**DS Grad Programme 6 (Machine Learning – Part Two) –   
Unsupervised Learning**

What is Unsupervised Learning?

* Type of machine learning where algorithms are trained on unlabelled data.
* Aims to find patterns, structures, or relationship in the data without predefined labels.
* Identify patterns, clusters, or relationships within data to provide valuable insights

Why use it?

* Plays an important role particularly in scenarios where labelled data is difficult to get or expensive to obtain
* Many real-world datasets are unlabelled
* Can be the perfect starting point to obtain a small subset of possible labels to inform supervised learning

Applications

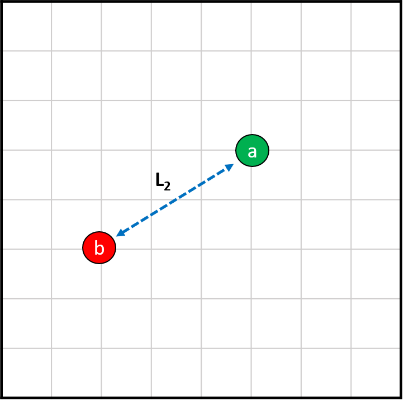
* Customer segmentation
* Anomaly detection
* Recommender systems
* Medical imaging

Main methods of unsupervised learning

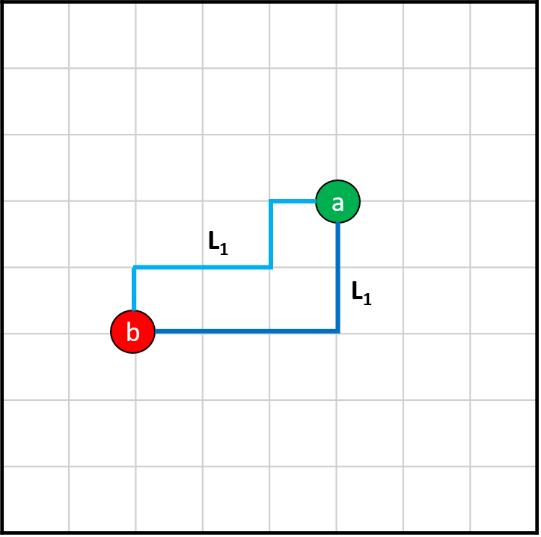
* Clustering
* Association (common in retail/finance)
* Dimensionality Reduction

Measuring distance

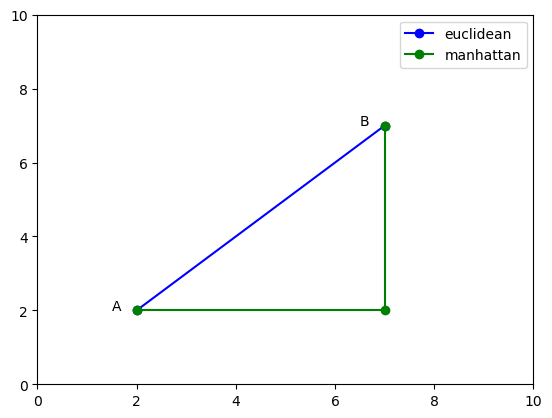
* Distance can be quantified in different ways, but the main two used to quantify clustering are:
  + Euclidean distance
    - The distance in a straight line measurement in “euclidean space”.
    - We can calculate the euclidean distance by finding the square root of the sum of squared differences in each of the dimensions/coordinates of the data.
    - The euclidean distance is often referred to as the L2 distance or “norm”.



* + Manhattan distance
    - Takes it’s name from the street grid layout of Manhattan in New York, as well as many other cities.
    - The intuition for this method is that distance travelled cannot always be described by the shortest path in space, but rather has to follow grid lines.
    - The Manhattan distance is also called the Taxicab metric or snake distance and we will use the notation L1.
    - There can often be more than one path between two points which have the same Manhattan Distance.
    - The distance is calculated as the sum of lengths between two points in each dimension. This uses the absolute value of the lengths, rather than the squared sum, it is given by:



* + Minkowski distance
    - We can generalise the L1 and L2 norms into one equation used for measuring distances. This is known as the Minkowski distance, where we have the parameter p, which specifies the type of distance used.



Clustering

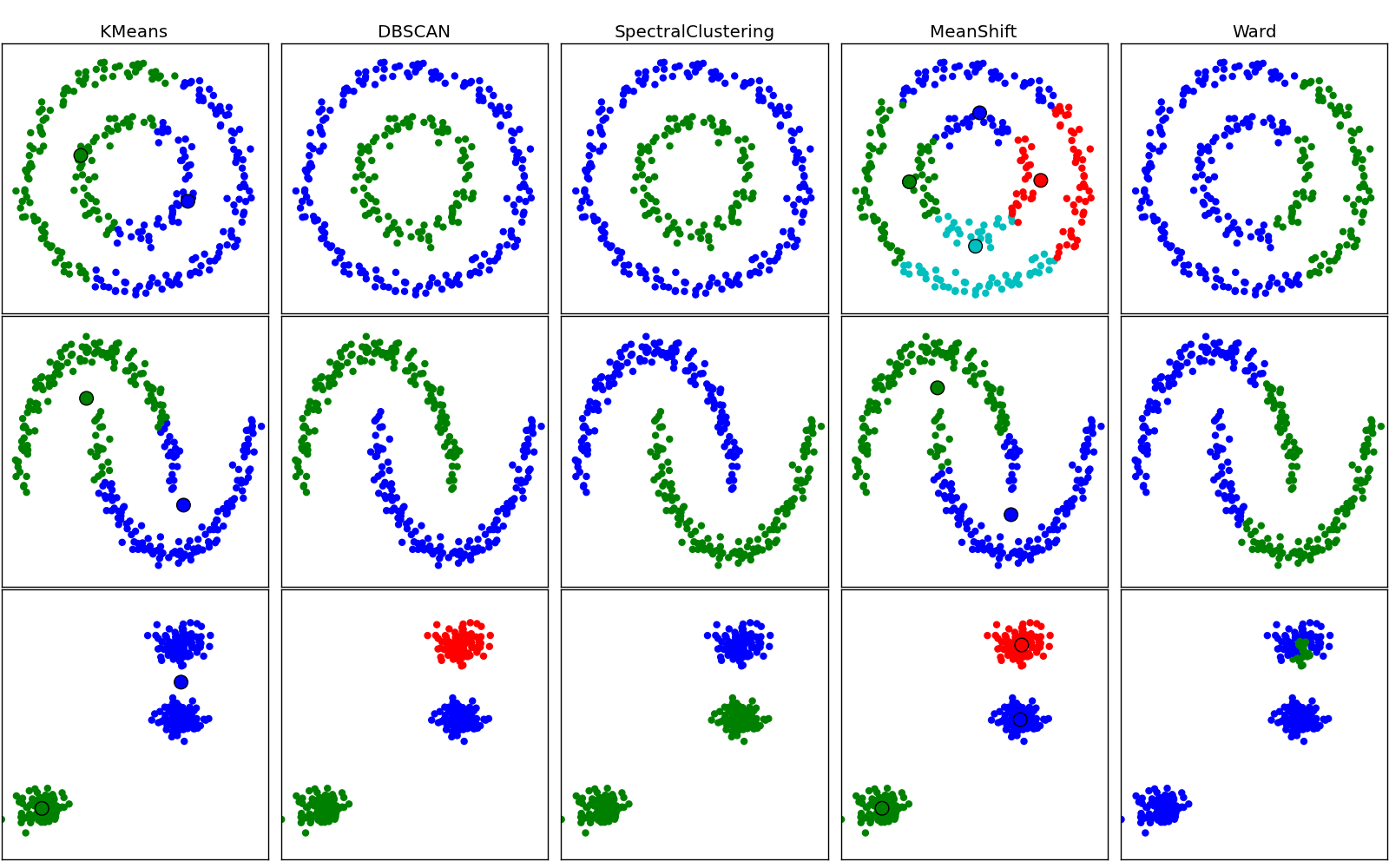
Aims

* Groups similar data points into cluster or groups
* Goal is to identify the inherent structure or patterns within the data
* Application includes image segmentation, anomaly detection and social network analysis.

Key types of algorithms

* K-means (K-means clustering) – good for clear splodges
* Density (eg. DBSCAN, MeanShift) – good for specific shapes
* Hierarchical (Ward’s Method)
* Probabilistic (Gaussian Mixture Models (GMMs))

Different clustering algorithms are more/less suitable for identifying specific shapes of clusters:

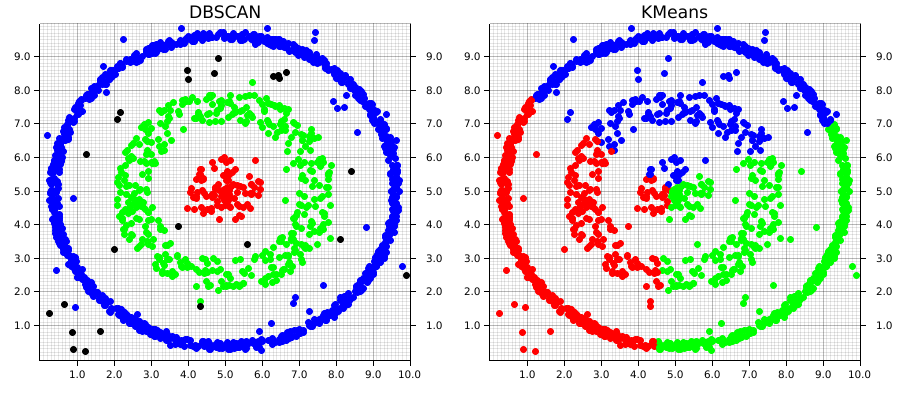


K-Means (see html for application of algorithm)

* Divides data into a predefined number of clusters (K)
* Each cluster represented by a centroid
* Assigns data points to nearest centroid
* Iterative process - updates centroid until convergence is achieved
* Method
  + We specify how many clusters we want to find
  + The algorithm will take the data and work out which points of the data belong to which cluster.
  + At the end of our algorithm we will have each of our data points having membership of a class. Our algorithm will not tell us what the different classes mean, but we can interpret the meaning using domain knowledge.
  + This method uses the mean position of each cluster to assign membership of the cluster, we will call this mean position a centroid.
* Step by step
  + Step 1 - Initialise Centroids
    - Select k initial samples of the data as the starting centroids. This can be done either randomly or using a more complex algorithm approach.
  + Step 2 - Assign To Nearest Centroid
    - Calculate the Euclidean (for lower dimensional, Manhattan or other for higher dimensional) distance between all points and each centroid. Assign each point to it’s nearest centroid.
  + Step 3 - Calculate New Centroids
    - Now that we have each data point as a member of a class, we can calculate the new centroid of that class as the mean position of all data points with that class.
  + Step 4 - Repeat Steps 2 & 3
    - Now that we have the new centroid values we can reassign each data point to an class based on it’s distance to each newly calculated centroid. Once the assignment takes place new centroids are calculated and so on.
  + This process continues for some specified number of iterations or until some convergence condition is met.
* Output
  + Will be dependent on the initial selection (seeds) of centroids chosen
    - For this reason the algorithm should be run multiple times and the output checked to avoid poorly generalised models.
  + How many steps this algorithm will take to run is dependent on several factors. These are:
    - n - The number of data points
    - k - The number of clusters
    - d - How many dimensions (axis) each data point has
    - i - The number of iterations until convergence (how many repeats, determined by how well clustered the data is)
  + How long it will take for our algorithm to run is linear with each of these factors. Therefore as all these increase so will our algorithm’s runtime.
* Evaluation
  + KMeans clustering does have some inherent problems and areas where it does not perform well.
  + KMeans clustering excels where there is a clear separation between classes when taking into account the distance from the center. You can think of this as clustering “as the crow flies”.
  + The KMeans method is unable to consider local structure, it assumes that all clusters are convex, and that the centroid calculated is a significant measurement.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) – see html for implementation

* Core algorithm for identifying clusters with specific spatial characteristics or shapes
  + You define the characteristics of the shape
* Identifies clusters based on density
* Divides dataset into core, border & noise points
  + Core points expand to form clusters based on density connectivity
  + Border points are on the edge of clusters
  + Noise points do not belong to any cluster



* Instead of clusters being convex, DBSCAN assumes that clusters are high density (many points in small area) and are separated by areas where there are fewer data points in a given area.
* Significantly, when compared to K-Means, a number of clusters does not need to be specified, DBSCAN will find this for us.
* Other than which distance metric to use (Euclidean/Manhattan etc), there are two characteristic values used by the DBSCAN algorithm.
  + The first is the “reach distance”, the radius around a point that a density will be calculated from. As with the rest of our methods using distances, we want to be sure we are centering the data.
  + The second is the number of points within our reach distance we want to count an area as high density (includes the point in question).
* These two quantities, the number of points and surrounding area, allow the density around each point to be calculated.
  + If the density is above the required amount then the new point is added to the cluster.
  + Higher number of points or lower distance will require the data to be more dense in order to be added to the cluster.

Evaluation

* Robust to outliers and capable of handling cluster with different shape and size
* Overkill for simple clusters
* Comparison to K-Means
  + Instead of measuring distances and creating centroids, a density based method takes into account how many data points are within the surrounding area of a “core sample”. A core sample is a sample that is within an area of high density.
  + The algorithm calculates the distance between points in and out of the core sample, this calculation spreads throughout the whole data in a search tree.
  + The result of this approach is that density-based methods can cluster in interesting shapes - so long as there is a dense enough link from points.
* Density challenges
  + When using a density based approach we expect our clusters are expected to have a similar density as each other. This helps us to tune an appropriate eps/min\_samples.
  + If this is not true then it can often be a challenge to find appropriate parameter values for the algorithm, and produce poor clustering labels.

Hierarchical Clustering (eg. Ward’s Method)

* Builds hierarchy of clusters with either a top-down or bottom-up approach.
  + Bottom-Up/Agglomerative
    - Starts with each points as a separate cluster
    - Merge clusters based on similarity
  + Top-Down/Divisive
    - Begins with all points in single cluster
    - Splits clusters recursively
  + May be useful to use elbow plots to identify the sweet spot
* Agglomerative Method (see html for implementation and a lot more detail on agglomerative clustering including Ward’s method and dendrograms)
  + Step 1
    - Initialise all data points as their own clusters. There will be n clusters at this step.
    - Each data point is within a cluster of one single point.
  + Step 2
    - Calculate the distance between all the clusters. Merge the two clusters which have the smallest distance between them (most similar). There will now be one fewer clusters in the data set.
  + Step 3
    - Repeat step 2 until the number of clusters is equal to the number of desired clusters (j).
    - If j=1 the algorithm will continue until all points are within the same cluster.
  + The end result of our clustering, and how well the method performs will depend on the point in the iteration that we stop combining clusters.

Which algorithms to choose?

* Analyse data characteristics
  + Assess dataset size, dimensionality, and distribution
  + Consider number of data points and features
  + Evaluate data distribution for complex structures
* Define cluster requirements
  + Determine desired properties of clusters
  + Assess level of separation between clusters
  + Consider whether clusters should be tightly packed or spread out
* Evaluate Computational Complexity
  + Assess scalability concerning dataset size and dimensionality
  + Evaluate runtime performance for real-time application
* Experiment and validate
  + Experiment with multiple clustering algorithms
  + Compare results
  + Choose the algorithms that best meets the clustering objectives and computational constraints

Evaluation Metrics – Known Class Membership (Extrinsic metrics)

* These metrics can only be used if we have access to labels, to carry out comparative analysis to the ground truth
* Adjusted Rand Index Properties (see html for implementation)
  + The Adjusted Rand Index (ARI) takes as an input the found cluster membership and true class values.
  + The score is bounded between [-1, 1]. A good ARI, indicating similarity between clusters and classes, is near 1. A bad ARI, indicating no similarity between clusters and classes is negative.
  + A random selection of clustering compared to true classes will give an ARI of near 0. This is important as it allows us to compare our clustering to a baseline selection.
  + We do not need to know anything about the structure of the clusters or classes to perform this measure.
* Other extrinsic methods (use true values) for measuring cluster performance include:
  + Mutual Information (normalise, adjusted)
  + Homogeneity, Completeness, V-measure (their harmonic mean)
  + Fowlkes-Mallows Index

Evaluation Metrics – Unknown Class Membership (Intrinsic metrics)

* These metrics can always be used, whether we have access to labels or not, to carry out any ground truth analysis
* For intrinsic evaluation we no longer have the true labels.
  + This means we need to generate meaningful quantities from the data and clusters generated.
  + We can do this using the distances between different positions within the data space.
* Silhouette Scores (see html for implementation)
  + Used to evaluate the quality of clusters produced by algorithms
  + Measures compactness (how close data points are to each other within the same cluster) separation (how far apart different clusters are from each other) between the clusters.
  + Ranges from -1 to 1:
    - Score close to 1 suggests the clusters are well separated and dense
    - Score close to -1 indicates overlapping clusters
* Davies-Bouldin Index
  + Computes average similarity between each cluster and its most similar cluster.
  + Helps identify optimal number of clusters for a dataset
  + Value closer to 0 indicates well separated and distinct clusters

Association

Aims

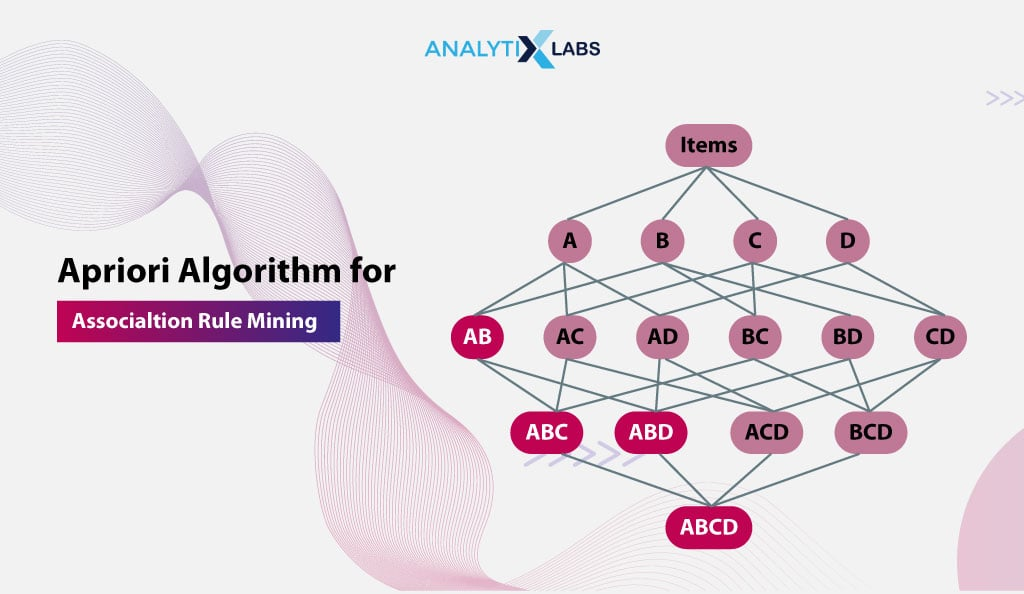
* Focuses on discovering relationships between variables
* Identifies frequent patterns, correlations or co-occurrences
* Commonly used for anomaly detection, recommendation systems and market basket analysis

Method

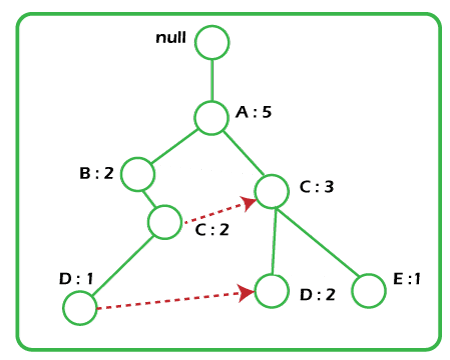
* Rely on measures of interestingness:
  + Support – How frequently the itemset {X,Y} (products) appear in the transaction database.
    - If the support is 1%, then 1% of the transactions contain the item.
  + Confidence – How strong the link between products X and Y are.

Common Methods

* Apriori
  + Generates frequent patterns by creating itemsets with pairing, grouping etc.



* FP-Growth (Frequent Pattern)
  + Generates frequent patterns from a tree without need for itemset generation.



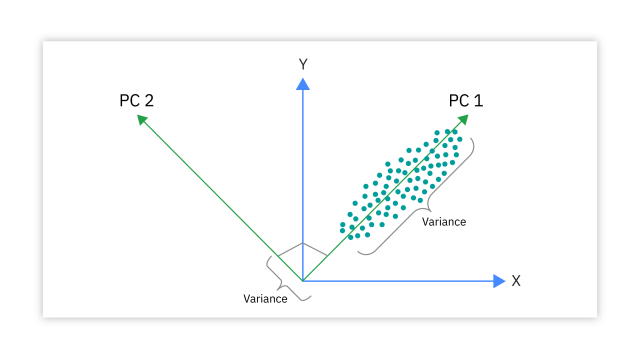
Dimensionality Reduction

Aims

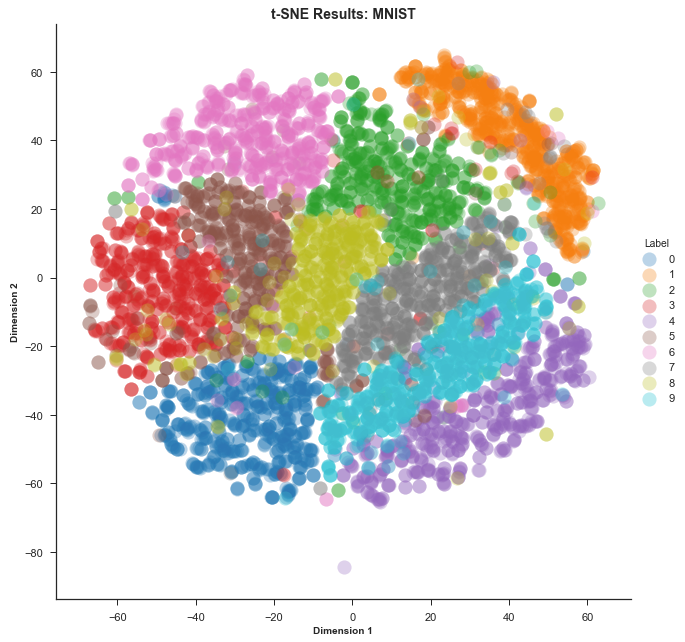
* Approach for reducing the number of features in a dataset
* Aims to capture meaningful properties while minimising loss of information.
* Commonly performed during pre-processing as a follow-up step for feature engineering

Common Methods

* Principal Component Analysis (PCA)
  + Linear technique that projects data from a high dimensional space to a lower dimensional space.
  + Takes linear combinations (sums) of features that are likely correlated to create a smaller set of uncorrelated features.
  + Each new feature is called a principal component, with the first capturing maximum variance and so on.
  + Must manually label the new components to clarify their relation to the original features (the components will be spat out as PCA1, PCA2 etc)



* t-distributed Stochastic Neighbour Embedding (t-SNE)
  + Allows separating of non-linear data that requires more than straight lines to be projected in lower dimensions.
  + Preserves relationships between data points as opposed to the minimising variance reduction.
  + Excellent for visualising high dimensional data.



Evaluating Dimensionality Reduction

* Reconstruction Error (key metric)
  + Measures the difference between original data and its reconstructed form.
  + Lower error indicates retained information from original data.
  + Helps preserve important features while reducing dimensionality.
  + Will; need to use elbow plots to identify the sweet spot in performance
* Computational resources
  + Unsupervised learning can be computationally intensive
  + Usually big data
  + Requires efficient algorithms
* Overfitting and Generalisability
  + Models may be prone to overfitting without labelled data
  + Capture noise or irrelevant patterns
* Interpretability
  + Output visualisations and clusters etc are complicated.
  + Challenge to interpret, especially for non-experts
  + Lack of transparency in how clusters/projections were formed.
  + When trying to communicate this to others, it is useful to build plots up piece by piece and overlay one element at a time