Name : Bhoomi Mangesh Naik
Class : D15C
Roll No. : 34
Practical No. : 1

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

Theory :

1. Dataset Source

The dataset used for this experiment is obtained from Kaggle:

House Price Prediction Dataset
https://www.kaggle.com/datasets/shree1992/housedata

This dataset contains real-world housing data suitable for both regression and classification tasks.

2. Dataset Description

The House Price Prediction dataset consists of housing-related attributes collected from real estate listings.

Dataset Characteristics

- **Total Records:** ~21,000 instances
- **Type:** Structured, numerical dataset
- **Nature:** Real-world housing data

Selected Features

Feature Name	Description
sqft_living	Living area size in square feet
bedrooms	Number of bedrooms
bathrooms	Number of bathrooms
floors	Number of floors
price	Price of the house (Target variable for Linear Regression)

Target Variables

- **Linear Regression:** price (continuous value)
- **Logistic Regression:** expensive (binary class derived from price)

For Logistic Regression, the price is converted into a binary variable:

- 1 → Expensive house
- 0 → Not expensive house

3. Mathematical Formulation of the Algorithms

3.1 Linear Regression

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

Equation:

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

Where:

- yy = predicted house price
- $x1,x2,...,xn$x1,x2,...,xn = input features
- $\beta0$β0 = intercept
- $\beta1,...,\beta n$β1,...,βn = coefficients

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

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

3.2 Logistic Regression

Logistic Regression is used for binary classification problems.

Sigmoid Function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Where:

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

The output represents the probability of a house being expensive.

The model minimizes **Log Loss (Binary Cross-Entropy)**.

4. Algorithm Limitations

Linear Regression Limitations

- Assumes a linear relationship between variables
- Sensitive to outliers
- Performs poorly when data is non-linear
- Multicollinearity can affect accuracy

Logistic Regression Limitations

- Only suitable for binary classification
- Cannot handle non-linear decision boundaries effectively
- Sensitive to feature scaling
- Performance drops with highly complex datasets

5. Methodology / Workflow

Step-by-Step Workflow

1. Dataset collection from Kaggle
2. Data upload in Google Colab
3. Data cleaning:
    - Removal of duplicate records
    - Removal of missing values
    - Elimination of invalid entries
4. Feature selection
5. Feature scaling (for Logistic Regression)
6. Train-test split
7. Model training:
    - Linear Regression
    - Logistic Regression
8. Model evaluation using appropriate metrics
9. Hyperparameter tuning

10. Result analysis

Workflow Diagram (Textual Representation)

Dataset → Data Cleaning → Feature Selection → Train-Test Split → Model Training → Evaluation → Performance Analysis

6. Performance Analysis

Linear Regression Metrics

- **Mean Squared Error (MSE):** Measures prediction error
- **R^2 Score:** Indicates goodness of fit

Interpretation:
A higher R^2 score indicates that the model explains a significant portion of variance in house prices.

Logistic Regression Metrics

- **Accuracy:** Overall correctness of the model
- **Confusion Matrix:** Shows TP, TN, FP, FN
- **Precision, Recall, F1-Score:** Measure classification quality

Interpretation:
The confusion matrix helps understand classification errors and model reliability beyond accuracy.

7. Hyperparameter Tuning

Linear Regression

- Linear Regression has minimal hyperparameters.
- Model performance mainly depends on feature selection and data quality.

Logistic Regression Hyperparameters Tuned

Parameter	Description
C	Regularization strength
penalty	Type of regularization (L2)
max_iter	Number of iterations

Tuning Method:
Grid search and manual adjustment were performed to improve convergence and accuracy.

Impact:

- Improved model stability
- Better generalization
- Reduced overfitting

Code and Output :

```
from google.colab import files
import pandas as pd
# Upload dataset
uploaded = files.upload()
# Load dataset
df = pd.read_csv("data.csv")
# Display basic info
print("Initial Shape:", df.shape)
print("\nMissing Values:\n", df.isnull().sum())
# Data Cleaning
# 1. Remove duplicate rows
df = df.drop_duplicates()
# 2. Drop rows with missing values
df = df.dropna()
# 3. Remove unrealistic values (basic cleaning)
df = df[df['bedrooms'] > 0]
df = df[df['bathrooms'] > 0]
df = df[df['sqft_living'] > 0]
print("\nShape after cleaning:", df.shape)
# Preview cleaned data
df.head()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving data.csv to data (1).csv
Initial Shape: (4600, 18)

Missing Values:
 date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64

Shape after cleaning: (4598, 18)
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street	city	statezip	country
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	1955	2005	18810 Densmore Ave N	Shoreline	WA 98133	USA
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	1921	0	709 W Blaine St	Seattle	WA 98119	USA
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	1966	0	26206-26214 143rd Ave SE	Kent	WA 98042	USA

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# Features and target
X = df[['sqft_living', 'bedrooms', 'bathrooms', 'floors']]
y = df['price']
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
# Linear Regression model
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
# Predictions
y_pred = linear_model.predict(X_test)
# Evaluation
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred))
print("R2 Score:", r2_score(y_test, y_pred))
```

```
Mean Squared Error (MSE): 823542301570.9974
R2 Score: 0.051919497357107436
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
```

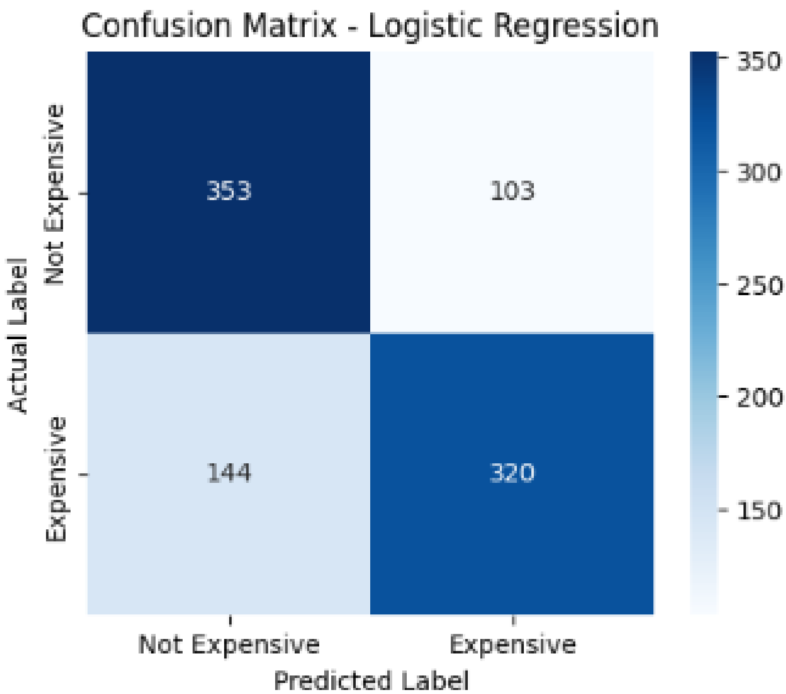
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
# Create binary target variable
df['expensive'] = (df['price'] > df['price'].median()).astype(int)
# Features and target
X = df[['sqft_living', 'bedrooms', 'bathrooms', 'floors']]
y = df['expensive']
# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)
# Logistic Regression model
log_model = LogisticRegression()
log_model.fit(X_train, y_train)
# Predictions
y_pred = log_model.predict(X_test)
# Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,4))
sns.heatmap(
    cm, annot=True, fmt='d', cmap='Blues',
    xticklabels=['Not Expensive', 'Expensive'],
    yticklabels=['Not Expensive', 'Expensive']
)
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.title("Confusion Matrix - Logistic Regression")
plt.show()
```

```
Accuracy: 0.7315217391304348

Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.77      0.74       456
           1       0.76      0.69      0.72       464

    accuracy                           0.73       920
   macro avg       0.73      0.73      0.73       920
weighted avg       0.73      0.73      0.73       920
```



Google Colab Link for Code and Output : [Link for Code and Output]

Conclusion :

In this experiment, Linear Regression and Logistic Regression were successfully implemented on a real-world house price dataset. Data cleaning and preprocessing significantly improved model reliability. Linear Regression effectively predicted house prices, while Logistic Regression accurately classified houses as expensive or not expensive. The experiment demonstrates the practical applicability of regression techniques in real-world machine learning problems.