**D15A / 26**

## Experiment No :01

## A.Linear Regression

### 1. Dataset Source

The dataset used for Linear Regression is the Insurance Premium Prediction Dataset, obtained from Kaggle.

Source Link:
Insurance Premium Prediction

This dataset is used to predict continuous insurance premium values, making it suitable for Linear Regression.

### 2. Dataset Description

The dataset consists of insurance-related customer information.

- Number of records: 1338
- Number of features: 7

| Feature | Description | Type |
|---|---|---|
| age | Age of the person | Numerical |
| sex | Gender (male / female (male / female) | Categorical |
| bmi | Body Mass Index | Numerical |
| children | Number of children covered by insurance | Numerical |
| smoker | Smoking status (yes / no) | Categorical |
| region | Residential area of the person | Categorical |
| charges | Insurance premium amount (Target) | Numerical |

Independent Variables:

- age

- sex
- bmi
- children
- smoker
- region

Target Variable:

- charges – Insurance premium amount (continuous numerical value)

Dataset Characteristics:

- No missing values
- Combination of numerical and categorical data
- Suitable for regression-based prediction

### 3. Mathematical Formulation

Mathematical & Algorithmic Logic

Linear Regression models the relationship between dependent and independent variables using a linear equation.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

Where:

- $y$ = Insurance charges
- $x_1, x_2, \ldots, x_n$ = Input features
- $\beta_0$ = Intercept
- $\beta_1, \beta_2, \ldots, \beta_n$ = Coefficients
- $\epsilon$ = Error term

The objective is to minimize **Mean Squared Error (MSE)**:

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

### 4. Algorithm Limitations

Assumes a linear relationship between variables

- Sensitive to outliers
- Poor performance on non-linear data
- Multicollinearity can affect coefficient stability

**5. Methodology / Workflow**

1. Dataset Collection
   The Insurance Premium Prediction dataset was collected from Kaggle and used exclusively for this experiment.

2. Data Loading
   The dataset was loaded into the Python environment using the Pandas library.

3. Data Preprocessing

    a. Checked the dataset for missing values and confirmed that no null values were present.

    b. Categorical variables such as *sex*, *smoker*, and *region* were converted into numerical format using One-Hot Encoding.

4. Feature Selection

    a. Independent variables: age, sex, bmi, children, smoker, region

    b. Dependent variable: charges

5. Dataset Splitting
   The dataset was divided into:

    a. 80% training data

    b. 20% testing data
       using the train-test split method to ensure unbiased evaluation.

6. Model Training
   A Multiple Linear Regression model was trained using the `LinearRegression()` algorithm on the training dataset.

7. Model Evaluation
   The trained model was evaluated on the test dataset using:

    a. Mean Squared Error (MSE)

    b. $R^2$ Score

8. Visualization
   To analyze model performance and data relationships, the following visualizations were generated:

    a. Actual vs Predicted scatter plot to compare predicted insurance charges with actual values.

    b. Correlation heatmap to identify relationships between features and the target variable.

    c. Regression plot (BMI vs Charges) to observe the influence of BMI on insurance cost.

**6. Performance Analysis**

- MAE measures average absolute prediction error
- MSE penalizes large errors
- $R^2$ Score indicates goodness of fit

A high $R^2$ score confirms that the model effectively explains the variance in insurance charges.

Observed Results:

```
MAE: 4181.561524000791
MSE: 33600065.355077825
R2 Score: 0.7835726930039906
```

Analysis:

- The **MAE value** indicates that, on average, the predicted insurance charges differ from the actual values by around 4181 units.
- The **$R^2$ score of approximately 0.78** shows that about **78% of the variance** in insurance charges is explained by the model.
- This confirms that the Linear Regression model provides a **good fit** for predicting insurance premium values based on customer attributes.

## 7. Hyperparameter Tuning

Linear Regression has very limited hyperparameters.
Default parameters were used as the model already achieved strong performance.
No advanced hyperparameter tuning was required.

### Code and Output:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborne as sns

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression,
LogisticRegression

from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

df = pd.read_csv("insurance.csv")

df.head()
```

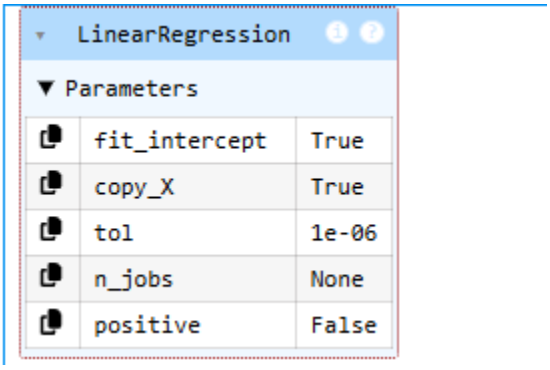
| | age | sex | bmi | children | smoker | region | expenses |
|---|---|---|---|---|---|---|---|
| 0 | 19 | female | 27.9 | 0 | yes | southwest | 16884.92 |
| 1 | 18 | male | 33.8 | 1 | no | southeast | 1725.55 |
| 2 | 28 | male | 33.0 | 3 | no | southeast | 4449.46 |
| 3 | 33 | male | 22.7 | 0 | no |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

```
if df['sex'].dtype == 'object':

    df['sex'] = df['sex'].map({'male': 1, 'female': 0})

if df['smoker'].dtype == 'object':
```

```
    df['smoker'] = df['smoker'].map({'yes': 1, 'no': 0})
if 'region' in df.columns:
    df = pd.get_dummies(df, columns=['region'], drop_first=True)
df.head()
```

| | age | sex | bmi | children | smoker | expenses | region_northwest | region_southeast | region_southwest |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 19 | 0 | 27.9 | 0 | 1 | 16884.92 | False | False | True |
| 1 | 18 | 1 | 33.8 | 1 | 0 | 1725.55 | False | True | False |
| 2 | 28 | 1 | 33.0 | 3 | 0 | 4449.46 | False | True | False |
| 3 | 33 | 1 | 22.7 | 0 | 0 | 21984.47 | True | False | False |
| 4 | 32 | 1 | 28.9 | 0 | 0 | 3866.86 | True | False | False |

```
X = df.drop('expenses', axis=1)
y = df['expenses']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
lr = LinearRegression()
lr.fit(X_train, y_train)
```

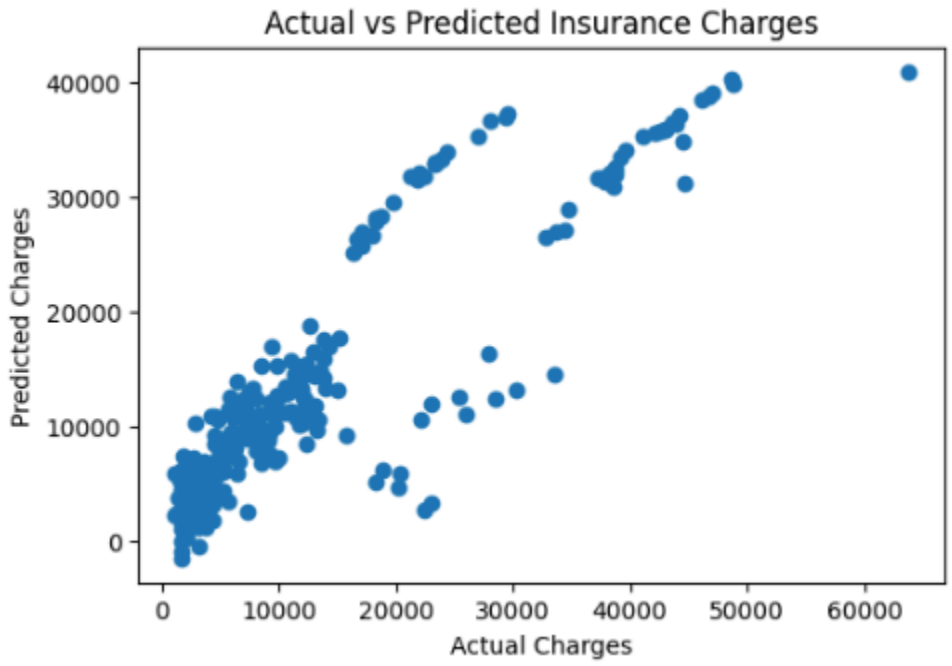



```
y_pred = lr.predict(X_test)
print("MAE:", mean_absolute_error(y_test, y_pred))
print("MSE:", mean_squared_error(y_test, y_pred))
```

```
print("R2 Score:", r2_score(y_test, y_pred))
```

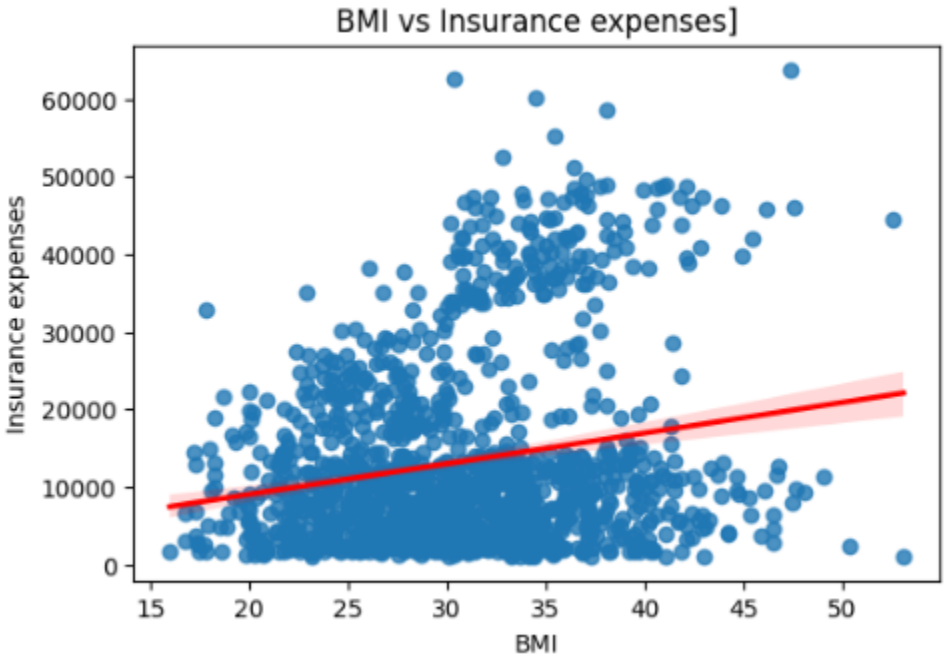
```
MAE: 4181.561524000791
MSE: 33600065.355077825
R2 Score: 0.7835726930039906
```

```
plt.figure(figsize=(6,4))

plt.scatter(y_test, y_pred)

plt.xlabel("Actual Charges")

plt.ylabel("Predicted Charges")

plt.title("Actual vs Predicted Insurance Charges")

plt.show()
```

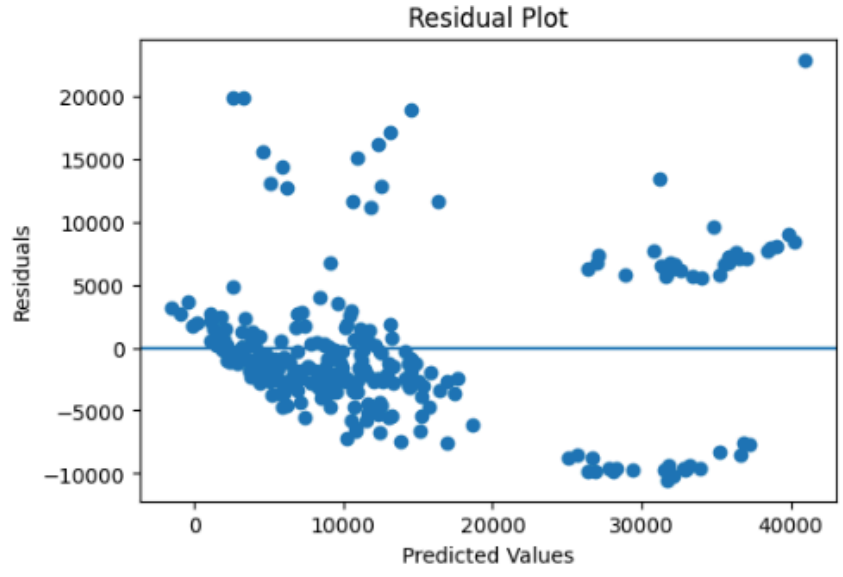



```
plt.figure(figsize=(6,4))

sns.regplot(x=df['bmi'], y=df['expenses'],
line_kws={"color":"red"})

plt.xlabel("BMI")

plt.ylabel("Insurance expenses")
```

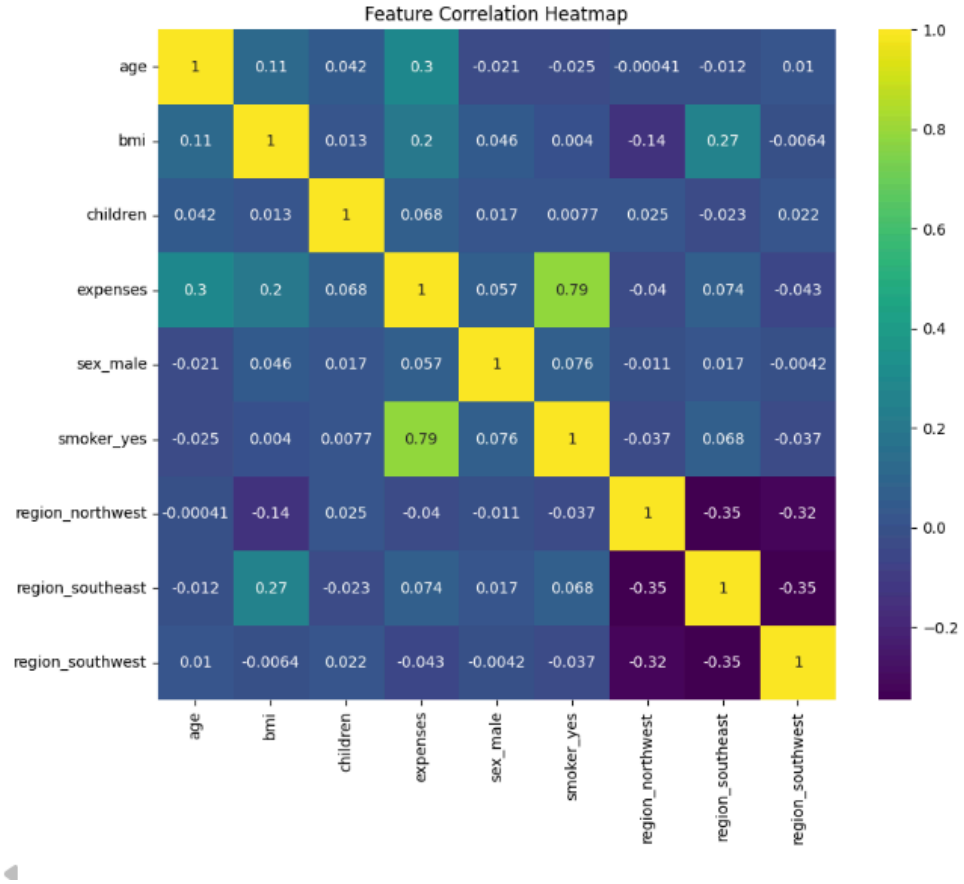
```
plt.title("BMI vs Insurance expenses]")

plt.show()
```




```
residuals = y_test - y_pred

plt.figure(figsize=(6,4))
plt.scatter(y_pred, residuals)
plt.axhline(y=0)
plt.xlabel("Predicted Values")
plt.ylabel("Residuals")
plt.title("Residual Plot")
plt.show()
```



```
plt.figure(figsize=(10,8))
sns.heatmap(data_enc.corr(), annot=True, cmap="viridis")
plt.title("Feature Correlation Heatmap")
plt.show()
```




## B.Logistic Regression

### 1. Dataset Source

The dataset used for Logistic Regression is the **Insurance Premium Prediction Dataset**, obtained from Kaggle.

Source Link:
Insurance Premium Prediction

The dataset was adapted for classification by converting the continuous insurance charges into binary premium categories.

## 2. Dataset Description

The dataset contains information about insurance customers and factors that influence insurance premiums.

- Total records: 1338
- Total attributes: 6 input features + 1 derived target variable

Input Features:

- age – Age of the customer
- sex – Gender (male/female)
- bmi – Body Mass Index
- children – Number of dependents
- smoker – Smoking status (yes/no)
- region – Residential area

Target Variable:

- premium_level
    - 1 → High premium (charges above dataset average)
    - 0 → Low premium (charges below or equal to average)

Dataset Characteristics:

- No missing values
- Structured tabular data
- Suitable for supervised binary classification

## 3. Mathematical Formulation

Logistic Regression predicts the probability of a binary outcome using the sigmoid function:

$$P(y = 1|x) = \frac{1}{1 + e^{-z}}$$

where,

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

The output probability always lies between 0 and 1.

## 4. Algorithm Limitations

Logistic Regression has the following limitations:

1. Assumes a linear relationship between features and the log-odds of the target variable.
2. Cannot capture complex non-linear decision boundaries.
3. Performance may degrade if classes are highly imbalanced.
4. Sensitive to irrelevant or highly correlated features.

The algorithm may not perform well when:

- The data has strong non-linear relationships
- Feature engineering is insufficient

## 5. Methodology / Workflow

**Step-by-Step Workflow:**

1. Dataset collection from Kaggle.
2. Data loading using Pandas.

3. Checking the dataset for missing values.
4. Encoding categorical variables into numerical form.
5. Converting the continuous target variable into binary classes.
6. Splitting the dataset into:
    - **80% Training data**
    - **20% Testing data**
7. Training the Logistic Regression model.
8. Evaluating the model using classification metrics.
9. Visualizing results using a confusion matrix.

### 6. Performance Analysis

The performance of the Logistic Regression model was evaluated using accuracy score, confusion matrix, and classification metrics such as precision, recall, and F1-score.

**Results Obtained:**

- **Accuracy: 0.9067** (≈ 90.67%)
- **Confusion Matrix:**

| Actual \ Predicted | 0 | 1 |
|---|---|---|
| 0 (Low Premium) | 189 | 0 |
| 1 (High Premium) | 25 | 54 |

- Classification Report:
    - Class 0 (Low Premium):
        - Precision: 0.88
        - Recall: 1.00

        - F1-Score: 0.94
    - Class 1 (High Premium):
        - Precision: 1.00
        - Recall: 0.68
        - F1-Score: 0.81
    - Weighted Average F1-Score: 0.90

Interpretation:

- The overall accuracy of 90.67% indicates that the model correctly classifies the majority of insurance customers.
- The model achieves perfect recall (1.00) for low-premium customers, meaning all low-premium cases were correctly identified.
- Precision for high-premium customers is 1.00, indicating no false positive predictions for this class.
- Slightly lower recall for the high-premium class suggests some high-premium cases were misclassified as low-premium.
- The balanced weighted F1-score of 0.90 confirms strong and reliable classification performance.

**7. Hyperparameter Tuning**

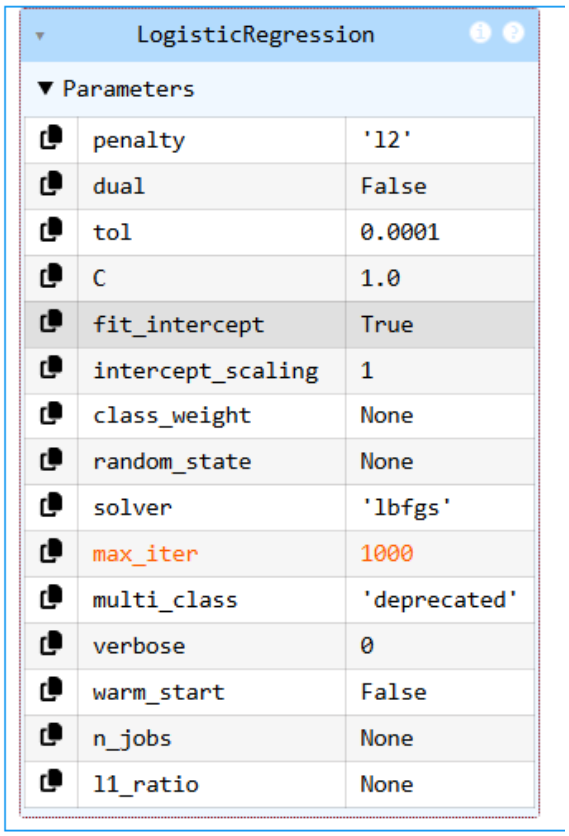
To ensure proper convergence of the Logistic Regression model, the following hyperparameter was adjusted:

- **max_iter = 1000**

Increasing the iteration limit ensured stable optimization and prevented premature convergence. No extensive tuning was required, as the model already achieved good performance with default regularization settings.

## Code and Output:

```
df['premium_level'] = (df['expenses'] > df['expenses'].mean()).astype(int)
X_log = df.drop(['expenses', 'premium_level'], axis=1)
y_log = df['premium_level']

X_train_l, X_test_l, y_train_l, y_test_l = train_test_split(
    X_log, y_log, test_size=0.2, random_state=42
)

log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train_l, y_train_l)
```



LogisticRegression

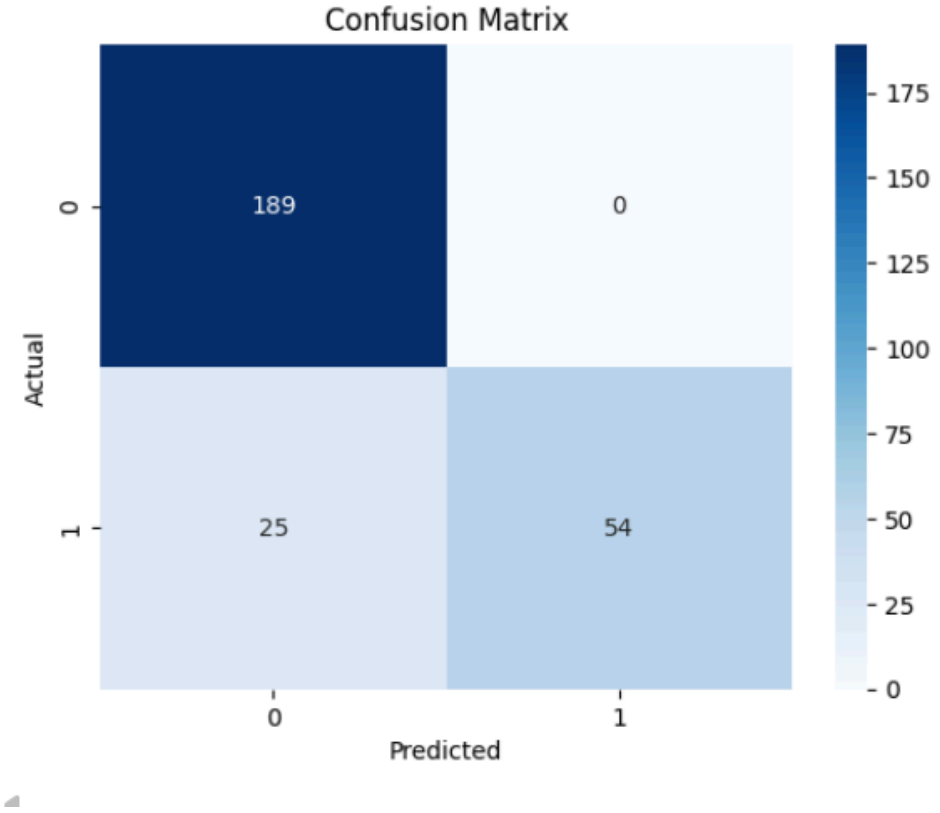
▼ Parameters

| | |
|---|---|
| penalty | 'l2' |
| dual | False |
| tol | 0.0001 |
| C | 1.0 |
| fit_intercept | True |
| intercept_scaling | 1 |
| class_weight | None |
| random_state | None |
| solver | 'lbfgs' |
| max_iter | 1000 |
| multi_class | 'deprecated' |
| verbose | 0 |
| warm_start | False |
| n_jobs | None |
| l1_ratio | None |

```
y_pred_l = log_model.predict(X_test_l)

print("Accuracy:", accuracy_score(y_test_l, y_pred_l))
print(confusion_matrix(y_test_l, y_pred_l))
print(classification_report(y_test_l, y_pred_l))
```

```
Accuracy: 0.9067164179104478
[[189   0]
 [ 25  54]]
              precision    recall  f1-score   support

           0       0.88      1.00      0.94       189
           1       1.00      0.68      0.81        79

    accuracy                           0.91       268
   macro avg       0.94      0.84      0.87       268
weighted avg       0.92      0.91      0.90       268
```

```
sns.heatmap(confusion_matrix(y_test_l, y_pred_l),
            annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```




**Conclusion:**

In this experiment, Linear and Logistic Regression were successfully implemented on a real-world insurance dataset. Linear Regression effectively predicted insurance charges with a good $R^2$ score, while Logistic Regression classified customers into high and low premium categories with an accuracy of **90.67%**. The results demonstrate that regression-based supervised learning algorithms are effective for both prediction and classification tasks on structured data.