Date : 25/04/2024

EXPERIMENT NO. 10

AIM: To implement TSNE(t-distributed stochastic neighbor embedding) using python programming.

SOFTWARE USED: Jupyter Notebook.

THEORY:

t-SNE, short for t-distributed stochastic neighbor embedding, is a machine learning technique used for dimensionality reduction and visualization of high-dimensional data. It was developed by Laurens van der Maaten and Geoffrey Hinton in 2008. t-SNE is particularly effective for visualizing complex datasets by projecting them into a two-dimensional or three-dimensional space while preserving local data structures and relationships.

Key Concepts:

- Dimensionality Reduction: t-SNE reduces high-dimensional data to a lower-dimensional space, typically 2D or 3D, to make the data easier to visualize and analyze.
- Preserving Local Structure: The technique focuses on maintaining the relationships between nearby data points from the high-dimensional space in the lower-dimensional space.
- Distance Metrics: t-SNE uses a distance metric (such as Euclidean distance) to compute pairwise similarities between data points in both high-dimensional and low-dimensional spaces.
- Gaussian and t-Distributions: In the high-dimensional space, similarities between data points are calculated using a Gaussian distribution. In the low-dimensional space, a heavy-tailed t-distribution is used to model similarities, which allows the technique to capture fine-grained relationships.
- Perplexity: The perplexity parameter controls the balance between focusing on local versus global data structures. It determines the bandwidth of the Gaussian kernel used in the high-dimensional space.
- Optimization: t-SNE optimizes the arrangement of data points in the lower-dimensional space to minimize the divergence (KL divergence) between the distributions in the high-dimensional and low-dimensional spaces.

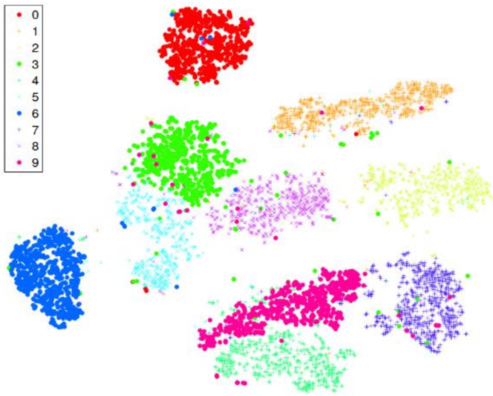
Applications:

- Data Visualization: t-SNE is widely used to visualize high-dimensional data in 2D or 3D, such as in natural language processing, genomics, and image processing.
- Clustering: t-SNE can reveal clusters in the data by grouping similar data points together in the low-dimensional space.
- Anomaly Detection: t-SNE can help identify anomalies in high-dimensional data by showing data points that are far from expected clusters.
- Feature Engineering: The lower-dimensional representation generated by t-SNE can be used as features for machine learning models.

Considerations:

- Interpretation: The resulting visualizations may not always be directly interpretable, as the technique is non-linear and can emphasize different aspects of the data.

- Parameter Selection: Choosing appropriate parameters (such as perplexity and learning rate) is important for achieving meaningful visualizations.
- Computational Cost: t-SNE can be computationally intensive, particularly for large datasets, and may require more time and memory compared to other dimensionality reduction techniques.
- Random Initialization: The algorithm uses random initialization, so results can vary between runs. It may be helpful to run t-SNE multiple times to verify the consistency of the visualizations.



OUTPUT CODE:

```
In [3]: #DSA experiment 10
        #TSNE
```

```
In [11]: !pip install bioinfokit
```

```
Collecting bioinfokit
  Using cached bioinfokit-2.1.3.tar.gz (87 kB)
Requirement already satisfied: pandas in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from bioinfokit) (2.0.0)
Requirement already satisfied: numpy in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from bioinfokit) (1.25.2)
Requirement already satisfied: matplotlib in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from bioinfokit) (3.7.2)
Requirement already satisfied: scipy in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from bioinfokit) (1.11.1)
Requirement already satisfied: scikit-learn in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from bioinfokit) (1.3.0)
Requirement already satisfied: seaborn in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from bioinfokit) (0.13.2)
Collecting matplotlib-venn
  Using cached matplotlib_venn-0.11.10-py3-none-any.whl (33 kB)
Collecting tabulate
  Using cached tabulate-0.9.0-py3-none-any.whl (35 kB)
Requirement already satisfied: statsmodels in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from bioinfokit) (0.14.1)
Collecting textwrap3
  Using cached textwrap3-0.9.2-py2.py3-none-any.whl (12 kB)
Collecting adjustText
  Using cached adjustText-1.1.1-py3-none-any.whl (11 kB)
Requirement already satisfied: cycler>=0.10 in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from matplotlib->bioinfok
it) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from matplotlib->bio
infokit) (4.25.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from matplotlib->bioinfo
kit) (9.0.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from matplotlib->
bioinfokit) (2.8.2)
Requirement already satisfied: packaging>=20.0 in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from matplotlib->bioin
fokit) (23.2)
```

```
Requirement already satisfied: packaging>=20.0 in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from matplotlib->bioin
fokit) (23.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from matplotlib->bioi
nfokit) (1.1.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from matplotlib->bio
infokit) (1.3.1)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from matplotlib-
>bioinfokit) (3.0.4)
Requirement already satisfied: six>=1.5 in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from python-dateutil>=2.7->ma
tplotlib->bioinfokit) (1.16.0)
Requirement already satisfied: tzdata>=2022.1 in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from pandas->bioinfoki
t) (2023.3)
Requirement already satisfied: pytz>=2020.1 in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from pandas->bioinfokit)
(2023.3)
Requirement already satisfied: joblib>=1.1.1 in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from scikit-learn->bioin
fokit) (1.3.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from scikit-learn
->bioinfokit) (3.2.0)
Requirement already satisfied: patsy>=0.5.4 in c:\users\anjal\anaconda3\envs\myenv\lib\site-packages (from statsmodels->bioinfo
kit) (0.5.6)
Building wheels for collected packages: bioinfokit
  Building wheel for bioinfokit (setup.py): started
  Building wheel for bioinfokit (setup.py): finished with status 'done'
  Created wheel for bioinfokit: filename=bioinfokit-2.1.3-py3-none-any.whl size=59093 sha256=e4aa605740063aac4485f83785ca9d957f
d7745afe916e4ae20f9f8e1cc6dd6b
  Stored in directory: c:\users\anjal\appdata\local\pip\cache\wheels\ac\67\a7\4e0b4172d5415933127e819d7d7080ae08a6220949ad2f6de
5
Successfully built bioinfokit
Installing collected packages: textwrap3, tabulate, matplotlib-venn, adjustText, bioinfokit
Successfully installed adjustText-1.1.1 bioinfokit-2.1.3 matplotlib-venn-0.11.10 tabulate-0.9.0 textwrap3-0.9.2
```

In [15]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
from bioinfokit.visuz import cluster
```

In [16]:
```python
data = pd.read_csv('TSNE_data.csv')
```

In [17]:
```python
data.head()
```

Out[17]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

5 rows × 31 columns

In [18]:
```python
data.describe()
```

Out[18]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fracta
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	

8 rows × 30 columns

```
In [19]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   diagnosis                569 non-null    object
 1   radius_mean              569 non-null    float64
 2   texture_mean             569 non-null    float64
 3   perimeter_mean           569 non-null    float64
 4   area_mean                569 non-null    float64
 5   smoothness_mean          569 non-null    float64
 6   compactness_mean         569 non-null    float64
 7   concavity_mean           569 non-null    float64
 8   concave points_mean      569 non-null    float64
 9   symmetry_mean            569 non-null    float64
 10  fractal_dimension_mean   569 non-null    float64
 11  radius_se                569 non-null    float64
 12  texture_se               569 non-null    float64
 13  perimeter_se             569 non-null    float64
 14  area_se                  569 non-null    float64
 15  smoothness_se            569 non-null    float64
 16  compactness_se           569 non-null    float64
 17  concavity_se             569 non-null    float64
 18  concave points_se        569 non-null    float64
 19  symmetry_se              569 non-null    float64
 20  fractal_dimension_se     569 non-null    float64
 21  radius_worst             569 non-null    float64
 22  texture_worst            569 non-null    float64
 23  perimeter_worst          569 non-null    float64
 24  area_worst               569 non-null    float64
 25  smoothness_worst         569 non-null    float64
 26  compactness_worst        569 non-null    float64
 27  concavity_worst          569 non-null    float64
 28  concave points_worst     569 non-null    float64
 29  symmetry_worst           569 non-null    float64
 30  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), object(1)
memory usage: 137.9+ KB
```

```
In [20]: data.shape
Out[20]: (569, 31)
```

```
In [21]: filename = "TSNE_data.csv"
         dataframe = pd.read_csv(filename)
```

```
In [22]: array = dataframe.values
         X = array[:,1:]
         Y = array[:,0]
```

```
In [23]: from bioinfokit.visuz import cluster
         data_tsne = TSNE(n_components=2).fit_transform(X)
         cluster.tsneplot(score=data_tsne)
```

```
In [25]: color_class = dataframe['diagnosis'].to_numpy()
         cluster.tsneplot(score=data_tsne, colorlist=color_class, legendpos='upper right',legendanchor=(1.15,1))
```

```
In [26]: data_tsne
Out[26]: array([[ 41.63356  , -11.834398 ],
                [ 41.613197 ,  -9.135637 ],
                [ 37.03936  ,  -9.464888 ],
                ...,
                [ 21.530272 ,  -3.0823648],
                [ 39.503273 ,  -9.354743 ],
                [-36.991283 , -26.944433 ]], dtype=float32)
```

CONCLUSION:

In conclusion, t-SNE is a powerful tool for reducing the dimensionality of high-dimensional data and creating visually interpretable representations. It is particularly effective for capturing local relationships within the data and can be useful for visualization, clustering, and anomaly detection in complex datasets.