Batuhan Erden

batuhan.erden@ozu.edu.tr

CS 452/552 – Data Science with Python

Project 1 Report

03.12.2017

Comparison of 6 different Clustering Algorithms on 3 different Datasets

Abstract

In this project, I have used 6 different Clustering Algorithms (*KMeans*, *Mini-Batch KMeans*, *Mean Shift*, *Affinity Propagation*, *Spectral Clustering*, and *Agglomerative Clustering*) in clustering module of **sklearn** library to compare their performances on 3 different datasets (*Boston*, *Digits3* and *Iris*). These datasets are also from **sklearn** library. I used **TSNE** (T-distributed Stochastic Neighbor Embedding) to convert the similarities between data points to joint probabilities. It is basically used to have a better-normalized dataset.

1 Introduction

I have applied *KMeans*, *Mini-Batch KMeans*, *Mean Shift*, *Affinity Propagation*, *Spectral Clustering*, and *Agglomerative Clustering* algorithms on *Boston*, *Digits3* and *Iris* datasets. Firstly, I have imported the necessary libraries and initialized the datasets. Since *Boston* is not a clustering but a regression dataset, I changed its target so that it only consisted of 0's, 1's and 2's. In doing so, I was able to show the class labels of *Boston* dataset. I have constructed a class, Dataset, to store datasets names and values. It was required to show the names of the datasets on the graphs. Secondly, I have initialized **TSNE** (T-distributed Stochastic Neighbor Embedding) to normalize the dataset for easier parameter selection. I initialized the clustering algorithms with the best parameters I calculated and compared their results on the datasets. I realized the fact that all of the clustering algorithms used the same lines of code except their instantiation part. In order for me to prevent code duplication, I constructed a class, ClusteringAlgorithm, to encapsulate the methods needed to do the comparison in a single class. Thus, all clustering algorithms used were ClusteringAlgorithm objects. Lastly, I plotted and tabulated the final results using **mathplotlib**.

2 Problem and Dataset Information

My aim on this project was to compare 6 different clustering algorithms on 3 different datasets. These are the datasets from **sklearn's** datasets library. I will briefly describe what is the meaning of each dataset.

2.1 Boston Dataset

Boston is a dataset for House Prices. The creators of this dataset are Harrison, D. and Rubinfeld, D.L. and it's currently being maintained in Carnegie Mellon University. It has 506 instances and 13 attributes. There was only one problem with this database. Since this database is not a suitable database for clustering purposes, I had to change its target value. Before any changes made, target was consisting of float between 5.0 and 50.0 and it didn't have any class labels displayed on the graph. So, I change the target values such that it was only consisting of 0's, 1's and 2's. The method used to assign correct 0's, 1's and 2's was dividing the array into 3-sorted portions that are sorted decreasingly and assigning 0's to the 1st portion, 1's to the 2nd portion and 2's to the 3rd portion. It made the dataset have class labels. This operation was necessary for comparison.

2.2 Digits Dataset

Digit is a dataset consisting of handwritten digits. In this assignment, I only used the 0, 1 and 2 (**n=3**). However, you could observe all digits by changing the **n** value in the constructors. The creator of this dataset is E. Alpaydin. The dataset has 5620 instances and 64 attributes. The **n** value is also equal to the number of clusters.

2.3 Iris Dataset

Iris is a dataset consisting of 3 types of iris plants. It was created by R.A. Fisher. It has 150 instances where each instance has 50 of them. Besides, it has 4 numeric, predictive attributes (*sepal length*, *sepal width*, *petal length* and *petal width*). It is one of the greatest databases used in clustering purposes.

3 Algorithms

In order to have an easier parameter selection, the dataset needs to be normalized. What this means is that we need to have similar data points stand together. To do so, I used **TSNE** (T-distributed Stochastic Neighbor Embedding), which is a prize-winning machine-learning algorithm for dimensionality reduction. It is commonly used to visualize high-dimensional datasets.

3.1 KMeans

KMeans, an algorithm that partitions n observations into k clusters. It is an un-supervised machine-learning algorithm. Each observation is child of the cluster that has the nearest mean. The algorithm first assigns each observation to the proper cluster that has the nearest main. Then it updates the new means to be the centroids of the observations. When there are no more assignments left, the algorithm is said to be converged. It takes number of clusters as argument.

I applied KMeans to my datasets and the exception would be that the clusters were complete and had the observations that were close to each other.

As can be seen in the *Figure 1*, all of the clusters have its observations close to each other. It is obvious that KMeans performed well on both digits and iris datasets. However, it couldn't find the clusters well because the normalized mutual info score of Boston is too low.

To sum up, the winner for this session is Digits dataset with 76.55% similarity. Iris dataset also performed well as it looks almost the same with the class labels. However, as Boston not being a suitable dataset for clustering, KMeans failed to find proper clusters for it.

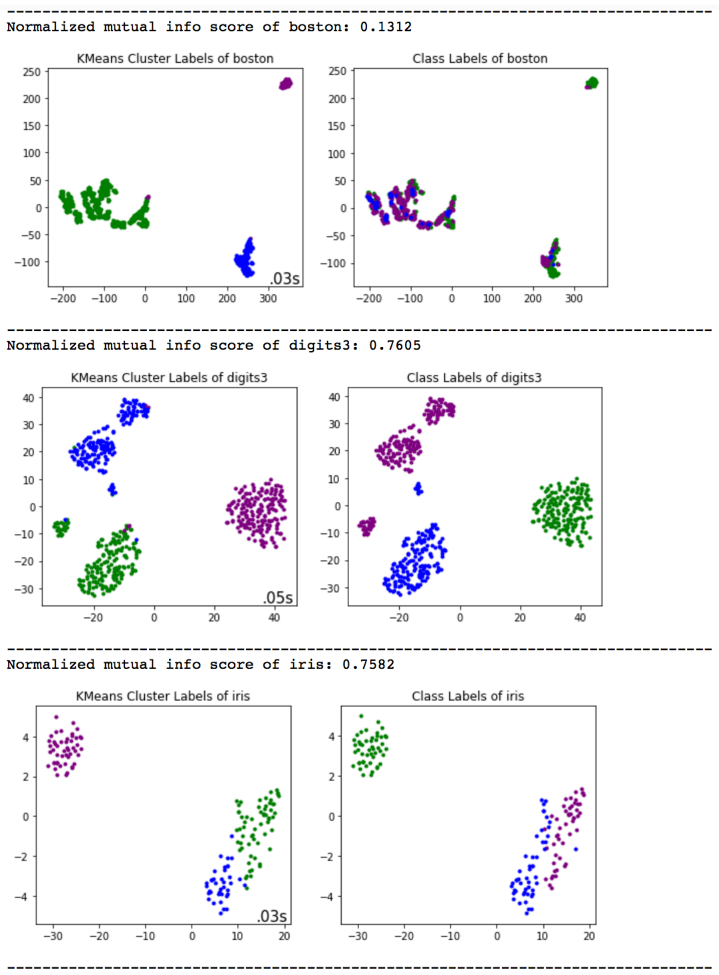

```
--------------------------------------------------------------------------------
Normalized mutual info score of boston: 0.1312
--------------------------------------------------------------------------------
Normalized mutual info score of digits3: 0.7605
--------------------------------------------------------------------------------
Normalized mutual info score of iris: 0.7582
--------------------------------------------------------------------------------
```

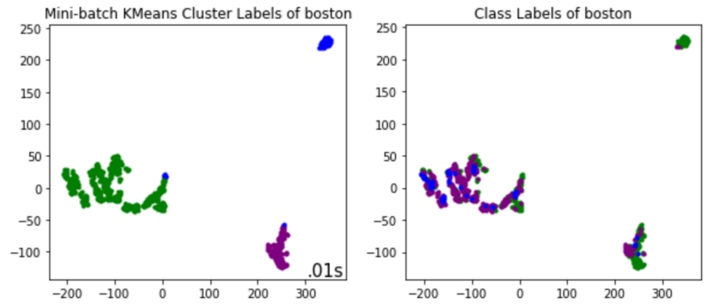
Figure 1: Results of KMeans on each dataset

3.2 Mini-Batch KMeans

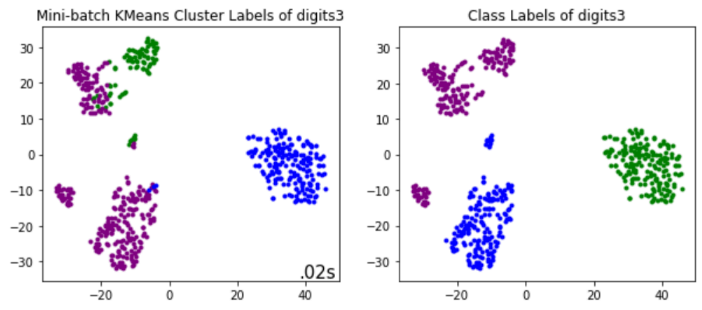
The implementation and the results of the Mini-Batch KMeans are almost the same with KMeans but Mini-Batch KMeans is slightly faster. The advantage of this algorithm is that it has less computational cost because it does not use all dataset in each iteration; it uses a subsample of a fixed size. It takes number of clusters as argument.

As in the *Figure 2*, the difference between Mini-Batch KMeans and KMeans is not that much. However, we can see from the time labels in the graphs that Mini-Batch KMeans was faster than the KMeans. On the other hand, Iris dataset is the best in this algorithm and Digits dataset is the 2nd while Digits dataset is the best and Iris dataset is the 2nd in KMeans. The algorithms differ more on some specific datasets.

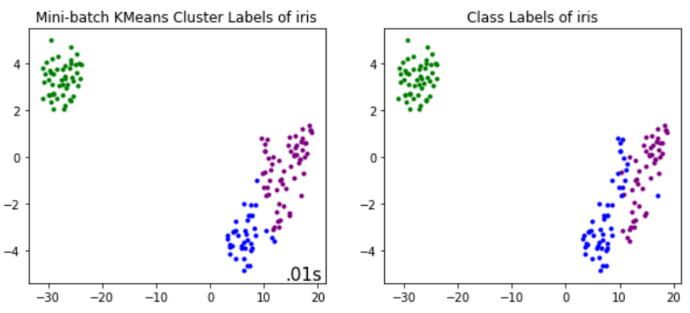
```
--------------------------------------------------------------------------------
Normalized mutual info score of boston: 0.1312
```



```
--------------------------------------------------------------------------------
Normalized mutual info score of digits3: 0.6538
```



```
--------------------------------------------------------------------------------
Normalized mutual info score of iris: 0.7419
```



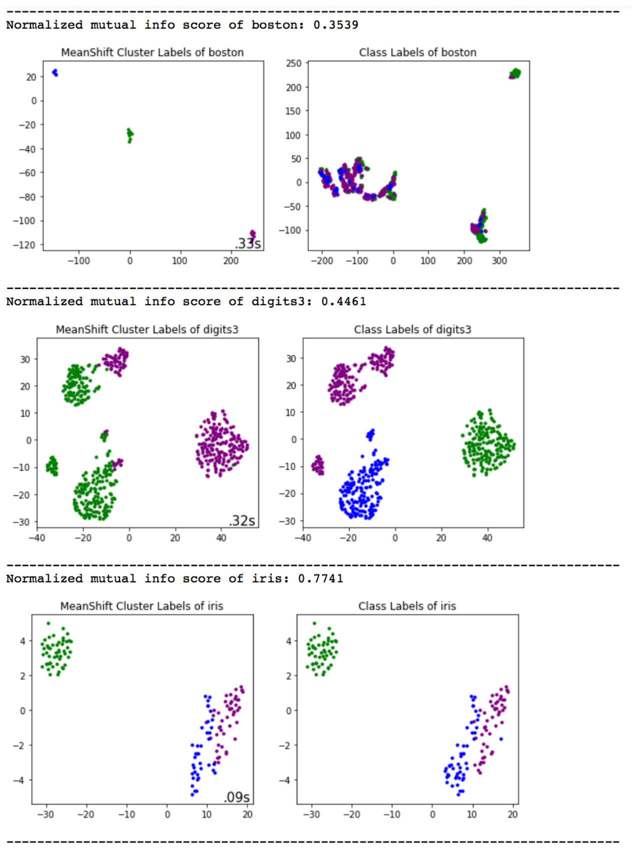
```
--------------------------------------------------------------------------------
```

Figure 2: Results of Mini-Batch KMeans on each dataset

3.3 Mean Shift

Mean Shift algorithm is a clustering technique that does not need to have the number of clusters before its assignment. It is a centroid-based algorithm. This algorithm shifts the kernel to a point with higher density until it converges. Convergence in this algorithm means that there is no direction left for the algorithm to shift the kernel. The algorithm looks for the highest density and shift to that direction. The algorithm relies on these arguments:

- **bandwidth**: Bandwidth used in **RBF Kernel**. It is estimated using **estimate_bandwidth** method of **sklearn.cluster** and it takes 2 arguments; **X**, which is the normalized output of **TSNE** and **quantile**, which is a value between [0, 1]. Quantile being 0.5 means that median of all pairwise distances is used. For Boston, Digits and Iris datasets, 0.01, 0.18 and 0.03 values are used respectively after a significant number of trials.
- **bin_seeding**: Bin Seeding being True means that initial kernel locations are not locations of all points. I used **True** for Bin Seeding to speed up the algorithm because fewer seeds are initialized.



```
--------------------------------------------------------------------------------
Normalized mutual info score of boston: 0.3539
--------------------------------------------------------------------------------
Normalized mutual info score of digits3: 0.4461
--------------------------------------------------------------------------------
Normalized mutual info score of iris: 0.7741
--------------------------------------------------------------------------------
```

Figure 3: Results of Mean Shift on each dataset

The results are shown in the *Figure 3* where the results are almost as expected. I expected observations to move to locations with higher density and construct its clusters in that location. As can be seen in the *Figure 3*, clusters for Boston are gathered in 3 different small points because there are the points with the higher density.

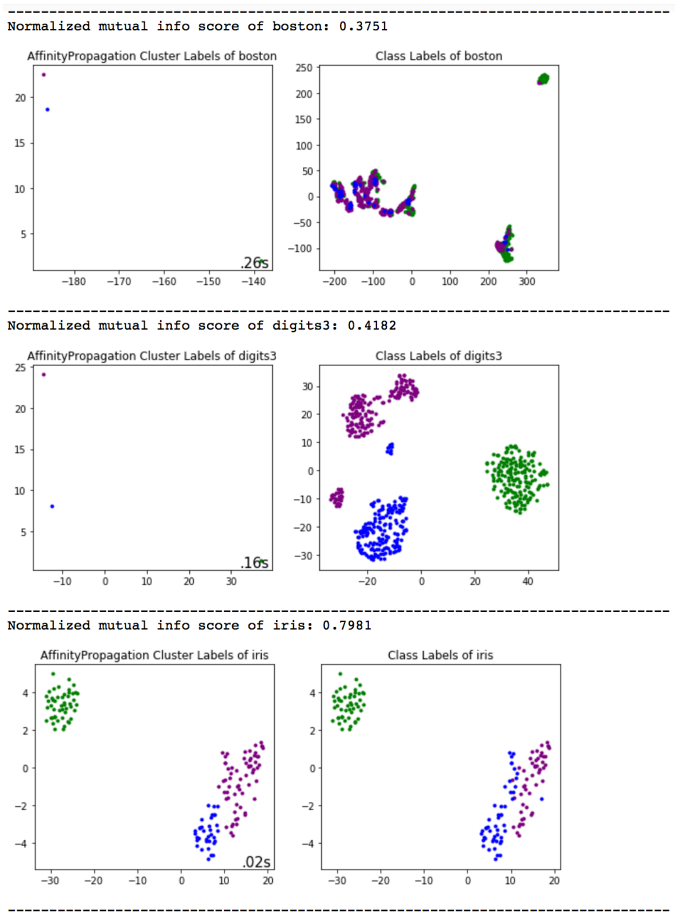
On the other datasets, the results look not much different than the other algorithms. For Digits dataset, it is obvious to state that there were 2 higher density points so clusters are gathered in these places and for iris dataset there were 3 main higher points. I expected this because iris dataset is great for clustering purposes and it’s easier to use it in all algorithms.

3.4 Affinity Propagation

Affinity Propagation, like Mean Shift, does not require the prior knowledge of the number of clusters. It basically finds what is called **exemplars** that are responsible to represent the clusters. This is a new algorithm and works well on clustering purposes. The algorithm relies on 2 arguments and these are:

- **damping:** Damping factor is a number between [0.5, 1.0]. It is the extent to which the current value is maintained relative to incoming values. After doing enough experiments, I chose damping to be 0.92, 0.98, and 0.06 for datasets Boston, Digits and iris respectively.
- **preference:** It is the preferences for each point. Points with large preference values are more likely to be selected as an exemplar.

In the *Figure 4*, the results of this algorithm are shown. The result of Boston was obvious but I did not expect the result of Digits like this. Since Digits dataset has 3 crowded clusters, I was expecting more accurate result. Lastly, amongst all algorithms, iris performed the best in terms of normalized mutual info score.



```
--------------------------------------------------------------------------------
Normalized mutual info score of boston: 0.3751
--------------------------------------------------------------------------------
Normalized mutual info score of digits3: 0.4182
--------------------------------------------------------------------------------
Normalized mutual info score of iris: 0.7981
--------------------------------------------------------------------------------
```

Figure 4: Results of Affinity Propagation on each dataset

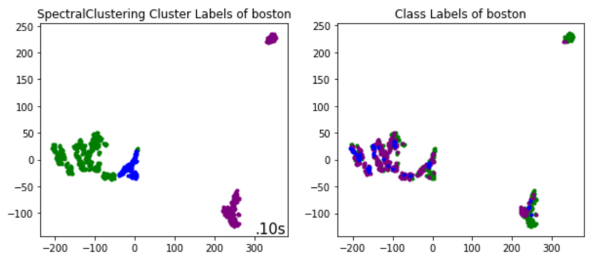
3.5 Spectral Clustering

Spectral Clustering makes use of the Normalized Laplacian. It benefits from the eigenvalues of the similarity matrix of the data to performance the reduction operations on the dimensionality before clustering operation. It takes number of clusters as argument and it relies on these 2 other arguments:

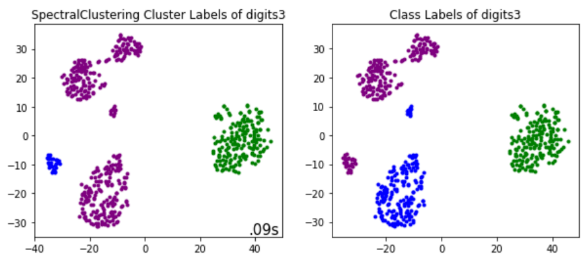
- **affinity:** It is the kernel. Only kernel with similarity scores should be used. I used **nearest_neighbors**.
- **eigen_solver:** It is the eigenvalue decomposition strategy. I chose **arpack** as it is recommended in most of the tutorials.

As can be seen in the *Figure 5*, Digits and iris datasets have great scores. Besides, this algorithm worked great on Iris dataset. I think that it is because of the rich similarity in the Iris dataset. Since Boston has poor similarity, I wouldn't expect anything better than this result. Lastly, on the Digits dataset, the algorithm seems to have 2 clusters combined because it resulted a very huge cluster (the purple one).

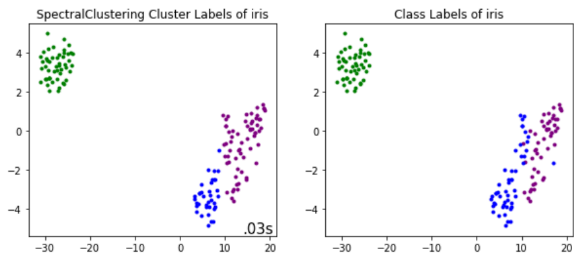
```
Normalized mutual info score of boston: 0.1363
```



```
--------------------------------------------------------------------------------
Normalized mutual info score of digits3: 0.7101
```



```
--------------------------------------------------------------------------------
Normalized mutual info score of iris: 0.8058
```



```
--------------------------------------------------------------------------------
```

Figure 5: Results of Spectral Clustering on each dataset

3.6 Agglomerative Clustering

Agglomerative Clustering, knows as Agglomerative Hierarchal Clustering, is a bottom-up clustering algorithm. In this algorithm, clusters have its sub-clusters, which also have their sub-clusters and so on. The goal for this algorithm to merge the closest pair of clusters until all data is put into a single cluster. It takes number of clusters as argument and relies on these 3 other arguments:

- **affinity:** It is the metric used to compute the linkage. **cityblock** is used in this project.
- **linkage:** It determines the distance to use between sets of observations. The algorithm merges pairs which minimizes this criterion. **average** is used for linkage because it uses the average of the distances.
- **connectivity:** It is the connectivity matrix. By making use of the **kneighbors_graph**, I made connectivity symmetric.

It can be seen in the *Figure 7 and Figure 8* that the number of clusters is almost 2. It means that the algorithm did its job and merged 2 clusters to decrease the number of cluster from 3 to 2. However, in the *Figure 6*, the algorithm was not successful enough. This is one of the algorithms that I liked the most because its bottom-up approach is so efficient. For instance, it calculates the result in *Figure 8* in 0.01 seconds and the result is satisfying.

```
Normalized mutual info score of boston: 0.1424
```

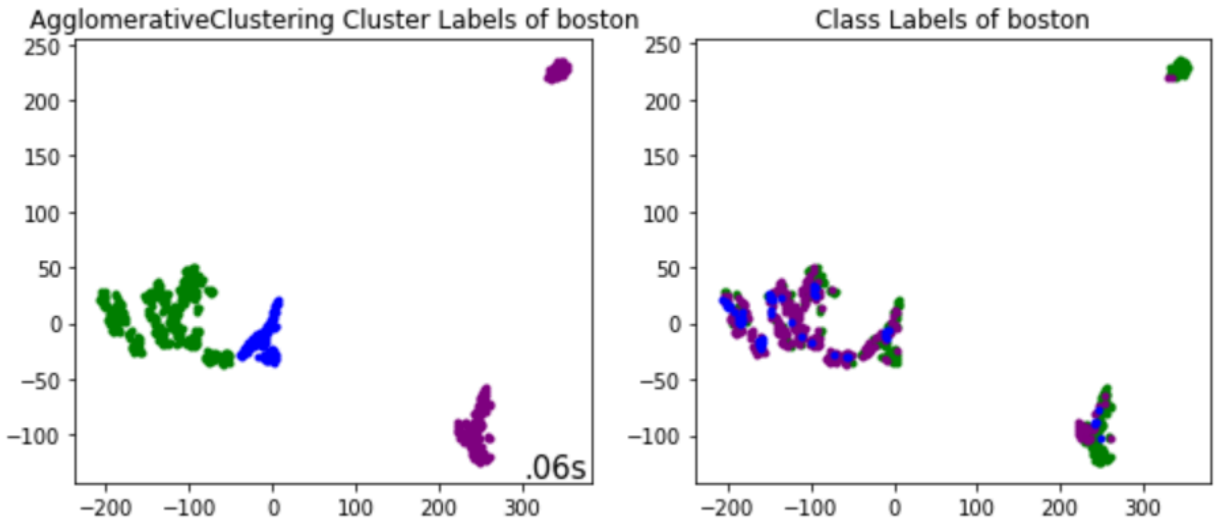
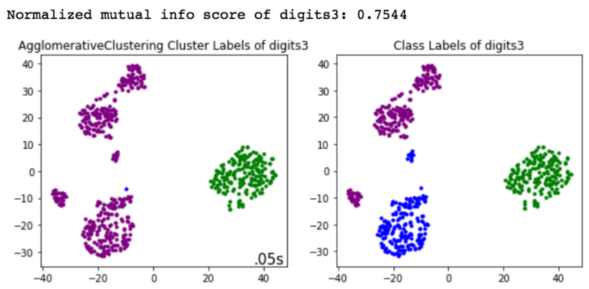
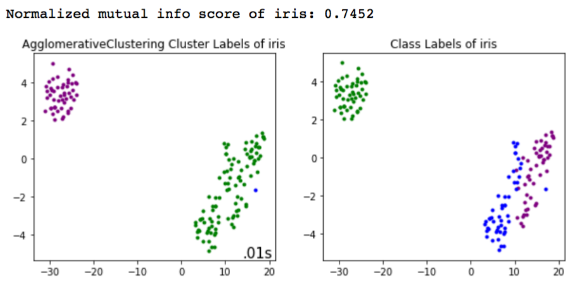


Figure 6: Results of Spectral Clustering on Boston dataset



```
Normalized mutual info score of digits3: 0.7544
```

Figure 7: Results of Spectral Clustering on Digits dataset



```
Normalized mutual info score of iris: 0.7452
```

Figure 8: Results of Spectral Clustering on Iris dataset

4 Final Results

Using **mathplotlib**, I plotted the overall results. There are 3 graphs. The graph in *Figure 9* shows the normalized mutual info score of each dataset on each clustering algorithm. The graph in *Figure 10* shows the time passed in fitting each dataset into each clustering algorithm. The graph in *Figure 11* shows the ratio of <u>the data in the 1st graph divided by the data in the 2nd graph (score / time_passed)</u>. Lastly, the table in *Figure 12* shows the performance of all clustering algorithms on each dataset. This is the last figure in this report and it concludes everything.

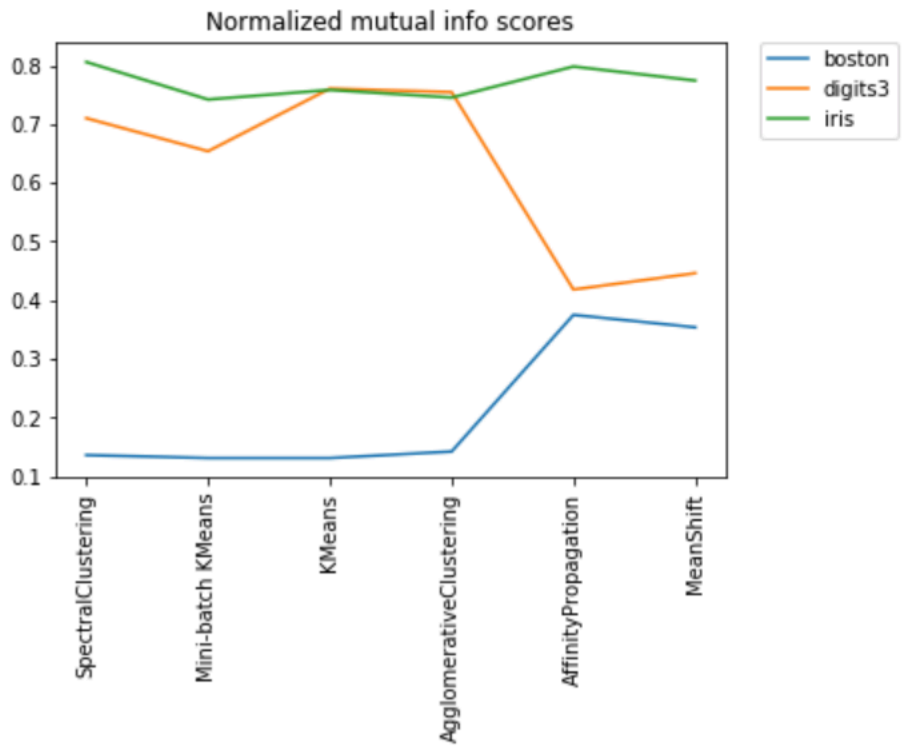


Figure 9: Normalized mutual info scores of each dataset on each Clustering Algorithm

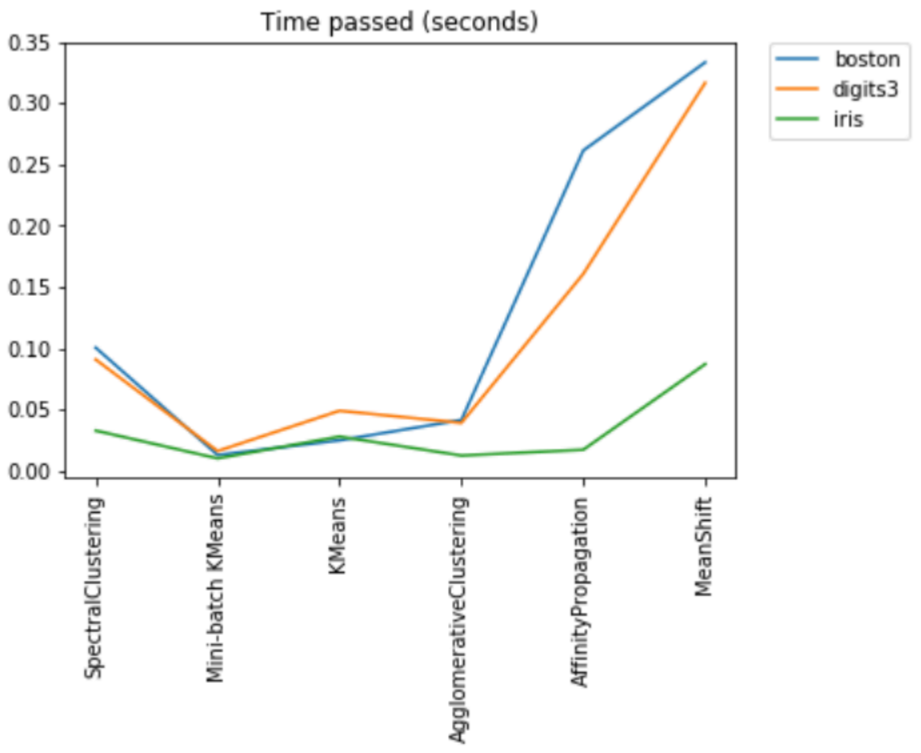


Figure 10: Time passed (seconds) in fitting each dataset into each Clustering Algorithm

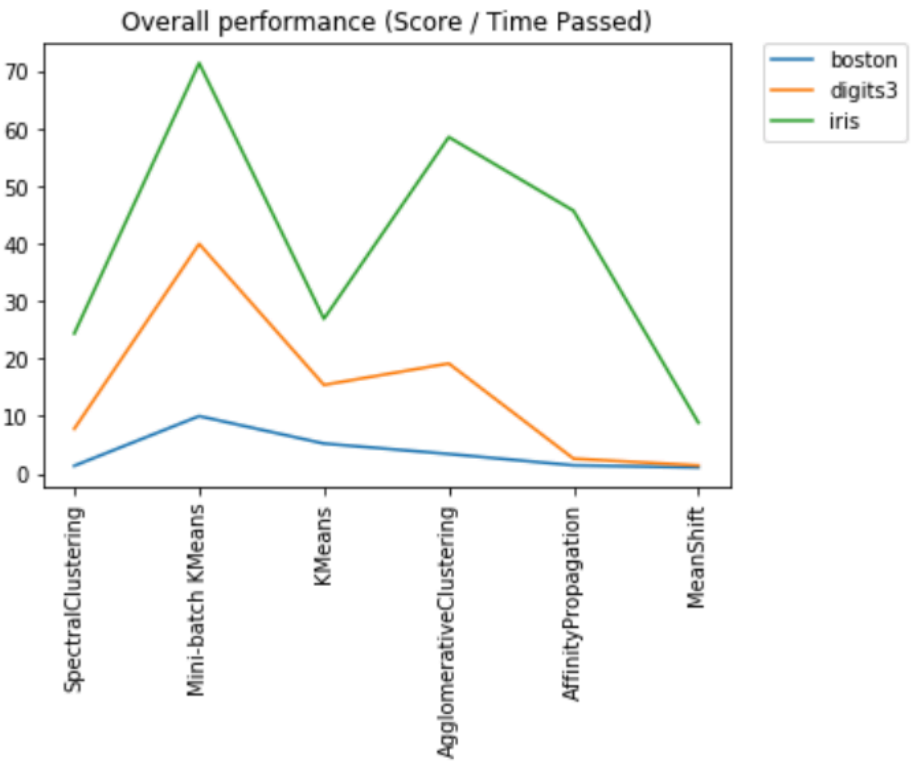
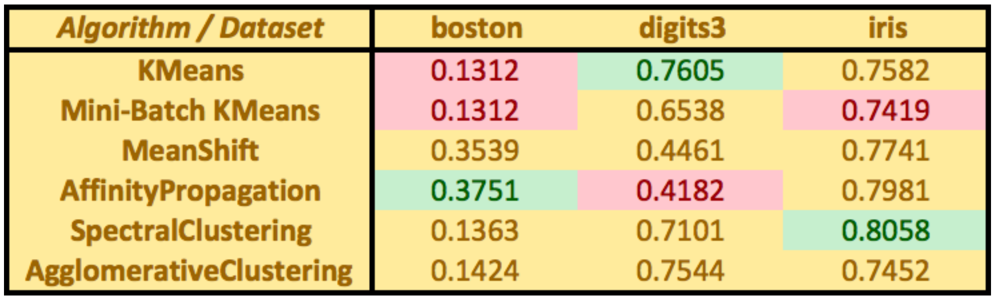


Figure 11: Normalized mutual info score divided by time passed



Algorithm / Dataset	boston	digits3	iris
KMeans	0.1312	0.7605	0.7582
Mini-Batch KMeans	0.1312	0.6538	0.7419
MeanShift	0.3539	0.4461	0.7741
AffinityPropagation	0.3751	0.4182	0.7981
SpectralClustering	0.1363	0.7101	0.8058
AgglomerativeClustering	0.1424	0.7544	0.7452

Figure 12: Table of performances

As can be seen in the *Figure 12*, For Boston dataset, Affinity Propagation worked the best (score=0.3751) and both KMeans and Mini-Batch KMeans worked the worst (scores=0.1312). For Digits dataset, KMeans worked the best (score=0.7605) and Affinity Propagation worked the worst (score=0.4182). And lastly for Iris dataset, Spectral Clustering worked the best (score=0.8058) and Mini-Batch KMeans worked the worst (score=0.7419).

5 Conclusion

As mentioned earlier, my aim for this project was to compare 6 different clustering approach techniques on 3 different dataset. I did my comparison and elaborated critical points. Some algorithms worked well on some datasets and some algorithms worked inefficient on some dataset. I learned that it mostly depends on the datasets and the proper choice of parameters. I learned a lot from this assignment about Data Science and Clustering.

References

- http://scikit-learn.org
- http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/TUZEL1/MeanShift.pdf
- http://www.improvedoutcomes.com