219 - Large-Scale Data Mining: Models and Algorithms

Report on

Project 2: Data Representations and Clustering

Authors

Jayanth Shreekumar (805486993)

Narayan Gopinathan (605624014)

Yuheng He (505686149)

Winter 2022

Part 1 - Clustering on Text Data

Introduction

Clustering algorithms are unsupervised techniques where we receive unlabelled data points and are tasked to find groups or "clusters" of data that have similar features. Clustering algorithms are often applied to problems such as pattern recognition and data analysis as they provide useful insights into the data that was not prominent at first glance.

Question 1.

In this part, we perform clustering on the text-based 20 newsgroups dataset . Since the dataset is textual, the first step would be to convert in into a matrix form that can be fed into the clustering methods. The following steps were followed for this conversion:

1. Fetch the dataset while setting the 'categories' parameter to extract the required classes and 'remove' parameter to remove headers and footers as required.
2. Create labels for the two classes.
3. Feed the dataset into a tf-idf vectorizer which performs both CountVectorizer() as well as TfidfTransformer(). Set the parameter 'min_df' to 3 and remove the english stopwords are specified.

The obtained TF-IDF matrix has the shape **(7882, 23522)**.

Question 2.
After obtaining the sparse matrix representation of our dataset, the K-Means clustering algorithm was used with the following parameters:

- The number of clustering centroids, k = 2.
- random_state = 0.
- max_iter = 1000.
- n_init = 500.

The contingency table is the equivalent of the confusion matrix for clustering algorithms, where the rows show the number of datapoints in the ground truth and columns show the number of datapoints in the clusters created by our clustering algorithm. Since there is no guarantee that the algorithm finds the clusters in the correct order(even though the clusters themselves are correct, the algorithm might just name them differently from the ground truth), we might not get the ideal diagonal matrix. Also, the contingency matrix will be square due to the constraint of sklearn, but some columns can have only zeros in it. Therefore, will the shape of the table is square, we see that the number of clusters detected is not equal to the number of unique ground truth labels. The contingency matrix that we obtained is shown below:

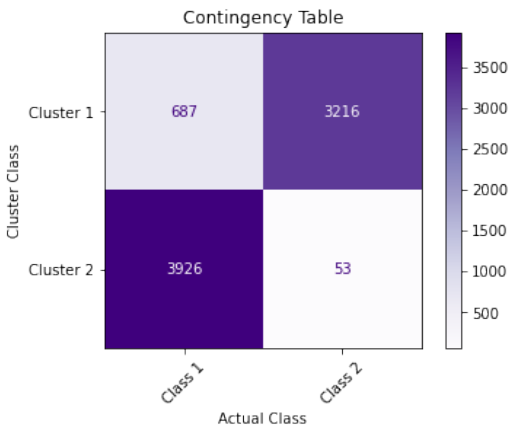


Figure 1: The contingency matrix

Question 3.
The five clustering measures explained in the introduction are

1. **Homogeneity** is a measure of how homogeneous or pure the clusters are. If each cluster contains data from only a single class, then the homogeneity is satisfied.
2. **Completeness** is a measure of whether a class of data is completely assigned to the same cluster. If all of the data in a dataset of a given class fall into the same cluster, completeness is satisfied.
3. **V-measure** is the harmonic average of the completeness and homogeneity.
4. **Adjusted Rand Index** reports the similarity between the clustering labels and the ground truth labels. It counts all the pairs of data points that fall either in the same cluster and the same class or in different clusters and different classes.
5. **Adjusted mutual information score** measures the mutual information between the cluster label distribution and the ground truth label distortions.

The metrics obtained for the clustering experiment in question 2 are given below:

Values for k-means clustering	
Homogeneity Score	0.595026
Completeness Score	0.607796
V-Measure Score	0.601343
Adjusted Rand Index:	0.659675
Adjusted Mutual Information	0.601306
Average	0.613029

Table 1: Scores for K-means clustering

Question 4.

Sometimes, it is not a good idea to utilize high dimensional TF-IDF matrices directly for clustering as they are memory intensive, and also, just do not perform well in high dimensional space where the definition of "distance" is much more complex and tends to be very similar.

Dimensionality reduction methods such as truncated SVD and non-negative matrix factorization can be used to alleviate these problems and are utilized frequently in textual experiments where the representational matrices are large and sparse. However, when applying these methods, we need to set the hyperparater to control the number of components that we would want to feed into the clustering algorithm. Choosing this can be difficult and therefore a visual representation depicting what ratio of the variance in the original dataset is retained. This plot, called the cumulative explained variance ratio vs the number of components of the output, is shown below:

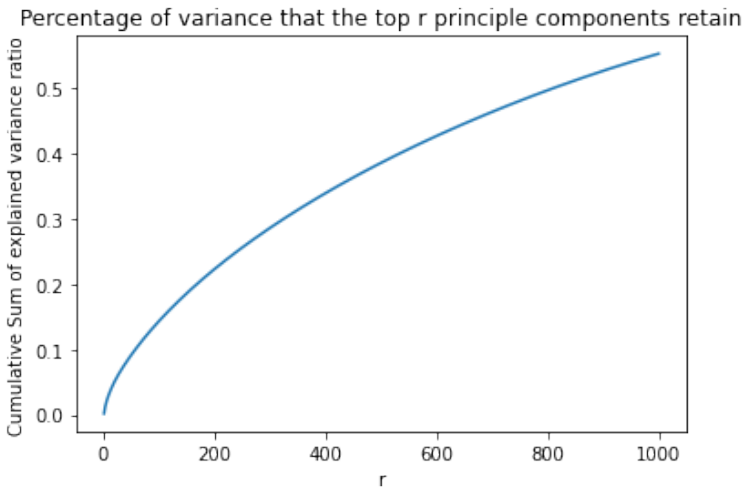
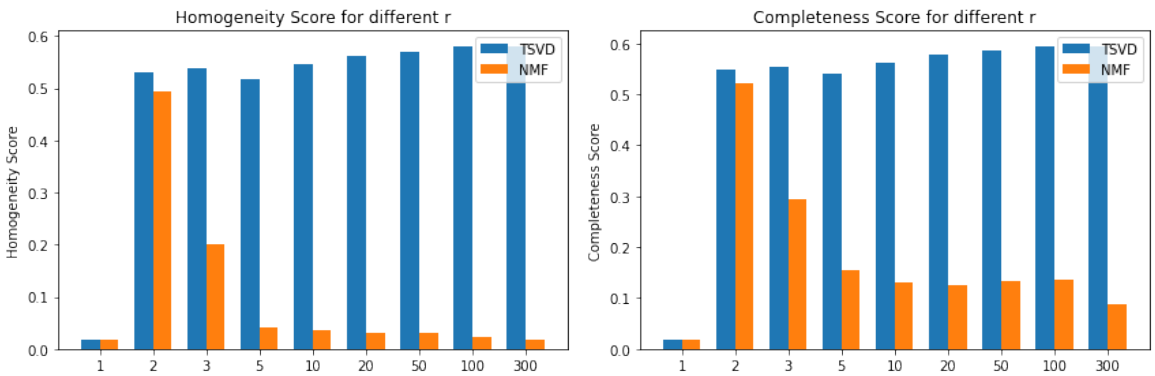


Figure 2: The percentage of variance that the top r principal components retain, for values of r from 1 to 1000.

It is straightforward to understand that the cumulative sum of the explained variance ratio increases as we increase the number of dimensions that are in the output.

Question 5.

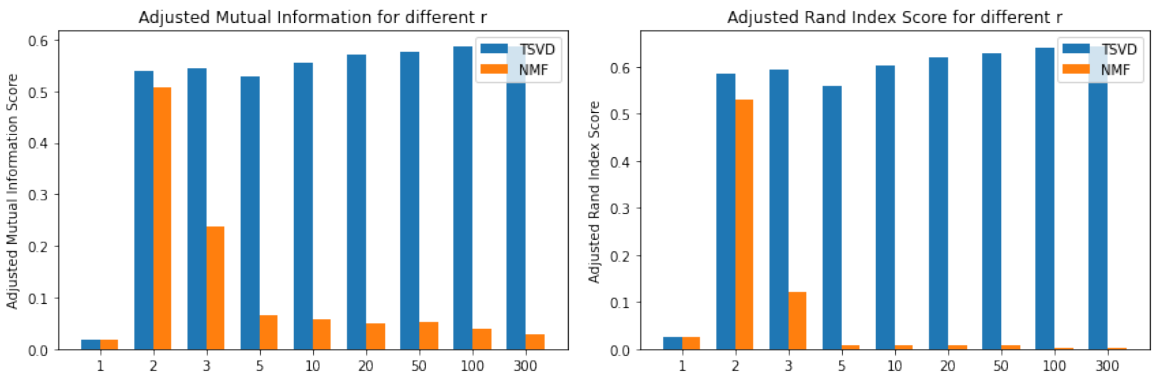
The other decision left is the choice of the dimensionality reduction method. Clustering methods are largely influenced by our dataset and so no single clustering method works best. In turn, clustering methods themselves are influenced by dimensionality reduction methods and therefore, choosing the right dimensionality technique is crucial. In this question, we are asked to change the number of output components of two different dimensionality reduction techniques and report the 5 measure scores after applying the K-Means clustering algorithm on each of the outputs. The results of our experiments are displayed below:



(a) The homogeneity scores for NMF and SVD for different r-values

(b) The completeness scores for NMF and SVD for different r-values..

Figure 3: Homogeneity and completeness scores for NMF and SVD for different r-values



(a) The adjusted mutual scores for NMF and SVD for different r-values

(b) The adjusted Rand scores for NMF and SVD for different r-values..

Figure 4: Adjusted mutual and Rand scores for NMF and SVD for different r-values

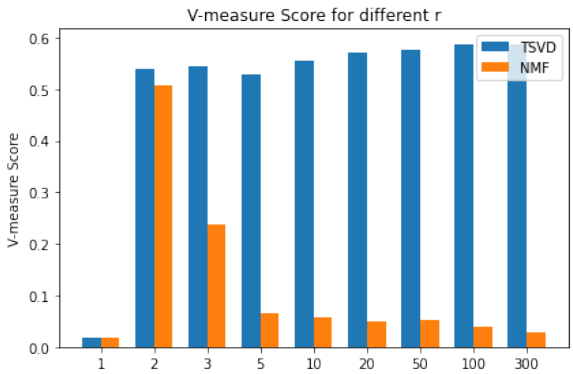


Figure 5: The V-measure scores for NMF and SVD for different r-values.

To figure out the best output dimension that would perform well in clustering, we took the average of all 5 measures, the results of which are shown below:

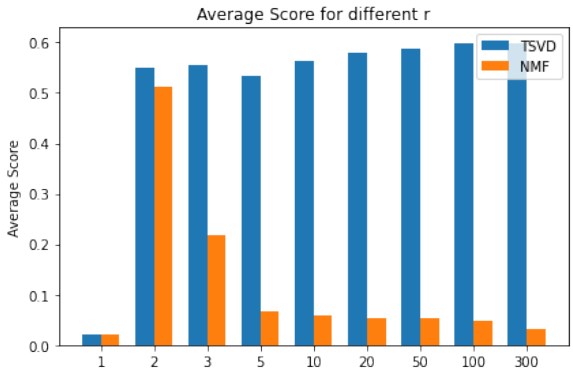


Figure 6: The average scores for NMF and SVD for different r-values.

Clearly, from the plot, the best values for r are 300 for SVD and 2 for NMF. However, the critical observation here is that as r, the output dimension increases, the computational cost of performing dimensionality reduction also increases. Therefore, we need to choose an optimal r value such that it is not so large that it is computationally expensive to compute, but contains enough information of the classes to perform a good job during clustering. **We see that this problem does not arise for NMF and the r value of 2 meets our requirements**. However, the average score at the r of 50 for the Truncated SVD, which is 0.587, is negligibly lesser when compared to the average score at r value 300, which is 0.599. **Therefore, the optimal choice of r for SVD would be 50**.

Question 6.

Our expectation is that any algorithm (in this case, clustering algorithms) will perform better when we increase the amount of information it is given and so, we expect monotonic behaviour. However, in contrast, we observe from the plots in question 5 that in general, as r increases, the clustering measures are non-monotonic. There are two reasons for this:

1. The definition of "distance" is much more complex in higher dimensions. The Euclidean distance that K-Means utilizes saturates at a certain value so that the distance between any pairs of samples are very similar.

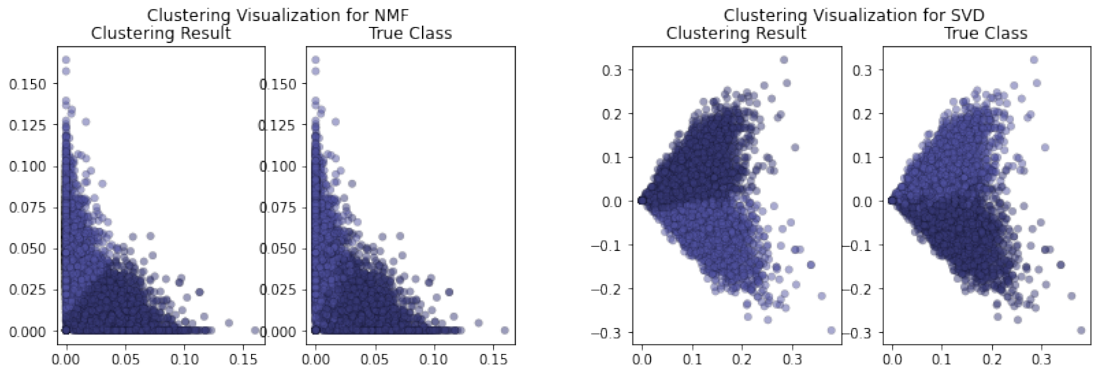
2. With respect to NMF, the major drawback when compared to SVD is that NMF does not allow negative values in its decomposition, due to which it is very inflexible in higher dimensions. In fact, NMF is computationally expensive in higher dimensions and also does not converge well.

Question 7.

No, these measures are not on average better than the ones computed in question 3. The average of scores computed in question 3 is .613029, which is higher than the average of the scores for NMF or SVD at any value of r. This is to be expected as using the original dataset will almost always perform better than using dimensionally-reduced data because it contains more information. However, the main purpose of dimensionality reduction methods is to drastically reduce computational cost while simultaneously capturing most of the information useful for clustering the data.

Question 8.

The visualised clustered data is shown below:



(a) Clustering visualization with NMF

(b) Clustering visualization with SVD

Figure 7: Clustering visualizations for NMF and SVD

Question 9.

There are mainly two observations to be made:

1. K-Means clustering inherently assumes that the clusters are normally distributed. However, this might not be the case, and therefore K-Means will not do a good job on any dataset.

2. In the figures in question 8, we see that neither SVD nor NMF produces normal distributions.

Thus, we conclude from these two observations that while K-Means clustering can be applied on any dataset, it is not ideal in this case.

Question 10.

In this question, we perform clustering on the text-based 20 newsgroups dataset. Since the dataset is textual, the first step would be to convert in into a matrix form that can be fed into the clustering methods. The following steps were followed for this conversion:

1. Fetch the full dataset while setting the 'remove' parameter to remove headers and footers and create labels for the 20 classes.
2. Feed the dataset into a tf-idf vectorizer which performs both CountVectorizer() as well as TfidfTransformer(). Set the parameter 'min_df' to 3 and remove english stopwords.
3. Perform dimensionality reduction of the dataset using SVD with n_components = 20.
4. Perform k-means clustering with number of clusters k=20.

The contingency table and the 5 measures for SVD and K-Means are shown below. Please refer the .ipynb file for results on using NMF with K-Means.

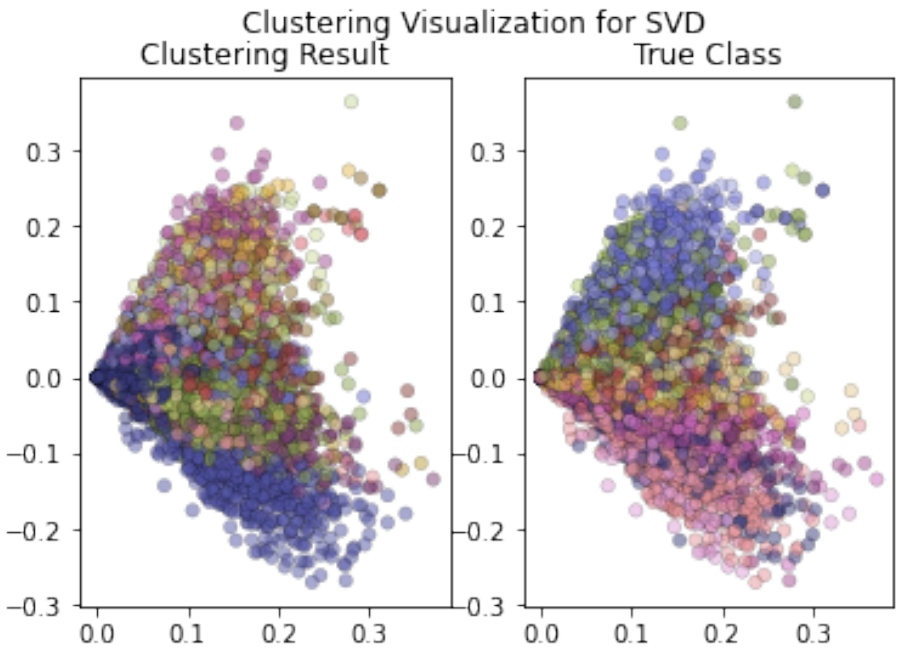


Figure 8: Clustering visualization created using SVD and K-Means for 20 clusters.

Homogeneity Score	0.323340
Completeness Score	0.366366
V-Measure Score	0.343511
Adjusted Rand Index:	0.108943
Adjusted Mutual Information	0.341251
Average	0.296682

Table 2: Clustering measures obtained using SVD and K-Means for 20 clusters.

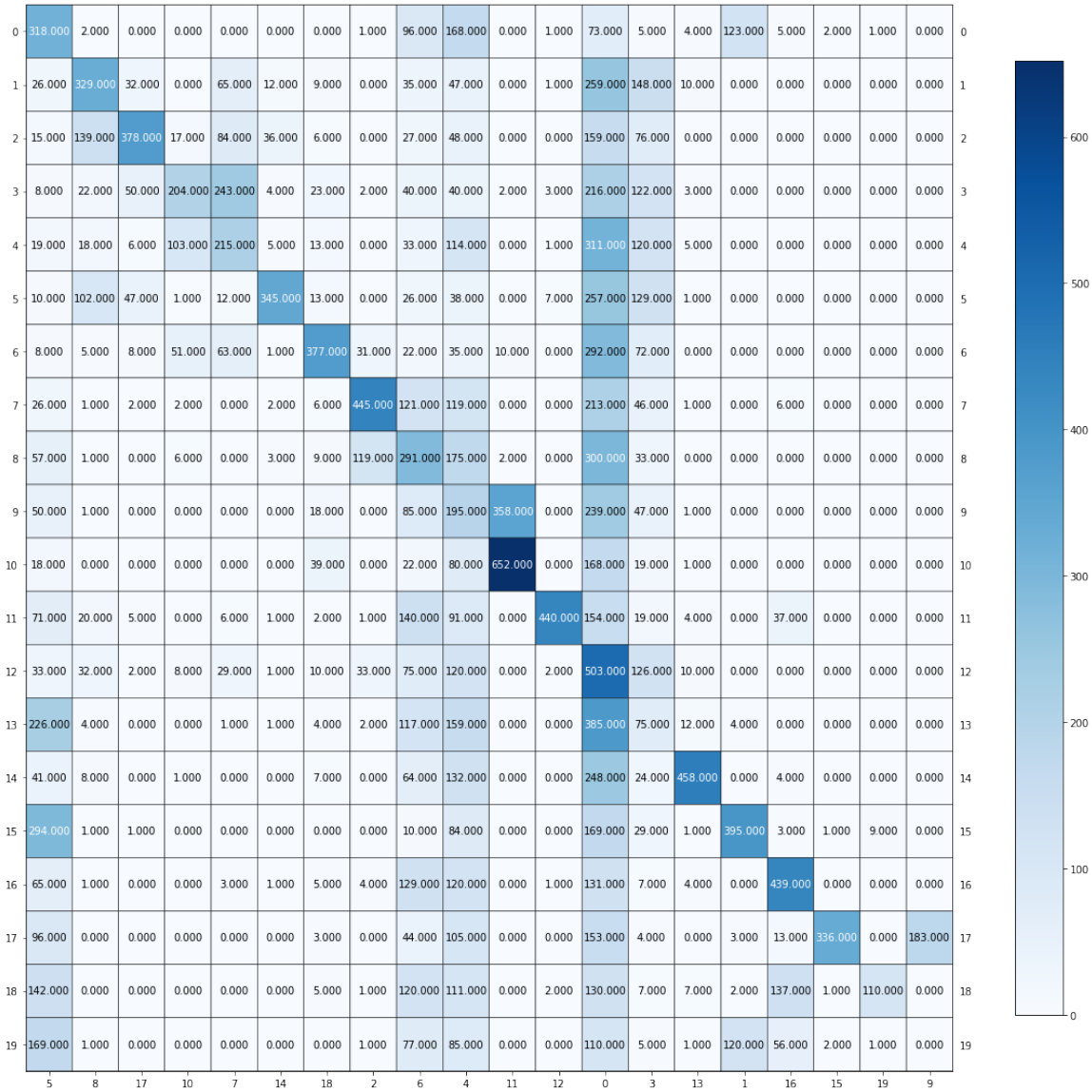


Figure 9: Contingency Table created using SVD and K-Means for 20 clusters.

Question 11.

Uniform Manifold Approximation and Projection or UMAP is a graph based dimensionality reduction technique that uses ideas from topological data analysis.

In this question, we perform dimensionality reduction using UMAP for varying n_components and two different metrics "euclidean" and "cosine". **The best n_components for cosine was found to be 5 while the best n_components for euclidean was found to be 10.**

We find that the cosine metric performs much better than the euclidean one and we believe that this is because the euclidean metric is constrained by its idea of distance, while the cosine metric uses other measures like angles.

Homogeneity Score	0.006865
Completeness Score	0.007127
V-Measure Score	0.006934
Adjusted Rand Index:	0.000900
Adjusted Mutual Information	0.003941

Table 3: Clustering metrics for Euclidean method of k-means clustering with 10 components

Homogeneity Score	0.579646
Completeness Score	0.590490
V-Measure Score	0.595018
Adjusted Rand Index:	0.460154
Adjusted Mutual Information	0.583665

Table 4: Clustering metrics for Cosine method of k-means clustering with 5 components

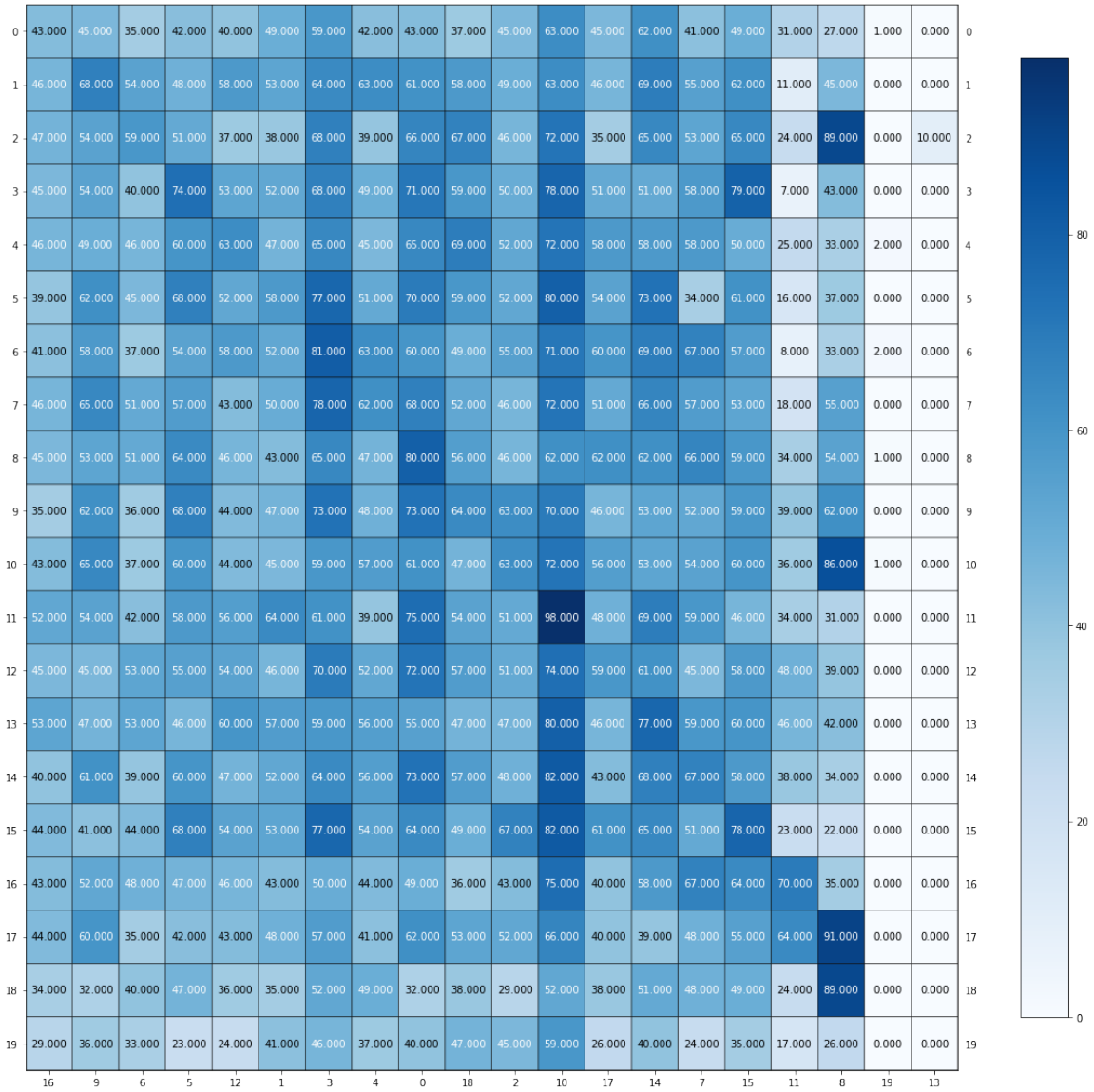


Figure 10: Contingency table of Euclidean UMAP dimensionality reduction

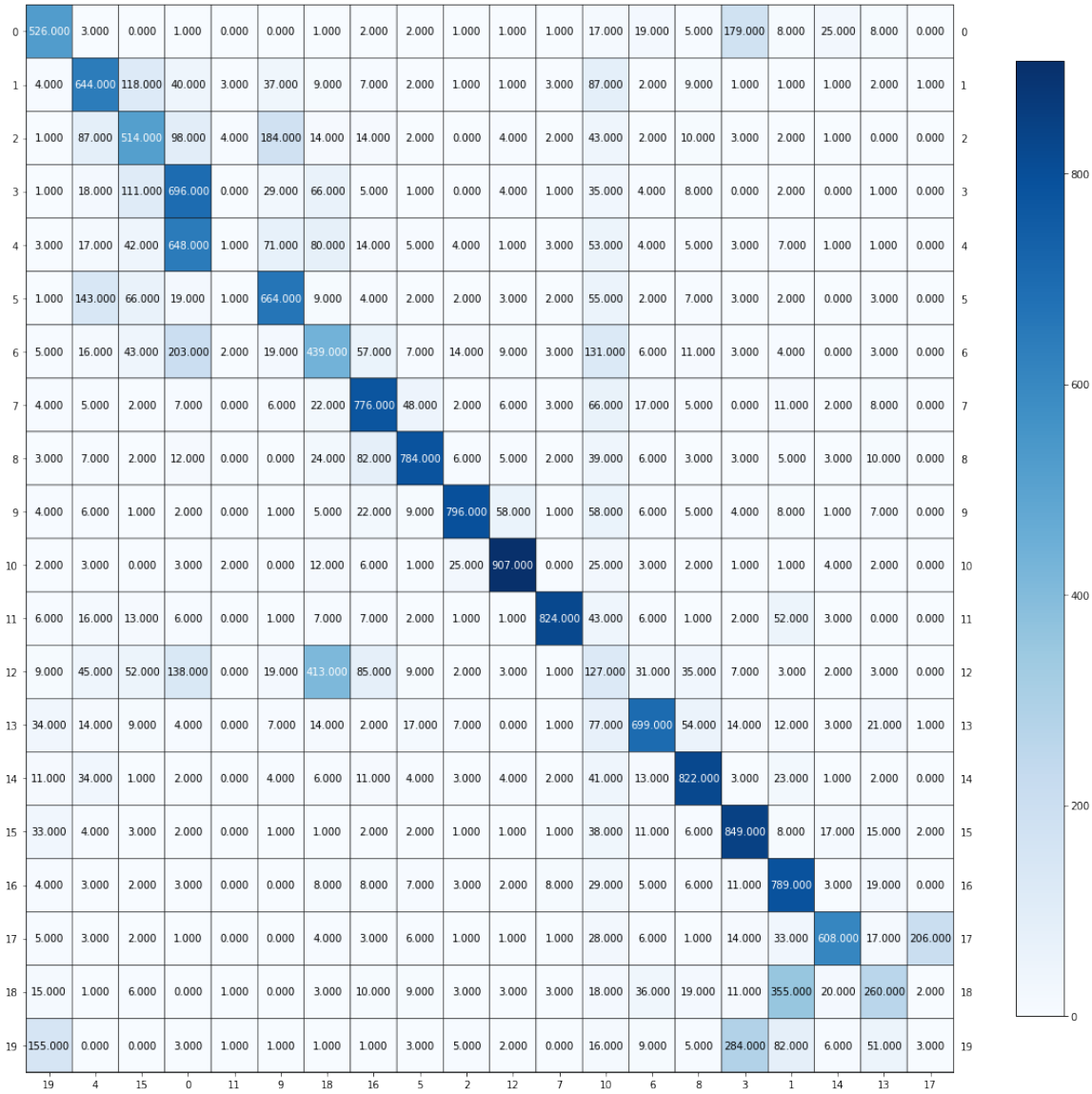


Figure 11: Contingency table of Cosine UMAP dimensionality reduction

Question 12.

The clustering evaluation metrics are very low for Euclidean clustering. For that reason Euclidean clustering is not a good method. The cosine clustering method is better. Its clustering evaluation metrics are higher and the contingency table gives high values mostly along the main diagonal as it should. However, some of the ground truths seem to be recognized as a single cluster. One explanation for this is that these classes might be related to each other and therefore, the clustering algorithm might confuse between them. In addition, the constraint that K-Means looks for Gaussian distributions might also be having an adverse effect.

Question 13.

Most of the clustering methods tried have very low clustering metric scores. If the clustering scores for a method receives scores above .5 then that puts it above the rest.
The NMF clustering with 2 components and 2 clusters is one of the best clustering methods, as it has metrics close to .59. The SVD clustering method with 300 components is another of the best clustering methods, as it also has metrics close to .59. The k-means clustering with X-train and X-labels had the highest metrics with all of them except homogeneity being at .60, and with homogeneity at .595.

SVD clustering, as long as $r > 1$, returns evaluation metrics between .5 and .6 . This means that SVD is a good clustering method and can return good results.

Conversely, the NMF method of clustering, when r is greater than or equal to 5, returned clustering evaluation metrics below 0.1. These are among the worst methods of k-means clustering.

Question 14.

Agglomerative clustering is a form of bottom-up hierarchical clustering method used to group clusters based on similarity. It works by merging pairs of clusters until a single large cluster called a dendrogram is formed. Slicing the dendrogram at a certain level yields a set of clusters at that level. The linkage is the metric that is used while merging clusters. We performed agglomerative clustering on both euclidean and cosine UMAP outputs the results of which are shown below:

	Ward-Euc	**Ward-Cosine**	**Single-Euc**	**Single-Cosine**
Homogeneity	0.007906	0.539638	0.006653	0.557766
Completeness	0.008241	0.570745	0.006844	0.578605
V-Measure	0.008706	0.554746	0.006747	0.567994
Adjusted Rand Index	0.000949	0.497392	0.001041	0.426737
Adjusted Mutual Info	0.004776	0.553254	0.003492	0.556562

Table 5: Clustering evaluation metrics for each case

The ward linkage criteria performs lightly better than the single linkage criteria on both accounts.

Question 15.

Note: For this question we use UMAP with cosine metric.

DBSCAN, short for density-based spatial clustering of applications with noise, is a density-based clustering method that is capable of finding arbitrarily shaped clusters with outliers. This property enables it to be much more flexible than K-means. The main idea behind DBSCAN is that if a point belongs to a cluster, then there are many points from that cluster close to it. The main parameters are:

1. "eps": two points are called neighbours of the distance between them is less than eps.
2. min_samples: minimum number of datapoints that define a cluster.

Using this parameter, points are classified as follows:

1. Core point: There are at least min_samples number of points(including itself) in its area within a distance of eps.
2. Border point: Point has less than min_samples points within a distance of eps and it should be reachable from a core point.
3. Outlier: Neither core nor border point.

HDBSCAN, or hierarchical DBSCAN converts DBSCAN into a heirarchical clustering algorithm. It creates dendrograms, as seen in agglomerative clustering.

We ran nested for loops to find the best combination of eps and min_samples for **DBSCAN**. We found the best combination to be **eps 0.5, and min_samples 90**. Similarly, for **HDBSCAN**, we found the best combination to be **eps 0.5, and min_samples 35**. The clustering measures for the best DBSCAN and HDBSCAN models are shown below:

	DBSCAN eps=0.5 min_samples=90	**HDBSCAN eps=0.5 min_samples=35**
Homogeneity	0.469266	0.433170
Completeness	0.592159	0.622251
V-Measure	0.523598	0.510773
Adjusted Rand Index	0.303203	0.243576
Adjusted Mutual Information	0.522423	0.509794

Table 6: Best DBSCAN and HDBSCAN model on cosine UMAP datapoints

Question 16.

The contingency matrices for the best DBSCAN and HDBSCAN models are given below. We found that both perform very similiarly. We see that DBSCAN has found a total of 11 clusters not including the outliers, while HDBSCAN has found a total of 8 clusters not including the outliers. The -1 label refers to these outliers.

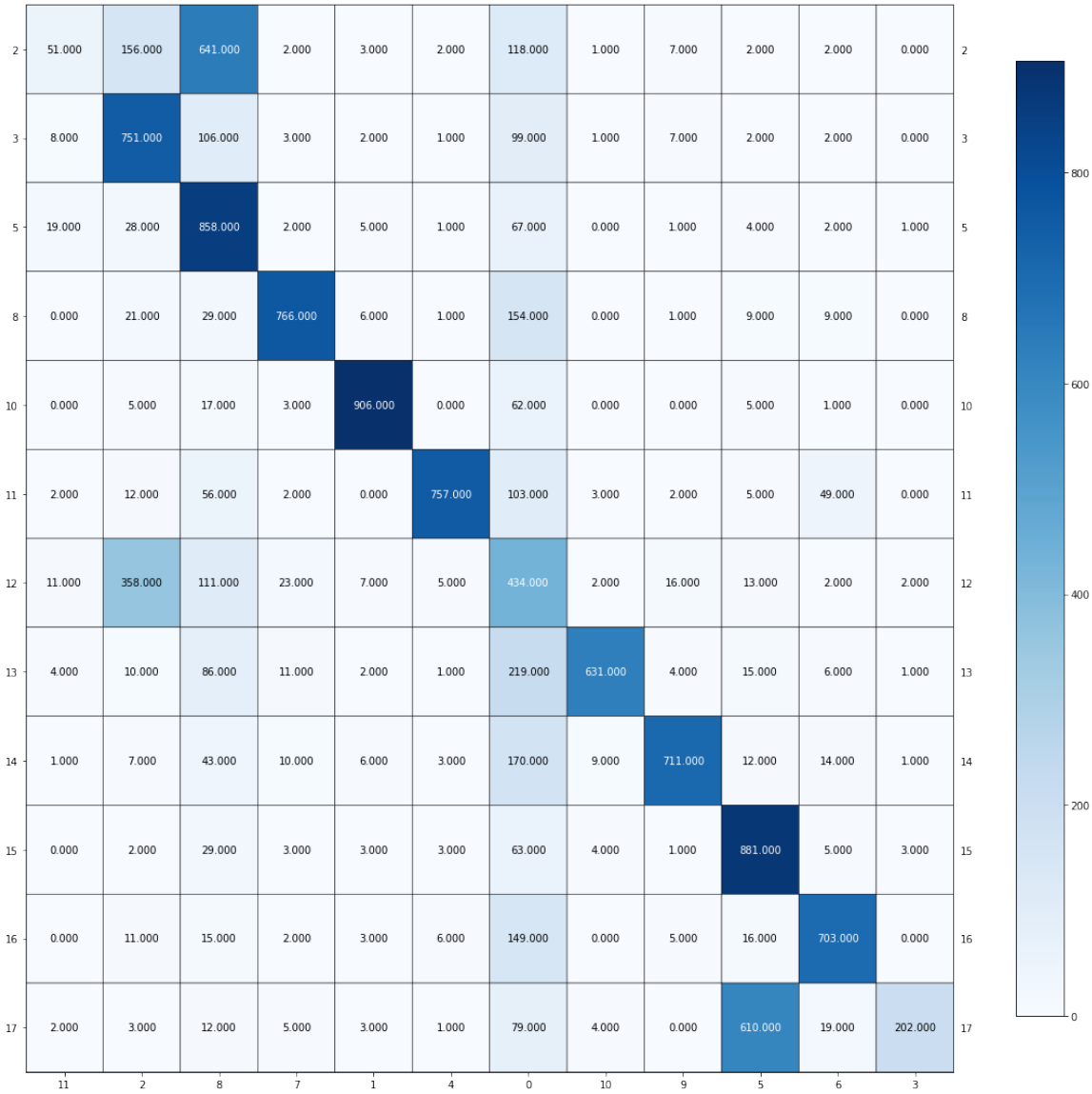


Figure 12: The contingency matrix for the DBSCAN clustering model with $\epsilon = 0.5$ and min_samples $= 90$

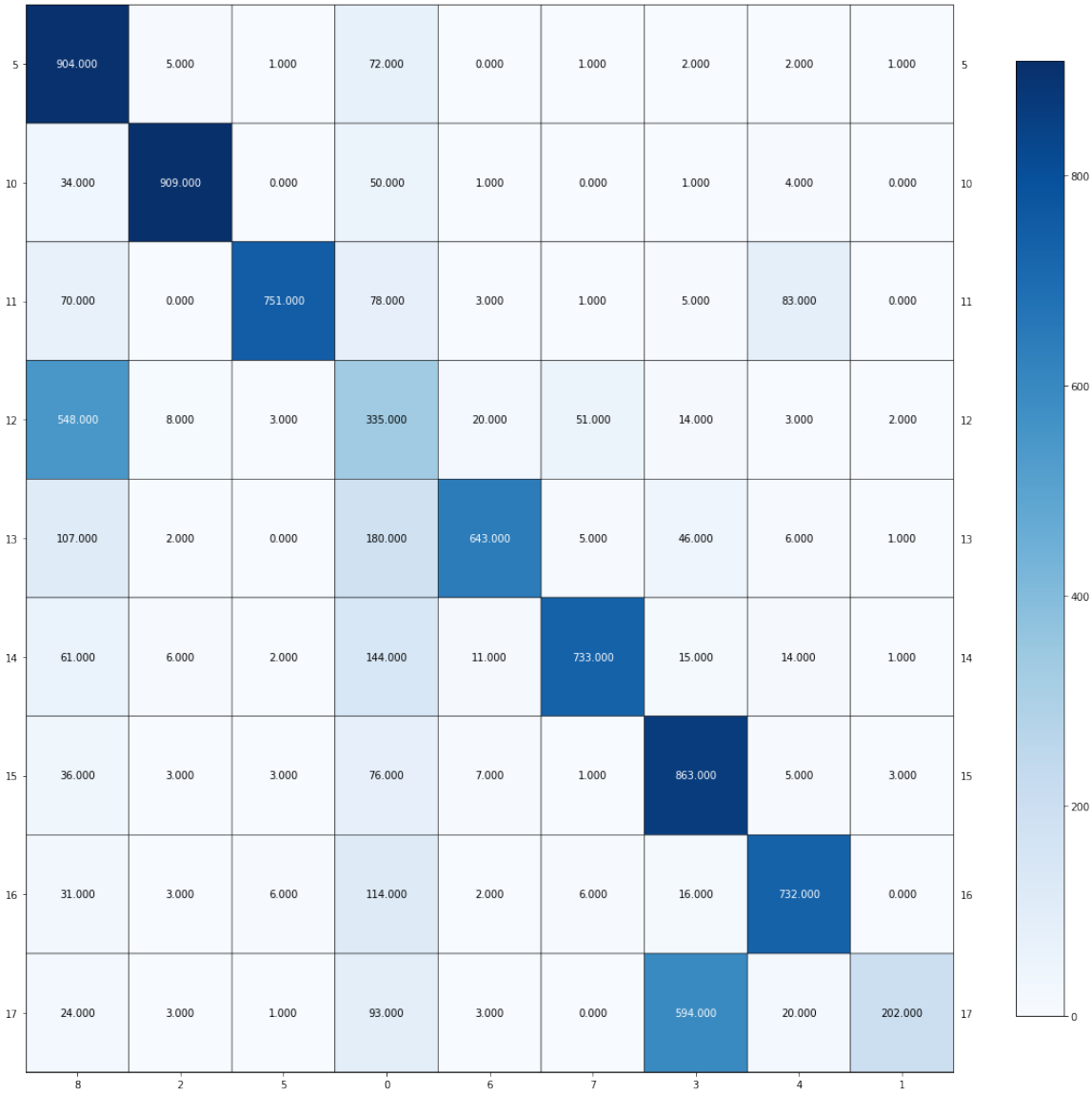


Figure 13: The contingency matrix for the HDBSCAN clustering model with $\epsilon = 0.5$ and min_samples $= 35$

Question 17. and 18.

In this question we are asked to perform a grid search over all combinations of dimensionality reduction methods and clustering methods. Due to memory and colab's time-limit constraints, we performed this task as follows:

1. Fix the dimensionality reduction method.
2. Perform dimensionality reduction over different r = [5, 20, 200].
3. Feed the output into all the different clustering methods.
4. Each clustering method is varied over its hyper-parameters. The hyper-parameters are:
   - K-Means: Number of clusters = [10, 20, 50].
   - Agglomerative: Linkage = [ward, single]; n_cluster = 20.
   - DBSCAN: min_samples = list(range(5, 100, 5)); eps = 0.5.
   - HDBSCAN min_samples = list(range(5, 100, 5)); min_cluster_size = [100, 200]; eps = 0.5.
5. Find the best clustering method in terns of adjusted rand score.

NOTE: We could not run any clustering method without performing dimensionality reduction because colab would crash after using the RAM completely.

The results of our experiments are as follows:

Dimensionality Reduction	**Best r value**	**Best Clustering Algorithm**	**Parameters**	**Best Adjusted Rand Score**
SVD	20	Agglomerative	Linkage = ward	0.167731
NMF	20	Agglomerative	Linkage = ward	0.129689
UMAP	200	K-Means	k_val = 20	0.448073

Table 7: Results obtained through grid search over all dimensionality reducers and clusterers.

As expected, UMAP is by far the best dimensionality reduction technique. While it is surprising the k-means outperformed the other techniques, this might be because UMAP transformed the datapoints into roughly Gaussian distributions. However, one point to be noted is that DBSCAN and HDBSCAN do perform better overall when average scores of the 5 clustering measures are used to find the best model. For a complete list of the grid search metrics obtained, please use the .ipynb file.

Part 2 - Deep Learning and Clustering of Image Data

Introduction

In this section, we aim to explore deep learning methods such as neural networks to extract features from our image dataset and conduct classification. The dataset we use for this part is those of `tf_flowers` dataset. We intend to utilize the trained networks to transform the scatters of images with high dimensions into smooth low dimension manifolds. During the process, our goal is to preserve most of the information while minimizing the loss.

Specifically, we use VGG network in this part and then use the pre-trained network to perform feature extraction on the `tf_flowers` dataset.

Question 19.

Hopefully, the VGG is trained on a broad set of classes. For example, networks trained on the ImageNet dataset can classify more than a thousand categories of objects. Just like a forensic investigator can come across a new crime scene and assess the evidence, a VGG that is trained on the ImageNet Dataset can transfer learning and assess and categorize the new evidence it sees. As such it would have discriminative power over a new custom dataset.

Question 20.

The dataset is imported from the url and extracted to obtain the images. Then the `FeatureExtractor()` class is implemented as follows: first imports the pre-trained VGG model from `pytorch`, and then breaks down the the forward pass into 4 parts. First, it implements the `features` layer to extract VGG-16 Feature Layers, and then it passes the output to the `pooling` layer which was extracted from the vgg architecture using `vgg.avgpool`. Next, it relays this to a `flatten` layer to convert the image into one-dimensional vectors, and then it extracts the first part of fully-connect layer from VGG-16 as the classifier.

A convolutional neural network like VGG accepts images that are of the same size as the input. As the flowers dataset has images of varying sizes, to feed it into a convolutional neural network, we need to reize the images. This process is done by the `transform` parameter. Here, we perform image resizing and cropping and convert it into a tensor. Another operation that we perform is **normalization**. Normalizing all images is an important step in the pre-processing steps of a computer vision task such as image classification: it is performed to make sure that we do not encounter, or at least alleviate, the problem of exploding gradients caused due to the accumulation of values in the gradients that are used to update the weight matrices using backpropagation. Finally, to make sure that we do not exceed the memory constraints, a dataloader is implemented for the whole process so that instead of processing the whole dataset at once, we instead process the images in batches of size 64.

Question 21.

The original images are of varying sizes and have different number of pixels. However, they are all resized to the shape (224, 224)(50176 pixels) to be fed into the VGG16 convolutional neural network. The VGG network extracts $4,096$ features per image.

Question 22.

The extracted features are dense since the majority entries are non-zero.

Question 23.

From the result shown in Table 8, we can see that there are 5 major clusters, which means the t-SNE method properly extracted the features and correctly mapped them. However, there are some sporadic outliers for each clusters. This indicates that some information or features are lost during the process of extraction.

Question 24.

Hyperparameters[1]	**Hyperparameters**[2]	**Dimensionality Reduction**	**Clustering**	**Rand Score**
N/A	k = 5	None	K-Means	0.19013
N/A	n_clusters = 5	None	Agglomerative	0.18855
N/A	min_cluster_size = 3 & min_samples = 5	None	HDBSCAN	0.00240
r = 50	k = 5	SVD	K-Means	0.19303
r = 50	n_clusters = 5	SVD	Agglomerative	0.18522
r = 50	min_cluster_size = 3 & min_samples = 5	SVD	HDBSCAN	0.01616
n_components = 50	k = 5	UMAP	K-Means	0.46356
n_components = 50	n_clusters = 5	UMAP	Agglomerative	0.46826
n_components = 50	min_cluster_size = 3 & min_samples = 5	UMAP	HDBSCAN	0.09452
num_features = 50	k = 5	Autoencoder	K-Means	0.17883
num_features = 50	n_clusters = 5	Autoencoder	Agglomerative	0.24470
num_features = 50	min_cluster_size = 3 & min_samples = 5	Autoencoder	HDBSCAN	0.01210

Table 8: Best Performed Dimensionality Reduction and Clustering Pair in terms of Rand Score.

[1] Hyperparameters of Dimensionlaity Reduction, [2] Hyperparameters of Clustering

Question 25.

The test accuracy of the MLP classifier on the original VGG features is 94%. The test accuracies of the MLP classifier when using reduced-dimension features vary according to different preserved components number. In this project, we use both SVD and UMAP as the dimensionality reduction methods and present the result in percentage for clear comparison.

As shown in Table 9, the accuracy increases as more components are preserved, however, a minor drop of the accuracy is reported when the number of components is set to 300. Generally, the reduced-dimension features are able to reach the accuracy rate of that with original VGG features

Components	**SVD**	**UMAP**
1	24%	83%
2	54%	85%
3	61%	86%
5	69%	88%
10	87%	86%
20	91%	84%
50	94%	85%
100	94%	84%
300	91%	82%

Table 9: Best Performed Dimensionality Reduction and Clustering Pair in terms of Rand Score.

when we implement SVD as the dimensionality reduction method. The performance of UMAP is different than that of SVD. It is seen that even with small number of preserved components, the UMAP-treated features are still able to reach at least 80% accuracy rate. However, as the number increases, no major increase of accuracy rate is reported, which means that when UMAP is implemented, the accuracy will not reach the same level as that of the original dataset.