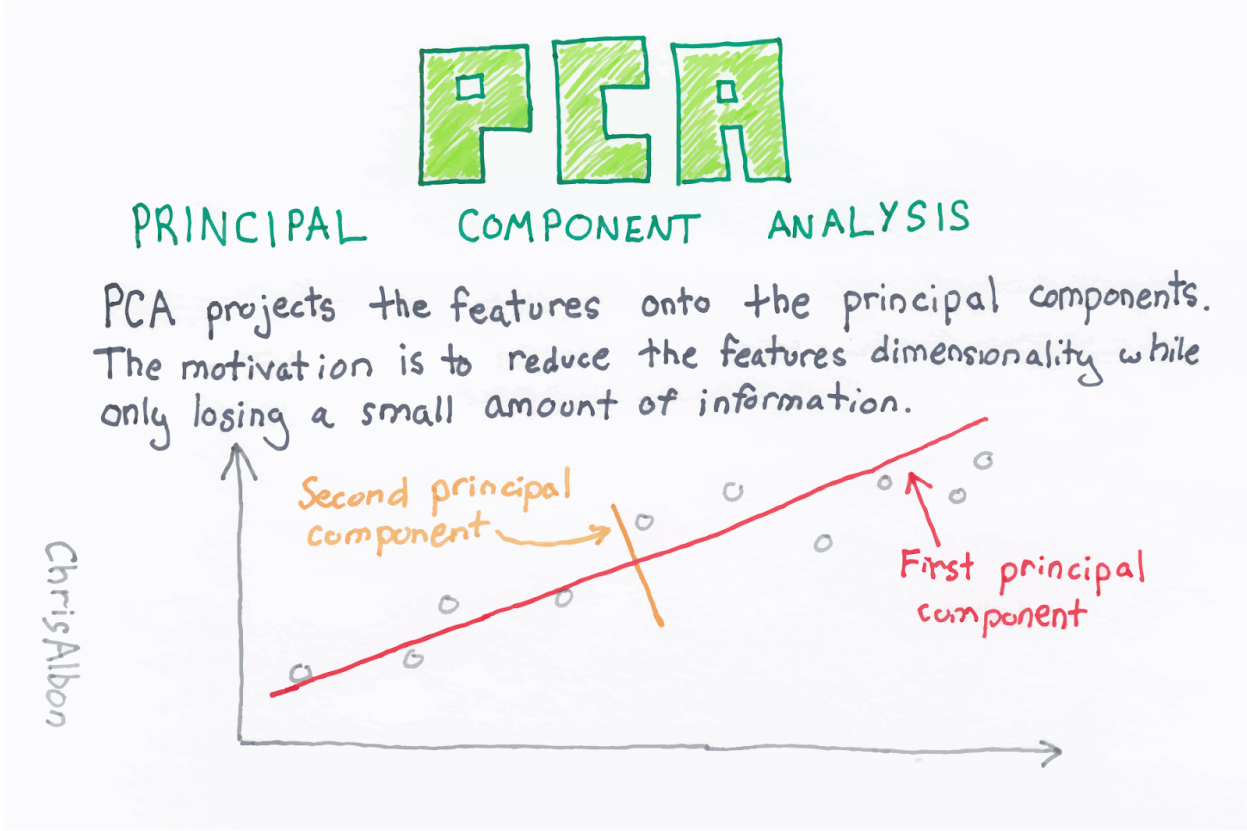
Data Mining

Part 1

Dimensionality Reduction



Index

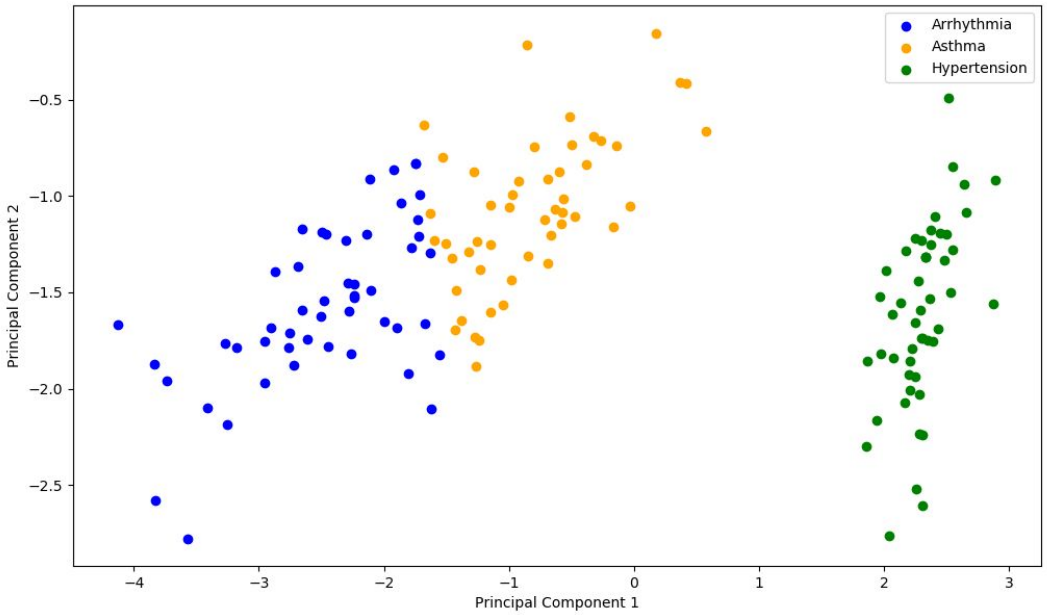
Introduction

The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent.

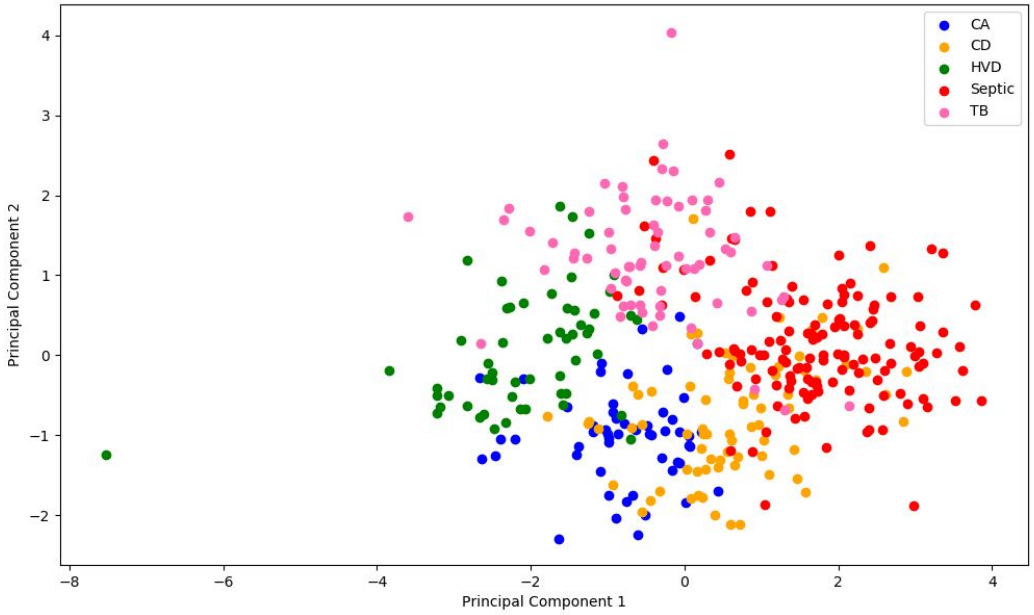
Scatter Plots

The Scatter plots shown below are of PCA using 3 datasets.

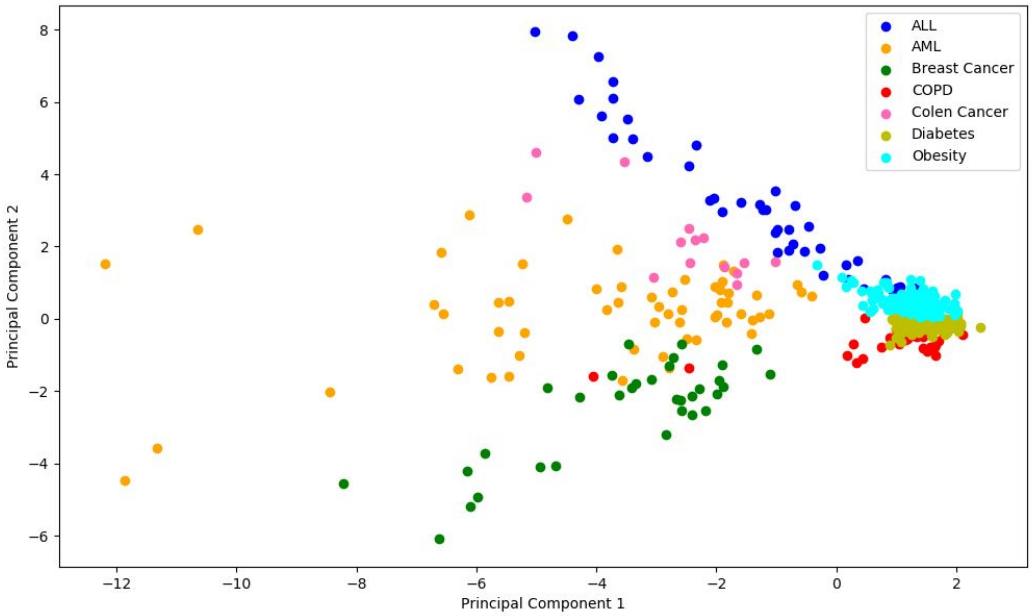
1. a.txt file



2. b.txt file

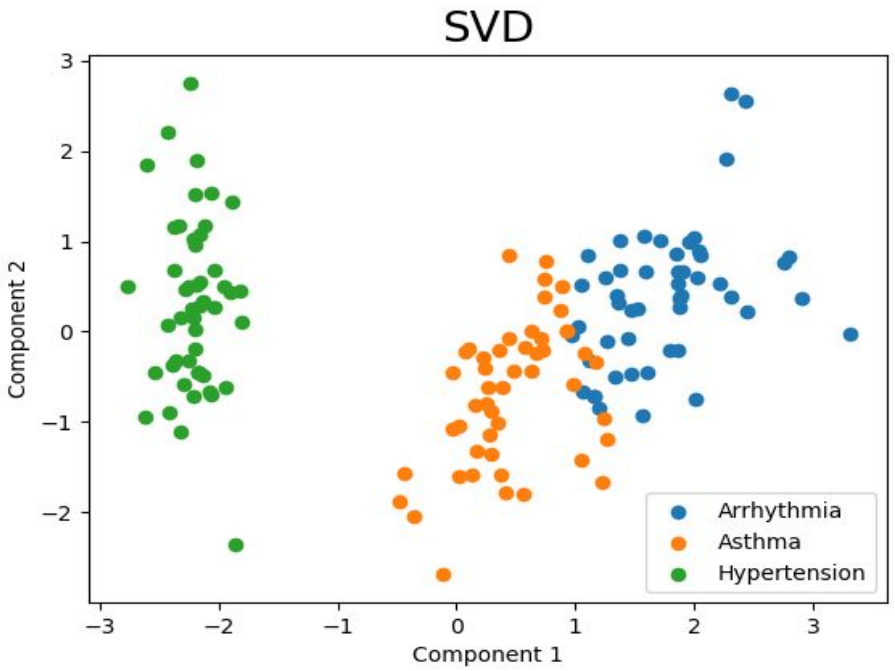


3. c.txt file

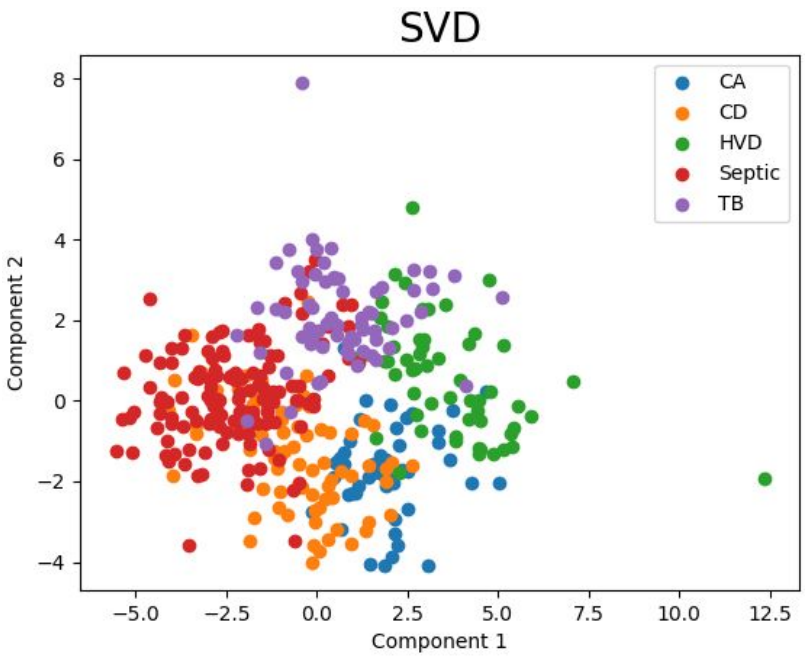


The Scatter plots shown below are of SVD using 3 datasets.

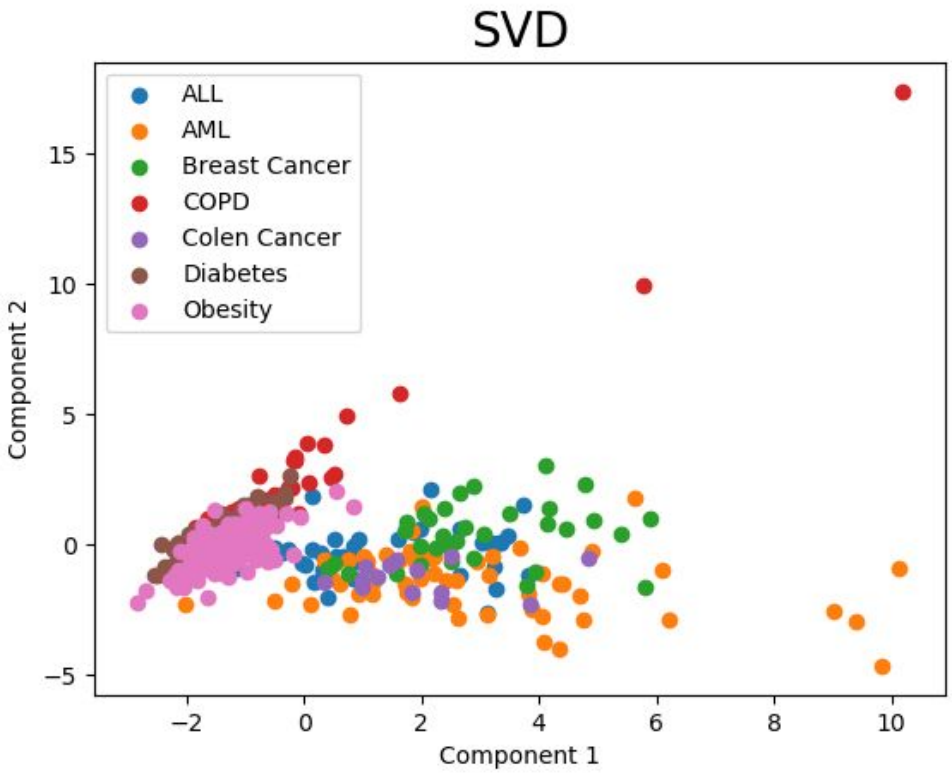
1. a.txt file



2. b.txt file

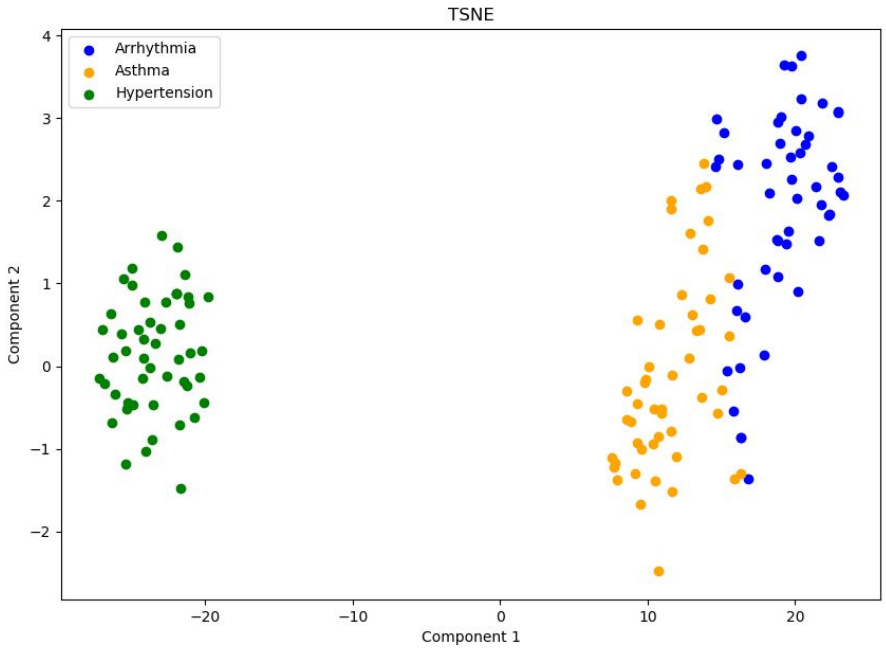


3. c.txt file

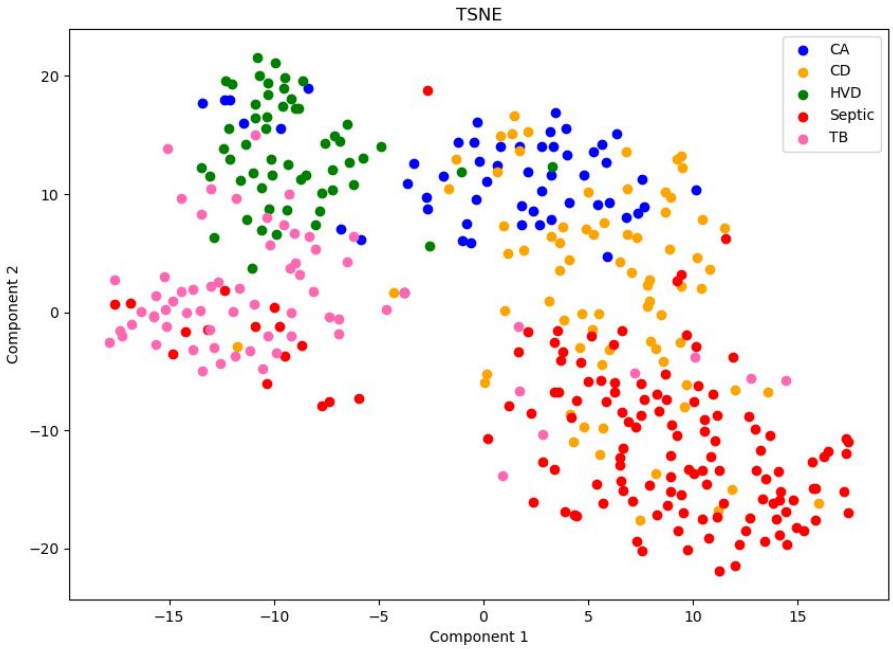


The Scatter plots shown below are of TSNE using 3 datasets.

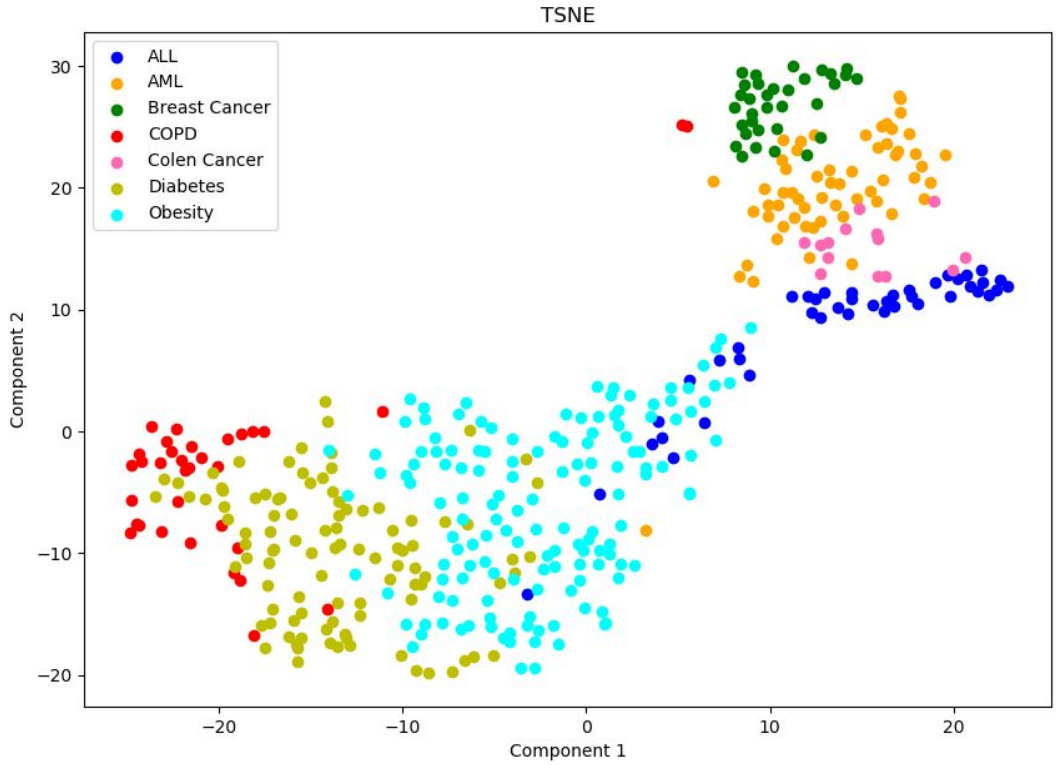
1. a. txt file



2. b.txt file



3. c.txt file


TSNE
ALL
AML
Breast Cancer
COPD
Colen Cancer
Diabetes
Obesity
Component 2
Component 1
30
20
10
0
−10
−20
−20
−10
0
10
20

Brief Explanation of PCA Implementation

1. Load the Data. The pandas dataframe is used to load the data and convert it into a csv file.

Load data

```
df=pd.read_csv("Desktop/c.txt",delimiter="\t")
df.to_csv('a.csv',encoding='utf-8', index=False)
print(df.head(5))
```

```
   0.88  1.16  1.47  1.38  1.61  1.09  0.85  0.97  1.27  1.51  1.12   Obesity
0  1.19  0.71  0.92  1.25  0.98  0.76  1.08  1.50  1.42  1.16  1.24  Diabetes
1  1.11  0.73  1.10  0.98  1.29  0.82  0.91  1.15  1.12  1.18  0.89   Obesity
2  1.48  1.39  5.25  1.94  7.76  1.59  1.11  1.97  1.51  2.73  1.54       ALL
3  1.38  1.05  0.91  1.58  0.82  1.13  1.28  2.01  1.43  0.85  1.44      COPD
4  1.13  1.22  1.13  1.98  1.15  1.22  0.94  1.45  1.55  1.30  1.62  Diabetes
```

2. Get number of rows and columns of the dataframe

```
row,col=df.shape
print("Row: ",row)
print("Col: ",col)
```

```
Row:  427
Col:  12
```

3. Separate the last columns which includes the name of the diseases and get all the disease name listed in the column. Then get the unique disease names from the last column.

Seperate last column (disease)

```
X = df.iloc[:,0:col-1].values
y = df.iloc[:,col-1].values
```

```
a=np.unique(y)
l=len(a)
print("unique no. of values in last column: ",l)
print(a)
```

```
unique no. of values in last column:  7
['ALL' 'AML' 'Breast Cancer' 'COPD' 'Colen Cancer' 'Diabetes' 'Obesity']
```

4. Standardize the data. Standardization rescales data to have a mean (μ) of 0 and standard deviation (σ) of 1 (unit variance). All the values in the matrix except the last column is standardized.

Adjust the original data with mean

```
X_std = X-X.mean()
cov_mat = np.cov(X_std.T)
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
```

5. Get Eigen-values and Eigen-vectors from the covariance matrix. In the code, the eigen-value of the first element and it's vector can be seen in the output.

Covariance Matrix and Eigen value and vector

```
cov_mat = np.cov(X_std.T)

eig_vals, eig_vecs = np.linalg.eig(cov_mat)

print(eig_vals[0])
print(eig_vecs[0])
```

```
4.987893022664105
[ 0.36615052  0.03626421 -0.38200982 -0.06348133  0.00716644 -0.28042667
  0.40229376 -0.08021667 -0.37523384 -0.55596987 -0.1333752 ]
```

6. Create Eigen pairs. After generating the pairs, reverse the list order so that highest eigen-value is stored first.

Create Eigen Pairs

```
# Make a list of (eigenvalue, eigenvector) tuples
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]

# Sort the (eigenvalue, eigenvector) tuples from high to low
eig_pairs.sort(key=lambda tup:tup[0])
eig_pairs.reverse()
```

7. Reduce the dimension. Reduce the multi-dimensional feature space to a 2-dimensional feature subspace, by choosing the "top 2" eigenvectors with the

highest eigenvalue. Numpy's "np.hstack" stacks the two eigen-vectors on top of first vector.

Dimensionality Reduction

```
matrix_w = np.hstack((eig_pairs[0][1].reshape(col-1,1),
                      eig_pairs[1][1].reshape(col-1,1)))

print('Matrix W:\n', matrix_w)
```

```
Matrix W:
 [[ 0.36615052  0.03626421]
 [ 0.38018128  0.12531359]
 [ 0.33299565  0.22796499]
 [ 0.14018047 -0.42709854]
 [ 0.26759055  0.17046331]
 [ 0.3823751   0.18850369]
 [ 0.33349487 -0.04885194]
 [ 0.16494132 -0.51254831]
 [ 0.19085099 -0.50350748]
 [ 0.36418155  0.23807446]
 [ 0.2560608  -0.3297231 ]]
```

8. Plot the Scatter Plot.

Projection Onto the New Feature Space

```
Y = X_std.dot(matrix_w)
print(Y.shape)
```

```
(427, 2)
```

```
colors = ['blue', 'orange', 'green', 'red', 'hotpink','y', 'cyan', 'purple', 'pink','yellow', 'gold',
        'lightcoral', 'salmon', 'yellowgreen', 'greenyellow', 'olive','gray', 'navy']
```

```
plt.figure(figsize=(12,7))

for c, i, target_name in zip(colors, a, a):
    pl.scatter(Y[y==i,0], Y[y==i,1], c=c, label=target_name)

pl.xlabel('Principal Component 1')
pl.ylabel('Principal Component 2')
pl.legend()
pl.title('PCA of a.txt')
pl.show()
```

The results of the code is displayed in the above section under PCA Scatter plots.

Brief Explanation of SVD Implementation

1. Load the Data. The pandas dataframe is used to load the data and convert it into a csv file.

Load data

```
df=pd.read_csv("Desktop/c.txt",delimiter="\t")
df.to_csv('a.csv',encoding='utf-8', index=False)
print(df.head(5))
```

```
   0.88  1.16  1.47  1.38  1.61  1.09  0.85  0.97  1.27  1.51  1.12   Obesity
0  1.19  0.71  0.92  1.25  0.98  0.76  1.08  1.50  1.42  1.16  1.24  Diabetes
1  1.11  0.73  1.10  0.98  1.29  0.82  0.91  1.15  1.12  1.18  0.89   Obesity
2  1.48  1.39  5.25  1.94  7.76  1.59  1.11  1.97  1.51  2.73  1.54       ALL
3  1.38  1.05  0.91  1.58  0.82  1.13  1.28  2.01  1.43  0.85  1.44      COPD
4  1.13  1.22  1.13  1.98  1.15  1.22  0.94  1.45  1.55  1.30  1.62  Diabetes
```

2. Get number of rows and columns of the dataframe

```
row,col=df.shape
print("Row: ",row)
print("Col: ",col)
```

```
Row:  427
Col:  12
```

3. Separate the last columns which includes the name of the diseases and get all the disease name listed in the column. Then get the unique disease names from the last column.

Seperate last column (disease)

```
X = df.iloc[:,0:col-1].values
y = df.iloc[:,col-1].values
```

```
a=np.unique(y)
l=len(a)
print("unique no. of values in last column: ",l)
print(a)
```

```
unique no. of values in last column:  7
['ALL' 'AML' 'Breast Cancer' 'COPD' 'Colen Cancer' 'Diabetes' 'Obesity']
```

4. Standardize the data and call SVD function.
    a. Standardization rescales data to have a mean (μ) of 0 and standard deviation (σ) of 1 (unit variance).
    b. All the values in the matrix except the last column is standardized. The standardized value of the first row is seen in the output.

Standardization

```
X_std = StandardScaler().fit_transform(X)
svd = TruncatedSVD(n_components=2)
Y_fitted = svd.fit_transform(X_std)
```

5. Plot the Scatter Plot.

Projection onto new feature space

```
colors = ['blue', 'orange', 'green', 'red', 'hotpink','y', 'cyan', 'purple', 'pink','yellow', 'gold',
        'lightcoral', 'salmon', 'yellowgreen', 'greenyellow', 'olive','gray', 'navy']

for labels, columns in zip(a,colors):
    plt.scatter(Y_fitted[y==labels, 0],Y_fitted[y==labels, 1],label=labels)

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(loc='best')
plt.title("SVD", fontsize=20)
plt.show()
```

Brief Explanation of t-SNE Implementation

1. Load the Data. The pandas dataframe is used to load the data and convert it into a csv file.

Load data

```
df=pd.read_csv("Desktop/c.txt",delimiter="\t")
df.to_csv('a.csv',encoding='utf-8', index=False)
print(df.head(5))
```

```
   0.88  1.16  1.47  1.38  1.61  1.09  0.85  0.97  1.27  1.51  1.12   Obesity
0  1.19  0.71  0.92  1.25  0.98  0.76  1.08  1.50  1.42  1.16  1.24  Diabetes
1  1.11  0.73  1.10  0.98  1.29  0.82  0.91  1.15  1.12  1.18  0.89   Obesity
2  1.48  1.39  5.25  1.94  7.76  1.59  1.11  1.97  1.51  2.73  1.54       ALL
3  1.38  1.05  0.91  1.58  0.82  1.13  1.28  2.01  1.43  0.85  1.44      COPD
4  1.13  1.22  1.13  1.98  1.15  1.22  0.94  1.45  1.55  1.30  1.62  Diabetes
```

2. Get number of rows and columns of the dataframe.

```
row,col=df.shape
print("Row: ",row)
print("Col: ",col)
```

```
Row:  427
Col:  12
```

3. Separate the last columns which includes the name of the diseases and get all the disease name listed in the column. Then get the unique disease names from the last column.

Seperate last column (disease)

```
X = df.iloc[:,0:col-1].values
y = df.iloc[:,col-1].values
```

```
a=np.unique(y)
l=len(a)
print("unique no. of values in last column: ",l)
print(a)
```

```
unique no. of values in last column:  7
['ALL' 'AML' 'Breast Cancer' 'COPD' 'Colen Cancer' 'Diabetes' 'Obesity']
```

4. Call t-sne function.
    a. t-Distributed Stochastic Neighbor Embedding (t-SNE) is an unsupervised, non-linear technique primarily used for data exploration and visualizing high-dimensional data.
    b. t-SNE differs from PCA by preserving only small pairwise distances or local similarities whereas PCA is concerned with preserving large pairwise distances to maximize variance.

```
tsne = TSNE(n_components=2, random_state=0)
X_2d = tsne.fit_transform(X)
X_2d.shape
```

5. Plot the scatter plot

```
colors = ['blue', 'orange', 'green', 'red', 'hotpink','y', 'cyan', 'purple', 'pink','yellow', 'gold',
        'lightcoral', 'salmon', 'yellowgreen', 'greenyellow', 'olive','gray', 'navy']

plt.figure(figsize=(12,7))

for c, i, target_name in zip( colors, a, a):
    pl.scatter(X_2d[y==i,0], X_2d[y==i,1], c=c, label=target_name)

pl.xlabel('Principal Component 1')
pl.ylabel('Principal Component 2')
pl.legend()
pl.title('TSNE of a.txt')
pl.show()
```

References

1. PCA
   http://www.sthda.com/english/wiki/print.php?id=206

2. SVD
   https://machinelearningmastery.com/singular-value-decomposition-for-machine-learning/

3. T-sne
   https://www.kdnuggets.com/2018/08/introduction-t-sne-python.html