ECE 156B: Unsupervised Learning Clustering

Vicki Chen
University of California, Santa Barbara

1 Introduction

In this assignment, we are given the task to use unsupervised learning clustering algorithms, to generate classifications. We are given a dataset called *glas_data_labeled.csv* which contains different chemical compounds as features and the type of glass as the class label.
There are 9 attributes: Refractive index (RI), Sodium (Na), Potassium (K), Magnesium (Mg), Aluminum (Al), calcium (Ca), Silicon (Si), Barium (Ba) and iron (Fe) content.
7 different classes: Building windows Float processed glass, Vehicle windows float processed glass, building windows non-float processed glass, vehicle windows non-float processed glass, containers non-window glass, table ware non-window glass and headlamps non-window glass, and 214 total instances.
By using different clustering methods, we are able to group the unlabeled dataset by examining their features and finding the correlation between them.

2 Method

2.1 Setup

To load and set up the data, I followed a similar format to how I loaded the dataset for the first assignment. Since this is a csv file, we will be using pandas instead of numpy.

```
data = pd.read_csv('glass.csv')
X = data[['RI', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe']]
y = data['Type']
```

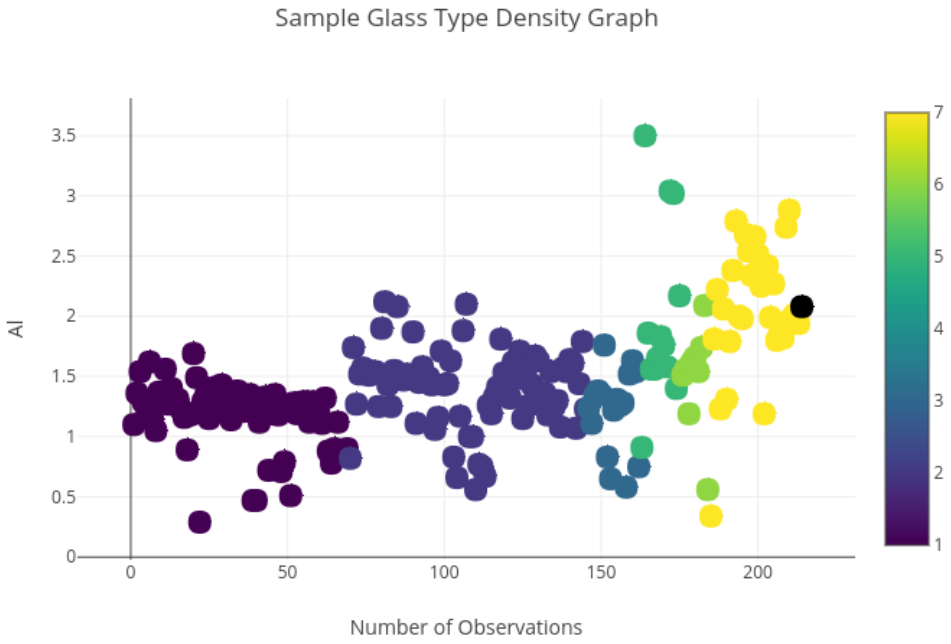
Similar to our setup for the first homework, we will create a loop to apply different clustering methods to our dataset and calculate the output accuracy.

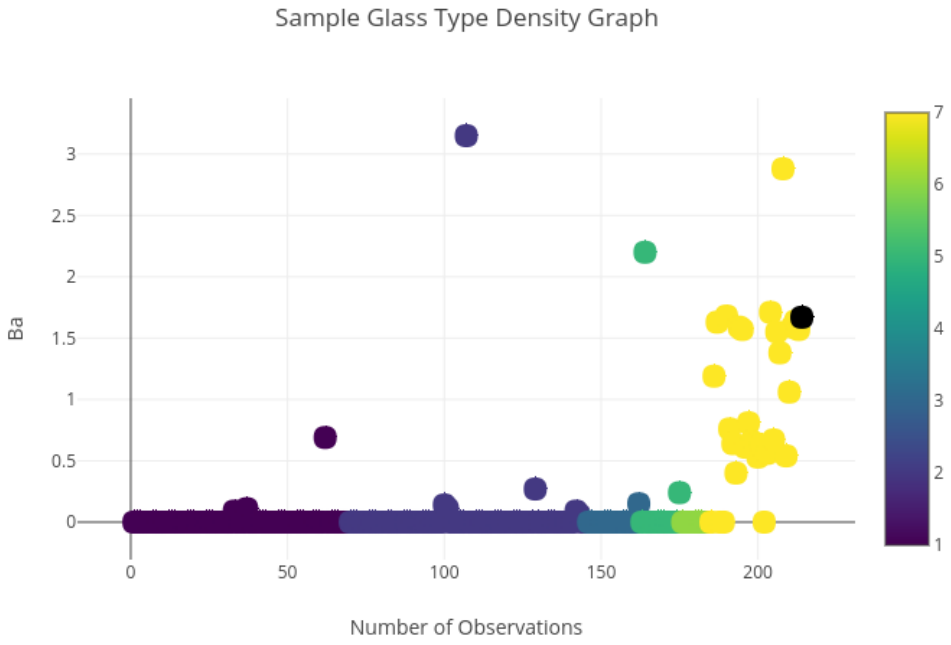
```
names = ["K-Means", "Affinity Propagation", "Spectral Clustering","Mean
Shift","Agglomerative Clustering","DBSCAN","Birch"]
```

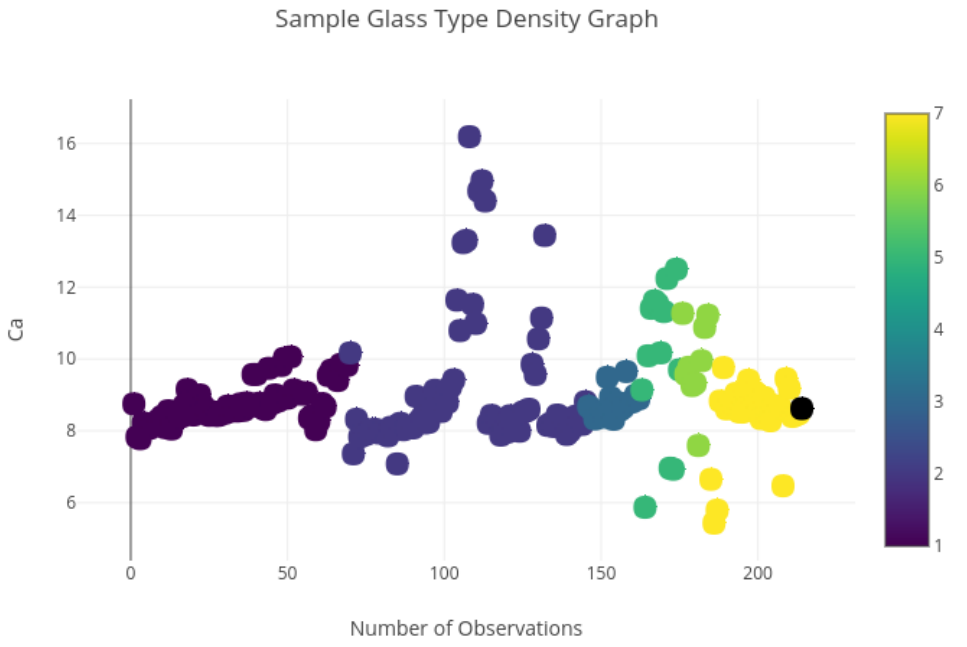
```
for name, cl in zip(names, clusters):
    labels = cl.fit(X).labels_
    score = metrics.fowlkes_mallows_score(labels, y)
```

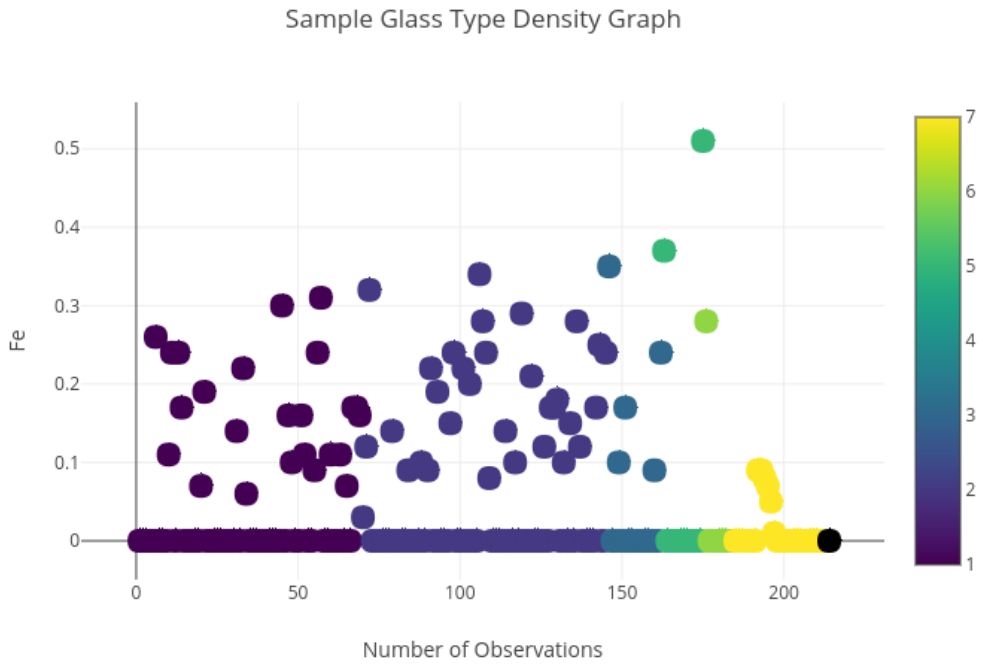
```
    print(name + ': ' + ('%.9f' % score).lstrip('0') + '\n')
```

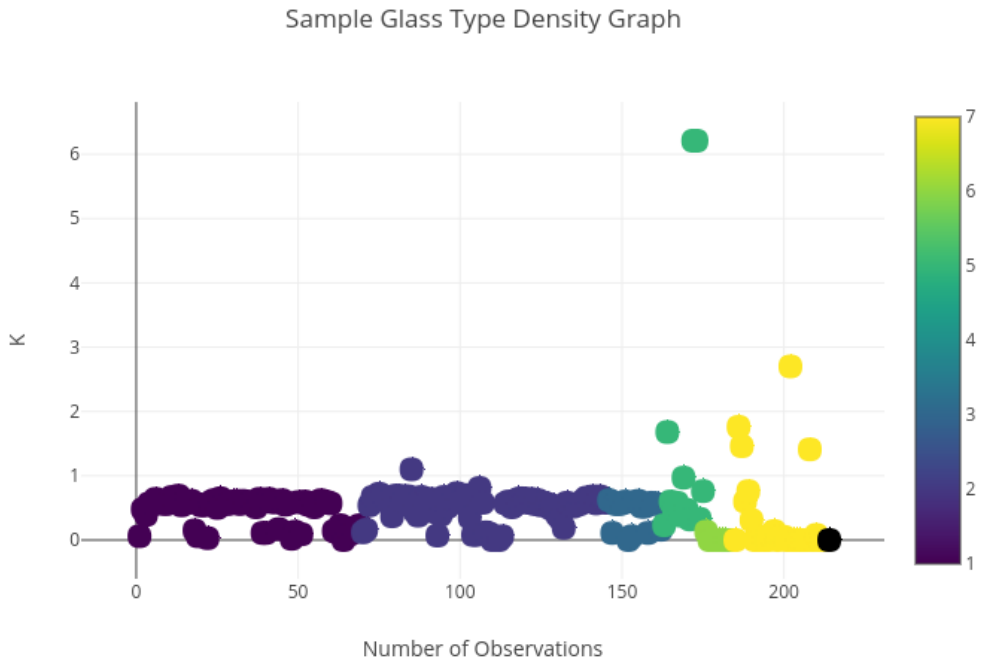
2.2 Density Plots

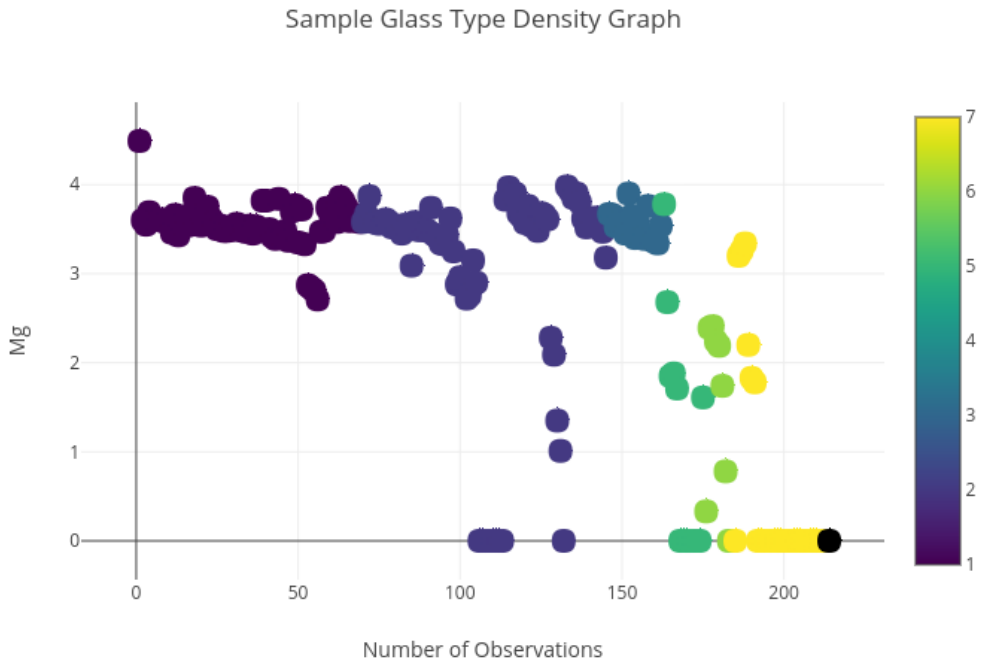


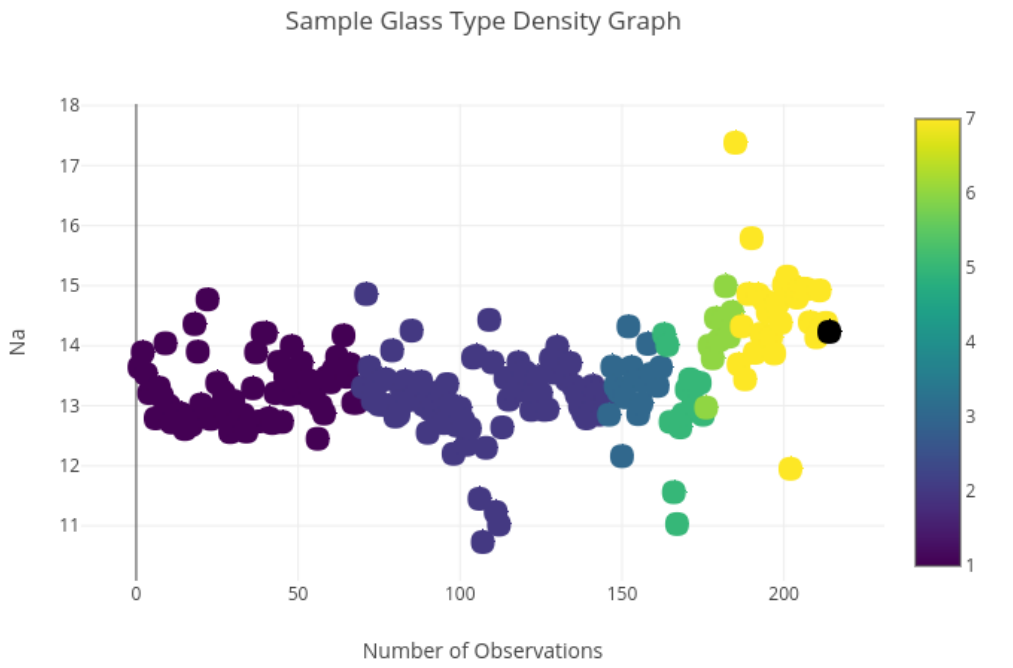


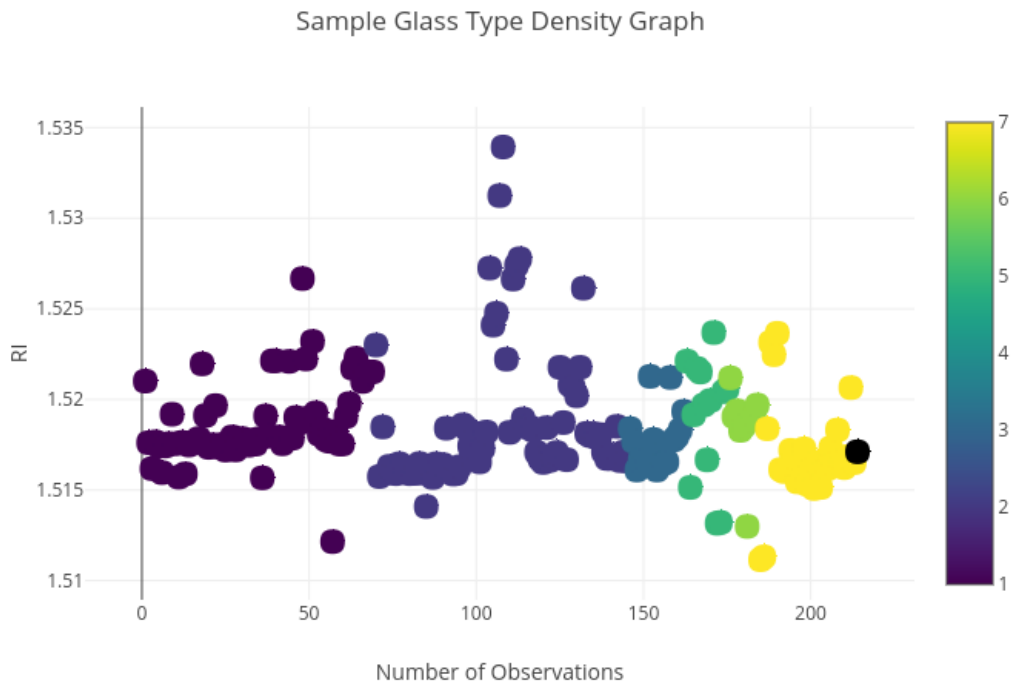

Sample Glass Type Density Graph
Ca
16
14
12
10
8
6
0
50
100
150
200
Number of Observations
7
6
5
4
3
2
1

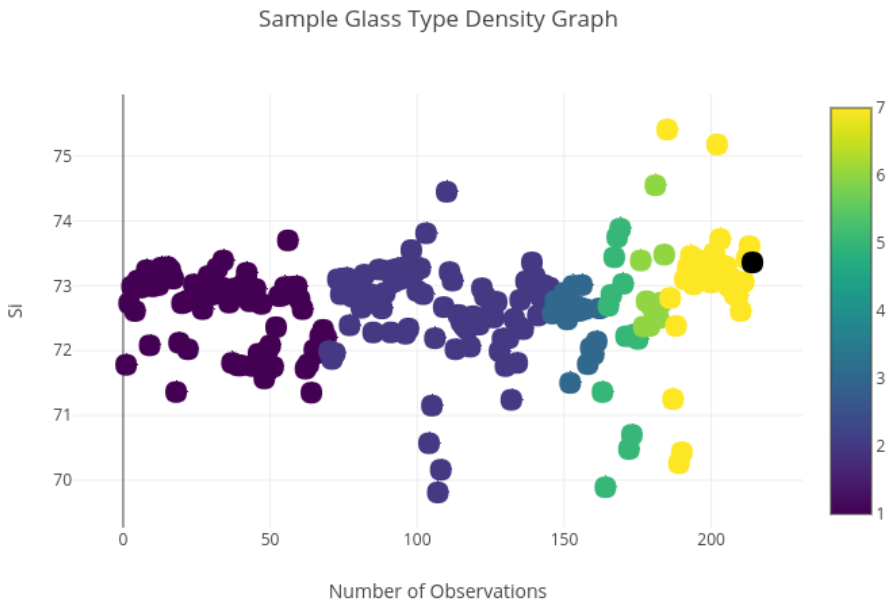

Sample Glass Type Density Graph
Fe
0.5
0.4
0.3
0.2
0.1
0
0
50
100
150
200
Number of Observations
7
6
5
4
3
2
1


Sample Glass Type Density Graph
K
Number of Observations
0
1
2
3
4
5
6
0
50
100
150
200
1
2
3
4
5
6
7


Sample Glass Type Density Graph
Mg
Number of Observations
0
1
2
3
4
0
50
100
150
200
1
2
3
4
5
6
7


Sample Glass Type Density Graph
Na
18
17
16
15
14
13
12
11
0
50
100
150
200
Number of Observations
7
6
5
4
3
2
1


Sample Glass Type Density Graph
RI
1.535
1.53
1.525
1.52
1.515
1.51
0
50
100
150
200
Number of Observations
7
6
5
4
3
2
1

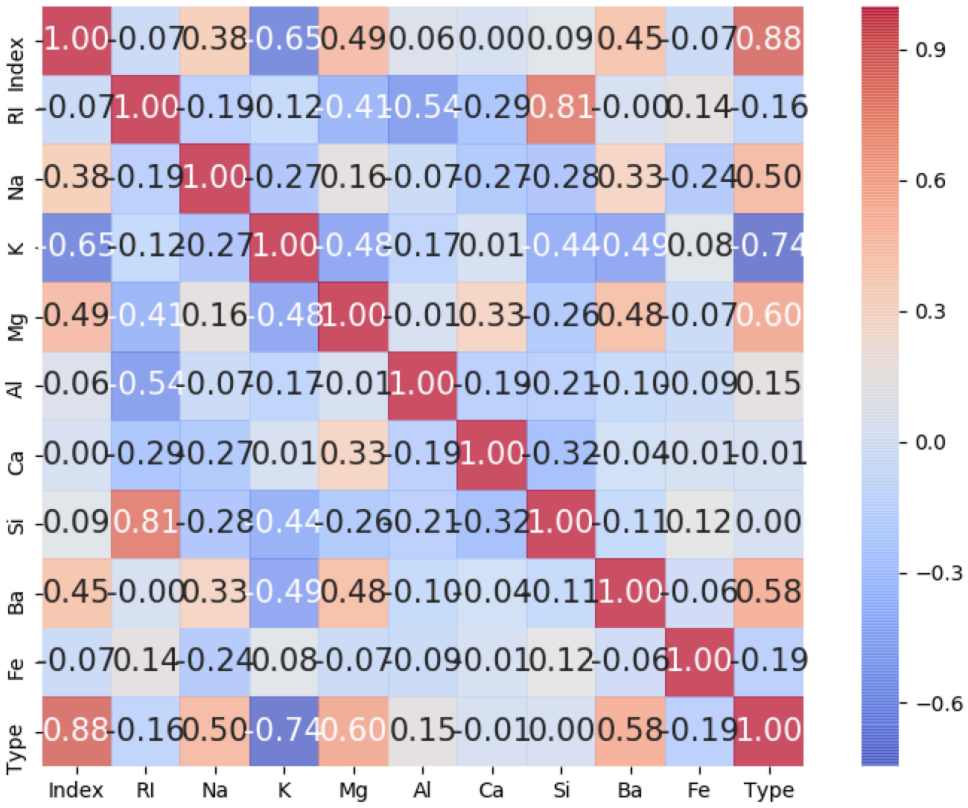


The figures above show the relationship between every single feature with all the targets types.

2.3 Correlation Matrix

To get more insight in how strongly each feature is correlated with the Type of glass, we can calculate and plot the correlation matrix for this dataset
We have obtained the heat map of correlation among the variables using the code below.

```
corr = df[features].corr()
    plt.figure(figsize=(16,16))
    sns.heatmap(corr, cbar = True,  square = True, annot=True, fmt= '.2f',annot_kws={'size': 15},
    xticklabels= features, yticklabels= features, alpha = 0.7,   cmap= 'coolwarm')
    plt.show()
```



The color palette in the side represents the amount of correlation among the variables. The lighter shade represents high correlation. The correlation matrix shows us for example that the oxides Mg and Al are most strongly correlated with the Type of glass. The content of Ca is least strongly correlated with the type of glass. There also seems to be a strong positive correlation between RI and Ca.

3 Preprocess

3.1 PCA

Since Clustering generally depends on some sort of distance measure. Points near each other are in the same cluster and points far apart are in different clusters. But in high dimensional spaces, distance measures do not work very well. PCA can drastically improve the accuracy of clustering methods by reduce the number of dimensions first so that your distance metric will have a better fit. Through abstracting the meaning of features with PCA, we are able to understand what factors influence the variance of samples The principle components are the 9 features for the dataset.

By apply PCA, we can calculate the variance of each principal component and realize that about 99 % of the variance can be explained with the first 7 principal components

PC1 Cumulative variance: 47.621%
PC2 Cumulative variance: 73.940%
PC3 Cumulative variance: 84.720%
PC4 Cumulative variance: 94.922%
PC5 Cumulative variance: 98.229%
PC6 Cumulative variance: 99.834%
PC7 Cumulative variance: 99.977%
PC8 Cumulative variance: 100.000%
PC9 Cumulative variance: 100.000%

To test and check to see if applying PCA improved performance, I applied both a classification & clustering method on the dataset before and after applying PCA. Here are the results:

Classification Accuracy on train & test data before PCA:
Accuracy of Logistic regression classifier on training set: 0.59
Accuracy of Logistic regression classifier on test set: 0.35

Classification Accuracy on train & test data after PCA
Accuracy of Logistic regression classifier on training set: 0.65
Accuracy of Logistic regression classifier on test set: 0.49

Clustering Accuracy before PCA
K-Means: .497678951

Affinity Propagation: .356047481

Spectral Clustering: .556455488

Mean Shift: .557108026

Agglomerative Clustering: .509199142

DBSCAN: .429615145

Birch: .547798118

Clustering Accuracy after PCA

K-Means: .504161228

Affinity Propagation: .357792417

Spectral Clustering: .566129697

Mean Shift: .553318621

Agglomerative Clustering: .506164719

DBSCAN: .438134337

Birch: .547798118

The results show that PCA did indeed improve performance, increasing the accuracy in both classification and clustering algorithms. However, the improvement in clustering accuracy is very minimal.

4 Clustering

Similar to classification, clustering is a learning method which characterize objects into groups by one or more features. However, instead of being used in supervised learning environments, clustering is used in unsupervised learning where similar instances are grouped based on their features or properties.

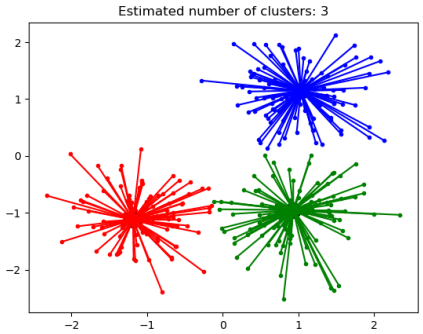
4.1 Kmeans Clustering

The KMeans algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares. This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields.
The algorithm divides a set of N samples X into K disjoint clusters C, each described by the mean *μj* of the samples in the cluster. The means are commonly called the cluster “centroids”; note that they are not, in general, points from X, although they live in the same space. The K-means algorithm aims to choose centroids that minimize the *inertia*, or within-cluster sum of squared criterion:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

4.2 Affinity Propagation

Affinity Propagation takes in input measures of similarity between pairs of data points, and simultaneously considers all data points as potential exemplars. Real-valued messages are exchanged between data points until a high-quality set of exemplars and corresponding clusters gradually emerges.
Exemplars are points that explain the other data points best and are the most significant of their cluster. A cluster only has one exemplar. All the data points want to collectively determine which data points are an exemplar for them. These messages are stored in two matrices. This updating happens iteratively until convergence, at which point the final exemplars are chosen, and hence the final clustering is given.



Affinity Propagation chooses the number of clusters based on the data provided. Due to this, the two important parameters are the *preference*, which controls how many exemplars are used, and the *damping factor* which damps the responsibility and availability messages to avoid numerical oscillations when updating these messages.

```
    AffinityPropagation(damping=0.5, max_iter=200, convergence_iter=15, copy=True,
preference=None, affinity='euclidean', verbose=False),
```

4.3 Agglomerative Clustering

Agglomerative clustering is also known as Hierarchical clustering and does not require the user to specify the number of clusters. Initially, each point is considered as a separate cluster, then it recursively clusters the points together depending upon the distance between them. The points are clustered in such a way that the distance between points within a cluster is minimum and distance between the cluster is maximum. In this assignment, we're measuring the distance using Euclidean algorithm. Unlike k-means clustering, it is bottom-up approach.

```
    AgglomerativeClustering(n_clusters=7, affinity='euclidean', memory=None,
connectivity=None, compute_full_tree='auto', linkage='ward',
pooling_func='deprecated'),
```

4.4 Spectral Clustering

Spectral clustering works by first transforming the data from Cartesian space into similarity space and then clustering in similarity space. The original data is projected into the new coordinate space which encodes information about how nearby data points are. The similarity transformation reduces the dimensionality of space and, loosely speaking, pre-clusters the data into orthogonal dimensions. This pre-clustering is non-linear and allows for arbitrarily connected *non-convex* geometries which is the main advantage of spectral clustering.
To perform a spectral clustering there are 3 main steps:

1. Create a similarity graph between our *N* objects to cluster.
2. Compute the first k eigenvectors of its Laplacian matrix to define a feature vector for each object.
3. Run k-means on these features to separate objects into k classes.

5 Results

5.1 Clustering Performance Evaluation

To evaluate the clustering algorithms, I went with the Fowlkes-Mallows index which is used when the ground truth class assignments of the samples is known. The Fowlkes-Mallows score FMI is defined as the geometric mean of the pairwise precision:

$$\mathrm{FMI} = \frac{\mathrm{TP}}{\sqrt{(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})}}$$

Where TP is the number of True Positive (i.e. the number of pair of points that belong to the same clusters in both the true labels and the predicted labels), FP is the number of False Positive (i.e. the number of pair of points that belong to the same clusters in the true labels and not in the predicted labels) and FN is the number of False Negative (i.e the number of pair of points that belongs in the same clusters in the predicted labels and not in the true labels). The score ranges from 0 to 1 and a high value indicates a good similarity between two clusters.

```
score = metrics.fowlkes_mallows_score(labels, y)
    print(name + ': ' + ('%.9f' % score).lstrip('0') + '\n')
```

The algorithm that had the highest score was spectral clustering. This is probably due to the advantages spectral clustering has since it does not make strong assumptions on the statistics of the clusters and it typically has good clustering results. One remarkable advantage of spectral clustering is its ability to cluster “points” which are not necessarily vectors, and to use for this a “similarity”, which is less restrictive than a distance. A second advantage of spectral clustering is its flexibility as it can find clusters of arbitrary shapes, under realistic separations.

6 Conclusion

In conclusion, all clustering methods performed around 50% accuracy on the dataset. The algorithm with the highest accuracy was spectral clustering. By testing different clustering methods, I was able to understand why certain methods worked better than others since there are different use cases for each method. This assignment proved that the difficulty with unsupervised learning is far higher in comparison to supervised learning tasks.

References

[1] Clustering. https://scikit-learn.org/stable/modules/clustering.html.
[2] Covariance correlation matrix.
https://en.wikipedia.org/wiki/Covariance_matrix#Relation_ to_the_correlation_matrix.
[3] Glass dataset from uci. https://www.kaggle.com/uciml/glass.
[4] sklearn. https://scikit-learn.org/stable/install.html.