Experiment 3

Aim

To implement and analyze Multiple Linear Regression, Lasso Regression, and Ridge Regression models on real-world datasets using Python and evaluate their performance using appropriate metrics.

---

1. Dataset Source

Dataset – Multiple, Lasso & Ridge Regression

Student Performance Dataset

Target Variable: Final_Score

Source:

https://github.com/jaykerkar0405/MLDL-Lab/blob/main/datasets/student_performance.csv

---

2. Dataset Description

Features:

- Hours_Studied
- Attendance
- Assignment_Score
- Midterm_Score

Target:

- Final_Score (Continuous)

Characteristics:

- Type: Numerical tabular dataset
- Problem Type: Regression
- No missing values
- Suitable for regularized regression techniques

---

3. Mathematical Formulation of the Algorithm

1. Multiple Linear Regression

Hypothesis Function:
$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta n x n$

Cost Function (MSE):
$MSE = (1 / N) \Sigma (yi - \hat{y}i)^2$

---

2. Ridge Regression (L2 Regularization)

Cost Function:
$J = (1/N) \Sigma (yi - \hat{y}i)^2 + \lambda \Sigma \beta j^2$

Where:
λ = regularization parameter
Penalizes large coefficients

---

3. Lasso Regression (L1 Regularization)

Cost Function:
$J = (1/N) \Sigma (yi - \hat{y}i)^2 + \lambda \Sigma |\beta j|$

Where:
λ = regularization parameter
Can shrink some coefficients to zero (Feature Selection)

4. Algorithm Limitations

Multiple Linear Regression

- Sensitive to multicollinearity
- Sensitive to outliers
- Assumes linear relationship

Ridge Regression

- Does not eliminate features completely
- Requires tuning of λ

Lasso Regression

- May eliminate important features if λ is large
- Not stable when features are highly correlated

5. Methodology / Workflow

Common Workflow

1. Load dataset using Pandas
2. Select independent and dependent variables
3. Split dataset into training and testing sets
4. Standardize features
5. Train models
6. Predict test values
7. Evaluate using MSE and R^2 score

6. Performance Analysis

Evaluation Metrics

- Mean Squared Error (MSE)
- R^2 Score

Interpretation:

- Lower MSE → Better model
- R^2 close to 1 → Good fit
- Regularization reduces overfitting
- Compare coefficients to observe shrinkage

---

7. Hyperparameter Tuning

Ridge Regression

- Alpha (λ value)

Lasso Regression

- Alpha (λ value)

Higher alpha → Stronger regularization
Lower alpha → Weaker regularization

Hyperparameter tuning improves generalization and prevents overfitting.

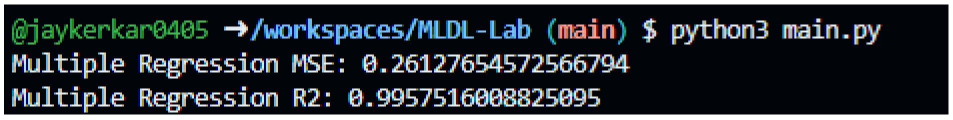
Exercise 1: Multiple Linear Regression

Code

```
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, r2_score
from sklearn.preprocessing import StandardScaler

# Load dataset
df = pd.read_csv('student_performance.csv')

X = df[['Hours_Studied', 'Attendance', 'Assignment_Score',
'Midterm_Score']]
y = df['Final_Score']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Model Training
model = LinearRegression()
model.fit(X_train, y_train)

# Prediction
y_pred = model.predict(X_test)

# Evaluation
print("Multiple Regression MSE:", mean_squared_error(y_test,
y_pred))
print("Multiple Regression R2:", r2_score(y_test, y_pred))
```

Output



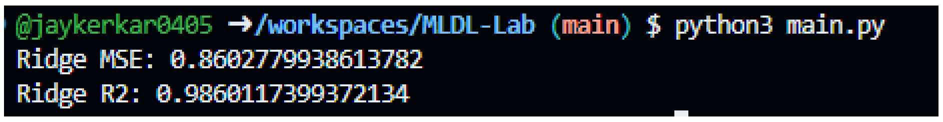
```
@jaykerkar0405 ➜/workspaces/MLDL-Lab (main) $ python3 main.py
Multiple Regression MSE: 0.26127654572566794
Multiple Regression R2: 0.9957516008825095
```

Exercise 2: Ridge Regression

Code

```
from sklearn.linear_model import Ridge

# Ridge Model
ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)

# Prediction
y_pred_ridge = ridge.predict(X_test)

# Evaluation
print("Ridge MSE:", mean_squared_error(y_test, y_pred_ridge))
print("Ridge R2:", r2_score(y_test, y_pred_ridge))
```

Output



```
@jaykerkar0405 ➜/workspaces/MLDL-Lab (main) $ python3 main.py
Ridge MSE: 0.8602779938613782
Ridge R2: 0.9860117399372134
```

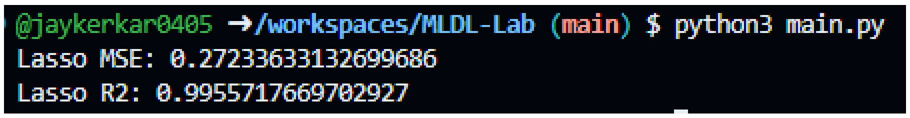
Exercise 3: Ridge Regression

Code

```
from sklearn.linear_model import Lasso

# Lasso Model
lasso = Lasso(alpha=0.1)
lasso.fit(X_train, y_train)

# Prediction
```

```
y_pred_lasso = lasso.predict(X_test)

# Evaluation
print("Lasso MSE:", mean_squared_error(y_test, y_pred_lasso))
print("Lasso R2:", r2_score(y_test, y_pred_lasso))
```

Output



```
@jaykerkar0405 ➜/workspaces/MLDL-Lab (main) $ python3 main.py
Lasso MSE: 0.27233633132699686
Lasso R2: 0.9955717669702927
```

Conclusion

This experiment successfully implemented Multiple Linear Regression, Ridge Regression, and Lasso Regression on a real-world dataset. While Multiple Linear Regression provides a baseline model, Ridge and Lasso Regression introduce regularization to reduce overfitting. Ridge shrinks coefficients without eliminating features, whereas Lasso can perform automatic feature selection by setting some coefficients to zero. These techniques are essential for improving model generalization and handling multicollinearity in supervised learning problems.