**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ**

**Кафедра информационных систем**

**ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ  
по курсу: Теория информационных процессов и систем**

**Отбор признаков**

**Работу выполнили студенты:**

**Прялкова Юлия**

**группа 3513**

**Буйчик Антон**

**группа 3514**

**САНКТ-ПЕТЕРБУРГ**

**2014**

В качестве исходных данных для лабораторной работы была взята статистика людей, проходивших исследования на наличие сердечно сосудистых заболеваний. Критериями являются 13 различных параметров, в их числе возраст, пол, тип боли в груди, уровень холестерина и другие. Все критерии имеют числовой тип, те легко их легко изучать. Результирующим критерием является отсутствие (1) или присутствие (2) сердечно сосудистых заболеваний. Данная статистика имеет 270 экземпляров.

Внешним критерием для оценки был выбран критерий скользящего контроля.

В качестве первого алгоритма отбора было выбрано последовательное добавление признаков. Мы выбираем тот признак, при использовании которого в контрольной выборке получается наибольшее количество правильных предсказаний. Останавливается алгоритм тогда, когда это количество начинает уменьшаться.

Алгоритм отбора признаков был написан на языке Java с применение библиотеки Weka.

Исходный код:

**package** temp;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Random;

**import** java.util.Stack;

**import** java.util.StringTokenizer;

**import** weka.classifiers.Evaluation;

**import** weka.classifiers.trees.J48;

**import** weka.core.Instances;

**import** weka.filters.Filter;

**import** weka.filters.unsupervised.attribute.Remove;

**public** **class** main {

**public** **static** **void** main(String[] args) **throws** Exception {

BufferedReader reader = **new** BufferedReader(

**new** FileReader("data.arff"));

Instances data = **new** Instances(reader);

reader.close();

// setting class attribute

data.setClassIndex(data.numAttributes() - 1);

ArrayList<Integer> atrInd=**new** ArrayList<Integer>();

Stack<Double> atrError=**new** Stack<Double>();

ArrayList<Double> temp=**new** ArrayList<Double>();

ArrayList<Integer> temp2=**new** ArrayList<Integer>();

**while** (**true**){

temp.clear();

temp2.clear();

**for** (**int** i=1; i<14; i++){

**boolean** flag=**false**;

String[] options = **new** String[3];

options[0] = "-R";

options[1] = "14,"+Integer.*toString*(i);

**for** (**int** j=0; j<atrInd.size(); j++){

**if** (atrInd.get(j)!=i){

options[1]+=","+Integer.*toString*(atrInd.get(j));

}**else**{

flag=**true**;

**break**;

}

}

**if** (flag){

**continue**;

}

options[2]="-V";

Remove remove = **new** Remove();

remove.setOptions(options);

remove.setInputFormat(data);

Instances newData = Filter.*useFilter*(data, remove);

String[] options2 = **new** String[1];

options2[0] = "-U";

J48 tree = **new** J48();

tree.setOptions(options2);

tree.buildClassifier(newData);

Evaluation eval = **new** Evaluation(newData);

eval.crossValidateModel(tree, newData, 10, **new** Random(1));

temp.add(eval.correct());

temp2.add(i);

}

**int** maxInd=temp.indexOf(Collections.*max*(temp));

//System.out.println(temp2.get(maxInd));

//System.out.println(temp.get(maxInd));

**if**(!atrError.isEmpty()){

**if** (temp.get(maxInd)<atrError.lastElement()){

**break**;

}

}

atrError.add(temp.get(maxInd));

atrInd.add(temp2.get(maxInd));

//System.in.read();

}

**for** (Integer ind : atrInd){

System.*out*.println(data.attribute(ind-1));

}

System.*out*.println(atrError.lastElement());

}

}

Результаты:

В результате отбора признаков были выбраны 4:

* результат сканирования сердца с помощью таллия (13 атрибут – thal)
* количество крупных сосудов, найденных с помощью рентгеноскопии (12 атрибут - number of major vessels colored by fluoroscopy)
* тип боли в груди (3 атрибут – chest pain type)
* уровень сахара (6 атрибут – fasting blood sugar)

В качестве второго алгоритма отбора было выбрано построение и обход дерева возможных наборов признаков с использованием поиска в глубину. Алгоритм ищем наилучший набор атрибутов с максимальным количеством правильных предсказаний.

Алгоритм отбора признаков был написан на языке Java с применение библиотеки Weka.

**package** temp;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Random;

**import** java.util.Stack;

**import** java.util.StringTokenizer;

**import** weka.classifiers.Evaluation;

**import** weka.classifiers.trees.J48;

**import** weka.core.Instances;

**import** weka.filters.Filter;

**import** weka.filters.unsupervised.attribute.Remove;

**public** **class** main2 {

**public** **static** Instances *data*;

**public** **static** BufferedReader *tree*;

**public** **static** Double *bestCorrect*=(**double**) 0;

**public** **static** String *bestAtribute*;

**public** **static** Remove *remove* = **new** Remove();

**public** **static** **void** func(**double** lastCorrect, String lastRangeList) **throws** Exception{

String[] splitted=lastRangeList.split(",");

**int** t;

**if** (Integer.*parseInt*(splitted[splitted.length-1])!=14){

t=Integer.*parseInt*(splitted[splitted.length-1]);

}**else**{

t=0;

}

**for** (**int** j=t+1; j<14; j++){

String[] options = **new** String[3];

options[0] = "-R";

options[1] = lastRangeList+","+Integer.*toString*(j);

String rangeList=options[1];

options[2]="-V";

*remove*.setOptions(options);

Instances newData = Filter.*useFilter*(*data*, *remove*);

String[] options2 = **new** String[1];

options2[0] = "-U";

J48 tree = **new** J48();

tree.setOptions(options2);

tree.buildClassifier(newData);

Evaluation eval = **new** Evaluation(newData);

eval.crossValidateModel(tree, newData, 10, **new** Random(1));

**if** (eval.correct()>*bestCorrect*){

*bestCorrect*=eval.correct();

*bestAtribute*=rangeList;

}

**if** (lastCorrect>eval.correct()){

**continue**;

}

*func*(eval.correct(), rangeList);

}

}

**public** **static** **void** main(String[] args) **throws** Exception {

BufferedReader reader = **new** BufferedReader(

**new** FileReader("data.arff"));

*data* = **new** Instances(reader);

reader.close();

*data*.setClassIndex(*data*.numAttributes() - 1);

*remove*.setInputFormat(*data*);

String s="14";

*func*(-1, s);

System.*out*.println(*bestCorrect*);

System.*out*.println(*bestAtribute*);

String[] spl=bestAtribute.split(",");

String[] options=new String[3];

ArrayList<Double> temp=new ArrayList<Double>();

ArrayList<Integer> temp2=new ArrayList<Integer>();

for (String st : spl){

if (Integer.parseInt(st)!=14){

options[0] = "-R";

options[1] = "14,"+Integer.parseInt(st);

options[2]="-V";

//System.out.println(options[1]);

Remove remove2 = new Remove();

remove2.setOptions(options);

remove2.setInputFormat(data);

Instances newData = Filter.useFilter(data, remove2);

String[] options2 = new String[1];

options2[0] = "-U";

J48 tree = new J48();

tree.setOptions(options2);

tree.buildClassifier(newData);

Evaluation eval = new Evaluation(newData);

eval.crossValidateModel(tree, newData, 10, new Random(1));

temp.add(eval.correct());

temp2.add(Integer.parseInt(st));

}

}

System.out.println(temp);

System.out.println(temp2);

}

}

Результаты:

В результате отбора признаков были выбраны 4:

* тип боли в груди (3 атрибут – chest pain type)
* уровень сахара (6 атрибут – fasting blood sugar)
* количество крупных сосудов, найденных с помощью рентгеноскопии (12 атрибут - number of major vessels colored by fluoroscopy)
* результат сканирования сердца с помощью таллия (13 атрибут – thal)

Из-за того, что при построении дерева в этом алгоритме атрибуты идут в порядке возрастания их индекса, результат нельзя интерпретировать правильно без дополнительных расчетов. Например, можно для каждого атрибута из полученного набора рассчитать ошибку предсказания, если для этот будет использоваться только один атрибут. А затем отсортировать их по уменьшению ошибки (в нашем случае, по увеличению количества правильных предсказаний). Тогда получается набор атрибутов вида 13, 12, 3, 6.

Вывод: доработав второй алгоритм и сравнив результаты с первым видно, что наиболее важным атрибутом является 13 - результат сканирования сердца с помощью таллия, что соответствует действительности.