Ridge and Lasso Regression using Scikit-Learn

Project Description

This project demonstrates the use of **Ridge and Lasso Regression**, which are two regularized linear regression techniques. These methods add penalties to the model to prevent overfitting and improve the generalization of the model.

- **Ridge regression** applies an L2 penalty (squared magnitude of coefficients).
- **Lasso regression** applies an L1 penalty (absolute value of coefficients), which can also result in sparse models by setting some coefficients to zero.

Why Ridge and Lasso Regression?

- **Ridge Regression** is helpful when you have many features and want to prevent multicollinearity and overfitting by shrinking the coefficients.
- **Lasso Regression** is useful when you have a large number of features, and you want to perform feature selection, as it tends to drive coefficients to zero, removing less important features.

Prerequisites

Required Libraries

- **Python 3.7 or later**
- `pandas` : For data manipulation and analysis.
- `numpy` : For numerical computations.
- `scikit-learn` : For machine learning models and evaluation metrics.
- `matplotlib` : For data visualization.

Installation

Run the following command to install the necessary libraries:

```
pip install pandas numpy scikit-learn matplotlib
```

Files Included

- **your_dataset.csv** : A placeholder dataset (replace with your actual dataset file).
- **Python code** for Ridge and Lasso regression.

Code Description

Steps in the Code

1. Dataset Loading:

```
 data = pd.read_csv('your_dataset.csv')
print(data.head())
```

The dataset is loaded, and the first few rows are printed for inspection.

2. Handling Missing Values:

```
data.fillna(data.mean(), inplace=True)
```

Missing values are filled with the mean of each column to ensure the model training is not disrupted.

3. Splitting Features and Target:

```
X = data.iloc[:, :-1]  # All columns except the last as features
y = data.iloc[:, -1]   # The last column as the target
```

4. Splitting into Training and Test Sets:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
```

5. Ridge Regression:

```
from sklearn.linear_model import Ridge

ridge_model = Ridge(alpha=1.0)
ridge_model.fit(X_train, y_train)
```

Ridge regression is applied with an **alpha** value of 1.0, which controls the strength of regularization.

6. Lasso Regression:

```
from sklearn.linear_model import Lasso

lasso_model = Lasso(alpha=0.1)
lasso_model.fit(X_train, y_train)
```

Lasso regression is applied with an **alpha** value of 0.1.

7. Model Predictions:

```
ridge_pred = ridge_model.predict(X_test)
lasso_pred = lasso_model.predict(X_test)
```

8. Evaluation Metrics:

Mean Squared Error (MSE):

```
from sklearn.metrics import mean_squared_error, r2_score

ridge_mse = mean_squared_error(y_test, ridge_pred)
lasso_mse = mean_squared_error(y_test, lasso_pred)
```

R^2 Score:

```
ridge_r2 = r2_score(y_test, ridge_pred)
lasso_r2 = r2_score(y_test, lasso_pred)

print(f"Ridge - MSE: {ridge_mse}, R² Score: {ridge_r2}")
print(f"Lasso - MSE: {lasso_mse}, R² Score: {lasso_r2}")
```

---

9. Visualization

Scatter Plot of Actual vs Predicted:

```
import matplotlib.pyplot as plt

plt.scatter(y_test, ridge_pred, color='blue', label='Ridge')
plt.scatter(y_test, lasso_pred, color='red', label='Lasso')
plt.title('Actual vs Predicted')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.legend()
plt.show()
```

Residual Plot:

```
ridge_residuals = y_test - ridge_pred
lasso_residuals = y_test - lasso_pred

plt.scatter(ridge_pred, ridge_residuals, color='blue', label='Ridge')
plt.scatter(lasso_pred, lasso_residuals, color='red', label='Lasso')
plt.title('Residuals vs Predicted')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.axhline(y=0, color='r', linestyle='--')
plt.legend()
plt.show()
```

Outputs

Metrics

- **Ridge - Mean Squared Error (MSE)**
- **Ridge - R^2 Score**
- **Lasso - Mean Squared Error (MSE)**
- **Lasso - R^2 Score**

Visualizations

1. **Actual vs Predicted**: Scatter plot comparing actual vs predicted values for both Ridge and Lasso models.
2. **Residuals vs Predicted**: Plot showing residuals for Ridge and Lasso, helping identify biases or patterns.

Example Output

```
 Ridge - Mean Squared Error: 1.23
Ridge - R² Score: 0.91
Lasso - Mean Squared Error: 1.45
Lasso - R² Score: 0.88
```

Use Cases

This project is beneficial for: ? **Predicting continuous variables** with regularization to prevent overfitting.
? **Feature selection** with Lasso to reduce model complexity.
? **Comparative analysis** of regularization methods.

Future Enhancements

- **Hyperparameter Tuning**: Experiment with different values of **alpha** for Ridge and Lasso regression to optimize performance.
- **Cross-Validation**: Implement **cross-validation** to validate model performance on different subsets of the data.
- **Feature Engineering**: Explore more sophisticated **feature engineering techniques** to improve model accuracy.
