AAYUSH PATIL - 56

MLDL EXPERIMENT 1

AIM-

Implement Linear and Logistic Regression on real-world datasets

THEORY-

Linear Regression

Linear Regression is a supervised machine learning algorithm used to predict a **continuous numerical value** based on one or more independent variables.

It assumes a linear relationship between the dependent variable and independent variables.

The main objective of linear regression is to find the best-fitting straight line that minimizes the difference between actual values and predicted values.

Linear Regression is used when:

- The output variable is continuous (numeric)
- We want to understand how independent variables influence the output
- We want to predict future numeric values

In business analytics, it is widely used for:

- Sales prediction
- Demand forecasting
- Revenue estimation
- Trend analysis

Linear Regression Formula

Simple Linear Regression (One variable)

$$Y = b_0 + b_1 X$$

Where:

- YYY = Dependent variable
- XXX = Independent variable
- b0b_0b0 = Intercept
- b1b_1b1 = Slope (coefficient)

Multiple Linear Regression (More than one variable)

$$Y = b_0 + b_1 X_1 + b_2 X_2 + ... + b_n X_n$$

Where:

- YYY = Target variable
- X1,X2,...XnX_1, X_2, ... X_nX1,X2,...Xn = Independent variables
- b0b_0b0 = Intercept
- b1,b2,...bnb_1, b_2, ... b_nb1,b2,...bn = Coefficients

Application on Our Amazon Sales Dataset

Dataset Used: Amazon Sales Dataset

Objective: To predict **quantity_sold**

Dependent Variable (Target)

Quantity_sold
Reason:

- It represents actual product sales
- It is not derived from other variables

Independent Variables Used

From the dataset:

- price
- discount_percent
- rating
- review_count

Model Used in Our Case

The regression equation becomes:

Quantity_sold =

$$quantity\_sold = b_0 + b_1(price) + b_2(discount\_percent) + b_3(rating) + b_4(review\_count)$$

Logistic regression

Logistic Regression is a supervised machine learning algorithm used for **classification problems**.

Unlike Linear Regression, which predicts continuous values, Logistic Regression predicts **categorical outcomes** (such as Yes/No, 0/1, True/False).

It estimates the probability that a given input belongs to a particular class.

Purpose of Using Logistic Regression

Logistic Regression is used when:

- The output variable is categorical
- We need probability-based classification
- We want to understand the influence of features on a binary outcome

In business analytics, it is commonly used for:

- Purchase prediction (Buy / Not Buy)
- Fraud detection
- Customer churn prediction
- High demand vs Low demand classification

Logistic Regression Formula

Unlike linear regression, logistic regression uses the **sigmoid function** to convert output into probability.

Step 1: Linear Combination

$$Z = b_0 + b_1X_1 + b_2X_2 + ... + b_nX_n$$

Step 2: Apply Sigmoid Function

$$P(Y = 1) = \frac{1}{1 + e^{-Z}}$$

Where:

- P(Y=1)P(Y=1)P(Y=1) = Probability of belonging to class 1
- eee = Euler's number
- ZZZ = Linear combination of inputs

Final Prediction Rule

If:

$$P(Y = 1) \geq 0.5$$

→ Class = 1
Else → Class = 0

How Parameters Are Calculated

Logistic Regression does NOT use Ordinary Least Squares.

Instead, it uses:

Maximum Likelihood Estimation (MLE)
It maximizes the likelihood function to find coefficients that best separate the classes.

Loss function used:

$$Loss = -\sum [y \log(p) + (1 - y) \log(1 - p)]$$

This is called **Binary Cross-Entropy Loss**.

Application on Our Amazon Sales Dataset

Dataset Used: Amazon Sales Dataset
Objective:

To classify whether a product has:

- High Sales (1)
- Low Sales (0)

Target Variable Creation

Since logistic regression requires categorical output, we converted:

$$quantity\_sold$$

into:

$$high\_sales$$

Where:

- 1 → quantity_sold greater than median
- 0 → quantity_sold less than or equal to median

Independent Variables Used

From dataset:

- price
- discount_percent
- rating

Model Used in Our Case

The logistic regression equation becomes:

$$Z = b_0 + b_1(price) + b_2(discount\_percent) + b_3(rating) + b_4(review\_count)$$

Then probability:

$$P(high\_sales = 1) = \frac{1}{1 + e^{-Z}}$$

Interpretation in Our Experiment

- If probability > 0.5 → Product is predicted to have High Sales
- If probability < 0.5 → Product is predicted to have Low Sales

OBSERVATIONS -

CODE :

```
# ==============================================
# 1. Import Required Libraries
# ==============================================
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error

# ==============================================
# 2. Load Dataset (No Upload Required)
# ==============================================
df = pd.read_csv('/content/amazon_sales_dataset.csv')

print("Dataset Loaded Successfully!\n")
print(df.head())
```

```python
# ==============================================
# 3. Select Features and Target
# ==============================================

X = df[['price', 'discount_percent', 'rating', 'review_count']]
y = df['quantity_sold']

# ==============================================
# 4. Handle Missing Values (Added Step)
# ==============================================

data = pd.concat([X, y], axis=1)
data.dropna(inplace=True)

X = data[['price', 'discount_percent', 'rating', 'review_count']]
y = data['quantity_sold']

# ==============================================
# 5. Train-Test Split
# ==============================================

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# ==============================================
# 6. Train Linear Regression Model
# ==============================================

model = LinearRegression()
model.fit(X_train, y_train)

# ==============================================
# 7. Model Evaluation
# ==============================================

y_pred = model.predict(X_test)

print("\nModel Performance:")
```

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

# =============================================
# 8. Display Derived Formula
# =============================================

intercept = model.intercept_
coefficients = model.coef_

print("\nDerived Linear Regression Formula:\n")
print(f"quantity_sold = {intercept:.4f} "
      f"+ ({coefficients[0]:.4f} * price) "
      f"+ ({coefficients[1]:.4f} * discount_percent) "
      f"+ ({coefficients[2]:.4f} * rating) "
      f"+ ({coefficients[3]:.4f} * review_count)")

# =============================================
# 9. Manual Prediction Section
# =============================================

print("\n--- Enter Values for Prediction ---")

price = float(input("Enter price: "))
discount = float(input("Enter discount_percent: "))
rating = float(input("Enter rating: "))
reviews = float(input("Enter review_count: "))

# Manual Calculation
calculated_value = (
    intercept
    + coefficients[0] * price
    + coefficients[1] * discount
    + coefficients[2] * rating
    + coefficients[3] * reviews
)

print("\nStep-by-step Calculation:")
print(f"Intercept: {intercept:.4f}")
print(f"{coefficients[0]:.4f} * {price} = {coefficients[0] * price:.4f}")
```

```python
print(f"{coefficients[1]:.4f} * {discount} = {coefficients[1] *
discount:.4f}")
print(f"{coefficients[2]:.4f} * {rating} = {coefficients[2] *
rating:.4f}")
print(f"{coefficients[3]:.4f} * {reviews} = {coefficients[3] *
reviews:.4f}")

print("\nPredicted Quantity Sold:", round(calculated_value, 2))
```

OUTPUT :

```
Dataset Loaded Successfully!
...
   order_id  order_date  product_id product_category   price  \
0         1  2022-04-13        2637            Books  128.75
1         2  2023-03-12        2300          Fashion  302.60
2         3  2022-09-28        3670           Sports  495.80
3         4  2022-04-17        2522            Books  371.95
4         5  2022-03-13        1717           Beauty  201.68

   discount_percent  quantity_sold customer_region payment_method  rating  \
0                10              4   North America            UPI     3.5
1                20              5            Asia    Credit Card     3.7
2                20              2          Europe            UPI     4.4
3                15              4     Middle East            UPI     5.0
4                 0              4     Middle East            UPI     4.6

   review_count  discounted_price  total_revenue
0           443            115.88         463.52
1           475            242.08        1210.40
2           183            396.64         793.28
3           212            316.16        1264.64
4           308            201.68         806.72

Model Performance:
R2 Score: -0.0003226473047468481
MSE: 1.9992428322547457

Derived Linear Regression Formula:

quantity_sold = 3.0163 + (0.0000 * price) + (0.0004 * discount_percent) + (-0.0068 * rating) + (-0.0000 * review_count)
```

```
quantity_sold = 3.0163 + (0.0000 * price) + (0.0004 * discount_percent) + (-0.0068 * rating) + (-0.0000 * review_count)

--- Enter Values for Prediction ---
Enter price: 400
Enter discount_percent: 30
Enter rating: 4
Enter review_count: 400

Step-by-step Calculation:
Intercept: 3.0163
0.0000 * 400.0 = 0.0138
0.0004 * 30.0 = 0.0134
-0.0068 * 4.0 = -0.0270
-0.0000 * 400.0 = -0.0143

Predicted Quantity Sold: 3.0
```

2. LOGISTIC REGRESSION

CODE :

```
# ============================================
# 1. Import Libraries
# ============================================
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

# ============================================
# 2. Load Dataset
# ============================================
df = pd.read_csv('/content/amazon_sales_dataset.csv')

print("Dataset Loaded Successfully!\n")
print(df.head())

# ============================================
# 3. Create Binary Target Variable
# ============================================

median_value = df['quantity_sold'].median()

df['high_sales'] = np.where(df['quantity_sold'] > median_value, 1, 0)

print("\nMedian Quantity Sold:", median_value)
print(df[['quantity_sold','high_sales']].head())

# ============================================
# 4. Select Features
# ============================================

X = df[['price', 'discount_percent', 'rating', 'review_count']]
y = df['high_sales']
```

```python
# ============================================
# 5. Train-Test Split
# ============================================

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# ============================================
# 6. Train Logistic Regression
# ============================================

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# ============================================
# 7. Model Evaluation
# ============================================

y_pred = model.predict(X_test)

print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# ============================================
# 8. Display Logistic Formula
# ============================================

intercept = model.intercept_[0]
coefficients = model.coef_[0]

print("\nLogistic Regression Equation (z):")
print(f"z = {intercept:.4f} "
      f"+ ({coefficients[0]:.4f} * price) "
      f"+ ({coefficients[1]:.4f} * discount_percent) "
      f"+ ({coefficients[2]:.4f} * rating) "
      f"+ ({coefficients[3]:.4f} * review_count)")

# ============================================
```

```python
# 9. Manual Prediction
# ============================================

print("\n--- Enter Values for Prediction ---")

price = float(input("Enter price: "))
discount = float(input("Enter discount_percent: "))
rating = float(input("Enter rating: "))
reviews = float(input("Enter review_count: "))

z = (intercept
     + coefficients[0]*price
     + coefficients[1]*discount
     + coefficients[2]*rating
     + coefficients[3]*reviews)

probability = 1 / (1 + np.exp(-z))

print("\nCalculated z value:", round(z,4))
print("Predicted Probability of High Sales:", round(probability,4))

if probability > 0.5:
    print("Prediction: HIGH SALES (1)")
else:
    print("Prediction: LOW SALES (0)")
```

OUTPUT :

```
•••    order_id  order_date  product_id product_category   price  \
    0         1  2022-04-13        2637            Books  128.75
    1         2  2023-03-12        2300          Fashion  302.60
    2         3  2022-09-28        3670           Sports  495.80
    3         4  2022-04-17        2522            Books  371.95
    4         5  2022-03-13        1717           Beauty  201.68

       discount_percent  quantity_sold customer_region payment_method  rating  \
    0                10              4   North America            UPI     3.5
    1                20              5            Asia    Credit Card     3.7
    2                20              2          Europe            UPI     4.4
    3                15              4     Middle East            UPI     5.0
    4                 0              4     Middle East            UPI     4.6

       review_count  discounted_price  total_revenue
    0           443            115.88         463.52
    1           475            242.08        1210.40
    2           183            396.64         793.28
    3           212            316.16        1264.64
    4           308            201.68         806.72

    Median Quantity Sold: 3.0
       quantity_sold  high_sales
    0              4           1
    1              5           1
    2              2           0
    3              4           1
    4              4           1

    Accuracy: 0.5988
```

```
    Accuracy: 0.5988
•••
    Confusion Matrix:
     [[5988    0]
     [4012    0]]

    Classification Report:
                  precision    recall  f1-score   support

               0       0.60      1.00      0.75      5988
               1       0.00      0.00      0.00      4012

        accuracy                           0.60     10000
       macro avg       0.30      0.50      0.37     10000
    weighted avg       0.36      0.60      0.45     10000


    Logistic Regression Equation (z):
    z = -0.4127 + (0.0001 * price) + (0.0014 * discount_percent) + (-0.0059 * rating) + (-0.0001 * review_count)

    --- Enter Values for Prediction ---
    Enter price: 200
    Enter discount_percent: 20
    Enter rating: 3.4
    Enter review_count: 333

    Calculated z value: -0.4087
    Predicted Probability of High Sales: 0.3992
    Prediction: LOW SALES (0)
```

CONCLUSION -

In this experiment, Logistic Regression was applied to the Amazon Sales dataset to classify products into High Sales and Low Sales categories. The continuous variable `quantity_sold` was converted into a binary target variable, and features such as price, discount_percent, rating, and review_count were used as independent variables.

The model used the sigmoid function to calculate the probability of a product achieving high sales and classified it based on a threshold value. This approach helps in analyzing demand patterns and supports better sales prediction and business decision-making.