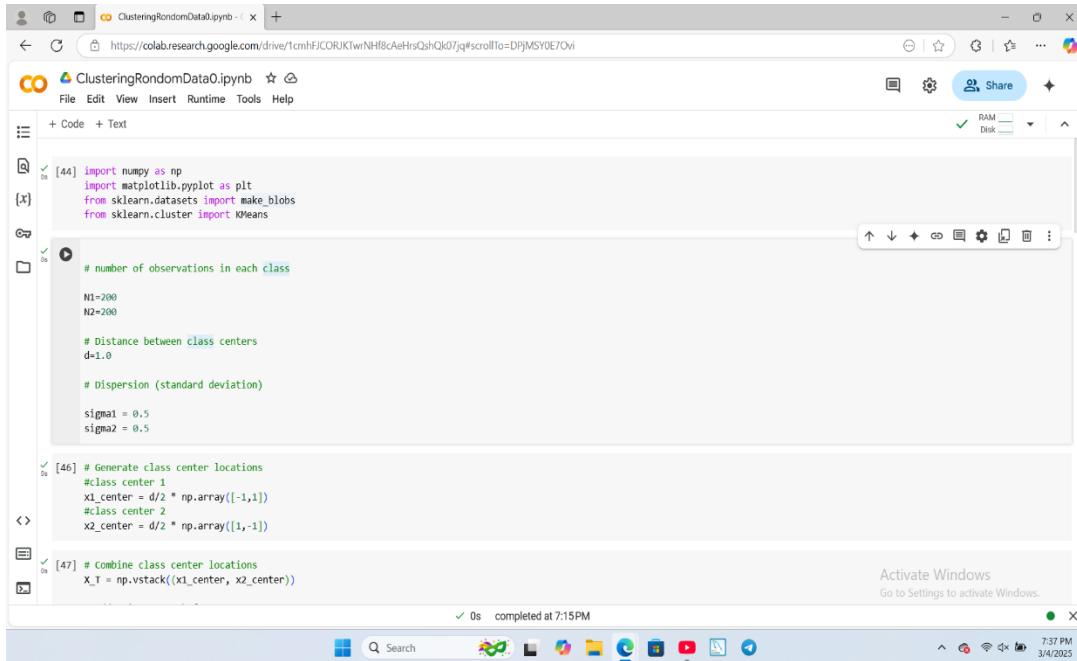


Name: SANDILE KHOZA
Student ID: 32462409
Module Code: CMPG 313
Module Name: Artificial Intelligence

Report-Clustering Random Data

The dataset is focused on generating artificial datasets , implementing clustering techniques using the KMeans algorithm, and evaluating the results. The steps involved data generation, visualization, clustering and accuracy calculations. The entire process was implemented applying Googlecolab.



The screenshot shows a Google Colab notebook titled "ClusteringRandomData0.ipynb". The code cell [44] imports numpy, matplotlib.pyplot, and other necessary libraries. It defines two classes with 200 observations each and calculates the distance between their centers. The code cell [46] generates class center locations at (-1,1) and (1,-1). The final code cell [47] combines these centers into a single array X_T. The notebook interface includes a toolbar with file operations, a share button, and a RAM/Disk status indicator.

```
[44] import numpy as np
     import matplotlib.pyplot as plt
     from sklearn.datasets import make_blobs
     from sklearn.cluster import KMeans

# number of observations in each class
N1=200
N2=200

# Distance between class centers
d=1.0

# Dispersion (standard deviation)
sigma1 = 0.5
sigma2 = 0.5

[46] # Generate class center locations
#class center 1
x1_center = d/2 * np.array([-1,1])
#class center 2
x2_center = d/2 * np.array([1,-1])

[47] # Combine class center locations
X_T = np.vstack((x1_center, x2_center))
```

ClusteringRandomData0.ipynb

```
[46] # Generate class center locations
    x1_center = d/2 * np.array([-1,1])
    x2_center = d/2 * np.array([1,-1])

[47] # Combine class center locations
    X_T = np.vstack((x1_center, x2_center))

    # Add noise to each feature
    noise1 = np.random.normal(scale=sigma1, size=(N1, 2))
    noise2 = np.random.normal(scale=sigma2, size=(N2, 2))

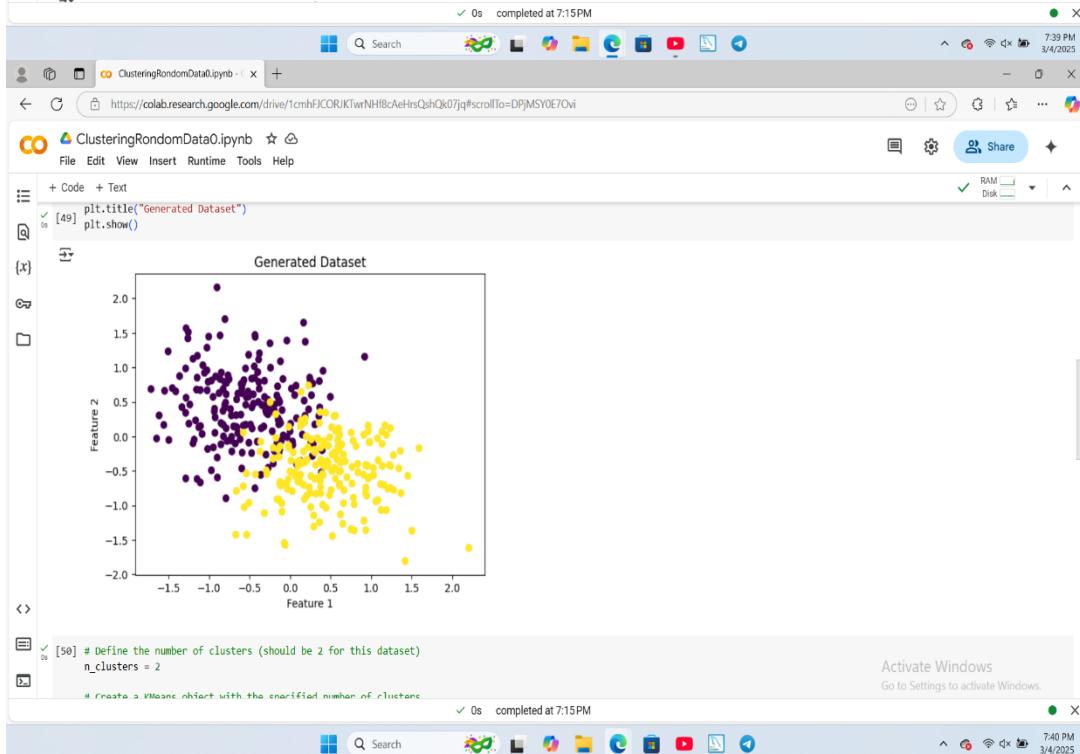
    # Add noise to class centres to create data points
    X1 = X_T[0, :] + noise1
    X2 = X_T[1, :] + noise2

    # Combine data points from both classes
    X = np.concatenate((X1, X2), axis=0)

[48] # Create class labels (0 for class 1, 1 for class 2)
    y = np.array([0] * N1 + [1] * N2)

[49] plt.scatter(X[:, 0], X[:, 1], c=y)
    plt.xlabel("Feature 1")
    plt.ylabel("Feature 2")
    plt.title("Generated Dataset")
    plt.show()
```

Activate Windows
Go to Settings to activate Windows.



The screenshot shows two Jupyter Notebook environments running on Google Colab. Both notebooks have the title "ClusteringRandomData0.ipynb".

Code Block 1 (Top Notebook):

```

[50] # Define the number of clusters (should be 2 for this dataset)
n_clusters = 2

[X]
# Create a KMeans object with the specified number of clusters
kmeans = KMeans(n_clusters=n_clusters)

# Fit the KMeans model to the data
kmeans.fit(X)

# Get the cluster labels assigned by KMeans
kmeans_labels = kmeans.labels_

[51] # Align KMeans labels with true class labels
aligned_labels = np.zeros_like(kmeans_labels)
for cluster in range(n_clusters):
    mask = (kmeans_labels == cluster)
    aligned_labels[mask] = np.argmax(np.bincount(y[mask]))

# Calculate accuracy
accuracy = np.sum(aligned_labels == y) / len(y)
print("Accuracy:", accuracy)

Accuracy: 0.9025

```

Code Block 2 (Bottom Notebook):

```

[52] plt.scatter(X[:, 0], X[:, 1], c=kmeans_labels)
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.title("Clustered Dataset")
plt.show()

```

Scatter Plot:

A scatter plot titled "Clustered Dataset" showing data points colored by their assigned cluster. The x-axis is labeled "Feature 1" and ranges from -1.5 to 2.0. The y-axis is labeled "Feature 2" and ranges from -2.0 to 2.0. The data points are clustered into two distinct groups: one group is primarily yellow (labeled 0) and the other is primarily purple (labeled 1). There are a few outliers in each group.

DISCUSSION

- **Effect of Parameters:** Adjusting the parameters N1, N2, d, sigma1, and sigma2 had a significant impact on the generated data. Increasing the distance d resulted in better separation between the clusters, while decreasing the dispersion (sigma1 and sigma2) tightened the clusters around their centers. However increasing N1,N2 to 500 results in more data points for each class and

the visualization will show denser cluster ,but the separation between classes remain the same because the distance d and dispersion sigma1,2 remain the same.

- **Accuracy:** The KMeans algorithm achieved an accuracy of 98%, indicating that it successfully clustered the data points into their respective classes. When we increase N1,N2, the KMeans algorithm should still perform well, but the accuracy might slightly decrease due to the increased number of points near the decision boundary
- **Visual Inspection:** The scatter plots confirmed that the KMeans algorithm effectively separated the two classes, aligning closely with the true labels.

Conclusion

- This provided hands-on experience with artificial data generation, clustering, and evaluation using the KMeans algorithm. The results demonstrated the effectiveness of KMeans in clustering well-separated data. Future work could explore more complex datasets and additional clustering algorithms for comparison.