Syed Ali COSC 750 HMWK 5

In [1]:
```
import numpy as np
import pandas
from matplotlib import pyplot as plt
from sklearn.datasets import load_breast_cancer
```

In [2]:
```
cancer = load_breast_cancer()
print(cancer.data[0])
```

```
[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
 1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
 6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
 1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
 4.601e-01 1.189e-01]
```

In [3]:
```
print(cancer.keys())
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filenam
e'])
```

In [4]:
```
print(cancer['feature_names'])
print(len(cancer['feature_names']))
```

```
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
30
```

In [5]:
```
"""converts the sklearn 'cancer' bunch
    Returns:
        pandas.DataFrame: cancer data
        """
data = np.c_[cancer.data, cancer.target]
columns = np.append(cancer.feature_names, ["target"])
df = pandas.DataFrame(data, columns=columns)
print(df)
```

```
     mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0          17.99         10.38          122.80     1001.0          0.11840
1          20.57         17.77          132.90     1326.0          0.08474
2          19.69         21.25          130.00     1203.0          0.10960
3          11.42         20.38           77.58      386.1          0.14250
4          20.29         14.34          135.10     1297.0          0.10030
..           ...           ...             ...        ...              ...
564        21.56         22.39          142.00     1479.0          0.11100
565        20.13         28.25          131.20     1261.0          0.09780
566        16.60         28.08          108.30      858.1          0.08455
567        20.60         29.33          140.10     1265.0          0.11780
568         7.76         24.54           47.92      181.0          0.05263

     mean compactness  mean concavity  mean concave points  mean symmetry  \
0             0.27760         0.30010              0.14710         0.2419
1             0.07864         0.08690              0.07017         0.1812
2             0.15990         0.19740              0.12790         0.2069
```

```
3            0.28390         0.24140              0.10520         0.2597
4            0.13280         0.19800              0.10430         0.1809
..               ...             ...                  ...            ...
564          0.11590         0.24390              0.13890         0.1726
565          0.10340         0.14400              0.09791         0.1752
566          0.10230         0.09251              0.05302         0.1590
567          0.27700         0.35140              0.15200         0.2397
568          0.04362         0.00000              0.00000         0.1587

     mean fractal dimension  ...  worst texture  worst perimeter  worst area  \
0                   0.07871  ...          17.33           184.60      2019.0
1                   0.05667  ...          23.41           158.80      1956.0
2                   0.05999  ...          25.53           152.50      1709.0
3                   0.09744  ...          26.50            98.87       567.7
4                   0.05883  ...          16.67           152.20      1575.0
..                      ...  ...            ...              ...         ...
564                 0.05623  ...          26.40           166.10      2027.0
565                 0.05533  ...          38.25           155.00      1731.0
566                 0.05648  ...          34.12           126.70      1124.0
567                 0.07016  ...          39.42           184.60      1821.0
568                 0.05884  ...          30.37            59.16       268.6

     worst smoothness  worst compactness  worst concavity  \
0             0.16220            0.66560           0.7119
1             0.12380            0.18660           0.2416
2             0.14440            0.42450           0.4504
3             0.20980            0.86630           0.6869
4             0.13740            0.20500           0.4000
..                ...                ...              ...
564           0.14100            0.21130           0.4107
565           0.11660            0.19220           0.3215
566           0.11390            0.30940           0.3403
567           0.16500            0.86810           0.9387
568           0.08996            0.06444           0.0000

     worst concave points  worst symmetry  worst fractal dimension  target
0                  0.2654          0.4601                  0.11890     0.0
1                  0.1860          0.2750                  0.08902     0.0
2                  0.2430          0.3613                  0.08758     0.0
3                  0.2575          0.6638                  0.17300     0.0
4                  0.1625          0.2364                  0.07678     0.0
..                    ...             ...                      ...     ...
564                0.2216          0.2060                  0.07115     0.0
565                0.1628          0.2572                  0.06637     0.0
566                0.1418          0.2218                  0.07820     0.0
567                0.2650          0.4087                  0.12400     0.0
568                0.0000          0.2871                  0.07039     1.0

[569 rows x 31 columns]
```

In [6]:
```
m1 = np.c_[np.array([1,2,3]), np.array([4,5,6]), np.array([7,8,9])]
m2 = np.r_[np.array([1,2,3]), np.array([4,5,6])]
print(m1)
print('\n', m2)
```

```
[[1 4 7]
 [2 5 8]
 [3 6 9]]

 [1 2 3 4 5 6]
```

In [7]:
```
print(df.shape)
```

```
(569, 31)
```

benign data set has the label 0.0 and malignant data set has label 1.0. Let us divide the data set as training and test

Split the data into training and testing

In [8]:
```
na = np.array(df)
m,n = na.shape
brtr = na[0:m//2,:]
brts = na[m//2:m,:]
```

In [9]:
```
m,n = df.shape
a = np.array(df)
print(a.shape)
print(a[0,])
```

```
(569, 31)
[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
 1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
 6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
 1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
 4.601e-01 1.189e-01 0.000e+00]
```

In [10]:
```
def knn_dist_vote(na,x,k):
    m1,n1 = na.shape
    c1 = 0
    for i in range (k):
        d1 = np.linalg.norm(x[0:n1-1]-na[0,0:n1-1])
        for j in range(1,m1):
            d = np.linalg.norm(x[0:n1-1]-na[j,0:n-1])
            if d < d1:
                d1 = d
                ind = j
        if(na[ind, n1-1] == 1.0):
            c1 += 1
        na = np.delete(na,ind,0)
        m1 -= 1
    if (c1 > k//2):
            return 1.0
    else:
            return 0.0
```

In [11]:
```
def acc_of_classification(trn,tst,k):
    tp = 0
    tn = 0
    fp = 0
    fn = 0
    m1,n1 = brts.shape
    for i in range (m1):
        c = knn_dist_vote(brtr,brts[i,],k)
        if (c == 1):
            if (brts[i,n1-1] == 1):
                tp += 1
        else:
            fp += 1
        if (c == 0):
```

```
            if (brts[i,n1-1] == 0):
                tn += 1
            else:
                fn += 1
    return tp,fp,tn,fn
```

In [12]:
```
tp,fp,tn,fn = acc_of_classification(brtr,brts,6)
print(tp,fp,tn,fn)
```

```
200 80 62 18
```

In [13]:
```
accuracy = (tp+tn)/(tp + fp + tn + fn)
prec = tp/(tp+fp)
recall = tp/(tp+fn)
```

In [14]:
```
print('Accuracy = ', accuracy)
print('Precision = ', prec)
print('Recall = ', recall)
```

```
Accuracy =  0.7277777777777777
Precision =  0.7142857142857143
Recall =  0.9174311926605505
```

In [15]:
```
CM = [[tp,fp],[fn,tn]] # Confusion Matrix
print('\t', CM[0], '\n\t',CM[1])
```

```
        [200, 80]
        [18, 62]
```

Let us plot accuracy vs K

In [16]:
```
kk = np.array([3,5,7,9])
n1 = kk.shape[0]
acc1 = np.zeros(n1,dtype = 'float')
for i in range(n1):
    tp, fp, tn,fn = acc_of_classification(brtr, brts, kk[i])
    acc1[i] = (tp+tn)/(tp+fp+tn+fn)
plt.plot(kk,acc1)
print(acc1)
```

```
[0.70994475 0.74229692 0.74229692 0.75211268]
```