**Practical 6**

**Aim:**

**Perform Principle Component Analysis on Iris dataset.**

**Code:**

import pandas as pd

from mpl\_toolkits.mplot3d import Axes3D

import matplotlib.pyplot as plt

import numpy as np

dt=pd.read\_csv('iris.csv')

sl=np.array(dt['sepal.length'])

sw=np.array(dt['sepal.width'])

pl=np.array(dt['petal.length'])

pw=np.array(dt['petal.width'])

t=np.array(dt['variety'])

print(t)

for item in list(t):

if item =='Setosa':

t[list(t).index(item)]='r'

elif item =='Virginica':

t[list(t).index(item)]='g'

else:

t[list(t).index(item)]='b'

print(t)

fig = plt.figure()

ax = fig.add\_subplot(221, projection='3d')

ax2 = fig.add\_subplot(222, projection='3d')

ax3 = fig.add\_subplot(223, projection='3d')

ax.set\_xlabel("sepal length")

ax.set\_ylabel("sepal.width")

ax.set\_zlabel("petal.length")

ax2.set\_xlabel("sepal length")

ax2.set\_ylabel("sepal width")

ax2.set\_zlabel("petal width")

ax3.set\_xlabel("sepal width")

ax3.set\_ylabel("sepal width")

ax3.set\_zlabel("petal width")

img = ax.scatter(sl, sw, pl, c=t, cmap=plt.hot())

img2 = ax2.scatter(sl, sw, pw, c=t, cmap=plt.hot())

img3 = ax3.scatter(pl, sw, pw, c=t, cmap=plt.hot())

fig.colorbar(img)

fig.colorbar(img2)

plt.show()

pca = PCA(n\_components=2)

principalComponents = pca.fit\_transform(x)

principalDf = pd.DataFrame(data = principalComponents

, columns = ['principal component 1', 'principal component 2'])

finalDf = pd.concat([principalDf, df[['target']]], axis = 1)

fig = plt.figure(figsize = (8,8))

ax = fig.add\_subplot(1,1,1)

ax.set\_xlabel('Principal Component 1', fontsize = 15)

ax.set\_ylabel('Principal Component 2', fontsize = 15)

ax.set\_title('2 component PCA', fontsize = 20)

targets = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']

colors = ['r', 'g', 'b']

for target, color in zip(targets,colors):

indicesToKeep = finalDf['target'] == target

ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']

, finalDf.loc[indicesToKeep, 'principal component 2']

, c = color

, s = 50)

ax.legend(targets)

ax.grid()

plt.show()

**Output:**



