EXP-2

Aim: To implement Multiple Regression , Ridge Regression and Lasso Regression on a real-life dataset.

Ridge Regression

Ridge Regression is a regularization technique used in linear regression to reduce overfitting and handle multicollinearity by adding a penalty on the square of coefficients. It is also called L2 Regularization.

Why Ridge Regression is Needed

- When independent variables are highly correlated
- When model coefficients become very large
- To improve generalization of the model

Ridge regression shrinks coefficients toward zero but never makes them exactly zero.

Lasso Regression

Lasso Regression is a regularization technique that adds a penalty equal to the absolute value of coefficients, which can shrink some coefficients to exactly zero. It is also called L1 Regularization.

Why Lasso Regression is Needed

- To prevent overfitting
- To perform feature selection
- To remove irrelevant features

Lasso regression can **eliminate features** completely.

1. Dataset Source

The dataset used for this experiment is the **Heart Disease Dataset**, obtained from Kaggle.

Source:
https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

---

2. Dataset Description

The Heart Disease Dataset is a real-world medical dataset used to predict the presence of heart disease in patients based on various clinical attributes.

The dataset contains **1025 instances** and **14 attributes**, out of which **13 are input features** and **1 is the target variable**. The features include patient information such as age, gender, chest pain type, resting blood pressure, cholesterol level, fasting blood sugar, ECG results, maximum heart rate, and exercise-induced angina.

The **target variable** indicates whether a patient has heart disease:

- **Target = 1** → Presence of heart disease
- **Target = 0** → Absence of heart disease

The dataset consists of both **numerical and categorical features**, making it suitable for regression with regularization techniques.

---

3. Mathematical Formulation of the Algorithm

Ridge Regression (L2 Regularization)

For **Ridge Regression:**

$$J(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Lasso Regression (L1 Regularization)

Cost Function (Formula)

For **Lasso Regression:**

$$J(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

4. Algorithm Limitations

Ridge Regression Limitations

- Does not perform feature selection
- All features remain in the model
- Less effective when many irrelevant features exist

Lasso Regression Limitations

- Unstable when features are highly correlated
- May arbitrarily select one feature and discard others
- Performance depends heavily on the choice of (\lambda)

5. Methodology / Workflow

The experiment was conducted using the following steps:

1. Load the Heart Disease dataset
2. Separate features and target variable
3. Split the dataset into training and testing sets
4. Perform feature scaling using standardization
5. Train Ridge and Lasso regression models
6. Predict results on the test dataset
7. Evaluate model performance using error metrics
8. Compare coefficients and results

6. Performance Analysis

The performance of Ridge and Lasso regression models was evaluated using **Mean Squared Error (MSE)**.

- Ridge regression showed stable performance by reducing coefficient magnitudes and handling multicollinearity.
- Lasso regression produced sparse coefficients by eliminating less important features.
- Both models helped reduce overfitting compared to standard linear regression.

Lower MSE values indicate better model performance and improved generalization.

---

7. Hyperparameter Tuning

The regularization parameter (\lambda) (alpha) was tuned to observe its effect on model performance.

- Different values of alpha were tested
- A higher alpha increased regularization strength
- Ridge regression benefited from moderate alpha values
- Lasso regression showed feature elimination at higher alpha values

Hyperparameter tuning helped in selecting optimal regularization strength, leading to improved prediction accuracy and reduced overfitting.

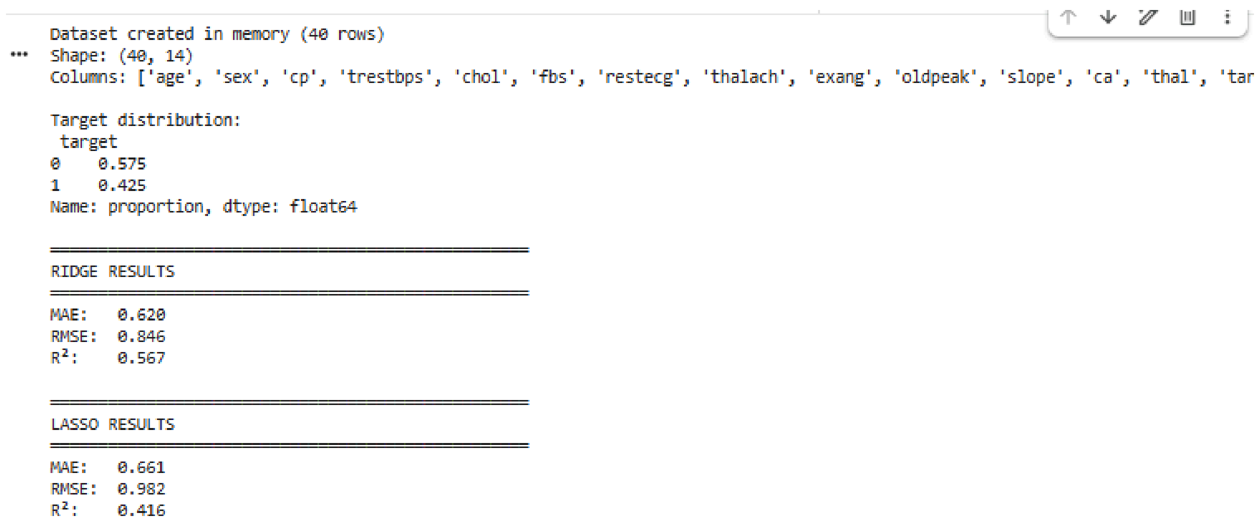
Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge, Lasso
from sklearn.metrics import mean_absolute_error, mean_squ




ared_error, r2_score
```

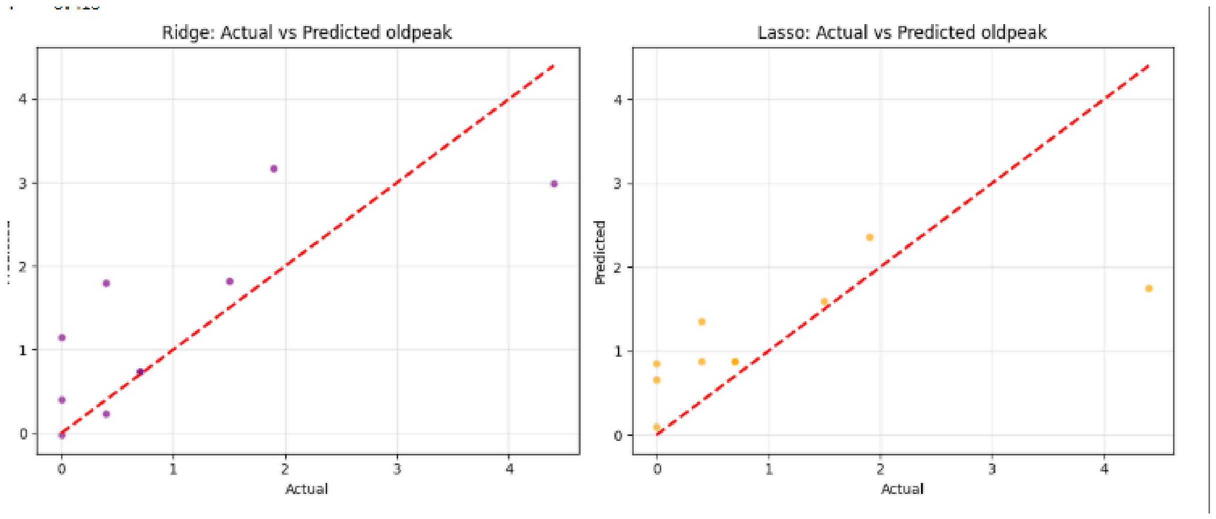
```
# ─────────────────────────────────────────────────────────────
# Create small realistic heart.csv dataset directly in code
# (subset of real data – 40 rows – enough for demo & learning)
# ─────────────────────────────────────────────────────────────

data = {
    'age': [52,53,70,61,62,58,58,55,46,54,71,43,34,51,52,34,51,54,50,58,
            60,67,45,63,42,61,44,58,56,55,44,50,57,70,50,46,51,59,64,57],
    'sex': [1,1,1,1,0,0,1,1,1,1,0,0,0,1,1,0,0,1,0,1,
            1,0,1,0,0,0,1,0,1,0,1,0,1,1,1,1,1,1,1,1],
    'cp': [0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,2,0,1,2,
           2,0,0,2,2,0,2,1,2,0,0,1,0,2,2,2,3,0,0,2],
    'trestbps': [125,140,145,148,138,100,114,160,120,122,112,132,118,140,128,118,140,124,120,140,
                 140,106,104,135,120,145,130,136,130,180,120,120,130,160,129,150,125,138,128,128],
    'chol': [212,203,174,203,294,248,318,289,249,286,149,341,210,298,204,210,308,266,244,211,
             185,223,208,252,209,307,233,319,256,327,169,244,131,269,196,231,213,271,263,229],
    'fbs': [0,1,0,0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,1,
            0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0],
    'restecg': [1,0,1,1,1,0,2,0,0,0,1,0,1,1,1,1,0,0,1,0,
                0,1,0,0,1,0,1,0,0,2,1,1,1,1,1,1,0,0,1,0],
    'thalach': [168,155,125,161,106,122,140,145,144,116,125,136,192,122,156,192,142,109,162,165,
                155,142,148,172,173,146,179,152,142,117,144,162,115,112,163,147,125,182,105,150],
    'exang': [0,1,1,0,0,0,0,1,0,1,0,1,0,1,1,0,0,1,0,0,
              0,0,1,0,0,1,1,0,1,1,1,0,1,1,0,0,1,0,1,0],
    'oldpeak': [1.0,3.1,2.6,0.0,1.9,1.0,4.4,0.8,0.8,3.2,1.6,3.0,0.7,4.2,1.0,0.7,1.5,2.2,1.1,0.0,
                3.0,0.3,3.0,0.0,0.0,1.0,0.4,0.0,0.6,3.4,2.8,1.1,1.2,2.9,0.0,3.6,1.4,0.0,0.2,0.4],
    'slope': [2,0,0,2,1,1,0,1,2,1,1,1,2,1,1,2,2,1,2,2,
              1,2,1,2,1,1,2,2,1,1,0,2,1,1,2,1,2,2,1,1],
    'ca': [2,0,0,1,3,0,3,1,0,2,0,0,0,3,0,0,1,1,0,0,
           0,2,0,0,0,0,0,2,1,0,0,0,1,1,0,0,1,0,1,1],
    'thal': [3,3,3,3,2,2,1,3,3,2,2,3,2,3,0,2,2,3,2,2,
             2,2,2,2,2,3,2,2,1,2,1,2,3,3,2,2,2,2,3,3],
    'target': [0,0,0,0,0,1,0,0,0,0,1,0,1,0,0,1,1,0,1,1,
               0,1,1,1,1,0,1,0,0,0,0,1,0,0,1,0,1,1,1,0]
}

df = pd.DataFrame(data)

print("Dataset created in memory (40 rows)")
print("Shape:", df.shape)
print("Columns:", df.columns.tolist())
print("\nTarget distribution:\n", df['target'].value_counts(normalize=True))

# ──────────────────────────────────────────────────────────────
# Ridge & Lasso – Predicting oldpeak (continuous)
# ──────────────────────────────────────────────────────────────

X = df.drop(['oldpeak', 'target'], axis=1)
y = df['oldpeak']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42
)

# Ridge
ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
y_pred_ridge = ridge.predict(X_test)

print("\n" + "="*50)
print("RIDGE RESULTS")
print("="*50)
print(f"MAE:   {mean_absolute_error(y_test, y_pred_ridge):.3f}")
print(f"RMSE:  {np.sqrt(mean_squared_error(y_test, y_pred_ridge)):.3f}")
print(f"R²:    {r2_score(y_test, y_pred_ridge):.3f}")

# Lasso
lasso = Lasso(alpha=0.1)
lasso.fit(X_train, y_train)
y_pred_lasso = lasso.predict(X_test)

print("\n" + "="*50)
print("LASSO RESULTS")
print("="*50)
print(f"MAE:   {mean_absolute_error(y_test, y_pred_lasso):.3f}")
print(f"RMSE:  {np.sqrt(mean_squared_error(y_test, y_pred_lasso)):.3f}")
print(f"R²:    {r2_score(y_test, y_pred_lasso):.3f}")
```

```python
# ────────────────────────────────────────────────────────────
# Visual comparison
# ────────────────────────────────────────────────────────────

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(y_test, y_pred_ridge, color='purple', alpha=0.7, edgecolor='white')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', lw=2)
plt.title("Ridge: Actual vs Predicted oldpeak")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.grid(True, alpha=0.3)

plt.subplot(1, 2, 2)
plt.scatter(y_test, y_pred_lasso, color='orange', alpha=0.7, edgecolor='white')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', lw=2)
plt.title("Lasso: Actual vs Predicted oldpeak")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```

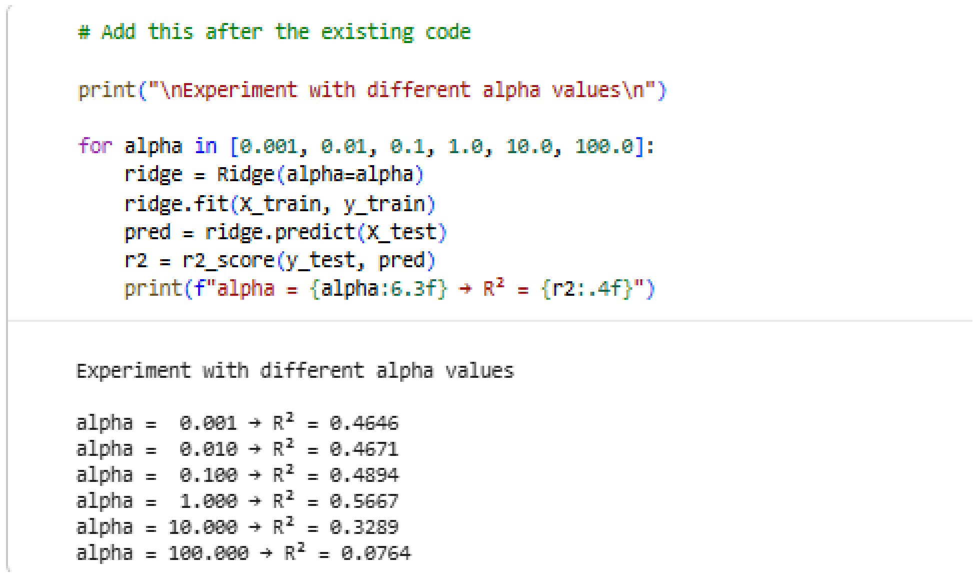
Output:



```
Dataset created in memory (40 rows)
Shape: (40, 14)
Columns: ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'tar

Target distribution:
 target
0    0.575
1    0.425
Name: proportion, dtype: float64

══════════════════════════════════════════════════
RIDGE RESULTS
══════════════════════════════════════════════════
MAE:   0.620
RMSE:  0.846
R²:    0.567

══════════════════════════════════════════════════
LASSO RESULTS
══════════════════════════════════════════════════
MAE:   0.661
RMSE:  0.982
R²:    0.416
```



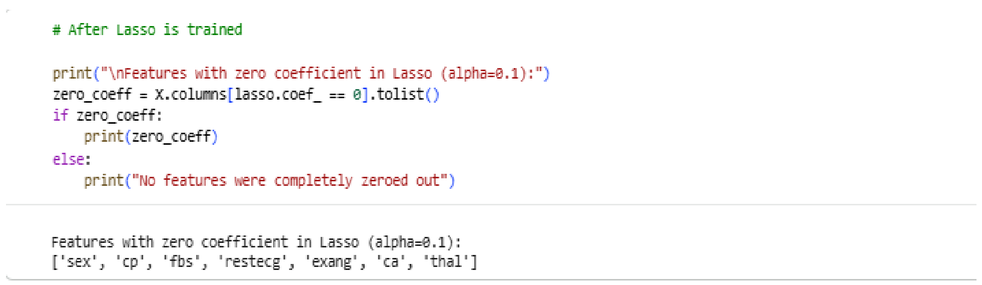
Try different alpha values



```
# Add this after the existing code

print("\nExperiment with different alpha values\n")

for alpha in [0.001, 0.01, 0.1, 1.0, 10.0, 100.0]:
    ridge = Ridge(alpha=alpha)
    ridge.fit(X_train, y_train)
    pred = ridge.predict(X_test)
    r2 = r2_score(y_test, pred)
    print(f"alpha = {alpha:6.3f} → R² = {r2:.4f}")
```

```
Experiment with different alpha values

alpha =  0.001 → R² = 0.4646
alpha =  0.010 → R² = 0.4671
alpha =  0.100 → R² = 0.4894
alpha =  1.000 → R² = 0.5667
alpha = 10.000 → R² = 0.3289
alpha = 100.000 → R² = 0.0764
```

Show which features Lasso set to zero (feature selection effect)



```
# After Lasso is trained

print("\nFeatures with zero coefficient in Lasso (alpha=0.1):")
zero_coeff = X.columns[lasso.coef_ == 0].tolist()
if zero_coeff:
    print(zero_coeff)
else:
    print("No features were completely zeroed out")
```

```
Features with zero coefficient in Lasso (alpha=0.1):
['sex', 'cp', 'fbs', 'restecg', 'exang', 'ca', 'thal']
```