Draft version April 3, 2020
Typeset using LaTeX default style in AASTeX63

Analysis of Performance of Clustering Algorithms
PH6130 Data Science Analysis - Project

Karthik Pagadala[1] and Jeel A. Bhavsar[2]

[1] *EE17BTECH11049*
[2] *ES16BTECH11005*

ABSTRACT

In today's age, data is raw gold and the information extracted from it can be compared to gold ornaments. Everyday, large amounts of data are encountered by organizations and people. An indispensible method to handle this data is to categorize or classify them into sets of groups, partitions or clusters. In data science, Clustering is the most commonly used unsupervised learning technique. It is a way of locating similar data objects into clusters based on similarity. In a cluster, objects have an affinity with other members in the same cluster and a disparity with members of any other cluster. Clustering algorithms can be broadly categorized into Hierarchical Clustering algorithm, Density-Based clustering algorithm, Partitioning clustering algorithm, Graph-based algorithm, Grid-based algorithm, Model-based clustering algorithm and Combinational clustering algorithm. These clustering algorithms give different result based on the data. Some clustering techniques work better for large data sets and some give good results for finding clusters with arbitrary shapes. In this project, we compare the performance of K-Means and Hierarchical clustering algorithms on the HTRU2 dataset.

Keywords: clustering, k-means, hierarchical

1. INTRODUCTION

The advancements in data collection methods have led to the availability of large amounts of data. There is an increased demand for extracting useful information from data. Data clustering, also refers to as unsupervised classification, is defined as a technique which groups data objects such that objects in one cluster are very identical and objects in different cluster(s) are relatively different. Unsupervised classification is exploratory data analysis where there is no training data set and the objective is to extract hidden patterns in data sets with no labelled responses. The prime focus is to increase proximity in data points belonging to the same cluster and increase dissimilarity among various clusters and all this is achieved through achieved through some similarity measure. On one hand, many tools for cluster analysis have been created, along with the information increase and subject intersection. On the other hand, each clustering algorithm has its own strengths and weaknesses, due to the complexity of information. A considerable number of clustering algorithms have been reviewed in (1). Clustering algorithms can be based on the following: Partition, Hierarchy, Distribution, Density, Hierarchy, Fuzzy Theory, Graph Theory, Grid, Fractal Theory, Model

In this project, we test the performance of the K-Means and Hierarchical clustering algorithms on the HTRU2 dataset from the UC Irvine Machine Learning Repository (2). We have also used Principal Component Analysis and tested the above clustering algorithms.

2. DATASET DESCRIPTION

HTRU2 is a data set which describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey (South). Pulsars are of considerable scientific interest as probes of space-time, the inter-stellar medium, and states of matter. Here the legitimate pulsar examples are a minority positive class, and spurious examples the majority negative class. This is a binary classification problem. The candidates must be classified into pulsar and non-pulsar classes to aid discovery. The data set contains 16,259 spurious examples caused by RFI/noise, and 1,639 real pulsar examples. We are trying unsupervised clustering on a dataset meant for supervised binary classification. We are trying to see if the predicted labels are the same as the ground truth and whether clustering algorithms can be

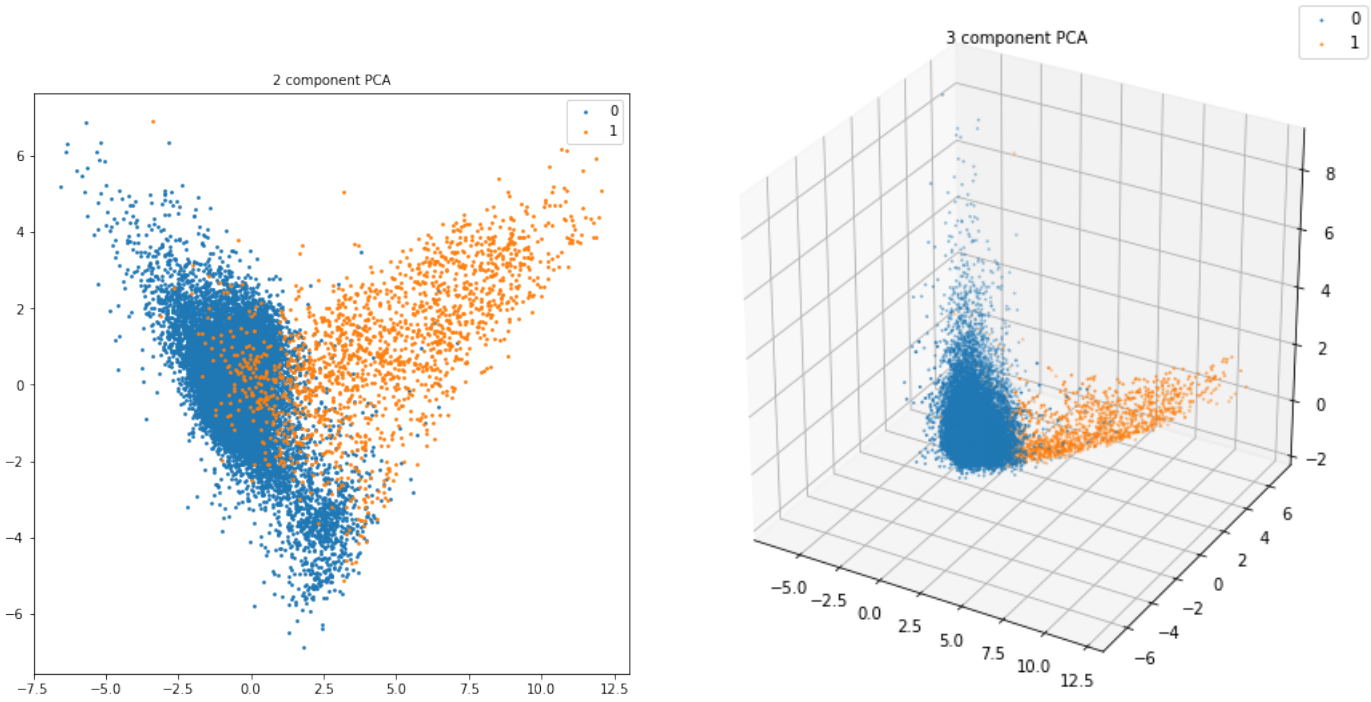

Figure 1. Visualisations of the dataset after applying 2 and 3 component PCA

used to annotate data. We have used python to code and the sklearn library to implement these algorithms. We also experimented with the parameters of the Agglomerative clustering function and have obtained a variety of results.

3. K-MEANS CLUSTERING

K-Means is an example of a clustering algorithm based on Partition. Partition based clustering algorithms are subclassified into Medoid based and Centroid based. K-Means is a centroid based Partitioning algorithm. The basic idea of this kind of clustering is to regard the center of data points as the center of the corresponding cluster. K-Means is the most widely-used centroid-based non supervised clustering algorithm. Centroid-based algorithms are efficient but sensitive to initial conditions and outliers.

The effectiveness of k-means technique depends on the objective function that is being used to calculate the distance between the instances.

K-Means Algorithm:

1. Select k cluster centres randomly.
2. Compute the distance between cluster centres and each data point.
3. Keep the data point in the cluster which is closest to the data point.
4. Update the cluster centres.
5. Compute the distance between updated cluster centres and all data points.
6. Check if any data point has changed cluster, and if there are changes, steps 3 onwards are repeated.

Complexity of K-Means:

Time Complexity: O(nki)
Space Complexity: O(k+n),
where n is number of points, k is the number of clusters and i is the number of iterations. Generally, the value of k and i is fixed prior to running the algorithm to attain the linear time complexity by the algorithm.

Shortcomings of K-Means clustering technique:

1. Does not clearly defines a universal method of deciding total number of partitions in the beginning , this algorithm relies heavily on user to provide in advance ,the number of k clusters.
2. Sensitive to outliers.
3. Can deal only with clusters with spherical symmetrical point distribution.
4. Not applicable to categorical data.

Mini Batch K-Means:
This algorithm uses mini-batches to reduce the computation time. Mini-batches are subsets of the input data, randomly sampled in each training iteration. The objective function which is to be optimised remains the same. These mini-batches drastically reduce the amount of computation required to converge to a local solution. In contrast to other algorithms that reduce the convergence time of k-means, mini-batch k-means produces results that are generally only slightly worse than the standard algorithm. Since the HTRU dataset is quite small, the computation time difference between Kmeans and Mini Batch Kmeans was not very prominent. Hence, the results have not been included here but can be viewed in the github repository at (8).

4. HIERARCHICAL CLUSTERING

This is a paradigm of cluster analysis which generates a sequence of nested clusters which can be visualized as a tree or a hierarchy of clusters known as cluster dendrogram. Hierarchical trees can provide a view of data at different levels of abstraction. When laid down as a tree, this hierarchy has the highest level as root and the lowest levels as leaves. Each point that resides in the leaf node has its own cluster whereas the root contains of all points in one cluster. The dendrogram can be cut at intermediate levels for obtaining clustering results.
Hierarchical clustering can be implemented through two methods: Agglomerative and Divisive Hierarchical clustering.

Agglomerative clustering works in a "bottom-up" manner. Each object is initially considered as a single-element cluster (leaf). At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (nodes). This procedure is iterated until all points are member of just one single big cluster (root).

Divisive clustering is a top-down approach. This algorithm also does not require the knowledge of the number of clusters. Top-down clustering needs a method for splitting a cluster that contains the entire data and proceeds by splitting clusters recursively until individual data have been split into a singleton cluster.

- In the divisive approach, a cluster is recursively split after starting with all the points in the same cluster, this step is repeated until all points are in separate clusters. For a cluster with M objects, there are 2^{M-1} possible two-subset divisions, which incurs a high computation cost.
- Divisive clustering is more complex, efficient and accurate.
- But both these approaches are biased against non-spherical clusters due to the centroid-based approach employed by both methods.

Complexity of Hierarchical clustering:
Time Complexity: O($n^2 log(n)$)
Space Complexity: O(n^2),
where n is the number of data objects.

In this project, we have implemented the Agglomerative Hierarchical Clustering algorithm using the sklearn library. The stepwise procedure of the algorithm is as follows:

i Compute the distance matrix containing the distance between each pair of patterns. Assume each pattern as a cluster.

ii Find the most similar pair of clusters using the data matrix. Combine these similar pairs of patterns into a single cluster.

iii If all points are in a single cluster the stop it, otherwise continue with step ii.

Similarity Measurement:
An important aspect which comes into picture now is to decide when clusters are similar. In order to decide which objects/clusters should be combined or divided, we need methods for measuring the similarity between objects. There are many methods to calculate the (dis)similarity information. This is mainly done through distance measurement and linkage. The AgglomerativeClustering function in sklearn provides the following similarity measures, also known as affinities: Euclidean, L1, L2, Manhattan, Cosine, Precomputed.

These measures are basically distance measurement in various methods. Now, the linkage function takes the distance information and groups pairs of objects into clusters based on their similarity. The linkage criterion determines which distance to use between sets of observation. The algorithm will merge the pairs of cluster that minimize this criterion. sklearn provides the following linkage criterion:

1. ward: minimizes the variance of the clusters being merged.
2. average: uses the average of the distances of each observation of the two sets.
3. complete (or maximum linkage): uses the maximum distances between all observations of the two sets.
4. single: uses the minimum of the distances between all observations of the two sets.

5. SIMULATIONS AND RESULTS

The results here show Accuracy of prediction of 0s (No. of True Negative %), 1s (No. of True Positive %), no. of False Positives and False Negatives. The data set contains 16,259 spurious examples caused by RFI/noise i.e. 0s, and 1,639 real pulsar examples i.e 1s.

True Positives: No of 1s classified correctly as 1s.
True Negatives: No of 0s classified correctly as 0s.
False Positives: No of 0s classified incorrectly as 1s.
False Negatives: No of 1s classified incorrectly as 0s.

We have also included the Homogeneity and Completeness scores in the tables computed using sklearn library (9). Those metrics are based on normalized conditional entropy measures of the clustering labeling to evaluate given the knowledge of a Ground Truth class labels of the same samples. A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class. A clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster. Both scores have positive values between 0.0 and 1.0, larger values being desirable.

5.1. *K-Means*

PCA components used	Acc. of 0's prediction(%)	Acc. of 1's prediction(%)	Overall Acc.(%)	Homogeneity Measure	Completeness Measure	# of False Positives	# of False Negatives
2	**99.54**	69.74	**96.81**	**0.54**	**0.67**	**75**	496
3	95.36	78.04	93.78	0.44	0.38	754	360
4	95.25	78.04	93.67	0.44	0.38	773	360
5	95.25	78.1	93.67	0.44	0.38	773	359
6	95.21	**78.16**	93.65	0.44	0.38	778	**358**
7	95.23	**78.16**	93.66	0.44	0.38	776	**358**
Without PCA	95.23	**78.16**	93.66	0.44	0.38	776	**358**

Table 1: Results for experiments with K-Means Algorithm

Observations for KMeans:

i) As we can see from the 2 Component PCA visualisation in figure 1, the 0s are clustered together in a more dense fashion compared to 1s. The Kmeans algorithm is able to predict the 0s with 99.55% accuracy with only 75 False Positives after applying 2 component PCA on the dataset.

ii) There is not a lot of difference in the accuracy of 0s and 1s prediction in the remaining cases. But The number of True Positives has increased i.e. Accuracy of 1s prediction has improved with increasing PCA components.

5.2. *Hierarchical Clustering*

We have implemented the sklearn funtion AgglomerativeClustering and have obtained several results by using different linkage and affinity modes.

The dendrogram for the dataset is shown in Figure 1. Since we have only 2 clusters (pulsars and non-pulsars), we splice the dendrogram at the blue line above the red tree.

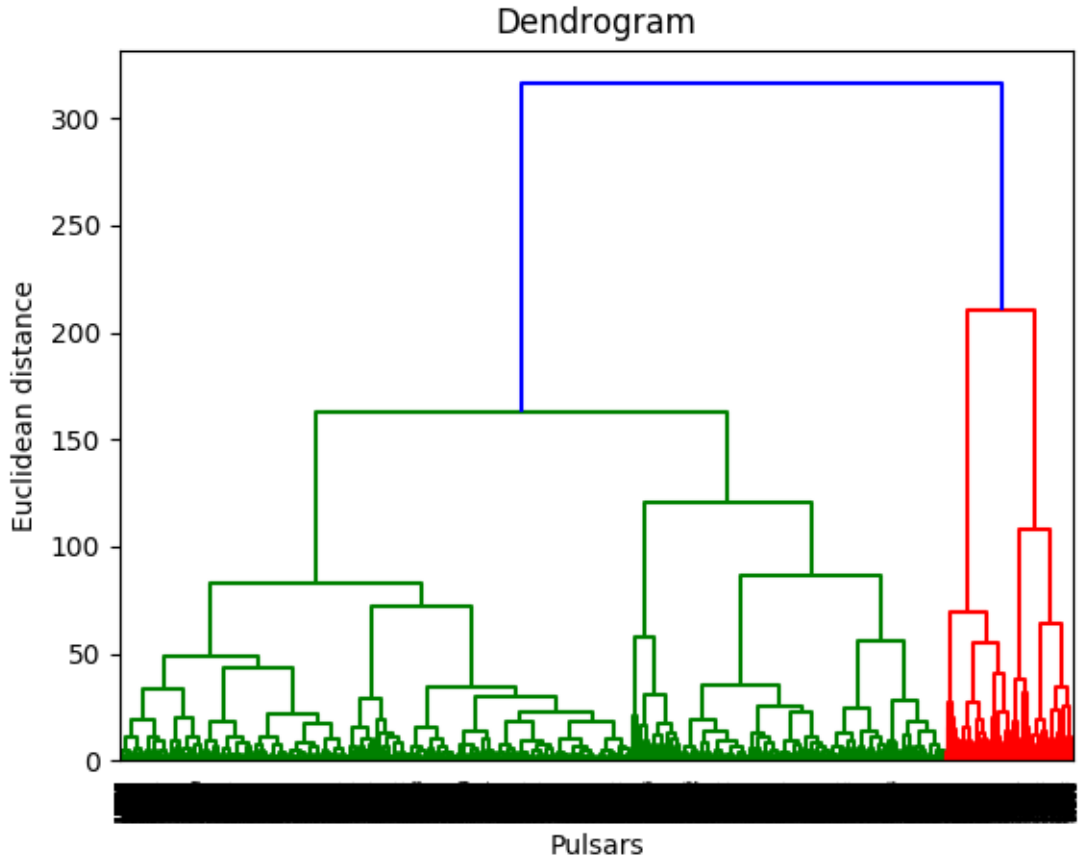

Figure 2. Dendrogram of the dataset

PCA components used	Acc. of 0's prediction(%)	Acc. of 1's prediction(%)	Overall Acc.(%)	Homogeneity Measure	Completeness Measure	# of False Positives	# of False Negatives
2	99.88	61.74	**96.39**	**0.49**	**0.69**	19	627
3	**99.92**	54.06	95.72	0.42	0.65	**13**	753
4	99.91	46.43	95.01	0.35	0.61	15	878
5	99.9	49.85	95.32	0.38	0.63	16	822
6	95.09	79.5	93.66	0.45	0.38	798	336
7	95.15	74.74	93.28	0.4	0.35	788	414
Without PCA	93.92	**84.14**	93.03	0.47	0.36	988	**260**

Table 2: Agglomerative Clustering using Ward + Euclidean

Observations for 'Ward' linkage:

i) When the linkage is ward, sklearn restricts the affinity to euclidean.

ii) From the table, we can observe that without PCA, we have the least number of false negatives and when we have the number of PCA components as 6, we have the second least number of false negatives, but the number of false positives has also decreased compared to without PCA.

PCA components used	Acc. of 0's prediction(%)	Acc. of 1's prediction(%)	Overall Acc.(%)	Homogeneity Measure	Completeness Measure	# of False Positives	# of False Negatives
2	**100.0**	**0.06**	90.85	0.0	0.22	**0**	**1638**
3	**100.0**	**0.06**	90.85	0.0	0.22	**0**	**1638**
4	99.99	0.0	90.84	0.0	0.01	1	1639
5	99.99	0.0	90.84	0.0	0.01	1	1639
6	99.99	0.0	90.84	0.0	0.01	1	1639
7	99.99	0.0	90.84	0.0	0.01	1	1639
Without PCA	99.99	0.0	90.84	0.0	0.01	1	1639

Table 3: Agglomerative Clustering using Single + Euclidean/l2

PCA components used	Acc. of 0's prediction(%)	Acc. of 1's prediction(%)	Overall Acc.(%)	Homogeneity Measure	Completeness Measure	# of False Positives	# of False Negatives
2	**100.0**	**0.06**	90.85	0.0	0.22	**0**	**1638**
3	**100.0**	**0.06**	90.85	0.0	0.22	**0**	**1638**
4	99.99	0.0	90.84	0.0	0.01	1	1639
5	99.99	0.0	90.84	0.0	0.01	1	1639
6	99.99	0.0	90.84	0.0	0.01	1	1639
7	99.99	0.0	90.84	0.0	0.01	1	1639
Without PCA	**100.0**	**0.06**	90.85	0.0	0.22	**0**	**1638**

Table 4: Agglomerative Clustering using Single + Manhattan/l1

PCA components used	Acc. of 0's prediction(%)	Acc. of 1's prediction(%)	Overall Acc.(%)	Homogeneity Measure	Completeness Measure	# of False Positives	# of False Negatives
2	99.38	**2.68**	90.52	0.0	0.03	101	**1595**
3	99.99	0.0	90.84	0.0	0.01	1	1639
4	**100.0**	0.06	90.85	0.0	0.22	**0**	1638
5	99.99	0.0	90.84	0.0	0.01	1	1639
6	99.99	0.0	90.84	0.0	0.01	1	1639
7	99.99	0.0	90.84	0.0	0.01	1	1639
Without PCA	99.99	0.0	90.84	0.0	0.01	1	1639

Table 5: Agglomerative Clustering using Single + cosine

Observations for 'Single' linkage:

i) It can be observed from tables 3,4 and 5 that when linkage is single, irrespective of the affinity used, hierarchical clustering always gave almost 0 accuracy of a data object being a pulsar.

ii) Also, the false negatives in this scenario is very close to the number of pulsars. It implies that pulsars are being wrongly classified as non-pulsar objects.

iii) This can also be seen from the homogeneity and completeness scores, which are almost zero. Therefore, the single linkage is not useful for clustering this dataset.

PCA components used	Acc. of 0's prediction(%)	Acc. of 1's prediction(%)	Overall Acc.(%)	Homogeneity Measure	Completeness Measure	# of False Positives	# of False Negatives
2	99.94	**48.08**	**95.19**	**0.37**	**0.63**	10	**851**
3	99.93	43.01	94.71	0.33	0.6	12	934
4	**99.98**	35.27	94.06	0.27	0.57	**3**	1061
5	99.85	0.06	90.71	0.0	0.0	24	1638
6	**99.98**	0.06	90.83	0.0	0.01	4	1638
7	**99.98**	0.06	90.83	0.0	0.01	**3**	1638
Without PCA	99.93	41.0	94.54	0.31	0.59	11	967

Table 6: Agglomerative Clustering using Average + Euclidean/l2

PCA components used	Acc. of 0's prediction(%)	Acc. of 1's prediction(%)	Overall Acc.(%)	Homogeneity Measure	Completeness Measure	# of False Positives	# of False Negatives
2	99.88	64.25	96.61	0.52	0.7	20	586
3	99.89	0.06	90.75	0.0	0.0	18	1638
4	99.99	27.46	93.35	0.21	0.54	1	1189
5	99.96	0.06	90.81	0.0	0.0	7	1638
6	99.98	0.0	90.82	0.0	0.01	4	1639
7	**100.0**	19.77	92.65	0.15	0.5	**0**	1315
Without PCA	99.77	**72.79**	**97.3**	**0.59**	**0.73**	38	**446**

Table 7: Agglomerative Clustering using Average + Manhattan/l1

PCA components used	Acc. of 0's prediction(%)	Acc. of 1's prediction(%)	Overall Acc.(%)	Homogeneity Measure	Completeness Measure	# of False Positives	# of False Negatives
2	89.99	72.42	88.38	0.28	0.2	1627	452
3	**90.15**	88.77	**90.02**	**0.43**	**0.29**	**1602**	184
4	62.89	95.67	65.9	0.22	0.1	6033	71
5	70.4	94.87	72.64	0.26	0.12	4812	84
6	76.01	18.36	70.73	0.0	0.0	3901	1338
7	74.48	94.57	76.32	0.3	0.14	4149	89
Without PCA	58.48	**96.03**	61.92	0.19	0.09	6751	**65**

Table 8: Agglomerative Clustering using Average + cosine

Observations for 'Average' linkage:
i) For euclidean/l2 affinities (table 6), number of false negatives is large, so we are missing a lot of pulsars.
ii) For manhattan/l1 affinities, sum total of false positives and false negatives is low and this is also reinforced by a high value of completeness measure. Also, the accuracy of true positives (predicting 1's) and true negatives (predicting 0's) is very good in the case PCA is not used.
iii) For cosine affinity, we observe a large number of false positives, so it is overestimating the number of pulsars. But for the cases when 2,3 princical components are used in PCA, This number is small compared to other PCA cases and even without PCA. But the drawback is that this small number is still close to the number of actual pulsars.
iv) The best performance with the Average linkage is when 3 principal components of PCA are used.

PCA components used	Acc. of 0's prediction(%)	Acc. of 1's prediction(%)	Overall Acc.(%)	Homogeneity Measure	Completeness Measure	# of False Positives	# of False Negatives
2	99.97	41.12	94.58	0.32	0.6	5	965
3	**100.0**	20.62	92.73	0.15	0.5	**0**	1301
4	97.39	**75.96**	95.43	0.49	0.48	424	**394**
5	99.88	58.39	96.08	0.46	0.67	19	682
6	**100.0**	17.02	92.4	0.13	0.48	**0**	1360
7	99.88	60.52	96.28	0.48	**0.68**	19	647
Without PCA	99.51	70.77	**96.88**	**0.55**	0.67	80	479

Table 9: Agglomerative Clustering using Complete + Euclidean/l2

PCA components used	Acc. of 0's prediction(%)	Acc. of 1's prediction(%)	Overall Acc.(%)	Homogeneity Measure	Completeness Measure	# of False Positives	# of False Negatives
2	99.99	28.55	93.45	0.21	0.54	2	1171
3	78.09	**82.86**	78.52	0.22	0.12	3563	**281**
4	99.15	64.73	95.99	0.46	0.58	139	578
5	**100.0**	17.63	92.46	0.13	0.48	**0**	1350
6	92.79	10.62	85.27	0.0	0.0	1172	1465
7	**100.0**	10.25	91.78	0.07	0.43	**0**	1471
Without PCA	99.03	72.91	**96.64**	**0.54**	**0.62**	157	444

Table 10: Agglomerative Clustering using Complete + Manhattan/l1

PCA components used	Acc. of 0's prediction(%)	Acc. of 1's prediction(%)	Overall Acc.(%)	Homogeneity Measure	Completeness Measure	# of False Positives	# of False Negatives
2	57.49	93.72	60.81	0.17	0.07	6911	103
3	68.69	94.69	71.07	0.25	0.12	5091	87
4	66.85	89.69	68.94	0.19	0.09	5390	169
5	57.98	2.07	52.86	0.13	0.06	6832	1605
6	59.27	1.95	54.02	0.12	0.06	6622	1607
7	**79.26**	92.68	**80.49**	**0.32**	**0.17**	**3372**	120
Without PCA	65.93	**94.87**	68.58	0.23	0.1	5540	**84**

Table 11: Agglomerative Clustering using Complete + cosine

Observations for 'Complete' linkage:

i) For euclidean/l2 linkages, best total accuracy is without PCA. The homogeneity and completeness scores also support this. Even though PCA=4 has lower false negatives, no PCA has the least sum of false positives and false negatives. Therefore, Complete + Euclidean is a good combination with low errors.

ii) For manhattan/l1 linkages, just like euclidean, the best accuracy, homogeneity and completeness scores are without PCA. But the number of false negatives are comparable to the number of pulsars, thereby giving a fairly wrong result.

iii) For cosine affinity, the best case is with 7 PCA components. This gave the least number of the sum of false positives and false negatives. Unlike the previous cases, the false positive number is high, so the number of pulsars is overestimated here.

6. CONCLUSIONS

I In this project, we have tested the K-Means and Mini Batch K-Means algorithms first. In some cases, Mini Batch K-Means performed better than K-Means even though it is uses less computation. But the overall highest accuracy is achieved by K-Means. The experiments with Mini Batch Kmeans have not been included here since the differences in computation time and accuracy were not prominent but can be viewed on our github repository at (8).

II After this, we have tested the AgglomerativeClustering funtion of sklearn using all it's possible combinations of affinities and linkages. When the 'complete' linkage was used, except for the case when affinity was cosine, we have obtained fairly accurate results. The best overall accuracy among all the results we have obtained is for the 'average' linkage with 'manhattan' affinity.

III The Overall Accuracy obtained from Hierarchical Clustering is greater than that obtained from K-Means. Also, the maximum homogeneity and completeness scores obtained from hierarchical clustering (0.59, 0.73 respectively, in average linkage with manhattan affinity) are greater than that obtained from K-Means (0.54, 0.67).

IV Using unsupervised clustering algorithms might not be the best way to annotate data. Using semi-supervised and self-supervised learning methods (when there is abundant unlabeled data and very little labeled data) will probably outperform these simple clustering algorithms.

REFERENCES

[1]Xu, D. & Tian, Y , "A Comprehensive Survey of Clustering Algorithms," Ann. Data. Sci. (2015) 2(2):165–193.

[2]https://archive.ics.uci.edu/ml/datasets/HTRU2

[3]https://scikit-learn.org/stable/modules/clustering.html

[4]Garima, H. Gulati and P. K. Singh, "Clustering techniques in data mining: A comparison," 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2015, pp. 410-415.

[5]G. Ahalya and H. M. Pandey, "Data clustering approaches survey and analysis," 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), Noida, 2015, pp. 532-537.

[6]K. M. A. Patel and P. Thakral, "The best clustering algorithms in data mining," 2016 International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, 2016, pp. 2042-2046.

[7]J. Oyelade et al., "Data Clustering: Algorithms and Its Applications," 2019 19th International Conference on Computational Science and Its Applications (ICCSA), Saint Petersburg, Russia, 2019, pp. 71-81.

[8]https://github.com/sayZeeel/DSAproject

[9]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity_completeness_v_measure.html