Logistic Regression is a supervised machine learning algorithm used for **classification**, not regression (despite the name). It's a linear model that estimates the **probability** that an instance belongs to a particular class.

It uses the **sigmoid function** (or logistic function) to map any real-valued number into a probability between 0 and 1.

## Theoretical Classification Types

Logistic Regression is categorized based on the number of classes the target variable contains:

### 1. Binary (or Binomial) Logistic Regression ☯️

This is the simplest and most common form, used when the dependent variable has **only two possible outcomes** (classes). The model predicts the probability P(Y=1), and if this probability exceeds a threshold (usually 0.5), it classifies the instance as class 1; otherwise, as class 0.

|  |  |
| --- | --- |
| Feature | Description |
| **Output** | 0 or 1 (e.g., No/Yes, False/True, Spam/Ham) |
| **Equation** | The probability is calculated as: P(Y=1)=1+e−(β0​+β1​x1​+⋯+βn​xn​)1​ |
| **Use Cases** | **Credit Default:** Will a borrower default (Yes/No)? ∙ **Email Spam:** Is an email spam (Spam/Ham)? ∙ **Churn Prediction:** Will a customer leave (Churn/Stay)? |

### 2. Multi-Class (or Multinomial) Logistic Regression 🚦

Used when the dependent variable has **three or more unordered categories**. The model calculates the probability of the instance belonging to each class and selects the class with the highest probability.

|  |  |
| --- | --- |
| Feature | Description |
| **Output** | 3 or more unordered classes (e.g., Cat/Dog/Bird, Class A/B/C) |
| **Mechanism** | Uses the **Softmax function** (a generalization of the sigmoid) to output a set of probabilities that sum to 1. |
| **Use Cases** | **Image Classification:** Identifying an object as one of many categories. ∙ **Handwriting Recognition:** Classifying a digit as 0,1,2,…,9. ∙ **Product Categorization:** Assigning a product to a department (e.g., Electronics/Clothing/Books). |

### 3. Ordinal Logistic Regression 📊

Used when the dependent variable has **three or more ordered categories**. The relationship between the independent variables and the predicted outcomes is maintained across all levels. This is less common in machine learning libraries like scikit-learn but is standard in statistical packages.

|  |  |
| --- | --- |
| Feature | Description |
| **Output** | 3 or more ordered classes (e.g., Low/Medium/High, 1-Star/2-Star/3-Star) |
| **Mechanism** | It doesn't treat the problem as independent classes; instead, it models the cumulative probability of being *below* a certain level. |
| **Use Cases** | **Customer Satisfaction:** Predicting a rating (e.g., Poor/Fair/Good/Excellent). ∙ **Disease Stage:** Classifying the severity of a disease. |

## Python Examples

The scikit-learn library handles both Binary and Multi-Class Logistic Regression using a single class: LogisticRegression.

### Example 1: Binary Logistic Regression (Churn Prediction)

We'll use a simplified dataset to predict whether a user will churn or stay.

Python

import numpy as np

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

# Sample Data: (X) features vs (y) target

# X: [Monthly Usage (hrs), Support Calls]

X = np.array([[100, 1], [50, 5], [120, 0], [40, 4], [80, 2]])

# y: Target (0=Stay, 1=Churn)

y = np.array([0, 1, 0, 1, 0])

# Split data (standard ML workflow)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=42)

# Initialize and train the Binary Logistic Regression model

# The 'solver' parameter dictates the optimization method

binary\_model = LogisticRegression(solver='liblinear', random\_state=42)

binary\_model.fit(X\_train, y\_train)

# Predict class (0 or 1)

predictions = binary\_model.predict(X\_test)

# Predict probabilities for each class

probabilities = binary\_model.predict\_proba(X\_test)

print(f"Predictions: {predictions}")

# print(f"Probabilities (P(0), P(1)):\n {probabilities.round(2)}")

print(f"Model Accuracy: {accuracy\_score(y\_test, predictions):.2f}")

### Example 2: Multi-Class Logistic Regression (Handwriting Recognition)

This example uses a famous built-in dataset to classify images of handwritten digits (0 through 9).

Python

from sklearn.datasets import load\_digits  
from sklearn.linear\_model import LogisticRegression  
  
# Load the multi-class dataset (10 classes: 0 to 9)  
digits = load\_digits()  
X\_multi, y\_multi = digits.data, digits.target  
  
# Split data  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_multi, y\_multi, test\_size=0.2, random\_state=42)  
  
# Initialize and train the Multi-Class Logistic Regression model  
# scikit-learn automatically uses the appropriate strategy (Multinomial/Softmax)  
multi\_model = LogisticRegression(solver='lbfgs', max\_iter=2000, random\_state=42)  
multi\_model.fit(X\_train, y\_train)  
  
# Predict the class (the digit 0-9)  
predictions\_multi = multi\_model.predict(X\_test)  
  
print(f"Number of classes: {len(np.unique(y\_multi))}")  
print(f"First 5 predictions: {predictions\_multi[:5]}")  
print(f"Multi-Class Model Accuracy: {multi\_model.score(X\_test, y\_test):.3f}")