УНИВЕРЗИТЕТ “Св. КИРИЛ И МЕТОДИЈ” - СКОПЈЕ

Факултет за Информатички науки и Компјутерско Инженерство

Деловна пракса

Прв дел – Следење био-вредности

Втор дел – Андроид апликација за препознавање на електрокардиограми

Ментор:
Проф./д-р Невена Ацковска

Прв дел-Изработиле:
Дарко Тосев, 111224
Наталија Велкоска, 122006
Ивана Коцева, 121158

Втор дел-Изработил:
Дарко Тосев, 111224

Прв дел – Следење и документирање на био-вредности

Целта на овој дел од проектот беше да се креираат биолошки податоци, добиени со рачно мерење на субјекти(луѓе од различен пол и старосни групи) преку користење на ,,e-Health'' сензор платформата за Ардуино.

Целосната опрема и сензори можат да се видат на Слика 1, додека за нашите потреби беа користени само сензорот за мерење на пулсот и нивото на кислород во крвта и сензорот за добивање на електрокардиограм(означени на сликата). Дополнително, се користеше и апарат за мерење на крвниот притисок(висок и низок), како и за број на отчукувања на срцето во една минута. Поставувањето на елктродите за добивање на кардиограм може да се види на Слика 1. Електрокардиограмите се користат во вториот дел од проектот за креирање на апликација за нивно препознавање.

Слика 1.
Опрема и поставување на електроди

Во текот на едно мерење се користеа сите сензори и апарати(набројани погоре) истовремено. Беа направени 300 мерења, сите главно во интервал од 10 секунди до 1 минута. Секое мерење беше документирано во посебен текст документ, каде што во секој ред се напишани вредностите од сензорите за пулс, нивото на кислород и електрокардиограмот(овие сензори произведуваат вредности континуирано се додека се поврзани со сензор платформата). Додека информациите за крвниот притисок и отчукувањата на срцето се запишани во името на документот(овој апарат се вклучуваше еднаш во текот на секое мерење, поради нивното времетраење).

Користевме 1.0.5 верзија на Ардуино, бидејќи ,,e-Health'' сензор платформата не беше компатибилна со поновите верзии на Ардуино. Кодот во Ардуино(дел може да се види на Слика 2) беше прилично едноставен – иницијализација на сензорите во setup() и примање на податоците од сензорите во loop() (дополнително имав и системски часовник за времетраењето на мерењето, наместо тоа да се прави рачно). Понатаму, го користевме Процесинг софтверот за да ги испечатиме вредностите што ги добиваме од серискатa порта во датотека, бидејќи Ардуино нема такви можности. Кодот во Процесинг е исто така едноставен и беше наменет единствено за таа цел(прикажан на Слика 2).

```
void setup() {
  Serial.begin(115200);
  eHealth.initPulsioximeter();

  timer.setInterval(15000, repeatMe);

  //Attach the inttruptions for using the pulsioximeter.
  PCintPort::attachInterrupt(6, readPulsioximeter, RISING);
}

void loop() {
  timer.run();

  output = (String)eHealth.getBPM();
  output.concat(",");
  output.concat(eHealth.getOxygenSaturation());
  output.concat(",");

  ECG = eHealth.getECG();
  char charVal[10];
  dtostrf(ECG, 3, 3, charVal);
  output.concat((String)charVal);

  Serial.println(output);
  delay(8);
}
```

```
void setup() {
  mySerial = new Serial(this, Serial.list()[0], 115200);
  output = createWriter("m30.txt");
  //time = millis();
}

void draw() {
    if (mySerial.available() > 0) {
        String value = mySerial.readString();
        if (value != null) {
            output.print(value);
        }
    }
```

Слика 2
Делови од кодовите во Ардуино(лево) и Процесинг(десно)

Во следната табела можат да се видат информациите за луѓето врз кои се направени мерењата, како и кое мерење на кој се однесува. Дополнително, луѓето врз кои се направени првите 100 мерења, немаат срцеви проблеми.

Број на мерење	Возраст	Пол
101-113	51	Женски
114, 115, 137-171	55	Машки
116-136, 172-195	28	Машки
196-200	23	Машки
201-220	22	Женски
221-240	19	Женски
241-260	23	Женски
261-280	22	Женски
281-300	53	Женски
301-310	47	Машки
311-320, 398-400	45	Женски
321-330	17	Машки
331-340	24	Женски
341-370	22	Женски
371-386	16	Машки
387-397	72	Женски

Втор дел – Андроид апликација

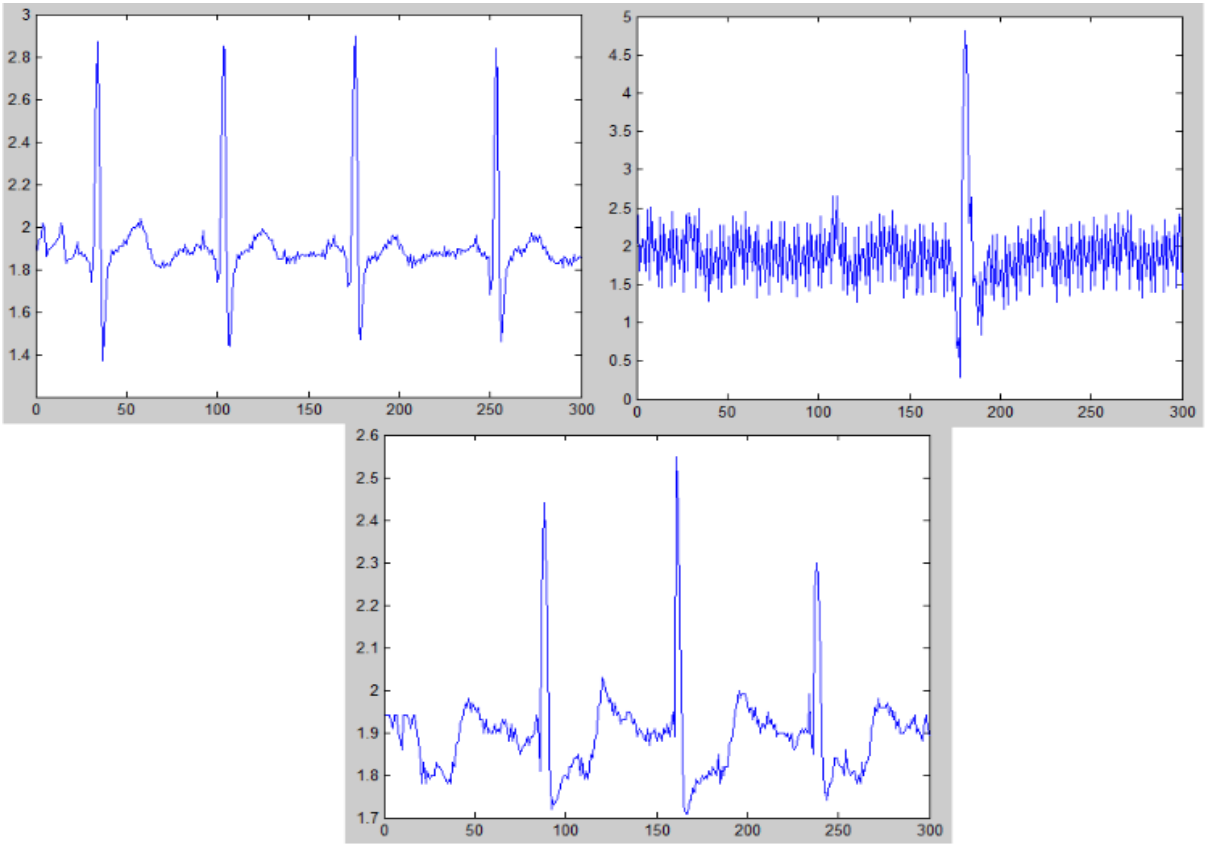
Креирањето на Андроид апликацијата се одвиваше во неколку фази:

- Анализа на податоците, одредување на атрибути коишто најдобро ќе ги опишат својствата на еден електрокардиограм и избор на класифицирачки модел.
- Создавање на негативни примероци, предпроцесирање, креирање на крајното податочно множество за тренирање и градење на моделот.

- Создавање на апликацијата со вграден модел за препознавање на електрокардиограм сигнал.

Анализа, атрибути и избор на модел

Како што може да видиме од Слика 3, каде што се прикажани примероци-електрокардиограми од изведените мерења, можат да се издвојат некои основни карактеристики што би ни помогнале во нивното препознавање. Најочигледно и наједноставно за било каков сигнал е да се гледа дистрибуцијата на вредностите. Прво нешто што се забележува за дистрибуцијата е нивниот ранг, односно дека се наоѓа помеѓу 0 и 5 (поради самата природа на сензорите, нема негативни вредности, додека максималната вредност добиена од сите мерења беше блиску до 5). Друга карактеристика што е очигледна за сигналите се пиковите (контракциите на срцето), односно нивната амплитуда во споредба со останатиот дел од сигналот.



Слика 3
Првите 300 вредности од неколку примероци

Следствено, ги одбрав следните атрибути:

- А1 – број на вредности надвор од интервалот од 0 до 5
- Б1 – број на вредности во интервалот од 0 до 1.4
- Б2 – број на вредности во интервалот од 1.4 до 1.9
- Б3 – број на вредности во интервалот од 1.9 до 2.5
- Б4 – број на вредности во интервалот од 2.5 до 3.5
- Б5 – број на вредности во интервалот од 3.5 до 5

- Ц1 – број на позитивни пикови. Притоа, за еден пик се смета низа од вредности кадешто разликата меѓу првата и последната е минимум 0.4. Ова го направив со цел да се избегне интерпретација на сигнали со сличен број на пикови, но мали амплитуди како електрокардиограми.

Бројот и границите на интервалите, како и големината на амплитудата во пиковите ги одбрав по разгледување на дел од примероците. Овие бројки во иднина можат да се подобрат преку тестирање и слично, со цел да добиеме поголема прецизност и точност. Секако, овие атрибути и параметри не се најдобрите за да се претстават електрокардиограмите и сигурен сум дека постојат други покомплексни карактеристики и начини на гледање на сигналот, што би можеле да се употребат во иднина од лица што поседуваат поголемо знаење за овој специфичен проблем.

Во однос на моделот, во основа потребен ни е модел којшто:

- Нема големи мемориски побарувања.
- Дава брзи резултати.
- Има прилично едноставна имплементација.
- Има голема точност.

Првите три побарувања се секако особено значајни поради ограничувањата што доаѓаат со мобилните телефони. Според ова моделите добиени со методи базирани на инстанци(како најпознат претставник е методот на к најблиски соседи) не се добар пристап(поради потребата од чување на тренирачкото множ. во меморија). Со тој исклучок, постојат добри опции за модели коишто би ги исполниле побарувањата. Баесовиот класификатор е еден таков пример – откако ќе се пресметаат потребните веројатности, лесно би се имплементирал моделот во Андроид апликацијата. Малку покомплексна имплементација би имале со невронски мрежи, но сепак изводливо. Постојат веќе многу добри модели добиени со невронски мрежи, како и ансамбл модели, опишани во многу трудови достапни на интернет (секако изработени на многу повисоко ниво, со користење на многу прецизна опрема за добивање на мерењата). Овие пристапи можат да се истражат во некоја идна работа.

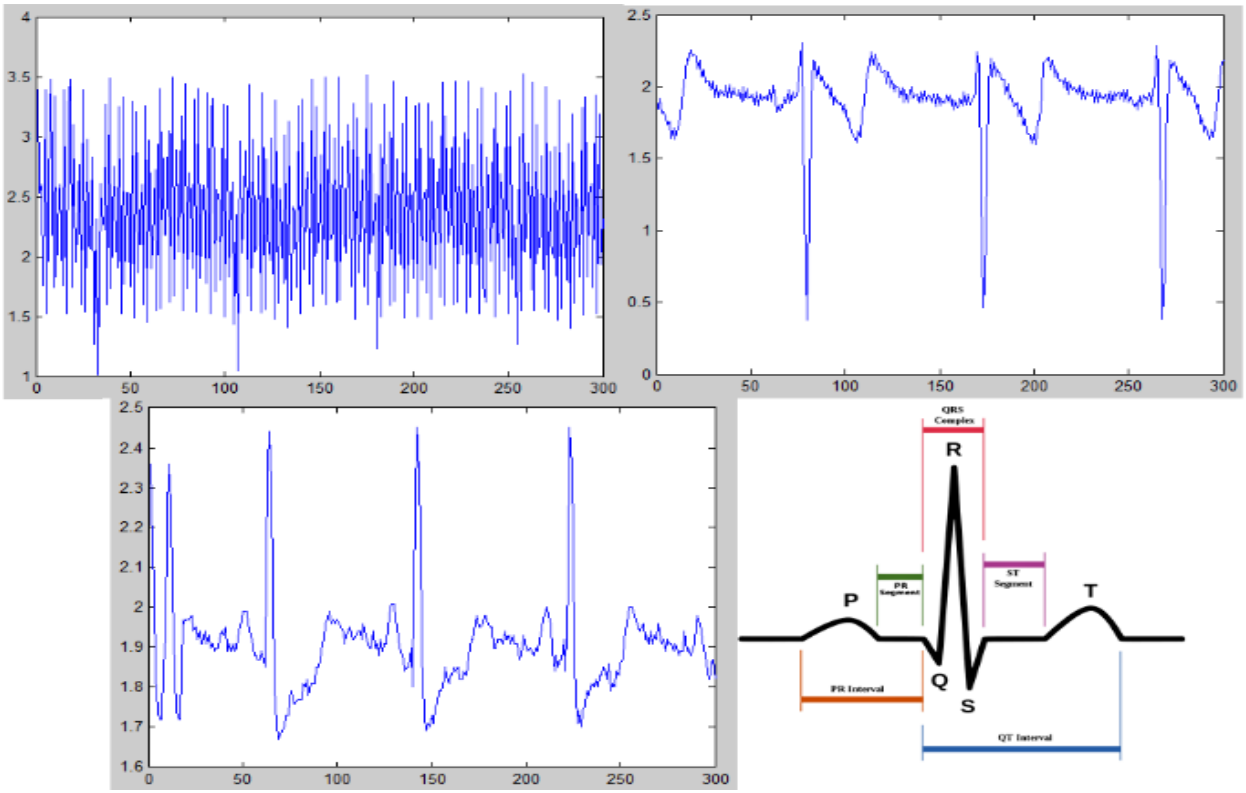
Модели коишто ги исполнуваат овие барања и коишто јас ги одбрав за изработување на апликацијата се моделите добиени од дрва на одлучување и класифицирачки правила. Моделите што се креираат се брзи, прецизни и при имплементација можат да се претстават во форма на ако-тогаш правила(класифицирачки правила извлечени од добиеното дрво на одлучување).

Податочно множество, предпроцесирање и градење на модел

Очигледно, покрај позитивните примероци(електрокардиограмите) потребни беа и негативни примероци(сигнали што не се електрокардиограми). Прво се направени околу 30 примероци од некои основни функции како синус, косинус, квадратна, логаритамска, експоненцијална и други функции, како и примероци со случајно одбрани вредности од неколку такви функции. Со оглед на тоа што ваквите функции ќе ги препознае многу лесно поради првиот атрибут, не се задржав да правам повеќе од нив. Понатаму, останатиот дел се всушност случајно одбрани вредности во рангот од -1000 до 1000, од -5 до 5, од 0 до 5, од 0 до 30 и од 2 до 3. Најголем дел

од примероците се во рангот од 0 до 5, поради тоа што позитивните примероци се наоѓаат во тој ранг, се со цел подобро да се истренира моделот за да ги препознава подобро вистинските електрокардиограми во однос на сигнали што се слични на нив во дистрибуцијата на вредностите (во чијшто случај би требало да го искористи атрибутот за бројот на пикови). Притоа, под случајно одбрани вредности, се подразбира користење на Random класата во Јава каде што е имплементирана униформната распределба. Една од главните цели на апликацијата е да препознае погрешно поставување на електродите или истрошени(искористени) лепенки на електродите, па според тоа, најдобри негативни примероци би биле тие добиени со намерно погрешно поставени електроди или истрошени лепенки при изведување на, но во периодот на правење на апликацијата не ми беше достапна опремата.

Сепак, возможно и најверојатно е некои од позитивните примероци што ги направивме во првиот дел да бидат всушност негативни, односно мерени со погрешно поставување на електродите или со истрошени лепенки на електродите. Решив да направам кластерирање со цел да видам дали навистина е така. На Слика 4 можат да се видат примероци што припаѓаат во различни кластери, каде што разликата е очигледна и исто така прикажан е перфектен електрокардиограм за споредба. Сепак, не ги земав тие како негативни примероци, со оглед на тоа што како што видов се во помал број, беше потребно да се погоди бројот на кластери и да се проверат сите примероци рачно за да знаеме дали навистина поделбата е добро направена. Исто така, требаше да се одделуваат рачно и поради тоа што ова ќе одземеше дополнително време, решив да ги оставам сите примероци како позитивни. Во идно доработување на овој проект, ова е насока каде што може да се подобри квалитетот на добиениот модел понатаму.



Слика 4
3 примероци од различни кластери и еден совршен екг

Следен чекор беше земање на 3464 вредности од секое мерење, што е еквивалентно на времетраење од 30 секунди. Ова го направив поради повеќе причини:

- Како дел од атрибутите ја користам дистрибуцијата на вредностите.
- Голем дел од податоците се со различен број на вредности поради тоа што имаме мерења од различни времетраења, како и рачно мерење на времетраењето при нивното извршување.
- Некои од мерењата имаат нули на почетокот, поради тоа што различните апарати не се пуштаа во истиот момент бидејќи требаше рачно да се вклучат, како и поради доцнењето при праќање на вредности од сензорите до Ардуино програмата, па потоа до Процесинг програмата на компјутерот.

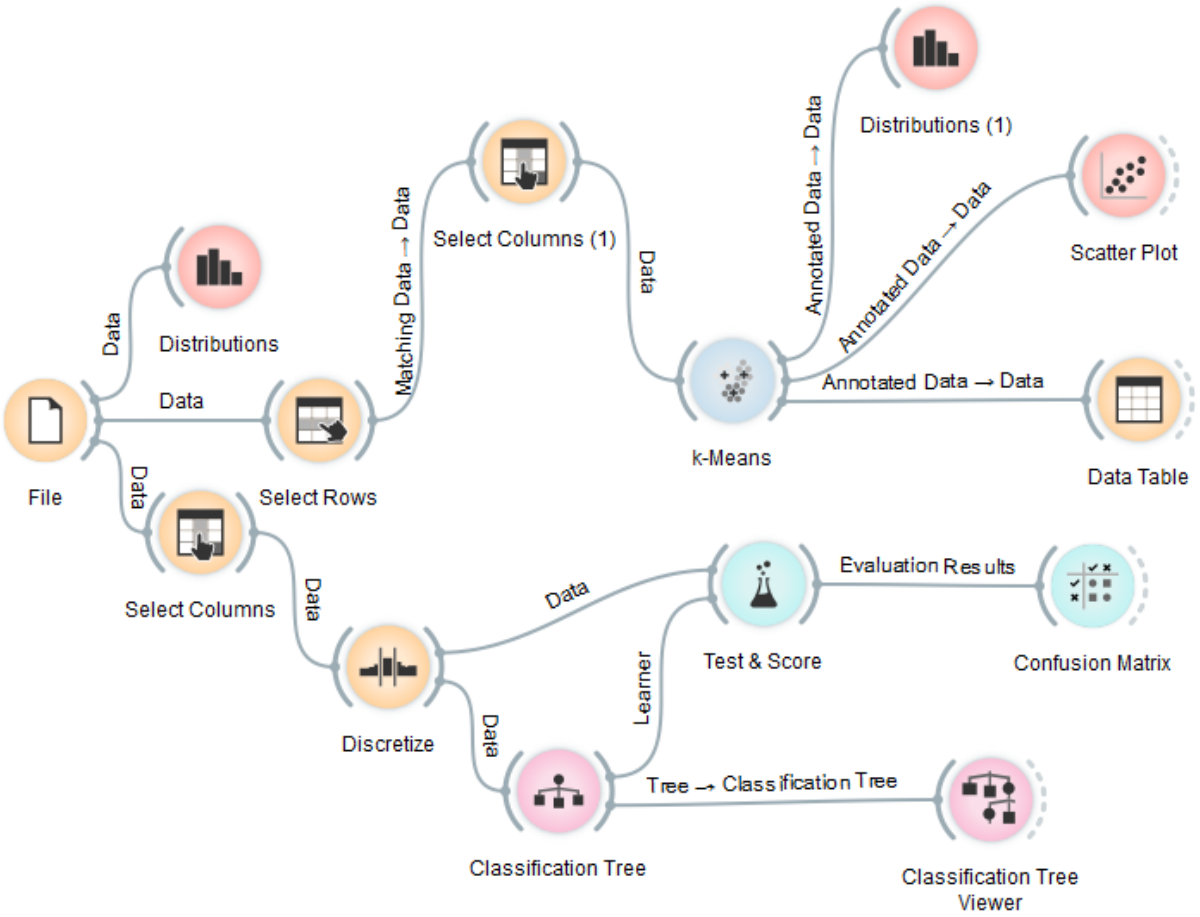
Со оглед на тоа што апликацијата може да се прилагоди да чита вредности од сензорите во реално време преку Bluetooth, подобра идеја ќе беше атрибутите да бидат процентуални репрезентации и во тој случај би можела да препознава електрокардиограми од различни времетраења. На овој начин би можеле да се искористат поголем дел од позитивните примероци при тренирање на моделот (на пример тие со времетраење од 15 секунди или повеќе од 30 секунди). До овој заклучок дојдов подоцна во изработката на проектот, па затоа не го направив на тој начин и ова би била уште една насока кадешто може да се подобри квалитетот и можностите на апликацијата во иднина.

По направеното предпроцесирање и креирање на негативни примероци, добиеното податочно множество врз коешто беше изграден моделот се состои од 151 позитивни примероци и 130 негативни примероци. Дел од кодот за креирање на множ. може да се види на Слика 5.

```
for (File measurement : data) {
    int a1 = 0, b1 = 0, b2 = 0, b3 = 0, b4 = 0, b5 = 0, c1 = 0, id;
    float value, vStart = 0, vEnd = 0;
    boolean first = true;

    input = new Scanner(new FileInputStream(measurement));
    while (input.hasNextLine()) {
        value = Float.parseFloat(input.nextLine());
        if (first) {
            vStart = vEnd = value;
            first = false;
        }

        if (value >= 0 && value < 1.4)
            b1++;
        else if (value >= 1.4 && value < 1.9)
            b2++;
        else if (value >= 1.9 && value < 2.5)
            b3++;
        else if (value >= 2.5 && value < 3.5)
            b4++;
        else if (value >= 3.5 && value <= 5)
            b5++;
        else
            a1++;

        if (value >= vEnd)
            vEnd = value;
        else {
            if (vEnd - vStart >= 0.4)
                c1++;
            vStart = vEnd = value;
        }
    }
    input.close();
    id = Integer.parseInt(measurement.getName().split("\\.")[0]);
    if (id > 100 && id < 400)
        sb.append(String.format("%s,%d,%d,%d,%d,%d,%d,%d,%d,P\n", measurement.getName().split("\\.")[0], a1,
                b1, b2, b3, b4, b5, c1));
    else
        sb.append(String.format("%s,%d,%d,%d,%d,%d,%d,%d,%d,N\n", measurement.getName().split("\\.")[0], a1,
                b1, b2, b3, b4, b5, c1));
```

Слика 5
Дел од кодот за креирање на податочното множество

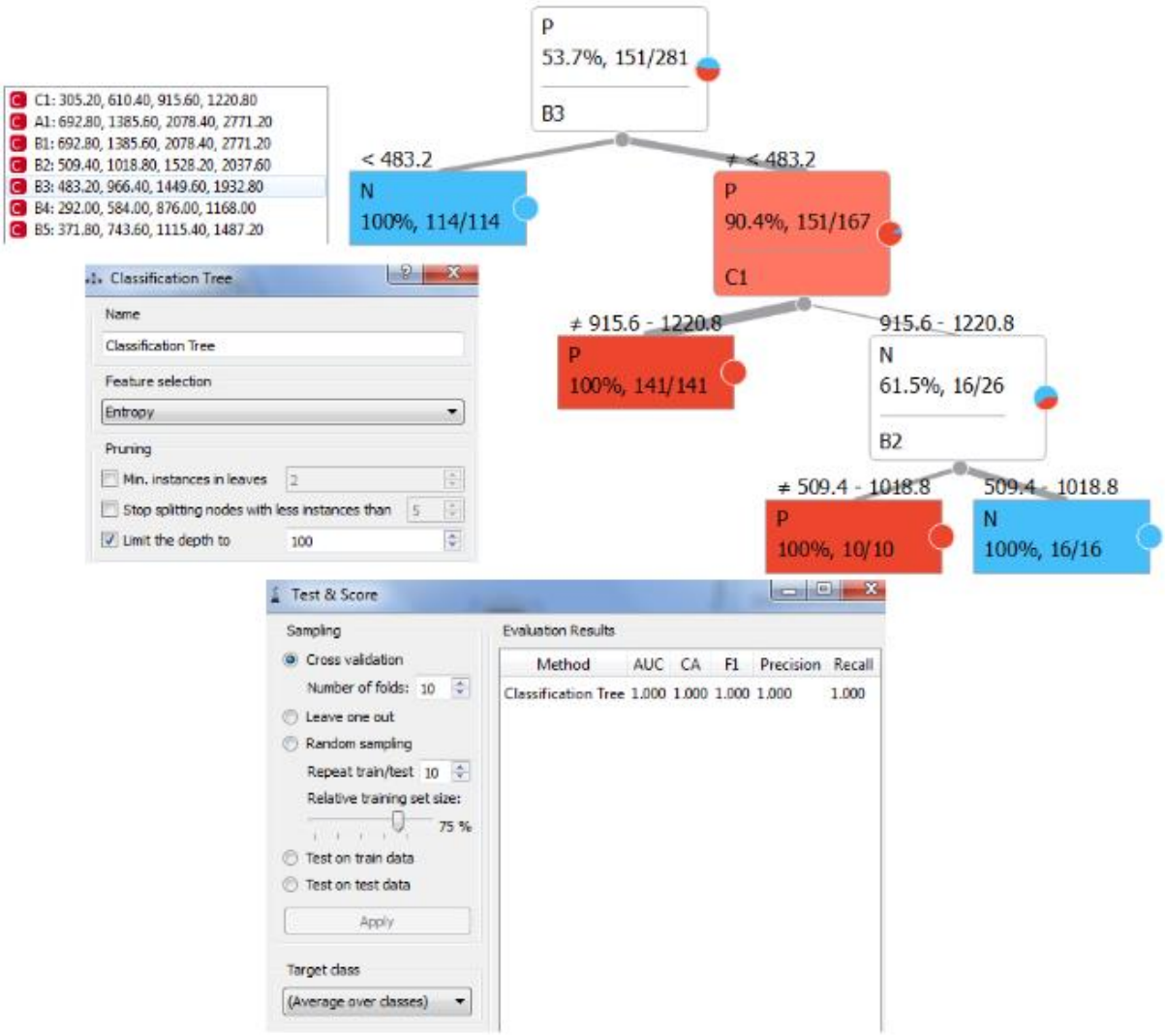
Поради тоа што моделот е дрво на одлучување, потребна е дискретизација на атрибутите. Дискретизацијата, како и дел од анализата на податочното множество, кластерирањето, градењето на моделот и тестирање на неговата точност се направени во Orange софтверот. Целиот процес може да се види на Слика 6. Дискретизацијата ја направив со бинување во 5 бина по еднакви должини (добиената дискретизација исто така е прикажана на Слика 7). Овој чекор исто така може дополнително да се подобрува преку тестирање на друг вид на дискретизација или бројот на биновите.



Слика 6
Градење на моделот и предпроцесирање

Откако се извршени сите овие чекори, го трениравме моделот, добиен со ентропија како мерка за најдобар тест при поделбата на податоците (добиениот модел и конфигурацијата се прикажани на Слика 7). Тоа што може да се забележи од моделот е дека се добива прилично едноставно дрво, коешто при класификација на невиден примерок не би ги искористило целосно сите атрибути. Неговата точност добиена со 10-фолд крос-валидација е 100% (Слика 7), што значи дека за податоците коишто се употребени имаме совршена поделба користејќи ги атрибутите што ги одбравме. Сепак, морам да нагласам дека најверојатно неговата генерализација нема да биде совршена и очекувам да прави погрешни класифицирања во некои случаи. Ова би се поправило со поквалитетно податочно множество (вклучувајќи подобри позитивни и подобри негативни примероци, во смисла дискутирана претходно). Исто така, очекувам и моделот изграден на такво множество да биде покомплексен и во најмала мерка, да ги искористи сите достапни атрибути при класифицирање на нови примероци.

Понатаму за имплементација на овој модел во Андроид апликацијата, се екстрактираат класификациски правила од дрвото на одлучување кои се претставени како ако-тогаш правила во кодот.



Слика 7
Дискретизација, конфигурација и точност на моделот, како и самиот модел

Апликација

Апликацијата е прилично едноставна и е составена од две активности: главната активност, односно почетната што се појавува откако ќе се упали апликацијата и активноста за пребарување на датотеката што ќе се тестира дали содржи електрокардиограм сигнал.

Главната активност е составена на следниот начин:

- activity_main - одговорна за поставување на структурата и изгледот. Главни елементи од тука се копчето Browse коешто води кон другата активност за пребарување на датотеката, текстот којшто го прикажува името на датотеката што се разгледува(иницијално празен) и текстот што го дава резултатот(исто така иницијално празен).

- MainActivity е класата одговорна за логиката. Составена е од дел што служи за барање на дозвола од корисникот за користење на меморијата на мобилниот за пребарување на датотеката што ќе се тестира и понатаму, главната функција за иницијализација, и две функции кои се поврзани со логиката на копчето Browse(функцијата каде што се клика копчето и се стартува активноста за пребарување на датотека и функцијата што чека резултат(односно одбрана датотека), го обработува и прикажува на екран).

FileAnalysis е класата кадешто најважниот дел од проектот се одвива, односно се анализира датотеката што е одбрана(се креира инстанца во MainActivity во функцијата што чека на резултат откако е одбрана датотека). Во оваа класа прво се проверува форматот на датотеката и се земаат потребните податоци(односно вредностите за атрибутите) и потоа според моделот се гледа дали е електрокардиограм или не е. Во однос на проверката на форматот тоа што се проверува е дали секоја линија е децимален број и дали се вкупно 3464 вредности. Овој дел може да се поправи во иднина ако се прават атрибутите процентуални. Како што беше претходно дискутирано, моделот(прикажан на Слика 7) е претставен со ако-тогаш правила и кодот за овој модел може да се види на Слика 8.

```
// Built Model
String positive = "The Signal Is ECG!", negative = "The Signal Is Not ECG!";
if (b3 >= 483.2) {
    if (c1 >= 915.6 && c1 <= 1220.8) {
        if (b2 >= 509.4 && b2 <= 1018.8)
            return negative;
        else
            return positive;
    } else
        return positive;
} else
    return negative;
```

Слика 8
Моделот за препознавање на електрокардиограм

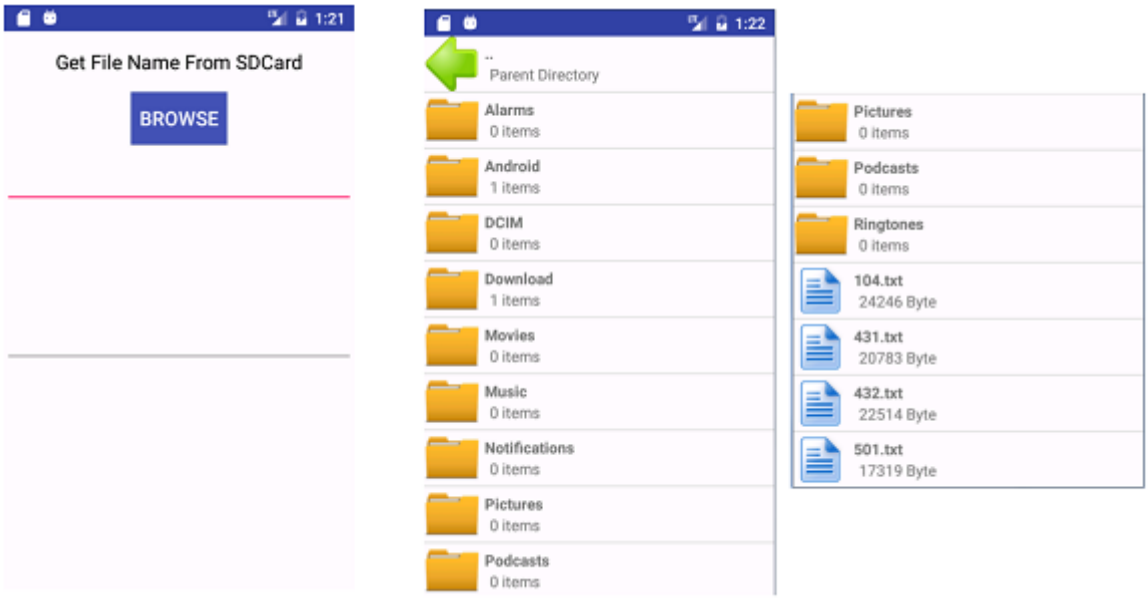
Активноста за пребарување на датотека се состои од:

- file_view – исто така одговорна за изгледот и структурата на активноста. Ја прикажува листата на директориуми и датотеки што се содржат во тековниот директориум. Има копче за да се вратиме назад во хиерархијата и соодветно на секој директориум/датотека може да се кликне. Ако се одбере датотека, тогаш таа се враќа како резултат кон главната активност. За директориум се прикажува бројот на датотеки/директориуми што ги содржи, додека за датотека големината во бајти.
- FileChooser е главната класа одговорна за логиката на прелистување на содржината во телефонот. При креирање на инстанцата, се поставува почетниот директориум и се креира листа на датотеки/директориуми што ги содржи, која понатаму се прикажува на екран(ова се одвива во FileArrayAdapter класата). Како почетен директориум се поставува мемориската картичка, што може да претставува проблем ако ја нема. Исто така, корисникот може намерно да предизвика грешка со константно стискање на иконата за враќање назад во хиерархијата на директориуми(во случај да сме стигнале до коренот). Друго што може да се додаде е ограничување на датотеките/директориумите што се прикажуваат(според некој критериуми) поради безбедност и ефикасност, со оглед на тоа што се земаат сите можни датотеки и директориуми. Сите овие потенцијални грешки се друго поле на можно подобрување

во иднина. Конечно, тука се наоѓа и функцијата за логиката при кликање на датотека/директориум што веќе ја дискутирав претходно.

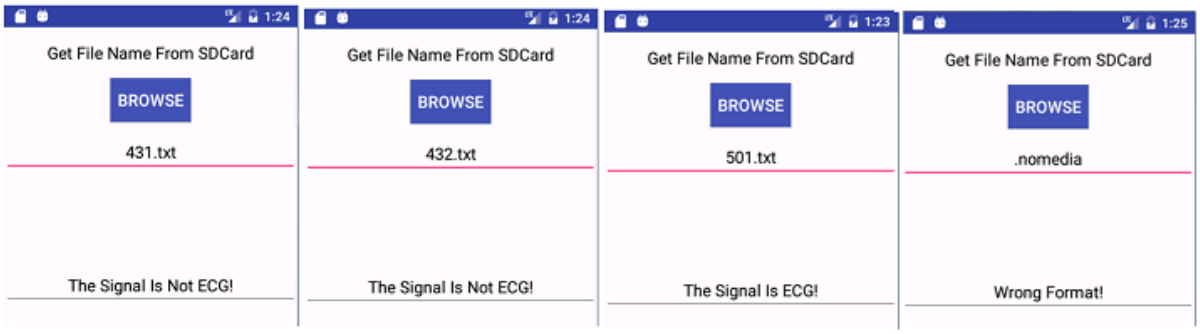
- FileArrayAdapter е помошна класа на FileChooser што содржи стандарден код за пополнување на Layoult елементот во file_view.
- Item е исто така помошна класа што го претставува секој елемент од листата(датотека/директориум) и го содржи името, големината и ознака за типот.

На слика 9 е прикажан изгледот на секоја активност.



Слика 9
Изглед на секоја активност

На Слика 10 е прикажано тестирање на апликацијата со 3 датотеки – примероци што не се користени при градење на моделот. Првата датотека е обичен синусен сигнал, втората се случајни вредности во рангот од 0 до 5, третата е електрокардиограм и дополнително имаме една датотека со погрешен формат.



Слика 10
Тестирање – синус, случајни вредности, електрокардиограм и погрешен формат