📆 DAY 15 — Train-Test Split & Data Leakage

Goal : Train models honestly and avoid cheating

1️⃣ What is Train-Test Split ? (foundation)

When training ML models, you must answer:

> “can my model predict unseen data?”

To test this, we **split data** into:

Part	Purpose
Training set	Model Learns patterns
Test set	Model is evaluated

Typical split

```
80% → Train
20% → Test
```

2️⃣ How to Split Data (Correct Way)

```
from sklearn.model_selection import train_test_split

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

🧠 Important Parameters

- `test_size` → how much data to hold out
- `random_state` → reproducibility (VERY IMPORTANT)

4️⃣ What Is Data Leakage? (🔥 VERY IMPORTANT 🔥)

Data Leakage = using future or test information during training

Your model is **cheating**.

Examples of leakage:

- Scaling entire dataset before splitting
- Using target information in features
- Feature engineering using test data

> If leakage exits → evaluation is meaningless.

5️⃣ Common Data Leakage Mistake (CRITICAL)

❌ WRONG

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)  # ❌ uses ALL data

X_train, X_test = train_test_split(X_scaled)
```

✅ CORRECT

```
X_train, X_test, y_train, y_test = train_test_split(X, y)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

🧠 Why?

- Test data must simulate **unseen future data**.
- `fit()` must NEVER see test data

6️⃣ Leakage Through Feature Engineering

❌ **WRONG**

```
df["salary_mean"] = df["salary"].mean()
```

(uses info from future/ test rows)

✅ **CORRECT**

Compute stats on **train only**.

7️⃣ Stratified Split (VERY IMPORTANT FOR CLASSIFICATION)

Problem

Imbalanced classes → random split may break distribution

Solution

```
train_test_split(
    X, y,
    stratify=y,
    test_size = 0.2
)
```

🧠 **Why this matters**

Ensures:

- Train & test have similar class ratios
- Fair evaluation

8️⃣ Validation Set (Next-Level Concept)

Instead of 2 splits:

```
Train → Learn
Validation → Tune
Test Final → evaluation
```

Typical:

- 70% Train

- 15% Validation
- 15% Test

9️⃣ ML Pipeline (Leak-Proof Way)

```
from sklearn.pipline import Pipeline

pipeline = Pipeline([
    ("scaler", StrandardScaler()),
    ("model", LogisticRegression())
])

pipeline.fit(X_train, y_train)
```

🧠 **Why pipelines matter**

- Prevent leakage
- Cleaner code
- Industry standard

🚨 VERY IMPORTANT WARNING

If you ignore data leakage:

- **Your model will fail in production**
- **Accuracy numbers become lies**