```
# Common imports
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import SGDClassifier
from sklearn.utils import shuffle
import matplotlib.pyplot as plt

# Fetch the MNIST dataset from openml
mnist = fetch_openml('mnist_784', version=1)

# Store samples in X and labels in y
X, y = mnist["data"], mnist["target"]

# Convert labels to integers
y = y.astype(np.int32)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/datasets/_openml.py:1022: FutureWarning: The default value of `parser` will change t
  warn(
```

```
# Splitting the dataset: 70% training, 30% testing
train_size = int(0.7 * len(X))

x_train, x_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
```

```
last_training_sample_digit = list(y_train)[-1]
print(f"The last training sample is of digit {last_training_sample_digit}.")
```

```
The last training sample is of digit 6.
```

```
# Collect digit-6 and digit-9 images
six_nine_indices = (y_train == 6) | (y_train == 9)
x_train_69 = x_train[six_nine_indices]
y_train_69 = y_train[six_nine_indices]

# Set labels: 1 for digit 6, 0 for digit 9
y_train_69 = (y_train_69 == 6).astype(int)

# Shuffle the data
x_train_69, y_train_69 = shuffle(x_train_69, y_train_69, random_state=1729)

# Similarly, create x_test_69 and y_test_69
six_nine_indices_test = (y_test == 6) | (y_test == 9)
x_test_69 = x_test[six_nine_indices_test]
y_test_69 = y_test[six_nine_indices_test]

y_test_69 = (y_test_69 == 6).astype(int)
```

```
sum_train_labels = np.sum(y_train_69)
sum_test_labels = np.sum(y_test_69)

print(f"Sum of labels in y_train_69: {sum_train_labels}")
print(f"Sum of labels in y_test_69: {sum_test_labels}")
```

```
Sum of labels in y_train_69: 4855
Sum of labels in y_test_69: 2021
```

```
# Apply StandardScaler
scaler = StandardScaler()
x_train_69Tf = scaler.fit_transform(x_train_69)

# Mean and standard deviation of the zeroth sample and feature
mean_zeroth_sample = np.mean(x_train_69Tf[0])
mean_zeroth_feature = np.mean(x_train_69Tf[:, 0])
std_zeroth_sample = np.std(x_train_69Tf[0])
std_zeroth_feature = np.std(x_train_69Tf[:, 0])

results_tuple = (mean_zeroth_sample, mean_zeroth_feature, std_zeroth_sample, std_zeroth_feature)
print(results_tuple)
```

```
(0.1285224869548995, 0.0, 3.888657702544401, 0.0)
```

```
# Train Logistic Regression model using SGDClassifier
sgd_clf = SGDClassifier(loss='log_loss', max_iter=10, random_state=10, learning_rate='constant', eta0=0.01)
sgd_clf.fit(x_train_69Tf, y_train_69)
```

```
▾                              SGDClassifier
SGDClassifier(eta0=0.01, learning_rate='constant', loss='log_loss', max_iter=10,
              random_state=10)
```

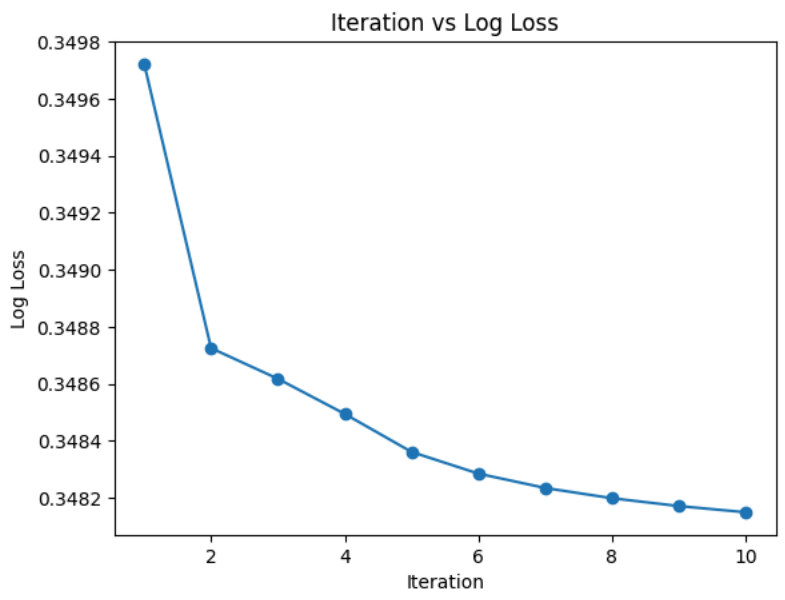
```
# Train Logistic Regression model using SGDClassifier
sgd_clf = SGDClassifier(loss='log_loss', max_iter=10, random_state=10, learning_rate='constant', eta0=0.01)

# Initialize the loss list to capture the loss after each iteration
losses = []
for _ in range(10):
    sgd_clf.partial_fit(x_train_69Tf, y_train_69, classes=np.unique(y_train_69))

    # Calculate the log loss
    y_pred_prob = sgd_clf.decision_function(x_train_69Tf)
    log_loss_value = np.mean(np.log(1 + np.exp(-y_train_69 * y_pred_prob)))

    # Append the log loss value for this iteration
    losses.append(log_loss_value)

# Plotting the iteration vs loss curve
import matplotlib.pyplot as plt
plt.plot(range(1, 11), losses, marker='o')
plt.xlabel('Iteration')
plt.ylabel('Log Loss')
plt.title('Iteration vs Log Loss')
plt.show()
```



Start coding or generate with AI.