CYTOAUTOCLUSTER

This notebook focuses on analyzing high-dimensional data, likely in the context of bioinformatics or clustering-based data exploration. The main goal appears to be implementing machine learning techniques to explore, visualize, and classify the data effectively.

1. **Data Loading and Exploration:**

- The dataset is loaded using pandas.
- Exploratory steps include checking column names, previewing the data, and performing descriptive analysis.

2. **Data Preprocessing:**

- Scaling features using StandardScaler to normalize data.
- Handling data dimensionality through PCA (Principal Component Analysis).

3. **Machine Learning Techniques:**

- **Clustering:** Using KMeans for unsupervised clustering.
- **Semi-Supervised Learning:** Applying LabelPropagation to leverage labeled and unlabeled data.
- Evaluating results with metrics such as accuracy, clustering scores, or visualization.

4. **Visualization:**

- Plotting PCA-reduced dimensions for better interpretability.
- Visualizing clustering results with matplotlib.

Day to day :

- 07-10-24
  - I searched for datasets containing both labeled and unlabeled data on Kaggle, Google Scholar, and Papers with Code.

- 08-10-24
  - My dataset was rejected because it lacked sufficient unlabeled data; the required proportion is 70% unlabeled and 30% labeled data.
  - The dataset must have at least 60-70% unlabeled data to perform semi-supervised learning effectively.
  - The data must be in a completely tabular format. The dataset should contain mass cytometry data with more than 40 columns.

- 09-10-24
  - I explored the components of cytometry using a sample dataset (Levine_32dim_notransform.csv) and gained an understanding of its structure and features.
  - Alongside this, I familiarized myself with basic Git commands to manage repositories on GitHub.
  - Learned how collaborative workflows operate in shared repositories, including actions like pushing changes, merging branches, and committing updates, which enable multiple users to work on the same codebase effectively.

- 10-10-24
  - Finalized the dataset (Levine_32dim_notransform.csv) for my analysis. After setting up the Python environment.
  - Successfully uploaded the dataset and created a DataFrame to begin working with the data.

- 11-10-24
  - Removed unnecessary columns (Cell_length, file_number, event_number) from the dataset to streamline the analysis. I then performed several Exploratory Data Analysis (EDA) techniques, including:
  - Data Information: Used info() to check the structure of the dataset.
  - Histogram: Visualized the distribution of data using histograms, which help represent the frequency distribution of the dataset by grouping data into bins.
  - Label and Unlabeled Data Percentages: Calculated and visualized the percentage of labeled vs. unlabeled data.
  - Non-null Values: Checked for missing data by counting non-null values in each column.

- 14-10-24
  - Performed additional Exploratory Data Analysis (EDA) techniques, including:
  - Range of Each Feature: Analyzed the minimum and maximum values for each feature to understand the scope of the data
  - .Correlation Matrix: Visualized the relationships between features to identify any strong correlations using a correlation matrix. This helps understand which features might be related to each other.
  - Class Label Distribution: Examined the distribution of the class labels to ensure balanced or imbalanced classes.
  - Box Plot: Created box plots to identify the spread of the data, including the median, quartiles, and potential outliers. Outliers in a box plot are defined as points outside the whiskers, which are typically set at 1.5 times the interquartile range (IQR) above the third quartile or below the first quartile.

- 15-10-24
  - Kurtosis measures the "tailedness" of a distribution, showing how much data lies in the extreme ends compared to a normal distribution. There are three types of kurtosis:
  - Mesokurtic: Normal distribution-like kurtosis.
  - Leptokurtic: High kurtosis, meaning more data in the tails and a higher probability of extreme values.

- Platykurtic: Low kurtosis, indicating a flatter distribution with fewer extreme values. High kurtosis suggests data is prone to extreme outliers, while low kurtosis indicates a more consistent dataset.
- Skewness measures the asymmetry of a distribution. A perfectly symmetrical distribution has zero skewness, but real-world data often exhibits skew:
- Right Skew (Positive Skew): The distribution tail is longer on the right.
- Left Skew (Negative Skew): The distribution tail is longer on the left. These insights were discussed with my mentor to understand the nature of the data better.

- 16-10-24
  - Performed individual plots for each feature in the dataset to visualize their Kurtosis and Skewness values. These plots provided a clear picture of the distribution and helped in understanding how each feature behaves in terms of symmetry and tail behavior.
  - Read a research paper related to the project titled "Mass cytometry: Single-cell cytometry for large-scale analysis of cellular phenotypes" .
  - This paper gave me insights into the theoretical aspects and applications of mass cytometry.
  - Also discussed the key points of the research paper with my mentor to deepen my understanding of the subject.

- 17-10-24
  - Encountered challenges while performing t-SNE and PCA on the dataset and did not achieve the expected results. After further analysis, realized the following steps were necessary:
  - Standardization: The relevant features must be scaled to a range from 0 to 1. This helps models like t-SNE and PCA perform better by ensuring all features contribute equally to the analysis. Standardizing values ensures that features have a mean of 0 and a standard deviation of .
  - Handling Missing Values: Columns containing null values should be removed before applying these techniques, as they can affect the performance and accuracy of dimensionality reduction models.
  - Applied t-SNE on the MNIST dataset, which helped me grasp how the model visualizes high-dimensional data in lower dimensions. This provided more clarity on improving results when applying t-SNE and PCA on my original dataset.

- 18-10-24
  - Gained brief knowledge about what are t-SNE and PCA and How they work

- 21-10-24
  - Gained a brief understanding of t-SNE (t-distributed Stochastic Neighbor Embedding) and PCA (Principal Component Analysis) and how they function as dimensionality reduction techniques.
  - t-SNE:A technique used to visualize high-dimensional data by reducing its dimensions to 2 or 3, making it possible to plot and understand complex structures.
  - PCA: PCA is a linear technique that transforms the data into a new coordinate system, where the axes (principal components) correspond to the directions of maximum variance in the data.

- 22-10-24
  - Performed several PCA operations on the dataset, including:
  - Standard Deviation: Calculated the standard deviation of each principal component to understand the spread of data along each axis.
  - Proportion of Variance: Analyzed how much variance is explained by each principal component, which helps in deciding how many components to retain for dimensionality reduction.
  - Cumulative Proportion: Calculated the cumulative proportion of variance to see how much total variance is explained by the first few principal components.
  - Additionally, visualized the data by performing a 3-D plot of PCA, which helped me observe the separation of data points in a reduced space and identify any distinct patterns or clusters.

- 23-10-24
  - Studied the Research paper (https://arxiv.org/pdf/2006.05278) about Deep SemiSupervised Learning.

- 25-10-24
  - Gained a brief knowledge about semi-super vised learning from the research paper .

- 28-10-24
  - Learned about two important concepts in semi-supervised learning:
  - Consistency Regularization: Consistency regularization ensures that a model's predictions remain stable when the input data undergoes slight perturbations or augmentations. This technique encourages the model to produce consistent outputs even if the input data is slightly altered, which helps the model generalize better, especially when dealing with unlabeled data. This approach is often used in methods like pseudo-labeling and virtual adversarial training.
  - Entropy Minimization: Entropy minimization focuses on reducing the uncertainty in the model’s predictions on unlabeled data. By encouraging the model to produce confident and sharp predictions, this method helps the model become more certain about the outputs, thus improving its performance on unlabeled data. Lower entropy indicates more confident and precise predictions, which is essential when dealing with sparse labeled data.

- 29-10-24
  - Performed Binary Masking on a set of rows in the dataset.Binary masking allows for the selective focus on specific data points by applying a binary mask (usually represented as 1s and 0s) to indicate which rows should be kept or altered.
  - This technique is particularly useful in semi-supervised learning, as it helps the model focus on the most relevant features of both labeled and unlabeled data, enhancing the model's ability to capture essential patterns while filtering out noise.
  - It improves performance when working with limited labeled data by directing the model's attention to more informative features.

- 30-10-24
  - Performed binary masking by randomly shuffling the column values in the DataFrame.
  - This technique helps simulate missing or corrupted data, allowing the model to learn and adapt to real-world scenarios where data may be incomplete or noisy.
  - Corrupted data refers to data that has been intentionally or unintentionally altered or distorted. This could occur due to errors in data collection, processing, or storage.
  - For testing or simulation purposes, corrupted data can help assess how models perform when faced with incomplete or noisy inputs.

- 01-11-24
  - Performed binary masking on the dataset to simulate corruption by creating a corrupted data frame.
  - The binary mask matrix was designed such that when it represents 1, the corresponding value in that index is considered corrupted, and when it represents 0, the value remains unchanged.
  - This allows the dataset to mimic real-world scenarios where some data might be missing or corrupted.Additionally, I created four variables to split the dataset into labeled and unlabeled categories:
  - x_labeled: Contains the feature rows with values (excluding the target column or label).
  - y_labeled: Contains the corresponding target (label) values for the labeled data.
  - x_unlabeled: Contains the feature rows with missing or null values (unlabeled data).
  - y_unlabeled: Contains the target values where the labels are missing or null (unlabeled data).

- 04-11-24
  - Performed the train-test split on the variables x_labeled, y_labeled, and y_unlabeled such that 70% of the data was used for training, and 30% was reserved for testingTrain-Test Split:
  - Train-test splitting is a common technique used to divide a dataset into two subsets:

- Training Set: This subset is used to train the model and learn the underlying patterns in the data.
- Test Set: This subset is used to evaluate the model's performance on unseen data, ensuring that the model generalizes well.The purpose of this split is to prevent overfitting, which occurs when a model performs well on the training data but fails to generalize to new, unseen data.
- By holding out a portion of the data for testing, we can get an unbiased estimate of the model's predictive capabilities. This helps in assessing how well the model will perform in real-world scenarios where it encounters data it hasn't been trained on.

- 05-11-24
  - Performed Logistic Regression and XGBoost on the training data split and also calculated the Log Loss values for model evaluation.
  - Logistic Regression: Logistic Regression is a statistical method used for binary classification. It models the relationship between a dependent binary variable (target) and one or more independent variables (features) by estimating probabilities using a logistic function.
  - XGBoost: XGBoost (Extreme Gradient Boosting) is a highly efficient implementation of gradient boosting, which enhances predictive performance through parallel processing and regularization techniques. It iteratively builds a series of decision trees, each tree learning from the mistakes of the previous one, allowing the model to capture complex patterns and interactions in the data.
  - Log Loss: Log Loss (also known as logarithmic loss) measures the accuracy of probabilistic predictions. In the case of binary classification, it quantifies how far off the predicted probabilities are from the actual binary labels (0 or 1). A lower Log Loss indicates better model performance, as it suggests that the predicted probabilities are closer to the true labels.

- 06-11-24
  - Created an model under the guidance of mentor created an function called self_supervised with some relevant parameters and a part of code on it.

- 07-11-24
  - Performed further coding on the self_supervised function like created encoder variable and to return encoder result.
  - Run the self_supervised funtion.

- 08-11-24
  - Encoders in a dataset are techniques used to convert categorical variables into numerical formats, making them suitable for machine learning algorithms.
  - Common methods include one-hot encoding, which creates binary columns for each category, and label encoding, which assigns a unique integer to each category.

- 11-11-24
  - Performed logistic regression and XGboost on encoded data.

- 12-11-24
  - Discussed and Corrected the mistakes in encoder model and took my inputs from the mentor.

- 13-11-24
  - Started working on a semi-supervised learning model and created three main functions to handle the different parts of the model pipeline:
  - Model Function: This function defines the structure of the neural network, where the input, hidden, and output layers are connected based on the specified dimensions.
  - Train Function : The train function performs the actual training by feeding labeled and unlabeled data into the model, calculating the loss, and updating the model's parameters.
  - Semi-supervised Function : This function (which you didn't detail further) likely involves the overall integration of the labeled and unlabeled data during training, allowing the model to leverage both types of data for improved learning performance.

- 14-11-24
  - Have continued developing the code for the semi-supervised learning model, and the new function is:
  - **x_train**: The labeled training features (input data for supervised learning).
  - **y_train**: The corresponding labeled targets or labels (output data for supervised learning).
  - **x_unlabeled**: The unlabeled features (input data for semi-supervised learning, where no labels are available).
  - **x_test**: The test data used for model evaluation after training.
  - **parameters**: A dictionary that contains various hyperparameters and model configurations. This may include the number of layers, learning rates, batch sizes, and other relevant parameters used during the training process.

- 15-11-24
  - I have successfully completed the code for your semi-supervised learning model and adjusted it according to the variables used in your notebook

- 19-11-24
  - I corrected the errors that occurred, redefined the labels, and then presented the updated work to the mentor.

- 20-11-24
  - Completed the semi-supervised learning model and showed the output to the mentor

- 21-11-24
  - I implemented a function to generate unlabeled data as part of the semi-supervised learning process and predicted the labels for the unlabeled data.

- 22-11-24
  - I did not get the predicted labels correctly, so I attempted to fix the issue and implemented t-SNE to verify the results against the previous t-SNE output in order to evaluate how the model is trained.
- 25-11-24
  - I got the tsne for the new dataset but it was not well clustred sir told to re-do it by predicting the labels again and find how many number of times the predicted labels occured
- 26-11-24
  - I didn’t standardized the data so I was not getting the desired tsne so I tried to standardized the new dataset and tried to get the tsne for that
- 27-11-24
  - After tsne implementation I tried to implement gradio and got the tsne and predicted labels for the 100 samples and created the presentation for the project