roject-cytoautocluster-aniruddh-2

November 28, 2024

0.1 #Infosys Springboard Project- CytoAutoCluster

0.2 Created by Aniruddh Joshi

0.3 Loading the Dataset

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.semi_supervised import LabelPropagation
from sklearn.metrics import silhouette_score
from sklearn.manifold import TSNE
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
import pandas as pd

# Provide the URL of the file
url = '/content/drive/MyDrive/dataset/data.csv'

# Load the dataset
df = pd.read_csv(url)

# Check the first few rows of the dataset
print(df.head())
```

```
   Event    Time  Cell_length      DNA1      DNA2    CD45RA     CD133  \
0      1  2693.0           22  4.391057  4.617262  0.162691 -0.029585
1      2  3736.0           35  4.340481  4.816692  0.701349 -0.038280
2      3  7015.0           32  3.838727  4.386369  0.603568 -0.032216
3      4  7099.0           29  4.255806  4.830048  0.433747 -0.027611
4      5  7700.0           25  3.976909  4.506433 -0.008809 -0.030297
```

```
       CD19      CD22     CD11b  …     CD117     CD49d    HLA-DR      CD64  \
0 -0.006696  0.066388 -0.009184  …  0.053050  0.853505  1.664480 -0.005376
1 -0.016654  0.074409  0.808031  …  0.089660  0.197818  0.491592  0.144814
2  0.073855 -0.042977 -0.001881  …  0.046222  2.586670  1.308337 -0.010961
3 -0.017661 -0.044072  0.733698  …  0.066470  1.338669  0.140523 -0.013449
4  0.080423  0.495791  1.107627  … -0.006223  0.180924  0.197332  0.076167

       CD41  Viability  file_number  event_number  label  individual
0 -0.001961   0.648429     3.627711           307    1.0           1
1  0.868014   0.561384     3.627711           545    1.0           1
2 -0.010413   0.643337     3.627711          1726    1.0           1
3 -0.026039  -0.026523     3.627711          1766    1.0           1
4 -0.040488   0.283287     3.627711          2031    1.0           1

[5 rows x 42 columns]
```

[ ]:
```
df.head()
```

[ ]:
```
   Event    Time  Cell_length      DNA1      DNA2    CD45RA     CD133  \
0      1  2693.0           22  4.391057  4.617262  0.162691 -0.029585
1      2  3736.0           35  4.340481  4.816692  0.701349 -0.038280
2      3  7015.0           32  3.838727  4.386369  0.603568 -0.032216
3      4  7099.0           29  4.255806  4.830048  0.433747 -0.027611
4      5  7700.0           25  3.976909  4.506433 -0.008809 -0.030297

       CD19      CD22     CD11b  …     CD117     CD49d    HLA-DR      CD64  \
0 -0.006696  0.066388 -0.009184  …  0.053050  0.853505  1.664480 -0.005376
1 -0.016654  0.074409  0.808031  …  0.089660  0.197818  0.491592  0.144814
2  0.073855 -0.042977 -0.001881  …  0.046222  2.586670  1.308337 -0.010961
3 -0.017661 -0.044072  0.733698  …  0.066470  1.338669  0.140523 -0.013449
4  0.080423  0.495791  1.107627  … -0.006223  0.180924  0.197332  0.076167

       CD41  Viability  file_number  event_number  label  individual
0 -0.001961   0.648429     3.627711           307    1.0           1
1  0.868014   0.561384     3.627711           545    1.0           1
2 -0.010413   0.643337     3.627711          1726    1.0           1
3 -0.026039  -0.026523     3.627711          1766    1.0           1
4 -0.040488   0.283287     3.627711          2031    1.0           1

[5 rows x 42 columns]
```

[ ]:
```
print("Basic Structure of the Data:")
display(df)
```

```
Basic Structure of the Data:
         Event        Time  Cell_length      DNA1      DNA2    CD45RA  \
0            1    2693.00           22  4.391057  4.617262  0.162691
1            2    3736.00           35  4.340481  4.816692  0.701349
2            3    7015.00           32  3.838727  4.386369  0.603568
3            4    7099.00           29  4.255806  4.830048  0.433747
4            5    7700.00           25  3.976909  4.506433 -0.008809
…          …        …          …       …         …        …
265622  265623  707951.44           41  6.826629  7.133022  1.474081
265623  265624  708145.44           45  6.787791  7.154026  0.116755
265624  265625  708398.44           41  6.889866  7.141219  0.684921
265625  265626  708585.44           39  6.865218  7.144353  0.288761
265626  265627  709122.44           41  6.887820  7.127359  0.360753

           CD133      CD19      CD22     CD11b  …     CD117     CD49d  \
0      -0.029585 -0.006696  0.066388 -0.009184  …  0.053050  0.853505
1      -0.038280 -0.016654  0.074409  0.808031  …  0.089660  0.197818
2      -0.032216  0.073855 -0.042977 -0.001881  …  0.046222  2.586670
3      -0.027611 -0.017661 -0.044072  0.733698  …  0.066470  1.338669
4      -0.030297  0.080423  0.495791  1.107627  … -0.006223  0.180924
…            …         …         …         …  …       …         …
265622 -0.019174 -0.055620 -0.007261  0.063395  … -0.011105  0.533736
265623 -0.056213 -0.008864 -0.035158 -0.041845  …  0.143869  1.269464
265624 -0.006264 -0.026111 -0.030837 -0.034641  …  0.087102 -0.055912
265625 -0.011310 -0.048786  0.073983 -0.031787  … -0.047971  0.101955
265626  0.128604 -0.006934  0.109846  3.864711  …  0.080195  0.037962

          HLA-DR      CD64      CD41  Viability  file_number  event_number  \
0       1.664480 -0.005376 -0.001961   0.648429     3.627711           307
1       0.491592  0.144814  0.868014   0.561384     3.627711           545
2       1.308337 -0.010961 -0.010413   0.643337     3.627711          1726
3       0.140523 -0.013449 -0.026039  -0.026523     3.627711          1766
4       0.197332  0.076167 -0.040488   0.283287     3.627711          2031
…            …         …         …          …           …             …
265622  0.123758 -0.042495 -0.027971   0.236957     3.669327        102686
265623  0.047215 -0.008000 -0.025811  -0.003500     3.669327        102690
265624  0.501536  0.053884 -0.042602   0.107206     3.669327        102701
265625  6.200001  0.296877  0.192786   0.620872     3.669327        102706
265626  3.675123 -0.000878 -0.052526   0.310466     3.669327        102720

        label  individual
0         1.0           1
1         1.0           1
2         1.0           1
3         1.0           1
4         1.0           1
…         …           …
265622    NaN           2
265623    NaN           2
265624    NaN           2
```

```
265625     NaN           2
265626     NaN           2

[265627 rows x 42 columns]
```

```
[ ]: print("\nData Information:")
     display(df.info())
```

```
Data Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 265627 entries, 0 to 265626
Data columns (total 42 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Event         265627 non-null  int64
 1   Time          265627 non-null  float64
 2   Cell_length   265627 non-null  int64
 3   DNA1          265627 non-null  float64
 4   DNA2          265627 non-null  float64
 5   CD45RA        265627 non-null  float64
 6   CD133         265627 non-null  float64
 7   CD19          265627 non-null  float64
 8   CD22          265627 non-null  float64
 9   CD11b         265627 non-null  float64
 10  CD4           265627 non-null  float64
 11  CD8           265627 non-null  float64
 12  CD34          265627 non-null  float64
 13  Flt3          265627 non-null  float64
 14  CD20          265627 non-null  float64
 15  CXCR4         265627 non-null  float64
 16  CD235ab       265627 non-null  float64
 17  CD45          265627 non-null  float64
 18  CD123         265627 non-null  float64
 19  CD321         265627 non-null  float64
 20  CD14          265627 non-null  float64
 21  CD33          265627 non-null  float64
 22  CD47          265627 non-null  float64
 23  CD11c         265627 non-null  float64
 24  CD7           265627 non-null  float64
 25  CD15          265627 non-null  float64
 26  CD16          265627 non-null  float64
 27  CD44          265627 non-null  float64
 28  CD38          265627 non-null  float64
 29  CD13          265627 non-null  float64
 30  CD3           265627 non-null  float64
 31  CD61          265627 non-null  float64
 32  CD117         265627 non-null  float64
 33  CD49d         265627 non-null  float64
 34  HLA-DR        265627 non-null  float64
 35  CD64          265627 non-null  float64
 36  CD41          265627 non-null  float64
 37  Viability     265627 non-null  float64
 38  file_number   265627 non-null  float64
 39  event_number  265627 non-null  int64
 40  label         104184 non-null  float64
 41  individual    265627 non-null  int64
dtypes: float64(38), int64(4)
memory usage: 85.1 MB
```

```
None
```

```
[ ]: print("\nMissing Values:")
     missing_values = df.isnull().sum()
     missing_percentage = (missing_values / len(df)) * 100
     missing_df = pd.DataFrame({'Missing Values': missing_values, 'Percentage':␣
      ↪missing_percentage})
     display(missing_df[missing_df['Missing Values'] > 0])
```

```
Missing Values:
```

```
       Missing Values  Percentage
label          161443   60.778084
```

```
[ ]: print("\nDescriptive Statistics:")
     display(df.describe())
```

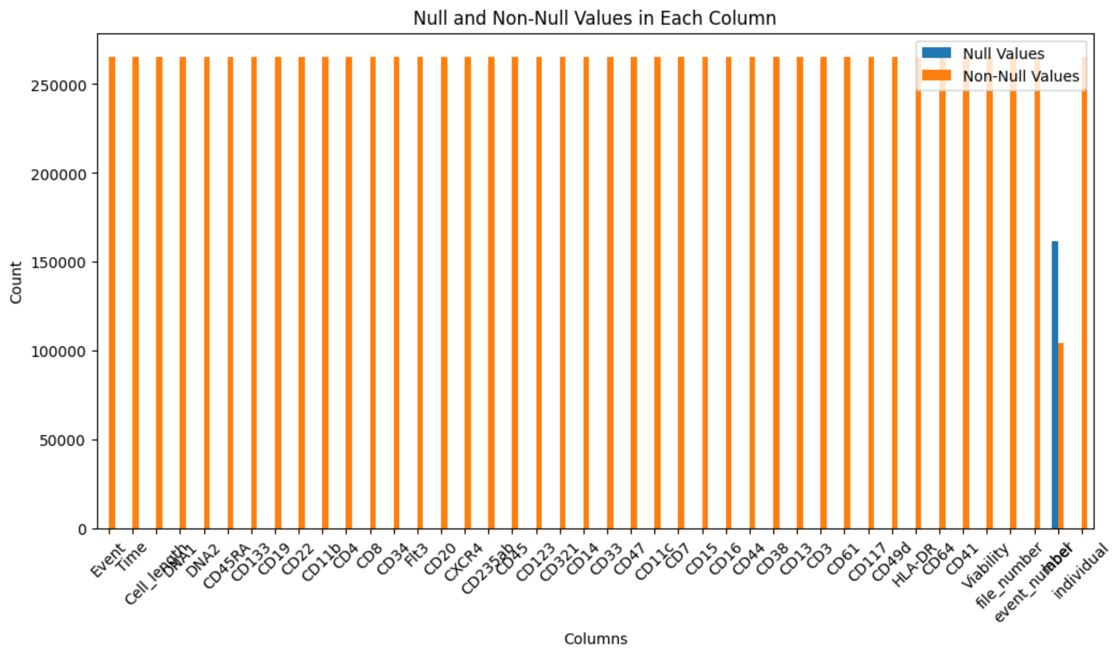
```
Descriptive Statistics:
```

```
               Event           Time    Cell_length           DNA1  \
count  265627.000000  265627.000000  265627.000000  265627.000000
mean   132814.000000  272948.345014      34.450572       4.606956
std     76680.054314  171220.139430      11.446694       1.312831
min         1.000000       1.000000      10.000000       2.786488
25%     66407.500000  120196.000000      26.000000       3.700023
50%    132814.000000  253276.000000      33.000000       4.022127
75%    199220.500000  424502.500000      41.000000       6.353313
max    265627.000000  709122.440000      65.000000       7.001489

                DNA2         CD45RA          CD133           CD19  \
count  265627.000000  265627.000000  265627.000000  265627.000000
mean        5.198308       0.688127       0.145960       0.509301
std         1.150357       0.609105       0.259267       0.857462
min         2.236450      -0.057305      -0.058081      -0.058089
25%         4.407822       0.204625      -0.022935      -0.018838
```

```
50%           4.698415       0.549387       0.025353       0.075210
75%           6.766268       1.031198       0.224299       0.548386
max           7.472308       6.691197       5.527494       4.990085

                CD22           CD11b  …          CD117          CD49d  \
count  265627.000000  265627.000000  …  265627.000000  265627.000000
mean        0.397323       0.710319  …       0.131199       0.794938
std         0.762126       1.011434  …       0.313208       0.627619
min        -0.057342      -0.058236  …      -0.057668      -0.058064
25%        -0.020689      -0.000294  …      -0.023957       0.283013
50%         0.058790       0.257923  …      -0.000410       0.677212
75%         0.386481       0.923517  …       0.154736       1.190787
max         5.160477       5.260789  …       5.502125       5.153438

              HLA-DR           CD64           CD41      Viability  \
count  265627.000000  265627.000000  265627.000000  265627.000000
mean        1.521812       0.551512       0.261754       0.570037
std         1.694211       0.888739       0.617065       0.589738
min        -0.057974      -0.058199      -0.058244      -0.057979
25%         0.057709      -0.010582      -0.020166       0.065523
50%         0.611335       0.122493       0.052229       0.398230
75%         2.888240       0.604131       0.305591       0.931058
max         7.052507       4.517843       7.718288       2.433031

         file_number   event_number          label     individual
count  265627.000000  265627.000000  104184.000000  265627.000000
mean        3.639348  171288.314234       8.116102       1.279625
std         0.018678  123904.361456       2.457486       0.448816
min         3.627711       1.000000       1.000000       1.000000
25%         3.627711   58679.500000       7.000000       1.000000
50%         3.627711  152783.000000       8.000000       1.000000
75%         3.669327  282369.000000      10.000000       2.000000
max         3.669327  400112.000000      14.000000       2.000000

[8 rows x 42 columns]
```

##NULL VS NOT NULL

```
[ ]: df = pd.DataFrame(df)
     null_values = df.isnull().sum()
     non_null_values = df.notnull().sum()
     plot_data = pd.DataFrame({
         'Null Values': null_values,
         'Non-Null Values': non_null_values
     })
     plot_data.plot(kind='bar', figsize=(12, 6))
     plt.title('Null and Non-Null Values in Each Column')
     plt.xlabel('Columns')
     plt.ylabel('Count')
     plt.xticks(rotation=45)
     plt.legend(loc='upper right')
     plt.show()
```



#IMPORTANT TO RUN (DROP PART)

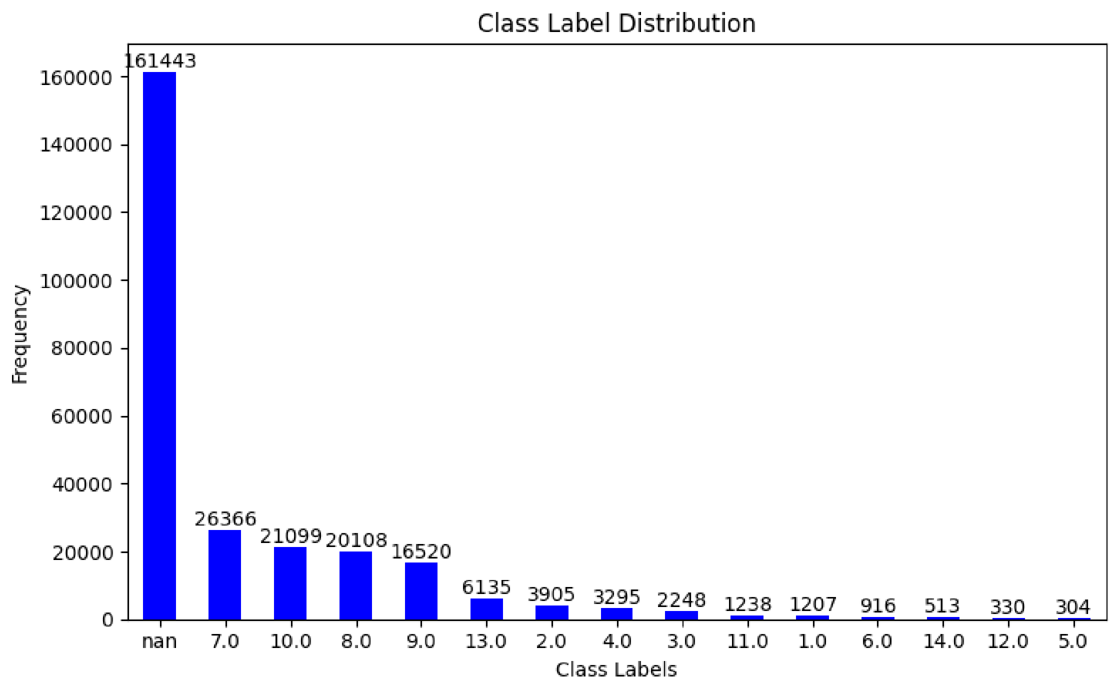
```
[ ]: df = df.drop(columns=['Event','Time','individual','file_number','event_number'])
```

##CLASS LABEL DISTRIBUTION

```
[ ]: import pandas as pd
     import matplotlib.pyplot as plt

     data = df


     label_distribution = df['label'].value_counts(dropna=False)
     print("Class Label Distribution:")
     print(label_distribution)


     label_distribution = df['label'].value_counts(dropna=False)

     plt.figure(figsize=(8, 5))
     bars = label_distribution.plot(kind='bar', color='blue')
```

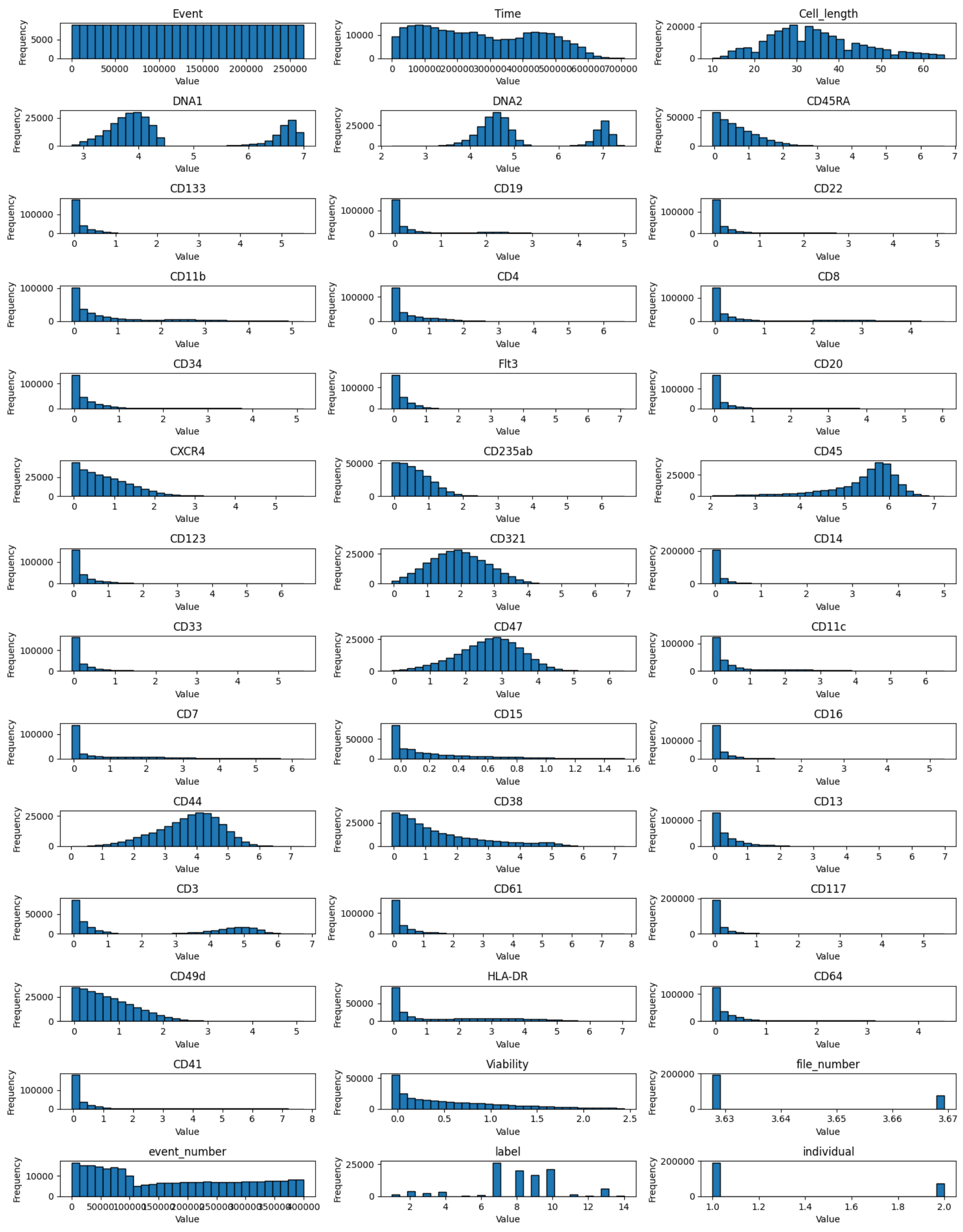
```
plt.title('Class Label Distribution')
plt.xlabel('Class Labels')
plt.ylabel('Frequency')
plt.xticks(rotation=0)

for bar in bars.patches:
    bars.annotate(bar.get_height(),
                  (bar.get_x() + bar.get_width() / 2, bar.get_height()),
                  ha='center',
                  va='bottom')

plt.tight_layout()
plt.show()
```

```
Class Label Distribution:
label
NaN     161443
7.0      26366
10.0     21099
8.0      20108
9.0      16520
13.0      6135
2.0       3905
4.0       3295
3.0       2248
11.0      1238
1.0       1207
6.0        916
14.0       513
12.0       330
5.0        304
Name: count, dtype: int64
```



##Histograms of Features

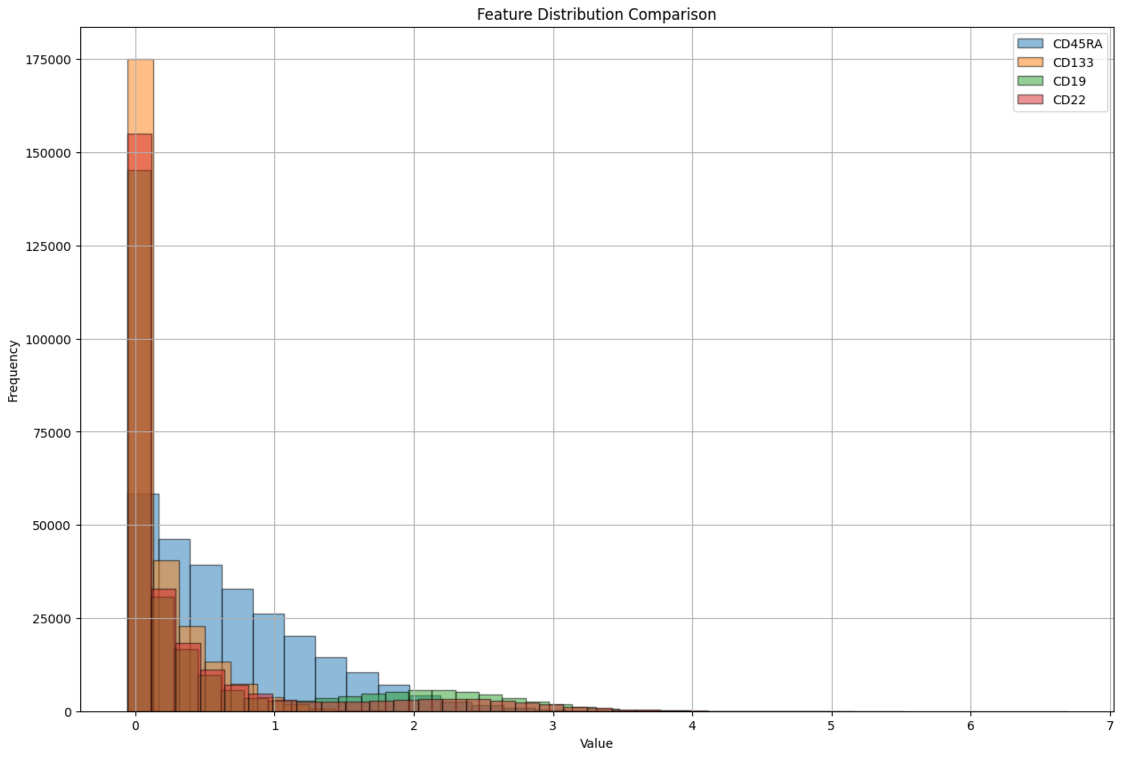
```
[ ]: import pandas as pd
import matplotlib.pyplot as plt

# data = pd.read_csv('/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv')

# Select only numerical columns for histogram plotting
numerical_columns = data.select_dtypes(include=['float64', 'int64']).columns

# Set up the figure for subplots
plt.figure(figsize=(15, 20))

# Iterate through numerical columns and create a histogram for each
for i, column in enumerate(numerical_columns, 1):
    plt.subplot(len(numerical_columns)//3 + 1, 3, i)
    plt.hist(data[column], bins=30, edgecolor='black')
    plt.title(column)
    plt.xlabel('Value')
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```

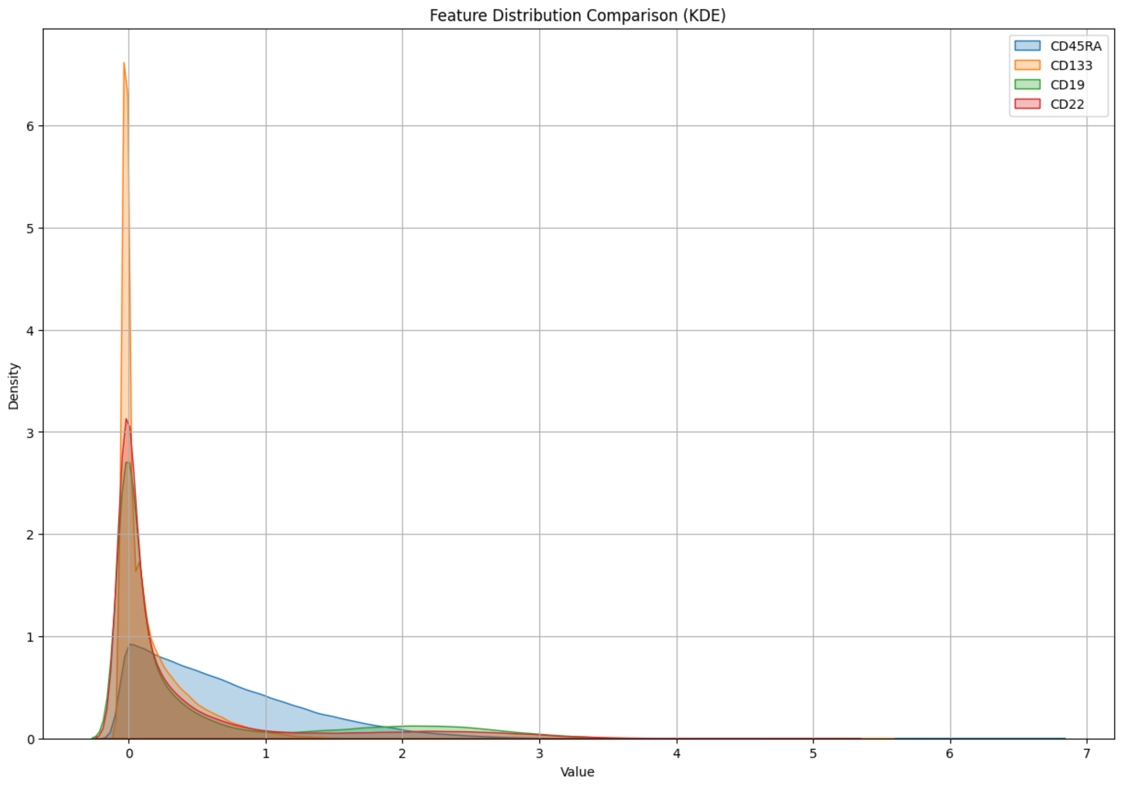


##Comparing Feature Distributions with Histograms and KDE Plots

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

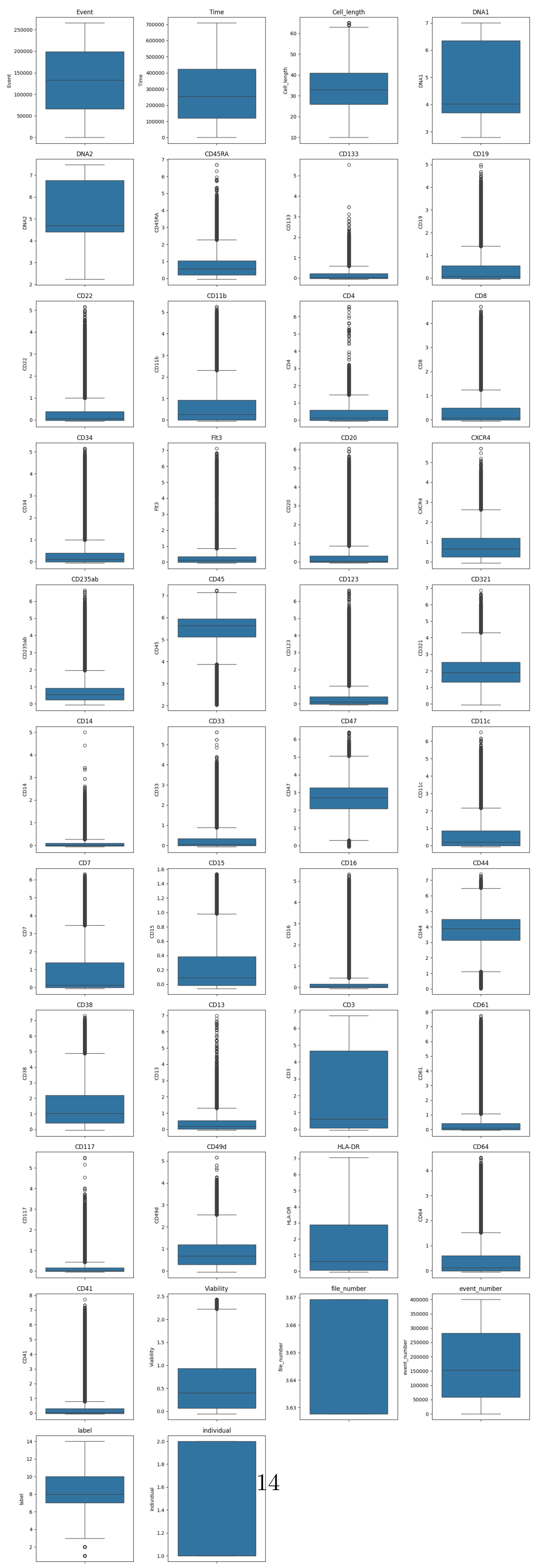
```
# Load the dataset
# data = pd.read_csv('/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv')

# Select features for comparison (adjust based on your dataset)
features_to_compare = ['CD45RA', 'CD133', 'CD19', 'CD22']  # Example features,␣
 ↪replace with your own
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728']  # Custom color palette

# Step 1: Histograms for feature distribution comparison
plt.figure(figsize=(15, 10))

for feature, color in zip(features_to_compare, colors):
    plt.hist(data[feature], bins=30, alpha=0.5, label=feature,␣
 ↪edgecolor='black', color=color)

plt.title('Feature Distribution Comparison')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.legend()
plt.grid(True)
plt.show()

# Step 2: Kernel Density Estimation (KDE) for smoother distribution comparison
plt.figure(figsize=(15, 10))

for feature, color in zip(features_to_compare, colors):
    sns.kdeplot(data[feature], label=feature, fill=True, alpha=0.3, color=color)

plt.title('Feature Distribution Comparison (KDE)')
plt.xlabel('Value')
plt.ylabel('Density')
plt.legend()
plt.grid(True)
plt.show()
```


Feature Distribution Comparison
CD45RA
CD133
CD19
CD22
Frequency
Value
175000
150000
125000
100000
75000
50000
25000
0
0
1
2
3
4
5
6
7


Feature Distribution Comparison (KDE)
CD45RA
CD133
CD19
CD22
Density
Value
6
5
4
3
2
1
0
0
1
2
3
4
5
6
7

##Box Plot Analysis of Feature Distributions

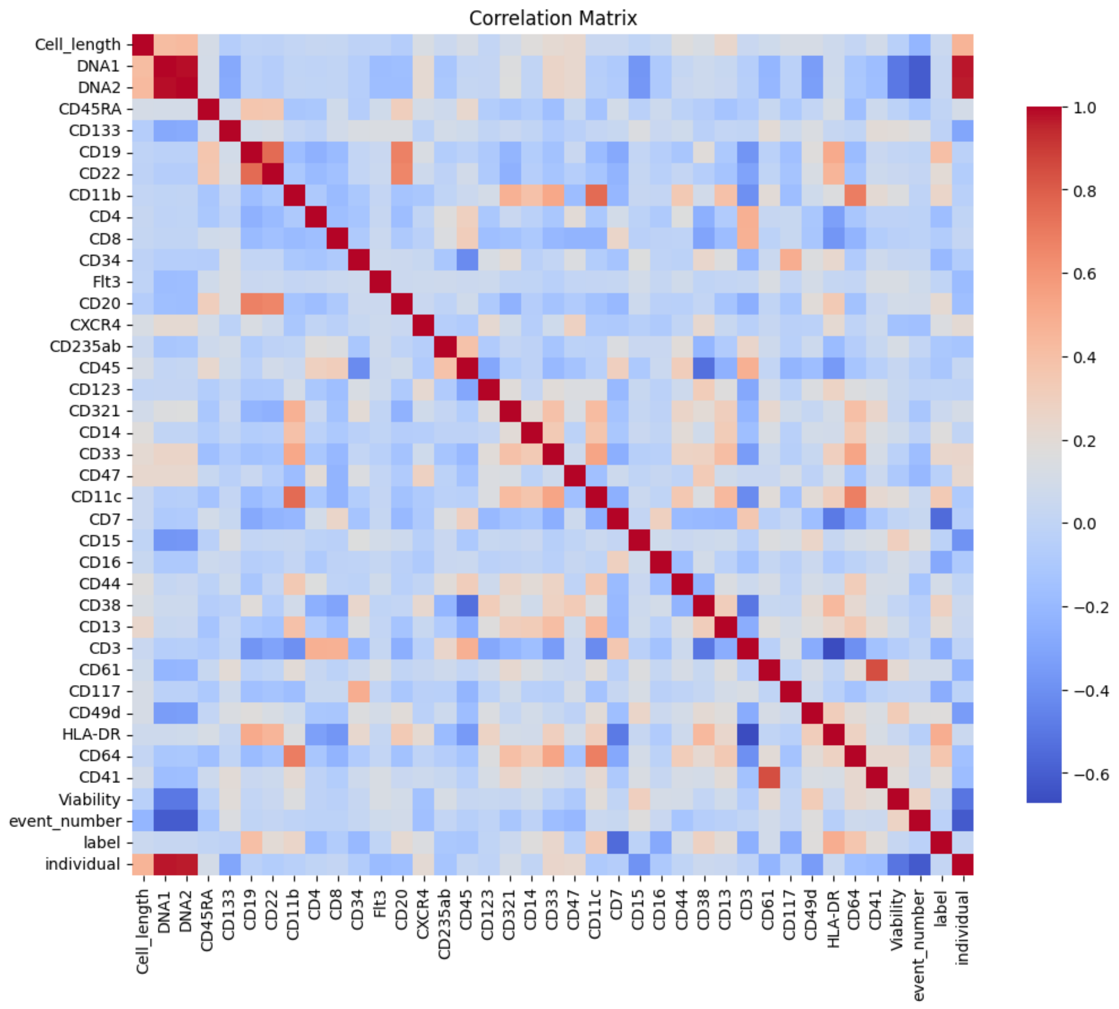
```
[ ]: import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns

     # Load the dataset
     # data = pd.read_csv('/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv')

     # Step 1: Box Plots for Numerical Features
     numerical_features = data.select_dtypes(include=['float64', 'int64']).columns ␣
      ↪# Select numerical columns
     rows = (len(numerical_features) // 4) + 1  # Calculate the number of rows needed

     plt.figure(figsize=(15, rows * 4))
     for i, feature in enumerate(numerical_features):
         plt.subplot(rows, 4, i + 1)
         sns.boxplot(data[feature])
         plt.title(feature)
     plt.tight_layout()
     plt.show()

     # Step 2: Count Plots for Categorical Features
     categorical_features = data.select_dtypes(include=['object']).columns  # Select␣
      ↪categorical columns

     plt.figure(figsize=(15, 10))
     for i, feature in enumerate(categorical_features):
         plt.subplot(2, 2, i + 1)
         sns.countplot(x=data[feature], order=data[feature].value_counts().index)
         plt.title(feature)
         plt.xticks(rotation=45)
     plt.tight_layout()
     plt.show()
```


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

```
<Figure size 1500x1000 with 0 Axes>
```

##Feature Correlation Matrix Analysis

```
[ ]: import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns

     # Load the data
     # data = pd.read_csv('/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv')

     # Drop the specified columns
     data = data.drop(columns=['file_number', 'Event', 'Time'])

     # Calculate the correlation matrix
     correlation_matrix = data.corr()

     # Set up the matplotlib figure
     plt.figure(figsize=(12, 10))

     # Create a heatmap using Seaborn without annotations
     sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm', square=True,␣
      ↪cbar_kws={"shrink": .8})

     plt.title('Correlation Matrix')
     plt.show()
```



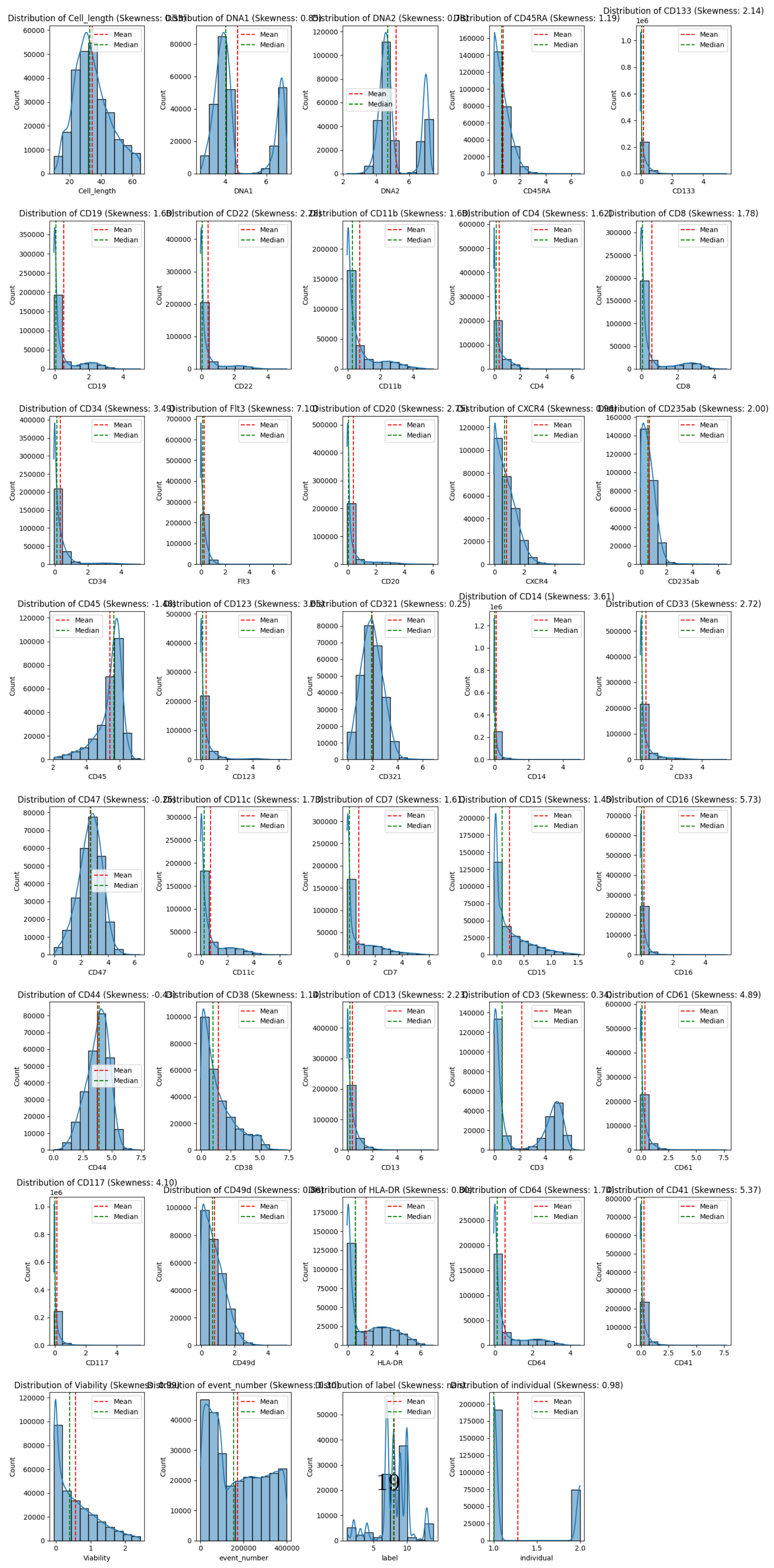
##Analysis of Feature Skewness

```
[ ]: import pandas as pd
     from scipy.stats import skew
     import matplotlib.pyplot as plt
     import seaborn as sns
     import math

     # Load the data
     # data = pd.read_csv('/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv')
     data = data.drop(columns=['file_number', 'Event', 'Time'])

     # Calculate skewness
     skewness = data.apply(skew)

     # Function to categorize skewness
```

```python
def categorize_skewness(value):
    if value > 0.5:
        return 'Right-skewed'
    elif value < -0.5:
        return 'Left-skewed'
    else:
        return 'Approximately symmetrical'

# Apply the categorization
skewness_category = skewness.apply(categorize_skewness)

# Display skewness and its categorization
skewness_df = pd.DataFrame({'Skewness': skewness, 'Category':␣
↪skewness_category})
print(skewness_df)

# Set the number of columns in the grid
n_cols = 5  # Adjust this value for number of plots per row
n_plots = len(data.columns)
n_rows = math.ceil(n_plots / n_cols)

# Create subplots grid
fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, 4 * n_rows))  # Adjust␣
↪figsize for larger or smaller plots
axes = axes.flatten()  # Flatten axes array to make it easier to index

# Loop through columns and plot histograms on each subplot
for idx, col in enumerate(data.columns):
    sns.histplot(data[col], bins=10, kde=True, ax=axes[idx])
    axes[idx].set_title(f'Distribution of {col} (Skewness: {skewness[col]:.
↪2f})')
    axes[idx].axvline(data[col].mean(), color='red', linestyle='--',␣
↪label='Mean')
    axes[idx].axvline(data[col].median(), color='green', linestyle='--',␣
↪label='Median')
    axes[idx].legend()

# Remove any unused subplots (if n_plots is not a perfect multiple of n_cols)
for i in range(n_plots, len(axes)):
    fig.delaxes(axes[i])

# Ensure the layout is tight and the plot is shown properly
plt.tight_layout()
plt.show(block=True)  # Ensure plt.show() does not block rendering
```

```
             Skewness                      Category
Cell_length  0.527832                  Right-skewed
```

```
DNA1           0.845010                Right-skewed
DNA2           0.779167                Right-skewed
CD45RA         1.191595                Right-skewed
CD133          2.141953                Right-skewed
CD19           1.682609                Right-skewed
CD22           2.283181                Right-skewed
CD11b          1.679089                Right-skewed
CD4            1.622044                Right-skewed
CD8            1.775713                Right-skewed
CD34           3.492437                Right-skewed
Flt3           7.098151                Right-skewed
CD20           2.754699                Right-skewed
CXCR4          0.955342                Right-skewed
CD235ab        2.001479                Right-skewed
CD45          -1.484824                 Left-skewed
CD123          3.648890                Right-skewed
CD321          0.247097  Approximately symmetrical
CD14           3.609006                Right-skewed
CD33           2.724977                Right-skewed
CD47          -0.250323  Approximately symmetrical
CD11c          1.733888                Right-skewed
CD7            1.606528                Right-skewed
CD15           1.445147                Right-skewed
CD16           5.733203                Right-skewed
CD44          -0.431589  Approximately symmetrical
CD38           1.141482                Right-skewed
CD13           2.234311                Right-skewed
CD3            0.342239  Approximately symmetrical
CD61           4.894707                Right-skewed
CD117          4.097508                Right-skewed
CD49d          0.856805                Right-skewed
HLA-DR         0.795359                Right-skewed
CD64           1.743733                Right-skewed
CD41           5.366314                Right-skewed
Viability      0.985417                Right-skewed
event_number   0.304116  Approximately symmetrical
label               NaN  Approximately symmetrical
individual     0.982030                Right-skewed
```


Distribution of Cell_length (Skewness: 0.53)
Distribution of DNA1 (Skewness: 0.85)
Distribution of DNA2 (Skewness: 0.78)
Distribution of CD45RA (Skewness: 1.19)
Distribution of CD133 (Skewness: 2.14)
Distribution of CD19 (Skewness: 1.69)
Distribution of CD22 (Skewness: 2.28)
Distribution of CD11b (Skewness: 1.69)
Distribution of CD4 (Skewness: 1.62)
Distribution of CD8 (Skewness: 1.78)
Distribution of CD34 (Skewness: 3.49)
Distribution of Flt3 (Skewness: 7.10)
Distribution of CD20 (Skewness: 2.75)
Distribution of CXCR4 (Skewness: 0.98)
Distribution of CD235ab (Skewness: 2.00)
Distribution of CD45 (Skewness: -1.48)
Distribution of CD123 (Skewness: 3.05)
Distribution of CD321 (Skewness: 0.25)
Distribution of CD14 (Skewness: 3.61)
Distribution of CD33 (Skewness: 2.72)
Distribution of CD47 (Skewness: -0.25)
Distribution of CD11c (Skewness: 1.73)
Distribution of CD7 (Skewness: 1.61)
Distribution of CD15 (Skewness: 1.45)
Distribution of CD16 (Skewness: 5.73)
Distribution of CD44 (Skewness: -0.43)
Distribution of CD38 (Skewness: 1.10)
Distribution of CD13 (Skewness: 2.23)
Distribution of CD3 (Skewness: 0.34)
Distribution of CD61 (Skewness: 4.89)
Distribution of CD117 (Skewness: 4.10)
Distribution of CD49d (Skewness: 0.36)
Distribution of HLA-DR (Skewness: 0.80)
Distribution of CD64 (Skewness: 1.78)
Distribution of CD41 (Skewness: 5.37)
Distribution of Viability (Skewness: 0.89)
Distribution of event_number (Skewness: 0.10)
Distribution of label (Skewness: nan)
Distribution of individual (Skewness: 0.98)
Mean
Median
Count

```
[ ]:
```

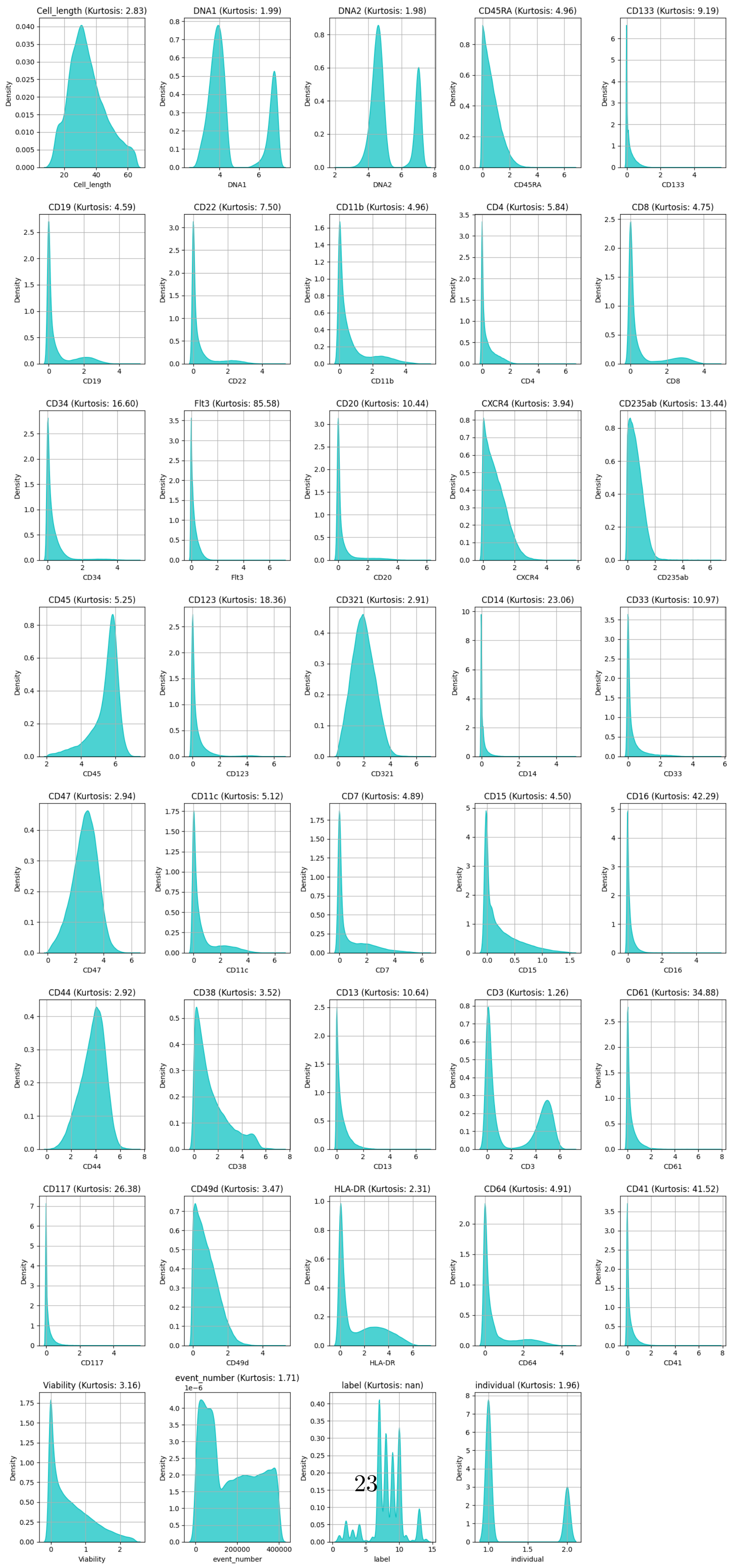
##Analysis of Feature Kurtosis

```
[ ]: import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     from scipy.stats import kurtosis
     import math

     # Load the data
     # data = pd.read_csv('/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv')

     # Drop the specified columns
     data = data.drop(columns=['file_number', 'Event', 'Time'])

     # Calculate kurtosis for each column
     kurtosis_values = data.apply(kurtosis, fisher=False)  # Fisher=False gives␣
      ↪Pearson kurtosis (normal kurtosis = 3)

     # Create a DataFrame with kurtosis values
     kurtosis_df = pd.DataFrame({'Column': data.columns, 'Kurtosis':␣
      ↪kurtosis_values})

     # Categorize the kurtosis values (Leptokurtic, Mesokurtic, Platykurtic)
     def categorize_kurtosis(value):
         if value > 3:
             return 'Leptokurtic (heavy tails)'
         elif value < 3:
             return 'Platykurtic (light tails)'
         else:
             return 'Mesokurtic (normal tails)'

     kurtosis_df['Category'] = kurtosis_df['Kurtosis'].apply(categorize_kurtosis)

     # Print the kurtosis values and their categories
     print(kurtosis_df)

     # Set the number of columns in the grid
     n_cols = 5  # You can adjust this to control how many plots per row
     n_plots = len(data.columns)
     n_rows = math.ceil(n_plots / n_cols)

     # Create subplots grid
```

```
fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, 4 * n_rows))  # Adjust␣
↪figsize for larger or smaller plots
axes = axes.flatten()  # Flatten axes array to make it easier to index

# Loop through columns and plot KDE on each subplot
for idx, column in enumerate(data.columns):
    sns.kdeplot(data[column].dropna(), color='c', fill=True, alpha=0.7,␣
↪ax=axes[idx])
    axes[idx].set_title(f'{column} (Kurtosis: {kurtosis_df.
↪loc[kurtosis_df["Column"] == column, "Kurtosis"].values[0]:.2f})')
    axes[idx].set_xlabel(column)
    axes[idx].set_ylabel('Density')
    axes[idx].grid(True)

# Remove any unused subplots (if n_plots is not a perfect multiple of n_cols)
for i in range(n_plots, len(axes)):
    fig.delaxes(axes[i])

plt.tight_layout()
plt.show()
```

```
                 Column  Kurtosis                  Category
Cell_length  Cell_length  2.834033  Platykurtic (light tails)
DNA1                DNA1  1.994037  Platykurtic (light tails)
DNA2                DNA2  1.975021  Platykurtic (light tails)
CD45RA            CD45RA  4.964272  Leptokurtic (heavy tails)
CD133              CD133  9.190066  Leptokurtic (heavy tails)
CD19                CD19  4.590887  Leptokurtic (heavy tails)
CD22                CD22  7.500223  Leptokurtic (heavy tails)
CD11b              CD11b  4.964495  Leptokurtic (heavy tails)
CD4                  CD4  5.844261  Leptokurtic (heavy tails)
CD8                  CD8  4.745776  Leptokurtic (heavy tails)
CD34                CD34 16.596416  Leptokurtic (heavy tails)
Flt3                Flt3 85.583534  Leptokurtic (heavy tails)
CD20                CD20 10.435449  Leptokurtic (heavy tails)
CXCR4              CXCR4  3.936307  Leptokurtic (heavy tails)
CD235ab          CD235ab 13.440586  Leptokurtic (heavy tails)
CD45                CD45  5.246770  Leptokurtic (heavy tails)
CD123              CD123 18.361217  Leptokurtic (heavy tails)
CD321              CD321  2.914593  Platykurtic (light tails)
CD14                CD14 23.062535  Leptokurtic (heavy tails)
CD33                CD33 10.967536  Leptokurtic (heavy tails)
CD47                CD47  2.943834  Platykurtic (light tails)
CD11c              CD11c  5.117156  Leptokurtic (heavy tails)
CD7                  CD7  4.885115  Leptokurtic (heavy tails)
CD15                CD15  4.504387  Leptokurtic (heavy tails)
CD16                CD16 42.287749  Leptokurtic (heavy tails)
```

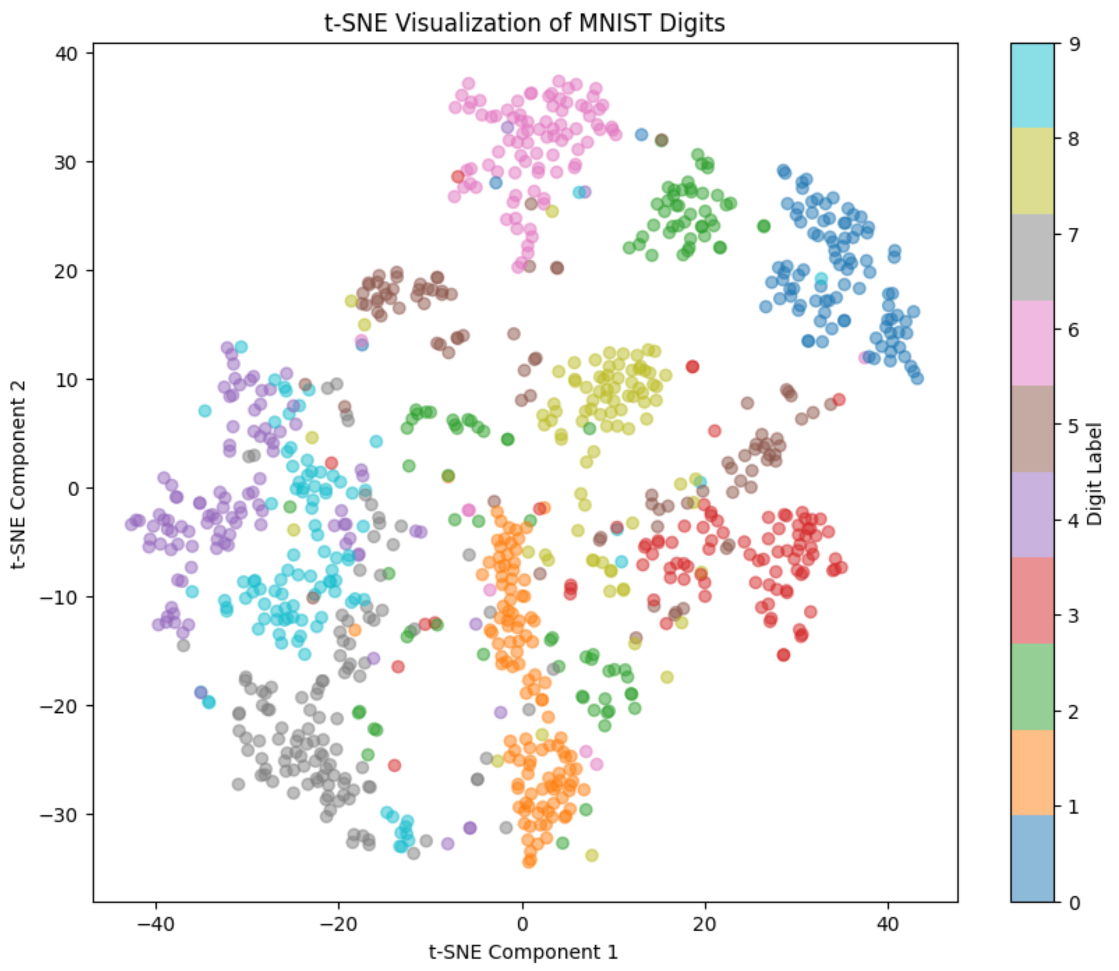
```
CD44                 CD44   2.918792  Platykurtic (light tails)
CD38                 CD38   3.521190  Leptokurtic (heavy tails)
CD13                 CD13  10.637564  Leptokurtic (heavy tails)
CD3                   CD3   1.264612  Platykurtic (light tails)
CD61                 CD61  34.878020  Leptokurtic (heavy tails)
CD117               CD117  26.375108  Leptokurtic (heavy tails)
CD49d               CD49d   3.468119  Leptokurtic (heavy tails)
HLA-DR             HLA-DR   2.309924  Platykurtic (light tails)
CD64                 CD64   4.910631  Leptokurtic (heavy tails)
CD41                 CD41  41.521113  Leptokurtic (heavy tails)
Viability       Viability   3.156935  Leptokurtic (heavy tails)
event_number event_number   1.706183  Platykurtic (light tails)
label               label        NaN  Mesokurtic (normal tails)
individual     individual   1.964382  Platykurtic (light tails)
```


Cell_length (Kurtosis: 2.83)
DNA1 (Kurtosis: 1.99)
DNA2 (Kurtosis: 1.98)
CD45RA (Kurtosis: 4.96)
CD133 (Kurtosis: 9.19)
CD19 (Kurtosis: 4.59)
CD22 (Kurtosis: 7.50)
CD11b (Kurtosis: 4.96)
CD4 (Kurtosis: 5.84)
CD8 (Kurtosis: 4.75)
CD34 (Kurtosis: 16.60)
Flt3 (Kurtosis: 85.58)
CD20 (Kurtosis: 10.44)
CXCR4 (Kurtosis: 3.94)
CD235ab (Kurtosis: 13.44)
CD45 (Kurtosis: 5.25)
CD123 (Kurtosis: 18.36)
CD321 (Kurtosis: 2.91)
CD14 (Kurtosis: 23.06)
CD33 (Kurtosis: 10.97)
CD47 (Kurtosis: 2.94)
CD11c (Kurtosis: 5.12)
CD7 (Kurtosis: 4.89)
CD15 (Kurtosis: 4.50)
CD16 (Kurtosis: 42.29)
CD44 (Kurtosis: 2.92)
CD38 (Kurtosis: 3.52)
CD13 (Kurtosis: 10.64)
CD3 (Kurtosis: 1.26)
CD61 (Kurtosis: 34.88)
CD117 (Kurtosis: 26.38)
CD49d (Kurtosis: 3.47)
HLA-DR (Kurtosis: 2.31)
CD64 (Kurtosis: 4.91)
CD41 (Kurtosis: 41.52)
Viability (Kurtosis: 3.16)
event_number (Kurtosis: 1.71)
label (Kurtosis: nan)
individual (Kurtosis: 1.96)
Density
1e−6

##T-SNE Visualization

```
[ ]: import tensorflow as tf
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import numpy as np

# Load the MNIST dataset
(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.
 ↪mnist.load_data()
train_images = train_images.astype('float32') / 255.0
test_images = test_images.astype('float32') / 255.0

# Flatten the images and take a subset
n_samples = 1000
train_images_flat = train_images[:n_samples].reshape(n_samples, -1)
train_labels_subset = train_labels[:n_samples]

# Perform t-SNE
tsne = TSNE(n_components=2, random_state=42, perplexity=30)
train_images_embedded = tsne.fit_transform(train_images_flat)

# Plot the t-SNE results
plt.figure(figsize=(10, 8))
scatter = plt.scatter(train_images_embedded[:, 0], train_images_embedded[:, 1],␣
 ↪c=train_labels_subset, cmap='tab10', alpha=0.5)
plt.colorbar(scatter, label='Digit Label')
plt.title('t-SNE Visualization of MNIST Digits')
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')
plt.show()
```

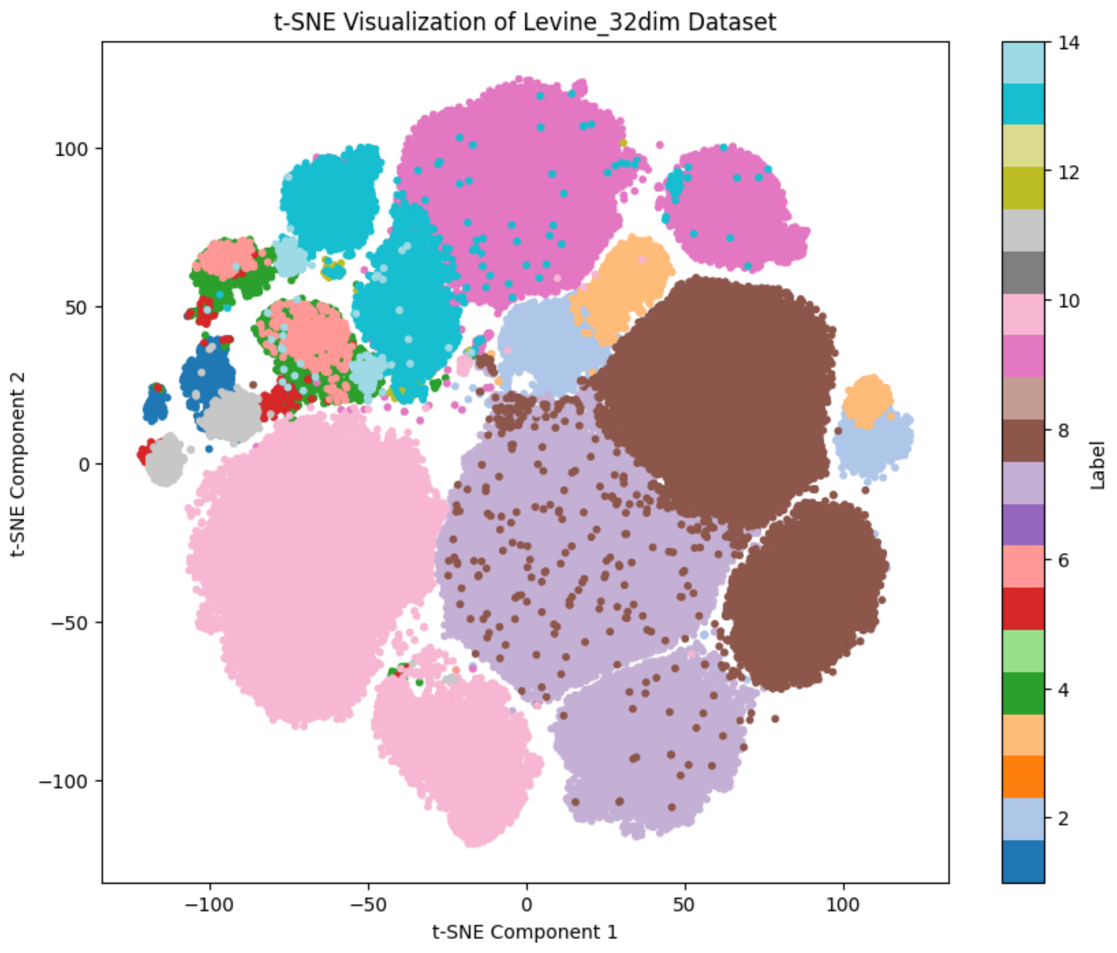
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz
11490434/11490434             0s
0us/step
```



##t-SNE Visualization for Dimensionality Reduction

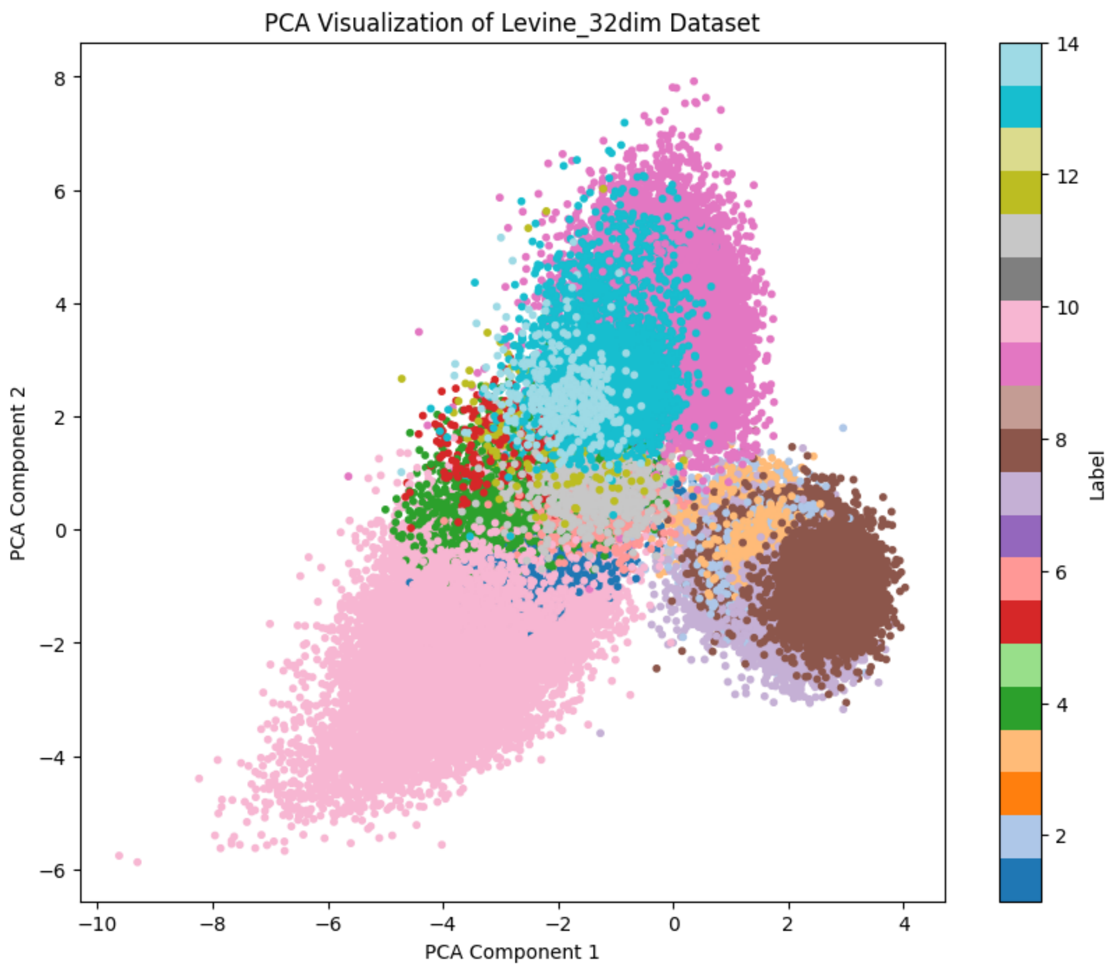
```
[ ]: import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

# Load the dataset
# data = pd.read_csv('/content/Levine_32dim.fcs.csv')

# Exclude the specified columns
exclude_columns = ['Event', 'Time', 'Cell_length', 'file_number',␣
  ↪'event_number', 'label', 'individual']
data_filtered = data.drop(columns=exclude_columns)

# Standardize the data (z-score normalization)
scaler = StandardScaler()
```

```
data_standardized = scaler.fit_transform(data_filtered)

# Perform t-SNE
tsne = TSNE(n_components=2, random_state=42, perplexity=30)  # You can adjust␣
↪perplexity as needed
tsne_results = tsne.fit_transform(data_standardized)

# Add the t-SNE results to the original data for visualization
data['t-SNE Component 1'] = tsne_results[:, 0]
data['t-SNE Component 2'] = tsne_results[:, 1]

# Plot the t-SNE visualization
plt.figure(figsize=(10, 8))
scatter = plt.scatter(data['t-SNE Component 1'], data['t-SNE Component 2'],␣
↪c=data['label'], cmap='tab20', s=10)
plt.colorbar(scatter, label='Label')
plt.title('t-SNE Visualization of Levine_32dim Dataset')
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')
plt.show()
```



##Principal Component Analysis (PCA) for Dimensionality Reduction

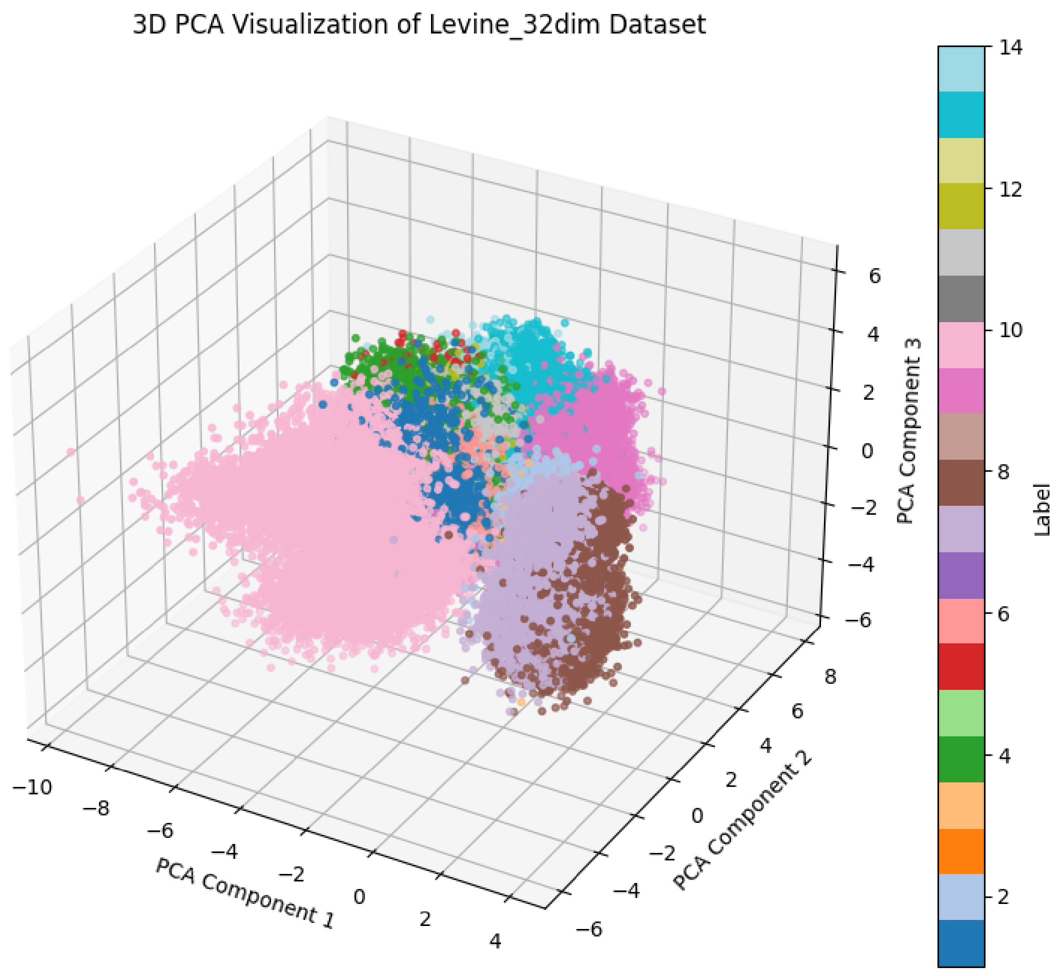
```
[ ]: import pandas as pd
     from sklearn.preprocessing import StandardScaler
     from sklearn.decomposition import PCA
     import matplotlib.pyplot as plt

     # Load the dataset
     # data = pd.read_csv('/content/Levine_32dim.fcs.csv')

     # Exclude the specified columns
     exclude_columns = ['Event', 'Time', 'Cell_length', 'file_number',␣
      ↪'event_number', 'label', 'individual']
     data_filtered = data.drop(columns=exclude_columns)

     # Standardize the data (z-score normalization)
     scaler = StandardScaler()
     data_standardized = scaler.fit_transform(data_filtered)

     # Perform PCA
     pca = PCA(n_components=2)  # Reduce to 2 dimensions for visualization
     pca_result = pca.fit_transform(data_standardized)

     # Add the PCA results to the original data for visualization
     data['PCA Component 1'] = pca_result[:, 0]
     data['PCA Component 2'] = pca_result[:, 1]

     # Plot the PCA results
     plt.figure(figsize=(10, 8))
     scatter = plt.scatter(data['PCA Component 1'], data['PCA Component 2'],␣
      ↪c=data['label'], cmap='tab20', s=10)
     plt.colorbar(scatter, label='Label')
     plt.title('PCA Visualization of Levine_32dim Dataset')
     plt.xlabel('PCA Component 1')
     plt.ylabel('PCA Component 2')
     plt.show()
```



##3D PCA graph

```
[ ]: import pandas as pd
     from sklearn.preprocessing import StandardScaler
     from sklearn.decomposition import PCA
     import matplotlib.pyplot as plt
     from mpl_toolkits.mplot3d import Axes3D  # Importing 3D plotting

     # Load the dataset
     # data = pd.read_csv('/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv')

     # Exclude the specified columns
     exclude_columns = ['Event', 'Time', 'Cell_length', 'file_number',␣
      ↪'event_number', 'label', 'individual']
     data_filtered = data.drop(columns=exclude_columns)

     # Standardize the data (z-score normalization)
```

```
scaler = StandardScaler()
data_standardized = scaler.fit_transform(data_filtered)

# Perform PCA
pca = PCA(n_components=3)  # Reduce to 3 dimensions for 3D visualization
pca_result = pca.fit_transform(data_standardized)

# Add the PCA results to the original data for visualization
data['PCA Component 1'] = pca_result[:, 0]
data['PCA Component 2'] = pca_result[:, 1]
data['PCA Component 3'] = pca_result[:, 2]

# Plot the PCA results in 3D
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Create a 3D scatter plot
scatter = ax.scatter(data['PCA Component 1'], data['PCA Component 2'],␣
 ↪data['PCA Component 3'],
                     c=data['label'], cmap='tab20', s=10)

# Add color bar and labels
plt.colorbar(scatter, label='Label')
ax.set_title('3D PCA Visualization of Levine_32dim Dataset')
ax.set_xlabel('PCA Component 1')
ax.set_ylabel('PCA Component 2')
ax.set_zlabel('PCA Component 3')

# Show the plot
plt.show()
```



##Variance, Cumulative Proportion, and Standard Deviation Analysis

```
[ ]: import pandas as pd
     from sklearn.preprocessing import StandardScaler
     from sklearn.decomposition import PCA

     # Load the dataset
     # data = pd.read_csv('/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv')

     # Exclude the specified columns
     exclude_columns = ['Event', 'Time', 'Cell_length', 'file_number',␣
       ↪'event_number', 'label', 'individual']
     data_filtered = data.drop(columns=exclude_columns)

     # Standardize the data (z-score normalization)
     scaler = StandardScaler()
     data_standardized = scaler.fit_transform(data_filtered)
```

```
# Perform PCA
pca = PCA(n_components=4)  # Use 4 principal components
pca.fit(data_standardized)

# Extract the required information
explained_variance = pca.explained_variance_ratio_
cumulative_variance = explained_variance.cumsum()
standard_deviation = pca.singular_values_ / (len(data_standardized) - 1)**0.5

# Create a DataFrame for the output
pca_summary = pd.DataFrame({
    'PC1': [standard_deviation[0], explained_variance[0],␣
 ↪cumulative_variance[0]],
    'PC2': [standard_deviation[1], explained_variance[1],␣
 ↪cumulative_variance[1]],
    'PC3': [standard_deviation[2], explained_variance[2],␣
 ↪cumulative_variance[2]],
    'PC4': [standard_deviation[3], explained_variance[3],␣
 ↪cumulative_variance[3]]
}, index=['Standard Deviation', 'Proportion of Variance', 'Cumulative␣
 ↪Proportion'])

# Round the numbers for better readability
pca_summary = pca_summary.map(lambda x: f'{x:.4f}')

# Apply styles to the DataFrame
styled_summary = (pca_summary.style
                  .set_caption("PCA Summary")
                  .set_table_styles(
                      [{'selector': 'caption', 'props': [('font-size', '16px'),␣
 ↪('color', 'black'), ('font-weight', 'bold')]}]
                  )
                  .background_gradient(cmap='coolwarm', axis=None)
                  .set_properties(**{'text-align': 'center'})
)

# Hiding the index column manually (workaround)
styled_summary.set_table_styles({
    'index': [{'selector': '', 'props': 'display:none;'}]  # Hides the index␣
 ↪column
})

# Display the styled DataFrame
styled_summary
```

[ ]: <pandas.io.formats.style.Styler at 0x782a6f557880>

[ ]:
```
# Separate labeled and unlabeled data based on non-NaN and NaN values in the␣
  ↪'label' column
df_labeled = df[df['label'].notnull()]
df_unlabeled = df[df['label'].isnull()]

# Print the shapes of labeled and unlabeled data
print("Labeled Data Shape:", df_labeled.shape)
print("Unlabeled Data Shape:", df_unlabeled.shape)
```

```
Labeled Data Shape: (104184, 37)
Unlabeled Data Shape: (161443, 37)
```

##**Binary Masking**

[ ]:
```
import numpy as np
import pandas as pd

# Set a random seed for reproducibility
np.random.seed(42)

# Create a sample DataFrame called 'demodata' for demonstration
demodata = pd.DataFrame({
    'column1': [5, 12, 18, 7],
    'column2': [10, 20, 15, 30],
    'column3': [25, 35, 40, 45]
})

# Define the probability of masking (e.g., 0.3 means a 30% chance each element␣
  ↪will be masked)
p_m = 0.3

# Convert 'demodata' to a NumPy array for masking
data_array = demodata.values

# Generate a binary mask based on the probability, where 1 = not masked, 0 =␣
  ↪masked
mask = np.random.binomial(1, 1 - p_m, data_array.shape)  # Reverse probability␣
  ↪for desired 1/0 output

# Convert to a DataFrame for easier analysis
binary_mask_df = pd.DataFrame(mask, columns=demodata.columns)

print("Original DataFrame:\n", demodata)
print("\nBinary Mask DataFrame:\n", binary_mask_df)
```

```
Original DataFrame:
    column1  column2  column3
0        5       10       25
1       12       20       35
2       18       15       40
3        7       30       45

Binary Mask DataFrame:
    column1  column2  column3
0        1        0        0
1        1        1        1
2        1        0        1
3        0        1        0
```

##**Random Shuffling of Data**

```
[ ]: import numpy as np
     import pandas as pd

     # Create a sample DataFrame called 'demodata' for demonstration
     demodata = pd.DataFrame({
         'column1': [5, 12, 18, 7],
         'column2': [10, 20, 15, 30],
         'column3': [25, 35, 40, 45]
     })

     # Shuffle each column in the DataFrame independently
     shuffled_demodata = demodata.apply(lambda col: np.random.permutation(col))

     print("Original DataFrame:\n", demodata)
     print("\nShuffled DataFrame:\n", shuffled_demodata)
```

```
Original DataFrame:
    column1  column2  column3
0        5       10       25
1       12       20       35
2       18       15       40
3        7       30       45

Shuffled DataFrame:
    column1  column2  column3
0        5       30       25
1       18       15       40
2       12       10       35
3        7       20       45
```

##**Corrupted DataFrame** Formula = (x.values * (1 - m) + x_shuffled.values * m)

```
[ ]: import numpy as np
     import pandas as pd

     # Create a sample DataFrame called 'x' (original data)
     x = pd.DataFrame({
         'column1': [5, 12, 18, 7],
         'column2': [10, 20, 15, 30],
         'column3': [25, 35, 40, 45]
     })

     # Define the probability of masking (e.g., 0.3 means a 30% chance each element␣
      ↪will be masked)
     p_m = 0.3

     # Generate a binary mask matrix 'm'
     m = np.random.binomial(1, 1 - p_m, x.shape)
     binary_mask_df = pd.DataFrame(m, columns=x.columns)

     # Shuffle each column in 'x' independently to create 'x_shuffled'
     x_shuffled = x.apply(lambda col: np.random.permutation(col))

     # Calculate the corrupted DataFrame 'x_corrupted' using the formula
     x_corrupted_array = x.values * (1 - m) + x_shuffled.values * m
     x_corrupted = pd.DataFrame(x_corrupted_array, columns=x.columns)

     # Display results
     print("Original DataFrame (x):\n", x)
     print("\nBinary Mask DataFrame (m):\n", binary_mask_df)
     print("\nShuffled DataFrame (x_shuffled):\n", x_shuffled)
     print("\nCorrupted DataFrame (x_corrupted):\n", x_corrupted)
```

```
Original DataFrame (x):
    column1  column2  column3
0        5       10       25
1       12       20       35
2       18       15       40
3        7       30       45

Binary Mask DataFrame (m):
    column1  column2  column3
0        1        1        1
1        1        1        0
2        1        1        1
3        1        0        1

Shuffled DataFrame (x_shuffled):
    column1  column2  column3
```

```
0       12       20       40
1        7       15       45
2       18       30       25
3        5       10       35

Corrupted DataFrame (x_corrupted):
    column1  column2  column3
0       12       20       40
1        7       15       35
2       18       30       25
3        5       30       35
```

[ ]:
```
# Separate labeled and unlabeled data based on non-NaN and NaN values in the␣
  ↪'label' column
df_labeled = df[df['label'].notnull()]
df_unlabeled = df[df['label'].isnull()]

# Print the shapes of labeled and unlabeled data
print("Labeled Data Shape:", df_labeled.shape)
print("Unlabeled Data Shape:", df_unlabeled.shape)
```

```
Labeled Data Shape: (104184, 37)
Unlabeled Data Shape: (161443, 37)
```

##Applying Binary Mask, Shuffling, and Handling Corrupted Data on the Original Dataset

[ ]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Load the dataset
# data = pd.read_csv('/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv')
data=df
# Exclude the specified columns
exclude_columns = [ 'Cell_length','label']
data_filtered = data.drop(columns=exclude_columns)

# Set the probability of masking
p_m = 0.3

# Generate a binary mask matrix 'm'
m = np.random.binomial(1, 1 - p_m, data_filtered.shape)
binary_mask_df = pd.DataFrame(m, columns=data_filtered.columns)

# Shuffle each column in 'data_filtered' independently to create 'data_shuffled'
data_shuffled = data_filtered.apply(lambda col: np.random.permutation(col))
```

```
# Calculate the corrupted DataFrame 'data_corrupted' using the formula
data_corrupted_array = data_filtered.values * (1 - m) + data_shuffled.values * m
data_corrupted = pd.DataFrame(data_corrupted_array, columns=data_filtered.
 ↪columns)

# Display results
print("Binary Mask DataFrame (m):\n", binary_mask_df)
print("\nShuffled DataFrame (data_shuffled):\n", data_shuffled)
print("\nCorrupted DataFrame (data_corrupted):\n", data_corrupted)
```

```
Binary Mask DataFrame (m):
         DNA1  DNA2  CD45RA  CD133  CD19  CD22  CD11b  CD4  CD8  CD34  …  \
0          0     1       1      1     1     0      1    1    1     0  …
1          1     1       1      0     1     1      1    0    1     1  …
2          1     1       1      1     0     1      1    1    1     1  …
3          0     1       0      1     0     1      1    0    0     1  …
4          1     1       1      0     1     0      1    1    1     0  …
…        …    …     …      …    …    …    …     …   …   …    …  …
265622     1     1       1      0     1     1      1    1    1     1  …
265623     1     0       0      1     1     0      1    0    0     1  …
265624     1     1       1      1     0     0      1    1    0     0  …
265625     1     1       0      1     1     1      0    1    1     0  …
265626     1     1       1      0     1     0      0    1    0     1  …

        CD38  CD13  CD3  CD61  CD117  CD49d  HLA-DR  CD64  CD41  Viability
0          1     1    0     1      1      1       1     0     1          0
1          1     1    1     0      1      1       1     1     1          1
2          1     1    1     1      1      1       1     1     1          1
3          1     1    1     1      0      1       1     1     1          0
4          0     1    1     0      1      1       1     1     1          1
…        …    …   …   …    …     …     …    …    …         …
265622     1     1    1     0      1      1       1     1     0          0
265623     1     1    0     1      0      0       1     1     1          0
265624     1     1    0     0      1      1       1     1     0          1
265625     1     0    1     1      1      1       1     1     0          1
265626     1     1    1     1      0      1       1     1     1          1

[265627 rows x 35 columns]

Shuffled DataFrame (data_shuffled):
             DNA1      DNA2    CD45RA     CD133      CD19      CD22     CD11b  \
0       3.832251  7.110342  0.056787 -0.047125 -0.019157  0.081593  0.861457
1       6.850977  6.938741  0.280154 -0.052424  3.194934 -0.025818  0.474615
2       3.898148  4.800649  0.803446 -0.005483 -0.039994 -0.010834  0.791976
3       3.983141  7.101164  0.895017 -0.024665 -0.004068  0.631010  2.461230
4       6.843494  4.833479  1.347323  0.181406 -0.011673  0.344872 -0.049675
…            …       …       …       …       …       …       …
```

```
265622  4.143692  4.165534  1.921162  0.503055  0.088089 -0.024726  1.640322
265623  3.520940  4.925819  0.844817 -0.019637 -0.042965  1.193971  1.318320
265624  3.374421  3.725582  1.700395 -0.035191 -0.000930  0.782723  0.125876
265625  6.266251  4.144004  0.527227 -0.009675  0.801529 -0.028198  0.137037
265626  3.963936  6.973443  0.149522 -0.004637  0.191710 -0.033149  2.969673

             CD4       CD8      CD34  …      CD38      CD13       CD3  \
0       0.044337  0.528761  0.303096  …  1.472629  0.141334  0.333051
1      -0.031446 -0.050283 -0.022875  …  2.356335  0.361061  5.299439
2       0.671094  0.335853  0.782878  …  3.766334  0.572558  0.533445
3      -0.031273 -0.030742  0.080948  …  0.246523  0.234318  5.297177
4      -0.015428  3.252921  0.628982  …  4.098167  0.141950  0.528495
…            …         …         …  …         …         …         …
265622  0.073016  0.266265 -0.019674  …  0.178304  0.110954  4.644802
265623  1.354430  0.258695  0.186385  …  1.646464  0.060831 -0.022053
265624  1.912192  0.015772  0.227752  …  0.082550  1.005197  0.180568
265625  0.465344 -0.000996  0.586408  …  1.920932  0.177863  0.181161
265626  0.022423 -0.008825  1.885905  …  1.724717  0.076569  0.080876

            CD61     CD117     CD49d    HLA-DR      CD64      CD41  Viability
0      -0.008143  0.095703  0.577297  2.074943  0.123180  0.365936   0.258184
1       0.580516 -0.026543  0.909052 -0.047745 -0.018654  0.529259   0.043941
2      -0.021106 -0.018753  0.688829  0.949675  2.952665 -0.015633   0.273634
3       0.625687 -0.005148  1.904124 -0.025244  2.868251  0.052811   0.024663
4       0.339656  0.030624  0.722506  3.447628  0.054154 -0.024944   1.405620
…            …         …         …         …         …         …         …
265622 -0.017365 -0.031729  0.212918  4.578154  1.962263 -0.032345   0.556676
265623  0.014400 -0.018247  0.068592  3.515773  0.266746 -0.002856   0.280610
265624  0.018446  0.016299  0.415913  0.167136  0.451329 -0.054842   0.237244
265625  0.071217  0.232223  2.509537  0.057068  0.058362  0.010580   0.110044
265626  0.056771 -0.022872  0.559208  3.188042 -0.047085  0.031154   0.690313

[265627 rows x 35 columns]

Corrupted DataFrame (data_corrupted):
            DNA1      DNA2    CD45RA     CD133      CD19      CD22     CD11b  \
0       4.391057  7.110342  0.056787 -0.047125 -0.019157  0.066388  0.861457
1       6.850977  6.938741  0.280154 -0.038280  3.194934 -0.025818  0.474615
2       3.898148  4.800649  0.803446 -0.005483  0.073855 -0.010834  0.791976
3       4.255806  7.101164  0.433747 -0.024665 -0.017661  0.631010  2.461230
4       6.843494  4.833479  1.347323 -0.030297 -0.011673  0.495791 -0.049675
…            …         …         …         …         …         …         …
265622  4.143692  4.165534  1.921162 -0.019174  0.088089 -0.024726  1.640322
265623  3.520940  7.154026  0.116755 -0.019637 -0.042965 -0.035158  1.318320
265624  3.374421  3.725582  1.700395 -0.035191 -0.026111 -0.030837  0.125876
265625  6.266251  4.144004  0.288761 -0.009675  0.801529 -0.028198 -0.031787
265626  3.963936  6.973443  0.149522  0.128604  0.191710  0.109846  3.864711
```

```
             CD4       CD8      CD34  ...      CD38      CD13       CD3  \
0       0.044337  0.528761 -0.012805  ...  1.472629  0.141334 -0.032596
1      -0.035424 -0.050283 -0.022875  ...  2.356335  0.361061  5.299439
2       0.671094  0.335853  0.782878  ...  3.766334  0.572558  0.533445
3      -0.019066  0.056109  0.080948  ...  0.246523  0.234318  5.297177
4      -0.015428  3.252921 -0.038895  ...  3.711521  0.141950  0.528495
...          ...       ...       ...  ...       ...       ...       ...
265622  0.073016  0.266265 -0.019674  ...  0.178304  0.110954  4.644802
265623  0.970120 -0.023903  0.186385  ...  1.646464  0.060831  5.112841
265624  1.912192  0.257884  0.107905  ...  0.082550  1.005197  5.098065
265625  0.465344 -0.000996  1.678589  ...  1.920932  0.275652  0.181161
265626  0.022423  0.113039  1.885905  ...  1.724717  0.076569  0.080876

            CD61     CD117     CD49d    HLA-DR      CD64      CD41  Viability
0      -0.008143  0.095703  0.577297  2.074943 -0.005376  0.365936   0.648429
1       1.258437 -0.026543  0.909052 -0.047745 -0.018654  0.529259   0.043941
2      -0.021106 -0.018753  0.688829  0.949675  2.952665 -0.015633   0.273634
3       0.625687  0.066470  1.904124 -0.025244  2.868251  0.052811  -0.026523
4       0.168609  0.030624  0.722506  3.447628  0.054154 -0.024944   1.405620
...          ...       ...       ...       ...       ...       ...        ...
265622  0.861068 -0.031729  0.212918  4.578154  1.962263 -0.027971   0.236957
265623  0.014400  0.143869  1.269464  3.515773  0.266746 -0.002856  -0.003500
265624 -0.008680  0.016299  0.415913  0.167136  0.451329 -0.042602   0.237244
265625  0.071217  0.232223  2.509537  0.057068  0.058362  0.192786   0.110044
265626  0.056771  0.080195  0.559208  3.188042 -0.047085  0.031154   0.690313

[265627 rows x 35 columns]
```

##**New Masking** Formula = (mask_new = 1 * (data_filtered != data_corrupted))

```
[ ]: import numpy as np
     import pandas as pd
     from sklearn.preprocessing import StandardScaler

     # Load the dataset
     # data = pd.read_csv('/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv')

     # Exclude the specified columns
     exclude_columns = ['Event', 'Time', 'Cell_length', 'file_number',␣
      ↪'event_number', 'label', 'individual']
     data_filtered = data.drop(columns=exclude_columns)

     # Set the probability of masking
     p_m = 0.3

     # Generate a binary mask matrix 'm' (changes every run)
     m = np.random.binomial(1, 1 - p_m, data_filtered.shape)
     binary_mask_df = pd.DataFrame(m, columns=data_filtered.columns)
```

```
# Shuffle each column in 'data_filtered' independently to create␣
↪'data_shuffled' (changes every run)
data_shuffled = data_filtered.apply(lambda col: np.random.permutation(col))

# Calculate the corrupted DataFrame 'data_corrupted' using the formula
data_corrupted_array = data_filtered.values * (1 - m) + data_shuffled.values * m
data_corrupted = pd.DataFrame(data_corrupted_array, columns=data_filtered.
↪columns)

# Generate mask_new to indicate differences between original and corrupted data
mask_new = 1 * (data_filtered != data_corrupted)

# Print only the new mask matrix
print("New Mask Matrix (mask_new):\n", mask_new)
```

```
New Mask Matrix (mask_new):
          DNA1  DNA2  CD45RA  CD133  CD19  CD22  CD11b  CD4  CD8  CD34  …  \
0           1     1       1      1     1     1      0    1    1     1  …
1           1     1       1      1     1     1      1    0    0     1  …
2           1     1       1      1     1     1      1    1    1     1  …
3           0     1       1      0     1     1      1    0    0     0  …
4           0     1       1      1     1     1      1    1    1     0  …
…         …   …     …    …   …   …    …   …   …   …   …  …
265622      1     1       1      0     0     1      1    1    1     1  …
265623      1     1       0      1     0     1      1    0    1     1  …
265624      1     1       0      1     1     1      1    0    1     1  …
265625      1     1       1      1     1     1      1    1    1     0  …
265626      1     1       1      1     1     1      0    1    0     0  …

        CD38  CD13  CD3  CD61  CD117  CD49d  HLA-DR  CD64  CD41  Viability
0          1     0    1     1      1      1       0     0     1          1
1          1     0    1     0      1      0       1     1     0          1
2          0     1    1     1      1      1       1     0     0          0
3          1     1    0     1      1      1       1     1     1          1
4          1     1    0     0      0      1       1     0     1          0
…        …   …   …   …    …    …    …    …   …   …        …
265622     0     1    1     1      0      0       1     1     1          1
265623     1     1    1     0      1      1       0     0     0          1
265624     1     0    1     1      1      1       1     1     0          0
265625     1     1    0     1      0      1       1     1     1          0
265626     1     0    1     1      1      1       0     1     1          1

[265627 rows x 35 columns]
```

##Separating Features and Labels in Unlabeled Data

```
[ ]: import numpy as np
     import pandas as pd

     # Load the dataset
     # df = pd.read_csv('/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv')

     # Define the target column used for labeling
     label_column = 'label'
     df=data
     # Separate labeled and unlabeled data using label_df
     label_df = df[df[label_column].notnull()]  # labeled data
     unlabeled_df = df[df[label_column].isnull()]  # unlabeled data

     # Split features and labels for labeled data
     x_labeled = label_df.drop(columns=[label_column])
     y_labeled = label_df[label_column]

     # Split features and labels for unlabeled data
     x_unlabeled = unlabeled_df.drop(columns=[label_column])
     y_unlabeled = unlabeled_df[label_column]

     # Display results
     print("Labeled Features (x_labeled):\n", x_labeled)
     print("\nLabeled Labels (y_labeled):\n", y_labeled)
     print("\nUnlabeled Features (x_unlabeled):\n", x_unlabeled)
     print("\nUnlabeled Labels (y_unlabeled):\n", y_unlabeled)
```

```
Labeled Features (x_labeled):
          Event       Time  Cell_length      DNA1      DNA2    CD45RA  \
0            1    2693.00           22  4.391057  4.617262  0.162691
1            2    3736.00           35  4.340481  4.816692  0.701349
2            3    7015.00           32  3.838727  4.386369  0.603568
3            4    7099.00           29  4.255806  4.830048  0.433747
4            5    7700.00           25  3.976909  4.506433 -0.008809
…          …        …           …        …         …         …
104179  104180  641812.44           58  6.827981  7.249403 -0.000106
104180  104181  653387.44           55  6.683204  7.166172  0.692668
104181  104182  671024.44           40  6.911546  7.152603 -0.036795
104182  104183  680006.44           48  6.700332  7.100771  0.308817
104183  104184  687494.44           64  6.559460  7.080928  0.519572

           CD133      CD19      CD22     CD11b  …      CD61     CD117  \
0      -0.029585 -0.006696  0.066388 -0.009184  … -0.002936  0.053050
1      -0.038280 -0.016654  0.074409  0.808031  …  1.258437  0.089660
2      -0.032216  0.073855 -0.042977 -0.001881  …  0.257137  0.046222
3      -0.027611 -0.017661 -0.044072  0.733698  … -0.041140  0.066470
4      -0.030297  0.080423  0.495791  1.107627  …  0.168609 -0.006223
```

```
…           …        …        …        …  …        …        …
104179 -0.030641  1.432347 -0.044946 -0.016534  …  0.188846 -0.002144
104180 -0.037335  1.639063  0.286325 -0.036985  … -0.029213 -0.031301
104181 -0.014477  1.637975 -0.021794 -0.020169  … -0.015220 -0.034755
104182  0.075762  1.455129  0.042576 -0.049737  … -0.016644 -0.047522
104183  0.097257  1.346523  0.279473 -0.021585  … -0.051973 -0.017015

           CD49d    HLA-DR      CD64      CD41  Viability  file_number  \
0       0.853505  1.664480 -0.005376 -0.001961   0.648429     3.627711
1       0.197818  0.491592  0.144814  0.868014   0.561384     3.627711
2       2.586670  1.308337 -0.010961 -0.010413   0.643337     3.627711
3       1.338669  0.140523 -0.013449 -0.026039  -0.026523     3.627711
4       0.180924  0.197332  0.076167 -0.040488   0.283287     3.627711
…           …        …        …        …         …           …
104179  1.115652  2.373524 -0.004620 -0.051592   0.157816     3.669327
104180  1.653418  4.367032  0.062683  0.158656   0.025255     3.669327
104181  1.083173  3.541526  0.110382  0.108349  -0.043739     3.669327
104182  0.432565  3.882030  0.058852  0.185295   0.204898     3.669327
104183  0.263008  4.332834 -0.017214  0.130106   0.023135     3.669327

        event_number  individual
0                307           1
1                545           1
2               1726           1
3               1766           1
4               2031           1
…             …          …
104179        100344           2
104180        100892           2
104181        101558           2
104182        101842           2
104183        102112           2

[104184 rows x 41 columns]

Labeled Labels (y_labeled):
 0          1.0
1          1.0
2          1.0
3          1.0
4          1.0
         …
104179    14.0
104180    14.0
104181    14.0
104182    14.0
104183    14.0
Name: label, Length: 104184, dtype: float64
```

```
Unlabeled Features (x_unlabeled):
         Event       Time  Cell_length      DNA1      DNA2    CD45RA  \
104184  104185      40.00           25  4.203073  4.837565  0.095543   
104185  104186     176.00           34  4.042991  4.808275  0.035310   
104186  104187     189.00           37  4.233125  4.922201  0.415954   
104187  104188     193.00           26  3.997143  4.685426 -0.038565   
104188  104189     204.00           20  4.115830  4.893428  0.177246   
…          …          …          …         …         …         …   
265622  265623  707951.44           41  6.826629  7.133022  1.474081   
265623  265624  708145.44           45  6.787791  7.154026  0.116755   
265624  265625  708398.44           41  6.889866  7.141219  0.684921   
265625  265626  708585.44           39  6.865218  7.144353  0.288761   
265626  265627  709122.44           41  6.887820  7.127359  0.360753   

           CD133      CD19      CD22     CD11b  …      CD61     CD117  \
104184 -0.027206  0.172384 -0.001950  0.505713  …  3.029787 -0.010093   
104185 -0.013869 -0.043922 -0.001871  0.180261  … -0.017628  0.346248   
104186  0.412757  0.431715 -0.025619  0.491190  …  0.000544  0.691393   
104187  0.125894  0.191383 -0.026497  0.342190  … -0.012887  0.033096   
104188  0.171916  0.028568 -0.029751  2.480689  … -0.015719 -0.043689   
…            …         …         …         …  …         …         …   
265622 -0.019174 -0.055620 -0.007261  0.063395  …  0.861068 -0.011105   
265623 -0.056213 -0.008864 -0.035158 -0.041845  …  0.565170  0.143869   
265624 -0.006264 -0.026111 -0.030837 -0.034641  … -0.008680  0.087102   
265625 -0.011310 -0.048786  0.073983 -0.031787  … -0.029347 -0.047971   
265626  0.128604 -0.006934  0.109846  3.864711  … -0.023831  0.080195   

           CD49d    HLA-DR      CD64      CD41  Viability  file_number  \
104184  0.387121  2.859639  2.709532  1.208795   0.102978     3.627711   
104185  0.089940 -0.017702  0.045091 -0.022009   0.092770     3.627711   
104186  2.996583  5.812406  1.713608  0.479122   1.888485     3.627711   
104187 -0.029722 -0.031126 -0.020739 -0.014693   0.067437     3.627711   
104188  0.027586  2.543139  3.323810 -0.002918   0.109243     3.627711   
…            …         …         …         …          …            …   
265622  0.533736  0.123758 -0.042495 -0.027971   0.236957     3.669327   
265623  1.269464  0.047215 -0.008000 -0.025811  -0.003500     3.669327   
265624 -0.055912  0.501536  0.053884 -0.042602   0.107206     3.669327   
265625  0.101955  6.200001  0.296877  0.192786   0.620872     3.669327   
265626  0.037962  3.675123 -0.000878 -0.052526   0.310466     3.669327   

        event_number  individual  
104184             1           1  
104185             6           1  
104186             7           1  
104187             8           1  
104188             9           1  
…                …           …  
```

```
265622        102686          2
265623        102690          2
265624        102701          2
265625        102706          2
265626        102720          2

[161443 rows x 41 columns]

Unlabeled Labels (y_unlabeled):
 104184   NaN
104185   NaN
104186   NaN
104187   NaN
104188   NaN
          ..
265622   NaN
265623   NaN
265624   NaN
265625   NaN
265626   NaN
Name: label, Length: 161443, dtype: float64
```

[ ]:
```python
from sklearn.model_selection import train_test_split
# Separate labeled and unlabeled data
df_labeled = df[df['label'].notnull()]  # Labeled data
df_unlabeled = df[df['label'].isnull()]  # Unlabeled data

# Separate features and target for labeled data
x_labeled = df_labeled.drop(columns=['label'])  # Features
y_labeled = df_labeled['label']                  # Target

# Separate features for unlabeled data
x_unlabeled = df_unlabeled.drop(columns=['label'])  # Features (no labels)

# Split the labeled data into training and testing sets (e.g., 70% train, 30%␣
 ↪test)
x_train, x_test, y_train, y_test = train_test_split(x_labeled, y_labeled,␣
 ↪test_size=0.3, random_state=42)

print("\nTraining Features (x_train):\n", x_train.head())
print("\nTraining Labels (y_train):\n", y_train.head())
print("\nTesting Features (x_test):\n", x_test.head())
print("\nTesting Labels (y_test):\n", y_test.head())
```

```

Training Features (x_train):
        Event       Time  Cell_length      DNA1      DNA2    CD45RA     CD133  \
64113  64114  401196.00           25  3.899656  4.594272  0.976652  0.302811
```

```
82744  82745  502826.44           31  6.592998  6.901888  0.431481 -0.052898
24294  24295  488377.00           41  3.543583  4.467671  0.377192  0.219081
7820    7821  225689.00           38  4.305227  4.881685  0.199351  0.100678
43295  43296  153333.00           26  4.159271  4.861015  0.831285  0.191518

           CD19      CD22     CD11b  …       CD61     CD117     CD49d  \
64113  0.154761 -0.011676  3.180236  …  0.051464 -0.003680  1.260410
82744 -0.037690 -0.029715 -0.040846  … -0.036430  0.021689  0.034946
24294  0.245478  0.193328  0.075123  …  1.003383  0.406137  1.928676
7820  -0.025812 -0.002898  1.437247  … -0.007282  1.421540  1.443145
43295  2.002712  3.387782  0.179219  … -0.040754  0.060944  1.294561

         HLA-DR      CD64      CD41  Viability  file_number  event_number  \
64113  0.700093  2.355886  0.125409   0.840205     3.627711        318320
82744 -0.055651 -0.023248 -0.054842  -0.009329     3.669327         80934
24294 -0.046849  0.229309  0.937020   1.231347     3.627711        366690
7820   2.461705  0.528679  0.072205   0.892480     3.627711        203131
43295  3.085858 -0.014128  0.479256   2.269233     3.627711        152117

       individual
64113           1
82744           2
24294           1
7820            1
43295           1

[5 rows x 41 columns]

Training Labels (y_train):
 64113    10.0
82744     7.0
24294     7.0
7820      6.0
43295     9.0
Name: label, dtype: float64

Testing Features (x_test):
        Event      Time  Cell_length      DNA1      DNA2    CD45RA     CD133  \
60544  60545  278003.0           49  3.618797  4.144135  0.198186  0.000282
50673  50674  490341.0           27  3.660988  4.497041  1.272625  0.129642
50682  50683  490912.0           23  3.854865  4.663734  1.527763  0.151383
1761    1762  170466.0           17  3.716473  4.465312  0.375236 -0.037150
98760  98761  423490.0           32  6.826030  7.007709  0.223441 -0.048813

           CD19      CD22     CD11b  …       CD61     CD117     CD49d  \
60544  0.253703 -0.018972  2.665005  …  0.307357  0.208639  2.039954
50673  3.054480  2.493220  0.189975  …  0.084448  0.033192  0.004637
50682  2.361353  2.281009  0.528589  … -0.041903 -0.026017  0.109363
```

```
1761   -0.035385  0.127904  0.415204  … -0.001024 -0.017034  0.023385
98760  -0.018816 -0.045954  4.067125  … -0.029816 -0.046020  0.140410

         HLA-DR      CD64      CD41  Viability  file_number  event_number  \
60544  2.847283  2.798986  1.090235   1.005784     3.627711        237532
50673  4.488360  0.866820 -0.002174   0.917810     3.627711        367731
50682  2.328828 -0.008223 -0.018680   1.091297     3.627711        367970
1761   0.120367  0.472159 -0.014919   0.620643     3.627711        164637
98760  0.735830  1.011186 -0.044875   0.149759     3.669327         62492

       individual
60544           1
50673           1
50682           1
1761            1
98760           2

[5 rows x 41 columns]

Testing Labels (y_test):
 60544    10.0
50673     9.0
50682     9.0
1761      2.0
98760    10.0
Name: label, dtype: float64
```

```
[ ]: # Separate labeled and unlabeled data based on non-NaN and NaN values in the␣
     ↪'label' column
     df_labeled = df[df['label'].notnull()]
     df_unlabeled = df[df['label'].isnull()]

     # Print the shapes of labeled and unlabeled data
     print("Labeled Data Shape:", df_labeled.shape)
     print("Unlabeled Data Shape:", df_unlabeled.shape)
```

```
Labeled Data Shape: (104184, 37)
Unlabeled Data Shape: (161443, 37)
```

##Splitting Labeled Dataset into Training and Testing Sets (70% Training, 30% Testing)**

```
[ ]: from sklearn.model_selection import train_test_split
     # Separate labeled and unlabeled data
     df_labeled = df[df['label'].notnull()]  # Labeled data
     df_unlabeled = df[df['label'].isnull()]  # Unlabeled data

     # Separate features and target for labeled data
```

```
x_labeled = df_labeled.drop(columns=['label'])  # Features
y_labeled = df_labeled['label']                  # Target

# Separate features for unlabeled data
x_unlabeled = df_unlabeled.drop(columns=['label'])  # Features (no labels)

# Split the labeled data into training and testing sets (e.g., 70% train, 30%␣
↪test)
x_train, x_test, y_train, y_test = train_test_split(x_labeled, y_labeled,␣
↪test_size=0.3, random_state=42)

print("\nTraining Features (x_train):\n", x_train.head())
print("\nTraining Labels (y_train):\n", y_train.head())
print("\nTesting Features (x_test):\n", x_test.head())
print("\nTesting Labels (y_test):\n", y_test.head())
```

```
Training Features (x_train):
        Cell_length      DNA1      DNA2    CD45RA     CD133      CD19  \
64113           25  3.899656  4.594272  0.976652  0.302811  0.154761
82744           31  6.592998  6.901888  0.431481 -0.052898 -0.037690
24294           41  3.543583  4.467671  0.377192  0.219081  0.245478
7820            38  4.305227  4.881685  0.199351  0.100678 -0.025812
43295           26  4.159271  4.861015  0.831285  0.191518  2.002712

           CD22     CD11b       CD4       CD8  …      CD38      CD13  \
64113 -0.011676  3.180236  1.465950  0.086209  …  1.563844  0.480488
82744 -0.029715 -0.040846  0.914311  0.022305  …  1.232765  0.100678
24294  0.193328  0.075123  0.936352 -0.044813  …  0.486930  0.046766
7820  -0.002898  1.437247 -0.013400 -0.001012  …  1.250272  0.731957
43295  3.387782  0.179219  0.115231 -0.010963  …  2.883403  0.345273

            CD3      CD61     CD117     CD49d    HLA-DR      CD64      CD41  \
64113  0.017010  0.051464 -0.003680  1.260410  0.700093  2.355886  0.125409
82744  5.722406 -0.036430  0.021689  0.034946 -0.055651 -0.023248 -0.054842
24294  4.061728  1.003383  0.406137  1.928676 -0.046849  0.229309  0.937020
7820   0.245939 -0.007282  1.421540  1.443145  2.461705  0.528679  0.072205
43295  0.226596 -0.040754  0.060944  1.294561  3.085858 -0.014128  0.479256

       Viability
64113   0.840205
82744  -0.009329
24294   1.231347
7820    0.892480
43295   2.269233

[5 rows x 36 columns]
```

```
Training Labels (y_train):
 64113    10.0
82744     7.0
24294     7.0
7820      6.0
43295     9.0
Name: label, dtype: float64

Testing Features (x_test):
       Cell_length      DNA1      DNA2    CD45RA     CD133      CD19  \
60544           49  3.618797  4.144135  0.198186  0.000282  0.253703
50673           27  3.660988  4.497041  1.272625  0.129642  3.054480
50682           23  3.854865  4.663734  1.527763  0.151383  2.361353
1761            17  3.716473  4.465312  0.375236 -0.037150 -0.035385
98760           32  6.826030  7.007709  0.223441 -0.048813 -0.018816

           CD22     CD11b       CD4       CD8  …       CD38      CD13  \
60544 -0.018972  2.665005  0.079150 -0.002045  …  2.479135  1.419488
50673  2.493220  0.189975 -0.024412  0.186744  …  2.212054 -0.020246
50682  2.281009  0.528589 -0.014516 -0.002732  …  0.787080 -0.010742
1761   0.127904  0.415204  0.226788  2.802413  …  0.042091 -0.018271
98760 -0.045954  4.067125  0.004401 -0.012083  …  1.382377  0.154702

            CD3      CD61     CD117     CD49d    HLA-DR      CD64      CD41  \
60544  0.643676  0.307357  0.208639  2.039954  2.847283  2.798986  1.090235
50673  0.054290  0.084448  0.033192  0.004637  4.488360  0.866820 -0.002174
50682  0.068448 -0.041903 -0.026017  0.109363  2.328828 -0.008223 -0.018680
1761  -0.039628 -0.001024 -0.017034  0.023385  0.120367  0.472159 -0.014919
98760  0.250393 -0.029816 -0.046020  0.140410  0.735830  1.011186 -0.044875

      Viability
60544  1.005784
50673  0.917810
50682  1.091297
1761   0.620643
98760  0.149759

[5 rows x 36 columns]

Testing Labels (y_test):
 60544    10.0
50673     9.0
50682     9.0
1761      2.0
98760    10.0
Name: label, dtype: float64
```

```
[ ]: from sklearn.preprocessing import StandardScaler

     # Initialize the scaler
     scaler = StandardScaler()

     # Fit and transform the unlabeled data
     x_unlabeled_scaled = scaler.fit_transform(x_unlabeled)

     # Convert back to a DataFrame if needed (optional, for better readability)
     x_unlabeled_scaled = pd.DataFrame(x_unlabeled_scaled, columns=x_unlabeled.
      ↪columns)


     from sklearn.model_selection import train_test_split
     df_labeled = df[df['label'].notnull()]  # Labeled data
     df_unlabeled = df[df['label'].isnull()]  # Unlabeled data

     # Separate features and target for labeled data
     X_labeled = df_labeled.drop(columns=['label'])  # Features
     y_labeled = df_labeled['label']                  # Target


     # Split the labeled data into training and testing sets (e.g., 70% train, 30%␣
      ↪test)
     X_train, X_test, y_train, y_test = train_test_split(X_labeled, y_labeled,␣
      ↪test_size=0.3, random_state=42)

     # Print the shapes of the training and testing sets
     print("Shape of Training Features (X_train):", X_train.shape)
     print("Shape of Training Labels (y_train):", y_train.shape)
     print("Shape of Testing Features (X_test):", X_test.shape)
```

```
Shape of Training Features (X_train): (72928, 36)
Shape of Training Labels (y_train): (72928,)
Shape of Testing Features (X_test): (31256, 36)
```

##**Logistic Regression and XGBoost Models**

```
[ ]: import numpy as np
     import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LogisticRegression
     from xgboost import XGBClassifier
     from sklearn.preprocessing import LabelEncoder, StandardScaler

     # Load the dataset
     # df = pd.read_csv('/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv')
     # Define the target column used for labeling
     label_column = 'label'

     # Separate labeled data
     label_df = df[df[label_column].notnull()]

     # Split features and labels for labeled data
     x_labeled = label_df.drop(columns=[label_column])
     y_labeled = label_df[label_column]

     # Encode labels if necessary (e.g., for non-numeric labels)
     label_encoder = LabelEncoder()
     y_labeled = label_encoder.fit_transform(y_labeled)

     # Split labeled data into training and testing sets (70%-30% split)
     x_train, x_test, y_train, y_test = train_test_split(x_labeled, y_labeled,␣
      ↪test_size=0.3, random_state=42)

     # Scale features for Logistic Regression and XGBoost
     scaler = StandardScaler()
     x_train_scaled = scaler.fit_transform(x_train)
     x_test_scaled = scaler.transform(x_test)

     # Logistic Regression Model with increased max_iter and scaled data
     logistic_model = LogisticRegression(max_iter=2000)
     logistic_model.fit(x_train_scaled, y_train)
     y_test_hat_logistic = logistic_model.predict_proba(x_test_scaled)

     # XGBoost Model (using scaled data)
     xgb_model = XGBClassifier(eval_metric='mlogloss')
     xgb_model.fit(x_train_scaled, y_train)  # Use scaled data for training
     y_test_hat_xgb = xgb_model.predict_proba(x_test_scaled)  # Use scaled test data␣
      ↪for prediction

     # Display the predicted probabilities for Logistic Regression and XGBoost
     print("Logistic Regression Predicted Probabilities:\n", y_test_hat_logistic)
     print("\nXGBoost Predicted Probabilities:\n", y_test_hat_xgb)
```

```
Logistic Regression Predicted Probabilities:
 [[3.80171505e-14 1.79677755e-16 6.39379100e-15 … 5.38996491e-12
  3.86042318e-11 1.08306939e-10]
 [3.47565068e-19 1.24709876e-15 1.38143214e-17 … 2.40264684e-11
  7.09480485e-05 1.10783946e-09]
 [4.63781842e-14 1.51592963e-11 2.34262947e-15 … 2.38911411e-14
  4.45010473e-07 2.94420163e-12]
 …
 [1.44640950e-10 2.47678160e-05 1.63766974e-09 … 2.97537303e-10
```

```
  1.62026427e-11 9.27725670e-10]
 [2.40986517e-15 2.18268999e-11 5.48917631e-13 … 6.86061570e-15
  1.29394377e-07 1.21871407e-08]
 [4.03181434e-12 4.30914823e-08 3.20390902e-12 … 3.78280383e-16
  1.88017463e-08 1.25449672e-13]]

XGBoost Predicted Probabilities:
 [[5.1860439e-07 5.7016132e-07 3.9342046e-07 … 8.7231723e-07
  7.8322529e-07 5.8524409e-07]
 [8.1559364e-07 1.7003977e-06 6.5709645e-07 … 1.0578570e-06
  1.6773010e-05 2.0757868e-06]
 [5.4707402e-07 7.7930986e-07 5.7252220e-07 … 1.0011920e-06
  4.1819781e-06 9.0937459e-07]
 …
 [7.2769018e-07 3.8097273e-06 7.3218297e-07 … 6.5145679e-07
  5.3072574e-07 4.8228475e-07]
 [2.6380076e-06 2.8984452e-06 2.4174913e-06 … 4.4621816e-06
  1.0261622e-05 3.1510925e-05]
 [8.4509213e-07 7.6313864e-07 6.7003242e-07 … 7.0236609e-07
  3.7610064e-06 9.6112535e-07]]
```

```
[ ]: from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import log_loss

     def logit(x_train, y_train, x_test):
         """Logistic Regression.

         Args:
             x_train: Training features.
             y_train: Training labels.
             x_test: Testing features.

         Returns:
             y_test_hat: Predicted probabilities for x_test.
         """
         # Convert labels into proper format
         if len(y_train.shape) > 1:
             y_train = donvert_matrix_to_vector(y_train)

         # Define and fit the model on the training dataset
         model = LogisticRegression()
         model.fit(x_train, y_train)

         # Predict probabilities on x_test
         y_test_hat = model.predict_proba(x_test)

         return y_test_hat
```

```
y_test_prob = logit(X_train, y_train, X_test)

# Display the probabilities
print("Predicted probabilities for the test set:")
print(y_test_prob)

# Compute log loss
log_loss_value = log_loss(y_test, y_test_prob)

# Display log loss
print("Log loss for the test set:", log_loss_value)
```

```
Predicted probabilities for the test set:
[[1.27832255e-12 2.06977665e-16 3.99046638e-17 … 7.92486068e-13
  3.66276613e-14 1.83700781e-13]
 [3.53829724e-14 5.62561775e-14 9.40919132e-16 … 1.07032765e-11
  2.42897888e-04 1.52985856e-10]
 [9.66721886e-11 2.33132685e-10 3.73727689e-12 … 1.13644612e-10
  1.28665515e-06 3.89190497e-11]
 …
 [8.97193682e-08 1.11777043e-05 1.13462283e-08 … 2.18504192e-08
  2.32788580e-10 2.44061608e-10]
 [4.43450554e-09 3.80180801e-10 2.59394355e-11 … 1.75572500e-08
  5.96201221e-06 2.02390897e-07]
 [2.07984818e-09 7.62212185e-09 7.21995065e-11 … 3.49653489e-11
  5.64646108e-08 4.47350113e-12]]
Log loss for the test set: 0.033144266653965554
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[ ]:
```
from xgboost import XGBClassifier
from sklearn.metrics import log_loss
import numpy as np

def xgboost_model(x_train, y_train, x_test):
    """XGBoost Classifier.

    Args:
        x_train: Training features.
        y_train: Training labels.
        x_test: Testing features.

    Returns:
        y_test_prob: Predicted probabilities for x_test.
    """
    # Convert labels to proper format and zero-based index if necessary
    if len(y_train.shape) > 1:
        y_train = donvert_matrix_to_vector(y_train)

    # Check if labels need to be shifted to start from 0
    if np.min(y_train) != 0:
        y_train = y_train - np.min(y_train)  # Shift labels to start from 0

    # Define and fit the XGBoost model on the training dataset
    model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
    model.fit(x_train, y_train)

    # Predict probabilities on x_test
    y_test_prob = model.predict_proba(x_test)

    return y_test_prob

# Example usage
# Assuming y_test is the true labels for X_test
# Check if y_test needs to be shifted to start from 0 for log loss calculation
if np.min(y_test) != 0:
    y_test_zero_based = y_test - np.min(y_test)
else:
    y_test_zero_based = y_test

y_test_prob = xgboost_model(X_train, y_train, X_test)

# Display the probabilities
print("Predicted probabilities for the test set:")
print(y_test_prob)

# Compute log loss
log_loss_value = log_loss(y_test_zero_based, y_test_prob)
print("Log loss for the test set:", log_loss_value)
```

```
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning:
[01:48:27] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)

Predicted probabilities for the test set:
```

```
[[5.1860439e-07 5.7017002e-07 3.9342234e-07 … 8.7231638e-07
  7.8322529e-07 5.8524296e-07]
 [8.1559443e-07 1.7004106e-06 6.5709958e-07 … 1.0578590e-06
  1.6773043e-05 2.0757868e-06]
 [5.4707510e-07 7.7939086e-07 5.7252436e-07 … 1.0011939e-06
  4.1819862e-06 9.0937459e-07]
 …
 [7.2769092e-07 3.8095675e-06 7.3217876e-07 … 6.5145679e-07
  5.3072574e-07 4.8228475e-07]
 [2.6380076e-06 2.8987354e-06 2.4175074e-06 … 4.4621897e-06
  1.0261622e-05 3.1510957e-05]
 [8.4509293e-07 7.6321942e-07 6.7003754e-07 … 7.0236740e-07
  3.7610098e-06 9.6112626e-07]]
Log loss for the test set: 0.00400363072165128
```

##XGBoost Log Loss

[ ]:
```
from sklearn.metrics import log_loss


# Calculate log loss for XGBoost
xgb_loss = log_loss(y_test, y_test_hat_xgb)
print("XGBoost Log Loss:", xgb_loss)
```

```
XGBoost Log Loss: 0.004003640001628129
```

##**ENCODER MODEL**

[ ]:
```
from keras.layers import Input, Dense
from keras.models import Model
import numpy as np

def binary_mask(p_m, data):
    """Generates a binary mask with probability p_m."""
    return np.random.binomial(1, 1 - p_m, data.shape)

def corruption(mask, data):
    num_samples, num_features = data.shape
    shuffled_data = np.zeros([num_samples, num_features])

    for feature_idx in range(num_features):
        shuffled_indices = np.random.permutation(num_samples)
        shuffled_data[:, feature_idx] = data[shuffled_indices, feature_idx]

    data_corrupted = data * (1 - mask) + shuffled_data * mask
    mask_new = (data != data_corrupted).astype(int)

    return mask_new, data_corrupted
```

```python
def self_supervised(x_unlabeled, p_m, alpha, parameters):
    epochs = parameters['epochs']
    batch_size = parameters['batch_size']
    _, dimension = x_unlabeled.shape

    # Define model architecture
    input_layer = Input(shape=(dimension,))
    h = Dense(int(dimension), activation='relu')(input_layer)

    output1 = Dense(int(dimension), activation='sigmoid',␣
 ↪name='mask_estimation')(h)
    output2 = Dense(int(dimension), activation='sigmoid',␣
 ↪name='feature_estimation')(h)

    model = Model(inputs=input_layer, outputs=[output1, output2])

    # Compile model with appropriate loss functions and weights
    model.compile(
        optimizer="rmsprop",
        loss={'mask_estimation': 'binary_crossentropy', 'feature_estimation':␣
 ↪'mean_squared_error'},
        loss_weights={'mask_estimation': 1.0, 'feature_estimation':␣
 ↪float(alpha)}  # Corrected to use float
    )

    # Generate corrupted input and mask labels
    corruption_binary_mask = binary_mask(p_m, x_unlabeled)
    x_unlabeled_corrupted,mask_label = corruption(corruption_binary_mask,␣
 ↪x_unlabeled)

    assert x_unlabeled_corrupted.shape == mask_label.shape


    # Train model
    model.fit(x_unlabeled_corrupted, {'mask_estimation': mask_label,␣
 ↪'feature_estimation': x_unlabeled},
              epochs=epochs, batch_size=batch_size)

    # Display model summary (this will print the model's parameters)
    model.summary()

    # Define encoder
    name_of_layer = model.layers[1].name
    layer_output = model.get_layer(name_of_layer).output
    encoder = Model(inputs=model.input, outputs=layer_output)
    return encoder
```

[ ]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler

# Load the dataset
# data = pd.read_csv('/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv')

# Exclude specified columns
exclude_columns = ['Event', 'Time', 'Cell_length', 'file_number',␣
 ↪'event_number', 'label', 'individual']
data_filtered = data.drop(columns=exclude_columns)

# Standardize the data
scaler = StandardScaler()
x_unlabeled_scaled = scaler.fit_transform(data_filtered)  # Now␣
 ↪x_unlabeled_scaled is defined

# Define other parameters
p_m = 0.3
alpha = 2.0
parameters = {
    'batch_size': 128,
    'epochs': 50,
}

# Run the self_supervised function with the scaled data
encoder_model = self_supervised(x_unlabeled_scaled, p_m, alpha, parameters)
```

```
Epoch 1/50
2076/2076             7s 3ms/step -
feature_estimation_loss: 0.0766 - loss: 2.1129 - mask_estimation_loss: 2.0363
Epoch 2/50
2076/2076             4s 2ms/step -
feature_estimation_loss: -0.0184 - loss: 1.9830 - mask_estimation_loss: 2.0013
Epoch 3/50
2076/2076             5s 2ms/step -
feature_estimation_loss: -0.0019 - loss: 1.9964 - mask_estimation_loss: 1.9984
Epoch 4/50
2076/2076             7s 3ms/step -
feature_estimation_loss: -0.0230 - loss: 1.9786 - mask_estimation_loss: 2.0017
Epoch 5/50
2076/2076             4s 2ms/step -
feature_estimation_loss: -0.0515 - loss: 1.9472 - mask_estimation_loss: 1.9987
Epoch 6/50
2076/2076             4s 2ms/step -
```

```
feature_estimation_loss: -0.0942 - loss: 1.9040 - mask_estimation_loss: 1.9982
Epoch 7/50
2076/2076              7s 3ms/step -
feature_estimation_loss: -0.1120 - loss: 1.8836 - mask_estimation_loss: 1.9956
Epoch 8/50
2076/2076              5s 2ms/step -
feature_estimation_loss: -0.0802 - loss: 1.9176 - mask_estimation_loss: 1.9979
Epoch 9/50
2076/2076              6s 3ms/step -
feature_estimation_loss: -0.2092 - loss: 1.7892 - mask_estimation_loss: 1.9984
Epoch 10/50
2076/2076              9s 2ms/step -
feature_estimation_loss: -0.1551 - loss: 1.8508 - mask_estimation_loss: 2.0059
Epoch 11/50
2076/2076              5s 2ms/step -
feature_estimation_loss: -0.3687 - loss: 1.6287 - mask_estimation_loss: 1.9974
Epoch 12/50
2076/2076              9s 2ms/step -
feature_estimation_loss: -0.2640 - loss: 1.7338 - mask_estimation_loss: 1.9979
Epoch 13/50
2076/2076              6s 3ms/step -
feature_estimation_loss: -0.4363 - loss: 1.5628 - mask_estimation_loss: 1.9991
Epoch 14/50
2076/2076              9s 2ms/step -
feature_estimation_loss: -0.2996 - loss: 1.6972 - mask_estimation_loss: 1.9968
Epoch 15/50
2076/2076              6s 3ms/step -
feature_estimation_loss: -0.4141 - loss: 1.5852 - mask_estimation_loss: 1.9993
Epoch 16/50
2076/2076              9s 2ms/step -
feature_estimation_loss: -0.4214 - loss: 1.5785 - mask_estimation_loss: 1.9999
Epoch 17/50
2076/2076              6s 3ms/step -
feature_estimation_loss: -0.5260 - loss: 1.4742 - mask_estimation_loss: 2.0002
Epoch 18/50
2076/2076              9s 2ms/step -
feature_estimation_loss: -0.5785 - loss: 1.4261 - mask_estimation_loss: 2.0045
Epoch 19/50
2076/2076              7s 3ms/step -
feature_estimation_loss: -0.7073 - loss: 1.2930 - mask_estimation_loss: 2.0004
Epoch 20/50
2076/2076              4s 2ms/step -
feature_estimation_loss: -0.7332 - loss: 1.2684 - mask_estimation_loss: 2.0016
Epoch 21/50
2076/2076              4s 2ms/step -
feature_estimation_loss: -1.1876 - loss: 0.8136 - mask_estimation_loss: 2.0013
Epoch 22/50
2076/2076              6s 3ms/step -
```

```
feature_estimation_loss: -1.5166 - loss: 0.4800 - mask_estimation_loss: 1.9967
Epoch 23/50
2076/2076           9s 2ms/step -
feature_estimation_loss: -1.7986 - loss: 0.1957 - mask_estimation_loss: 1.9943
Epoch 24/50
2076/2076           6s 3ms/step -
feature_estimation_loss: -1.3329 - loss: 0.6703 - mask_estimation_loss: 2.0033
Epoch 25/50
2076/2076           8s 2ms/step -
feature_estimation_loss: -1.3981 - loss: 0.6025 - mask_estimation_loss: 2.0006
Epoch 26/50
2076/2076           7s 3ms/step -
feature_estimation_loss: -2.1693 - loss: -0.1622 - mask_estimation_loss: 2.0071
Epoch 27/50
2076/2076           4s 2ms/step -
feature_estimation_loss: -0.6803 - loss: 1.3197 - mask_estimation_loss: 2.0000
Epoch 28/50
2076/2076           4s 2ms/step -
feature_estimation_loss: -1.1905 - loss: 0.8115 - mask_estimation_loss: 2.0019
Epoch 29/50
2076/2076           6s 3ms/step -
feature_estimation_loss: -2.5179 - loss: -0.5181 - mask_estimation_loss: 1.9998
Epoch 30/50
2076/2076           8s 2ms/step -
feature_estimation_loss: -2.8251 - loss: -0.8252 - mask_estimation_loss: 1.9999
Epoch 31/50
2076/2076           6s 3ms/step -
feature_estimation_loss: -2.6824 - loss: -0.6825 - mask_estimation_loss: 1.9999
Epoch 32/50
2076/2076           5s 2ms/step -
feature_estimation_loss: -1.9845 - loss: 0.0137 - mask_estimation_loss: 1.9982
Epoch 33/50
2076/2076           4s 2ms/step -
feature_estimation_loss: -4.0104 - loss: -2.0138 - mask_estimation_loss: 1.9967
Epoch 34/50
2076/2076           6s 3ms/step -
feature_estimation_loss: -1.8072 - loss: 0.1993 - mask_estimation_loss: 2.0065
Epoch 35/50
2076/2076           5s 2ms/step -
feature_estimation_loss: -3.4241 - loss: -1.4235 - mask_estimation_loss: 2.0006
Epoch 36/50
2076/2076           5s 2ms/step -
feature_estimation_loss: -5.5387 - loss: -3.5382 - mask_estimation_loss: 2.0005
Epoch 37/50
2076/2076           5s 3ms/step -
feature_estimation_loss: -5.0089 - loss: -3.0070 - mask_estimation_loss: 2.0018
Epoch 38/50
2076/2076           5s 2ms/step -
```

```
feature_estimation_loss: -2.6510 - loss: -0.6501 - mask_estimation_loss: 2.0009
Epoch 39/50
2076/2076                11s 3ms/step -
feature_estimation_loss: -7.2637 - loss: -5.2667 - mask_estimation_loss: 1.9969
Epoch 40/50
2076/2076                5s 2ms/step -
feature_estimation_loss: -4.1740 - loss: -2.1737 - mask_estimation_loss: 2.0003
Epoch 41/50
2076/2076                4s 2ms/step -
feature_estimation_loss: -7.4531 - loss: -5.4564 - mask_estimation_loss: 1.9967
Epoch 42/50
2076/2076                6s 3ms/step -
feature_estimation_loss: -5.8372 - loss: -3.8401 - mask_estimation_loss: 1.9972
Epoch 43/50
2076/2076                5s 2ms/step -
feature_estimation_loss: -5.5592 - loss: -3.5622 - mask_estimation_loss: 1.9969
Epoch 44/50
2076/2076                4s 2ms/step -
feature_estimation_loss: -4.4282 - loss: -2.4249 - mask_estimation_loss: 2.0032
Epoch 45/50
2076/2076                8s 3ms/step -
feature_estimation_loss: -7.0175 - loss: -5.0199 - mask_estimation_loss: 1.9974
Epoch 46/50
2076/2076                5s 2ms/step -
feature_estimation_loss: -7.8107 - loss: -5.8059 - mask_estimation_loss: 2.0048
Epoch 47/50
2076/2076                5s 2ms/step -
feature_estimation_loss: -5.6008 - loss: -3.6003 - mask_estimation_loss: 2.0004
Epoch 48/50
2076/2076                6s 3ms/step -
feature_estimation_loss: -10.0767 - loss: -8.0819 - mask_estimation_loss: 1.9948
Epoch 49/50
2076/2076                9s 2ms/step -
feature_estimation_loss: -9.0966 - loss: -7.0964 - mask_estimation_loss: 2.0002
Epoch 50/50
2076/2076                6s 3ms/step -
feature_estimation_loss: -7.2386 - loss: -5.2414 - mask_estimation_loss: 1.9973

Model: "functional"


 Layer (type)                Output Shape                 Param #   Connected␣
↪to

 input_layer (InputLayer)    (None, 35)                         0   -          ␣
↪
```

```
dense (Dense)                (None, 35)                  1,260  ␣
↪input_layer[0][0]

mask_estimation (Dense)      (None, 35)                  1,260  ␣
↪dense[0][0]

feature_estimation           (None, 35)                  1,260  ␣
↪dense[0][0]
(Dense)                                                                    ␣
↪
```

```
Total params: 7,562 (29.54 KB)

Trainable params: 3,780 (14.77 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 3,782 (14.78 KB)
```

[ ]:
```
from keras.layers import Input,Dense
from keras.models import Model
from keras import models
import numpy as np

def binary_mask(p_m, data):
    """Generates a binary mask with probability p_m for corruption."""
    return pd.DataFrame(np.random.binomial(1, p_m, data.shape), columns=data.
 ↪columns)

def x_corruption(mask, data):
    """Applies corruption to the data using the mask."""
    shuffled = data.apply(lambda col: np.random.permutation(col))
    return data * (1 - mask) + shuffled * mask
def self_supervised(x_unlabeled_scaled,p_m, alpha, parameters):

  # extract the batch_size and epochs
  epochs = parameters['epochs']
  batch_size = parameters['batch_size']
  _,dimension = x_unlabeled_scaled.shape

  # model creation
  # defining an encoder
```

```
    # auto encoder ---> corrupted input ---> encoder ----> latent space --->␣
    ↪decoder
    # working on the encoder part and extracting the latent space
    # creating a fully connecting network with the number of neurons in the forst␣
    ↪layer equal to the number of features present in the dataset
    # input_layer will be of size 37
    input_layer = Input(shape=(dimension,))

    #encoder model
    h = Dense(int(dimension),activation='relu')(input_layer)

    #output1 ---> mask estimation
    output1 = Dense(int(dimension) , activation='sigmoid',␣
    ↪name='mask_estimation')(h)

    #output2 ---> feature estimation
    output2 = Dense(int(dimension) , activation='sigmoid',␣
    ↪name='feature_estimation')(h)

    model = Model(inputs = input_layer, outputs=[output1,output2])
    model.compile(optimizer="rmsprop",loss={'mask_estimation':␣
    ↪'binary_crossentropy', 'feature_estimation':␣
    ↪'mean_squared_error'},loss_weights={'mask_estimation': 1.0,␣
    ↪'feature_estimation': alpha})

    # Generate corrupted data and mask
    corruption_mask = binary_mask(p_m,x_unlabeled_scaled)
    x_unlabeled_corrupted = x_corruption(corruption_mask, x_unlabeled_scaled)
    m_label = (x_unlabeled_scaled != x_unlabeled_corrupted).astype(int) #␣
    ↪Calculate m_label

    # Fit the model
    model.fit(x_unlabeled_corrupted,{'mask_estimation':
    ↪m_label,'feature_estimation':
    ↪x_unlabeled_scaled},epochs=epochs,batch_size=batch_size)

    name_of_layer = model.layers[1].name # Assuming the encoder layer is the␣
    ↪second layer
    layer_output = model.get_layer(name_of_layer).output
    encoder = models.Model(inputs=model.input , outputs=layer_output)
    model.summary()
    return encoder


x_unlab = x_unlabeled_scaled
```

```
p_m=0.3

alpha= 2.0

parameters={'batch_size':128,
            'epochs':50,
            }

encoder_model =self_supervised(x_unlab,p_m, alpha, parameters)
```

```
Epoch 1/50
1262/1262            4s 2ms/step -
feature_estimation_loss: 0.6403 - loss: 2.3843 - mask_estimation_loss: 1.7439
Epoch 2/50
1262/1262            7s 3ms/step -
feature_estimation_loss: 0.6104 - loss: 1.9914 - mask_estimation_loss: 1.3810
Epoch 3/50
1262/1262            3s 2ms/step -
feature_estimation_loss: 0.6084 - loss: 1.9778 - mask_estimation_loss: 1.3694
Epoch 4/50
1262/1262            3s 2ms/step -
feature_estimation_loss: 0.6079 - loss: 1.9711 - mask_estimation_loss: 1.3632
Epoch 5/50
1262/1262            2s 2ms/step -
feature_estimation_loss: 0.6076 - loss: 1.9710 - mask_estimation_loss: 1.3634
Epoch 6/50
1262/1262            3s 2ms/step -
feature_estimation_loss: 0.6071 - loss: 1.9661 - mask_estimation_loss: 1.3590
Epoch 7/50
1262/1262            5s 2ms/step -
feature_estimation_loss: 0.6066 - loss: 1.9597 - mask_estimation_loss: 1.3530
Epoch 8/50
1262/1262            2s 2ms/step -
feature_estimation_loss: 0.6064 - loss: 1.9615 - mask_estimation_loss: 1.3551
Epoch 9/50
1262/1262            3s 2ms/step -
feature_estimation_loss: 0.6059 - loss: 1.9585 - mask_estimation_loss: 1.3527
Epoch 10/50
1262/1262            2s 2ms/step -
feature_estimation_loss: 0.6056 - loss: 1.9610 - mask_estimation_loss: 1.3554
Epoch 11/50
1262/1262            4s 3ms/step -
feature_estimation_loss: 0.6055 - loss: 1.9594 - mask_estimation_loss: 1.3539
Epoch 12/50
1262/1262            3s 3ms/step -
feature_estimation_loss: 0.6051 - loss: 1.9591 - mask_estimation_loss: 1.3541
Epoch 13/50
```

```
1262/1262              4s 2ms/step -
feature_estimation_loss: 0.6049 - loss: 1.9574 - mask_estimation_loss: 1.3525
Epoch 14/50
1262/1262              2s 2ms/step -
feature_estimation_loss: 0.6047 - loss: 1.9571 - mask_estimation_loss: 1.3524
Epoch 15/50
1262/1262              3s 2ms/step -
feature_estimation_loss: 0.6046 - loss: 1.9560 - mask_estimation_loss: 1.3515
Epoch 16/50
1262/1262              4s 3ms/step -
feature_estimation_loss: 0.6046 - loss: 1.9564 - mask_estimation_loss: 1.3518
Epoch 17/50
1262/1262              3s 2ms/step -
feature_estimation_loss: 0.6042 - loss: 1.9545 - mask_estimation_loss: 1.3503
Epoch 18/50
1262/1262              2s 2ms/step -
feature_estimation_loss: 0.6046 - loss: 1.9526 - mask_estimation_loss: 1.3479
Epoch 19/50
1262/1262              2s 2ms/step -
feature_estimation_loss: 0.6043 - loss: 1.9562 - mask_estimation_loss: 1.3519
Epoch 20/50
1262/1262              2s 2ms/step -
feature_estimation_loss: 0.6041 - loss: 1.9556 - mask_estimation_loss: 1.3515
Epoch 21/50
1262/1262              4s 3ms/step -
feature_estimation_loss: 0.6040 - loss: 1.9497 - mask_estimation_loss: 1.3457
Epoch 22/50
1262/1262              4s 3ms/step -
feature_estimation_loss: 0.6044 - loss: 1.9511 - mask_estimation_loss: 1.3467
Epoch 23/50
1262/1262              4s 2ms/step -
feature_estimation_loss: 0.6039 - loss: 1.9552 - mask_estimation_loss: 1.3512
Epoch 24/50
1262/1262              3s 2ms/step -
feature_estimation_loss: 0.6036 - loss: 1.9527 - mask_estimation_loss: 1.3492
Epoch 25/50
1262/1262              2s 2ms/step -
feature_estimation_loss: 0.6038 - loss: 1.9491 - mask_estimation_loss: 1.3453
Epoch 26/50
1262/1262              3s 3ms/step -
feature_estimation_loss: 0.6037 - loss: 1.9543 - mask_estimation_loss: 1.3506
Epoch 27/50
1262/1262              4s 3ms/step -
feature_estimation_loss: 0.6038 - loss: 1.9491 - mask_estimation_loss: 1.3453
Epoch 28/50
1262/1262              4s 2ms/step -
feature_estimation_loss: 0.6036 - loss: 1.9518 - mask_estimation_loss: 1.3483
Epoch 29/50
```

```
1262/1262              3s 2ms/step -
feature_estimation_loss: 0.6036 - loss: 1.9497 - mask_estimation_loss: 1.3461
Epoch 30/50
1262/1262              2s 2ms/step -
feature_estimation_loss: 0.6035 - loss: 1.9501 - mask_estimation_loss: 1.3466
Epoch 31/50
1262/1262              4s 3ms/step -
feature_estimation_loss: 0.6038 - loss: 1.9471 - mask_estimation_loss: 1.3433
Epoch 32/50
1262/1262              4s 3ms/step -
feature_estimation_loss: 0.6034 - loss: 1.9484 - mask_estimation_loss: 1.3450
Epoch 33/50
1262/1262              4s 2ms/step -
feature_estimation_loss: 0.6032 - loss: 1.9497 - mask_estimation_loss: 1.3465
Epoch 34/50
1262/1262              3s 2ms/step -
feature_estimation_loss: 0.6035 - loss: 1.9524 - mask_estimation_loss: 1.3489
Epoch 35/50
1262/1262              2s 2ms/step -
feature_estimation_loss: 0.6033 - loss: 1.9514 - mask_estimation_loss: 1.3481
Epoch 36/50
1262/1262              3s 3ms/step -
feature_estimation_loss: 0.6035 - loss: 1.9457 - mask_estimation_loss: 1.3422
Epoch 37/50
1262/1262              4s 2ms/step -
feature_estimation_loss: 0.6036 - loss: 1.9534 - mask_estimation_loss: 1.3498
Epoch 38/50
1262/1262              3s 2ms/step -
feature_estimation_loss: 0.6035 - loss: 1.9478 - mask_estimation_loss: 1.3444
Epoch 39/50
1262/1262              2s 2ms/step -
feature_estimation_loss: 0.6033 - loss: 1.9538 - mask_estimation_loss: 1.3505
Epoch 40/50
1262/1262              2s 2ms/step -
feature_estimation_loss: 0.6032 - loss: 1.9510 - mask_estimation_loss: 1.3478
Epoch 41/50
1262/1262              4s 3ms/step -
feature_estimation_loss: 0.6032 - loss: 1.9499 - mask_estimation_loss: 1.3467
Epoch 42/50
1262/1262              3s 3ms/step -
feature_estimation_loss: 0.6033 - loss: 1.9535 - mask_estimation_loss: 1.3502
Epoch 43/50
1262/1262              4s 2ms/step -
feature_estimation_loss: 0.6031 - loss: 1.9506 - mask_estimation_loss: 1.3475
Epoch 44/50
1262/1262              3s 2ms/step -
feature_estimation_loss: 0.6032 - loss: 1.9515 - mask_estimation_loss: 1.3483
Epoch 45/50
```

```
1262/1262           3s 2ms/step -
feature_estimation_loss: 0.6030 - loss: 1.9482 - mask_estimation_loss: 1.3452
Epoch 46/50
1262/1262           4s 3ms/step -
feature_estimation_loss: 0.6032 - loss: 1.9515 - mask_estimation_loss: 1.3482
Epoch 47/50
1262/1262           4s 2ms/step -
feature_estimation_loss: 0.6033 - loss: 1.9509 - mask_estimation_loss: 1.3476
Epoch 48/50
1262/1262           3s 2ms/step -
feature_estimation_loss: 0.6032 - loss: 1.9455 - mask_estimation_loss: 1.3423
Epoch 49/50
1262/1262           2s 2ms/step -
feature_estimation_loss: 0.6030 - loss: 1.9492 - mask_estimation_loss: 1.3462
Epoch 50/50
1262/1262           3s 2ms/step -
feature_estimation_loss: 0.6030 - loss: 1.9491 - mask_estimation_loss: 1.3461
Model: "functional"


 Layer (type)                Output Shape                 Param #   Connected␣
↪to

 input_layer (InputLayer)    (None, 36)                         0   -          ␣
↪

 dense (Dense)               (None, 36)                     1,332  ␣
↪input_layer[0][0]

 mask_estimation (Dense)     (None, 36)                     1,332  ␣
↪dense[0][0]

 feature_estimation          (None, 36)                     1,332  ␣
↪dense[0][0]
 (Dense)                                                                       ␣
↪


 Total params: 7,994 (31.23 KB)

 Trainable params: 3,996 (15.61 KB)

 Non-trainable params: 0 (0.00 B)
 Optimizer params: 3,998 (15.62 KB)
```

```
[ ]: import os

     # Define the path where you want to save the model
     encoder_path = "content/encoder_model.keras"

     # Create the directory if it doesn't exist
     os.makedirs(os.path.dirname(encoder_path), exist_ok=True)

     # Save the model
     encoder_model.save(encoder_path)

     print(f"Model saved to {encoder_path}")
```

```
Model saved to content/encoder_model.keras
```

```
[ ]: from keras.models import load_model
     encoder=load_model(encoder_path)
```

```
[ ]: # import pandas as pd
     # import numpy as np
     # from sklearn.preprocessing import StandardScaler
     # from keras.models import load_model

     # # ... (Load your data and define exclude_columns as before) ...
     # # Exclude specified columns
     # exclude_columns = ['Event', 'Time', 'Cell_length', 'file_number',␣
      ↪'event_number', 'label', 'individual']
     # data_filtered = data.drop(columns=exclude_columns)

     # # Get the column names used during training
     # training_columns = data_filtered.columns  # Assuming data_filtered was used␣
      ↪for training

     # # Select the same columns from x_train and x_test
     # x_train_filtered = x_train[training_columns]
     # x_test_filtered = x_test[training_columns]

     # # Standardize using the same scaler used during training
     # # Assuming you saved the scaler, otherwise recreate it with the same␣
      ↪parameters
     # # scaler = load_scaler("path/to/scaler.pkl")  # If saved
     # scaler = StandardScaler()  # If not saved, recreate it
     # x_train_scaled = scaler.fit_transform(x_train_filtered)
     # x_test_scaled = scaler.transform(x_test_filtered)
     # # Load the encoder model
     # encoder_model = load_model(encoder_path)

     # # Now predict using the correctly preprocessed data
     # X_train_scaled_encoded = encoder_model.predict(x_train_scaled)
     # X_test_scaled_encoded = encoder_model.predict(x_test_scaled)

     # # ... (Rest of your code) ...

     # logistic_model = LogisticRegression(max_iter=5000)
     # logistic_model.fit(X_train_scaled_encoded, y_train)
     # y_encoded = logistic_model.predict_proba(X_test_scaled_encoded)

     # from sklearn.metrics import log_loss
     # print("Logistic Regression Log Loss:", log_loss(y_test, y_encoded))

     # xgb_model = XGBClassifier(eval_metric='mlogloss')
     # xgb_model.fit(X_train_scaled_encoded, y_train)
     # y_encoded_xgb = xgb_model.predict_proba(X_test_scaled_encoded)

     # print("XGBoost Log Loss:", log_loss(y_test, y_encoded_xgb))

     import numpy as np
     from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import log_loss
     import xgboost as xgb

     # Adjust y_train and y_test labels to start from 0 by subtracting the minimum␣
      ↪label value
     y_train -= y_train.min()
     y_test -= y_test.min()

     scaler = StandardScaler()
     x_train_scaled = scaler.fit_transform(x_train)  # Scale training data
     x_test_scaled = scaler.transform(x_test)

     # Step 1: Define the encoder model and train it on x_unlab (assumed to be done␣
      ↪beforehand)
     # For demonstration, use the encoder to transform train and test data

     # Use the encoder to get the encoded data for training and testing
     x_train_scaled_encoded = encoder.predict(x_train_scaled)
     x_test_scaled_encoded = encoder.predict(x_test_scaled)

     # Check shapes
     print("Encoded x_train shape:", x_train_scaled_encoded.shape)
     print("Encoded x_test shape:", x_test_scaled_encoded.shape)

     # Step 2: Logistic Regression
     log_reg = LogisticRegression(max_iter=1000)  # Set max_iter to a higher value␣
      ↪for convergence
     log_reg.fit(x_train_scaled_encoded, y_train)

     # Predict on the test set using Logistic Regression
     y_encoded_log_reg = log_reg.predict_proba(x_test_scaled_encoded)

     # Compute log loss for logistic regression predictions
     log_reg_loss = log_loss(y_test, y_encoded_log_reg)
     print("Log Loss for Logistic Regression:", log_reg_loss)

     # Step 3: XGBoost Model
     xgb_model = xgb.XGBClassifier(eval_metric='logloss', random_state=42)
     xgb_model.fit(x_train_scaled_encoded, y_train)

     # Predict on the test set using XGBoost
     y_encoded_xgb = xgb_model.predict_proba(x_test_scaled_encoded)

     # Compute log loss for XGBoost predictions
     xgb_loss = log_loss(y_test, y_encoded_xgb)
     print("Log Loss for XGBoost:", xgb_loss)
```

```
2279/2279            3s 1ms/step
977/977            2s 2ms/step
Encoded x_train shape: (72928, 36)
Encoded x_test shape: (31256, 36)
Log Loss for Logistic Regression: 0.03542085939608436
Log Loss for XGBoost: 0.057810137316703855
```

#Overview of Function

Function for the model, train, semi_supervised

```
[ ]: import tensorflow as tf
     from tensorflow.keras import layers, models, optimizers
     import numpy as np  # Ensure numpy is imported

     import tensorflow as tf
     from tensorflow.keras import layers, models, optimizers

     # Define the model function
     def build_model(input_dimension, hidden_dimension, label_dimension,␣
      ↪activation=tf.nn.relu):
         inputs = tf.keras.Input(shape=(input_dimension,), name='model_input')
```

```
    x = layers.Dense(hidden_dimension, activation=activation,␣
↪name='model_dense_layer_1')(inputs)
    x = layers.Dense(hidden_dimension, activation=activation,␣
↪name='model_dense_layer_2')(x)
    y_logit = layers.Dense(label_dimension, activation=None,␣
↪name='model_logit_output')(x)
    y = layers.Activation('softmax', name='model_output')(y_logit)
    model = models.Model(inputs=inputs, outputs=[y_logit, y], name="model")
    return model

# Define the training function
def train(feature_batch, label_batch, unlabeled_feature_batch, model, beta,␣
↪supv_loss_fn, optimizer):
    with tf.GradientTape() as tape:
        y_logit, y = model(feature_batch, training=True)
        y_loss = supv_loss_fn(label_batch, y)

        unlabeled_y_logit, unlabeled_y = model(unlabeled_feature_batch,␣
↪training=True)
        unlabeled_y_loss = tf.reduce_mean(tf.nn.moments(unlabeled_y_logit,␣
↪axes=0)[1])

        total_loss = y_loss + beta * unlabeled_y_loss
        grads = tape.gradient(total_loss, model.trainable_weights)
    optimizer.apply_gradients(zip(grads, model.trainable_weights))
    return total_loss

# Define the semi-supervised function
def semi_supervised(x_train, y_train, x_unlabeled, x_test, parameters,␣
↪mask_probability, K, beta):
    hidden_dimension = parameters['hidden_dimension']
    batch_size = parameters['batch_size']
    epochs = parameters['epochs']
    input_dimension = x_train.shape[1]
    label_dimension = len(np.unique(y_train)) if y_train.ndim == 1 else y_train.
↪shape[1]

    # Map class labels if y_train is categorical
    if y_train.ndim == 1 or y_train.shape[1] == 1:
        class_mapping = {label: idx for idx, label in enumerate(np.
↪unique(y_train))}
        y_train = np.vectorize(class_mapping.get)(y_train)

    # Split training data into training and validation sets
    index = np.random.permutation(x_train.shape[0])
    train_index = index[:int(len(index) * 0.9)]
    valid_index = index[int(len(index) * 0.9):]

    splitted_train_x = x_train[train_index, :]
    splitted_train_y = y_train[train_index]
    splitted_valid_x = x_train[valid_index, :]
    splitted_valid_y = y_train[valid_index]

    # Data encoding
    encoder_model_path = "/content/encoder_model.keras"
    encoder = tf.keras.models.load_model(encoder_model_path)

    x_valid_encoded = encoder.predict(splitted_valid_x)
    x_test_encoded = encoder.predict(x_test)

    # Initialize the supervised learning model
    supervised_model = build_model(
        input_dimension=encoder.output_shape[1],
        hidden_dimension=hidden_dimension,
        label_dimension=label_dimension
    )
    optimizer = optimizers.Adam()
    supv_loss_fn = tf.keras.losses.CategoricalCrossentropy(from_logits=True)

    for epoch in range(epochs):
        batch_index = np.random.choice(len(splitted_train_x), batch_size,␣
↪replace=False)
        batch_x = splitted_train_x[batch_index]
        batch_y = splitted_train_y[batch_index]
        batch_x_encoded = encoder.predict(batch_x)

        batch_unlabeled_index = np.random.choice(len(x_unlabeled), batch_size,␣
↪replace=False)
        batch_unlabeled_x = x_unlabeled[batch_unlabeled_index]

        batch_unlabeled_x_shuffled = []
        for _ in range(K):
            mask_batch_unlabeled = binary_mask(mask_probability,␣
↪batch_unlabeled_x)
            _, unlabeled_shuffled_temp = corruption(mask_batch_unlabeled,␣
↪batch_unlabeled_x)
            unlabeled_shuffled_temp_encoded = encoder.
↪predict(unlabeled_shuffled_temp)
            batch_unlabeled_x_shuffled.append(unlabeled_shuffled_temp_encoded)

        batch_unlabeled_x_shuffled = np.concatenate(batch_unlabeled_x_shuffled,␣
↪axis=0)
```

```
        total_loss = train(batch_x_encoded, batch_y,␣
↪batch_unlabeled_x_shuffled, supervised_model, beta, supv_loss_fn, optimizer)

        y_valid_logit, y_valid = supervised_model(x_valid_encoded,␣
↪training=False)
        y_valid_loss = supv_loss_fn(splitted_valid_y, y_valid_logit)

        if epoch % 100 == 0:
            print(f"Epoch: {epoch}/{epochs}, Validation Loss: {y_valid_loss:.
↪4f}")

    y_test_logit, y_test = supervised_model(x_test_encoded, training=False)
    return y_test_logit, supervised_model
```

[ ]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, optimizers, losses
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import load_model
import pandas as pd

# Define the model
def model(input_dimension, hidden_dimension, label_dimension, activation=tf.nn.
↪relu):
    inputs = tf.keras.Input(shape=input_dimension, name='model_input')
    x = layers.Dense(hidden_dimension, activation=activation,␣
↪name='model_dense_layer_1')(inputs)
    x = layers.Dense(hidden_dimension, activation=activation,␣
↪name='model_dense_layer_2')(x)
    y_logit = layers.Dense(label_dimension, activation=None,␣
↪name='model_logit_output')(x)
    y = layers.Activation('softmax', name='model_output')(y_logit)
    return tf.keras.Model(inputs=inputs, outputs=[y_logit, y], name="model")

# Training function
def train(feature_batch, label_batch, unlabeled_feature_batch, model, beta,␣
↪supv_loss_fn, optimizer):
    with tf.GradientTape() as tape:
        # Labeled data loss
        y_logit, _ = model(feature_batch, training=True)
        y_loss = supv_loss_fn(label_batch, y_logit)

        # Unlabeled data loss
        unlabeled_y_logit, _ = model(unlabeled_feature_batch, training=True)
        _, variance = tf.nn.moments(unlabeled_y_logit, axes=0)
        unlabeled_y_loss = tf.reduce_mean(variance)

        # Total loss
        total_loss = y_loss + beta * unlabeled_y_loss

    # Gradient computation and update
    grads = tape.gradient(total_loss, model.trainable_weights)
    optimizer.apply_gradients(zip(grads, model.trainable_weights))
    return total_loss

# Semi-supervised function
def semi_supervised(x_train, y_train, x_unlabeled, x_test, parameters,␣
 ↪mask_probability, K, beta, encoder_path):
    # Ensure NumPy arrays
    if isinstance(x_train, pd.DataFrame):
        x_train = x_train.values
    if isinstance(y_train, pd.Series):
        y_train = y_train.values
    if isinstance(x_unlabeled, pd.DataFrame):
        x_unlabeled = x_unlabeled.values
    if isinstance(x_test, pd.DataFrame):
        x_test = x_test.values

    # Hyperparameters
    hidden_dimension = parameters['hidden_dim']
    batch_size = parameters['batch_size']
    epochs = parameters['iterations']
    input_dimension = x_train.shape[1]

    # Label preprocessing: One-hot encoding for CategoricalCrossentropy
    unique_classes = np.unique(y_train)
    label_dimension = len(unique_classes)
    class_mapping = {label: idx for idx, label in enumerate(unique_classes)}
    y_train_mapped = np.vectorize(class_mapping.get)(y_train)
    y_train_one_hot = to_categorical(y_train_mapped,␣
 ↪num_classes=label_dimension)

    # Data splitting
    index = np.random.permutation(x_train.shape[0])
    train_index = index[:int(len(index) * 0.9)]
    valid_index = index[int(len(index) * 0.9):]
    splitted_train_x, splitted_train_y = x_train[train_index],␣
 ↪y_train_one_hot[train_index]
    splitted_valid_x, splitted_valid_y = x_train[valid_index],␣
 ↪y_train_one_hot[valid_index]

    # Load pre-trained encoder
    encoder = load_model(encoder_path)
    x_valid_encoded = encoder.predict(splitted_valid_x)
    x_test_encoded = encoder.predict(x_test)

    # Initialize the supervised model
    supervised_model = model(input_dimension=(encoder.output_shape[1],),
                             hidden_dimension=hidden_dimension,
                             label_dimension=label_dimension)

    optimizer = optimizers.Adam()
    supv_loss_fn = losses.CategoricalCrossentropy(from_logits=True)

    # Training loop
    for epoch in range(epochs):
        batch_index = np.random.choice(splitted_train_x.shape[0], batch_size,␣
↪replace=False)
        batch_x, batch_y = splitted_train_x[batch_index],␣
↪splitted_train_y[batch_index]
        batch_x_encoded = encoder.predict(batch_x)

        batch_unlabeled_index = np.random.choice(x_unlabeled.shape[0],␣
↪batch_size, replace=False)
        batch_unlabeled_x = x_unlabeled[batch_unlabeled_index]

        batch_unlabeled_x_shuffled = []
        for _ in range(K):
            mask = np.random.binomial(1, mask_probability, batch_unlabeled_x.
↪shape)
            corrupted_data = batch_unlabeled_x * (1 - mask) + np.random.
↪permutation(batch_unlabeled_x) * mask
            corrupted_data_encoded = encoder.predict(corrupted_data)
            batch_unlabeled_x_shuffled.append(corrupted_data_encoded)
        batch_unlabeled_x_shuffled = np.concatenate(batch_unlabeled_x_shuffled,␣
↪axis=0)

        total_loss = train(batch_x_encoded, batch_y,␣
↪batch_unlabeled_x_shuffled, supervised_model, beta, supv_loss_fn, optimizer)

        y_valid_logit, _ = supervised_model(x_valid_encoded, training=False)
        y_valid_loss = supv_loss_fn(splitted_valid_y, y_valid_logit)

        if epoch % 100 == 0:
            print(f'Epoch: {epoch}/{epochs}, Validation Loss: {y_valid_loss:.
↪4f}')

    y_test_logit, _ = supervised_model(x_test_encoded, training=False)
    return y_test_logit, supervised_model

# Hyperparameters
mask_probability = 0.3
K = 3
beta = 1.0
parameters = {
    'hidden_dim': 100,
    'batch_size': 128,
    'iterations': 800
}

# Assuming x_train, y_train, x_unlabeled_scaled, x_test are defined
encoder_path = "content/encoder_model.keras"  # Replace with your encoder path
y_test_model, model_instance = semi_supervised(x_train, y_train,␣
 ↪x_unlabeled_scaled, x_test,
                                         parameters, mask_probability, K, beta,␣
 ↪encoder_path)
```

```
228/228              0s 1ms/step
977/977              1s 1ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
Epoch: 0/800, Validation Loss: 4.3181
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
```

```
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 2ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 5ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 4ms/step
4/4               0s 4ms/step
4/4               0s 4ms/step
4/4               0s 4ms/step
4/4               0s 4ms/step
4/4               0s 4ms/step
4/4               0s 4ms/step
4/4               0s 2ms/step
4/4               0s 2ms/step
4/4               0s 5ms/step
4/4               0s 4ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
```

```
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 5ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
```

```
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
```

```
4/4                0s 4ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 5ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 5ms/step
4/4                0s 4ms/step
4/4                0s 5ms/step
4/4                0s 5ms/step
4/4                0s 5ms/step
4/4                0s 4ms/step
4/4                0s 5ms/step
4/4                0s 5ms/step
4/4                0s 4ms/step
4/4                0s 5ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 5ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
```

```
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 5ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 6ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 5ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
```

```
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 6ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 2ms/step
4/4               0s 5ms/step
4/4               0s 5ms/step
4/4               0s 5ms/step
4/4               0s 5ms/step
4/4               0s 5ms/step
4/4               0s 5ms/step
4/4               0s 5ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
```

```
4/4               0s 5ms/step
4/4               0s 5ms/step
4/4               0s 6ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 5ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 5ms/step
4/4               0s 2ms/step
4/4               0s 2ms/step
4/4               0s 2ms/step
4/4               0s 5ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 5ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
```

```
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 7ms/step
4/4             0s 6ms/step
4/4             0s 6ms/step
4/4             0s 3ms/step
Epoch: 100/800, Validation Loss: 0.3418
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
```

```
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 6ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
```

```
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
```

```
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 6ms/step
4/4             0s 6ms/step
4/4             0s 8ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 16ms/step
4/4             0s 9ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 6ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
```

```
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
```

```
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 2ms/step
4/4                 0s 3ms/step
4/4                 0s 6ms/step
4/4                 0s 3ms/step
4/4                 0s 2ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 4ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 2ms/step
4/4                 0s 5ms/step
4/4                 0s 2ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 4ms/step
4/4                 0s 4ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 5ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 5ms/step
4/4                 0s 4ms/step
4/4                 0s 5ms/step
4/4                 0s 3ms/step
4/4                 0s 3ms/step
4/4                 0s 5ms/step
4/4                 0s 4ms/step
4/4                 0s 3ms/step
```

```
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 5ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 5ms/step
4/4               0s 4ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 8ms/step
4/4               0s 3ms/step
4/4               0s 5ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
```

```
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
```

```
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 8ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 6ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 6ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
```

```
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
Epoch: 200/800, Validation Loss: 0.1842
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
```

```
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 6ms/step
4/4              0s 4ms/step
4/4              0s 7ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
```

```
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 6ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
```

```
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
```

```
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 6ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
```

```
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 2ms/step
4/4               0s 5ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
```

```
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 5ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 6ms/step
4/4              0s 5ms/step
4/4              0s 4ms/step
```

```
4/4                0s 4ms/step
4/4                0s 4ms/step
4/4                0s 6ms/step
4/4                0s 5ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 6ms/step
4/4                0s 5ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 5ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 7ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 5ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 4ms/step
```

```
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
Epoch: 300/800, Validation Loss: 0.1814
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
```

```
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 6ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 7ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 8ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
```

```
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
```

```
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 6ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 5ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 5ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 6ms/step
4/4                0s 3ms/step
```

```
4/4              0s 5ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 6ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
```

```
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
```

```
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 5ms/step
4/4                0s 6ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 5ms/step
4/4                0s 3ms/step
4/4                0s 7ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 6ms/step
4/4                0s 2ms/step
4/4                0s 4ms/step
4/4                0s 2ms/step
4/4                0s 7ms/step
4/4                0s 5ms/step
4/4                0s 5ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
```

```
4/4            0s 4ms/step
4/4            0s 2ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 2ms/step
4/4            0s 4ms/step
4/4            0s 3ms/step
4/4            0s 2ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 2ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 2ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 2ms/step
4/4            0s 4ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 2ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 2ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 3ms/step
4/4            0s 2ms/step
```

```
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 6ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
Epoch: 400/800, Validation Loss: 0.1252
4/4             0s 2ms/step
```

```
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 6ms/step
4/4              0s 7ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
```

```
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
```

```
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 6ms/step
4/4             0s 3ms/step
4/4             0s 6ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 6ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
```

```
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
```

```
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 7ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 6ms/step
4/4              0s 4ms/step
4/4              0s 5ms/step
4/4              0s 2ms/step
4/4              0s 6ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 5ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
```

```
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 6ms/step
4/4             0s 6ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
```

```
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 5ms/step
4/4                0s 5ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 6ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
```

```
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 6ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 6ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 8ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 5ms/step
4/4              0s 4ms/step
```

```
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
Epoch: 500/800, Validation Loss: 0.0845
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
```

```
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 5ms/step
4/4              0s 5ms/step
4/4              0s 7ms/step
4/4              0s 5ms/step
4/4              0s 5ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
```

```
4/4             0s 6ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 6ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 7ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 6ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 6ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
```

```
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
```

```
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 5ms/step
4/4               0s 4ms/step
4/4               0s 4ms/step
4/4               0s 5ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 7ms/step
4/4               0s 5ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 4ms/step
4/4               0s 4ms/step
4/4               0s 5ms/step
4/4               0s 2ms/step
4/4               0s 2ms/step
4/4               0s 5ms/step
4/4               0s 3ms/step
4/4               0s 5ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 6ms/step
4/4               0s 5ms/step
4/4               0s 6ms/step
4/4               0s 5ms/step
4/4               0s 6ms/step
4/4               0s 4ms/step
4/4               0s 4ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
```

```
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 6ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
```

```
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 5ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
```

```
4/4             0s 3ms/step
4/4             0s 7ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
```

```
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 6ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
Epoch: 600/800, Validation Loss: 0.0867
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
```

```
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 6ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 5ms/step
4/4             0s 6ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 2ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
```

```
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 5ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 6ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
```

```
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 5ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 5ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
```

```
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
```

```
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 8ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
```

```
4/4                0s 6ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 5ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 6ms/step
4/4                0s 4ms/step
4/4                0s 4ms/step
4/4                0s 5ms/step
4/4                0s 4ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 4ms/step
4/4                0s 5ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 4ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 3ms/step
4/4                0s 2ms/step
4/4                0s 3ms/step
```

```
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
```

```
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 6ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
```

```
4/4             0s 3ms/step
Epoch: 700/800, Validation Loss: 0.0749
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
```

```
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 6ms/step
4/4              0s 5ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 5ms/step
4/4              0s 6ms/step
4/4              0s 2ms/step
```

```
4/4               0s 2ms/step
4/4               0s 6ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 5ms/step
4/4               0s 4ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 5ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 4ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 5ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 2ms/step
4/4               0s 4ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
4/4               0s 3ms/step
```

```
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
```

```
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 4ms/step
4/4             0s 6ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 5ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 5ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 7ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 6ms/step
4/4             0s 4ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
4/4             0s 2ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 4ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 3ms/step
4/4             0s 2ms/step
```

```
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
```

```
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 6ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
```

```
4/4              0s 4ms/step
4/4              0s 4ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 6ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 5ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
```

```
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 4ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 3ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 2ms/step
4/4              0s 3ms/step
```

```
[ ]: from sklearn.metrics import accuracy_score, roc_auc_score
from sklearn.preprocessing import label_binarize
import numpy as np

def perf_metric(metric, y_test, y_test_hat):
    """
    Evaluate the performance of a classification model using accuracy or AUROC.

    Parameters:
    - metric (str): 'acc' for accuracy or 'auc' for AUROC.
    - y_test (np.array): Ground truth labels, integer encoded, shape:␣
↪(n_samples,).
    - y_test_hat (np.array): Predicted probabilities, shape: (n_samples,␣
↪n_classes).

    Returns:
    - float: Calculated performance metric.
    """
    # Validate input
    if metric not in ['acc', 'auc']:
        raise ValueError("Unsupported metric. Use 'acc' for accuracy or 'auc'␣
↪for AUROC.")

    # Accuracy metric
    if metric == 'acc':
        # Convert predicted probabilities to class labels
        y_pred = np.argmax(y_test_hat, axis=1)
        return accuracy_score(y_test, y_pred)

    # AUROC metric
    elif metric == 'auc':
        n_classes = y_test_hat.shape[1]
        if n_classes == 2:  # Binary classification
            # Use probabilities of the positive class
            y_pred_prob = y_test_hat[:, 1]
            return roc_auc_score(y_test, y_pred_prob)
        elif n_classes > 2:  # Multiclass classification
            # Use one-vs-rest approach
            y_test_bin = label_binarize(y_test, classes=np.unique(y_test))
            return roc_auc_score(y_test_bin, y_test_hat, average='macro',␣
↪multi_class='ovr')
        else:
            raise ValueError("AUROC is not defined for single-class tasks.")
```

[ ]:
```
## Perf Metric
```

[ ]:
```
# Evaluate Accuracy
accuracy = perf_metric('acc', y_test, y_test_model)
print(f"Accuracy: {accuracy:.4f}")

# Evaluate AUROC
auroc = perf_metric('auc', y_test, y_test_model)
print(f"AUROC: {auroc:.4f}")
```

```
Accuracy: 0.9738
AUROC: 0.9934
```

[ ]:
```
def generate_unlabeled_predictions(x_unlab, encoder, predictor):
    """
    Generate predictions for unlabeled data using an encoder and predictor.

    Parameters:
    - x_unlab: Unlabeled feature data.
    - encoder: Pretrained encoder model to encode features.
    - predictor: Trained classification model.

    Returns:
    - y_unlab_pred: Predicted labels for unlabeled data.
    """
    # Encode unlabeled data
    x_unlab_encoded = encoder.predict(x_unlab)

    # Predict with the classifier
    _, y_unlab_hat = predictor(x_unlab_encoded, training=False)

    # Convert probabilities to predicted class labels
    y_unlab_pred = np.argmax(y_unlab_hat, axis=1)
    return y_unlab_pred
```

```
# Generate predictions for the unlabeled data
y_unlab_pred = generate_unlabeled_predictions(x_unlabeled_scaled, encoder,␣
↪model_instance)
print(f"Predicted Labels for Unlabeled Data:\n{y_unlab_pred}")
```

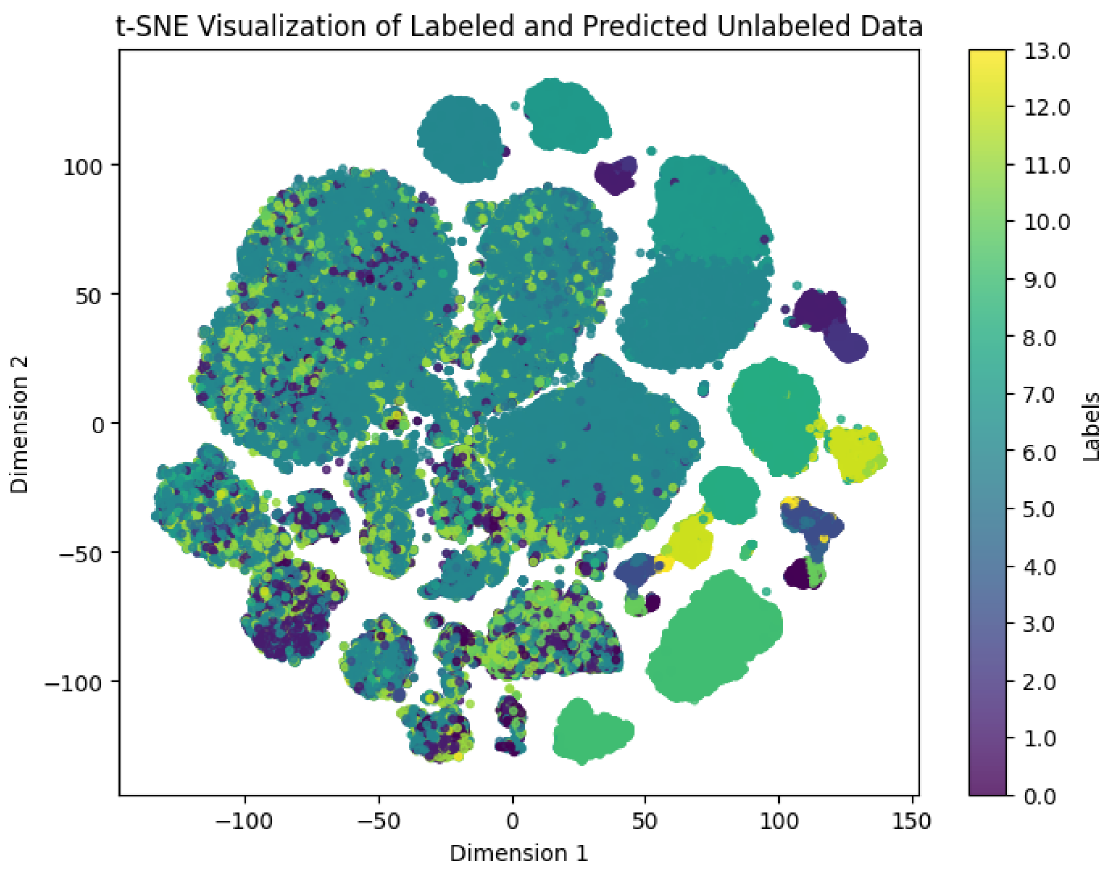
```
5046/5046            15s 3ms/step
Predicted Labels for Unlabeled Data:
[ 6  6  6 …  6  6 11]
```

[ ]:
```
pip install openTSNE
```

```
Collecting openTSNE
  Downloading openTSNE-1.0.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86
_64.whl.metadata (7.8 kB)
Requirement already satisfied: numpy>=1.16.6 in /usr/local/lib/python3.10/dist-
packages (from openTSNE) (1.26.4)
Requirement already satisfied: scikit-learn>=0.20 in
/usr/local/lib/python3.10/dist-packages (from openTSNE) (1.5.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages
(from openTSNE) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn>=0.20->openTSNE) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20->openTSNE)
(3.5.0)
Downloading
openTSNE-1.0.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.0
MB)
                         3.0/3.0 MB
8.8 MB/s eta 0:00:00
Installing collected packages: openTSNE
Successfully installed openTSNE-1.0.2
```

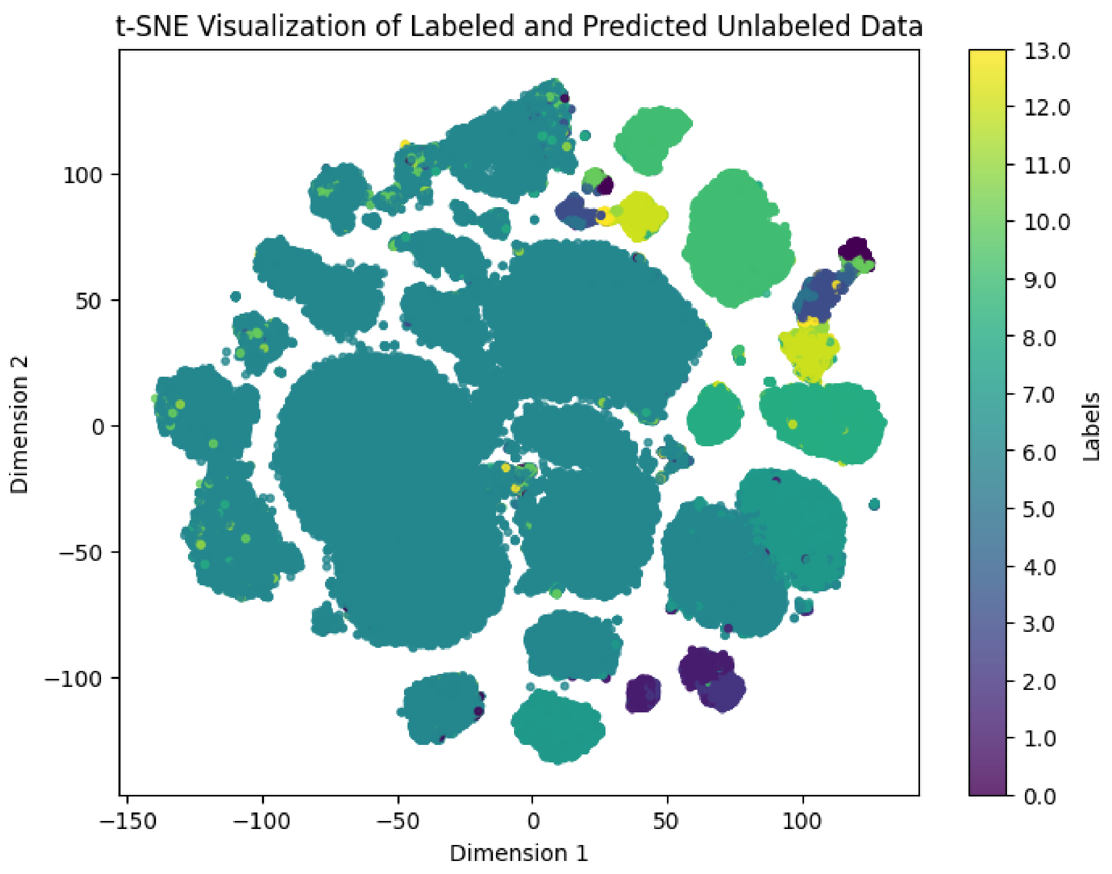
[ ]:
```
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt
from openTSNE import TSNE
import numpy as np
from matplotlib import cm

def plot_tsne_custom_color(features, labels, title="t-SNE Visualization"):
    """
    Generate t-SNE visualization using OpenTSNE with a custom color scheme.

    Parameters:
    - features: The feature matrix (e.g., encoded or raw features).
    - labels: Labels corresponding to the features (should be integers or␣
↪categories).
```

```
    - title: Title of the plot.
    """
    # Perform t-SNE with OpenTSNE
    tsne = TSNE(n_components=2, perplexity=30, n_iter=1000, random_state=42)
    tsne_result = tsne.fit(features)

    # Use a continuous colormap such as 'viridis' to match the color scheme in␣
 ↪the image
    cmap = plt.cm.viridis

    # Plot the results
    plt.figure(figsize=(8, 6))
    scatter = plt.scatter(tsne_result[:, 0], tsne_result[:, 1], c=labels,␣
 ↪cmap=cmap, s=10, alpha=0.8)
    plt.title(title)
    plt.xlabel('Dimension 1')
    plt.ylabel('Dimension 2')
    cbar = plt.colorbar(scatter, ticks=np.unique(labels))
    cbar.set_label('Labels')
    cbar.ax.set_yticklabels([str(label) for label in np.unique(labels)])
    plt.show()

# Features (scaled unlabeled data) and predicted labels
scaled_features = np.vstack([x_train_scaled_encoded, x_unlabeled_scaled])  #␣
 ↪Combine labeled and unlabeled features
combined_labels = np.hstack([y_train, y_unlab_pred])  # Combine true and␣
 ↪predicted labels

# Call the t-SNE plotting function
plot_tsne_custom_color(scaled_features, combined_labels, title="t-SNE␣
 ↪Visualization of Labeled and Predicted Unlabeled Data")
```



```
[ ]: from matplotlib.colors import ListedColormap
     import matplotlib.pyplot as plt
     from openTSNE import TSNE
     import numpy as np
     from matplotlib import cm

     def plot_tsne_custom_color(features, labels, title="t-SNE Visualization"):
         """
         Generate t-SNE visualization using OpenTSNE with a custom color scheme.

         Parameters:
         - features: The feature matrix (e.g., encoded or raw features).
         - labels: Labels corresponding to the features (should be integers or␣
       ↪categories).
         - title: Title of the plot.
         """
         # Perform t-SNE with OpenTSNE
         tsne = TSNE(n_components=2, perplexity=30, n_iter=1000, random_state=42)
         tsne_result = tsne.fit(features)
```

```
    # Use a continuous colormap such as 'viridis' to match the color scheme in␣
↪the image
    cmap = plt.cm.viridis

    # Plot the results
    plt.figure(figsize=(8, 6))
    scatter = plt.scatter(tsne_result[:, 0], tsne_result[:, 1], c=labels,␣
↪cmap=cmap, s=10, alpha=0.8)
    plt.title(title)
    plt.xlabel('Dimension 1')
    plt.ylabel('Dimension 2')
    cbar = plt.colorbar(scatter, ticks=np.unique(labels))
    cbar.set_label('Labels')
    cbar.ax.set_yticklabels([str(label) for label in np.unique(labels)])
    plt.show()

# Features (scaled unlabeled data) and predicted labels
scaled_features = np.vstack([x_train_scaled_encoded, x_unlabeled_scaled])  #␣
↪Combine labeled and unlabeled features
combined_labels = np.hstack([y_train, y_unlab_pred])  # Combine true and␣
↪predicted labels

# Call the t-SNE plotting function
plot_tsne_custom_color(scaled_features, combined_labels, title="t-SNE␣
↪Visualization of Labeled and Predicted Unlabeled Data")
```



#Intial TSNE for Comparison

```
[ ]: def generate_unlabeled_predictions(x_unlab, encoder, predictor):
       x_unlab_encoded = encoder.predict(x_unlab)
       _, y_unlab_hat = predictor(x_unlab_encoded, training=False)
       y_unlab_pred = np.argmax(y_unlab_hat, axis=1)
       return y_unlab_pred
     '''
     performance metric -> generate unlabeled predictions -> inout prediction labels␣
      ↪for the unlabeled part of the dataset
     generate tsne for this new dataset
     '''
```

```
[ ]: '\nperformance metric -> generate unlabeled predictions -> inout prediction
     labels for the unlabeled part of the dataset\ngenerate tsne for this new
     dataset\n'
```

```
[ ]: pip install gradio
```

```
Collecting gradio
  Downloading gradio-5.7.0-py3-none-any.whl.metadata (16 kB)
Collecting aiofiles<24.0,>=22.0 (from gradio)
  Downloading aiofiles-23.2.1-py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: anyio<5.0,>=3.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (3.7.1)
Collecting fastapi<1.0,>=0.115.2 (from gradio)
  Downloading fastapi-0.115.5-py3-none-any.whl.metadata (27 kB)
Collecting ffmpy (from gradio)
  Downloading ffmpy-0.4.0-py3-none-any.whl.metadata (2.9 kB)
Collecting gradio-client==1.5.0 (from gradio)
  Downloading gradio_client-1.5.0-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.10/dist-
packages (from gradio) (0.27.2)
Requirement already satisfied: huggingface-hub>=0.25.1 in
/usr/local/lib/python3.10/dist-packages (from gradio) (0.26.2)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.10/dist-
packages (from gradio) (3.1.4)
Collecting markupsafe~=2.0 (from gradio)
  Downloading MarkupSafe-2.1.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x
86_64.whl.metadata (3.0 kB)
Requirement already satisfied: numpy<3.0,>=1.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (1.26.4)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.10/dist-
packages (from gradio) (3.10.11)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from gradio) (24.2)
Requirement already satisfied: pandas<3.0,>=1.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (11.0.0)
Requirement already satisfied: pydantic>=2.0 in /usr/local/lib/python3.10/dist-
packages (from gradio) (2.9.2)
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting python-multipart==0.0.12 (from gradio)
  Downloading python_multipart-0.0.12-py3-none-any.whl.metadata (1.9 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (6.0.2)
Collecting ruff>=0.2.2 (from gradio)
  Downloading ruff-0.8.0-py3-none-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (25 kB)
Collecting safehttpx<1.0,>=0.1.1 (from gradio)
  Downloading safehttpx-0.1.1-py3-none-any.whl.metadata (4.1 kB)
Collecting semantic-version~=2.0 (from gradio)
```

```
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
  Downloading starlette-0.41.3-py3-none-any.whl.metadata (6.0 kB)
Collecting tomlkit==0.12.0 (from gradio)
  Downloading tomlkit-0.12.0-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typer<1.0,>=0.12 in
/usr/local/lib/python3.10/dist-packages (from gradio) (0.13.0)
Requirement already satisfied: typing-extensions~=4.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (4.12.2)
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.32.1-py3-none-any.whl.metadata (6.6 kB)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from gradio-client==1.5.0->gradio) (2024.10.0)
Collecting websockets<13.0,>=10.0 (from gradio-client==1.5.0->gradio)
  Downloading websockets-12.0-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64
.manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.6 kB)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-
packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist-
packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-
packages (from anyio<5.0,>=3.0->gradio) (1.2.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-
packages (from httpx>=0.24.1->gradio) (2024.8.30)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-
packages (from httpx>=0.24.1->gradio) (1.0.7)
Requirement already satisfied: h11<0.15,>=0.13 in
/usr/local/lib/python3.10/dist-packages (from
httpcore==1.*->httpx>=0.24.1->gradio) (0.14.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub>=0.25.1->gradio) (3.16.1)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub>=0.25.1->gradio) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub>=0.25.1->gradio) (4.66.6)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas<3.0,>=1.0->gradio) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas<3.0,>=1.0->gradio) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas<3.0,>=1.0->gradio) (2024.2)
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.10/dist-packages (from pydantic>=2.0->gradio) (0.7.0)
Requirement already satisfied: pydantic-core==2.23.4 in
/usr/local/lib/python3.10/dist-packages (from pydantic>=2.0->gradio) (2.23.4)
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.10/dist-
packages (from typer<1.0,>=0.12->gradio) (8.1.7)
Requirement already satisfied: shellingham>=1.3.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from typer<1.0,>=0.12->gradio) (1.5.4)
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.10/dist-
packages (from typer<1.0,>=0.12->gradio) (13.9.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas<3.0,>=1.0->gradio) (1.16.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from
rich>=10.11.0->typer<1.0,>=0.12->gradio) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from
rich>=10.11.0->typer<1.0,>=0.12->gradio) (2.18.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub>=0.25.1->gradio) (3.4.0)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub>=0.25.1->gradio) (2.2.3)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich>=10.11.0->typer<1.0,>=0.12->gradio)
(0.1.2)
Downloading gradio-5.7.0-py3-none-any.whl (57.1 MB)
                         57.1/57.1 MB
14.8 MB/s eta 0:00:00
Downloading gradio_client-1.5.0-py3-none-any.whl (320 kB)
                         320.1/320.1 kB
20.5 MB/s eta 0:00:00
Downloading python_multipart-0.0.12-py3-none-any.whl (23 kB)
Downloading tomlkit-0.12.0-py3-none-any.whl (37 kB)
Downloading aiofiles-23.2.1-py3-none-any.whl (15 kB)
Downloading fastapi-0.115.5-py3-none-any.whl (94 kB)
                         94.9/94.9 kB
7.2 MB/s eta 0:00:00
Downloading
MarkupSafe-2.1.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25
kB)
Downloading ruff-0.8.0-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(11.1 MB)
                         11.1/11.1 MB
87.8 MB/s eta 0:00:00
Downloading safehttpx-0.1.1-py3-none-any.whl (8.4 kB)
Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Downloading starlette-0.41.3-py3-none-any.whl (73 kB)
                         73.2/73.2 kB
5.9 MB/s eta 0:00:00
Downloading uvicorn-0.32.1-py3-none-any.whl (63 kB)
                         63.8/63.8 kB
5.4 MB/s eta 0:00:00
Downloading ffmpy-0.4.0-py3-none-any.whl (5.8 kB)
Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Downloading websockets-12.0-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.m
anylinux_2_17_x86_64.manylinux2014_x86_64.whl (130 kB)
                         130.2/130.2 kB
10.9 MB/s eta 0:00:00
Installing collected packages: pydub, websockets, uvicorn, tomlkit,
semantic-version, ruff, python-multipart, markupsafe, ffmpy, aiofiles,
starlette, safehttpx, gradio-client, fastapi, gradio
  Attempting uninstall: markupsafe
    Found existing installation: MarkupSafe 3.0.2
    Uninstalling MarkupSafe-3.0.2:
      Successfully uninstalled MarkupSafe-3.0.2
Successfully installed aiofiles-23.2.1 fastapi-0.115.5 ffmpy-0.4.0 gradio-5.7.0
gradio-client-1.5.0 markupsafe-2.1.5 pydub-0.25.1 python-multipart-0.0.12
ruff-0.8.0 safehttpx-0.1.1 semantic-version-2.10.0 starlette-0.41.3
tomlkit-0.12.0 uvicorn-0.32.1 websockets-12.0
```

```
[ ]: import gradio as gr
     import pandas as pd
     import numpy as np
     from openTSNE import TSNE
     import matplotlib.pyplot as plt
     from matplotlib import colormaps
     from tensorflow.keras.models import load_model

     # Function to generate predictions for unlabeled data
     def generate_predictions_for_unlabeled(x_unlab, encoder, predictor):
         """Encode data and generate predictions for unlabeled samples."""
         encoded_data = encoder.predict(x_unlab)  # Encode data using the encoder
         predictions = predictor(encoded_data, training=False)  # Get the predictions
         predicted_classes = np.argmax(predictions, axis=1)  # Get predicted class␣
      ↪labels
         return predicted_classes


     # Function for t-SNE visualization
     def create_tsne_visualization(features, labels, title="t-SNE Visualization"):
         """Perform t-SNE dimensionality reduction and visualize with distinct␣
      ↪cluster colors."""
         tsne = TSNE(n_components=2, perplexity=30, n_iter=1000, random_state=42)
         tsne_result = tsne.fit(features)

         unique_labels = np.unique(labels)
         label_to_index = {label: idx for idx, label in enumerate(unique_labels)}
         color_indices = np.array([label_to_index[label] for label in labels])

         cmap = colormaps.get_cmap('tab10')  # Get the colormap
```

```
    fig, ax = plt.subplots(figsize=(8, 6))
    scatter = ax.scatter(
        tsne_result[:, 0],
        tsne_result[:, 1],
        c=color_indices,
        cmap=cmap,
        s=5,
        alpha=0.7
    )
    ax.set_title(title)
    ax.set_xlabel('t-SNE Dimension 1')
    ax.set_ylabel('t-SNE Dimension 2')

    # Add legend for cluster labels
    legend_handles = [
        plt.Line2D([], [], marker='o', color=cmap(idx / len(unique_labels)),␣
↪linestyle='', markersize=10)
        for idx in range(len(unique_labels))
    ]
    ax.legend(legend_handles, unique_labels, title="Clusters", loc="best",␣
↪bbox_to_anchor=(1, 1))

    return fig

# Function to process and visualize predictions and t-SNE
def process_and_visualize_data(start_row, end_row):
    """
    Process a subset of the x_unlabeled dataset, predict labels, and generate a␣
↪t-SNE visualization.
    """
    global x_unlabeled  # Ensure x_unlabeled is loaded

    # Parse input row indices
    start_row = int(start_row)
    end_row = int(end_row)

    # Select subset of data
    x_subset = x_unlabeled[start_row:end_row]

    # Load pre-trained encoder and predictor
    encoder = load_model(encoder_path)
    trained_model = encoder
    predictor = trained_model  # Assume predictor is preloaded

    # Predict labels for the subset
    predicted_labels = generate_predictions_for_unlabeled(x_subset, encoder,␣
↪predictor)
```

```
    # Create t-SNE visualization
    tsne_figure = create_tsne_visualization(x_subset, predicted_labels,␣
  ↪title="t-SNE Visualization of Subset")

    # Return visualization and predictions
    return tsne_figure, pd.DataFrame({"Predicted Labels": predicted_labels}).
  ↪head(10)

# Set up Gradio interface
inputs = [
    gr.Number(label="Start Row", value=3, precision=0),  # Starting row input
    gr.Number(label="End Row", value=109, precision=0)   # Ending row input
]

outputs = [
    gr.Plot(label="t-SNE Visualization"),               # t-SNE plot
    gr.Dataframe(label="Predicted Labels (Top 10)")     # Dataframe for␣
  ↪predicted labels
]

gr.Interface(
    fn=process_and_visualize_data,
    inputs=inputs,
    outputs=outputs,
    title="Self-Supervised Learning Visualization",
    description="Generate predictions and visualize data with t-SNE."
).launch(debug=True)
```

```
Running Gradio in a Colab notebook requires sharing enabled. Automatically
setting `share=True` (you can turn this off by setting `share=False` in
`launch()` explicitly).

Colab notebook detected. This cell will run indefinitely so that you can see
errors and logs. To turn off, set debug=False in launch().
* Running on public URL: https://df4a3f63116d66a9d1.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU
upgrades, run `gradio deploy` from the terminal in the working directory to
deploy to Hugging Face Spaces (https://huggingface.co/spaces)
```

```
<IPython.core.display.HTML object>
```

```
4/4          0s 11ms/step
```

1 Project Completed!

Infosys Springboard Project successfully completed by **Aniruddh Joshi**!

```
[ ]:
```