Model Drift — Advanced

1️⃣ Label Drift (Target Drift)

What is Label Drift?

Label Drift occurs when the **distribution of target labels changes over time**, even if input features remain stable.

$$P(Y_{train}) \neq P(Y_{prod})$$

This is **the most dangerous drift** because:

- Feature drift tests won't detect it
- Model confidence can stay high
- Business metrics silently degrade

Why Label Drift Happens

- Economic changes (recession → more loan defaults)
- User behavior changes
- Policy or regulation changes
- Feedback loops caused by the model itself

Key Property (IMPORTANT)

⚠️ **Labels are delayed**

You **cannot detect label drift instantly**.

You must wait for ground truth.

This is why:

- Feature drift = real-time
- Label drift = lagged monitoring

How Label Drift Is Detected (Practically)

You do **NOT** compare distributions directly.

Instead, you monitor:

- Accuracy / F1 / AUC over time
- Error rate trend
- Calibration decay

Using **rolling windows**.

Python: Label Drift via Performance Decay

Example: Binary Classification

```
import numpyas np
from sklearn.metricsimport accuracy_score, f1_score

# Simulated predictions and true labels
np.random.seed(42)

y_true_past = np.random.binomial(1,0.3,1000)
y_pred_past = y_true_past.copy()# good model

y_true_recent = np.random.binomial(1,0.5,1000)# label shift
y_pred_recent = np.random.binomial(1,0.3,1000)# model unchanged

# Metrics
past_f1 = f1_score(y_true_past, y_pred_past)
```

```
recent_f1 = f1_score(y_true_recent, y_pred_recent)

print(f"Past F1: {past_f1:.3f}")
print(f"Recent F1: {recent_f1:.3f}")

if recent_f1 < past_f1 *0.9:
print("🚨 LABEL DRIFT DETECTED (Performance Decay)")
else:
print("✅ Performance Stable")
```

When to Act

Scenario	Action
Metric drop < 5%	Monitor
5–15%	Retrain
> 15%	Redesign model / features

2️⃣ Multivariate Data Drift

What is Multivariate Drift?

Multivariate drift occurs when **relationships between features change**, even if individual feature distributions look stable.

$$P(X_1, X_2, ..., X_n) \neq P'(X_1, X_2, ..., X_n)$$

Univariate tests (KS) **fail here**.

Real Example

- Age distribution: same

- Salary distribution: same
- **Age–Salary correlation changed**

Model breaks.

Technique: PCA Reconstruction Error

How PCA-Based Drift Detection Works

1. Fit PCA on training data
2. Reconstruct data using limited components
3. Measure reconstruction error
4. Compare baseline vs production error

If error increases → drift

Python: Multivariate Drift with PCA

```
import numpyas np
from sklearn.decompositionimport PCA

np.random.seed(42)

# Training data (correlated features)
X_train = np.random.multivariate_normal(
    mean=[0,0],
    cov=[[1,0.9],
         [0.9,1]],
    size=1000
)

# Production data (correlation changed)
X_prod = np.random.multivariate_normal(
    mean=[0,0],
    cov=[[1,0.2],
        [0.2,1]],
    size=1000
)

# Fit PCA on training data
pca = PCA(n_components=1)
pca.fit(X_train)

# Reconstruction
X_train_recon = pca.inverse_transform(pca.transform(X_train))
X_prod_recon = pca.inverse_transform(pca.transform(X_prod))

# Reconstruction error
train_error = np.mean(np.linalg.norm(X_train - X_train_recon, axis=1))
prod_error = np.mean(np.linalg.norm(X_prod - X_prod_recon, axis=1))

print(f"Train Error: {train_error:.4f}")
print(f"Prod Error: {prod_error:.4f}")

if prod_error > train_error *1.2:
print("🚨 MULTIVARIATE DRIFT DETECTED")
else:
print("✅ No Multivariate Drift")
```

Why This Works

- PCA captures **feature interaction structure**
- Reconstruction error increases when relationships break
- Scales to many features