import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.ArrayList;

class treeNode{

private String sname;

public treeNode(String str) {

sname=str;

}

public String getsname() {

return sname;

}

ArrayList<String> label=new ArrayList<String>();

ArrayList<treeNode> node=new ArrayList<treeNode>();

}

public class ID3 {

private ArrayList<String> label=new ArrayList<String>();

private ArrayList<ArrayList<String>> date=new ArrayList<ArrayList<String>>();

private ArrayList<ArrayList<String>> test=new ArrayList<ArrayList<String>>();

private ArrayList<String> sum=new ArrayList<String>();

private String kind;

public ID3(String path,String path0) throws FileNotFoundException {

getDate(path);

gettestDate(path0);

init(date);

}

public void init(ArrayList<ArrayList<String>> date) {

sum.add(date.get(0).get(date.get(0).size()-1));

for(int i=0;i<date.size();i++) {

if(sum.contains(date.get(i).get(date.get(0).size()-1))==false) {

sum.add(date.get(i).get(date.get(0).size()-1));

}

}

}

public void gettestDate(String path) throws FileNotFoundException {

String str;

int i=0;

try {

FileInputStream fis = new FileInputStream(path);

InputStreamReader isr = new InputStreamReader(fis, "UTF-8");

BufferedReader in = new BufferedReader(isr);

while((str=in.readLine())!=null) {

String[] strs=str.split(",");

ArrayList<String> line =new ArrayList<String>();

for(int j=0;j<strs.length;j++) {

line.add(strs[j]);

}

test.add(line);

i++;

}

in.close();

}catch(Exception e) {

e.printStackTrace();

}

}

public void getDate(String path) throws FileNotFoundException {

String str;

int i=0;

try {

FileInputStream fis = new FileInputStream(path);

InputStreamReader isr = new InputStreamReader(fis, "UTF-8");

BufferedReader in = new BufferedReader(isr);

while((str=in.readLine())!=null) {

if(i==0) {

String[] strs=str.split(",");

for(int j=0;j<strs.length;j++) {

label.add(strs[j]);

}

i++;

continue;

}

String[] strs=str.split(",");

ArrayList<String> line =new ArrayList<String>();

for(int j=0;j<strs.length;j++) {

line.add(strs[j]);

}

date.add(line);

i++;

}

in.close();

}catch(Exception e) {

e.printStackTrace();

}

}

public double Ent(ArrayList<ArrayList<String>> dat) {

int all=0;

double amount=0.0;

for(int i=0;i<sum.size();i++) {

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

if(sum.get(i).equals(dat.get(j).get(dat.get(0).size()-1))) {

all++;

}

}

if((double)all/dat.size()==0.0) {

continue;

}

amount+=((double)all/dat.size())\*(Math.log(((double)all/dat.size()))/Math.log(2.0));

all=0;

}

if(amount==0.0) {

return 0.0;

}

return -amount;

}

public double condtion(int a,ArrayList<ArrayList<String>> dat) {

ArrayList<String> all=new ArrayList<String>();

double c=0.0;

all.add(dat.get(0).get(a));

for(int i=0;i<dat.size();i++) {

if(all.contains(dat.get(i).get(a))==false) {

all.add(dat.get(i).get(a));

}

}

ArrayList<ArrayList<String>> plus=new ArrayList<ArrayList<String>>();

ArrayList<ArrayList<ArrayList<String>>>

count=new ArrayList<ArrayList<ArrayList<String>>>();

for(int i=0;i<all.size();i++) {

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

if(true==all.get(i).equals(dat.get(j).get(a))) {

plus.add(dat.get(j));

}

}

count.add(plus);

c+=((double)count.get(i).size()/dat.size())\*Ent(count.get(i));

plus.removeAll(plus);

}

return (Ent(dat)-c);

}

public int Gain(ArrayList<ArrayList<String>> dat) {

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

for(int i=0;i<dat.get(0).size()-1;i++) {

num.add(condtion(i,dat));

}

int index=0;

double max=num.get(0);

for(int i=1;i<num.size();i++) {

if(max<num.get(i)) {

max=num.get(i);

index=i;

}

}

return index;

}

public treeNode creattree(ArrayList<ArrayList<String>> dat) {

int index=Gain(dat);

treeNode node=new treeNode(label.get(index));

ArrayList<String> s=new ArrayList<String>();

s.add(dat.get(0).get(index));

for(int i=1;i<dat.size();i++) {

if(s.contains(dat.get(i).get(index))==false) {

s.add(dat.get(i).get(index));

}

}

ArrayList<ArrayList<String>> plus=new ArrayList<ArrayList<String>>();

ArrayList<ArrayList<ArrayList<String>>>

count=new ArrayList<ArrayList<ArrayList<String>>>();

for(int i=0;i<s.size();i++) {

node.label.add(s.get(i));

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

if(true==s.get(i).equals(dat.get(j).get(index))) {

plus.add(dat.get(j));

}

}

count.add(plus);

int k;

String str=count.get(i).get(0).get(count.get(i).get(0).size()-1);

for(k=1;k<count.get(i).size();k++) {

if(false==str.equals(count.get(i).get(k).get(count.get(i).get(k).size()-1))) {

break;

}

}

if(k==count.get(i).size()) {

treeNode dd=new treeNode(str);

node.node.add(dd);

}

else {

node.node.add(creattree(count.get(i)));

}

plus.removeAll(plus);

}

return node;

}

public void print(ArrayList<ArrayList<String>> dat) {

System.out.println("构建的决策树如下：");

treeNode node=null;

node=creattree(dat);//类

put(node);//递归调用

}

public void put(treeNode node) {

System.out.println("结点："+node.getsname()+"\n");

for(int i=0;i<node.label.size();i++) {

System.out.println(node.getsname()+"的标签属性:"+node.label.get(i));

if(node.node.get(i).node.isEmpty()==true) {

System.out.println("叶子结点："+node.node.get(i).getsname());

}

else {

put(node.node.get(i));

}

}

}

public void testdate(ArrayList<ArrayList<String>> test,String path) throws IOException {

treeNode node=null;

int count=0;

node=creattree(this.date);

try {

BufferedWriter out=new BufferedWriter(new FileWriter(path));

for(int i=0;i<test.size();i++) {

testput(node,test.get(i));

for(int j=0;j<test.get(i).size();j++) {

out.write(test.get(i).get(j)+",");

}

if(kind.equals(date.get(i).get(date.get(i).size()-1))==true) {

count++;

}

out.write(kind);

out.newLine();

}

System.out.println("该次分类结果正确率为："+(double)count/test.size()\*100+"%");

out.flush();

out.close();

}catch(IOException e) {

e.printStackTrace();

}

}

public void testput(treeNode node,ArrayList<String> t) {

int index=0;

for(int i=0;i<this.label.size();i++) {

if(this.label.get(i).equals(node.getsname())==true) {

index=i;

break;

}

}

for(int i=0;i<node.label.size();i++) {

if(t.get(index).equals(node.label.get(i))==false) {

continue;

}

if(node.node.get(i).node.isEmpty()==true) {

this.kind=node.node.get(i).getsname();//取出分类结果

}

else {

testput(node.node.get(i),t);

}

}

}

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

String data="C:\\Users\\zfw\\Desktop\\data1.txt";

String test="C:\\Users\\zfw\\Desktop\\test.txt";

String result="C:\\Users\\zfw\\Desktop\\result.txt";

ID3 id=new ID3(data,test);

id.print(id.date);

id.testdate(id.test,result);

}

}