3a)

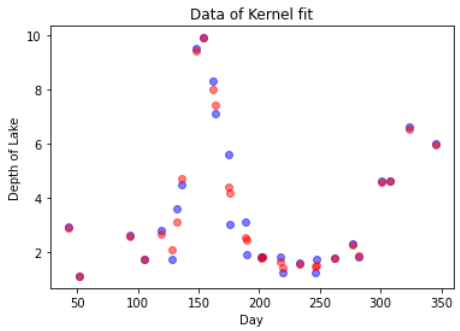
In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

df = pd.read_csv('mendota_secchi_depth.txt', delimiter='\t')
x = df['day_of_year']
y = df['secchi_depth']
sigma = 10
lam = 0.01
n = len(x)
```

In [2]:
```
def KERNEL(X, sigma):
    distsq = np.zeros((n, n),dtype = float)
    for i in range(0, n):
        for j in range(0, n):
            d = np.linalg.norm(X[i] - X[j])
            distsq[i, j] = d**2
    K = np.exp(-distsq/(2*sigma**2))
    return K

def alpha(kernmat, lam):
    alpha = np.linalg.inv(kernmat + lam*np.identity(n))@y
    return alpha
```

In [3]:
```
kernmat = KERNEL(x, sigma)
alphamat = alpha(kernmat, lam)
Yhat = kernmat@alphamat
```

In [5]:
```
plt.plot(x, y, 'bo',label= 'Measured Data', alpha = 0.5)
plt.plot(x, Yhat, 'ro',label= 'Kernel Fit', alpha = 0.5)
plt.xlabel('Day')
plt.ylabel('Depth of Lake')
plt.title('Data of Kernel fit')
```

Out[5]:
```
Text(0.5, 1.0, 'Data of Kernel fit')
```



The resulting fit approximates the measured data very well in the middle of the dataset (days), in the start it overshoots it by a little and in the end it also tapers off the actual data by quite a lot. The parameters thus underfit the data as some notable features are not taken.

3b)

We can use k-fold cross validation to get better values of sigma and lambda. Since we can iterate through sigma values, and find the error at each value of sigma we can use that advantage to compare the errors and find the lowest. The error graph should have a local minima and that is the desired value of sigma for us.

We can run a similar experiment for lamda values, finding the minima of the error graph and thus finding the appropriate lamda.

In [ ]:
```
```

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import loadmat
dataset = loadmat('face_emotion_data.mat')

X, y = dataset['X'], dataset['y']
n, p = np.shape(X)

y[y==-1] = 0  # use 0/1 for labels instead of -1/+1
X = np.hstack((np.ones((n,1)), X))  # append a column of ones

print(X.round(2))
print(X.shape)
print(y.shape)
```

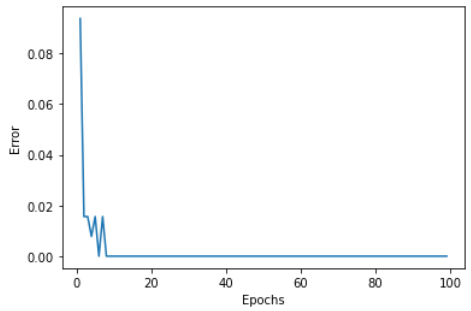
```
[[ 1.   -0.16  0.99 ...  2.01  1.8   2.94]
 [ 1.    1.51  2.43 ...  0.91  1.84  1.68]
 [ 1.   -0.08  0.95 ...  2.4   1.66  2.54]
 ...
 [ 1.    0.05 -0.11 ...  0.21 -1.17 -0.82]
 [ 1.    1.39  0.99 ...  1.73  0.92  0.73]
 [ 1.    1.61  0.36 ...  0.6   0.53 -0.15]]
(128, 10)
(128, 1)
```

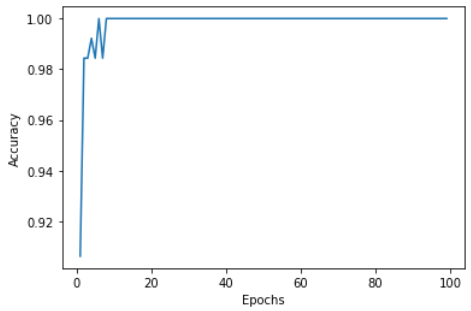
1a)

In [2]:
```
def logsig(_x):
    return 1/(1+np.exp(-_x))

def train(Xb, yb, L):
    n, p = np.shape(Xb)

    V = np.random.randn(M+1, 1)
    W = np.random.randn(p, M)

    for epoch in range(L):
        ind = np.random.permutation(n)
        for i in ind:
            H = logsig(np.hstack((np.ones((1,1)), Xb[[i],:]@W)))
            Yhat = logsig(H@V)
            delta = (Yhat-yb[[i], :])*Yhat*(1-Yhat)
            Vnew = V-alpha*H.T@delta
            gamma = delta@V[1:,:].T*H[:,1:]*(1-H[:,1:])
            Wnew = W - alpha*Xb[[i], :].T@gamma
            V = Vnew
            W =Wnew

    return W, V

def test(Xb, yb, W, V):
    H =logsig(np.hstack((np.ones((np.shape(Xb)[0],1)), Xb@W)))
    Yhat = logsig(H@V)
    # ERROR CALC
    error = np.mean(abs(np.round(Yhat[:,0]-yb[:,0])))
    return error
```

1b)

In [3]:
```
q = np.shape(y)[1]
M = 32 # Number of nodes

alpha = 0.5
L_arr = list(range(1,100))
err_arr = []
accuracy_arr = []

for L in L_arr:
    Wcurr, Vcurr = train(X, y, L)
    error = test(X, y, Wcurr, Vcurr)
    err_arr.append(error)
    accuracy_arr.append(1-error)

plt.figure()
plt.plot(L_arr, err_arr)
plt.xlabel("Epochs")
plt.ylabel("Error")
plt.show()

plt.figure()
plt.plot(L_arr, accuracy_arr)
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.show()
```





Yes after about 20 epochs the error drops to 0. Or accuracy goes to near perfect. This however varies from run to run, but on average is around the 20th mark (based on several runs).

1c)

In [4]:
```
L = 100
err_total = []

for iteration in range(L):
    err_somerun = []
    for i in range(8):
        start = i*16
        end = (i+1)*16

        X_train = np.vstack((X[0:start, :], X[end:-1, :]))
        y_train = np.vstack((y[0:start, :], y[end:-1, :]))

        X_test = X[start:end, :]
        y_test = y[start:end, :]

        V = np.random.randn(M+1, 1)
        W = np.random.randn(p, M)

        W, V = train(X_train, y_train, L)
        err = test(X_test, y_test, W, V)

        err_somerun.append(err)
    print(" run ", iteration, "error rate: ", np.mean(err_somerun), " and accuracy rate: ", 1-np.mean(err_somerun))
    err_total.append(err_somerun)
```

```
run  0 error rate:  0.0390625  and accuracy rate:  0.9609375
run  1 error rate:  0.03125  and accuracy rate:  0.96875
run  2 error rate:  0.0390625  and accuracy rate:  0.9609375
run  3 error rate:  0.0390625  and accuracy rate:  0.9609375
run  4 error rate:  0.03125  and accuracy rate:  0.96875
run  5 error rate:  0.03125  and accuracy rate:  0.96875
run  6 error rate:  0.0390625  and accuracy rate:  0.9609375
run  7 error rate:  0.0390625  and accuracy rate:  0.9609375
run  8 error rate:  0.0234375  and accuracy rate:  0.9765625
run  9 error rate:  0.046875  and accuracy rate:  0.953125
run  10 error rate:  0.03125  and accuracy rate:  0.96875
run  11 error rate:  0.0390625  and accuracy rate:  0.9609375
run  12 error rate:  0.03125  and accuracy rate:  0.96875
run  13 error rate:  0.03125  and accuracy rate:  0.96875
run  14 error rate:  0.0234375  and accuracy rate:  0.9765625
run  15 error rate:  0.0390625  and accuracy rate:  0.9609375
run  16 error rate:  0.0390625  and accuracy rate:  0.9609375
run  17 error rate:  0.0390625  and accuracy rate:  0.9609375
run  18 error rate:  0.03125  and accuracy rate:  0.96875
run  19 error rate:  0.0390625  and accuracy rate:  0.9609375
run  20 error rate:  0.03125  and accuracy rate:  0.96875
run  21 error rate:  0.046875  and accuracy rate:  0.953125
run  22 error rate:  0.0390625  and accuracy rate:  0.9609375
run  23 error rate:  0.03125  and accuracy rate:  0.96875
run  24 error rate:  0.0234375  and accuracy rate:  0.9765625
run  25 error rate:  0.03125  and accuracy rate:  0.96875
run  26 error rate:  0.03125  and accuracy rate:  0.96875
run  27 error rate:  0.015625  and accuracy rate:  0.984375
run  28 error rate:  0.0390625  and accuracy rate:  0.9609375
run  29 error rate:  0.03125  and accuracy rate:  0.96875
run  30 error rate:  0.0390625  and accuracy rate:  0.9609375
run  31 error rate:  0.046875  and accuracy rate:  0.953125
run  32 error rate:  0.0390625  and accuracy rate:  0.9609375
run  33 error rate:  0.0390625  and accuracy rate:  0.9609375
run  34 error rate:  0.0390625  and accuracy rate:  0.9609375
run  35 error rate:  0.046875  and accuracy rate:  0.953125
run  36 error rate:  0.0390625  and accuracy rate:  0.9609375
run  37 error rate:  0.0390625  and accuracy rate:  0.9609375
run  38 error rate:  0.0390625  and accuracy rate:  0.9609375
run  39 error rate:  0.03125  and accuracy rate:  0.96875
run  40 error rate:  0.0390625  and accuracy rate:  0.9609375
run  41 error rate:  0.0390625  and accuracy rate:  0.9609375
run  42 error rate:  0.0390625  and accuracy rate:  0.9609375
run  43 error rate:  0.03125  and accuracy rate:  0.96875
run  44 error rate:  0.046875  and accuracy rate:  0.953125
run  45 error rate:  0.0390625  and accuracy rate:  0.9609375
run  46 error rate:  0.046875  and accuracy rate:  0.953125
run  47 error rate:  0.0390625  and accuracy rate:  0.9609375
run  48 error rate:  0.046875  and accuracy rate:  0.953125
run  49 error rate:  0.046875  and accuracy rate:  0.953125
run  50 error rate:  0.03125  and accuracy rate:  0.96875
run  51 error rate:  0.03125  and accuracy rate:  0.96875
run  52 error rate:  0.0390625  and accuracy rate:  0.9609375
run  53 error rate:  0.03125  and accuracy rate:  0.96875
run  54 error rate:  0.046875  and accuracy rate:  0.953125
run  55 error rate:  0.0390625  and accuracy rate:  0.9609375
run  56 error rate:  0.0390625  and accuracy rate:  0.9609375
run  57 error rate:  0.0390625  and accuracy rate:  0.9609375
run  58 error rate:  0.0390625  and accuracy rate:  0.9609375
run  59 error rate:  0.03125  and accuracy rate:  0.96875
run  60 error rate:  0.0390625  and accuracy rate:  0.9609375
run  61 error rate:  0.046875  and accuracy rate:  0.953125
run  62 error rate:  0.0390625  and accuracy rate:  0.9609375
run  63 error rate:  0.0390625  and accuracy rate:  0.9609375
run  64 error rate:  0.0390625  and accuracy rate:  0.9609375
run  65 error rate:  0.0390625  and accuracy rate:  0.9609375
run  66 error rate:  0.046875  and accuracy rate:  0.953125
run  67 error rate:  0.0390625  and accuracy rate:  0.9609375
run  68 error rate:  0.046875  and accuracy rate:  0.953125
run  69 error rate:  0.03125  and accuracy rate:  0.96875
run  70 error rate:  0.0546875  and accuracy rate:  0.9453125
run  71 error rate:  0.0390625  and accuracy rate:  0.9609375
run  72 error rate:  0.03125  and accuracy rate:  0.96875
run  73 error rate:  0.0390625  and accuracy rate:  0.9609375
run  74 error rate:  0.0390625  and accuracy rate:  0.9609375
run  75 error rate:  0.03125  and accuracy rate:  0.96875
run  76 error rate:  0.0390625  and accuracy rate:  0.9609375
```

```
run  77 error rate:  0.0390625  and accuracy rate:  0.9609375
run  78 error rate:  0.046875  and accuracy rate:  0.953125
run  79 error rate:  0.0390625  and accuracy rate:  0.9609375
run  80 error rate:  0.0390625  and accuracy rate:  0.9609375
run  81 error rate:  0.0390625  and accuracy rate:  0.9609375
run  82 error rate:  0.046875  and accuracy rate:  0.953125
run  83 error rate:  0.0234375  and accuracy rate:  0.9765625
run  84 error rate:  0.0390625  and accuracy rate:  0.9609375
run  85 error rate:  0.0390625  and accuracy rate:  0.9609375
run  86 error rate:  0.0390625  and accuracy rate:  0.9609375
run  87 error rate:  0.0390625  and accuracy rate:  0.9609375
run  88 error rate:  0.03125  and accuracy rate:  0.96875
run  89 error rate:  0.03125  and accuracy rate:  0.96875
run  90 error rate:  0.0390625  and accuracy rate:  0.9609375
run  91 error rate:  0.03125  and accuracy rate:  0.96875
run  92 error rate:  0.03125  and accuracy rate:  0.96875
run  93 error rate:  0.046875  and accuracy rate:  0.953125
run  94 error rate:  0.0546875  and accuracy rate:  0.9453125
run  95 error rate:  0.0390625  and accuracy rate:  0.9609375
run  96 error rate:  0.046875  and accuracy rate:  0.953125
run  97 error rate:  0.03125  and accuracy rate:  0.96875
run  98 error rate:  0.0234375  and accuracy rate:  0.9765625
run  99 error rate:  0.0390625  and accuracy rate:  0.9609375
```

No. Even over 100 trials, the error rate was not 0. Thus we infer that the error rate for cross validation does not go to 0. This makes sense intuitively as the hold out set of the data could have features that the test set does not, thus leading to errors.

In [5]:
```
print("mean error rate over 100 iterations", np.mean(err_total))
```

```
mean error rate over 100 iterations 0.037578125
```

Thus we cannot achieve perfect accuracy. There is always some error.

In [ ]:

2a)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import loadmat
dataset = loadmat('face_emotion_data.mat')

X, y = dataset['X'], dataset['y']
n, p = np.shape(X)

y[y==-1] = 0  # use 0/1 for labels instead of -1/+1
X = np.hstack((np.ones((n,1)), X))  # append a column of ones

def kernel(X, sigma):
    n, p = np.shape(X)

    distsquared = np.zeros((n,n ), dtype=float)

    for i in range(n):
        for j in range(n):
            dist = np.linalg.norm(X[i,:]-X[j,:])
            distsquared[i,j]=dist*dist

    K = np.exp(-1*distsquared/(2*sigma**2))
    return K

def train(X, y, lam, sigma):

    K = kernel(X,sigma)

    return np.linalg.inv(K+lam*np.identity(len(K)))@y


def errorCrossValidation(Xb, yb, W, V):
    H =logsig(np.hstack((np.ones((np.shape(Xb)[0],1)), Xb@W)))
    Yhat = logsig(H@V)
    # ERROR CALC

    error = np.mean(abs(np.round(Yhat[:,0]-yb[:,0])))
    return error
```

sigmas = np.linspace(0, 50, 100) print(sigmas) errors = []

for sigma in sigmas: train_ind = np.arange(128) alpha = train(X, y, 0.5, sigma) errors.append(error) print(error)

plt.figure() plt.plot(sigmas, errors)

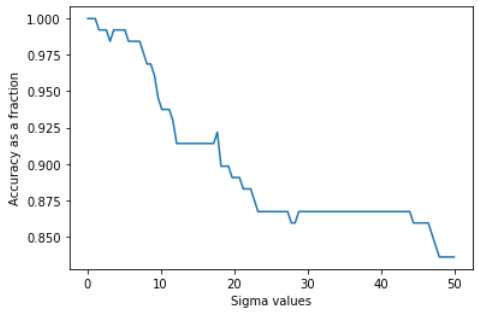
In [2]:
```
def test(X, y, Xv, yv, alpha):
    numAlpha = len(alpha)

    K = kernel(X,sigma)

    y_hat = K@alpha
    y_hat_thresh = np.sign(y_hat)
    y_hat_thresh[y_hat_thresh>0.5] = 1
    y_hat_thresh[y_hat_thresh<=0.5] = 0

    error = np.sum(np.abs(y_hat_thresh-yv))

    return error/n
```

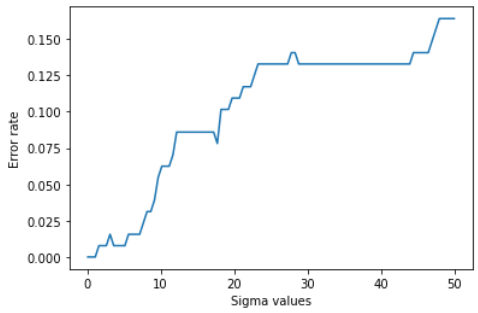
Type *Markdown* and LaTeX: α^2

2b)

In [3]:
```
sigmas = np.linspace(0.1, 50, 100)

errors = []
accuracies = []
lamda = 0.5
for sigma in sigmas:
    K = kernel(X,sigma)
    alpha = train(X,y, lamda, sigma)
    Yhat = abs(np.round(K@alpha))

    correct = [1 if m[0]==m[1] else 0 for m in np.hstack((Yhat, y))]

    accuracy = sum(correct)/len(correct)

    accuracies.append(accuracy)
    errors.append(1-accuracy)

plt.figure()
plt.xlabel("Sigma values")
plt.ylabel("Accuracy as a fraction")
plt.plot(sigmas, accuracies)

plt.figure()
plt.xlabel("Sigma values")
plt.ylabel("Error rate")
plt.plot(sigmas, errors)
```

Out[3]:
```
[<matplotlib.lines.Line2D at 0x2a16ebfabe0>]
```





2c)

In [4]:
```
L = 100
lamda = 0.5
sigma = 0.1
err_total = []
sigmas = np.linspace(0.1, 50, 100)
for sigma in sigmas:
    err_somerun = []
    err_somerun_Train = []
    err_somerun_Test = []
    for i in range(7):
        start = i*16
        end = (i+1)*16

        X_train = np.vstack((X[0:start, :], X[end:-1, :]))
        y_train = np.vstack((y[0:start, :], y[end:-1, :]))

        X_test = X[start:end, :]
        y_test = y[start:end, :]

        alpha1 = train(X_train, y_train, lamda, sigma)
        alpha2 = train(X_test, y_test, lamda, sigma)

        K = kernel(X,sigma)

        Yhat1 = abs(np.round(K@np.vstack((alpha1,np.ones((17, alpha1.shape[1]))))))

        correct_vec = [1 if m[0]==m[1] else 0 for m in np.hstack((X.T@Yhat1, X_test.T@y_test))]
        accuracy = sum(correct_vec)/len(correct_vec)

        accuracies.append(accuracy)
        errors.append(1-accuracy)


        err_somerun.append(errors)

    print(" run with sigma: ", sigma, "error rate: ", np.mean(err_somerun), " and accuracy rate: ", 1-np.mean(err_somerun))
    err_total.append(err_somerun)

print("mean error rate over 100 iterations", np.mean(err_total))
```

```
 run with sigma:  0.1 error rate:  0.16004672897196262  and accuracy rate:  0.8399532710280374
 run with sigma:  0.604040404040404 error rate:  0.21162280701754385  and accuracy rate:  0.7883771929824561
 run with sigma:  1.1080808080808082 error rate:  0.2572314049586777  and accuracy rate:  0.7427685950413223
 run with sigma:  1.6121212121212123 error rate:  0.2978515625  and accuracy rate:  0.7021484375
 run with sigma:  2.1161616161616164 error rate:  0.33425925925925926  and accuracy rate:  0.6657407407407407
 run with sigma:  2.6202020202020204 error rate:  0.3670774647887324  and accuracy rate:  0.6329225352112676
 run with sigma:  3.1242424242424245 error rate:  0.39681208053691275  and accuracy rate:  0.6031879194630873
 run with sigma:  3.6282828282828286 error rate:  0.4238782051282051  and accuracy rate:  0.5761217948717949
 run with sigma:  4.132323232323232 error rate:  0.4486196319018405  and accuracy rate:  0.5513803680981595
 run with sigma:  4.636363636363637 error rate:  0.4713235294117647  and accuracy rate:  0.5286764705882353
 run with sigma:  5.14040404040404 error rate:  0.4922316384180791  and accuracy rate:  0.5077683615819208
 run with sigma:  5.644444444444444 error rate:  0.5115489130434783  and accuracy rate:  0.48845108695652173
 run with sigma:  6.148484848484848 error rate:  0.5294502617801047  and accuracy rate:  0.4705497382198953
 run with sigma:  6.652525252525253 error rate:  0.5460858585858586  and accuracy rate:  0.45391414141414144
 run with sigma:  7.156565656565657 error rate:  0.5615853658536586  and accuracy rate:  0.4384146341463414
 run with sigma:  7.66060606060606 error rate:  0.5760613207547169  and accuracy rate:  0.42393867924528306
 run with sigma:  8.164646464646465 error rate:  0.5896118721461188  and accuracy rate:  0.41038812785388123
 run with sigma:  8.66868686868687 error rate:  0.6023230088495575  and accuracy rate:  0.3976769911504425
 run with sigma:  9.172727272727274 error rate:  0.6142703862660944  and accuracy rate:  0.3857296137339056
 run with sigma:  9.676767676767676 error rate:  0.6255208333333333  and accuracy rate:  0.3744791666666667
 run with sigma:  10.180808080808081 error rate:  0.6361336032388664  and accuracy rate:  0.36386639676113364
 run with sigma:  10.684848484848485 error rate:  0.6461614173228346  and accuracy rate:  0.3538385826771654
 run with sigma:  11.188888888888888 error rate:  0.6556513409961686  and accuracy rate:  0.3443486590038314
 run with sigma:  11.692929292929293 error rate:  0.6646455223880597  and accuracy rate:  0.33535447761194026
 run with sigma:  12.196969696969697 error rate:  0.6731818181818182  and accuracy rate:  0.3268181818181818
 run with sigma:  12.701010101010102 error rate:  0.6812943262411347  and accuracy rate:  0.31870567375886527
 run with sigma:  13.205050505050506 error rate:  0.6890138408304498  and accuracy rate:  0.3109861591695502
 run with sigma:  13.709090909090909 error rate:  0.6963682432432432  and accuracy rate:  0.3036317567567568
 run with sigma:  14.213131313131314 error rate:  0.7033828382838284  and accuracy rate:  0.2966171617161716
 run with sigma:  14.717171717171718 error rate:  0.7100806451612903  and accuracy rate:  0.2899193548387097
 run with sigma:  15.22121212121212 error rate:  0.7164826498422713  and accuracy rate:  0.2835173501577287
 run with sigma:  15.725252525252525 error rate:  0.722608024691358  and accuracy rate:  0.277391975308642
 run with sigma:  16.22929292929293 error rate:  0.7284743202416919  and accuracy rate:  0.2715256797583081
 run with sigma:  16.733333333333334 error rate:  0.7340976331360947  and accuracy rate:  0.26590236686390534
 run with sigma:  17.23737373737374 error rate:  0.7394927536231884  and accuracy rate:  0.26050724637681155
 run with sigma:  17.741414141414143 error rate:  0.7446732954545454  and accuracy rate:  0.2553267045454546
 run with sigma:  18.24545454545455 error rate:  0.7496518105849582  and accuracy rate:  0.2503481894150418
 run with sigma:  18.749494949494952 error rate:  0.7544398907103825  and accuracy rate:  0.24556010928961747
 run with sigma:  19.253535353535355 error rate:  0.7590482573726541  and accuracy rate:  0.24095174262734587
 run with sigma:  19.75757575757576 error rate:  0.7634868421052632  and accuracy rate:  0.23651315789473681
 run with sigma:  20.261616161616164 error rate:  0.7677648578811369  and accuracy rate:  0.2322351421188631
 run with sigma:  20.765656565656567 error rate:  0.7718908629441624  and accuracy rate:  0.22810913705583757
 run with sigma:  21.269696969696973 error rate:  0.7758728179551122  and accuracy rate:  0.2241271820448878
 run with sigma:  21.773737373737376 error rate:  0.7797181372549019  and accuracy rate:  0.2202818627450981
 run with sigma:  22.27777777777778 error rate:  0.783433734939759  and accuracy rate:  0.21656626506024101
 run with sigma:  22.781818181818185 error rate:  0.7870260663507109  and accuracy rate:  0.21297393364928907
 run with sigma:  23.285858585858588 error rate:  0.7905011655011654  and accuracy rate:  0.20949883449883455
 run with sigma:  23.789898989898994 error rate:  0.7938646788990825  and accuracy rate:  0.20613532110091748
 run with sigma:  24.293939393939397 error rate:  0.7971218961625283  and accuracy rate:  0.20287810383747173
 run with sigma:  24.7979797979798 error rate:  0.8002777777777778  and accuracy rate:  0.19972222222222225
 run with sigma:  25.302020202020206 error rate:  0.8033369803063457  and accuracy rate:  0.1966630196936543
 run with sigma:  25.80606060606061 error rate:  0.8063038793103449  and accuracy rate:  0.19369612068965514
 run with sigma:  26.310101010101015 error rate:  0.8091825902335457  and accuracy rate:  0.19081740976645434
 run with sigma:  26.814141414141417 error rate:  0.8119769874476988  and accuracy rate:  0.18802301255230125
 run with sigma:  27.31818181818182 error rate:  0.8146907216494845  and accuracy rate:  0.18530927835051547
 run with sigma:  27.822222222222226 error rate:  0.8173272357723578  and accuracy rate:  0.18267276422764223
 run with sigma:  28.32626262626263 error rate:  0.8198897795591182  and accuracy rate:  0.18011022044088176
 run with sigma:  28.830303030303032 error rate:  0.8223814229249012  and accuracy rate:  0.17761857707509876
 run with sigma:  29.334343434343438 error rate:  0.8248050682261209  and accuracy rate:  0.17519493177387913
 run with sigma:  29.83838383838384 error rate:  0.8271634615384615  and accuracy rate:  0.17283653846153846
 run with sigma:  30.342424242424244 error rate:  0.8294592030360531  and accuracy rate:  0.17054079696394686
 run with sigma:  30.84646464646465 error rate:  0.8316947565543071  and accuracy rate:  0.16830524344569286
```

```
run with sigma:  31.350505050505053 error rate:  0.8338724584103512  and accuracy rate:  0.16612754158964882
run with sigma:  31.85454545454546 error rate:  0.8359945255474452  and accuracy rate:  0.16400547445255476
run with sigma:  32.35858585858586 error rate:  0.8380630630630631  and accuracy rate:  0.1619369369369369
run with sigma:  32.862626262626264 error rate:  0.8400800711743772  and accuracy rate:  0.15991992882562278
run with sigma:  33.36666666666667 error rate:  0.8420474516695958  and accuracy rate:  0.15795254833040417
run with sigma:  33.87070707070708 error rate:  0.8439670138888888  and accuracy rate:  0.15603298611111116
run with sigma:  34.37474747474748 error rate:  0.8458404802744426  and accuracy rate:  0.15415951972555741
run with sigma:  34.87878787878788 error rate:  0.8476694915254237  and accuracy rate:  0.1523305084745763
run with sigma:  35.382828282828285 error rate:  0.8494556113902848  and accuracy rate:  0.1505443886097152
run with sigma:  35.88686868686869 error rate:  0.8512003311258278  and accuracy rate:  0.14879966887417218
run with sigma:  36.3909090909091 error rate:  0.8529050736497545  and accuracy rate:  0.14709492635024546
run with sigma:  36.8949494949495 error rate:  0.8545711974110033  and accuracy rate:  0.1454288025889967
run with sigma:  37.3989898989899 error rate:  0.8562  and accuracy rate:  0.14380000000000004
run with sigma:  37.903030303030306 error rate:  0.8577927215189873  and accuracy rate:  0.14220727848101267

run with sigma:  38.40707070707071 error rate:  0.8593505477308294  and accuracy rate:  0.14064945226917058
run with sigma:  38.91111111111111 error rate:  0.860874613003096  and accuracy rate:  0.13912538699690402
run with sigma:  39.41515151515152 error rate:  0.8623660030627871  and accuracy rate:  0.1376339969372129
run with sigma:  39.919191919191924 error rate:  0.8638257575757575  and accuracy rate:  0.13617424242424248
run with sigma:  40.42323232323233 error rate:  0.8652548725637181  and accuracy rate:  0.13474512743628186
run with sigma:  40.92727272727273 error rate:  0.8666543026706232  and accuracy rate:  0.13334569732937684
run with sigma:  41.43131313131313 error rate:  0.8680249632892805  and accuracy rate:  0.13197503671071953
run with sigma:  41.93535353535354 error rate:  0.8693677325581395  and accuracy rate:  0.13063226744186052
run with sigma:  42.439393939393945 error rate:  0.87068345323741  and accuracy rate:  0.12931654676258997
run with sigma:  42.94343434343435 error rate:  0.8719729344729344  and accuracy rate:  0.12802706552706555
run with sigma:  43.44747474747475 error rate:  0.8732369534555712  and accuracy rate:  0.12676304654442883
run with sigma:  43.95151515151515 error rate:  0.8744762569832403  and accuracy rate:  0.12552374301675973
run with sigma:  44.455555555555556 error rate:  0.8756915629322268  and accuracy rate:  0.12430843706777317
run with sigma:  44.959595959595966 error rate:  0.8768835616438356  and accuracy rate:  0.12311643835616437
run with sigma:  45.46363636363637 error rate:  0.8780529172320217  and accuracy rate:  0.12194708276797828
run with sigma:  45.96767676767677 error rate:  0.8792002688172043  and accuracy rate:  0.12079973118279574
run with sigma:  46.471717171717174 error rate:  0.8803262316910786  and accuracy rate:  0.1196737683089214
run with sigma:  46.97575757575758 error rate:  0.8814313984168866  and accuracy rate:  0.11856860158311344
run with sigma:  47.479797979797986 error rate:  0.882516339869281  and accuracy rate:  0.11748366013071898
run with sigma:  47.98383838383839 error rate:  0.8835816062176166  and accuracy rate:  0.11641839378238339
run with sigma:  48.48787878787879 error rate:  0.8846277278562259  and accuracy rate:  0.1153722721437741
run with sigma:  48.991919191919195 error rate:  0.8856552162849872  and accuracy rate:  0.11434478371501278
run with sigma:  49.4959595959596 error rate:  0.8866645649432535  and accuracy rate:  0.11333543505674648
run with sigma:  50.0 error rate:  0.88765625  and accuracy rate:  0.11234374999999996
mean error rate over 100 iterations 0.88765625
```

We cannot achieve perfect accuracy. Even with 100 trials, the error rate was not 0. Thus we infer that the error rate for cross validation does not go to 0. This makes sense intuitively as the hold out set of the data could have features that the test set does not, thus leading to errors.

In [ ]: