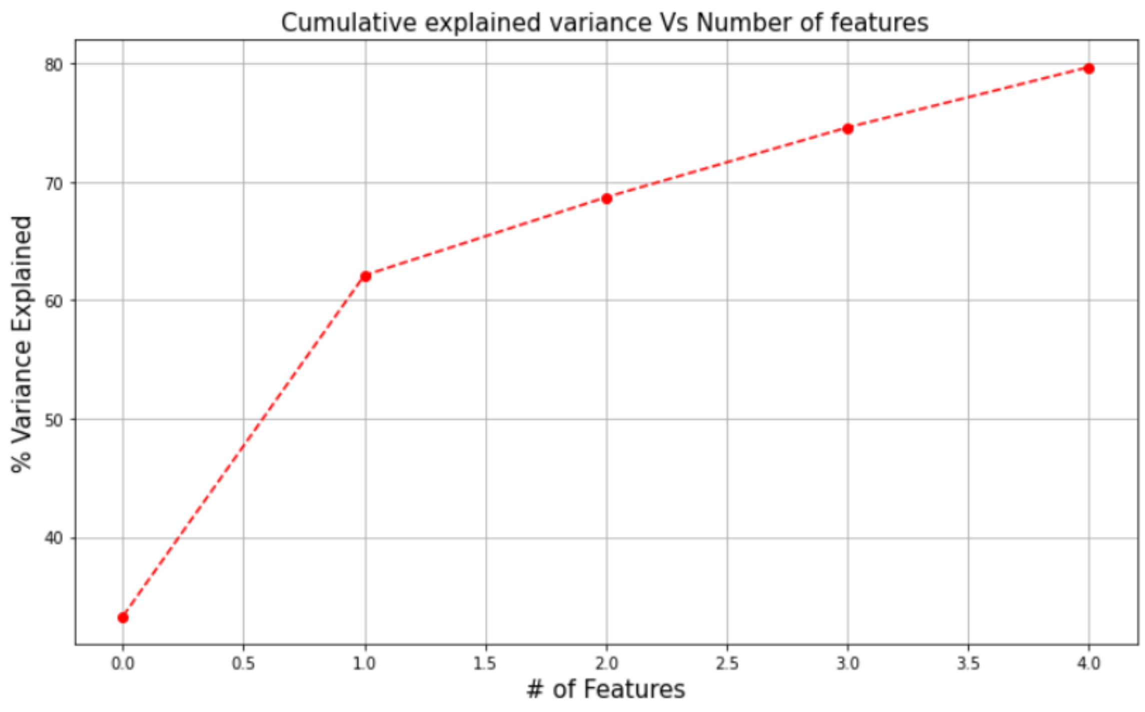
```
array([0.33264187, 0.28755372, 0.06617143, 0.05898404, 0.05123977])
```

**cumulative sum of variance explained with 5 features:**

```
var=np.cumsum(np.round(pca.explained_variance_ratio_, decimals=3)*100)
var #cumulative sum of variance explained with [n] features
```

```
array([33.3, 62.1, 68.7, 74.6, 79.7])
```

Lets plot a graph,”Cumulative explained variance Vs Number of features”



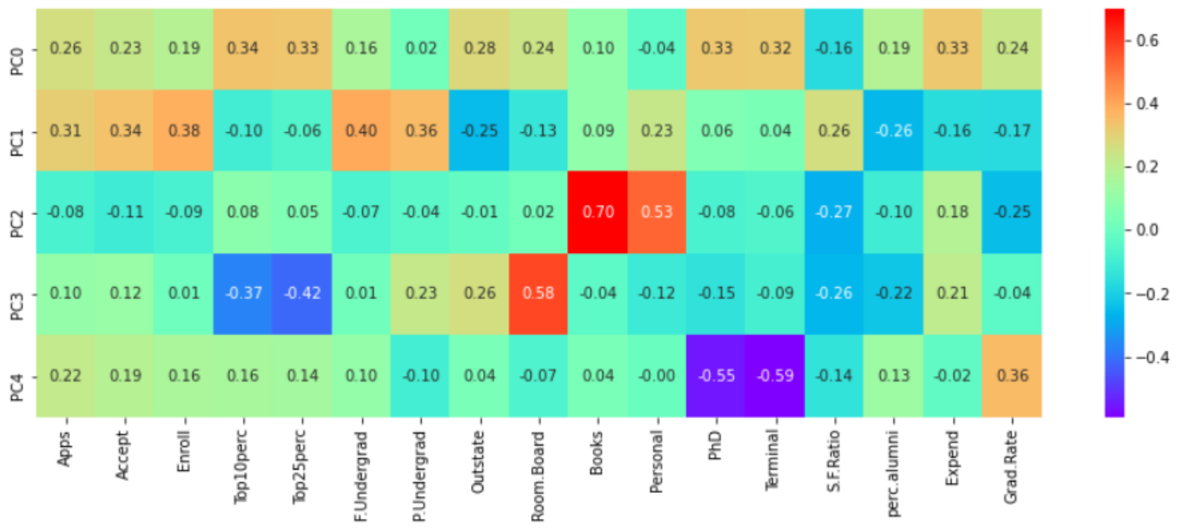
The plot above clearly shows that maximum variance (somewhere around 33.3%) can be explained by the first principal component alone. The second,third,fourth and fifth principal component share almost less amount of information.Comparatively 6th onwards components share less amount of information as compared to the rest of the Principal components.But those information cannot be ignored since they both contribute almost 21% of the data.We are covering 80% variance by considering 5 features.
The eigenvectors (principal components) determine the directions of the new feature space, and the eigenvalues determine their magnitude.

## *Correlation between components and features*

```
df_comp = pd.DataFrame(pca.components_,columns=list(data_new))
df_comp.head()
```

```
        Apps     Accept     Enroll  Top10perc  Top25perc  F.Undergrad  P.Undergrad   Outstate  Room.Board      Books   Personal        PhD   Terminal  S.
0   0.262230   0.230624   0.189345   0.338871   0.334694     0.163364     0.022537   0.283419    0.244165   0.096734  -0.035198   0.326418   0.323126  -0
1   0.314086   0.344579   0.382776  -0.099381  -0.059567     0.398603     0.357544  -0.251926   -0.131948   0.093949   0.232450   0.055092   0.042982  0.
2  -0.080996  -0.107629  -0.085522   0.078731   0.050686    -0.073699    -0.040299  -0.014779    0.021247   0.697109   0.530972  -0.081157  -0.059012  -0.
3   0.098893   0.118256   0.009421  -0.369062  -0.416766     0.014042     0.225334   0.263127    0.580772  -0.036305  -0.115165  -0.147493  -0.089232  -0.
4   0.219854   0.189592   0.162328   0.157346   0.144606     0.102740    -0.095765   0.037403   -0.069435   0.035393  -0.000373  -0.550741  -0.590385  -0.
```

```
plt.figure(figsize=(15,5))
sns.heatmap(df_comp,cmap='rainbow',annot=True,fmt='.2f',yticklabels=['PC0','PC1','PC2','PC3','PC4'])
plt.show()
```



## *2.8) Mention the business implication of using the Principal Component Analysis for this case study*

This heatmap and the color bar basically represent the correlation between the various feature and the principal component itself

1.PC0:Scholars : This Component Combines Top10% and Top25% Students in HSC, PhD candidates, Percentage of faculties with terminal degree--> It is Section where Scholars of College Comes.

2.PC1:Overall Admissions and Intake : No of Applications, No of Enrolled Students , No of Accepted Students, Full time Undergrad students, Part time Undergrad Students shows that they Mostly Contribute to Principal Component 1. we can name this Component as Overall Admissions and Intake as per our knowledge

3.PC2:Expenses: Personal and Books Expenses

4.PC3: Room and Boarding Expenses

5.PC4: Graduation Rate

So by using PCA we can Finally reduce 18 variables to 5 Principal Components.On basis of this we can give College Rankings,Acceptance Probability or Prediction of any random student will get admit or not from Particular college, College can decide its Fees on basis of This. College Can make Strategic Moves like Fees, How to attract more number of Scholars to College by providing them Scholarships, Fundings.

Whether to Increase Professors or Decrease it, Its impact on Admissions.

All this Kind Of Decision can be Made By Analysing data Ahead which is Produced by PCA.

Variables are Reduced, So now complex data is Looking Simpler, Without missing much Information