Atharva Chaudhari
42

EXPERIMENT NO. 2

Aim:

Implementation of Multiple Linear Regression, Ridge Regression, and Lasso Regression on Insurance Dataset.

Dataset Source:

Dataset Name: Insurance Premium Prediction
Source: Kaggle

🔗 Source Link:
https://www.kaggle.com/datasets/noordeen/insurance-premium-prediction/data

This dataset is publicly available and widely used for regression-based cost prediction problems.

Dataset Description:

Overview:

The dataset contains medical insurance information of individuals and is used to predict medical insurance charges based on demographic and lifestyle factors.

Dataset Size:

- Total Records: 1338
- Total Features: 7
- Problem Type: Regression
- Target Variable: Continuous (charges)

📁 **Feature Description**

Feature	Description
age	Age of the person
sex	Gender (male/female)
bmi	Body Mass Index
children	Number of children/dependents
smoker	Smoking status (yes/no)
region	Residential region
charges	Medical i[illegible]ance cost (Target Variable)

Target Variable:

charges → Medical insurance cost (continuous numeric value)

Characteristics

- Mix of numerical and categorical data
- No missing values
- Moderate multicollinearity
- Linear relationship between BMI, smoker, and charges

Mathematical Formulation of the Algorithms:

1. Multiple Linear Regression

Linear Regression models the relationship between independent variables and a dependent variable.

Equation:

y=β0+β1x1+β2x2+...+βnxn+ϵ

Where:

- yyy = Target variable (charges)
- xix_ixi = Input features
- βi\beta_iβi = Coefficients
- ϵ\epsilonϵ = Error term

Objective Function:

$$MSE = \frac{1}{n}\sum(y_i - \hat{y}_i)^2$$

Minimize Mean Squared Error.

2. Ridge Regression (L2 Regularization)

Ridge adds penalty to reduce overfitting.

Cost Function:

$$J(\beta) = \frac{1}{n}\sum(y - \hat{y})^2 + \lambda\sum\beta^2$$

λ = Regularization parameter

Penalizes large coefficients

Reduces multicollinearity impact

3. Lasso Regression (L1 Regularization)

Lasso performs feature selection.

Cost Function:

$$J(\beta) = \frac{1}{n}\sum(y - \hat{y})^2 + \lambda\sum|\beta|$$

Can shrink some coefficients to zero

Performs automatic feature selection

Algorithm Limitations:

1. **Linear Regression:**

- Assumes linear relationship
- Sensitive to outliers
- Affected by multicollinearity
- Not suitable for non-linear datasets

2. **Ridge Regression:**

- Does not remove irrelevant features completely
- Selection of λ is critical

3. **Lasso Regression:**

- Unstable when features are highly correlated
- May eliminate useful correlated variables

4. **Not Suitable For:**

- Highly non-linear data
- Image or text datasets without preprocessing
- Extremely small datasets

Methodology / Workflow:

Step 1: Data Upload

Dataset uploaded in Google Colab.

Step 2: Data Preprocessing

- Automatic detection of target column
- Categorical encoding using get_dummies()
- Train-Test split (80-20)
- Feature scaling using StandardScaler

Step 3: Model Training

- Linear Regression
- Ridge Regression
- Lasso Regression

Step 4: Hyperparameter Tuning

Used GridSearchCV to find optimal alpha values for Ridge and Lasso.

Step 5: Model Evaluation

Used:

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

Performance Analysis:

Evaluation Metrics Used:

1. **Mean Squared Error (MSE)**

   Measures prediction error.

   Lower MSE → Better performance.

2. **R^2 Score:**

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

   Value between 0 and 1

   Closer to 1 → Better fit.

Observations from Output:

- Linear Regression provides strong baseline performance.
- Ridge slightly improves generalization by shrinking coefficients.
- Lasso performs feature selection but gives similar accuracy.
- R^2 score around 0.75–0.80 indicates good predictive performance.

- Slight deviations in higher charge values due to outliers (e.g., smokers).

Hyperparameter Tuning:

1. Ridge Regression

Parameter tuned:

α\alphaα

Tested values:
[0.01, 0.1, 1, 10, 100]

Used **GridSearchCV (5-fold cross-validation)**.

Best alpha selected automatically.

Impact:

- Small alpha → Less regularization
- Large alpha → Strong shrinkage
- Optimal alpha improves generalization

2. Lasso Regression:

Parameter tuned:

A\alphaα

Tested values:
[0.001, 0.01, 0.1, 1, 10]

Impact:

- Higher alpha → More coefficients shrink to zero
- Performs feature selection
- Prevents overfitting

CODE:

```python
# STEP 1: Upload File
from google.colab import files
uploaded = files.upload()

filename = list(uploaded.keys())[0]

# STEP 2: Import Libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler

# STEP 3: Load Dataset

data = pd.read_csv(filename)

print("\nColumn Names in Dataset:")
print(data.columns.tolist())

# STEP 4: Automatically Detect Target Column

# Assume last column is target
target_column = data.columns[-1]
print("\nDetected Target Column:", target_column)

y = data[target_column]
X = data.drop(target_column, axis=1)

# STEP 5: Convert Categorical to Numeric

X = pd.get_dummies(X, drop_first=True)

# STEP 6: Train-Test Split

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

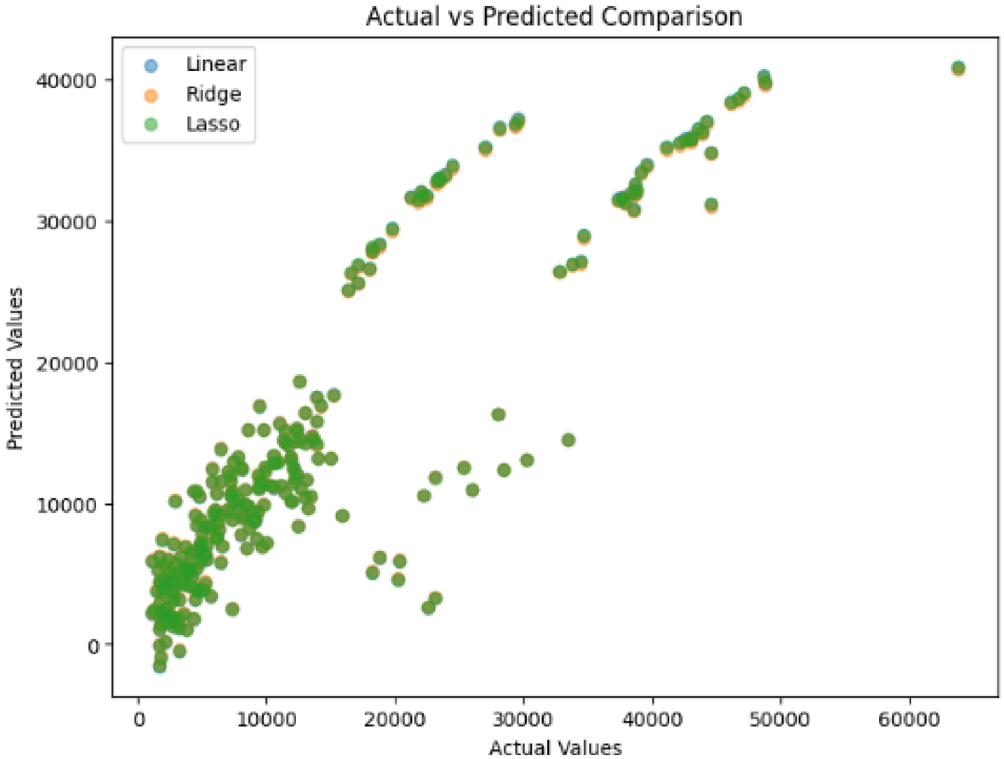
```python
# STEP 7: Feature Scaling

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# STEP 8: Linear Regression

lr = LinearRegression()
lr.fit(X_train, y_train)

y_pred_lr = lr.predict(X_test)

print("\n===== Linear Regression =====")
print("MSE:", mean_squared_error(y_test, y_pred_lr))
print("R2 Score:", r2_score(y_test, y_pred_lr))

# STEP 9: Ridge Regression

ridge = Ridge()
param_grid_ridge = {'alpha': [0.01, 0.1, 1, 10, 100]}

grid_ridge = GridSearchCV(ridge, param_grid_ridge, cv=5)
grid_ridge.fit(X_train, y_train)

best_ridge = grid_ridge.best_estimator_
y_pred_ridge = best_ridge.predict(X_test)

print("\n===== Ridge Regression =====")
print("Best Alpha:", grid_ridge.best_params_)
print("MSE:", mean_squared_error(y_test, y_pred_ridge))
print("R2 Score:", r2_score(y_test, y_pred_ridge))


# STEP 10: Lasso Regression

lasso = Lasso(max_iter=10000)
param_grid_lasso = {'alpha': [0.001, 0.01, 0.1, 1, 10]}

grid_lasso = GridSearchCV(lasso, param_grid_lasso, cv=5)
grid_lasso.fit(X_train, y_train)

best_lasso = grid_lasso.best_estimator_
y_pred_lasso = best_lasso.predict(X_test)
```

```
print("\n===== Lasso Regression =====")
print("Best Alpha:", grid_lasso.best_params_)
print("MSE:", mean_squared_error(y_test, y_pred_lasso))
print("R2 Score:", r2_score(y_test, y_pred_lasso))

# STEP 11: Visualization

plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred_lr, alpha=0.5, label="Linear")
plt.scatter(y_test, y_pred_ridge, alpha=0.5, label="Ridge")
plt.scatter(y_test, y_pred_lasso, alpha=0.5, label="Lasso")

plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs Predicted Comparison")
plt.legend()
plt.show()
```

OUTPUT:

```
Choose Files insurance.csv
insurance.csv(text/csv) - 50264 bytes, last modified: 2/15/2026 - 100% done
Saving insurance.csv to insurance (4).csv

Column Names in Dataset:
['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'expenses']

Detected Target Column: expenses

===== Linear Regression =====
MSE: 33600065.35507784
R2 Score: 0.7835726930039905

===== Ridge Regression =====
Best Alpha: {'alpha': 10}
MSE: 33688841.98244828
R2 Score: 0.7830008582119171

===== Lasso Regression =====
Best Alpha: {'alpha': 10}
MSE: 33642353.592636935
R2 Score: 0.783300302778679\8
```

...



Actual vs Predicted Comparison
Linear
Ridge
Lasso
Predicted Values
Actual Values
0
10000
20000
30000
40000
50000
60000