

1.4 결과 해석

1.4.1 MSE와 R² 계

회귀 모델의 성능을 평가할 때 대표적으로 사용되는 지표는 다음 두 가지입니다.

(1) 평균 제곱 오차 (Mean Squared Error, MSE)

MSE는 실제값과 예측값 사이의 **제곱 차이의 평균**입니다. 작을수록 예측이 실제와 가깝다는 뜻입니다.

$$MSE = \frac{1}{m} \sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)})^2$$

(2) 결정 계수 (R² Score)

R² 는 예측값이 실제값을 얼마나 잘 설명하는지를 나타냅니다. **1에 가까울수록 좋은 성능**입니다.

$$R^2 = 1 - \frac{\sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^m (y^{(i)} - \bar{y})^2} p$$

1.4.2 평가 코드

```
#1.4.2

from sklearn.metrics import mean_squared_error, r2_score

# 네 가지 회귀 모델 평가
models = {
    "선형 회귀": (lin_reg, X),
    "다항 회귀": (lin_poly, X_poly),
    "릿지 회귀": (ridge, X_poly),
    "라쏘 회귀": (lasso, X_poly),
}

print("모델\t\tMSE\t\tR²")
print("-" * 50)
for name, (model, X_input) in models.items():
    y_pred = model.predict(X_input)
    mse = mean_squared_error(y, y_pred)
    r2 = r2_score(y, y_pred)
    print(f"{name:<10s}\t{mse:.4f}\t{r2:.4f}")
```

모델	MSE	R ²
선형 회귀	0.6133	0.0022
다항 회귀	0.5405	0.1207
릿지 회귀	0.5433	0.1162
라쏘 회귀	0.5776	0.0603

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.model_selection import learning_curve
```

```

# 1. 데이터 생성 (복잡도 + 잡음 )
np.random.seed(42)
X = np.linspace(-1.5, 1.5, 200).reshape(-1, 1)
y = (
    0.3 * np.sin(9 * np.pi * X).ravel() # 고주파수로 변경
    + 0.3 * X.ravel()**5                # 곡선성
    + 0.6 * np.random.randn(200)        # 잡음
)

# 2. 모델 정의 (15차 다항)
lin_reg = LinearRegression()
poly15_lin = make_pipeline(PolynomialFeatures(degree=15), LinearRegression())
poly15_ridge = make_pipeline(
    PolynomialFeatures(degree=15),
    StandardScaler(),
    Ridge(alpha=0.005)
)

poly15_lasso = make_pipeline(
    PolynomialFeatures(degree=15),
    StandardScaler(),
    Lasso(alpha=0.001, max_iter=100000)
)

models = {
    "선형 회귀 (1차)": (lin_reg, X),
    "다항 회귀 (15차)": (poly15_lin, X),
    "릿지 회귀 (15차,  $\alpha=0.005$ )": (poly15_ridge, X),
    "라쏘 회귀 (15차,  $\alpha=0.001$ )": (poly15_lasso, X),
}

# 3. 학습 곡선 시각화
fig, axes = plt.subplots(2, 2, figsize=(14, 10))
axes = axes.ravel()

for idx, (name, (model, X_input)) in enumerate(models.items()):
    train_sizes, train_scores, val_scores = learning_curve(
        model, X_input, y,
        train_sizes=np.linspace(0.1, 1.0, 10),
        scoring='neg_mean_squared_error',
        cv=5,
        shuffle=True,
        random_state=42
    )

    # 음수 제거 + log 스케일 호환성 확보
    epsilon = 1e-6
    train_errors = np.clip(-np.mean(train_scores, axis=1), epsilon, None)
    val_errors = np.clip(-np.mean(val_scores, axis=1), epsilon, None)

    ax = axes[idx]
    ax.plot(train_sizes, train_errors, 'o-', label="훈련 MSE", color='orange', linewidth=2)
    ax.plot(train_sizes, val_errors, 'o--', label="검증 MSE", color='orangered', linewidth=2)
    ax.set_title(f"{name}")
    ax.set_xlabel("훈련 샘플 수")
    ax.set_ylabel("평균 제곱 오차 (MSE)")

```

```
# 조건부 로그 스케일 적용
if "선형 회귀" not in name:
    ax.set_yscale("log")

ax.legend()
ax.grid(True, which='both', linestyle='--', linewidth=0.5)
```

```
plt.tight_layout()
plt.savefig("learning_curve_poly15_mixed_scale.png")
plt.show()
```

