**Bansilal Ramnath Agarwal Charitable Trust’s**

**Vishwakarma Institute of Technology, Pune-37  *(An autonomous institute of Savitribai Phule Pune University)***

******

**STATISTICAL INFERENCE LAB 1**

|  |  |
| --- | --- |
| **Year** | **Third** |
| **Branch** | **AI & DS** |
| **Division** | **AI-A** |
| **Batch** | **3** |
| **PRN** | **12210992** |
| **Roll no.** | **58** |
| **Name** | **Pavanraje Namdev Gaikwad** |
| **Subject** | **Statistical Inference** |

**Regularization Techniques: Lasso and Ridge**

**What are Regularization Techniques (Lasso and Ridge)?**

Regularization techniques are essential in machine learning to prevent overfitting, especially in complex models like linear regression. Overfitting occurs when a model captures the noise in the dataset, leading to poor generalization on unseen data. To address overfitting, Lasso and Ridge regression introduce penalties on the size of the coefficients, shrinking them towards zero and simplifying the model.

* **Lasso (Least Absolute Shrinkage and Selection Operator)**: Lasso regression uses L1 regularization, which adds a penalty equal to the absolute value of the magnitude of coefficients. This can lead to feature selection by shrinking some coefficients to zero, effectively removing irrelevant features.
* **Ridge**: Ridge regression uses L2 regularization, which adds a penalty proportional to the square of the magnitude of coefficients. Unlike Lasso, Ridge tends to shrink coefficients uniformly but does not set them to zero.

**Algorithm and Code Explanation**

**Base Model: Linear Regression without Regularization**

Before implementing Lasso and Ridge, let's start with a simple linear regression model on a dataset.

**Step 1: Import Libraries and Load Data**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

# Load the dataset (for this example, we'll create a synthetic dataset)

np.random.seed(0)

X = np.random.rand(100, 10)

y = 5 \* X[:, 0] + 2 \* X[:, 1] + np.random.randn(100) # Linear relationship with noise

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

**Step 2: Train the Base Linear Regression Model**

# Train a base linear regression model

base\_model = LinearRegression()

base\_model.fit(X\_train, y\_train)

# Make predictions

y\_pred\_base = base\_model.predict(X\_test)

# Compute the mean squared error

mse\_base = mean\_squared\_error(y\_test, y\_pred\_base)

print(f"Base Model MSE: {mse\_base}")

* **Explanation**: We create a synthetic dataset with 10 features and train a basic linear regression model. The model is evaluated using Mean Squared Error (MSE), a common metric for regression models.

**Ridge Regression (L2 Regularization)**

**Step 3: Implement Ridge Regression**

from sklearn.linear\_model import Ridge

# Train a Ridge regression model

ridge\_model = Ridge(alpha=1.0) # alpha is the regularization strength

ridge\_model.fit(X\_train, y\_train)

# Make predictions

y\_pred\_ridge = ridge\_model.predict(X\_test)

# Compute the mean squared error

mse\_ridge = mean\_squared\_error(y\_test, y\_pred\_ridge)

print(f"Ridge Model MSE: {mse\_ridge}")

* **Explanation**: Ridge regression adds an L2 penalty term to the cost function, which prevents large coefficients and helps mitigate overfitting. The alpha parameter controls the strength of regularization.

**Lasso Regression (L1 Regularization)**

**Step 4: Implement Lasso Regression**

from sklearn.linear\_model import Lasso

# Train a Lasso regression model

lasso\_model = Lasso(alpha=0.1) # alpha is the regularization strength

lasso\_model.fit(X\_train, y\_train)

# Make predictions

y\_pred\_lasso = lasso\_model.predict(X\_test)

# Compute the mean squared error

mse\_lasso = mean\_squared\_error(y\_test, y\_pred\_lasso)

print(f"Lasso Model MSE: {mse\_lasso}")

* **Explanation**: Lasso regression uses L1 regularization, which can shrink some coefficients to zero. This leads to feature selection, where irrelevant features are excluded from the model.

**Step 5: Compare Model Performance**

**Plot the Coefficients**

# Plot the coefficients for comparison

plt.figure(figsize=(12, 6))

plt.plot(base\_model.coef\_, 'o-', label='Base Model')

plt.plot(ridge\_model.coef\_, 'o-', label='Ridge')

plt.plot(lasso\_model.coef\_, 'o-', label='Lasso')

plt.xlabel('Features')

plt.ylabel('Coefficient Value')

plt.title('Comparison of Coefficients')

plt.legend()

plt.show()

* **Explanation**: This plot compares the coefficients of the base model, Ridge, and Lasso. Ridge shrinks coefficients uniformly, while Lasso shrinks some to zero, leading to feature selection.

**Results Comparison**

print(f"Base Model MSE: {mse\_base}")

print(f"Ridge Model MSE: {mse\_ridge}")

print(f"Lasso Model MSE: {mse\_lasso}")

**Conclusion**

* **Base Model**: Without regularization, the model fits the data well but may overfit, as the coefficients can grow large, capturing noise in the data.
* **Ridge Regression**: Ridge adds a penalty that shrinks the coefficients uniformly, preventing overfitting by controlling large coefficients. It does not eliminate features but reduces the model's complexity.
* **Lasso Regression**: Lasso adds an L1 penalty that can force some coefficients to zero, leading to feature selection. It is useful for sparse models where only a few features are relevant.

In terms of **MSE**, Ridge typically performs slightly better when all features contribute to the output, whereas Lasso is useful when some features can be excluded.