NIKHIL MALVI

Text Block: Model Evaluation in Multi-Class Classification

In this code, we used the **Iris dataset**, a well-known dataset in machine learning, to train a **Logistic Regression** model for multi-class classification. Here's a breakdown of the process and results:

1. **Data Preprocessing**:
   - We loaded the dataset and selected two features, **'sepal length'** and **'sepal width'**, for simplicity.
   - The dataset was split into training and testing sets (80% training, 20% testing).
   - **Standardization** was performed on the features to ensure that all input features have a mean of 0 and a standard deviation of 1, which helps improve model performance.
2. **Model Training**:
   - We used a **Logistic Regression** model and trained it on the scaled training data (`X_train` and `y_train`).
   - The model was trained for a maximum of 200 iterations.
3. **Predictions and Performance Evaluation**:
   - After training the model, we made predictions on the test data (`X_test`).
   - We evaluated the model's performance using various metrics:
     - **Confusion Matrix**: This 3x3 matrix shows the number of true positives, false positives, true negatives, and false negatives for each of the three classes (Iris-setosa, Iris-versicolor, Iris-virginica). It helps us understand how well the model performs for each class.
     - **Accuracy**: This is the proportion of correct predictions to the total predictions, providing a simple measure of the model's performance.
     - **Precision**: Measures the model’s ability to correctly predict positive instances of each class, taking into account the imbalance in the class distribution.
     - **Recall**: Reflects how well the model identifies all actual instances of each class.
     - **F1 Score**: The harmonic mean of precision and recall, providing a balance between them.
4. **Evaluation Metrics Output**:
   - **Accuracy**: Indicates the overall performance of the model.
   - **Precision, Recall, and F1 Score**: These metrics are important when we care about both the correctness (precision) and completeness (recall) of the classification.

New Section

```
# Importing necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_
from sklearn.datasets import load_iris

# Load the inbuilt Iris dataset
iris = load_iris()

# Create DataFrame from the Iris dataset
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['target'] = iris.target  # Adding the target variable (species)

# Preview the dataset
print(df.head())

# Selecting features (X) and target variable (y)
X = df[['sepal length (cm)', 'sepal width (cm)']]  # Using first two features for simplicity
y = df['target']

# Split the dataset into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize Logistic Regression model
model = LogisticRegression(max_iter=200)

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model performance
cm = confusion_matrix(y_test, y_pred)

# Print confusion matrix
print(f"Confusion Matrix:\n{cm}")

# Compute accuracy, precision, recall, and F1 score
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
```

```
# Display the results
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")

```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0
Confusion Matrix:
[[10  0  0]
 [ 0  7  2]
 [ 0  1 10]]
Accuracy: 0.90
Precision: 0.90
Recall: 0.90
F1 Score: 0.90
```