```
def getJenksBreaks( dataList, numClass ):

	dataList.sort()

	mat1 = []
	for i in range(0,len(dataList)+1):
		temp = []
		for j in range(0,numClass+1):
			temp.append(0)
		mat1.append(temp)

	mat2 = []
	for i in range(0,len(dataList)+1):
		temp = []
		for j in range(0,numClass+1):
			temp.append(0)
		mat2.append(temp)

	for i in range(1,numClass+1):
		mat1[1][i] = 1
		mat2[1][i] = 0
		for j in range(2,len(dataList)+1):
			mat2[j][i] = float('inf')

	v = 0.0
	for l in range(2,len(dataList)+1):
		s1 = 0.0
		s2 = 0.0
		w = 0.0
		for m in range(1,l+1):
			i3 = l - m + 1

			val = float(dataList[i3-1])

			s2 += val * val
			s1 += val

			w += 1
			v = s2 - (s1 * s1) / w
			i4 = i3 - 1

			if i4 != 0:
				for j in range(2,numClass+1):
					if mat2[l][j] >= (v + mat2[i4][j - 1]):
						mat1[l][j] = i3
						mat2[l][j] = v + mat2[i4][j - 1]
		mat1[l][1] = 1
		mat2[l][1] = v

	k = len(dataList)
	kclass = []
	for i in range(0,numClass+1):
		kclass.append(0)

	kclass[numClass] = float(dataList[len(dataList) - 1])

	countNum = numClass
	while countNum >= 2:
```

```
            #print "rank = " + str(mat1[k][countNum])
            id = int((mat1[k][countNum]) - 2)
            #print "val = " + str(dataList[id])

            kclass[countNum - 1] = dataList[id]
            k = int((mat1[k][countNum] - 1))
            countNum -= 1

      return kclass

def getGVF( dataList, numClass ):
        """
        The Goodness of Variance Fit (GVF) is found by taking the
            difference between the squared deviations
        from the array mean (SDAM) and the squared deviations from the
            class means (SDCM), and dividing by the SDAM
        """
        breaks = getJenksBreaks(dataList, numClass)
        dataList.sort()
        listMean = sum(dataList)/len(dataList)
        print listMean
        SDAM = 0.0
        for i in range(0,len(dataList)):
                sqDev = (dataList[i] - listMean)**2
                SDAM += sqDev

        SDCM = 0.0
        for i in range(0,numClass):
                if breaks[i] == 0:
                        classStart = 0
                else:
                        classStart = dataList.index(breaks[i])
                        classStart += 1
                classEnd = dataList.index(breaks[i+1])

                classList = dataList[classStart:classEnd+1]

                classMean = sum(classList)/len(classList)
                print classMean
                preSDCM = 0.0
                for j in range(0,len(classList)):
                        sqDev2 = (classList[j] - classMean)**2
                        preSDCM += sqDev2

                SDCM += preSDCM

        return (SDAM - SDCM)/SDAM
```