The AI-ML and Data Science Club of IITH

Dimensionality Reduction

Contents

Compiled by : Srinith

Dimensionality reduction is a set of techniques used to reduce the number of features or variables in a dataset while preserving important information. It is commonly employed in machine learning, data analysis, and data visualization tasks. Here are four popular dimensionality reduction techniques:

1 T-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is an unsupervised, non-linear technique primarily used for data exploration and visualizing high-dimensional data.t-SNE gives you a feel or intuition of how the data is arranged in a high-dimensional space.It is usually only used for visualization of high-dimensional data and not for clustering.

1.1 Mathematical Formulation

Let our data set $\mathcal{D}$ contain the data points $\{x_1, \cdots, x_m\}$. It starts by converting the high dimensional Euclidean distance between the data points into conditional probabilities that represents similarities.The t-SNE algorithm calculates a similarity measure between pairs of instances in the high dimensional space and in the low dimensional space.Then it optimizes the similarity values using the cost function.

The similarity of high-dimensional datapoint x_j to datapoint x_i is the conditional probability, $p_{j|i}$, that xi would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at xi.As we are only interested in pairwise similarity hence $p_{i|i}$ is consider to be 0. Mathematically,

$$p_{j|i} = \frac{\exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(\frac{-\|x_i - x_k\|^2}{2\sigma_i^2}\right)}$$

σ_i is the variance of the Gaussian Kernel which is an user defined value. It is calculated based on desired perplexity ($\mathcal{P}$) . Perplexity is defined as,

$$\mathcal{P} = 2^{\mathcal{H}}$$
$$\mathcal{H} = -\sum_{j \neq i} p_{j|i} \log_2 p_{j|i}$$

It is a user given value which ranges typically between 5 and 50.

The similarity of x_j with respect to x_i is not necessarily equal to that of x_i with respect to x_j. Thus, we choose the similarity of x_i and x_j as,

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2m}$$

to be the average of the two.

For the low-dimensional counterparts y_i and y_j of the high-dimensional datapoints x_i and x_j , it is possible to compute a similar conditional probability, which we denote by $qj|i$. In T-SNE it is taken to be,

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq j}(1 + \|y_k - y_l\|^2)^{-1}} = \frac{w_{ij}}{Z}$$

We first construct a similarity matrix containing the pairwise similarity scores for each pair of points in the original dataset say of size (mxn) , where m is the number of points and n is the number of input dimensions. We then randomly project the input data points into d-dimensional space and construct a similarity matrix for this as well of size (mxd). For each point in this lower-dimensional space, we move it such that the row corresponding to it in the new similarity matrix becomes like the corresponding row in the original similarity matrix. Finally, we are left with the appropriate clusters that visualize the original dataset.

This is achieved by minimizing the cost function, C, which is defined as the sum of Kullback-Leibler divergences over all datapoints using gradient descent.

$$C = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

The minimization of the cost function is performed using a gradient descent method. The gradient has a surprisingly simple form

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}$$

The gradient update is given by ,

$$\gamma^t = \gamma^{t-1} + \eta \frac{\delta C}{\delta y_i} + \alpha(t)(\gamma^{t-1} - \gamma^{t-2})$$

where γ^t indicates the solution at iteration t, η indicates the learning rate, and $\alpha(t)$ represents the momentum at iteration t.

1.2 Questions

1. What is the major difference between SNE and t-SNE algorithms?
2. What is the value of $p_{i|i}$ taken?
3. Why is t-kernel used over Gaussian kernel in the low dimension?
4. Why is the disadvantage of using t-SNE?
5. What is Kullback-Leibler divergence?

2 PCA

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set. It is often used for exploratory data analysis.

2.1 Mathematical Formulation

Suppose we are given m samples $x_1, x_2, ..., x_m$ where x_i belongs to R^n. And we want to project the data to k dimensional data , where k<n. Therefore we have to project data onto k vectors say $u_1, u_2, ..., u_k$ such that it represents maximum variance of data.

Prior to running PCA, we first pre-process the data to zero out the mean of the data and normalize the variance of data.

$$\mu = \sum_{i=1}^{m} x^{(i)}$$
$$\sigma_j^2 = \frac{1}{m} \sum_i x_j^{(i)}$$
$$x_j^{(i)} = \frac{x_j^{(i)} - \mu}{\sigma_j}$$

Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (i.e., principal components) are uncorrelated i.e, there is no linear relationship between them .i.e , The n principal components are a sequence of n mutually perpendicular lines passing through centroid of data , which we pre-processed to 0.

As there are as many principal components as there are variables in the data, principal components are constructed in such a manner that the first principal component accounts for the largest possible variance in the data set. i.e, we consider the best-fit line for the data as first principal component (PC1). The line should be such that the we minimise the sum of square of distances from line which is same as maximising sum of squares of projection distances on the line .

Now , the second principal component has to be calculated which should be uncorrelated i.e, should be perpendicular to the first principal component and that it accounts for the next highest variance and soon.

Considering any vector u from $\{u_1, u_2, ...\}$, considering st.$||u|| = 1$, we have to find u such that it maximises the following variance ,

$$\max \frac{1}{m} \sum_i (x^{(i)^T} u)^2$$
$$\max \frac{1}{m} \sum_i (u^T x^{(i)} x^{(i)^T} u)$$
$$\max u^T \left(\frac{1}{m} \sum_i (x^{(i)} x^{(i)^T}) \right) u$$

here we can see that as we that the covariance matrix of the data is,

$$S = \frac{1}{m} \sum_i (x^{(i)} x^{(i)^T})$$

we can rewrite it as,

$$\max u^T S u$$

We can use the Lagrangian multiplier and maximise it ,

$$\begin{aligned}\mathcal{L}(u,\lambda) &= u^T S u + \lambda(1 - u^T u)\\ \frac{\partial \mathcal{L}}{\partial u} &= 2u^T S - 2\lambda u^T = 0\\ Su &= \lambda u\\ \frac{\partial \mathcal{L}}{\partial \lambda} &= 1 - u^T u\\ u^T u &= 1\end{aligned}$$

which implies that u is a eigen vector of the covariance matrix S , and λ is the eigen value which is equal to variance since,

$$\begin{aligned}V &= u^T S u\\ V &= \lambda u^T u = \lambda\end{aligned}$$

Therefore , the vectors $u_1, u_2, ..., u_k$ are the top k eigen vectors of the covariance matrix S , corresponding to highest k eigen values of matrix S.

2.2 Questions

1. What's the difference between PCA and t-SNE algorithms?
2. Why do we do standardisation in PCA ?
3. Can we use PCA for feature selection?
4. Explain the curse of dimensionality.
5. What are the results of PCA when used on noisy data?

3 DBSCAN

The DBSCAN(Density-based spatial clustering of applications with noise) algorithm is based on this intuitive notion of "clusters" and "noise" .The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points. DBSCAN is used because , the real-life data may contain noise or the clusters can be of arbitrary shape ,it is especially useful when the training data consists of nested clusters, where algorithms like k-means will not work because the centroids of nested clusters can be very close to each other.

2.1 Algorithm

DBSCAN's clustering principle is congruent with how people typically think of clusters, which is why it performs so well. A human would base their decision to locate clusters given an arbitrary set of data points on how packed the points are in a specific area. DBSCAN thus imitates the human brain when looking for clusters.

DBSCAN algorithm requires two parameters:

- eps : It defines the neighborhood around a data point i.e. if the distance between two points is lower or equal to 'eps' then they are considered neighbors.
- MinPts : Minimum number of neighbors (data points) within eps radius.As a general rule, MinPts $\geq$ D +1 , where D is the number of dimensions.

Based on the eps , the datapoints are also classified into 3 types ,

- Core Point : A point is a core point if it has more than MinPts points within eps.
- Border Point : A point which has fewer than MinPts within eps but it is in the neighborhood of a core point.
- Noise or outlier : A point which is not a core point or border point

The algorithm works as follows,

1. For all the points in the database , it classifies them into core or non-core points.
2. Then it randomly picks a core point and assign it to the first cluster.
3. Next , the core points closer to the first cluster are added to it .
4. Then , the core points closer to the growing first cluster are added to it.
5. Finally , the non-core points closer to the points of the final cluster are added to it.

After forming of all the clusters , the left points are called outliners , or noise points.

3.2 Questions

1. How to get the optimum parameters for DBSCAN ?
2. What are core points?
3. Is the number of clustered formed given as a parameter to DBSCAN?
4. What is the worst case time complexity of DBSCAN algorithm?
5. What points would you keep in mind while determining the MinPts parameter?

4 SVD

Singular value decomposition (SVD) is a matrix factorization technique. SVD breaks down a given matrix into three components: a unitary matrix, a diagonal matrix, and another unitary matrix.

4.1 Mathematical Formulation

Let A belongs to R^{mn} be a real matrix.The Singular value decomposition theorem, states that there always exists a unique decomposition of A of the form

$$A = USV^T$$

where U belongs to $R^{m\times r}$ is the left singular matrix, V belongs to $R^{n\times r}$ is the right singular matrix and S belongs to $R^{r\times r}$ is a diagonal matrix containing the singular values along its diagonal. Here r is the rank of A.

Let us look at its proof.

- Firstly ,We know that is a matrix is symmetric , that is $B = B^T$ then we can say that the matrix B is diagonalizable with a unitary matrix and it has real eigen values. Which is the spectral theorem

Now, let us consider A^TA ,which is symmetric matrix ,as

$$(A^TA)^T = A^T(A^T)^T = A^TA$$

So, from the spectral theorem we can say that,

$$(A^TA)V = DV$$

where D is a diagonal matrix containing the eigen values in dereasing order in the diagonal and V is unitary.So,

$$V^T = V^-1$$

- We can prove that the eigen values of A^TA are always non-negative.

Consider ,

$$\begin{aligned} A^TAx &= \lambda x \\ x^TA^TAx &= \lambda x^Tx \\ (Ax)^T(Ax) &= \lambda x^Tx \end{aligned}$$

So ,

$$\lambda = \frac{||Ax||^2}{||x||^2} \geq 0$$

Now , continuing, let v_i denote the ith coloumn of V ,and $w_i = Av_i$. Consider the inner product of any two w_i, w_j which would imply,

$$\begin{aligned} < w_i, w_j > &= w_i^Tw_j \\ &= v_i^TA^TAv_j \\ &= (V^TA^TAV)_{ij} \\ &= D_{ij} \end{aligned}$$

that is inner product is 0 is i$\neq$ j implying they are orthogonal to each other , else the value is D_{ii}. So , we can say that w_i's form an orthogonal set . We can also say something about its length,

$$\begin{aligned} ||w_i||^2 &= D_{ii} \\ ||w_i|| &= \sqrt{D_{ii}} \end{aligned}$$

Since we have seen that eigen values are positive for A^TA , the length of the orthogonal vectors will be valid and real.

So, we considered AV = W , where we found out W to be a matrix with orthogonal columns whose lengths we found out (not always one ,i,e not a unitary matrix) ,so we can write W as,

$$W = US$$

where u is a unitary matrix and $S_{ii} = \sigma_i = \sqrt{D_{ii}}$,which scales the columns of the unitary matrix U.

Which leaves us with,

$$\begin{aligned} AV &= US \\ A &= USV^{-1} \\ A &= USV^T \end{aligned}$$

Moreover ,

$$\begin{aligned} A^TA &= (USV^T)^T(USV^T) \\ &= (VS^TU^T)(USV^T) \\ &= VS^2V^T \\ A^TAV &= VS^2 \end{aligned}$$

Similarly,

$$AA^TU = US^2$$

Thus , the columns of U are the eigenvectors of AA^T and the columns of V are the eigenvectors of A^TA.

4.2 Questions

1. Is SVD possible for only square matrices or any size of matrices?
2. If dimensions of A are $a \times b$ and its SVD is USV^T then dimensions of U and V are?
3. Is the no. of non-zero elements in S is equal to rank of matrix A.
4. How can SVD we used in Machine learning?
5. Let A = $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$, find AA^T, A^TA, eigen values and vectors of the above matrices and thus write the singular valued decomposition of A.

5 Examples

SVD

Let us decompose the following matrix , A = $\begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix}$

First we will have to find AA^T and A^TA and their eigen values, it is important to note that both the matrices have same non-zero eigen values which we will see,

$$A^T = \begin{bmatrix} 3 & 2 \\ 2 & 3 \\ -2 & -2 \end{bmatrix}$$

$$AA^T = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix} \begin{bmatrix} 3 & 2 \\ 2 & 3 \\ -2 & -2 \end{bmatrix}$$

$$AA^T = \begin{bmatrix} 17 & 8 \\ 8 & 17 \end{bmatrix}$$

$$A^TA = \begin{bmatrix} 13 & 12 & 2 \\ 12 & 13 & -2 \\ 2 & -2 & 8 \end{bmatrix}$$

From the matrix we get the eigen values of AA^T to be 25,9. And the eigen values of A^TA are 25,9,0.

Now we will have to find eigen vectors corresponding to eigen values,

$$\begin{bmatrix} 17-\lambda & 8 \\ 8 & 17-\lambda \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0$$

$$(17-\lambda)v_1 + 8v_2 = 0$$

$$\frac{v_1}{v_2} = \frac{8}{\lambda - 17}$$

So ,the normalised eigen vector of AA^T are,

$$v_{\lambda=25} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$v_{\lambda=9} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

and similarly the normalised eigen vectors for non-zero eigen values of A^TA are,

$$v_{\lambda=25} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$v_{\lambda=9} = \frac{1}{\sqrt{18}} \begin{bmatrix} 1 \\ -1 \\ 4 \end{bmatrix}$$

Now, we will have to find the singular values of A which are the square roots of eigen values,

$$\sigma_1 = 5$$

$$\sigma_2 = 3$$

Now , we can write the U,V,S matrices which form the decomposition.

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

$$V = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{18}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{18}} \\ 0 & \frac{4}{\sqrt{18}} \end{bmatrix}$$

$$S = \begin{bmatrix} 5 & 0 \\ 0 & 3 \end{bmatrix}$$

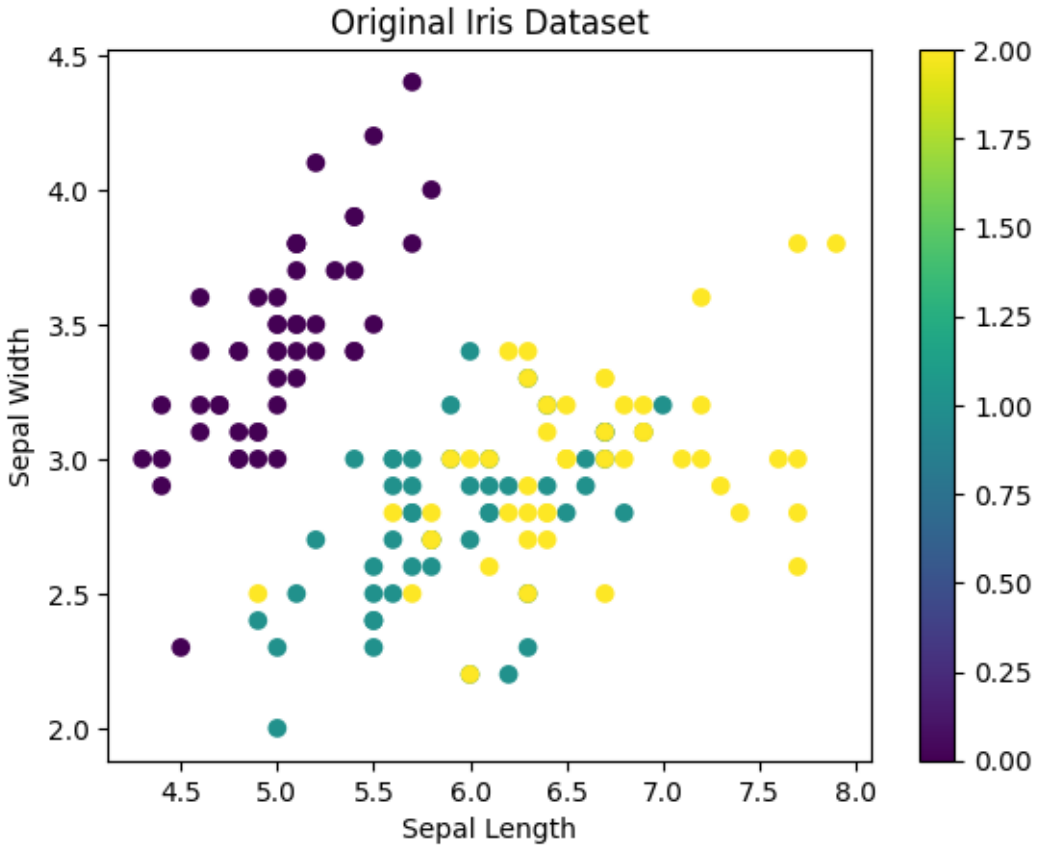
Finally , the decomposition is,

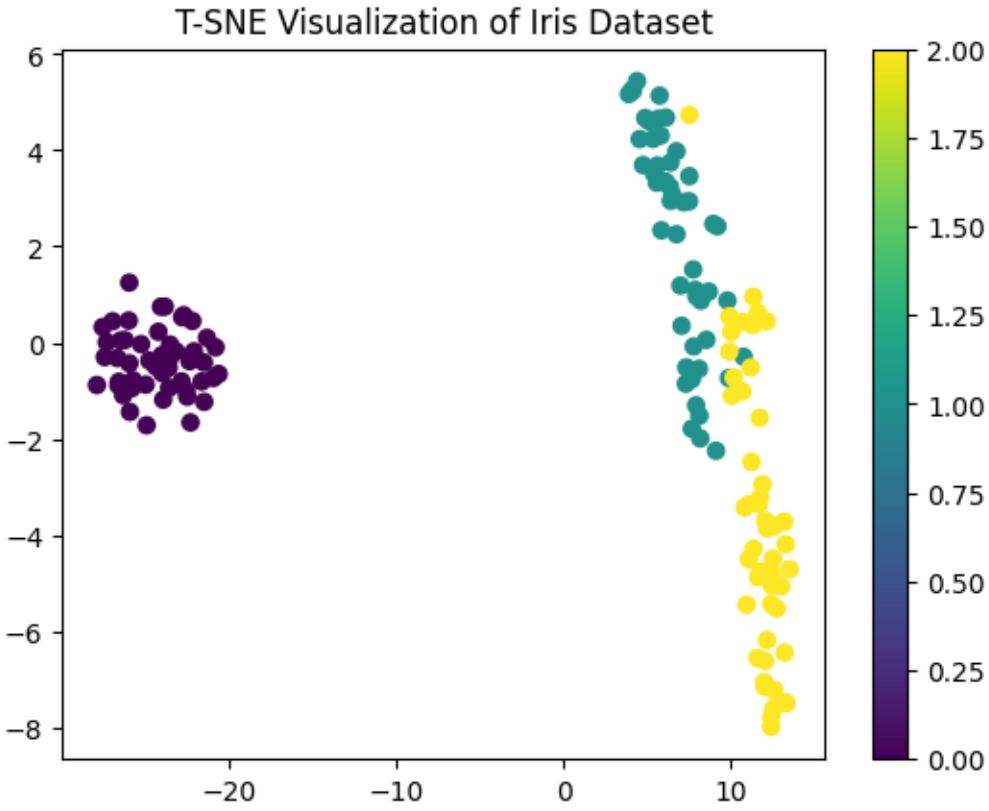
$$\begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 5 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{18}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{18}} \\ 0 & \frac{4}{\sqrt{18}} \end{bmatrix}^T$$

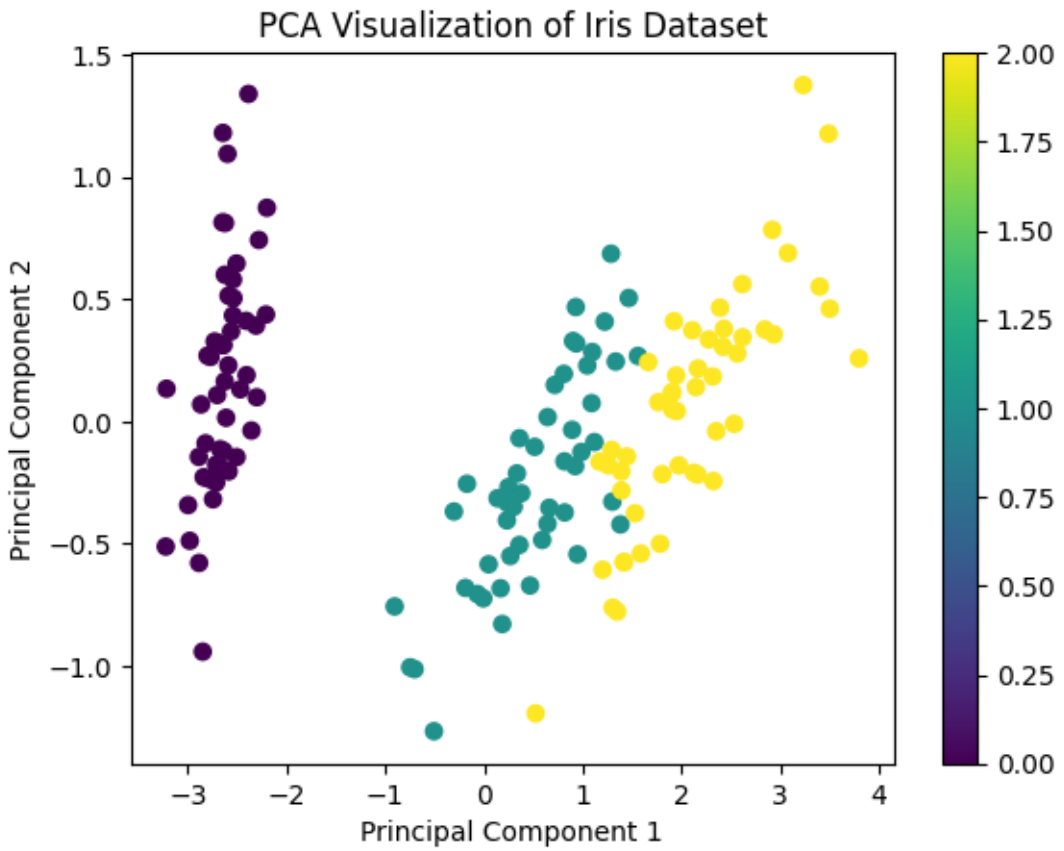
Data Visualisation

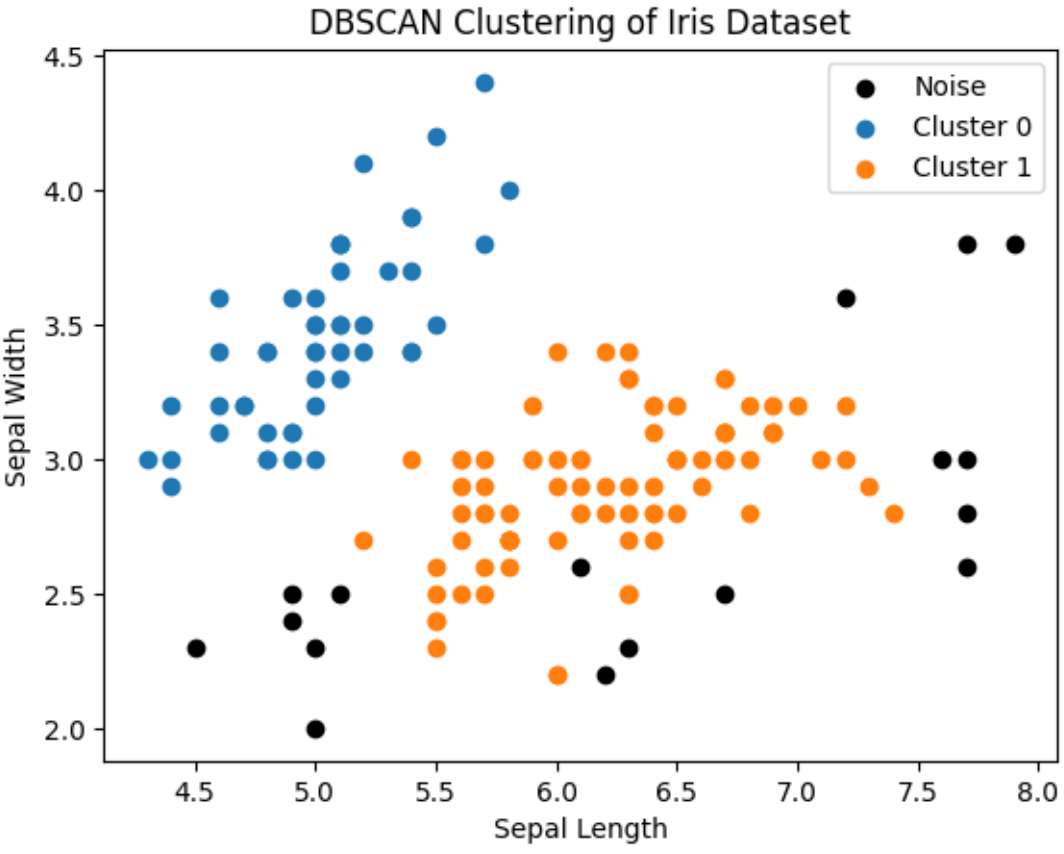
```
import numpy as np
from sklearn import datasets
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

# Load Iris dataset
iris = datasets.load_iris()
data = iris.data
target = iris.target

# Plot the original data
plt.scatter(data[:, 0], data[:, 1], c=target)
plt.title("Original Iris Dataset")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.colorbar()
plt.show()

# Perform T-SNE embedding
tsne = TSNE(n_components=2, random_state=42)
embedded_data_tsne = tsne.fit_transform(data)

# Perform PCA
pca = PCA(n_components=2)
embedded_data_pca = pca.fit_transform(data)
```

```
    # Perform DBSCAN clustering
    dbscan = DBSCAN(eps=0.5, min_samples=5)
    clusters = dbscan.fit_predict(data)

    # Plot the embedded data for TSNE
    plt.scatter(embedded_data_tsne[:, 0], embedded_data_tsne[:, 1], c=
       target)
    plt.title("T-SNE Visualization of Iris Dataset")
    plt.colorbar()
    plt.show()


    # Plot the embedded data for PCA
    plt.scatter(embedded_data_pca[:, 0], embedded_data_pca[:, 1], c=
       target)
    plt.title("PCA Visualization of Iris Dataset")
    plt.xlabel('Principal Component 1')
    plt.ylabel('Principal Component 2')
    plt.colorbar()
    plt.show()

    # Get unique cluster labels for DBSCAN
    unique_labels = np.unique(clusters)

    # Plot the clustered data
    for label in unique_labels:
        if label == -1:
            # Plot points labeled as noise in black
            plt.scatter(data[clusters == label, 0], data[clusters ==
               label, 1], color='black', label='Noise')
        else:
            # Plot points for each cluster
            plt.scatter(data[clusters == label, 0], data[clusters ==
               label, 1], label=f'Cluster {label}')

    plt.title("DBSCAN Clustering of Iris Dataset")
    plt.xlabel("Sepal Length")
    plt.ylabel("Sepal Width")
    plt.legend()
    plt.show()
```


Original Iris Dataset
Sepal Width
Sepal Length
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00


T-SNE Visualization of Iris Dataset
6
4
2
0
−2
−4
−6
−8
−20
−10
0
10
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00


PCA Visualization of Iris Dataset
Principal Component 2
Principal Component 1
1.5
1.0
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3
4
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00


DBSCAN Clustering of Iris Dataset
Noise
Cluster 0
Cluster 1
Sepal Width
Sepal Length
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0