**Practical-10**

**Implementation of K NN clustering on jupyter notebook using python.**

In [ ]:

import numpy as np

from sklearn import datasets

In [2]:

iris = datasets.load\_iris()

iris\_data = iris.data

iris\_labels = iris.target

print(iris\_data[0], iris\_data[79], iris\_data[100])

print(iris\_labels[0], iris\_labels[79], iris\_labels[100])

[5.1 3.5 1.4 0.2] [5.7 2.6 3.5 1. ] [6.3 3.3 6. 2.5]

0 1 2

In [3]:

np.random.seed(42)

indices = np.random.permutation(len(iris\_data))

n\_training\_samples = 12

learnset\_data = iris\_data[indices[:-n\_training\_samples]]

learnset\_labels = iris\_labels[indices[:-n\_training\_samples]]

testset\_data = iris\_data[indices[-n\_training\_samples:]]

testset\_labels = iris\_labels[indices[-n\_training\_samples:]]

In [14]:

print(learnset\_data[:4], learnset\_labels[:4])

print(testset\_data[:4], testset\_labels[:4])

[[6.1 2.8 4.7 1.2]

[5.7 3.8 1.7 0.3]

[7.7 2.6 6.9 2.3]

[6. 2.9 4.5 1.5]] [1 0 2 1]

[[5.7 2.8 4.1 1.3]

[6.5 3. 5.5 1.8]

[6.3 2.3 4.4 1.3]

[6.4 2.9 4.3 1.3]] [1 2 1 1]

In [15]:

%matplotlib inline

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

colours = ("r", "b")

X = []

for iclass in range(3):

X.append([[], [], []])

for i in range(len(learnset\_data)):

if learnset\_labels[i] == iclass:

X[iclass][0].append(learnset\_data[i][0])

X[iclass][1].append(learnset\_data[i][1])

X[iclass][2].append(sum(learnset\_data[i][2:]))

colours = ("r", "g", "y")

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

for iclass in range(3):

ax.scatter(X[iclass][0], X[iclass][1], X[iclass][2], c=colours[iclass])

plt.show()

