CytoAutoCluster

Project Overview:

The **CytoAutoCluster** project is designed to cluster cytometry data using a semi-supervised machine learning approach. The goal is to utilize both labeled and unlabeled data to improve the accuracy of cell population identification. The project involves data preprocessing, feature extraction using an encoder, model training, and interactive demonstration via a Gradio interface.

Dataset:

- Source: 'Levine 32 dim' dataset containing cell marker measurements.
- Features: Includes markers like CD45, CD133, CD4, CD8, etc.
- Label Info: Presence of NaN values indicating unlabeled data points.

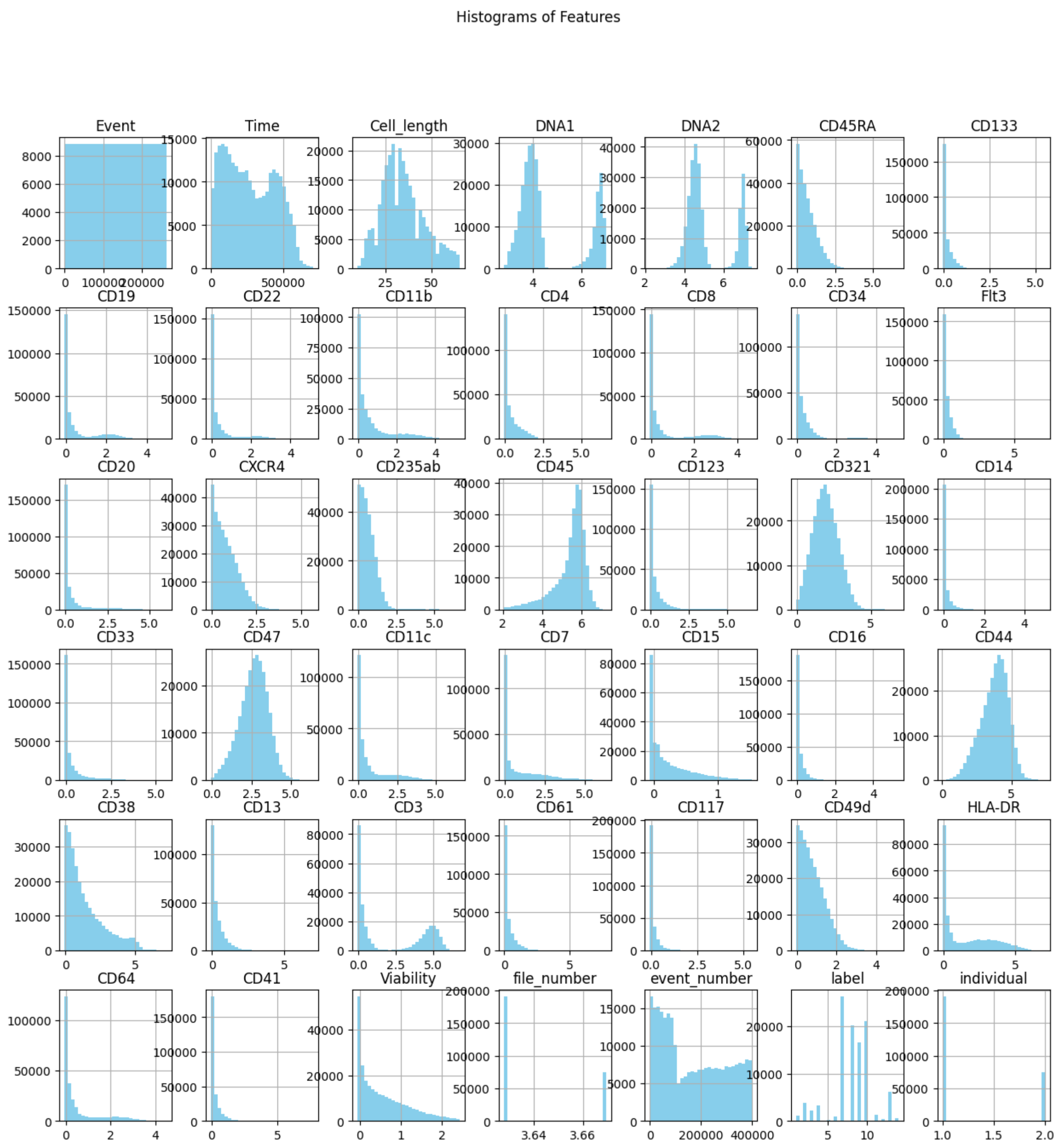
Data Loading and Exploration:

Objective: Load the dataset and conduct initial exploration to understand its structure.

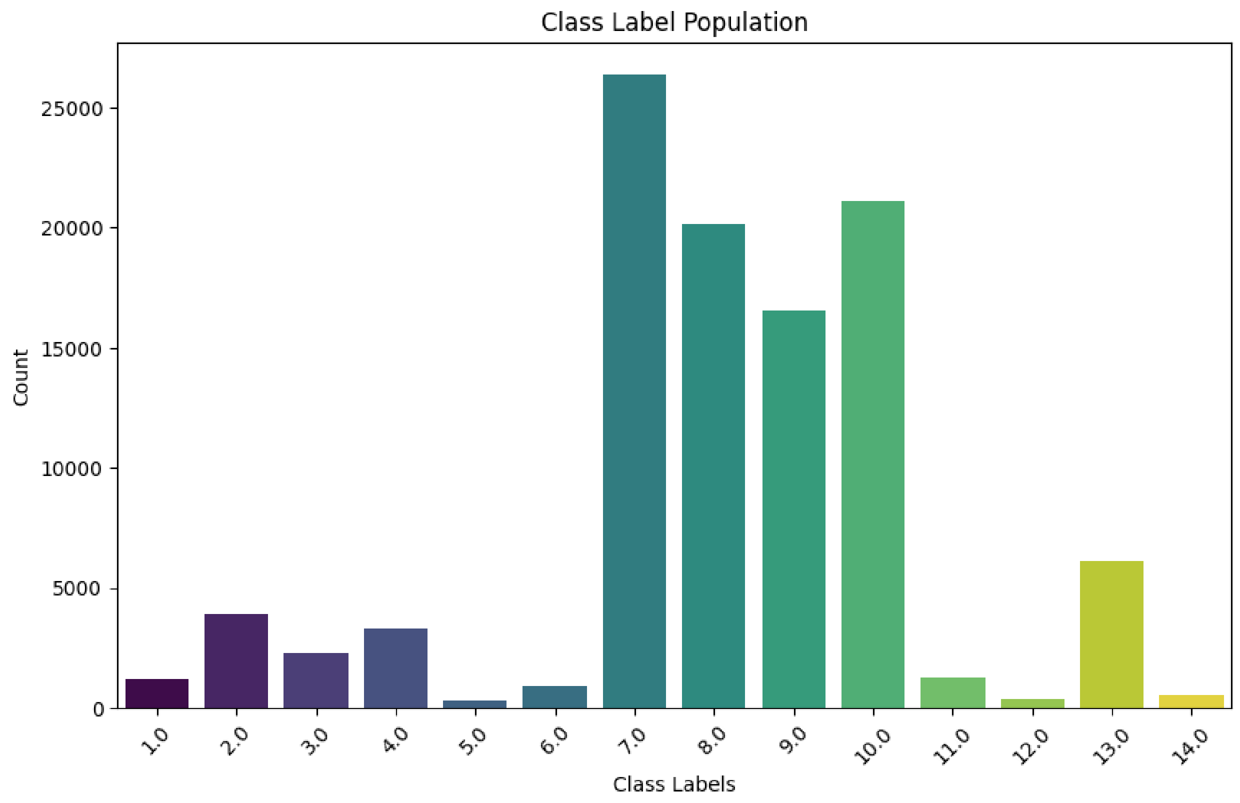
Explanation:

- The dataset, known as the Levine 32 dim dataset, is loaded using Python's pandas library. This dataset includes multiple features related to cell markers and other attributes.
- Initial exploration involves checking the basic information, such as data types, null values, and the shape of the dataset, to ensure that the data is suitable for preprocessing.
- Visualizations like heatmaps are used to detect missing values, while distribution plots help identify the spread and variability of the data.

Histogram:


Histograms of Features
Event
Time
Cell_length
DNA1
DNA2
CD45RA
CD133
CD19
CD22
CD11b
CD4
CD8
CD34
Flt3
CD20
CXCR4
CD235ab
CD45
CD123
CD321
CD14
CD33
CD47
CD11c
CD7
CD15
CD16
CD44
CD38
CD13
CD3
CD61
CD117
CD49d
HLA-DR
CD64
CD41
Viability
file_number
event_number
label
individual

Class Label Distribution:



<u>Data Preprocessing:</u>

Objective: Prepare the dataset by cleaning and scaling it for further analysis.

Explanation:

- **Handling Missing Values**: Missing data points are handled by either dropping or imputing values. Common strategies include filling missing entries with the mean or median of respective columns.
- **Feature Scaling**: The data is normalized using a scaling technique to bring all features into a comparable range. This ensures that algorithms work efficiently, particularly those sensitive to feature magnitudes like the encoder.
- **Splitting into Labeled and Unlabeled Data**: The dataset is divided into labeled and unlabeled subsets based on the availability of labels. This step is crucial for training models in a semi-supervised manner.

t-SNE Visualization:

Objective: Visualize high-dimensional data to gain insights before and after feature extraction.

Explanation:

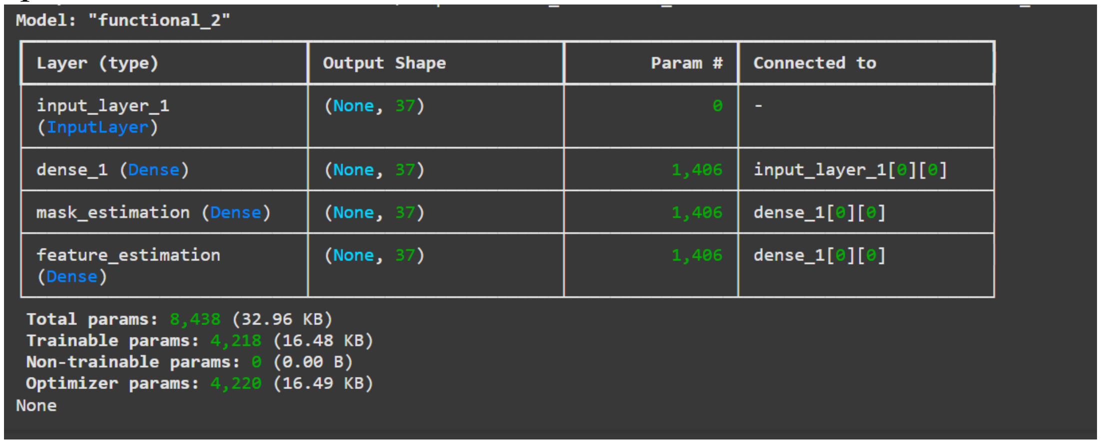
- **t-SNE (t-distributed Stochastic Neighbor Embedding)** is applied to reduce the dimensionality of the dataset to 2D for visualization purposes. This helps in understanding how data points are distributed and whether there are visible clusters.
- The visualization helps to observe potential groupings in the data and the effect of feature transformation done by the encoder.

Encoder Model:

Objective: Create a neural network-based encoder to extract meaningful features from the input data.

Explanation:

- An encoder is used to reduce the dimensionality of the input data and create a latent space that captures the essential information in a compressed form. This process is useful for learning representations that are beneficial for downstream tasks such as clustering.
- The encoder model is compiled with an optimizer (e.g., Adam) and trained using a suitable loss function (e.g., mean squared error) for a set number of epochs.



```
Model: "functional_2"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)              ┃ Output Shape           ┃        Param # ┃ Connected to            ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━┩
│ input_layer_1             │ (None, 37)             │              0 │ -                       │
│ (InputLayer)              │                        │                │                         │
├───────────────────────────┼────────────────────────┼────────────────┼─────────────────────────┤
│ dense_1 (Dense)           │ (None, 37)             │          1,406 │ input_layer_1[0][0]     │
├───────────────────────────┼────────────────────────┼────────────────┼─────────────────────────┤
│ mask_estimation (Dense)   │ (None, 37)             │          1,406 │ dense_1[0][0]           │
├───────────────────────────┼────────────────────────┼────────────────┼─────────────────────────┤
│ feature_estimation        │ (None, 37)             │          1,406 │ dense_1[0][0]           │
│ (Dense)                   │                        │                │                         │
└───────────────────────────┴────────────────────────┴────────────────┴─────────────────────────┘
 Total params: 8,438 (32.96 KB)
 Trainable params: 4,218 (16.48 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 4,220 (16.49 KB)
None
```

Model Training (Logistic Regression, XGBoost, Semi-Supervised Learning):

Objective: Train models on encoded features to evaluate their effectiveness in classifying and clustering data.

Explanation:

- **Logistic Regression**: Used as a baseline to understand the performance of the encoded features. It is a simple and interpretable model for binary or multiclass classification. Achieved an log loss of **0.11511803835329242** using encoded features as input.
- **XGBoost**: A more complex, gradient-boosted tree algorithm that often performs well on structured data. It is trained using the encoder's output as features to achieve higher predictive accuracy. Achieved a log loss of **0.07188946205290861**, validating the quality of latent features.
- **Semi-Supervised Learning**: This method is applied to leverage both labeled and unlabeled data. The encoder's features are combined with a semi-supervised loss function that uses a clustering algorithm to incorporate unlabeled data into the training process. This helps improve the model's ability to generalize and identify patterns in unseen data. [**Accuracy: 0.910543744301003 ,AUROC: 0.9858937931143635**],

Unlabeled Predictions:

Objective: Predict the labels for the unlabeled portion of the data.

Explanation:

- The trained encoder is applied to the unlabeled data to extract latent features. These features are then processed by a clustering algorithm, such as K-Means, to assign cluster labels.
- This step helps group similar data points together and provides insights into potential cell population clusters that have not been labeled previously.

<u>Gradio Interface:</u>

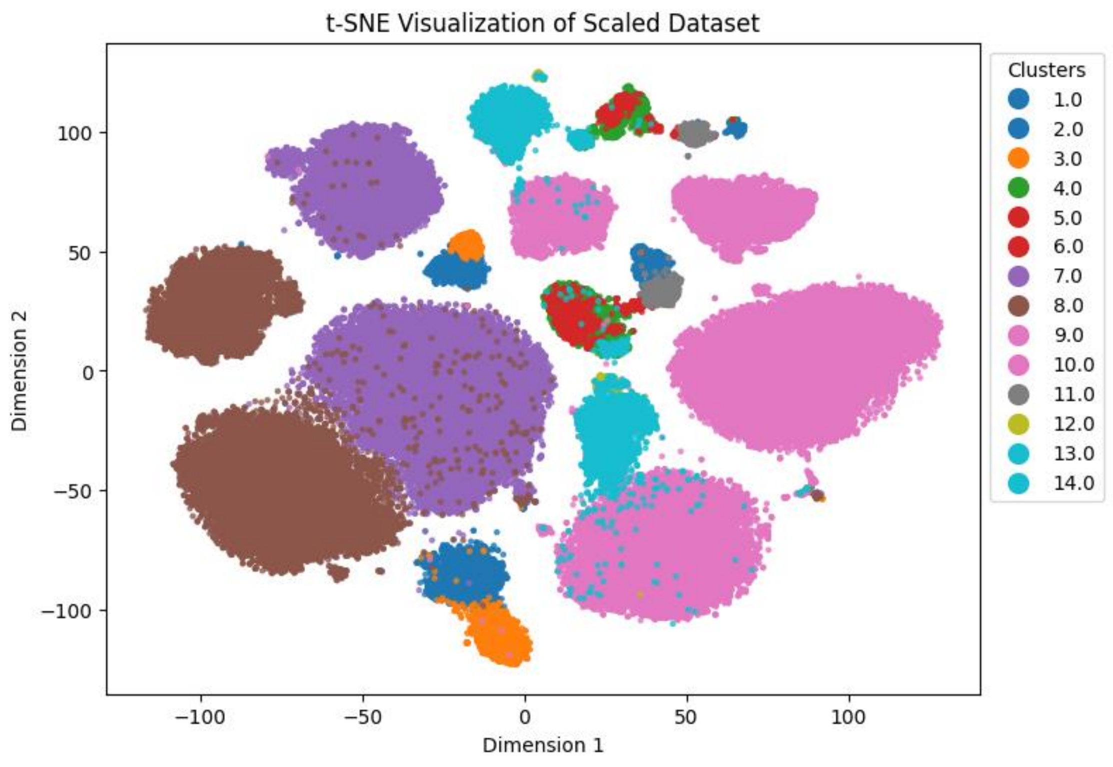
Objective: Create an interactive tool to demonstrate the model's capabilities and allow users to make predictions.

Explanation:

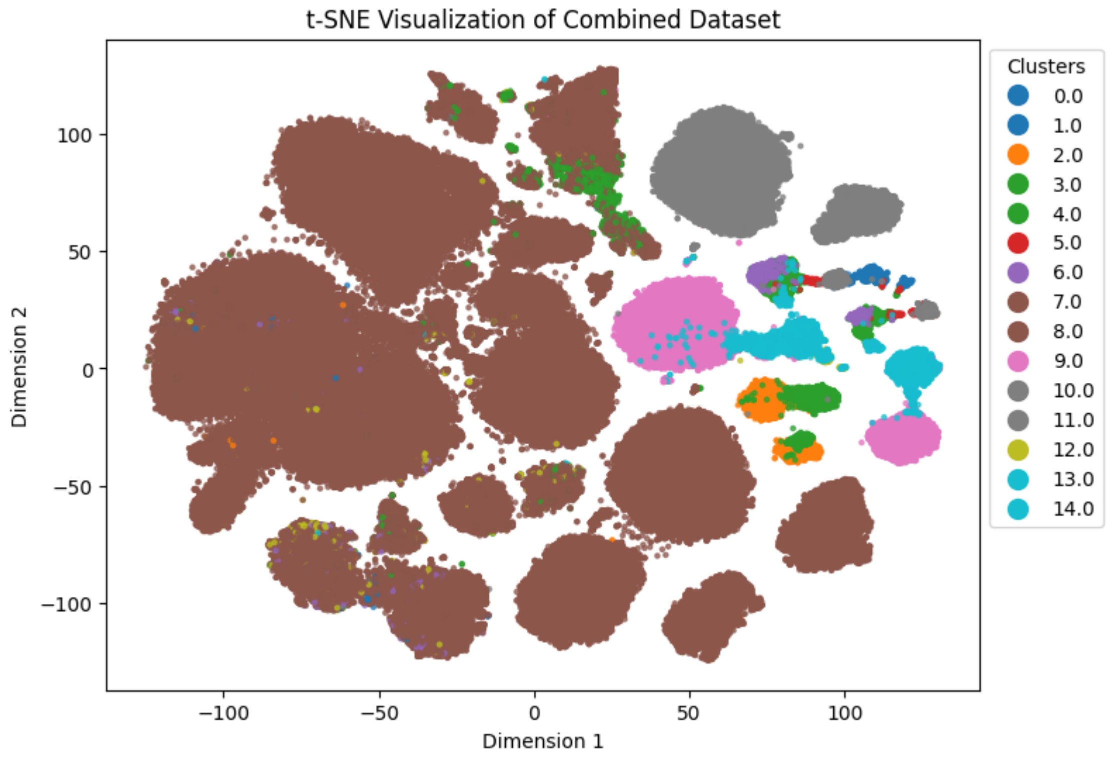
- A Gradio interface is set up to provide a user-friendly way to interact with the trained model. Users can upload data points or manually input information.
- The interface processes the input data, runs it through the encoder, and displays the predictions and clustering results. This makes the tool accessible for demonstration purposes and practical use.

<u>Visualizations and Outputs:</u>

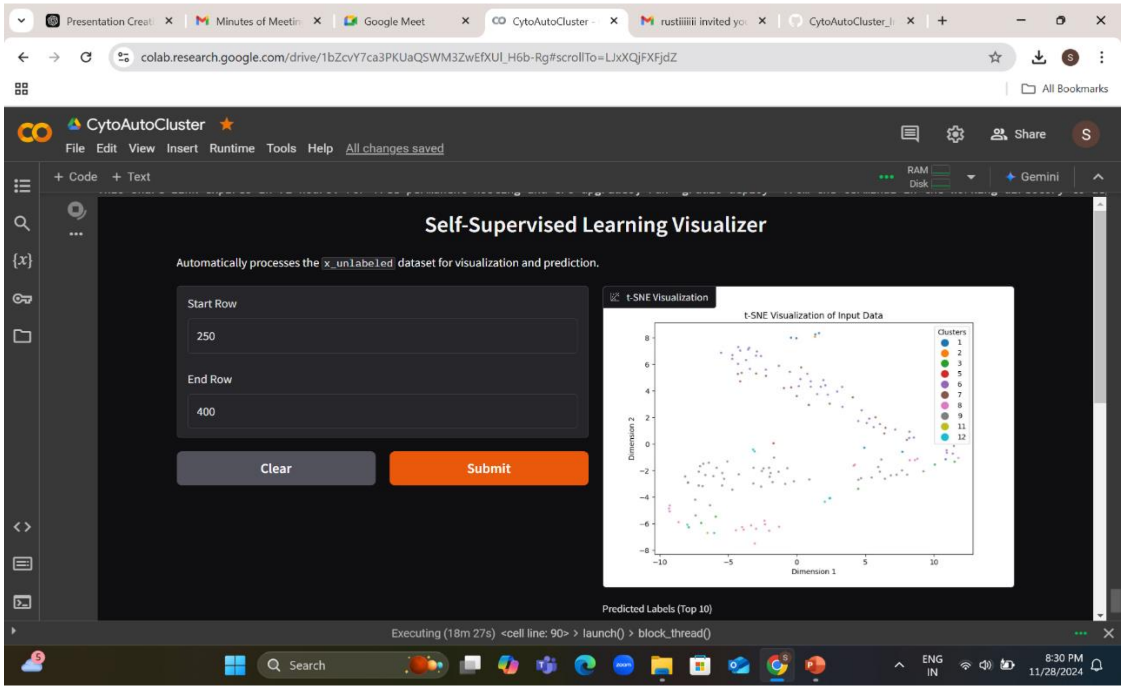
t-SNEPlots-Before:



After:



Gradio Interface:


colab.research.google.com/drive/1bZcvY7ca3PKUaQSWM3ZwEfXUl_H6b-Rg#scrollTo=LJxXQjFXFjdZ
CytoAutoCluster
File
Edit
View
Insert
Runtime
Tools
Help
All changes saved
Share
+ Code
+ Text
Gemini
t-SNE Visualization
t-SNE Visualization of Input Data
Dimension 2
Dimension 1
Clusters
Clear
Submit
Executing (18m 27s) <cell line: 90> > launch() > block_thread()
8:30 PM
11/28/2024

Self-Supervised Learning Visualizer

Automatically processes the `x_unlabeled` dataset for visualization and prediction.

Start Row: 250

End Row: 400

Predicted Labels (Top 10)

<u>Conclusion:</u>

The **CytoAutoCluster** project demonstrates how semi-supervised learning techniques, combined with feature extraction using an encoder, can enhance the analysis and clustering of cytometry data. The interactive Gradio interface further aids in showcasing the model's capabilities and making it accessible for users.