

# **Experiment - 2**

Nikhil Kale  
D15A/50

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

## **1. Dataset Source**

**Dataset Name:** Insurance Premium Prediction Dataset

**Source:** Kaggle

<https://www.kaggle.com/datasets/noordeen/insurance-premium-prediction/data>

## **2. Dataset Description**

The dataset contains medical insurance information and corresponding charges.

### **Dataset Size**

- 1338 records
- 7 features

### **Target Variable**

- expenses (continuous numeric value)

### **Features**

- age (numeric)
- sex (categorical)
- bmi (numeric)
- children (numeric)
- smoker (categorical)
- region (categorical)

### **Data Characteristics**

- Mixed categorical and numeric data
- Regression problem
- Moderate correlation between smoker and charges

## **3. Mathematical Formulation**

## Multiple Linear Regression

$$\hat{y} = \beta_0 + \sum_{i=1}^n \beta_i X_i$$

Minimizes:

$$RSS = \sum (y - \hat{y})^2$$

## Ridge Regression (L2 Regularization)

$$Loss = RSS + \lambda \sum \beta_i^2$$

Shrinks coefficients to prevent overfitting.

## Lasso Regression (L1 Regularization)

$$Loss = RSS + \lambda \sum |\beta_i|$$

Can shrink some coefficients exactly to zero (feature selection).

## 4. Algorithm Limitations

### Multiple Regression

- Sensitive to multicollinearity
- Assumes linearity
- Sensitive to outliers

### Ridge

- Does not eliminate features completely
- Requires tuning of lambda

### Lasso

- Can eliminate useful features if lambda is large
- Performance depends heavily on alpha value

## **5. Methodology / Workflow**

### **Step 1: Data Loading**

Loaded dataset using Pandas.

### **Step 2: Data Preprocessing**

- One-Hot Encoding for categorical variables
- Train-Test Split (80-20)
- Feature Scaling using StandardScaler

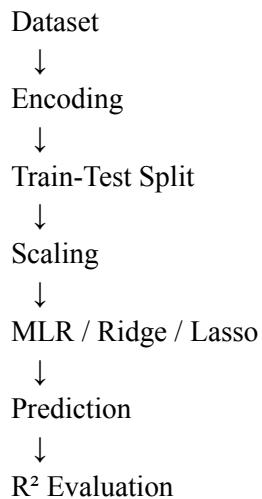
### **Step 3: Model Training**

- Multiple Linear Regression
- Ridge Regression
- Lasso Regression

### **Step 4: Evaluation**

- R<sup>2</sup> Score
- RMSE
- Coefficient comparison

## **Workflow Diagram**



## **6. Performance Analysis**

Model	R <sup>2</sup> Score
Multiple Linear Regression	0.78357
Ridge Regression	0.78352
Lasso Regression	0.78356

All three models achieved similar R<sup>2</sup> (~78%).

Interpretation:

- 78% of variation in insurance charges is explained.
- Regularization had minimal effect.
- Indicates low multicollinearity in the dataset.

## 7. Hyperparameter Tuning

Parameter tuned: `alpha`

Used GridSearchCV with:

$$\alpha \in 10^{-3} \text{ to } 10^3$$

Observation:

- Small alpha → behaves like Linear Regression
- Large alpha → over-regularization
- Optimal alpha improves generalization and reduces overfitting

## Code and Output -

```
[1] > # Basic libraries
> import numpy as np
> import pandas as pd
> import matplotlib.pyplot as plt
> import seaborn as sns

> # Sklearn
> from sklearn.model_selection import train_test_split, GridSearchCV
> from sklearn.preprocessing import StandardScaler, LabelEncoder
> from sklearn.linear_model import LinearRegression, Ridge, Lasso
> from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

> # Settings
> plt.style.use('seaborn-v0_8')
> sns.set_palette("Set2")
```

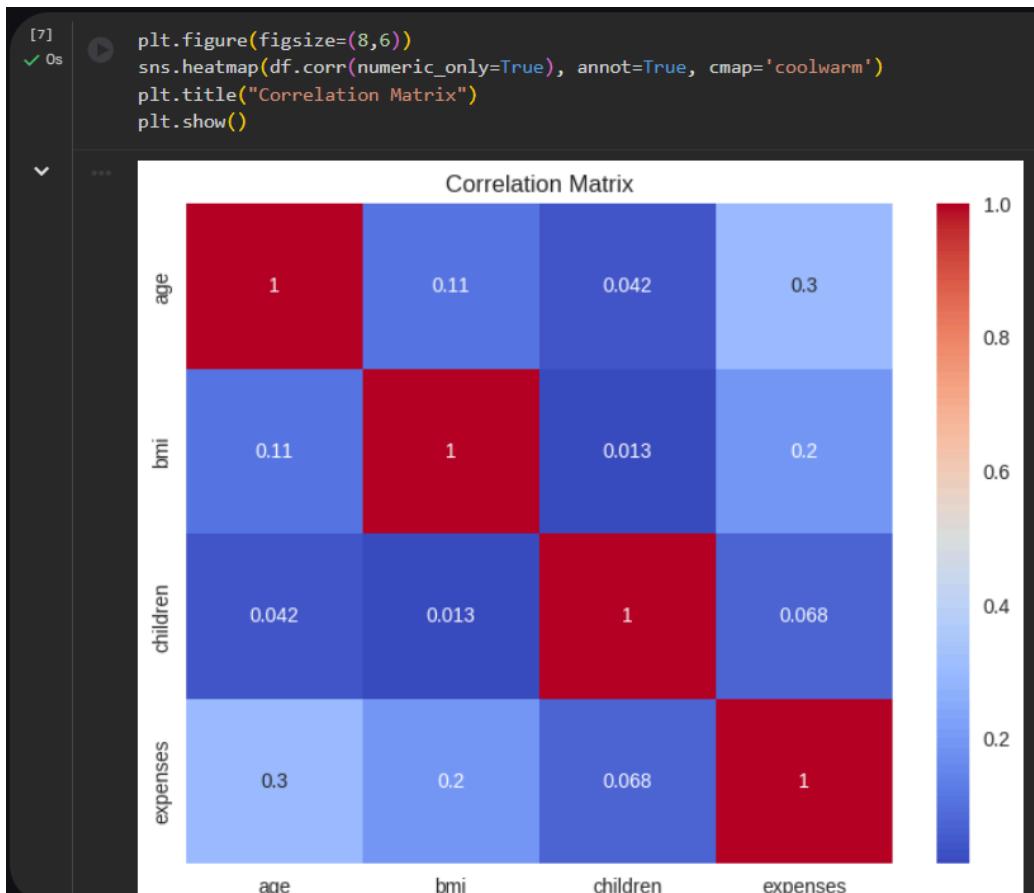
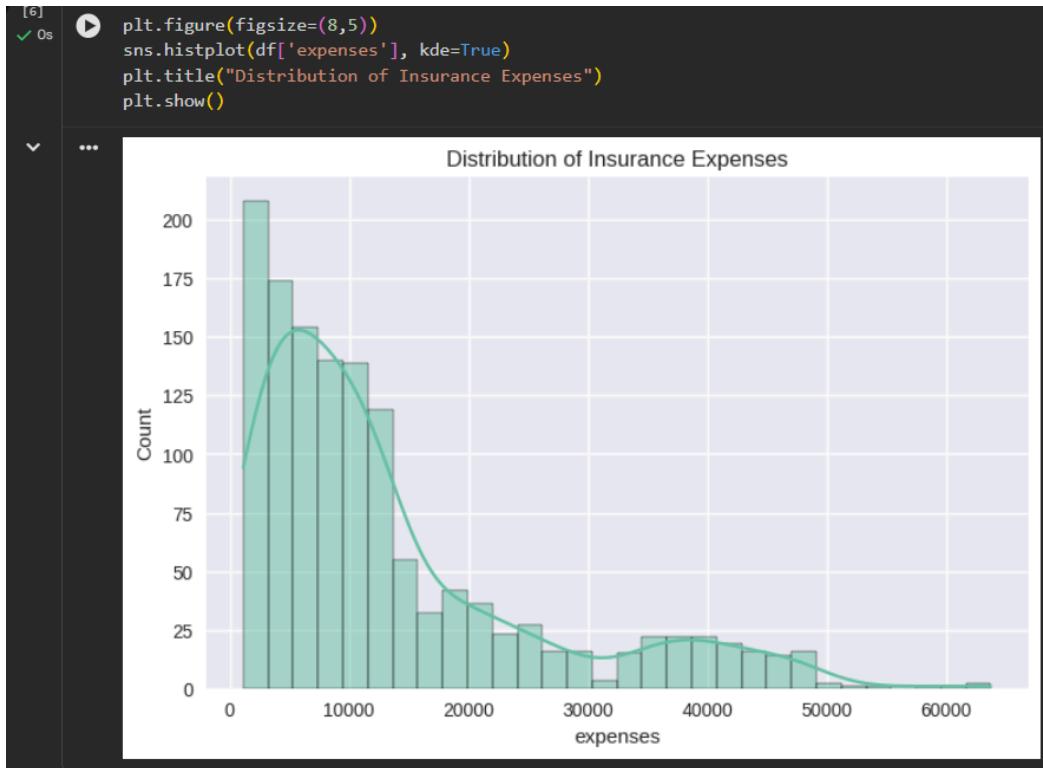
```
[2] ✓ 0s ⏎ df = pd.read_csv('/content/insurance.csv')
df.head()

...
   age    sex  bmi  children  smoker     region  expenses
0   19  female  27.9       0      yes  southwest  16884.92
1   18     male  33.8       1      no  southeast  1725.55
2   28     male  33.0       3      no  southeast  4449.46
3   33     male  22.7       0      no  northwest  21984.47
4   32     male  28.9       0      no  northwest  3866.86
```

```
[3] ✓ 0s ⏎ df.info()
df.describe()
df.isnull().sum()

...
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  --          -----          ----- 
 0   age         1338 non-null   int64  
 1   sex          1338 non-null   object 
 2   bmi          1338 non-null   float64 
 3   children     1338 non-null   int64  
 4   smoker        1338 non-null   object 
 5   region        1338 non-null   object 
 6   expenses      1338 non-null   float64 
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB

      0
age   0
sex   0
```



```
[8] ✓ 0s le = LabelEncoder()

df['sex'] = le.fit_transform(df['sex'])
df['smoker'] = le.fit_transform(df['smoker'])
df['region'] = le.fit_transform(df['region'])

[9] ✓ 0s X = df.drop('expenses', axis=1)
y = df['expenses']

[10] ✓ 0s ⏪ X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

[11] ✓ 0s ⏪ scaler = StandardScaler()

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

[12] ✓ 0s ⏪ lr = LinearRegression()
lr.fit(X_train, y_train)

y_pred_lr = lr.predict(X_test)

print("Linear Regression Performance:")
print("R2 Score:", r2_score(y_test, y_pred_lr))
print("MAE:", mean_absolute_error(y_test, y_pred_lr))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_lr)))

... Linear Regression Performance:
R2 Score: 0.7833214205203847
MAE: 4186.9401063170135
RMSE: 5799.920265829358

[13] ✓ 0s ⏪ ridge = Ridge()
ridge.fit(X_train, y_train)

y_pred_ridge = ridge.predict(X_test)

print("Ridge Regression Performance:")
print("R2 Score:", r2_score(y_test, y_pred_ridge))
print("MAE:", mean_absolute_error(y_test, y_pred_ridge))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_ridge)))

... Ridge Regression Performance:
R2 Score: 0.7832789709998405
MAE: 4188.40301104253
```

```
[14] 0s   lasso = Lasso()
        lasso.fit(X_train, y_train)

        y_pred_lasso = lasso.predict(X_test)

        print("Lasso Regression Performance:")
        print("R2 Score:", r2_score(y_test, y_pred_lasso))
        print("MAE:", mean_absolute_error(y_test, y_pred_lasso))
        print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_lasso)))

... Lasso Regression Performance:
R2 Score: 0.7833045151713598
MAE: 4187.054780221619
RMSE: 5800.146517460348
```

```
[15] 0s   ridge_params = {'alpha': np.logspace(-3, 3, 50)}

        ridge_grid = GridSearchCV(Ridge(), ridge_params, cv=5)
        ridge_grid.fit(X_train, y_train)

        print("Best Alpha for Ridge:", ridge_grid.best_params_)

        best_ridge = ridge_grid.best_estimator_
        y_pred_ridge_tuned = best_ridge.predict(X_test)

        print("Tuned Ridge R2:", r2_score(y_test, y_pred_ridge_tuned))

... Best Alpha for Ridge: {'alpha': np.float64(8.286427728546842)}
Tuned Ridge R2: 0.7829319370377956
```

```
[16]  lasso_params = {'alpha': np.logspace(-3, 3, 50)}

lasso_grid = GridSearchCV(Lasso(max_iter=10000), lasso_params, cv=5)
lasso_grid.fit(X_train, y_train)

print("Best Alpha for Lasso:", lasso_grid.best_params_)

best_lasso = lasso_grid.best_estimator_
y_pred_lasso_tuned = best_lasso.predict(X_test)

print("Tuned Lasso R2:", r2_score(y_test, y_pred_lasso_tuned))

...
... Best Alpha for Lasso: {'alpha': np.float64(104.81131341546852)}
Tuned Lasso R2: 0.7815906455716939
```

```
[17]  models = ['Linear', 'Ridge', 'Lasso']
r2_scores = [
    r2_score(y_test, y_pred_lr),
    r2_score(y_test, y_pred_ridge_tuned),
    r2_score(y_test, y_pred_lasso_tuned)
]

plt.figure(figsize=(6,4))
sns.barplot(x=models, y=r2_scores)
plt.title("Model Comparison (R2 Score)")
```

