Logistic Regression is a popular algorithm for binary classification tasks. A simple example of classifying images using Logistic Regression.

use the MNIST dataset, which contains images of handwritten digits and classify whether the digit is a '0' or a '1'.

Step-by-Step Explanation and Code

1. Import Libraries: Import necessary libraries such as numpy, pandas, matplotlib, and sklearn.
2. Load the Dataset: use the fetch_openml function from sklearn.datasets to load the MNIST dataset.
3. Preprocess the Data: This involves selecting only the digits '0' and '1', normalizing the pixel values, and splitting the data into training and testing sets.
4. Train the Model: use LogisticRegression from sklearn.linear_model to fit the model on the training data.
5. Evaluate the Model: use accuracy, confusion matrix, and visualization to understand the performance of the model.

1. **Import Libraries:**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay

```

✓ 3.4s

2. **Load the dataset**

```
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784', version=1)

```

✓ 16.9s

Preprocess the Data: Select only the digits '0' and '1':

```
mask = mnist.target.astype(int) <= 1
X, y = mnist.data[mask], mnist.target[mask].astype(int)

```

✓ 0.1s

Data normalization:

```
X = X / 255.0

0.0s
```

- **Pixel Value Range**: In most image formats, pixel intensity values range from 0 to 255 (8-bit representation)
  - 0 = black
  - 255 = white (or maximum color intensity)

Data Split:

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

  - test_size=0.2: 20% of the data will be used for testing, 80% for training
  - random_state=42: Sets a specific random seed for reproducible results
    random seed(input) = randomly selecting the 42 images
- X_train: Features for training the model (80% of X)
- X_test: Features for testing the model (20% of X)
- y_train: Target values for training (80% of y)
- y_test: Target values for testing (20% of y)

Training the Logistic Regression Model:

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

[6]  0.5s
```

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

Make predictions:

```
y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)

0.1s
```

Model Performance Evaluation:

```
1  train_accuracy = accuracy_score(y_train, y_train_pred)
2  test_accuracy = accuracy_score(y_test, y_test_pred)
3  print("Training Accuracy:", train_accuracy)
4  print("Testing Accuracy:", test_accuracy)
5
```

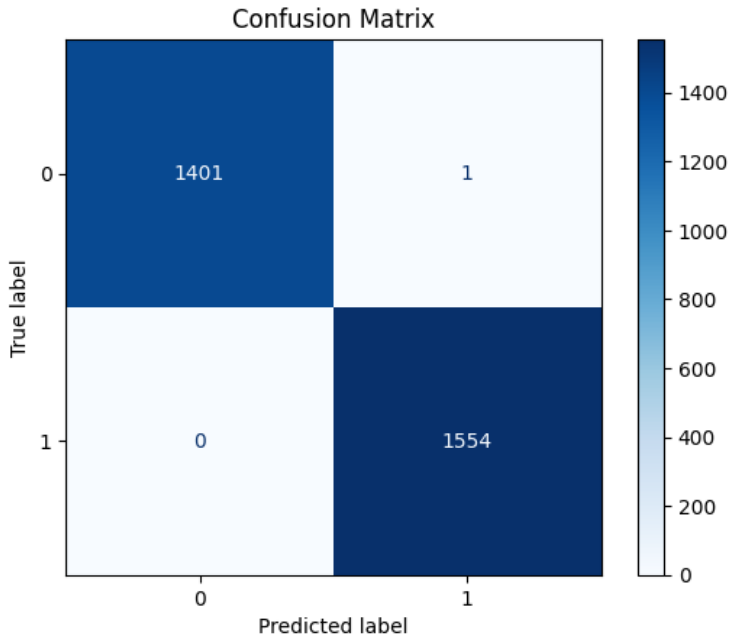
[8] ✓ 0.0s

```
Training Accuracy: 0.9999154262516915
Testing Accuracy: 0.9996617050067659
```

Confusion Matrix:

```
1  conf_matrix = confusion_matrix(y_test, y_test_pred)
2  ConfusionMatrixDisplay(conf_matrix).plot(cmap='Blues')
3  plt.title("Confusion Matrix")
4  plt.show()
5
```

✓ 0.3s Python



1. **True Positives (TP): 1554 instances**
   - These are cases where the model correctly predicted class 1
   - Bottom right cell (true label = 1, predicted label = 1)
2. **True Negatives (TN): 1401 instances**
   - These are cases where the model correctly predicted class 0
   - Top left cell (true label = 0, predicted label = 0)
3. **False Positives (FP): 0 instances**
   - These would be cases where the model incorrectly predicted class 1 when the true class was 0
   - Bottom left cell (true label = 0, predicted label = 1)

- In this case, there are zero false positives, which is excellent

4. **False Negatives (FN): 1 instance**
   - This is a case where the model incorrectly predicted class 0 when the true class was 1
   - Top right cell (true label = 1, predicted label = 0)

Performance Metrics

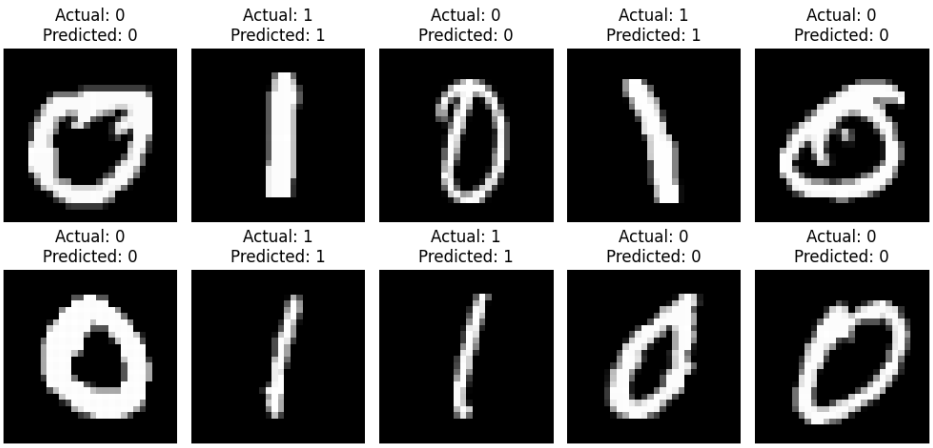
Based on these values, we can calculate key performance metrics:

1. Accuracy: 99.97% (2955/2956)
   - Formula: (TP + TN) / (TP + TN + FP + FN)
   - (1554 + 1401) / (1554 + 1401 + 0 + 1) = 2955/2956
2. Precision (Positive Predictive Value): 100%
   - Formula: TP / (TP + FP)
   - 1554 / (1554 + 0) = 1.0
3. Recall (Sensitivity): 99.94%
   - Formula: TP / (TP + FN)
   - 1554 / (1554 + 1) = 0.9994
4. Specificity: 100%
   - Formula: TN / (TN + FP)
   - 1401 / (1401 + 0) = 1.0
5. F1-Score: 99.97%
   - Formula: 2 * (Precision * Recall) / (Precision + Recall)
   - 2 * (1.0 * 0.9994) / (1.0 + 0.9994) = 0.9997

Final Results:

```
1  fig, axes = plt.subplots(2, 5, figsize=(10, 5))
2  for i, ax in enumerate(axes.flat):
3      ax.imshow(X_test.iloc[i].values.reshape(28, 28), cmap='gray')
4      ax.set_title(f'Actual: {y_test.iloc[i]}\nPredicted: {y_test_pred[i]}')
5      ax.axis('off')
6  plt.tight_layout()
7  plt.show()
8
```

✓ 0.9s



A confusion matrix is a tool often used in machine learning to evaluate the performance of a classification ML model. It allows to understand how well ML model is performing by comparing the true labels of dataset with the predictions made by ML model.

Structure of a Confusion Matrix

For a binary classification problem, a confusion matrix typically looks like this:

	Predicted Negative	**Predicted Positive**
Actual Negative	True Negative (TN)	False Positive (FP)
Actual Positive	False Negative (FN)	True Positive (TP)

For multi-class classification, the matrix will be larger, with rows and columns for each class.

- True Positive (TP): The number of instances correctly predicted as positive.

True Positive (TP) A True Positive (TP) is when the model correctly predicts the positive class. Disease is there model predicts disease is there

Example:

- Imagine you have a medical test for detecting a disease.
- If a patient has the disease (actual positive) and the test correctly identifies the disease (predicted positive), this is a True Positive.

Scenario: The test says "You have the disease," and indeed, the patient has the disease.

True Negative (TN) - A True Negative (TN) is when the model correctly predicts the negative class. Disease is there model predicts disease is not there

Example:

- Using the same medical test scenario:
- If a patient does not have the disease (actual negative) and the test correctly identifies that the patient does not have the disease (predicted negative), this is a True Negative.

Scenario: The test says "You do not have the disease," and indeed, the patient does not have the disease.

False Positive (FP) A False Positive (FP) is when the model incorrectly predicts the positive class. This is also known as a Type I error.

Example:

- Continuing with the medical test example:
- If a patient does not have the disease (actual negative) but the test incorrectly says the patient has the disease (predicted positive), this is a False Positive.

Scenario: The test says "You have the disease," but in reality, the patient does not have the disease.

False Negative (FN) A False Negative (FN) is when the model incorrectly predicts the negative class. This is also known as a Type II error.

Example:

- Again, using the medical test example:
- If a patient has the disease (actual positive) but the test incorrectly says the patient does not have the disease (predicted negative), this is a False Negative.

Scenario: The test says "You do not have the disease," but in reality, the patient does have the disease.

- **Accuracy** measures the overall correctness: $(TP + TN)/(TP + TN + FP + FN)$.
- **Precision** measures the correctness of positive predictions: $TP/(TP + FP)$.
- **Recall (Sensitivity)** measures the ability to identify positive cases: $TP/(TP + FN)$.
- **F1 Score** balances precision and recall: $2 \cdot (\text{Precision} \cdot \text{Recall})/(\text{Precision} + \text{Recall})$.
- **Specificity** measures the ability to identify negative cases: $TN/(TN + FP)$.

1. **Accuracy:**

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

   It measures the overall correctness of the model.
2. **Precision:**

$$\text{Precision} = \frac{TP}{TP + FP}$$

   It measures the correctness of positive predictions.
3. **Recall (Sensitivity or True Positive Rate):**

$$\text{Recall} = \frac{TP}{TP + FN}$$

   It measures the ability of the model to identify positive instances.

3. **Recall (Sensitivity or True Positive Rate):**

$$\text{Recall} = \frac{TP}{TP + FN}$$

   It measures the ability of the model to identify positive instances.
4. **F1 Score:**

$$\text{F1 Score} = \frac{2 \cdot (\text{Precision} \cdot \text{Recall})}{\text{Precision} + \text{Recall}}$$

   It is the harmonic mean of precision and recall, providing a balance between the two.
5. **Specificity (True Negative Rate):**

$$\text{Specificity} = \frac{TN}{TN + FP}$$

   It measures the ability of the model to identify negative instances.