

CS 422 DATA MINING

HW 5 SOLUTIONS

1 Mapreduce

I am using 2 Map Reduce Jobs. 1st Job gives count of even integers and total no in all files. 2nd sums up all the count and gives the percentage of the even no

1st mapper reads file content and then outputs the even integers and total no count together.

1st reducer (we are considering a system having files), it sums up even integers and total count for all files in the system

2nd mapper, it just passes the input as output.

2nd reducer now will sum up all the even and total for all files and output the percentage of files

INPUT/OUTPUT:

Map Reduce Job1

Mapper1 # It takes in the file content and counts the even and total in respective fileID

Input: (fileID, content) ; where content = content of the input file

Output: (fileID, <Even, Total>);

Even = even numbers per file, Total = total no of integers per file

Reducer1 # considering there are multiple files with same fileID in a system and hence the input is an array of even & total count in each file

Input: (fileID, array<Even, Total>);

Output: (1, <Even_S, Total_S>);

where Even_S = Total no of even no across all files M = Total no of integers across all files in a system

Map Reduce Job2

Mapper2

Input: (1, <Even_S, Total_S>);

Output: (1, <Even_MS, Total_MS>);

Reducer2

Input: (1, <Even_MS, Total_MS>);

Output: Percentage of even no

Pseudo Code:

Map Reduce Job1

Class Mapper(fileID, file_content)

```
Even=0, total =0;
String [] tokens = file_content.split("\n")
for (String token: tokens)
{
    Int value = Integer.parseInt(token);
    if(value%2==0)
        Even ++;
    Total++
}
```

Emit (fileID, < Even, Total>)

Class Reducer (fileID, array < Even, Total>)

```
arr = array< Even, Total>
for values in arr:
{
    Even_S+=arr.Even
    Total_S += arr.Total
}
Emit (1,<Even_S, Total_S>)
```

Map Reduce Job2

Class Mapper (1, (1,<Even_S, Total_S>)

Input: (1 , <Even_S, Total_S>);

Output: (1 , <Even_MS, Total_MS>);

Class Reducer (1 ,array <Even_MS, Total_MS>);

arr = array <Even_MS, Total_MS>

for values in arr:

{

Even_Total+=arr.Even_MS

Total_Total += arr.Total_MS

}

Emit (1, (Even_S/ Total_S)*100)

- For faster and more computation in parallel we need to use more mappers and reducers. More no of mappers and reducers provides an increase in efficiency for this system as multiple files in the filesystem have the mapper running in parallel this allows for the counts to be provided faster.
- If we consider that we have small file then using multiple reducers and mappers would not be useful as it might take more time than 1 mapreduce job, so in this as we have multiple files it is better to use multiple files/large file.

METHOD 2

INPUT/OUTPUT:

Map Reduce

Mapper

Input: (fileID, content) ; where content = content of the input file

Output: (fileID, <Even, Total>);

Even = even numbers per file, Total = total no of integers per file

Combiner

Input: (fileID, array <Even, Total>);

Output: (1, <Even_S, Total_S>);

where Even_S = Total no of even no across all files M = Total no of integers across all files in a system

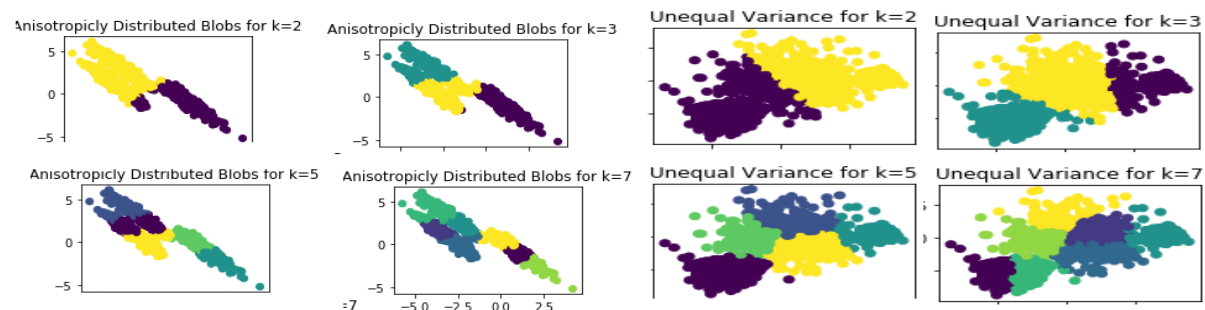
Reducer

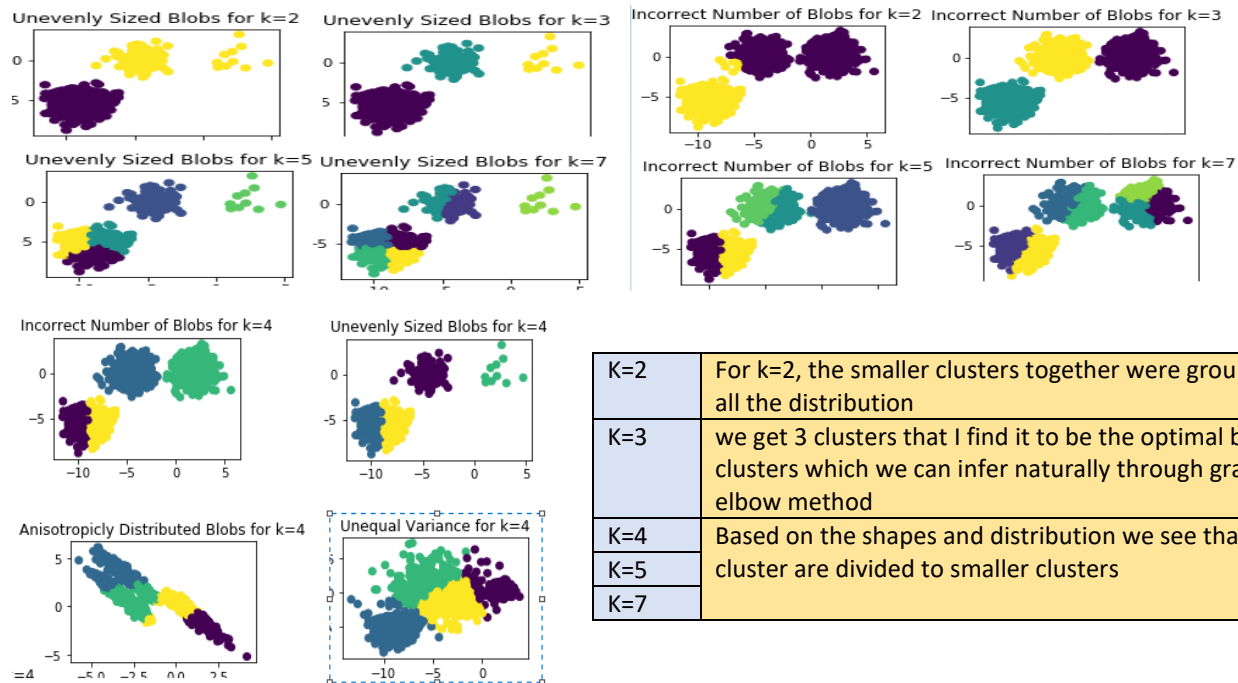
Input: (1 , <Even_S, Total_S>);

Output: Percentage of even no

2) Clustering

A



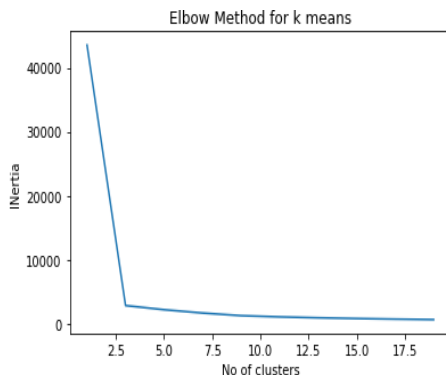


- For “Unevenly Sized and normal distribution, a value of k=3 performed the best. k=2 was too low and while k=4 was too high and splits the largest cluster into smaller (choosing the proper value of k is needed based on the distribution else there is misclassification)
- For unequal variance, all the data points have different variance and distance between all points in the same cluster is not same as there is change in SD, variance and hence as the data is changed k means is not able to properly distinguish clusters. As the data is not properly separated to clusters kmeans doesn't perform well. For larger variance, the cluster having higher variance are used to split into smaller because these have larger size/distance from centroid and the clusters with lower variance doesn't get divided
- Higher density cluster are used for splitting into smaller ones for larger k which can be seen for k=7
- when the data is anisotropic distributed, it's difficult for the K-means algorithm to correctly classify points because as the clusters are assigned based on the distance from centroid
- We can see from the fig that the optimal no of clusters is 3 and when k<3, the clusters that are closer are formed to one cluster .For more than k=3 leads to form larger centroids to smaller ones in order to form k clusters(they become granular and having no proper clustering with little value for clusters

K-means CONCLUSION

K-means has problems when clusters are of below types.

- Densities, Non-globular shapes, anisotropic distribution because it works on Euclidean /Manhattan distance as the data points are allocated to the centroids based on the distance from it not different shapes or density parameter (not able to decide which centroid each data point is closer)
- Sizes (k-means doesn't handle non-spherical type, unevenly sized of data and works well for spherical) .
- when the data contains outliers (points which are far away from the cluster centroid when comparison to the rest of the points in that cluster) and hence doesn't work well with unequal variance as the outliers influence the centroids. Need to have even distribution of data while performing K means.
- The clusters need to be well separated or with proper distribution of density and shape.



Note: We can use the elbow method to determine the value of k, one of the disadvantages of k means you need to feed the value of k which is not the case in other clustering algorithms. Hence, we can use Elbow method for determining the value of k.

This Elbow method was applied to the data without any transformation we can see that for no of clusters =3, there is a sharp decrease in the value and this point is called an elbow point which decide the optimal value for the given data

we see that for value k=3 we get an elbow point in the graph which is the optimal value of k

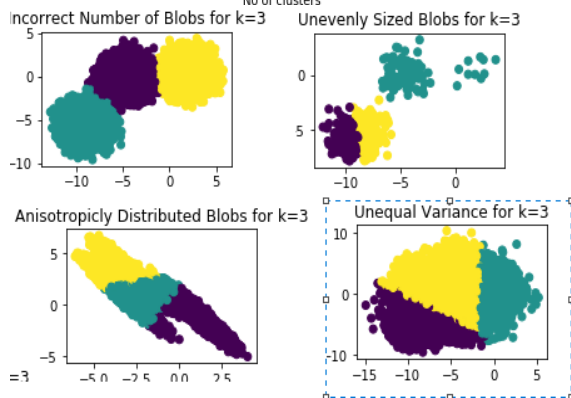


Fig a

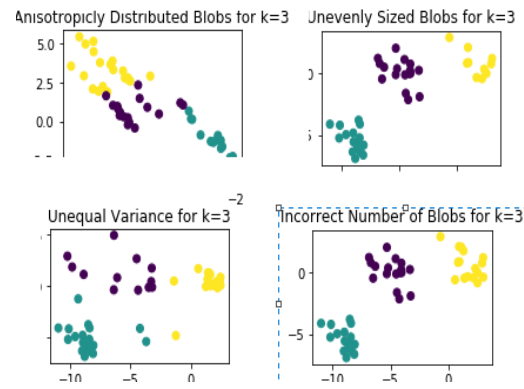


Fig b

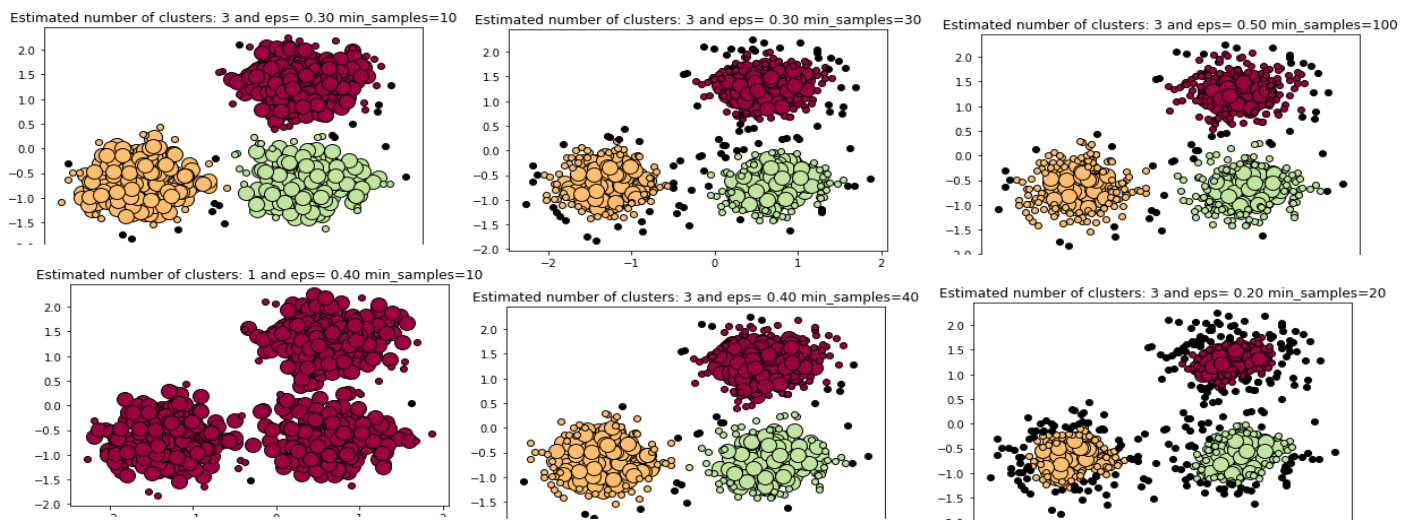
Fig a: INCREASING THE DATA POINTS TO 50000 when I tried to increase the data samples to 50000, I see that it performed very badly on unevenly sized blobs whereas it had performed very well with 1500 samples. This explains to us that k means doesn't work well for unequal sized data. Thus we can conclude that k means works well only for equal size, distribution with proper shape like spherical.

Fig b: DECREASING THE DATA POINTS TO 500: Even after trying this we can see that kmeans doesn't work well for anisotropic data.

b) DBSCAN			
	Min samples 3	Min samples 10	Min samples 20
Epsilon 0.1	Estimated number of clusters: 26 Homogeneity: 0.810 Completeness: 0.397 V-measure: 0.533 Adjusted Rand Index: 0.393 Adjusted Mutual Information: 0.386 Silhouette Coefficient: -0.123	Estimated number of clusters: 12 Homogeneity: 0.313 Completeness: 0.249 V-measure: 0.277 Adjusted Rand Index: 0.024 Adjusted Mutual Information: 0.240 Silhouette Coefficient: -0.366	Estimated number of clusters: 1 Homogeneity: 0.019 Completeness: 0.225 V-measure: 0.035 Adjusted Rand Index: 0.001 Adjusted Mutual Information: 0.018 Silhouette Coefficient: -0.162
Epsilon 0.3	Estimated number of clusters: 2 Homogeneity: 0.003	Estimated number of clusters: 3 Homogeneity: 0.953	Estimated number of clusters: 3 Homogeneity: 0.941

	Completeness: 0.048 V-measure: 0.005 Adjusted Rand Index: 0.000 Adjusted Mutual Information: -0.000 Silhouette Coefficient: -0.082	Completeness: 0.883 V-measure: 0.917 Adjusted Rand Index: 0.952 Adjusted Mutual Information: 0.883 Silhouette Coefficient: 0.626	Completeness: 0.851 V-measure: 0.894 Adjusted Rand Index: 0.933 Adjusted Mutual Information: 0.851 Silhouette Coefficient: 0.619
Epsilon 0.5	Estimated number of clusters: 1 Homogeneity: -0.000 Completeness: 1.000 V-measure: -0.000 Adjusted Rand Index: 0.000 Adjusted Mutual Information: -0.000	Estimated number of clusters: 1 Homogeneity: -0.000 Completeness: 1.000 V-measure: -0.000 Adjusted Rand Index: 0.000 Adjusted Mutual Information: -0.000	Estimated number of clusters: 1 Homogeneity: -0.000 Completeness: 1.000 V-measure: -0.000 Adjusted Rand Index: 0.000 Adjusted Mutual Information: -0.000

- Epsilon is the maximum distance a point is within the neighborhood/cluster
- min_samples is the minimum number of points needed to be found within epsilon.
- Points that are below the min_samples threshold and their epsilon is closer to a core point are called border points. All other outlier points that are not core or border points are called noise points.
- For DBSCAN the shape of cluster can be anything whereas k means needed spherical shape. This works based on density, points that are closer and dense are considered to belong to same cluster and the ones that are far away are considered as noise.
- A low minPts and low epsilon means it will build more clusters from noise, so don't choose it too small
- A higher value of epsilon means less no of clusters, so the best value of epsilon is around 0.3 and min samples near to 10 where we are getting the optimal clustering and better homogeneity and v measure
- A positive value of the scores indicate that the points were assigned to their clusters correctly. A negative value of the measures indicates the points were assigned incorrect clusters.

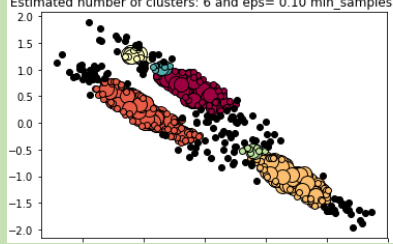
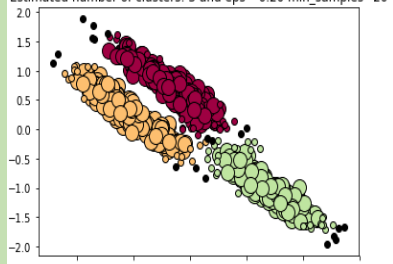
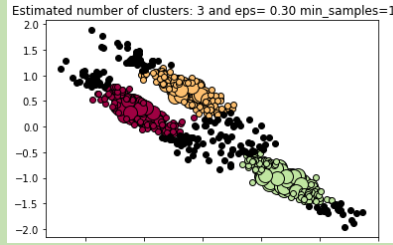


- I tried giving lesser eps=0.1 and min_samples =10, I got more clusters greater than 10. We can infer that by giving less eps, the cluster radius is very small and hence it leads to more no of clusters.
- When I tried for eps=0.2 and min_samples =20, I got 3 clusters and considering anything outside the dense clusters as outliers but if I tried to increase the value of min_samples, it considered the data points

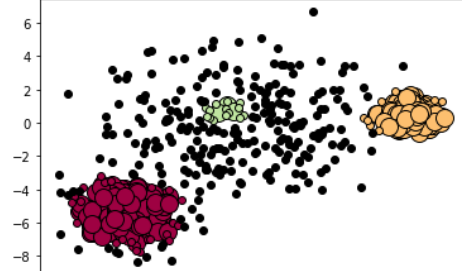
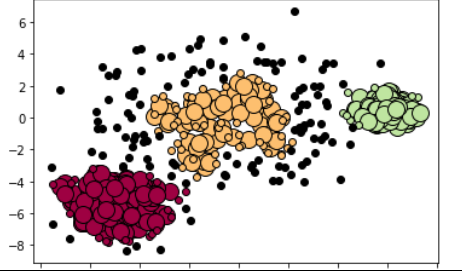
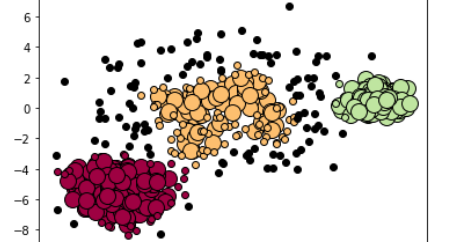
that belong to certain cluster as outliers. Hence increasing the value of min samples beyond certain range considers data points as outliers and decreases the no clusters as more points begin to fall below the threshold of minimum points required to be considered as a border point.

- When we decrease the value of eps, we need to decrease the value of min_samples to get the optimal no of clusters.
- I gave eps=0.4 and min_samples =10, this gave only one cluster. Thus, if we give very small value it leads to less no of cluster. I increased the min_samples to 40 and it resulted in correct no of clusters. We can infer that if we try to increase the epsilon then we have to increase the no of samples inorder to obtain the optimal no of clusters but the outliers are not identified correctly(inorder to get 3 cluster for eps=0.5, I had to increase min_samples to greater than 70).The explanation is if we increase the distance between two clusters we need to increase the minimum no of points to form a dense cluster
- If min_samples is too high, and epsilon is less, then all points will be considered as noise points, and causes the program a run-time error.
- Changing epsilon affects the size of the clusters because it decides whether points are in the same cluster and min_sample decide which data points are chosen as core points. Thus, giving a small no epsilon leads to more no of clusters and high epsilon leads to less no of clusters. Hence, we need to use the optimal value for eps and min_sample such that value of homogeneity, completeness, V measure is better, and outliers are also detected.

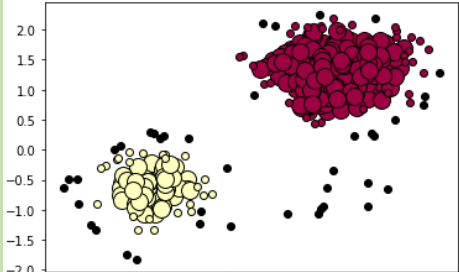
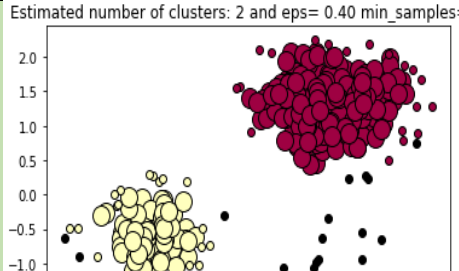
C- For Anisotropic data:

For this If the value of eps was taken 0.1 and below which the no of cluster were more.Min_sample I considered 15 and I got 6 clusters.With increase in no of min_samples we get more dense clusters and outliers/noise and the no of clusters keep decreasing ,but we don't get the optimal clusters with proper classification .Hence low value of 0.1eps doesn't work	Estimated number of clusters: 6 Estimated number of noise points: 145 Homogeneity: 0.799 Completeness: 0.549 V-measure: 0.651 Adjusted Rand Index: 0.604 Adjusted Mutual Information: 0.547 Silhouette Coefficient: 0.168	Estimated number of clusters: 6 and eps= 0.10 min_samples=15 
For eps=0.2 ,I tried min_samples=10 and got 3 clusters but with increase in min_samples I could get more dense clusters and able to detect the outliers at min_samples=20. Greater than 20 leads to more noise (more data points are considered as outliers) So eps=0.2 and min_samples=20 is the best possible parameter value for the anisotropic data.	Estimated number of clusters: 3 Estimated number of noise points: 19 Homogeneity: 0.849 Completeness: 0.784 V-measure: 0.815 Adjusted Rand Index: 0.877 Adjusted Mutual Information: 0.783 Silhouette Coefficient: 0.444	Estimated number of clusters: 3 and eps= 0.20 min_samples=20 
For eps=0.3 and min_samples<100 I got 1 cluster and hence for min_samples =101 I got 3 cluster but consists of more core points allocated as outliers with low scores. Even though 0.3 eps provides optimal no of cluster it doesn't provide better score when compared to eps=0.2 and min_samples=20	Estimated number of clusters: 3 Estimated number of noise points: 182 Homogeneity: 0.733 Completeness: 0.581 V-measure: 0.648 Adjusted Rand Index: 0.600 Adjusted Mutual Information: 0.580 Silhouette Coefficient: 0.346	Estimated number of clusters: 3 and eps= 0.30 min_samples=101 

For different variance:

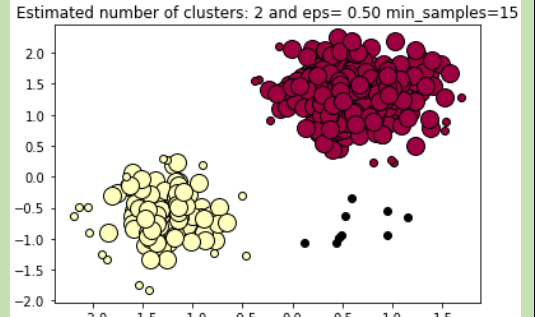
When I tried for less than eps=.6 I got more noise and less clusters. As there is high variance in the data we can see that having a small value of eps doesn't work and hence we have to more than 0.6	Estimated number of clusters: 3 Estimated number of noise points: 250 Homogeneity: 0.885 Completeness: 0.823 V-measure: 0.853 Adjusted Rand Index: 0.875 Adjusted Mutual Information: 0.823 Silhouette Coefficient: 0.422	Estimated number of clusters: 3 and eps= 0.60 min_samples=10 
eps=0.7, min_samples=8 gave a very good clustering with high scores and classifying the data points as noise with increase of min samples the scores value decreases and hence eps=0.7 and min_sampels =8 provides the best results	Estimated number of clusters: 3 Estimated number of noise points: 120 Homogeneity: 0.929 Completeness: 0.767 V-measure: 0.840 Adjusted Rand Index: 0.830 Adjusted Mutual Information: 0.766 Silhouette Coefficient: 0.559	Estimated number of clusters: 3 and eps= 0.70 min_samples=8 
At eps=0.9 it provides 3 clusters with proper noise classification and with increase in the min_samples the value of the scores decreases.	Estimated number of clusters: 3 Estimated number of noise points: 104 Homogeneity: 0.921 Completeness: 0.766 V-measure: 0.836 Adjusted Rand Index: 0.831 Adjusted Mutual Information: 0.765 Silhouette Coefficient: 0.572	Estimated number of clusters: 3 and eps= 0.90 min_samples=12 

For uneven sized data

I tried with lower than 0.2 eps but as the clusters are dense and larger, we require little more eps than 0.2 else lot of data points were considered as noise. So for eps0.3 and num_samples=15 we get good scores but if we increase num_samples it results in more noise and less scores	Estimated number of clusters: 2 Estimated number of noise points: 40 Homogeneity: 0.833 Completeness: 0.690 V-measure: 0.755 Adjusted Rand Index: 0.850 Adjusted Mutual Information: 0.688 Silhouette Coefficient: 0.583	Estimated number of clusters: 2 and eps= 0.30 min_samples=15 
Again we get better score for eps=0.4 than 0.3 but if we increase num_samples>15 we get more noise and lesser scores	Estimated number of clusters: 2 Estimated number of noise points: 21 Homogeneity: 0.915 Completeness: 0.836 V-measure: 0.873 Adjusted Rand Index: 0.945 Adjusted Mutual Information: 0.834 Silhouette Coefficient: 0.655	Estimated number of clusters: 2 and eps= 0.40 min_samples=15 

For $\text{eps}=0.5$ and $\text{min_samples}=15$ we get very good scores(homogeneity ,v measure) when compared to other eps values
Hence for $\text{eps}=0.5$ and $\text{num_samples}= 15$ we get very good clustering

Estimated number of clusters: 2
Estimated number of noise points: 9
Homogeneity: 0.978
Completeness: 0.987
V-measure: 0.983
Adjusted Rand Index: 0.997
Adjusted Mutual Information: 0.978
Silhouette Coefficient: 0.670



Anisotropic Comparision

- The best result is when epsilon (0.2) and min_sample (20) where few points are considered as noise.
- Even though each cluster has same size,shape with high density ,kmeans cant distinguish the two cluster that are near to each other as center based method(Euclidian distance) fails to distinguish clusters.
- Hence using small eps value helps to figure out denser region and as they are not separated by a large distance . If we use large eps we wont be able to distinguish clusters at all and therefore we have to use small value of epsilon for anisotropic data .
- Must not decrease below 1 eps as it will have a negative effect and create more clusters.
- DBscan performs well as we are reducing the neighborhood around a data point that belongs to the same cluster and clusters of high dense are formed regardless of their different shape.

Unequal Variance blobs:

- We can see that 3 clusters have different variance and some cluster are denser when compared to others. At $\text{eps}=0.7$ and $\text{min_samples}=8$ we get best classification and includes less no of noise.
- For this type of data distribution we can see that the ones that are not dense are considered as noise ,inorder to capture the cluster having large variance we need to give a larger eps value .A small eps value will detect only few cluster and have large no of noise.
- As there is large variance in the middle cluster, if we try to give more no of min_samples and eps, it will lead to more clusters and noise
- **Thus one disadvantage is that is the data/cluster is sparse ,then dbscan doesn't recognize all the data points as core which we can see in the middle cluster as it density based.**

Unevenly sized blobs:

- We can see that as there are dense cluster of different size, we cannot give small value of eps and hence for $\text{eps}=0.5$ we get a very good clustering and $\text{min_samples}=15$. If we try more min_samples it didn't affect score (no improvement)
- Hence a larger epsilon 0.5 is required to find the clusters with min no of noise.

DBSCAN CONCLUSION

- *DBSCAN cannot cluster data sets well with large differences in densities, since the eps combination cannot then be chosen appropriately for all clusters.*
- Also, it doesn't work well for sparser dense cluster as it considers many points as noise
- Hence if the data distribution is spherical and with equal density kmeans works well but if there is unequal density, variable size/shape dbcan works well.
- The other advantage of dbscan is we don't need to specify/know the value of k as in k means
- The kmeans takes less time when compared to dbscan
- DBScan is robust in detecting clusters of arbitrary shapes and sizes, given that the density among different clusters are similar. kMeans, on the other hand, works well with spherical shaped data, and can provide intuitive clusters even in some cases of different variance.

D)

The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. The 20 newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering. Approximately 4% of the articles are cross-posted. The articles are typical news groups postings and thus have headers including subject lines, signature files, and quoted portions of other articles. The data is organized into 20 different newsgroups, each corresponding to a different topic. Some of the newsgroups are very closely related to each other (e.g. **comp.sys.ibm.pc.hardware** / **comp.sys.mac.hardware**), while others are highly unrelated (e.g **misc.forsale** / **soc.religion.christian**). Here is a list of the 20 newsgroups, partitioned (more or less) according to subject matter:

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

For this assignment we choose 4 groups to train our data: '**alt.atheism**', '**talk.religion.misc**', '**comp.graphics**' and '**sci.space**'. Out of these "'alt.atheism','talk.religion.misc' are closely related and can be considered as one.

Homogeneity:

If clusters contain data points of a single class, then there is a high homogeneity. It represents the compactness of the cluster. Higher value means each cluster has only one type of data point of a particular class.

Completeness is obtained when the data points of a given class are elements of same cluster. Thus, higher value means more data points of class A belong to only one cluster

V-measure is the harmonic mean between homogeneity and completeness. A higher V-measure is expected

$$V \text{ measure} = 2 * (\text{homogeneity} * \text{completeness}) / (\text{homogeneity} + \text{completeness})$$

Silhouette Coefficient: determine the degree of separation between clusters. The best and worst value is 1 and -1.(0 indicates overlapping clusters, negative value indicates data sample has been assigned to wrong cluster)
 All the scores are need to be closer to 1 in order for good clustering

K means

K	Homogeneity	Completeness	V-measure	Silhouette	Time
2	0.142	0.289	0.190	0.01	6.335
4	0.175	0.204	0.189	0.011	5.217
9	0.325	0.219	0.262	0.016	9.228
14	0.342	0.195	.249	0.019	5.4
20	0.4	0.203	0.27	0.018	8.3
100	.597	.192	0.292	0.022	64

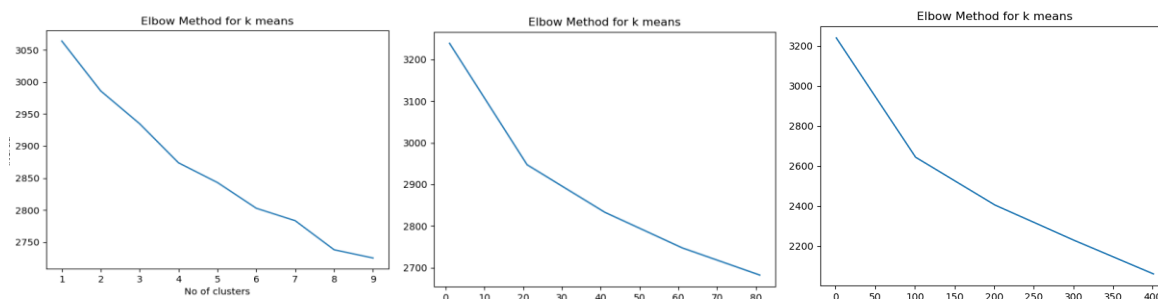
K means Mini Batch

K	Homogeneity	Completeness	V-measure	Silhouette	Time
---	-------------	--------------	-----------	------------	------

2	0.081	0.168	0.109	0.111	0.877
3	0.015	0.02	0.017	0.01	0.94
4	0.192	0.200	0.196	0.01	0.9
9	0.372	0.267	0.311	0.013	1.89
14	0.42	0.237	0.303	0.016	1.7
20	0.407	0.206	0.273	0.016	2.99
100	0.563	0.207	0.303	0.022	41.6

MiniBatch kmeans uses subsample of the data at each iteration and hence it is faster than kmeans.

- In the above table, there is no clear output that confirms the value of k.
- If we look at the k means table for k=100 we got v measure and silhouette score high, but this is not the no of clusters based on the data.
- When we try the mini batch, this also doesn't give a proper k value as v measure is high for k=9 but silhouette is high for k=100.
- We can conclude that the k means or minibatch fails at clustering
- We can also see that the minbatch takes lesser time when compared to the k means.



- As I didn't get any proper k value I try to plot the elbow method and this shows that the value of k is not determined for this data. All the 3 graph are same but the range of no of clusters in X axis are different. We cannot determine the elbow point in any of these plots.
- Hence we have to try other means of the transformation to the data inorder to obtain the optimal k value.

Try different values of the number of clusters k. Discuss what seems to be the best value, according to the evaluation, and how does it compare to what you've learned about the data

Lsa= 200					
K	Homogeneity	Completeness	V-measure	Silhouette	Time
3	0.1771	0.224	0.197	0.023	0.65
4	0.392	0.391	0.391	0.031	0.8
9	0.372	0.262	0.307	0.031	0.654
20	0.422	0.207	0.278	0.055	2.99

- LSA is one of the NLP technique used analyze relation among the documents, terms by producing concepts related to terms and document.
- LSA reduces the dimension of the data and hence faster computation, data is generalized. It converts words to topics related to document.
- Latent semantic analysis determines the relation between doc and words (we can that word having same meaning will occur in document having same topic)
- LSA result faster processing and dimensionality reduction .
- I have used 200 value ,we can see that the k=4 .We have used for 4 and even though two of them belong to the same group by applying lsa we are able to get a proper clustering than normal k means

- Homogeneity is a good score for k=20 but V measure is high for 4 and which is considered as it is a harmonic mean between Homogeneity and completeness .

TF IDF					
K	Homogeneity	Completeness	V-measure	Silhouette	Time
3	0.59	.746	.6	0.08	.07
4	0.335	0.449	0.384	0.006	0.8
9	0.583	0.479	0.526	0.008	1.3
20	0.648	0.351	0.455	0.012	2.8

Term frequency-inverse document frequency, effect how important a word is to a document in a collection or corpus. The tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.

- Tf-idf can be successfully used for stop-words filtering.

Term frequency

- The weight of a term that occurs in a document is simply proportional to the term frequency

Inverse document frequency

- inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.
- The tf-idf is the product of two statistics, term frequency and inverse document frequency.
- The terms with the highest tfidf score are the ones that describe the topics best

Parameters

- **Min_df** ignores terms that have a document frequency lower than the given threshold. (term must appear in 2% of the documents of this case).
- **max_df**. ignores terms that have a document frequency higher than the given threshold. (In this case it remove more than 50%, ie to remove the stop words)
- Homogeneity is high for k=20 but V measure is high for k=3 along with completeness.TF-IDf determined the optimal clusters.

HASHING:					
K	Homogeneity	Completeness	V-measure	Silhouette	Time
3	0.146	0.196	0.165	0.014	0.8
4	0.146	0.147	0.144	0.013	0.69
9	0.34	0.24	.282	0.014	1.46
20	0.36	0.198	0.256	0.018	2.97

- Hash functions are an efficient way of mapping terms to features and uses low memory
- Hashing is conversion of a collection of text documents to a matrix of token occurrences. Hashing is another technique of transformation. HashingVectorizer applies a hashing function to term frequency counts in each document.
- By looking at the table the results is similar to the normal k means and hence using hashing doesn't improve the results in this case.
- Hashing provides scalability(Converting the data into a sparse matrix for faster processing), time and memory-efficiency.

NUM_FEATURES=2500					
K	Homogeneity	Completeness	V-measure	Silhouette	Time
3	0.142	.205	.168	0.012	0.374
4	0.214	0.246	0.229	0.012	0.39
9	0.355	0.236	0.283	0.018	1.177

20	0.406	0.202	0.27	0.029	1.146
----	-------	-------	------	-------	-------

- We can see that the results don't make any sense and we are not able to decide the optimal k value. Even though after reducing the features from 10000 to 2500 we don't see any improvement because it does not eliminate the common words
- when I increase number of features to 30,000 in the dataset, cluster quality decreases because of overfitting
- Increasing the no of features causes overfitting and high dimension ie data points are sparse when we increase the dimension.
- Low no of features might lead to high generalization and losing of important features.

USING ALL TFIDF,LSA,NUM_FEATURES Together and performing K means

K	Homogeneity	Completeness	V-measure	Silhouette	Time
3	0.59	0.749	0.66	0.027	0.349
4	0.534	0.546	0.54	0.025	.507
9	0.5	0.37	0.426	0.047	0.69
20	0.633	0.323	0.428	0.057	1.59

- LSA transformation and Max num features transformation performs dimensionality reduction, unimportant /discriminative features and removal of redundant features. These increase the speed and performance
- Hashing improves speed rather than clustering quality and TFIDF removes stop words.
- By looking at the above table we can see that for k=3 we get a higher value is obtained than when we used any other transformation.

20 NG CONCLUSION

- K-means works well for text data
- It is important to pre-process the data to select features (e.g. based on term frequencies) and to apply appropriate weights (e.g. tf-idf)
- Dimensionality reduction (e.g. LSA) help to denoise data and improve the similarity (or distance) calculations. That leads to better clustering

CONCLUSION

Clustering is a great way to discover structure in the data and groups similar data records together.

K-means and DBScan are two clustering algorithms that use very different approach to creating clusters. K-means is fast and works well on many data types. However, it cannot handle well data that has certain complex shapes and densities DBScan often performs well on the data for which K-means won't produce good clusters. However, it also doesn't handle certain data densities well.

For both algorithms It may be better to create many smaller clusters if the data shapes and densities are complicated. That may produce many more clusters than are present in the data but at least similar records will more likely end up in the same cluster. We see it in the examples for k-Means with k=7 and for DBScan with smaller values for epsilon (see example for anisotropic data with eps=0.1 and 6 resulting clusters).

K-means is a good algorithm for clustering text data. It is more efficient than DBScan on data with high dimensionality. Using an appropriate feature selection and weighting is very important to get good clustering results.