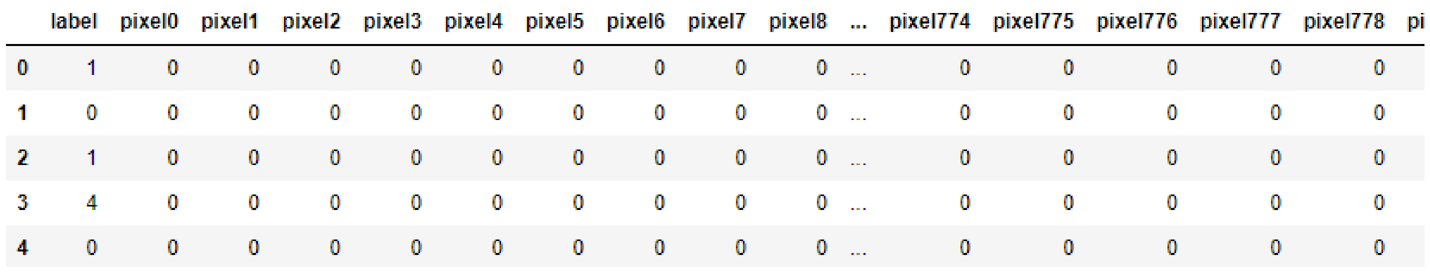
MSDS 422 Assignment 5
Husein Adenwala

Introduction

The Purpose of this assignment is to compare K-mean clustering by using Principal component analysis for dimension reduction and K-Mean clustering without using principal component analysis. I will use the MINST dataset from Kaggle that contains 28x28 pixel images of hand-written numbers where each column represents a pixel.

Data preparation, exploration, visualization

The train data set has 42000 rows and 785 columns. Each row has 784 columns that represent 28x28 pixels for the hand-written images and a label column that represents that number that the image represents. The image below shows the general structure of the datasets.



	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pi
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	

Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer ranging from 0 to 255.

Using the pixel columns for each row, I plotted the hand-written image and assigned the value from the label columns as shown below.

1 0 1 4 0 0 7 3 5 3
8 9 1 3 3 1 2 0 7 5
8 6 2 0 2 3 6 9 9 7
8 9 4 9 2 1 3 1 1 4
9 1 4 4 2 6 3 7 7 4
7 5 1 9 0 2 2 3 9 1
1 1 5 0 6 3 4 8 1 0

Principal component analysis

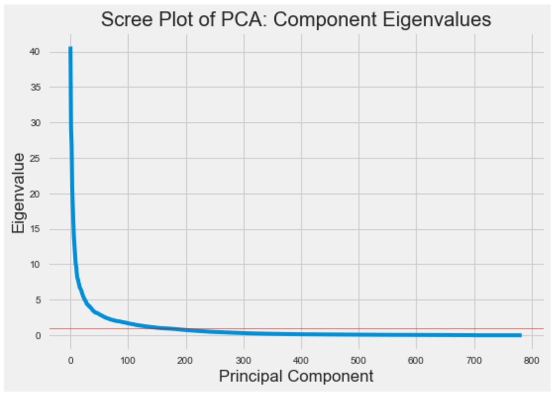
I used the scree plot and total percent variance plot to determine the number of components to select for the PCA data transformation. I wanted to reduce the dimension but also keep the approximately 90% variation in the dataset in order for the dimension reduction to be effective for any modeling purpose.

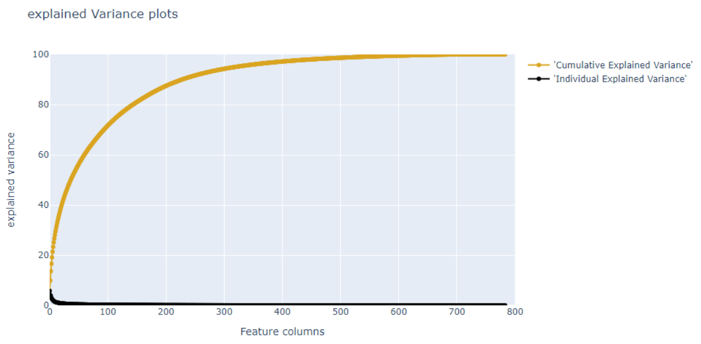
A scree plot shows the eigenvalues on the y-axis and the number of factors on the x-axis. The Scree plot displays how much variation each principal component captures from the dataset.

The total percent variance plot is a cumulative plot that shows the total variability of each principal component axis.

In this case, the total percent variance plot is more useful compared to the scree plot as the steep slope in the scree plot is misleading. It indicates a lower optimum number of principal components (e.g. it looks as though less than 100 would be sufficient) but in order to represent 90% variation, I included approximately 220 components (as indicated by the total percent variance plot).

The first two principal components only show 13% variation and might not be very helpful to visualize the clusters.





Review research design and modeling methods

For this analysis, I used k-means clustering model that tries to partition the dataset into distinct non-overlapping subgroups/clusters, where each datapoint belongs to only one group. It tries to make the intra-cluster datapoints as similar as possible while also keeping the clusters as different/far as possible.

In k-means clustering, I pre-defined the number of clusters. There are several metrics/methods to identify the optimal number of clusters but the task is largely based on subjective decisions by the analyst or subject matter expert.

MINST digits data has 10 labels that can be divided into clusters using k-means but MINST is high dimensional with 784 columns and very high-dimensional spaces. Euclidean distances tend to become inflated (curse of high dimensionality); therefore, I used two k-means models, one with PCA prior to k-means clustering to alleviate the curse of high dimensionality and one without PCA on the original data. I evaluated which of these two models performed better.

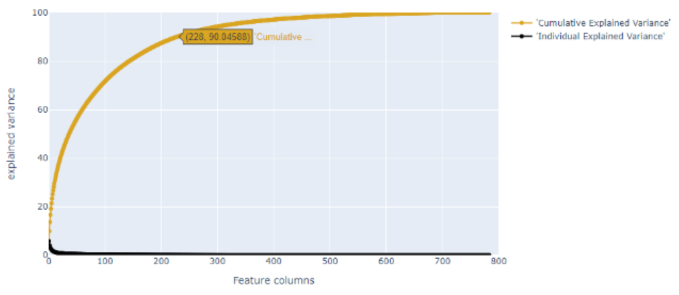
Review results, evaluate models

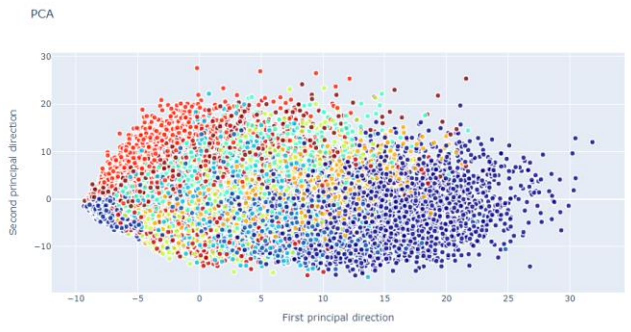
K-means clustering is sensitive to outliers and generally, I would standardize the data. PCA requires standardized data and in the 1st model, I standardized the data before performing PCA. In the second model, I used the original data without standardizing as the data does not have outliers and performed better without standardization.

K-Means clustering using principal component analysis (PCA):

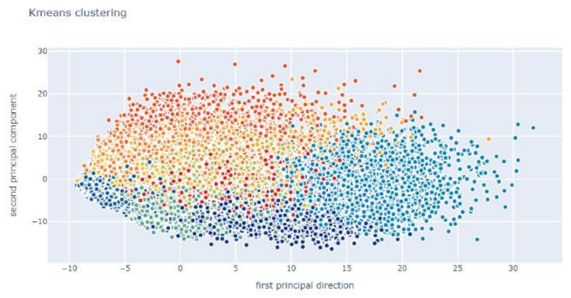
To represent 90% variation in the data set, I will use the first 228 principal components. This way I reduced the number of columns from 784 to 228, which is significant in terms of performance.

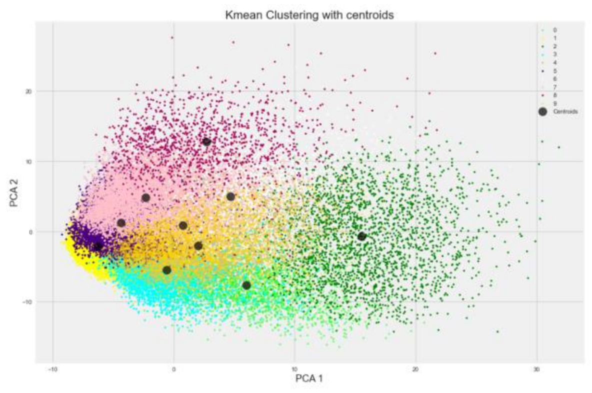
I created a scatterplot using the first 2 principal axes, PCA 1 and PCA 2, and color coding it by label to identify any pattern. As the PCA1 and 2 only covers 13% of the variation, the plots are not very useful to identify any patterns as there is a lot of overlap between the clusters, and there are no clear cluster boundaries. Also, the silhouette plot shows unbalanced clusters.

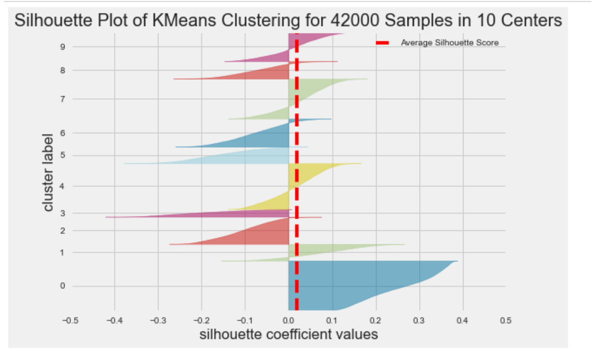




Creating 10 clusters and plotting them:







```
<matplotlib.axes._subplots.AxesSubplot at 0x25a401247f0>
```

```
1 visualizer.silhouette_score_
```

```
0.019812050736058535
```

```
n_clusters: 10,          n_samples 42000,         n_features 228
__________________________________________________________________________________
init           time     inertia          homo    compl   v-meas  ARI     AMI     silhouette
PCA Kmeans     6.76s    22058380         0.419   0.441   0.430   0.319   0.429   0.0199
```

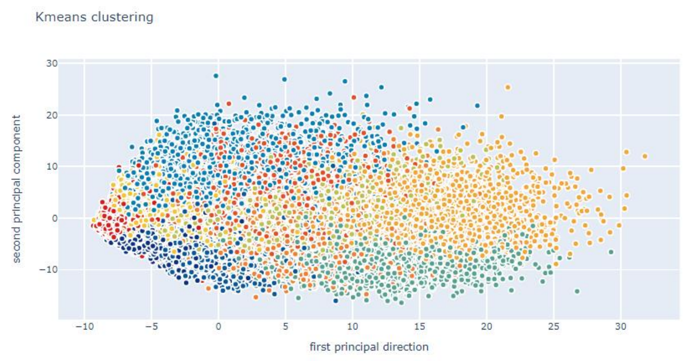
Silhouette score comparison for number of clusters from 2 to 11:

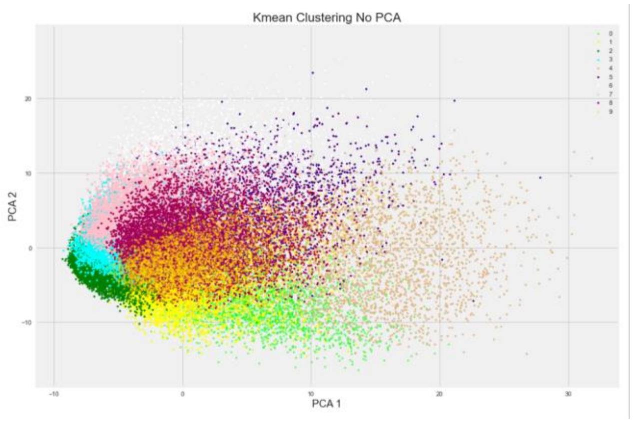
```
For n_clusters = 2 The average silhouette_score is : 0.1358503002899895
For n_clusters = 3 The average silhouette_score is : 0.05045011432823784
For n_clusters = 4 The average silhouette_score is : 0.045188306116045615
For n_clusters = 5 The average silhouette_score is : 0.038562754177665116
For n_clusters = 6 The average silhouette_score is : 0.008491663634972224
For n_clusters = 7 The average silhouette_score is : 0.000591911358764725
For n_clusters = 8 The average silhouette_score is : 0.009428174969705928
For n_clusters = 9 The average silhouette_score is : 0.019950326259297685
For n_clusters = 10 The average silhouette_score is : 0.018820192775440295
```

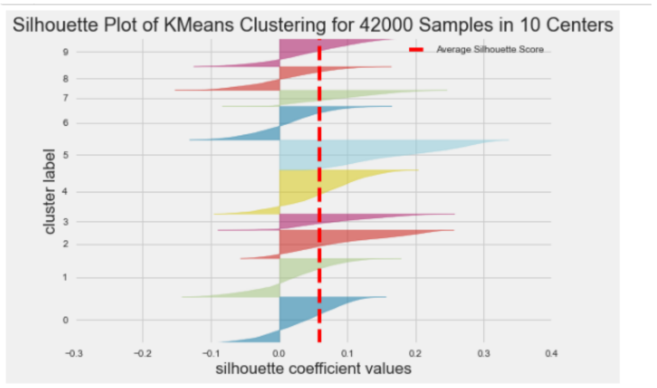
K-Means clustering no PCA:

I performed k-means clustering on all original 784 columns. I calculated the performances, plotted the clusters and imposed the model cluster on 1st and 2nd principal component that I got from the PCA analysis above.

Similar to the plots above, there is a lot of overlapping between the clusters and there are no clear cluster boundaries here either, although the silhouette plot shows more balanced clusters compared to above.







```
: <matplotlib.axes._subplots.AxesSubplot at 0x25a401428e0>

:   1 visualizer.silhouette_score_

: 0.0586296812620352
```

Silhouette score comparison for number of clusters from 2 to 11:

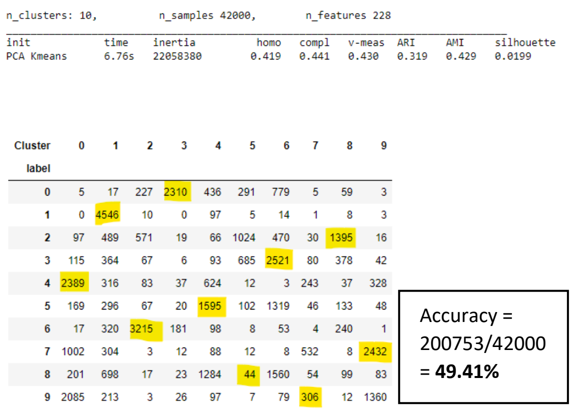
```
For n_clusters = 2 The average silhouette_score is : 0.12836395248974397
For n_clusters = 3 The average silhouette_score is : 0.0438993731671025
For n_clusters = 4 The average silhouette_score is : 0.037154085105357094
For n_clusters = 5 The average silhouette_score is : 0.029183908527397686
For n_clusters = 6 The average silhouette_score is : -0.0017514345378291535
For n_clusters = 7 The average silhouette_score is : -0.010150513358895883
For n_clusters = 8 The average silhouette_score is : -0.0007203086316785063
For n_clusters = 9 The average silhouette_score is : 0.00819952852197786
For n_clusters = 10 The average silhouette_score is : 0.006587709106964628
For n_clusters = 11 The average silhouette_score is : 0.005635315030916063
For n_clusters = 12 The average silhouette_score is : 0.006138354244299793 4
```

Summary of the two models' data

The k-means no PCA model performed slightly better. It had a higher Adjusted Rand index and homogeneity score which suggest better clustering. The silhouette score is significantly higher for the k-means with PCA model, but silhouette score is a better metric to determine the optimal clusters and may not be the best metric to compare models. The silhouette plot of the no PCA model shows more balanced clusters.

It took the k-means without PCA twice the amount of time to create the model. This could be a consideration for an extremely large dataset.

K-means with PCA

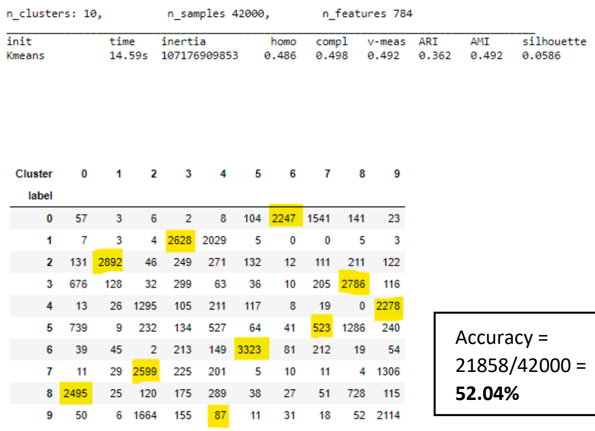


```
n_clusters: 10,         n_samples 42000,        n_features 228
__________________________________________________________________________________
init            time    inertia         homo    compl   v-meas  ARI     AMI     silhouette
PCA Kmeans      6.76s   22058380        0.419   0.441   0.430   0.319   0.429   0.0199
```

Cluster label	0	1	2	3	4	5	6	7	8	9
0	5	17	227	2310	436	291	779	5	59	3
1	0	4546	10	0	97	5	14	1	8	3
2	97	489	571	19	66	1024	470	30	1395	16
3	115	364	67	6	93	685	2521	80	378	42
4	2389	316	83	37	624	12	3	243	37	328
5	169	296	67	20	1595	102	1319	46	133	48
6	17	320	3215	181	98	8	53	4	240	1
7	1002	304	3	12	88	12	8	532	8	2432
8	201	698	17	23	1284	44	1560	54	99	83
9	2085	213	3	26	97	7	79	306	12	1360

Accuracy = 200753/42000 = **49.41%**

K-Means no PCA



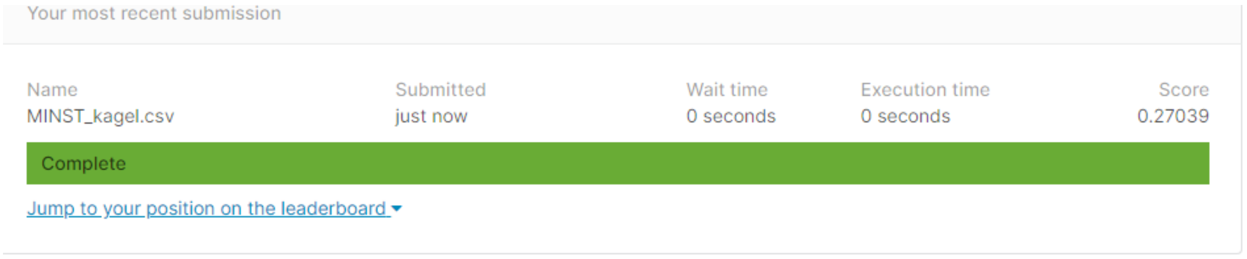
```
n_clusters: 10,         n_samples 42000,        n_features 784
__________________________________________________________________________________
init            time    inertia         homo    compl   v-meas  ARI     AMI     silhouette
Kmeans          14.59s  107176909853    0.486   0.498   0.492   0.362   0.492   0.0586
```

Cluster label	0	1	2	3	4	5	6	7	8	9
0	57	3	6	2	8	104	2247	1541	141	23
1	7	3	4	2628	2029	5	0	0	5	3
2	131	2892	46	249	271	132	12	111	211	122
3	676	128	32	299	63	36	10	205	2786	116
4	13	26	1295	105	211	117	8	19	0	2278
5	739	9	232	134	527	64	41	523	1286	240
6	39	45	2	213	149	3323	81	212	19	54
7	11	29	2599	225	201	5	10	11	4	1306
8	2495	25	120	175	289	38	27	51	728	115
9	50	6	1664	155	87	11	31	18	52	2114

Accuracy = 21858/42000 = **52.04%**

Kaggle Submission

User name = haden



Your most recent submission

Name	Submitted	Wait time	Execution time	Score
MINST_kagel.csv	just now	0 seconds	0 seconds	0.27039

Complete

Jump to your position on the leaderboard

Implementation and programming

Using Python Pandas package, I loaded the MINST Train and test csv file to the Data Frame object.

Exploratory data analysis was done utilizing the function in the Pandas, Numpy, plotly and matplotlib packages.

I then created a pandas series object by sub-setting the label column and subsequently dropping the column from the test data frames.

I created two copies of the data frame. I used the scikit-learn standard scalar function to normalize the data on the first copy; the second I left as is.

I used PCA() function from scikit learn to reduce the number of dimensions in the standardized data from 784 to 228 and then created a clustering model using KMeans() function from scikit learn to group data into 10 clusters. I used matplotlib and plotly to plot the data on PCA1 and PCA2 and used the metrics method from scikit learn to get performance metrics such as adjusted rand index homogeneity score and silhouette score. I used pandas crosstab to create a confusion matrix (the cluster labels were mislabeled).
I used the python time it function to time the performance of the k-means model using PCA.

I used all the original data (784 columns) for the next model using the KMeans() function from scikit learn to group data into 10 clusters. I used matplotlib and plotly to plot the data on PCA1 and PCA2 and used the metrics method from scikit learn to get the performance metrics such as adjusted rand index homogeneity score and silhouette score.
I again used pandas crosstab to create a confusion matrix (the cluster labels were mislabeled).

I used marplot lib and scikit learn functions to create the scree plot and silhouette plot for both models.

I selected the k-means model no PCA to predict the labels on the test data but before that, I created a dictionary to rename the cluster labels by utilizing the confusion matrix by selecting the maximum frequency in a row/column to be the actual label.

I used the k-means no PCA model to predict the label and finally I re-labeled the data using the dictionary created above to create final predicted labels that was submitted to Kaggle.

Exposition, problem description, and management recommendations

Based on my analysis above, I would recommend using k-means clustering without PCA. This model performed better based on the majority of performance metrics such as Rand index and homogeneity score. Also, the silhouette plot shows more balanced clusters. The re-labeled calculated accuracy was better in the without PCA model as well.

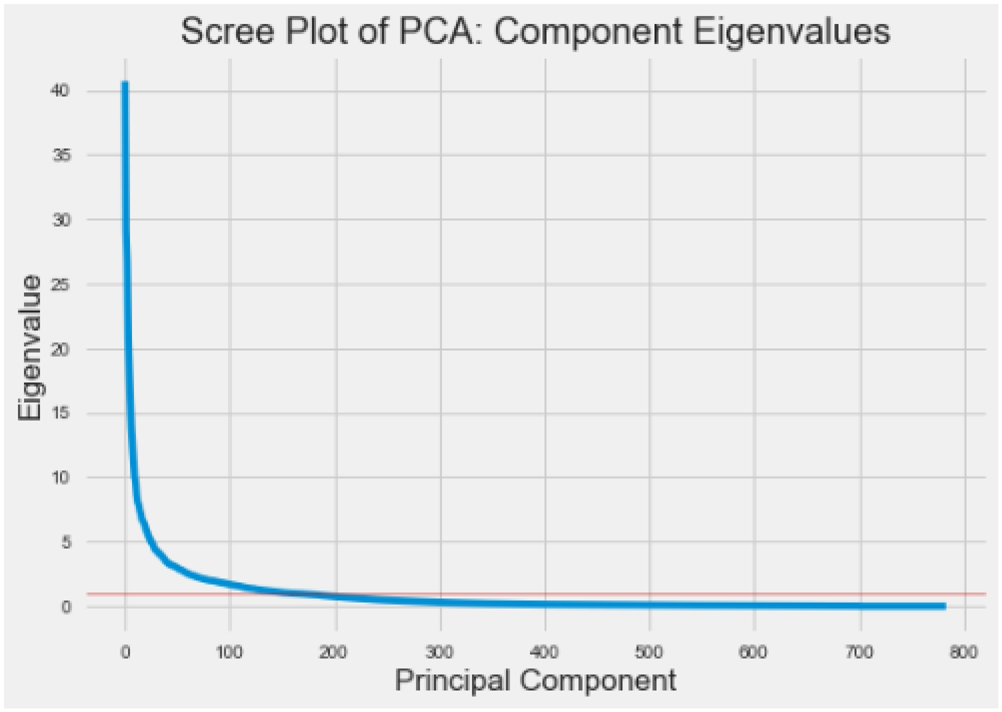
K-means clustering is one of the most popular clustering algorithms. The limitations of k-means are that it primarily performs well only if the clusters have convex, non-overlapping boundaries and attributes.
MINST database has high dimensions and PCA can reduce the dimensions but it’s not enough reduction in the dimensions to transform the high-dimensional input into meaningful outputs of smaller n-dimensional. In order to improve performance, we will need a better method to extract the relevant and meaningful features from the data.

Neither model performed well in terms of clustering but relatively, the k-means without PCA performed better.

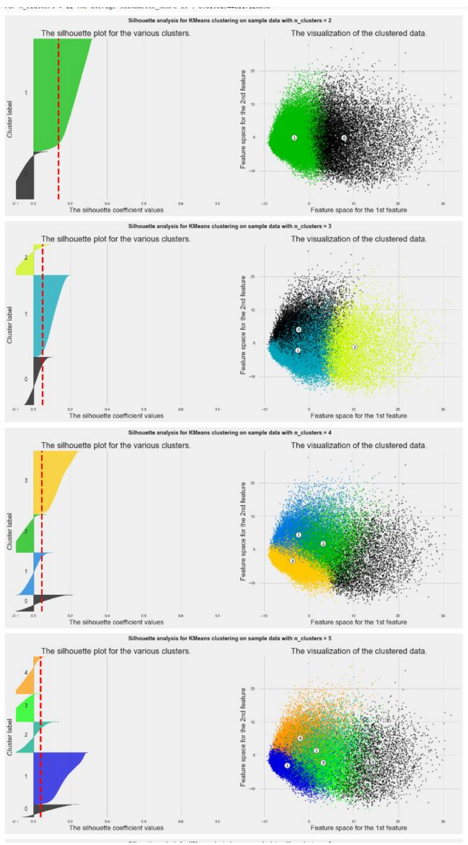
Appendix

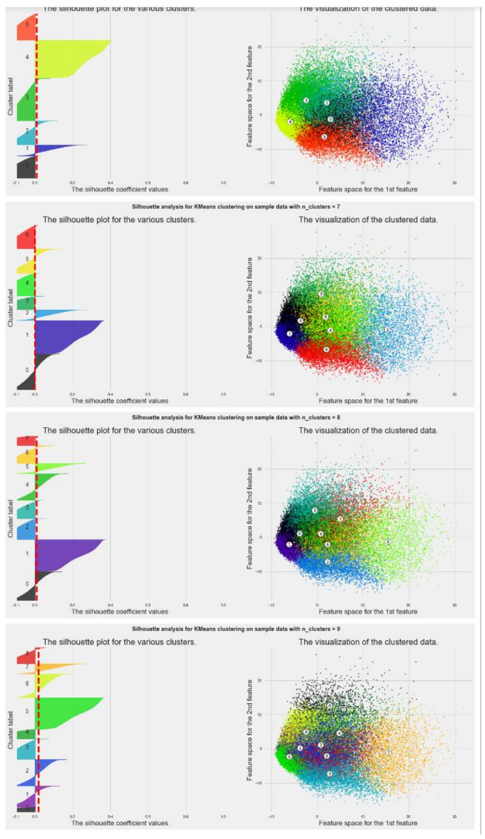
Scree plot

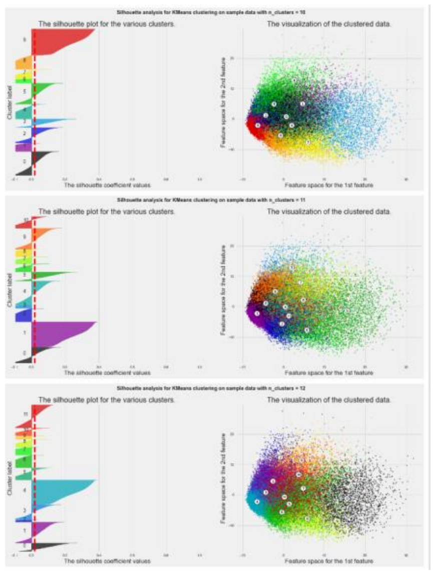
```
pca = PCA()
pca.fit(X_std_)

def scree_plot():

    from matplotlib.pyplot import figure, show
    from matplotlib.ticker import MaxNLocator

    ax = figure().gca()
    ax.plot(pca.explained_variance_)
    ax.xaxis.set_major_locator(MaxNLocator(integer=True))
    plt.xlabel('Principal Component')
    plt.ylabel('Eigenvalue')
    plt.axhline(y=1, linewidth=1, color='r', alpha=0.5)
    plt.title('Scree Plot of PCA: Component Eigenvalues')
    show()

scree_plot()
```



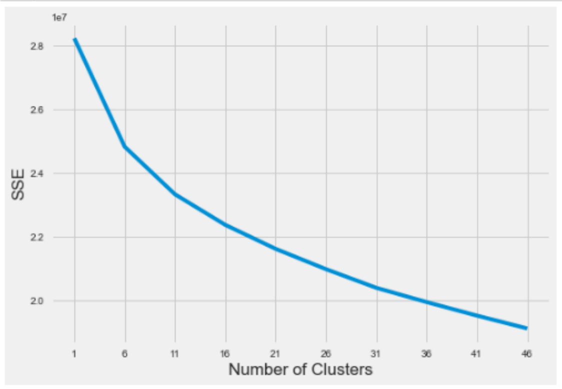
Silhouette graph comparing number of cluster 2 to 11

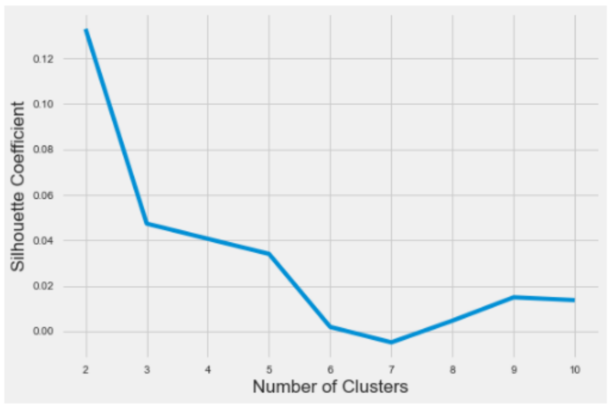






Scree plot and silhouette coefficient plot to determine the optimal number of clusters





```python
kl = KneeLocator(range(1, 50,5), sse, curve="convex", direction="decreasing")


kl.elbow
```

```
11
```