**Code:**

import pandas as pd

import numpy as np

wbcd = pd.read\_csv("C:\\Users\CSE-14\Desktop\wbcd.csv")

# converting B to Benign and M to Malignant

wbcd['diagnosis'] = np.where(wbcd['diagnosis'] == 'B', 'Benign ', wbcd['diagnosis'])

wbcd['diagnosis'] = np.where(wbcd['diagnosis'] == 'M', 'Malignant ', wbcd['diagnosis'])

wbcd = wbcd.iloc[:, 1:] # Excluding id column

desc = wbcd.describe()

# Normalization function

def norm\_func(i):

x = (i-i.min()) / (i.max()-i.min())

return (x)

# Normalized data frame (considering the numerical part of data)

wbcd\_n = norm\_func(wbcd.iloc[:, 1:])

norm\_data = wbcd\_n.describe()

X = np.array(wbcd\_n.iloc[:,:]) # Predictors

Y = np.array(wbcd['diagnosis']) # Target

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.2)

# Imbalance check

wbcd.diagnosis.value\_counts()

ytrain = pd.DataFrame(Y\_train)

ytest = pd.DataFrame(Y\_test)

ytrain.value\_counts()

ytest.value\_counts()

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors = 21)

knn.fit(X\_train, Y\_train)

pred = knn.predict(X\_test)

pred

# Evaluate the model

from sklearn.metrics import accuracy\_score

print(accuracy\_score(Y\_test, pred))

pd.crosstab(Y\_test, pred, rownames = ['Actual'], colnames= ['Predictions'])

# error on train data

pred\_train = knn.predict(X\_train)

print(accuracy\_score(Y\_train, pred\_train))

pd.crosstab(Y\_train, pred\_train, rownames=['Actual'], colnames = ['Predictions'])

# creating empty list variable

acc = []

# running KNN algorithm for 3 to 50 nearest neighbours(odd numbers) and

# storing the accuracy values

for i in range(1, 50, 2):

neigh = KNeighborsClassifier(n\_neighbors = i)

neigh.fit(X\_train, Y\_train)

train\_acc = np.mean(neigh.predict(X\_train) == Y\_train)

test\_acc = np.mean(neigh.predict(X\_test) == Y\_test)

acc.append([train\_acc, test\_acc])

import matplotlib.pyplot as plt # library to do visualizations

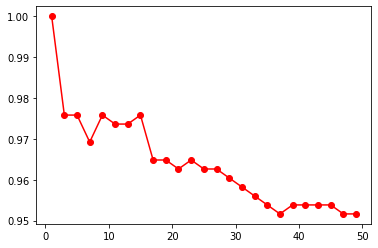
# train accuracy plot

plt.plot(np.arange(1,50,2),[i[0] for i in acc],"ro-")

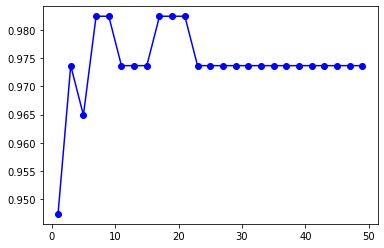
# test accuracy plot

plt.plot(np.arange(1,50,2),[i[1] for i in acc],"bo-")

**Outputs:**

****

**Scatter plot for accuracy of Train dataset**



**Scatter plot for accuracy of Test dataset**