CSE 601: DATA MINING AND BIOINFORMATICS
FALL 2019

PROJECT 1: PCA AND APRIORI ALGORITHM

PART 1: DIMENSIONALITY REDUCTION

REPORT

Keerthana Baskaran UB NO: 50288944 (UBIT: kbaskara)

Sriram Venkataramanan UB NO : 50289666 (UB IT: sv84)

Vivek Nagaraju UB NO : 50288711 (UB IT: viveknag)

PROBLEM:

For the first part, we should implement PCA (Principle Components Analysis) algorithm, project the high-dimensional data to 2 dimensions, and plot the 2-dimensional data points.

Dataset Description:

In this part, you are expected to 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 columns are features.
Note that your code should be able to handle the data with different numbers of rows/columns.

Requirements:

Task 1: Implement PCA algorithm by choosing our convenience of language.

Task 2:

Implement PCA and then run it on three data files (*pca_a.txt*, *pca_b.txt*, *pca_c.txt*) 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.

Task 3:

Apply existing packages to run SVD and t-SNE algorithms (Do not need to implement them by yourself) and get the two-dimensional data points. Visualize the data points of the two algorithms on the three datasets in the same way as the visualization of PCA results in step 2.

SOLUTION:

TASK 1:

We implemented PCA in Python without using any existing packages.

Before explaining the steps used in implementing the PCA algorithm, we need to know the basic concept of Principal Component Analysis.

PCA:

PCA is a method for **compressing a lot of data into something that captures the essence of original data.** To be precise, it is a dimensionality reduction technique where the dimensions of very big or large data sets are reduced by transforming a larger set of variables into smaller ones. The smaller set of variables consists most of the information that was present from the larger set.

In sum, PCA is a concept which is **simple by reducing the number of variables in a dataset,** while trying to **preserve as much meaningful information as possible.**

IMPLEMENTATION STEPS:

The packages used for implementation are: numpy and matplotlib.

1) The file is passed as an argument in command line argument. It can be one of the three files:
   Pca_a.txt
   Pca_b.txt
   Pca_c.txt

2) The `pca_svd_tnse_plot(file )` method is called: The file is passed as argument to the method.
3) The data from the file is imported, and it is converted into an array using **numpy.asarray()**, a function used when we want to convert input to an array. Input data, in any form that can be converted to an array.
4) All columns from the data file, except the last column are taken as attributes and converted into a matrix using **numpy.matrix()** .
5) The last column disease is kept in a separate list "disease_col".

6) **PCA IMPLEMENTATION INSIDE THE METHOD AND EXPLANATION:**

Step 1)

- The mean vector is computed i.e. "attr_mean_colwise ". It consists of column wise mean values.
- Then the standardized or modified attributes are computed by subtracting the column wise mean against respective values. "Standardized_attr" consists those values.

EXPLANATION:

The first two steps are called standardisation steps. They are performed so that all the variables are transformed to the same scale. The aim is to standardize the range of continuous initial variable, so that every variable contributes equally to analysis.

Step 2)

- The next step is to calculate the covariance matrix. It is done by calling the np.cov() package provided by numpy on the standardized or modified attributes which we calculated in the previous step.

EXPLANATION:

This step is done to understand how the variables of input data set vary from the mean with respect to each other. It checks if there is any relationship between them.

Sometimes the variables are highly correlated in a way; it contains most of the redundant information. Thus in order to identify those correlations, we compute the covariance matrix.

From this we know that covariance matrix is a table which summarizes the correlations between all possible pairs of variables.

Step 3)

- The eigan values and eigan vectors are computed from the covariance matrix applied on standaredized_attributes using the function np.linalg.eig(standardized_attr):

EXPLANATION:

This step is essential to obtain the principal components from the covariance matrix. PC are nothing but the directions of the data that explain a maximal amount of variance, that is to say, the lines that capture the most information of the data.

Step4)

- The eigan values and vectors are sorted in descending order, so that the first two are chosen as 2 principal components which will be used for plotting the data points.

EXPLANATION:PC1: Most of the information within initial variables is compressed into the first components. This will allow us to reduce the dimensionality without losing much information and it discards the components with low information.

PC2: The second principal component is calculated in the same way, with the condition that it is uncorrelated with (i.e., perpendicular to) the first principal component and that it accounts for the next highest variance.

Step 5) pca_plot_matrix is formed by computing the dot product between standardized attributes and Eigen vectors

EXPLANATION: This matrix is used for plotting the final output.

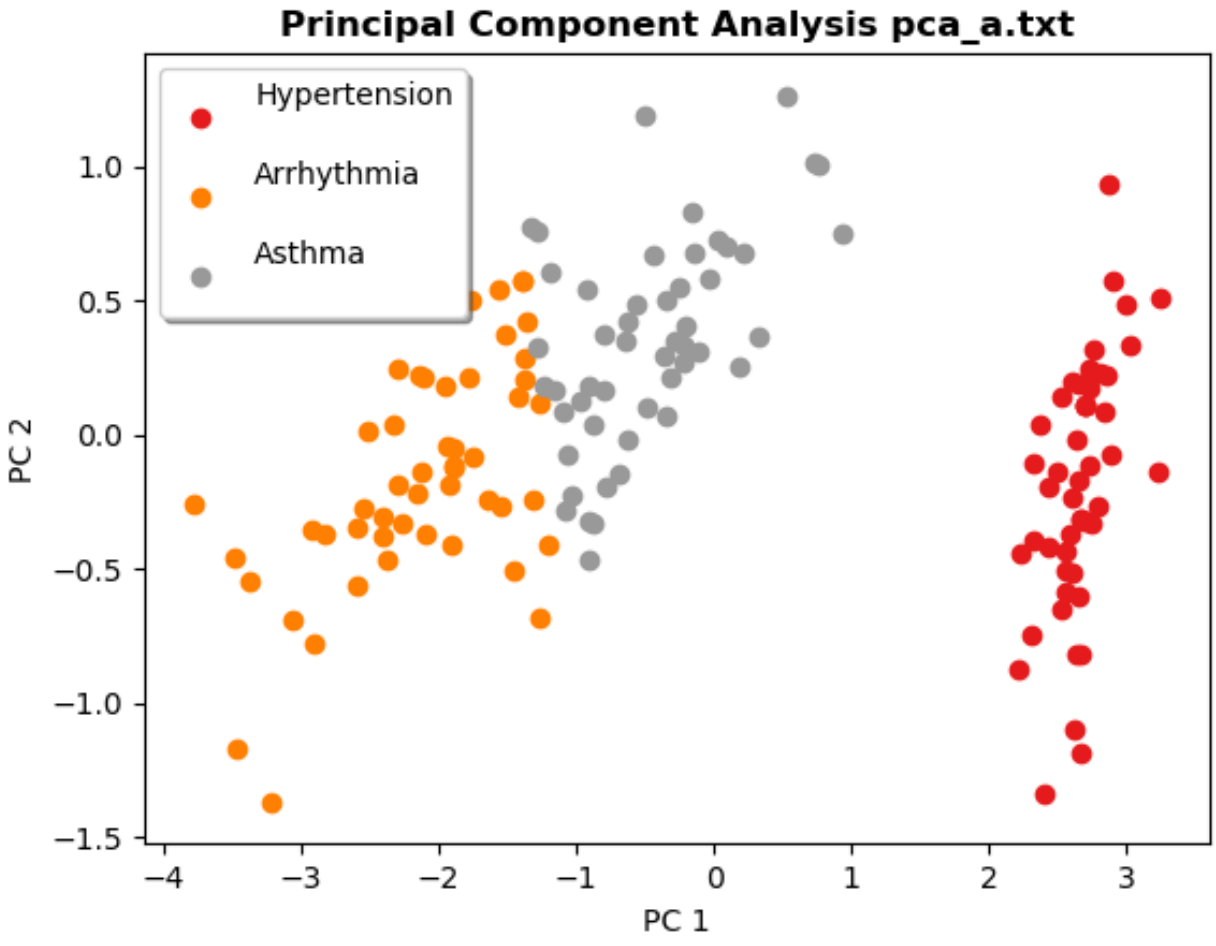
PLOTTING THE PCA

7) The unique diseases present in the file are identified by calling the set() method and this list "disease_unique" will have **unique diseases**.
8) Different numbers of colours are assigned in the beginning according to the unique number of diseases by calling **the cm.Set1().**
9) By looping to the length of unique diseases, the data from the pca_plot_matrix are plotted. Initially all data of same disease are plotted, followed by the next disease and continued till every data is plotted.
10) The plotting is done by using **matplotlib.pyplot**, by setting all the legends and attributes.

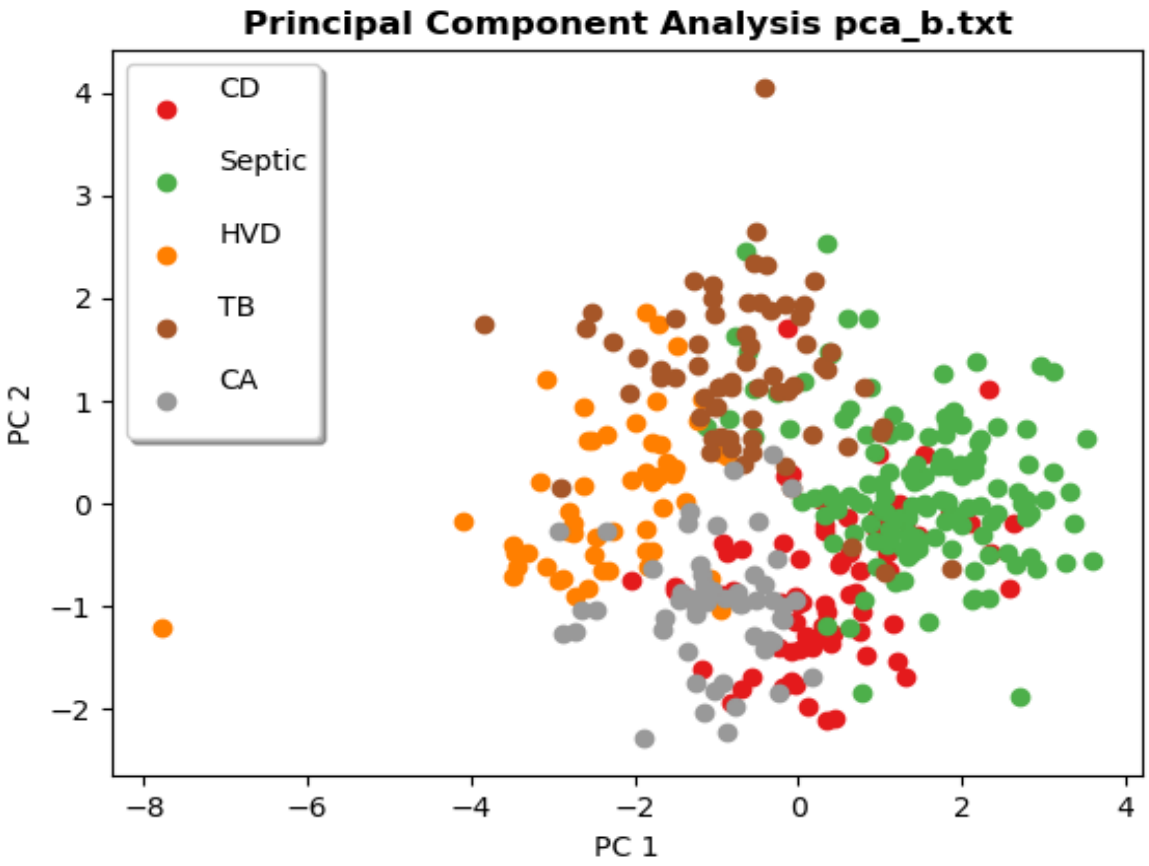
Task 2: Run the PCA algorithm on the three text files and record the results.

RESULTS OBTAINED:

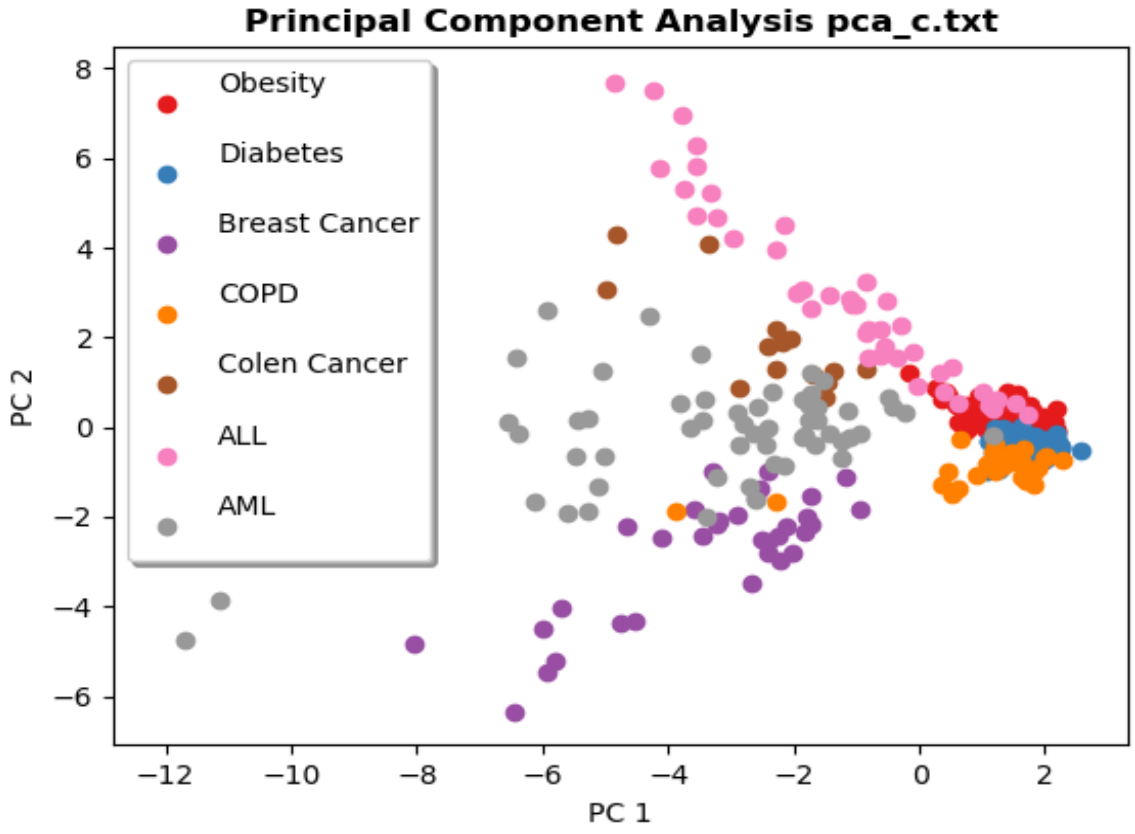
1) **Plotting PCA on Pca_a.txt**



2) Plotting PCA on pca_b.txt



3) Plotting PCA on pca_c.txt



TASK 3: Implement SVD and TNSE using the existing packages and visulaize the data points in the same way on the three files.

Before implementing SVD, we need to know the basics of SVD.

Singular Value Decomposition:
It is another decomposition method which is used for both real and complex matrices. **It generally decomposes a matrix into product of two unitary matrices (U,V*) and a rectangular diagonal matrix of singular values.**

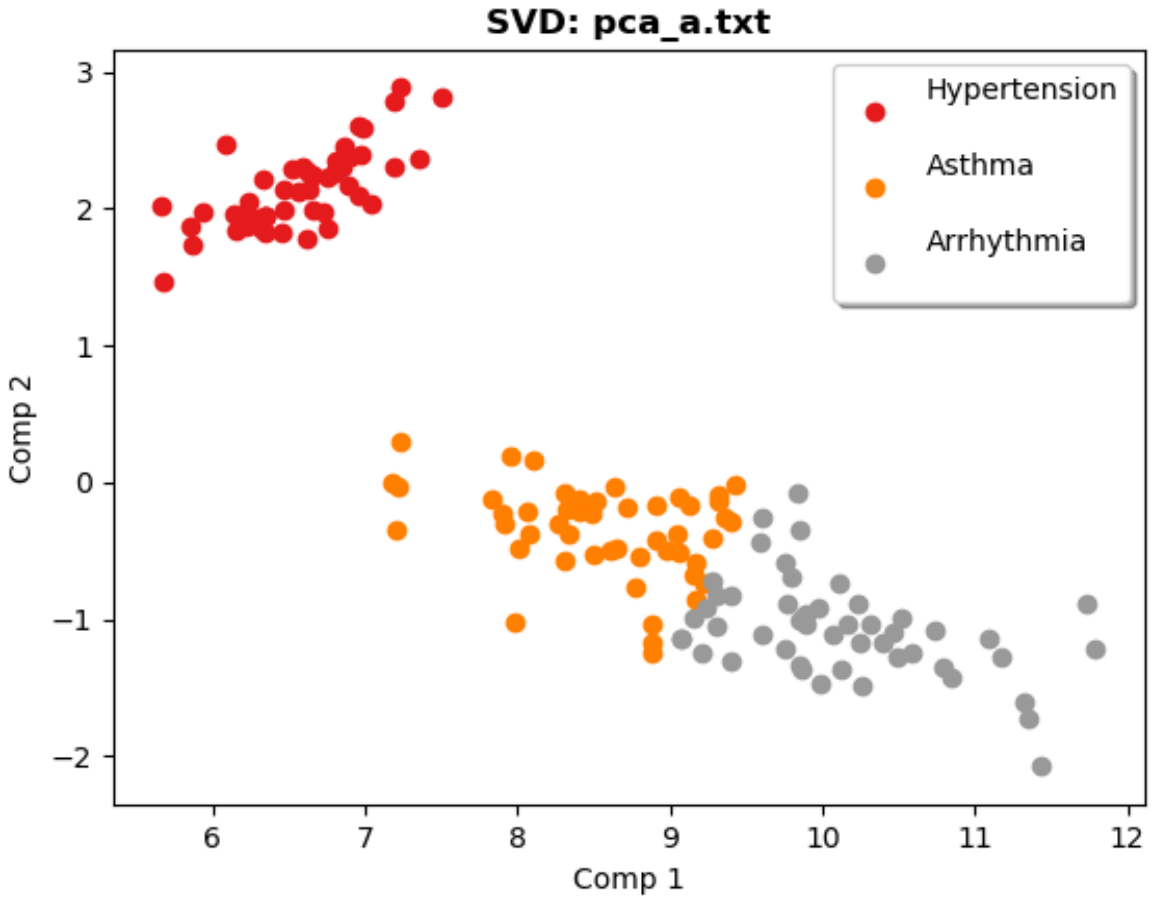
Just like PCA, it is used for variable reduction.
In sum, it is a method of decomposing a matrix into three other matrices.

IMPLEMENTATION:

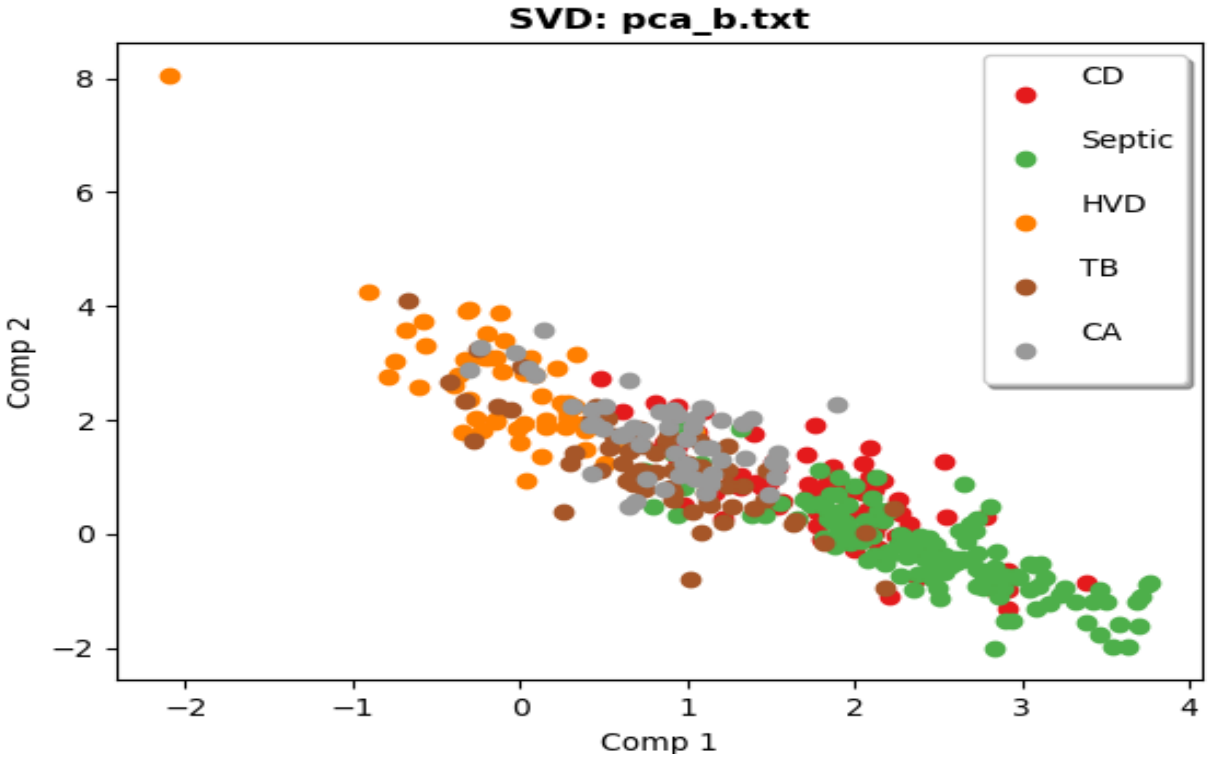
1) The first 5 steps are common for PCA and SVD
2) For implementing SVD we used existing function from sklearn
   TruncatedSVD(**n_iteration**=2,**algorithm**='randomized',**n_iter**=5,tol=0.0,**random_state**=None)
3) The **n_iteration** refers to dimensionality of output data and it should be strictly less than number of features.
   n_iter corresponds to Number of iterations for randomized SVD solver
   tol corresponds to ARPACK. 0 means machine precision.
   random_state corresponds to If int, random_state is the seed used by the random number generator;
4) The plotting method is same as that of the PCA and the results are obtained.

RESULTS OBTAINED FOR SVD on three text files

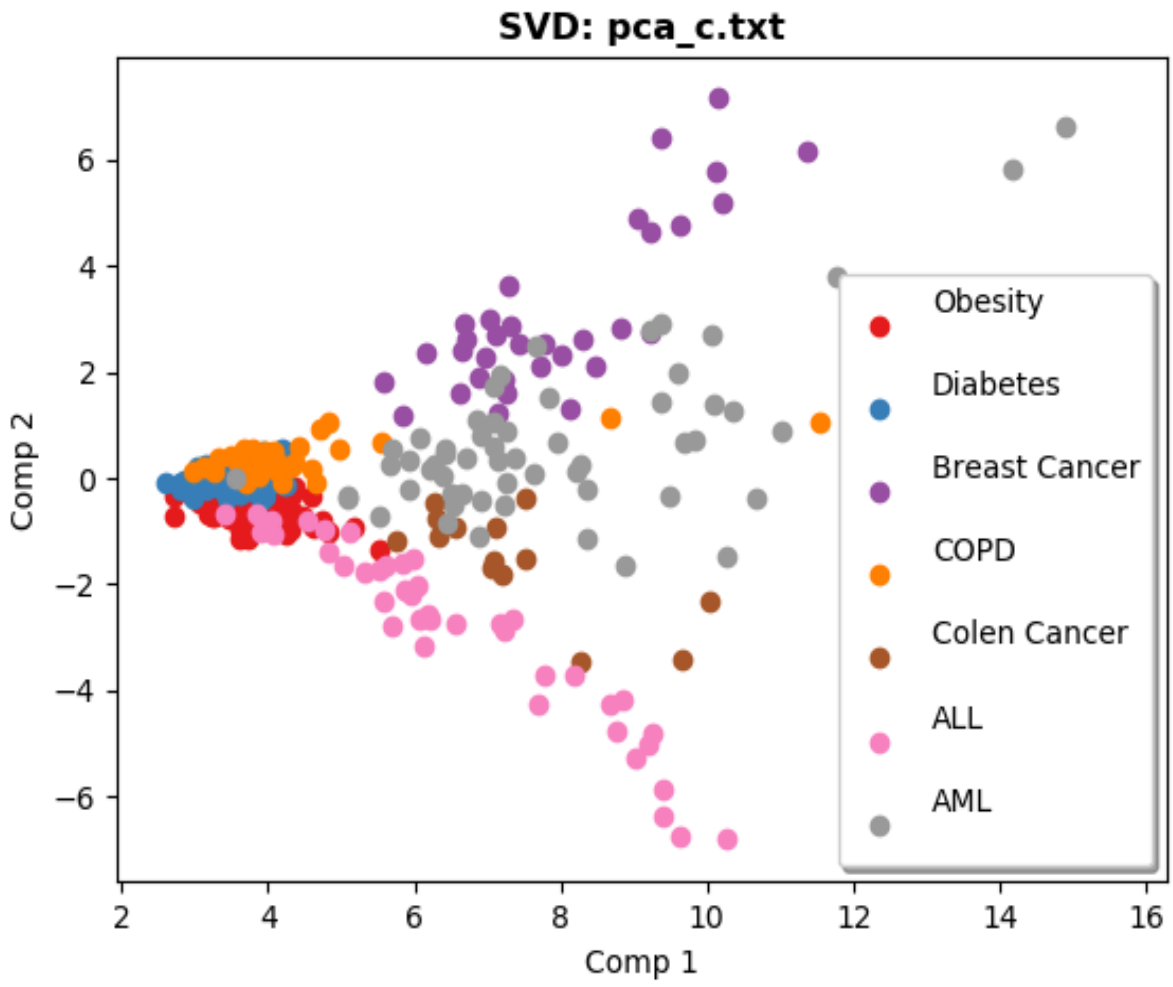
1) **Plotting SVD on Pca_a.txt**



2) **Plotting SVD on Pcb_.txt**



3) **Plotting SVD on pca_c.txt**



COMPARISON OF RESULTS BETWEEN PCA AND SVD:

- **Similar results are obtained for PCA and SVD algorithms.** It is because both the algorithms use a similar technique. They use eigenvalue methods to reduce high dimensional data set into fewer dimensions while retaining the most important information.
- In both the cases **as the dimensionality increases, the data points wouldn't be clearly distinct.**
- **Thus we can actually perform PCA using SVD and vice versa.**

T- SNE:

It is a non-linear technique which again performs dimensionality reduction i.e. reducing a larger set of variables into lesser ones while preserving the most important information. **It is well suited for visualising high dimensional data sets.**

ALGORITHM:

- The **algorithm calculates the probability of similarity of points in high dimensional space** and does the same in low dimensional space. The similarity is calculated as a **conditional probability.**
- After finding the similarity, it tries to minimize the difference between conditional probabilities in high dimensional space and low dimensional space, **to get a perfect representation of data points in low dimensional space.**

- To calculate the minimum sum of difference of conditional probability, t-sne minimizes the sum of **KL divergence of overall data** points using a gradient descent method.

IMPLEMENTATION:

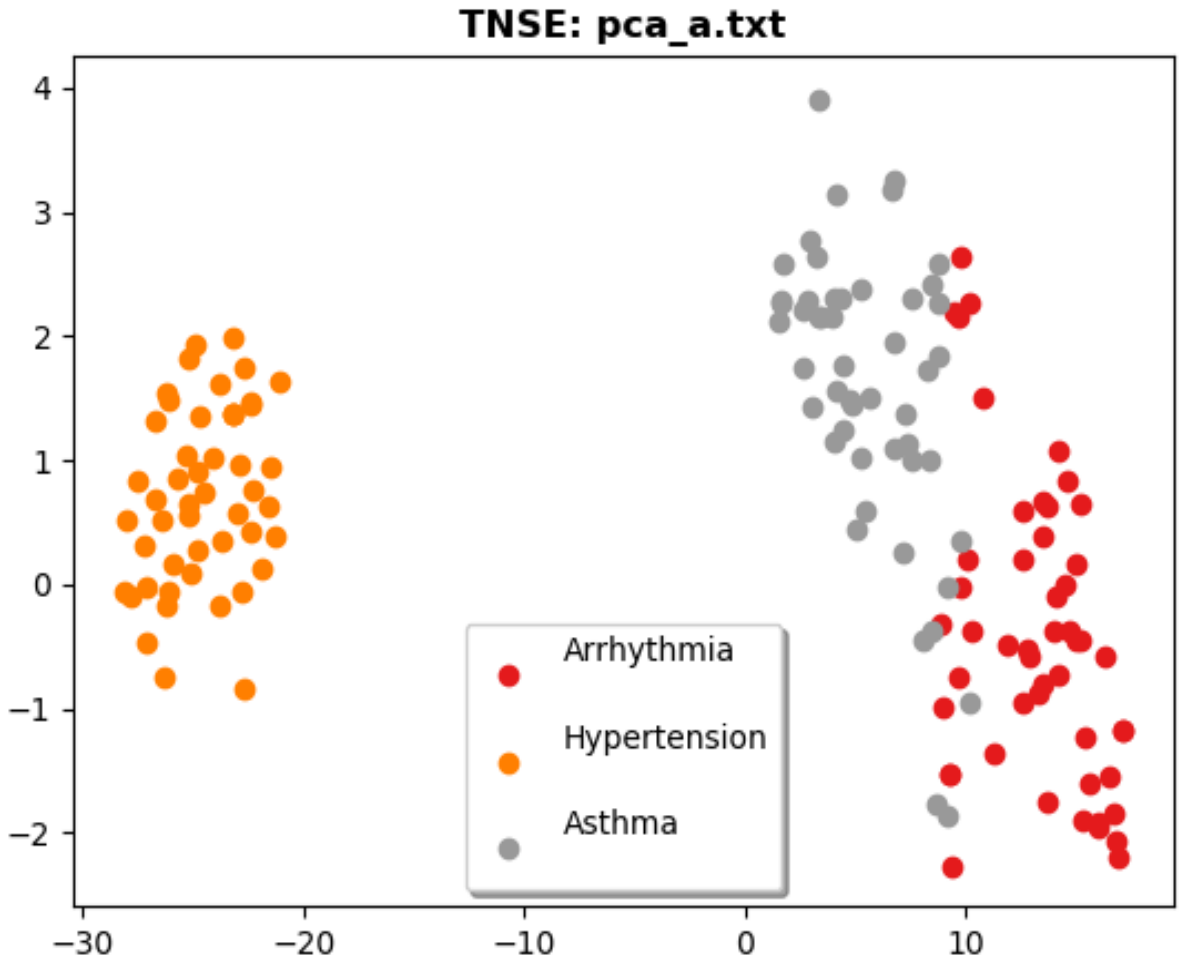
1) The first 5 steps are common as that of PCA.
2) For implementing TNSE, we used an existing method **TNSE imported from sklearn. The** arguments passed to the method **are n_components=2, init=pca and learning rate =100 TNSE(n_components=2,init=pca,learning_rate=100, perplexity:30, n_iter=1000, early_exaggeration=12.0,angle=0.5**

3) These are some of the parameters passed to TNSE and each having its own significance.
   n_components corresponds to dimensionality of output.
   init possible options are pca and random
   n_iter sets the number of iteraitons.
   learning_rate =The learning rate for t-SNE is usually in the range [10.0, 1000.0]. If the learning rate is too high, the data may look like a 'ball' with any point approximately equidistant from its nearest neighbours.

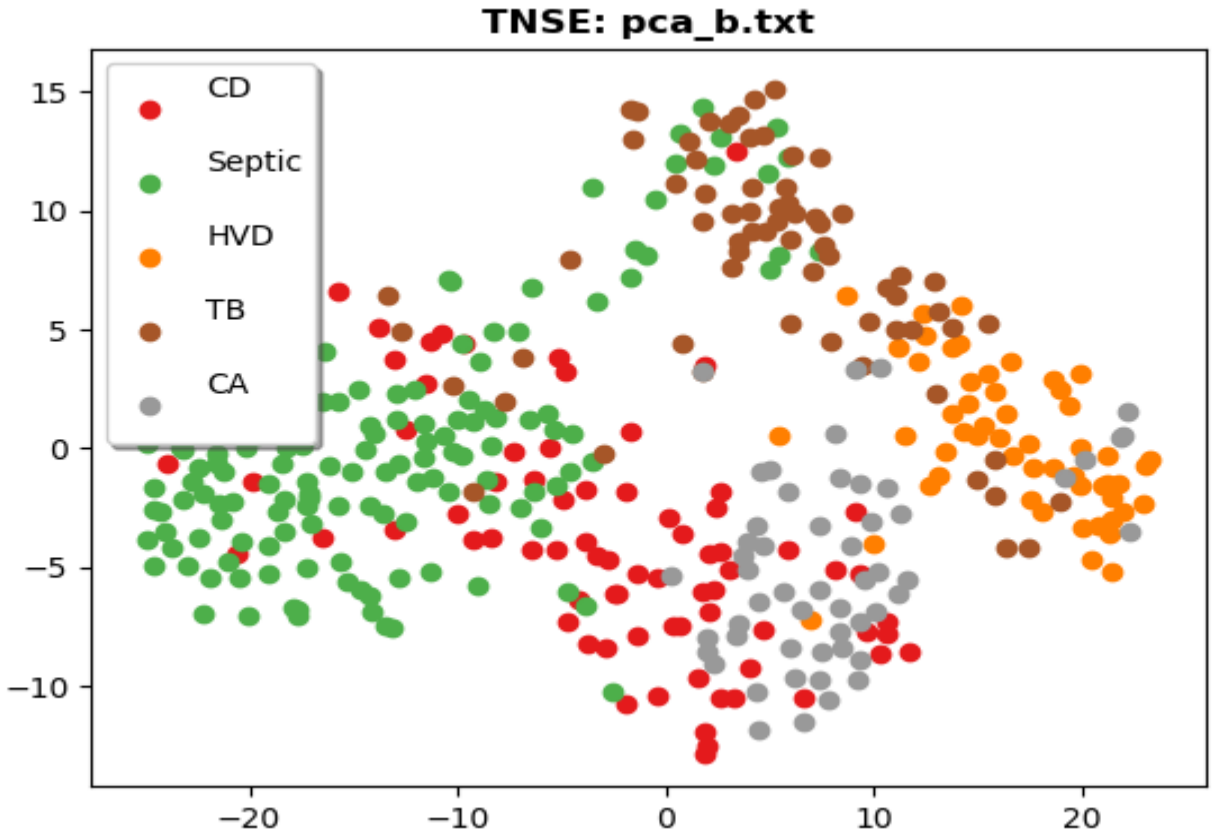
4) The plotting is done just like above and the results are obtained.

RESULTS OBTAINED:

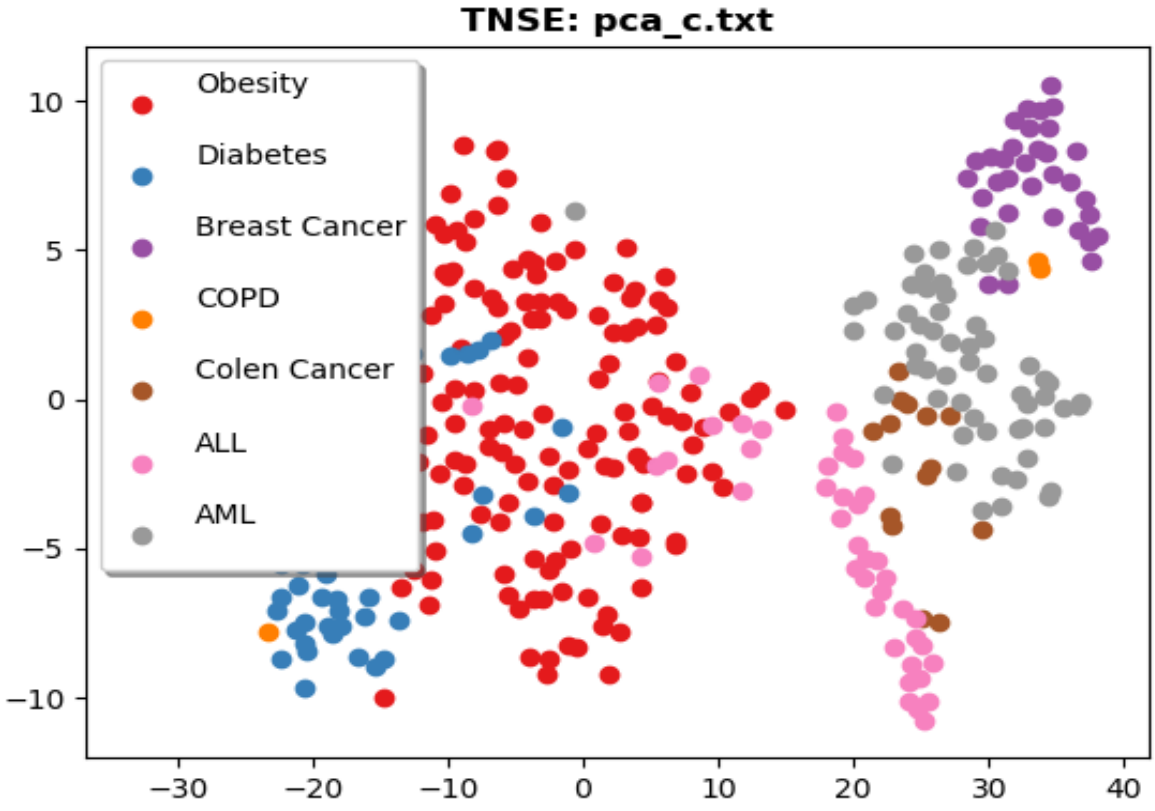
1) Plotting t-sne on pca_a.txt



2) **Plotting t-sne on pca_b.txt**



3) **Plotting t-sne on pca_c.txt**

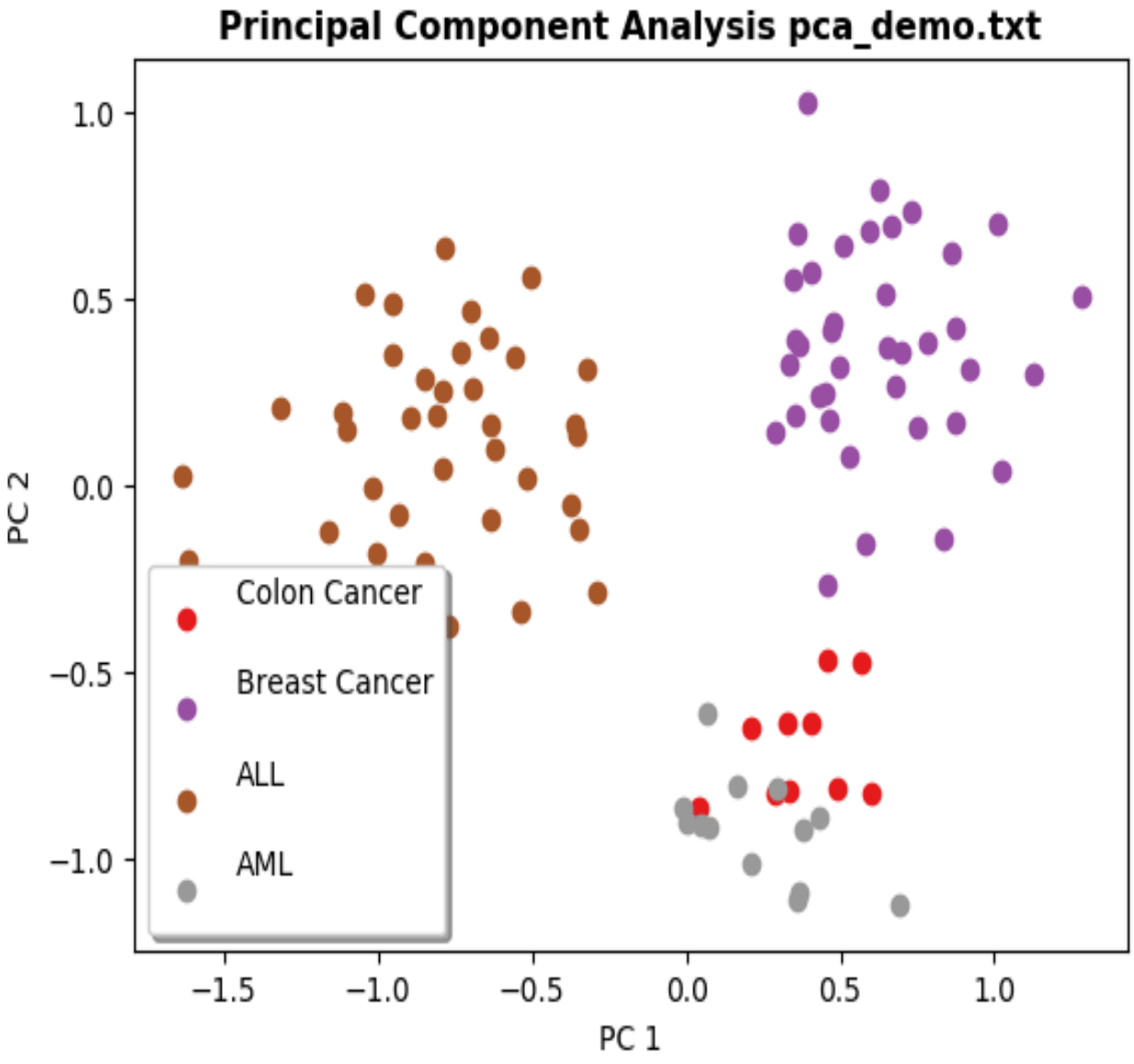


COMPARISON BETWEEN PCA AND T-SNE

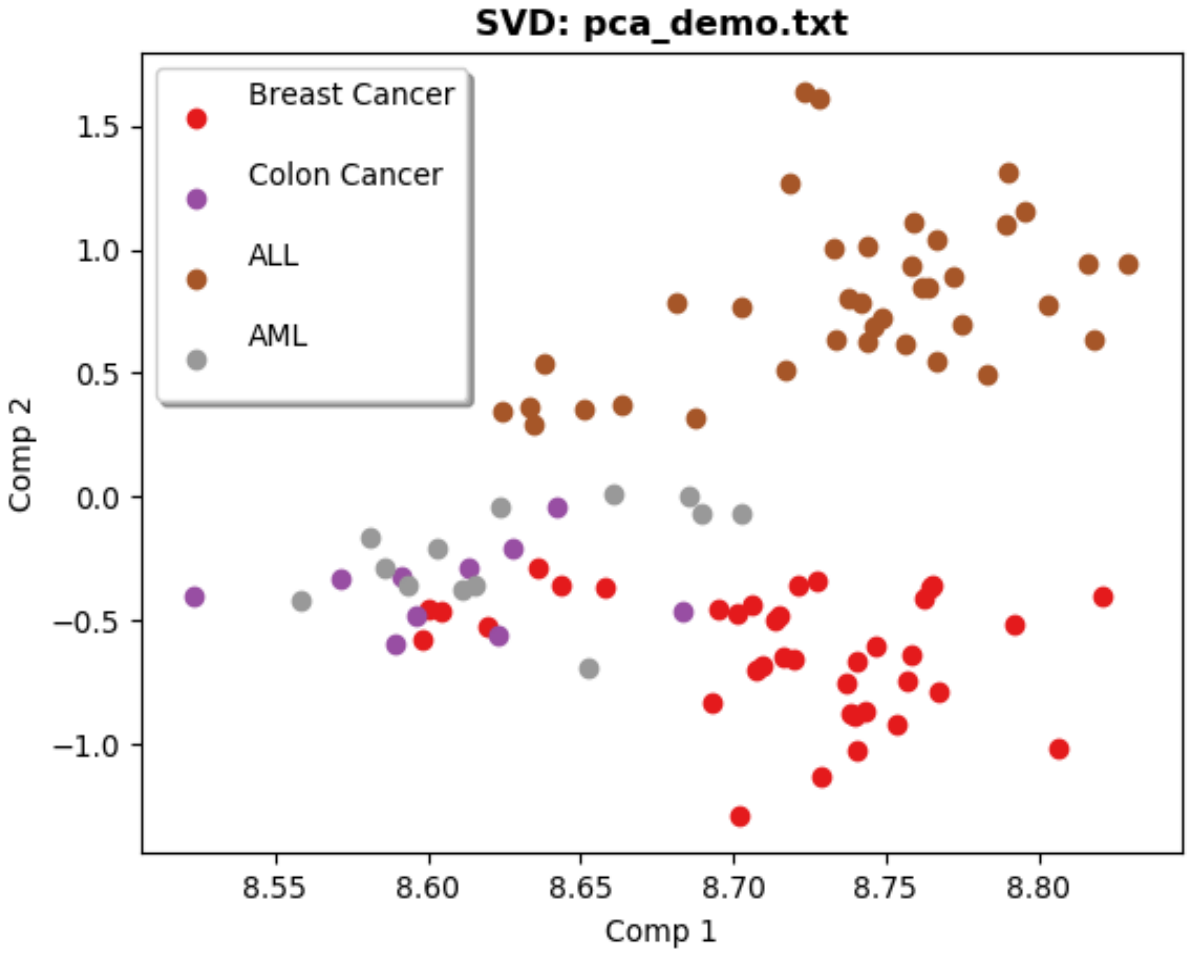
- The main distinguisher between PCA and T-SNE is that, **PCA is a mathematical method for dimensionality reduction whereas, T-SNE is a probabilistic one.**
- PCA is a statistical procedure which uses orthogonal transformation technique to convert a series of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.
- **t-SNE maps high dimensional data to lower dimensions to find patterns in the data by observing clusters based on similarity of data points with multiple features.**
- For very high dimensions of data**, t-Sne tends to give a better visualisation and exploration compared to pca.**
- One of the **disadvantages of t-Sne is that it requires a lot of computations and memory storage.**

RESULTS OBTIANED ON DEMO FILE

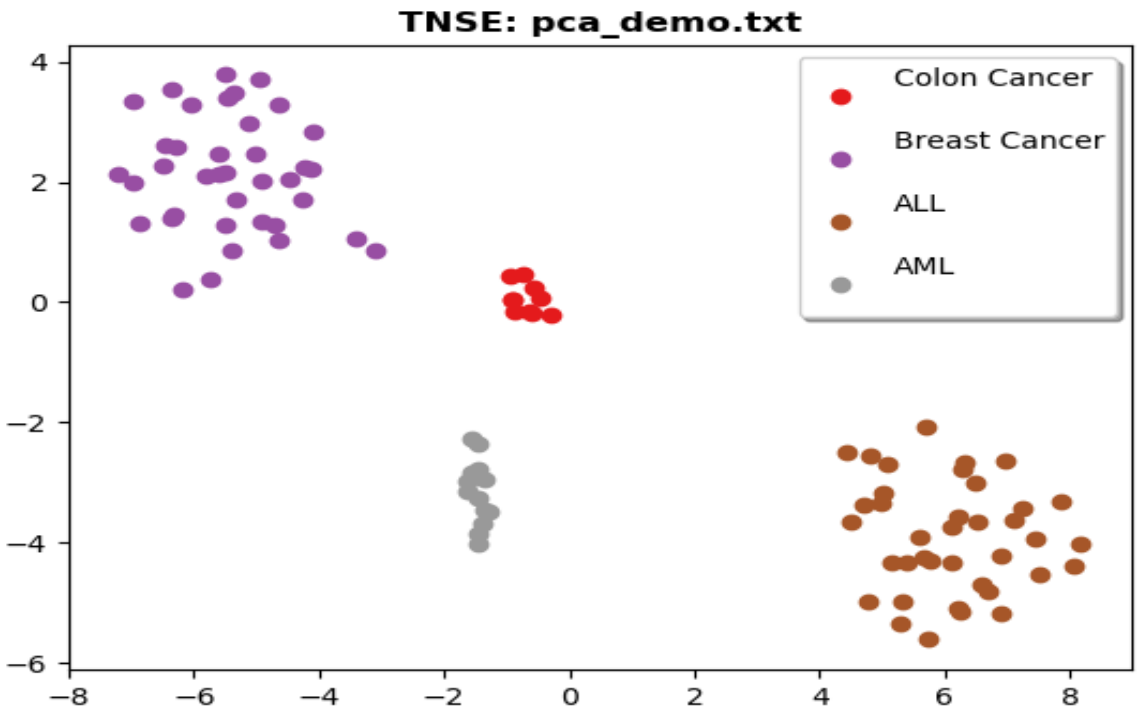
1) **Plotting PCA on demo.txt**



2) Plotting SVD on demo.txt



3) Plotting TNSE on demo.txt



CONCLUSION

- Implemented PCA in python without using any existing packages.
- Implemented SVD and t-Sne using the existing methods from skylearn
- The graphs are plotted and the results are analysed.
- **PCA and SVD tend to perform similar as the methods used are quite same.**
- T-Sne uses a different technique, a probabilistic one to reduce high dimensional data set into lower ones, while preserving the important information.
- **T-sne gives a better visualisation for very high dimensional data sets, example in case of pca_c.txt.**
- **T-sne takes more computation time and memory storage.**

REFERENCES:

1) https://www.datacamp.com/community/tutorials/introduction-t-sne
2) https://towardsdatascience.com/a-step-by-step-explanation-of-principal-component-analysis-b836fb9c97e2
3) https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.svd.html
4) https://scikitlearn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html
5) https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html