CSE 601: Data Mining and Bioinformatics
Project-1
Part-1
Dimensionality Reduction

Professor
Jing Gao

Submitted By
Akshara Santharam (50313950)
Swathi Ravindran (50314300)
Sai Kumar Aindla (50321372)

Dimensionality Reduction

Dimensionality Reduction is the process of discovering new set of dimensions against which to represent, describe and evaluate the older set by retaining the meaningful properties. This is very useful when dealing with high set of dimensions. Reduction is often performed by projecting data to a lower set of dimensions, which when transformed will still preserve the essence of the data.

Dimensionality reduction can be performed using Principal Component Analysis (PCA) by transferring a set of correlated dimensions into a new set of uncorrelated dimensions. These new set of dimensions are called as "Principal Components". Similarly, this can be accomplished using Singular Value Decomposition (SVD) and t-Distributed Stochastic Neighbor embedding (t-SNE).

Implementation

Principal Component Analysis (PCA):

1. Pass the file name as command line arguments when executing the python file.
2. Using pandas, read the data from the file by using separator "\t" and store it in a pandas data frame.

```
file = sys.argv[1]
df = pd.read_csv(file,sep="\t", header=None)
```

3. Store the diseases(labels) and features in different numpy array and data frame.

```
diseases = (df[len(df.columns) - 1]).to_numpy()
features = df.iloc[:,0:len(df.columns)-1]
```

4. Perform the PCA implementation on the features data frame.
5. Calculate the mean vector ($\bar{X}$) of all the features and adjust the original data by the mean using the formula.

$$X' = X - \bar{X}$$

6. Evaluate the Co-Variance matrix (S) of adjusted features vector (X') by using the below formula on features.

$$S = \frac{1}{n-1} X'^{T} X'$$

```
mean_vector = pd.DataFrame(features.mean(axis = 0))
mean_vector = pd.DataFrame(mean_vector.to_numpy().T)
mean_vector = mean_vector.append([mean_vector.iloc[0]]*(len(features)-1))
mean_vector = mean_vector.reset_index(drop=True)
Adjust_data = features.to_numpy() - mean_vector.to_numpy()
CoVar_Mat = 1/(len(features)-1)
CoVar_Mat = CoVar_Mat * np.matmul(Adjust_data.T,Adjust_data)
```

7. Compute the eigen values and eigen vectors of the Co-Variance matrix using numpy's np.linalg.eig(S).

```
eigen_values, eigen_vectors = np.linalg.eig(CoVar_Mat)
```

8. Extract the eigen indexes of the top two eigen values and accordingly their respective eigen vectors having these highest eigen values.
9. Later, multiply the eigen vectors with adjusted data to get the new dimensions.

```
top_eigen_indexes = sorted(range(len(eigen_values)), key = lambda i : eigen_values[i], reverse=True)[:2]
top_eigen_vectors = eigen_vectors[:,top_eigen_indexes]
new_data = np.matmul(Adjust_data,top_eigen_vectors)
```

10. Scatter plots are created using seaborn package by using the new dimensions and the diseases data frame that is created in the first step.

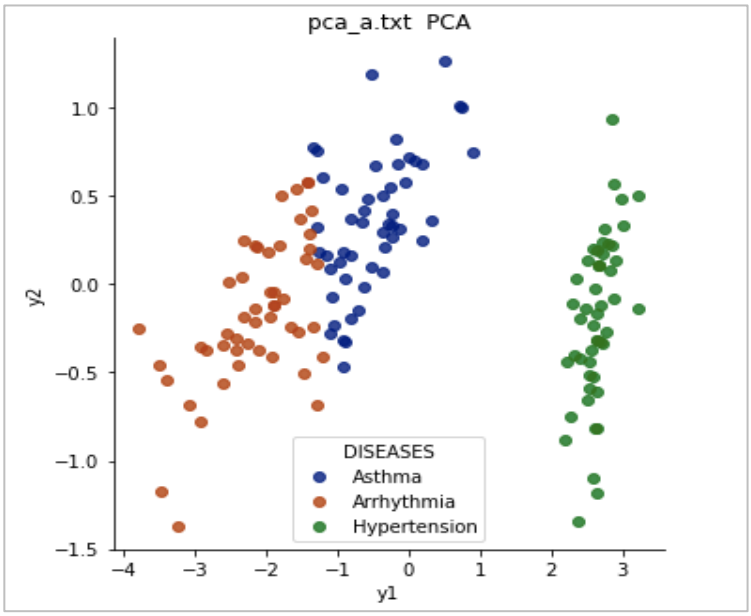
SVD and t-SNE:

1. Pass the features numpy array that is created in the first step of PCA to the SVD function and t-SNE function.
2. Apply the existing SVD package and multiply the left singular matrix (u) with the diagonal matrix(s) to get the new dimensions.
3. Similarly apply the existing t-SNE package to get the new dimensions by mentioning the number of required components.
4. Follow, the same process of PCA to create the scatter plots using seaborn package using the new dimensions and the diseases (labels(arrays.

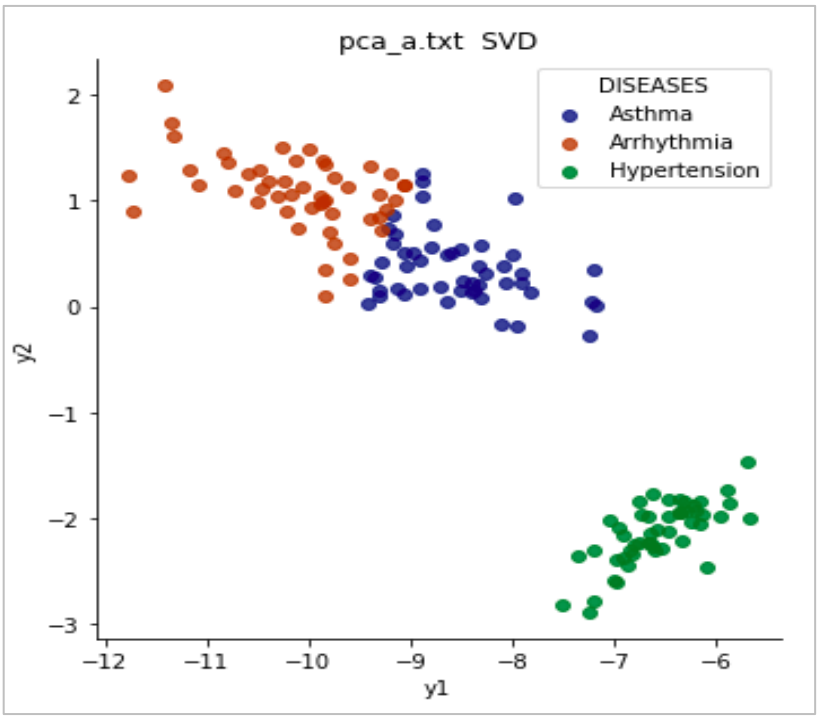
Scatter Plots

PCA_A Results

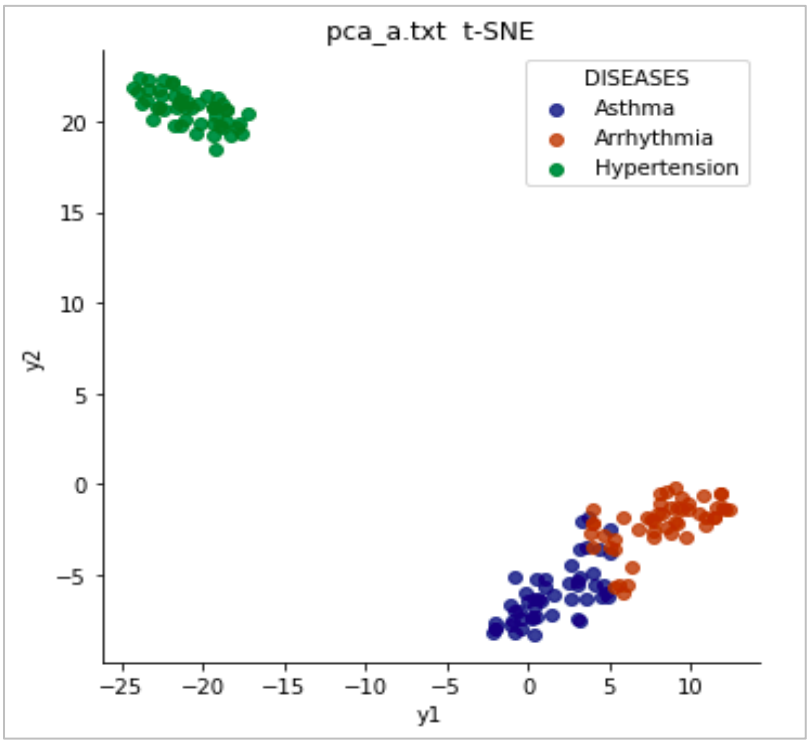
1. PCA plot



2. SVD plot

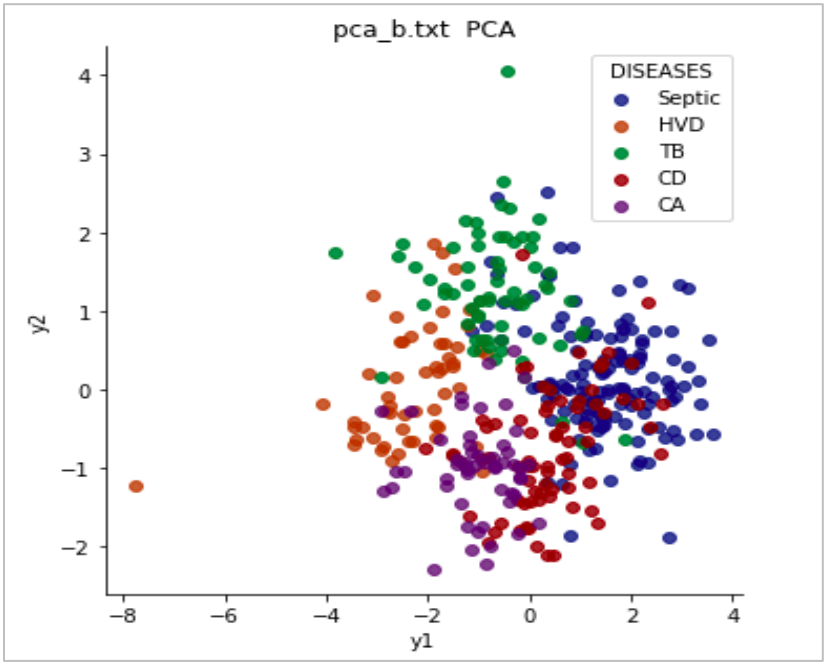


3. t-SNE plot

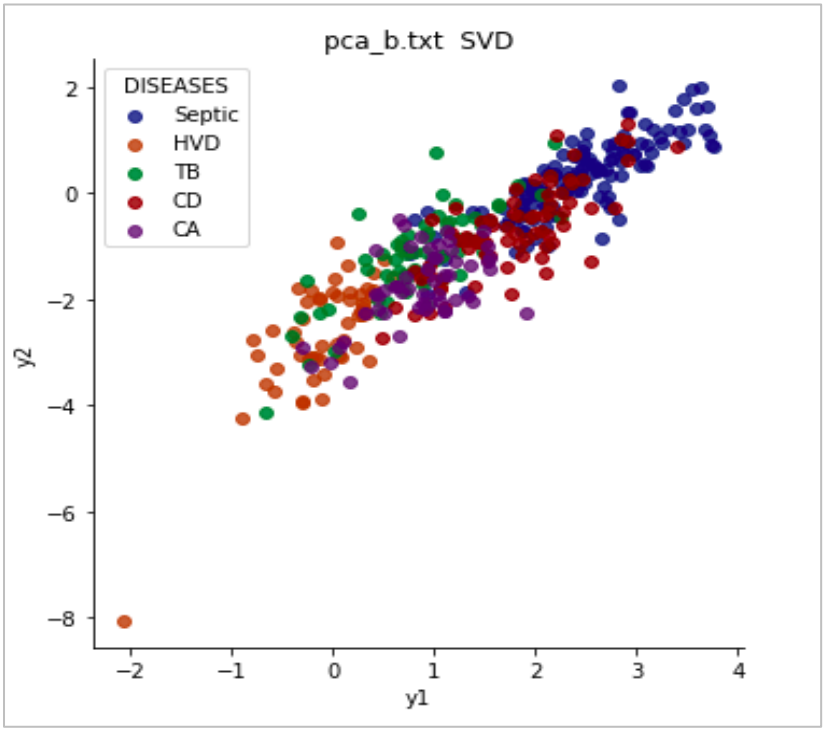


PCA_B Results

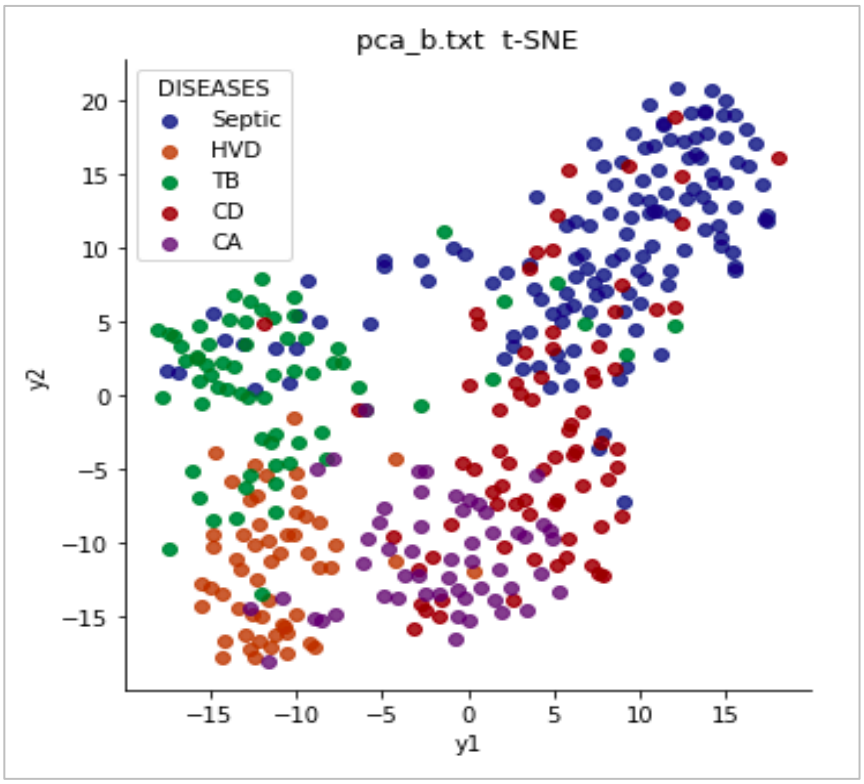
1. PCA plot



2. SVD plot

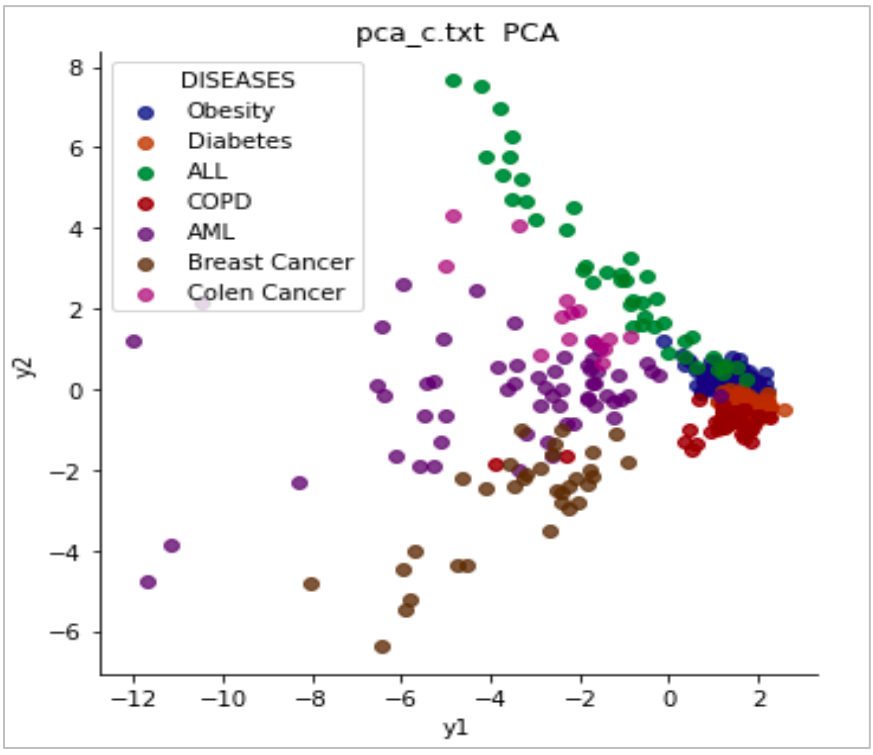


3. t-SNE plot

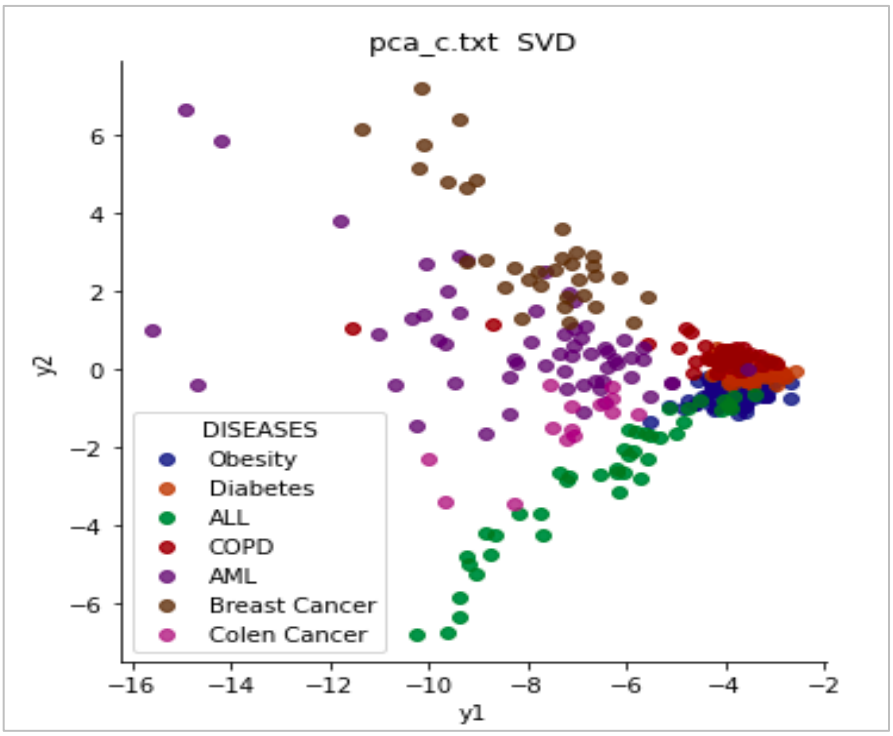


PCA_C Results

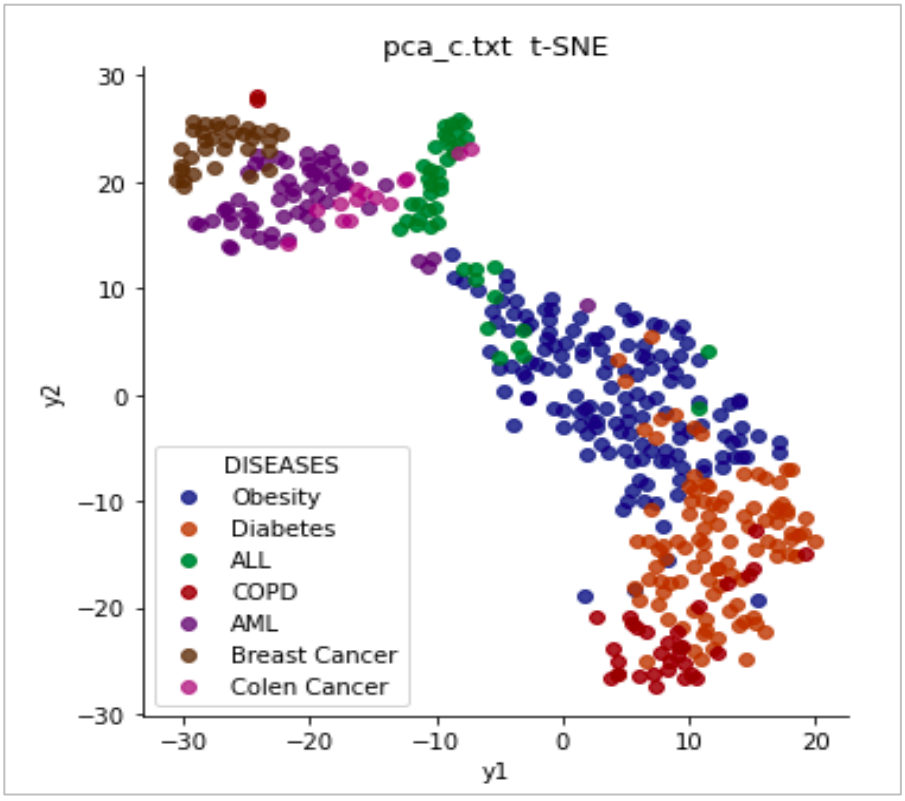
1. PCA plot



2. SVD plot



3. t-SNE plot



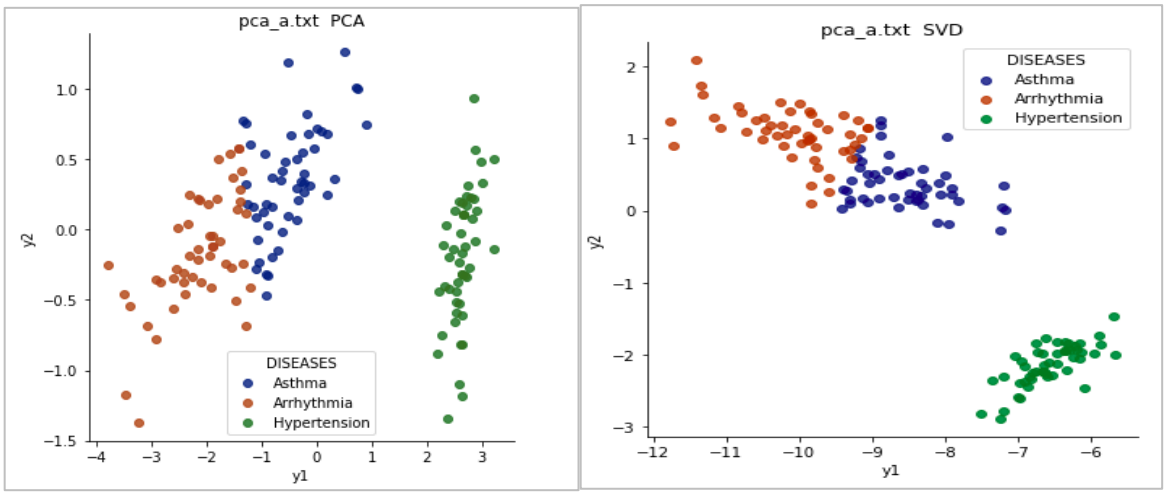
Discussion

Principal Component Analysis (PCA) essentially work by separating points as far as possible based on highest varying field using mathematical analysis.

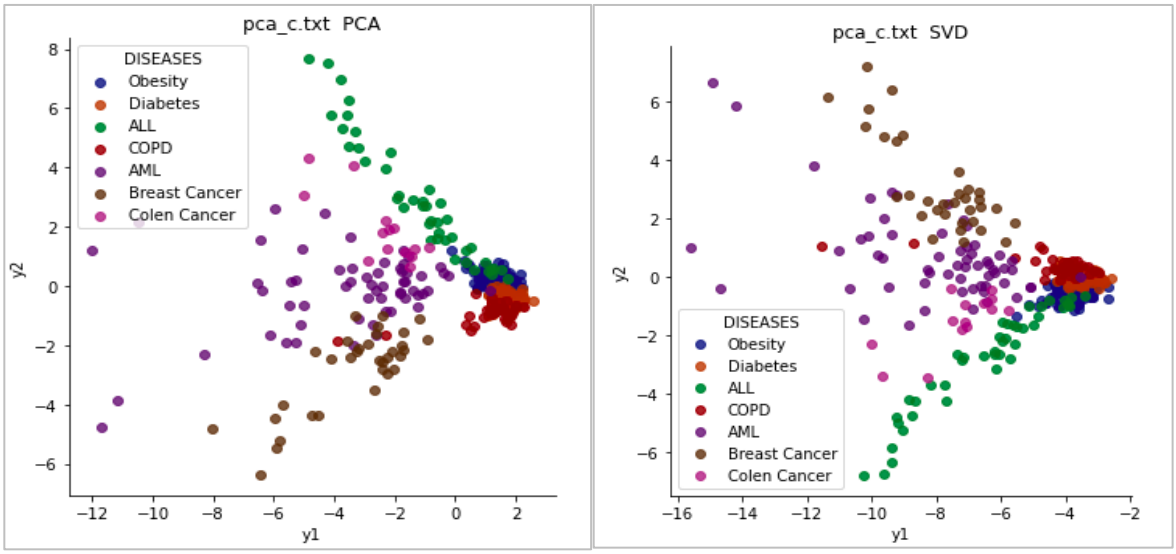
PC1 > PC2 > PC3

Singular Value Decomposition (SVD) is alike to PCA because of the fact that in both of them eigen vectors with less eigen values will be ignored for computing the new dimensions. Therefore, the results will be almost similar (or mirror images) when used with the mean-centered matrix.

Example – For figures of "pca_a" file both "PCA" and "SVD" plots can be observed as mirror images as shown below.



Similar behavior can be observed with the "PCA" and "SVD" plots of "pca_c" file.



t-Distributed Stochastic Neighbor Embedding (t-SNE) differs from both PCA and SVD in addressing the visualization of the points. This is more of a probabilistic distribution.

In t-SNE, the points are grouped as closely as possible based on characteristics of the point. It preserves the neighborhood of the points by forming clusters and uses the probabilistic approach to form the new dimensions by reducing original dimensions to required number of components/dimensions. Hence, we can see that plots are varied in each execution.

At the same time, it forms the clusters more clearly when compared with PCA and SVD. The t-SNE output plots of all the datasets resembles this notion. In short, t-SNE helps in providing better visualizations of the points in comparison with PCA and SVD.