PNDI : Rapport du projet

Table des matières

[DA 2](#_Toc104213881)

[Phase 1 2](#_Toc104213882)

[Phase 2 6](#_Toc104213883)

[Phase 3 9](#_Toc104213884)

[Phase 4 11](#_Toc104213885)

[Interprétation phase 4 16](#_Toc104213886)

[Amélioration 16](#_Toc104213887)

[Classification actuelle 16](#_Toc104213888)

[Solution personnelle 16](#_Toc104213889)

[Critique du code 17](#_Toc104213890)

[Réalisation du projet 17](#_Toc104213891)

# DA

## Phase 1

// structures

data { userAcceleration.x

. { userAcceleration.y

. { userAcceleration.z

sujet { code

. { gender

PATH = {

"dws\_1/","dws\_2/","dws\_11/",

"jog\_9/", "jog\_16/",

"sit\_5/", "sit\_13/",

"std\_6/", "std\_14/",

"ups\_3/", "ups\_4/",

"wlk\_7/", "wlk\_8/", "wlk\_15/",

}

DATA\_SUBJECT "../data/data\_subjects\_info.csv"

DATA "../data/A\_DeviceMotion\_data/A\_DeviceMotion\_data/"

TRAIN\_SET "../trainSet.csv"

TEST\_SET "../testSet.csv"

NB\_SUJETS 24

NB\_FORLDER 15

NB\_SUJET\_TEST 3

TIME\_ONE\_MINUTE 600

o─────────────o ↓ file

│ readSubject │

o─────────────o ↓ subject

┌─── \*

│ // lis les infos du sujet

│

│ line[100];

│ trash;

│ fgets(line, sizeof(line), file);

│

│ sscanf\_s(line, "%d; %d; %d; %d; %d", &subject.code, &trash, &trash, &trash, &subject.gender);

└──────────

o──────────────o ↓ file

│ readDataBase │

o──────────────o ↓ data

┌─── \*

│ // retire les titres de colonne

│ fgets(line, sizeof(line), file);

│ // lit dans le file

│ fgets(line, sizeof(line), file);

│

│ sscanf\_s(line, "%lf, %lf, %lf, %lf, %lf, %lf, %lf, %lf, %lf, %lf, %lf,

│ %lf, %lf", &trash, &trash, &trash, &trash, &trash, &trash, &trash,

│ &trash, &trash, &trash, &data.userAccX, &data.userAccY, &data.userAccZ);

└──────────

o────────────────o ↓ file

│ writeTitleVacc │

o────────────────o

┌─── \*

│ // écrit 600 titres "Vacc"

│ i = 0

│╔══ while(i < TIME\_FOR\_A\_MIN)

│║ fprintf(file, "; %s", "Vacc");

│║ i++

│╙──

└──────────

o────────────────o ↓ typeMovement

│ defineMovement │

o────────────────o ↓ movement

┌─── \*

│┌── if (typeMovement == "dws")

││ movement = "downstair"

│├── if (typeMovement == "jog")

││ movement = "jogging"

│├── if (typeMovement == "sit")

││ movement = "sit down"

│├── if (typeMovement == "std")

││ movement = "stand up"

│├── if (typeMovement == "ups")

││ movement = "upstair"

│├── if (typeMovement == "wlk")

││ movement = "walking"

│└──

└──────────

o──────────────o ↓ realClasses[], estimatedClasses[], nbTests, nbClasses

│ createMatrix │

o──────────────o ↓ matrix[],

┌─── \*

│ matrix[][] = new ARRAY( nbClasses)( nbClasses)

│ // titre mouvement

│ iTitre = 0

│╔══ while (iTitre < nbClasses)

│║ matrix[0][iTitre] = iTitre +1

│╙──

│ // init matrix full à 0

│ i = 1

│╔══ while ( i < nbClasses)

│║ matrix[i][0] = i

│║ j = 1

│║╔══ while (j < nbClasses)

│║║ matrix[i][j] = 0

│║╙──

│╙──

│ i = 0

│╔══ while (i < nbTest)

│║ matrix[realClasses[i] -1][extimatedClasses[i] -1]++

│╙──

│

└──────────

o───────────o

│ Principal │

o───────────o

┌─── \*

│ subjects = new Array(NB\_SUBJECTS)

│ ouvrir DATA\_SUBJECT dans dataSubject

│

│┌── if (dataSubject == null)

││ sortir "erreur ouverture du fichier"

│├── else

││ // retire les titres de colonne

││ fgets(trash, sizeof(trash), dataSubject);

││

││ // lit les info des Subject et les sauvegarde

││ i = 0

││╔══ while( i < NB\_Subject)

││║ subjects[i] = readSubject(dataSubject);

││║ i++

││╙──

││

││ ouvrir TRAIN\_SET dans trainSet

││ ouvrir TEST\_SET dans testSet

││┌── if (trainSet == null OR testSet == null)

│││ sortir "erreur création des fichiers"

││├── else

│││ // ecrit

│││ fprintf(trainSet, "%s; %s; %s", "mouvement", "gender", "index");

│││ fprintf(testSet, "%s; %s; %s", "mouvement", "gender", "index");

│││

│││ writeTilteVacc(trainSet);

│││ writeTilteVacc(testSet);

│││

│││ index = 1

│││ //permet la repartition entre trainSet et testSet (+- 90 - 10 %)

│││ iSubjectForTest = 0;

│││

│││ iFolder = 0

│││╔══ while (i Folder < NB\_FOLDER)

│││║ folder = DATA + PATH[iFolder]

│││║ o────────────────o ↓ PATH[iFolder]

│││║ │ defineMovement │

│││║ o────────────────o ↓ movement

│││║

│││║ nbTestSubjects = 0

│││║ iSubjects = 0

│││║╔══ while (iSubject < NB\_SUBJECTS)

│││║║┌── if (nbTestSubjects < NB\_TEST\_SUBJECTS AND iSubject ==

│││║║│ iSubjectForTest + nbTestSubjects)

│││║║│ fileToWrite = testSet;

│││║║│ nbTestSubjects++;

│││║║├── else

│││║║│ fileToWrite = trainSet;

│││║║└──

│││║║ //passe a la ligne pour le Subject

│││║║ fprintf(fileToWrite, "%s", "\n");

│││║║

│││║║ //ecrit le type de mouvement

│││║║ fprintf(fileToWrite, "%s;", mouvement);

│││║║

│││║║ //colonne genre

│││║║ fprintf(fileToWrite, "%d;", subjects[iSubject].gender);

│││║║ //colonne index

│││║║ fprintf(fileToWrite, "%d;", index);

│││║║

│││║║ sprintf\_s(code, sizeof(code), "%d", subjects[iSubject].code);

│││║║ pathDataFile = folder + "sub\_" + code + ".csv"

│││║║

│││║║ ouvrir pathDataFile dans dataFile

│││║║

│││║║┌── if (dataFile == null)

│││║║│ sortir "erreur création fichier"

│││║║├── else

│││║║│ i = 0

│││║║│ o──────────────o ↓ dataFile

│││║║│ │ readDataBase │

│││║║│ o──────────────o ↓ data

│││║║│╔══ while (i < TIME\_ONE\_MINUTE AND !feof(dataFile))

│││║║│║ Vacc = sqrt(data.userAccX \* data.userAccX + data.userAccY \*

│││║║│║ data.userAccY + data.userAccZ \* data.userAccZ);

│││║║│║ fprintf(fileToWrite, "%lf;", Vacc);

│││║║│║ data = readDataBase(dataFile);

│││║║│║ i++;

│││║║│╙──

│││║║│ fermer dataFile;

│││║║│ iSubject++;

│││║║│ index++;

│││║║└──

│││║╙──

│││║ iSubjectForTest += 3;

│││║┌── if (iSubjectForTest ≥ NB\_SUBJECTS)

│││║│ iSubjectForTest = 0;

│││║└──

│││║ iFolder++;

│││╙──

│││ fermer trainSet;

│││ fermer testSet;

││└──

││ fermer dataSubject;

│└──

│

└──────────

## Phase 2

NB\_MAX\_CLASSES 20

o─────────────────────────o ↓ realClasses[], estimatedClasses[], nbTests

│ displayConfusionMatrice │

o─────────────────────────o

┌─── \*

│ o─────────────o ↓ realClasses[], estimatedClasses[], nbTests

│ │ maxMovement │

│ o─────────────o ↓ maxClasses

│ o──────────────o ↓ realClasses[], extimatedClasses[], nbTestsn,

│ │ │ maxClasses

│ │ createMatrix │

│ o──────────────o ↓ matrix[]

│

│ o──────────────────o ↓ matrix[0]], NULL

│ │ displayRowMatrix │

│ o──────────────────o

│ i = 1

│╔══ while (i < NB\_MAX\_CLASSES AND i < nbClasses)

│║ // sort le type du mouvement

│║ sortir mouvement +'\t'

│║ i = 0

│║╔══ while (i < NB\_MAX\_CLASSES AND ligne[i] ≠ NULL)

│║║ // sort le nombres d'hestimation

│║║ sortir ligne[i] + '\t'

│║╙──

│╙──

└──────────

o─────────────o ↓ realClasses[], estimatedClasses[], nbTests

│ maxMovement │

o─────────────o ↓ maxClasses

┌─── \*

│ maxClasses = 0

│ i = 0

│╔══ while (i < nbTests)

│║┌── if( realClasses[i] > maxClasses OR estimatedClasses[i] > maxClasses )

│║│ maxClasses = realClasses[i] > estimatedClasses[i] ? realClasses[i] :

│║│ estimatedClasses[i]

│║└──

│║ i++

│╙──

└──────────

==========================================================================

o─────────────────o ↓ realClasses[],estimatedClasses[],nbTests

│ displayAccuracy │

o─────────────────o

┌─── \*

│ NB\_CLASSES = 6

│

│ // boucler sur les deux tableaux en incrémentant le total et potentiellement totalCorrect

│ o─────────────────o ↓ realClasses[],estimatedClasses[],nbTest

│ │ countTotCorrect │

│ o─────────────────o ↓ totalCorrect, total

│

│ // calculer et sortir l'accuracy

│ accuracy = totalCorrect/total100

│ sortir "L'accuracy est de " + accuracy + "%"

└──────────

o─────────────────o ↓ realClasses[],estimatedClasses[],nbTest

│ countTotCorrect │

o─────────────────o ↓ totalCorrect, total

┌─── \*

│ totalCorrect = 0

│ total = 0

│ i = 0

│╔══ while (i < nbTest)

│║┌── if (realClasses[i] == estimatedClasses[i])

│║│ totalCorrect++

│║└──

│║ total++

│║ i++

│╙──

└──────────

==========================================================================

o───────────────────────o ↓ realClasses[],estimatedClasses[],nbTests)

│ displayResultsByClass │

o───────────────────────o

┌─── \*

│ NB\_CLASSES = 6

│

│ // creation d'un tableau de structures de NB\_CLASSES cellules (ici 6)

│ classes[] = new ARRAY (NB\_CLASSES)

│ movement { number

│ . { totalCorrect = 0

│ . { total = 0

│

│ iClasses = 0

│╔══ while (iClasses < NB\_CLASSES)

│║ classes[iClasses].number = iClasses+1

│║ iClasses++

│╙──

│

│ //parcourir et comparer les deux tableaux realClasses et estimatedClasses

│ o─────────────o ↓ realClasses[],estimatedClasses[], classes, nbTest

│ │ countResult │

│ o─────────────o ↓ classes[]

│

│ // afficher resultByCLass

│ o────────────o ↓ classes, NB\_CLASSES

│ │ displayRow │

│ o────────────o

└──────────

o─────────────o ↓ realClasses[],estimatedClasses[], classes, nbTest

│ countResult │

o─────────────o ↓ classes[]

┌─── \*

│ i = 0

│╔══ while(i < nbTest)

│║ iClasses = realClasses[i]-1

│║┌── if (realClasses[i] == estimatedClasses[i])

│║│ classes[iClasses].totalCorrect++

│║└──

│║ classes[iClasses].total++

│║ i++

│╙──

└──────────

o────────────o ↓ classes, NB\_CLASSES

│ displayRow │

o────────────o

┌─── \*

│ sortir "classe | bien classes | total | Pourcentage\n"

│ iClasses = 0

│╔══ while (iClasses < NB\_CLASSES)

│║┌── if (classes[iClasses].total ≠ 0)

│║│ sortir classes[iClasses].number + " | "

│║│ sortir classes[iClasses].totalCorrect + " | "

│║│ sortir classes[iClasses].total + " | "

│║│ pourcentage = classes[iClasses].totalCorrect /

│║│ classes[iClasses].total\*100

│║│ sortir pourcentage + "%\n"

│║└──

│║ i++

│╙──

└──────────

o─────────────────o ↓ realClasses[],estimatedClasses[],nbTests

│ displayAccuracy │

o─────────────────o

┌─── \*

│ NB\_CLASSES = 6

│

│ // boucler sur les deux tableaux en incrÃ©mentant le total et

│ potentiellement totalCorrect

│ o─────────────────o ↓ realClasses[],estimatedClasses[],nbTest

│ │ countTotCorrect │

│ o─────────────────o ↓ totalCorrect, total

│

│ // calculer et sortir l'accuracy

│ accuracy = totalCorrect/total\*100

│ sortir "L'accuracy est de " + accuracy + "%"

└──────────

o─────────────────o ↓ realClasses[],estimatedClasses[],nbTest

│ countTotCorrect │

o─────────────────o ↓ totalCorrect, total

┌─── \*

│ totalCorrect = 0

│ total = 0

│ i = 0

│╔══ while (i < nbTest)

│║┌── if (realClasses[i] == estimatedClasses[i])

│║│ totalCorrect++

│║└──

│║ total++

│║ i++

│╙──

└──────────

## Phase 3

trainSetPAth = "../../phase 1/trainSet.csv"

fiModelPath= "../fiModel.csv"

//structures

dataDixiemeSeconde {

. totVacc

. totCarreVacc

. totLigne

}

o───────────o ↓ trainsetPath

│ Principal │

o───────────o

┌─── \*

│ ouvrir traineSet.csv en lecture

│ cree fiModel.csv

│

│┌── if(trainSet == NULL || fiModel == NULL

││ sortir "erreur ouverture des fichiers"

│├── else

││ o─────────────o ↓ fiModel

││ │ initFiModel │

││ o─────────────o

││ // parcours du fichier complet

││╔══ while( !feof(trainset))

││║ o───────────o

││║ │ initDatas │

││║ o───────────o ↓ datas

││║ totGenerale = 0

││║ nbTot = 0

││║ line = lire ligne dans trainSet

││║ currentmovement= lire "mouvement" dans line

││║ movement = currentmovement

││║╔══ while( movement == currentMovemet AND !feof(trainSet))

││║║ passer "index" dans line

││║║ passer "genre" dans line

││║║ iColonne = 0

││║║ // parcours de chaque colonne de dixieme de seconde

││║║ vacc = lire "Vacc" dans case

││║║╔══ while( vacc ≠ null)

││║║║ // structure

││║║║ datas[iColonne].totVacc += vacc

││║║║ datas[iColonne].totCarreVacc += vacc \* vacc

││║║║ datas[iColonne].totLigne++

││║║║ // moyenne generale

││║║║ totGenerale += vacc

││║║║ nbTot++

││║║║ // dixieme de seconde suivant

││║║║ iColonne++

││║║║ vacc = lire "Vacc" dans case

││║║╙──

││║║ line = lire ligne dans trainSet

││║║ currentmovement= lire "mouvement" dans line

││║╙──

││║ module(writeData; fiModel, data[], totVacc, nbVacc;)

││╙──

││ fermer trainSet

││ fermer fiModel

│└──

└──────────

o─────────────o ↓ fiModel

│ initFiModel │

o─────────────o

┌─── \*

│ ecrire mouvement dans foModel

│ i = 0

│╔══ while(i < 600)

│║ ecrire "Vacc" dans fiModel

│╙──

└──────────

o───────────o

│ initDatas │

o───────────o ↓ datas

┌─── \*

│ datas= ARRAY(600)

│ i = 0

│╔══ while (i < 600)

│║ datas[i].totVacc = 0

│║ datas[i].totCarreVacc = 0

│║ datas[i].totLigne = 0

│║ i++

│╙──

└──────────

o───────────o ↓ file, data[], totVacc, nbVacc

│ writeData │

o───────────o

┌─── \*

│ i = 0

│ // écrit la moyennes

│ ecrire \n dans file

│ ecrire mouvement dans file

│╔══ while( i < 600 AND datasDixiemeSeconde[i].totVacc ≠ null)

│║ ecrire datasDixiemeSeconde[i].totVacc / datasDixiemeSeconde[i].totLigne

│║ dans file

│╙──

│ // écrit la ecart types

│ ecrire \n dans file

│ ecrire mouvement dans file

│ i = 0

│╔══ while( i < 600 AND datasDixiemeSeconde[i].totCarreVacc≠ null)

│║ ecrire datasDixiemeSeconde[i].totCarreVacc /

│║ datasDixiemeSeconde[i].totLigne –datasDixiemeSeconde[i].totVacc /

│║ datasDixiemeSeconde[i].totLigne dans file

│╙──

│ // écrit moyennes générale

│ ecrire \n dans file

│ ecrire mouvement dans file

│ ecrire totGenerale / nbTot dans file

└──────────

## Phase 4

struct model {

type

averages[TIME\_ONE\_MINUTE ]

nbValues

stds[TIME\_ONE\_MINUTE ]

globalAvg

}

TIME\_ONE\_MINUTE 600

NB\_TESTS 1000

NB\_MOVEMENTS 6

FIMODEL\_PATH "../../phase3/fimodel.csv"

TEST\_SET\_PATH "../../phase1/testSet.csv"

o───────────o ↓ file

│ readModel │

o───────────o ↓ models

┌─── \*

│ models = new array(NB\_TESTS)

│

│ // supprime la ligne des titres

│ line = lire ligne dans file

│ i = 0

│╔══ while (!feof(file))

│║ // récupere le mouvement

│║ line = lire ligne dans file

│║ token = strtok(line, ";")

│║ models[i].type = token

│║ // récupère les vacc moyennes

│║ iVacc = 0

│║╔══ while(iVacc < TIME\_ONE\_MINUTE)

│║║ token = strtok(line, ";")

│║║ models[i].avergages[iVacc] = token

│║║ iVacc++

│║╙──

│║ models[i].nbValues = iVacc

│║

│║ // récupère les vacc écarts types

│║ line = lire ligne dans file

│║ iVacc = 0

│║╔══ while(iVacc < TIME\_ONE\_MINUTE)

│║║ token = strtok(line, ";")

│║║ models[i].stds[iVacc] = token

│║║ iVacc++

│║╙──

│║ // récupère la Vacc moyenne générale

│║ line = lire ligne dans file

│║ token = strtok(line, ";")

│║ models[i].globalAvg = token

│║ i++

│╙──

└──────────

o───────────o

│ Principal │

o───────────o

┌─── \*

│ ouvrir fiModel en lecture

│ ouvrir testSet en lecture

│

│

│ realClasses = new array(NB\_TESTS)

│ estimatedClasses= new array(NB\_TESTS)

│ // module pour lire fiModel et stocker dans models

│ o───────────o ↓ fiModel

│ │ readModel │

│ o───────────o ↓ models

│

│ // supprime la lignes de titres

│ line = lire ligne dans testSet

│ int i = 0

│╔══ while (i < NB\_TESTS AND !feof(trainSet))

│║ line = lire ligne dans testSet

│║ o───────────────o ↓ line

│║ │ decomposition │

│║ o───────────────o ↓ data, movementType

│║ realClasses[i] = movementType

│║

│║ // estimation sur la moyenne

│║ o───────────────────o ↓ models, data

│║ │ EstimationAverage │

│║ o───────────────────o ↓ movementUsingAverage

│║ // estimation sur l'écartType

│║ o─────────────────────────────o ↓ models, data

│║ │ EstimationStandartDeviation │

│║ o─────────────────────────────o ↓ movementUsingStandartDeviation

│║ // estimation sur la moyenne generale

│║ o───────────────────────────────o ↓ models, data

│║ │ EstimationUsingGeneralAverage │

│║ o───────────────────────────────o ↓ movementUsingGeneralAverage

│║ o──────────────────o ↓ movementUsingAverage,

│║ │ movementUsingStandartDeviation, movementUsingGeneralAverage

│║ │ identifyMovement │

│║ o──────────────────o ↓ movement

│║

│║ // enregistre l'estimation dans estimatedClasses

│║ estimatedClasses[i] = movement

│║ i++

│╙──

│

│ // affichage

│ displayResultsForEachClass(realClasses, estimatedClasses, i)

│ displayAccuracy(realClasses, estimatedClasses, i)

│ displayClass(realClasses, estimatedClasses, i)

│

│

│ fermer fiModel

│ fermer testSet

└──────────

o──────────────────o ↓ movementUsingAverage,

│ movementUsingStandartDeviation, movementUsingGeneralAverage

│ identifyMovement │

o──────────────────o ↓ movement

┌─── \*

│ // regarde si movementUsingStandartDeviation même que un autre

│┌── if( movementUsingAverage == movementUsingStandartDeviation ||

││ movementUsingStandartDeviation == movementUsingGeneralAverage)

││ movement = movementUsingGeneralAverage

│├── else

││ // pas de check d'égalité entre movementUsingAverage et

││ movementUsingGeneralAverage

││ // car prendra la valeur de movementUsingAverage dans les 2 cas

││

││ // cas de base ⇒ movementUsingAverage

││ movement = movementUsingAverage

│└──

└──────────

o───────────────o ↓ line

│ decomposition │

o───────────────o ↓ data, movementType

┌─── \*

│ data = new array(NB\_TESTS)

│ // recupere le mouvement

│ token = strtok(line, ";")

│ movementType = token

│

│ // passe index et genre

│ token = strtok(NULL, ";")

│ token = strtok(NULL, ";")

│

│ // prend la premiere valeur de Vacc

│ token = strtok(NULL, ";")

│ i = 0

│╔══ while (token ≠ NULL)

│║ data[i] = token

│║ token = strtok(NULL, ";")

│║ i++

│╙──

└──────────

// 3 modules d'estimation

o───────────────────o ↓ models[], data[]

│ EstimationAverage │

o───────────────────o ↓ movementUsingAverage

┌─── \*

│ standartDevMin = MAX\_VALUE

│ iMovement = 0

│ // boucle pour calculer la distance min

│╔══ while (iMovement < NB\_MOVEMENTS)

│║ standartDev = 0

│║ i = 0

│║ // calcul la distance euclidienne

│║╔══ while (i < TIME\_ONE\_MINUTE)

│║║ standartDev += pow((models[iMovement]. averages[i] - data[i]), 2)

│║║ i++

│║╙──

│║ standartDev = sqrt(standartDev)

│║ // regarde si la distance est la plus petite trouvée jusqu'à présent

│║┌── if( standartDev < standartDevMin )

│║│ movementUsingAverage= iMovement

│║│ standartDevMin = standartDev

│║└──

│║ iMovement++

│╙──

└──────────

o─────────────────────────────o ↓ models[], data[]

│ EstimationStandartDeviation │

o─────────────────────────────o ↓ movementUsingStandartDeviation

┌─── \*

│ standartDevMin = MAX\_VALUE

│ iMovement = 0

│ // boucle pour calculer la distance min

│╔══ while (iMovement < NB\_MOVEMENTS)

│║ standartDev = 0

│║ i = 0

│║ // calcul la distance euclidienne

│║╔══ while (i < TIME\_ONE\_MINUTE)

│║║ standartDev += pow((models[iMovement].stds[i] - data[i]), 2)

│║║ i++

│║╙──

│║ standartDev = sqrt(standartDev)

│║ // regarde si la distance est la plus petite trouvée jusqu'à présent

│║┌── if( standartDev < standartDevMin )

│║│ movementUsingStandartDeviation = iMovement

│║│ standartDevMin = standartDev

│║└──

│║ iMovement++

│╙──

└──────────

o───────────────────────────────o ↓ models[], data[]

│ EstimationUsingGeneralAverage │

o───────────────────────────────o ↓ movementUsingGeneralAverage

┌─── \*

│ // calcule de la moyenne des data

│ vaccTot = 0

│ nbVacc = 0

│ i = 0

│╔══ while (i < TIME\_ONE\_MINUTE)

│║ vaccTot += data[i]

│║ nbVacc++

│║ i++

│╙──

│ vaccMiddle = vaccTot / nbVacc

│ diffMin = MAX\_VALUE

│ i = 0

│╔══ while (i < NB\_MOVEMENT)

│║ diff = abs(models[i].globalAvg - vaccMiddle)

│║┌── if(diff < diffMin)

│║│ diffMin = diff

│║│ movementUsingGeneralAverage = i

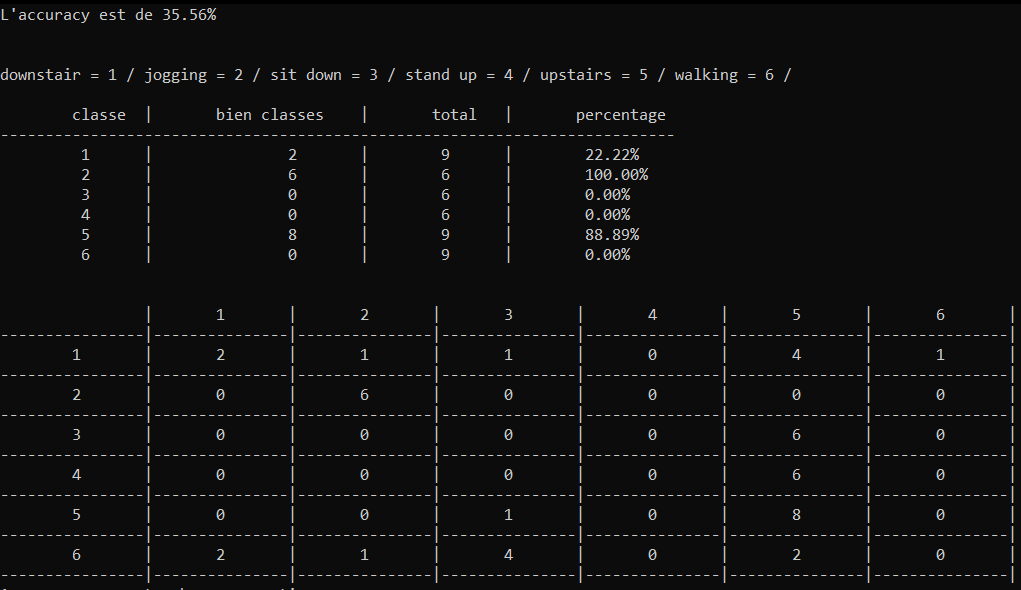
│║└──

│║ i++

│╙──

└──────────

# Interprétation phase 4



L’algorithme de triage de type de mouvement à une réussite de 35.56%. Il est capable de différencier les mouvements « jogging » et « upstairs » facilement des autres mouvements mais est mauvais pour déterminer le reste.

Les mouvements « sit down » et stand up » sont toujours confondu pour un mouvement « upstairs ».

# Amélioration

## Classification actuelle

Dans l’optique d’améliorer les résultats, les données des sujets non-utilisé (âge, taille, poids) devraient être intégrer au model sous forme d’intervalle. Cette intégration permettrait de déterminer le type de mouvement d’un sujet de test en le comparant avec les données d’un sujet se rapprochant physiquement de lui. La comparaison en deviendrait plus pertinente et donc plus précise.

## Solution personnelle

Le meilleur algorithme de classification, pour moi, devrait modifier les ses valeurs de références en fonction des résultats des test qu’il effectue. Pour cela, Il faudrait une quantité beaucoup plus importante de données et implémenter une partie à cet algorithme qui analyse les résultat.

# Critique du code

Le point d’amélioration principal de mon code serait probablement que d’autre personne y regarde. La phase 1 notamment semble plus complexe que nécessaire et des idées d’amélioration que d’autres auraient ne sont implémentée car seul moi suis repassé sur la manière dont fonctionne le code. Des commentaires pourraient également être ajouté à certains endroits pour améliorer la compréhension.

Si je devais donné un point positif au code, ce serait probablement l’utilisation de fonction pour simplifier la vue des boucles principale.

# Réalisation du projet

Dans notre groupe, J’ai réalisé l’entièreté des codes. Je réalisais également les DA durant les cours prévu.

En termes d’organisation, aucune n’a été mise en place. Une fois le DA fini, je réalisais le code de la phase sur base du DA dans les 3 jours suivant la fin de ce dernier. Cette organisation était relativement inefficace et mauvaise bien qu’elle me permette de réaliser tout en temps et en heure. Cela est uniquement dû à la taille de ce projet relativement petite. Elle ne pourra pas être appliquée lors d’autres projet de groupe.

Concernant les cours en lui-même, la partie pratique (réalisation du projet) m’as permit d’avoir une base sur la manière dont de grande quantité de données sont transformée et réarranger pour permettre de les utiliser plus facilement. La partie théorique est plus abstraite quant à elle. Le faible nombre de cours et le fait qu’ils soient placé en début de quadrimestre uniquement les rends vite oubliable. Le syllabus est incomplet ce qui rend la réalisation des exercices peut attrayante.