K-Fold CV

K-Fold Cross-Validation is one of the most popular and reliable techniques in ML to evaluate how well a model will perform on unseen data. It helps get a more stable and trustworthy estimate of model performance compared to a single train-test split.

Core Idea

Instead of splitting your data once into train and test sets (which can be lucky or unlucky depending on the split), you split the entire dataset into **k** roughly equal-sized parts ("folds").

You then train and evaluate the model **k times**:

- Each time, use **k-1 folds** for training
- Use the remaining **1 fold** as the test/validation set
- Every data point gets used exactly once as test data across all k runs

Finally, you average the performance scores (accuracy, MSE, F1, etc.) from all k folds → this average is your cross-validated performance estimate.

Why use k-fold CV?

- Makes better use of your data (every sample is used for both training and testing)
- Reduces the risk that your performance depends on a lucky/unlucky single split
- Gives a more realistic estimate of how the model will generalize
- Especially useful when you have limited data

Most common values for k

- **k = 5** — good balance between bias and computation time
- **k = 10** — very widely used (often considered the "default")
- **k = 3** — when you want faster computation
- **k = n** (where n = number of samples) → **Leave-One-Out CV** (LOOCV) — very accurate but very slow

Visual Explanation

FOLD 1	FOLD 2	FOLD 3	FOLD 4	FOLD 5
Test	Train	Train	Train	Train
Train	Test	Train	Train	Train
Train	Train	Test	Train	Train
Train	Train	Train	Test	Train

FOLD 1	FOLD 2	FOLD 3	FOLD 4	FOLD 5
Train	Train	Train	Train	Test

Scikit-Learn Implementation

```python
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np

# Example data
X = ...  # your features
y = ...  # your target

kf = KFold(n_splits=5, shuffle=True, random_state=42)
# shuffle=True is recommended unless you have time-series data

scores = []

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    model = LinearRegression()
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    score = mean_squared_error(y_test, y_pred)
    scores.append(score)

print("Mean CV MSE:", np.mean(scores))
print("Std of CV MSE:", np.std(scores))
```