CYTOAUTOCLUSTER

Enhancing Cytometry with Deep Learning

INTRODUCTION :

CytoAutoCluster aims to integrate semi-supervised learning approaches within cytometry workflows. By utilizing both labeled and unlabeled data, this project seeks to develop a robust clustering algorithm that can adaptively learn from the inherent structure of cytometric data. This innovative approach not only aims to enhance the accuracy of cell classification but also to reduce the reliance on extensive labeled datasets, which can be labor-intensive and time-consuming to create.

PROBLEM OVERVIEW :

Cytometry generates vast amounts of high-dimensional data, often leading to challenges in the interpretation and classification of cell populations. Traditional clustering methods, such as k-means or hierarchical clustering, may struggle with the complexity and variability of cytometric data. Key challenges include:

1. **High Dimensionality**: Cytometric data often consists of multiple parameters, making it difficult to visualize and analyze using conventional methods.

2. **Label Scarcity**: High-quality labeled datasets are crucial for supervised learning but are often scarce in biological research, limiting the effectiveness of traditional machine learning approaches.

3. **Noise and Variability**: Cytometric data can be noisy and exhibit significant variability due to biological differences, which can adversely affect clustering performance.

OBJECTIVES :

The primary objectives of CytoAutoCluster are as follows:

1. **Develop a Semi-Supervised Learning Framework**: Create an algorithm that can effectively utilize both labeled and unlabeled data to enhance clustering performance in cytometric analyses.

2. **Improve Clustering Accuracy**: Implement deep learning techniques to achieve higher accuracy in classifying complex cell populations compared to traditional methods.

3. **Reduce Labeling Requirements**: Minimize the need for extensive labeled datasets by leveraging unlabeled data, thereby reducing the time and resources required for data preparation.

4. **Enhance Interpretability**: Provide tools and visualizations that help researchers understand the clustering results and the underlying biological significance of identified cell populations.

5. **Scalability and Efficiency**: Ensure that the developed algorithm is scalable and efficient, capable of handling large datasets commonly encountered in cytometry.

DAY-1 | 07-10-2024 :

- ✓ Referred and Downloaded a valid semi-supervised cytometry mass data set from resources like kaggle,google scholar,paperswithcode.

DAY-2 | 08-10-2024 :

- ✓ My data set got rejected because it contains complete labelled data as we need some I showcased my downloaded data set to the mentor and took my inputs.
- ✓ percentage of unlabelled data to perform semi supervised learning.
- ✓ As per the feedback from mentor, I must find mass cytometry data with more than 40 columns and unlabelled data must be more than 60% and it is to be completely in tabular format.

DAY-3 | 09-10-2024 :

- ✓ Gained a brief knowledge on the components of the cytometry in the data set with an sample data set (Levine_32dim_notransform.csv).
- ✓ Learned about some basic GIT commands to use in the repository of the github.
- ✓ Bagged the idea of how the code in collabed repository is used by the multiple users (push ,merge and commit).

DAY-4 | 10-10-2024 :

- ✓ Finalized the data set i.e., Levine_32dim_notransform.csv
- ✓ Created Python Environment and uploaded the data set , created a dataframe.

```
import numpy as np
import pandas as pd
import seaborn as sns
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
```

- Imported the Required Python Packages for Data Analytics.

```
#upload the data set
data=pd.read_csv('Levine_32dim_notransform.csv')
```

- Uploaded the data set.

```
#create a DataFrame
df=pd.DataFrame(data)
```

- Created a Data Frame for the data set.

DAY-5 | 11-10-2024 :

✓ Removed Unnecessary columns (Cell_length,file_number, event_number) from the data set.

```
data=data.drop(columns=['Time','file_number','event_number'])
```

- Dropped the Unnecessary Columns from the data set.

```
#Columns of the data set
df.columns
```

```
Index(['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', 'label', 'individual'],
      dtype='object')
```

- The Columns in the data set after removing the unnecessary columns.

✓ Performed EDA Techniques (Info,Histogram,Label and Unlabel percentage) and took inputs from the mentor.

```
#Information of data set
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 265627 entries, 0 to 265626
Data columns (total 38 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Cell_length  265627 non-null  int64
 1   DNA1         265627 non-null  float64
 2   DNA2         265627 non-null  float64
 3   CD45RA       265627 non-null  float64
 4   CD133        265627 non-null  float64
 5   CD19         265627 non-null  float64
 6   CD22         265627 non-null  float64
 7   CD11b        265627 non-null  float64
 8   CD4          265627 non-null  float64
 9   CD8          265627 non-null  float64
 10  CD34         265627 non-null  float64
 11  Flt3         265627 non-null  float64
 12  CD20         265627 non-null  float64
 13  CXCR4        265627 non-null  float64
 14  CD235ab      265627 non-null  float64
 15  CD45         265627 non-null  float64
 16  CD123        265627 non-null  float64
 17  CD321        265627 non-null  float64
 18  CD14         265627 non-null  float64
 19  CD33         265627 non-null  float64
 20  CD47         265627 non-null  float64
 21  CD11c        265627 non-null  float64
 22  CD7          265627 non-null  float64
 23  CD15         265627 non-null  float64
 24  CD16         265627 non-null  float64
 25  CD44         265627 non-null  float64
 26  CD38         265627 non-null  float64
 27  CD13         265627 non-null  float64
 28  CD3          265627 non-null  float64
 29  CD61         265627 non-null  float64
 30  CD117        265627 non-null  float64
 31  CD49d        265627 non-null  float64
 32  HLA-DR       265627 non-null  float64
 33  CD64         265627 non-null  float64
 34  CD41         265627 non-null  float64
 35  Viability    265627 non-null  float64
 36  label        104184 non-null  float64
 37  individual   265627 non-null  int64
dtypes: float64(36), int64(2)
memory usage: 77.0 MB
```

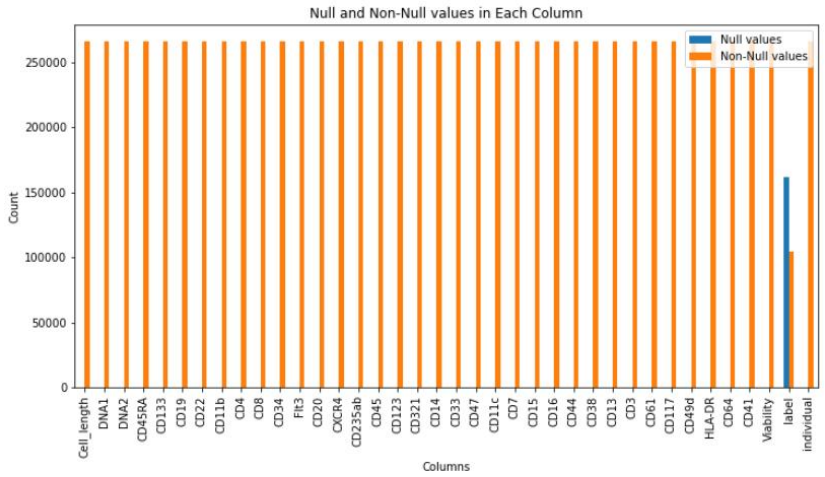
- The Information about the data type , non-null count of the data set.

```
# Calculate label and unlabel percentage
label_count = df['label'].count()
unlabel_count = df['label'].isna().sum()

label_percentage = (label_count / len(df)) * 100
unlabel_percentage = (unlabel_count / len(df)) * 100

print(f"Label percentage: {label_percentage:.2f}%")
print(f"Unlabel percentage: {unlabel_percentage:.2f}%")
```

```
Label percentage: 39.22%
Unlabel percentage: 60.78%
```

- Calculated the Label and Unlabel Percentage present in the label column so to futher perform semi-supervised learning.

```
#plot comparision of null values vs non-null values
null_counts=data.isnull().sum()
non_null_counts=data.notnull().sum()
plot_data=pd.DataFrame({"Null values": null_counts,
                        "Non-Null values": non_null_counts
                       })
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=90)
plt.legend(loc="upper right")
plt.show()
```

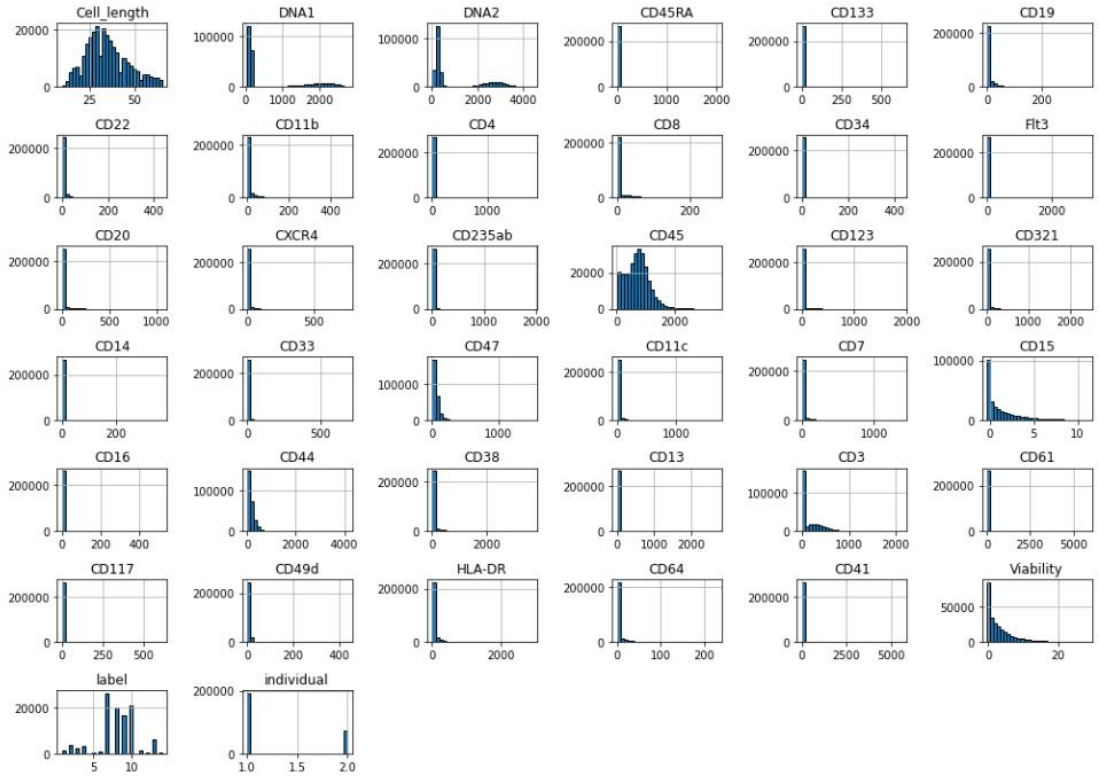


- The plotted graph representation so that to understand the null values present in the each column in the data set.

HISTOGRAM

A histogram is a type of bar chart that represents the frequency distribution of a dataset. It displays data by grouping it into bins or intervals, with each bin representing a range of values. The height of each bar shows the frequency or count of data points within that range, helping to visualize the distribution, spread, and central tendencies of the data.

```
#histrogram
df.hist(figsize=(14,10),bins=30,edgecolor='black')
plt.tight_layout()
plt.show()
```



- The Histogram representation of each feature values in the data set.

DAY-6 | 14-10-2024 :

✓ Performed EDA Techniques (Range of each feature , Correlation matrix, Class Label Distribution ,Box Plot) and took inputs from mentor.

```python
#Range of Each Feature in data set
for column in df.columns:
    feature_range = df[column].max() - df[column].min()
    print(f"Range of {column}: {feature_range}")
```

```
Range of Cell_length: 55
Range of DNA1: 2705.260757446289
Range of DNA2: 4373.519323348999
Range of CD45RA: 2013.4970097541834
Range of CD133: 629.0630275011063
Range of CD19: 367.64589342474915
Range of CD22: 435.891390711069
Range of CD11b: 481.8620219230657
Range of CD4: 1804.810034424064
Range of CD8: 273.4073664546013
Range of CD34: 430.49149709939906
Range of F1t3: 3083.149202555413
Range of CD20: 1062.0647517740726
Range of CXCR4: 744.96462726593
Range of CD235ab: 1925.8784293532397
Range of CD45: 3459.6277294158986
Range of CD123: 1914.2225522100975
Range of CD321: 2401.3572410941124
Range of CD14: 373.5840930938722
Range of CD33: 684.8300481736662
Range of CD47: 1508.6325097084045
Range of CD11c: 1698.3260714411736
Range of CD7: 1388.134843885896
Range of CD15: 11.344912439584778
Range of CD16: 520.675962328911
Range of CD44: 4108.539601758122
Range of CD38: 3675.5327078402042
Range of CD13: 2690.7746390104294
Range of CD3: 2131.9424919188073
Range of CD61: 5795.503212273121
Range of CD117: 613.3092513978481
Range of CD49d: 432.8393713831899
Range of HLA-DR: 2889.668938398361
Range of CD64: 229.35811600089048
Range of CD41: 5623.056521087885
Range of Viability: 28.55403268337251
Range of label: 13.0
Range of individual: 1
```

- Calculated the range value of each feature present in the data set
  i.e; range=(max value- min value)

```python
#class label distribution of data set
label_distribution = df['label'].value_counts()

print("Class Label Distribution:")
print(label_distribution)
```

```
Class Label Distribution:
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: label, dtype: int64
```

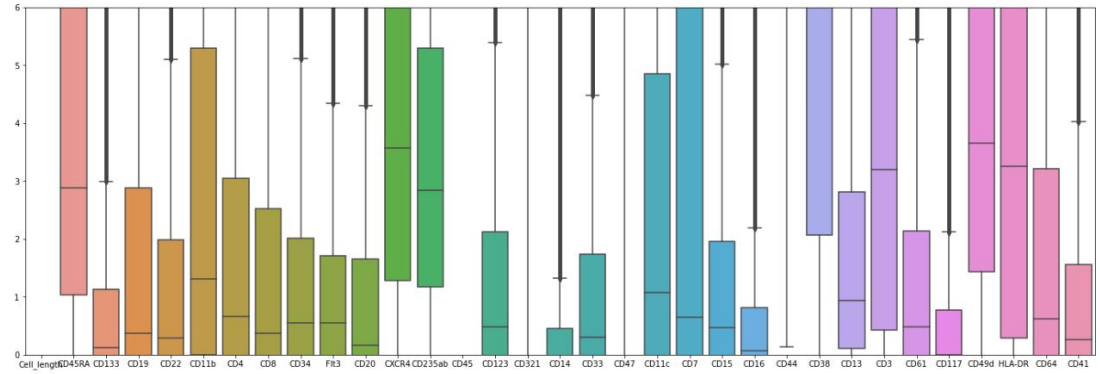
- Calculated the Label distribution in each feature of the data set.

BOXPLOT

Outliers in a boxplot are data points that fall outside the whiskers, typically defined as 1.5 times the interquartile range (IQR) above the third quartile or below the first quartile.

```
#box plot
fig, ax = plt.subplots(figsize=(24, 8))
sns.boxplot(data=df[['Cell_length','CD45RA','CD133','CD19','CD22','CD11b','CD4','CD8','CD34','Flt3','CD20','CXCR4','CD235ab','CD4
ax.set_ylim(0,6)
```

```
(0.0, 6.0)
```

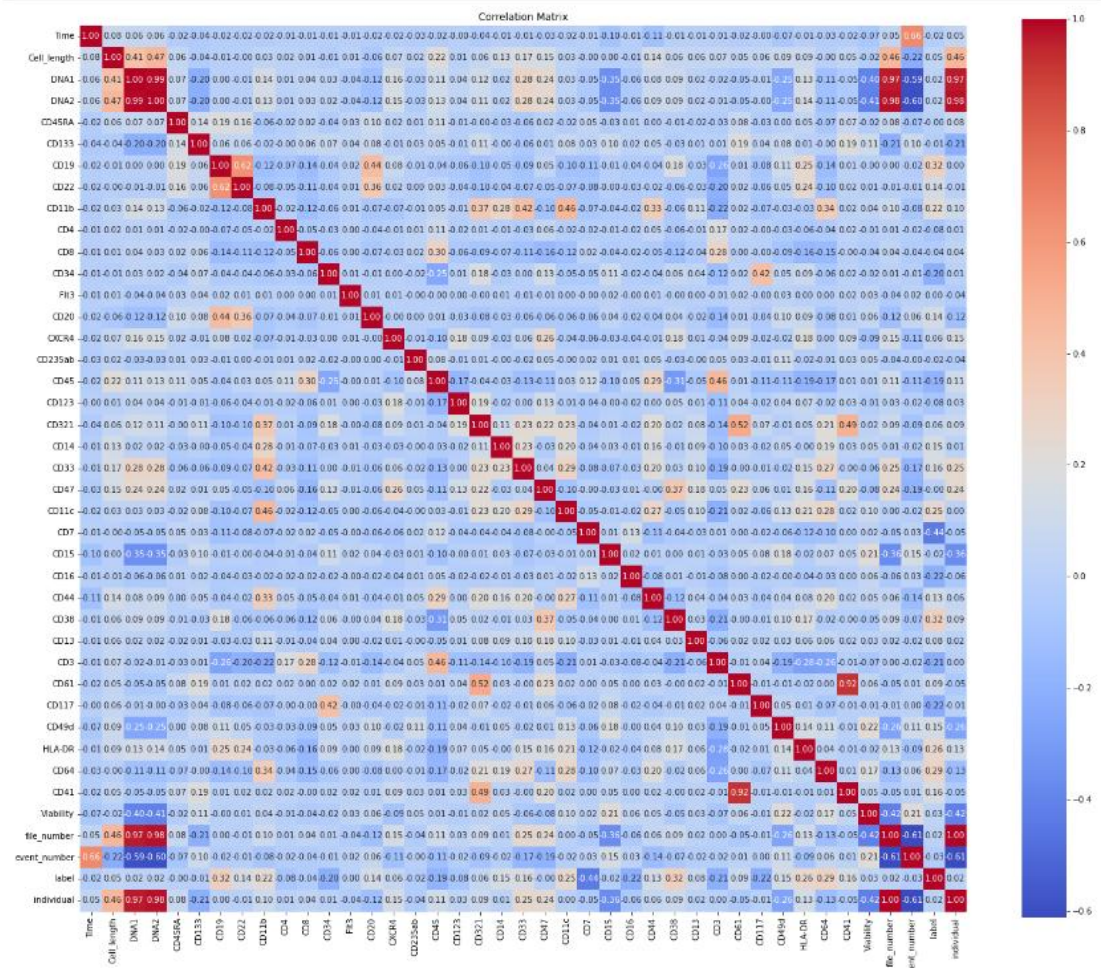


- The box-plot of all protien features represented within a range of values from 0 to 6.

CORRELATION MATRIX

A correlation matrix is a table displaying the correlation coefficients between multiple variables in a dataset, showing the degree and direction of linear relationships. Each cell in the matrix contains the correlation value between a pair of variables, ranging from -1 (perfect negative correlation) to +1 (perfect positive correlation), with 0 indicating no correlation. This matrix is a valuable tool for identifying relationships, multicollinearity, and feature dependencies using heatmaps to make patterns and strengths of relationships clearer.

```
correlation_matrix=df.corr()
plt.figure(figsize=(24,20))
sns.heatmap(correlation_matrix,annot=True,fmt='.2f',cmap='coolwarm',square=True)
plt.title('Correlation Matrix')
plt.show()
```



- Calculated the correlation value of each feature in the data set and represented it in the form of a correlated matrix using a heatmap so that the color represents the value of particular features is highly correlated or low or neutral.

DAY-7 | 15-10-2024 :

✓ Performed EDA Techniques (Kurtosis, Skewness , pair plot) and took inputs from my mentor.

KURTOSIS

Kurtosis measures the "tailedness" of a probability distribution, indicating how much data is in the tails compared to a normal distribution. There are three types of kurtosis:

1. **Mesokurtic**: This is a normal distribution with kurtosis close to zero, indicating average tail presence, like the bell curve.
2. **Leptokurtic**: Distributions with high kurtosis (>0) are leptokurtic. They have heavy tails, meaning more data falls in the tails, suggesting more outliers. This results in a sharp peak and flatter tails.
3. **Platykurtic**: Distributions with low kurtosis (<0) are platykurtic, with thin tails and fewer outliers. They have a flatter peak and less extreme values in the tails.

High kurtosis implies data is prone to extreme values, while low kurtosis shows a more consistent, predictable dataset.

```
#Calculate kurtosis of each featurebin the data set
from scipy.stats import kurtosis
# Calculate kurtosis for each column
kurtosis_values = df.apply(kurtosis, fisher=False)  # Fisher=False gives Pearson kurtosis (normal kurtosis = 3)

# Create a DataFrame with kurtosis values
kurtosis_df = pd.DataFrame({'Column': df.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)# 4.1 Histogram for each column
```

	Column	Kurtosis	Category
Cell_length	Cell_length	2.834033	Platykurtic (light tails)
DNA1	DNA1	2.534998	Platykurtic (light tails)
DNA2	DNA2	2.427060	Platykurtic (light tails)
CD45RA	CD45RA	8914.445042	Leptokurtic (heavy tails)
CD133	CD133	39762.277510	Leptokurtic (heavy tails)
CD19	CD19	33.044641	Leptokurtic (heavy tails)
CD22	CD22	76.673855	Leptokurtic (heavy tails)
CD11b	CD11b	41.254044	Leptokurtic (heavy tails)
CD4	CD4	17443.057242	Leptokurtic (heavy tails)
CD8	CD8	16.299477	Leptokurtic (heavy tails)
CD34	CD34	95.275706	Leptokurtic (heavy tails)
Flt3	Flt3	862.869549	Leptokurtic (heavy tails)
CD20	CD20	206.215616	Leptokurtic (heavy tails)
CXCR4	CXCR4	577.158590	Leptokurtic (heavy tails)
CD235ab	CD235ab	1785.466381	Leptokurtic (heavy tails)
CD45	CD45	3.652132	Leptokurtic (heavy tails)
CD123	CD123	398.071278	Leptokurtic (heavy tails)
CD321	CD321	533.082177	Leptokurtic (heavy tails)
CD14	CD14	16355.281414	Leptokurtic (heavy tails)
CD33	CD33	468.243060	Leptokurtic (heavy tails)
CD47	CD47	54.322868	Leptokurtic (heavy tails)
CD11c	CD11c	127.718580	Leptokurtic (heavy tails)
CD7	CD7	76.440845	Leptokurtic (heavy tails)
CD15	CD15	6.609792	Leptokurtic (heavy tails)
CD16	CD16	271.364229	Leptokurtic (heavy tails)
CD44	CD44	31.102720	Leptokurtic (heavy tails)
CD38	CD38	109.335486	Leptokurtic (heavy tails)
CD13	CD13	12442.953303	Leptokurtic (heavy tails)
CD3	CD3	4.910874	Leptokurtic (heavy tails)
CD61	CD61	394.035613	Leptokurtic (heavy tails)
CD117	CD117	7830.374159	Leptokurtic (heavy tails)
CD49d	CD49d	173.700138	Leptokurtic (heavy tails)
HLA-DR	HLA-DR	33.940301	Leptokurtic (heavy tails)
CD64	CD64	21.793802	Leptokurtic (heavy tails)
CD41	CD41	680.890565	Leptokurtic (heavy tails)
Viability	Viability	7.500375	Leptokurtic (heavy tails)
label	label	NaN	Mesokurtic (normal tails)
individual	individual	1.964382	Platykurtic (light tails)

- Calculated the Kurtosis value of each feature representing the category of kurtosis(platykurtic,leptokurtic,mesokurtic) present in the data set.

SKEWNESS

Skewness measures the asymmetry of a probability distribution. A perfectly symmetrical distribution has zero skewness, but real-world data often leans to one side. There are two main types:

1. **Right Skewness (Positive Skew)**: Here, the tail on the right side of the distribution is longer, meaning the majority of data points lie on the left. It indicates that the mean is typically greater than the median, and it's common in distributions with high outliers, like income data.

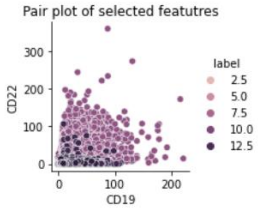
2. **Left Skewness (Negative Skew)**: In left-skewed distributions, the tail on the left side is longer, with most data points on the right. Here, the mean is often less than the median, and it occurs in data with low outliers, such as age at retirement.

```
# calculate skewness of each feature in the data set
from scipy.stats import skew

skewness = df.apply(skew)

# Function to categorize skewness
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)
```

	Skewness	Category
Cell_length	0.527832	Right-skewed
DNA1	1.155424	Right-skewed
DNA2	1.108669	Right-skewed
CD45RA	65.251655	Right-skewed
CD133	126.096395	Right-skewed
CD19	4.007221	Right-skewed
CD22	6.131244	Right-skewed
CD11b	5.264678	Right-skewed
CD4	114.022325	Right-skewed
CD8	3.313920	Right-skewed
CD34	8.397363	Right-skewed
Flt3	26.230625	Right-skewed
CD20	10.655454	Right-skewed
CXCR4	14.332247	Right-skewed
CD235ab	35.288190	Right-skewed
CD45	0.514492	Right-skewed
CD123	13.956222	Right-skewed
CD321	15.415273	Right-skewed
CD14	74.327532	Right-skewed
CD33	11.659128	Right-skewed
CD47	4.327074	Right-skewed
CD11c	7.679567	Right-skewed
CD7	7.405451	Right-skewed
CD15	1.860822	Right-skewed
CD16	14.520519	Right-skewed
CD44	3.436531	Right-skewed
CD38	7.733425	Right-skewed
CD13	99.104480	Right-skewed
CD3	1.479010	Right-skewed
CD61	17.909078	Right-skewed
CD117	54.242959	Right-skewed
CD49d	6.622882	Right-skewed
HLA-DR	4.612054	Right-skewed
CD64	3.752616	Right-skewed
CD41	22.140511	Right-skewed
Viability	2.013592	Right-skewed
label	NaN	Approximately symmetrical
individual	0.982030	Right-skewed

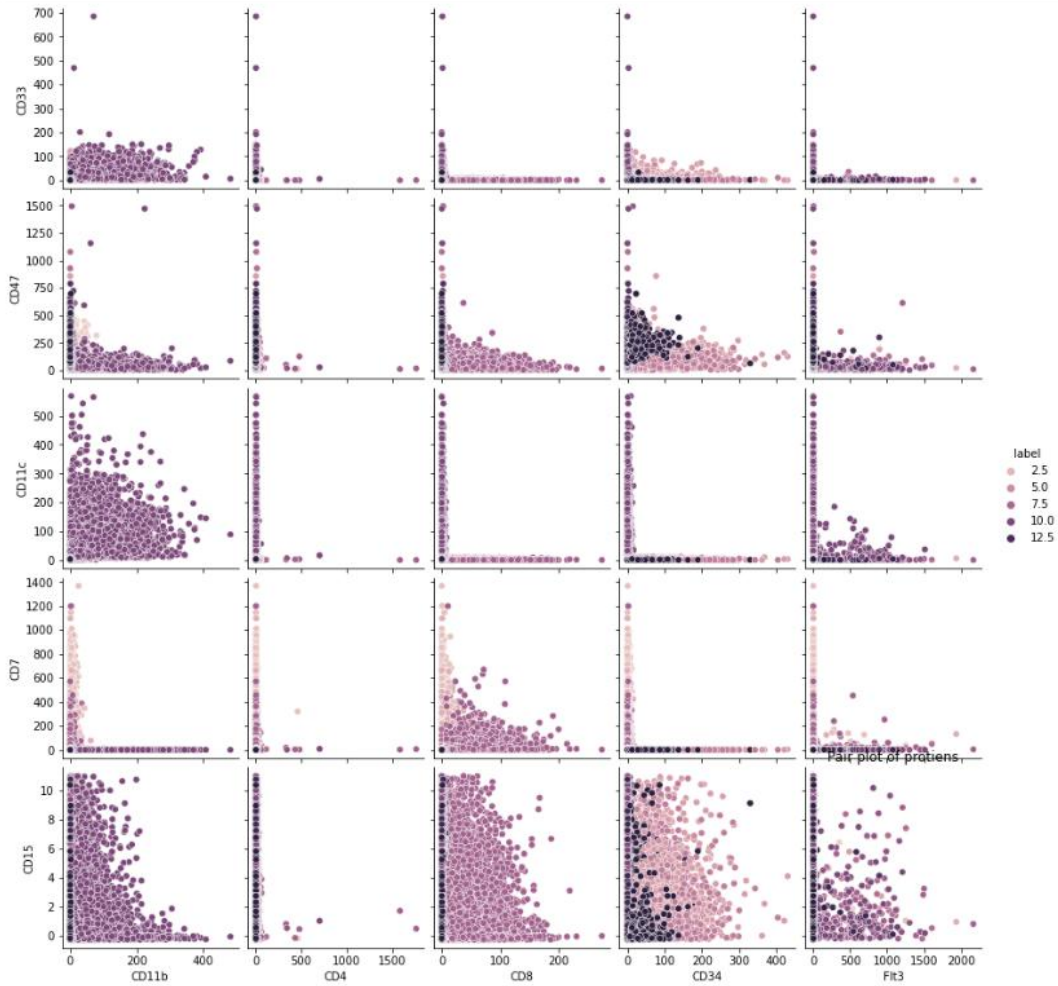
- Calculated the skewness values of each feature in the data set i.e(right-skewed or left skewed or symmetrical.

```
#pairplot
import seaborn as sns
sns.pairplot(df,hue='label',x_vars=['CD19'],y_vars=['CD22'])
plt.title('Pair plot of selected featutres')
plt.show()
```



- Calculated a pair plot for two selected protiens and the plot representing the label values in different colors I.e; the darker the color the higher the label value , the lighter the color the lower the label value and vice versa.

```
sns.pairplot(df,hue='label',x_vars=['CD11b','CD4','CD8','CD34','Flt3'],y_vars=['CD33','CD47','CD11c','CD7','CD15'])
plt.title('Pair plot of protiens')
plt.show()
```



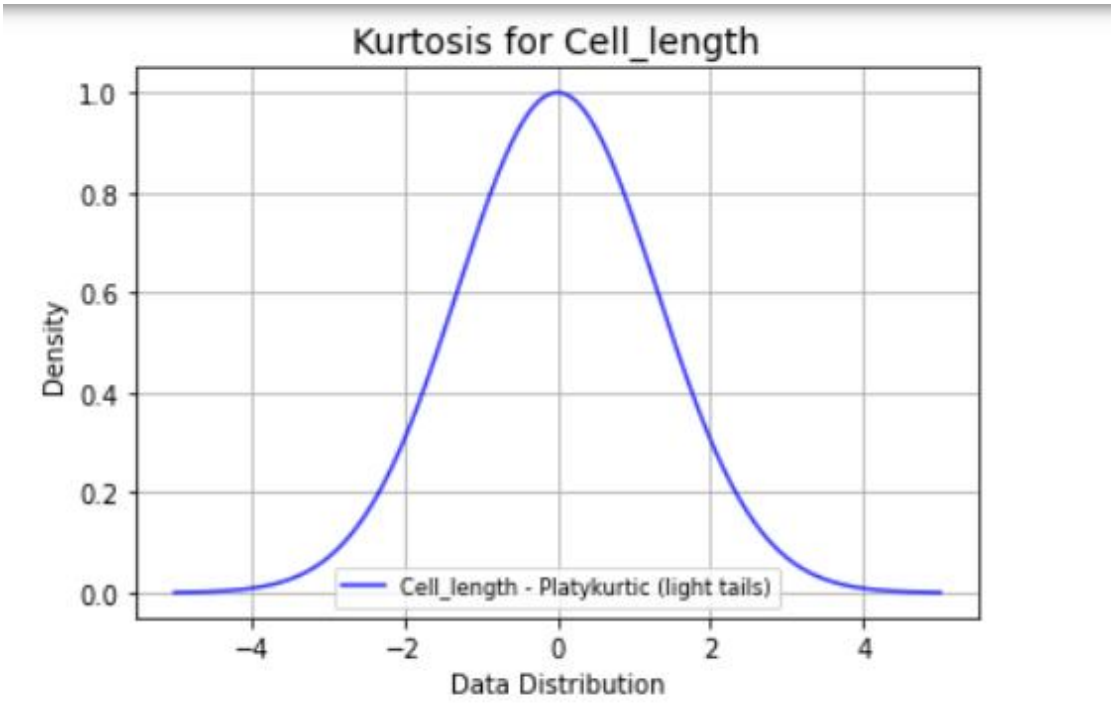
- Calculated a pair plot for 10 selected protiens and the plot representing the label values in different colors I.e; the darker the color the higher the label value , the lighter the color the lower the label value and vice versa.

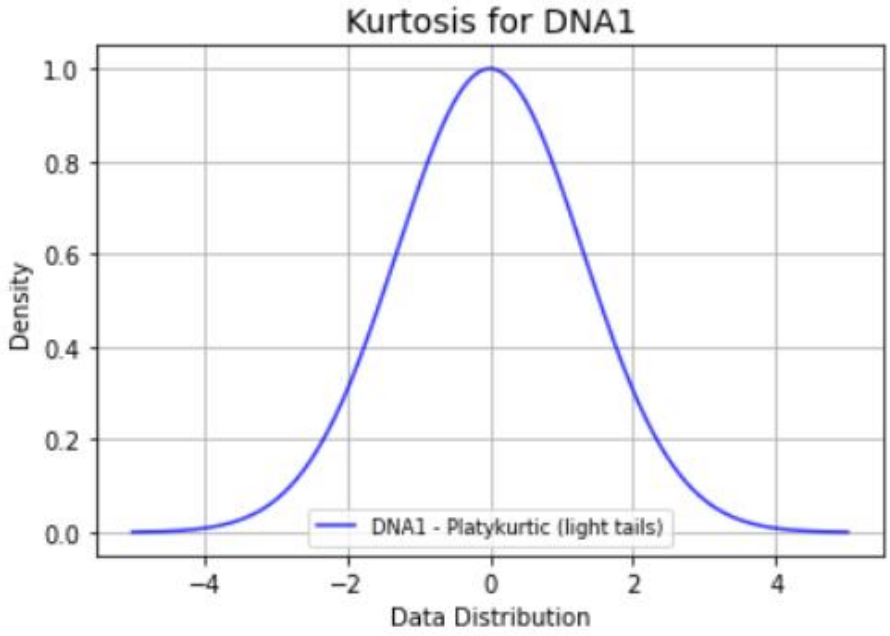
DAY-8 | 16-10-2024 :

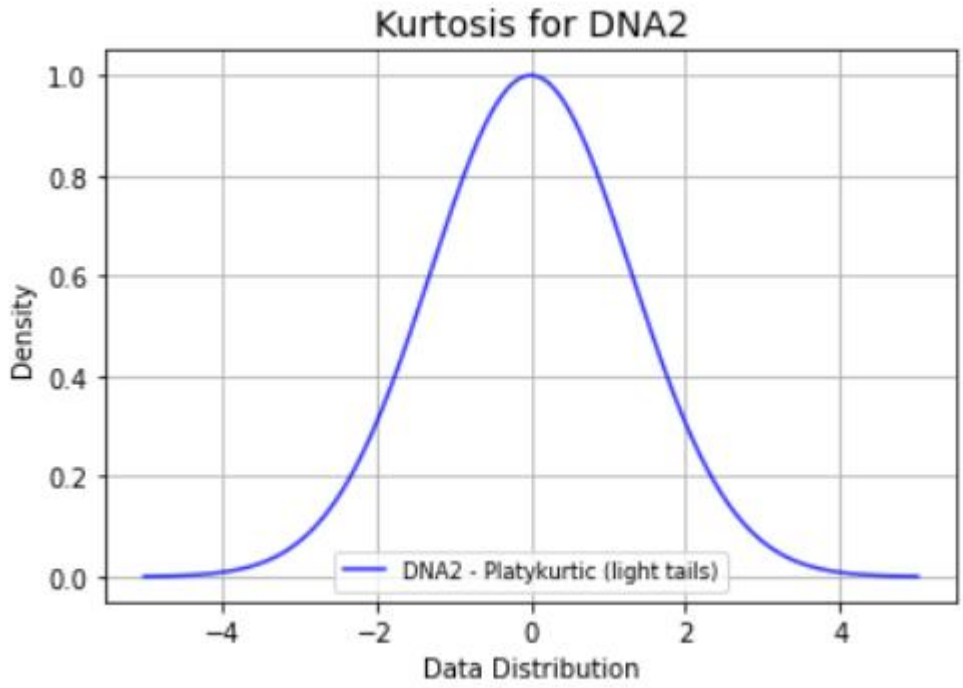
✓ Performed the individual plotting of each feature of kurtosis and skewness in the data set.

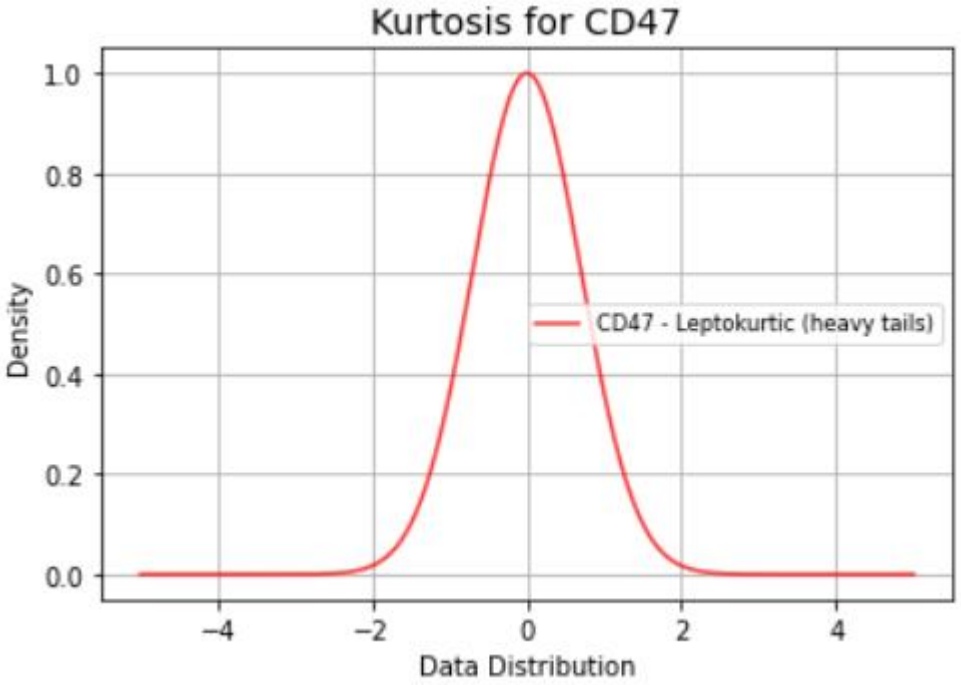
```python
# graph(kurtosis) of each individual feature in the data set
import numpy as np
import matplotlib.pyplot as plt

# Features and their respective kurtosis values
features = ['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', 'individual']

kurtosis_values = [2.834033, 2.534998, 2.427060, 8914.445042, 39762.277510, 33.044641, 76.673855, 41.254044,
                   17443.057242, 16.299477, 95.275706, 862.869549, 206.215616, 577.158590, 1785.466381, 3.652132,
                   398.071278, 533.082177, 16355.281414, 468.243060, 54.322868, 127.718580, 76.440845, 6.609792,
                   271.364229, 31.102720, 109.335486, 12442.953303, 4.910874, 394.035613, 7830.374159, 173.700138,
                   33.940301, 21.793802, 680.890565, 7.500375, 1.964382]

# Categories based on kurtosis values
categories = ['Platykurtic (light tails)', 'Platykurtic (light tails)', 'Platykurtic (light tails)',
              'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)',
              'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)',
              'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)',
              'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)',
              'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)',
              'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)',
              'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)',
              'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)',
              'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)',
              'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)',
              'Leptokurtic (heavy tails)', 'Leptokurtic (heavy tails)', 'Mesokurtic (normal tails)',
              'Platykurtic (light tails)']

# Generate x-axis for curve plotting
x = np.linspace(-5, 5, 1000)

# Different colors for categories
colors = {'Leptokurtic (heavy tails)': 'red', 'Platykurtic (light tails)': 'blue', 'Mesokurtic (normal tails)': 'green'}

# Plot individual graphs for each feature
for i in range(len(features)):
    plt.figure(figsize=(6, 4))

    if categories[i] == 'Leptokurtic (heavy tails)':
        y = np.exp(-0.5 * (x / 0.7) ** 2)  # More peaked for leptokurtic
    elif categories[i] == 'Platykurtic (light tails)':
        y = np.exp(-0.5 * (x / 1.3) ** 2)  # Flatter for platykurtic
    else:
        y = np.exp(-0.5 * (x / 1.0) ** 2)  # Normal for mesokurtic

    # Plot each feature's kurtosis curve
    plt.plot(x, y, color=colors[categories[i]], label=f'{features[i]} - {categories[i]}', alpha=0.8)

    # Add title and labels
    plt.title(f'Kurtosis for {features[i]}', fontsize=14)
    plt.xlabel('Data Distribution')
    plt.ylabel('Density')

    # Display the legend
    plt.legend(loc="best", fontsize='small')

    # Show grid
    plt.grid(True)

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

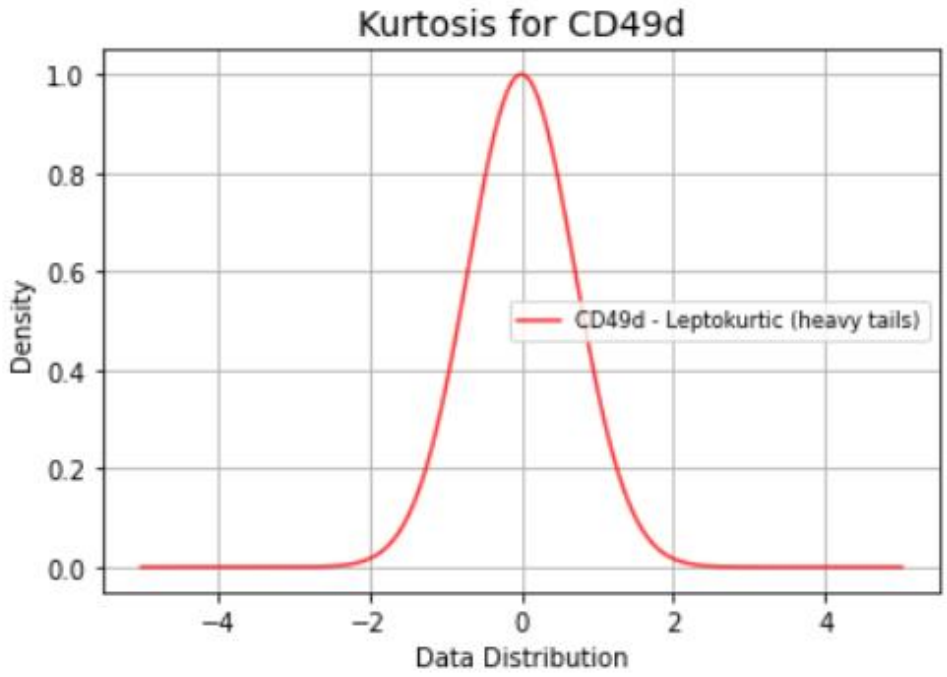
- Code to create individual plots to each feature representing the type of the kurtosis for that particular feature with a unique color for each type respectively in the data set.

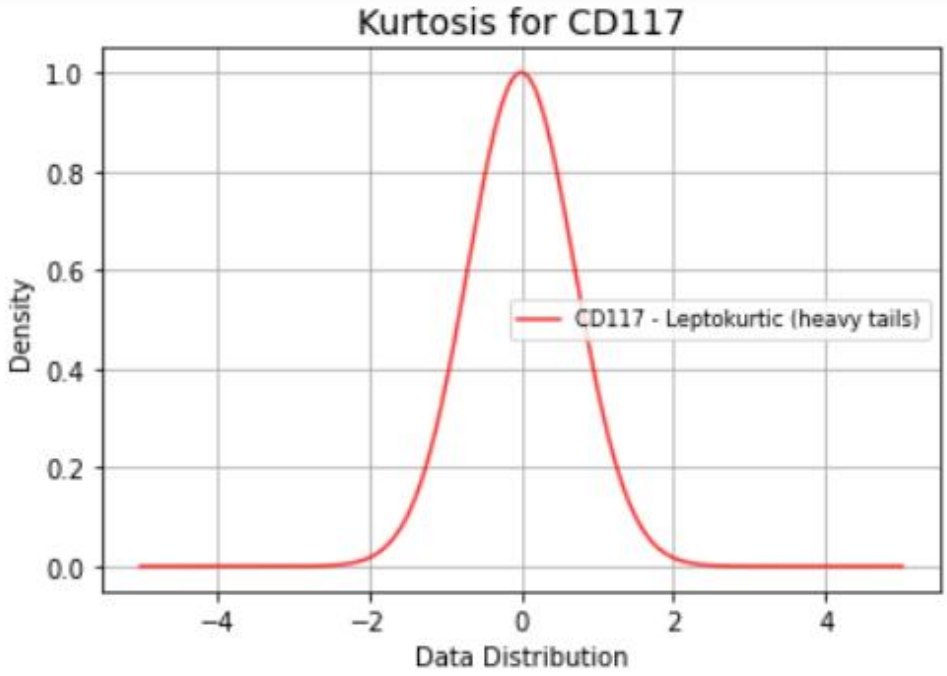

Kurtosis for Cell_length
Density
Data Distribution
Cell_length - Platykurtic (light tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

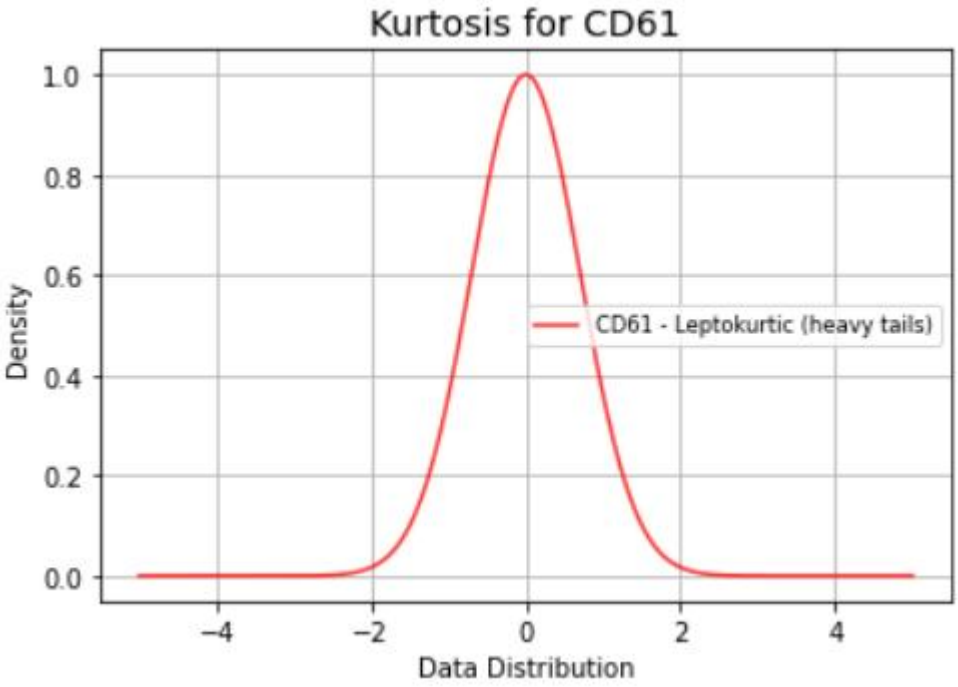

Kurtosis for DNA1
Density
Data Distribution
DNA1 - Platykurtic (light tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

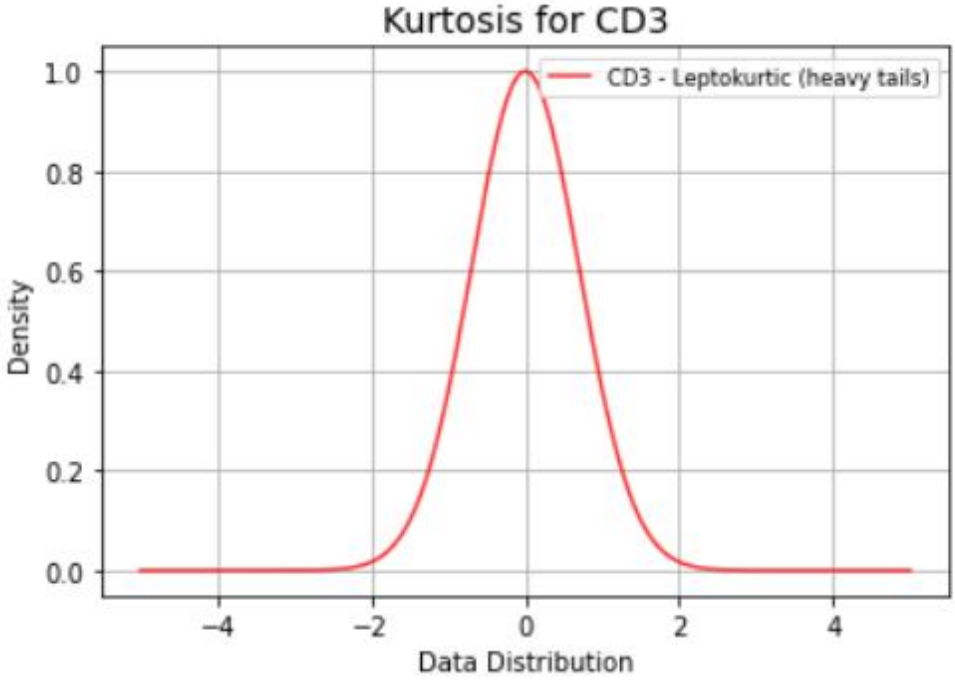

Kurtosis for DNA2
Density
Data Distribution
DNA2 - Platykurtic (light tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

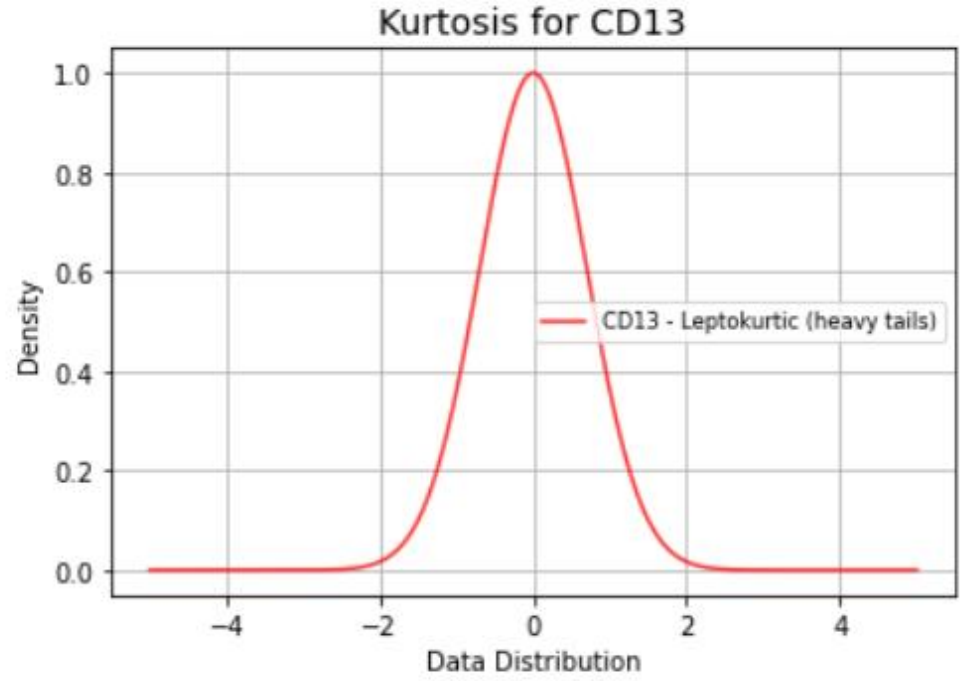

Kurtosis for CD47
Density
Data Distribution
CD47 - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

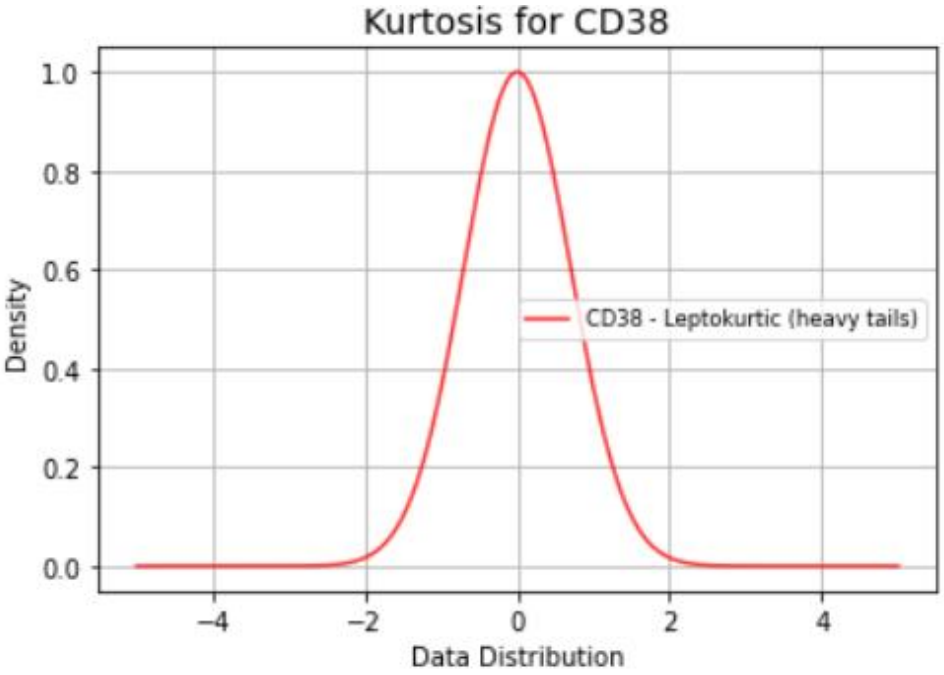

Kurtosis for CD49d
1.0
0.8
0.6
0.4
0.2
0.0
Density
CD49d - Leptokurtic (heavy tails)
−4
−2
0
2
4
Data Distribution

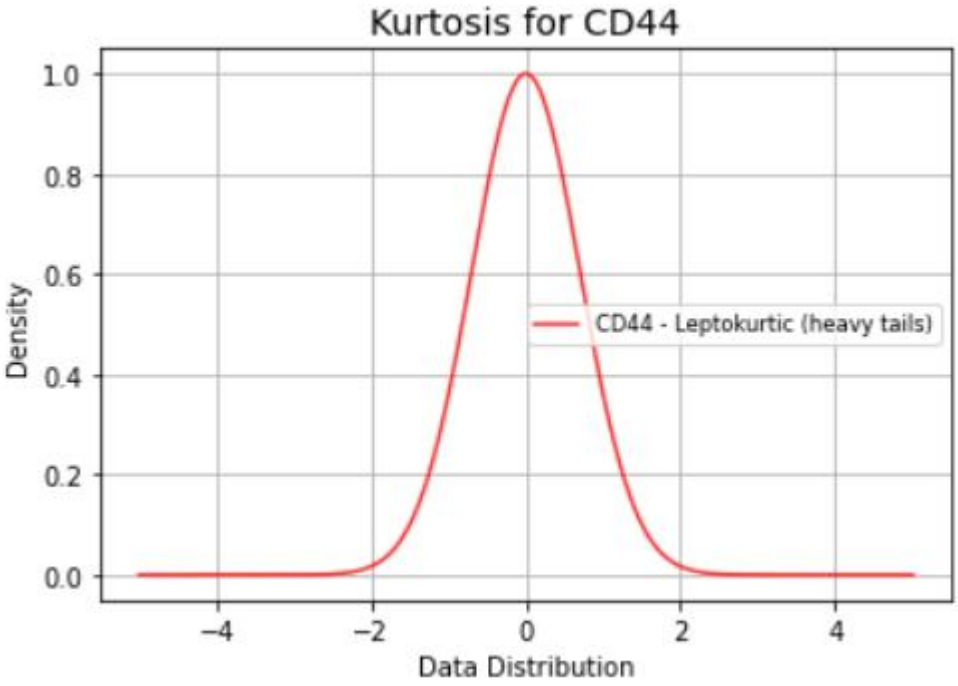

Kurtosis for CD117
1.0
0.8
0.6
0.4
0.2
0.0
Density
CD117 - Leptokurtic (heavy tails)
−4
−2
0
2
4
Data Distribution

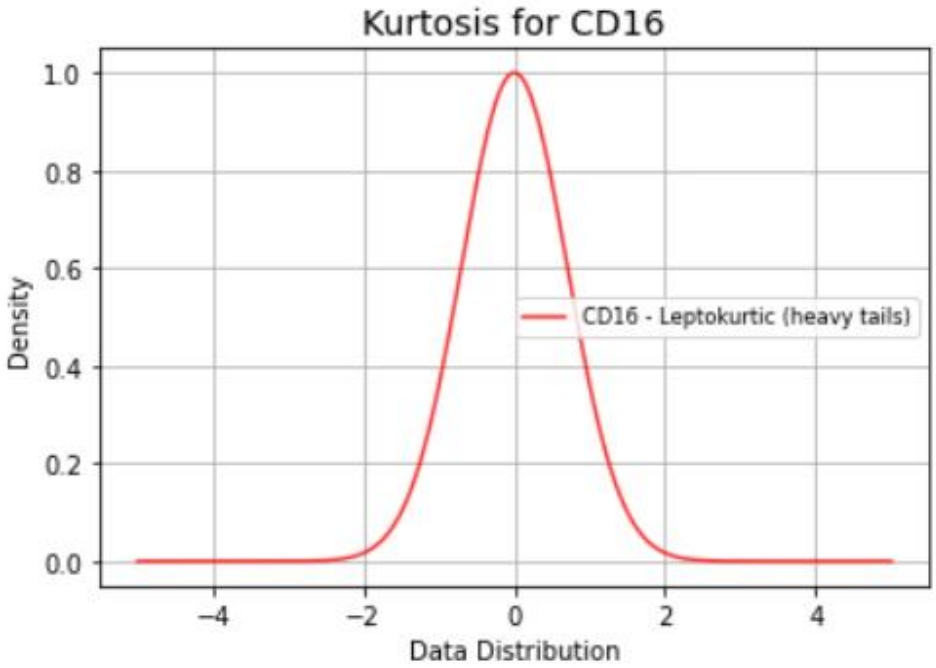

Kurtosis for CD61
CD61 - Leptokurtic (heavy tails)
Density
Data Distribution
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

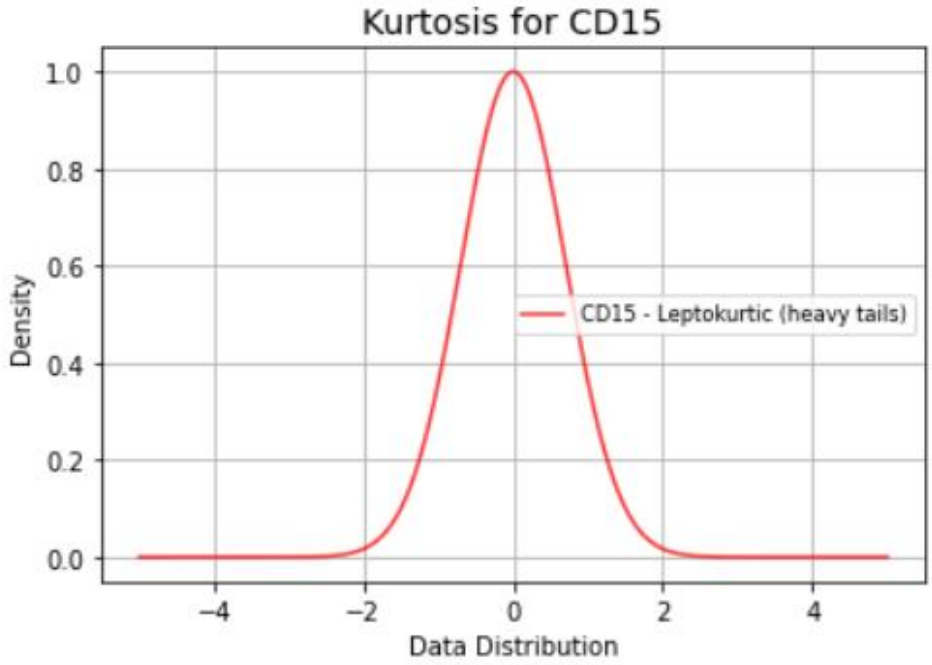

Kurtosis for CD3
CD3 - Leptokurtic (heavy tails)
Density
Data Distribution
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

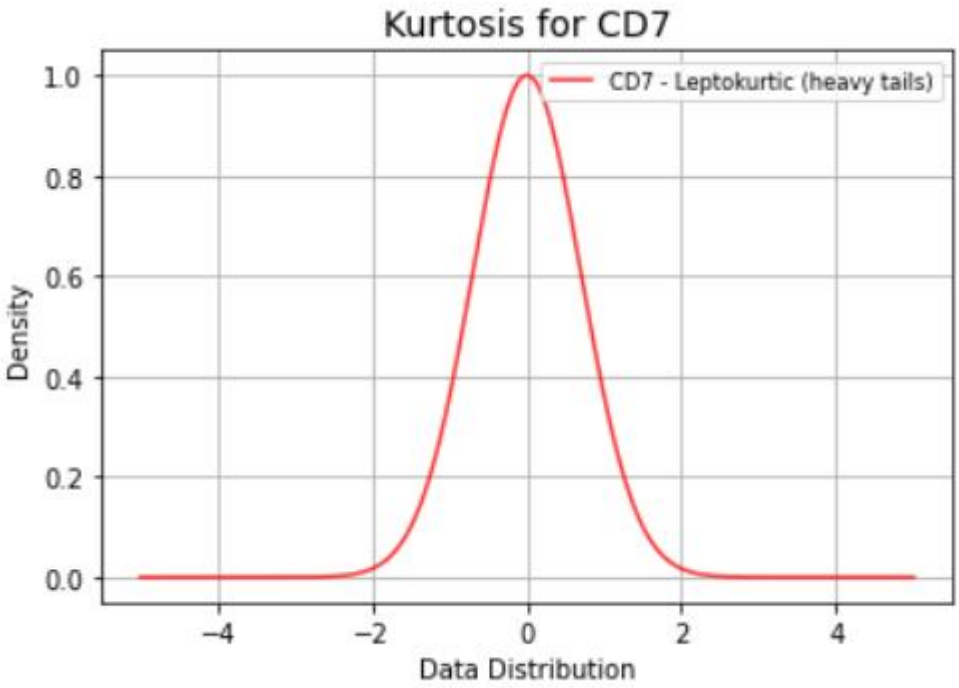

Kurtosis for CD13
Density
Data Distribution
CD13 - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

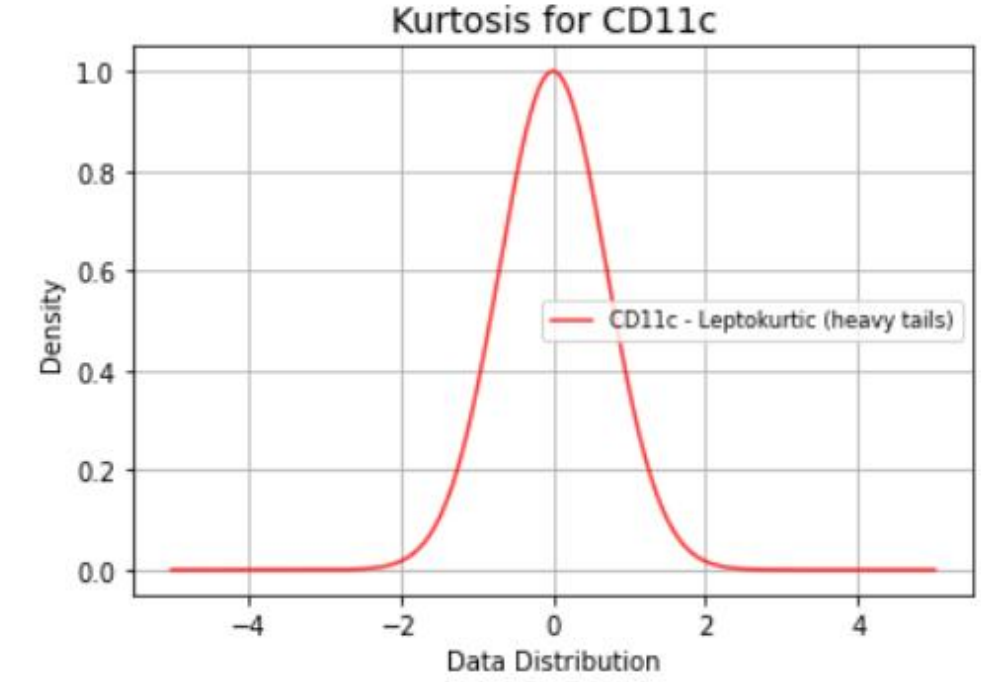

Kurtosis for CD38
Density
Data Distribution
CD38 - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

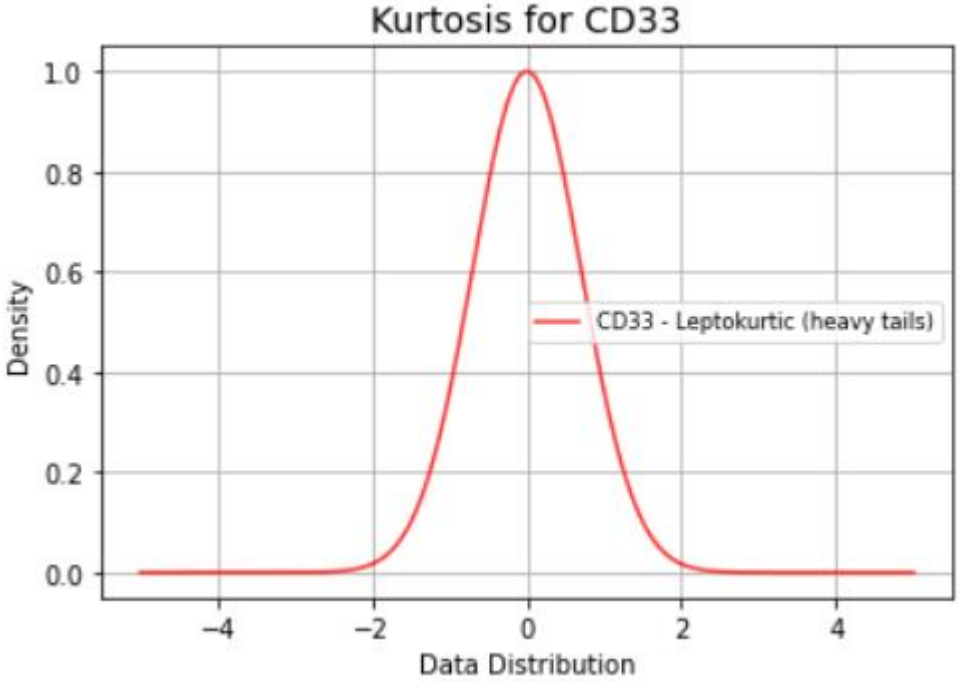

Kurtosis for CD44
1.0
0.8
0.6
0.4
0.2
0.0
Density
CD44 - Leptokurtic (heavy tails)
−4
−2
0
2
4
Data Distribution

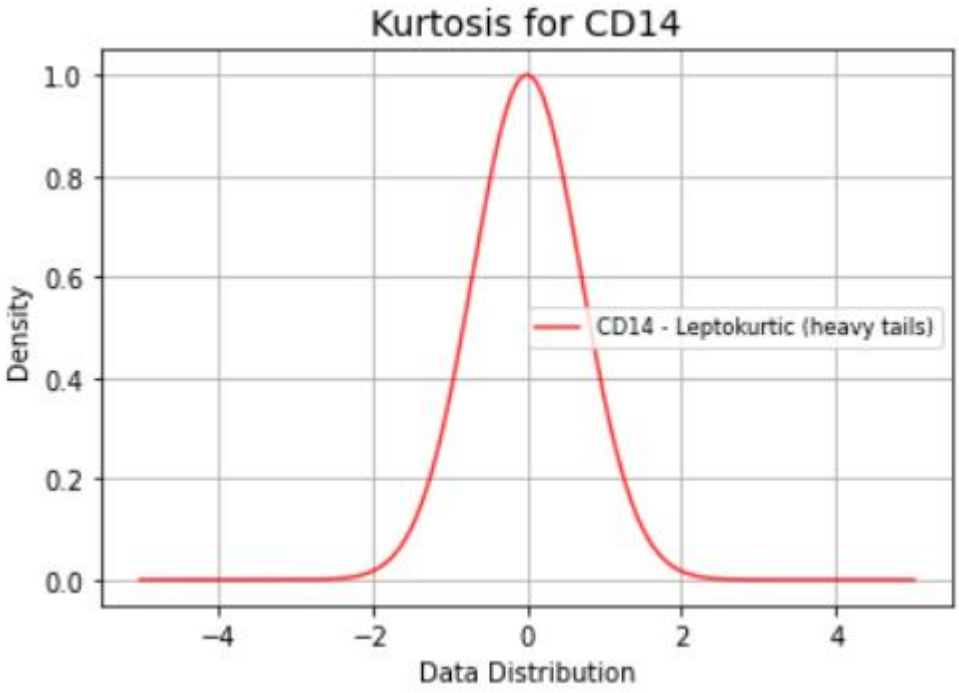

Kurtosis for CD16
1.0
0.8
0.6
0.4
0.2
0.0
Density
CD16 - Leptokurtic (heavy tails)
−4
−2
0
2
4
Data Distribution

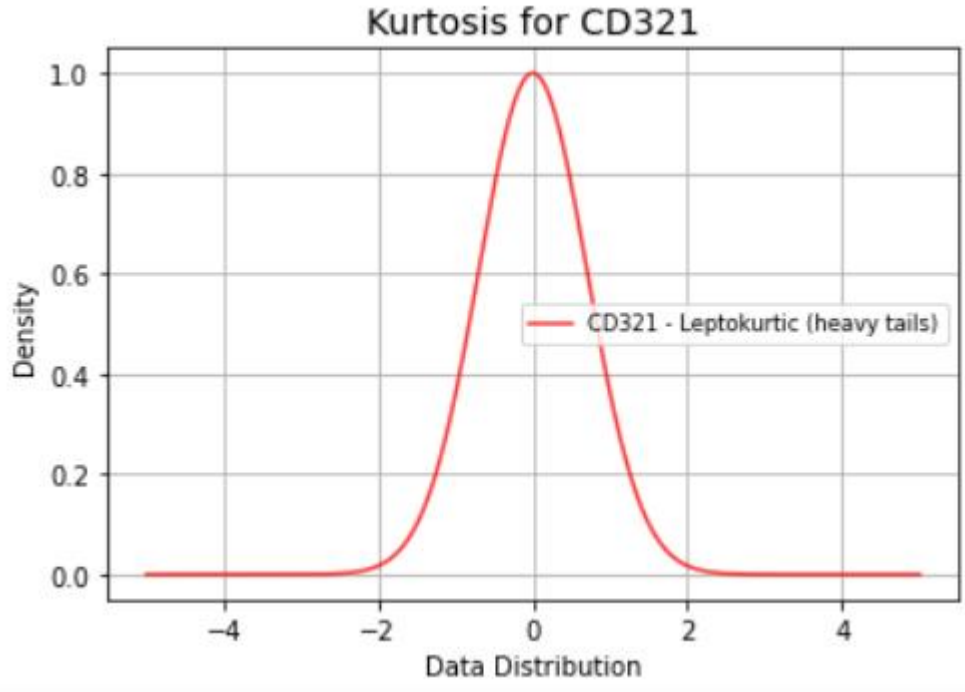

Kurtosis for CD15
CD15 - Leptokurtic (heavy tails)
Density
Data Distribution
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

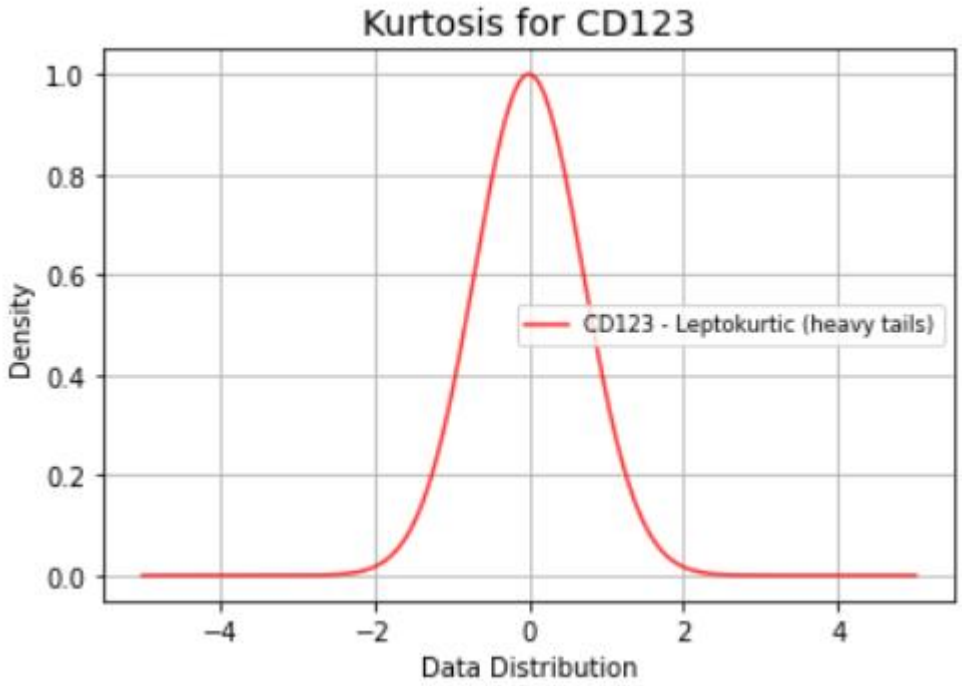

Kurtosis for CD7
CD7 - Leptokurtic (heavy tails)
Density
Data Distribution
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

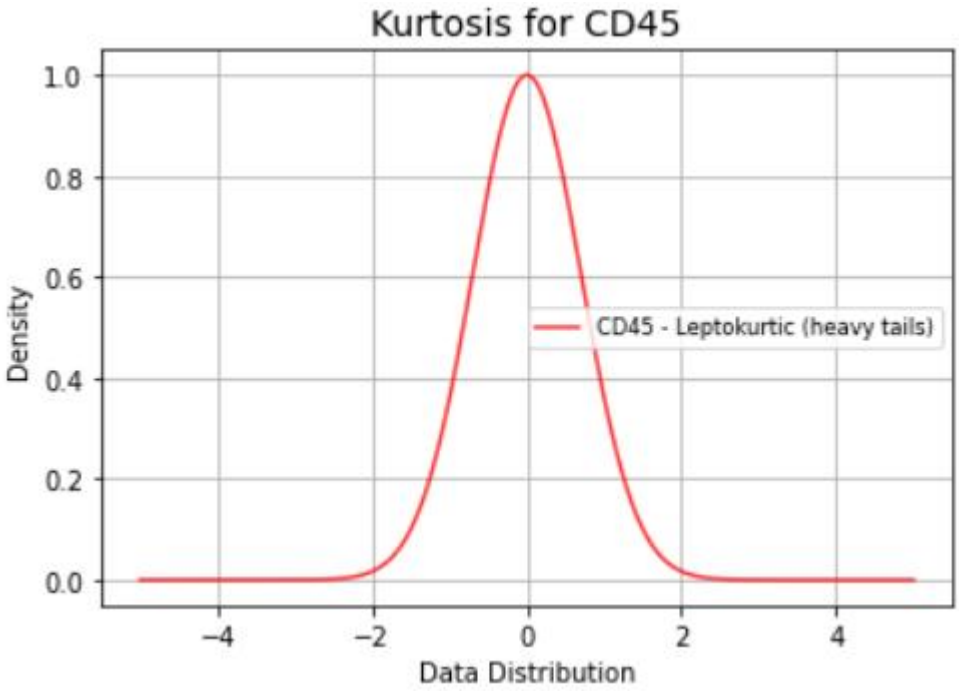

Kurtosis for CD11c
Density
Data Distribution
CD11c - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

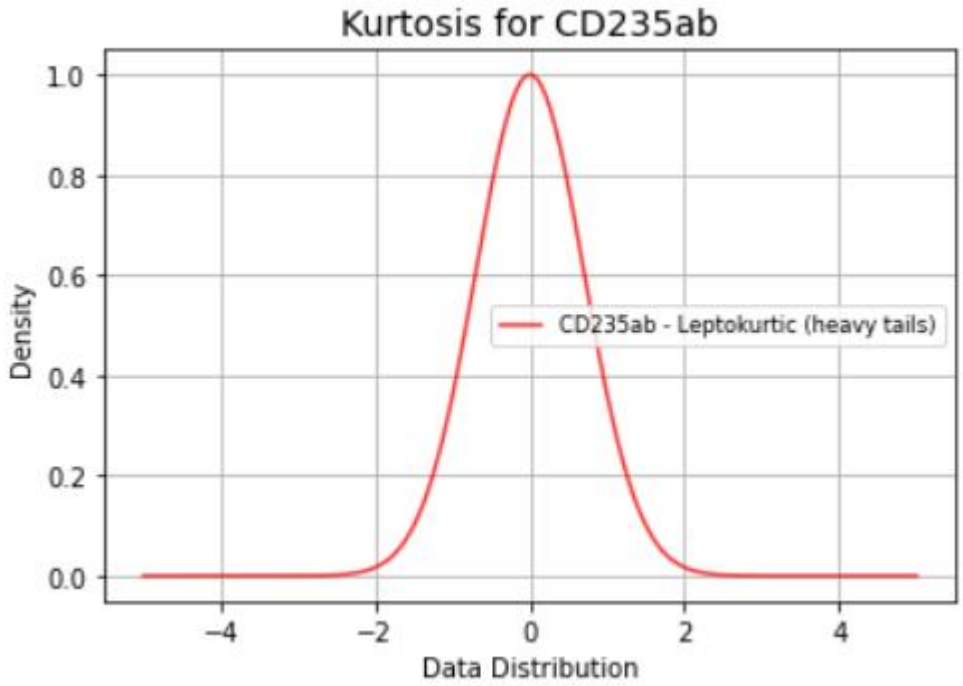

Kurtosis for CD33
Density
Data Distribution
CD33 - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

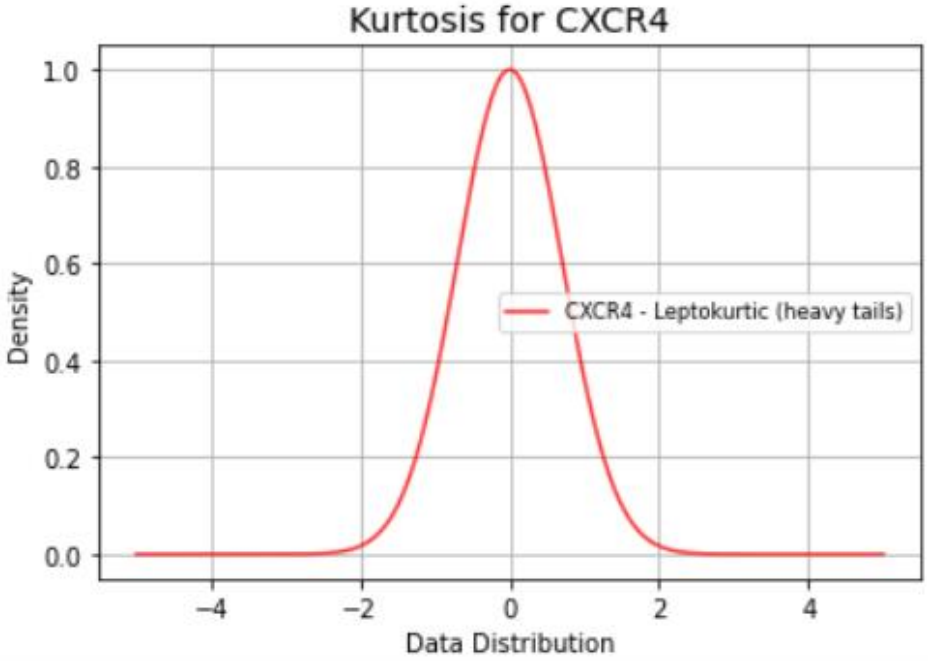

Kurtosis for CD14
Density
Data Distribution
CD14 - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

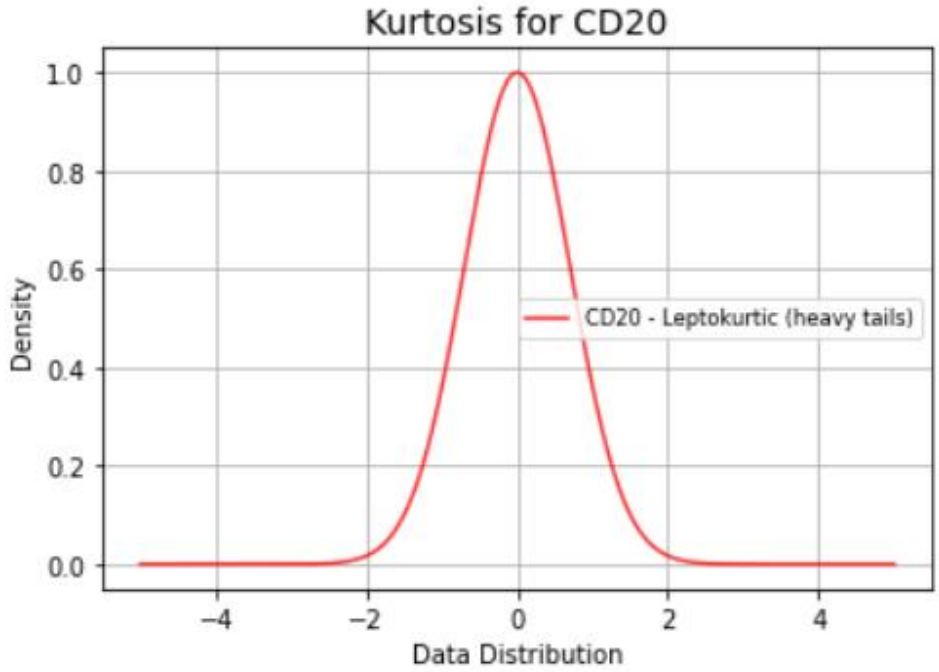

Kurtosis for CD321
Density
Data Distribution
CD321 - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

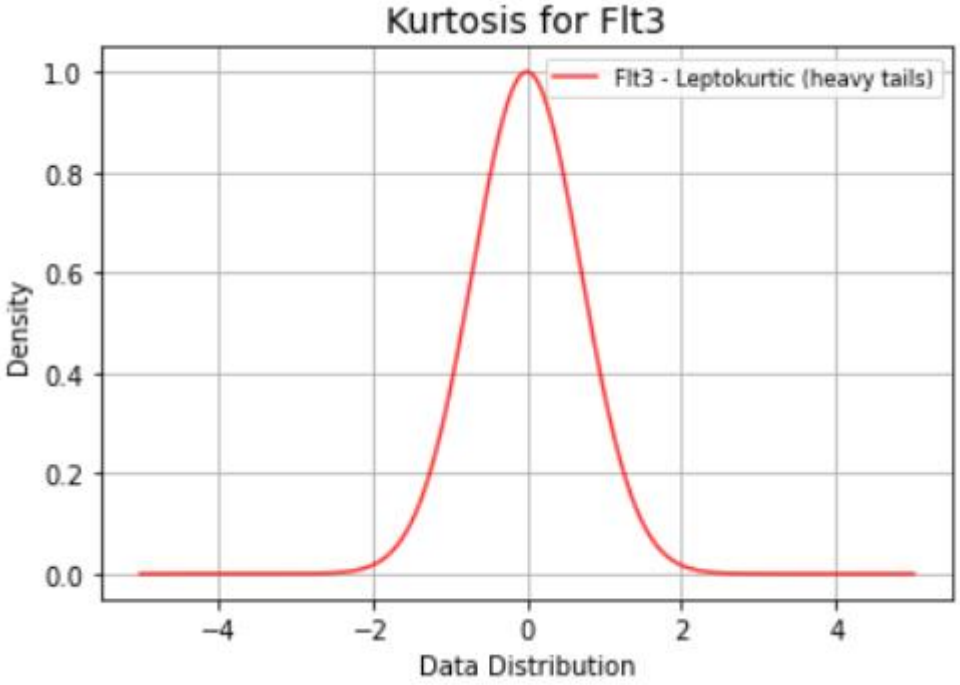

Kurtosis for CD123
Density
Data Distribution
CD123 - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

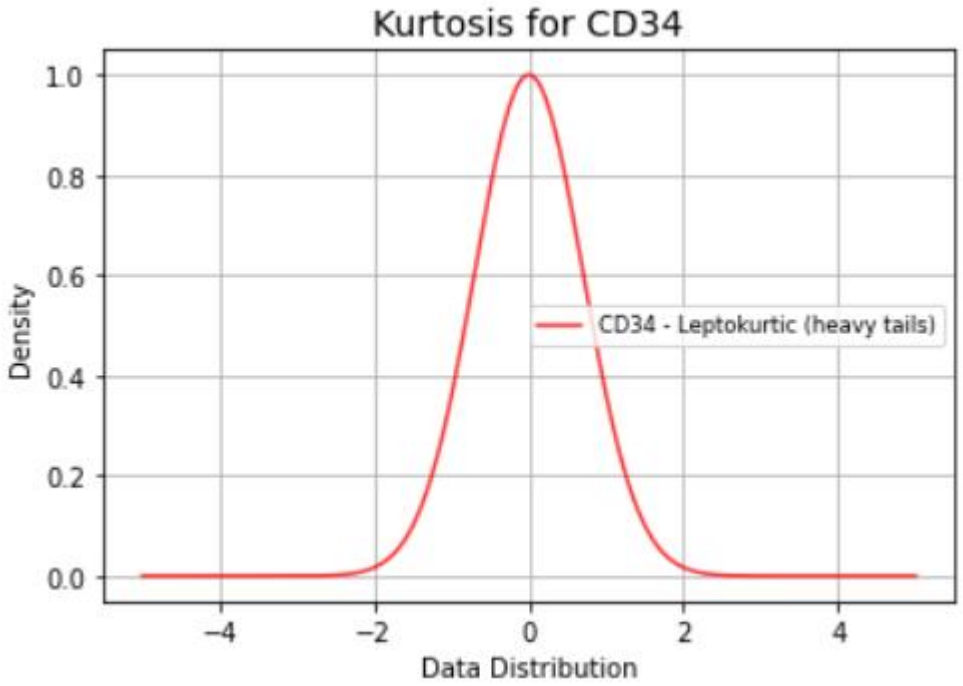

Kurtosis for CD45
Density
Data Distribution
CD45 - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

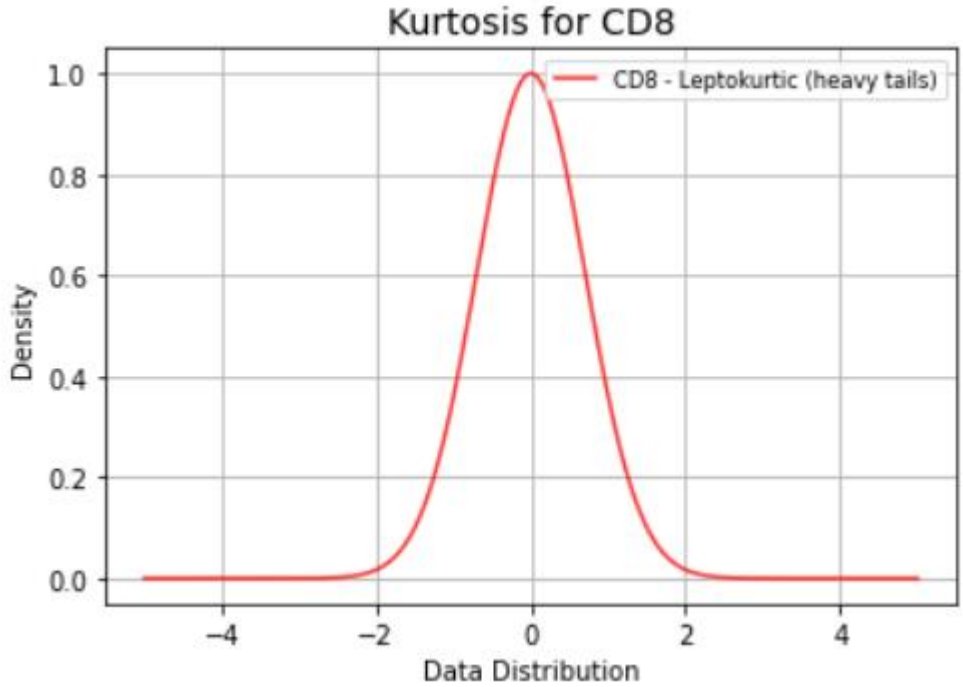

Kurtosis for CD235ab
Density
Data Distribution
CD235ab - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

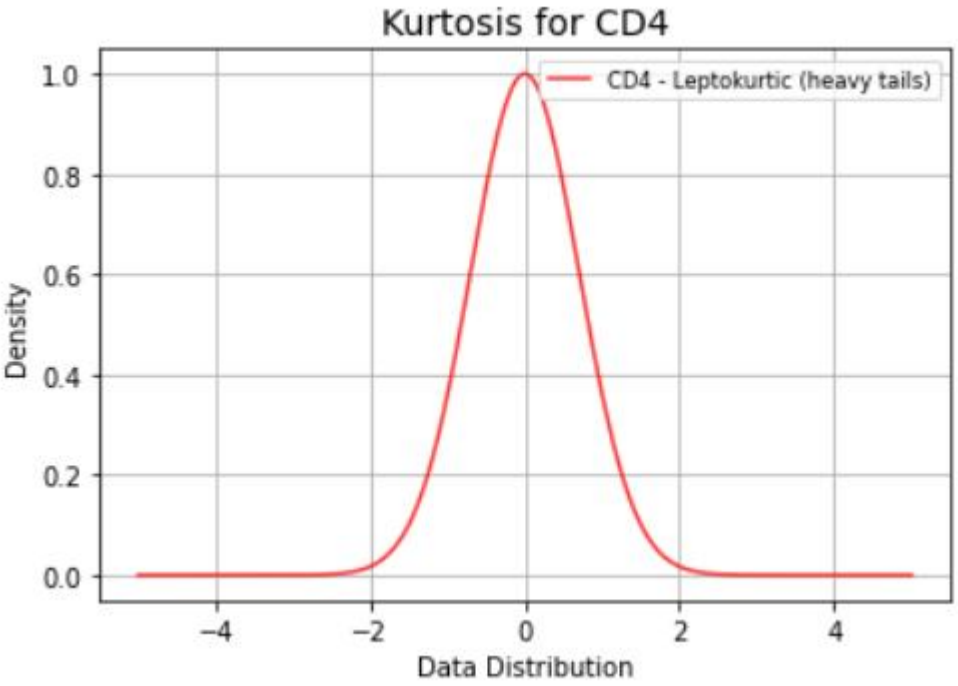

Kurtosis for CXCR4
Density
Data Distribution
CXCR4 - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

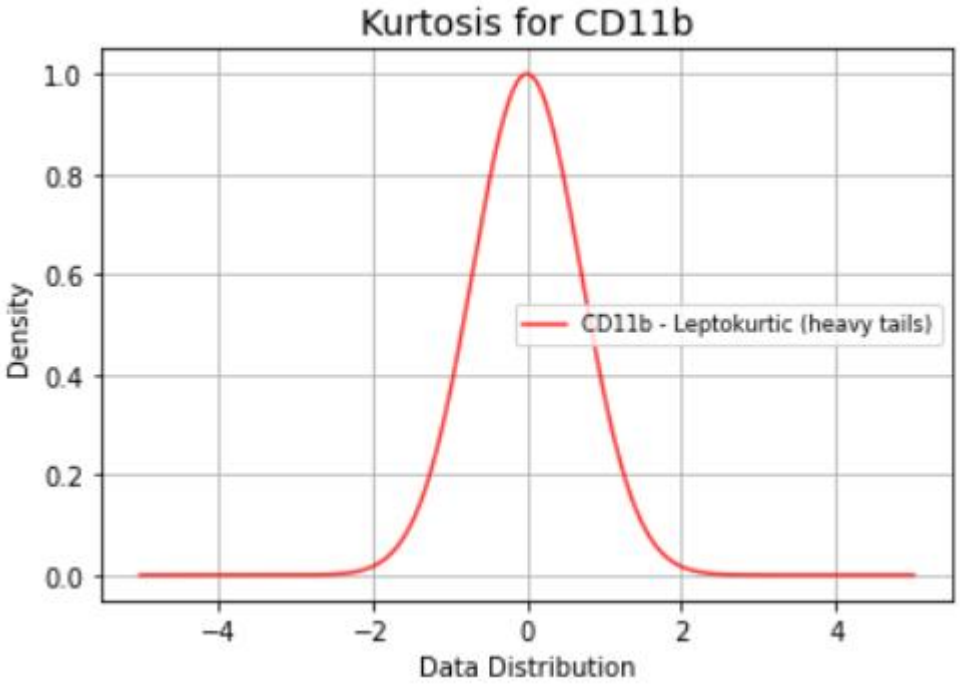

Kurtosis for CD20
CD20 - Leptokurtic (heavy tails)
Density
Data Distribution
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

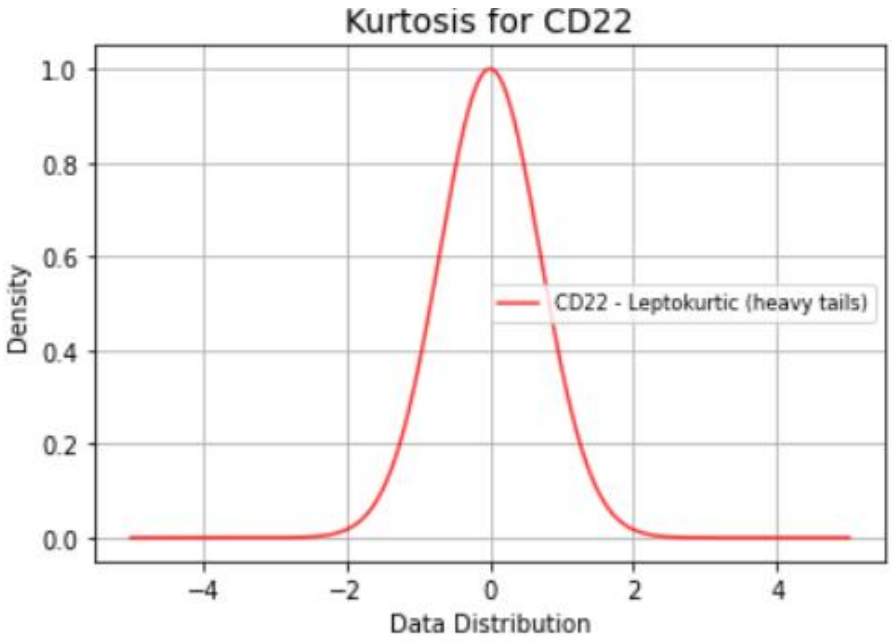

Kurtosis for Flt3
Flt3 - Leptokurtic (heavy tails)
Density
Data Distribution
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

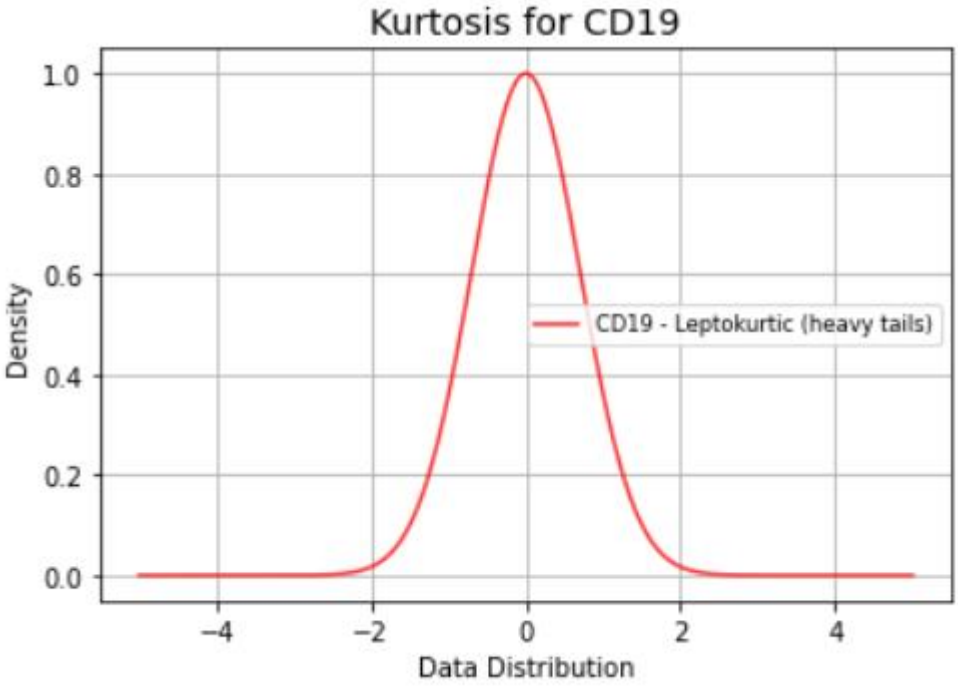

Kurtosis for CD34
CD34 - Leptokurtic (heavy tails)
Density
Data Distribution
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

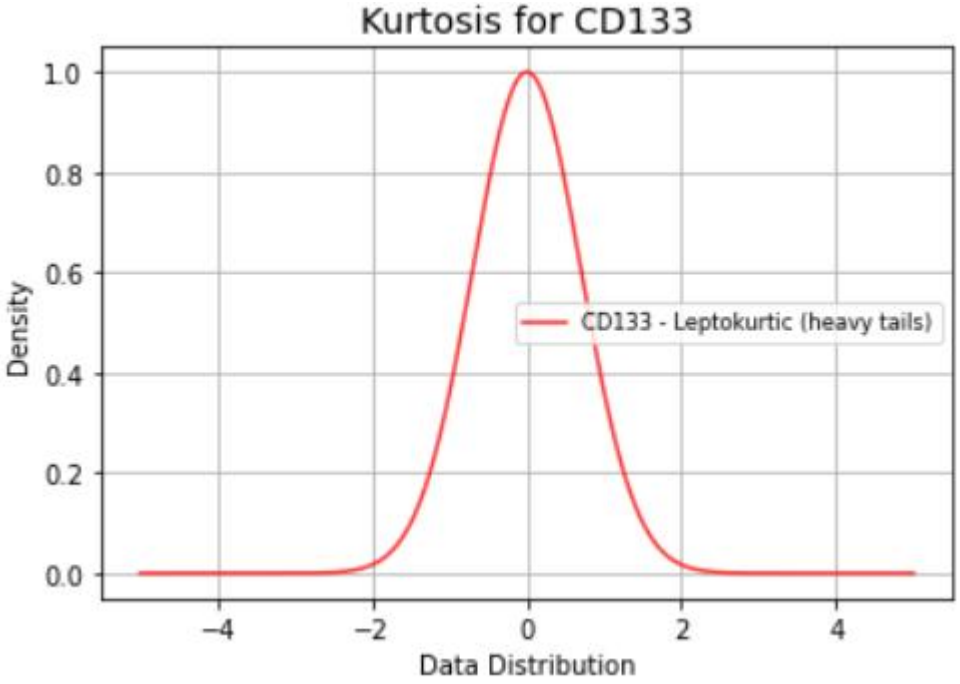

Kurtosis for CD8
CD8 - Leptokurtic (heavy tails)
Density
Data Distribution
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

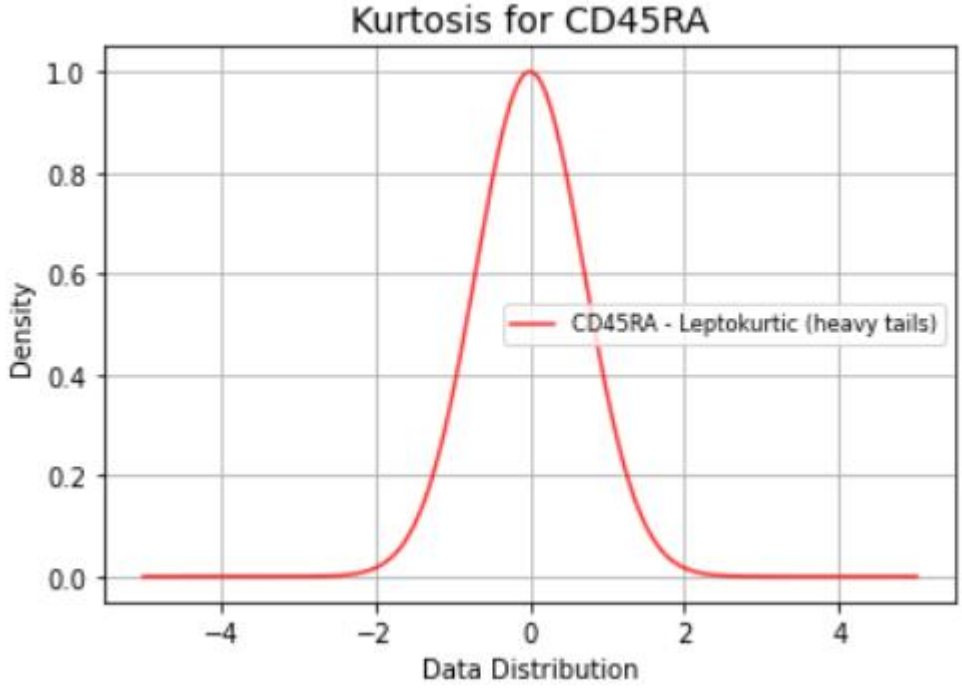

Kurtosis for CD4
CD4 - Leptokurtic (heavy tails)
Density
Data Distribution
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

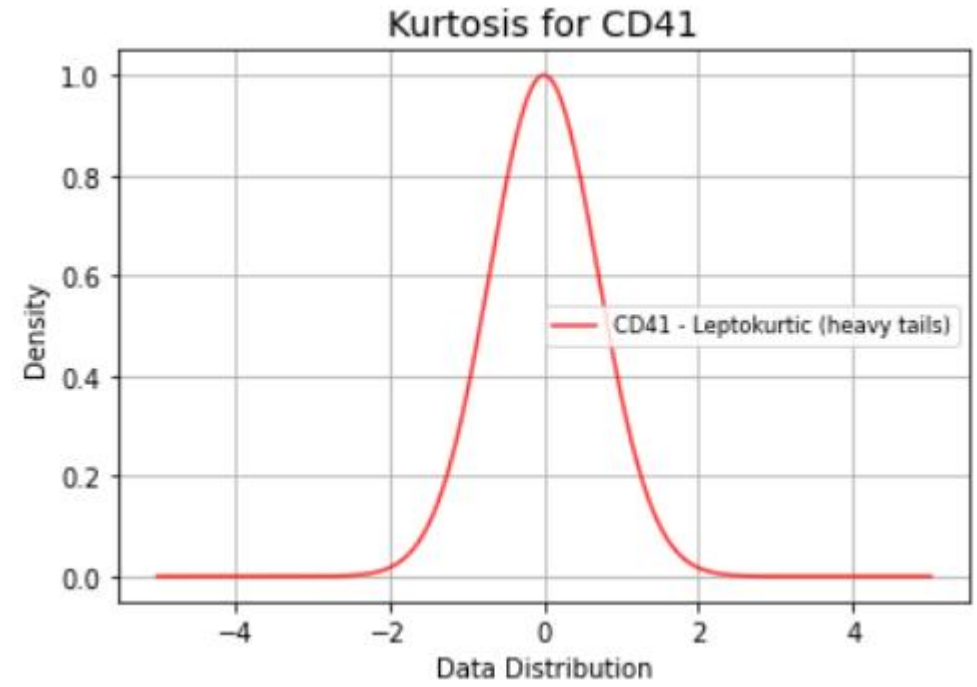

Kurtosis for CD11b
CD11b - Leptokurtic (heavy tails)
Density
Data Distribution
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

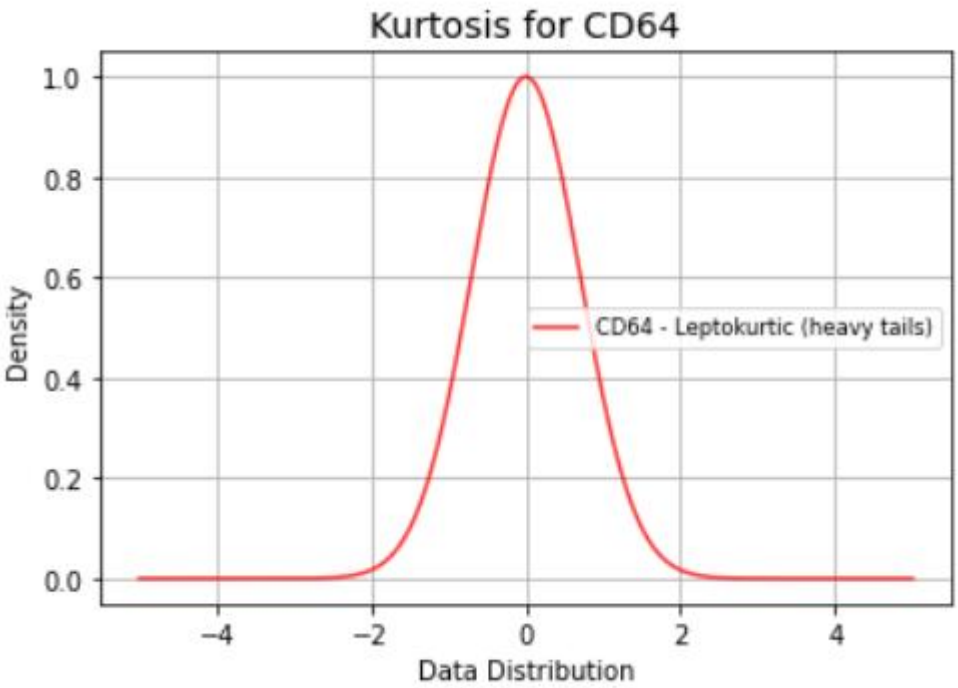

Kurtosis for CD22
Density
Data Distribution
CD22 - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

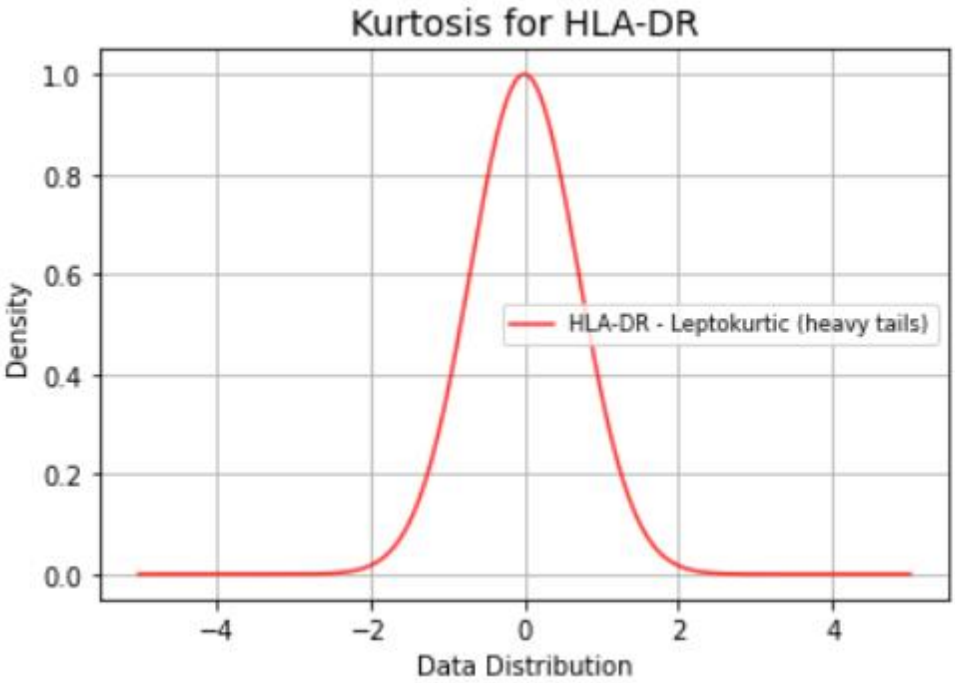

Kurtosis for CD19
Density
Data Distribution
CD19 - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

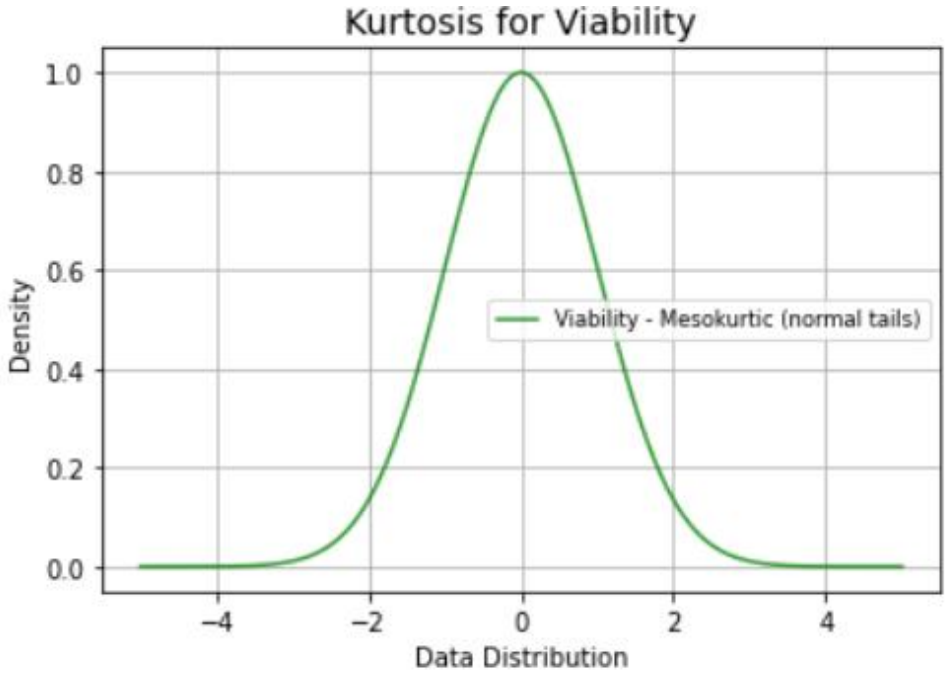

Kurtosis for CD133
Density
Data Distribution
CD133 - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

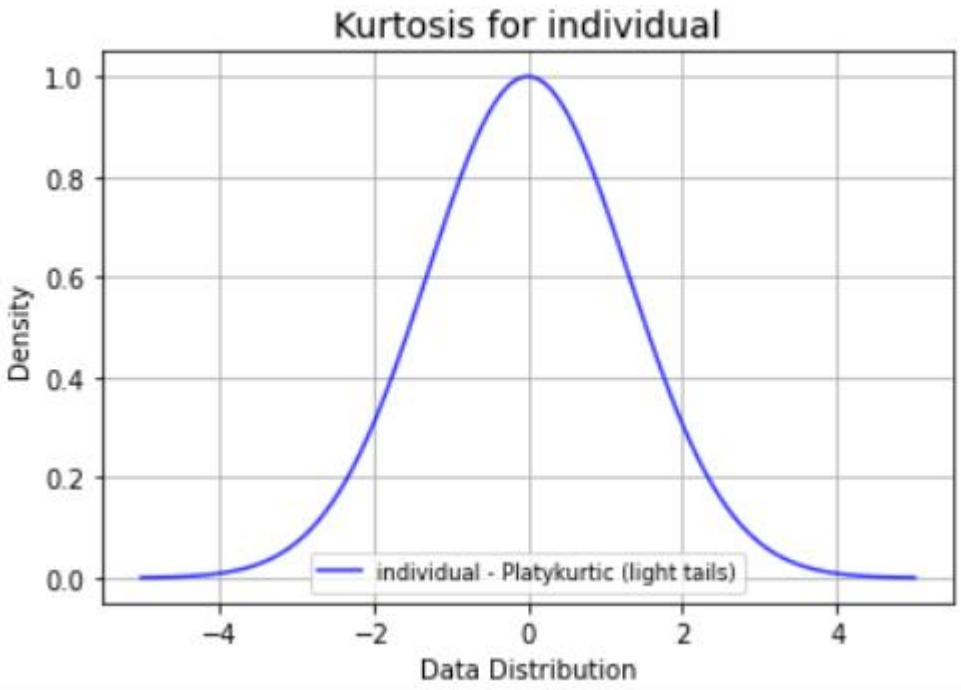

Kurtosis for CD45RA
Density
Data Distribution
CD45RA - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4


Kurtosis for CD41
Density
Data Distribution
CD41 - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4


Kurtosis for CD64
Density
Data Distribution
CD64 - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4


Kurtosis for HLA-DR
Density
Data Distribution
HLA-DR - Leptokurtic (heavy tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

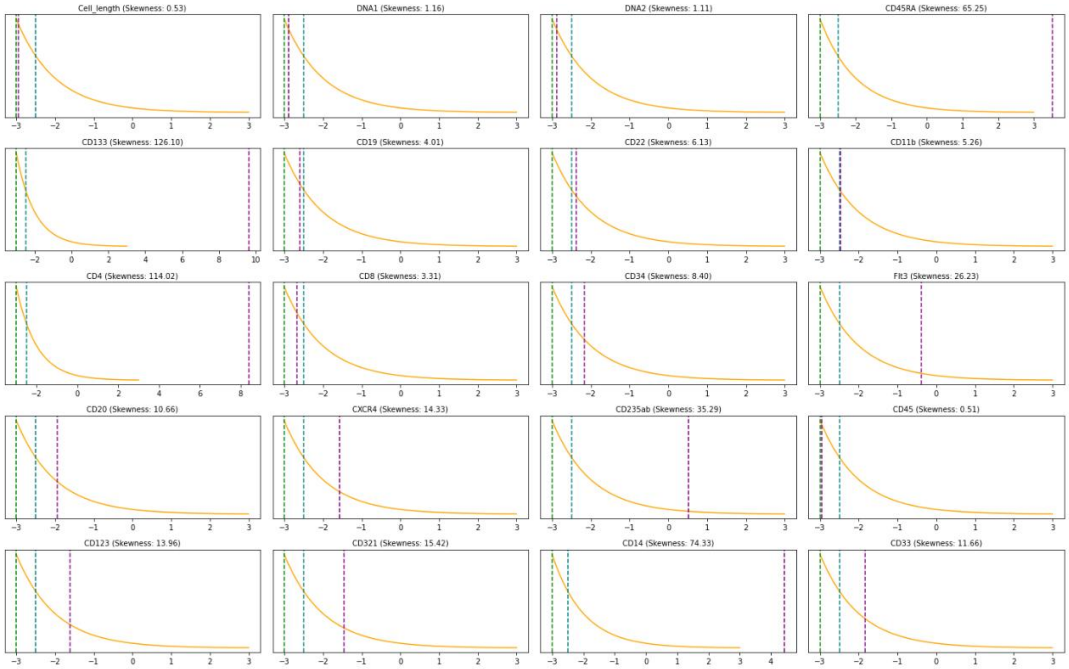

Kurtosis for Viability
Density
Data Distribution
Viability - Mesokurtic (normal tails)
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

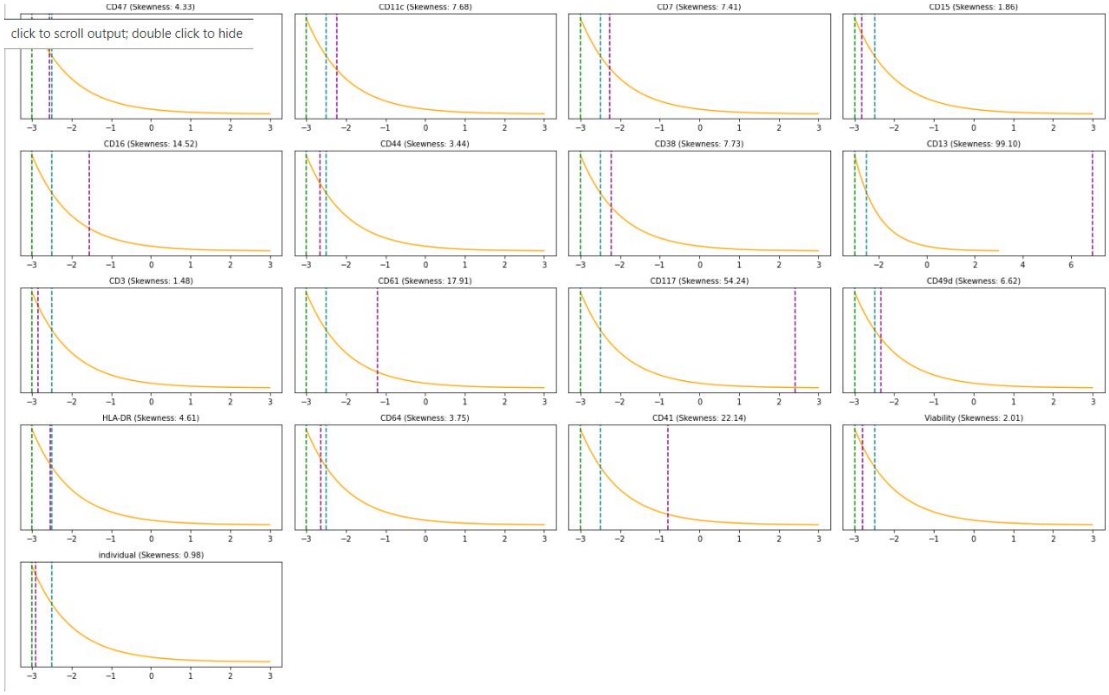


- Output of each feature individually.

```python
# graph(skewness) of each individual feature in the data set
import numpy as np
import matplotlib.pyplot as plt

# Skewness data for each feature
skewness_data = {
    'Cell_length': 0.527832, 'DNA1': 1.155424, 'DNA2': 1.108669, 'CD45RA': 65.251655,
    'CD133': 126.096395, 'CD19': 4.007221, 'CD22': 6.131244, 'CD11b': 5.264678,
    'CD4': 114.022325, 'CD8': 3.313920, 'CD34': 8.397363, 'Flt3': 26.230625,
    'CD20': 10.655454, 'CXCR4': 14.332247, 'CD235ab': 35.288190, 'CD45': 0.514492,
    'CD123': 13.956222, 'CD321': 15.415273, 'CD14': 74.327532, 'CD33': 11.659128,
    'CD47': 4.327074, 'CD11c': 7.679567, 'CD7': 7.405451, 'CD15': 1.860022,
    'CD16': 14.520519, 'CD44': 3.436531, 'CD38': 7.733425, 'CD13': 99.104480,
    'CD3': 1.479010, 'CD61': 17.909078, 'CD117': 54.242959, 'CD49d': 6.622882,
    'HLA-DR': 4.612054, 'CD64': 3.752616, 'CD41': 22.140511, 'Viability': 2.013592,
    'individual': 0.982030
}

# Create subplots
fig, axes = plt.subplots(10, 4, figsize=(20, 25))
axes = axes.flatten()

# Plot each feature with skewness visualization
for i, (feature, skewness_val) in enumerate(skewness_data.items()):
    # Generate x values to represent the distribution curve
    x = np.linspace(-3, 3, 500)
    y = np.exp(-x) * (1 + skewness_val / 10)  # Skewed right

    # Normalize the y values to make the plot consistent
    y /= y.max()

    # Plot skewness curve
    axes[i].plot(x, y, color="orange")

    # Markers for Mode, Median, and Mean positions
    mode = x[np.argmax(y)]
    median = mode + 0.5
    mean = mode + skewness_val / 10  # approximate mean based on skewness

    # Add lines and labels for mode, median, and mean
    axes[i].axvline(mode, color='green', linestyle="--", label='Mode')
    axes[i].axvline(median, color='teal', linestyle="--", label='Median')
    axes[i].axvline(mean, color='purple', linestyle="--", label='Mean')

    # Set title and remove y-axis for simplicity
    axes[i].set_title(f"{feature} (Skewness: {skewness_val:.2f})", fontsize=10)
    axes[i].get_yaxis().set_visible(False)

# Hide any remaining empty subplots
for j in range(i + 1, len(axes)):
    axes[j].axis('off')

# Adjust layout
plt.tight_layout()
plt.show()
```

- Code to create individual plots to each feature representing the type of the skewness for that particular feature with a unique color for each type respectively in the data set.





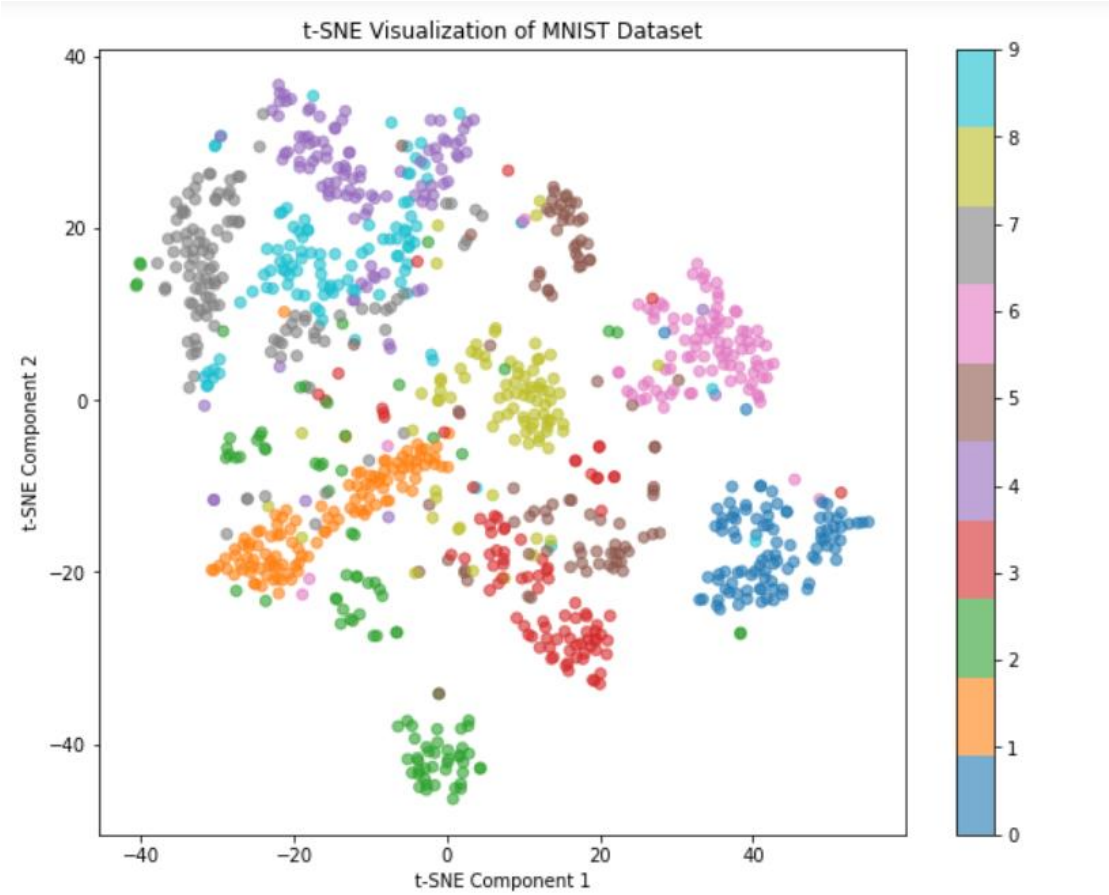
- Output of each feature individually.

- ✓ Read the Research paper about the project (https://www.sciencedirect.com/science/article/pii/S0092867415006376)
- ✓ Learnt a Brief about the research paper from the mentor.

DAY-9 | 17-10-2024 :

- ✓ Could not get expected results by performing t-SNE and pca on the data set.
- ✓ Found that we must Standardize the relevant features range from 0 to 1 and remove columns with null values .
- ✓ So performed the t-SNE on a MNIST data set to understand the working of the model.

```python
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()

# Normalize the images
train_images = train_images.astype('float32') / 255.0
test_images = test_images.astype('float32') / 255.0

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

# Apply t-SNE
tsne = TSNE(n_components=2, random_state=42, perplexity=30)  # Adjust perplexity as needed
train_images_tsne = tsne.fit_transform(train_images_flat)

# Plot the results
plt.figure(figsize=(10, 8))
scatter = plt.scatter(train_images_tsne[:, 0], train_images_tsne[:, 1], c=train_labels_subset, cmap='tab10', alpha=0.6)
plt.colorbar(scatter, ticks=range(10))
plt.title('t-SNE Visualization of MNIST Dataset')
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')
plt.show()
```



- A t-SNE representation of a data set.

DAY-10 | 18-10-2024 :

✓ Gained brief knowledge about what are t-SNE and PCA and How they work.

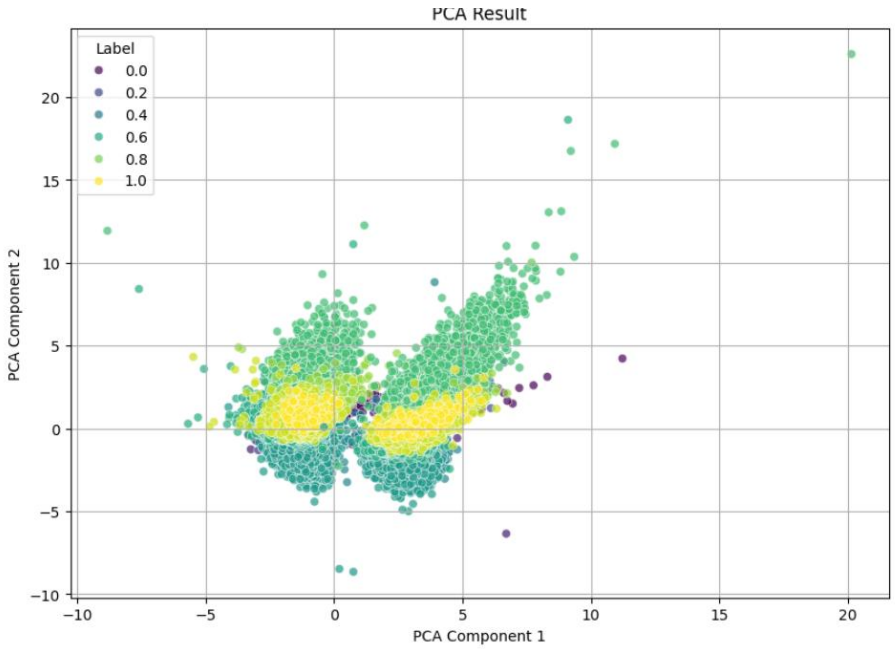
DAY-11 | 21-10-2024 :

✓ Performed t-SNE and PCA Techniques.

PCA

Principal Component Analysis (PCA) is a dimensionality reduction technique used in transforming the data into a set of new, uncorrelated variables called principal components, PCA captures the most important information (or variance) with fewer dimensions.This reduction helps in visualizing high-dimensional data, speeding up algorithms, and minimizing noise.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Preprocessing: Standardizing the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df.drop(['label'], axis=1).dropna())  # Exclude the label column

# Applying PCA
pca = PCA(n_components=2)  # Reduce to 2 components for visualization
pca_result = pca.fit_transform(scaled_data)

# Create a DataFrame with the PCA results
pca_df = pd.DataFrame(data=pca_result, columns=['PCA1', 'PCA2'])
pca_df['label'] = df['label'].dropna().reset_index(drop=True)

# Plot PCA results
plt.figure(figsize=(10, 7))
sns.scatterplot(data=pca_df, x='PCA1', y='PCA2', hue='label', palette='viridis', alpha=0.7)
plt.title('PCA Result')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend(title='Label')
plt.grid()
plt.colorbar(scatter, label='Labels')
plt.show()
```



- The 2-D output of PCA by giving different colors to represent different clusters respectively in a data set.

```
df_cleaned_after = df.drop(columns=[  'Cell_length', 'label', 'individual'])


len(df_cleaned_after.columns)
```

- Removed particular columns that are not needed while performing PCA.

```
# Separate features and labels
features = df_cleaned_after
labels = df['label']
```

- Seperated features and labels.

t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is used for visualizing high-dimensional data in a low-dimensional space, typically 2D or 3D.t-SNE preserves the local structure of data, so similar points in the high-dimensional space stay close in the visualization. It does this by minimizing the differences in probability distributions of point distances between high and low dimensions, resulting in clusters that reflect the original relationships.

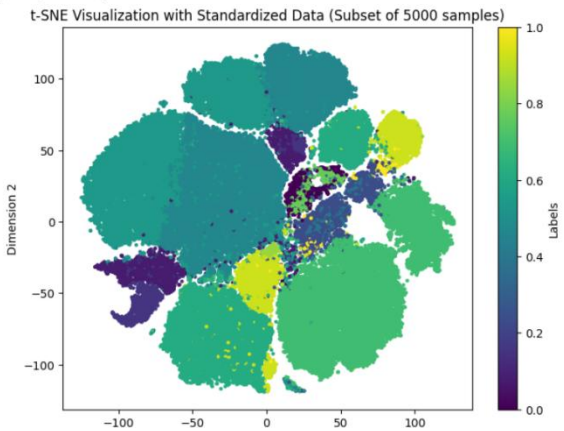
```
import numpy as np
scaler = StandardScaler()
features_standardized = scaler.fit_transform(features)


n_samples = 215000
subset_indices = np.random.choice(len(features_standardized), n_samples, replace=False)


subset_features = features_standardized[subset_indices]
subset_labels = labels[subset_indices]


tsne = TSNE(n_components=2, random_state=42, perplexity=30, n_iter=1000, verbose=1)
tsne_result = tsne.fit_transform(subset_features)


plt.figure(figsize=(8, 6))
scatter = plt.scatter(tsne_result[:, 0], tsne_result[:, 1], c=subset_labels, cmap='viridis', s=5)
plt.title('t-SNE Visualization with Standardized Data (Subset of 5000 samples)')
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')


plt.colorbar(scatter, label='Labels')
plt.show()
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 215000 samples in 0.005s...
[t-SNE] Computed neighbors for 215000 samples in 318.636s...
[t-SNE] Computed conditional probabilities for sample 1000 / 215000
[t-SNE] Computed conditional probabilities for sample 2000 / 215000
[t-SNE] Computed conditional probabilities for sample 3000 / 215000
[t-SNE] Computed conditional probabilities for sample 4000 / 215000
[t-SNE] Computed conditional probabilities for sample 5000 / 215000
[t-SNE] Computed conditional probabilities for sample 6000 / 215000
[t-SNE] Computed conditional probabilities for sample 7000 / 215000
[t-SNE] Computed conditional probabilities for sample 8000 / 215000
[t-SNE] Computed conditional probabilities for sample 9000 / 215000
[t-SNE] Computed conditional probabilities for sample 10000 / 215000
[t-SNE] Computed conditional probabilities for sample 11000 / 215000
[t-SNE] Computed conditional probabilities for sample 12000 / 215000
[t-SNE] Computed conditional probabilities for sample 13000 / 215000
[t-SNE] Computed conditional probabilities for sample 14000 / 215000
[t-SNE] Computed conditional probabilities for sample 15000 / 215000
[t-SNE] Computed conditional probabilities for sample 16000 / 215000
[t-SNE] Computed conditional probabilities for sample 17000 / 215000
[t-SNE] Computed conditional probabilities for sample 18000 / 215000
[t-SNE] Computed conditional probabilities for sample 19000 / 215000
[t-SNE] Computed conditional probabilities for sample 20000 / 215000
[t-SNE] Computed conditional probabilities for sample 21000 / 215000
[t-SNE] Computed conditional probabilities for sample 22000 / 215000
[t-SNE] Computed conditional probabilities for sample 23000 / 215000
[t-SNE] Computed conditional probabilities for sample 24000 / 215000
[t-SNE] Computed conditional probabilities for sample 25000 / 215000
[t-SNE] Computed conditional probabilities for sample 26000 / 215000
[t-SNE] Computed conditional probabilities for sample 27000 / 215000
[t-SNE] Computed conditional probabilities for sample 28000 / 215000
[t-SNE] Computed conditional probabilities for sample 29000 / 215000
```

```
[t-SNE] Computed conditional probabilities for sample 30000 / 215000
[t-SNE] Computed conditional probabilities for sample 31000 / 215000
[t-SNE] Computed conditional probabilities for sample 32000 / 215000
[t-SNE] Computed conditional probabilities for sample 33000 / 215000
[t-SNE] Computed conditional probabilities for sample 34000 / 215000
[t-SNE] Computed conditional probabilities for sample 35000 / 215000
[t-SNE] Computed conditional probabilities for sample 36000 / 215000
[t-SNE] Computed conditional probabilities for sample 37000 / 215000
[t-SNE] Computed conditional probabilities for sample 38000 / 215000
[t-SNE] Computed conditional probabilities for sample 39000 / 215000
[t-SNE] Computed conditional probabilities for sample 40000 / 215000
[t-SNE] Computed conditional probabilities for sample 41000 / 215000
[t-SNE] Computed conditional probabilities for sample 42000 / 215000
[t-SNE] Computed conditional probabilities for sample 43000 / 215000
[t-SNE] Computed conditional probabilities for sample 44000 / 215000
[t-SNE] Computed conditional probabilities for sample 45000 / 215000
[t-SNE] Computed conditional probabilities for sample 46000 / 215000
[t-SNE] Computed conditional probabilities for sample 47000 / 215000
[t-SNE] Computed conditional probabilities for sample 48000 / 215000
[t-SNE] Computed conditional probabilities for sample 49000 / 215000
[t-SNE] Computed conditional probabilities for sample 50000 / 215000
[t-SNE] Computed conditional probabilities for sample 51000 / 215000
[t-SNE] Computed conditional probabilities for sample 52000 / 215000
[t-SNE] Computed conditional probabilities for sample 53000 / 215000
[t-SNE] Computed conditional probabilities for sample 54000 / 215000
[t-SNE] Computed conditional probabilities for sample 55000 / 215000
[t-SNE] Computed conditional probabilities for sample 56000 / 215000
[t-SNE] Computed conditional probabilities for sample 57000 / 215000
[t-SNE] Computed conditional probabilities for sample 58000 / 215000
[t-SNE] Computed conditional probabilities for sample 59000 / 215000
[t-SNE] Computed conditional probabilities for sample 60000 / 215000
[t-SNE] Computed conditional probabilities for sample 61000 / 215000
[t-SNE] Computed conditional probabilities for sample 70000 / 215000
[t-SNE] Computed conditional probabilities for sample 71000 / 215000
[t-SNE] Computed conditional probabilities for sample 72000 / 215000
[t-SNE] Computed conditional probabilities for sample 73000 / 215000
[t-SNE] Computed conditional probabilities for sample 74000 / 215000
[t-SNE] Computed conditional probabilities for sample 75000 / 215000
[t-SNE] Computed conditional probabilities for sample 76000 / 215000
[t-SNE] Computed conditional probabilities for sample 77000 / 215000
[t-SNE] Computed conditional probabilities for sample 78000 / 215000
[t-SNE] Computed conditional probabilities for sample 79000 / 215000
[t-SNE] Computed conditional probabilities for sample 80000 / 215000
[t-SNE] Computed conditional probabilities for sample 81000 / 215000
[t-SNE] Computed conditional probabilities for sample 82000 / 215000
[t-SNE] Computed conditional probabilities for sample 83000 / 215000
[t-SNE] Computed conditional probabilities for sample 84000 / 215000
[t-SNE] Computed conditional probabilities for sample 85000 / 215000
[t-SNE] Computed conditional probabilities for sample 86000 / 215000
[t-SNE] Computed conditional probabilities for sample 87000 / 215000
[t-SNE] Computed conditional probabilities for sample 88000 / 215000
[t-SNE] Computed conditional probabilities for sample 89000 / 215000
[t-SNE] Computed conditional probabilities for sample 90000 / 215000
[t-SNE] Computed conditional probabilities for sample 91000 / 215000
[t-SNE] Computed conditional probabilities for sample 92000 / 215000
[t-SNE] Computed conditional probabilities for sample 93000 / 215000
[t-SNE] Computed conditional probabilities for sample 94000 / 215000
[t-SNE] Computed conditional probabilities for sample 95000 / 215000
[t-SNE] Computed conditional probabilities for sample 96000 / 215000
[t-SNE] Computed conditional probabilities for sample 97000 / 215000
[t-SNE] Computed conditional probabilities for sample 98000 / 215000
[t-SNE] Computed conditional probabilities for sample 99000 / 215000
[t-SNE] Computed conditional probabilities for sample 100000 / 215000
[t-SNE] Computed conditional probabilities for sample 101000 / 215000
[t-SNE] Computed conditional probabilities for sample 102000 / 215000
[t-SNE] Computed conditional probabilities for sample 103000 / 215000
[t-SNE] Computed conditional probabilities for sample 104000 / 215000
[t-SNE] Computed conditional probabilities for sample 105000 / 215000
[t-SNE] Computed conditional probabilities for sample 106000 / 215000
[t-SNE] Computed conditional probabilities for sample 107000 / 215000
[t-SNE] Computed conditional probabilities for sample 108000 / 215000
[t-SNE] Computed conditional probabilities for sample 109000 / 215000
[t-SNE] Computed conditional probabilities for sample 110000 / 215000
[t-SNE] Computed conditional probabilities for sample 111000 / 215000
[t-SNE] Computed conditional probabilities for sample 112000 / 215000
[t-SNE] Computed conditional probabilities for sample 113000 / 215000
[t-SNE] Computed conditional probabilities for sample 114000 / 215000
[t-SNE] Computed conditional probabilities for sample 115000 / 215000
[t-SNE] Computed conditional probabilities for sample 116000 / 215000
[t-SNE] Computed conditional probabilities for sample 117000 / 215000
[t-SNE] Computed conditional probabilities for sample 118000 / 215000
[t-SNE] Computed conditional probabilities for sample 119000 / 215000
[t-SNE] Computed conditional probabilities for sample 120000 / 215000
[t-SNE] Computed conditional probabilities for sample 121000 / 215000
[t-SNE] Computed conditional probabilities for sample 122000 / 215000
[t-SNE] Computed conditional probabilities for sample 123000 / 215000
[t-SNE] Computed conditional probabilities for sample 124000 / 215000
[t-SNE] Computed conditional probabilities for sample 125000 / 215000
[t-SNE] Computed conditional probabilities for sample 126000 / 215000
[t-SNE] Computed conditional probabilities for sample 127000 / 215000
[t-SNE] Computed conditional probabilities for sample 128000 / 215000
[t-SNE] Computed conditional probabilities for sample 129000 / 215000
[t-SNE] Computed conditional probabilities for sample 130000 / 215000
[t-SNE] Computed conditional probabilities for sample 131000 / 215000
[t-SNE] Computed conditional probabilities for sample 132000 / 215000
[t-SNE] Computed conditional probabilities for sample 133000 / 215000
```

```
[t-SNE] Computed conditional probabilities for sample 134000 / 215000
[t-SNE] Computed conditional probabilities for sample 135000 / 215000
[t-SNE] Computed conditional probabilities for sample 136000 / 215000
[t-SNE] Computed conditional probabilities for sample 137000 / 215000
[t-SNE] Computed conditional probabilities for sample 138000 / 215000
[t-SNE] Computed conditional probabilities for sample 139000 / 215000
[t-SNE] Computed conditional probabilities for sample 140000 / 215000
[t-SNE] Computed conditional probabilities for sample 141000 / 215000
[t-SNE] Computed conditional probabilities for sample 142000 / 215000
[t-SNE] Computed conditional probabilities for sample 143000 / 215000
[t-SNE] Computed conditional probabilities for sample 144000 / 215000
[t-SNE] Computed conditional probabilities for sample 145000 / 215000
[t-SNE] Computed conditional probabilities for sample 146000 / 215000
[t-SNE] Computed conditional probabilities for sample 147000 / 215000
[t-SNE] Computed conditional probabilities for sample 148000 / 215000
[t-SNE] Computed conditional probabilities for sample 149000 / 215000
[t-SNE] Computed conditional probabilities for sample 150000 / 215000
[t-SNE] Computed conditional probabilities for sample 151000 / 215000
[t-SNE] Computed conditional probabilities for sample 152000 / 215000
[t-SNE] Computed conditional probabilities for sample 153000 / 215000
[t-SNE] Computed conditional probabilities for sample 154000 / 215000
[t-SNE] Computed conditional probabilities for sample 155000 / 215000
[t-SNE] Computed conditional probabilities for sample 156000 / 215000
[t-SNE] Computed conditional probabilities for sample 157000 / 215000
[t-SNE] Computed conditional probabilities for sample 158000 / 215000
[t-SNE] Computed conditional probabilities for sample 159000 / 215000
[t-SNE] Computed conditional probabilities for sample 160000 / 215000
[t-SNE] Computed conditional probabilities for sample 161000 / 215000
[t-SNE] Computed conditional probabilities for sample 162000 / 215000
[t-SNE] Computed conditional probabilities for sample 163000 / 215000
[t-SNE] Computed conditional probabilities for sample 164000 / 215000
[t-SNE] Computed conditional probabilities for sample 165000 / 215000
[t-SNE] Computed conditional probabilities for sample 184000 / 215000
[t-SNE] Computed conditional probabilities for sample 185000 / 215000
[t-SNE] Computed conditional probabilities for sample 186000 / 215000
[t-SNE] Computed conditional probabilities for sample 187000 / 215000
[t-SNE] Computed conditional probabilities for sample 188000 / 215000
[t-SNE] Computed conditional probabilities for sample 189000 / 215000
[t-SNE] Computed conditional probabilities for sample 190000 / 215000
[t-SNE] Computed conditional probabilities for sample 191000 / 215000
[t-SNE] Computed conditional probabilities for sample 192000 / 215000
[t-SNE] Computed conditional probabilities for sample 193000 / 215000
[t-SNE] Computed conditional probabilities for sample 194000 / 215000
[t-SNE] Computed conditional probabilities for sample 195000 / 215000
[t-SNE] Computed conditional probabilities for sample 196000 / 215000
[t-SNE] Computed conditional probabilities for sample 197000 / 215000
[t-SNE] Computed conditional probabilities for sample 198000 / 215000
[t-SNE] Computed conditional probabilities for sample 199000 / 215000
[t-SNE] Computed conditional probabilities for sample 200000 / 215000
[t-SNE] Computed conditional probabilities for sample 201000 / 215000
[t-SNE] Computed conditional probabilities for sample 202000 / 215000
[t-SNE] Computed conditional probabilities for sample 203000 / 215000
[t-SNE] Computed conditional probabilities for sample 204000 / 215000
[t-SNE] Computed conditional probabilities for sample 205000 / 215000
[t-SNE] Computed conditional probabilities for sample 206000 / 215000
[t-SNE] Computed conditional probabilities for sample 207000 / 215000
[t-SNE] Computed conditional probabilities for sample 208000 / 215000
[t-SNE] Computed conditional probabilities for sample 209000 / 215000
[t-SNE] Computed conditional probabilities for sample 210000 / 215000
[t-SNE] Computed conditional probabilities for sample 211000 / 215000
[t-SNE] Computed conditional probabilities for sample 212000 / 215000
[t-SNE] Computed conditional probabilities for sample 213000 / 215000
[t-SNE] Computed conditional probabilities for sample 214000 / 215000
[t-SNE] Computed conditional probabilities for sample 215000 / 215000
```

```
[t-SNE] Mean sigma: 0.693408
[t-SNE] KL divergence after 250 iterations with early exaggeration: 96.404488
[t-SNE] KL divergence after 1000 iterations: 2.989786
```



- The 2-D output of a subset of 5000 samples by giving different colors to represent different clusters respectively in a data set.

DAY-12 | 22-10-2024 :

✓ Performed some operations of PCA like Standard Deviation , Proportion of Variance, Cumulative Proportion.

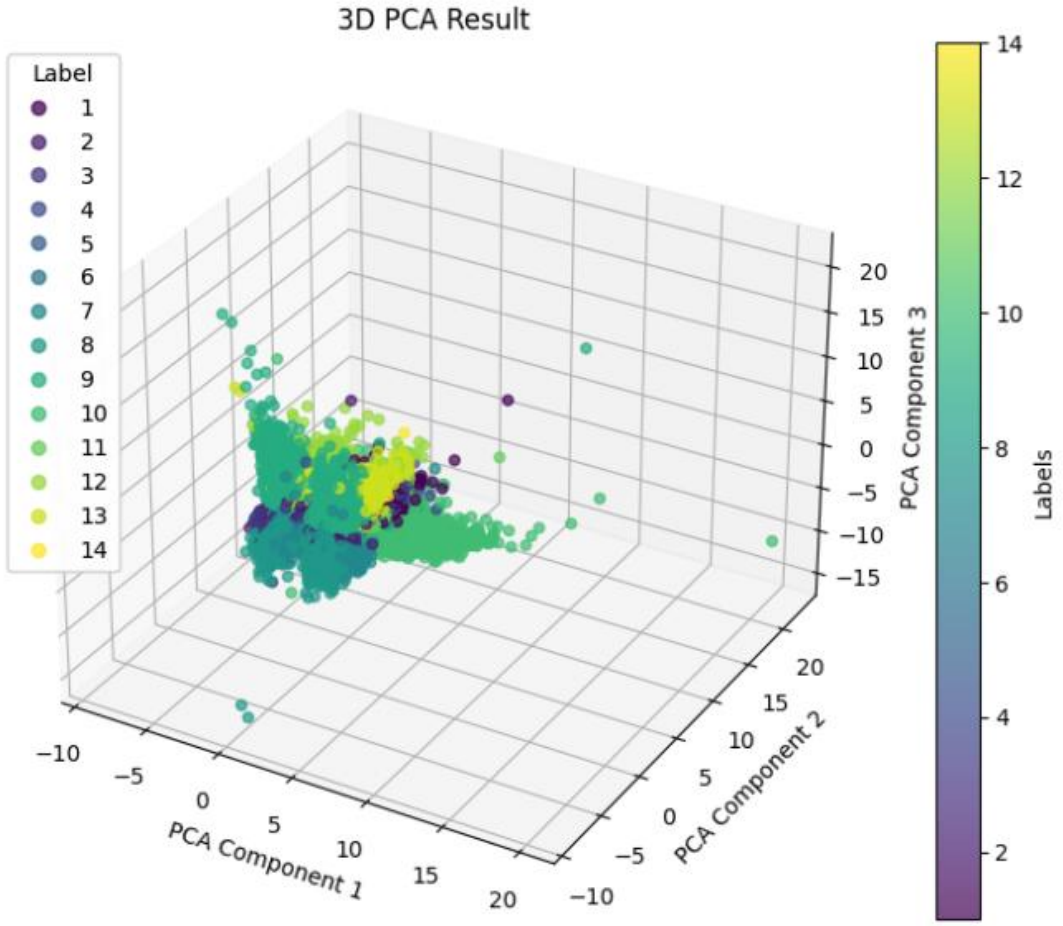
```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Convert data into a DataFrame
df = pd.DataFrame(data)

# Remove rows with NaN values
df_cleaned = df.dropna()

# Standardize the cleaned data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df_cleaned)

# Perform PCA
pca = PCA(n_components=4)
pca.fit(scaled_data)

# Extracting results
explained_variance = pca.explained_variance_ratio_
cumulative_variance = np.cumsum(explained_variance)
std_deviation = np.sqrt(pca.explained_variance_)

# Creating the summary table
summary_table = pd.DataFrame({
    'Standard deviation': std_deviation,
    'Proportion of Variance': explained_variance,
    'Cumulative Proportion': cumulative_variance
}, index=[f'PC{i+1}' for i in range(4)])

print(summary_table.T)
```

	PC1	PC2	PC3	PC4
Standard deviation	2.091349	1.981198	1.728052	1.494168
Proportion of Variance	0.115097	0.103292	0.078583	0.058750
Cumulative Proportion	0.115097	0.218390	0.296972	0.355723

- The values of Standard Deviation , Proportion of Variance, Cumulative Proportion for 4 components of PCA respectively.

✓ Performed 3-D plot of PCA.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from mpl_toolkits.mplot3d import Axes3D  # Import for 3D plotting
# Preprocessing: Standardizing the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df.drop(['label'], axis=1).dropna())  # Exclude the label column

# Applying PCA for 3 components
pca = PCA(n_components=3)  # Reduce to 3 components for 3D visualization
pca_result = pca.fit_transform(scaled_data)

# Create a DataFrame with the PCA results
pca_df = pd.DataFrame(data=pca_result, columns=['PCA1', 'PCA2', 'PCA3'])
pca_df['label'] = df['label'].dropna().reset_index(drop=True)

# 3D Plot PCA results
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')

# Scatter plot with color mapping based on the label
scatter = ax.scatter(pca_df['PCA1'], pca_df['PCA2'], pca_df['PCA3'], c=pca_df['label'], cmap='viridis', alpha=0.7)

# Add axis labels and a title
ax.set_xlabel('PCA Component 1')
ax.set_ylabel('PCA Component 2')
ax.set_zlabel('PCA Component 3')
ax.set_title('3D PCA Result')

# Adding a color bar and a legend
legend1 = ax.legend(*scatter.legend_elements(), title="Label")
ax.add_artist(legend1)
fig.colorbar(scatter, ax=ax, label='Labels')
plt.show()
```



- The 3-D output of PCA by giving different colors to represent different clusters respectively in a data set.

✓ Learnt a brief on Auto Encoders like on what domain encoders are used and some model names that are implemented using auto encoders.

DAY-13 | 23-10-2024 :

✓ Studied the Research paper (https://arxiv.org/pdf/2006.05278) about Deep Semi-Supervised Learning.

DAY-14 | 25-10-2024 :

✓ Gained a brief knowledge about semi-super vised learning from the research paper .