Dimensionality Reduction and Association Analysis

Part 1: Dimensionality Reduction

Introduction:

Dimensionality reduction is the process of decreasing the dimensions (the number of attributes) in a given data. There could be several compelling reasons to do so, including the convenience of visualizing, reduction in noise, representing the data in a more comprehensive way by removing redundant attributes etc. One technique of dimensionality reduction is known as Principal Component Analysis (PCA).

Principal Component Analysis (PCA): In PCA, we discover a new set of dimensions against which to represent, describe or evaluate the data where each new dimension is known as a Principal Component and each principal component is mutually orthogonal to all other Principal components.

Implementation:

From a bird's eye view, the process of finding the principal components is as follows:

- Let X* the mean vector found by taking the mean of all rows.
- The original data is adjusted by subtracting each value of each row by X
    - $x` = x - X^*$;
- Next, the covariance matrix, S, is calculated from the above adjusted data(X).

$$S = (1/n-1)XTX$$

- Now, the eigenvectors and eigenvalues of S are found such that $Sa = \lambda a$
- The eigenvalues λj corresponds to variance on each component j, so the λ values are sorted.
- Finally, we take the first 'd' eigenvectors ai and these represent the directions with largest variance.

The new dimensions (principal components) are produced as:

y1 = a11x1 + a12x2 + ... + a1kxk
y2 = a21x1 + a22x2 + ... + a2kxk
...
yk = ak1x1 + ak2x2 + ... + akkxk

Code Snippets:

The following piece of code is used to calculate the mean of all rows

```
for col in range(matrix.shape[1]):
        sum = 0
       for row in range(matrix.shape[0]):
           sum+=matrix[row,col]
       meanList.append(sum/n)
```

After the mean (X*) is found, the following method is used to adjust the original data:

```
def adjustment(matrix, meanList):
    tmp = 0
    for col in range(matrix.shape[1]):
        for row in range(matrix.shape[0]):
            matrix[row,col] = matrix[row,col] - meanList[tmp]
        tmp+=1
    return matrix
```

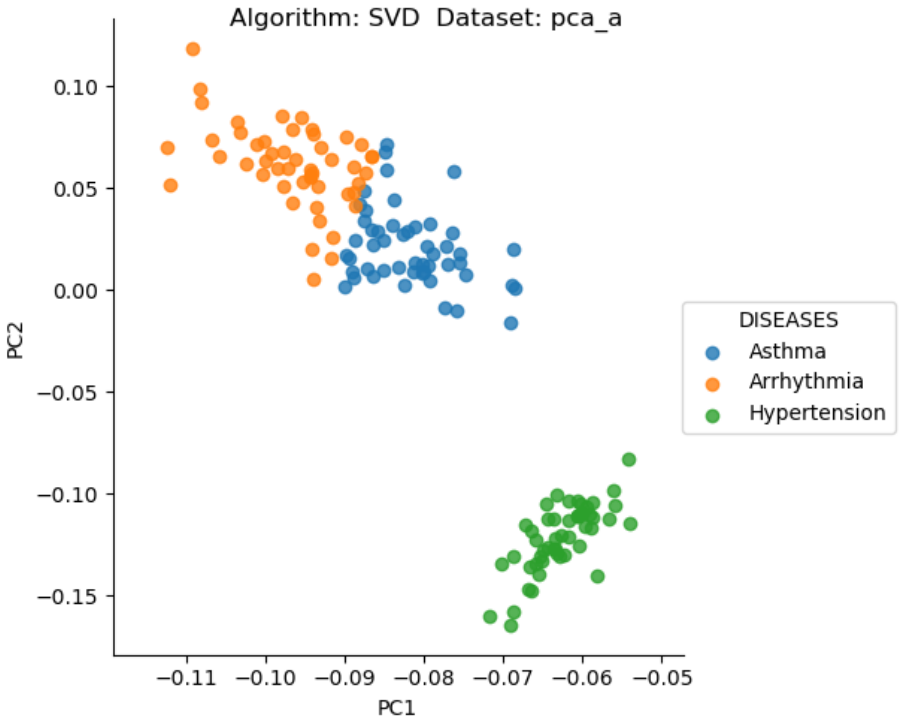
To calculate the eigen vectors, the following code is used (using the numpy library methods for matrix multiplication and finding the eigen vectors)

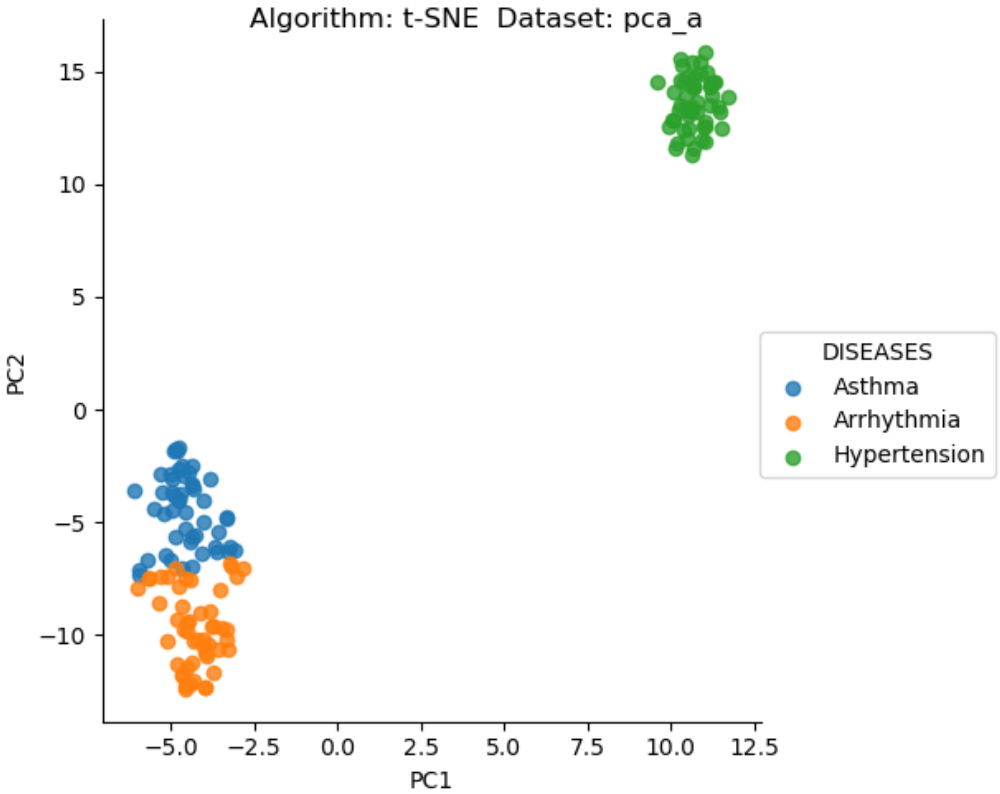
```
S = (1/(n-1))*np.matmul(matrix.T,matrix)
    eigenValues, eigenVectors = np.linalg.eig(S)
```

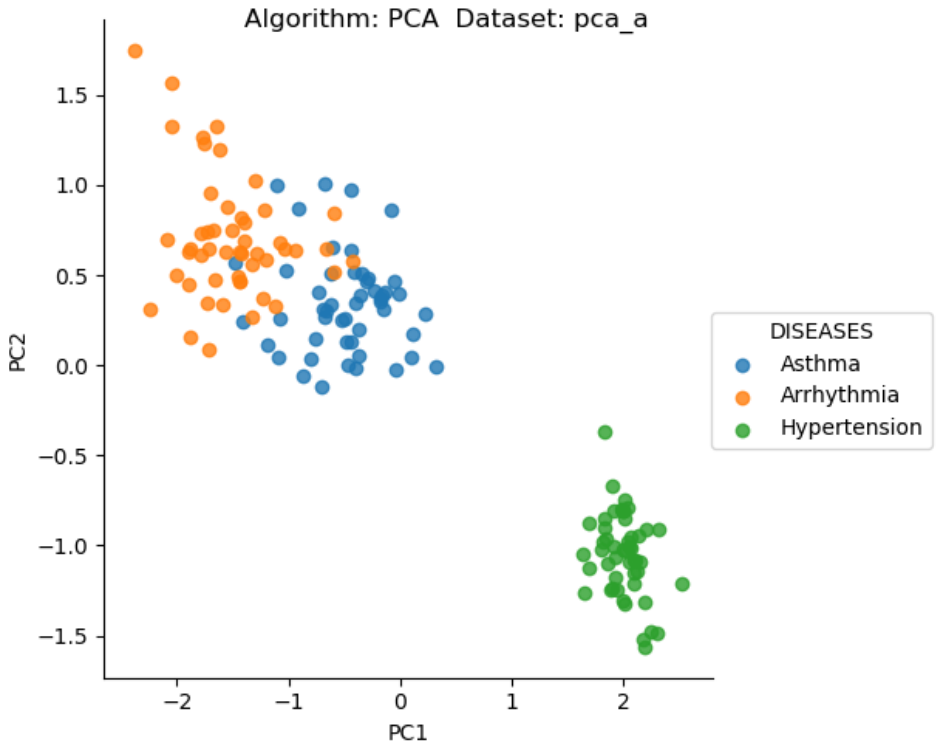
Finally, the top n principal components are found using the following method:

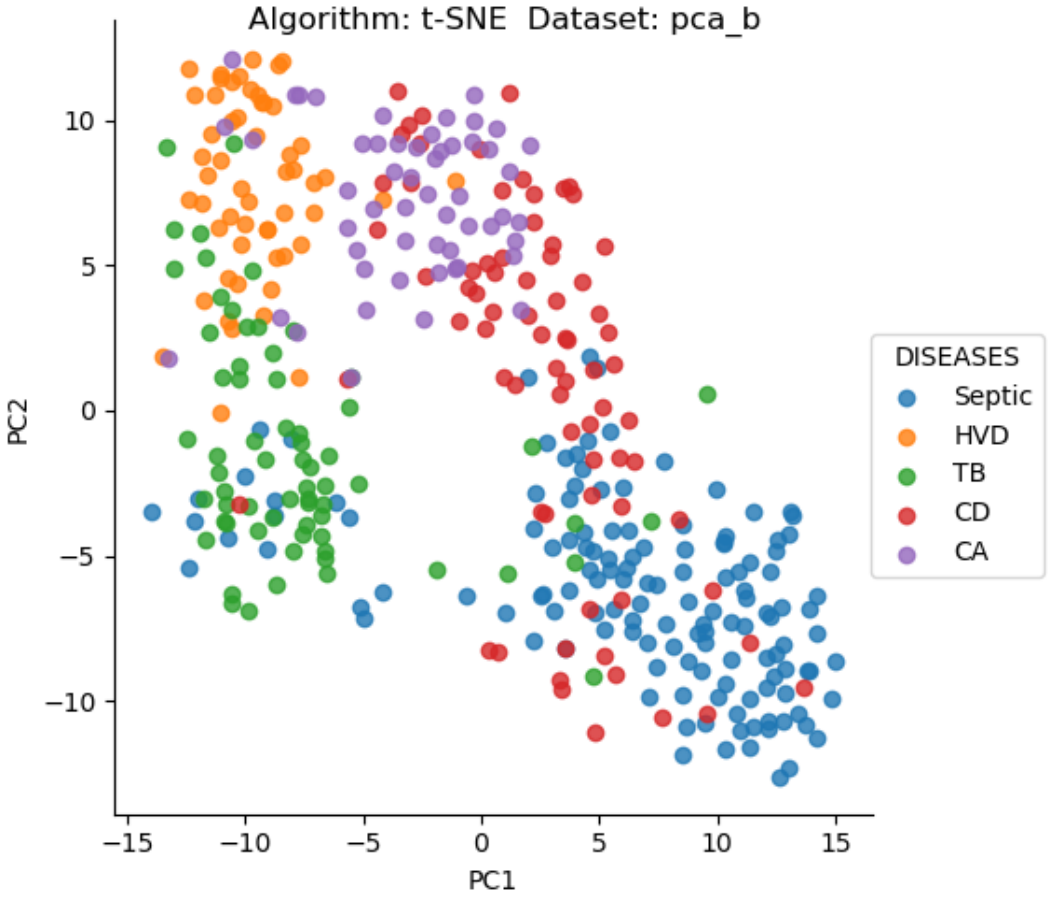
```
def topN(eigenValues, eigenVectors,n = 2):
    indices = eigenValues.argsort()[::-1][:n]
    return eigenVectors[indices[0]],eigenVectors[indices[1]]
```

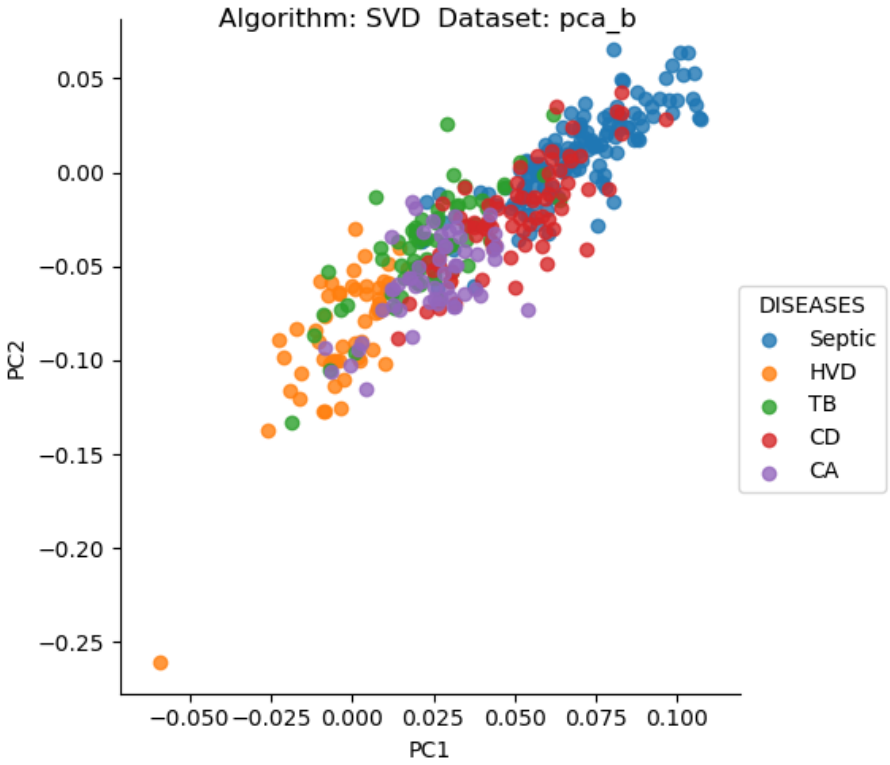
Results:

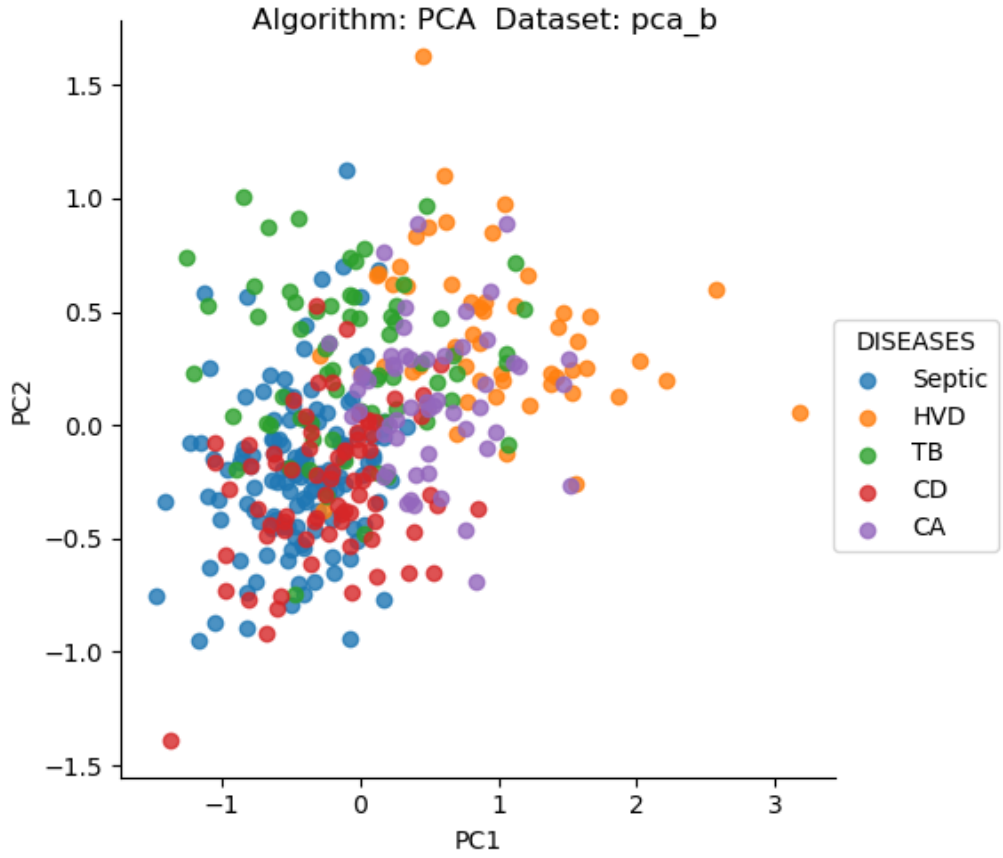

Algorithm: SVD Dataset: pca_a
0.10
0.05
0.00
−0.05
−0.10
−0.15
PC2
−0.11
−0.10
−0.09
−0.08
−0.07
−0.06
−0.05
PC1
DISEASES
Asthma
Arrhythmia
Hypertension

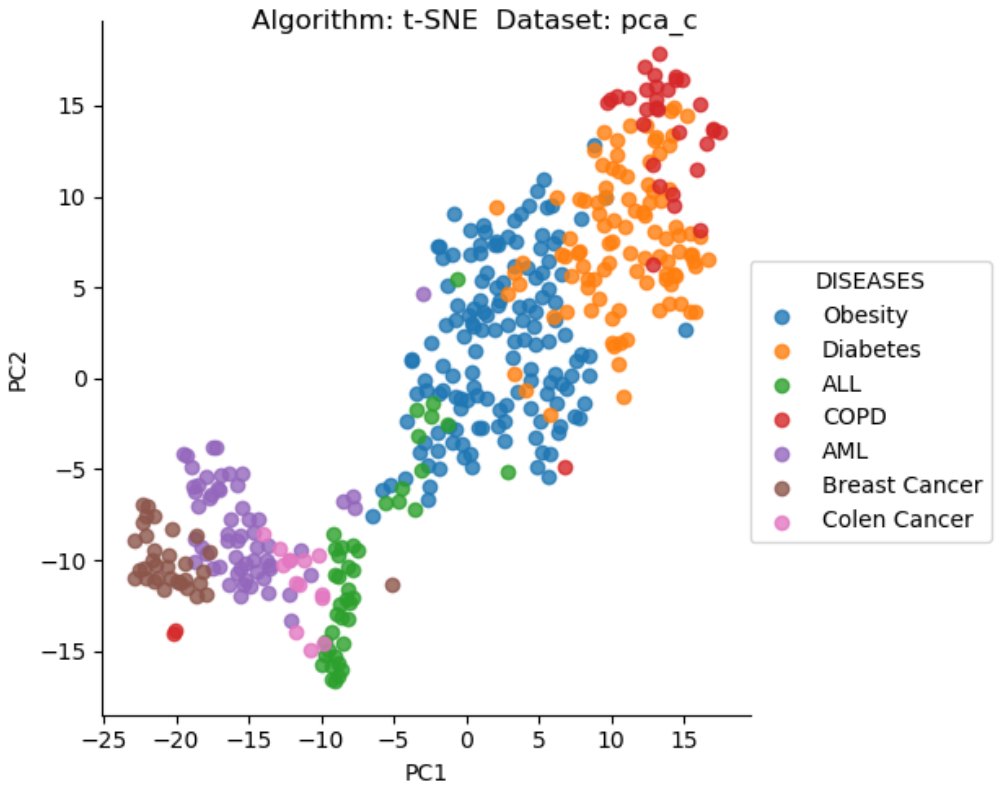

Algorithm: t-SNE Dataset: pca_a
15
10
5
0
−5
−10
PC2
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
PC1
DISEASES
Asthma
Arrhythmia
Hypertension

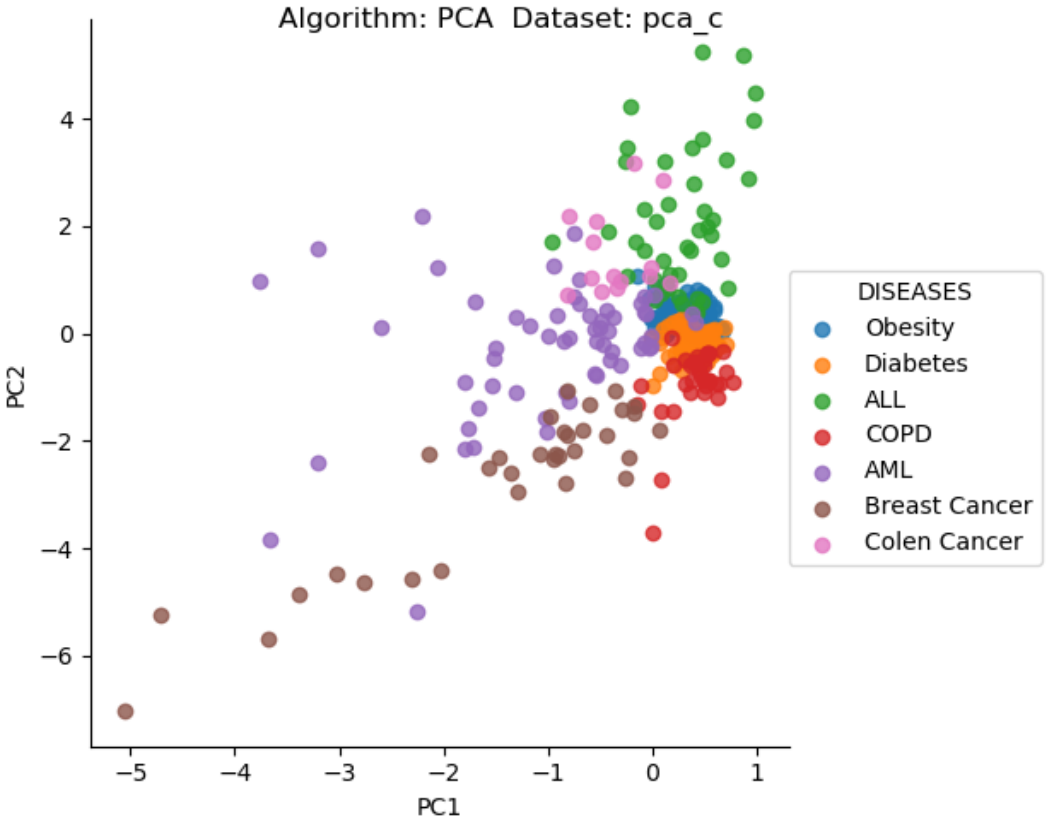

Algorithm: PCA Dataset: pca_a
PC2
PC1
DISEASES
Asthma
Arrhythmia
Hypertension

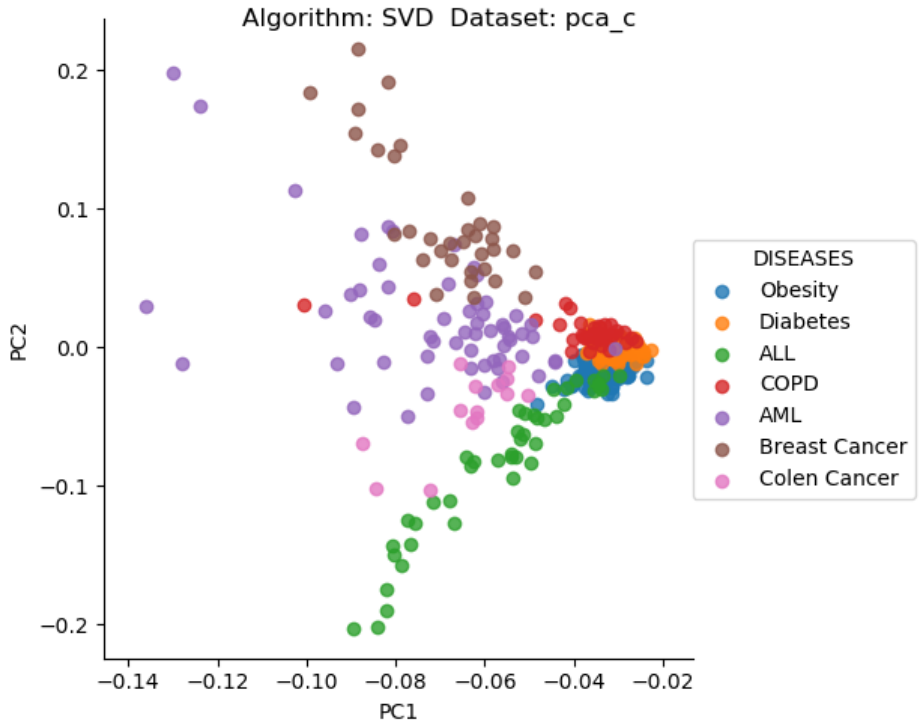

Algorithm: t-SNE Dataset: pca_b
PC2
PC1
DISEASES
Septic
HVD
TB
CD
CA


Algorithm: SVD Dataset: pca_b
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
PC2
−0.050 −0.025 0.000 0.025 0.050 0.075 0.100
PC1
DISEASES
Septic
HVD
TB
CD
CA


Algorithm: PCA Dataset: pca_b
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
PC2
−1
0
1
2
3
PC1
DISEASES
Septic
HVD
TB
CD
CA


Algorithm: t-SNE Dataset: pca_c
PC2
PC1
DISEASES
Obesity
Diabetes
ALL
COPD
AML
Breast Cancer
Colen Cancer


Algorithm: PCA Dataset: pca_c
PC2
PC1
DISEASES
Obesity
Diabetes
ALL
COPD
AML
Breast Cancer
Colen Cancer



Inferences:

As it can be evidently seen, the scatter plots for PCA and SVD are much similar as both the algorithms follow a somewhat similar process for determining the principal components. PCA is just a special case of SVD where it needs the data to be normalized and hence, this justifies why the graphs for PCA and SVD are similar.

t-SNE (short for t-Distributed Stochastic Neighbor Embedding) maps the multi-dimensional data to a lower dimensional space by minimizing the divergence between two distributions: a distribution that measures pairwise similarities of the input objects and a distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding and attempts to find patterns in the data by identifying observed clusters based on similarity of data points with multiple features. When t-SNE is compared to PCA, PCA is just a linear technique for reducing the dimensions while keeping the information intact. Whereas in t-SNE, after the process, the input features are no longer identifiable, and no inference can be made based only on the output of t-SNE.