**Predictive Analysis Lab**

**Isha Yadav**

**500107798**

**Batch 4 AIML Non Hons**

Ques: Differentiate between logistic regression and linear regression through two real-world scenarios.

Hint: Differentiate in terms of i) Definition, ii) Datasets compatibility, iii) Model, iv) Validation Metrics, v) Visualization ( through graphs)

Solution:

**Scenario 1: Linear Regression (Predicting House Prices)**

Linear regression predicts a continuous variable based on one or more input features. Let's say we are predicting house prices based on the house's area in square feet.

**i) Definition:**

Linear regression models the relationship between input features (e.g., area) and the target variable (e.g., price) by fitting a linear equation. It assumes that there is a linear relationship between input features and the target.

**ii) Dataset Compatibility:**

Linear regression is suitable for datasets where the target variable is continuous, such as predicting house prices.

**iii) Model:**

The mathematical model for linear regression is:

y=β0+β1x1+β2x2+…+βnxn+ϵ

Where y is the predicted price, and x1,x2,…,xn​ are features like the area, number of rooms, etc.

**iv) Validation Metrics:**

Common metrics include:

* **Mean Squared Error (MSE)**.
* **Root Mean Squared Error (RMSE)**.
* **R-squared (R²)**: Proportion of variance explained by the model.

**v) Visualization:**

You can visualize linear regression using a scatter plot with the fitted regression line.

**Code for Linear Regression:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

# Sample dataset: House area (sqft) vs. price (in thousands)

data = {'area': [1500, 2000, 2500, 3000, 3500, 4000],

'price': [300, 400, 500, 600, 700, 800]}

df = pd.DataFrame(data)

# Reshape the data for model fitting

X = df['area'].values.reshape(-1, 1) # Feature

y = df['price'].values # Target

# Create a Linear Regression model

model = LinearRegression()

model.fit(X, y)

# Make predictions

y\_pred = model.predict(X)

# Calculate metrics

mse = mean\_squared\_error(y, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y, y\_pred)

print(f"Mean Squared Error: {mse}")

print(f"Root Mean Squared Error: {rmse}")

print(f"R-squared: {r2}")

# Visualization: Scatter plot and the regression line

plt.scatter(X, y, color='blue', label='Actual Prices')

plt.plot(X, y\_pred, color='red', label='Predicted Line')

plt.xlabel('Area (sq ft)')

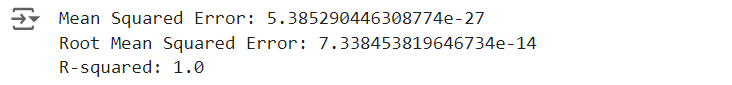
plt.ylabel('Price (in $1000s)')

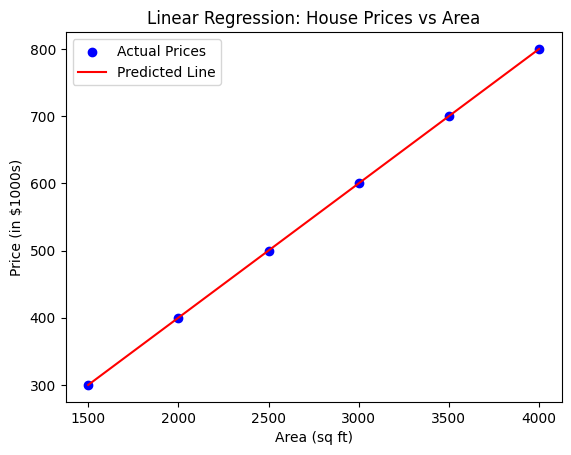
plt.title('Linear Regression: House Prices vs Area')

plt.legend()

plt.show()

**Output:**

****



* **Mean Squared Error (MSE)**: A measure of the average squared difference between actual and predicted values.
* **Root Mean Squared Error (RMSE)**: Square root of MSE, giving the average difference in the same units as the target.
* **R-squared (R²)**: Tells how well the regression model fits the data (ranges from 0 to 1).

The **plot** will show a scatter plot of house area vs. price, with a red regression line showing the predictions.

**Scenario 2: Logistic Regression (Predicting Disease Presence)**

In this case, let’s say we are predicting whether a patient has a disease (binary outcome: yes/no) based on input features such as age, BMI, cholesterol level, etc.

**i) Definition:**

Logistic regression is used for binary classification tasks. It predicts the probability of an event happening (e.g., disease presence) and maps it between 0 and 1 using the **logistic (sigmoid) function**.

**ii) Dataset Compatibility:**

Logistic regression is suited for datasets where the target variable is categorical, particularly binary (yes/no, 0/1). In this case, the target is whether the patient has a disease (1 for yes, 0 for no).

**iii) Model:**

The mathematical model for logistic regression is:

p(y=1)=1 / 1+e− (β0+β1 x1+⋯+βn xn)​

Where p(y=1) is the probability that the patient has the disease given the features x1,x2,…

**iv) Validation Metrics:**

Common metrics for logistic regression:

* **Accuracy**: Proportion of correct predictions.
* **Precision, Recall, F1-Score**: Useful when dealing with imbalanced datasets.
* **AUC-ROC**: Area under the ROC curve for evaluating classification performance.

**v) Visualization:**

Logistic regression is visualized using the sigmoid curve, showing the probability of an event (disease presence) vs. input features.

**Code for Logistic Regression:**

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, roc\_curve, auc

# Sample dataset: Age and BMI vs. Disease presence (1 = yes, 0 = no)

data = {'age': [25, 35, 45, 50, 60, 70],

'bmi': [22, 28, 26, 32, 35, 40],

'disease': [0, 0, 0, 1, 1, 1]} # Binary target

df = pd.DataFrame(data)

# Features and target

X = df[['age', 'bmi']]

y = df['disease']

# Create a Logistic Regression model

log\_model = LogisticRegression()

log\_model.fit(X, y)

# Make predictions

y\_pred = log\_model.predict(X)

y\_prob = log\_model.predict\_proba(X)[:, 1] # Probability of disease

# Calculate accuracy

accuracy = accuracy\_score(y, y\_pred)

conf\_matrix = confusion\_matrix(y, y\_pred)

print(f"Accuracy: {accuracy}")

print(f"Confusion Matrix:\n{conf\_matrix}")

# ROC curve

fpr, tpr, \_ = roc\_curve(y, y\_prob)

roc\_auc = auc(fpr, tpr)

# Plotting the ROC curve

plt.figure()

plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc\_auc:0.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

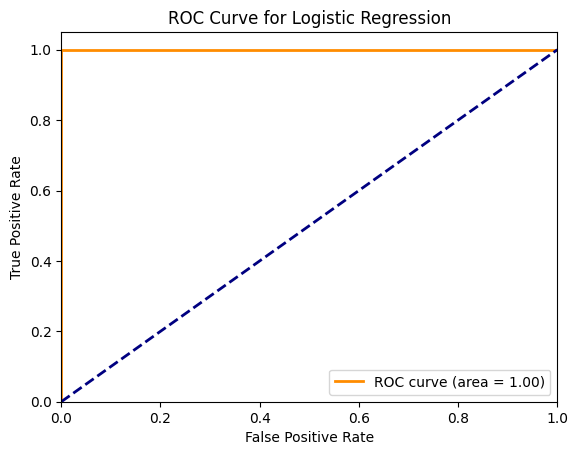
plt.title('ROC Curve for Logistic Regression')

plt.legend(loc="lower right")

plt.show()

**Output:**

****



* **Accuracy**: Shows how often the model predicts correctly.
* **Confusion Matrix**: Provides insight into the model's true/false positive and negative rates.
* **ROC Curve and AUC**: Visualizes the model’s ability to distinguish between classes. The closer the curve is to the top-left corner, the better.

The **ROC curve** will show the trade-off between the true positive rate and the false positive rate for various threshold settings.