Experiment No.01

Aim: Implement Linear and Logistic Regression on real-world datasets.

Theory:

PART A: LINEAR REGRESSION

1. Dataset Source

Dataset:Insurance Dataset
Source: Kaggle

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

2. Dataset Description

The dataset contains medical insurance information used to predict healthcare charges.

Features:

Feature	Description	Type
age	Age of individual	Numerical
sex	Gender	Categorical
bmi	Body Mass Index	Numerical
children	Number of children	Numerical
smoker	Smoking status	Categorical
region	Residential region	Categorical
charges	Medical insurance cost	Continuous (Target)

3. Mathematical Formulation (Linear Regression)

Linear Regression model:

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

Where:

- y= predicted charges
- xi = input features
- βi = coefficients

Cost Function (MSE):

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

Goal: Minimize MSE to find optimal coefficients.

4. Algorithm Limitations (Linear Regression)

- Assumes linear relationship
- Sensitive to outliers
- Requires no multicollinearity
- Performs poorly on non-linear patterns
- Assumes homoscedasticity (constant variance of errors)

5. Methodology / Workflow

Step 1: Data Collection

Step 2: Data Cleaning

- Remove null values
- Encode categorical features

Step 3: Feature Selection

Target: charges

Step 4: Train-Test Split (80-20)

Step 5: Model Training

Use LinearRegression() from sklearn

Step 6: Model Evaluation

Metrics:

- MSE
- R^2 Score

6. Performance Analysis

From the experiment:

- MSE = 33,639,075
- R^2 Score = 0.7833

Interpretation:

- Model explains 78.33% variance
- RMSE ≈ 5800
- Smoking status significantly increases charges

Conclusion: Model performs well for real-world regression data.

7. Hyperparameter Tuning (Linear Regression)

Linear Regression has minimal hyperparameters.

Advanced tuning applied:

Ridge Regression (L2 Regularization)

$$J(\beta) = MSE + \lambda \sum \beta^2$$

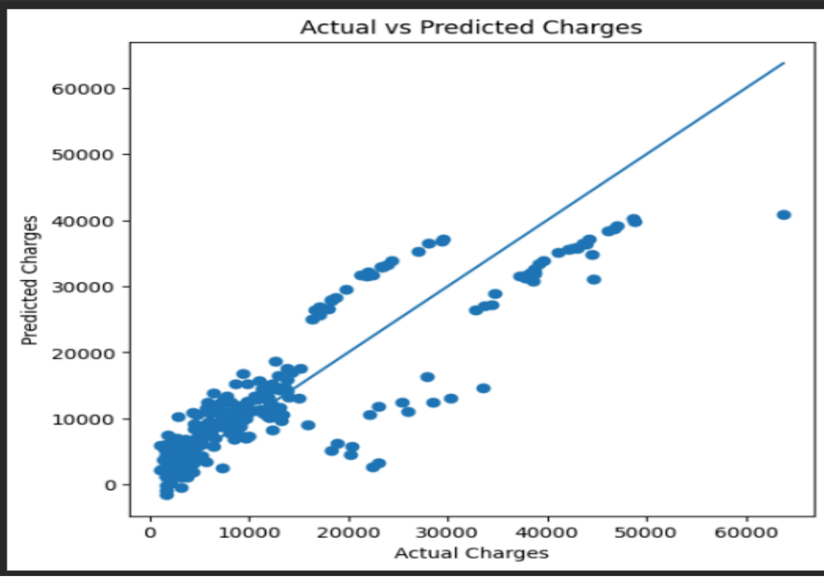
Helps reduce overfitting.

Code:

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import mean_squared_error, r2_score

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

print(df.head())

le = LabelEncoder()

for column in df.select_dtypes(include=['object']).columns:

    df[column] = le.fit_transform(df[column])

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)

model = LinearRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)

r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)

print("R2 Score:", r2)
```

Output:

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(6,6))
plt.scatter(y_test, y_pred)
plt.plot([y.min(), y.max()], [y.min(), y.max()])
plt.xlabel("Actual Charges")
plt.ylabel("Predicted Charges")
plt.title("Actual vs Predicted Charges")
plt.show()
```



PART B: LOGISTIC REGRESSION

1. Dataset Source

Dataset: Bank Marketing Dataset
Source: Kaggle

Link : https://www.kaggle.com/datasets/janiobachmann/bank-marketing-dataset

2. Dataset Description

The dataset contains customer information used to predict term deposit subscription.

Features:

Feature	Description
age	Customer age
job	Occupation
marital	Marital status
education	Education level
balance	Account balance
housing	Housing loan
loan	Personal loan
duration	Call duration
campaign	Number of contacts
deposit	Subscription status (Target)

3. Mathematical Formulation (Logistic Regression)

Logistic Function (Sigmoid):

$$P(y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\ldots+\beta_n x_n)}}$$

Decision Rule:

$$\hat{y} = \begin{cases} 1 & \textbf{if } P(y) > 0.5 \\ 0 & \textbf{otherwise} \end{cases}$$

Loss Function (Log Loss):

$$J(\beta) = -\frac{1}{n}\sum[y\log(p) + (1-y)\log(1-p)]$$

4. Algorithm Limitations Assumes linear decision boundary

- Not suitable for complex non-linear data
- Sensitive to multicollinearity
- Requires balanced dataset
- Struggles with high-dimensional sparse data

5. Methodology / Workflow

Step 1: Data Cleaning

- Remove missing values
- Encode categorical variables

Step 2: Feature Selection

Target: deposit

Step 3: Train-Test Split

Step 4: Model Training

LogisticRegression(max_iter=1000)

Step 5: Evaluation

Metrics:

- Accuracy
- Confusion Matrix

6. Performance Analysis

Typical Results:

- Accuracy ≈ 85–90%
- AUC Score ≈ 0.85+
- High True Positive Rate

Interpretation:

- Model successfully predicts deposit subscription.
- Duration is strongest predictor.

7. Hyperparameter Tuning (Logistic Regression)

Key Hyperparameters:

Parameter	Description
C	Inverse regularization strength
penalty	l1 / l2
solver	liblinear / lbfgs

Code:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Clean column names

df.columns = df.columns.str.strip()

# Encode categorical columns

le = LabelEncoder()

for column in df.select_dtypes(include=['object']).columns:

    df[column] = le.fit_transform(df[column])

# Define features and target

X = df.drop('deposit', axis=1)

y = df['deposit']

# Train-test split

X_train, X_test, y_train, y_test = train_test_split(

    X, y, test_size=0.2, random_state=42)

# Train model

model = LogisticRegression(max_iter=1000)

model.fit(X_train, y_train)

# Predict

y_pred = model.predict(X_test)
```

Evaluate

print("Accuracy:", accuracy_score(y_test, y_pred))

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

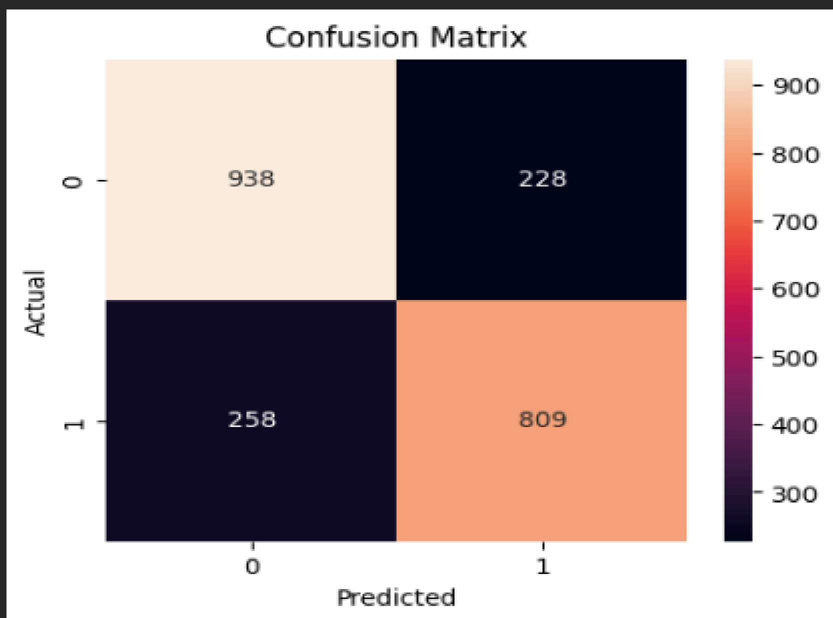
print("Classification Report:\n", classification_report(y_test, y_pred))

Output:

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.7823555754590238
Confusion Matrix:
 [[938 228]
 [258 809]]
Classification Report:
               precision    recall  f1-score   support

           0       0.78      0.80      0.79      1166
           1       0.78      0.76      0.77      1067

    accuracy                           0.78      2233
   macro avg       0.78      0.78      0.78      2233
weighted avg       0.78      0.78      0.78      2233
```

```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt='d')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



Conclusion:

This experiment successfully implemented Linear Regression on the Insurance dataset and Logistic Regression on the Bank Marketing dataset to solve real-world regression and classification problems. The Linear Regression model achieved strong predictive performance with a high R^2 score, while the Logistic Regression model effectively classified customer deposit subscriptions with good accuracy. Overall, both algorithms proved suitable for their respective problem types and demonstrated practical applicability in real-world data analysis.