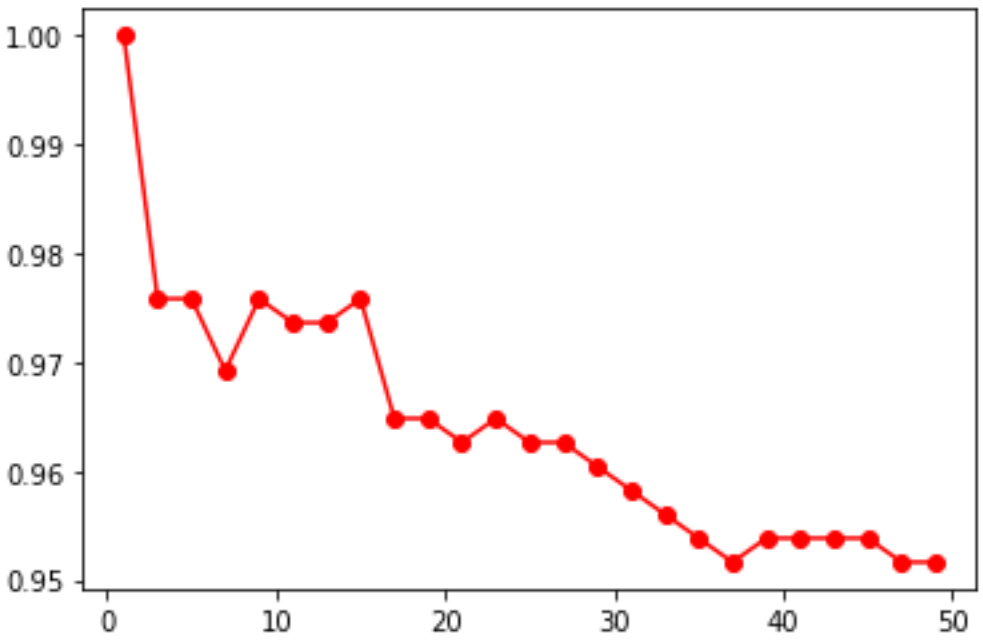
Code:

```
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()
```

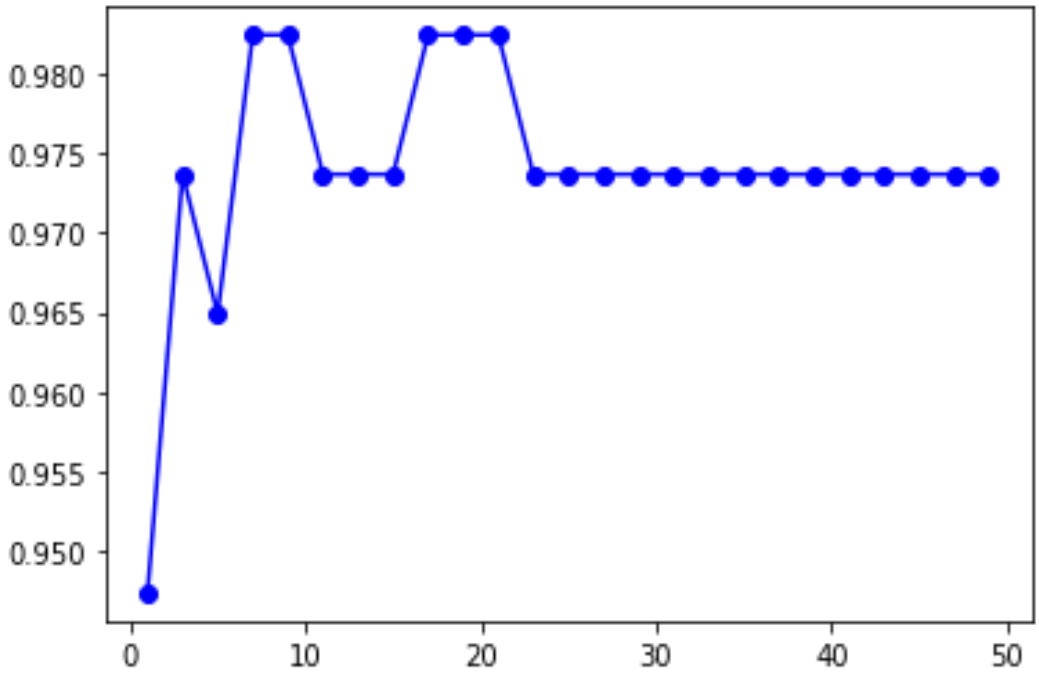
```
ytrain = pd.DataFrame(Y_train)
ytest = pd.DataFrame(Y_test)

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