Project 1 : Data Representations and Clustering

Sidarth Srinivasan (UID - 005629203), Shweta Katti (UID - 505604846)

February 8, 2022

Course : ECE 219 Large Scale Data Mining
Term : Winter 2022

The first part of the project explores the various unsupervised clustering models applied on textual data. The dataset being used for the same is the 20 Newsgroups and is available in scikit-learn.

We start by analyzing the performance of K-means clustering algorithm with 2 clusters. We consider the various dimensionality reduction techniques such as SVD, NMF and also analyze the performance of the model with no reduction techniques. We then move on to implement clustering models for the entire 20 classes. Along with the above mentioned reduction techniques, we also explore another linear reduction technique called UMAP. We then extend our analysis to other clustering models such as Agglomerative clustering, DBSCAN and HDBSCAN. All these models are compared based on the evaluation metrics mentioned below and the best models are highlighted in the report.

The evaluation metrics used are: Homogeneity, Completeness, V-Measure, Adjusted Rand Index, Adjusted Mutual Information Score.

In the second part of this project, we employ the VGG-16 architecture to extract features from the given tf_flowers dataset. Deep Learning and Clustering of image data. Here, we use the tf_flowers dataset provided to us with a VGG network which has been pretrained on the ImageNet dataset. We test this with various dimensionality reduction methods such as SVD, UMAP and autoencoder.

1 Question 1

The dimensions of the TF-IDF matrix obtained is **(7882, 18469)**. The sparse matrix was generated after excluding stop-words, removing headers/footers and setting min_df = 3.

2 Question 2

We observe the results o=for TF-IDF matrix by plotting the contingency matrix as shown in Figure 1.
Consider a contingency matrix with elements C_{ij} where i represents the actual class and j represents the predicted cluster class. We can say that a contingency table is usually square-shaped but this need not always be true because a category can exist in the reference data that does not exist in the mapped cluster, and vice versa.

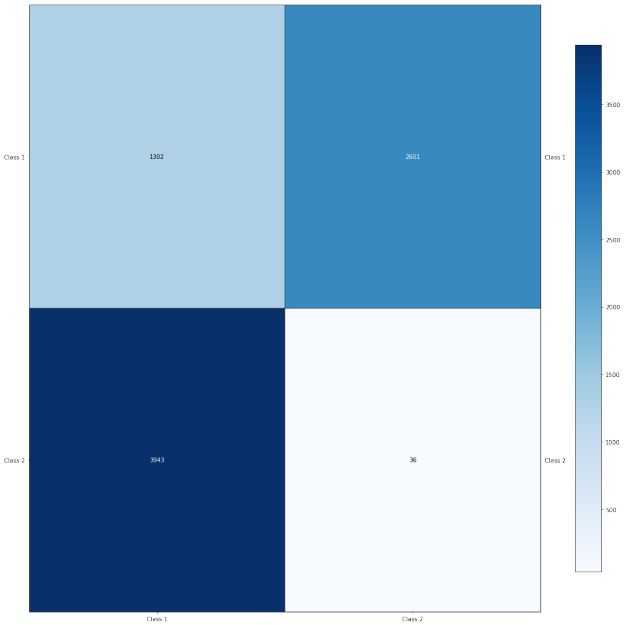


Figure 1: Contingency Matrix for K means Clustering with TF-IDF

3 Question 3

The 5 clustering measures obtained are as follows:

- Homogeneity = 0.427
- Completeness = 0.464
- V-measure = 0.445
- Adjusted Rand-Index = 0.436
- Adjusted Mutual Information Score = 0.445

4 Question 4

Figure 2 shows the percentage of variance that the top r principle components retain v.s. r. We observe that as we increase r, we also see a monotonic increase in the variance retention percentage.

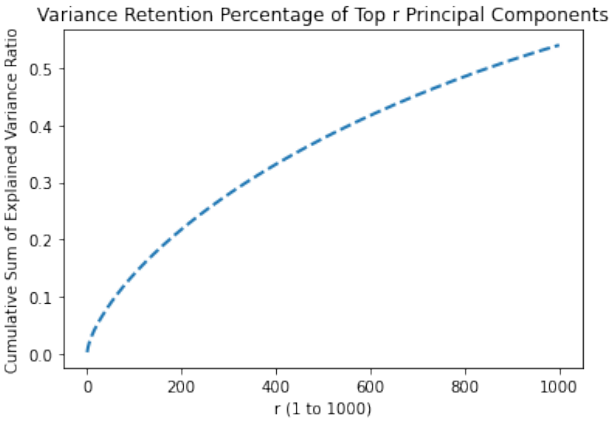
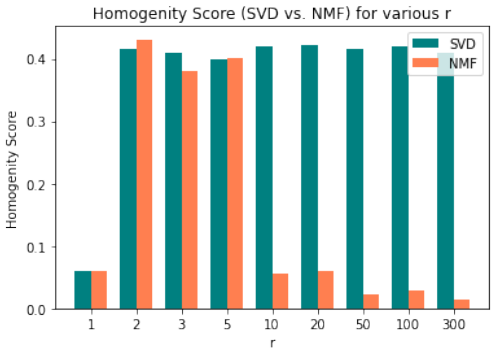


Figure 2: percentage of variance that the top r principle components retain v.s. r

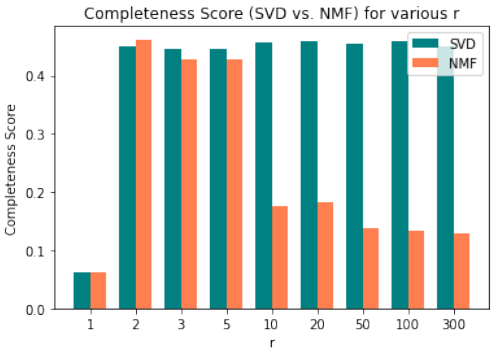
5 Question 5

From the figures and the results obtained we note that the best value for r for

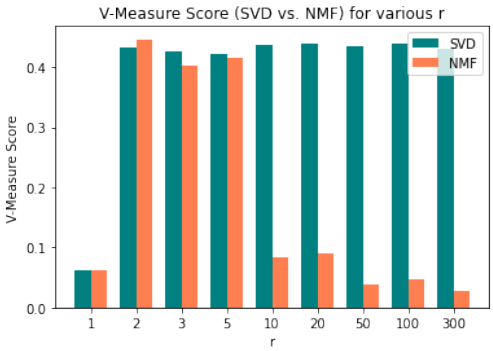
- SVD = 20
- NMF = 2



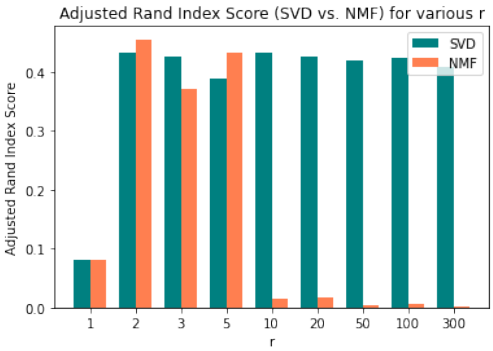
(a) Homogeneity



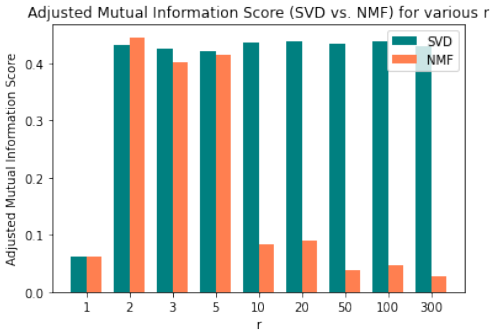
(b) Completeness



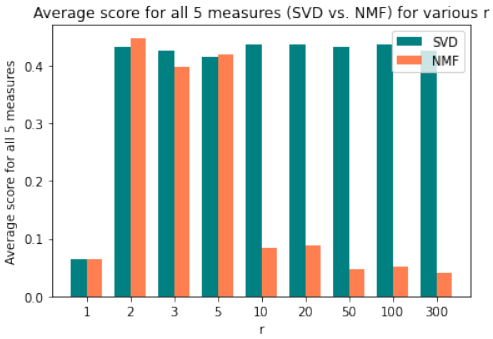
(c) V-Measure



(d) ARI



(e) AMI



(f) Average Scores for 5 measures

Figure 3: Metrics for SVD and NMF

6 Question 6

We observe from the results obtained that the clustering measures are non monotonic as r increases and moreover, all of the measures show a similar pattern. As r increases, the values of the five measures first increases and then drops. For low dimensional r value, the values are close to zero. As r increases, we have more information and hence the values of the clustering measures increases as more information helps with the clustering accuracy but as r becomes too large, we notice a drop in the measures. This is because it becomes a high dimensional matrix. K-means algorithm uses Euclidean distance to measure the distance between data points and we know that euclidean distance is not a good metric for a high dimensional space as the ratio between the nearest and farthest points approaches 1, i.e. the points essentially become uniformly distant from each other and hence the distinction of distance between data points becomes meaningless.

7 Question 7

K-Means on SVD with r = 20:

- Homogeneity: 0.421
- Completeness: 0.460
- V-measure: 0.440
- Adjusted Rand-Index: 0.427
- Adjusted Mutual Information Score: 0.440

K-Means on NMF with r = 2:

- Homogeneity: 0.431
- Completeness: 0.462
- V-measure: 0.446
- Adjusted Rand-Index: 0.455
- Adjusted Mutual Information Score: 0.446

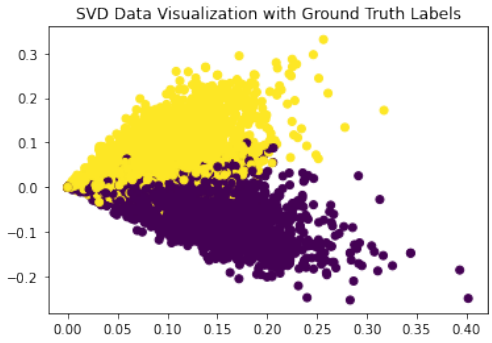
K-Means on sparse data:

- Homogeneity = 0.427
- Completeness = 0.464
- V-measure = 0.445
- Adjusted Rand-Index = 0.436
- Adjusted Mutual Information Score = 0.445

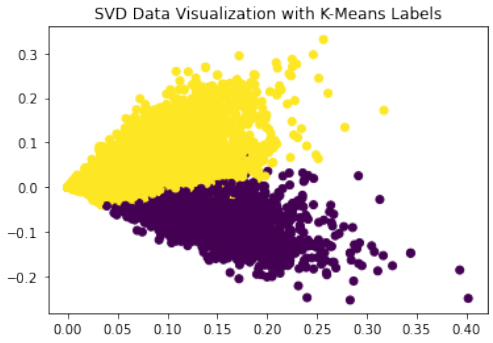
We can note that NMF with a best value of r=2 provided marginally better results when compared to SVD with r=20 and with the measures obtained for K means clustering on just sparse TF-IDF representations.

8 Question 8

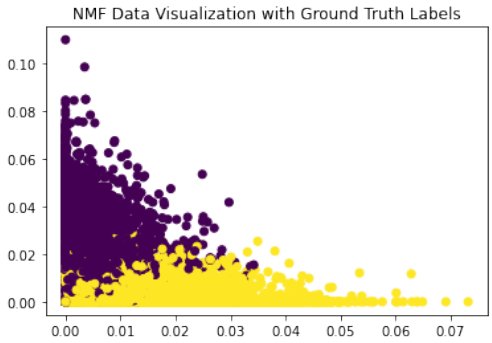
The clusters are visualized for SVD of r = 20 and NMF for r = 2 components. The reduced data are again projected onto a 2D plane using SVD to obtain the following clusters.



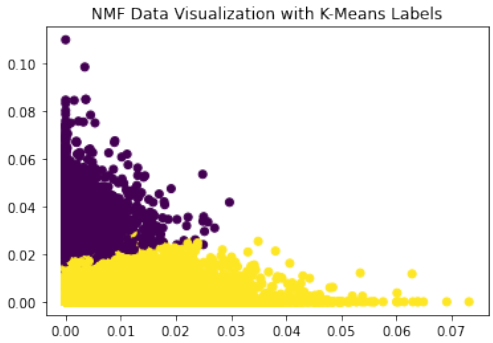
(a) SVD



(b) SVD with k-means



(c) NMF



(d) NMF with k-means

Figure 4: Cluster Visualization

9 Question 9

From the above plots in Figure 4 we may note the following:

- K means clustering algorithm assumes that the clusters are isotropic in nature. While SVD and NMF both do not show a spherical blob for the plot, NMF also shows uneven sizes of blobs for both the cluster classes.
- SVD and NMF both show unequal variances for the two cluster classes which is more eminent in NMF with one cluster more compactly packed than the other.
- We can also visualize that there is an overlap of clusters in both SVD and NMF, signalling a small euclidean distance between the centroids of the clusters which explains the low performance/ smaller values of the measures.

10 Question 10

We performed K-Means clustering for the entire 20 classes and recorded metric values for both SVD and NMF. The best r value for SVD was found to be r = 20 and the best r value for NMF was found to be r = 10. The contingency matrix for the SVD is plotted below:

20 Classes with SVD									
Metrics	r=1	r=2	r=3	r=5	r=10	**r=20**	r=50	r=100	r=300
Homogeneity	0.0242	0.212	0.247	0.3202	0.3249	**0.3330**	0.2838	0.3307	0.2976
Completeness	0.0267	0.2246	0.2654	0.3483	0.3543	**0.3756**	0.3557	0.3999	0.3721
V-Measure	0.0254	0.2183	0.2559	0.3336	0.3389	**0.3530**	0.3157	0.3621	0.3307
ARI	0.0053	0.0657	0.0831	0.1259	0.1228	**0.1201**	0.0793	0.1138	0.0879
AMI	0.0221	0.2157	0.2534	0.3314	0.3367	**0.3132**	0.3508	0.3598	0.3283

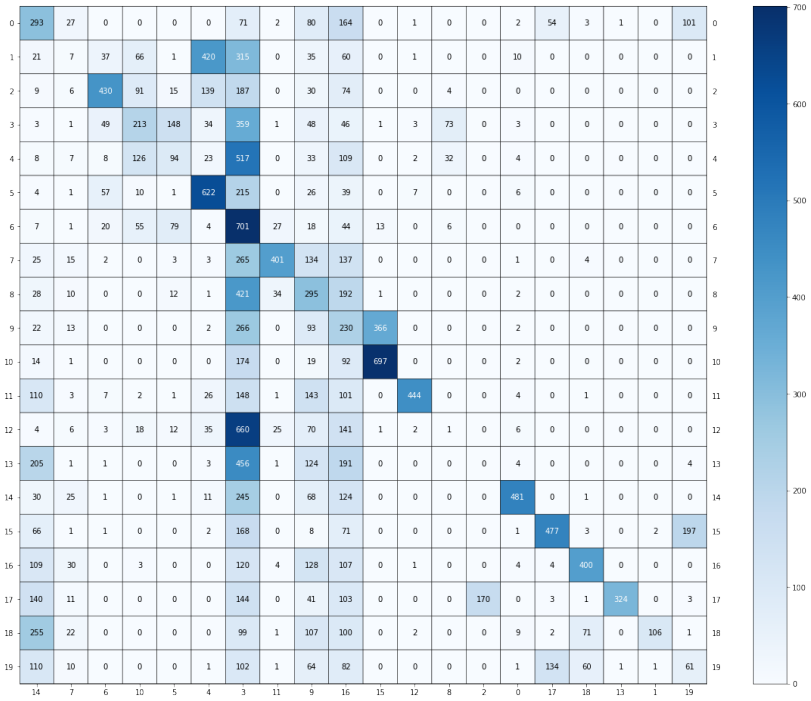


Figure 5: Contingency Matrix for SVD with 20 classes

20 Classes with NMF								
Metrics	r=1	r=2	r=3	r=5	**r=10**	r=20	r=50	r=100
Homogeneity	0.0242	0.1922	0.2176	0.2676	**0.2879**	0.2745	0.2331	0.1485
Completeness	0.0267	0.2050	0.2562	0.2894	**0.3274**	0.3526	0.3397	0.1910
V-Measure	0.0254	0.1984	0.2353	0.2781	**0.3064**	0.3086	0.2765	0.1671
ARI	0.0053	0.0582	0.0672	0.0874	**0.0995**	0.0717	0.0532	0.0331
AMI	0.0221	0.1957	0.2326	0.2757	**0.3039**	0.3061	0.2736	0.1640

11 Question 11

We observe the best value of r for Euclidean UMAP (according to avg. metric): 2. The results for various r's are tabulated as well.

- Homogeneity: 0.009524537446715267
- Completeness: 0.009704272289063731
- V-measure: 0.009613564862345788

- Adjusted Rand-Index: 0.001865160869070743
- Adjusted Mutual Information Score: 0.0063842821082159724

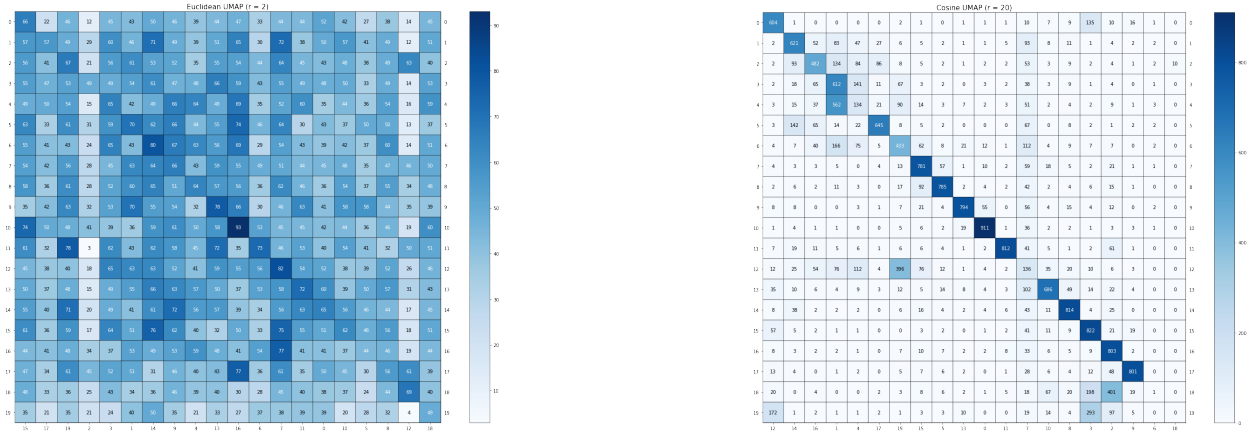
UMAP (Euclidean)								
Metrics	r=1	**r=2**	r=3	r=5	r=10	r=20	r=50	r=100
Homogeneity	0.0094	**0.0095**	0.0075	0.0076	0.0077	0.0082	0.0075	0.0076
Completeness	0.0097	**0.0097**	0.0075	0.0079	0.0079	0.0084	0.0076	0.0079
V-Measure	0.0095	**0.0096**	0.0075	0.0077	0.0078	0.0083	0.0076	0.0077
ARI	0.0010	**0.0018**	0.0013	0.0010	0.0010	0.0012	0.0012	0.0010
AMI	0.0063	**0.0063**	0.0043	0.0045	0.0045	0.0050	0.0043	0.0045

We observe the best value of r for Cosine UMAP (according to avg. metric): 20. The results for various r values are tabulated below:

- Homogeneity: 0.5794601012917987
- Completeness: 0.6014533980318418
- V-measure: 0.5902519484203353
- Adjusted Rand-Index: 0.4646765141350141
- Adjusted Mutual Information Score: 0.5888922998184412

UMAP (Cosine)								
Metrics	r=1	r=2	r=3	r=5	r=10	**r=20**	r=50	r=100
Homogeneity	0.3957	0.5575	0.5637	0.5709	0.5674	**0.5794**	0.5692	0.5795
Completeness	0.4128	0.5741	0.5857	0.5804	0.6026	**0.6014**	0.5946	0.5937
V-Measure	0.4041	0.5656	0.5745	0.5756	0.5845	**0.5902**	0.5816	0.5866
ARI	0.2602	0.4299	0.4485	0.4510	0.4457	**0.4646**	0.4454	0.4630
AMI	0.4021	0.5642	0.5731	0.5742	0.5831	**0.5888**	0.5803	0.5852

12 Question 12



(a) Euclidean UMAP

(b) Cosine UMAP

Figure 6: Contingency Matrices for UMAP

From the figure, we can observe that the Eucledian's UMAP contingency matrix is non-diagonal and stochastic. But in the case of Cosine's UMAP contingency matrix, we observe that some of the ground truth clusters are being lumped to a single cluster. We do observe specific clusters such as *sci.crypt, comp.windows.x, sci.electronics, talk.religion.misc, soc.religion.christian* are under-performing. This could be due to the use of common words between these classes. Also, we could infer that the lack of headers and footers that provide semantic information about the topic of newsgroup posts are missing.

13 Question 13

The best results for each of the feature-reduction techniques are listed below:

K-Means with SVD for r = 20:

- Homogeneity: 0.421
- Completeness: 0.460
- V-measure: 0.440
- Adjusted Rand-Index: 0.427
- Adjusted Mutual Information Score: 0.440

K-Means with NMF for r = 2:

- Homogeneity: 0.431
- Completeness: 0.462
- V-measure: 0.446
- Adjusted Rand-Index: 0.455
- Adjusted Mutual Information Score: 0.446

K-Means with UMAP (Cosine) for r = 20:

- Homogeneity: 0.579
- Completeness: 0.601
- V-measure: 0.591
- Adjusted Rand-Index: 0.464
- Adjusted Mutual Information Score: 0.588

K-Means on sparse data:

- Homogeneity: 0.328
- Completeness: 0.376
- V-measure: 0.350
- Adjusted Rand-Index: 0.116
- Adjusted Mutual Information Score: 0.348

Based on the above metrics calculated for different feature reduction techniques applied on K-Means clustering model, we could infer that UMAP reduction technique performs the best. The reason could be attributed to the reason that UMAP constructs a graphical representation of the data in the high dimension and attempts to optimize a low dimensional graphical embedding to be geometrically similar to the high dimensional graph. Hence this non-linear technique performs better than the linear SVD and NMF techniques for this case. We could further say that the UMAP for cosine distance performs better than euclidean distance. This is expected because cosine similarity is not affected by the magnitude of vectors, i.e the length of the documents does not affect the distance metric, but rather it associates clusters based on angle between sample points. In addition, in higher dimensions, the increase in the features causes the data to be sparse in each dimension, causing the euclidean distances to converge to a constant value and since all feature points becomes equidistant, UMAP euclidean would not be able to cluster accordingly.

14 Question 14

We observe the following metric values for Agglomerative Clustering:

FOR WARD:

- Homogeneity: 0.538
- Completeness: 0.580
- V-measure: 0.558
- Adjusted Rand-Index: 0.408
- Adjusted Mutual Information Score: 0.557

FOR SINGLE:

- Homogeneity: 0.017
- Completeness: 0.387
- V-measure: 0.033

- Adjusted Rand-Index: 0.001
- Adjusted Mutual Information Score: 0.029

From the above results we can clearly see that the ward linkage criteria performed much better than the single linkage criteria. This may be due to the fact that the single linkage criteria uses the nearest neighbor approach while ward linkage criteria uses minimum increase in sum of squares. Single linkage is thus fast but does not work well with outliers. It forms a long chain-like cluster while ward forms spherical blobs. Thus, in chain like clusters it is easy to cause errors while clustering the data points.

15 Question 15

A pipeline was constructed to obtain the best set of hyper-parameters epsilon and minimum number of samples. The average of 5 metrics were taken for choosing the best performing hyper-parameters.

For DBSCAN, we obtained the best results when epsilon and minimum number of samples were 0.9 and 500 respectively. The five metrics for the same model are given below :

Metrics for epsilon = 0.9, min_samples = 500 :

- Homogeneity: 0.4505706176253225
- Completeness: 0.6696252561297367
- V-measure: 0.5386798368047954
- Adjusted Rand-Index: 0.2525887602515154
- Adjusted Mutual Information Score: 0.53783880676293891

For HDBSCAN, we obtained the best results when epsilon and minimum number of samples were 0.3 and 5 respectively. The five metrics for the same model are given below :

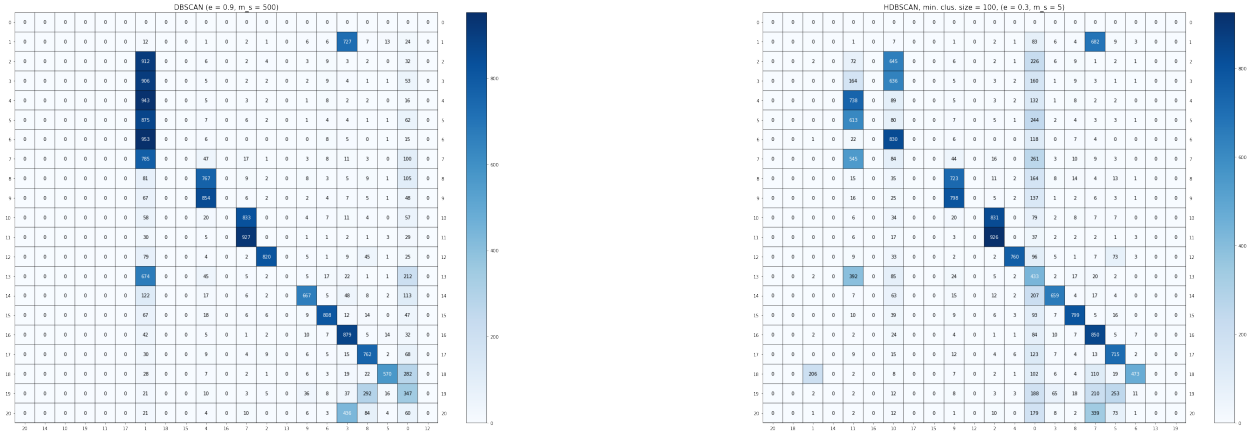
Metrics for epsilon = 0.3, min_samples = 5 :

- Homogeneity: 0.46070057414336
- Completeness: 0.5997040711586736
- V-measure: 0.5210916627401566
- Adjusted Rand-Index: 0.30250027340533325
- Adjusted Mutual Information Score: 0.5200804256766023

16 Question 16

The contingency matrices are plotted for the best performing models obtained from Question 15. From the matrices, we can infer that the DBSCAN forms 10 clusters including the -1 cluster and HDBSCAN forms 11 clusters including the -1 cluster. We can further confirm this by using the set operation on the class labels obtained from DBSCAN and HDBSCAN.

We can see a lesser number of clusters than expected as both DBSCAN and HDBSCAN do not take the number of clusters as inputs, but instead try to determine the clusters based on the density of the feature vectors. The "-1" signifies all the points that were classified as outliers. We can also infer that the sensitivity to the density of determining clusters is influenced by the hyper-parameters epsilon and min_samples.



(a) DBSCAN

(b) HDBSCAN

Figure 7: Contingency Matrices for DBSCAN and HDBSCAN

17 Question 17

The given table was implemented for identifying the best model along with its hyper parameters. The top 4 performing models are summarized below:

Aggolmerative Clustering (Ward) applied on UMAP (cosine) n = 20 components, r = 20 :

- Homogeneity: 0.553
- Completeness: 0.582
- V-measure: 0.567
- Adjusted Rand-Index: 0.412
- Adjusted Mutual Information Score: 0.566

Aggolmerative Clustering (Ward) applied on UMAP (cosine) n = 20 components, r = 200 :

- Homogeneity: 0.551
- Completeness: 0.578
- V-measure: 0.564
- Adjusted Rand-Index: 0.429
- Adjusted Mutual Information Score: 0.563

K-Means Clustering applied on UMAP (cosine) n = 20 components, r = 20 :

- Homogeneity: 0.548
- Completeness: 0.580
- V-measure: 0.563
- Adjusted Rand-Index: 0.425

- Adjusted Mutual Information Score: 0.562

K-Means Clustering applied on UMAP (cosine) n = 20 components, r = 200 :

- Homogeneity: 0.584
- Completeness: 0.603
- V-measure: 0.594
- Adjusted Rand-Index: 0.473
- Adjusted Mutual Information Score: 0.592

18 Question 18

The following techniques were explored :

- Apply Lemmatization on the data and extract POS tags and then apply UMAP feature reduction. The reduced feature matrix could then be applied on KMeans clustering could aid marginally better results.
- Include dataset without removing headers and footers. We observed that headers and footers might contain semantic information that could assist in classification. Hence, extracting tf-idf matrix for the dataset without removing headers and footers and then using the UMAP reduction, followed by Aggolomerative/K-Means clustering could give marginal performance boost.
- We explored the LDA feature reduction technique that could preserve class information. The reduced feature matrix then could be applied on K-Means or Aggolomerative clustering.
- Employ Transformer base architecture such as BERT that could extract complex features.

19 Question 19

Although the VGG network trained on the Image net dataset would have learnt to classify different target labels, we could still use the weights learnt by the model as our initial weights and use that in training our dataset. This technique of using weights of pre-trained models in custom dataset training is called Transfer learning. Rather than initializing the weights to random which is the case during neural network training, we use the weights from the pre-trained model as our initial weights. By doing so, the model does not have to learn from scratch as it can directly inherit the simpler features by using the pre-trained weights and thus the model can now concentrate to learn the complex features present in the custom dataset. The transfer learning technique offers two primary advantages: drastically reduces the training time as the model need not learn from scratch. This technique is useful when our custom dataset is not large enough to be trained from scratch, we could use the pre-trained weights and offer initial learnings to the model.

20 Question 20

The helper code helps us implement the following :

- It begins with loading the flower dataset and applying various transformation techniques such as resizing, center cropping and normalizing to ensure dataset homogeneity.
- The dataset is split into 59 batches with 64 samples each.

- The class **feature extractor** then calls the pre-trained VGG model. The VGG architecture which consists of 2D convolutions, ReLU activations and Max pooling. The 2D convolutions would aid us in down-sampling the image and help us extract complex images as we progress through the network followed my ReLU activation and average/max pooling. This is finally followed by a fully-connected layer and softmax layer that provides the probability estimates of the target labels.
- The training is done for 58 epochs and we obtain the extracted features of 4096 features per sample along with the predicted class labels.

21 Question 21

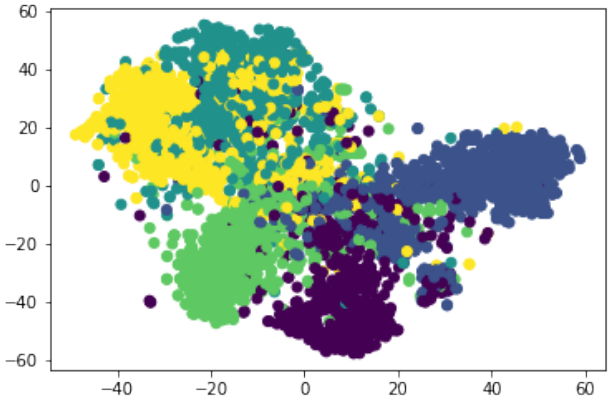
We observe that the images in the dataset are of varied sizes and hence there is no definite/standard size for the original images in the dataset. Hence a transform is applied to resize the images to the same size (224 x 224). The VGG network extracts 4096 features per image sample.

22 Question 22

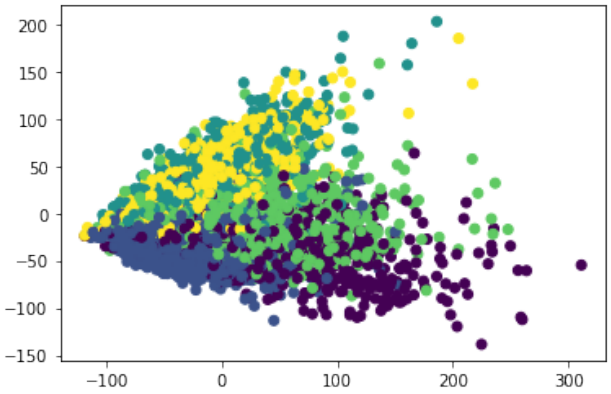
The density or sparsity of a given matrix can be determined by counting the number of non zero entities in them. For the TF-IDF features from the textual data, we observed the TF-IDF is a sparse matrix of 363490 non zero elements with the total size of the matrix being 7882 x 18469 (calculated using scipy.sparse.find). But whereas, from the features extracted from the VGG network, we observed that all entities from the feature space are non zero (calculated using np.count_nonzero) and hence is a denser than the sparse TF-IDF matrix obtained from text features.

23 Question 23

As seen from the figure below, we can infer that t-SNE preserves class information better than PCA for this case since t-SNE adopts a probabilistic approach and tries to group points as close as possible based on probability that two close points come from the same probability distribution. But whereas, in case of the PCA, it tries separate points as far as possible based on highest variance. PCA also being a linear dimensionality reduction technique while t-SNE is a non linear dimensionality reduction technique. t-SNE is a much more popular choice for dimensionality reduction than PCA as it is more robust to outliers and tends to preserve the local structure of the data.



(a) t-SNE



(b) PCA

Figure 8: Cluster Visualization for t-SNE & PCA

24 Question 24

The Adjusted Random Index for various Dimensionality Reduction techniques and clustering models are tabulated below. We can infer that the **UMAP reduced feature with K-Means clustering** performs the best with a rand score of **0.466** followed by **UMAP with Agglomerative clustering** with a rand score of **0.3**.

CLUSTERING			
Dim Red	k-means	Agglomerative Clustering	HDBSCAN (min_cluster_size, min_samples)
SVD	0.192	0.216	0.010 (15,5)
UMAP	**0.466**	**0.300**	0.095 (5,5)
Autoencoder	0.225	0.181	0.013 (5,5)

For HDBSCAN, the following grid values for grid of values for min_cluster_size and min_samples were considered :

- min_cluster_size = [5, 15, 30, 60, 100, 200, 500, 1000, 3000]
- min_samples = [5, 15, 30, 60, 100, 200, 500]

25 Question 25

The MLP classifier was applied on the original VGG features to obtain the following:

- Test Accuracy on VGG features without any dimensionality reduction - 90.599 %
- Test Accuracy on VGG features with UMAP (n = 50) feature reduction - 87.602 %
- Test Accuracy on VGG features with Autoencoder (n = 50) feature reduction - 87.329 %

Based on the test accuracy results, we can observe that there is a drop in the accuracy by 3% of the MLP classifier when using the reduced-dimension features. One could not comment on its significance as this could be attributed to the trade-off between feature complexity and accuracy.

Further, we can comment that the clusters in clustering models are formed based on distances, but whereas in the case of the MLP classifier, the model learns the complex features and predicts the classes based on the learning. Thus in general, MLP classifier would in general perform better than Clustering methods for complex data.