ธนกิจ เหล่ายาวิระ
600632014

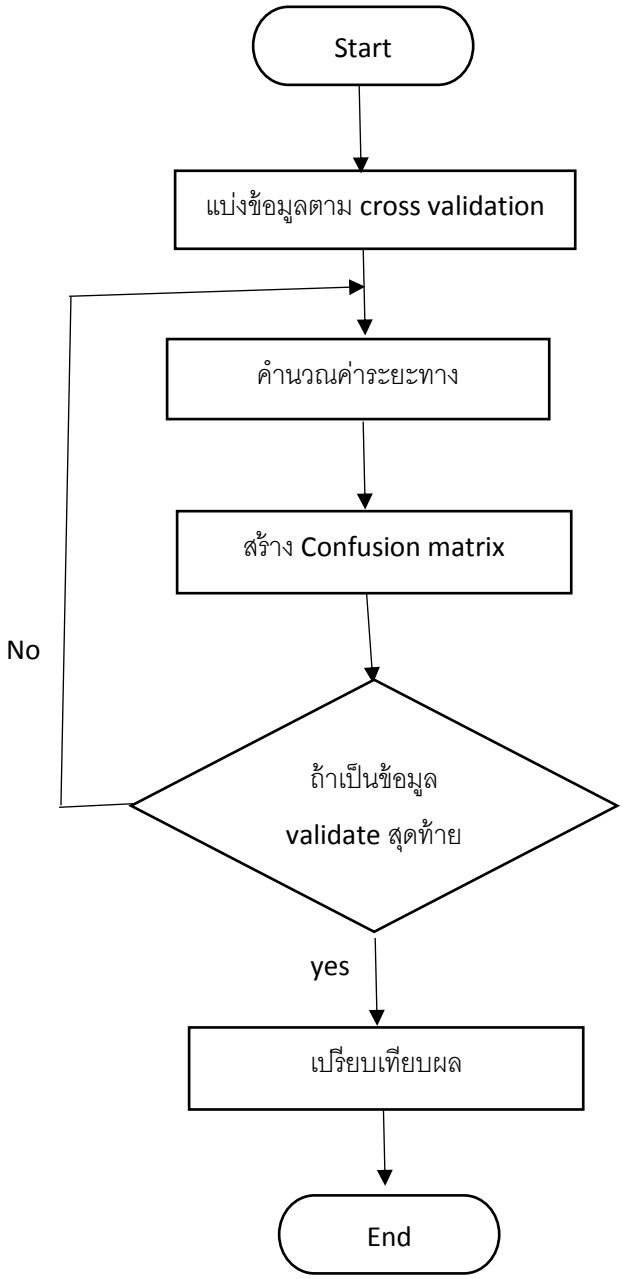
Computer Assignment 2.1

1.รายละเอียดของทฤษฎีหรือวิธีการต่าง ๆ ที่ใช้

1. $D_{Euclidean}(Point\ \ Test, Point\ \ Train) = \sqrt{(F1_{test} - F1_{train})^2 + (F2_{test} - F2_{train})^2}$;Euclidean Distant

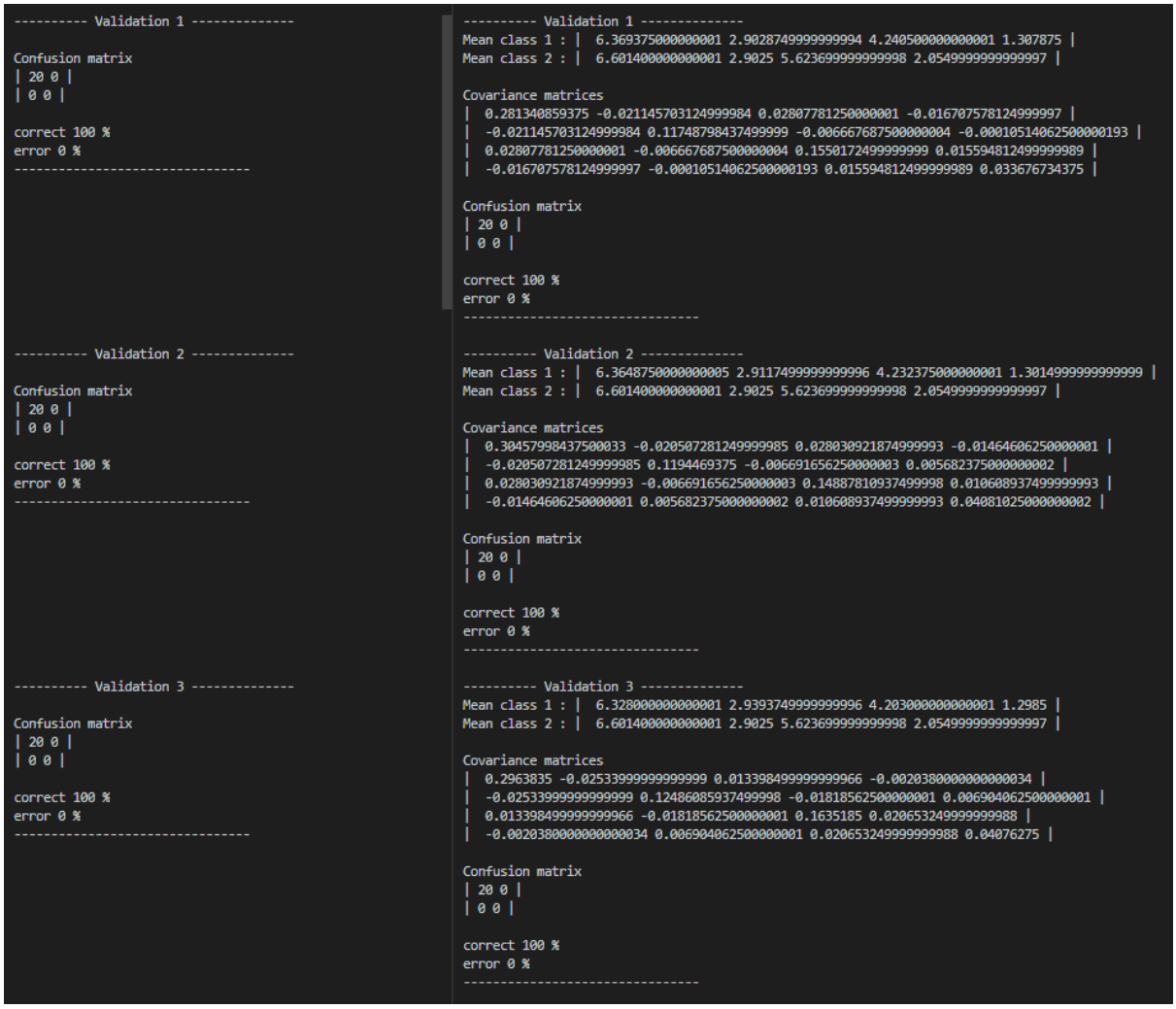
2. ใช้ NodeJS ในการพัฒนาซอฟท์แวร์

2.Flowchart



3. ผลการทดลอง

เมื่อทดลองกับ KNN k = 4 กับ Bayes Classifier 4 feature



```
---------- Validation 1 --------------

Confusion matrix
| 20 0 |
| 0 0 |

correct 100 %
error 0 %
--------------------------------
```

```
---------- Validation 1 --------------
Mean class 1 : |  6.369375000000001 2.9028749999999994 4.240500000000001 1.307875 |
Mean class 2 : |  6.601400000000001 2.9025 5.623699999999998 2.0549999999999997 |

Covariance matrices
|  0.281340859375 -0.021145703124999984 0.02807781250000001 -0.016707578124999997 |
|  -0.021145703124999984 0.11748798437499999 -0.006667687500000004 -0.0001051406250000193 |
|  0.02807781250000001 -0.006667687500000004 0.1550172499999999 0.015594812499999989 |
|  -0.016707578124999997 -0.0001051406250000193 0.015594812499999989 0.033676734375 |

Confusion matrix
| 20 0 |
| 0 0 |

correct 100 %
error 0 %
--------------------------------
```

```
---------- Validation 2 --------------

Confusion matrix
| 20 0 |
| 0 0 |

correct 100 %
error 0 %
--------------------------------
```

```
---------- Validation 2 --------------
Mean class 1 : |  6.3648750000000005 2.9117499999999996 4.232375000000001 1.3014999999999999 |
Mean class 2 : |  6.601400000000001 2.9025 5.623699999999998 2.0549999999999997 |

Covariance matrices
|  0.30457998437500033 -0.020507281249999985 0.028030921874999993 -0.01464606250000001 |
|  -0.020507281249999985 0.1194469375 -0.006691656250000003 0.005682375000000002 |
|  0.028030921874999993 -0.006691656250000003 0.14887810937499998 0.010608937499999993 |
|  -0.01464606250000001 0.005682375000000002 0.010608937499999993 0.04081025000000002 |

Confusion matrix
| 20 0 |
| 0 0 |

correct 100 %
error 0 %
--------------------------------
```

```
---------- Validation 3 --------------

Confusion matrix
| 20 0 |
| 0 0 |

correct 100 %
error 0 %
--------------------------------
```

```
---------- Validation 3 --------------
Mean class 1 : |  6.328000000000001 2.9393749999999996 4.203000000000001 1.2985 |
Mean class 2 : |  6.601400000000001 2.9025 5.623699999999998 2.0549999999999997 |

Covariance matrices
|  0.2963835 -0.02533999999999999 0.013398499999999966 -0.0020380000000000034 |
|  -0.02533999999999999 0.12486085937499998 -0.01818562500000001 0.006904062500000001 |
|  0.013398499999999966 -0.01818562500000001 0.1635185 0.020653249999999988 |
|  -0.0020380000000000034 0.006904062500000001 0.020653249999999988 0.04076275 |

Confusion matrix
| 20 0 |
| 0 0 |

correct 100 %
error 0 %
--------------------------------
```

```
---------- Validation 4 --------------

Confusion matrix
| 19 1 |
| 0 0 |

correct 95 %
error 5 %
--------------------------------
```

```
---------- Validation 4 --------------
Mean class 1 : |  6.328499999999999 2.9276249999999995 4.207375000000001 1.295125 |
Mean class 2 : |  6.601400000000001 2.9025 5.623699999999998 2.0549999999999997 |

Covariance matrices
|  0.3060902499999999 -0.011257312499999993 0.025039812499999998 -0.014812312500000006 |
|  -0.011257312499999993 0.105040609375 -0.006343734375000009 0.007082171874999998 |
|  0.025039812499999998 -0.006343734375000009 0.14476685937500003 0.013334703124999998 |
|  -0.014812312500000006 0.007082171874999998 0.013334703124999998 0.04043498437500003 |

Confusion matrix
| 19 1 |
| 0 0 |

correct 95 %
error 5 %
--------------------------------
```

```
---------- Validation 5 --------------

Confusion matrix
| 18 2 |
| 0 0 |

correct 90 %
error 10 %
--------------------------------
```

```
---------- Validation 5 --------------
Mean class 1 : |  6.36125 2.9268749999999994 4.194749999999999 1.271 |
Mean class 2 : |  6.601400000000001 2.9025 5.623699999999998 2.0549999999999997 |

Covariance matrices
|  0.33078593749999996 -0.017017343749999997 0.034995312500000014 -0.009652500000000012 |
|  -0.017017343749999997 0.09231648437499998 0.00115109375 0.009499374999999997 |
|  0.034995312500000014 0.00115109375 0.1448399375 0.013088999999999993 |
|  -0.009652500000000012 0.009499374999999997 0.013088999999999993 0.037364 |

Confusion matrix
| 18 2 |
| 0 0 |

correct 90 %
error 10 %
--------------------------------
```

```
---------- Validation 6 --------------

Confusion matrix
| 0 0 |
| 1 19 |

correct 95 %
error 5 %
--------------------------------
```

```
---------- Validation 6 --------------
Mean class 1 : |  6.350399999999999 2.9216999999999986 4.2156 1.2948 |
Mean class 2 : |  6.636374999999997 2.903250000000001 5.639375000000001 2.0322500000000003 |

Covariance matrices
|  0.3041698400000001 -0.01924767999999999 0.026098759999999985 -0.01157192000000001 |
|  -0.01924767999999999 0.11199610999999994 -0.007550520000000006 0.005738839999999999 |
|  0.026098759999999985 -0.007550520000000006 0.15171663999999996 0.014833120000000005 |
|  -0.01157192000000001 0.005738839999999999 0.014833120000000005 0.03876896000000001 |

Confusion matrix
| 0 0 |
| 1 19 |

correct 95 %
error 5 %
--------------------------------
```

```
---------- Validation 7 --------------

Confusion matrix
| 0 0 |
| 0 20 |

correct 100 %
error 0 %
--------------------------------
```

```
---------- Validation 7 --------------
Mean class 1 : |  6.350399999999999 2.9216999999999986 4.2156 1.2948 |
Mean class 2 : |  6.599874999999997 2.902000000000001 5.626500000000002 2.0653750000000004 |

Covariance matrices
|  0.3041698400000001 -0.01924767999999999 0.026098759999999985 -0.01157192000000001 |
|  -0.01924767999999999 0.11199610999999994 -0.007550520000000006 0.005738839999999999 |
|  0.026098759999999985 -0.007550520000000006 0.151716639999999996 0.014833120000000005 |
|  -0.01157192000000001 0.005738839999999999 0.014833120000000005 0.03876896000000001 |

Confusion matrix
| 0 0 |
| 0 20 |

correct 100 %
error 0 %
--------------------------------
```

```
---------- Validation 8 --------------

Confusion matrix
| 0 0 |
| 0 20 |

correct 100 %
error 0 %
--------------------------------
```

```
---------- Validation 8 --------------
Mean class 1 : |  6.350399999999999 2.9216999999999986 4.2156 1.2948 |
Mean class 2 : |  6.571874999999997 2.8836250000000003 5.617500000000015 2.043 |

Covariance matrices
|  0.3041698400000001 -0.01924767999999999 0.026098759999999985 -0.01157192000000001 |
|  -0.01924767999999999 0.11199610999999994 -0.007550520000000006 0.005738839999999999 |
|  0.026098759999999985 -0.007550520000000006 0.151716639999999996 0.014833120000000005 |
|  -0.01157192000000001 0.005738839999999999 0.014833120000000005 0.03876896000000001 |

Confusion matrix
| 0 0 |
| 0 20 |

correct 100 %
error 0 %
--------------------------------
```

```
---------- Validation 9 --------------

Confusion matrix
| 0 0 |
| 0 20 |

correct 100 %
error 0 %
--------------------------------
```

```
---------- Validation 9 --------------
Mean class 1 : |  6.350399999999999 2.9216999999999986 4.2156 1.2948 |
Mean class 2 : |  6.605249999999998 2.9152499999999995 5.615375000000001 2.0576250000000003 |

Covariance matrices
|  0.3041698400000001 -0.01924767999999999 0.026098759999999985 -0.01157192000000001 |
|  -0.01924767999999999 0.11199610999999994 -0.007550520000000006 0.005738839999999999 |
|  0.026098759999999985 -0.007550520000000006 0.151716639999999996 0.014833120000000005 |
|  -0.01157192000000001 0.005738839999999999 0.014833120000000005 0.03876896000000001 |

Confusion matrix
| 0 0 |
| 0 20 |

correct 100 %
error 0 %
--------------------------------
```

```
---------- Validation 10 --------------

Confusion matrix
| 0 0 |
| 2 18 |

correct 90 %
error 10 %
--------------------------------
```

```
---------- Validation 10 --------------
Mean class 1 : |  6.350399999999999 2.9216999999999986 4.2156 1.2948 |
Mean class 2 : |  6.593625 2.9083750000000004 5.619749999999998 2.076749999999999 |

Covariance matrices
|  0.3041698400000001 -0.01924767999999999 0.026098759999999985 -0.01157192000000001 |
|  -0.01924767999999999 0.11199610999999994 -0.007550520000000006 0.005738839999999999 |
|  0.026098759999999985 -0.007550520000000006 0.151716639999999996 0.014833120000000005 |
|  -0.01157192000000001 0.005738839999999999 0.014833120000000005 0.03876896000000001 |

Confusion matrix
| 0 0 |
| 2 18 |

correct 90 %
error 10 %
--------------------------------
```

4. สรุปผลการทดลอง

เมื่อใช้จำนวน K และ จำนวน feature เท่ากัน ผลของการเลือกคลาสจะเหมือนกัน

ภาคผนวก

Code: KNN (NodeJS)

```
let fs = require('fs')
const math = require('mathjs')

let input = fs.readFileSync('TWOCLASS.dat', 'utf8')
const fNum = 4 //Change number of features here
const percentValidate = 10
const kValue = 14

const fetchData = Promise.resolve(
    input.trim().split('\r\n').map(x => x.split('\t'))
)

const setUpTestData = (percentValidate, data) => {
    const testDataNum = data.length / percentValidate
    const round = data.length / testDataNum
    let testDatas = []
    for (i = 0; i < round; i++) {
        testDatas.push(data.slice((i * testDataNum), (i * testDataNum +
testDataNum)))
    }
    return testDatas
}

const setUpTrainData = (percentValidate, data) => {
    const testDataNum = data.length / percentValidate
    const round = data.length / testDataNum
    let trainDatas = []
    for (i = 0; i < round; i++) {
        let trainData = []
        for (j = 0; j < data.length; j++) {
            if (j < (i * testDataNum) || j >= (i * testDataNum + testDataNum))
{
                trainData.push(data[j])
            }
        }
        trainDatas.push(trainData)
    }
    return trainDatas
}

const calDistance = (sources, destinations) => {
    return destinations.map(
        (destination) => {
            let res = []
            for (let j = 0; j < (destination.length - 1); j++) {
```

```
                res.push(
                    Math.pow(
                        (parseFloat(sources[j]) - parseFloat(destination[j]))
                        , 2)
                )
            }
            let resSq = Math.sqrt(res.reduce((prev, curr) => prev + curr))
            res = []
            res.push(resSq, destination[4])
            return res
        }
    )
}

const main = async () => {
    let sourceData = await fetchData.then((value) => {
        value.shift()
        return value
    })
    const testDatas = setUpTestData(percentValidate, sourceData)
    const trainDatas = setUpTrainData(percentValidate, sourceData)
    for (let i = 0; i < 10; i++) {
        let a = 0, b = 0, c = 0, d = 0
        for (let j = 0; j < testDatas[i].length; j++) {
            let dist = await calDistance(testDatas[i][j], trainDatas[i])
            const distSorteds = dist.sort((a, b) => a[0] - b[0])
            const distSliceds = distSorteds.slice(0, kValue)
            const classes = [0, 0]
            distSliceds.forEach(distSliced => {
                if (distSliced[1] === '1') { classes[0]++ }
                else if (distSliced[1] === '2') { classes[1]++ }
            })

            let classChoose
            if (classes[0] === classes[1]) { classChoose =
Math.floor((Math.random() * 2) + 1) }
            else if (classes[0] > classes[1]) { classChoose = 1 }
            else if (classes[0] < classes[1]) { classChoose = 2 }
            if (parseInt(testDatas[i][j][4]) === 1 && classChoose === 1) { a =
a + 1 }
            else if (parseInt(testDatas[i][j][4]) === 1 && classChoose === 2)
{ b = b + 1 }
            else if (parseInt(testDatas[i][j][4]) === 2 && classChoose === 1)
{ c = c + 1 }
            else if (parseInt(testDatas[i][j][4]) === 2 && classChoose === 2)
{ d = d + 1 }

        }
```

```
        console.log(`---------- Validation ${i + 1} --------------`)
        console.log(`\nConfusion matrix`)
        console.log(`| ${a} ${b} |`)
        console.log(`| ${c} ${d} |`)
        let correct = 100 * (a + d) / (a + b + c + d)
        let error = 100 - correct
        console.log(`\ncorrect ${correct} %`)
        console.log(`error ${error} %`)
        console.log(`-------------------------------\n\n\n\n\n\n\n\n\n`)
    }
}

main()
```

Code: Bayes (NodeJS)

```
let fs = require('fs')
const math = require('mathjs')

let input = fs.readFileSync('TWOCLASS.dat', 'utf8')
const fNum = 4 //Change number of features here
const percentValidate = 10
const pw1 = 1
const pw2 = 1

const fetchData = Promise.resolve(
    input.trim().split('\r\n').map(x => x.split('\t'))
)

const setUpTestData = (percentValidate, data) => {
    const testDataNum = data.length / percentValidate
    const round = data.length / testDataNum
    let testDatas = []
    for (i = 0; i < round; i++) {
        testDatas.push(data.slice((i * testDataNum), (i * testDataNum +
testDataNum)))
    }
    return testDatas
}

const setUpTrainData = (percentValidate, data) => {
    const testDataNum = data.length / percentValidate
    const round = data.length / testDataNum
    let trainDatas = []
    for (i = 0; i < round; i++) {
        let trainData = []
```

```
        for (j = 0; j < data.length; j++) {
            if (j < (i * testDataNum) || j >= (i * testDataNum + testDataNum))
{
                trainData.push(data[j])
            }
        }
        trainDatas.push(trainData)
    }
    return trainDatas
}

const separateClass = (dataSource, classLabel, f) => {
    return dataSource.filter((data) => { return data[4] === classLabel
}).map((val) => {
        return val.slice(0, f)
    })
}

const average = datas => {
    let initP = []
    initP = datas[0].map(a => 0)
    return datas.reduce((p, c) => {
        let res = []
        for (let i = 0; i < p.length; i++) {
            res.push(parseFloat(p[i]) + parseFloat(c[i]))
        }
        return res
    }, initP).map((sum) => {
        return sum / datas.length
    })
}

const xMinusMean = (datas, means) => {
    return datas.map((data) => {
        let res = []
        for (let i = 0; i < data.length; i++) {
            res.push(parseFloat(data[i]) - parseFloat(means[i]))
        }
        return res
    })
}

const fx = (numClass, cov, xMinusMean) => {
    return (1 / (Math.sqrt(((2 * Math.PI) ^ numClass) * (math.det(cov))))) *
math.exp(math.multiply(math.multiply(math.multiply(math.transpose(xMinusMean),
-0.5), math.inv(cov)), xMinusMean))
}
```

```
const main = async () => {
    let sourceData = await fetchData.then((value) => {
        value.shift()
        return value
    })
    const testDatas = setUpTestData(percentValidate, sourceData)
    const trainDatas = setUpTrainData(percentValidate, sourceData)
    for (i = 0; i <= 9; i++) {
        let testData = testDatas[i].map((val) => {
            return val.slice(0, fNum)
        })
        let testClass = testDatas[i].map((val) => {
            return val.slice(-1)
        })

        let trainClass1 = separateClass(trainDatas[i], '1', fNum)
        let trainClass2 = separateClass(trainDatas[i], '2', fNum)

        let meanClass1 = average(trainClass1)
        let meanClass2 = average(trainClass2)

        let trainXMinusMean1 = xMinusMean(trainClass1, meanClass1)
        let trainXMinusMean2 = xMinusMean(trainClass2, meanClass2)

        let cov1 =
math.multiply(math.multiply(math.transpose(trainXMinusMean1),
trainXMinusMean1), (1 / trainClass1.length))
        let cov2 =
math.multiply(math.multiply(math.transpose(trainXMinusMean2),
trainXMinusMean2), (1 / trainClass2.length))
        let a = 0, b = 0, c = 0, d = 0
        for (j = 0; j < testData.length; j++) {
            let testXMinusMean1 = xMinusMean(testData, meanClass1)
            let testXMinusMean2 = xMinusMean(testData, meanClass2)
            let classChoose
            let fx1 = fx(2, cov1, testXMinusMean1[j])
            let fx2 = fx(2, cov2, testXMinusMean2[j])
            if (fx1 * pw1 === fx2 * pw2) { classChoose =
Math.floor((Math.random() * 2) + 1) }
            else if (fx1 * pw1 > fx2 * pw2) { classChoose = 1 }
            else if (fx1 * pw1 < fx2 * pw2) { classChoose = 2 }

            if (parseInt(testClass[j]) === 1 && classChoose === 1) { a = a + 1
}
            else if (parseInt(testClass[j]) === 1 && classChoose === 2) { b =
b + 1 }
            else if (parseInt(testClass[j]) === 2 && classChoose === 1) { c =
c + 1 }
```

```
            else if (parseInt(testClass[j]) === 2 && classChoose === 2) { d =
d + 1 }
        }
        console.log(`---------- Validation ${i + 1} --------------`)
        let smeanClass1 = meanClass1.reduce((res, mean) => res + ' ' + mean,
'')
        let smeanClass2 = meanClass2.reduce((res, mean) => res + ' ' + mean,
'')
        console.log(`Mean class 1 : | ${smeanClass1} |`)
        console.log(`Mean class 2 : | ${smeanClass2} |`)
        console.log(`\nCovariance matrices`)

        cov1.forEach((row) => {
            let cov = row.reduce((res, r) => res + ' ' + r, '')
            console.log(`| ${cov} |`)
        })
        console.log(`\nConfusion matrix`)
        console.log(`| ${a} ${b} |`)
        console.log(`| ${c} ${d} |`)
        let correct = 100 * (a + d) / (a + b + c + d)
        let error = 100 - correct
        console.log(`\ncorrect ${correct} %`)
        console.log(`error ${error} %`)
        console.log(`-------------------------------\n`)
    }
}

main()
```