Part 1: Dimensionality Reduction

Team Members

Siddharth Pandey

(sppandey@buffalo.edu)

Abhijeet Jadhav

(ajadhav4@buffalo.edu)

Anirudh Dalmia

(aldmia@buffalo.edu)

Overview: In the first part of project we have implemented Principle Component Analysis, Singular Vector Decomposition and t- Distributed Stochastic Neighbor Embedding (t-SNE) as methods of dimensionality reduction for high dimensional data.

Resources: In this project we have tested our data on 4 files, pca_a.txt, pca_b.txt, pca_c.txt for testing and pca_demo.txt for final demo.

1. **Principle Component Analysis on the files:**

   Principal Component Analysis or PCA is a linear feature extraction technique. It performs a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized. PCA combines your input features in a specific way that you can drop the least important feature while still retaining the most valuable parts of all of the features. As an added benefit, each of the new features or components created after PCA are all independent of one another.

- **Scatter plots for PCA on pca_a.txt, pca_b.txt, pca_c.txt.:**

Fig.1 PCA on pca_a.txt file

```
--------------------------------------------------------------------------------
SCATTER PLOT FOR PCA OF FILE: pca_b.txt
```

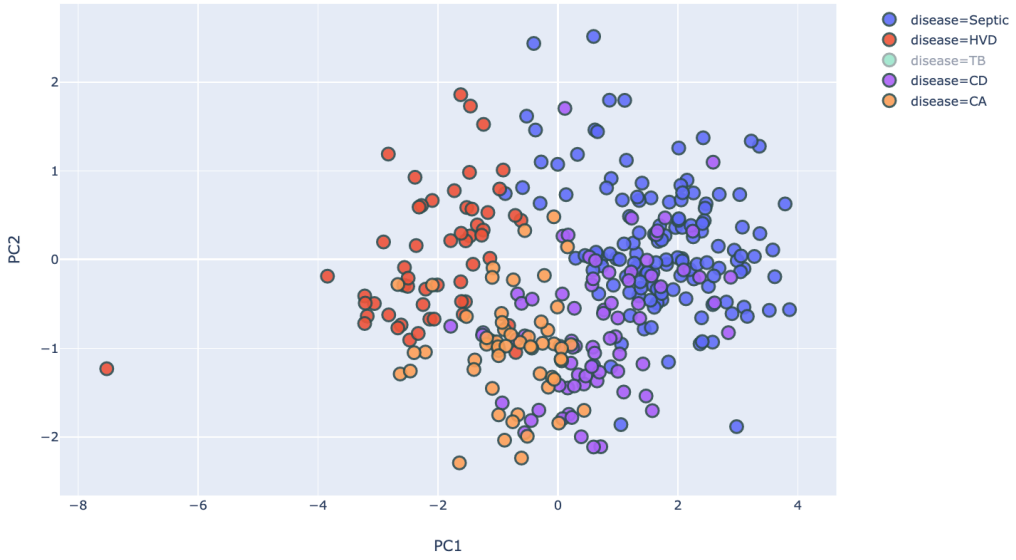

Fig.2 PCA on pca_b.txt file

```
--------------------------------------------------------------------------------
SCATTER PLOT FOR PCA OF FILE: pca_c.txt
```

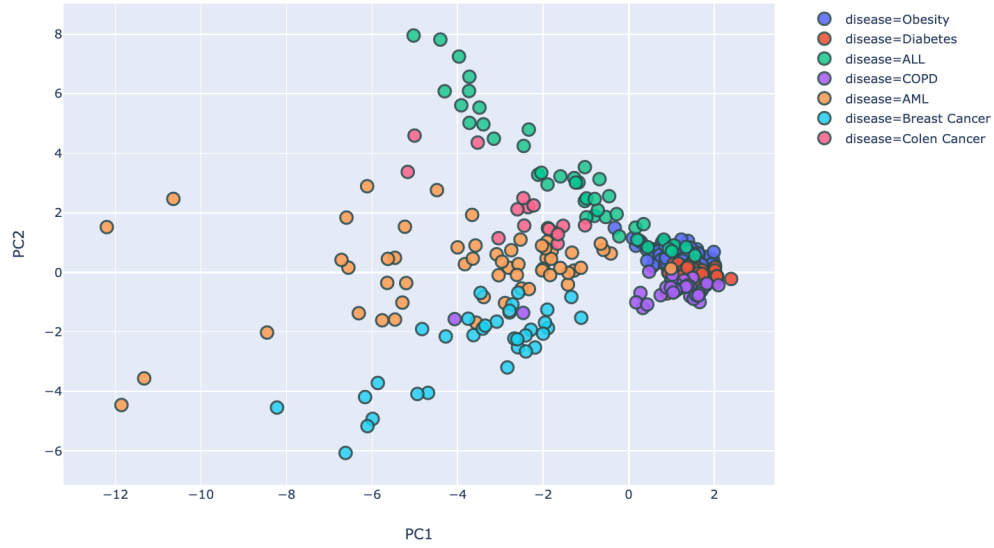

Fig.3 PCA on pca_b.txt file

2. Singular Vector Decomposition:

In linear algebra, the singular value decomposition (SVD) is a factorization of a real or complex matrix. It is the generalization of the eigendecomposition of a positive semidefinite normal matrix (for example, a symmetric matrix with non-negative eigenvalues) to any {\displaystyle m\times n}m\times n matrix via an extension of the polar decomposition.

- **Scatter plots for SVD on pca_a.txt, pca_b.txt, pca_c.txt.:**

```
--------------------------------------------------------------------------------
SCATTER PLOT FOR SVD OF FILE: pca_a.txt
```

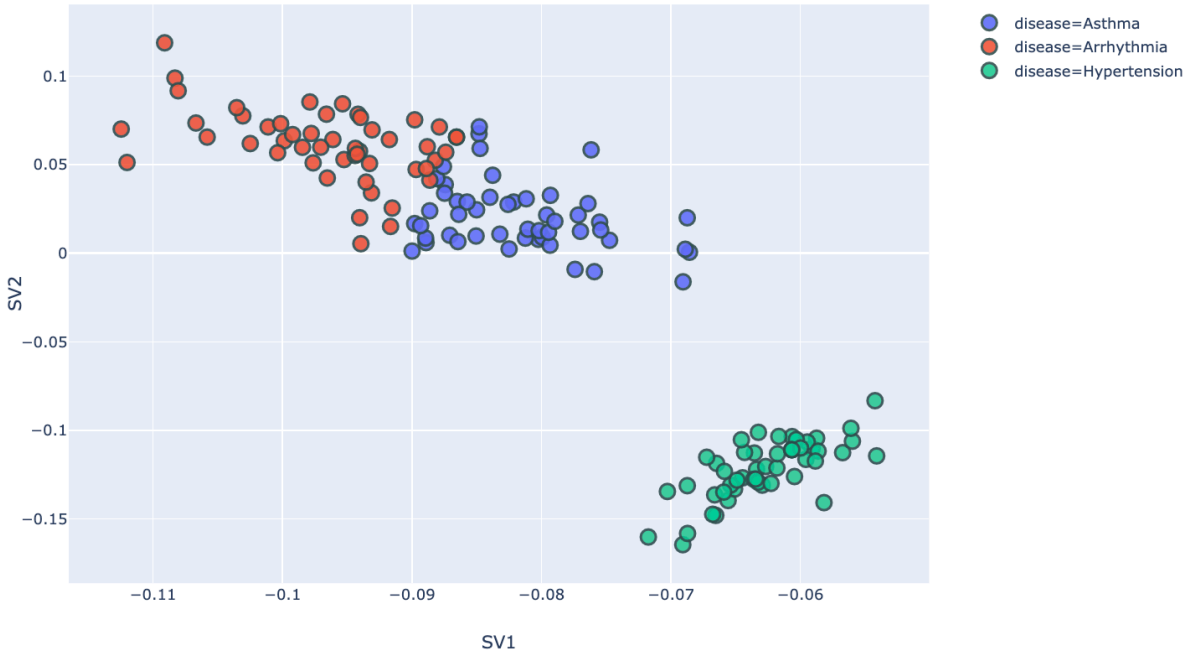

Fig.4 SVD on pca_b.txt file

```
--------------------------------------------------------------------------------
SCATTER PLOT FOR SVD OF FILE: pca_b.txt
```

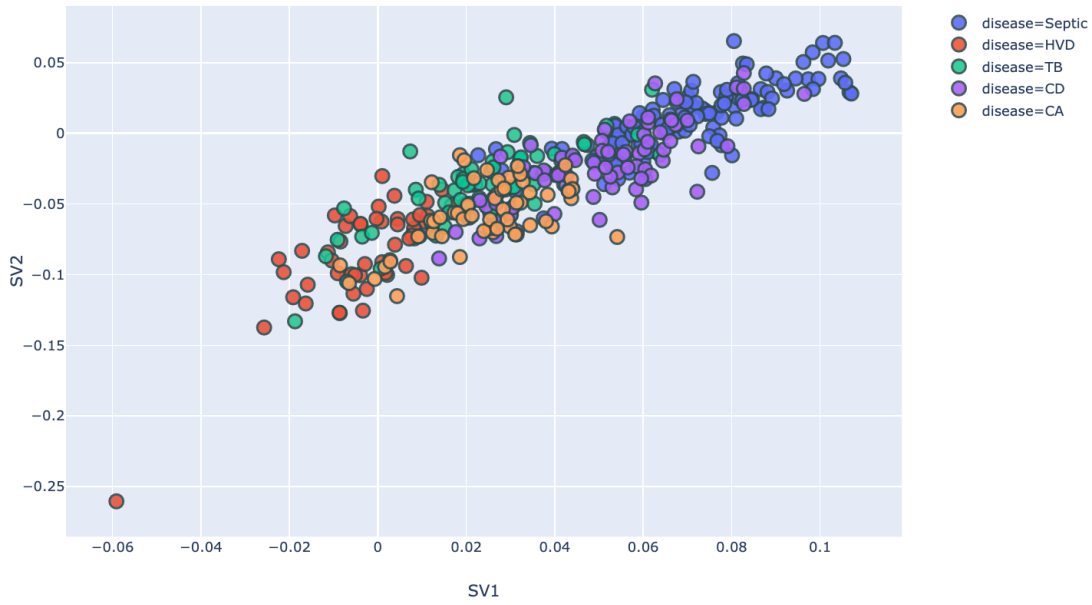

Fig.5 SVD on pca_b.txt file

```
--------------------------------------------------------------------------------
SCATTER PLOT FOR SVD OF FILE: pca_c.txt
```

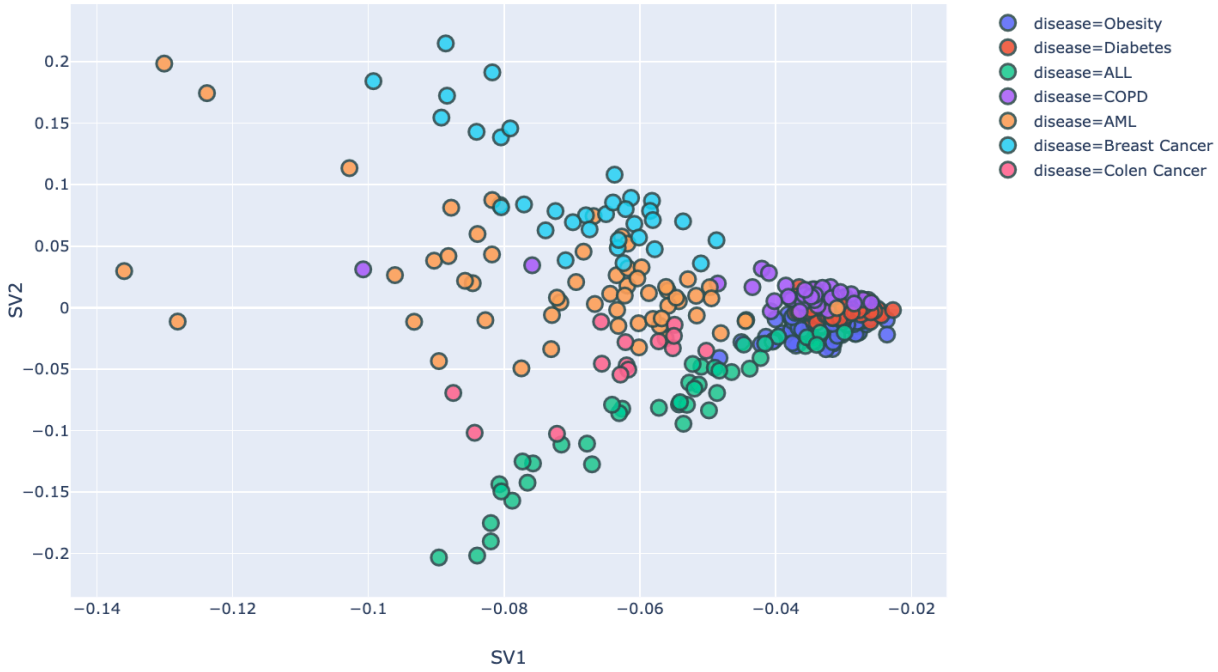

Fig.6 SVD on pca_c.txt file

3. t- Distributed Stochastic Neighbor Embedding (t-SNE) :

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets.

- **Scatter plots for t-SNE on pca_a.txt, pca_b.txt, pca_c.txt.:**

```
--------------------------------------------------------------------------------
NOW WORKING ON FILE pca_a.txt
--------------------------------------------------------------------------------
[t-SNE] Computing 121 nearest neighbors...
[t-SNE] Indexed 150 samples in 0.000s...
[t-SNE] Computed neighbors for 150 samples in 0.002s...
[t-SNE] Computed conditional probabilities for sample 150 / 150
[t-SNE] Mean sigma: 0.669058
[t-SNE] KL divergence after 250 iterations with early exaggeration: 48.583202
[t-SNE] Error after 400 iterations: 0.084539
SCATTER PLOT FOR t-SNE OF FILE: pca_a.txt
```

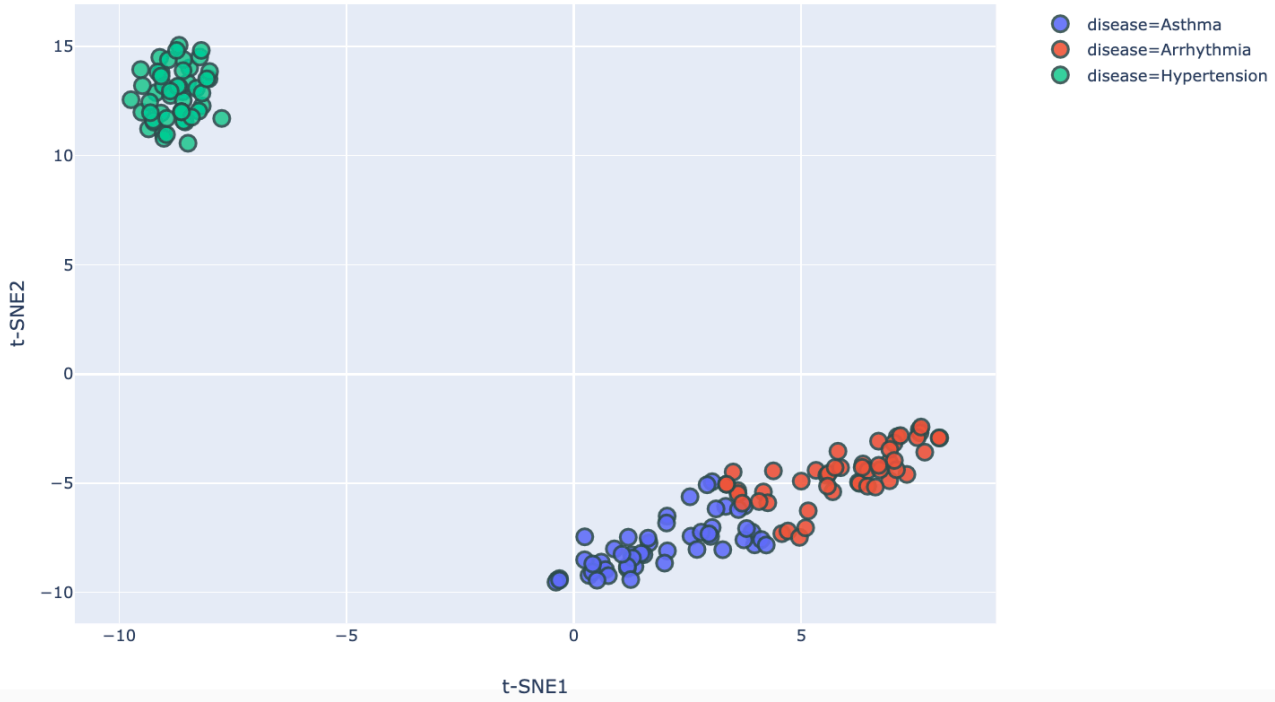

Fig.7 t-SNE on pca_c.txt file

```
--------------------------------------------------------------------------------
NOW WORKING ON FILE pca_b.txt
--------------------------------------------------------------------------------
[t-SNE] Computing 121 nearest neighbors...
[t-SNE] Indexed 386 samples in 0.001s...
[t-SNE] Computed neighbors for 386 samples in 0.036s...
[t-SNE] Computed conditional probabilities for sample 386 / 386
[t-SNE] Mean sigma: 0.824576
[t-SNE] KL divergence after 250 iterations with early exaggeration: 65.691788
[t-SNE] Error after 400 iterations: 0.603519
SCATTER PLOT FOR t-SNE OF FILE: pca_b.txt
```

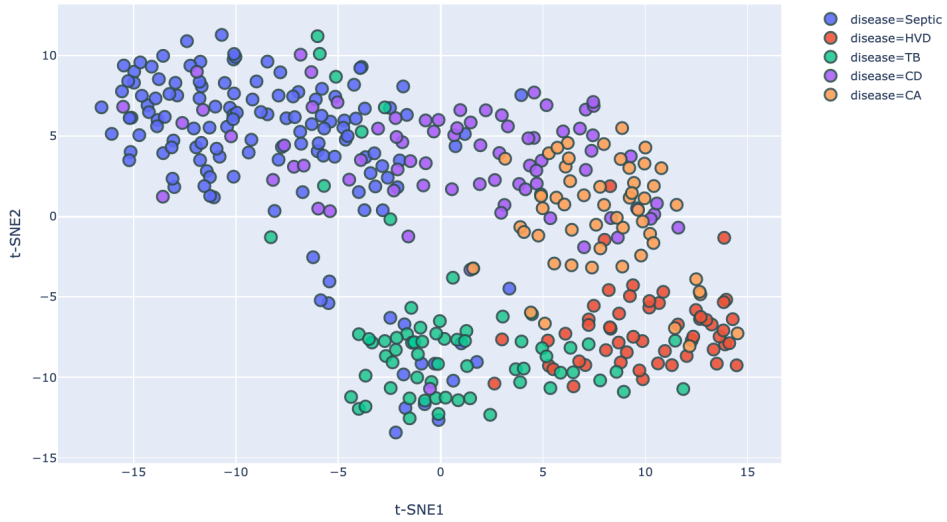

Fig.8 t-SNE on pca_c.txt file

```
--------------------------------------------------------------------------------
NOW WORKING ON FILE pca_c.txt
--------------------------------------------------------------------------------
[t-SNE] Computing 121 nearest neighbors...
[t-SNE] Indexed 428 samples in 0.000s...
[t-SNE] Computed neighbors for 428 samples in 0.011s...
[t-SNE] Computed conditional probabilities for sample 428 / 428
[t-SNE] Mean sigma: 0.365075
[t-SNE] KL divergence after 250 iterations with early exaggeration: 62.174610
[t-SNE] Error after 400 iterations: 0.525936
SCATTER PLOT FOR t-SNE OF FILE: pca_c.txt
```

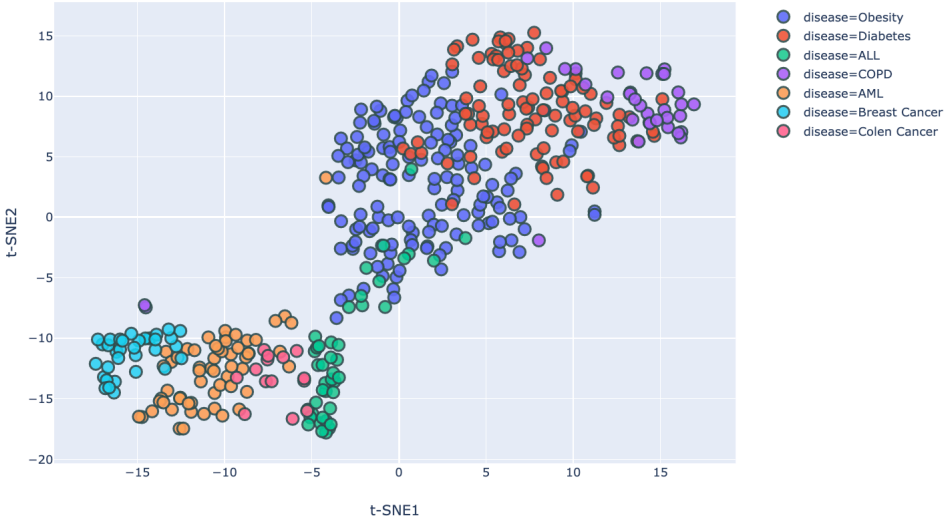

Fig.9 t-SNE on pca_c.txt file

PCA code Implementation Overview:

In our code implementation we have written the code from scratch in python. The basic flow of the data in the code is as follows:

1. All text files for dimensionality reduction are parsed and stored in dataframe in a loop.
2. To center the values of the data we have subtracted the mean of each column value from the respective column elements.

```
centeredFeatureArrayPCA = featureArrayPCA-featureArrayPCA.mean()
```

3. To generate the covariance matrix we made use of the np.cov() method from the numpy package to save computational time.
4. 4. In addition to the previous step we also calculated the eigen values and eigen vectors using the numpy package.

```
covarianceMatrix = np.cov(centeredFeatureArrayPCA.T)
eigen_values, eigen_vectors = np.linalg.eig(covarianceMatrix)
eig_pairs = [(np.abs(eigen_values[i]), eigen_vectors[:,i]) for i in range(len(eigen_values))]
eig_pairs = sorted(eig_pairs, key=lambda x : x[0])
eig_pairs.reverse()
```

5. Once the eigen values are obtained we use these values to determine the top 2 dimensions that show maximum variance and then plot it using the plotly express package.