Assignment 3

Atharv Karkar

Foundation of Machine Learning

Course Instructor : Arun Rajkumar

This report discusses the application of Principal Component Analysis and K-Means clustering on two different datasets to achieve dimensionality reduction and effective data segmentation respectively.

PCA is one of the very commonly applied linear transformation techniques in transforming complex data into simpler ones with most retained variance. In showing the application of PCA to the MNIST dataset on handwritten digits, it is made possible to extract meaningful features that may be used for visualization, reconstruction, and proper usage for downstream tasks.

K-Means clustering, a basic unsupervised learning algorithm, is applied to separate a sample dataset into clusters. The basic Lloyd's algorithm is also benchmarked with various numbers of clusters.

We also extend the approach using kernel-based K-Means that overcome the disadvantage of linear boundaries, and effective segmentation of non-linearly separable data.

Q.1) PCA

In this report, we apply PCA to the MNIST dataset, a collection of 28x28 grayscale images of handwritten digits. The goal is to explore how PCA can be used to reduce the dimensionality of the dataset while maintaining the ability to recognize digits.

1. Data Loading and Preprocessing

The MNIST dataset contains 28x28 pixel images of handwritten digits from 0 to 9. It loads into one image tagged with a label representing the number. We used the Hugging Face load_dataset function to load this dataset, and preprocessing it for PCA was done.

We first extract the image data and labels. Then we randomly select a subset of 1000 images with 100 from each class. Dividing the pixel values by 255 is used to normalize them into the [0, 1] interval.

- Dataset Loading This dataset is loaded using the MNIST library from Hugging Face, containing image data (X) with their labels (y).
- Random Sampling: Let us randomly select 1000 images, each class comprising 100 images, using np.random.choice in order to preserve the class distribution.
- Normalisation: Normalize the pixel values to between 0 and 1. This can be achieved by dividing the original values by 255.

2. PCA Implementation

The major steps of PCA are as follows:

- Mean Centering: Scale the data such that each feature or pixel has a zero mean.
- Covariance Matrix Computation: Computations of covariance matrix on centered data help explain their relationships.
- Eigenvector decomposition: A more computationally intensive technique that will feed the covariance matrix into an eigenvector decomposition function to calculate eigenvectors. The amount of variance explained for each component will be indicated by the eigenvalues, with the eigenvectors representing the principal components.

- Sorting and Selection: Order the eigenvectors by their corresponding eigenvalues in ascending order, then pick the largest num_components number of eigenvectors.
- Projection: Project the data on the selected eigenvectors so that the dimension of that data comes down.

Step 1: Flattening and Mean-Centering: Flatten the data (from 28x28 to 784 pixels) per image to be easily worked on. The feature, which is a pixel in this case, means-centres by subtracting the mean across each feature's images.

Step 2: Covariance Matrix: The covariance matrix is computed on the centred data to see how different pixel values covary with each other.

Step 3: Eigenvalue Decomposition: To find the eigenvalues and their respective eigenvectors, np.linalg.eigh is employed. This is a decent algorithm for dealing with symmetric matrices, such as a covariance matrix.

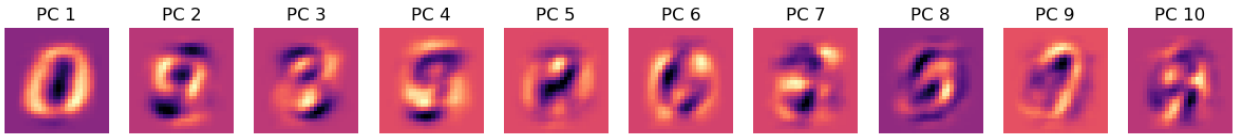
Step 4: Sorting Eigenvectors: The eigenvectors are sorted in the descending order of their eigenvalues; hence the components, that explained most variability in the data, go first.

Step 5: Component Selection: The highest num_components number of eigenvectors are chosen to compose the new basis for the reduced-dimensionality space.

Step 6: Projection: The data is projected onto the selected eigenvectors to end up with reduced data.

3. Plot of Principal Components

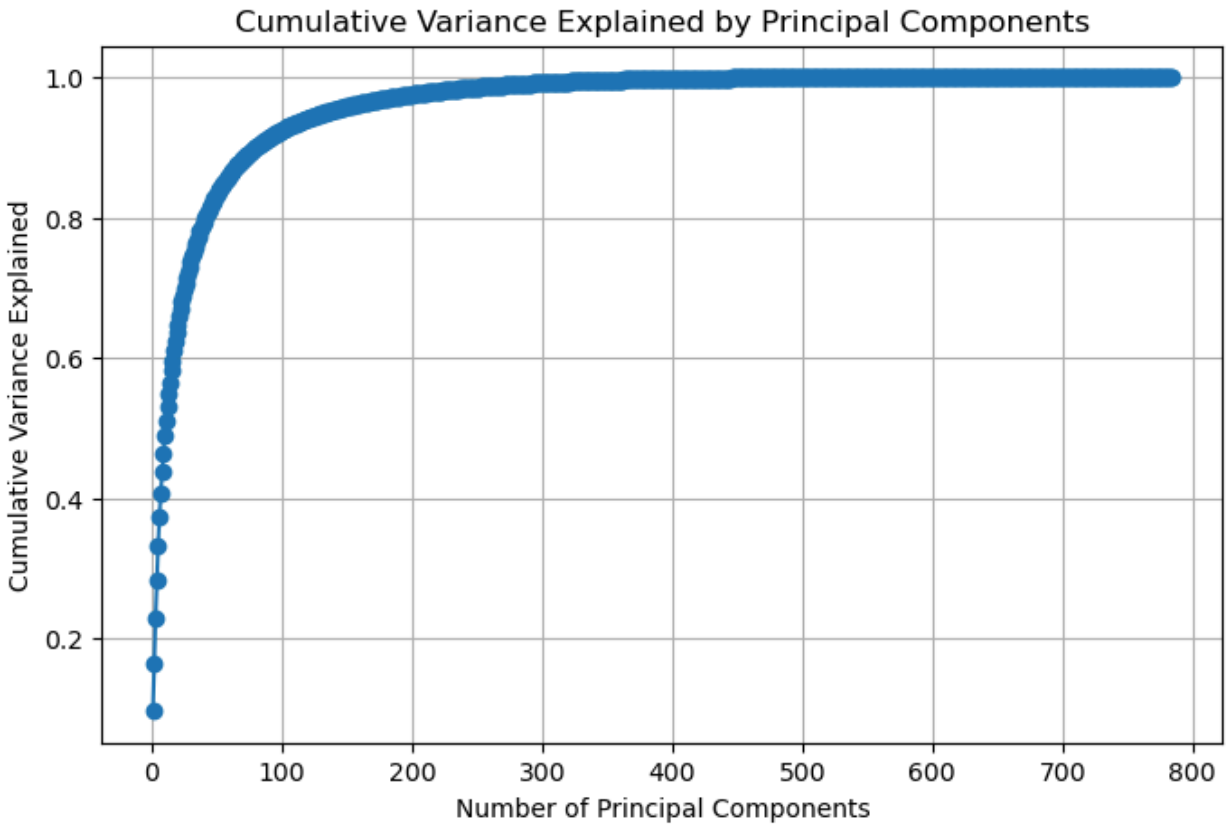
The first few of the principal components (eigenvectors) may be visualized to get an understanding of what features they capture. Each of the principal components might be reshaped back into a 28 x 28 image and then displayed.



4. Variance Explained by Principal Components

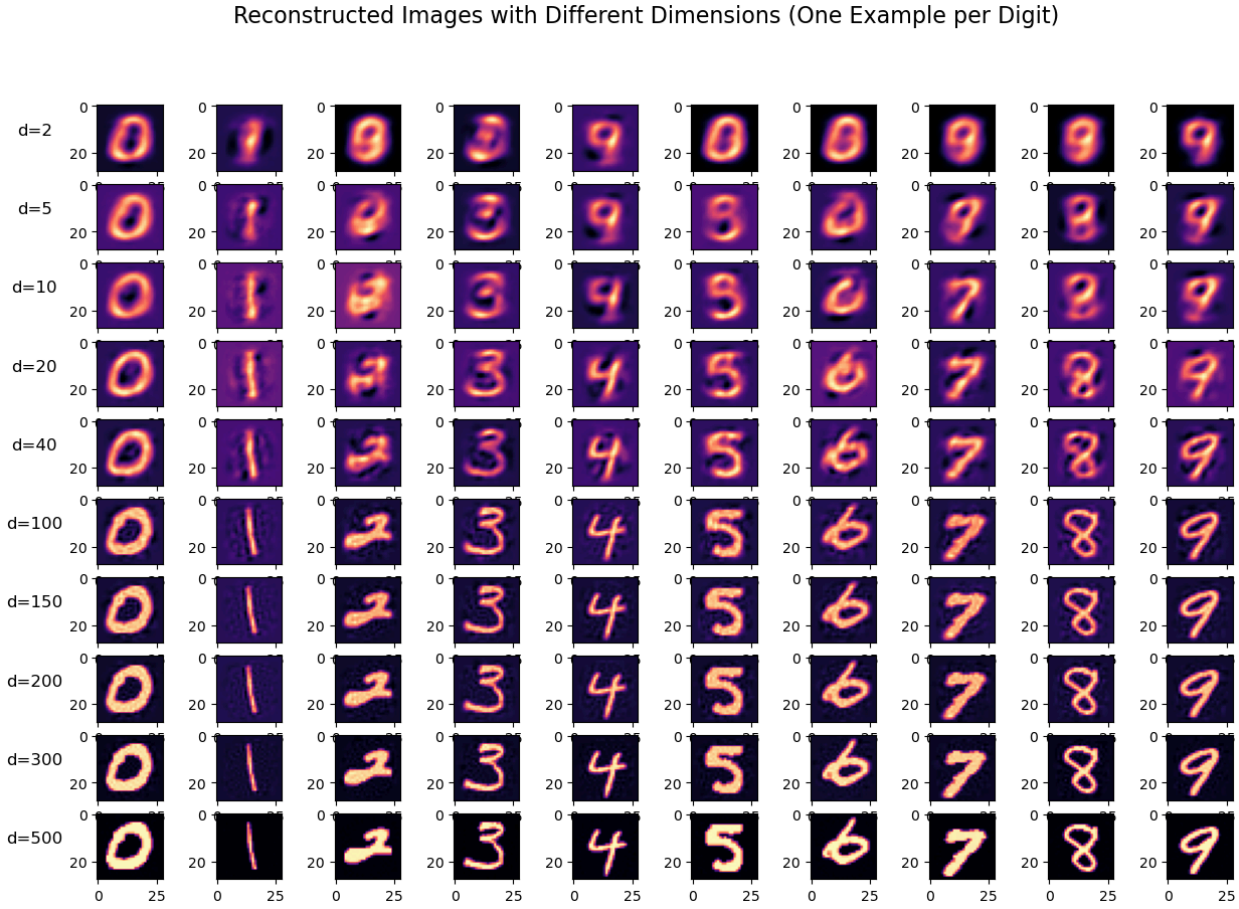
To understand how much information is retained by every principal component, we calculate the explained variance ratio. This ratio would be the amount of all total variance that each particular component explains. We take a cumulative variance in order to see how many components are required for one to explain a given percentage of the total variance.

- Explained Variance: For each of the principal components, we calculate the explained variance ratio to know how much of the total data variation is captured by a component. We are also able to depict the cumulative variance plot, showing us the cumulative percentage of variance explained as we continue increasing the number of components.
- Optimal Number of Components Based on the cumulative variance plot, we choose an optimal number of components (around 134 components), so as to retain at least 95% of the total variance.



5. Reconstructing Data with Varying Dimensions

- To assess the performance of PCA, we look into how well the images can be reconstructed using a changing number of dimensions - principal components. We assess the effectiveness of the representation by seeing how well the information is retained at reconstruction.



- Reconstruct the images with different numbers of principal components (from 2 to 500). For a large number of components, they are reconstructed very close to the original images.
- Visual Inspection: Clearly, the reconstructed images of digits have to 100 to 150 features. Dimensionality reduction merits in this case appear to include the ability to retain most features of the data.

Conclusion

PCA is very efficient in reducing the dimension of the MNIST dataset while retaining the majority of the variance. Hence with an optimal number of components selected, we can obtain a retention of 95% variance to about 134 components. The reconstructed images reveal that the

digits can be identified with a number of around 100 to 150 components retained. Thus, it reflects the power of PCA in compressing the data without losing major information. These lower dimensions are useful when trying to accomplish various kinds of machine learning such as classification and clustering that support improvements in computational efficiency.

Q.2) K-Means

This assignment has been done to cluster a data file, cm_dataset_2.csv, which is comprised of 1000 data points. Major goals are,

- Running the K-means algorithm with k=2 for 5 different random initializations and analyzing the error function and clusters.
- Exploring the clustering results for k={2,3,4,5} using Voronoi diagrams.
- Determine whether Lloyd's algorithm is appropriate for clustering the data set and discuss further alternatives.

i) Lloyd’s Algorithm with k=2

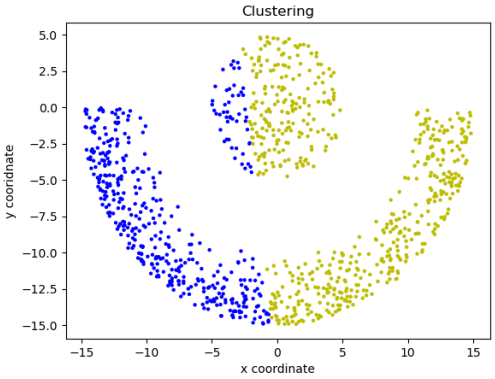
i. Lloyd's Algorithm with K = 2 and Random Initializations

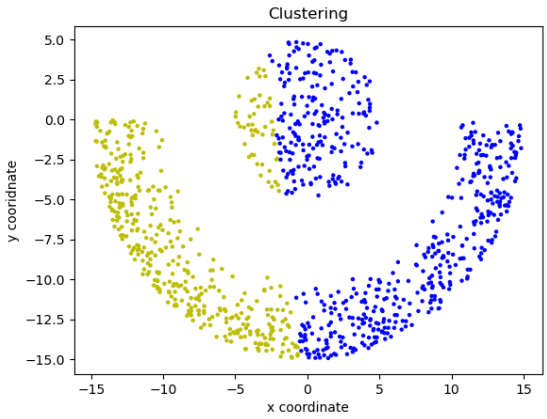
Methodology:

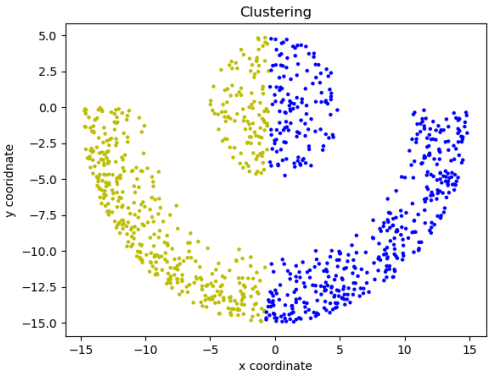
- Obtain two random centroids for clusters.
- Assign the data point to the closest centroid iteratively until converged. That is, a minimum of intra-cluster variance is reached.
- Plot the error function at each iteration for every random initialization.

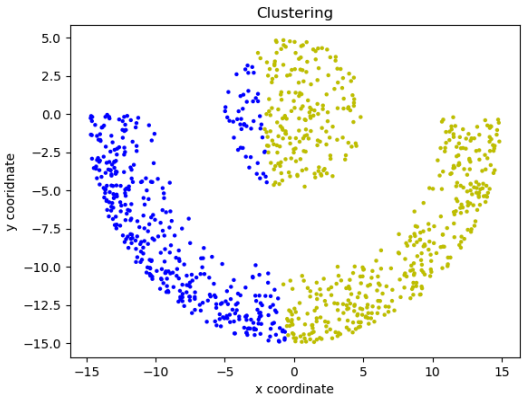
Observations:

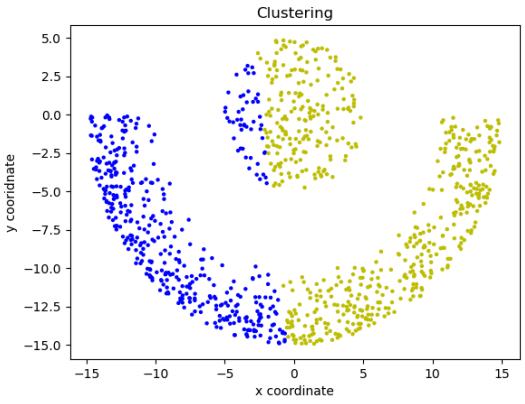
- For k=2, Lloyd's algorithm converges for all five random initialisations. The data points are grouped into two clusters as depicted by distinct colour shades. For the experiment, the result indicates that clustering quality is related to the intrinsic structure of the data.
- The centroids moved iteratively and stabilised by the end of the algorithm. Due to random initialisation, the starting points for the centroids are quite a long way from each other; thus, the convergence paths will be different.
- The result of the clustering algorithm does not separate between the semi-circular region and the circle that well. Some points belonging to the different regions were put into the same cluster, mainly at places where the two regions are similar in orientation.
- K-Means will assume linear, hyperplane-based cluster boundaries and therefore will not adapt to a non-linear shape.





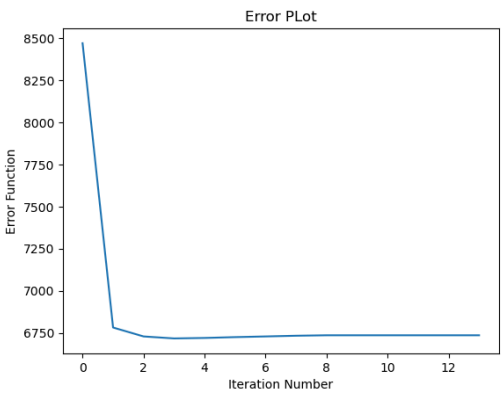


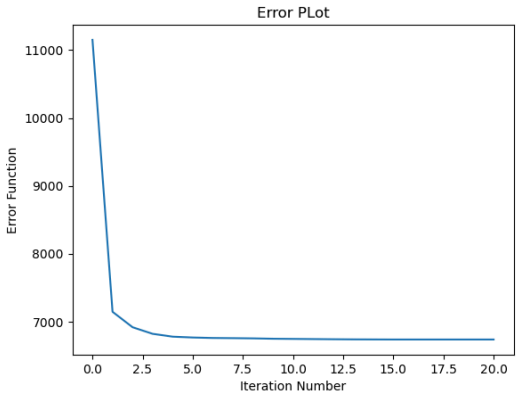


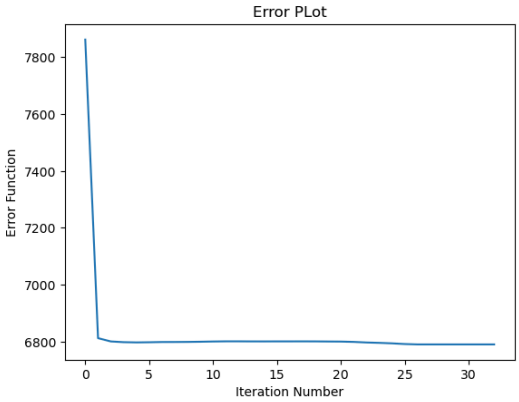


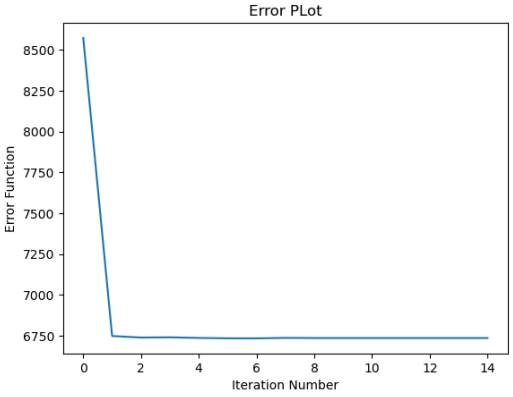
Behaviour of the Error Function:

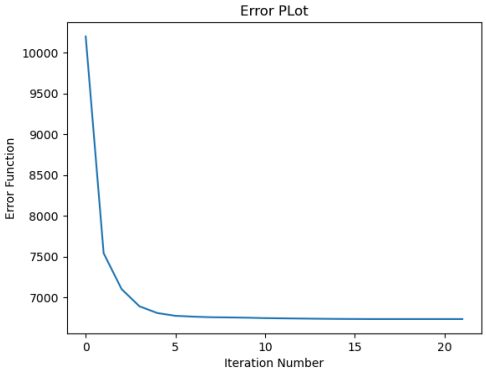
- The curve for the error function plotted against the number of iterations indicated a significant sharp drop in the early iterations and subsequent stabilization. This illustrates the fact that as the algorithm executes, it is minimizing the within-cluster variance.
- Although the initial errors were different due to the differences in the initial centroids of the five runs, the final error values all converged close to about the same point, and thus indicate robustness in minimizing the objective function.











ii) Clustering and Voronoi Regions for Different k

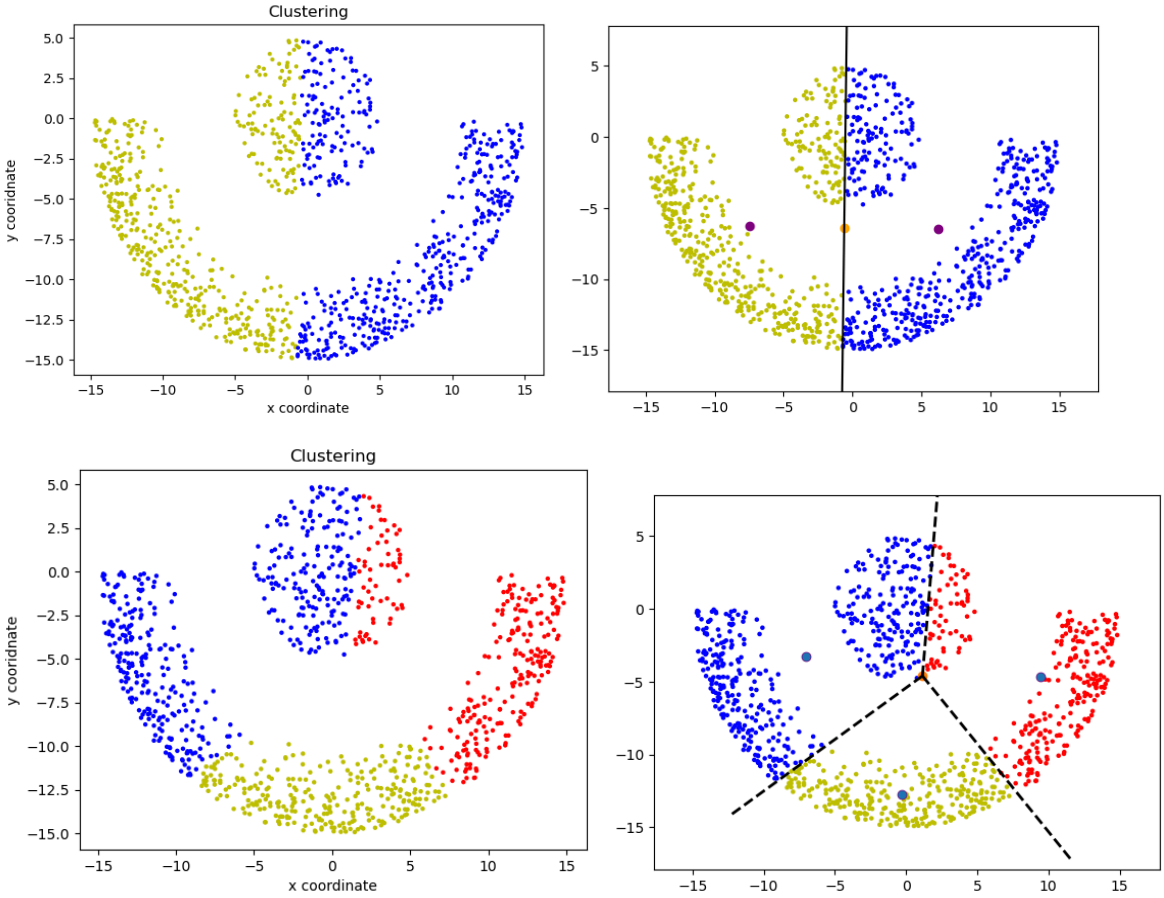
For k={2,3,4,5}, the algorithm does the following:

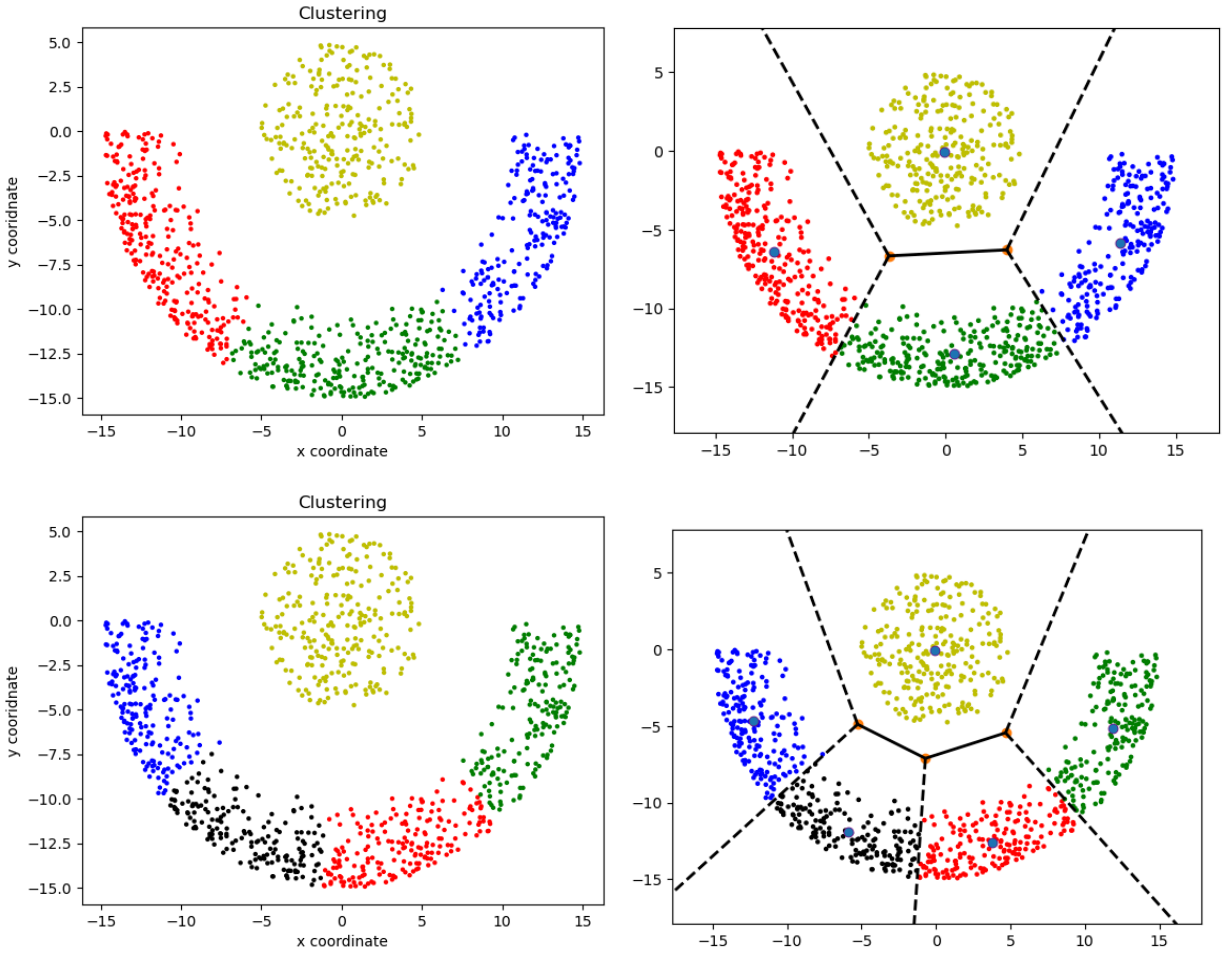
- Cluster Initialization:
    - Randomly initialize the cluster centers for k.
- K-means Algorithm:
    - The same exact process of iteration steps above to update and classify the cluster centers.
- Visualisation:
    - Plot the final clusters of each of the different k in colors.

- Voronoi regions can be computed from the cluster centres and overlaid on the clusters to show how each region falls within some center.
- Voronoi regions offer a natural intuitive way to explain how the space is partitioned between the cluster centers-how the clustering algorithm partitions the data set.

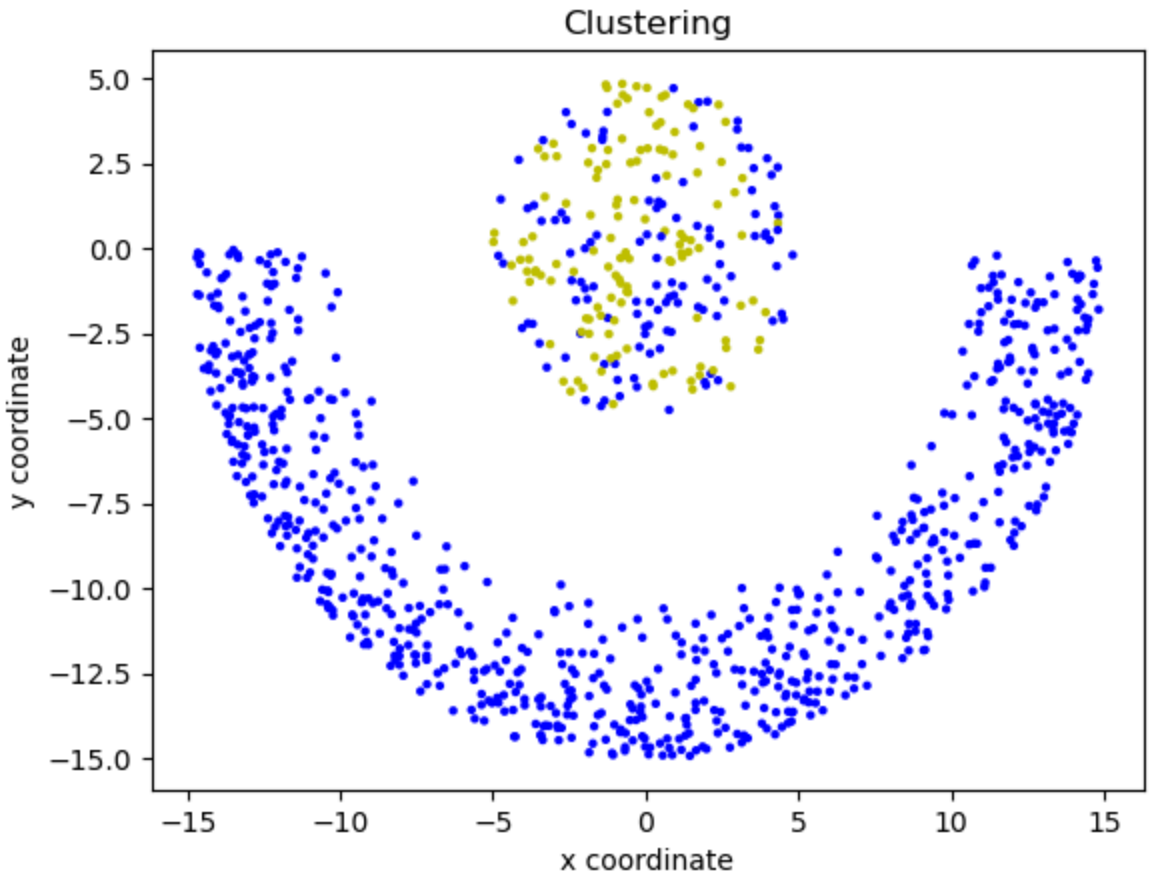
Effect of Cluster Count (k)

- The number of clusters, k increases from 2 to 5 and the dataset is divided into more and more refined groupings.
- These clusters then tend to cover smaller regions and capture more localized data patterns with a higher k.
- However, a value of k too large leads to overfitting where clusters represent more noise than meaningful groups.





- k=2: The clustering results aren't that good, probably because the complexity or separation of the data isn't represented well with just two clusters. Non-linear datasets usually have regions that a small k cannot represent at all.
- k=3: Not good enough. This implies that one extra cluster is not enough to separate the data.
- k=4: Results are somewhat better. This implies that the increment in cluster count does partially include some structure in the data but is not entirely aligned with the intrinsic clusters.
- k=5: Results are further improved. This implies that a higher
- k is closer to the actual underlying distribution of the data in the feature space.



iii) Kernel-Based K-Means

Kernel Transformation:
Makes use of a Gaussian kernel for transforming data into a feature space of higher dimensions to allow for the clustering of data that could not be linearly separable.
Precomputes a kernel matrix by using the formula of RBF kernel:

$$K[i][j] = e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}}$$

where σ^2 is a kernel parameter (set to 1 in this case).

Clustering in Feature Space:

It performs K-means clustering on the transformed data, that is rows of the normalized eigenvectors of the kernel matrix.
Visualization:
The plot of the clustered data, shows the separability in the transformed space.