Activity: Visualizing Ridge Regression

Tutorial-2

Objective

Understand how **Ridge Regression (L2 Regularization)** prevents overfitting by visualizing how the penalty term α (alpha) "tames" the model's complexity.

Part 1: The Setup

We will fit a high-degree polynomial (Degree 10) to a noisy sine wave. A standard Linear model will try to hit every noise point, creating wild "wiggles." We will use Ridge Regression to smooth it out.

Instructions: Copy and run the following code in your Python environment (Jupyter Notebook or Colab).

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

# 1. Generate Synthetic Data (A sine wave with noise)
np.random.seed(42)
X = np.sort(np.random.rand(20) * 5)[:, np.newaxis] # 20 random points
y = np.sin(X).ravel() + np.random.normal(0, 0.2, 20) # True function + noise

# 2. Define a Test Set (Smooth line for plotting)
X_plot = np.linspace(0, 5, 100)[:, np.newaxis]

# 3. Compare 3 Models
plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='black', label='Noisy Data')

# Model A: Standard Linear Regression (No Penalty)
# We use degree=10 to force the model to overfit
model_ols = make_pipeline(PolynomialFeatures(10), LinearRegression())
model_ols.fit(X, y)
plt.plot(X_plot, model_ols.predict(X_plot), color='red', linewidth=2, label='
    OLS (No Penalty)')

# Model B: Ridge with Low Penalty
model_ridge_low = make_pipeline(PolynomialFeatures(10), Ridge(alpha=0.1))
model_ridge_low.fit(X, y)
plt.plot(X_plot, model_ridge_low.predict(X_plot), color='blue', linestyle='--',
     label='Ridge (alpha=0.1)')

# Model C: Ridge with High Penalty
model_ridge_high = make_pipeline(PolynomialFeatures(10), Ridge(alpha=10.0))
model_ridge_high.fit(X, y)
plt.plot(X_plot, model_ridge_high.predict(X_plot), color='green', label='Ridge
    (alpha=10)')

```

```
plt.title('Taming the Wiggles: OLS vs Ridge')
plt.ylim(-2, 2)
plt.legend()
plt.show()
```

Part 2: Exploration Questions

Run the code above and analyze the graph to answer the following:

1. **The "Explosion" (Variance):** Look at the **Red Line** (OLS).
   - Why does it curve wildly at the edges compared to the Green line?
   - Does this model have High Bias or High Variance?
2. **The "Flatline" (Bias):**
   - Modify the code for **Model C**. Change `alpha=10.0` to `alpha=1000.0`.
   - Re-run the code. What happens to the Green line? Why?
3. **The "Sweet Spot":**
   - Try to find an `alpha` value for **Model B** (Blue dashed line) that fits the pattern smoothly without being too flat or too wiggly.
   - Optimal Alpha found: _____

Part 3:

Why did the Red line wiggle so much? It used massive coefficients to twist the line. Run this snippet to see the difference in the "budget" used by the models:

```
print("Sum of squared coefficients (OLS):", (model_ols.steps[1][1].coef_**2).
    sum())
print("Sum of squared coefficients (Ridge):", (model_ridge_high.steps[1][1].
    coef_**2).sum())
```

- **OLS Budget:** _____ (Likely Huge)
- **Ridge Budget:** _____ (Likely Small)