**1. Data Loading and Preprocessing**

* **Reading Data:**

df = pd.read\_csv('/content/drive/MyDrive/cytoauto/Levine\_32dimm.fcs.csv')

This line reads a CSV file containing data. The file is stored in a specified Google Drive path.

* **Basic Information:**
  + df.describe(): Provides a statistical summary of numerical columns.
  + df.info(): Displays data types, non-null counts, and memory usage.
* **Index and Column Details:**
  + df.index: Retrieves index labels.
  + df.columns: Retrieves column names.
  + df.shape: Returns the dataset dimensions (rows, columns).
* **Handling Missing Values:**
  + df.isnull(): Identifies missing values.
  + df.notnull(): Identifies non-missing values.

**2. Exploratory Data Analysis (EDA)**

* **Count of Null and Non-Null Values:**

counts = df.notnull().sum().to\_frame('Non-Null Values')

counts['Null Values'] = df.isnull().sum()

This calculates null and non-null value counts for each column and visualizes them using a bar plot.

* **Correlation Analysis:**

correlation\_matrix = df.corr()

Displays the relationship between numerical features in a heatmap.

* **Range Calculation:**

ranges = df.max() - df.min()

range\_df = pd.DataFrame({'Min': df.min(), 'Max': df.max(), 'Range': ranges})

Computes the range for each numerical column.

* **Class Distribution:**

sns.countplot(data=df, x='label')

Visualizes the frequency of each class label.

**3. Statistical Measures**

* **Skewness & Kurtosis:**

print(df.skew())

print(df.kurtosis())

Measures the asymmetry and peakedness of data distributions.

**4. Dimensionality Reduction**

* **Principal Component Analysis (PCA):**

pca = PCA(n\_components=2).fit\_transform(scaled\_features)

Reduces data dimensions for visualization. A 2D scatter plot and a 3D plot with KMeans clustering are created.

* **t-SNE Visualization:**

tsne = TSNE(n\_components=2, perplexity=30, n\_iter=1000)

tsne\_results = tsne.fit\_transform(scaled\_features)

Generates a 2D projection of high-dimensional data using t-SNE.

**5. Machine Learning Models**

* **Logistic Regression:**
  + A classification algorithm is applied to predict labels. It uses:
    - StandardScaler for feature scaling.
    - log\_loss for performance evaluation.
* **XGBoost Classifier:**
  + Applied for advanced classification using XGBClassifier.
  + Evaluates using log\_loss.

**6. Self-Supervised Learning**

* **Custom Encoder Model:**
  + Built using Keras with an architecture containing input, hidden, and output layers.
  + Trains using a binary mask for data corruption.
* **Encoded Features for Classification:**
  + Encodes features using a trained encoder.
  + Classification models (Logistic Regression and XGBoost) are applied to encoded data.

**7. Semi-Supervised Learning**

* **Custom Semi-Supervised Model:**
  + Combines supervised and unsupervised data to improve predictions.
  + Custom loss functions integrate labeled and unlabeled data.
* **Metrics Evaluation:**
  + **Accuracy:** Measures correct predictions.
  + **AUROC (Area Under ROC):** Evaluates model performance across thresholds.
* **Corrupted Data Creation:**

corrupted\_data = create\_corrupted\_data(df, keep\_probability=0.4)

Introduces randomness for self-supervised training.

* **Training with a Custom Neural Network:**

def custom\_model(input\_dimension, hidden\_dimension, label\_dimension):

Trains a semi-supervised neural network for improved label prediction.