

MLDL EXPERIMENT NO: 2

Aim:

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

Dataset Description

- The Insurance Charges Dataset is a real-world multivariate dataset commonly used in the machine learning domain for regression analysis, particularly Multiple Linear Regression, Ridge Regression, and Lasso Regression. Unlike simple or synthetic datasets, this dataset captures realistic variability in medical insurance costs influenced by demographic, lifestyle, and regional factors.
- The primary objective of this dataset is to predict medical insurance charges incurred by individuals based on attributes such as age, body mass index (BMI), smoking habits, and family size. Due to the presence of both numerical and categorical variables, as well as correlations among features (for example, BMI and smoking status), the dataset is well suited for demonstrating the effectiveness of regularization techniques like Ridge and Lasso regression.
- The dataset contains over 1,300 observations, making it large enough for robust model training and evaluation while remaining computationally efficient for academic experiments.

Column Name	Data Type	Description
Age	Numerical (Integer)	Age of the insured individual in years.
Sex	Categorical (Binary)	Gender of the individual (male or female).
BMI	Numerical (Float)	Body Mass Index, an indicator of body fat and health risk.
Children	Numerical (Integer)	Number of dependents covered by the insurance policy.
Smoker	Categorical (Binary)	Indicates whether the individual is a smoker (yes or no).
Region	Categorical	Residential region in the United States (northeast, northwest, southeast, southwest).
Charges	Numerical (Float)	Medical insurance charges billed to the individual (Target variable).

Dataset Link: <https://www.kaggle.com/datasets/mirichoi0218/insurance>

Purpose and Applications: The Insurance Charges Dataset is widely used to:

- Analyze the impact of lifestyle and demographic factors on healthcare costs
- Implement Multiple Linear Regression for baseline prediction
- Apply Ridge Regression to reduce coefficient variance caused by multicollinearity
- Apply Lasso Regression for feature selection by shrinking less important coefficients toward zero

MULTIPLE LINEAR REGRESSION

Theory:

- Multiple Linear Regression (MLR) is a statistical technique used to predict a continuous dependent variable based on two or more independent variables. It is an extension of Simple Linear Regression. While Simple Linear Regression models the relationship using a straight line, Multiple Linear Regression models the relationship using a multidimensional plane (in three dimensions) or a hyperplane (in higher dimensions).
- In the Insurance Charges Dataset, Multiple Linear Regression is used to predict medical insurance charges based on multiple influencing factors such as age, BMI, number of children, smoking status, gender, and region. Since insurance cost is affected by several variables simultaneously, Multiple Linear Regression is an appropriate modeling technique.

Mathematical Formulation

In Simple Linear Regression, the model is expressed as:

$$y = \beta_0 + \beta_1 x$$

In Multiple Linear Regression, the equation is extended to include multiple features:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \varepsilon$$

Where:

- **y**: Medical insurance charges (target variable)
- **x₁, x₂, ..., x_n**: Independent variables such as age, BMI, children, smoker, etc.
- **β₀**: Intercept (predicted charges when all features are zero)
- **β₁, ..., β_n**: Regression coefficients representing the effect of each feature on insurance charges while holding other variables constant
- **ε**: Error term representing unexplained variation

Each coefficient β_i indicates how much the insurance cost changes for a one-unit change in the corresponding feature, assuming all other variables remain constant.

Assumptions of Multiple Linear Regression: For the Multiple Linear Regression model to be valid, the following assumptions should be satisfied:

- **Linearity:** Insurance charges have a linear relationship with the independent variables.
- **No Multicollinearity:** Independent variables should not be highly correlated with each other.
- **Homoscedasticity:** The variance of the residuals remains constant across all predicted values.
- **Normality of Errors:** The residuals of the model are normally distributed.

Limitations:

1. Multicollinearity: Multiple Linear Regression assumes that all predictor variables are independent. Multicollinearity occurs when two or more features are strongly correlated, such as age and BMI or BMI and smoking-related health costs.

Condition of Failure: When multicollinearity is present, the model struggles to accurately estimate the individual contribution of each variable. This leads to unstable regression coefficients, making interpretation unreliable and sensitive to small data changes.

2. Overfitting (Curse of Dimensionality): As the number of independent variables increases, the complexity of the regression model also increases.

Condition of Failure: If the number of features becomes large relative to the number of observations, the model may start learning noise instead of the true pattern. This results in overfitting, where the model performs well on training data but poorly on unseen data.

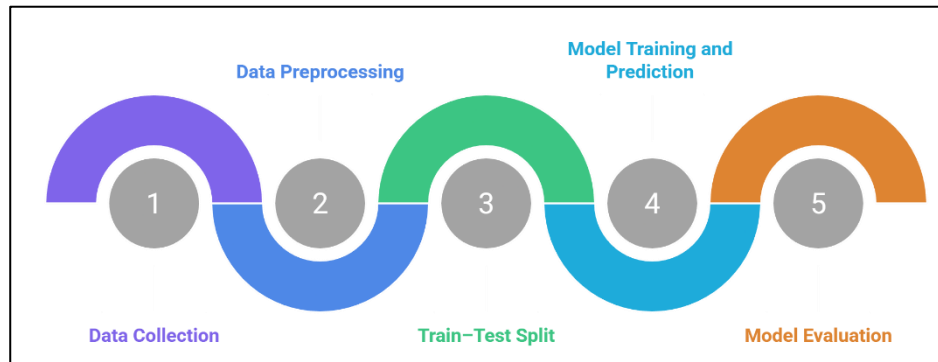
3. Assumption of Additive Feature Effects (Ignoring Interactions): Multiple Linear Regression assumes that each feature affects insurance charges independently and additively, as shown below:

$$y = \beta_1 x_1 + \beta_2 x_2$$

Condition of Failure: In real-world insurance data, features often interact. For example, the impact of BMI on insurance charges may be significantly higher for smokers than for non-smokers. Standard Multiple Linear Regression does not capture such interactions unless interaction terms are explicitly added to the model.

Workflow

- **Data Collection:** The Insurance Charges Dataset (insurance.csv) is loaded into a Pandas DataFrame. It contains demographic and lifestyle-related attributes of insured individuals, including age, gender, BMI, smoking status, region, and the corresponding medical insurance charges.
- **Data Preprocessing:** Categorical features such as sex, smoker, and region are converted into numerical form using one-hot encoding. The target variable charges is separated from the input features. Since the dataset contains no missing values, no imputation is required.
- **Train-Test Split:** The dataset is split into 80% training and 20% testing subsets. This ensures that the regression models are evaluated on unseen data, helping to measure generalization performance and reduce overfitting.
- **Model Training and Prediction:** A Multiple Linear Regression model is trained using the training data to learn the optimal regression coefficients that minimize the squared prediction error. The trained model is then used to predict insurance charges for the test dataset.
- **Model Evaluation:** Model performance is evaluated using Root Mean Squared Error (RMSE) to measure the average prediction error in monetary units and the R^2 score to determine the proportion of variance in insurance charges explained by the model. Actual and predicted values are compared to assess prediction quality.



Performance Analysis

The performance of the implemented Multiple Linear Regression model on the Insurance Charges Dataset is evaluated using the Root Mean Squared Error (RMSE) and the R^2 score.

1. Root Mean Squared Error (RMSE): The obtained RMSE value is 5,796.28. RMSE represents the standard deviation of the prediction errors, measuring the average difference between the predicted insurance charges and the actual charges.

Considering that insurance charges in the dataset range from a few thousand to over forty thousand units, an average prediction error of approximately 5,800 units is reasonable for a real-world healthcare cost prediction problem. This indicates that the model captures the general trend in insurance charges, although some variability remains due to complex factors such as medical history and lifestyle differences.

2. R^2 Score (Coefficient of Determination): The R^2 score achieved by the model is 0.78, which means that 78% of the variation in insurance charges is explained by the independent variables included in the model, such as age, BMI, number of children, smoking status, and region.

This value indicates a strong predictive performance, confirming that the selected features are significant contributors to insurance cost estimation. The remaining 22% of unexplained variance may be due to factors not present in the dataset or non-linear relationships that standard linear regression cannot capture.

Overall, the model demonstrates a good fit for the insurance data.

Hyperparameter Tuning

- Standard Multiple Linear Regression using the Ordinary Least Squares (OLS) method does not involve any tunable hyperparameters. The model computes a single optimal solution by minimizing the sum of squared errors using a closed-form mathematical approach.
- Unlike algorithms such as K-Nearest Neighbors or Random Forests, where parameters can be adjusted to influence learning behavior, OLS-based linear regression does not provide options for tuning. To improve model performance or handle issues such as multicollinearity and overfitting, regularized regression techniques like Ridge Regression or Lasso Regression must be applied.

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import
mean_squared_error, r2_score
from sklearn.linear_model import
LinearRegression
from sklearn.model_selection import
train_test_split

df = pd.read_csv('/content/insurance.csv')
df = pd.get_dummies(df, columns=['sex',
'smoker', 'region'], drop_first=True)

X = df.drop('charges', axis=1)
y = df['charges']
print("DataFrame after one-hot encoding and
splitting:")
print("X shape:", X.shape)
print("y shape:", y.shape)

X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Predictions on the test set made
successfully.")
```

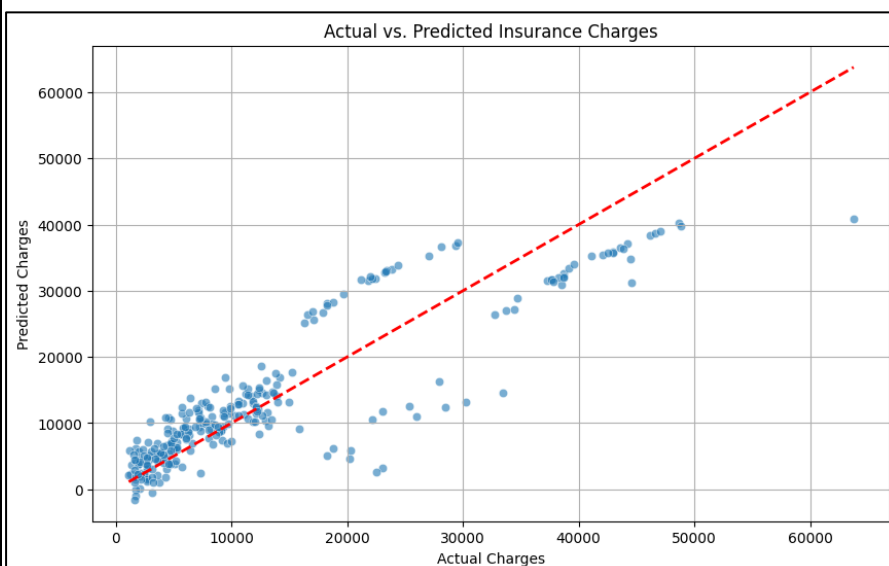
```
print("First 5 predictions:")
print(y_pred[:5])

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
print(f"Root Mean Squared Error (RMSE):
{rmse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")

plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.6)
plt.plot([min(y_test), max(y_test)],
[min(y_test), max(y_test)], color='red',
linestyle='--', lw=2)
plt.xlabel('Actual Charges')
plt.ylabel('Predicted Charges')
plt.title('Actual vs. Predicted Insurance
Charges')
plt.grid(True)
plt.show()

print("Plot of Actual vs. Predicted Insurance
Charges displayed.")
comparison_df = pd.DataFrame({'Actual':
y_test, 'Predicted': y_pred})
print("First 10 Actual vs. Predicted Values:")
print(comparison_df.head(10))
```

Output:



```
Root Mean Squared Error (RMSE): 5796.28
R-squared (R2) Score: 0.78
First 5 Actual vs. Predicted Values:
```

	Actual	Predicted
764	9095.06825	8969.550274
887	5272.17580	7068.747443
890	29330.98315	36858.410912
1293	9301.89355	9454.678501
259	33750.29180	26973.173457

LASSO REGRESSION

Theory:

- Lasso Regression is a type of regularized linear regression that improves upon standard Multiple Linear Regression by addressing overfitting and performing automatic feature selection. In traditional Multiple Linear Regression, the model minimizes prediction error without considering the magnitude of the coefficients. This can result in overly complex models that rely heavily on less important or noisy features.
- Lasso Regression overcomes this issue by adding a penalty term equal to the absolute value of the regression coefficients to the cost function. This penalty forces coefficients to shrink toward zero. A unique property of Lasso is its ability to shrink some coefficients exactly to zero, thereby completely removing irrelevant features from the model.

$$Cost = \underbrace{\sum_{i=1}^n (y_i - \hat{y}_i)^2}_{\text{Error (RSS)}} + \lambda \underbrace{\sum_{j=1}^p |\beta_j|}_{\text{L1 Penalty}}$$

- In the context of the Insurance Charges Dataset, Lasso Regression helps identify the most influential factors affecting insurance costs—such as smoking status, BMI, and age—while eliminating less impactful variables. The objective of Lasso Regression is to minimize the Residual Sum of Squares (RSS) along with an L1 regularization penalty.

Limitations:

1. Handling of Correlated Variables (Arbitrary Selection): Lasso Regression struggles when independent variables are **highly correlated** (multicollinearity).

Condition of Failure: In the insurance dataset, features such as BMI and smoking-related health impact may be correlated. Lasso tends to select one variable arbitrarily and shrink the other to zero, rather than distributing importance across both. This can reduce interpretability, as the retained feature may not necessarily be the most meaningful from a real-world perspective.

2. Number of Predictors vs Observations ($p > n$): Lasso has a mathematical limitation when the number of predictors exceeds the number of observations.

Condition of Failure: Lasso can select at most n features when there are n data points. While this is not a major issue for the insurance dataset, it becomes a limitation in high-dimensional scenarios such as genomic or text data.

Workflow:

1. **Data Collection:** The **Insurance Charges Dataset (insurance.csv)** is loaded into a Pandas DataFrame. The dataset contains demographic and lifestyle attributes along with corresponding medical insurance charges.
2. **Data Preprocessing:** Categorical variables such as **sex**, **smoker**, and **region** are converted into numerical form using **one-hot encoding**. The target variable **charges** is separated from the

independent variables. Since the dataset does not contain missing values, no imputation is required.

3. **Train–Test Split and Feature Scaling:** The dataset is split into **80% training** and **20% testing** sets. All numerical features are standardized using **StandardScaler**, which is essential for Lasso Regression because the regularization penalty is sensitive to feature scale.
4. **Model Training and Prediction:** Lasso Regression with cross-validation (**LassoCV**) is applied to automatically determine the optimal value of the regularization parameter **α (alpha)**. The trained model is then used to predict insurance charges on the test dataset.
5. **Model Evaluation and Feature Selection:** Model performance is evaluated using **RMSE** and **R^2 score**. The learned Lasso coefficients are analyzed to identify important predictors, with less relevant variables having their coefficients reduced to zero, effectively removing them from the model.
6. **Conclusion:** Lasso Regression provides a balance between predictive performance and model simplicity for the insurance dataset. By reducing overfitting and performing feature selection, it improves interpretability while retaining strong prediction accuracy, making it suitable for real-world insurance cost modeling.

Performance Analysis:

The performance of the implemented Lasso Regression model on the Insurance Charges Dataset:

1. **RMSE Analysis:** The model achieved an RMSE of 5,835.80, indicating that the predicted insurance charges deviate from the actual charges by approximately 5,836 monetary units on average. Considering the wide range of insurance charges in the dataset—from a few thousand to over forty thousand units—this level of error is reasonable for a real-world healthcare cost prediction problem. The RMSE value suggests that the model captures the general cost trends while smoothing out less significant variations.
2. **R^2 Score Analysis:** The obtained R^2 score is 0.78, which means that 78% of the variation in insurance charges is explained by the features selected by the Lasso model. This indicates a strong predictive capability, comparable to standard Multiple Linear Regression, while benefiting from reduced model complexity.

Hyperparameter Tuning:

Hyperparameter tuning was performed using **LassoCV** to optimize the Lasso Regression model and improve generalization. The key hyperparameter tuned was **alpha (α)**, which controls the strength of **L1 regularization**. **Smaller α :** Weaker regularization, more features retained **Larger α :** Stronger regularization, more coefficients shrunk toward zero

Using cross-validation, the optimal value obtained was: **Best Alpha (α): 100.0**. This indicates that the insurance dataset benefits from strong regularization, helping reduce the impact of less significant features. With this alpha value, the model achieved an **RMSE of 5,835.80** and an **R^2 score of 0.78**, demonstrating a good balance between accuracy and model simplicity.

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import
train_test_split
from sklearn.preprocessing import
StandardScaler
from sklearn.linear_model import LassoCV
from sklearn.metrics import
mean_squared_error, r2_score

df = pd.read_csv('/content/insurance.csv')

df = pd.get_dummies(df, columns=['sex',
'smoker', 'region'], drop_first=True)

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

print("X shape:", X.shape)
print("y shape:", y.shape)

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)

lasso = LassoCV(
    alphas=np.logspace(-3, 2, 50),
    cv=5,
    random_state=42
)

lasso.fit(X_train_scaled, y_train)
y_pred = lasso.predict(X_test_scaled)
```

```
print("Best Alpha:", lasso.alpha_)

rmse = np.sqrt(mean_squared_error(y_test,
y_pred))
r2 = r2_score(y_test, y_pred)

print(f"RMSE: {rmse:.2f}")
print(f"R2 Score: {r2:.2f}")

plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.6)
plt.plot(
    [y_test.min(), y_test.max()],
    [y_test.min(), y_test.max()],
    'r--', lw=2
)
plt.xlabel("Actual Charges")
plt.ylabel("Predicted Charges")
plt.title("Lasso Regression: Actual vs Predicted
Charges")
plt.grid(True)
plt.show()

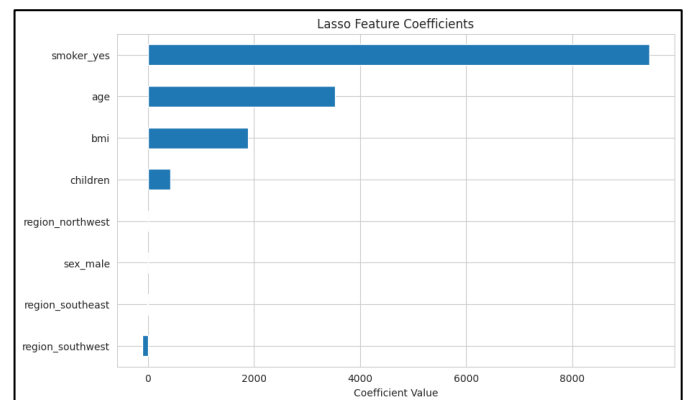
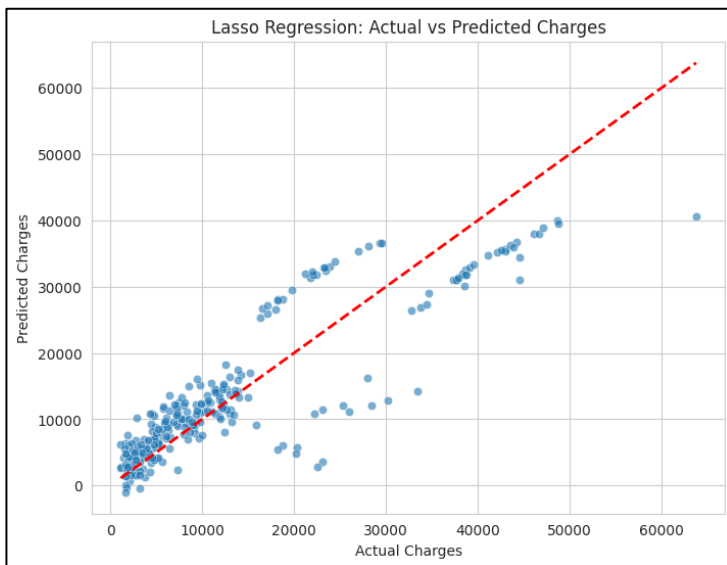
lasso_coef = pd.Series(lasso.coef_,
index=X.columns)
lasso_coef = lasso_coef.sort_values()

plt.figure(figsize=(10, 6))
lasso_coef.plot(kind='barh')
plt.title("Lasso Feature Coefficients")
plt.xlabel("Coefficient Value")
plt.show()

comparison_df = pd.DataFrame({
    'Actual': y_test.values,
    'Predicted': y_pred
})

print("\nFirst 5 Actual vs Predicted:")
print(comparison_df.head(5))
```

Output:



Best Alpha: 100.0
RMSE: 5835.80
R2 Score: 0.78

RIDGE REGRESSION

Theory:

- Ridge Regression (L2 Regularization) is a regression technique used to handle multicollinearity, where independent variables are highly correlated. In the presence of multicollinearity, standard Multiple Linear Regression produces unbiased coefficient estimates but with high variance, making the model unstable and sensitive to small changes in the data.
- Ridge Regression addresses this issue by adding an L2 penalty, which is the square of the magnitude of the regression coefficients, to the loss function. Unlike Lasso Regression, which penalizes the absolute value of coefficients and can eliminate features, Ridge Regression shrinks coefficients toward zero without setting them exactly to zero. As a result, all features remain in the model, but with reduced influence.
- Therefore, Ridge Regression is primarily a method of coefficient shrinkage rather than feature selection. Its objective is to minimize the Residual Sum of Squares (RSS) while applying a squared penalty to control model complexity and improve stability.

$$Cost = \underbrace{\sum_{i=1}^n (y_i - \hat{y}_i)^2}_{\text{Error (RSS)}} + \lambda \underbrace{\sum_{j=1}^p \beta_j^2}_{\text{L2 Penalty}}$$

Limitations:

1. Lack of Feature Selection (Model Interpretability): A key limitation of Ridge Regression is that it cannot eliminate features by shrinking coefficients exactly to zero.

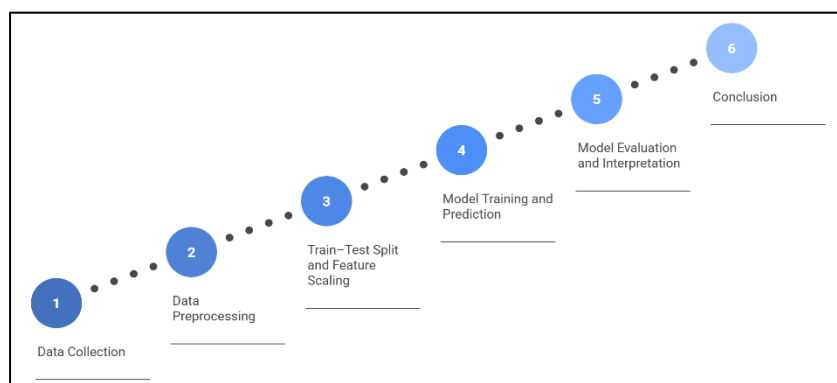
Condition of Failure: In the insurance dataset, if only a few variables (such as smoking status or BMI) are truly dominant, Ridge Regression will still retain all features (age, sex, region, children, etc.) with reduced weights. This results in a dense model, which can make interpretation and explanation to non-technical stakeholders more difficult.

2. Bias–Variance Tradeoff: Ridge Regression introduces bias into the model to reduce variance and stabilize coefficient estimates.

Condition of Failure: If the regularization strength (α) is set too high, the model may become overly constrained and underfit the data, failing to capture important relationships between predictors and insurance charges.

Workflow:

- 1. Data Collection:** The Insurance Charges Dataset (insurance.csv) is loaded into a Pandas DataFrame. It contains demographic and lifestyle attributes such as age, BMI, smoking status, region, and the corresponding medical insurance charges.
- 2. Data Preprocessing:** Categorical features (sex, smoker, and region) are converted into numerical form using one-hot encoding. The target variable charges is separated from the input features. The dataset contains no missing values, so no imputation is required.
- 3. Train–Test Split and Feature Scaling:** The dataset is split into 80% training and 20% testing subsets. All numerical features are standardized using StandardScaler, which is essential for Ridge Regression because the L2 penalty is sensitive to feature scale.
- 4. Model Training and Prediction:** Ridge Regression with cross-validation (RidgeCV) is trained on the training data to automatically select the optimal regularization parameter α . The trained model is then used to predict insurance charges on the test dataset.
- 5. Model Evaluation and Interpretation:** Model performance is evaluated using RMSE and R^2 score. The Ridge coefficients are analyzed to understand feature influence, noting that coefficients are reduced in magnitude but not eliminated.
- 6. Conclusion:** Ridge Regression effectively stabilizes the regression model by handling multicollinearity among insurance features while maintaining strong predictive performance.



Performance Analysis:

The performance of the Ridge Regression model is evaluated using **Root Mean Squared Error (RMSE)** and the **R² score**.

1. RMSE Analysis: The model achieved an RMSE of 5,802.53, indicating that predicted insurance charges differ from actual values by approximately 5,803 monetary units on average. Given the wide range of insurance charges in the dataset, this error level is acceptable for real-world cost prediction.

2. R² Score Analysis: The R² score of 0.78 indicates that 78% of the variation in insurance charges is explained by the independent variables. This demonstrates a strong linear relationship between the predictors and insurance costs and confirms effective learning from the dataset.

Hyperparameter Tuning:

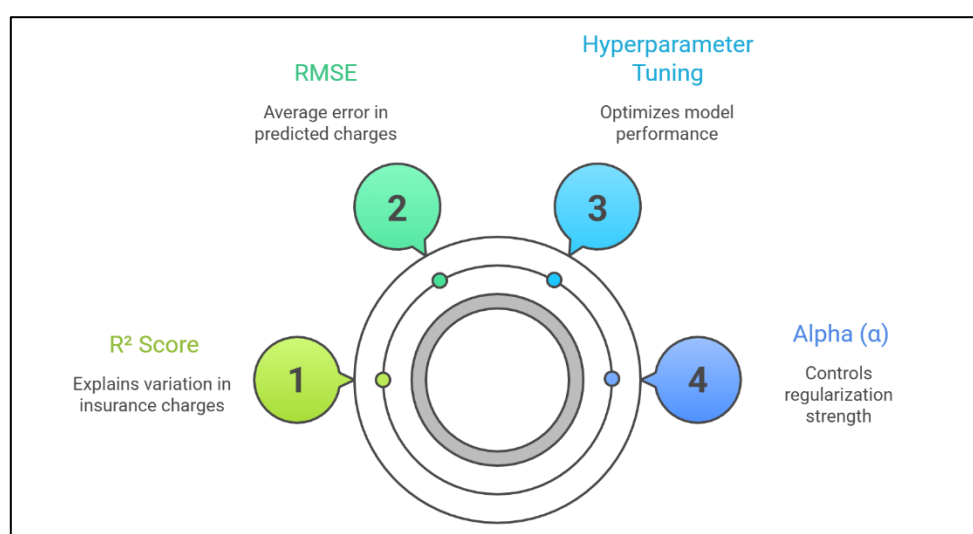
Hyperparameter tuning was performed using RidgeCV to optimize model performance and reduce overfitting. The tuned hyperparameter was alpha (α), which controls the strength of L2 regularization.

- **Smaller α values:** Weaker regularization, coefficients remain closer to original values
- **Larger α values:** Stronger regularization, coefficients are shrunk more aggressively to address multicollinearity

Using cross-validation, the optimal value obtained was:

Best Alpha (α): 8.2864

This value indicates that moderate regularization is sufficient to stabilize coefficient estimates in the insurance dataset while preserving all predictive features.



Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import
train_test_split
from sklearn.preprocessing import
StandardScaler
from sklearn.linear_model import RidgeCV
from sklearn.metrics import
mean_squared_error, r2_score

df = pd.read_csv('/content/insurance.csv')

df = pd.get_dummies(df, columns=['sex',
'smoker', 'region'], drop_first=True)

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

print("X shape:", X.shape)
print("y shape:", y.shape)

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)

ridge = RidgeCV(
    alphas=np.logspace(-3, 3, 50),
    cv=5
)

ridge.fit(X_train_scaled, y_train)
y_pred = ridge.predict(X_test_scaled)

print("Best Alpha (Ridge):", ridge.alpha_)
```

```
rmse = np.sqrt(mean_squared_error(y_test,
y_pred))
r2 = r2_score(y_test, y_pred)

print(f"RMSE: {rmse:.2f}")
print(f"R2 Score: {r2:.2f}")

plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.6)
plt.plot(
    [y_test.min(), y_test.max()],
    [y_test.min(), y_test.max()],
    'r--', lw=2
)
plt.xlabel("Actual Charges")
plt.ylabel("Predicted Charges")
plt.title("Ridge Regression: Actual vs Predicted
Charges")
plt.grid(True)
plt.show()

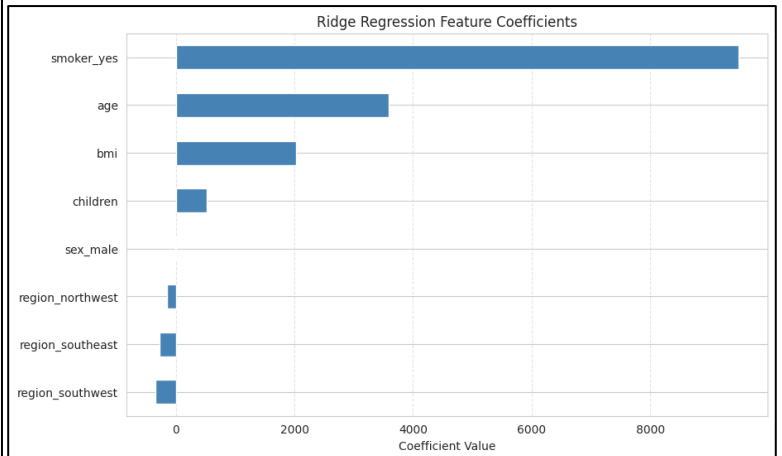
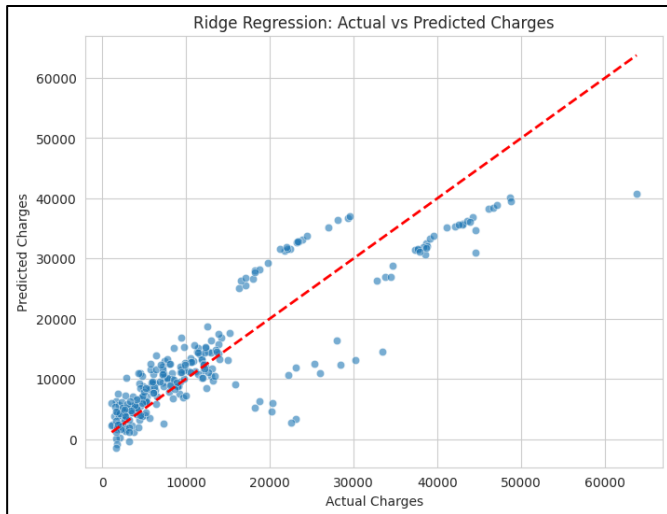
ridge_coef = pd.Series(ridge.coef_,
index=X.columns)
ridge_coef = ridge_coef.sort_values()

plt.figure(figsize=(10, 6))
ridge_coef.plot(kind='barh', color='steelblue')
plt.title("Ridge Regression Feature
Coefficients")
plt.xlabel("Coefficient Value")
plt.grid(True, axis='x', linestyle='--', alpha=0.5)
plt.show()

comparison_df = pd.DataFrame({
    'Actual': y_test.values,
    'Predicted': y_pred
})

print("\nFirst 10 Actual vs Predicted:")
print(comparison_df.head(10))
```

Output:



Best Alpha (Ridge): 8.286427728546842
RMSE: 5802.53
R2 Score: 0.78

Conclusion:

- In this experiment, Multiple Linear Regression, Ridge Regression, and Lasso Regression were studied theoretically and implemented practically on a real-world insurance dataset. The concept behind each regression technique was analyzed to understand their assumptions, strengths, and limitations.
- Using the insurance dataset, Multiple Linear Regression was applied as a baseline model to predict medical insurance charges.
- Ridge Regression was used to handle multicollinearity by shrinking coefficients.
- Lasso Regression further improved the analysis by performing regularization along with feature selection.
- The implementation demonstrated how theoretical concepts of regression and regularization translate effectively into real-world predictive modeling.