In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold
#1.Reads in the data, ignoring the first row (header) and first column (index).
data = pd.read_csv('x06Simple.csv')
y = np.array(data)
y = y[:,1:]
s3 = []
s5 = []
s20 = []
N = []
x1 = np.ones((20,1))
mean_ave = np.ones((20,1))
std_ave = np.ones((20,1))
y = data.to_numpy()
y = y[:,1:]
size1 = np.shape(y)[0]
s = [3,5,20,size1]

for i in s:
    #20 times does the following:
    for j in range(20):
        #For i= 1 to S
        for k in range(i):

            np.random.seed(j)
            y = data.to_numpy()
            y = y[:,1:]
            #Randomizes the data
            np.random.shuffle(y)
            x = y

            #Creates S folds.
            x = np.array_split(x,i)
            matfinal = []
            matfinal =np.ones((i,3))

            #Select fold i (= k) as your testing data and the remaining (S - 1) folds as your training data
            mat_test = x[k]
```

```
        mat=np.ma.array(x,mask=False)
        mat.mask[k] = True
        mat = mat.compressed()
        mat = np.array(mat)

        if(i == size1):
            matfinal = np.reshape(mat,(43,3))
        else:
            for k in mat:

                matfinal = np.concatenate((matfinal, k), axis=0)

            matfinal = matfinal[i:,:]
        Ans = matfinal[:,2]
        matfinal = matfinal[:,0:2]
        TAns = mat_test[:,2]
        test = mat_test[:,0:2]

        #Standardizes the data (except for the last column of course) based on the training data
        mean2 = np.mean(matfinal,axis=0)
        std2 = np.std(matfinal, axis=0,ddof=1)

        matfinal = (matfinal-mean2)/std2
        test = (test-mean2)/std2
        shape = np.shape(matfinal)
        one = np.ones((shape[0],1))
        matfinal = np.concatenate((one, matfinal), axis=1)
        shape = np.shape(test)
        one = np.ones((shape[0],1))
        test = np.concatenate((one, test), axis=1)

        #Train a closed-form linear regression model
        theta = np.linalg.inv(matfinal.T @ matfinal) @ matfinal.T @ Ans
        y1 = test@theta

        #Compute the squared error for each sample in the current testing fold
        error = mean_squared_error(y1,TAns)
        #You should now have N squared errors. Compute the RMSE for these.
        error_root = np.sqrt(error)
        if(i ==3):
            s3.append(error_root)
        if(i ==5):
```

```
                s5.append(error_root)
            if(i ==20):
                s20.append(error_root)
            if(i == size1):
                N.append(error_root)

#You should now have 20 RMSE values. Compute the mean and standard deviation of these. The former should give us
#better “overall” mean, whereas the latter should give us feel for the variance of the models that were created.

#The average and standard deviation of the root mean squared error for S = 3 over the 20 different seed values..
print("mean and std of all S=3")
mean3 = np.mean(s3,axis=0)
std3 = np.std(s3, axis=0,ddof=1)
print(mean3,std3)

#The average and standard deviation of the root mean squared error for S = 5 over the 20 different seed values.
print("mean and std of all S=5")
mean5 = np.mean(s5,axis=0)
std5 = np.std(s5, axis=0,ddof=1)
print(mean5,std5)

#The average and standard deviation of the root mean squared error for S = 20 over 20 different seed values.
print("mean and std of all S=20")
mean20 = np.mean(s20,axis=0)
std20 = np.std(s20, axis=0,ddof=1)
print(mean20,std20)

#The average and standard deviation of the root mean squared error for S = N (where N is the number of samples)
#over 20 different seed values. This is basically leave-one-out cross- validation.
print("mean and std of all S=N")
meanN = np.mean(N,axis=0)
stdN = np.std(N, axis=0,ddof=1)
print(meanN,stdN)
```

```
mean and std of all S=3
613.0985569002445 114.3532421630119
mean and std of all S=5
```

```
602.393598211715 128.8843428065632
mean and std of all S=20
567.6287950180362 283.8600037719564
mean and std of all S=N
496.4487056300912 380.17838776606726
```

In [ ]: