□ Day 11 – K-Means Clustering (Unsupervised Learning)

📓 Session Overview

Today's session introduced us to one of the most important concepts in unsupervised machine learning: **K-Means Clustering**. Unlike classification, where labels are provided, clustering helps uncover hidden patterns in **unlabeled data**.
We explored this concept hands-on using a **real-world social media dataset**, focusing on engagement patterns in posts.

💡 What is K-Means Clustering?

K-Means is an unsupervised algorithm that groups similar data points into **k clusters** based on feature similarity.

🔁 **How It Works:**

1. Randomly selects **k centroids**
2. Assigns each data point to the nearest centroid (cluster)
3. Recalculates centroids based on new assignments
4. Repeats the above steps until centroids stabilize (converge)

It's widely used in areas like **customer segmentation**, **behavioral grouping**, and **market analysis**.

🗂 Dataset Used: Live.csv

We worked on a dataset representing **user interaction with social media posts**, including:

- Number of **likes, shares, comments**
- **Post type** (status_type)
- Total reactions per post

□ Initial Cleaning:

Dropped irrelevant or non-informative columns such as:

- Column1, Column2, Column3, Column4
- status_id, status_published

Then, we label-encoded the categorical column status_type and scaled the dataset using **MinMaxScaler** to normalize feature values for better clustering performance.

🔧 Preprocessing Code Snapshot

```
df = pd.read_csv("Live.csv")

df.drop(['Column1', 'Column2', 'Column3', 'Column4', 'status_id', 'status_published'], axis=1, inplace=True)

# Encode post type
```

```
le = LabelEncoder()
df['status_type'] = le.fit_transform(df['status_type'])
# Feature scaling
scaler = MinMaxScaler()
X = scaler.fit_transform(df)
```

📊 K-Means Clustering Implementation

We applied **K-Means** with k = 2 to divide the posts into 2 behavioral clusters.

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2, random_state=0)
kmeans.fit(X)
labels = kmeans.labels_
```

To evaluate clustering (since we had true labels), we compared the model's labels to the actual encoded values:

```
correct_labels = sum(le.transform(df['status_type']) == labels)
accuracy = correct_labels / float(len(df))
print(f"Accuracy score: {accuracy:.2f}")
```

📈 Result & Accuracy

- Correctly clustered: **604 out of 1000 posts**
- Approximate clustering **accuracy: 60%**

Note: In unsupervised learning, accuracy isn't always the best evaluation metric — since there's no real "ground truth" — but it can still offer insight when labels are known.

🔍 Finding the Optimal Number of Clusters – Elbow Method

We also used the **Elbow Method** to visualize the best value for k (number of clusters).

```
cs = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=0)
    kmeans.fit(X)
    cs.append(kmeans.inertia_)

# Plot the elbow curve
plt.plot(range(1, 11), cs)
plt.xlabel("Number of Clusters")
```

```
plt.ylabel("Inertia")

plt.title("Elbow Method for Optimal K")

plt.show()
```

📌 The plot helps us identify the point where increasing clusters stops giving significant improvements — called the **"elbow point"**.

✅ **Key Observations**

- K-Means was able to group similar posts based on engagement metrics.
- Clustering accuracy was moderate, but useful for pattern discovery.
- The **Elbow Method** is a reliable technique to find the ideal cluster count.
- Feature scaling was essential for proper distance-based grouping.

📝 Conclusion

- Understood how **K-Means** clusters data without any labels
- Preprocessed and scaled real-world social media data
- Implemented and evaluated clustering models
- Explored **evaluation methods** and **optimal k selection** using the Elbow Method

This session was a great step toward learning **pattern discovery** in unlabeled datasets and opened up the world of **unsupervised learning** in a practical way.