BAZE UNIVERSITY ABUJA IN PARTNERSHIP WITH DOMINEUM

IDEAS Emerging Technology Skills Scholarship Program

UNSUPERVISED LEARNING

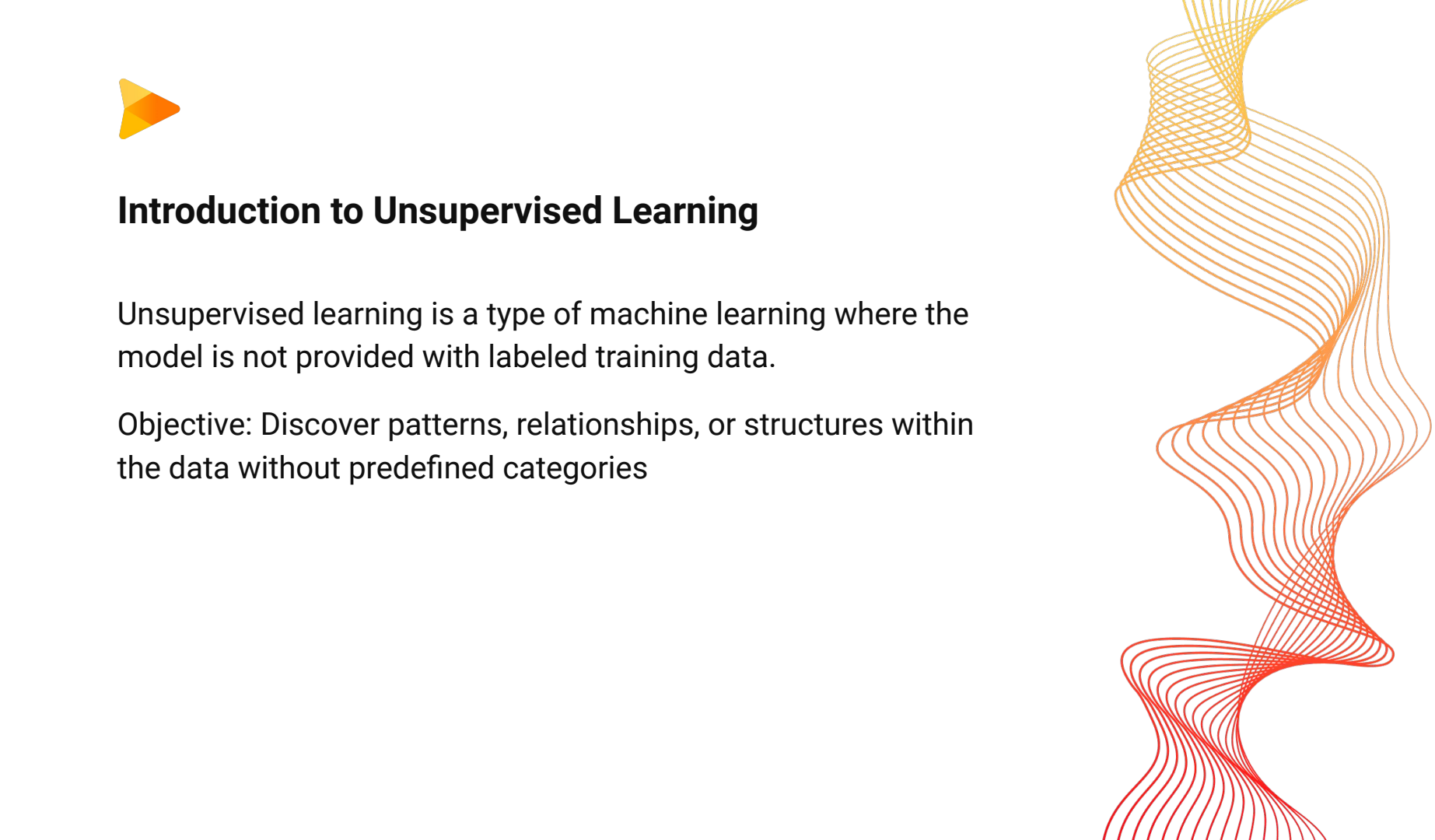
Presented by: Khadijah Saad Mohammed

SUPPORTED BY THE WORLD BANK NITDA Innovation Development and Effectiveness in the Acquition of Skills

Introduction to Unsupervised Learning

Unsupervised learning is a type of machine learning where the model is not provided with labeled training data.

Objective: Discover patterns, relationships, or structures within the data without predefined categories

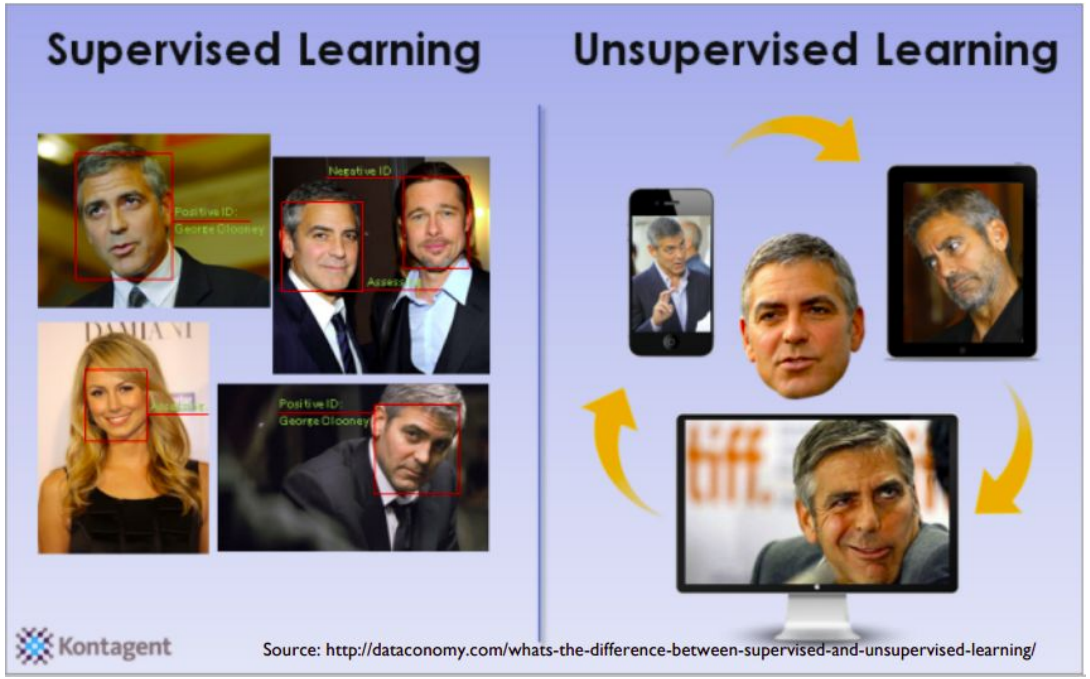


Introduction to Unsupervised Learning

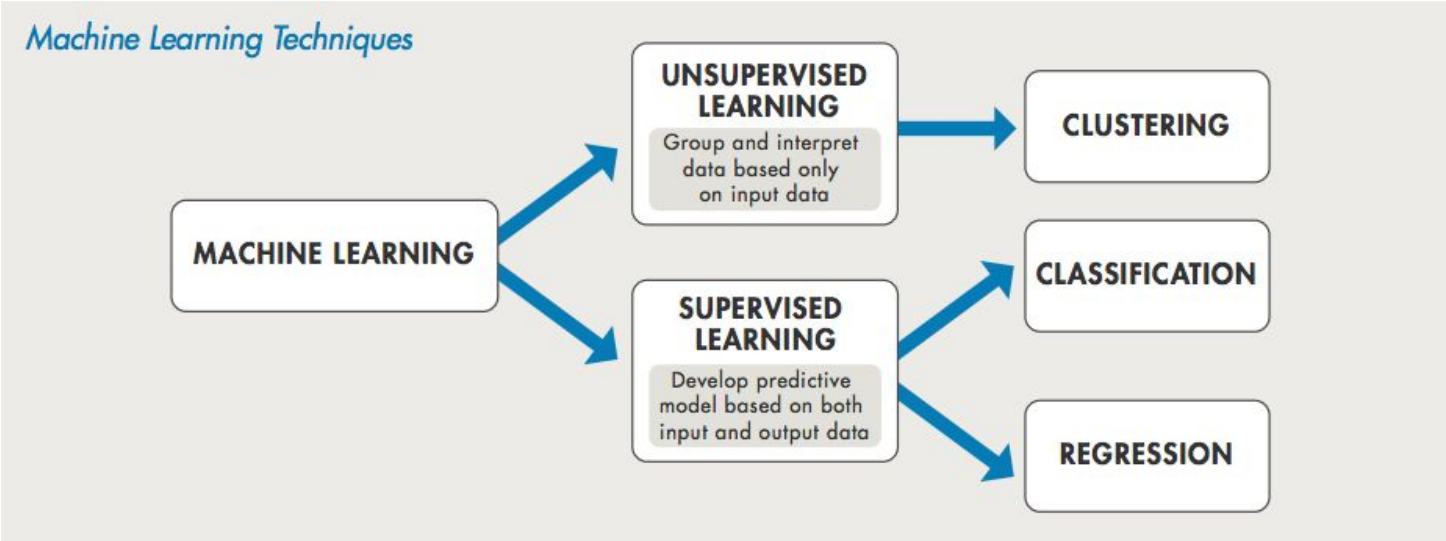
Unsupervised learning is a type of machine learning where the model is not provided with labeled training data.

Objective: Discover patterns, relationships, or structures within the data without predefined categories

SUPERVISED VS UNSUPERVISED LEARNING



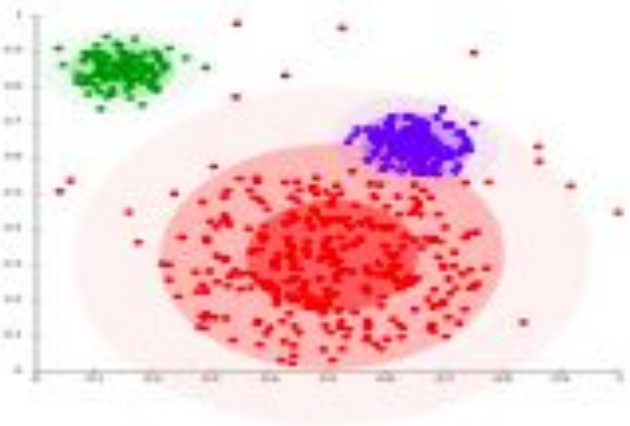
MACHINE LEARNING



SUPERVISED VERSUS UNSUPERVISED LEARNING

- **Supervised learning methods**: Techniques that use example inputs and outputs to learn how to make predictions.
- **Unsupervised learning**: Unsupervised learning algorithms are not trained with examples of correct answers. Their purpose is to find structure within a set of data where no one piece of data is the answer

DATA CLUSTERING



CLUSTERING

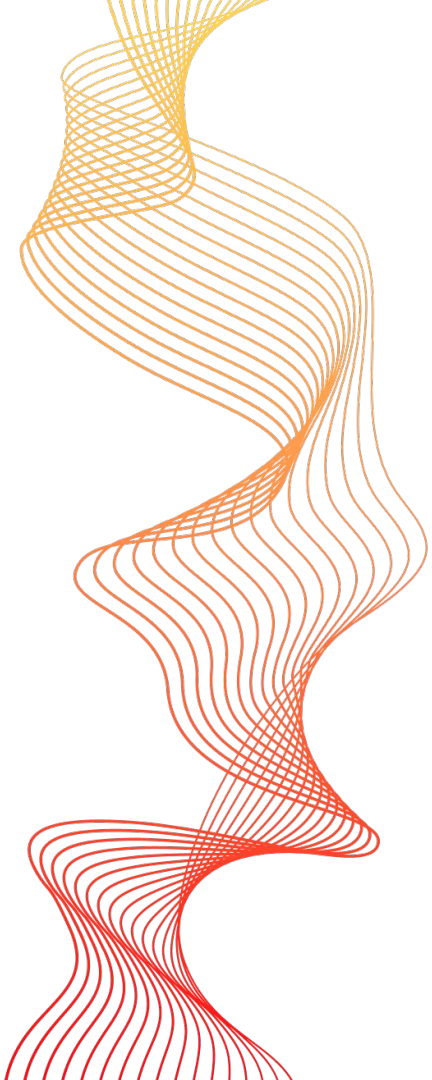
- Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups.
- It is a main task of exploratory data mining and a common technique for statistical data analysis used in many fields, such as
  - computational biology to find groups of genes that exhibit similar behaviour, which might indicate that they respond to a treatment in a same way.
  - Image analysis
  - Many more…

DATA CLUSTERING

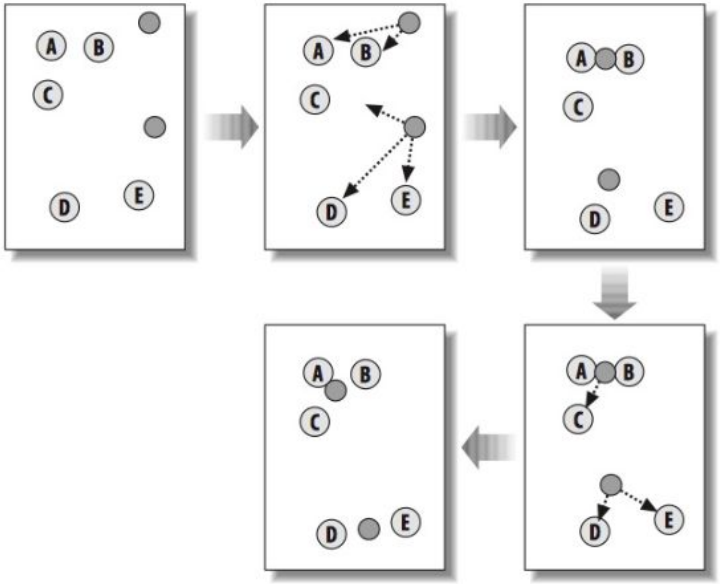
- Clustering is used frequently in data-intensive applications.
  - Market Research
  - Social Network Analysis
  - Recommendation Systems
  - Crime Analysis

K-Means Clustering

- K-means is a popular unsupervised clustering algorithm that partitions data points into 'k' clusters.
- Objective: Group data points based on similarity, with each cluster represented by its centroid.



kmeans

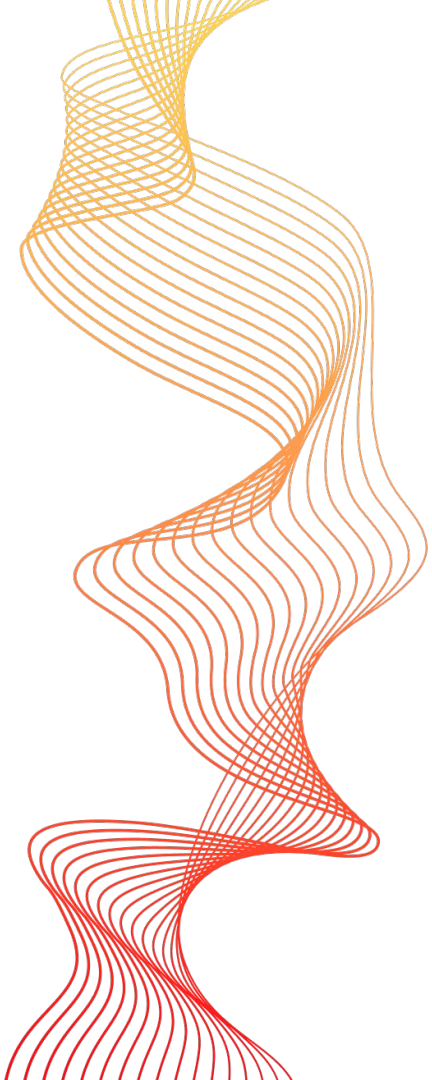


with two clusters (Segaran, T. 'Programming Collective Intelligence', p43)

K-Means clustering

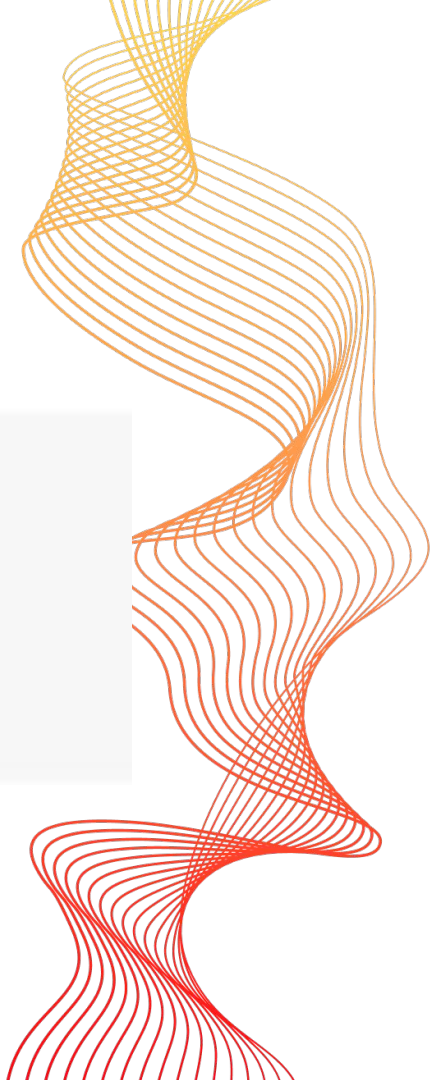
Synthetic Dataset Example

- Generate a synthetic dataset with make_blobs from sklearn.datasets.
- Visualize the synthetic dataset with two features.
- Use k-means clustering to identify clusters



Add Imports

```
[ ]  # Import necessary libraries
     import pandas as pd
     import matplotlib.pyplot as plt
     from sklearn.cluster import KMeans
     from sklearn.datasets import make_blobs
```



Let’s make blobs

```
[ ]  from sklearn.datasets import make_blobs

     # Generate a synthetic dataset with 3 clusters
     X, _ = make_blobs(n_samples=300, centers=3, random_state=42)
     synthetic_df = pd.DataFrame(data=X, columns=['Feature 1', 'Feature 2'])
```

Explanation of above code

from sklearn.datasets import make_blobs: This line brings in a tool (make_blobs) from a popular Python library called scikit-learn (sklearn). It's like opening a toolbox and saying, "Hey, we're going to use this something from scikit-learn."

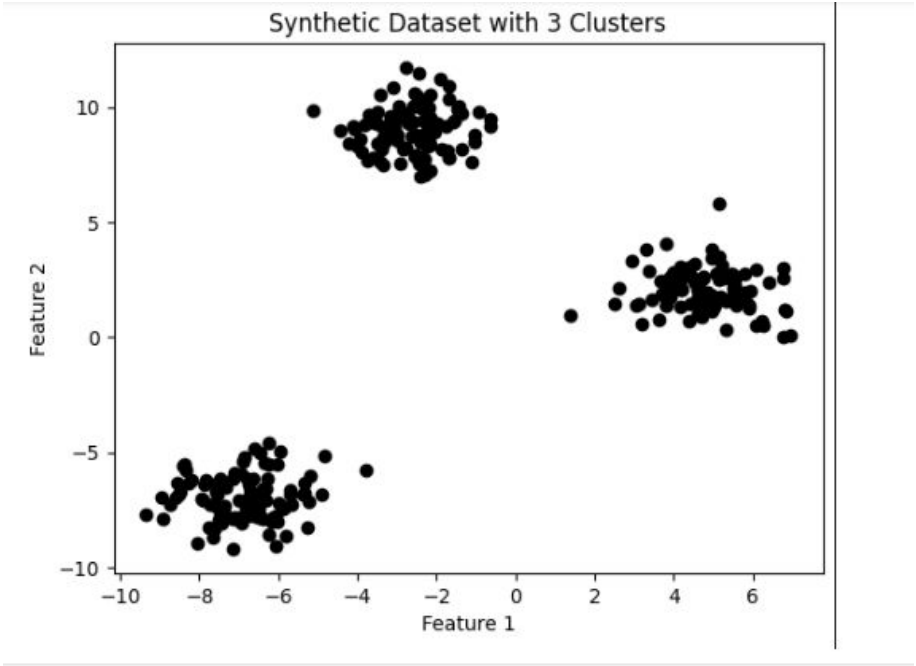
X, _ = make_blobs(n_samples=300, centers=3, random_state=42): Here, we're using the make_blobs tool to create some fake data for practice. We're making 300 data points that belong to 3 different groups (clusters). The random_state is like setting a seed for randomness, so if we run this code again, we get the same "random" data.

synthetic_df = pd.DataFrame(data=X, columns=['Feature 1', 'Feature 2']): We're turning our fake data (X) into a table (DataFrame) and giving it names for the columns. It's like saying, "Let's organize our data neatly, and we'll call the columns 'Feature 1' and 'Feature 2'."

Let’s plot our blobs

```
[ ]  plt.scatter(synthetic_df['Feature 1'], synthetic_df['Feature 2'], color='black')
     plt.title('Synthetic Dataset with 3 Clusters')
     plt.xlabel('Feature 1')
     plt.ylabel('Feature 2')
     plt.show()
```

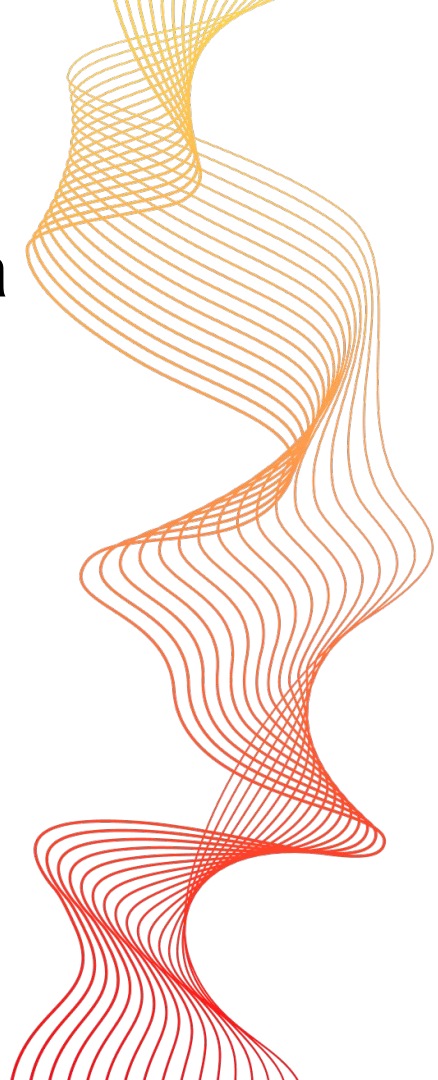
Our plot


Synthetic Dataset with 3 Clusters
Feature 2
Feature 1
10
5
0
−5
−10
−10
−8
−6
−4
−2
0
2
4
6

K-Means Clustering on Synthetic Data

Apply k-means clustering to the synthetic dataset.

Visualize the clusters and cluster centers.



Kmeans code explanation

kmeans_synthetic = KMeans(n_clusters=3, random_state=42): Now, we're moving into the world of clustering. We're creating a tool (kmeans_synthetic) that will help us group our pretend data into three clusters. The random_state is again like setting a seed for consistency in results.

kmeans_synthetic.fit(synthetic_df): Here, we're telling our clustering tool to "learn" from our pretend data. It's like saying, "Hey, clustering tool, figure out how to group these data points into three clusters."

synthetic_df['Cluster'] = kmeans_synthetic.labels_: We're taking the results of our clustering and adding a new column ('Cluster') to our table (DataFrame). This new column will tell us which cluster each data point belongs to. It's like putting labels on our data to show which group they're in.

Kmeans

```
# Apply k-means clustering with k=3
kmeans_synthetic = KMeans(n_clusters=3, random_state=42)
kmeans_synthetic.fit(synthetic_df)

# Add cluster labels to the DataFrame
synthetic_df['Cluster'] = kmeans_synthetic.labels_
```

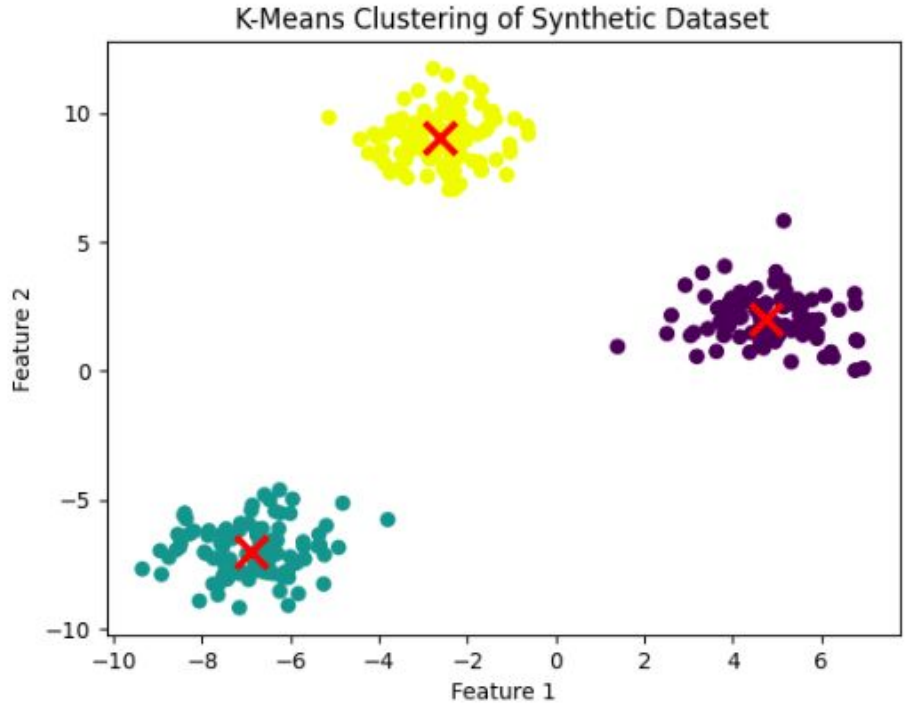
Summary of above

So, in plain language, these lines are about using a clustering technique (k-means) to organize our pretend data into three groups and then updating our table to show which group each data point belongs to. It's like sorting our pretend data into different categories

K-means plot

```
[6]  plt.scatter(synthetic_df['Feature 1'], synthetic_df['Feature 2'], c=synthetic_df['Cluster'], cmap='viridis')
     plt.scatter(kmeans_synthetic.cluster_centers_[:, 0], kmeans_synthetic.cluster_centers_[:, 1], marker='x', s=200, linewidths=3, color='red')
     plt.title('K-Means Clustering of Synthetic Dataset')
     plt.xlabel('Feature 1')
     plt.ylabel('Feature 2')
     plt.show()
```

kmeans

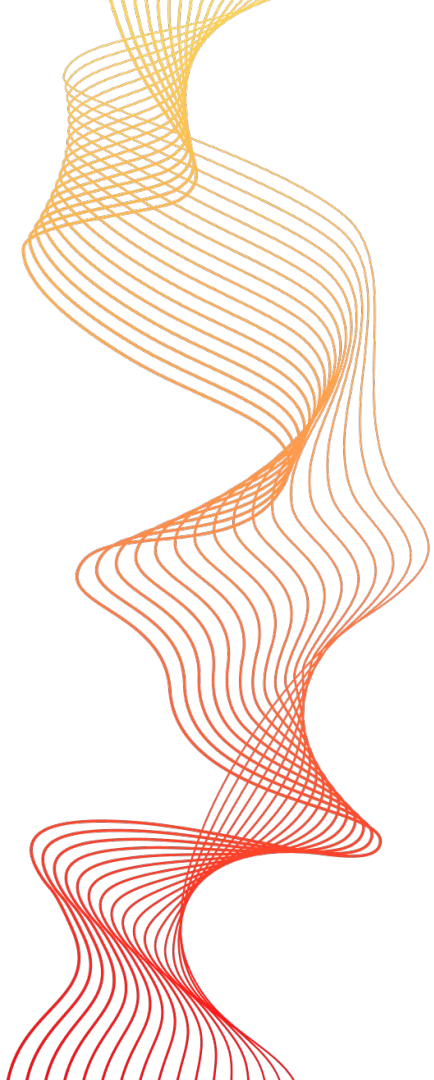

K-Means Clustering of Synthetic Dataset
Feature 2
Feature 1
10
5
0
−5
−10
−10
−8
−6
−4
−2
0
2
4
6

Iris Dataset Example

Use the Iris dataset from sklearn.datasets.

Visualize the dataset with two features ('sepal length', 'sepal width').

Apply k-means clustering to identify clusters.



Iris dataset

```
from sklearn.datasets import load_iris
iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
```

```
plt.scatter(df['sepal length (cm)'], df['sepal width (cm)'], cmap='viridis')
plt.title('Synthetic Dataset with 3 Clusters')
plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')
plt.show()
```

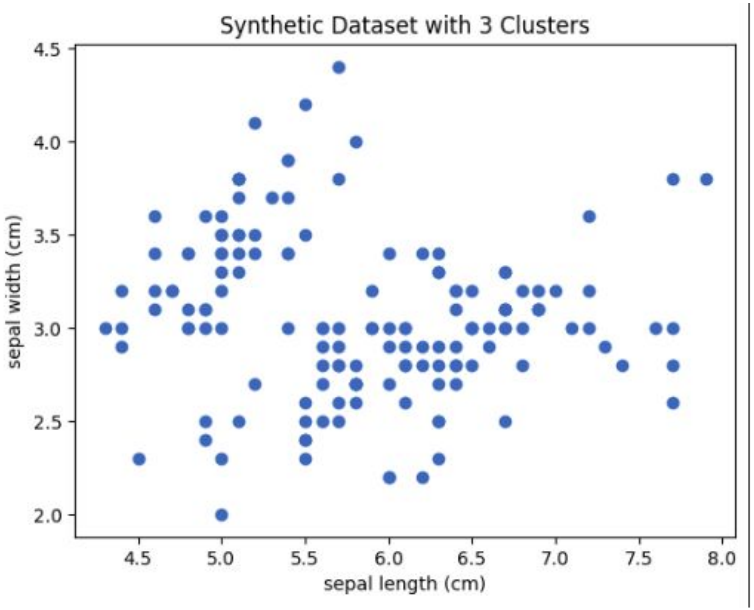
Explanation

from sklearn.datasets import load_iris: We're importing a dataset called Iris from scikit-learn. It's a famous dataset in the world of machine learning and contains measurements of different features of iris flowers.

iris = load_iris(): We're loading the Iris dataset into a variable called iris. It's like saying, "Hey, let's grab that Iris dataset and put it in a box named 'iris' for us to use."

df = pd.DataFrame(data=iris.data, columns=iris.feature_names): Now, we're turning the Iris dataset into a table (DataFrame). The data inside the Iris dataset, which includes measurements, is being organized into rows and columns. The column names are taken from the feature names of the Iris dataset. It's like saying, "Let's make a table out of the Iris data, and we'll name the columns after the features of the flowers."

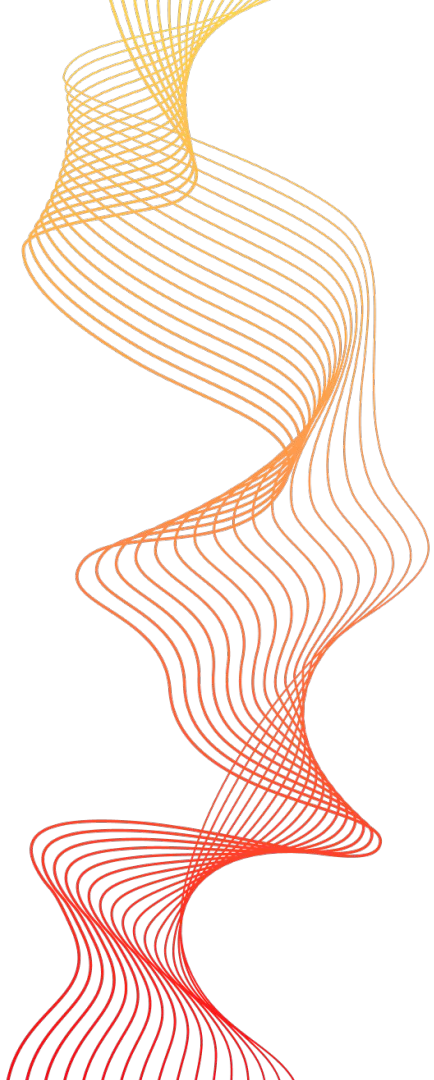
Iris sepal features plot



K-Means Clustering on Iris Data

Apply k-means clustering to the Iris dataset.

Visualize the clusters and cluster centers.



Applying K-means to Iris dataset

```
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(df)
df['Cluster'] = kmeans.labels_
```

```
plt.scatter(df['sepal length (cm)'], df['sepal width (cm)'], c=df['Cluster'], cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], marker='x', s=200, linewidths=3, color='red')
plt.title('K-Means Clustering of Iris Dataset')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
```

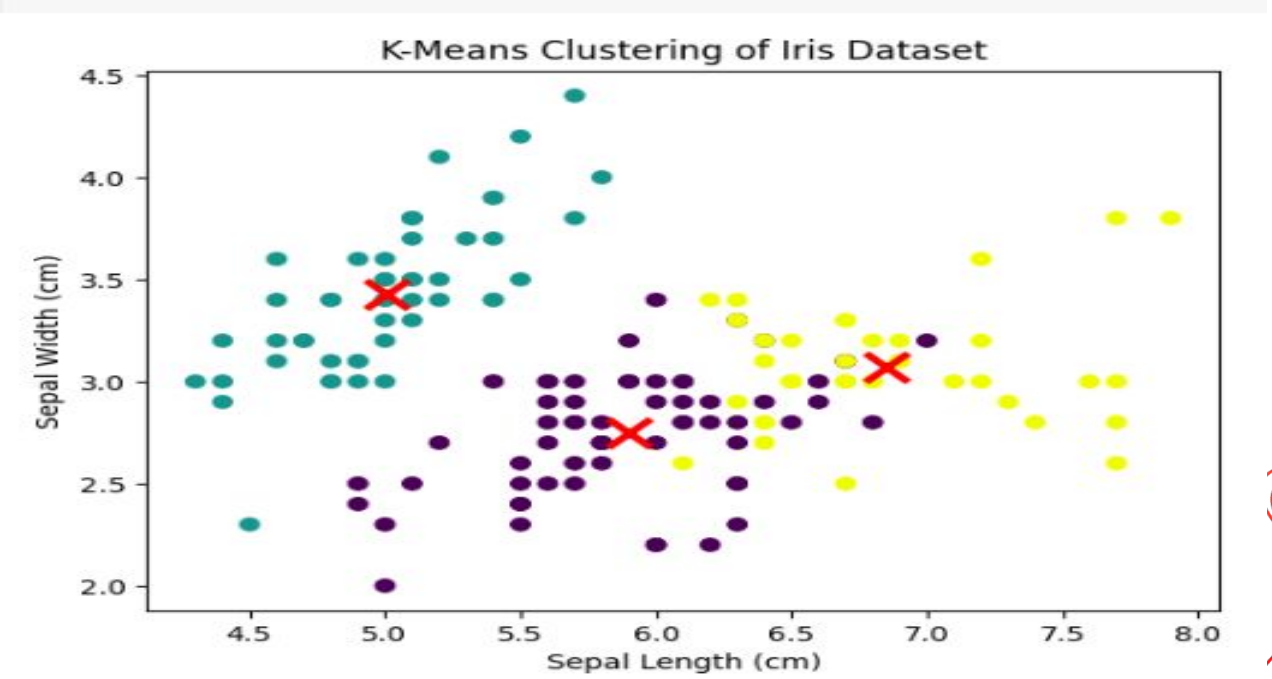
Explanation

kmeans = KMeans(n_clusters=3, random_state=42): We're creating another clustering tool named kmeans. This time, we're planning to group our data into three clusters. The random_state is like setting a starting point for randomness, making sure our results are consistent if we run this code again.

kmeans.fit(df): Now, we're telling our clustering tool to learn from the Iris dataset. It's like saying, "Hey, clustering tool, figure out how to group these flowers based on their measurements into three clusters."

df['Cluster'] = kmeans.labels_: We're taking the results of our clustering and adding a new column ('Cluster') to our Iris dataset table. This new column will tell us which cluster each flower belongs to according to our clustering model. It's like labeling each flower with its cluster group.

KMeans clustering of iris dataset



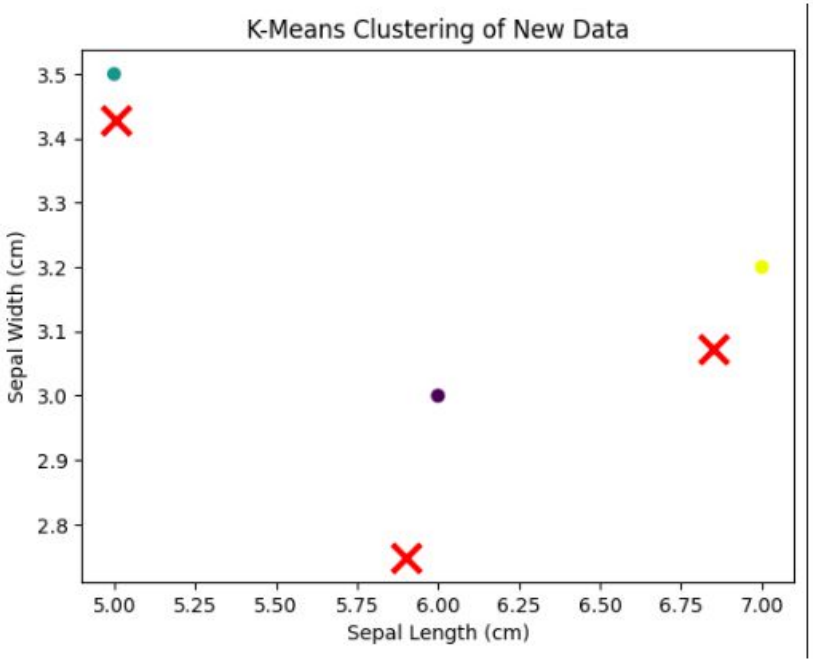
Kmeans predict

```
# Assuming you have new_data with the same features as the Iris dataset
new_data = pd.DataFrame([[5.0, 3.5, 1.5, 0.2], [6.0, 3.0, 4.0, 1.3], [7.0, 3.2, 5.5, 2.0]],
                        columns=iris.feature_names)

# Predict clusters for the new data
new_data['Cluster'] = kmeans.predict(new_data)

# Visualize the results for the new data
plt.scatter(new_data['sepal length (cm)'], new_data['sepal width (cm)'], c=new_data['Cluster'], cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], marker='x', s=200, linewidths=3, color='red')
plt.title('K-Means Clustering of New Data')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
```

Kmeans new data



Explanation part 1

new_data = pd.DataFrame([[5.0, 3.5, 1.5, 0.2], [6.0, 3.0, 4.0, 1.3], [7.0, 3.2, 5.5, 2.0]], columns=iris.feature_names): We're creating a new DataFrame (new_data) with some made-up measurements. It's like saying, "Let's imagine we have new flowers, and here are their measurements. We'll organize this into a table with the same features as the Iris dataset."

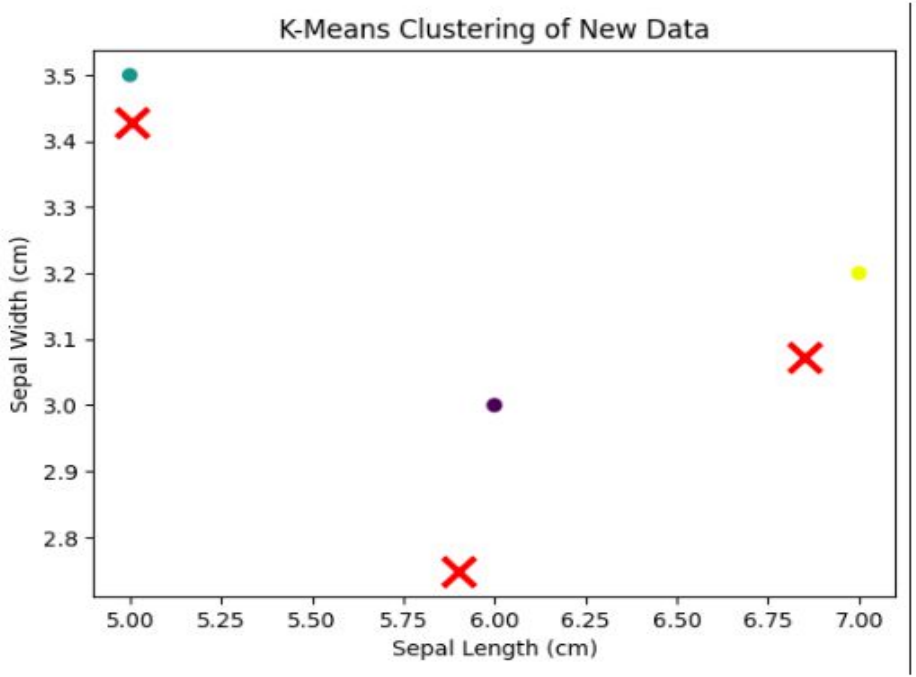
new_data['Cluster'] = kmeans.predict(new_data): We're using our trained clustering model (kmeans) to predict which cluster each of the new flowers belongs to. This is like saying, "Let's use our clustering knowledge to classify these new flowers into one of the clusters."

Explanation 2

Visualizing the results:

plt.scatter(new_data['sepal length (cm)'], new_data['sepal width (cm)'], c=new_data['Cluster'], cmap='viridis'): We're creating a scatter plot to visualize the new flowers, coloring them based on the clusters they belong to. It's like plotting the new flowers on a graph and using colors to show which group they are in.

Kmeans clustering of new data



CONCLUSION

- Recap of Week 2:
  - Covered the basics of unsupervised learning and the difference between Supervised and Unsupervised Learning

1.


BAZE UNIVERSITY
ABUJA
BAZE UNIVERSITY ABUJA
IN PARTNERSHIP WITH
DOMINEUM

THANK YOU

Q&A