
📅 Day 6: Data Splitting and Feature Engineering

🎯 Goal

By the end of Day 7, interns will:

- Understand **why** and **how** to split data for model training and evaluation.
- Apply **feature engineering techniques** to improve model performance.
- Learn **how to prepare raw data** into usable inputs for machine learning models.

🧩 1. What is Data Splitting?

✅ Why We Split the Dataset

In Machine Learning, we **never train and test on the same data**. Why?

- To **prevent overfitting** (learning too well on training data)
- To evaluate how well your model generalizes to new data

📦 Types of Splits

- **Training Set** – used to train the model (usually 70–80%)
- **Test Set** – used to evaluate the model (usually 20–30%)

```
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)
```

🧪 2. What is Feature Engineering?

Feature engineering is the process of **transforming raw data** into features that are more informative for your machine learning model.

🧰 3. Types of Feature Engineering

A. Feature Scaling (Standardization)

Why?

- Some algorithms (like Linear Regression, KNN) are sensitive to feature scale.

📏 **Standardization Formula**:

$z=(x-\mu)\sigma z = \frac{(x - \mu)}{\sigma}$

```
from sklearn.preprocessing import StandardScaler

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

B. Polynomial Features (Interaction Terms)

Use when:

- Relationships between variables may not be linear
- Polynomial regression helps model non-linearity

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X_scaled)
```

> Adds combinations like:
> `x1`, `x2`, `x1^2`, `x2^2`, `x1*x2`

C. One-Hot Encoding (for categorical variables)

Not used in Day 7 since California dataset is numeric, but here's how it works:

```
pd.get_dummies(df['Category'])
```

D. Feature Selection

Why?

- Reduce dimensionality
- Remove irrelevant or noisy features

```
from sklearn.feature_selection import SelectKBest, f_regression

selector = SelectKBest(score_func=f_regression, k=5)
X_selected = selector.fit_transform(X_scaled, y)
```

🔍 Mini Project Summary

Using the **California Housing** dataset, interns will:

1. Load the dataset
2. Split it into train/test
3. Standardize the features
4. Create polynomial features
5. Optionally select the top 5 features

🎮 Gamification / Intern Challenge Ideas

- **Experiment** with different degrees in PolynomialFeatures.
- **Compare models**: Try building a Linear Regression on scaled vs. unscaled vs. poly-transformed features.
- **Leaderboard**: Who gets the lowest MSE on test set with top 5 features?

🧠 Key Takeaways

- Always split data to avoid overfitting.
- Scaling and feature engineering are critical for many ML algorithms.
- Polynomial features help capture non-linear patterns.
- Feature selection can boost speed and reduce overfitting.

📅 Classification – Logistic Regression

🎯 Objective

Understand the fundamentals of classification and implement **logistic regression** using a real-world binary dataset.

🔍 What is Classification?

Classification is a type of **Supervised Learning** where:

- The **output** is **categorical** (e.g., Yes/No, Spam/Not Spam, Malignant/Benign)
- The goal is to **predict the correct class** for new data

✨ Examples of Classification:

Problem	Input Features	Output
Email Spam Detection	Email content, sender	Spam / Not Spam
Tumor Classification	Cell size, nucleus shape	Malignant / Benign

---

⚙️ Logistic Regression

Despite the name, **Logistic Regression** is a **classification algorithm**, not a regression algorithm.

🔐 Core Idea:

- Instead of predicting a continuous value, we predict a **probability between 0 and 1**
- If the predicted probability > 0.5, class = 1; else, class = 0

📈 It uses the Sigmoid Function:

σ(z)=11+e−z\sigma(z) = \frac{1}{1 + e^{-z}}

Where `z` is a linear combination of input features (`z = w*x + b`)

📘 Dataset Used: Breast Cancer Dataset

- Built-in dataset in `sklearn`
- Predict whether a tumor is **malignant** or **benign**
- Input features: Mean radius, texture, area, etc.
- Target: 0 = malignant, 1 = benign

from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()

🪜 Step-by-Step Implementation

✂️ Step 1: Split the Dataset

Use `train_test_split()` to divide data into training and test sets.

```
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)
```

- **Training set**: Used to train the model
- **Test set**: Used to evaluate performance

🔄 Step 2: Feature Scaling

Why scale?

- Logistic regression is sensitive to feature magnitudes

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

🤖 Step 3: Train Logistic Regression Model

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train_scaled, y_train)
```

- Fits the logistic function to the training data
- Learns the weights `w` and bias `b`

📊 Step 4: Evaluate the Model

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

y_pred = model.predict(X_test_scaled)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

🔍 Evaluation Metrics:

Metric	What It Measures
Accuracy	Overall correctness
Precision	How many predicted positives are correct
Recall	How many actual positives were correctly predicted
F1-Score	Balance between precision and recall

✅ Summary

Concept	Description
Classification	Predict categories
Logistic Regression	Maps inputs to probabilities
Sigmoid Function	Converts output to 0–1
Feature Scaling	Normalizes input features
Evaluation	Accuracy, precision, recall, F1

🧠 Intern Exercise

1. Change test size to 30%. Does accuracy change?
2. Print model coefficients using `model.coef_`. Which features are most impactful?

3. Add a challenge: Try with another binary dataset like `load_digits()` (2-class version).
