Disha Kulkarni roll no 38 D15B

Aim: Implement Multi Regression, Lasso, and Ridge Regression on real-world datasets

Theory:
Multiple Linear Regression, Ridge Regression, and Lasso Regression

Regression is a supervised machine learning technique used to model the relationship between a dependent variable and one or more independent variables. In this experiment, **Multiple Linear Regression**, **Ridge Regression**, and **Lasso Regression** are implemented on a real-world **house price prediction dataset** to analyze performance, overfitting, and feature importance.

Multiple Linear Regression

Multiple Linear Regression models the relationship between a target variable (y) and multiple input features (x_1, x_2, ..., x_n) by fitting a linear equation:

[
y = \beta_0 + \beta_1x_1 + \beta_2x_2 + \dots + \beta_nx_n + \epsilon
]

Where:

- (\beta_0) is the intercept
- (\beta_1, \beta_2, ..., \beta_n) are coefficients
- (\epsilon) is the error term

Characteristics:

- Assumes linear relationship between features and target
- Sensitive to multicollinearity
- Can overfit when number of features is large

In the house price dataset, Multiple Linear Regression acts as a **baseline model** to evaluate how regularization improves performance.

Ridge Regression (L2 Regularization)

Ridge Regression is an extension of linear regression that applies **L2 regularization** to reduce overfitting. It adds a penalty term proportional to the square of the magnitude of coefficients.

[
\text{Loss} = \sum (y_i - \hat{y}_i)^2 + \lambda \sum \beta_j^2
]

Where:

- (\lambda) (alpha) controls the strength of regularization

Key Features:

- Shrinks coefficients but does **not eliminate** any feature
- Handles multicollinearity effectively
- Improves generalization on unseen data

In datasets like house prices with many correlated features (e.g., area, rooms, floors), Ridge Regression stabilizes coefficient estimates and reduces variance.

Lasso Regression (L1 Regularization)

Lasso Regression introduces **L1 regularization**, which adds a penalty proportional to the absolute value of coefficients:

[
\text{Loss} = \sum (y_i - \hat{y}_i)^2 + \lambda \sum |\beta_j|
]

Key Features:

- Performs **automatic feature selection**
- Some coefficients become exactly zero
- Produces simpler and more interpretable models

For the house price dataset, Lasso helps identify the most influential factors affecting house prices by removing irrelevant or weak predictors.

Importance of Feature Scaling

Both Ridge and Lasso are sensitive to feature scale. Therefore, **standardization** is applied so that all features contribute equally to the regularization process.

Dataset Description

The **House Price Prediction dataset** contains real-world housing information such as:

- Property size, location, quality, and condition
- Construction year, number of rooms, and amenities
- Target variable: SalePrice

This dataset is ideal for regression experiments due to its **high dimensionality** and **feature correlation**.

Code:

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score
# Load dataset (update path if needed)
data = pd.read_csv("House Price Prediction Dataset.csv")

print(data.shape)
data.head()
# Drop columns with more than 40% missing values
threshold = 0.4
data = data.loc[:, data.isnull().mean() < threshold]

# Separate target
X = data.drop("Price", axis=1)
y = data["Price"]

# Fill missing values
X = X.fillna(X.median(numeric_only=True))
X = X.fillna("None")

# One-hot encoding
X = pd.get_dummies(X, drop_first=True)

print(X.shape)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
lr = LinearRegression()
lr.fit(X_train_scaled, y_train)

y_pred_lr = lr.predict(X_test_scaled)

print("Multiple Linear Regression")
print("MSE:", mean_squared_error(y_test, y_pred_lr))
print("R2 Score:", r2_score(y_test, y_pred_lr))
lr = LinearRegression()
```

```
lr.fit(X_train_scaled, y_train)

y_pred_lr = lr.predict(X_test_scaled)

print("Multiple Linear Regression")
print("MSE:", mean_squared_error(y_test, y_pred_lr))
print("R2 Score:", r2_score(y_test, y_pred_lr))
ridge = Ridge(alpha=1.0)
ridge.fit(X_train_scaled, y_train)

y_pred_ridge = ridge.predict(X_test_scaled)

print("\nRidge Regression")
print("MSE:", mean_squared_error(y_test, y_pred_ridge))
print("R2 Score:", r2_score(y_test, y_pred_ridge))
lasso = Lasso(alpha=0.001, max_iter=10000)
lasso.fit(X_train_scaled, y_train)

y_pred_lasso = lasso.predict(X_test_scaled)

print("\nLasso Regression")
print("MSE:", mean_squared_error(y_test, y_pred_lasso))
print("R2 Score:", r2_score(y_test, y_pred_lasso))
lasso_coefficients = pd.Series(
    lasso.coef_, index=X.columns
)

selected_features = lasso_coefficients[lasso_coefficients != 0]

print("Total features:", len(X.columns))
print("Selected features by Lasso:", len(selected_features))
from sklearn.linear_model import RidgeCV, LassoCV

ridge_cv = RidgeCV(alphas=[0.1, 1, 10, 50])
ridge_cv.fit(X_train_scaled, y_train)

lasso_cv = LassoCV(alphas=[0.0005, 0.001, 0.01], max_iter=10000)
lasso_cv.fit(X_train_scaled, y_train)

print("Best Ridge alpha:", ridge_cv.alpha_)
print("Best Lasso alpha:", lasso_cv.alpha_)
```

Screenshots:

(2000, 10)

	Id	Area	Bedrooms	Bathrooms	Floors	YearBuilt	Location	Condition	Garage	Price
0	1	1360	5	4	3	1970	Downtown	Excellent	No	149919
1	2	4272	5	4	3	1958	Downtown	Excellent	No	424998
2	3	3592	2	2	3	1938	Downtown	Good	No	266746
3	4	966	4	2	2	1902	Suburban	Fair	Yes	244020
4	5	4926	1	4	2	1975	Downtown	Fair	Yes	636056

```
(2000, 13)
```

```
Multiple Linear Regression
MSE: 78279764120.86243
R2 Score: -0.006181784611834162
```

```
Ridge Regression
MSE: 78279030968.43915
R2 Score: -0.006172360916937736
```

```
Lasso Regression
MSE: 78279763978.74977
R2 Score: -0.006181782785166012
```

```
Total features: 13
Selected features by Lasso: 13
```

```
Best Ridge alpha: 50.0
Best Lasso alpha: 0.01
```

CONCLUSION

In this experiment, Multiple Linear Regression, Ridge Regression, and Lasso Regression were successfully implemented on a real-world house price prediction dataset.

Multiple Linear Regression provided a baseline model but showed limitations due to multicollinearity and the presence of many correlated features. Ridge Regression improved model performance by reducing overfitting through coefficient shrinkage while retaining all

features. Lasso Regression further enhanced model interpretability by performing feature selection and eliminating irrelevant predictors.

The results demonstrate that **regularization techniques significantly improve regression models**, especially for high-dimensional real-world datasets. Ridge Regression is preferable when all features are potentially useful, while Lasso Regression is ideal when feature selection is required.

Thus, Ridge and Lasso regression techniques offer better generalization and robustness compared to standard Multiple Linear Regression in complex datasets like house price prediction.