ML Workflow and Best Practices

Data Splitting & Model Validation

Data Splitting & Model Validation

Learning Objectives

By the end of this lesson, you will be able to:

- Understand the importance of data splitting in ML workflows.
- Implement train-test splits and cross-validation techniques.
- Compare different model validation approaches and their trade-offs.
- Evaluate models effectively using appropriate performance metrics.

Overview

A well-validated model ensures reliable performance in real-world scenarios. This lesson focuses on how to **split data correctly** and apply **validation techniques** to prevent overfitting and underfitting.

Iris Flower Dataset

For this lesson, we will use the **Iris dataset**, a small, well-balanced dataset available in Scikit-learn. It consists of **150 samples** of iris flowers, each with **four numerical features**:

- Sepal length
- Sepal width
- Petal length
- Petal width

The target variable represents the species of the flower (**Setosa, Versicolor, Virginica**). This dataset is ideal for learning about data splitting and model validation because:

- **It's small and efficient**, allowing fast computations.
- **It has well-balanced classes**, making it suitable for train-test splitting and cross-validation.

Copy

```
from sklearn.datasets import load_iris
import pandas as pd

# Load sample dataset
data = load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target

print(df.head())
```

The Need for Data Splitting

When training a machine learning model, we must ensure that it generalizes well to unseen data. This requires **dividing our dataset** into subsets:

- **Training Set** – Used to train the model.
- **Validation Set** (optional) – Used for hyperparameter tuning.
- **Test Set** – Used to evaluate the model's final performance.

Implementing Train-Test Split

A **train-test split** is the simplest way to evaluate a model. It ensures that the model does not memorize the training data but generalizes to unseen examples.

Example Code:

Copy

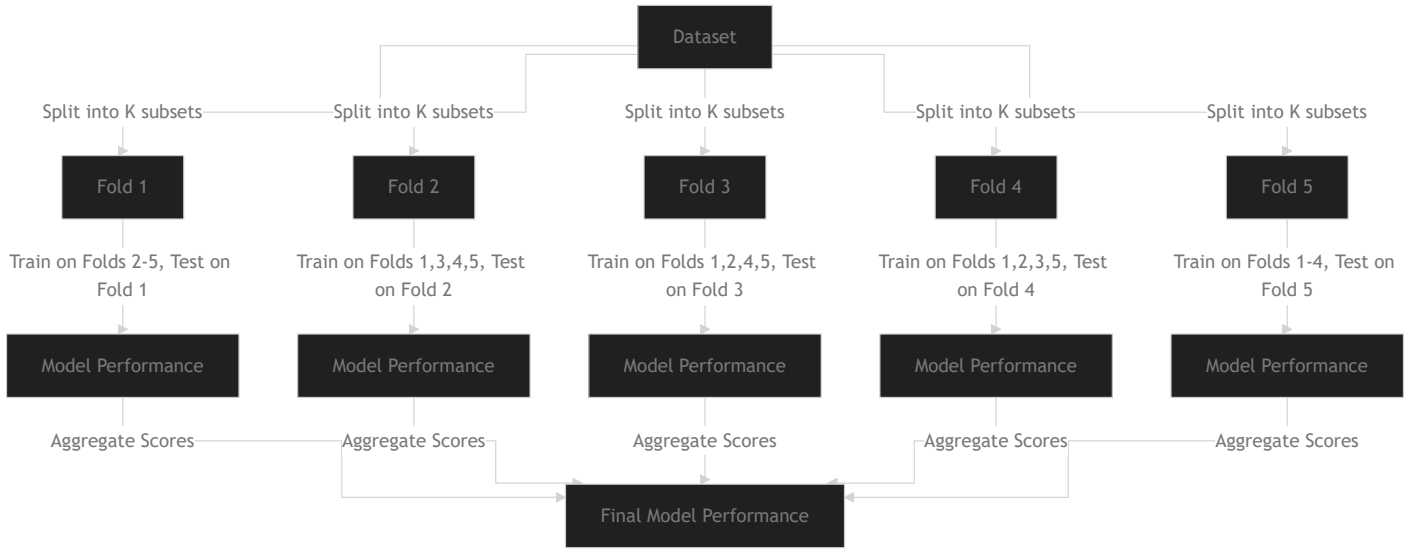
```
from sklearn.model_selection import train_test_split

# Split data into training (80%) and testing (20%)
train_data, test_data, train_labels, test_labels = train_test_split(
    df.drop(columns=['target']), df['target'], test_size=0.2, random_state=42)

print(f"Training Data Shape: {train_data.shape}")
print(f"Testing Data Shape: {test_data.shape}")
```

Understanding Cross-Validation

Cross-validation (CV) is a more **robust method** than a simple train-test split, as it helps mitigate variance in performance evaluation.

Types of Cross-Validation:

- **K-Fold Cross-Validation** – Splits data into **K subsets** and trains the model multiple times.
- **Stratified K-Fold** – Ensures that class distributions remain balanced across folds.
- **Leave-One-Out Cross-Validation (LOOCV)** – Uses all but one data point for training in each iteration.



Example Code:

Copy

```
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier

# Define a model
model = RandomForestClassifier()

# Perform 5-Fold Cross-Validation
scores = cross_val_score(model, train_data, train_labels, cv=5)

print("Cross-validation scores:", scores)
print("Mean CV Score:", scores.mean())
```

Choosing the Right Validation Approach

Validation Method	Pros	Cons
Train-Test Split	Simple, fast	High variance, results depend on split
K-Fold Cross-Validation	More reliable, less variance	Computationally expensive
Stratified K-Fold	Ensures balanced class representation	Still requires tuning

Validation Method	Pros	Cons
Leave-One-Out CV	Uses all data for training	Extremely slow on large datasets

Train-Test Split & CV Practice

Task: Modify the provided code to use different test sizes (e.g., 10%, 30%) and compare results.

- How does changing the test size affect model accuracy?
- Try switching from **train-test split** to **K-Fold Cross-Validation** and analyze score variation.

Key Takeaways

- **Train-test splits** prevent data leakage and help estimate model performance.
- **Cross-validation** is a more reliable method for evaluation, reducing the impact of random splits.
- Choosing the right validation approach depends on dataset size, model complexity, and computational constraints.

< Previous

© 2025 General Assembly
Attributions

Next >