State University of New York at Buffalo
CSE- 601 Data Mining and Bioinformatics

Project1 Part 1: Dimensionality Reduction

Karthikeyan Ramachandriya Subramanian 50289080
Riya Hazra 50290600
Shwetasree Chowdhury 50296995

Problem Statement:

Conduct dimensionality reduction on three biomedical data files (pca_a.txt, pca_b.txt, pca_c.txt). In each file, each row represents the record of a patient/sample; the last column is the disease name, and the rest of the columns are features. Algorithms used: PCA, SVD, t-SNE.

PCA and Dimensionality Reduction:

The need for PCA and Dimensionality Reduction stems from the fact that certain datasets may contain data with too many attributes, so much so that representing them on any plane may be difficult, hence hindering possible data visualisation.

- PCA is an algorithm that solves the above conundrum by reducing the muti-dimensional data into a 2 dimensional space for better representation.
- PCA works by retaining the variation present in the original variables. In a 2 dimensional plane, the 1st principal component retains maximum variation that was present in the original components, which can be viewed as a rotation of the existing axes in the space defined by original dimensions.
- The axes, so plotted are orthogonal and represent directions with maximum variability.
- The principal components are the eigenvectors of a covariance matrix, and hence they are orthogonal.

PCA Algorithm

1 – Standardize: The scale of data is standardized. The original data is centered around the mean by adjusting original data by the mean. $X = X - \bar{X}$
2 – Calculate covariance.
3 – Deduce eigens: Find the eigenvectors and eigenvalues of S. Select the top 2 eigen values and their corresponding eigen vectors which form the principal components.
4 – Reorient data: Data is re-oriented from a n-dimensional space to a 2-dimensional space.

PCA Steps implemented

1. reading the data from .txt file to the dataframe and partitioning the data into features and class label.
   data = pd.read_csv("pca_c.txt", sep="\t", header=None)
   mean_vector = mean_vector(data)

2. Take data in dataframe as an input and produce k dimensional mean vector - function mean_vector
   col_mean = np.mean(data[i,:])

3. subtract the mean from the original data to get the new data using function mean_vector

4. k_largest_eigen_val returns the eigen vector with k largest eigen value
   eig_val, eig_vec = np.linalg.eig(cov_matrix)

5. calculating the covariance matrix for the dataset using cov() function
   cov_matrix = data.cov()

6. Sorting the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigen value
   k_largest = k_largest_eigen_val(eig_val,eig_vec,k)

7. Transforming the samples onto the new subspace
8. Visualization of the 2D data

SVD Algorithm

- SVD can be used for dimensionality reduction, most often used in digital signal processing for noise reduction, image compression, and other areas.
- SVD is an algorithm that factors an m x n matrix, M, of real or complex values into three component matrices, where the factorization has the form USV*.
- U is an m x p matrix. S is a p x p diagonal matrix. V is an n x p matrix, with V* being the transpose of V, a p x n matrix, or the conjugate transpose if M contains complex values. The value p is called the rank.
- SVD differs from PCA, because it avoids the use of eigenvectors and eigenvalues for calculation of covariance matrix and instead uses a simple 3 component USV.

SVD Steps implemented

1. reading the data from .txt file to the dataframe and partitioning the data into features and class label.

data = pd.read_csv("pca_c.txt", sep="\t", header=None)

label = data.iloc[:,-1]

2. Calculate SVD value
   U, s, V = np.linalg.svd(data)

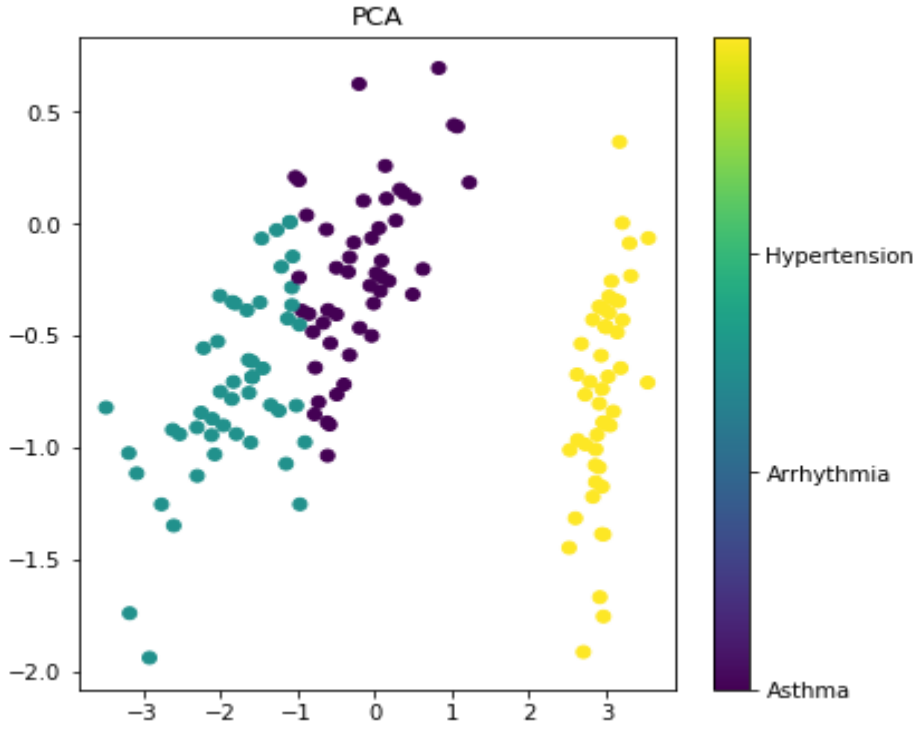
3. Transforming the samples onto the new subspace
4. Visualization of the 2D data

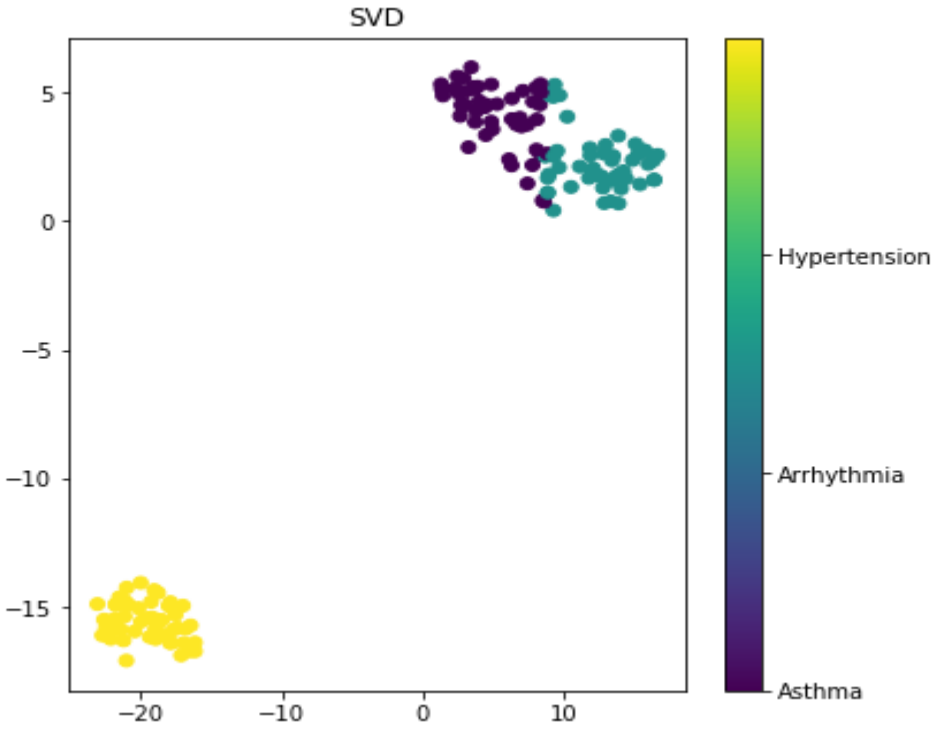
t-SNE Algorithm

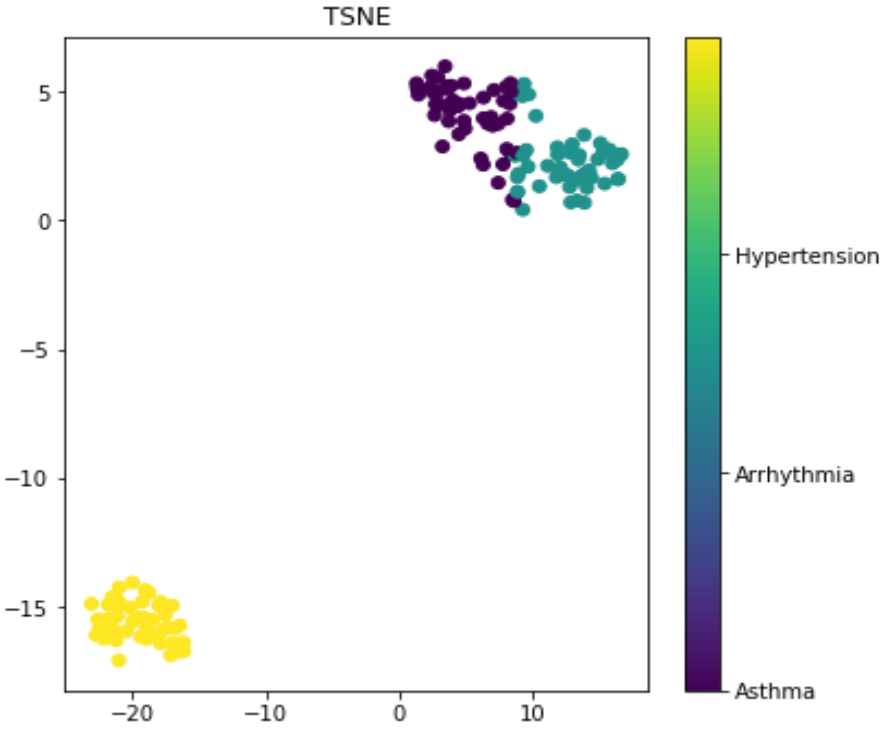
- t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets.
- t-SNE constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a high probability of being picked while dissimilar points have an extremely small probability of being picked.
- t-SNE defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the Kullback–Leibler divergence between the two distributions with respect to the locations of the points in the map

Scatter Plots for Datasets

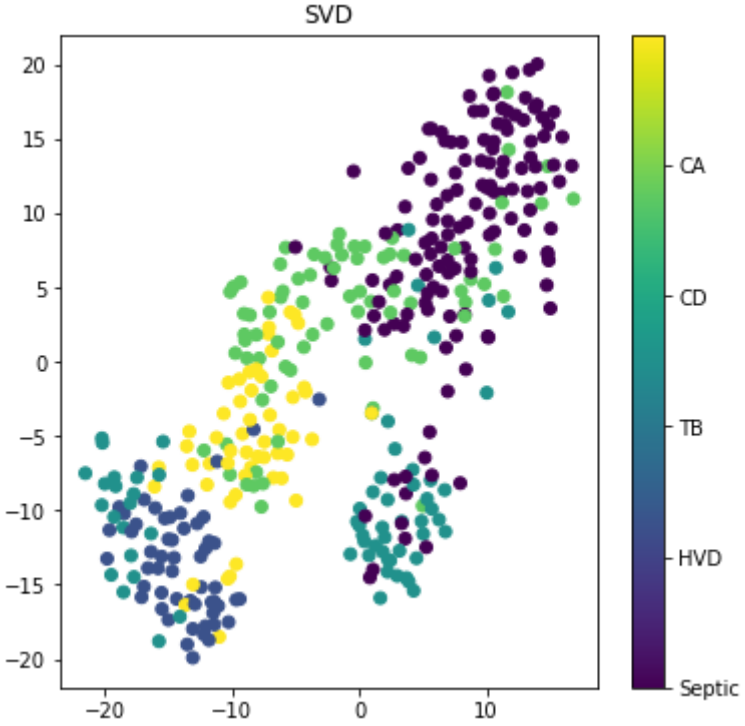
1. **Pca_a.txt**

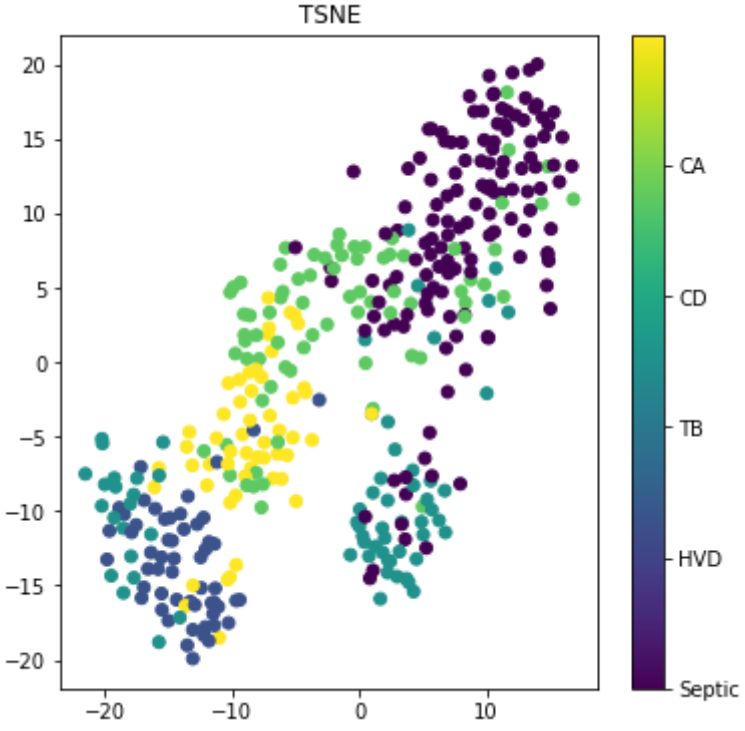


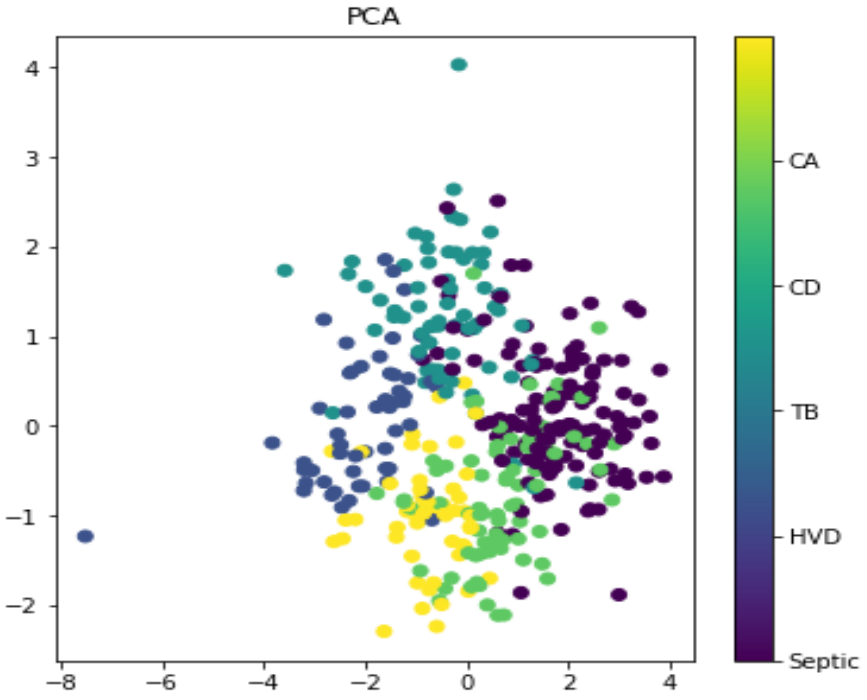

SVD
5
0
−5
−10
−15
−20
−10
0
10
Hypertension
Arrhythmia
Asthma


TSNE
5
0
−5
−10
−15
−20
−10
0
10
Hypertension
Arrhythmia
Asthma

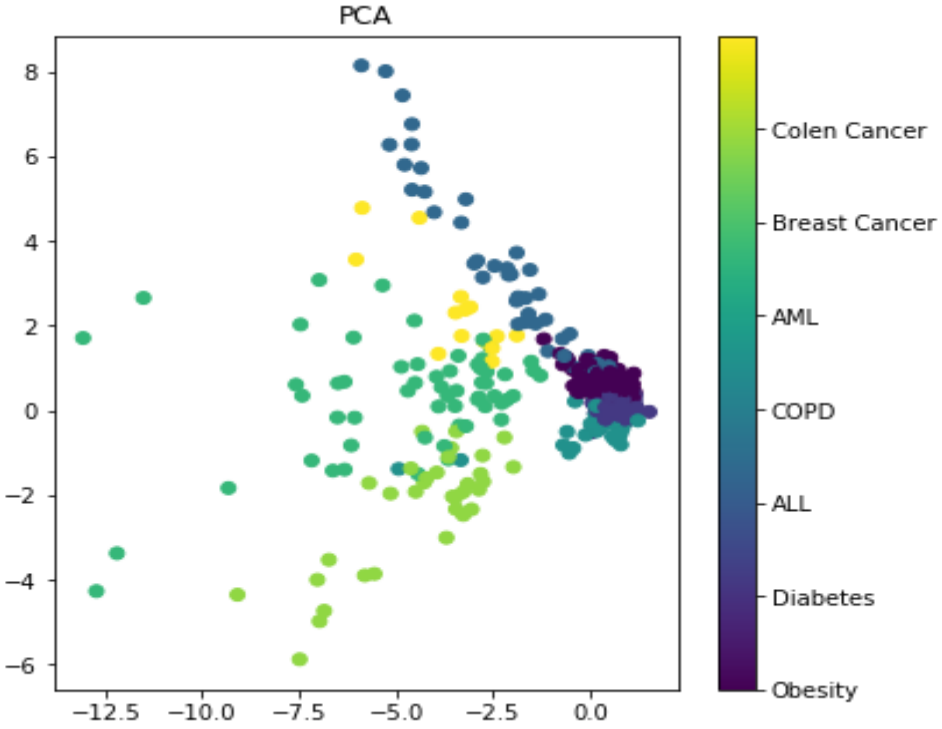
2. pca_b.txt

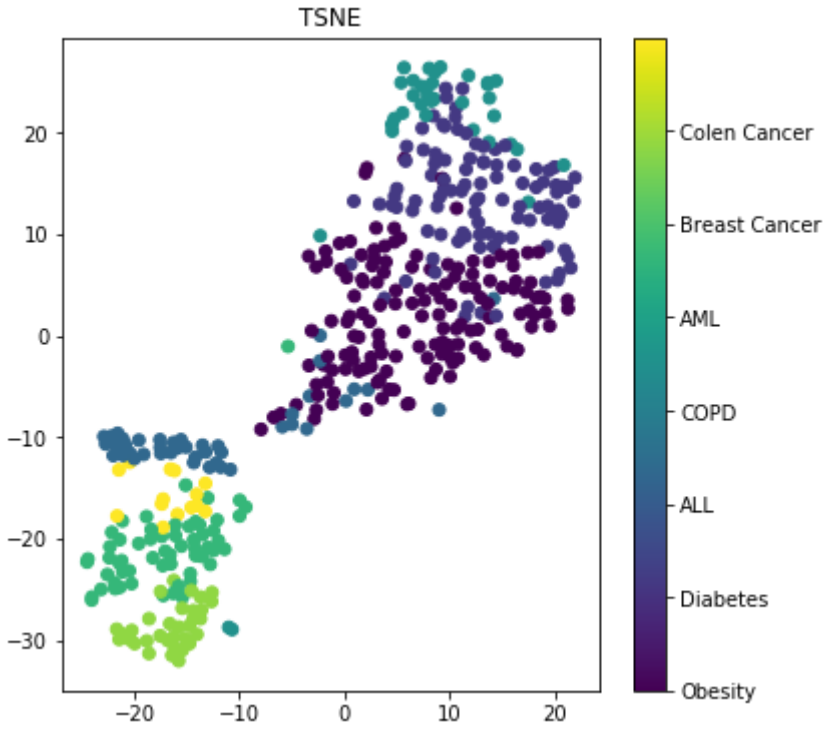

SVD
20
15
10
5
0
−5
−10
−15
−20
−20
−10
0
10
CA
CD
TB
HVD
Septic

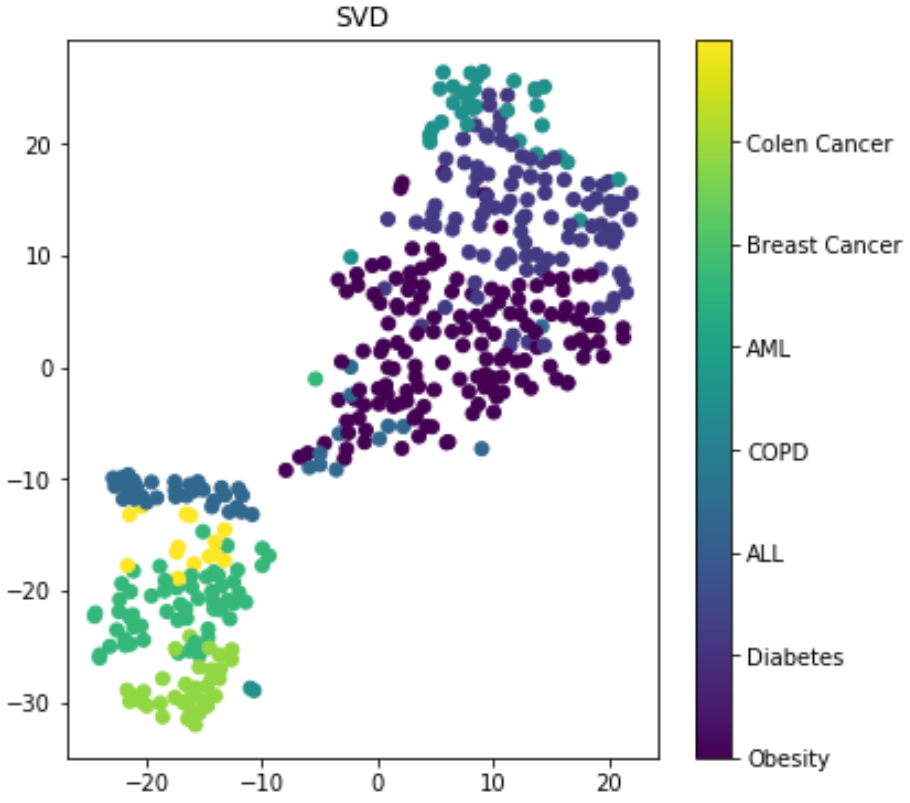

TSNE
20
15
10
5
0
−5
−10
−15
−20
−20
−10
0
10
CA
CD
TB
HVD
Septic


PCA
4
3
2
1
0
−1
−2
−8
−6
−4
−2
0
2
4
CA
CD
TB
HVD
Septic

3. Pca_c.txt


PCA
8
6
4
2
0
−2
−4
−6
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
Colen Cancer
Breast Cancer
AML
COPD
ALL
Diabetes
Obesity





Inferences

1. SVD and t-SNE will provide similar graphs as compared to PCA, as PCA takes into account only the difference of the mean with 0 because PCA takes difference of the mean, relative to the dataset present in the set.

2. PCA is a mathematical technique, but t-SNE is a probabilistic one.
3. Linear dimensionality reduction algorithms, like PCA, concentrate on placing dissimilar data points far apart in a lower dimension representation. But in order to represent high dimension data on low dimension, non-linear manifold, it is essential that similar data points must be represented close together, which is something t-SNE does, not PCA.
4. Sometimes in t-SNE different runs with the same hyperparameters may produce different results hence multiple plots must be observed before making any assessment with t-SNE, while this is not the case with PCA.
5. Since PCA is a linear algorithm, it will not be able to interpret the complex polynomial relationship between features while t-SNE is made to capture exactly that.
6. From the above plots, one can notice that t-SNE and SVD methods have similarity in terms of their plotting style, while PCA scatter plots are a little scattered and spread out. And therefore not suited for extremely high manifold data.

References:

1. https://en.wikipedia.org/wiki/Principal_component_analysis
2. https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b
3. https://stats.stackexchange.com/questions/238538/are-there-cases-where-pca-is-more-suitable-than-t-sne