**CytoAutoCluster: A Semi-Supervised Approach to Cell Classification**

**Introduction**

In data-driven environments, assessing the features of objects and classifying them into distinct groups is critical for various applications. CytoAutoCluster focuses on clustering cells into different groups based on their features. The objective is purely computational, aiming to develop an efficient clustering system using semi-supervised learning techniques.

**Problem Overview**

The challenge of clustering cytometry data arises primarily due to the following reasons:

* **Lack of Labels**: Most of the available cytometry data is unlabeled, making it difficult to directly classify the cells into groups.
* **Data Collection Difficulty**: It is challenging to obtain sufficient data that explicitly defines which group a cell belongs to.
* **Complex Features**: High-dimensional data from cytometry requires advanced algorithms to capture and process relevant features for classification.

**Objectives**

The aim of the CytoAutoCluster project is to design a clustering algorithm capable of classifying cells into distinct groups based on their features using a semi-supervised learning approach. This involves:

* Learning from a limited amount of labeled data to identify the most important features.
* Applying these features to classify a large volume of unlabeled data.
* Developing a computational system that works efficiently.

**Approach: Semi-Supervised Learning**

Since obtaining labeled data is resource-intensive, semi-supervised learning provides an optimal approach by leveraging both labeled and unlabeled data. The strategy includes:

* **Labeled Data**: A small fraction of the dataset has labels indicating the groupings of cells based on their features.
* **Unlabeled Data**: The majority of the dataset remains unlabeled, and the model must infer groupings from the labeled data.

Through this method, the system can learn significant patterns from the labeled data and apply these to the unlabeled portion for efficient classification.

**Dataset Selection**

Various datasets were considered for this project, but the following were key:

1. **Rejected Dataset**:  
   [CellCnn Learning Disease-Associated Cell Subsets](https://www.kaggle.com/datasets/kmader/cellcnn-learning-disease-associated-cell-subsets)  
   Reason for Rejection: This dataset was rejected as it pertains to medical applications, which do not align with the project's goal of purely computational classification of cells.
2. **Accepted Dataset**:  
   [Levine CytOF 32 Dimensional Data](https://www.kaggle.com/datasets/sibongo/levinecytof32dim)  
   Reason for Acceptance: This dataset has more than 60% unlabeled data and includes 40 features, making it suitable for semi-supervised learning where the majority of data is unlabeled, and feature-rich data is required.

**Code Implementation for CytoAutoCluster**

**Loading the Dataset:**

The project begins with loading the Levine CytOF 32 Dimensional Data dataset into the environment. This dataset is stored in CSV format and is uploaded to Google Colab for analysis.

**Exploring the Data:**

After loading the dataset, initial exploration is conducted to understand its structure. This includes examining the first few rows and determining the overall shape of the data, which helps in identifying the number of features and instances available for analysis.

**Checking for Null Values:**

Given the emphasis on utilizing semi-supervised learning, assessing the presence of null values in the dataset is crucial. A thorough analysis is performed to identify which columns contain null values and the specific rows that are affected. This step is essential for understanding the completeness of the data.

**Comparison between Null and Non-Null Values:**

To gain insights into the distribution of null and non-null values across the dataset, visualizations are created. These graphical representations help to identify which features are impacted by missing values and the extent of this impact, informing subsequent preprocessing decisions.

**Data Analysis Techniques:**

Histograms: Distribution of values for each feature.

Correlation Matrix: Identifies relationships between features, assessing feature importance**.**

**Correlation Matrix:**

A correlation matrix is constructed to analyze relationships between features. This matrix helps identify:

* + **Positive Correlations**: Features that increase or decrease together.
  + **Negative Correlations**: Features where one increases as the other decreases.
  + **Independent Features**: Features with minimal correlation, often useful for dimensionality reduction.

Understanding correlations aids in feature selection and helps to reduce redundancy by removing highly correlated features, which simplifies the model.

**Range of Feature Values :**

Each feature's value range is calculated, providing insights into the scale and variability of each attribute.

**Box Plot :**

Box plots are used to visually examine the distribution of each feature:

* + **Median and Quartiles**: Centered around the median, it shows data spread and skewness.
  + **Outliers**: Box plots highlight outliers beyond the interquartile range, guiding further decisions about feature engineering or data cleaning.

**Skewness and Kurtosis Analysis:**

* + **Skewness**: Measures the asymmetry of feature distributions:
    - Positive Skew: Right-skewed, with a longer tail on the right.
    - Negative Skew: Left-skewed, with a longer tail on the left.
  + **Kurtosis:** Measures the "tailedness" or peakedness of the distribution:
    - High Kurtosis: Leptokurtic, with sharper peaks and heavier tails.
    - Low Kurtosis: Platykurtic, with flatter distributions.

This analysis helps identify features requiring transformation to improve normality and stabilize variance, often beneficial for machine learning algorithms.

**Dimensionality Reduction Techniques:**

1. **Principal Component Analysis (PCA)**

PCA is used to reduce dimensionality by retaining essential features while preserving variance. It simplifies the data for visualization and analysis, especially with high-dimensional data like this project’s dataset.

* Variance Explained: Calculated to assess the significance of each component, which helps retain 95%+ of the variance using fewer dimensions.
* Transformation to Lower Dimensions: This simplification reduces computational costs and overfitting risks, enhancing model efficiency.

1. **t-Distributed Stochastic Neighbor Embedding (t-SNE)**

t-SNE is primarily a visualization tool that maps high-dimensional data into 2D or 3D space to reveal underlying clusters and patterns:

* + Preserving Local Similarities: Unlike PCA, t-SNE captures non-linear relationships by preserving local structures rather than global.
  + Cluster Visualization: Highlights potential groupings within data, particularly effective in semi-supervised learning for better understanding of unlabeled data structure.

**Autoencoders**

Autoencoders are neural networks used for feature extraction in semi-supervised learning, capable of handling both labeled and unlabeled data.

Autoencoder Mechanism**:**

* **Encoding**: Compresses input data into latent features.
* **Latent Space**: Stores reduced-dimensional data.
* **Decoding**: Reconstructs input data from latent space, minimizing dimensionality while preserving key information.

Applications and Data Requirements:

1. **Input Dimensions**: Autoencoders for tabular data input high-dimensional cytometry data, adapting it for reduced-dimension representations.
2. **Corruption Techniques**: Artificial noise or dropout can be applied to input data to improve feature extraction.

**Consistency Regularization**:

It is a semi-supervised learning technique that encourages the model to produce consistent predictions when small, irrelevant changes (like noise or augmentations) are made to the input data. The idea is that, regardless of minor alterations, the model should still classify the input in the same way. This helps the model generalize better and reduces reliance on labeled data, as it can leverage unlabeled data more effectively.

**Entropy Minimization**:

It aims to improve model confidence by minimizing the entropy of predictions, pushing outputs to be more confident (i.e., closer to 0 or 1 probabilities for class labels) rather than uncertain. In semi-supervised learning, entropy minimization is often used alongside consistency regularization to encourage the model to confidently classify unlabeled examples, improving performance when labeled data is scarce.

**Binary Mask**

A binary mask is an array of 0s and 1s used to filter or select specific elements in datasets or images. It highlights areas of interest, allowing focused analysis by marking relevant pixels or data points.

**Random Shuffling of Data**

Random shuffling of data involves rearranging the elements in a dataset randomly to eliminate order bias. This technique ensures that both training and testing datasets represent the overall data distribution accurately.

**Corrupted DataFrame**

A corrupted DataFrame contains missing, altered, or invalid values that compromise its integrity and usability. Handling such corruption often involves data cleaning techniques to restore the dataset for reliable analysis.

**Classification of Labeled and Unlabeled Data**

Labeled data consists of input-output pairs where each input has an associated output or label, making it suitable for supervised learning tasks. In contrast, unlabeled data only contains inputs without corresponding labels, requiring methods like semi-supervised or unsupervised learning. Combining labeled and unlabeled data is useful in scenarios where labeling is expensive or time-consuming. The labeled data provides guidance to learn patterns, while the unlabeled data enhances the model's ability to generalize. Effective classification techniques balance the use of both data types for robust learning.

**Logistic Regression**

Logistic regression is a statistical model used for binary or multi-class classification tasks. It predicts probabilities using a sigmoid function, converting linear combinations of inputs into values between 0 and 1. The model is trained using a loss function, typically cross-entropy, to minimize the difference between predicted and actual labels. Logistic regression is computationally efficient and works well for linearly separable data. Despite its simplicity, it forms the foundation for many advanced classification models.

**XGBoost**

XGBoost (Extreme Gradient Boosting) is an optimized machine learning algorithm based on decision trees, designed for both regression and classification tasks. It uses boosting, where weak learners are sequentially improved by correcting their predecessor's errors. XGBoost incorporates regularization techniques, handling overfitting effectively while being computationally efficient. It is widely known for its superior performance in competitive machine learning challenges. The algorithm supports missing value handling and works well with large datasets.

**Logistic Regression and XGBoost Loss**

Both logistic regression and XGBoost rely on loss functions to evaluate their predictions. Logistic regression often uses binary cross-entropy or log loss, measuring the divergence between predicted probabilities and actual labels. XGBoost utilizes a custom loss function, like log loss for classification, but augments it with gradient boosting to minimize errors iteratively. Both methods aim to optimize their respective loss functions during training, enabling the model to make accurate predictions on unseen data.

**Encoder Model**

An encoder model is typically used in neural networks for representation learning. It compresses input data into a latent space, reducing dimensionality while retaining meaningful features. The encoder is often the first component in architectures like autoencoders, where it transforms raw data into encoded vectors. These vectors serve as compact representations, useful for downstream tasks like classification or clustering. Encoder models are key in semi-supervised learning, enabling feature extraction from unlabeled data.

**Semi-Supervised Learning**

Semi-supervised learning combines labeled and unlabeled data to train models, leveraging the abundance of unlabeled data to improve performance. The labeled data guides the learning process, while the unlabeled data helps in discovering patterns and structures in the input space. This approach is beneficial in domains like healthcare or image recognition, where acquiring labels is expensive or difficult. Techniques such as self-training, consistency regularization, and pseudo-labeling are commonly used. Semi-supervised learning balances efficiency and accuracy by utilizing limited labeled data effectively.

**Performance Metrics: Accuracy and AUROC**

Accuracy measures the proportion of correctly predicted instances to the total instances, providing a straightforward evaluation of model performance. However, it can be misleading for imbalanced datasets. The Area Under the Receiver Operating Characteristic Curve (AUROC) evaluates a model's ability to distinguish between classes across various thresholds. A higher AUROC indicates better discriminatory power, making it ideal for tasks with imbalanced data. Both metrics provide complementary insights into model performance.

**t-SNE After Encoder**

t-SNE (t-Distributed Stochastic Neighbor Embedding) is a dimensionality reduction technique used to visualize high-dimensional data. After applying an encoder model, t-SNE can be used to map the encoded data to two or three dimensions for visualization. This helps in identifying clusters, patterns, or anomalies in the latent space. t-SNE works by minimizing the divergence between pairwise similarity distributions in high and low dimensions. It is especially useful for understanding the structure of encoded data in semi-supervised learning.

**Gradio**

Gradio is a Python library for creating interactive web-based interfaces for machine learning models. It allows users to test models in real-time by providing inputs through a user-friendly interface and visualizing predictions instantly. Gradio supports various input types, including text, images, and audio, making it versatile for multiple use cases. It is widely used for prototyping, sharing models, and gathering user feedback. Gradio's simplicity and integration capabilities make it popular for demonstrating machine learning applications.