Data Mining Project 1
Abhav Luthra, Tanmay Singh, Krishna Sehgal

Aim

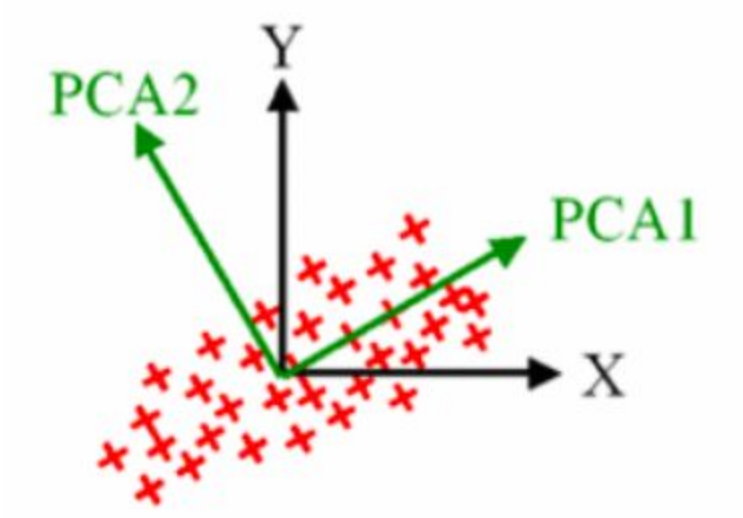
Implementation of PCA, SVD, and T-SNE and then run it on three data files (to get the two-dimensional data points. For each dataset, draw the data points with a scatter plot, and color them according to their disease names.

Background

A high-dimensional dataset is a dataset that has a great number of columns (or variables). Most of the times these variables (or called features) are correlated. So, we can transform the variables into a new set of variables without losing much information.
Few of the popular ones are implemented in this project:

1. **Principal Component Analysis**

The idea is to reduce the dimensionality of a dataset consisting of a large number of related variables while retaining as much variance in the data as possible. PCA finds a set of new variables that the original variables are just their linear combinations. The new variables are called *Principal Components (PCs)*. These principal components are *orthogonal*: In a 3-D case, the principal components are perpendicular to each other. X can not be represented by Y or Y cannot be presented by Z. The first principal component will capture most of the variance in the data, then followed by the second, third, and so on. As a result, the new data will have fewer dimensions.



PCA was implemented in python, we first used **StandardScaler** in scikit-learn to standardize the dataset's features onto the unit scale (mean = 0 and variance = 1). Then we calculated the

eigenvalue and vector and used the top 2 eigenvalue vectors and dot product that with the matrix and used that 2D matrix to plot a 2D graph representing the data points.

2. **Singular Value Decomposition**

Let's start with explaining this matrix operation, we have a matrix A, if A is a symmetric real $n \times n$ matrix, there exists an orthogonal matrix V and a diagonal D such that

$$A = VDV^T$$

Columns V are eigenvectors for A and the diagonal entries of D are the eigenvalues of A. This process is called the *Eigenvalue Decomposition*, or *EVD*, for matrix *A*. It tells us how to choose orthonormal bases so that the transformation is represented by a matrix with the simplest possible form, that is, diagonal. Extending the symmetric matrix, the SVD works with any real $m \times n$ matrix *A*. Given a real m× n matrix *A*, there exists an orthogonal $m \times m$ matrix *U,* an orthogonal matrix $m \times m$ *V,* and a diagonal $m \times n$ matrix Σ such that

$$A = U\boldsymbol{\Sigma}V^T$$

We have implemented SVD, using sklearn library in python

3. ***t*-distributed Stochastic Neighbor Embedding (t-SNE)**

t-SNE is developed by Laurens van der Maaten and Geoggrey Hinton. It is a machine learning algorithm for visualization that presents embedding high-dimensional data in a low-dimensional space of two or three dimensions.

t-SNE models the similarities among points. First, it is defined by the Euclidean distance between point *Xi* and *Xj*. Second, it is defined as the conditional probability that “the similarity of data point *i* to point *j* is the conditional probability *p* that point *i* would pick data *j* as its neighbor if other neighbors were picked according to their probabilities under a Gaussian distribution.” In the following conditional expression, if point *j* is closer to point *i* than other points, it has a higher probability (notice the negative sign) to be chosen.

$$p_{j|i} = \frac{\exp\left(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-||\mathbf{x}_i - \mathbf{x}_k||^2/2\sigma_i^2\right)}$$

t-SNE aims to match the above conditional probability *p* between *j* and *i* as well as possible by a low-dimensional space *q* between point *Yi* and *Yj,* as shown below. The probability *q* follows a fat-tailed Student-t distribution, thus the “*t*” in t-SNE comes from.

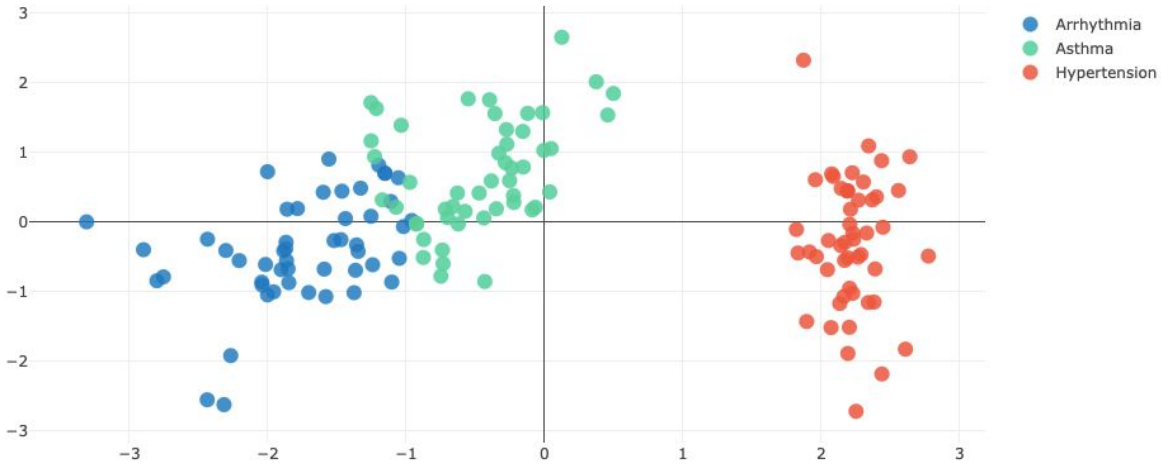
$$q_{ij} = \frac{\left(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2\right)^{-1}}{\sum_{k \neq l} \left(1 + ||\mathbf{y}_k - \mathbf{y}_l||^2\right)^{-1}}$$

The next step is to find *Yi* such that the distribution *q* will be as close to the distribution *p* as possible. t-SNE uses the gradient descent technique, an optimization technique, to find the values. We have implemented t-SNE, using sklearn library in python

Results

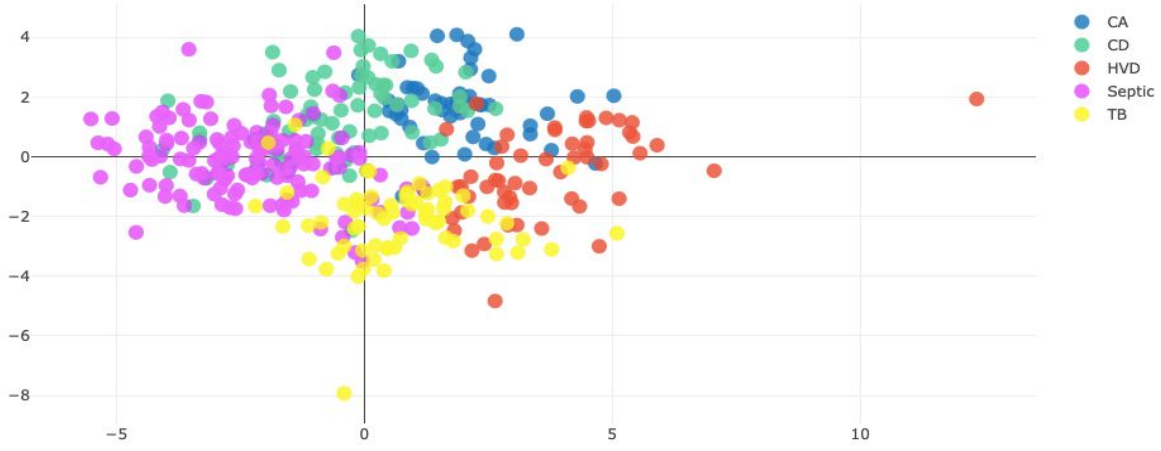
PCA

1. Dataset A



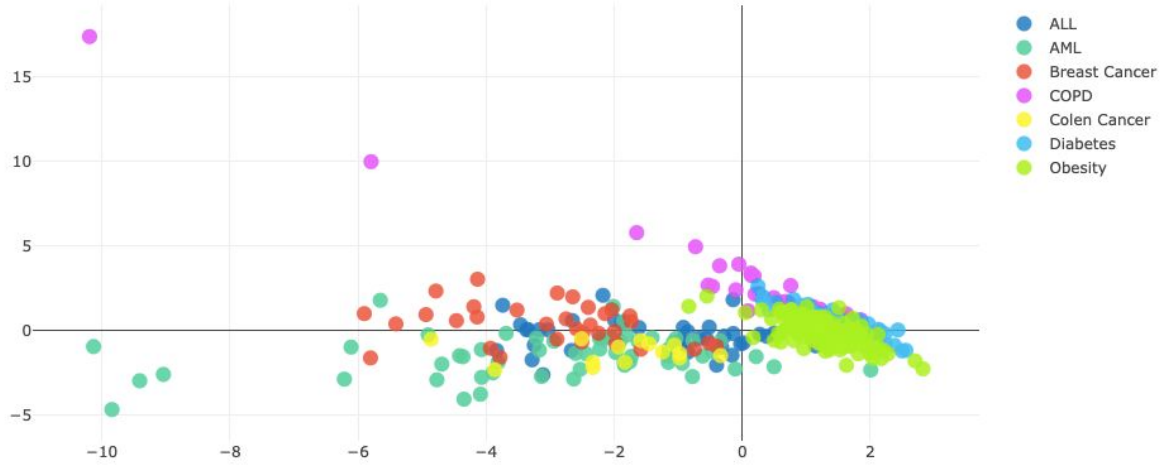
The graph has some overlapping of Asthma and Arrhythmia, but most of the time the diagnosis and independent of each other.

2. Dataset B



The graph has overlapping of many of the diagnosed diseases.

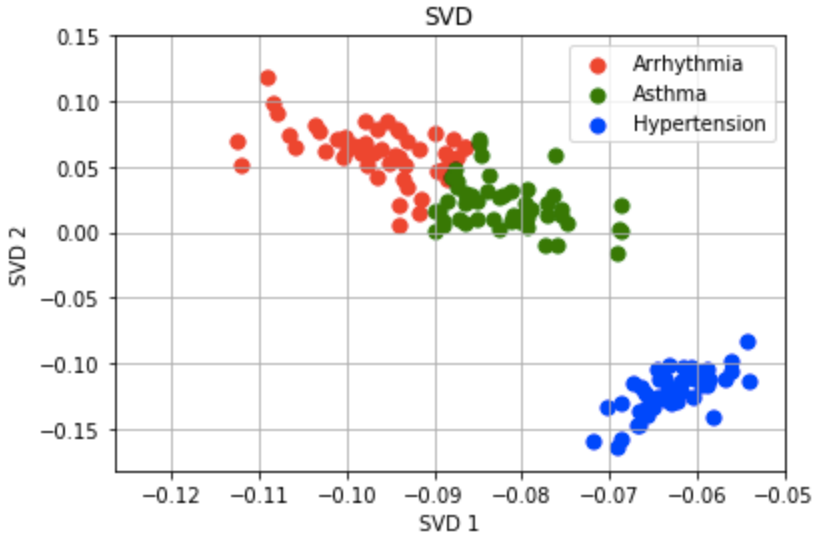
3. Dataset C



The graph has overlapping of many of the diagnosed diseases.

SVD

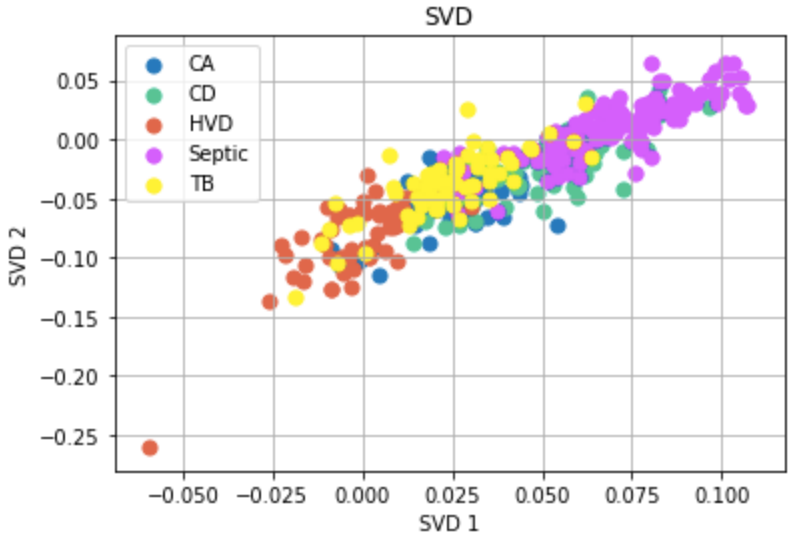
1. Dataset A



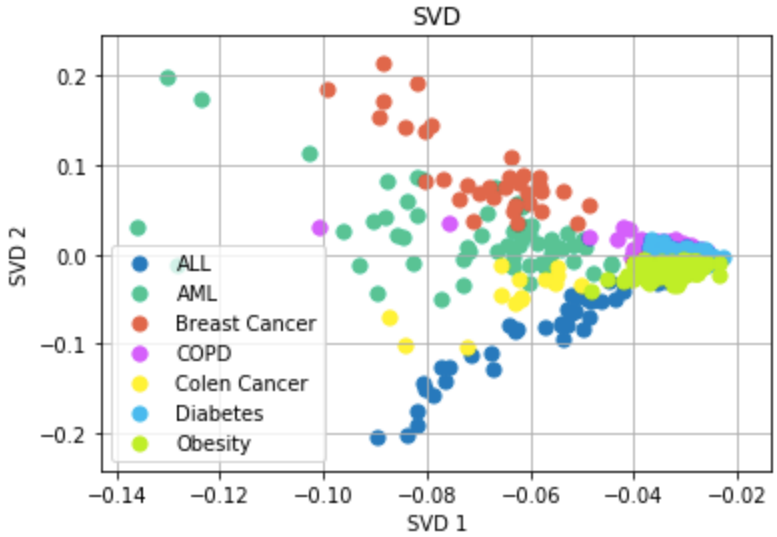
Similar to PCA, the graph has some overlapping of Asthma and Arrhythmia, but most of the time the diagnosis and independent of each other.

2. Dataset B

   Similar to PCA, The graph has overlapping of many of the diagnosed diseases.



3. Dataset C

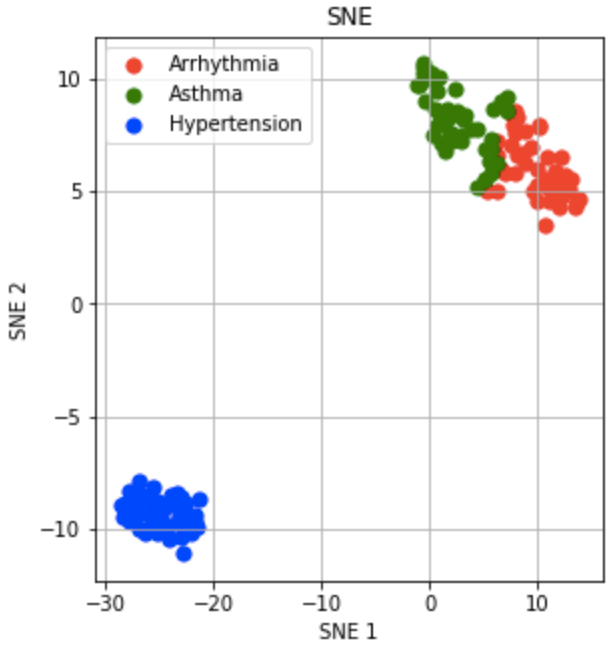


   Similar to PCA, The graph has overlapping of many of the diagnosed diseases.

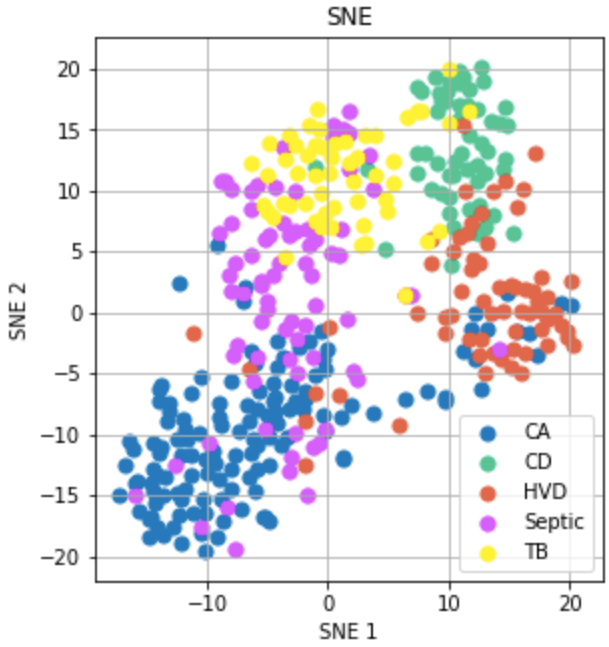
t-SNE

1. Dataset A

Similar to PCA, the graph has some overlapping of Asthma and Arrhythmia, but most of the time the diagnosis and independent of each other.

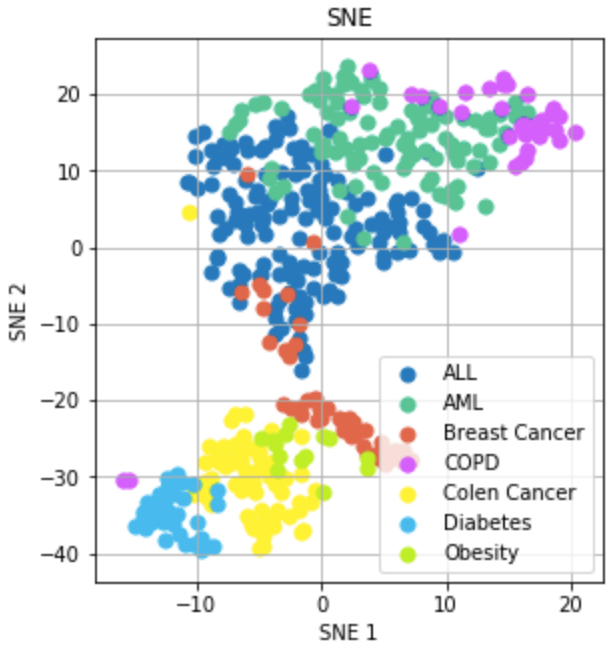


2. Dataset B



Similar to PCA, The graph has overlapping of many of the diagnosed diseases.

3. Dataset C



Similar to PCA, The graph has overlapping of many of the diagnosed diseases.

References:

1. https://towardsdatascience.com/dimension-reduction-techniques-with-python-f36ca7009e5c
2. https://plot.ly/ipython-notebooks/principal-component-analysis/