# CYTOAUTOCLUSTER

## **INTRODUCTION**:

CytoAutoCluster seeks to incorporate semi-supervised learning methods into cytometry workflows. By leveraging both labeled and unlabeled data, this project is designed to create a powerful clustering algorithm capable of adapting to the underlying patterns in cytometric data. This approach aims to improve the accuracy of cell classification while minimizing the need for large labeled datasets, which are often resource-intensive and time-consuming to generate

**DAY-1 | 07-10-2024 :**

- Sourced and downloaded a relevant semi-supervised cytometry mass dataset from resources such as Kaggle, Google Scholar, and Papers with Code.

**DAY-2 | 08-10-2024 :**

- The dataset was rejected as it contained fully labeled data, whereas semi-supervised learning requires a portion of unlabeled data.
- Presented the dataset to my mentor and received feedback to ensure the dataset includes more than 60% unlabeled data.
- The mentor advised that the final dataset should consist of over 40 columns and be completely in tabular format.

**DAY-3 | 09-10-2024 :**

- Acquired a foundational understanding of the components of cytometry using a sample dataset (Levine_32dim_notransform.csv).
- Learned basic Git commands for working with the project's GitHub repository.
- Understood collaborative repository management processes, including pushing, merging, and committing code changes.

**DAY-4 | 10-10-2024 :**

- Finalized the dataset: Levine_32dim_notransform.csv.
- Set up the Python environment, uploaded the dataset, and created a DataFrame for further analysis.
- Imported the necessary Python packages for data analysis.
- Uploaded the dataset into the environment.

**DAY-5 | 11-10-2024 :**

- Removed unnecessary columns (Cell_length, file_number, and event_number) from the dataset to focus on relevant features.
- Calculated the count of null and non-null values in each column using data.isnull().sum() and data.notnull().sum().
- Created a bar plot to compare the number of null and non-null values in each column, using a DataFrame to organize the counts.
- Visualized the plot to gain insights into the presence of missing data, with labeled axes and a legend for clarity.

**DAY-6 | 14-10-2024 :**

- Performed additional EDA techniques, including:
- Analyzing the range of each feature to understand their spread and distribution.
- Generating a correlation matrix to identify relationships between different features.
- Examining the class label distribution to gain insights into the balance of labeled data.
- Calculated the range of each feature in the dataset using the formula:

- Range = Maximum Value − Minimum Value

**DAY-7 | 15-10-2024 :**

Performed EDA Techniques more like Kurtosis, Skewness, and Pair Plot

**SKEWNESS**

- Calculated the skewness for each feature in the dataset using the scipy.stats library to measure asymmetry in data distribution.
- Defined a function to categorize skewness as:
  - "Right-skewed" for values greater than 0.5,
  - "Left-skewed" for values less than -0.5, and
  - "Approximately symmetrical" for values between -0.5 and 0.5.
- Applied the categorization function to the calculated skewness values and displayed the results in a DataFrame showing each feature's skewness and corresponding category.

**KURTOSIS**

- Here's a refined version of your description for the kurtosis analysis:
- Calculated the kurtosis for each feature in the dataset using the scipy.stats library to measure the "tailedness" of data distribution.
- Defined a function to categorize kurtosis as:
  - **Leptokurtic (Heavy Tails)** for values greater than 3,
  - **Platykurtic (Light Tails)** for values less than 3, and
  - **Mesokurtic (Normal Tails)** for values equal to 3.
- Applied the categorization function to the calculated kurtosis values and displayed the results in a DataFrame showing each feature's kurtosis and corresponding category.

**PAIRPLOT**

- Generated a pair plot to visualize the relationship between two selected protein features (CD19 and CD22), with different colors representing the label values.
- In the plot, darker shades indicate higher label values, while lighter shades indicate lower label values, highlighting the distribution of data points based on their labels.

**DAY-8 | 16-10-2024 :**

Performed individual plots for each feature to visualize kurtosis and skewness, allowing for a detailed examination of the distribution and "tailedness" of each feature in the dataset.

- **Plotted Kurtosis and Skewness for Individual Features**
  Created individual plots to analyze the kurtosis and skewness of each feature in the dataset.
- **Reviewed Relevant Research Paper**
  Read the research paper related to the project: Research Paper.
- **Mentor Discussion on Research Paper**
  Had a brief discussion with the mentor to gain insights into the key aspects of the research paper.

**DAY-9 | 17-10-2024 :**

- **Challenges with t-SNE and PCA**
  Encountered issues with obtaining the expected results after performing t-SNE and PCA on the dataset.
- **Identified Data Preprocessing Needs**
  Discovered that standardizing the features (scaling them between 0 and 1) and removing columns with null values are essential for better results.
- **Performed t-SNE on MNIST Dataset**
  To better understand how the t-SNE model works, I applied it to the MNIST dataset.
- **Standardizing Values**
  Standardizing data involves scaling features so that they have a mean of 0 and a standard deviation of 1, ensuring more reliable results in machine learning models.
- **MNIST Dataset**
  The MNIST dataset consists of 70,000 grayscale images of handwritten digits (0–9), widely used for training and evaluating image processing systems.

**DAY-10 | 18-10-2024 :**

**Learned About t-SNE and PCA**

Acquired a foundational understanding of how t-SNE and PCA work and their applications in dimensionality reduction and data visualization.

**DAY-11 | 21-10-2024 :**
As per my mentor's guidance, the following columns were excluded from the analysis to ensure only relevant features are considered:

```
# Define the columns to exclude
excluded_columns = ['Event', 'Time', 'Cell_length', 'file_number', 'event_number', 'label', 'individual']
```

**Conducted PCA and t-SNE Technique**

**PCA(Principal Component Analysis)**
Principal Component Analysis (PCA) is a technique for reducing data dimensionality by transforming it into a set of new, uncorrelated variables called principal components. These components capture the data's significant variance with fewer dimensions, which assists in visualizing complex datasets, optimizing algorithm performance, and reducing noise.

**t-SNE (t-Distributed Stochastic Neighbor Embedding)**
t-SNE is a visualization technique that maps high-dimensional data into a low-dimensional space, typically 2D or 3D, while preserving the local structure of the data. It achieves this by minimizing differences between probability distributions of distances in the high and low dimensions, allowing similar points in the original space to remain close in the visualization. This results in clusters that accurately reflect the original data relationships.

**DAY-12 | 22-10-2024 :**

- Performed Operations on PCA
  Carried out various operations on Principal Component Analysis (PCA), including the calculation of standard deviation, proportion of variance, and cumulative proportion.
- Created 3D Visualization of PCA
  Generated a 3-dimensional plot to visualize the results of the PCA technique.
- Studied Autoencoders
  Gained an overview of Autoencoders, including their application in various domains and learned about some popular models built using autoencoders.

**DAY-13 | 23-10-2024 :**

Studied the research paper on "Deep Semi-Supervised Learning" (https://arxiv.org/pdf/2006.05278).

**DAY-14 | 25-10-2024 :**

Gained a basic understanding of semi-supervised learning concepts from the research paper.

**DAY-15 | 28-10-2024 :**

Learned About Consistency Regularization and Entropy Minimization in Semi-Supervised Learning
Gained insights into key techniques used in semi-supervised learning:

- **Consistency Regularization**
  Consistency regularization ensures that the model's predictions stay consistent even when the input data undergoes slight perturbations or augmentations.
- **Entropy Minimization**
  Entropy minimization reduces uncertainty in the model's predictions, especially for unlabeled data, by encouraging the model to make more confident predictions.
- **Learned Binary Masking Techniques**
  Gained knowledge on how to apply binary masking to a set of rows in a dataframe.

**DAY-16 | 29-10-2024 :**

- **Performed Binary Masking on a Set of Rows**
  Applied binary masking to a set of rows in the dataset.
- **Binary Masking**
  Binary masking involves using a binary mask to selectively focus on certain parts of the input data. This technique allows the model to effectively learn from both labeled and unlabeled data.

By highlighting relevant features and ignoring noise, binary masking enhances the model's performance, especially in tasks with limited labeled data.

- **Reviewed Abstract of VIME (Value Imputation and Mask Estimation)**
  Read the abstract of the research paper "VIME: Extending the Success of Self- and Semi-supervised Learning to the Tabular Domain," which focuses on methods for value imputation and mask estimation in tabular data.

**DAY-17 | 30-10-2024 :**

- **Performed Binary Masking with Column Shuffling**
  Applied binary masking by randomly shuffling the values in the columns of the dataset.
- **Corruption of Data**
  Data corruption refers to the alteration or distortion of values in a dataset, which can occur either intentionally (for testing) or unintentionally (due to errors during data collection, processing, or storage).
- **Created Corrupted Data**
  Performed data corruption by creating a new dataframe called "corrupted_data_frame" using the equation:
- x * (1−m) + x_shuffled * m = x_corrupted
- Where:
  - x is the original data frame.
  - m is the binary mask matrix.
  - x_shuffled is shuffled data frame.

**DAY-18 | 01-11-2024:**

- **Performed Binary Masking on Dataset**
  Applied binary masking to the dataset.
- **Created Corrupted Data Frame**
  Created a corrupted data frame using a binary mask matrix, where a value of 1 indicates that the corresponding entry is corrupted, and a value of 0 indicates the entry is not corrupted.
- **Categorized Data into Labeled and Unlabeled Sets**
  Created four variables to split the dataset into labeled and unlabeled categories:
- **x_labeled:** Rows of the feature columns (excluding the target column) with non-null values.
- **y_labeled**: Rows of the label (target) column with non-null values.
- **x_unlabeled:** Rows of the feature columns with null values.
- **y_unlabeled:** Rows of the label (target) column with null values.

**DAY-19 | 04-11-2024:**

- **Performed Data Split on x_labeled, y_labeled, and y_unlabeled**
  Split the data into training and testing sets, allocating 70% of the data for training and 30% for testing.

**Train-Test Split:** The train-test split technique divides a dataset into two parts: a training set, used to train the model, and a test set, used to evaluate the model's performance on unseen data. This helps prevent overfitting and ensures that the model generalizes well to new data by providing an unbiased evaluation of its predictive capabilities.

**DAY-20 | 05-11-2024:**

- **Implemented Logistic Regression and XGBoost on Training Data Split**
  Applied both logistic regression and XGBoost algorithms to the training subset of the dataset, and calculated log loss values to evaluate the models' effectiveness.
- **Logistic Regression**
  Logistic regression is a method used for binary classification. It models the relationship between a binary dependent variable and independent variables by estimating the probability of outcomes using a logistic function. When applied to the training data, it predicts outcomes based on input features, helping assess the model's ability to generalize to unseen validation data.
- **XGBoost**
  XGBoost (Extreme Gradient Boosting) is an efficient, high-performance implementation of gradient boosting that leverages parallel processing and regularization techniques. When used on the training data, XGBoost builds multiple decision trees sequentially, capturing complex patterns and relationships within the data to enhance predictive performance.

- **Log Loss**
  Log loss, also known as logistic loss or cross-entropy loss, serves as the objective function for training models in binary classification. It measures how well the predicted probabilities match the actual outcomes, guiding the model to make more accurate predictions by minimizing the loss during training.

**DAY-21 | 06-11-2024:**

- **Developed a Model with Mentor Guidance**
  Created a function named self-supervised with relevant parameters as part of the model-building process, under the mentorship and guidance provided.

**DAY-22 | 07-11-2024:**

- **Continued Work on the Self-Supervised Function**
  Extended the self-supervised function by adding an encoder variable and implemented code to return encoder results. Ran the function to verify its correctness.

**DAY-23 | 08-11-2024:**

- **Implemented Encoder Method**
  **Encoder**
  Encoders are techniques used to convert categorical data into numerical format, making it suitable for machine learning models. Two common encoding methods are one-hot encoding, which creates binary columns for each category, and label encoding, where a unique integer is assigned to each category. This helps algorithms better process non-numerical features.

**DAY-24 | 11-11-2024:**

- **Performed Logistic Regression and XGBoost on Encoded Data**
  Implemented logistic regression and XGBoost algorithms on the encoded data to assess model performance. Both techniques were applied, and the outcomes were analyzed.

**DAY-25 | 12-11-2024:**

- **Reviewed and Refined Encoder Model**
  Discussed and reviewed the issues in the encoder model with the mentor and corrected the mistakes based on the feedback provided. Adjustments were made to ensure proper model implementation.

**DAY-26 | 13-11-2024:**

- **Initiated Development of Semi-Supervised Function**
  Started working on the semi-supervised function, focusing on setting up the training and model functions.
- Defined the training and model functions in the code.
- Discussed and analyzed the errors encountered in the outputs of the logistic regression and XGBoost models after encoding.

**DAY-27 | 14-11-2024:**