Step 2 – K-means Application

1. Apply the **K-means** algorithm with *k = 2*.
2. Visualize the resulting **clusters**, plotting the groups in different colors.
3. Compute the **confusion matrix**.
4. Compute the **Adjusted Rand Index (ARI)** for this clustering.

⚙️ When to Use K-means

- ✅ **Use it when:**
  - The data naturally forms **compact, spherical clusters**.
  - You know or can estimate the **number of clusters (k)** in advance.
  - You need a **fast, simple, and scalable** clustering method.
  - You're working with **continuous numerical data** (not categorical).
- 🚫 **Avoid it when:**
  - Clusters have **irregular shapes or very different sizes/densities**.
  - The data contains **many outliers**, which distort centroids.
  - The dataset mixes **categorical and numerical variables**.
  - You don't have a good idea of the correct **number of clusters (k)**.

🤖 Model Hyperparameters

- `n_clusters = 2` — number of clusters to form
- `n_init = "auto"` — number of random centroid initializations
- `max_iter = 300` — maximum number of iterations per run
- `random_state = 42` — ensures consistent cluster initialization

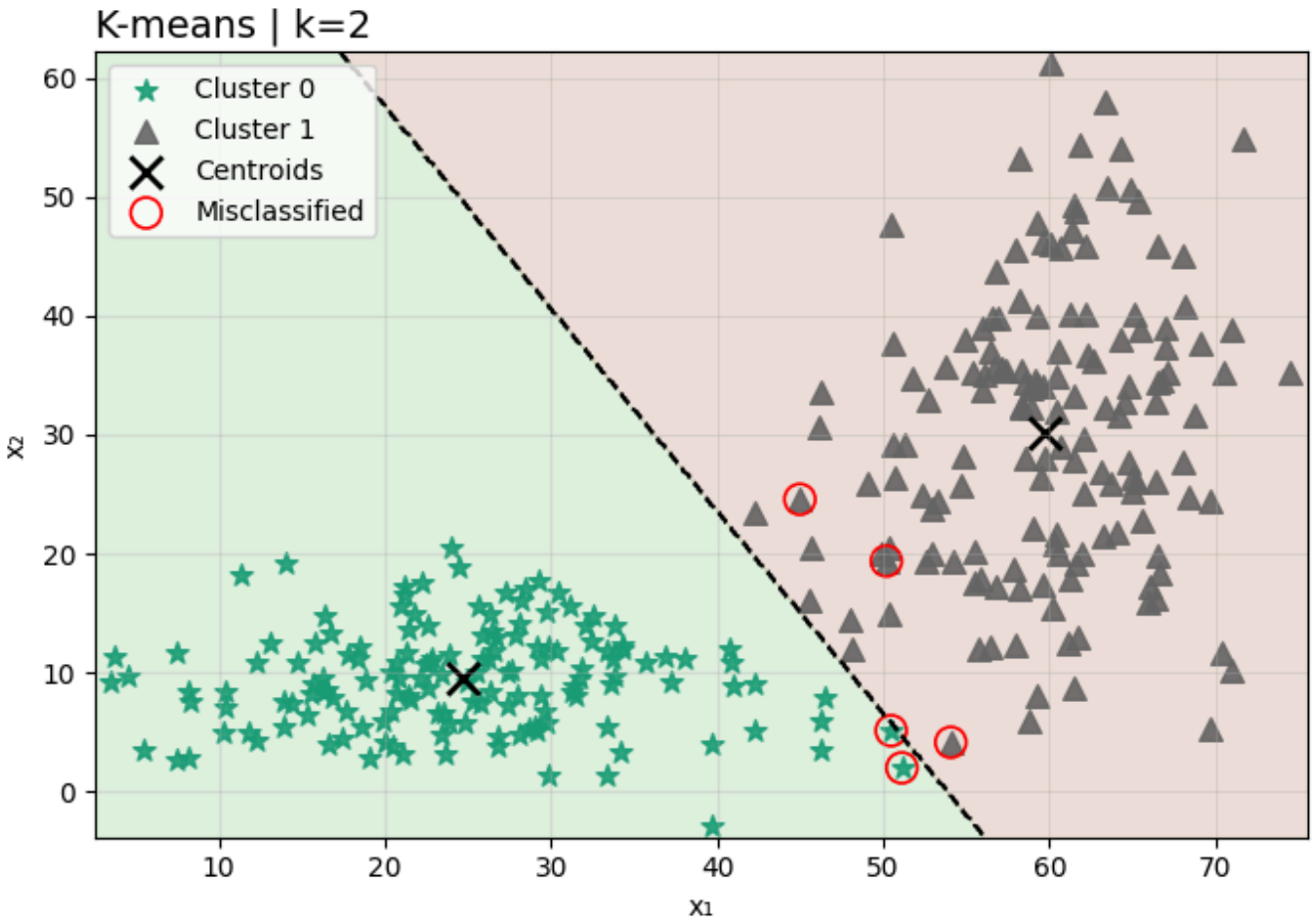
```
%run 00-setup.py
```

```
from ml.data import load_dataset
from tasks.kmeans import run_kmeans
from ml.viz import plt_clusters, plt_cmatrix
```

```
X, y, meta = load_dataset("../data/data_bivariate_gaussian.npz")
```

```
res = run_kmeans(X, y, params={"k": 2, "n_init": "auto", "max_iter": 300, "seed": 42})
```

```
plt_clusters(X, res["y_pred"], res["model"], y_true=y, title="K-means | k=2")
```

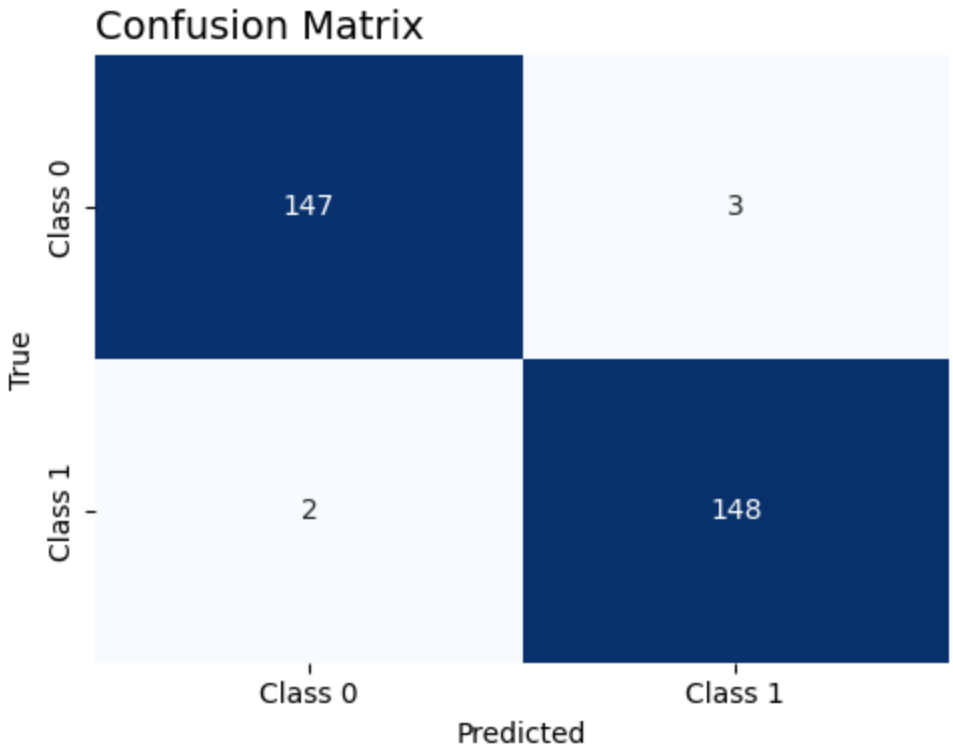



```
(<Figure size 700x500 with 1 Axes>,
 <Axes: title={'left': 'K-means | k=2'}, xlabel='x₁', ylabel='x₂'>)
```

```
cm = res["metrics"]["confusion_matrix"]

plt_cmatrix(cm=cm)

print("Adjusted Rand Index:", res["metrics"]["ari"])
```



Adjusted Rand Index: 0.9342244691686101