Clustering and Generative Modelling

Gurvir Dhillon

Submitted for the Degree of Master of Science in

Artificial Intelligence

Department of Computer Science
Royal Holloway University of London
Egham, Surrey TW20 0EX, UK

June 16, 2025

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 14709

Student Name: Gurvir Dhillon

Date of Submission: 01/09/2025

Abstract

The objective of this project is to evaluate the performance of various clustering algorithms using unsupervised learning methodologies and providing background research to support further development on the models explored. The background research will help to understand the context behind the clustering algorithms as well as the appropriate metrics to evaluate the study. Each method used will evaluate both real-world data and synthetically generated datasets, allowing for a controlled investigation into the strengths, limitations and adaptability to different data types for each algorithm.
Another area investigated will be dimensionality reduction techniques and its effects on the models output of data these include PCA (principle component analysis), tDistributed Stochastic Neigbhour Embedding(tSNE) and Uniform Manifold Approximation Projection(UMAP). This will be used to enhance the interpretability in high dimensional spaces within both real datasets and artificial datasets by performing feature extraction. Clustering performance will be assessed using a combination of internal and external evaluation metrics which will address questions surrounding the data structure including the quality of the clustering and its overall adaptation to the model. The models will be compared and contrasted with equal dimensional problems to see the overall impact the datasets have on a controlled dataset will be used to ensure the experiment is fair.

The primary algorithms studied include KMeans, Gaussian Mixture Models and Agglomerative Hierarchical Clustering each offering a different approach to modelling data structures ranging from hard partitioning to probabilistic soft assignment. These models will be compared to one another using the agreed metrics for interpretive comparisons. Furthermore, experiments will investigate the role of dimensionality, clustering shape and how noise can affect the model.

Spectral clustering has shown to outperform traditional clustering algorithms such as the traditional methodologies stated above[Luxburg 2007], the implementation would be considered the secondary study to compare this to traditional methodologies and its average performance. This secondary study will investigate the benefits that flexible non-linear models will give us compared to other methods.

Contents

Acknowledgement

I would like to acknowledge the many hours I have dedicated to this project, driven by a deep personal interest and passion for this subject. This journey has been both challenging and rewarding, and it has strengthened my enthusiasm for the field of machine learning and data science.

Firstly, I would like to offer a special thank you to my supervisor whose wisdom, passion for Data Science and expertise had been of great benefit. Thank you for being a great contribution to the project and for providing me with the guidance necessary.

I would like to express my heartfelt gratitude for my Mother and Father for their unwavering support emotionally, financially and for being a constant source of inspiration throughout this challenging yet rewarding journey. I would like to thank my sisters Gurleen and Mandeep for being great role models and providing an environment for me to improve myself. I would like to thank Ashy for her belief and motivation during my dissertation as well as reminding me why this all matters when I needed it the most. Thank you for lifting me up during moments of weakness and providing me with the energy and strength to move forward. Finally, to my best friend Gurtegh Purewal, thank you so much for being there for me during times of difficulty and for sticking by me all these years.

1 Introduction

Clustering plays a fundamental role in the process of unsupervised learning, widely used for grouping data which are alike and discovering patterns within the process. Its application spans a wide range of domains including customer segmentation, web mining, information retrieval, astronomy, medical sciences and many other disciplines which are data dependent and highly data driven.[Rui & Wunsch 2009].

Clustering uses unlabelled data based on feature similarity - this makes it an important tool as obtaining features without data labels can reduce the computational cost. This spanned over thousands of data points may be costly. Clustering has various applications such as detecting unusual patterns of activities within a banking spending system or in information retrieval clustering can improve the search results by organising documents according to the importance of a word[Jardine & Van Rijsbergen 1971]. Furthermore, KMeans has shown to be sensitive to its initialisation as well as the added challenge of noise/outliers which can distort the algorithm from matching the result of ground truth [Jain 2010].

While traditional methods such as K-means are effective by providing hard assignments of data points to clusters. This limits the level of interpretability and accuracy of the data, especially when points can belong to different clusters as it is close ties. Another limitation of K-means is the number of clusters should be predefined before running the algorithm[Steinley 2006]. K-Means has 3 steps in which it models data, first is its initialisation, the choice is made of the k number of clusters defined, then an assignment is done of each data point to a centroid using the euclidean distance(the mathematical foundation will be discussed in a later section labelled background research). After this recalculation, the centroids are made and updates are based on the current assignments[Hastie et al 2009].

Generative modelling such as Gaussian Mixture Modelling(GMM) offers a more flexible model. This provides us with the ability to learn the underlying distribution by soft assigning data points to clusters, helping to identify patterns and behaviour. This allows for more expressive models where it can fit different shapes and sizes of data whereas K-Means which needs an exact number of clusters [Reynolds 2015]. GMM has been used widely in speech recognition, bioinformatics where the ability to model overlaps with complex distributions of data[Bishop 2006]. K-means assume a spherically shaped cluster with equal distribution of variance in contrast GMMs allow for full covariance matrices. Making them suitable for elongated clusters and for assigning probabilities to each group unlike KMeans[Murphy 2012]. GMM can handle more realistic data and is more flexible in its models due to its parameter flexibility.

Beyond partitioning methods such as GMM and K-Means, hierarchical clustering provides a different adaptation to data clustering by building a dendrogram that represents groups of data points. This method, unlike KMeans, does not require a predefinement of the number of clusters, instead the number can be chosen after building the dendrogram by cutting it to its desired level. In the agglomerative approach, each data point starts its own cluster and pairs merge based on the distance. Various linkage methods are used including single-linkage(shortest distance) and complete linkage(furthest distance) to name a few. These linkage methods determine the structure of the dendrogram. With important applications to image processing and many other areas there is a key issue in terms of time performance and scalability especially with larger datasets[Murtagh 2012]. According to research the two approaches for agglomerative algorithms include naive and optimised. Naive has a time complexity of $O(n^3)$ due to continuous searches over the distance matrix performed whereas improvements in the algorithm(the optimised method) showed a lower complexity of $O(n^2)$ using priority queues and utilising the nearest neighbours whilst efficiently improving the updating process[Day & Edelsbrunner 1984]. It has many applications such as image processing but due to its limitation in scalability struggles in certain contexts which require upscaling.

2 Background research

Clustering algorithms can be broadly categorised by how they define their structure clusters within data. These categories include partitioning based methods, hierarchical clustering methods, density based methods and model approaches. In clustering there is no "one-size fits all" approach, each can differ significantly with how it models data, its distribution and the overall structure. Clusters have its limitations as well as its advantages based on the model provided. Each method can differ in its interpretations of distribution and the underlying structure[Jain, Murty Flynn 1999].

2.1 The evolution of clustering

One of the earliest links to clustering would be the K-means clustering algorithm which was complimented on its ability to handle large datasets and was originally used as a way to handle classification of unsupervised dataset[MacQueen 1967]. This introduced the initial foundation for partition based clustering using hard assignments. This had laid the foundation for scalable, iterative clustering methods that aimed to minimise the variance within-clusters. Due to the predefined clusters mentioned earlier and the assumption for spherically shaped data alternatives were at the forefront of priority for development as not all datasets were linear.

This was followed by the invention of hierarchical clustering which was the idea that the data follows a greedy-like algorithm, where every point is its own cluster and merges pairs based on the objective function. This was a better response compared to the previous model as no predefinement on the number of clusters were necessary and its ability to handle non-linear data structures[Ward 1963]. Further investigation performed by researchers had invented the use of the silhouette metrics to indicate the number of clusters that should be used to support the agglomerative hierarchical clustering[Kaufman & Rousseeuw 1990]. This helped to gather key feedback on the key question that was needed of "how many clusters is enough?". The silhouette score had then become a revolutionary tool used in many facets of clustering technologies and led to better optimisation of models.

Model-based approaches such as the GMM further advanced clustering by introducing probabilistic soft assignments where each datapoint was given a probability of belonging to each cluster. This allows for flexibility and to restructure the data to irregular data structures which performs better for real life data as the shape is unpredictable. Furthermore, GMMs are typically trained using EM algorithms, which iterates over the best fit for the given parameter for the data. More detail on the algorithm will be provided in later sections.

GMMs have become a popularised model type for data as it has been shown to outperform the likes of K-Means clustering[Banfield & Raftery 1993]. These are typically trained using the Expectation Maximisation(EM) algorithms which continuously estimate the parameters that

best fit the data. These methods also support automatic model based selection using criterias like Bayesian Information Criterion(BIC) or Akaike Information Criterion(AIC) which helps to determine the optimal number of components required by the model. As such, GMM offers an adaptive model framework that can be applied to real world datasets[Fraley and Raftery 2002].

2.1.1 Theoretical Foundations

To gain a deeper understanding of clustering methods and its capabilities, it's important to understand the underlying mechanics in place. Fundamentally, a clustering algorithm aims to partition datasets into its own clusters so that the points are similar to each other in the same group with minimal variance. But to understand the quality of the cluster a metric has to be defined[Bishop 2006]. Clusters usually refer to this metric as the objective function or distance metric.

2.1.2 K-Means foundations

The aim of K-Means is to partition N data by P features into k clusters. So that there is minimal variation in the cluster space. The K-Means method assigns each datapoint to clusters using the squared euclidean distance(see figure 1) by calculating the distance from the current centroid to the data point. The following shows the calculation performed by K-Means to compute the distance from the centroid. If the number is closer to the centroid value there is a higher possibility it is a part of this given cluster. However, a key thing to note is that K-Means relies on hard assignments of their data points therefore once it is assigned to a cluster there are no reassignments.

$$dist(X,\ Y)\ =\ ||\ xi\ -\ yk||\ =\sqrt{\sum_{j=1}^{d}\ (Xij\ -\ Ykj)^2}$$

[Figure 1]

Representation	Meaning
$X \in R^{N\times P}$	This is the data matrix for Xij is the value of the j-th variable for the i-th object.
R	This is a real number
C_k	set number of points in cluster k
N_k	number of objects in k clusters.

x̄(k)	This is the centroid of the given cluster

$$\bar{x}_j^{(k)} = \frac{1}{Nk} \sum_{i \in Ck} X_{ij}$$

[Figure 2]

^[Steinley 2006]
Figure 2 shows the calculation of the j-th feature for all the data points that were assigned to cluster k. This shows the mean of the cluster which can then be used to update the centroid of the cluster.

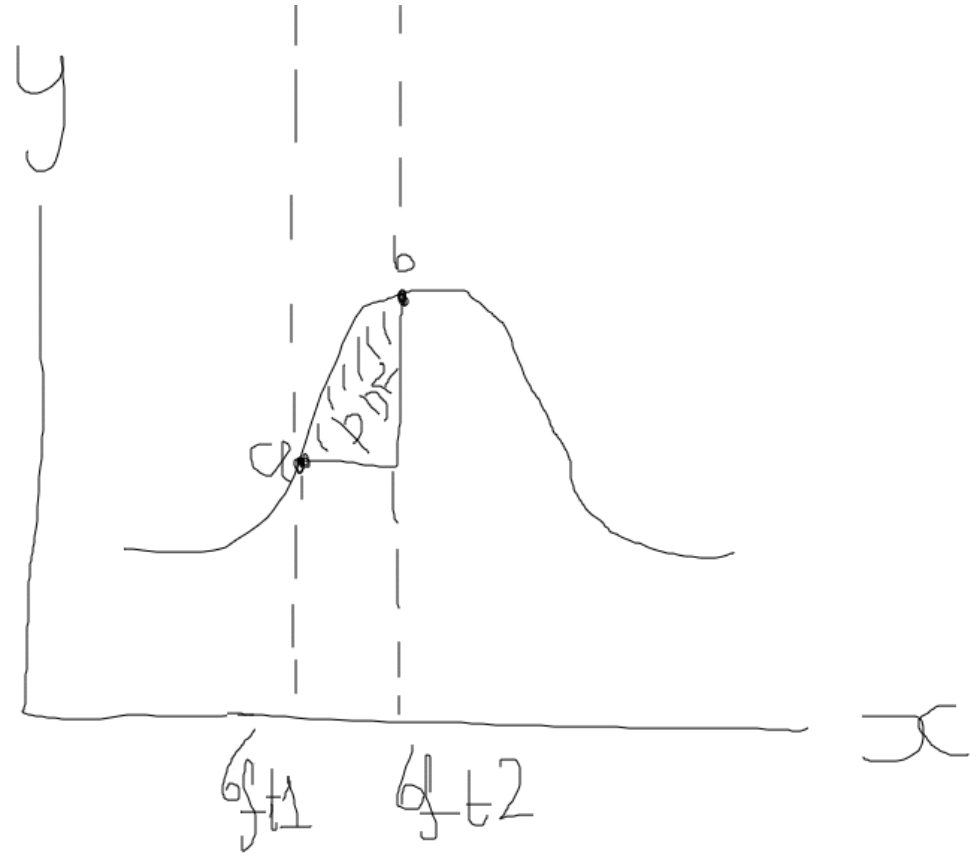
Another approach to the cluster centre is medoids(which is not a part of k-means) but uses any distance metric where the centre is based on a real data point that is most central to the cluster[Tang 2022]. However, due to computationally heaviness this can be inefficient and therefore is not feasible to the study. Therefore traditional K-Means is preferred as opposed to k-centroids.

Depending on the clustering type we get different forms including hard clustering which is most notable in K-means clustering where a given data point is assigned directly to the cluster. But gaussian models which will be discussed in the next section offer a different methodology.

2.1.3 Gaussian Mixture Model foundations

Gaussian models are a probabilistic based model used to represent a gaussian based distribution of data. GMM allows for soft clustering where each data point is associated with belonging to each component. The EM algorithm provides a general solution to incomplete data problems[Dempster, Laird and Rubin 1977]. It is assumed that each data point can be probabilistically belonging to multiple groups at once and each gaussian follows its own bellcurve. GMM uses the maximum likelihood function to find out the likelihood a data point belongs to the given component.
From this maximum likelihood function soft assignment is used to assign data points to the components from the decided probability density function. The probability density function does not give us a probability directly, instead it gives a density which acts as a probability based on a range of data. For example “what is the likelihood of a 21 year old healthy caucasian

male height being between 6ft 1inch and 6ft 2inch". This is then graphically represented by a bellcurve shape called a gaussian.



[Figure 3]
[Used canvas website software linked in reference - G Dhillon Canvas 2023]

Where 6ft 1 inch is the lower bound and 6ft 2inches is the upper bound. Figure 3 shows the probability density function where a and b are two points in a range, for example 6ft 1inch is a and 6ft 2inch is b. The area in the middle presents the density of the probability for the given distribution and this would be between the two assigned values(a and b). From this, probabilistic clustering is used to determine the likelihood of the cluster being given the current assignment and the higher the chance the more likely it will be assigned to that specific cluster. This is referred to as Bayesian probability where it updates the probability of the cluster being assigned to its given cluster.

$$\gamma k(x) = P(z = k \mid x) = \frac{\Pi k \cdot N(x \mid \mu k, \Sigma k)}{\Sigma^{K} j=1 \cdot N(x \mid \mu j, \Sigma j)}$$

[Figure 4]

Representation	Meaning
x	data point
k	cluster
K	total number of clusters
Πk	the probability of cluster k before/ mixing coefficient
$p(x\mid \mu k, \Sigma k)$	likelihood of data point x under cluster k's Gaussian
$\mu k, \Sigma k$	mean and covariance of cluster k (covariance shows the relation of two variables)
$\gamma k(x)$	The probability that point x belongs to the given cluster of k
$N(x\mid \mu k, \Sigma k)$	the gaussian likelihood of how likely x is under the distribution of cluster k.

2.1.4. Expectation step

In the E step of the EM algorithm, the probability of the data point x belonging to a particular gaussian component k is computed using the Bayesian rule. The Bayesian rule is the measure of the likelihood of the data point being assigned to the given cluster based on prior knowledge and new evidence for computation. This probability depends on the given distribution of the data and its set parameters. The denominator in figure 4 shows that all K components must sum up to 1, with a probability for each distribution which enables the use of soft probabilistic assignments. The numerator demonstrates the overall likeliness the data point x belongs to cluster k[Bilmes 1998].

2.1.5 Maximisation step

In the maximisation step of the EM algorithm, the parameters of the Gaussian model namely the mean, covariance and the mixing coefficient are updated to maximise the expected log likelihood based on soft clustering from the expectation step.

$$\mu_k = \frac{\sum_{i=1}^{N} \gamma ik \bullet xi}{\sum_{i=1}^{N} \gamma ik}$$

[Figure 5]

Figure 5 shows the calculation of the centroid of cluster k whilst taking into account the weighted average of the data point.

Representation	**Meaning**
x_i	The i-th data point.
γ_{ik}	The amount cluster k claims point x_i as its own.
$\sum_{i=1}^{N} \gamma ik \bullet x_i$	A weighted sum of all points giving more importance to points that strongly belong to cluster k.
$\sum_{i=1}^{N} \gamma ik$	The total responsibility for cluster k.

Covariance matrix of cluster k

$$\mu_k = \frac{\sum_{i=1}^{N} \gamma ik(xi - \mu k)(xi - \mu k)^T}{\sum_{i=1}^{N} \gamma ik}$$

[Figure 6]

[Formula derived from Deng & Han 2018]

Representation	**Meaning**
$(x_i - \mu_k)$	How far the point x is from the cluster centre.
$(xi - \mu k)(xi - \mu k)^T$	Turn the distance into a matrix that measures directional variance using euclidean distance.
γik	Gives a weighting to the point that the cluster is a part of.
$\sum \gamma ik$	This is used to normalise or standardise the data that we're working with.

This tells the shape and orientation of the cluster. How wide it is and which way it points towards.

Mixing coefficient(cluster weight)

$$\prod_k = \frac{1}{N} \sum_{i=1}^{N} \gamma ik$$

[Figure 7]

[Formula derived from Murphy 2014]

Representation	Meaning
γik	The percentage of responsibility that point x belongs to cluster k
$\sum_{i=1}^{N} \gamma ik$	Sum of the entire percentage of responsibility that cluster k has over all the data points.
$\frac{1}{N}$	Normalise it by total number of points

This gives the proportion of the dataset that is expected to belong to cluster k.

In the maximisation step of the EM algorithm, parameters estimates are continuously updated through iterative cycles using the posterior responsibility from the estimation step. The mean μk is updated as the weighted average (or the centroid), whilst the covariance shown in figure 6 captures the weighted distribution around the mean. The mixing coefficient reflects the expected proportion of data assigned to cluster k and estimates the distribution of the dataset. Together these updates refine the gaussian parameters iteratively to better model the underlying distribution[Ververidis and Kotropolous 2008].

2.1.6 Agglomerative hierarchical clustering foundations

Ward's most notable contribution to the agglomerative method is the implementation of the ESS method [Murtagh & Legendre 2011]. Agglomerative methods rely on the step by step build-up of clusters by using individual data points and merging data points which causes a minimal variance[Müllner 2011]. The ESS objective function is used to find the distance a given data point is from its own centre cluster. The ESS formula has been provided below:

$$ESS = \sum_{q \in Q} \sum_{i \in q} ||x_i - \mu_q||^2$$

[Figure 7a - Error sum of square]

Representation	Meaning

x_i	Represents a data point
μ_q	This is the mean of all clusters for q
Q	This is the set of all clusters

Ward's criterion was originally motivated by the variance decomposition identity which is the total variance of the dataset is the result of the between cluster variance and the within-cluster variance. Noted as the following:

$$T = B + W$$

[Figure 7b - value decomposition]

Representation	**Meaning**
T	Total variance in the dataset
B	The variation between-cluster
W	The within-cluster variance

Since T is a constant for the given dataset, minimising the within-cluster variance *W* maximises the separation between clusters *B*. This balance ensures that clusters remain consistent internally and represents distinctions externally.

2.1.6.1 Merge Cost Formula

At each step the algorithm considers two clusters which will be formally defined as c1 and c2. The associated increase is calculated via the error sum of square(ESS) discussed earlier.

$$\Delta ESS = \frac{|c1| \bullet |c2|}{|c1| + |c2|} ||\mu_{c1} - \mu_{c2}||^2$$

[Figure 7c - merge cost]

Representation	**Meaning**

|c1| |c2|	Size of the clusters
μ_{c1} μ_{c2}	The centroids for its respective clusters

2.1.6.2 Lance-Williams Recurrence Formula

For computational efficiency the ESS had become redundant due to the time to process. As such the Lance-Williams update formula efficiently updates between newly formed clusters by updating its parameters accordingly[Murtagh & Legendre 2014]. The formal definition has been stated in figure 7d:

$$d(i \cup j, k) = \alpha_I d(I, K) + \alpha_J d(J, K) + \beta d(I, J) + \gamma |d(I, K) - d(J, K)|$$

[Figure 7d - Lance William Update]

Representation	**Meaning**
d(I, J)	This is the dis-similiarity between cluster I and J.
$\alpha_I \alpha_J \beta, \gamma$	These are coefficients depending on the linkage method used (these are often referred to as the parameters)
K	Another cluster

So using the formula provided in figure 7d the reusement of old distances for example d(i, j), the combination the weight is done using the coefficients with reference to the linkage rule. Which then informs about the new distance without having to recompute from scratch. Depending on the parameters used this can inform you which linkage methods are used.

2.1.7 Validation metric foundations

Validation of a model is difficult as unsupervised learning lacks ground truth labels during the process. Therefore, we must rely on indirect measures, which are often referred to as validation criteria. The criteria is essential for evaluating how well the resulting clusters reflect meaningful groups and they come in 3 different forms.

External criteria compares the clusters against the actual labels if it is still available. These are fundamental for checking the truth of the label. There are many external validation metrics such

as the Adjusted Rand Index. This metric checks how good the results were and is not directly a part of the clustering process itself(Tang 2022).

Internal validation criteria are particularly valuable in unsupervised context as it is more accessible where there is no ground truth available. These criteria evaluate clustering performances using the properties of the data itself such as the compactness of the cluster (how closely related the data points are within the cluster) and the separation (the distance between clusters from one another). However, whilst internal criteria may provide practical tools for assessing the clustering quality without labels, they do not offer a complete picture of cluster validity in all scenarios.

Relative criteria is the third metric which compares multiple clustering results on the same algorithm. The idea is to test different algorithms and pick the greatest performance based on internal or external scores. In practice it helps to choose the best algorithm or number of clusters for the given dataset.

Studies have been conducted and have suggested the use of clustering tendency measurements which are pre-validation methods before applying the clustering algorithm to assess whether the data can be clustered altogether. To do this it was suggested to use the Hopkins Statistic. This statistic indicates whether the data is structured or unstructured[Amigo & Gonzalo et al 2009].

2.1.7.1 Metric foundations

Whilst purity and entropy are commonly used for external validation metrics they are known for their preference for purity classes. This can lead to potential deception within the data as there will be a higher recorded score leading to overfitting the data as it separates the clusters into many small singleton clusters due to its internal mechanics which rewards homogeneity. This has led to the development of Normalised Mutual Information to counter this effect[Amigó, Gonzalo, Artiles et al 2009]. The F-measure is another metric used which combines both the precision and recall which is how many of the points in the cluster are correctly assigned and how many items are in its true class. The F-measure tends to favour larger clusters due to its sensitivity to recall which reduces its ability to reflect on the fine grained structure it may have underneath. Research has suggested that purity can directly be enforced to ensure homogeneity, so that each cluster is internally consistent. However, it does not account for the completeness of class representation and may therefore overestimate the clustering quality when data is overfragmented.

As it has been mentioned purity compares the clustering results with true categories using precision and recall in information retrieval. The formula shown in figure 8 represents purity as a formula:

$$\text{Purity} = \sum_{i} \frac{|Ci|}{N} max_{j}\text{Precision}(C_i, L_j)$$

[Figure 8]

Representation	**Meaning**
Ci	The i-th cluster created by the algorithm
L	Reflects on the true category of the data given.
N	The total number of data points
| Ci |	The total of the elements in the cluster i
$\text{Precision}(Ci, Lj) = \frac{\lvert Ci \cap Lj \rvert}{\lvert Ci \rvert}$	The proportion of the cluster Ci that actually belongs to the true category Lj

Similarly purity can be reversed (inverse purity) which checks all the items in a class are in one cluster. It favours clustering that groups all the members of the same class. It emphasises no items from the same true group should spread across different clusters. However, there is an issue with the penalisation process as it may not penalise all categories with no group structures at all. Inverse purity will give it a higher score because it only checks when categories are grouped together and not if the grouping is appropriate.

The inverse purity is defined as:

$$\text{Inverse Purity} = \sum_{i} \frac{|Li|}{N} max_{j}\ \text{Precision}(Ci, Lj)$$

[Amigó, Gonzalo, Artiles et al 2009]

[Figure 9]

This does not penalise items mixing from different categories(once the ground truth is known). Therefore, we can reach the maximum value of the inverse purity function by making a single

cluster with all the items. Other metrics such as Entropy based clustering does not penalise outliers to the data.

The introduction of Davies-Bouldin Index highlighted an evolution in clustering which evaluated clustering performed based on intra-clustering similarity and inter-cluster distances. This method evaluates clusters without the need for ground truth in a quantifiable way(unlike entropy and purity). For each cluster, the index calculates a ratio of the distance between points in the same cluster to its centroids and then compares it to all other clusters. Mathematically it can be written as:

$$\frac{1}{k}\sum_{i=1}^{k} max_{j\neq i}\left(\frac{Si + Sj}{Mij}\right)$$

[Davies and Bouldin 1979]

[Figure 10]

Where Si denotes the average distance between members in cluster i from its centroid and Mij shows the distance between centroids of cluster indices i and j(usually using the euclidean distance). A lower DBI value signifies better clustering (compact and well separated clusters). Despite being effective, it can perform suboptimally when clusters vary in shape or size or even in its distribution[Davies and Bouldin 1979]. Moreover, as the DBI requires the use of averaging it is sensitive to the worst case (highest) similarity ratio with any other cluster.

Other techniques such as the Silhouette score had been adapted to perform cluster analysis which evaluates how close each point is to its own cluster(cohesion) and compares how far it is from other clusters(seperation). This offered a visualisation tool and a score based system to interpret the quality of the clustering results. This paired with Davies-Bouldin Index has become an asset which will be used when computing information about the output metric.

The silhouette formula is calculated by the following where i is represented as a single data point:

$$s(i) \;=\; \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

[Rousseeuw 1987]

[Figure 11]

This is where a(i) is the average distance from point i to all other points in its own cluster. b(i) measures the average separation from point i to all nearest neighbouring clusters. This is the smallest average distance from point i to all nearby points[Rousseeuw 1987]. The range varies

from -1 to 1, where clusters are shown to be well structured if they are closer to one and misclassified(worse than randomly assigned) if points are closer to -1. This similar to DBI can be applied to many clustering algorithms including K-means, divisive hierarchical approach and many other clustering algorithms. The silhouette score has been praised to identify overlapping clusters well and research has shown robustness.
As it has been shown that there was no need to use external labelling, silhouette works primarily on the dataset provided unlike purity and entropy. Moreover, Rousseeuw 1987 has shown that it provides us with more information than alternatives such as WCSS(within cluster sum of squares) silhouette balances compactness with separation in its model process. This makes it a versatile tool to use as it provides us with an array of information as well as providing us with the flexibility to use it in many clustering algorithms.

2.1.7.2 Generative modelling principles

Gaussian Mixture Models represent a foundational approach to generative models in the context of clustering. Unlike hard clustering types such as KMeans, GMMs use a soft probabilistic assignment strategy based on Bayesian probability which offers more flexibility as well as allows us to alter clusters depending on the updated information that has been provided by its parameters. Each component in a GMM corresponds to a gaussian distribution with learnable parameters. These parameters are its mean (so the average from the center of each gaussian), the covariance (shape/spread of each gaussian) which describes the correlation and variance between features. Lastly, its mixing coefficient which is the weight of each component(which tells the model how much influence each cluster has on the overall data distribution)[Jiang and Arias Castro 2024].

Furthermore, the EM algorithm is used to iteratively estimate the parameters that maximise the data likelihood. While GMM offers flexibility, they are sensitive to its initialization and can suffer overlaps in its data. To counter this issue, Jiang and Arias-Castro 2024 had proposed a separation constraint variant of GMM(called sc-GMM) which penalises models similar to its components to enforce distinct clusters.

However, standard GMMs have a key issue as it assumes that the data is always modeled as a gaussian distribution given a euclidean space but this is not always the case [He and Cai et al 2010]. These tend to favour structures that implicitly favor data structures that are well approximated by ellipsoids. This can be problematic when the data lies on a nonlinear manifold or if it contains a complex non-Gaussian data structure. These limitations become quite clear in high dimensional settings where noise can be visible, sparsely distributed and have no meaningful correlations.

Developments have been made to incorporate graph-based regularisation to improve the performance of the clustering on non-linear and manifold based structures of data. LAPGMM preserves the local geometric relations between data points during clustering.

Leading on, this would fail in high dimensions in an example of clustering detection for human faces as gaussian models assume a bellcurved structure in a euclidean space however facial data is a nonlinear curved manifold. Furthermore, pixels are not independent from one another - neighbouring pixels entail correlation. In such models data tends to become sparse and noisy therefore the distance between the data loses meaning. So GMMs may assign a face to a different cluster depending on things such as the angle, lighting and pose.

LAPGMM works much better as it preserves local structure of data so it keeps nearby faces in the same cluster even if they don't follow a perfect gaussian structure. It resists distortions caused by high dimensional noise such as variation of lighting, blurred features and so on.

A study had suggested building on the strengths of deep learning models by integrating Variational autoencoders(VAE) with Wasserstein GAN(generative adversarial networks), by using gaussian models or a Student's-t Mixture model as a prior over the latent space. A student's-t Mixture Model is similar to GMM but handles outliers better. VAE is a method which compresses and summarises the underlying data to help capture the most important parts. Wasserstein GAN generates new data samples to improve the quality of data samples in VAE and refines the data to fit the clusters provided. The incorporation of the mixture models allows the latent space to show the structure into clusters while the use of Student's-t mixture model enhances the robustness to outliers. Experiments had shown an improved performance for the overall accuracy and sample quality produced[Yang and Fan 2020].

2.1.7.3 Current challenges in clustering and generative modelling

Despite recent advancements, clustering and generative modelling still faces several persistent issues such as the curse of dimensionality, its sensitivity to outliers and the assumptions on the clustering shape (for example GMMs have a requirement of a bellcurve distribution of data) which limits its applicability[Kriegel, Kroger and Zimek 2009]. Furthermore, evaluating clustering performances presents its very own challenge as traditional metrics like the silhouette score often perform poorly in high dimensional settings. Increasing the number of dimensions also exacerbates scalability problems, making these methods computationally expensive and reducing the interpretability when aiming to extract meaningful patterns[Wang and Ye 2024]. As a result, hidden tradeoffs were involved which included scalability, interpretation and evaluation of the model that are involved. Recent recommendations suggest that the use of deep learning would address key limitations faced by uncovering latent structures and hidden trends within the data itself.

Clustering has seen substantial advancement with the use of deep learning. Combining these two paradigms where representation learning and clustering performed jointly introduces several key challenges which remain active areas of research. An issue in particular is the mutual dependency on clustering performance and representation quality. Traditional clustering methods assume that well structured feature spaces but high dimensional often require more feature extractions which generative models can provide such as GANs and VAE's. Therefore, representations learned by generative models are not inherently clustering friendly which can lead to suboptimal clustering results.

In deep clustering the incorporation of models such as GANs and VAEs introduce a new layer of complexity such as instability in training as generation of data points can lead to unpredictable behaviours as well as convergence issues. Other things such as high computational overhead may hinder the systems scalability to large datasets.

Furthermore, deep clustering allows for mutually exclusive clustering groups however in real world application this may not be a good approach as some data requires an exact cluster for an answer. To lead on from this, deep clustering methods can be very sensitive to initial conditions, as poor initialisation can lead to poor solutions [Ren and Pu 2024]. This can be a big problem in unsupervised learning where there are no ground truths to guide the model to correcting and improving its current model accuracy.

2.2 Differential clustering approaches

2.2.1 Spectral clustering

Spectral clustering introduces an alternative perspective to clustering which is a graph based clustering technique that uses eigenvalues of a similarity matrix to group its data points. Instead of directly clustering in the original feature space, it transforms the problem into a graph partitioning task.

The spectral clustering approach constructs a similarity graph based on the dataset where nodes represent data points and edges are the strength of the similarity/relation between data points. The normalised graph laplacian gets the top eigenvector which is then embedded into a lower dimensional space that emphasises the data structure construction.

This method varies from centroid based methods like KMeans and probabilistic approaches such as GMM as it does not assume the cluster shape. Instead it relies on the graph structure and makes it effective for irregular shaped clusters[Liu and Han 2018].

The scaling parameter is the affinity matrix which plays a crucial role in controlling the locality of connections in the graph. A smaller scaling parameter focuses on a very local relation whereas a larger scaling parameter connects distance points potentially merging clusters together. This has proven to be more effective than using traditional methods such as euclidean distance[Jordan and Weiss 2001].

2.2.2 Semi-supervised learning

So far, unsupervised learning has been explored however to get a holistic view on the effectiveness of unsupervised learning research was undertaken to see its effectiveness.

Semi-supervised learning is a paradigm that sits between supervised and unsupervised learning. It is useful in domains where obtaining labelled data is computationally expensive, but labelled data is abundant. The goal of semi-supervised learning(SSL) is to leverage both labelled and unlabeled data to build more accurate models than unsupervised learning whilst requiring fewer labelled samples than supervised learning.

A major limitation with unsupervised learning is it operates only on unlabelled data, making it prone to producing clusters that may not be in parallel with the ground-truth(if its available). SSL uses a small amount of unlabeled data to guide the learning process and to improve its accuracy and training. There are two forms of SSL which include: semi-supervised classification and semi supervised clustering[Van Engelen & Hoos 2020]. Standard SSL methods like to self-train by: training a model on a small labeled dataset, using it to pseudo label an unlabeled dataset then retraining a new model[Zhou & Belkin 2014].

2.2.2.1 Kernel based methods for semi supervised learning

In semi-supervised learning, a kernel based method aims to use both labelled and unlabeled data by embedding the data into a high dimensional feature space via a kernel function and then enforcing that points close in the space share similar labelling. With its key principle being the manifold assumption which means the decision boundary should lie within a low density region of the data distribution and similar points in the kernel induced space should have the same label.

Kernel based methods extend the capabilities of traditional supervised kernel machines such as SVMs to the semi supervised learning settings where only a small portion of training data is labelled and the majority is unlabeled. The core objective is to see how well the decision function generalises to the data even when the labeled samples are of the essence.

Semi supervised support vector machines

Vector machines look for a pattern in data and separates it according to margins, S3VMs have managed to repeat this process using unlabelled data points. It then places the data according to the decision boundary and sees which density group of points it belongs to. The issue with this approach is that the problem is non-convex so there may be many plausible solutions and distinguishing the best one can be difficult[Skabar and Juneja 2007].

2.3 Dimensional reduction strategies

High dimensional dataset often suffer the curse of dimensionality where the distance metrics become less meaningful, noise becomes easier to capture and does not identify the underlying pattern. This can cause a deterioration of the clustering algorithm. Dimensional reduction strategies are utilised to address this common issue by projecting the data in a lower dimensional space and aims to retain the structure for analysing data and investigating the underlying trend underneath.

The two strategies consist of linear and non-linear approaches with both offering differing approaches to the same issue. Linear strategies assume that the data lies in a linear-like structure or a flat subspace of the high dimensional data therefore it is limited in its capacity to capture bell-curved structures in data. An example of this is shown in appendices 2 which showcases the failure of clustering in a bellcurve shape like appendices 3.

2.3.1 Principle Component Analysis

The most notable examples of linear strategies include principle component analysis where the main objective is to capture most dimensions by merging features and keeping the overall structure and shape of the data. PCA looks at the direction in which the data represents the highest variation(known as the orthogonal axes). The principal component is captured by the new direction in the data space which showcases that high variation. This principal component then follows an order in which the first highest variation is captured, then the second highest variation, then the third and so forth. By retaining only its top components PCA reduces the dimensionality of the dataset while preserving most of its informational content and its structure.

PCA is achieved through using something called eigenvectors(which is the direction where the data varies the most). It's important to note that not all eigenvectors share the same weighting, ones with higher variation often hold larger weighting in comparison to one with little variation. The eigenvectors present the new direction (called principle components) and eigenvalues show the importance of each direction and the variance between the different directions.

An important concept to note is when the principle component captures a lot of variation, it means the data is stretched data and that it is stretched out a lot in that direction. This large spread within the data is referred to as high inertia. This matters as the magnitude of the inertia can often dominate the shape of the cluster and is important when summarising the dataset[Davies & Fearn 2004]. PCA achieves this by using singular value decomposition to identify its main components and its relative importance. This creates fewer dimensions by reconstructing lower dimensional representations and allows us to gain insights to the overall structure for the given dataset.

PCA can run in two different ways: covariance PCA and correlation PCA. Covariance PCA assumes that all variables are measured on the same scale. If covariance variables don't utilise the same scale this can be a problem as one variable can become dominant when analysing the data. For example, if we were to use a handwritten letter recognition system to recognise words given the same scale and the 8 by 8 image utilised each pixel as a variable, if one pixel was measured higher this can be problematic for the training system as all other pixels may be treated with less importance.

Correlation PCA first standardises each variable to the same scale so that the mean is 0 and the variance is at most 1. This is useful when variables are all measured in different ranges or units and this makes sure that nobody is weighed differently. This has more real world application as data can often lack standardisation in reality[Abdi & Williams 2010].

PCA has been described as computationally efficient by removing redundancies by combining the features with effectiveness, PCA is great at preserving the global structure and presenting useful visualisations or insights into the clustering models. But PCA can also have its known limitations which was discussed earlier and its assumption on datasets having a linear relationship which may not hold-up in real world complex datasets. PCA has shown evidence for failing at runtime to accept non-linear data structures which makes it a liability when dealing with real world data[Arora, Hu & Kothari 2018].

2.3.2 Independent Component Analysis (ICA)

Independent component analysis is another linear dimensional reduction method that unlike PCA doesn’t look for direction that maximises variance. Instead ICA uncovers statistical independent components which underlie the observed data. Meaning that if two variables were to be uncovered one should not interrupt the other's observation, similar to how two human voices should be treated independently in a recording device. The components are often referred to as latent factors as they represent hidden signals mixed together in a dataset. Because of its ability to separate mixed signals proficiently, ICA has been widely applied in the healthcare industry in particular including fMRI scanning and EEG’s(Electroencephalogram).

However, ICA has its limitations including its ability to perform depends heavily on the type of dataset - whether it is real or synthetic. It can often struggle with high dimensional spaces, in particular its sensitivity to noise and since it remains a linear method, it fails to capture its non-linear relations which is an inherited flaw within the model itself. This can reduce its overall effectiveness in practice[Nanga & Bawah et al 2021].

2.3.3 tDistributed-Stochastic Neighbour Embedding(t-SNE)

The assumption on linear models and the dataset having to be a linear shape can reduce its overall effectiveness in a real world scenario therefore to uncover complex shapes and alternative structures we can use non-linear structures to uncover the patterns. The first model explored is the t-stochastic neighbour embedding model which focuses on the local neighbourhoods rather than preserving the global structure.

The method does this by converting the distance between data points in high dimensional spaces into gaussian based probabilities(shown in figure 3) which represent how likely two points are to be neighbours[Maaten & Hinton 2008]. It then tries to reproduce these neighbourhood relationships in a lower dimensional space. So this means that points which are closer together in the original dataset will be closer in the visualisation for tSNE. This makes it stand out more clearly.

One of t-SNEs strengths is its ability to reveal non-linear structures that PCA often misses. This makes it suitable for more complex datasets, where it can uncover natural groupings and patterns that would have otherwise remained hidden.

However, t-SNE has been shown to be computationally very expensive making it less practical for larger datasets which require higher computational power. The optimisation process is non-convex, meaning there is no guarantee there will be a best solution[Wattenberg, Viegeas and Johnson 2016]. The data structure can distort its global structure, sometimes exaggerating the distance between clusters therefore when it comes to metric evaluation this can become less accurate when reading due to this exaggeration. The results are highly sensitive to its parameters(such as the learning rate and perplexity), which makes it difficult to reproduce or interpret consistently and due to it using random initialisation its runtime may vary depending on where exactly it initiated, producing varied results. Despite the drawbacks, t-SNE is often used as a human interpretable plot for visualising datasets where local similarities are grouped together in accordance.

2.3.4 Uniform Manifold Approximation and Projection(UMAP)

Another popular form of non-linear model is referred to as UMAP. Research shows it can produce visualisation that can be faster and more scalable which makes it more suitable for large datasets [McInnes, Healy & Melville 2018]. Unlike PCA, which focuses on preserving the overall variance and t-SNE which focuses mainly on local neighborhoods, UMAP aims to preserve both the global and local structure.

UMAP has a theoretically solid foundation. Which was built on Riemannian geometry(which is the mathematics of curved spaces) and algebraic topology(used to study connectivity and shape). This allows UMAP to reframe manifold learning and dimensionality reduction as a mathematical problem rather than relying on heuristic values like other methodologies such as t-SNE.

UMAP works by constructing graph based representation of the dataset where the points are connected according to its local neighbourhood relationship. These connections are encoded using fuzzy sets which assign probabilities to reflect the probabilities of the likelihood for it being linked. Once the structure is built in high dimensional spaces, UMAP maps it into fewer dimensions, while preserving as much of the local and global structure as possible. Optimisation is carried out using stochastic gradient descent, outputting the visualisations which highlight both cluster separation and broader patterns.

A key advantage is its ability to maintain proximity and keep points close together in a reduced space after its transformation[Becht & McInnes et al 2019]. This makes it particularly valuable for not only visualisation but also as a preprocessing step in machine learning where the data structures integrity is of utmost importance.

3 Preparation in clustering

3.1 Source code dependencies

This section outlines the key tools and libraries used throughout the project. The code was written in python. The project was split into 3 main sections for the main criteria: GMM, KMeans and AHC. All the main experiments used the same datasets as a baseline for modelling the data to compare the performance of the different algorithms. Each experiment had the same dimensionality reduction methods applied to show the performance of the data clustering model and on the real datasets standardisation was introduced to ensure that all data was treated with the same weighting. The program used object oriented programming as its base for creating the models of the various clustering algorithms[please see appendices 5 to 8 for the details].

The core dependencies are listed below:

- Python version 3.12.4
- Numpy (library)
- UMAP (library)
- Scipy(library)
- Time (library)

The Scikit-learn library was used including:

- make_blobs
- make_moons(utilised in later sections)
- Load_digits
- tSNE
- KMeans++
- Evaluation metrics such as silhouette score, ari, DBI

3.1.1 Source code structure

The code was modularised into its uses and followed this structure:

```
base) gurvi@Mac fy_project_v2 % tree
.
├── ACHC
│   ├── achc_digits.ipynb
│   └── achc_fake_data.ipynb
├── GMM
│   ├── gmm_digits.ipynb
```

```
│   └── gmm_fake_data.ipynb
├── Kernel Based methods
│   ├── kernel_digits.ipynb
│   └── kernel_fake_data.ipynb
├── KMeans
│   ├── kmeans_digits.ipynb
│   ├── KMeans_fake_data.ipynb
│   └── KMeans++
├── reading_docs
│   ├── 1-s2.0-S0925231217311815-main.pdf
etc…
│   └── umap_proj_dim_red.pdf
└── Spectral Clustering
    ├── SpectralClustering_digits.ipynb
    └── SpectralClustering_fake_data.ipynb
```

[Figure 12 Code Structure]

3.2.1 Synthetic dataset

A synthetic dataset was generated using make_blobs from Scikit-Learn library to create well-separated, clustering of data. This approach allowed precise control over the number of clusters, sample size and providing a clean interpretable dataset for a baseline clustering of performance. The dataset was constructed with 64 features and a predefined number of clusters(10 clusters) similar to the real world dataset used later as well as 1797 data points. Synthetic data allows for flexibility within the model and to test out key areas needed.

By using synthetic datasets the ground truth labels were known which enables more controlled evaluation of clustering performance using both internal and external metrics. Furthermore, it helps to analyse vital information regarding clustering. Furthermore, make_blobs serves as a way to test for dimensionality reduction methods in a setting where the underlying structure is understood. This provides us with a good baseline to compare to real data structures after reduction of dimensions.

3.2.2 Real dataset

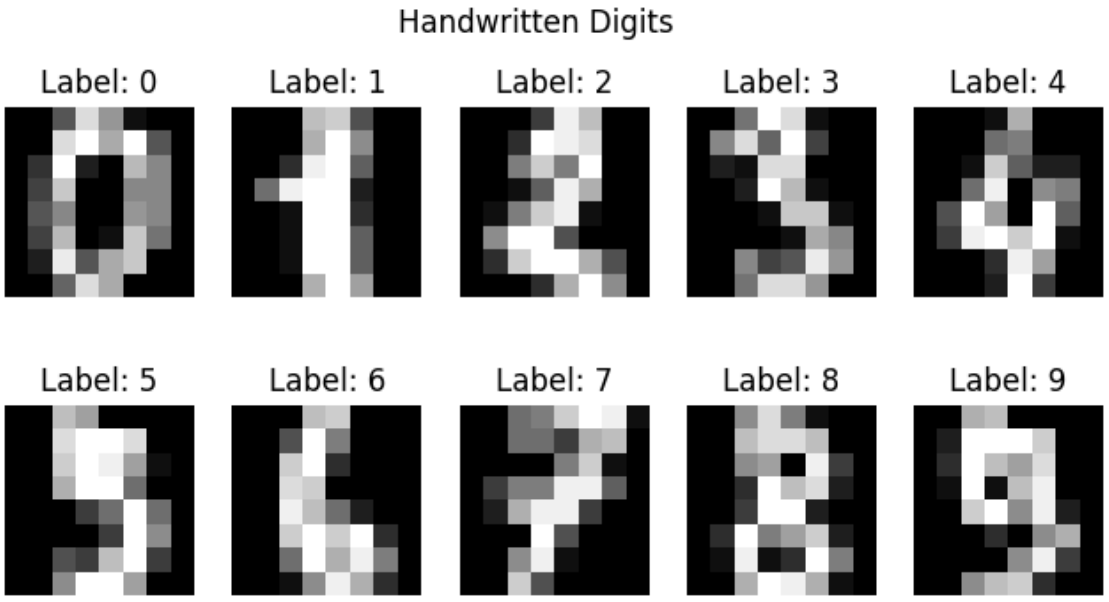
The real dataset will use load_digits which is available through the Scikit-learn dataset, this shows a real world performance on a set of data. This contains 1797 samples of handwritten digits (ranging from 0-9) with 64 dimensions. Unlike synthetic data this will contain practical

challenges such as overlapping datasets, class imbalance and nonlinear separability which makes clustering complex.

Ground truth is still available but is withheld during clustering to stimulate an unsupervised learning scenario. These were later used to validate the model using metrics such as Adjusted Rand Index.

The real world dataset was preprocessed with the use of standardisation before applying dimensional reductional algorithms to ensure that all the variables carry the same weighting and ensuring the level of data bias is reduced.

By applying each clustering algorithm to the digits dataset we evaluate the effectiveness of each method to uncover meaningful structures given the context of high-dimensional data with the added noise. Using dimensionality reduction methods like PCA, t-SNE and UMAP we can discover better clusters of data and discover underlying patterns in data.
The digits dataset serves as a bridge between the theoretical validation on synthetic data and real world complexities faced, providing a solid foundation for evaluating both clustering performance and its visualisation strategies.



[Figure 13 - sample handwritten digits dataset]
Figure 13 shows the sample digit handwritten dataset which demonstrates an 8 by 8 image of digits ranging from 0 to 9. With 64 dimensions and shown in an 8 by 8 image. This shows how the digits are represented and how the labels interpret each dataset. It can be seen due to the

low resolution visual similarities can be made for example a similarity between 1 and 7 or 9 and 8.

3.2.3 Project environment

The main development and execution of the project were conducted using JupyterNotebook hosted on a remote Linux-based virtual server provided by the university. Access to this environment was bridged using NoMachine which allowed me to gain full graphical interaction with the interface. Jupyter Notebook provides a flexible and user-friendly interface that allows for the integration of code, providing visualisation and works well for documentation purposes. It works well for seamless integration with libraries such as Matplotlib, pandas and numpy without further need to pre-install all required libraries. Furthermore, its UI supports a logical flow which allows for clearer debugging as the code is organised into codeblocks compared to Visual Studio Code's interface.

Server environment: Linux Ubuntu 20.04.3 LTS 64bit
Interface: NoMachine
Development tool: Jupyter Notebook

3.2.4 Dimensionality reduction methods

To reduce computational cost and improve the clustering performance, a total of 3 dimensional reduction methods were applied to this project on both the synthetic and real data. This includes PCA, UMAP and t-SNE. These were chosen because they represent a mixture of linear and non-linear techniques, offering a different perspective on the data.

First selection: Principle component analysis (PCA)
PCA was used as a baseline linear model. To reduce the 64 dimensional feature space in the digits dataset and the make_blobs dataset into smaller segmentations of principal components whilst retaining most of the variance. This made the cluster much more efficient and provided interpretable axes of variation.

Second selection: t-SNE
This model was selected for its visualisation, as it preserves local neighbourhoods and it produces clear cluster separation in 2d plots, which helped in evaluating how well the data grouped together. Its computational cost and sensitivity to hyperparameters were also duly noted but it gives us a good comparison to PCA in terms of preservation of global structure and local structure.

Third selection: UMAP

UMAP was selected as a more scalable non-linear method which balances out the approaches noted above (pca and t-SNE). This produces a more stable visualisation strategy compared to the non-linear method mentioned prior. UMAPs ability to reveal manifold structures in the data makes it perfect for projecting the given clusters.

3.2.5 Metrics chosen

This section outlines the evaluation metrics used to assess and evaluate the performance of the model developed. The selection chosen were used across all models to identify and compare the performance.

Metric used	Purpose	Categorisation	Interpretation	Insights & considerations
Silhouette Score	Measures the cluster cohesion and its separation. Measures how similar a point is in its own cluster compared to the next nearest cluster.	Internal	Values can range between -1 to 1. ● Closer to 1 = well clustered and good separation. ● Closer to -1 shows data points may be assigned to the wrong cluster. ● A value near 0 represents ambiguity and overlapping of the data.	Silhouette scores are usually sensitive to cluster shapes and these metrics do not require ground truth to perform. Sensitive to cluster shapes and densities thus can be less flexible as a model.
Davies-Bouldin Index	Assesses the average similarity between clusters and its most similar clusters. DBI gives a bigger picture point of view compared to silhouette score.	Internal	A value closer to 0 is better clustering.	DBI looks at the evaluation more broadly compared to Silhouette scores. Good for comparison between algorithms.
Adjusted Rand Index(ARI)	Measures the similarity between the	External	1 = perfect assignment 0 = random assignment 0 > inaccurate assignment	Useful when ground truth is available after training is completed.

	cluster and its true label.			
Runtime	Measures computational efficiency of the algorithm	Performance	shorter time shows more efficiency	Useful to compare algorithms runtime on larger datasets and its ability to provide a response within a reasonable timeframe.

[Figure 14 metrics table summary]

4 Model development and evaluation

4.1 KMeans clustering

4.1.1 Synthetic data KMeans

KMeans Model Synthetic Data Metrics - make_blobs()

Method	Silhouette Score	Davies-Bouldin Index	Adjusted Random Index	Runtime
PCA + KMeans	0.860	3.92	0.753	0.405 seconds
UMAP + KMeans	0.867	2.720	0.773	17.198 seconds
TSNE + KMeans	0.818	2.484	0.773	16.383 seconds

[Figure 15]

From the data shown in figure 15 this demonstrates a higher silhouette score for UMAP and a lower score for Davies-Bouldin Index. This indicates compact and well-separated clustering. t-SNE and KMeans also performed strongly with a high silhouette score(despite being the lowest silhouette score out of the 3 dimension reduction models). Which implies that t-SNE effectively captured some essence of non-linear data structures within the make_blobs dataset. PCA mixed with KMeans showed a lower score for the Silhouette score and the highest recorded Davies-Bouldin Index to counteract, which shows that clusters are less compact. In terms of the three methods similar performance was achieved across the board varying from 0.75 to 0.77. This shows that KMeans is consistently aligning reasonably well with the ground truth regardless of dimensionality reduction methods application. However, in terms of runtime performance PCA represented far better efficiency whereas UMAP and t-SNE were slower due to computational complexities within its projections.

4.1.2 Real data standard KMeans results

Method	Silhouette Score	Davies-Bouldin Index	Adjusted Random Index	Runtime
PCA + KMeans	0.055	6.630	0.320	0.738 seconds
UMAP + KMeans	0.585	1.135	0.664	10.174 seconds
TSNE + KMeans	0.494	1.159	0.743	19.476 seconds

[Figure 16]

PCA on a real dataset had recorded the worst performance on average with a high DBI and a low Silhouette score showing overlap between clusters and ambiguous assignments between clusters as well as a large spread of data. This is further supported by the PCA graph provided in Figure 19a. This shows a clear lack of boundaries and an overlap within the cluster set. PCA had shown a poor performance in the ARI scale with 0.320 indicating that it fails to recover the true digits grouping effectively.

UMAP achieves the highest silhouette score and a lower DBI suggesting well defined and separated clusters. This is confirmed by the UMAP projection which clusters appear more compact and distinct. t-SNE also performs similarly to UMAP with a respectable silhouette score(second highest) and the lowest Davies-Bouldin Index indicating tight and separated clusters of data. t-SNE visualisation shows this with minimalistic overlap(please see figure 19c). Furthermore, t-SNE recorded the highest ARI implying it suits the dataset in both a synthetic environment as well as in the real world for the KMeans algorithm.

The runtime had shown to be most effective for PCA, which was expected as it requires less computational resources but performs the clustering quality and separation poorly. t-SNE shows a tradeoff due to high clustering quality but with the added expense of computational time as this was the slowest runtime performance overall.

4.1.3 KMeans++ with the digits dataset

After evaluating both KMeans using real data and synthetic data, the choice was made to use KMeans++ to understand the role of initialisation and its importance when clustering data. Standard KMeans as mentioned previously can lead to suboptimal clustering based on its initialisation therefore the choice was to see its impact on clustering. KMeans++ intelligently selects initial centroids that are well spread out and can lead to faster convergence and higher quality clustering(therefore it is assumed it will give a better silhouette score and lower DBI).

Method	Silhouette Score	Davies-Bouldin Index	Adjusted Random Index	Runtime
PCA & KMeans++	0.105	2.183	0.361	0.202 seconds
UMAP & KMeans++	0.610	1.140	0.790	6.933 seconds
TSNE & KMeans++	0.556	0.980	0.890	13.965 seconds

[Figure 17]

From the results and visualisations of KMeans++ on the real digits dataset after dimensionality reduction we can observe several improvements and behavioural differences compared to traditional KMeans.

Notably the KMeans++ better initialisation had led to an increase in clustering performance, particularly in the ARI scores. For instance ARI for t-SNE had reached its peak of 0.89 suggesting the strongest ground truth alignment compared to previous trials(synthetic standard KMeans and real data standard KMeans). The Davies-Bouldin Index improves across all the methods especially with t-SNE and UMAP which shows higher levels of intra-clustering and inter-clustering distances.

Furthermore, KMeans++ had decreased the computational expense across the board due to smarter placement of the centroid initialisations. UMAP still is slower but has sped up compared to the standardised KMeans model. Visually UMAP and t-SNE show a clear and well defined grouping but the intra-cluster compactness appears randomised under KMeans++ with fewer scattered or ambiguous points showcasing that KMeans still contains flaws when running the algorithm.

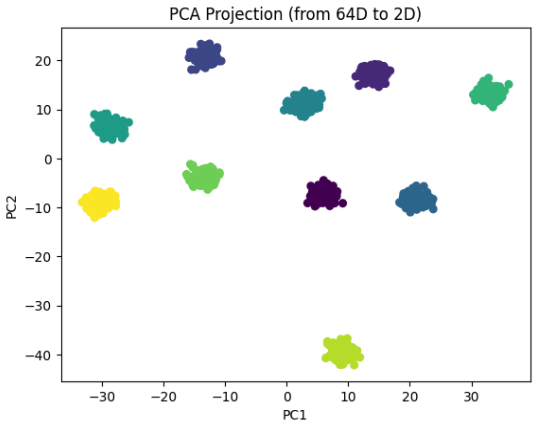
4.1.4 Analysis of KMeans

The application of KMeans clustering across both synthetic and real-world datasets reveals the complex relations between data structures, dimensionality reduction techniques and initialisation methods. With the use of synthetic data generation tools(make_blobs), the underlying cluster structure was well defined and well separated. This resulted in higher quality of clustering which was evident from Figures 15 - 17 in the data provided.
However, a shift in performance is noticeable when transitioning to real world datasets. Due to its complexities such as the overlapping structures and its non-linear distributions. The first recognisable issue was with the PCA and KMeans, which showed a significant deterioration in performance reflecting the inability of non-linear projections and naive initialisation to capture

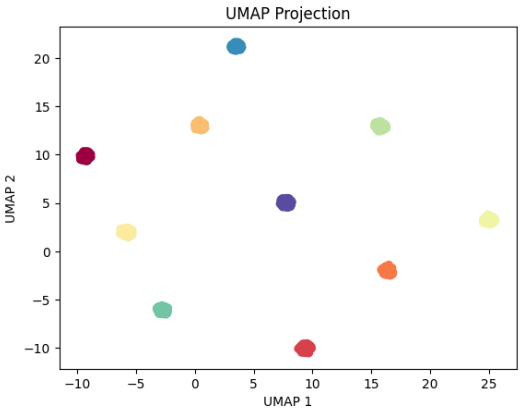
meaningful grouping patterns in the data. UMAP and t-SNE demonstrated the complete opposite, showing preservation of local and global structures. Providing an improvement of the ARI and silhouette score with runtime costs as its only drawback.

To see the overall impact of centroid initialisation, KMeans++ was used to see the advantages and had shown an increase in clustering performance for the real data with an increased ARI which rose significantly for t-SNE and UMAP projections. This has shown that smart initialisation has the impact to avoid leading to suboptimal partitions.
Furthermore, using the validation metrics explored: synthetic data and real data had shown that t-SNE and UMAP had performed stronger with the digits dataset particularly when paired with KMeans++ which offer not just better metrics but also better graphical results with clearer clusters and structure in the digits data.

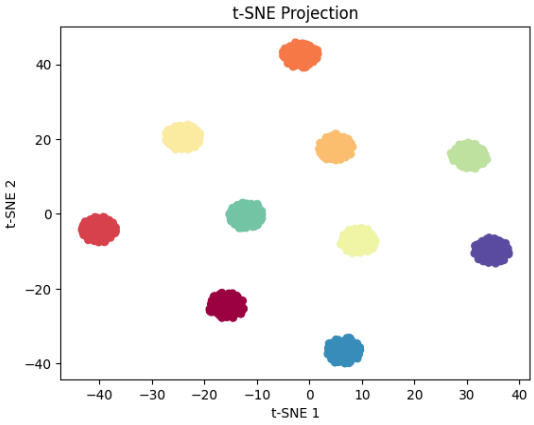
KMeans Artificial Data Results



[Figure 18a KMeans synthetic pca]

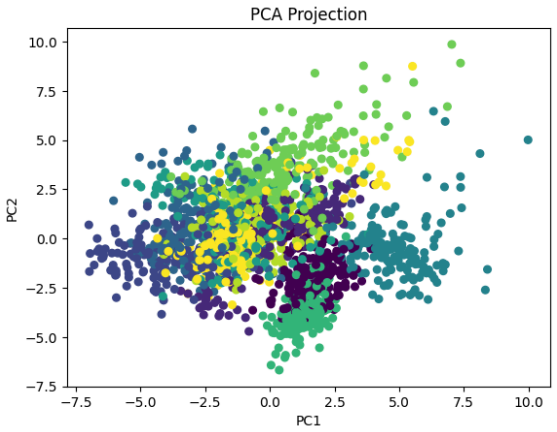


[Figure 18b KMeans synthetic umap]

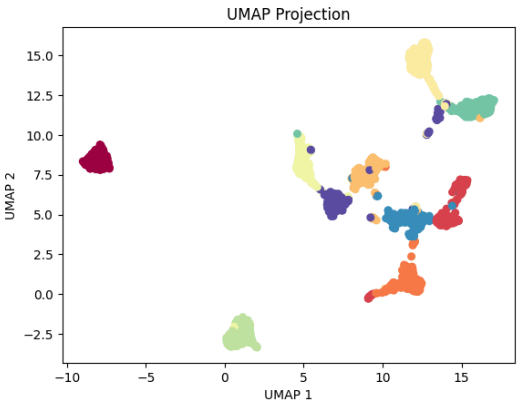


[Figure 18c KMeans synthetic t-sne]

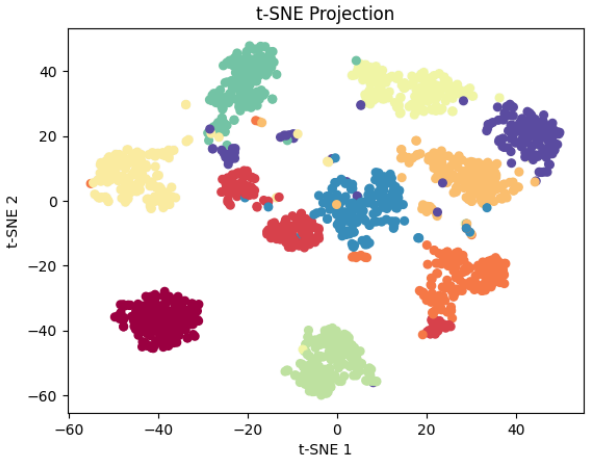
KMeans Digits Data:



[Figure 19a - pca kmeans real data]

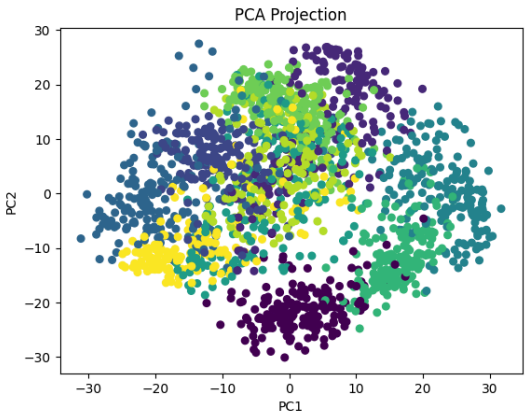


[Figure 19b - UMAP KMeans real data]

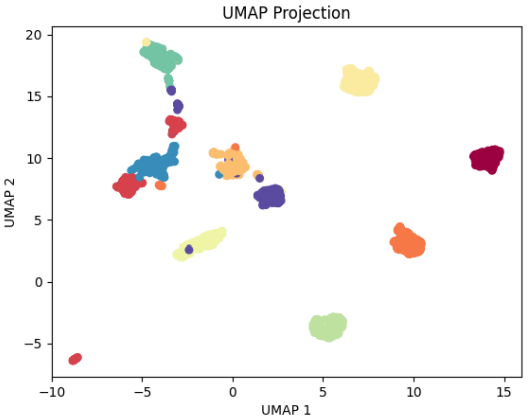


[Figure 19c t-sne KMeans real data]

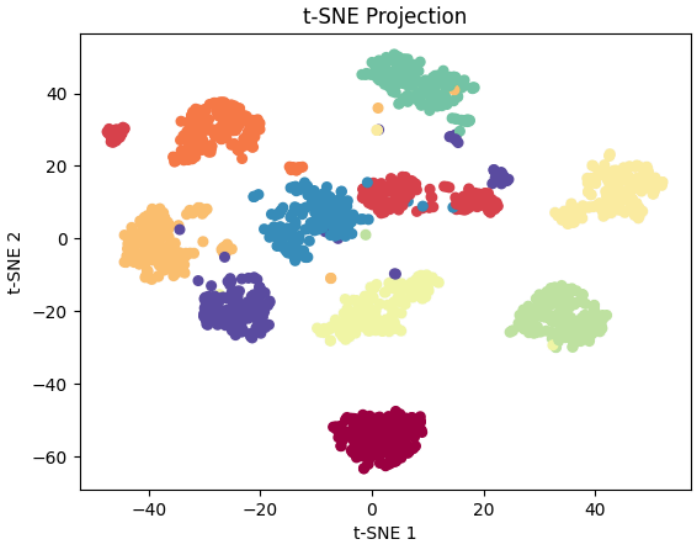
KMeans++ Real data



[Figure 20a KMeans++ pca]



[Figure 20b KMeans++ umap]



[Figure 20c KMeans++ t-SNE]

4.2 GMM Clustering

4.2.1 Synthetic dataset GMM

Method	Silhouette Score	Davies-Bouldin Index	Adjusted Random Index	Runtime
PCA + GMM	0.731	0.396	0.326	0.212 seconds
UMAP + GMM	0.953	0.068	0.513	0.129 seconds
TSNE + GMM	0.879	0.172	0.502	0.218 seconds

[Figure 21 - synthetic data GMM]

The synthetic dataset had shown a good performance based on Figure 16 provided above, with UMAP showing the best performance all around combined with gaussian models. UMAP showed the highest silhouette score indicating points that match its cluster and that it separates well from the points that do not correlate to its data type. Furthermore, the runtime had shown higher efficiency in comparison to t-SNE and PCA. With a higher level of ground truth of 0.513 compared to a lower score in the other dimensional reduction methods. In comparison PCA had performed better than expected however PCA tends to show a higher silhouette score based on artificial data but deteriorates on real data. t-SNE had also performed well on this dataset closely matching UMAP but had less accuracy as it lacked clustering quality by a small amount and was shown to be less than 0.1 second slower than UMAP.

4.2.2 Real dataset GMM

Method	Silhouette Score	Davies-Bouldin Index	Adjusted Random Index	Runtime
PCA + GMM	0.105	2.183	0.239	0.763 seconds
UMAP + GMM	0.610	1.140	0.338	0.157 seconds
TSNE + GMM	0.556	0.980	0.268	0.160 seconds

[Figure 22 - Real Data GMM]

As expected PCA had deterioration based on its silhouette score showing that data is borderlining between two clusters based on probability. PCA has shown it does not pair well with gaussian models based on the fact it has the lowest recorded accuracy throughout the models, however gaussian models have shown the lowest accuracy for prediction compared to the KMeans model(including KMeans++).

However, it can be stated that Gaussian models show a more computationally efficient performance across the board (including t-SNE and UMAP) as can be seen by the runtime.

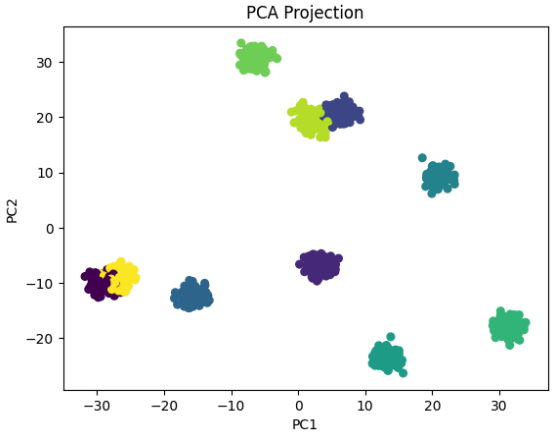
UMAP had shown a better average across the board with the highest silhouette score, the DBI had shown there was some overlap however had performed better than PCA if we compare figure 27b with its counterparts such as 27c. This shows less overlap with the data and data is less "spread out" in terms of data point distribution. For example t-SNE projects the purple cluster shown to have points further away in figure 24c whereas UMAP deals with this better as points show better distancing however there is still some overlap but is less than t-SNE.

4.2.3 GMM Comparative overview

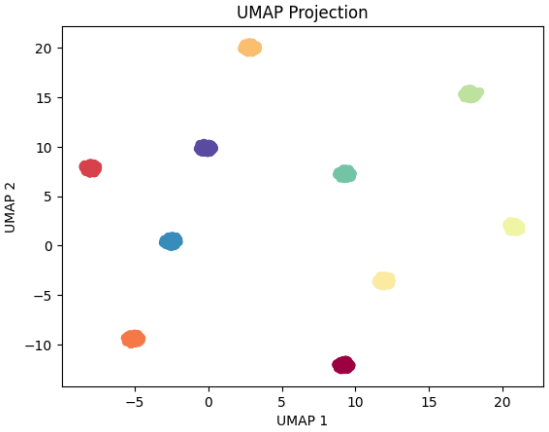
When comparing GMM performance across dimensionality reduction techniques some distinct patterns emerge. For the synthetic dataset GMM generally adapted more flexibly to the underlying data distribution which was expected due to its ability to model based on probabilities rather than enforcing the dataset to a rigid spherical boundary. This flexibility became evident in the Davies-Bouldin Index through the t-SNE and UMAP highlighting that GMM can more effectively exploit compact manifolds. PCA showed a mismatch with GMM as PCA is a linear structure and GMM projects non-linear relations therefore this caused constraint on the gaussian model.

On real data, the difference started to become much more obvious when comparing GMM and KMeans. PCA had yet again performed poorly showing that linear variance maximisation is not sufficient to capture the complex data structure of the digits dataset. Both UMAP and t-SNE capture better clustering structures but GMM tends to capture broader cluster overlap than KMeans, leading to improvements in the silhouette score and DBI but lower adjusted random score in some cases. This tradeoff reflects the probabilistic nature shown in GMM: which shows uncertainty in boundary assignment but this can represent the real world data more compared to KMeans as data may overlap therefore probability will be useful for the real-world.

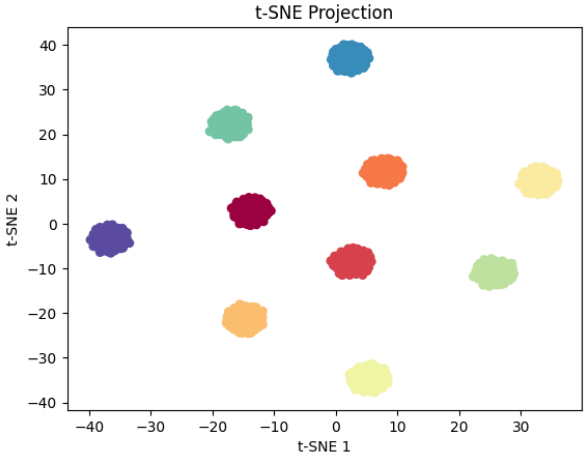
GMM Synthetic Results:



[Figure 23a - pca GMM synthetic dataset]

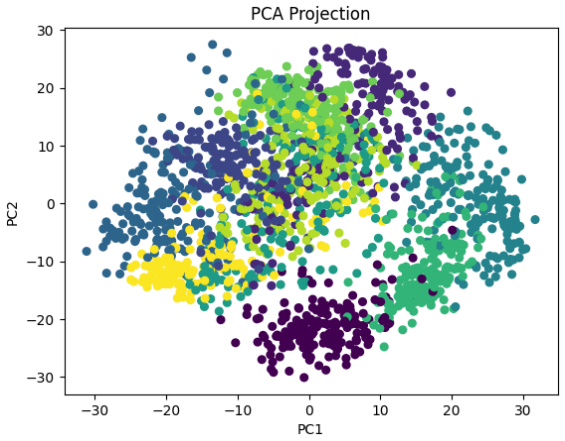


[Figure 23b umap GMM synthetic]

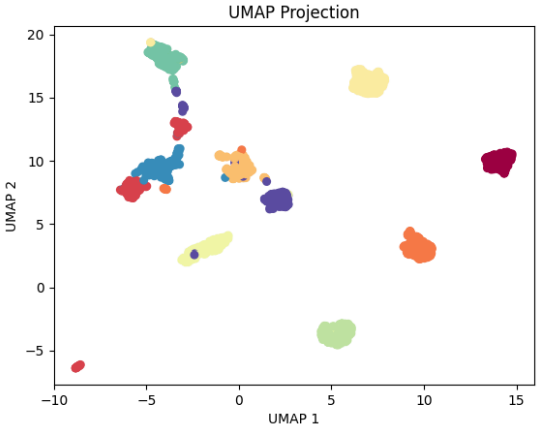


[Figure 23c t-SNE GMM synthetic]

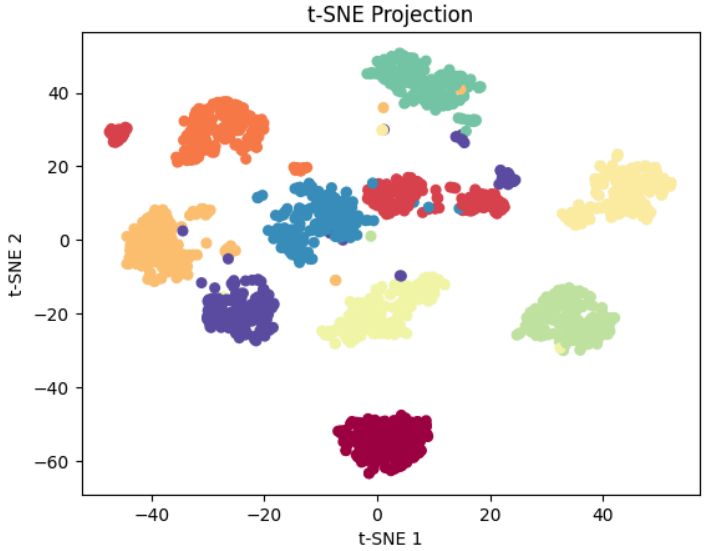
GMM digits dataset results:



[Figure 24a - PCA GMM real dataset]



[Figure 24b UMAP GMM real dataset]



[Figure 24c t-SNE GMM real dataset]

4.3 Agglomerative Hierarchical Clustering

Agglomerative clustering lacks a noise handling mechanism as points may be merged into clusters during the hierarchy-building process. This can be a problem with single linkage where the chain effect may link completely unrelated points based on distance. Moreover, agglomerative clustering requires the completion of data to compute the pairwise distances. Missing values must be addressed beforehand typically by user intervention to ensure reliable clustering outcomes[Alpaydin 1998]. However, luckily the datasets chosen have no missing values therefore this was not an issue highlighted in the discovery of the results(as can be seen by appendices 1).

4.3.1 Synthetic dataset Agglomerative Clustering

Method	Silhouette Score	Davies-Bouldin Index	Adjusted Random Index	Runtime
PCA + AC	0.731	0.396	0.444	0.403 seconds
UMAP + AC	0.953	0.068	0.380	0.178 seconds
TSNE + AC	0.879	0.172	0.380	13.405 seconds

[Figure 25]

AHC showed a strong silhouette score for the synthetic dataset across all dimensional reduction techniques similar to the previous two indicating well separated clusters. The method that stands out the most would be UMAP since it has the highest silhouette score and the lowest DBI score highlighting patterns of compact and well-isolated clustering. This suggests that UMAP can adapt well to this clustering type. t-SNE also performed very well with a high silhouette score and a low DBI showing clusters that are tight and visually distinct(shown in figure 28a).

PCA and AHC still showed reasonable results but showed a weaker approach compared to UMAP and tSNE with a higher DBI. But interestingly it had reported the highest accuracy in the adjusted random index which may be due to PCA keeping the global structure of the

dataset closer to its original labels. Whereas the other reduction mechanisms (UMAP and t-SNE) distort the global relationships as they are misaligned with the true labels.

The runtime for the AHC algorithm was incredibly tedious. Therefore this should be taken into consideration when modelling real world data as the computational output may cause incredibly expensive overheads.

4.3.2 Real dataset Agglomerative Hierarchical Clustering

Method	Silhouette Score	Davies-Bouldin Index	Adjusted Random Index	Runtime
PCA + AC	0.269	0.338	0.736	0.270 seconds
UMAP + AC	0.713	0.338	0.421	0.151 seconds
TSNE + AC	-0.027	1.689	0.270	0.153 seconds

[Figure 26]

PCA had shown patterned behaviour of achieving a higher ARI showing that the scaled data shows higher accordance with the ground truth. But the silhouette score shows poor clustering with clusters that are not well separated in the reduced dimensional space. Therefore it can be suggested whilst it provides accuracy it does not provide compact clusters.

t-SNE provides a visually clear and compact cluster which can be seen by figure 28b with minimal overlap being shown. The silhouette score shows a negative score(-0.02) which represents poor internal cohesion and the ARI has shown to be low. Which shows that there is a gap between visualisation and metric performance which highlights a known issue with t-SNE: it emphasises local separation for visualisation but distances are not always meaningful for hierarchical linkage.

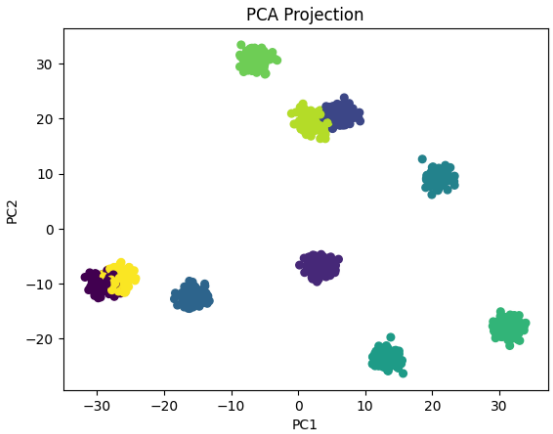
UMAP offers the best alternative by achieving the highest silhouette score showing clear intra-cluster compactness but its ARI is lower than PCA for reasons spoken about previously. UMAP has shown the lowest DBI across dimension reduction methods supporting that clusters are compact and distinct.

4.3.3 Agglomerative Clustering Comparative Overview

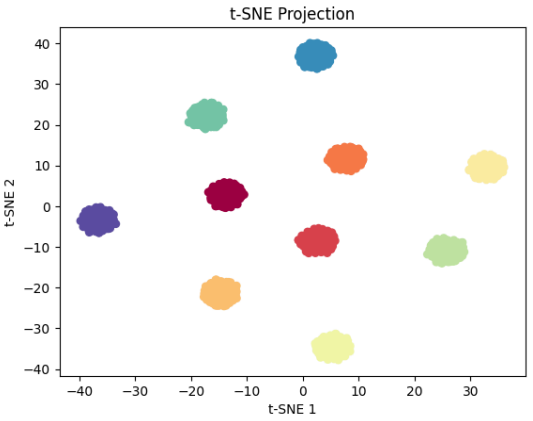
As expected UMAP had performed most consistently as the optimal data reduction method both with the use of synthetic data and real data. Whereas t-SNE had shown the greatest variation in the case of both datasets with a huge deterioration in its performance even showing lower cohesion. t-SNE gives visual groupings but has failed to translate into a reliable metric for the AHC.

UMAP provided the best structural separation in all 3 cases of dimensional reduction methods for the AHC and had proved to be most effective in the context.

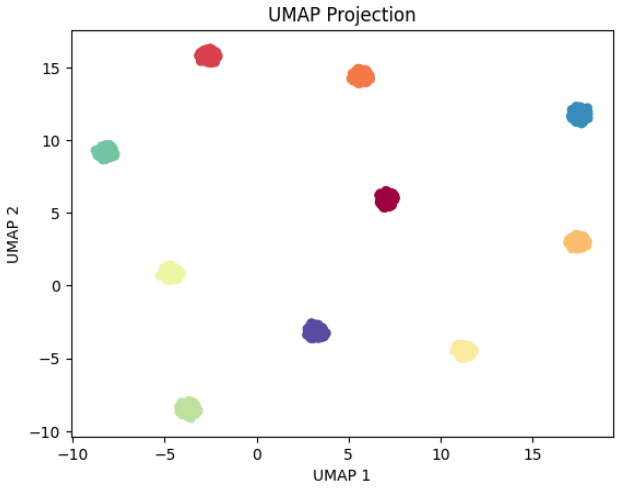
AHC Fake Results:



[Figure 27a PCA AHC fake data]

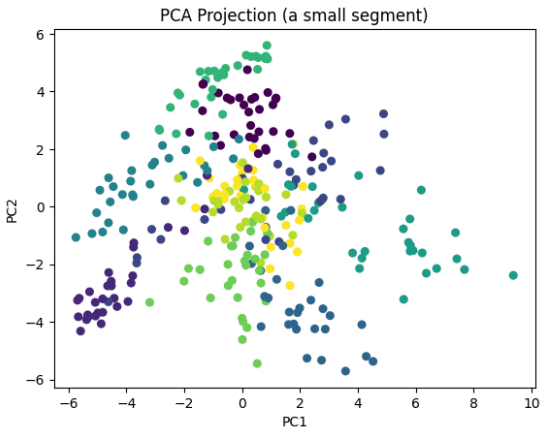


[Figure 27b t-SNE fake data]

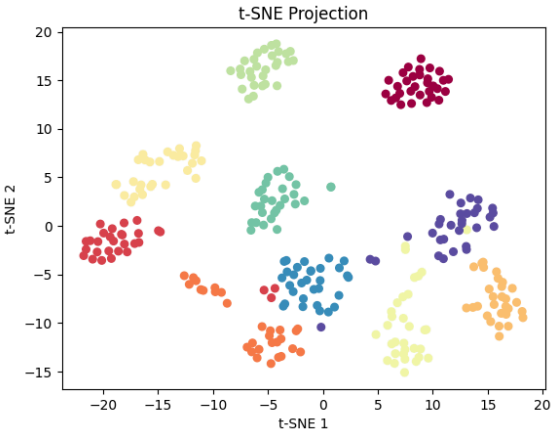


[Figure 27c UMAP fake data]

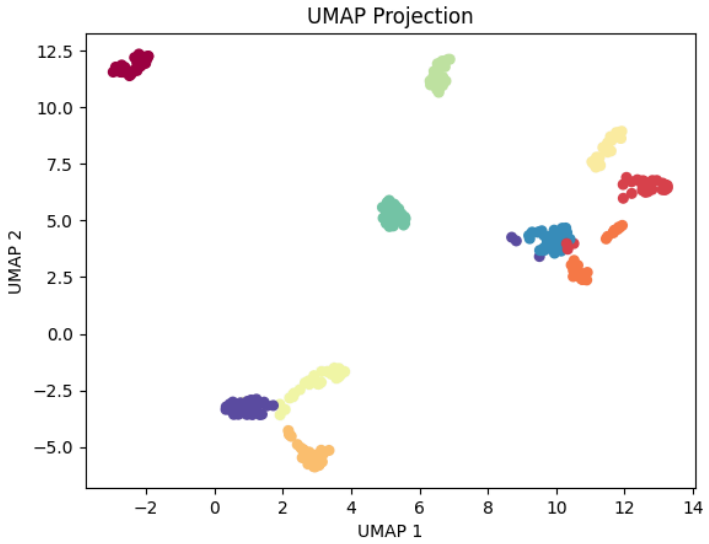
AHC Real Data:



[Figure 28a PCA Real data]



[Figure 28b t-SNE real data]



[Figure 28c UMAP real data]

4.3.4 Comparative review of GMM, KMeans and AHC

In summary, the optimal pairing of the clustering algorithm and dimensionality reduction methods is highly dataset dependent. UMAP offers consistent support through the different clustering algorithms recording reliable data throughout with visualisations showing compact cluster formations. PCA offers stability and alignment in label sensitive metrics for the hierarchical approaches and tSNE whilst powerful for visualisation may not provide the best embedding space for algorithmic clustering especially in distance sensitive methods like AHC.

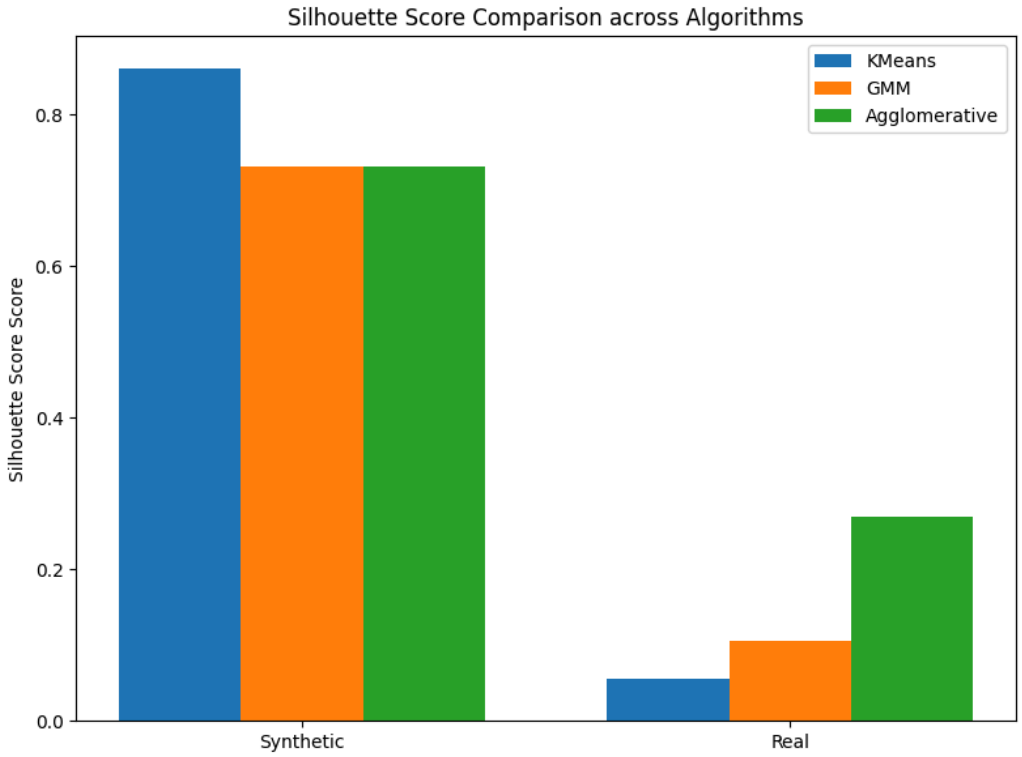
GMM had an improved silhouette score and a lower DBI with the use of UMAP but the ARI was lower compared to KMeans, this can suggest that due to GMM's soft assignment it captured broader overlaps at the cost of hard-label alignment.

t-SNE had offered visually appealing graphs with great separation for all the clustering algorithms modelled but demonstrated limitations in its projections as it did not show better clustering metric results for hierarchical methods.

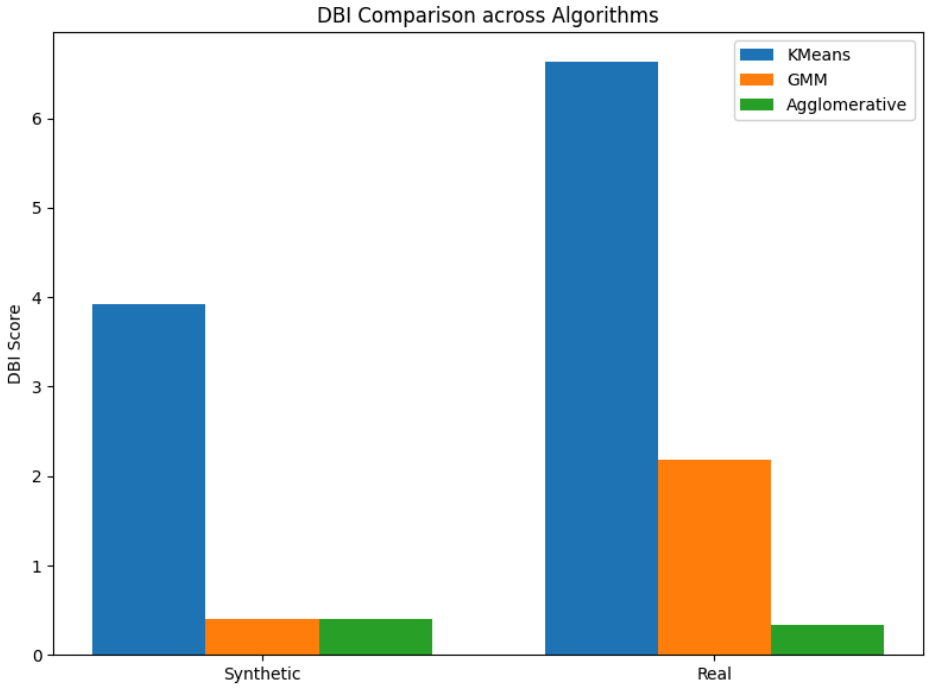
AHC showcased the highest level of variations within its dataset with PCA outperforming non-linear methods in its ARI due to its ability to preserve the global structure even when the clusters were less geometrically distinct.

All the datasets showcased better results for the synthetic datasets in comparison to the real datasets which was expected. KMeans on the synthetic dataset showed similar patterns under UMAP and tSNE, with tSNE offering a slightly lower silhouette scores but still outperforming PCA. PCA had struggled with the dataset as it was capturing non-linear data in its own constricted environment which had caused a deterioration in the system which therefore had a knockover effect into things such as the recorded metrics for its silhouette score, ARI and DBI.

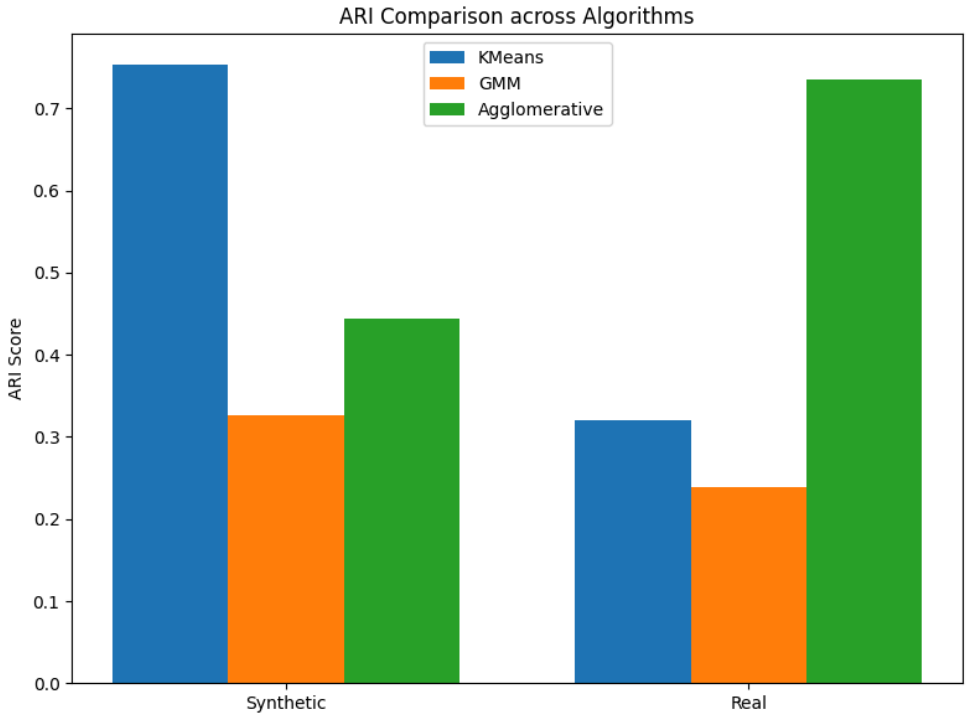
4.3.4.1 Comparative overview graph PCA performance



[Figure 29 Silhouette Score - synthetic data vs real data]

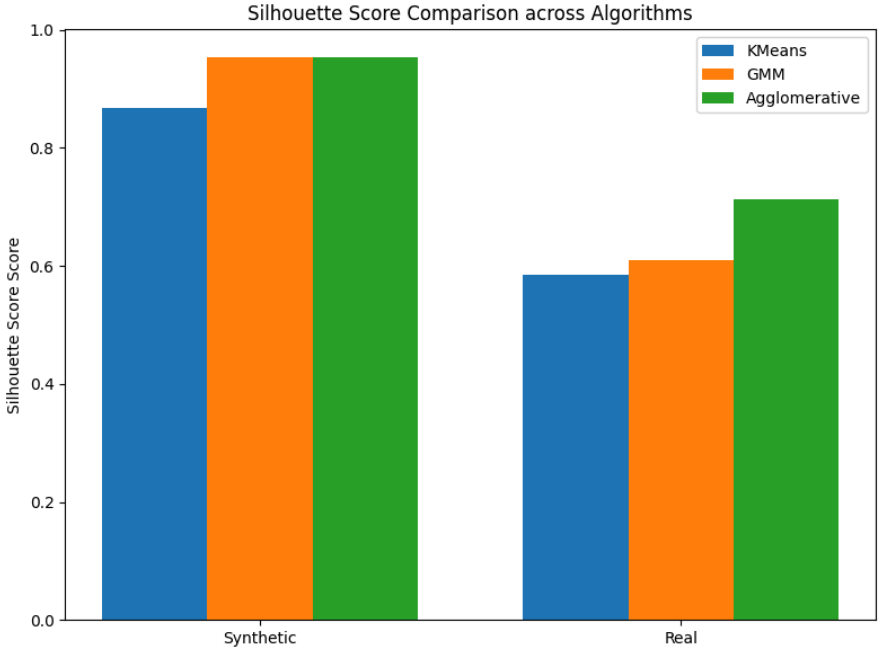


[Figure 30 DBI - synthetic data vs real data]

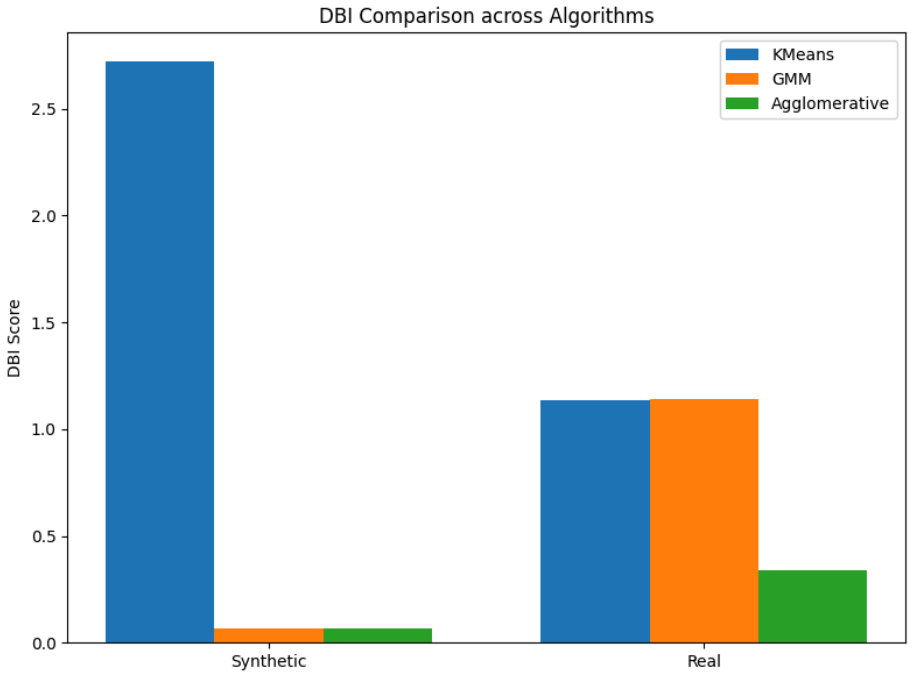


[Figure 31 ARI score synthetic vs real data]

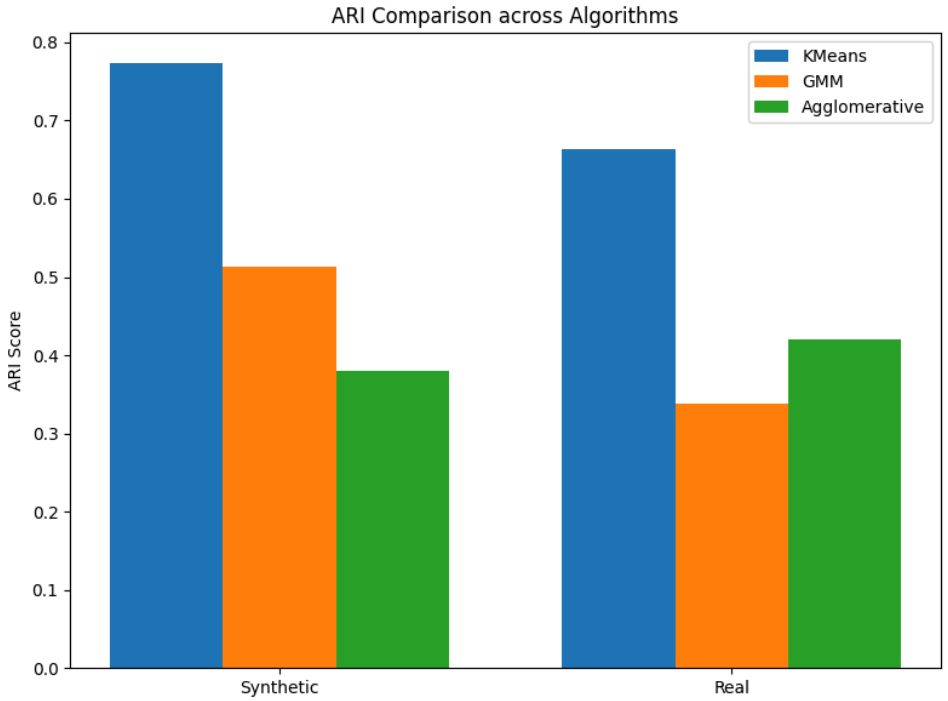
4.3.4.2 Comparative overview graph UMAP performance



[Figure 32 - silhouette score UMAP synthetic vs real data]

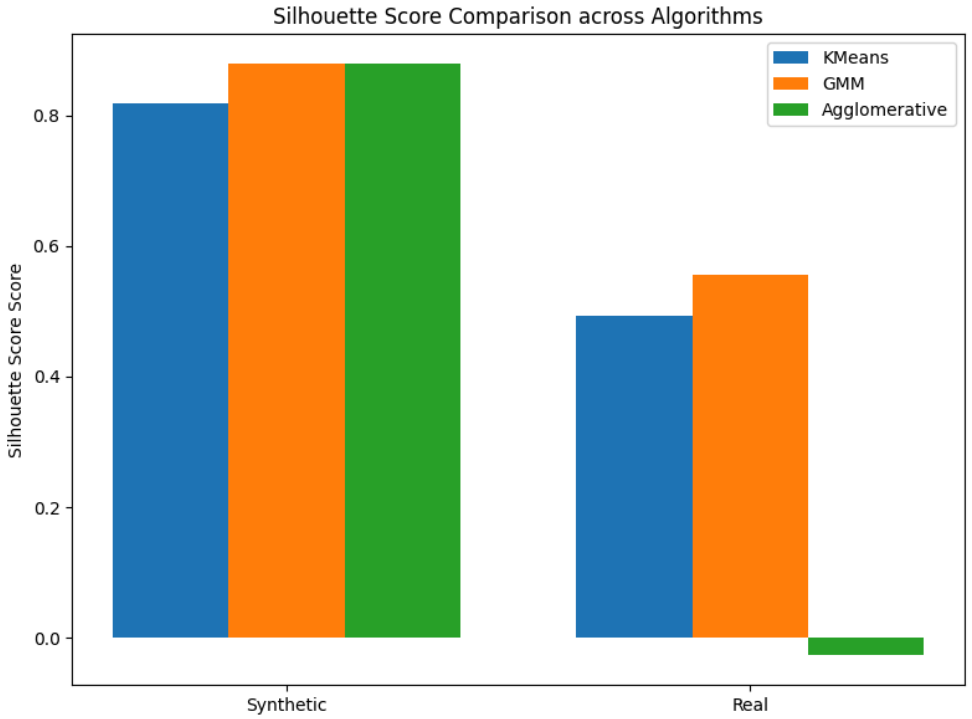


[Figure 33 DBI score synthetic vs real data]

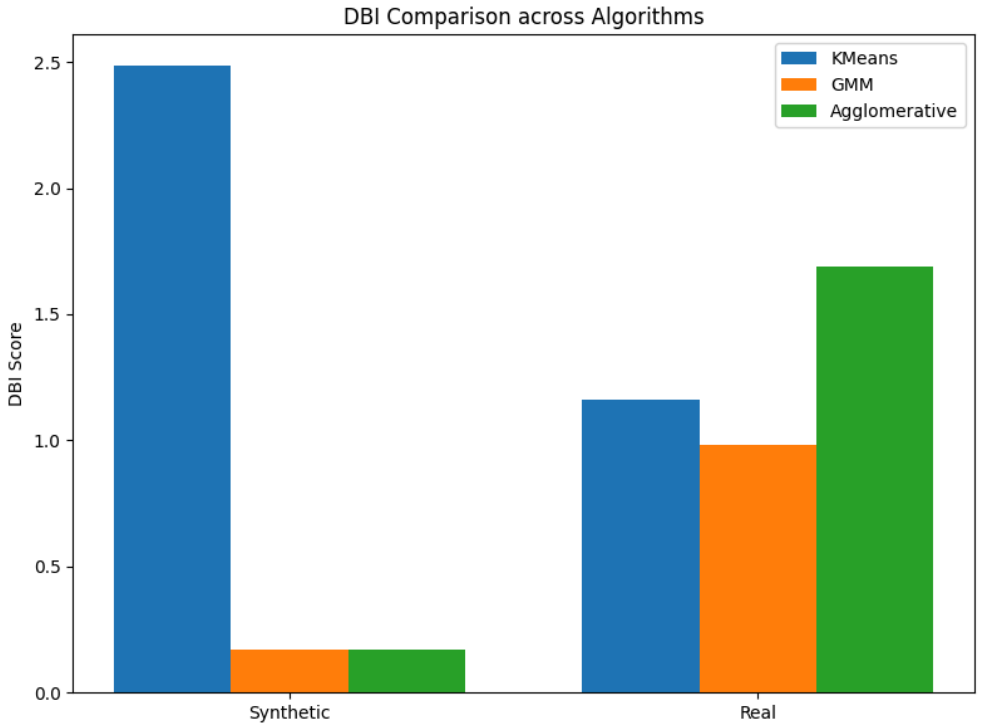


[Figure 34 ARI synthetic vs real data]

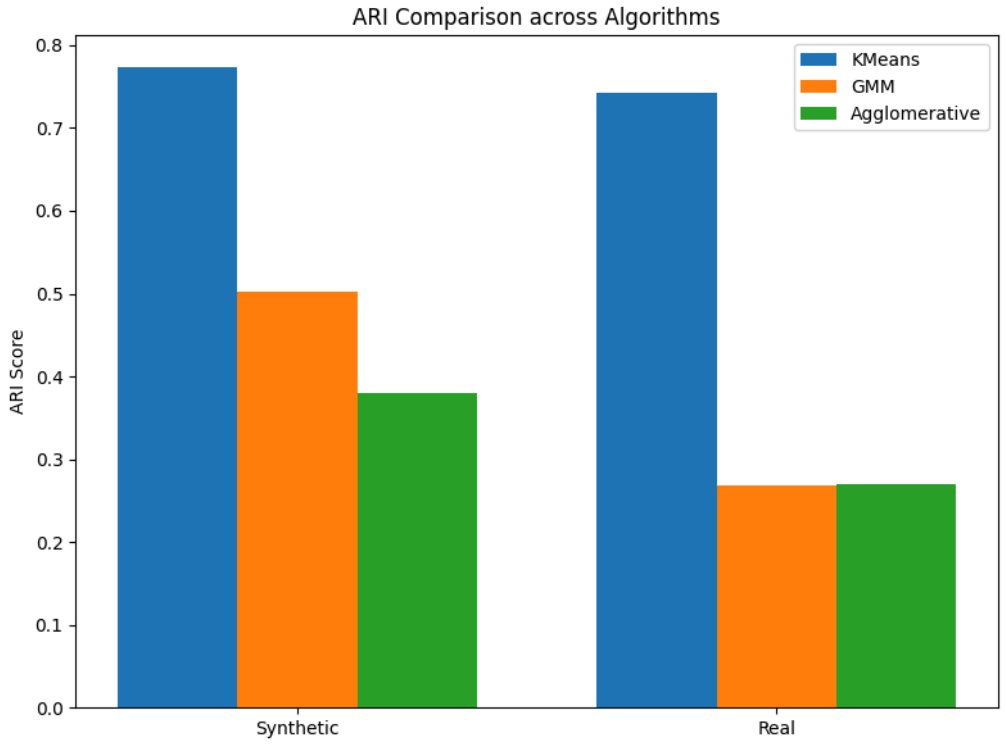
4.3.4.3 Comparative overview graph t-SNE performance



[Figure 35 silhouette score t-SNE synthetic vs real data]



[Figure 36 DBI tSNE synthetic vs real data]



[Figure 37 ARI tSNE synthetic vs real data]

4.4 Spectral Clustering

Spectral clustering(SC) can group non-linear data structures well unlike other techniques explored earlier. This section will give a direct overview of the effectiveness of spectral clustering and compare it to structures which capture only linear forms of data.

Synthetic data

Method	Silhouette Score	Davies-Bouldin Index	Adjusted Random Index	Runtime
PCA + SC	0.688	0.541	0.864	0.485 seconds
UMAP + SC	0.559	0.847	0.671	12.662 seconds
tSNE + SC	0.519	0.839	0.671	21.656 seconds

[Figure 38]

Spectral clustering had shown an overall preference for PCA showing a better performance with a higher recorded level of ground level truth assignment compared to UMAP and tSNE. Which is contrary to the previous methods discussed. UMAP and tSNE had performed similar to one another on the synthetic dataset demonstrating better similar clustering with average truth assignment but had shown higher DBI scores and therefore lower silhouette scores.

Real Data

Method	Silhouette Score	Davies-Bouldin Index	Adjusted Random Index	Runtime
PCA + SC	0.394	0.798	0.395	0.963 seconds
UMAP + SC	0.707	0.410	0.757	35.902 seconds
tSNE + SC	0.652	0.488	0.882	13.384 seconds

[Figure 39]

As expected PCA had shown on the real dataset it had deteriorated in performance levels reaching 0.394 which is low for its silhouette score showing its poor assignments to boundaries and its inability to decide where the clusters lie which is further supported by figure 41a.

On the other hand UMAP and tSNE had shown better results on real data compared to the synthetic data which may be due to make_blobs being linearly designed and the digits dataset being highly non-linear in its structure. UMAP and tSNE are better at handling non-linear data therefore the digits dataset helps optimise its performance.

UMAP had shown a performance increase with a higher recorded silhouette score for both the UMAP and tSNE models and recorded a reduction by just over a half in the DBI scores(from 0.84 in the synthetic dataset to 0.41 in real data). The ground truth had shown a slight performance increase as a result of the non-linear alignment of spectral clustering data structure that it is currently dealing with.

4.4.1 Spectral clustering against linear structures

As has been discussed, spectral clustering is an example of a non-linear data structure. In this section we will be discussing the key comparisons between spectral clustering and the way it models its clusters with a traditional non-linear method(KMeans) highlighting its key areas and how effective it is at identifying clusters.

Firstly, spectral clustering makes no assumption regarding the shape of the data structure therefore it can work with any data structure unlike KMeans which is restricted to a convex and spherically shaped data which can be clustered in a euclidean space. KMeans relies on its centroids to cluster its data therefore any sensitivity to data can impact and deteriorate the accuracy and can pull away the centroid from its true centroid point(where it should be). It often fails when trying to map on non-linear separable clusters. To counter this, kernel k-means would be advisable which runs data in high dimensional spaces[Dhillon & Guan et al 2004]. This instead of picking the initial centroids in an arbitrary fashion, it computes the kernel matrix of a pairwise value and then initialises the clusters.

Whereas spectral clustering depends on the similarity graph built, if the graph is well tuned the noise will have less of an impact on the cluster structure. KMeans is prone to mislabelling data compared to spectral clustering. Spectral clustering depends on the build and if the graph connects noise strongly together it can distort the eigenvector having an impact on the overall performance. Bottlenecks in large datasets for spectral clustering can be computationally expensive as the eigen decomposition can be in the worst case $O(N^3)$ or $O(N^2)$ due to the similarity matrix[Tremblay and Puy et al 2016].

As can be seen by Figure 15 and 16, the KMeans shows a huge dive in its performance in the adjusted random index which when compared to spectral clustering(by figures 38 and 39) had shown a more stable model in its tSNE and UMAP with higher score values when using real world data.

Figure 19a and 41a show the difference between the clustering methodologies where they both show overlap however KMeans objectively shows a worse performance with lower intra-clustering and inter-clustering distance. Whilst figure 19c and 41c showcase spectral clustering and how it handles the data better with less overlap with only a few data points as outliers in comparison to figure 19c.

Furthermore, according to research KMeans has shown to be more scalable and evenly distribute the clustering sizes. Whereas, spectral clustering uses a function which will customise to the similarity measures required which provide it with increased flexibility which means that the overall performance of the model is dependent on the function chosen.

4.4.1 Non-linear vs linear clustering algorithms

This demonstration used the make_moons and the make_blobs library to showcase non-linear points.

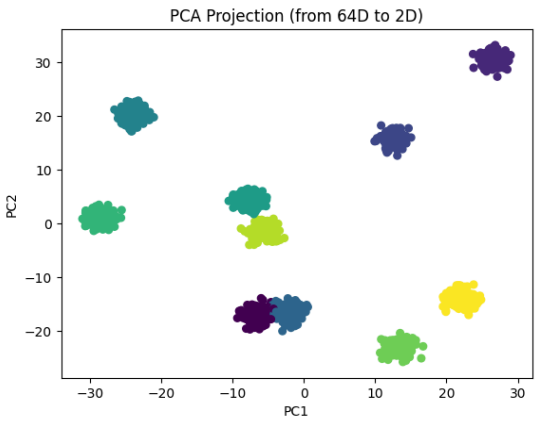
Appendix 2 shows how KMeans would naturally cluster KMeans data in a linear formation. It shows that it is well separated in a euclidean space within a KMeans environment as well as similar sizing and densities.

Appendices 3 shows the flaws of KMeans where it identifies the non-linear cluster boundaries in the two moon datasets. Despite having two distinct groups the algorithm partitioned the boundaries using a distance based method (euclidean distance). This caused a misclassification as can be seen where less than a quarter of a cluster has been wrongly identified for being "closer" to cluster 1 whereas it belonged to its cluster 2.

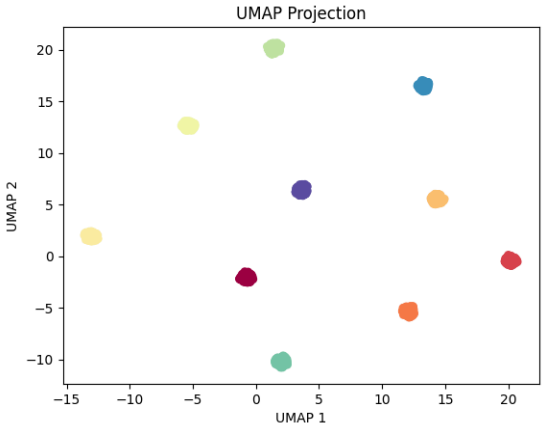
In contrast, appendices 4 showcases a successful separation of the same non-linear dataset where each cluster is actively identified. This is because spectral clustering has captured the relations between points as discussed before. It then applied a laplacian eigen-decomposition to map the data into a new space where clusters become linearly separable. From this a KMeans structure was used and provided better accuracy for complex datasets such as make_moons. Spectral clustering had actively captured the underlying graph as seen by appendices 4 however took longer to compute.

The choice of the algorithm should be guided by the geometry of the dataset and the trade-off between speed and flexibility. This is crucial for which algorithm to choose for efficiency and linear data KMeans would be ideal however non-linear data can present more complex problems and may introduce slower speeds.

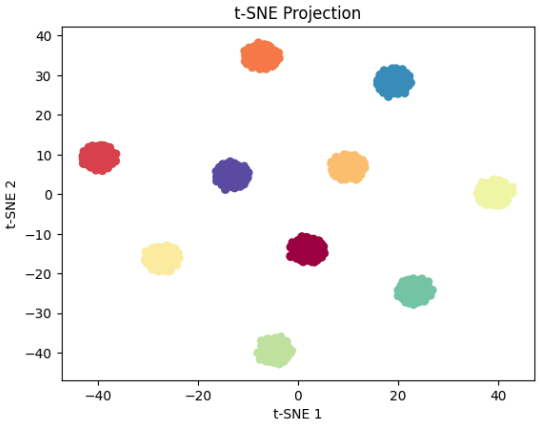
Spectral clustering synthetic data results:



[Figure 40a PCA spectral clustering synthetic]

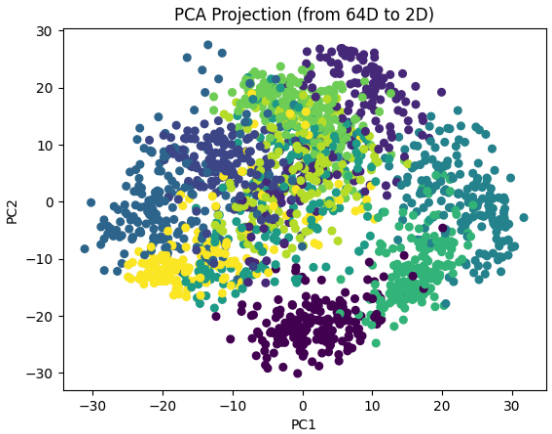


[Figure 40b UMAP spectral clustering synthetic]

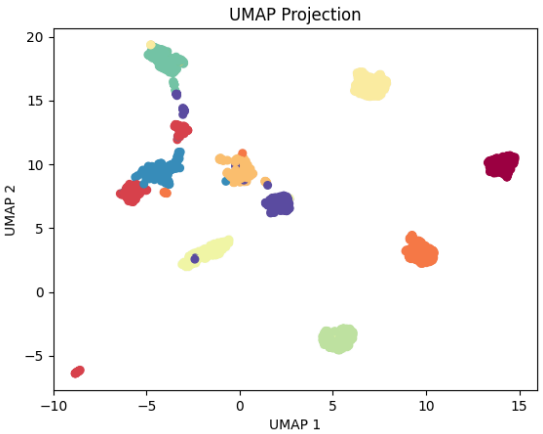


[Figure 40c t-SNE spectral clustering synthetic]

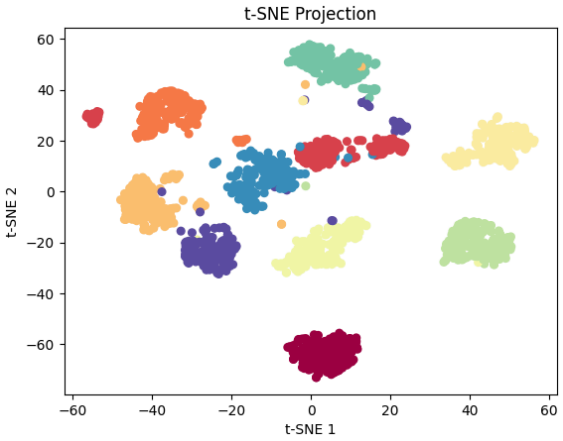
Spectral clustering real data results:



[Figure 41a UMAP spectral clustering real]



[Figure 41b t-SNE spectral clustering real]



[Figure 41c t-SNE spectral clustering real]

5 Future works

Extensive research can be done in semi-supervised learning methodologies which would play a crucial role in gaining a holistic view of clustering methodologies altogether as so far the focus has predominantly been on unsupervised learning methods. However, due to time constraints this was not possible to fulfill. For this, the dataset I would have used would be the load_digits dataset to then be able to make cross comparisons between the previous methods used as well as using the load_digits dataset and plot the outcome using matplotlib and then the metrics would help to identify key information surrounding the clusters developed.

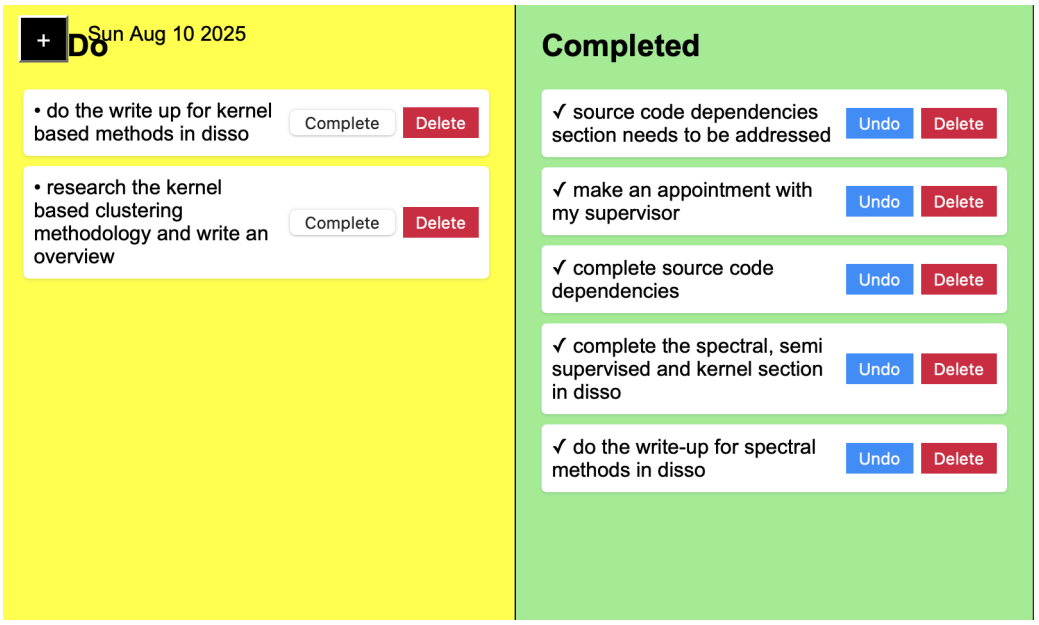
Moreover, I would have used normalised mutual information(NMI) which is a type of measurement which measures how much information the cluster shares with the ground truth. This is different from Adjusted random Index as this looks at points counting measures and sees the agreement levels within the cluster for the predicted and its true labels. This would have given more context behind the data itself and not just the structure/cluster which ARI focused on.

Dimensionality reduction had been a key issue which the clustering had faced. tSNE and UMAP were both written as library imports, a future work of mine would be to use non library based strategies instead. But due to technical difficulty and time limitations this had been an area of weakness and the decision was made to use library methods for dimensional reduction methods. Initially, the aim was to use non-library based methods throughout except for the datasets.

6 Self assessment

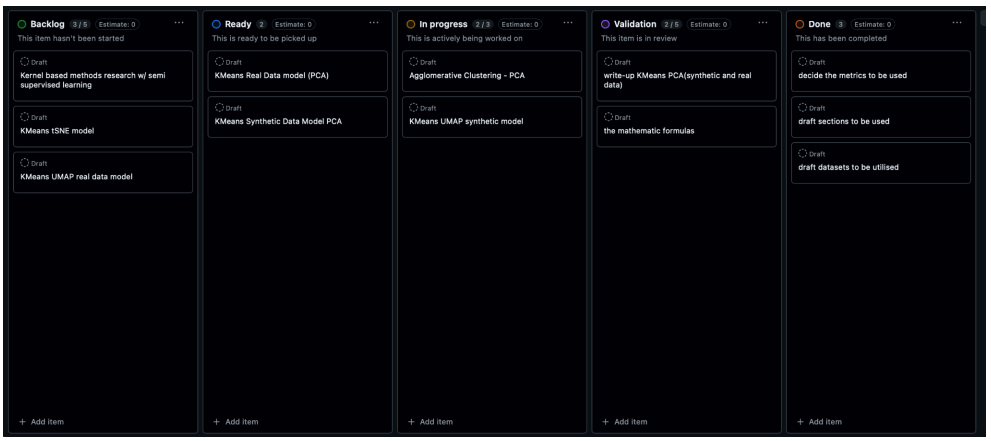
6.1 Strength

Organisation and project management skills: Utilised a "to do" website, which allowed me to keep track of things that have been completed and things yet to do(G Dhillon 2025). This helped to identify key areas where the work needed to be done and served as a checkpoint for identifying how much progress has been made during the week(or that particular day in some scenarios).



[Figure 42 ToDo list G Dhillon]

Furthermore, Github kanban had helped to identify key areas of development and was used to see the process stages entirely and where current stages were at in its development.



[Figure 43 extracted from Github Kanban]

Resource management: with thanks to the Linux subserver provided by the university(please see section 3.2.3 for more information). Jupyter Notebook was a crucial tool that had required minimal local setup or manual library installation and had allowed for full focus on the experiment rather than the environment configuration.

6.2 Weakness

Honing in on deep learning(generative modelling): Within the theory section VAE and GANs models were briefly discussed however this section could have been built upon by doing experimental studies on this particular area of generative models. Leveraging GPU-accelerated training and large scale datasets would allow for experimentation with architectures that go beyond the scope of traditional methods explored within the study. Direction for the generative modelling segment in particular had shown to be broad in my original project plan and if I was to redo it all again, I would research with more emphasis in the generative modelling strategy to use before deciding the aims and objectives.

Dataset diversity: The use of the load_digits() dataset had restricted the generalisability to datasets with a small set of domains (as load_digits has 10 classes) and this restricts the generalisability to other data types including text, audio or large scale unstructured data.

6.3 Opportunities

Dataset generalisability: The work completed lays groundwork for future exploration in several areas however, broadening the real world datasets used would present benefits in many domains including healthcare, finance, natural language processing and many other areas which can enhance the generalisability of the study.

Future use of the model in real world scenarios: Using the results of the experiment this can be considered for future use when modelling real world data such as natural language processing or character recognition. Furthermore, similar experiments can be taken on to use different datasets for a similar purpose making this a flexible model to use for future use.

Combining algorithms: Leading on from what has been discussed, the use of a combining clustering algorithm could have been used as a technique to see if it outperforms individual models. This would stem further research into the most optimal algorithm to combine and the areas of tradeoff.

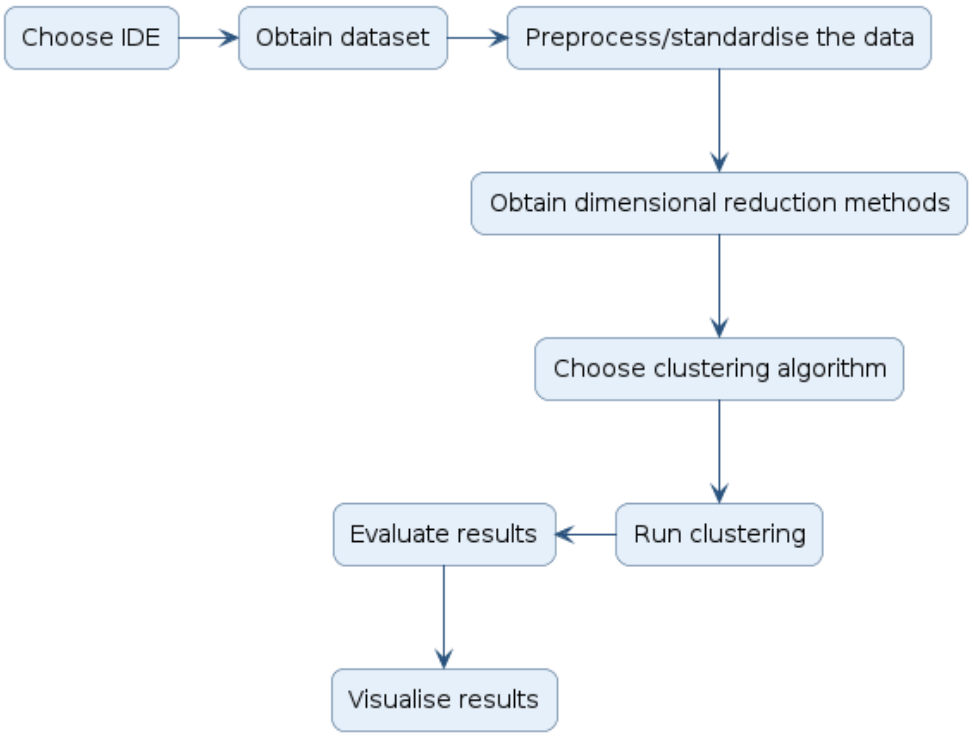
6.4 Threats

Changes within the dataset availability can prevent me from producing the results. This would be particularly difficult however luckily it was not the case at the time of submission and other datasets were kept in mind as a spare throughout the project process. This can be applied to all other libraries however all the libraries used were used sparingly and had a low risk of removal.

Otherwise, the main obstacle faced was time constraint due to technical difficulty during the implementation process that had caused progress issues during the process of development.

7 How to use my project

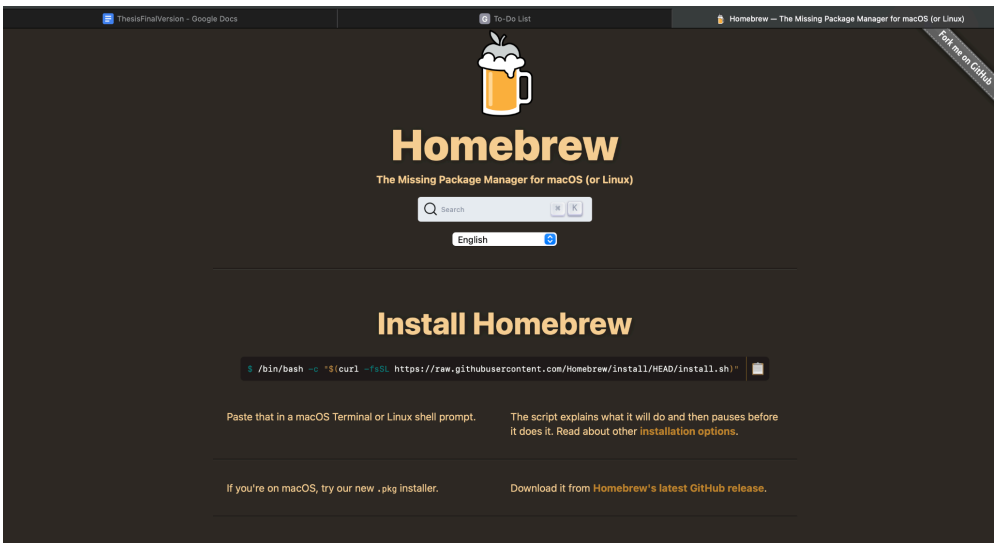
The flow of how to run the project is shown below. The recommended IDE(integrated development environment) is Jupyter Notebook as there are fewer installation requirements(please visit section 2.2.4 for more details). It is also recommended to use one file per programming algorithm and modularise the code accordingly. For example synthetic and real data should run in separate files.



[Figure 44 extracted from Plantuml(self-made)]

7.2 Replicating the environment

1. Install homebrew(This is a package manager for macOS)



[Figure 45 - homebrew website photo]

Click on the code block and copy it into a terminal of your choice

2. Install the updates if necessary

```
brew update
```

3. Install python:

```
brew install python
```

4. Verify the versions using:

```
python3 --version

pip3 --version
```

5. Install jupyter notebook

```
pip3 install notebook

pip3 install notebook
```

6. Run Jupyter Notebook

```
jupyter notebook
```

This command should automatically open up a http://localhost:8888 server in the browser with the jupyter notebook user interface.

References

[1] Xu, R. and Wunsch, D., 2008. Clustering. John Wiley & Sons.

[2] Jardine, N. and van Rijsbergen, C.J., 1971. The use of hierarchic clustering in information retrieval. *Information storage and retrieval*, *7*(5), pp.217-240.

[3]Jain, A.K., 2010. Data clustering: 50 years beyond K-means. *Pattern recognition letters*, *31*(8), pp.651-666.

[4] Steinley, D., 2006. K-means clustering: a half-century synthesis. *British Journal of Mathematical and Statistical Psychology*, *59*(1), pp.1-34.

[5] Hastie, T., Tibshirani, R., Friedman, J.H. and Friedman, J.H., 2009. *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2, pp. 1-758). New York: Springer.

[6] Reynolds, D., 2015. Gaussian mixture models. In *Encyclopedia of biometrics* (pp. 827-832). Springer, Boston, MA.

[7] Bishop, C.M. (2006) Pattern recognition and machine learning. New York: Springer.

[8] Murphy, K.P., 2012. *Machine learning: a probabilistic perspective*. MIT press.

[9] Murtagh, F. and Contreras, P., 2012. Algorithms for hierarchical clustering: an overview. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, *2*(1), pp.86-97.

[10] Day, W.H. and Edelsbrunner, H., 1984. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, *1*(1), pp.7-24.

[11] Jain, A.K., Murty, M.N. and Flynn, P.J., 1999. Data clustering: a review. *ACM computing surveys (CSUR)*, *31*(3), pp.264-323.

[12] MacQueen, J., 1967, January. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics* (Vol. 5, pp. 281-298). University of California press.

[13] Ward Jr, J.H., 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, *58*(301), pp.236-244.

[14] Kaufman, L. and Rousseeuw, P.J. (1990) Finding groups in data : an introduction to cluster analysis. New York ; Wiley.

[15] Fraley, C. and Raftery, A.E., 2002. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association*, *97*(458), pp.611-631.

[16] Dempster, A.P., Laird, N.M. and Rubin, D.B., 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society: series B (methodological)*, *39*(1), pp.1-22.

[17] Tang, N. (ed.) (2022) Data Clustering. London: IntechOpen. Available at: https://doi.org/10.5772/intechopen.95124.

[18]Dhillon,G. (2023) *Noter*. Available at: https://gurvirdhillon.github.io/art_canvas/

[19] Bilmes, J.A., 1998. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. *International computer science institute*, *4*(510), p.126.

[20] Deng, H. and Han, J., 2018. Probabilistic models for clustering. In *Data Clustering*(pp. 61-86). Chapman and Hall/CRC.

[21] Murphy, K.P. (2014) Machine Learning : A Probabilistic Perspective. Cambridge: MIT Press.

[22] Ververidis, Dimitrios & Kotropoulos, C.. (2008). Gaussian Mixture Modeling by Exploiting the Mahalanobis Distance. Signal Processing, IEEE Transactions on. 56. 2797 - 2811. 10.1109/TSP.2008.917350.

[23] Amigó, E., Gonzalo, J., Artiles, J. and Verdejo, F., 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. Information retrieval, 12(4), pp.461-486.

[24] Davies, D.L. and Bouldin, D.W., 1979. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2), pp.224-227.

[25] Rousseeuw, P.J., 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, *20*, pp.53-65.

[26] Jiang, H. and Arias-Castro, E. (2024) 'K-means and gaussian mixture modeling with a separation constraint', Communications in statistics. Simulation and computation, pp. 1–15. Available at: https://doi.org/10.1080/03610918.2024.2354747.

[27] He, X., Cai, D., Shao, Y., Bao, H. and Han, J., 2010. Laplacian regularized Gaussian mixture model for data clustering. *IEEE transactions on knowledge and data engineering*, *23*(9), pp.1406-1418.

[28] Yang, L., Fan, W. and Bouguila, N., 2020. Clustering analysis via deep generative models with mixture models. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(1), pp.340-350.

[29] Kriegel, H.P., Kröger, P. and Zimek, A., 2009. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *Acm transactions on knowledge discovery from data (tkdd)*, *3*(1), pp.1-58.

[30] Wang, Z. and Ye, C., 2024. Deep Clustering Evaluation: How to Validate Internal Clustering Validation Measures. *arXiv preprint arXiv:2403.14830*.

[31] Ren, Y., Pu, J., Yang, Z., Xu, J., Li, G., Pu, X., Yu, P.S. and He, L., 2024. Deep clustering: A comprehensive survey. *IEEE transactions on neural networks and learning systems*, *36*(4), pp.5858-5878.

[32] Abdi, H. and Williams, L.J., 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, *2*(4), pp.433-459.

[33] Davies, T. and Fearn, T., 2004. Back to basics: the principles of principal component analysis. *Spectroscopy Europe*, *16*(6), p.20.

[34] Maaten, L.V.D. and Hinton, G., 2008. Visualizing data using t-SNE. *Journal of machine learning research*, *9*(Nov), pp.2579-2605.

[35] Arora, S., Hu, W. and Kothari, P.K., 2018, July. An analysis of the t-sne algorithm for data visualization. In *Conference on learning theory* (pp. 1455-1462). PMLR.

[36] Wattenberg, M., Viégas, F. and Johnson, I., 2016. How to use t-SNE effectively. *Distill*, *1*(10), p.e2.

[37] McInnes, L., Healy, J. and Melville, J., 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

[38] Ghojogh, B., Ghodsi, A., Karray, F. and Crowley, M., 2021. Uniform manifold approximation and projection (UMAP) and its variants: tutorial and survey. *arXiv preprint arXiv:2109.02508*.

[39] Becht, E., McInnes, L., Healy, J., Dutertre, C.A., Kwok, I.W., Ng, L.G., Ginhoux, F. and Newell, E.W., 2019. Dimensionality reduction for visualizing single-cell data using UMAP. *Nature biotechnology*, *37*(1), pp.38-44.

[40] Alpaydin (1998) Optical recognition of handwritten digits dataset Available at: https://scikit-learn.org/stable/datasets/toy_dataset.html (Accessed: 1 August 2025).

[41] Von Luxburg, U., 2007. A tutorial on spectral clustering. *Statistics and computing*, *17*(4), pp.395-416.

[42] Ng, A., Jordan, M. and Weiss, Y., 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, *14*.

[43] Liu, J. and Han, J., 2018. Spectral clustering. In *Data clustering* (pp. 177-200). Chapman and Hall/CRC.

[44] Van Engelen, J.E. and Hoos, H.H., 2020. A survey on semi-supervised learning. *Machine learning*, *109*(2), pp.373-440.

[45] Zhou, X. and Belkin, M., 2014. Semi-supervised learning. In *Academic press library in signal processing* (Vol. 1, pp. 1239-1269). Elsevier.

[46] Skabar, A. and Juneja, N., 2007, July. A kernel-based method for semi-supervised learning. In *6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007)*(pp. 112-117). IEEE.

[47] S. Basu, A. Bamerjee and R. Mooney, "Semi-supervised clustering by seeding", Proceedings of the 19th International Conference on Machine Learning, pp.19-26, 2002.

[48] Dhillon, I.S., Guan, Y. and Kulis, B., 2004, August. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 551-556).

[49] Tremblay, N., Puy, G., Gribonval, R. and Vandergheynst, P., 2016, June. Compressive spectral clustering. In *International conference on machine learning* (pp. 1002-1011). PMLR.

[50] Dhillon G. (2025) *To Do.* Available at: (https://gurvirdhillon.github.io/todolist/)

[51] Nellutla, I. (2017). Getting started with Jupyter Notebooks. [online image] Qxf2 BLOG. Available at: https://qxf2.com/blog/getting-started-with-jupyter-notebooks/ [Accessed 20 Aug. 2025].

[52] Müllner, D., 2011. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

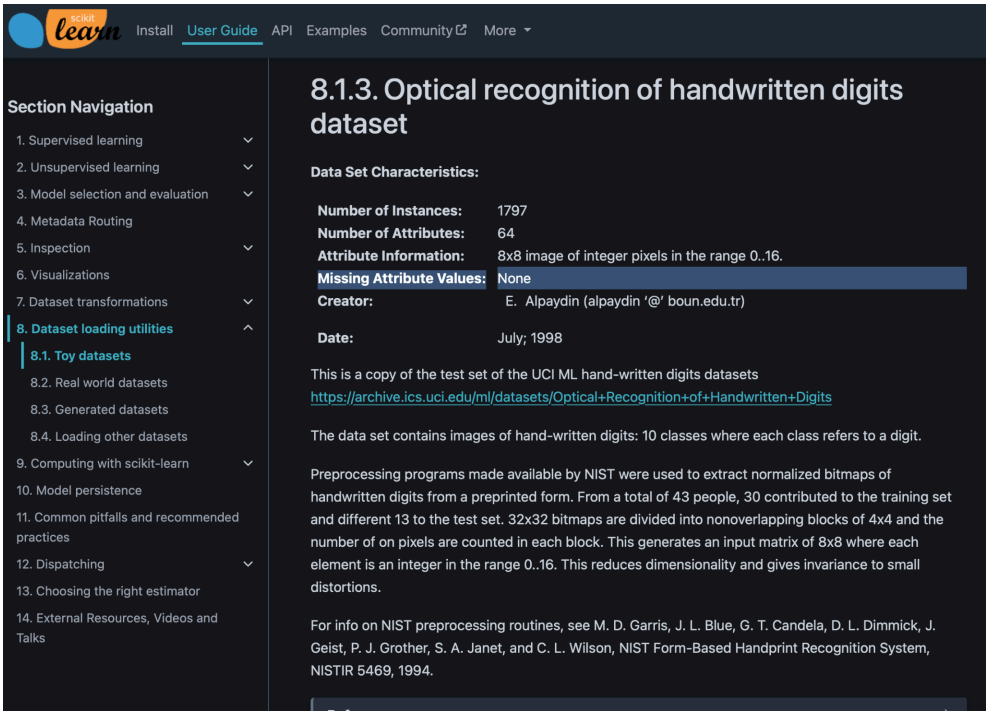
[53] Cohen-Addad, V., Kanade, V., Mallmann-Trenn, F. and Mathieu, C., 2019. Hierarchical clustering: Objective functions and algorithms. *Journal of the ACM (JACM)*, *66*(4), pp.1-42.

[54] Murtagh, F. and Legendre, P., 2011. Ward's hierarchical clustering method: clustering criterion and agglomerative algorithm. *arXiv preprint arXiv:1111.6285*.

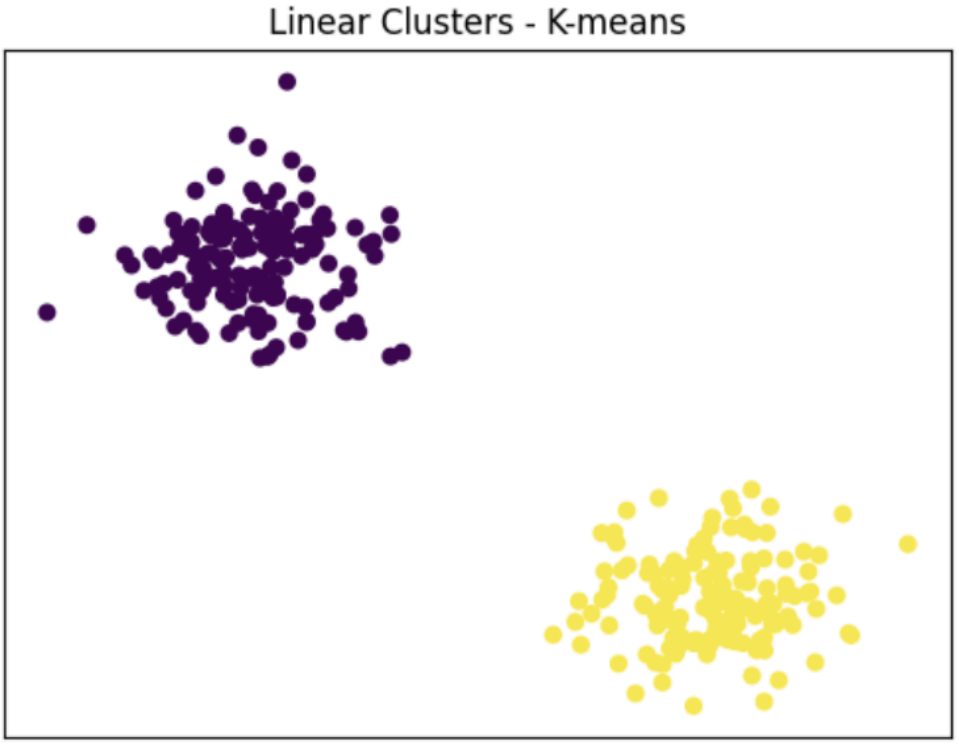
[55] Murtagh, F. and Legendre, P., 2014. Ward's hierarchical agglomerative clustering method: which algorithms implement Ward's criterion?. *Journal of classification*, *31*(3), pp.274-295.

[56] Nanga, S., Bawah, A.T., Acquaye, B.A., Billa, M.I., Baeta, F.D., Odai, N.A., Obeng, S.K. and Nsiah, A.D., 2021. Review of dimension reduction methods. *Journal of Data Analysis and Information Processing*, *9*(3), pp.189-231

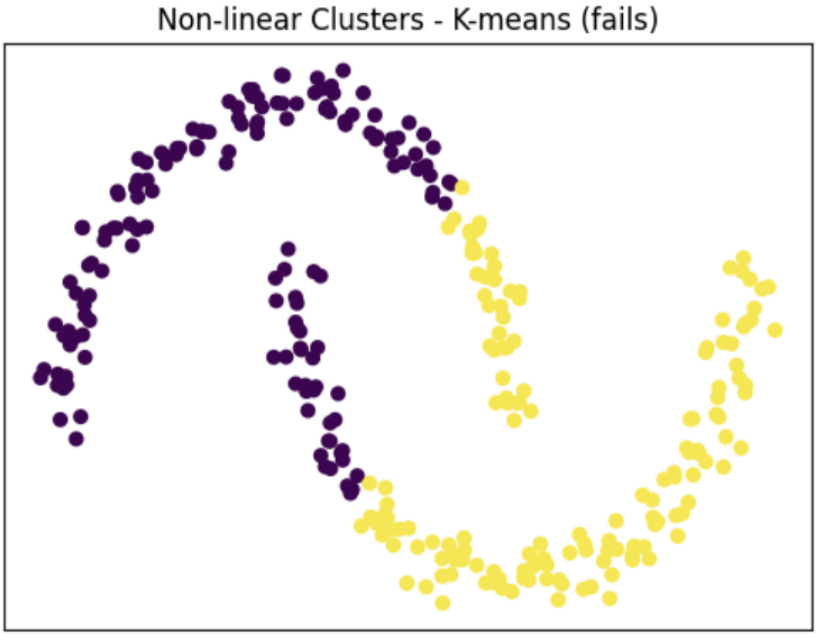
Appendices



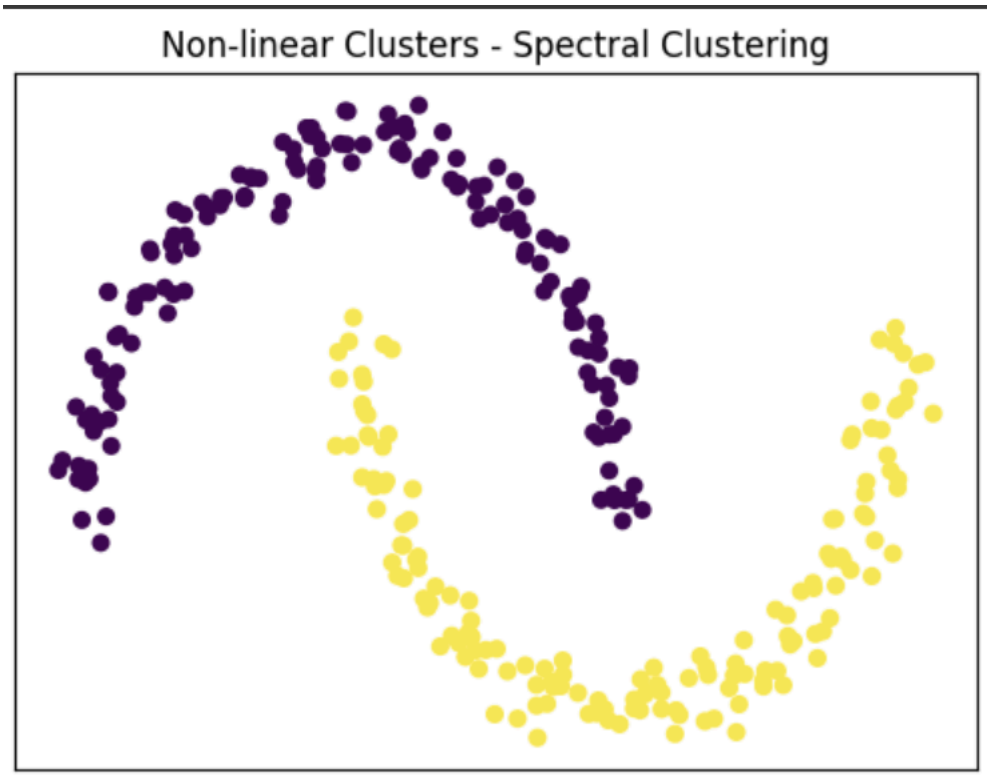
[Appendices 1 handwritten digits No missing values]



[Appendices 2 Linear Clustering KMeans]



[Appendices 3 Non Linear KMeans (attempt) - fail]



[Appendices 4 Non-linear Spectral Clustering]

```
class KMeans:

 def __init__(self, k=5, max_iterations=100, tol=1e-4,
random_state=12):

   # initialises the kmeans, number of k clusters, number of
iterations to run the algorithm

   # tol is the threshold for when the algorithm should stop
iterating
```

```
    self.k = k
    self.max_iterations = max_iterations
    self.tol = tol


 def initialise_centroids(self, x):
    # select k unique data points from x to serve as the initial
centroids
    random_index = np.random.choice(len(x), self.k, replace=False)
    return x[random_index]


 def compute_distance(self, X, centroids):
    # Compute the Euclidean distance between each data point and each
centroid
    # Returns a number of samples
    return np.linalg.norm(X[:, np.newaxis] - centroids, axis=2)


 def assign_clusters(self, distances):
    # assign each data point to the closest centroid with the smallest
distance
    # returns a vecotr of the cluster labels
    return np.argmin(distances, axis=1)


 def update_centroids(self, X, labels):
```

```
  # recalculating the centroid point as the mean of all points in
each clusters
  new_centroids = np.zeros((self.k, X.shape[1]))
  for i in range(self.k):
    cluster_points = X[labels == i]
    if len(cluster_points) > 0:
      new_centroids[i] = cluster_points.mean(axis=0)
  return new_centroids


def fit(self, X):
  # train kmeans model on data x
  # initialise first step
  self.centroids = self.initialise_centroids(X)
  for _ in range(self.max_iterations):
    # compute distance to centroids
      distances = self.compute_distance(X, self.centroids)
      self.labels = self.assign_clusters(distances)
      new_centroids = self.update_centroids(X, self.labels)
      if np.all(np.abs(new_centroids - self.centroids) < self.tol):
          break
      self.centroids = new_centroids
```

```
    # assign points to the nearest cluster, update the centroids
based on new assignments, check for convergence, update the centroids



 def predict(self, X):

   # assign cluster labels to new unseen data points

   distances = self.compute_distance(X, self.centroids)

   return self.assign_clusters(distances)
```

[Appendices 5 KMeans Configurations]

```
class GMM:

 def __init__(self, n_components, max_iterations=100, tol=1e-3):

   # components are the number of gaussian models and iterations are
how many times the algorithm will run

   # tolerance is the threshold for when the algorithm stops
iterating

   self.k = n_components

   self.max_iterations = max_iterations

   self.tol = tol


 def fit(self, x):

     n_samples, n_features = x.shape

     rng = np.random.RandomState(12)

     self.means = x[rng.choice(n_samples, self.k, replace=False)]

     self.covariances = [np.cov(x.T) for _ in range(self.k)]

     self.weights = np.ones(self.k) / self.k


     log_likelihood_old = 0


     for iteration in range(self.max_iterations):

           # E-step: compute responsibilities

         responsibilities = np.zeros((n_samples, self.k))
```

```
        for k in range(self.k):
            rv = multivariate_normal(mean=self.means[k],
cov=self.covariances[k])
            responsibilities[:, k] = self.weights[k] * rv.pdf(x)
        responsibilities /= responsibilities.sum(axis=1,
keepdims=True)


          # M-step: update parameters
        Nk = responsibilities.sum(axis=0)
        self.weights = Nk / n_samples
        self.means = (responsibilities.T @ x) / Nk[:, np.newaxis]
        self.covariances = []
        for k in range(self.k):
            diff = x - self.means[k]
            cov_k = (responsibilities[:, k][:, np.newaxis] * diff).T
@ diff / Nk[k]
            self.covariances.append(cov_k)
          # Check for convergence
        log_likelihood =
np.sum(np.log(responsibilities.sum(axis=1)))
        if np.abs(log_likelihood - log_likelihood_old) < self.tol:
            break
        log_likelihood_old = log_likelihood
```

```
def predict(self, X):

    likelihood = np.zeros((X.shape[0], self.k))

    for k in range(self.k):

        rv = multivariate_normal(mean=self.means[k],
cov=self.covariances[k])

        likelihood[:, k] = self.weights[k] * rv.pdf(X)

    return np.argmax(likelihood, axis=1)

  # this function assigns each point to the highest weighted
likelihood
```

[Appendices 6 GMM Configurations]

```
class AgglomerativeClustering:

    def __init__(self, n_clusters=2):

        self.n_clusters = n_clusters



    def fit(self, X):

        # Start with each point in its own cluster

        n_samples = X.shape[0]

        clusters = [[i] for i in range(n_samples)]

        distances = cdist(X, X)

        np.fill_diagonal(distances, np.inf)



        while len(clusters) > self.n_clusters:

            # Find closest pair of clusters

            min_dist = np.inf

            to_merge = (0, 1)

            for i in range(len(clusters)):

                for j in range(i+1, len(clusters)):

                    d = self._linkage(clusters[i], clusters[j], distances)

                    if d < min_dist:

                        min_dist = d

                        to_merge = (i, j)
```

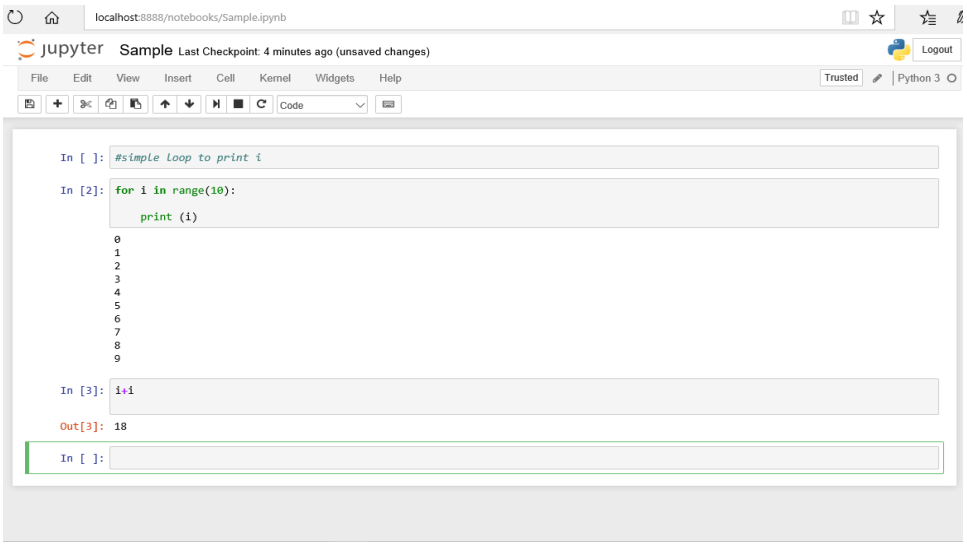
```
            # Merge clusters
            i, j = to_merge
            clusters[i] += clusters[j]
            del clusters[j]

        self.labels_ = np.zeros(n_samples, dtype=int)
        for cluster_id, cluster in enumerate(clusters):
            for idx in cluster:
                self.labels_[idx] = cluster_id

        return self

    def _linkage(self, cluster1, cluster2, distances):
        return min(distances[i, j] for i in cluster1 for j in cluster2)
```

[Appendices 7 Agglomerative Clustering Configurations]

```
def Spectral_Clustering(X, n_clusters=3, gamma=1.0):

 # computes squared euclidean distance matrix sq_dists[i, j] = ||X[i]
       - X[j]||^2

 sq_dists = np.sum(X**2, axis=1)[:, None] + np.sum(X**2, axis=1)[None,
       :] - 2 * np.dot(X, X.T)



 # convert distance into similarities using RBF kernel



 W = np.exp(-gamma * sq_dists)

 np.fill_diagonal(W, 0)

 # set self similarities to 0  so it doesnt self loop



 D = np.diag(np.sum(W, axis=1))

 # compute the degree matrix

 L = D - W

 eigenvals, eigenvecs = np.linalg.eigh(L)

 # compute the eigenvalues and eigenvecs of the laplacian

 idx = np.argsort(eigenvals)[:n_clusters]

 # choose eigenvectors corresponding to the smallest n_clusters
       eigenvalues

 H = eigenvecs[:, idx]

 H_norm = H / np.linalg.norm(H, axis=1, keepdims=True)

 # normalise h rows to unit length
```

```
 # then apply the kmeans algorithm

 kmeans = KMeans(k=n_clusters, max_iterations=100, tol=1e-4,
       random_state=12)

 kmeans.fit(H_norm)

 labels = kmeans.predict(H_norm)



 return labels

 # return final assignments

X, y = make_blobs(n_samples=1797, centers=10, n_features=64,
       random_state=19)

labels_syn = Spectral_Clustering(X_syn, n_clusters=10)
```

[Appendices 8 Spectral Clustering Configurations]



[Appendices 9 Jupyter Notebook User Interface]