Experiment 2

Aim: Implementation of Multiple Linear Regression, Lasso Regression, and Ridge Regression on a Real-World Insurance Premium Dataset

Theory:

1. Dataset Source

The dataset used in this experiment is the **Insurance Premium Prediction** dataset available on Kaggle.

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

This dataset is publicly available and is commonly used for regression tasks related to healthcare cost prediction.

2. Dataset Description

The dataset contains information about individuals and their medical insurance expenses. The objective is to predict the **insurance expenses** based on demographic and lifestyle-related attributes.

Features (Independent Variables)

Feature	Type	Description
age	Numeric	Age of the person in years
sex	Categorical	Gender of the person (male/female)
bmi	Numeric	Body Mass Index
children	Numeric	Number of dependent children
smoker	Categorical	Smoking status (yes/no)
region	Categorical	Residential region in the US

expenses	Numeric	Medical insurance cost (target variable)

Target Variable

- **expenses** – Continuous numerical variable representing medical insurance charges.

Dataset Size

- Total Rows: ~1,300+
- Total Columns: 7

Characteristics

- Mix of numerical and categorical features.
- Real-world healthcare cost prediction problem.
- Suitable for linear and regularized regression models.

3. Mathematical Formulation of the Algorithms

(a) Multiple Linear Regression

Multiple Linear Regression models the relationship between a dependent variable and multiple independent variables:

y=β0+β1x1+β2x2+⋯+βnxn+εy=β0+β1x1+β2x2+⋯+βnxn+ε

Where:

- yy = Target variable (expenses)
- xixi = Feature variables
- βiβi = Model coefficients
- εε = Error term

The objective is to minimize the **Residual Sum of Squares (RSS)**:

RSS=∑i=1n(yi−y^i)2RSS=i=1∑n(yi−y^i)2

(b) Ridge Regression (L2 Regularization)

Ridge Regression adds an L2 penalty to control large coefficients:

Cost=∑i=1n(yi−y^i)2+λ∑j=1$p\beta j$2Cost=i=1∑n(yi−y^i)2+λj=1∑pβj2

Where λλ is the regularization parameter.

(c) Lasso Regression (L1 Regularization)

Lasso Regression adds an L1 penalty and can force coefficients to zero:

Cost=∑i=1n(yi−y^i)2+λ∑j=1$p|\beta j|$Cost=i=1∑n(yi−y^i)2+λj=1∑p|βj|

This enables **feature selection**.

4. Algorithm Limitations

Algorithm	Limitations
Multiple Linear Regression	Sensitive to multicollinearity, assumes linear relationship, affected by outliers
Ridge Regression	Does not perform feature selection, only shrinks coefficients
Lasso Regression	Can be unstable when features are highly correlated

General Limitations:

- Assumes linear relationship between variables.
- Not suitable for highly non-linear patterns without feature engineering.
- Sensitive to outliers.

5. Methodology / Workflow

Step-by-Step Process

1. **Dataset Loading**
    - Load dataset from Kaggle into Pandas.
2. **Data Preprocessing**
    - Handle categorical variables using One-Hot Encoding.
    - Separate features (X) and target (y).
    - Standardize numerical features.
3. **Train–Test Split**
    - Split data into 80% training and 20% testing.

4. **Model Training**
    - Train:
        - Multiple Linear Regression
        - Ridge Regression
        - Lasso Regression
5. **Model Evaluation**
    - Use metrics:
        - Mean Squared Error (MSE)
        - Root Mean Squared Error (RMSE)
        - R^2 Score
6. **Hyperparameter Tuning**
    - Tune regularization parameter αα for Ridge and Lasso using GridSearchCV.
7. **Performance Comparison**
    - Compare models and analyze results.

Workflow Diagram (Textual)

Data → Preprocessing → Train/Test Split →
→ Train Models → Hyperparameter Tuning → Evaluation → Comparison

6. Performance Analysis

The models are evaluated using:

- **Mean Squared Error (MSE)**
- **Root Mean Squared Error (RMSE)**
- **R^2 Score**

Interpretation

- Lower MSE/RMSE indicates better prediction accuracy.
- Higher R^2 (closer to 1) indicates better fit.

Typically:

- Multiple Linear Regression performs well but may overfit.
- Ridge improves stability.
- Lasso may reduce model complexity by eliminating weak features.

7. Hyperparameter Tuning

Ridge and Lasso Parameters

The regularization parameter αα is tuned using:

$\alpha \in \{0.01, 0.1, 1, 10, 100\}$α∈{0.01,0.1,1,10,100}

Using **GridSearchCV**, the best value of alpha is selected based on cross-validation performance.

Impact

- Optimal αα reduces overfitting.
- Ridge improves generalization.
- Lasso simplifies the model by removing irrelevant features.

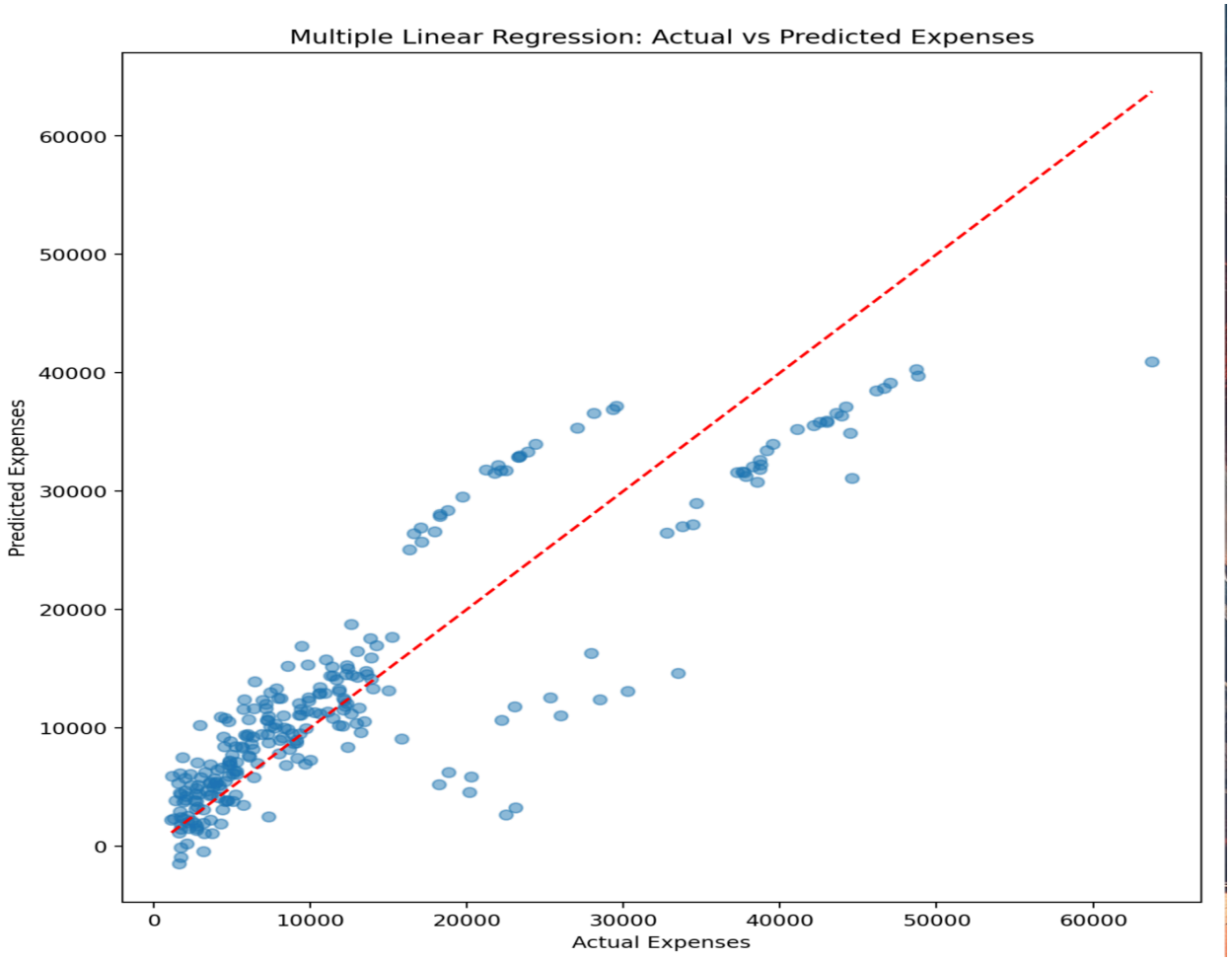
Code:

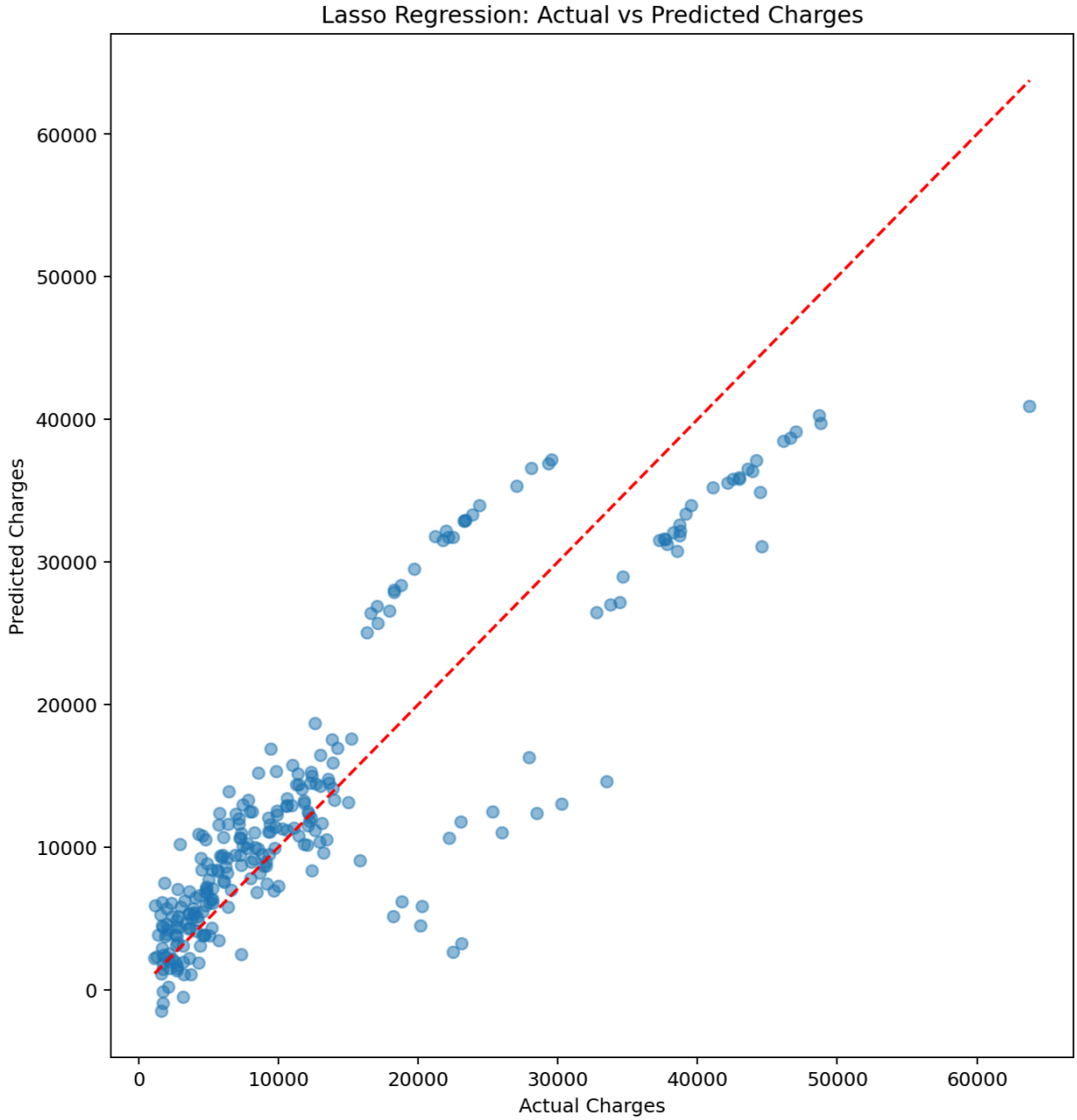
```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression, Lasso, Ridge

from sklearn.metrics import mean_squared_error, r2_score

from sklearn.preprocessing import LabelEncoder, StandardScaler

import matplotlib.pyplot as plt

import numpy as np



# Load dataset

file_path = 'archive (1)/insurance.csv'

df = pd.read_csv(file_path)



# Preprocessing: Encode categorical variables

le = LabelEncoder()
```

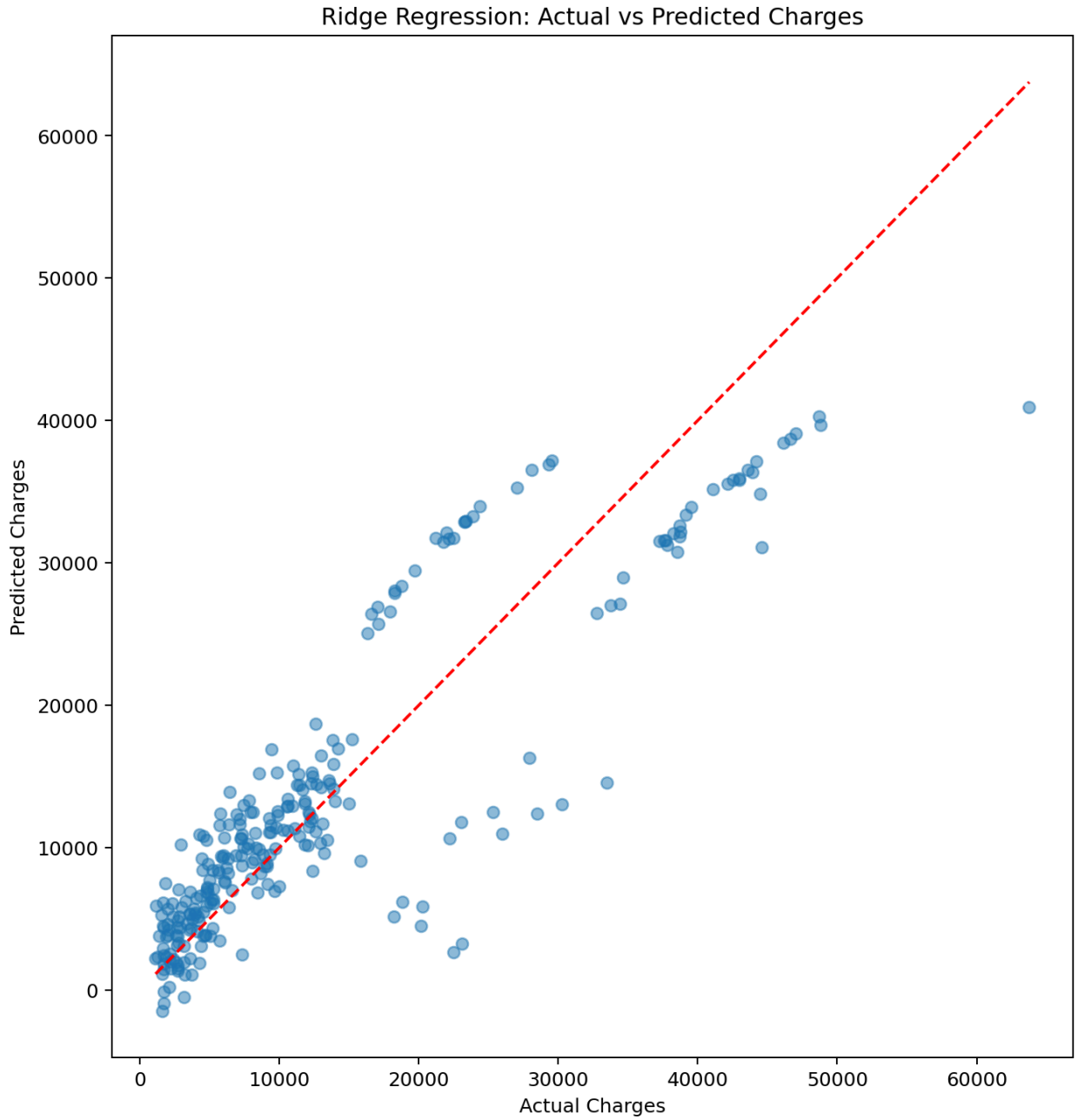
```
df_clean = df.copy()

for col in df_clean.select_dtypes(include=['object']).columns:

   df_clean[col] = le.fit_transform(df_clean[col])



# Features and target

y = df_clean['expenses']

X = df_clean.drop('expenses', axis=1)



# Split and scale

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)



# Models

models = {

   'Multiple Linear Regression': LinearRegression(),

   'Lasso Regression': Lasso(alpha=1.0),

   'Ridge Regression': Ridge(alpha=1.0)

}



results = {}

for name, model in models.items():
```

```
    model.fit(X_train_scaled, y_train)

    y_pred = model.predict(X_test_scaled)

    mse = mean_squared_error(y_test, y_pred)

    r2 = r2_score(y_test, y_pred)

    results[name] = (y_test, y_pred, mse, r2)

    print(f'{name} MSE:', mse)

    print(f'{name} R2 Score:', r2)

    plt.figure(figsize=(8,5))

    plt.scatter(y_test, y_pred, alpha=0.5)

    plt.xlabel('Actual Expenses')

    plt.ylabel('Predicted Expenses')

    plt.title(f'{name}: Actual vs Predicted Expenses')

    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')

    plt.tight_layout()

    plt.show()
```

8. Output:


Multiple Linear Regression: Actual vs Predicted Expenses
Predicted Expenses
Actual Expenses
0
10000
20000
30000
40000
50000
60000


Lasso Regression: Actual vs Predicted Charges
Predicted Charges
Actual Charges
0
10000
20000
30000
40000
50000
60000




```
[chirag@axos exp2]$ python exp2.py
Multiple Linear Regression MSE: 33639075.0899781
Multiple Linear Regression R2 Score: 0.7833214205203846
Lasso Regression MSE: 33641699.62400741
Lasso Regression R2 Score: 0.7833045151713598
Ridge Regression MSE: 33645665.32428989
Ridge Regression R2 Score: 0.7832789709998405
```

Conclusion

In this experiment, Multiple Linear Regression, Ridge Regression, and Lasso Regression were successfully implemented on a real-world insurance premium dataset to predict medical expenses. The results show that while Multiple Linear Regression provides a strong baseline, regularization techniques improve model stability and generalization. Ridge Regression effectively reduces overfitting by shrinking coefficient values, and Lasso Regression further simplifies the model by performing feature selection. Overall, regularized regression models offer better robustness and reliability for predicting insurance costs in real-world scenarios where multicollinearity and noise are present.