Name:- Aniket Kumar

Roll No:- PF 45

Sr. No:- 30

PRN No:- 1032171203

LAB Assignment -02

Aim:- To understand dimensionality reduction concept by comparing PCA and t-SNE

Theory:-

PCA

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

The idea of PCA is simple- reduce the number of variables of a data set, while preserving as much information as possible.

PCA Steps for dimensionality reduction:

1) Column standardization of data.
2) Find Covariance matrix
3) Find eigen values and eigen vectors
4.] Find Principal components
5.] Reducing dimensions of dataset.

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. In simpler terms, t-SNE gives you a feel or intuition of how the data is arranged in a high-dimensional space.

PCA	T-SNE
1.] It is a linear Dimensionality reduction technique	It is a non-linear Dimensionality reduction technique
2.] It tries to preserve the global structure of the data	It tries to preserve the local structure of data
3.] It doesnot work well as compared to t-SNE	It is one of the best dimensionality reduction technique
4.] It gets highly affected by Outliers	It can handle outliers.
5.] It doesnot involve Hyperparameters	It involve Hyperparameters such as perplexity, learning.

FAQ:-

1. What are various dimensionality reduction techniques?

Ans:- The various dimensionality reduction techniques are:-

1] Linear Discriminant Analysis

2] Auto encoder

3] t-SNE

4] Missing Values Ratio

5] Low Variance Filter.

2. Define feature engineering.

Ans:- Feature engineering is the process of using domain knowledge to extract features from raw data via data mining techniques. These features can be used to improve the performance of machine learning algorithm.

In [1]:
```
#Name:- Aniket Kumar
#PRN:- 1032171203
#Roll No:- PF45
#S.No:-30
#Lab Assignment 02
```

PCA Using Scratch

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
# Importing data and display
data=pd.read_csv(r"C:\Users\Aniket\Desktop\Aml\2\archive\winequality-red.csv")
data
```

Out[2]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

In [3]:
```
data.isna().sum()
```

Out[3]:
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

In [4]:
```
data.head(5)
```

Out[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

In [5]:
```
# save the labels into a variable l.
l = data['quality']
```

In [6]:
```
# Drop the label feature and store the pixel data in d.
d = data.drop("quality",axis=1)
```

In [7]:
```
d.shape
```

Out[7]: (1599, 11)

In [8]:
```
l.shape
```

Out[8]: (1599,)

In [9]:
```
# Standardization   in 0.....1 format
from sklearn.preprocessing import StandardScaler
standardized_data = StandardScaler().fit_transform(d)
print(standardized_data.shape)
print(standardized_data)
```

```
(1599, 11)
[[-0.52835961  0.96187667 -1.39147228 ...  1.28864292 -0.57920652
  -0.96024611]
 [-0.29854743  1.96744245 -1.39147228 ... -0.7199333   0.1289504
  -0.58477711]
 [-0.29854743  1.29706527 -1.18607043 ... -0.33117661 -0.04808883
  -0.58477711]
 ...
 [-1.1603431  -0.09955388 -0.72391627 ...  0.70550789  0.54204194
   0.54162988]
 [-1.39015528  0.65462046 -0.77526673 ...  1.6773996   0.30598963
  -0.20930812]
 [-1.33270223 -1.21684919  1.02199944 ...  0.51112954  0.01092425
   0.54162988]]
```

In [10]:
```
# Covariance Matrix
#find the co-variance matrix which is : A^T * A
sample_data = standardized_data

# matrix multiplication using numpy
covar_matrix = np.matmul(sample_data.T , sample_data)
print ( "The shape of variance matrix = ", covar_matrix.shape)
```

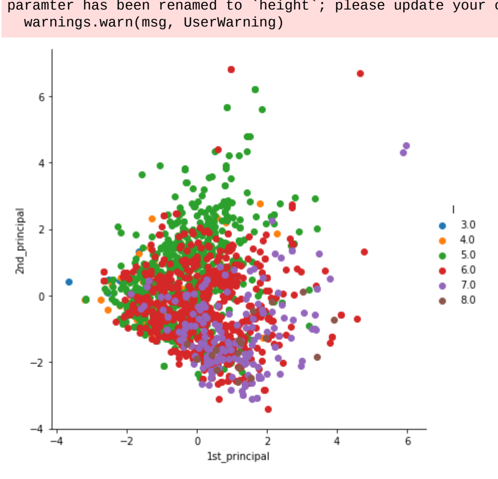
```
The shape of variance matrix =  (11, 11)
```

In [11]:
```
#eigen values, eigen vector
# finding the top two eigen-values and corresponding eigen-vectors
# for projecting onto a 2-Dim space.
from scipy.linalg import eigh
# the parameter 'eigvals' is defined (low value to heigh value)
# eigh function will return the eigen values in asending order
# this code generates only the top 2 (782 and 783) eigenvalues.
values, vectors = eigh(covar_matrix, eigvals=(8,9))
print(values)

print("Shape of eigen vectors = ",vectors.shape)
# converting the eigen vectors into (2,d) shape for easyness of further computations
vectors = vectors.T

print("Updated shape of eigen vectors = ",vectors.shape)
# here the vectors[1] represent the eigen vector corresponding 1st principal eigen vector
# here the vectors[0] represent the eigen vector corresponding 2nd principal eigen vector


# projecting the original data sample on the plane
#formed by two principal eigen vectors by vector-vector multiplication.
import matplotlib.pyplot as plt
new_coordinates = np.matmul(vectors, sample_data.T)
print (" resultanat new data points' shape ", vectors.shape, "X", sample_data.T.shape," = ", new_coordinates.shape)

#  resultanat new data points' shape (2,784) X (784, 15000)= (2, 15000)


import pandas as pd

# appending label to the 2d projected data
new_coordinates = np.vstack((new_coordinates, l)).T

# creating a new data frame for ploting the labeled points.
dataframe = pd.DataFrame(data=new_coordinates, columns=("1st_principal", "2nd_principal",
```

```
[2479.31903855 3079.52959367]
Shape of eigen vectors =  (11, 2)
Updated shape of eigen vectors =  (2, 11)
 resultanat new data points' shape  (2, 11) X (11, 1599)  =  (2, 1599)
   1st_principal  2nd_principal    l
0      -1.774454       0.450950  5.0
1      -0.911690       1.856553  5.0
2      -1.171394       0.882039  5.0
3       0.243489      -0.269976  6.0
4      -1.774454       0.450950  5.0
```

In [12]:
```
# ploting the 2d data points with seaborn
import seaborn as sn
sn.FacetGrid(dataframe, hue="l", size=6).map(plt.scatter, '1st_principal', '2nd_principal').add_legend()
```

```
C:\Users\Aniket\Anaconda3\lib\site-packages\seaborn\axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```



PCA direct (Using Algorithm)

In [16]:
```
# initializing the pca
from sklearn import decomposition
pca = decomposition.PCA()
```

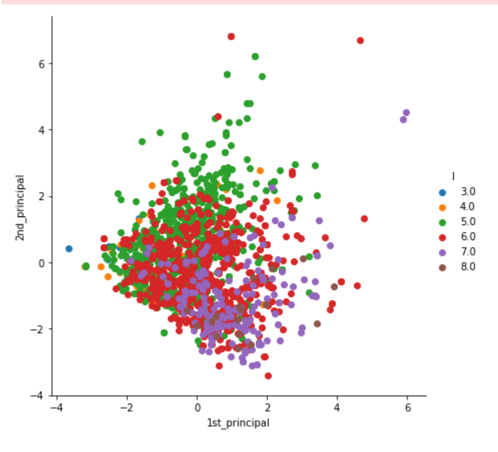
In [17]:
```
# configuring the parameteres
# the number of components = 2
pca.n_components = 2
pca_data = pca.fit_transform(sample_data)
```

In [18]:
```
# pca_reduced will contain the 2-d projects of simple data
print("shape of pca_reduced.shape = ", pca_data.shape)
```

```
shape of pca_reduced.shape =  (1599, 2)
```

In [19]:
```
# attaching the label for each 2-d data point
pca_data = np.vstack((pca_data.T, l)).T

# creating a new data fram which help us in ploting the result data
pca_df = pd.DataFrame(data=pca_data, columns=("1st_principal", "2nd_principal", "l"))


#import seaborn as sn
sn.FacetGrid(dataframe, hue="l", size=6).map(plt.scatter, '1st_principal', '2nd_principal').add_legend()
```

```
C:\Users\Aniket\Anaconda3\lib\site-packages\seaborn\axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```



T-SNE

In [21]:
```
from sklearn.manifold import TSNE
```

In [22]:
```
# Picking the top 1000 points as TSNE takes a lot of time for 15K points
data_1000 = standardized_data[0:1000,:]
```

In [23]:
```
labels_1000 = l[0:1000]
```

In [24]:
```
model = TSNE(n_components=2, random_state=0)
```

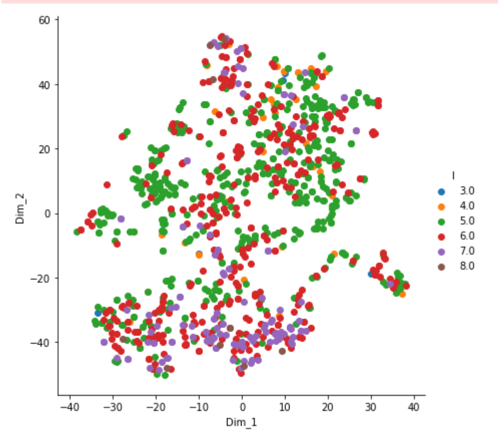
In [25]:
```
# configuring the parameteres
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data = model.fit_transform(data_1000)
tsne_data.shape
```

Out[25]: (1000, 2)

In [26]:
```
# creating a new data frame which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, labels_1000)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "l"))
```

In [27]:
```
# Ploting the result of tsne
sn.FacetGrid(tsne_df, hue="l", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_legend()
plt.show()
```

```
C:\Users\Aniket\Anaconda3\lib\site-packages\seaborn\axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

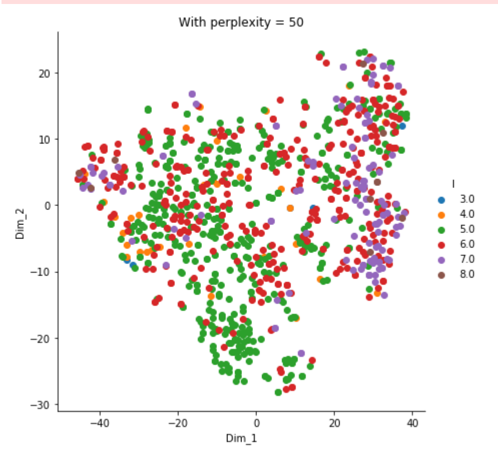


In [28]:
```
model = TSNE(n_components=2, random_state=0, perplexity=50)
tsne_data = model.fit_transform(data_1000)
```

In [29]:
```
# creating a new data fram which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, labels_1000)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "l"))
```

In [30]:
```
# Ploting the result of tsne
sn.FacetGrid(tsne_df, hue="l", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_legend()
plt.title('With perplexity = 50')
plt.show()
```

```
C:\Users\Aniket\Anaconda3\lib\site-packages\seaborn\axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

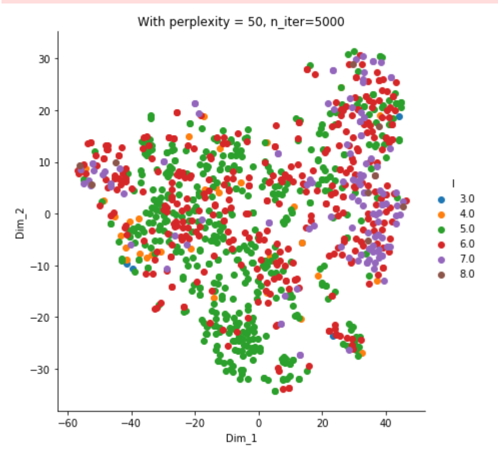


In [31]:
```
model = TSNE(n_components=2, random_state=0, perplexity=50,  n_iter=5000)
tsne_data = model.fit_transform(data_1000)
```

In [32]:
```
# creating a new data fram which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, labels_1000)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "l"))
```

In [33]:
```
# Ploting the result of tsne
sn.FacetGrid(tsne_df, hue="l", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_legend()
plt.title('With perplexity = 50, n_iter=5000')
plt.show()
```

```
C:\Users\Aniket\Anaconda3\lib\site-packages\seaborn\axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

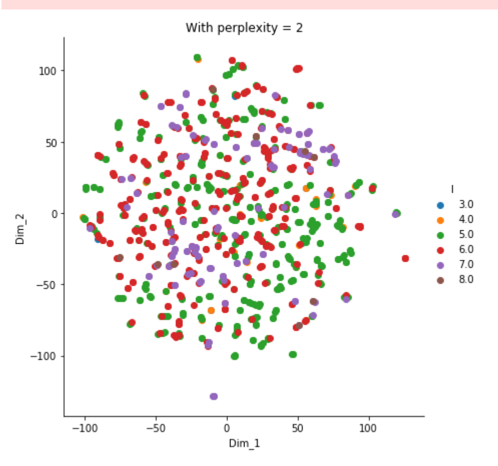


In [34]:
```
model = TSNE(n_components=2, random_state=0, perplexity=2)
tsne_data = model.fit_transform(data_1000)
```

In [35]:
```
# creating a new data fram which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, labels_1000)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "l"))
```

In [36]:
```
# Ploting the result of tsne
sn.FacetGrid(tsne_df, hue="l", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_legend()
plt.title('With perplexity = 2')
plt.show()
```

```
C:\Users\Aniket\Anaconda3\lib\site-packages\seaborn\axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```



In [ ]: