Step 8 – K-Nearest Neighbors (KNN)

1. Train a KNN model using the training data.
2. Plot the **decision boundary** for each chosen k.
3. Compare performance metrics (accuracy, precision, recall, F1-score) with other models.

⚙️ When to Use K-Nearest Neighbors (KNN)

- ✅ Use it when:
    - You want a **simple, non-parametric** model that adapts directly to data distribution.
    - The dataset is **small** and the input space is **low-dimensional**.
    - You expect **non-linear** decision boundaries that depend on local structure.
- 🚫 Avoid it when:
    - The dataset is **large** — inference is slow because every prediction compares all samples.
    - You have **many features** (curse of dimensionality).
    - The data is not **standardized** or contains **outliers**, since KNN is sensitive to distance scaling.

🤖 Model Hyperparameters

- `StandardScaler()` — feature scaling for distance-based metric
- `n_neighbors = 2` — number of nearest neighbors used in classification
- `weights = "uniform"` — all neighbors have equal voting weight

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

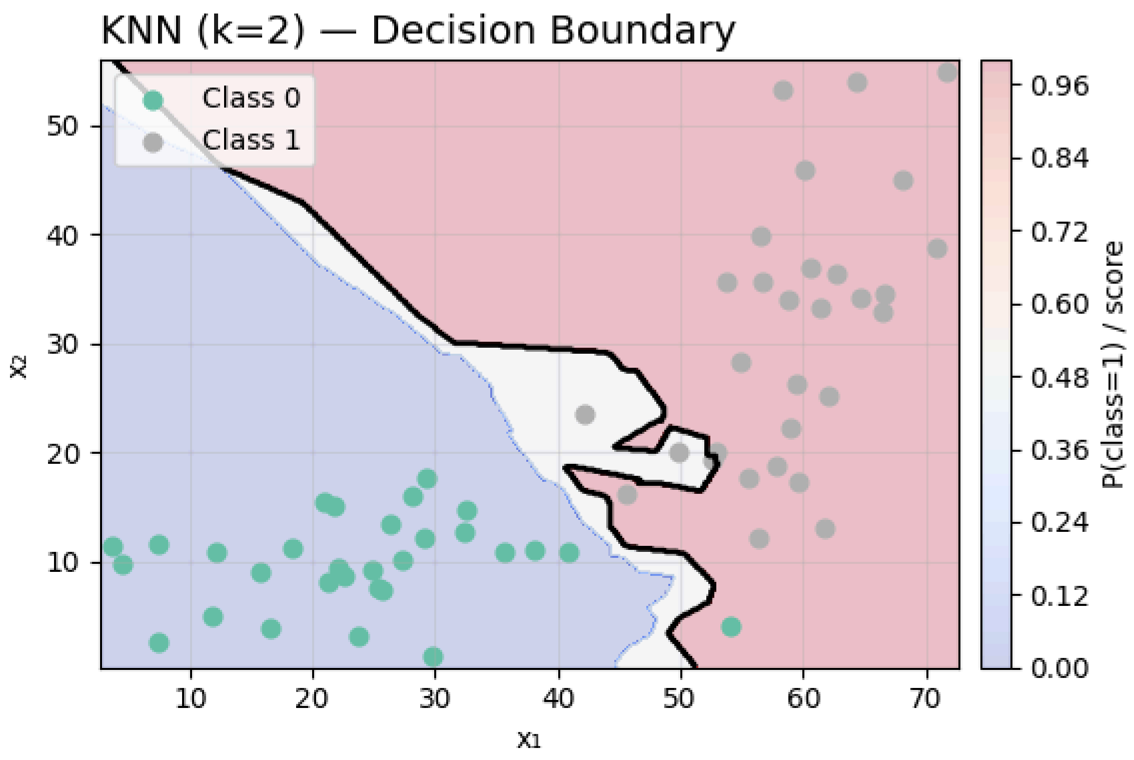
```
from ml.data import load_dataset
from tasks.knn import run_knn, run_knn_grid
from ml.viz import plt_dboundary, plt_cmatrix
from sklearn.model_selection import train_test_split
```

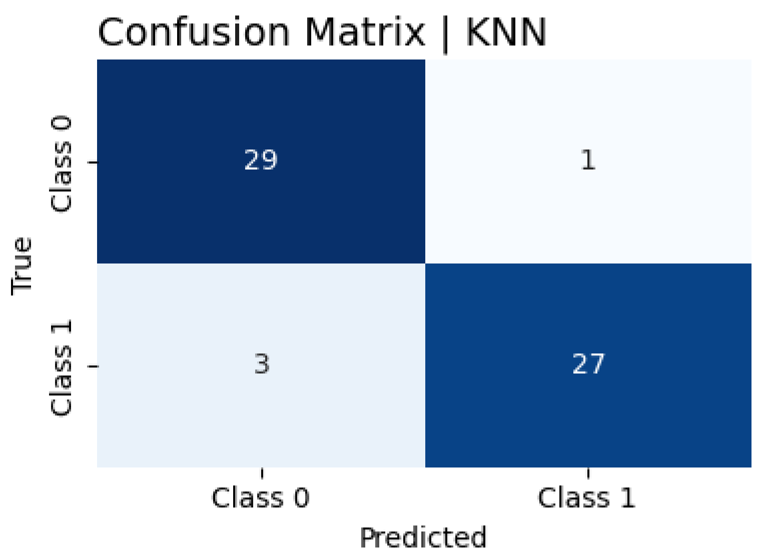
```
X, y, _ = load_dataset("../data/data_bivariate_gaussian.npz")

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

```
res = run_knn(X_train, y_train, X_test, y_test, params={"n_neighbors": 2, "weights": "uniform'

res["test"]["metrics"]
```

```
{'accuracy': 0.9333333333333333,
 'precision': 0.9642857142857143,
 'recall': 0.9,
 'f1': 0.9310344827586207}
```

```
plt_dboundary(res["model"], X_test, y_test, title="KNN (k=2) — Decision Boundary")

plt_cmatrix(
    y_true=y_test,
    y_pred=res["test"]["y_pred"],
    title="Confusion Matrix | KNN"
)
```





```
(<Figure size 400x300 with 1 Axes>,
 <Axes: title={'left': 'Confusion Matrix | KNN'}, xlabel='Predicted', ylabel='True'>)
```

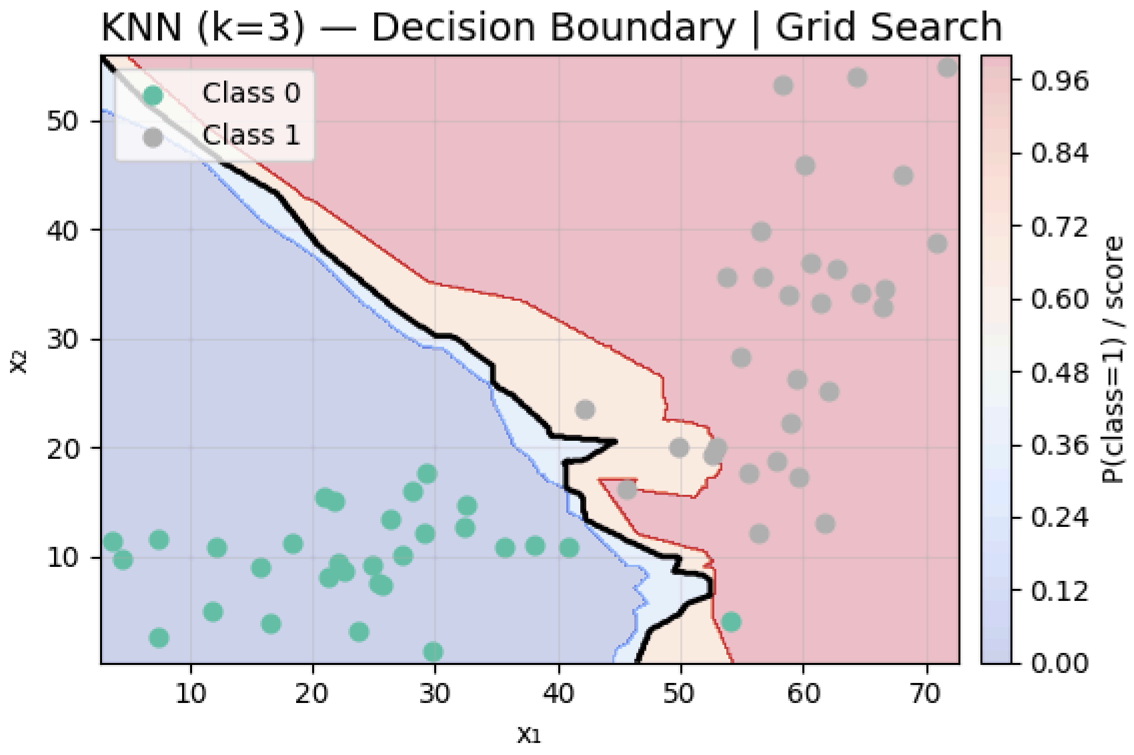
GRID

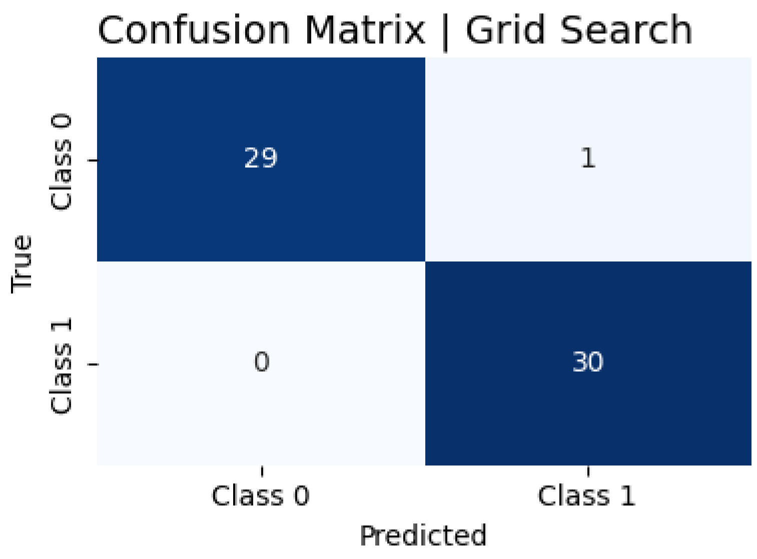
```
ks = range(1, 20, 1)
grid = run_knn_grid(X_train, y_train, X_test, y_test, ks=ks, weights="uniform")

best_k = max(ks, key=lambda k: grid[k]["test"]["metrics"]["accuracy"])
best_res = grid[best_k]

print("Best k:", best_k, "| Test metrics:")
best_res["test"]["metrics"]
```

```
Best k: 3 | Test metrics:
```

```
{'accuracy': 0.9833333333333333,
 'precision': 0.967741935483871,
 'recall': 1.0,
 'f1': 0.9836065573770492}
```

```
plt_dboundary(best_res["model"], X_test, y_test, title=f"KNN (k={best_k}) — Decision Boundary

plt_cmatrix(
    y_true=y_test,
    y_pred=best_res["test"]["y_pred"],
    title="Confusion Matrix | Grid Search"
)
```





Confusion Matrix | Grid Search
True
Class 0
Class 1
29
1
0
30
Class 0
Class 1
Predicted

```
(<Figure size 400x300 with 1 Axes>,
 <Axes: title={'left': 'Confusion Matrix | Grid Search'}, xlabel='Predicted', ylabel='True'>)
```