DIMENSIONALITY REDUCTION AND CLUSTERING

Machine Learning: Project 2

Hossain, Rayhan (rhossai2)

Table of Contents

Introduction

In machine learning classification of data is a very common problem. For classification, there are often too many factors of data on the basis of which the final classification is done. These factors are basically variables called features. The higher number of features can make the program complex. Sometimes, most of these features are correlated, and hence redundant. And for this reason, we need to do the dimensionality reduction to make our result better. Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. Clustering is another machine learning algorithm which groups multi-dimensional data-set into closely related groups. There are several clustering algorithms. Among these, K-means is the popular and simplest clustering algorithm.

About the Project

This project implements the above described dimensionality reduction and K-means algorithm without using any built-in module. Applying these implemented algorithms University Peers dataset is tested and analyzed for dividing them into closely related groups. Considering the given factors(information) of universities, it helps to cluster them.

Dataset

For this project, the given dataset has information of 57 universities including our one the University of Tennessee, Knoxville. And, for each of the universities there are 65 attributes or columns which describes a specific information of the university. Considering this dataset, this project finds the universities which are closely related to UTK, and clusters them into meaningful groups. A sample screenshot of the data is given below (partial view of the data).

	Name	IPEDS#	Carm R1	HBC	% Blk Total Students	% Hisp Total Students	US News top 65
0	**Univ. of Tennessee - Knoxville**	**221759**	**1**	**No**	**7**	**3**	**46**
1	**Univ. of Georgia**	139959	1	No	8	5	18
2	**Purdue Univ.**	243780	1	No	3	4	20
3	**Texas A&M Univ.**	228723	1	No	3	19	27
4	**Michigan State Univ.**	171100	1	No	7	4	33
5	**Univ. of Minnesota**	174066	1	No	4	3	26
6	**Clemson Univ.**		1	No	6	3	23
7	**Indiana Univ.**	151351	1	No	4	5	36
8	**Rutgers Univ.**	186380	1	No	8	12	25
9	**Auburn Univ.**	100858	2	No	7	3	43
10	**Iowa State Univ.**	153603	1	No	3	4	51
11	**NC State Univ.**	199193	1	No	6	4	38

Figure-1: raw data in excel

Data Preprocessing

Data were given in three formats. For preprocessing, the excel version of the data is a good option. Because, it makes the data visualizing and pre-editing easier. Among the 65 attributes, there are some unnecessary columns for clustering the data. Like the column HBC, there are others whose value is same for all the universities. So, removing these columns is a good option. Additionally, there are some columns with around 50% data missing. These may hamper the prediction. That's why those columns were also deleted from the actual dataset. After removing these, the final size of the clean data is a 57 X 57 matrix.

Data Formatting

After the preprocess, next step, data formatting starts with a 57 X 57 data matrix. But, the data yet is not clean to start the analyzation phase. There are some special characters in the data column like- $, -, comma, and multiple spaces. To clean these special characters a regular expression was written, and the data were converted into numeric value only.

```
new_data_list = []
for row in res:
    new_row = []
    for item in row:
        new_item = re.sub('[^A-Za-z0-9\.?]', '', item)
        if(new_item == '' or new_item == 'NA'):
            new_item = 0
        new_row.append(new_item)
    new_data_list.append(new_row)
```

Figure-2: data formatting

Now, all the data are numeric value. And, the missing data is replaced by zero. Though this is not a good idea. There are other options like- replacing with mean were also tested.

After this step, a new csv file was generated and imported into Pandas data frame. Pandas data frame is a very useful tool for handling some of the data processing functions. The clean numeric value only data set was the following one (partial view).

	IPEDS	BlkTotalStudents	HispTotalStudents	2017USNewstop65	2014MedSchool	VetSchool	TotalEnroll	GradEnroll
52	153658	3	6	35	1.0	0	30844	24
53	196088	6	5	43	1.0	0	29796	33
54	230764	1	10	52	1.0	0	31592	25
55	110671	4	33	58	1.0	0	21385	13
56	104179	4	23	62	1.0	0	42595	22

Figure-3: clean data (numeric)

Dimensionality Reduction

Dimensionality reduction is a series of techniques in machine learning and statistics to reduce the number of random variables to consider. Dimensionality reduction techniques are divided into two major categories: feature selection and feature extraction. Feature selection techniques find a smaller subset of a many-dimensional data set to create a data model. Feature extraction involves transforming high-dimensional data into spaces of fewer dimensions. Methods include principal component analysis, kernel PCA, graph-based kernel PCA, linear discriminant analysis and generalized discriminant analysis.

Principal Component Analysis (PCA) is a dimension-reduction tool that can be used to reduce a large set of variables to a small set that still contains most of the information in the large set. PCA seeks a linear combination of variables such that the maximum variance is extracted from the variables. It then removes this variance and seeks a second linear combination which explains the maximum proportion of the remaining variance, and so on. This is called the principal axis method and results in orthogonal (uncorrelated) factors. PCA analyzes total (common and unique) variance.

Eigenvectors: Principal components (from PCA - principal components analysis) reflect both common and unique variance of the variables and may be seen as a variance-focused approach seeking to reproduce both the total variable variance with all components and to reproduce the correlations.

Eigenvalues: Also called characteristic roots. The eigenvalue for a given factor measures the variance in all the variables which is accounted for by that factor. The ratio of eigenvalues is the ratio of explanatory importance of the factors with respect to the variables. If a factor has a low eigenvalue, then it is contributing little to the explanation of variances in the variables and may be ignored as redundant with more important factors. Eigenvalues measure the amount of variation in the total sample accounted for by each factor. A factor's eigenvalue may be computed as the sum of its squared factor loadings for all the variables.

K-Means Clustering

K-means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problems. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed, and an early group age is done. At this point we need to re-calculate k new centroids as barycenter of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the

nearest new center. A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more.

Given a set of observations (x1, x2, …, x_n), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k (≤ n) sets S = {S1, S2, …, S_k} so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

Figure-4: equation-1

where μ_i is the mean of points in S_i. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{\mathbf{x},\mathbf{y} \in S_i} \left\| \mathbf{x} - \mathbf{y} \right\|^2$$

Figure-5: equation-2

Implementation and Analysis

The implementation process starts with the clean data set of dimensions 57 X 57. At the beginning, data standardization was done for scaling the value. For doing the data standardization sklearn module was used. As the standardization is not the main focus sklearn was preferable to make the process faster.

```
1  from sklearn.preprocessing import StandardScaler
2  X_std = StandardScaler().fit_transform(X)
```

Figure-6: data standardization code

The classic approach to PCA is to perform the eigendecomposition on the covariance matrix Σ, which is a d × d matrix where each element represents the covariance between two features. The covariance between two features is calculated as follows:

$$\sigma_{jk} = \frac{1}{n-1} \sum_{i=1}^{N} \left(x_{ij} - \bar{x}_j\right)\left(x_{ik} - \bar{x}_k\right).$$

Figure-7: equation-3

There are several ways to calculate the covariance matrix. Here only one process is shown.

```
1  import numpy as np
2  mean_vec = np.mean(X_std, axis=0)
3  cov_mat = (X_std - mean_vec).T.dot((X_std - mean_vec)) / (X_std.shape[0]-1)
4  print('Covariance matrix \n%s' %cov_mat)
```

```
Covariance matrix
[[ 1.01785714e+00  1.25541804e-02 -3.17137467e-01 ...  3.38467789e-01
   3.32860105e-01  3.82293976e-01]
 [ 1.25541804e-02  1.01785714e+00 -3.66737589e-01 ... -1.38642671e-01
  -6.92302497e-02 -1.51886338e-01]
 [-3.17137467e-01 -3.66737589e-01  1.01785714e+00 ... -2.92645194e-04
  -1.16418242e-01 -3.46658698e-02]
 ...
 [ 3.38467789e-01 -1.38642671e-01 -2.92645194e-04 ...  1.01785714e+00
   9.24461915e-01  9.83898841e-01]
 [ 3.32860105e-01 -6.92302497e-02 -1.16418242e-01 ...  9.24461915e-01
   1.01785714e+00  8.84337162e-01]
 [ 3.82293976e-01 -1.51886338e-01 -3.46658698e-02 ...  9.83898841e-01
   8.84337162e-01  1.01785714e+00]]
```

Figure-8: covariance matrix code

In the next step, eigendecomposition is done on the covariance matrix:

```
cor_mat1 = np.corrcoef(X_std.T)

eig_vals, eig_vecs = np.linalg.eig(cor_mat1)

print('Eigenvectors \n%s' %eig_vecs)
print('\nEigenvalues \n%s' %eig_vals)
```

```
Eigenvectors
[[ 6.50875253e-02  4.08592563e-02  1.59321981e-01 ... -4.97220327e-06
   1.99426611e-11 -1.17704261e-10]
 [-6.70815436e-02  3.41510157e-02  8.87068473e-02 ... -5.50223345e-06
   2.20139816e-11 -1.30242087e-10]
 [ 1.86158055e-02  2.40483112e-02 -2.04253135e-01 ... -1.44122832e-05
   5.75643074e-11 -3.40998493e-10]
 ...
 [ 1.44986056e-01  2.18295677e-02  4.04055056e-02 ...  1.47480060e-04
  -5.94311138e-10  3.48889622e-09]
 [ 1.18605998e-01 -7.64746404e-02  9.64872515e-02 ... -1.06859582e-04
   4.30297383e-10 -2.52818228e-09]
 [ 1.50693939e-01  9.50854163e-03  3.69113167e-02 ... -5.66430147e-05
   2.28441286e-10 -1.33991236e-09]]

Eigenvalues
[ 2.16435286e+01  6.29553404e+00  5.44553938e+00  4.09938021e+00
  2.84946694e+00  2.49425974e+00  2.08388444e+00  1.75037901e+00
  1.51954264e+00  1.36212266e+00  1.01915308e+00  9.59682563e-01
  7.06443529e-01  6.70550553e-01  5.71160340e-01  4.77612105e-01
  3.96776636e-01  3.59507224e-01  3.42126922e-01  2.67056844e-01
  2.39789947e-01  1.99793919e-01  1.74564942e-01  1.52181942e-01
  1.39806529e-01  1.15892434e-01  1.06076937e-01  9.32682205e-02
  6.89607133e-02  6.72068131e-02  6.02329886e-02  4.76555274e-02
  4.21651318e-02  3.66657883e-02  2.68787416e-02  2.25917455e-02
  1.95504986e-02  1.63235916e-02  1.36460064e-02  1.03064925e-02
  7.85143539e-03  6.58423808e-03  4.41849567e-03  4.15755672e-03
  3.27205118e-03  2.76264575e-03  1.46162143e-03  1.01095046e-03
  5.86395557e-04  3.82521084e-04  1.34409938e-04  7.66988544e-05
  2.99750690e-05  4.59934918e-06  3.11533424e-11  2.35339125e-17
 -3.35367944e-16]
```

Figure-9: eigenvectors and eigenvalues

While the eigendecomposition of the covariance or correlation matrix may be more intuitive, most PCA implementations perform a Singular Vector Decomposition (SVD) to improve the computational efficiency. Here an SVD was also performed to confirm that the result is indeed the same. From the SVD, the singular values are extracted. By plotting these values of s, it is easier to find the best k. For this project the best value was chosen k = 10. Here the concept of elbow method was applied.

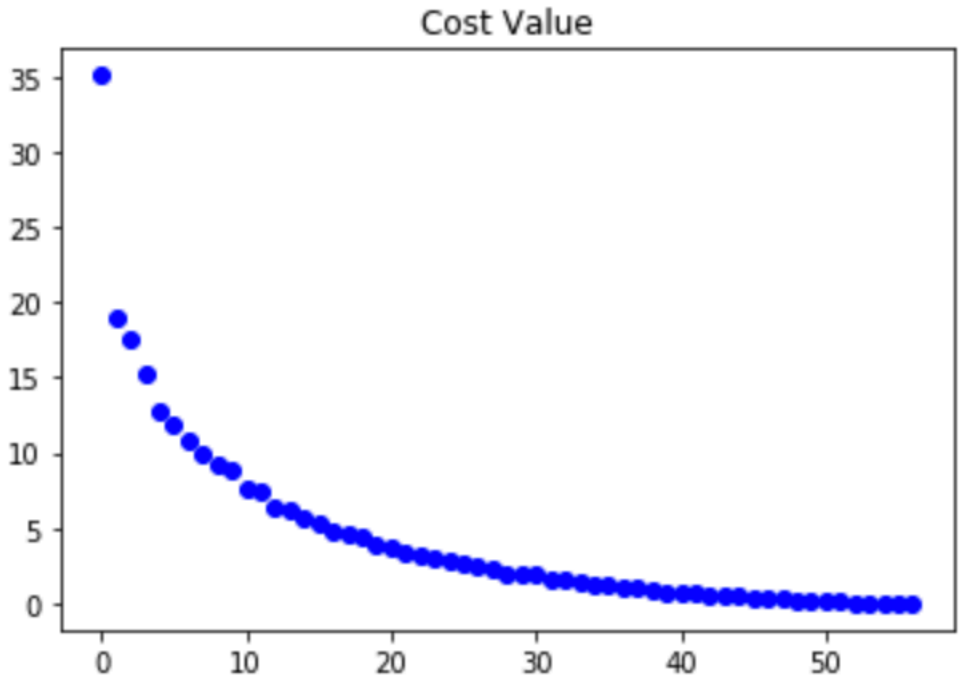

Figure-10: find k (elbow method)

In order to decide which eigenvector(s) can dropped without losing too much information for the construction of lower-dimensional subspace, it was necessary to inspect the corresponding eigenvalues: The eigenvectors with the lowest eigenvalues bear the least information about the distribution of the data; those are the ones can be dropped. In order to do so, the common approach is to rank the eigenvalues from highest to lowest in order choose the top k eigenvectors.

Eigenvalues in descending order (Top ten):
21.64352863775041
6.2955340363143355
5.445539381313991
4.099380212712964
2.849466937959404
2.4942597377483637
2.0838844389599123
1.7503790146050393
1.5195426426373189
1.3621226628420573

Then the data were reduced to a k-dimensional space. Below is the projection of data for the two PC values.

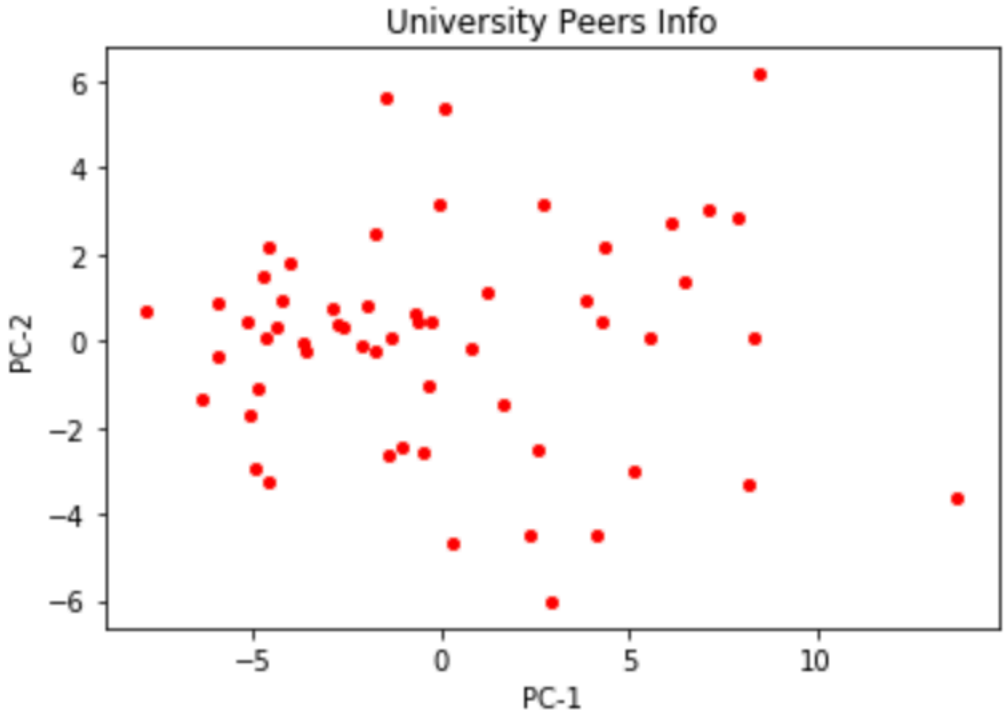

Figure-11: University peers data (based on 2 PCs)

After doing all these, the next part of the project starts which is K-means clustering. Before applying the PC when the K-mean was applied into the raw data, the result clusters was not satisfying.

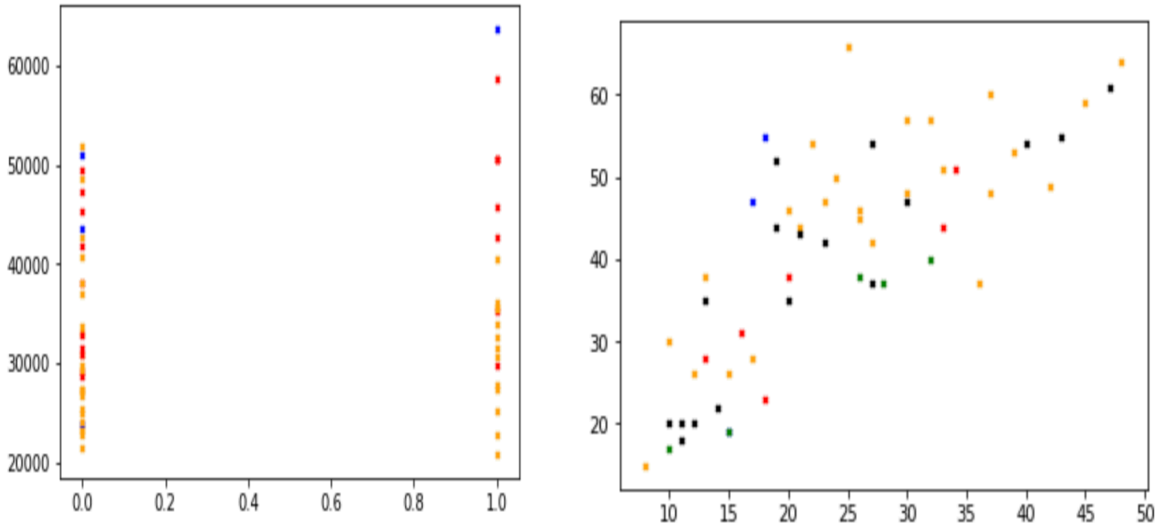

Figure-12: K-means on raw data (k = 4, 6)

After applying K-means for two PC's, and for various values of k, the resulting cluster was good and satisfying. And we can easily figure the peer universities.

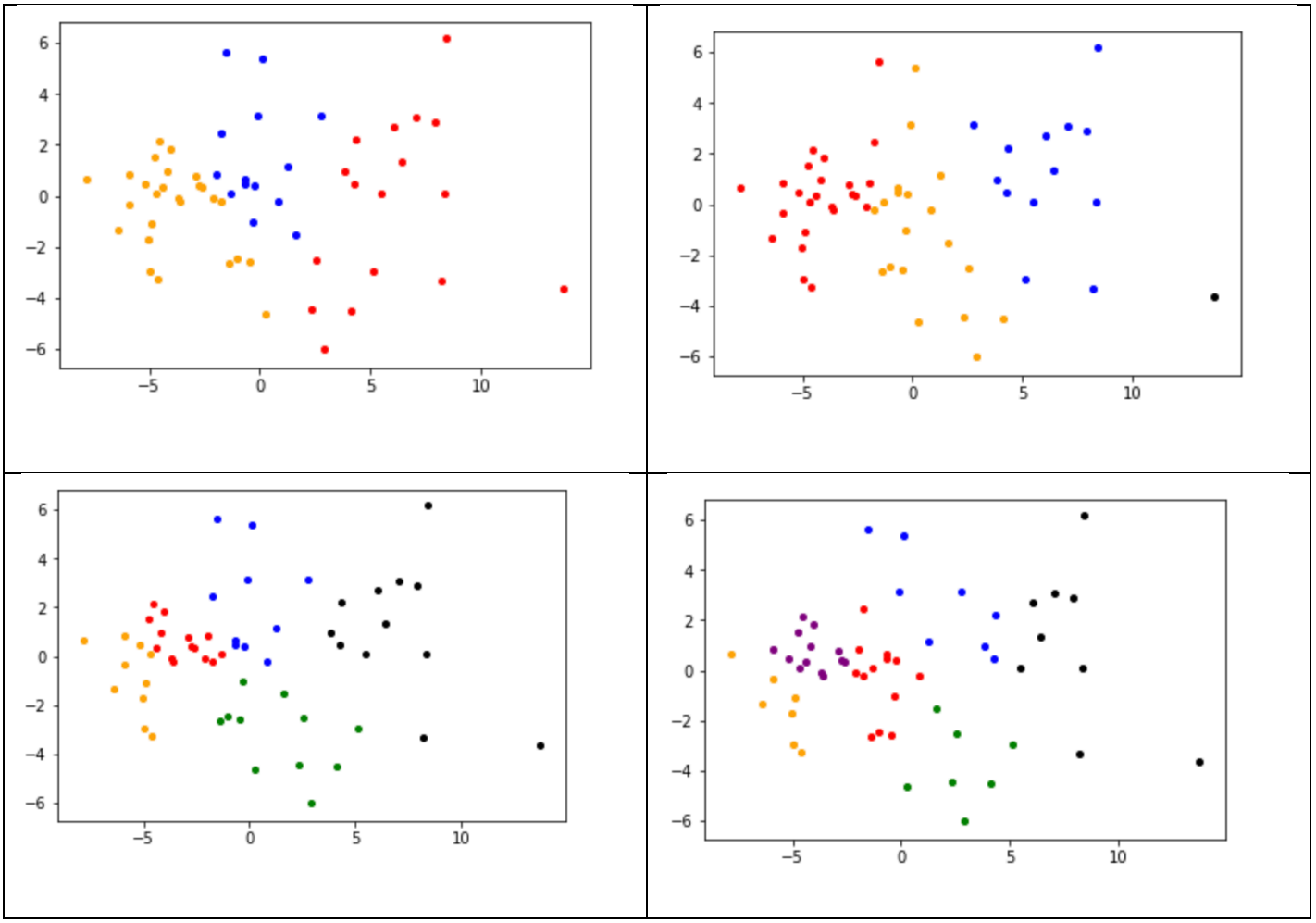

Figure-13: K-means on 2 PC's (k = 3,4,5,6)

Calculating Dunn index

The Dunn index is a metric for evaluating clustering algorithms. It is an internal evaluation scheme, where the result is based on the clustered data itself. As do all other such indices, the aim is to identify sets of clusters that are compact, with a small variance between members of the cluster, and well separated, where the means of different clusters are sufficiently far apart, as compared to the within cluster variance. For a given assignment of clusters, a higher Dunn index indicates better clustering.

The Dunn index for above four cluster is: 0.4821818962425446, 0.28985675342650447, 0.3321987664742322, and 0.41114000078642426.

Extra Credit Work

To get the extra credit, I started with the given large dataset- IPEDS-big-trimmed.csv. When I applied k-means without the dimensionality reduction, the result was a mess. It was really tough to find the correlation among the universities.
Two example screenshots for various dimensions is given below.

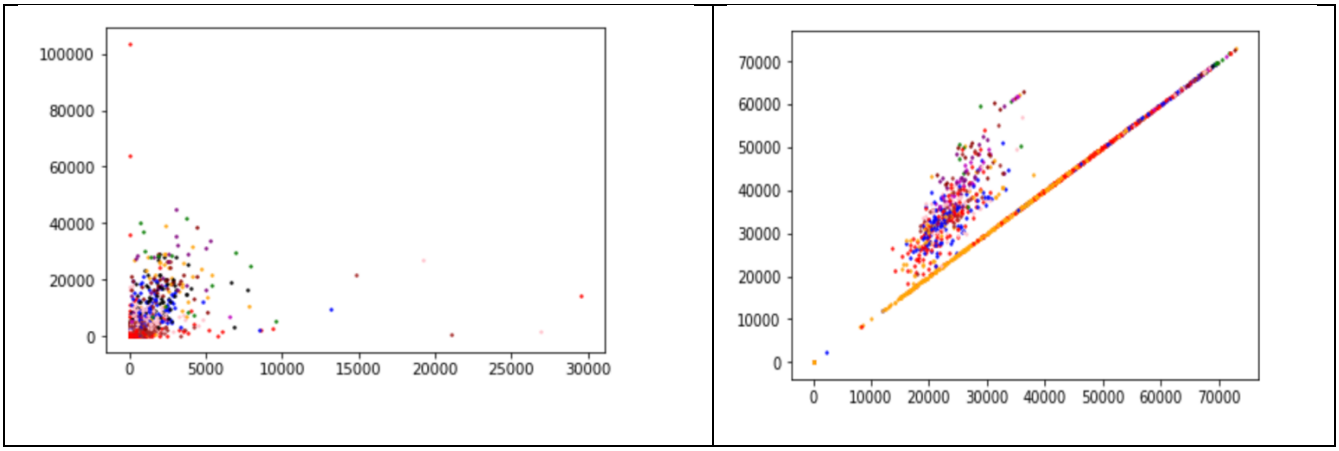

Figure-14: K-means on raw data (k = 10)

After applying the PC's, for first two PC's the clustering was improved, and looks far better than previous.

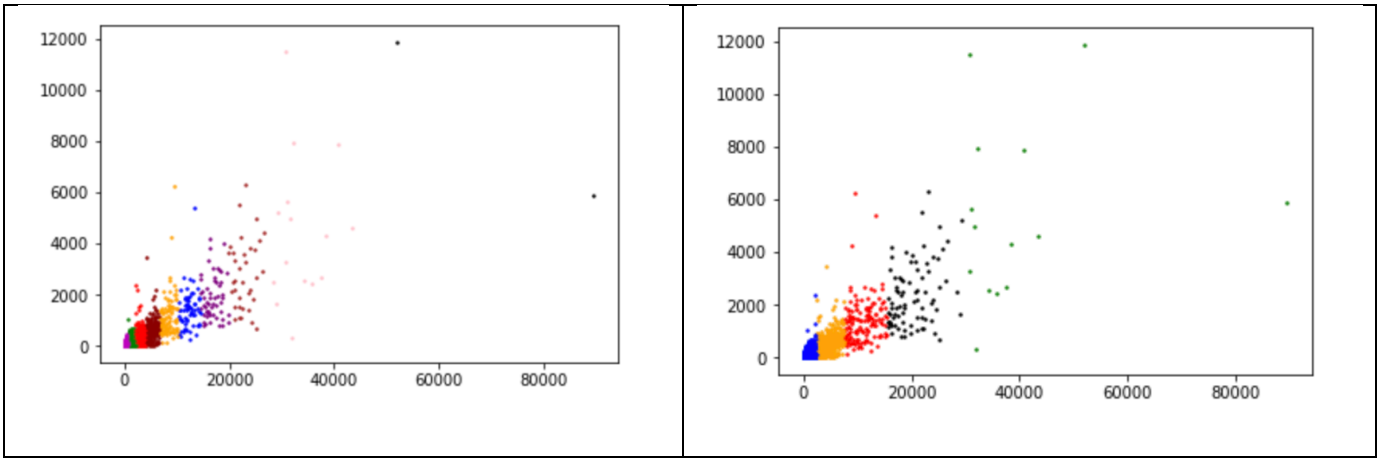

Figure-15: K-means on 2 PC's (k = 10, 5)

Conclusion

Reducing the unnecessary and less important dimensions improve the quality of clustering. Higher number of features always makes the data and algorithm complex. To do this task, main challenge was the data processing. It took almost 50% of the development time.

Reference

[1] ftp://statgen.ncsu.edu/pub/thorne/molevoclass/AtchleyOct19.pdf
[2]https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxkYXRhY2x1c3RlcmluZ2FsZ29yaXRobXN8Z3g6N2U0N2JjZTEzMTMxNjc3ZA
[3]https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxkYXRhY2x1c3RlcmluZ2FsZ29yaXRobXN8Z3g6M2QwZjdlZjRiMDM3ZGNhMQ
[4] https://en.wikipedia.org/wiki/K-means_clustering