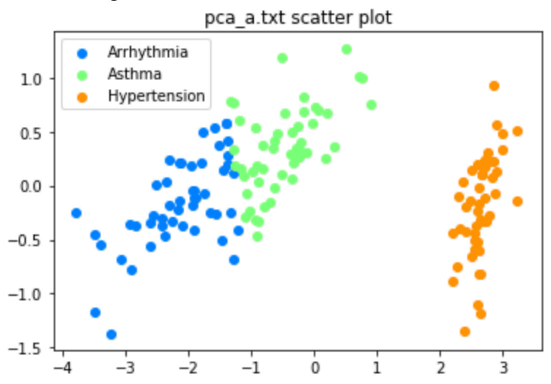
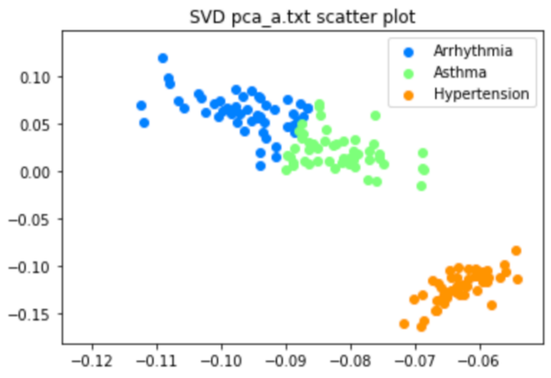
PCA Report

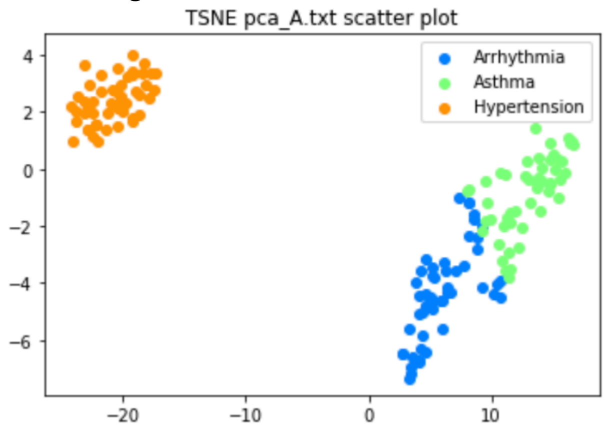
1. Scatter Plots:
    a. Dataset pca_a.txt
        i. PCA Algorithm



        ii. SVD Algorithm

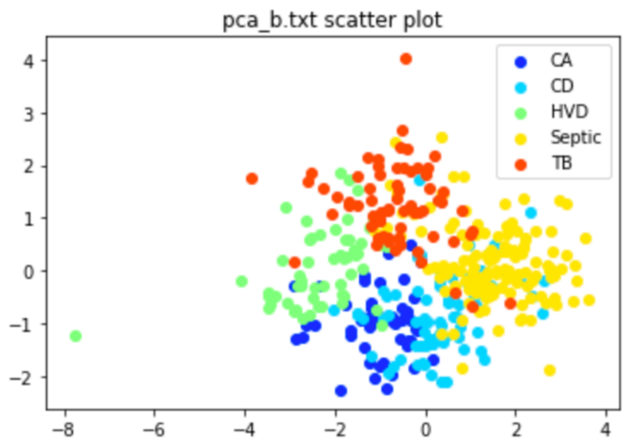


        iii. t-SNE Algorithm

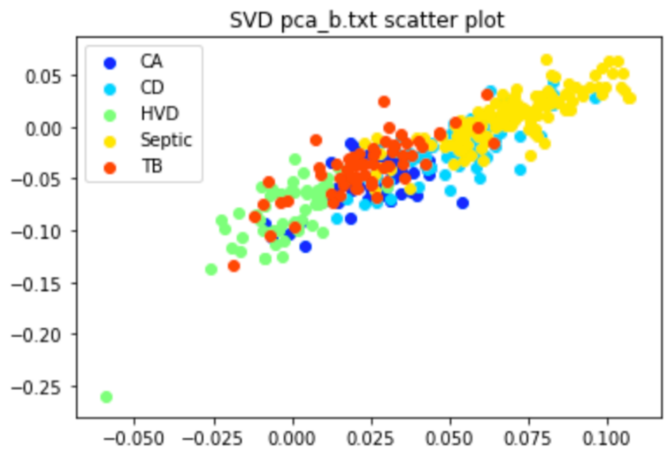


    b. Dataset pca_b.txt

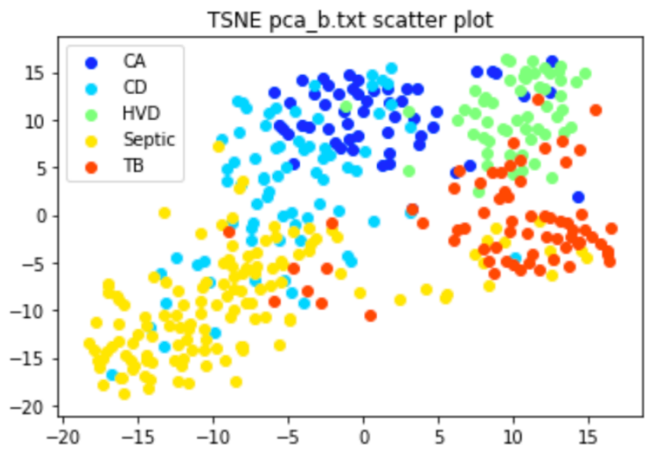
i. PCA Algorithm



ii. SVD Algorithm

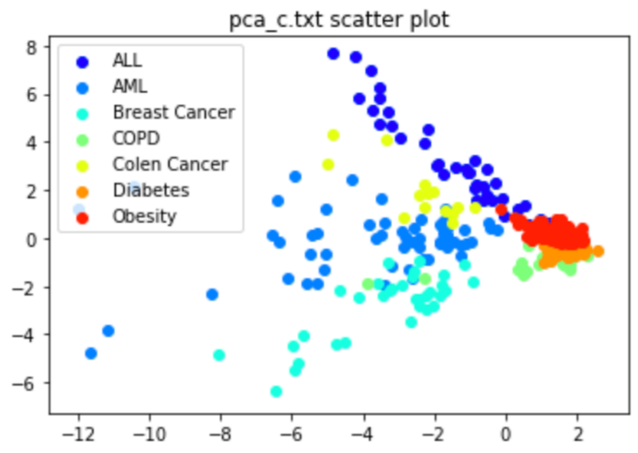


iii. t-SNE Algorithm

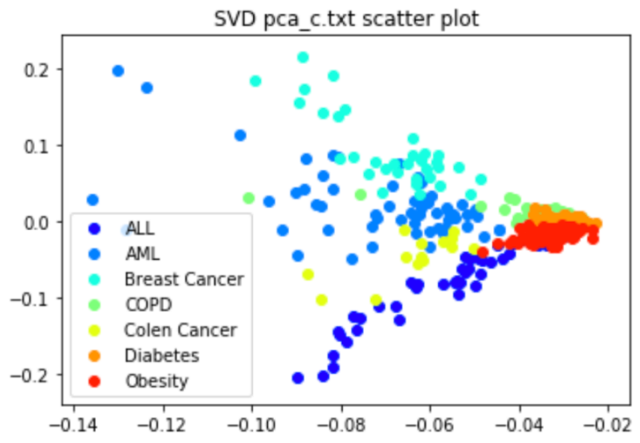


c. Dataset pca_c.txt

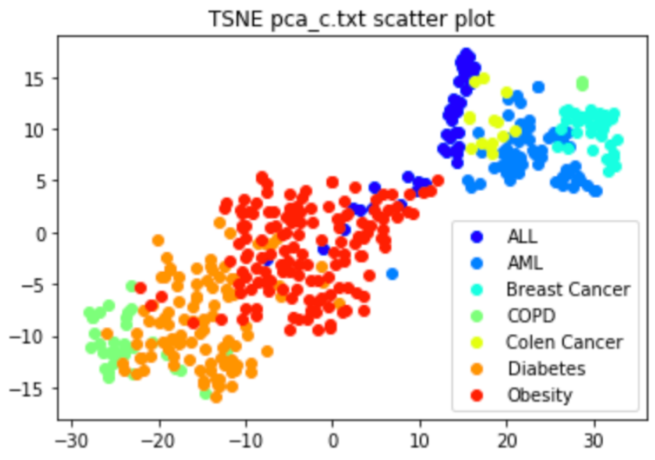
i. PCA Algorithm



ii. SVD Algorithm

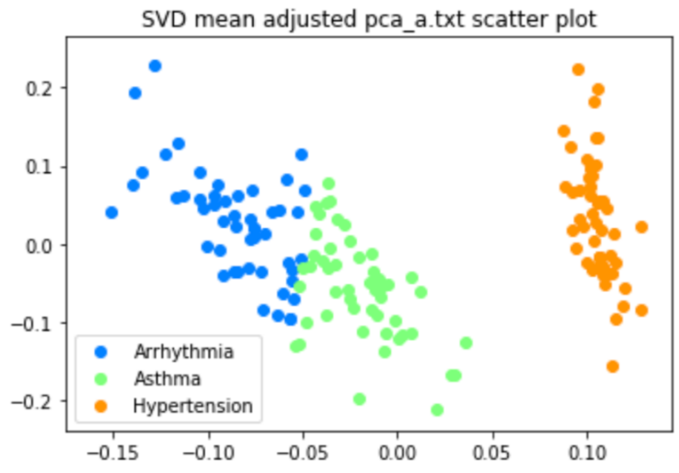


iii. t-SNE Algorithm

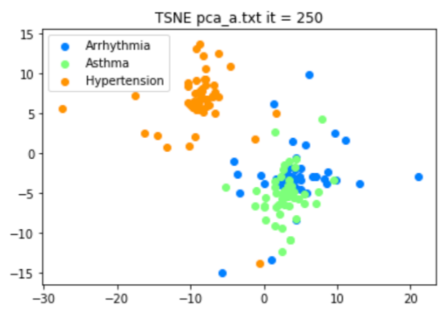


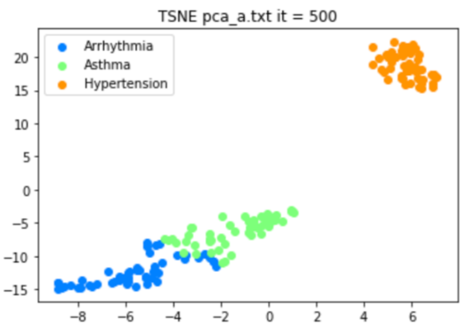
2. PCA implementation:
   1. Extract the features from the text file
      ```
      matrix = np.loadtxt(data_file, delimiter="\t", usecols = range(no_of_columns - 1))
      ```
   2. Calculate mean vector of matrix
      ```
      mean_vector = np.mean(matrix, axis=0)
      ```
   3. Subtract the mean vector from the original feature matrix to get the adjusted matrix
      ```
      adjusted_orig_data = matrix - mean_vector
      ```
   4. Compute the covariance matrix of the adjusted matrix
      ```
      covariance_matrix = np.cov(adjusted_orig_data.T)
      ```
   5. Compute eigen vector and eigen values from the covariance matrix
      ```
      eigen_value, eigen_vector = np.linalg.eig(covariance_matrix)
      ```
   6. Choose 2 eigen vectors with the highest eigen values
   7. Calculate principal components by multiplying the eigen vectors and the feature matrix
      ```
      pca[:,it] = np.dot(adjusted_orig_data, v.T)
      ```

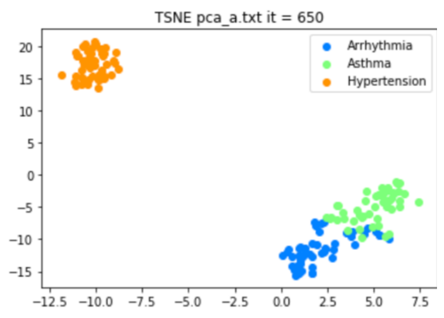
3. Inference
   1. PCA tells how much the features deviates from the mean
   2. SVD tells how much the features deviates from 0. And so, if we apply SVD to the mean adjusted matrix, we will get a similar plot as in PCA



   3. t-SNE uses probability distribution for dimensionality reduction
   4. In the TSNE function, as the number of iterations increases, clustered groups become more defined. We have selected the value 500 with default learning rate as 200, since there was not much difference in the results as the iterations increased







beyond 500