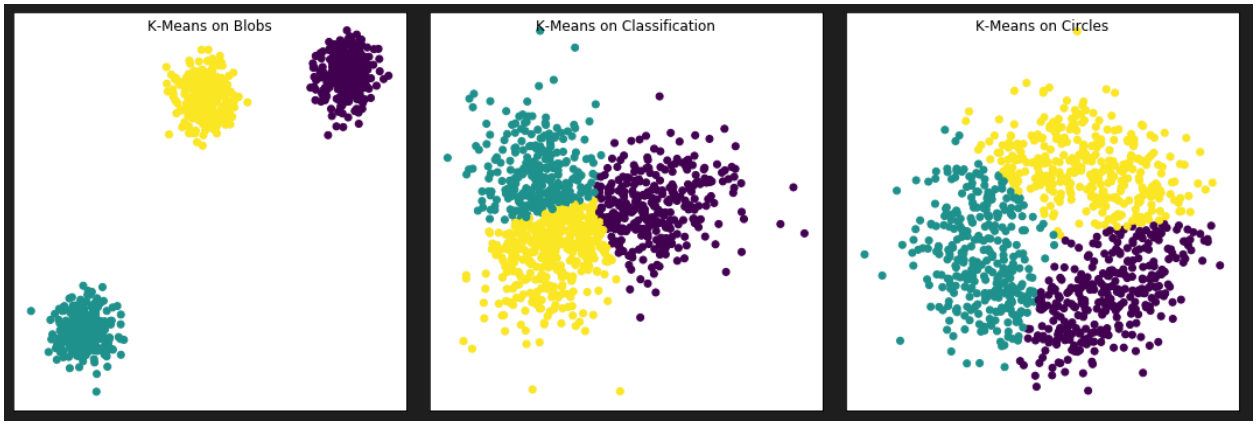
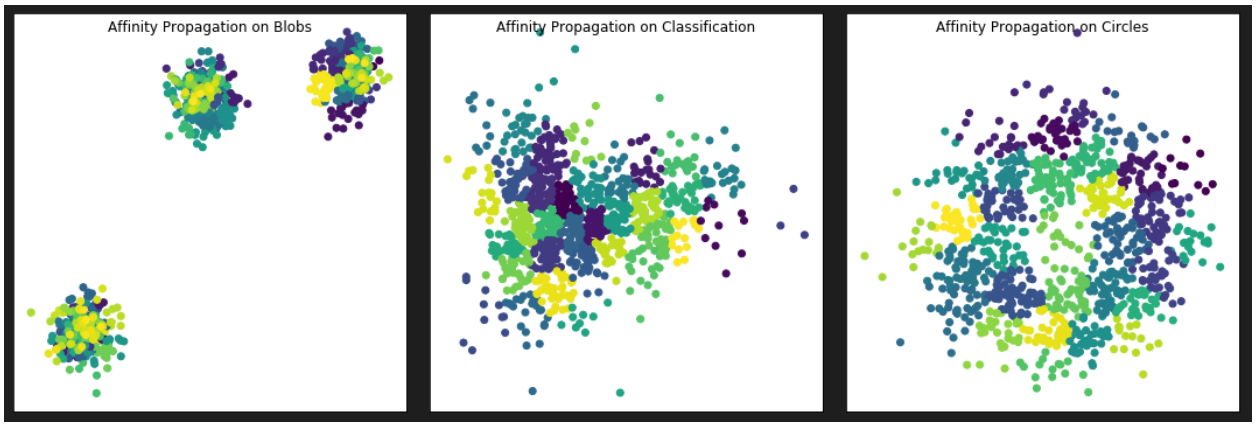
# Analysis of Algorithms

## K-means




K-Means is a clustering algorithm that groups data into k numbers of clusters, where each cluster is represented by the average of its points (the centroid). For the Blobs dataset, K-Means did a great job, clearly identifying all clusters and colouring each blob differently with no outliers. This shows how good it is with well separated round clusters. In the classification dataset, K-Means split the big central cluster into three equal parts, which shows its tendency to create equally-sized groups even if it doesn't match the actual data structure. For the circles dataset, K-Means divided the circular data into three parts, showing its limitations with non-round clusters. I set the number of clusters (k) to 3 for this analysis. This choice worked well for the Blobs dataset, matching the structure of the data . However, it didn't work as well for the Classification and Circles datasets, highlighting how important it is to choose the right k based on the data. Overall, K-Means is simple and efficient, especially for large datasets with well-separated clusters, but it struggles with non-round and overlapping clusters, as seen with the Circles dataset.

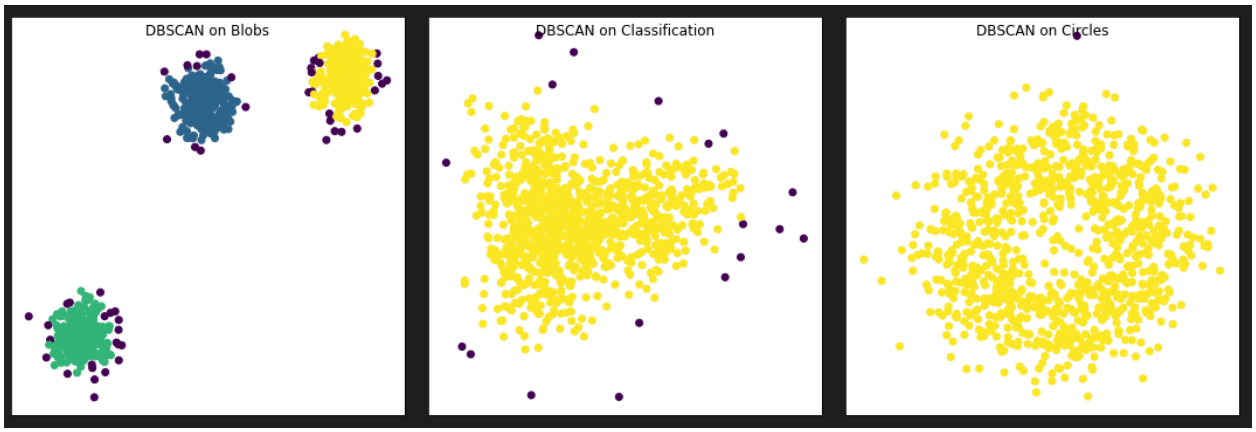
## Affinity Propagation




Affinity Propagation is a clustering algorithm that finds which data points should be the representative points and uses them to form clusters. For the Blobs dataset, it ended up with

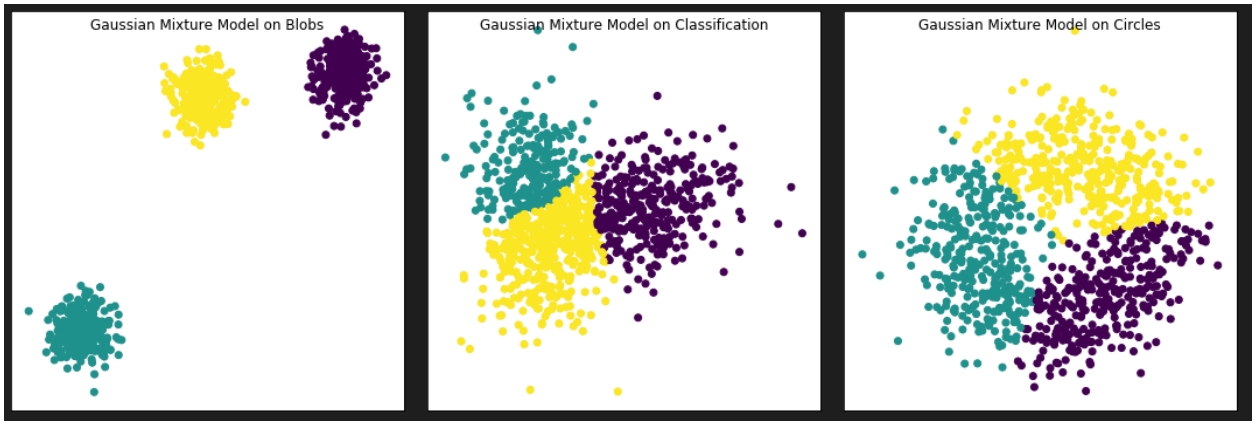
lots of different colours, which made the visualisation pretty confusing and didn't clearly show the clustering pattern. This suggests that Affinity Propagation can end up creating a lot of clusters, which might not always match the expected structure of the data.
In the Classification dataset, Affinity Propagation split the big central cluster into a bunch of smaller sections with lots of colours, showing a lot of tiny, isolated clusters scattered around. This shows its tendency to find more clusters than  expected. The Circles dataset had a similar issue, with many small groups and scattered colours, reflecting how sensitive the algorithm is to the data's structure and its ability to create lots of clusters.
While Affinity Propagation automatically determines the number of clusters, it seems how well it works can depend on the nature of the dataset. It's good for finding clusters of different shapes and sizes, but it can also lead to over clustering or confusing results if not adjusted right. It's useful for complex datasets where you don't know the number of clusters ahead of time, but can have too many clusters or less clear results.

## DBSCAN



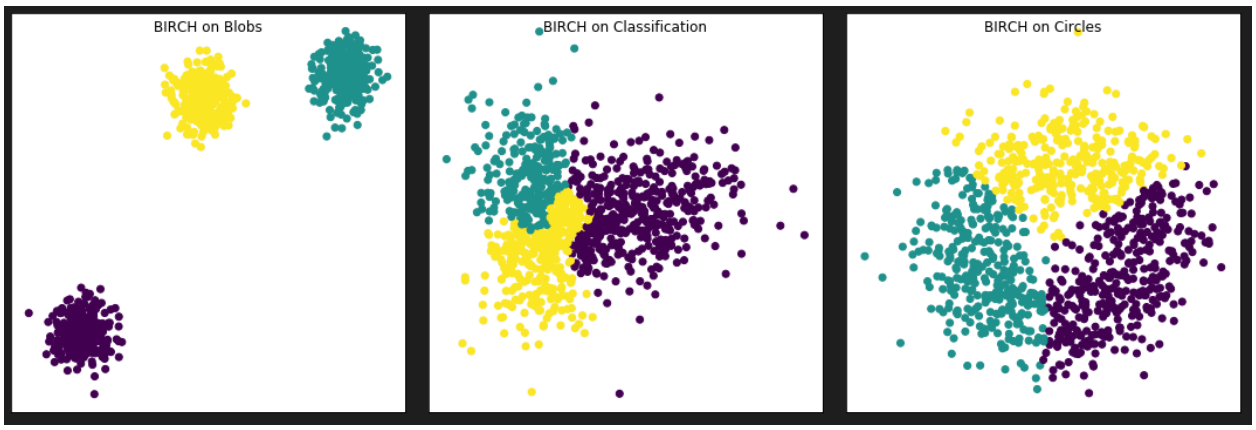

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that groups points that are close together and marks points in low-density areas as outliers. For the Blobs dataset, DBSCAN did a good job finding the main clusters, each with its own colour. However, it also picked up some outliers around each blob, probably because they were outside the default eps (epsilon) value of 0.5. This shows that DBSCAN can be sensitive to the eps setting and may flag sparse points as noise. In the Classification dataset, DBSCAN found a big central cluster and some outliers scattered around the edges, showing it can handle dense areas and isolated points. But also making the assumption that they are all one class. For the Circles dataset, DBSCAN ended up with one big cluster and just one outlier, which is similar to the Classification results. This suggests DBSCAN works well with circular shapes if the density settings are right and there is only one class. Overall, DBSCAN is good at finding clusters of different shapes and sizes and managing noise, but its performance depends a lot on getting the eps and min_samples settings right to avoid too many or too few clusters.

## Gaussian Mixture Model




The Gaussian Mixture Model is a clustering algorithm that assumes data comes from a mix of several Gaussian distributions. For the Blobs dataset, GMM did a great job finding all the clusters, with each blob in its own colour and no outliers. This shows GMM's strength in accurately modelling Gaussian distributions. In the Classification dataset, GMM split the big central cluster into three somewhat even sections, similar to K-Means, and didn't produce any outliers, showing how well it can partition data into distinct Gaussian components. For the Circles dataset, GMM also divided the circular structure into three parts, much like K-Means. This happens because both algorithms  try to minimise the distance between points and their cluster centres, but GMM uses a probabilistic approach. Overall, GMM is great for datasets with Gaussian-like clusters. However, like K-Means, it can struggle with non-Gaussian shapes and requires you to specify the number of components beforehand. n_components  was set to 3 so it could identify the 3 clusters otherwise the default value was 1 meaning  all points were the same class.

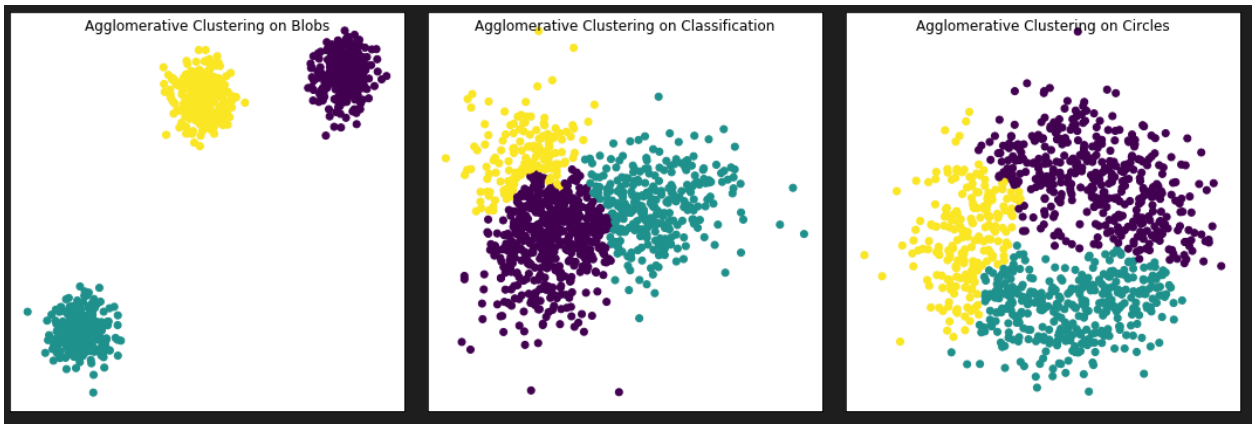
## BIRCH




BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) is a hierarchical clustering algorithm that builds a clustering feature tree (CF tree) to summarise large datasets and then clusters these summaries. For the Blobs dataset, BIRCH did a great job finding all the clusters, with each blob in its own colour and no outliers, efficiently handling well-separated, round clusters. In the Classification dataset, BIRCH split the big central cluster into three sections of different sizes. The splits weren't even and didn't have clear borders, showing that

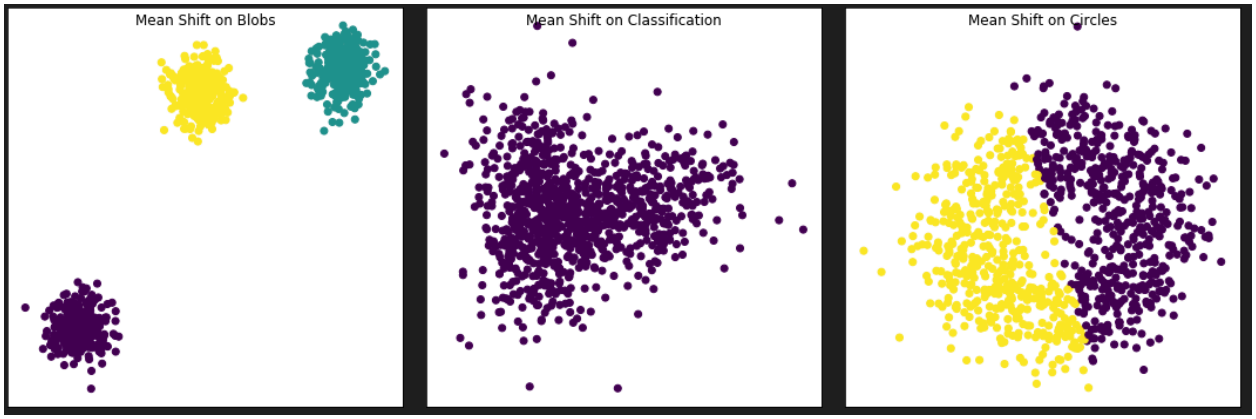
BIRCH can handle clusters of different shapes and sizes but might be a bit imprecise in defining clear cluster boundaries. For the Circles dataset, BIRCH divided the circular data into three roughly equal parts without any outliers, managing circular structures better than some centroid-based methods. BIRCH's hierarchical approach makes it scalable for large datasets and good at handling noise. However, the uneven and sometimes unclear splits in the Classification dataset point to a possible weakness in accurately partitioning clusters when there aren't clear boundaries.

### Agglomerative Clustering




Agglomerative Clustering is a hierarchical clustering algorithm that builds nested clusters by successively merging or  splitting them using a bottom-up approach. It starts with each data point as its own cluster and merges the closest pairs of clusters until the desired number of clusters is reached. For the Blobs dataset, Agglomerative Clustering successfully identified all clusters, with each blob in its own colour, showing its effectiveness in handling well-separated, spherical clusters. In the Classification dataset, the algorithm split the large central cluster into three sections with distinct boundary lines, though the splits were not completely even. This highlights Agglomerative Clustering's ability to create clear divisions, even if the boundaries are somewhat harsh. For the Circles dataset, Agglomerative Clustering produced results similar to the Classification dataset, splitting the circular data into three sections with softer boundary lines. This reflects its hierarchical nature, which can adapt to different cluster shapes but create clearer separations than other algorithms. Overall it seems Agglomerative Clustering is versatile and can handle various cluster shapes. Its ability to create clear boundaries makes it useful for datasets where distinct separations are needed, although this can sometimes lead to less natural splits in uneven data distributions.

## Mean Shift




Mean Shift is a clustering algorithm that finds clusters by shifting data points towards the areas with the highest density. For the Blobs dataset, Mean Shift did a great job picking out all the clusters, with each blob in its own colour and no outliers. This shows it's good at handling well separated clusters without needing to know the number of clusters in advance. In the Classification dataset, Mean Shift lumped all the data points into one big cluster, meaning it couldn't tell different density regions apart and treated everything as one dense area. For the Circles dataset, Mean Shift split the data into two sections along a clear line, effectively separating the data in half. This shows that Mean Shift can identify dense regions and split clusters based on density changes, although the harsh boundary lines seem not to indicate a very clear classification of the pattern in the data. Overall, Mean Shift is flexible and doesn't need you to specify the number of clusters ahead of time, but it can struggle with datasets that don't have clear density peaks, like the Classification dataset.