CYTOAUTOCLUSTER

- **Data split:**
  1. The data is split into labeled (df_labeled) and unlabeled (df_unlabeled) subsets based on the presence or absence of values in the 'label' column. The feature matrix (x_labeled, x_unlabeled) and target variable (y_labeled, y_unlabeled) are then separated.
  2. Separate scalers (scaler_labeled and scaler_unlabeled) are applied to labeled and unlabeled data, respectively.
  3. The labeled data is split into training and test sets using train_test_split from sklearn.model_selection, with an 80-20 split.
- **Logistic Regression & XGBoost:**
  1. A logistic regression model is trained on the standardized labeled data (x_train and y_train). The cross-entropy loss (log loss) for this model was 0.0119073425627856, indicating a preliminary level of model performance prior to encoder-based adjustments.
  2. An XGBoost classifier (XGBClassifier) is trained on adjusted labels (y_train_adjusted), and probability predictions are generated for the test set. The log loss for the XGBoost model was 0.005201324112962294, offering an initial baseline for comparison before applying the encoder function.
- **Self-Supervised Model:**
  1. A binary mask is generated using np.random.binomial. This mask determines which elements in the data will be masked (corrupted). For instance, with masking_probability = 0.5, each element has a 50% chance of being masked.
  2. The corruption function uses the binary mask to selectively corrupt x_unlab by mixing it with a shuffled version of the dataset (df_shuffle).
  3. A neural network model is built with an input layer and one hidden dense layer (Dense(dimension, activation='relu')). It has two outputs:
     - output1 for mask estimation (trained with binary_crossentropy loss),
     - output2 for feature estimation (trained with mean_squared_error loss).
  4. The encoder_model.save(encoder_path) command saves the trained encoder model to the specified file path, allowing for easy model reuse or deployment.
  5. Logistic Regression and XGBoost models are trained on the encoded data. Logarithmic loss (log loss) is computed for both models' predictions, where Logistic Regression yielded a log loss of 0.1935, while XGBoost achieved a lower log loss of 0.0783, indicating better performance.