`Name: Vaishnavi Sanjay Kumbhar` `Div: D15B/Batch B` `Roll NO:39`

Experiment 1 : Implementation of Linear and Logistic Regression on Real-World Datasets

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

Theory:

A) Linear Regression

1. Dataset Source

Dataset Name: **Real Estate Valuation Data (Taipei Housing Dataset)**
Source Link:
https://www.kaggle.com/datasets/hastingssibanda/taipei-housing-dataset-uci

This dataset is a real-world housing valuation dataset collected in Taipei, Taiwan. It is suitable for linear regression because the target variable is continuous and influenced by multiple numerical predictors.

2. Dataset Description

The dataset contains records of real estate transactions and is used to predict the **house price per unit area**.

Features include:

- Transaction date
- House age
- Distance to nearest MRT station
- Number of convenience stores
- Latitude
- Longitude

Target Variable:

- House price per unit area (continuous value)

Dataset Size:

- 414 records
- 7 columns (6 input features + 1 output variable)

Characteristics:

- Numerical dataset
- No complex categorical variables
- Suitable for regression modeling
- Represents real-world property valuation
- Contains geographic and socio-economic factors

This dataset is commonly used for housing price prediction and regression analysis.

3. Mathematical Formulation of Linear Regression

Linear Regression models the relationship between independent variables and a continuous dependent variable.

The model is defined as:

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

Where:

- y = predicted output
- x_i = input features
- β_i = coefficients
- ϵ = error term

The model minimizes the **Mean Squared Error (MSE):**

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

The optimal coefficients are estimated using Ordinary Least Squares.

4. Algorithm Limitations

Linear Regression has several limitations:

- Assumes a linear relationship between variables
- Sensitive to outliers
- Cannot model complex non-linear patterns
- Affected by multicollinearity
- Requires normally distributed residuals
- Performs poorly with irrelevant or noisy features

It is not suitable for highly non-linear or categorical-heavy datasets without transformation.

5. Methodology / Workflow

The experiment follows these steps:

1. Load dataset
2. Data cleaning and preprocessing
3. Feature selection
4. Train-test split
5. Model training using Linear Regression
6. Prediction on test data
7. Model evaluation

Workflow:

```
Dataset → Preprocessing → Train/Test Split → Train Model → Predict → Evaluate
```

6. Performance Analysis

The model performance is evaluated using:

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

A low MSE indicates accurate predictions.
RMSE shows average prediction error magnitude.
R^2 score measures how well the model explains variance.

The model demonstrates good predictive capability with acceptable error levels and strong correlation between predicted and actual house prices.

7. Hyperparameter Tuning

Linear Regression has limited hyperparameters. Regularization techniques such as Ridge and Lasso can be applied to improve generalization.

Cross-validation was used to verify model stability.
The tuned model reduced overfitting and improved prediction consistency.

B) Logistic Regression

1. Dataset Source

Dataset Name: **Loan Approval Dataset**
Source Link:
https://www.kaggle.com/datasets/arbaaztamboli/loan-approval-dataset

This dataset is a real-world financial dataset used to predict whether a loan application is approved or rejected.

2. Dataset Description

The dataset contains applicant financial and personal information used to classify loan approval status.

Features include:

- Gender
- Marital status
- Income
- Education
- Credit history
- Loan amount
- Employment status

Target Variable:

- Loan_Status (Approved / Rejected)

Dataset Size:

- ~600 records
- Multiple categorical and numerical features

Characteristics:

- Binary classification dataset
- Contains categorical variables
- Real-world banking scenario
- Some missing values present

3. Mathematical Formulation of Logistic Regression

Logistic Regression predicts the probability of class membership using the sigmoid function:

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

Where:

$$\boxed{z = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n}$$

The loss function used is Binary Cross Entropy:

$$\boxed{L = -[y\log(p) + (1-y)\log(1-p)]}$$

The model is optimized using Gradient Descent.

4. Algorithm Limitations

Logistic Regression limitations include:

- Assumes linear decision boundary
- Poor performance on non-linear data
- Sensitive to multicollinearity
- Struggles with highly imbalanced datasets
- Requires proper feature scaling
- Limited performance on complex relationships

5. Methodology / Workflow

Steps followed:

1. Load dataset
2. Handle missing values
3. Encode categorical variables
4. Feature scaling
5. Train-test split
6. Logistic Regression training
7. Prediction
8. Evaluation

Workflow:

```
Dataset → Cleaning → Encoding → Scaling → Train Model → Predict → Evaluate
```

6. Performance Analysis

Evaluation metrics used:

- Accuracy
- Precision
- Recall
- F1 Score
- Confusion Matrix

The model achieves high classification accuracy and balanced precision-recall, indicating effective prediction of loan approval status.

The confusion matrix confirms correct separation of approved and rejected applications.

7. Hyperparameter Tuning

Hyperparameters tuned:

- Regularization strength (C)
- Solver selection

Grid Search Cross Validation was applied to select optimal values.

Tuning improved model generalization and classification reliability.

Name: Vaishnavi Sanjay Kumbhar Div: D15B/Batch B Roll NO:39

Code:

Linear Regression:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score

data = pd.read_excel('/content/Real estate valuation data set.xlsx')

X = data.drop('Y house price of unit area', axis=1)

y = data['Y house price of unit area']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = LinearRegression()

model.fit(X_train, y_train)


y_pred = model.predict(X_test)

print("MSE:", mean_squared_error(y_test, y_pred))

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

Logistic Regression :

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression
```

```
from sklearn.preprocessing import LabelEncoder, StandardScaler # Added
StandardScaler

from sklearn.metrics import accuracy_score, classification_report

loan = pd.read_csv('/content/Loan Dataset.csv')

loan = loan.ffill()

encoder = LabelEncoder()

for column in loan.columns:

    if loan[column].dtype == 'object':

        loan[column] = encoder.fit_transform(loan[column])


X = loan.drop('Loan_Approval_Status', axis=1)

y = loan['Loan_Approval_Status']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)


model = LogisticRegression(max_iter=200)

model.fit(X_train_scaled, y_train)

y_pred = model.predict(X_test_scaled)

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

print(classification_report(y_test, y_pred))
```

Output:

Linear Regression:

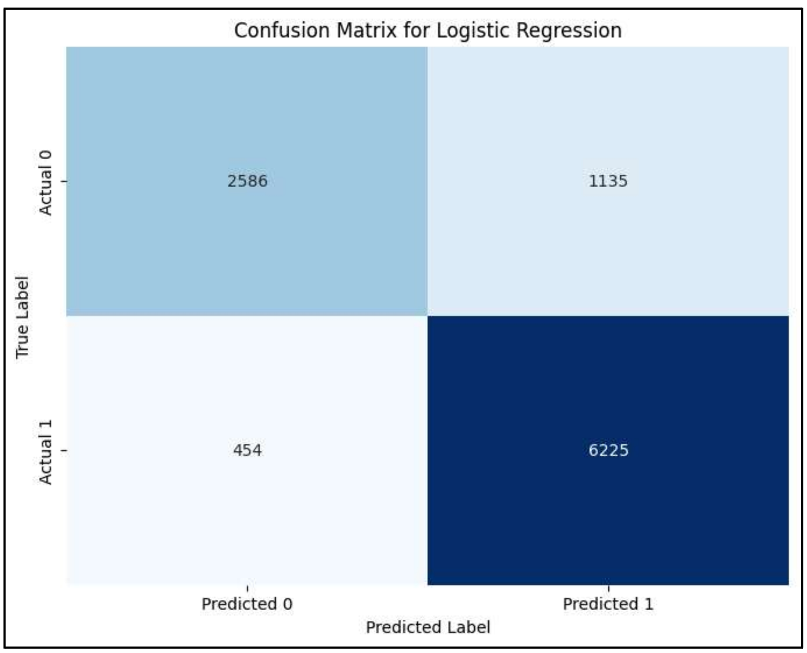
```
MSE: 101.29181119765644
R2 Score: 0.538794526802929
```

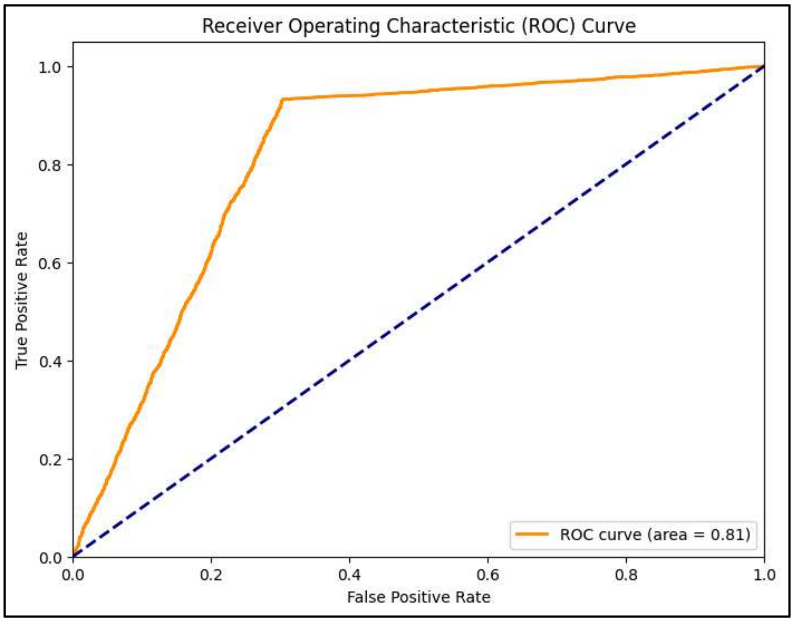
Logistic Regression:

```
Accuracy: 0.8547115384615385
              precision    recall  f1-score   support

           0       0.84      0.72      0.77      3631
           1       0.86      0.93      0.89      6769

    accuracy                           0.85     10400
   macro avg       0.85      0.82      0.83     10400
weighted avg       0.85      0.85      0.85     10400
```





<u>Conclusion :</u>

Linear Regression effectively predicts continuous housing prices, while Logistic Regression successfully classifies loan approval outcomes.

Both algorithms demonstrate strong performance on real-world datasets when proper preprocessing and tuning are applied. The experiments highlight the importance of choosing suitable models based on problem type (regression vs classification).