**package** WRD\_Project;

**import** weka.core.Instances;

**import** weka.attributeSelection.\*;

**import** weka.core.converters.CSVLoader;

**import** weka.filters.Filter;

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

**import** java.io.File;

**import** java.lang.Math;

**import** java.util.\*;

**public** **class** WRD\_Ensemble\_Class {

/\* Variable declaration:

\* F is a two dimensional array for Weight and Mean Weight of features for GR, ReliefF, PCC and WRD\_Ensemble Method

\* Rank is a two dimensional array for Rank and Absolute Rank Difference of Features for GR,ReliefF, PCC and WRD\_Ensemble Method

\* indices is a two dimensional array to keep index of a feature which is a candidate to remove from feature list

\* W is a two dimensional array for Rank and Weight of each feature of a FS method

\* Rank\_diff\_abs is a two dimensional array for absolute rank difference of each feature

\*/

**public** **static** **double** [][] *F* = **new** **double**[1100][1000];

**public** **static** **int** [][] *Rank* = **new** **int** [1100][1000];

**public** **static** **int** [] *indices* = **new** **int** [1000];

**public** **static** **double** [][] *Data\_Array*=**new** **double**[1100][1000];

**public** **static** **double**[][] *W* = **new** **double**[1100][1000];

**public** **static** **double**[][] *Rank\_diff\_abs* = **new** **double**[1000][1000];

**public** **static** **void** sortbyColumn(**double** arr[][], **int** col)

{

// Function sortbyColumn to sort an array by column

Arrays.*sort*(arr, **new** Comparator<**double**[]>() {

//built-in sort function Arrays.sort

@Override

// Compare values according to columns

**public** **int** compare(**final** **double**[] entry1,

**final** **double**[] entry2) {

//sort in descending order revert

**if** (entry1[col] < entry2[col])

**return** 1;

**else**

**return** -1;

}

});

}

**protected** **static** **void** GainRatio(Instances data) **throws** Exception {

/\*Function GainRatio(),input to this function is the dataset D and it generates rank and weight of features using GR FS method \*/

// AttributeSelection(),GainRatioAttributeEval() and Ranker() are built in classes of Weka

AttributeSelection attsel = **new** AttributeSelection(); // create and initiate a new AttributeSelection instance

GainRatioAttributeEval eval = **new** GainRatioAttributeEval(); // choose GainRatio as an evaluation method

Ranker search = **new** Ranker(); // choose a ranking search method

attsel.setEvaluator(eval);

attsel.setSearch(search);

**try** {

attsel.SelectAttributes(data);

*W* = attsel.rankedAttributes();

} **catch** (Exception e) {

e.printStackTrace();

}

**int** n=*W*.length;

**int** row;

**for** (**int** j=0;j<n;j++) {

row=(**int**)(*W*[j][0]);

*Rank*[row][0]=j+1;// rank of attributes

*F*[row][0]=*W*[j][1];//weight or score of attribute

System.***out***.println("Attributes "+ (row+1)+" Rank "+*Rank*[row][0]+" Weight "+ *W*[j][1]);

}

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

}

**protected** **static** **void** RelF(Instances data) **throws** Exception {

/\*Function RelF(),input to this function is the dataset D and it generates rank and weight of features using ReleifF FS method \*/

//AttributeSelection(),ReliefFAttributeEval() and Ranker() are built in classes of Weka

AttributeSelection attsel = **new** AttributeSelection(); // create and initiate a new AttributeSelection instance

ReliefFAttributeEval eval= **new** ReliefFAttributeEval(); // choose ReliefF as an evaluation method

Ranker search = **new** Ranker(); // choose a ranking search method

attsel.setEvaluator(eval);

attsel.setSearch(search);

**try** {

attsel.SelectAttributes(data);

*W* = attsel.rankedAttributes();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**int** n=*W*.length;

**int** row;

**for** (**int** j=0;j<n;j++) {

row=(**int**)(*W*[j][0]);

*Rank*[row][1]=j+1;

*F*[row][1]=*W*[j][1];

System.***out***.println("Attributes "+(row+1)+" Rank "+*Rank*[row][1]+" Weight "+ *W*[j][1]);

}

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

}

**protected** **static** **void** Corr(Instances data) **throws** Exception {

/\*Function Corr(),input to this function is the dataset D and it generates rank and weight of features using PCC FS method \*/

//AttributeSelection(),CorrelationAttributeEval() and Ranker() are built in classes of Weka

AttributeSelection attsel = **new** AttributeSelection(); // create and initiate a new AttributeSelection instance

CorrelationAttributeEval eval = **new** CorrelationAttributeEval(); // choose sPCC as an evaluation method

Ranker search = **new** Ranker(); // choose a ranking search method

attsel.setEvaluator(eval);

attsel.setSearch(search);

**try** {

attsel.SelectAttributes(data);

*W* = attsel.rankedAttributes();

} **catch** (Exception e) {

e.printStackTrace();

}

**int** n=*W*.length;

**int** row;

**for** (**int** j=0;j<n;j++) {

row=(**int**)(*W*[j][0]);

*Rank*[row][2]=j+1;

*F*[row][2]=*W*[j][1];

System.***out***.println("Attributes "+ (row+1)+" Rank "+*Rank*[row][2]+" Weight "+ *W*[j][1]);

}

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

}

**protected** **static** **void** WRD\_Ensemble(Instances data,**int** NumAtt) **throws** Exception {

/\*Function WRD\_Ensemble(),input to this function is the dataset D and number of attributes. It generates mean weight and absolute rank difference of features using WRD\_Ensemble FS method \*/

**int** row,col;

**double** cal\_mean=0.0;

**int** rank\_dis,rank\_dis1,rank\_dis2;

System.***out***.println("Rank number of an Attribute in GR,ReliefF and PCC FS methods Respectively");

**for**(row=0;row<NumAtt;row++)

{

**for**(col=0;col<3;col++)

System.***out***.println("Attribute "+(row+1)+" Ranks "+(*Rank*[row][col]));

System.***out***.println("\n");

}

//calculate absolute distances

**for**(row=0;row<NumAtt;row++)

{

col=0;

rank\_dis=*Rank*[row][0]-*Rank*[row][1];

rank\_dis=Math.*abs*(rank\_dis);

rank\_dis1=(*Rank*[row][0]-*Rank*[row][2]);

rank\_dis+=Math.*abs*(rank\_dis1);

rank\_dis2=*Rank*[row][1]-*Rank*[row][2];

rank\_dis+=Math.*abs*(rank\_dis2);

*Rank*[row][3]=rank\_dis;

}

**int** Max\_dis=-999,min\_dis=999;

**for**(row=0;row<NumAtt;row++)

{

**if**(*Rank*[row][3]>Max\_dis)

Max\_dis=*Rank*[row][3];

**if**(*Rank*[row][3]<min\_dis)

min\_dis=*Rank*[row][3];

}

**double** db\_rank,db\_max\_dis,db\_min\_dis,abs\_rank\_diff;

db\_max\_dis=Max\_dis;

db\_min\_dis=min\_dis;

//calculate mean weight and absolute rank difference

**for**(row=0;row<NumAtt;row++)

{

cal\_mean=0.0;

**for**(col=0;col<3;col++)

cal\_mean+=*F*[row][col];

*F*[row][col]=cal\_mean/3;

db\_rank=*Rank*[row][col];

*Rank\_diff\_abs*[row][1] = (db\_rank-db\_min\_dis)/(db\_max\_dis-db\_min\_dis);

// print Attribute, Mean Weight and Normalized Absolute Rank difference

System.***out***.println("Attribute "+(row+1) +" Mean Weight "+*F*[row][col]+" Rank\_difference "+(db\_rank-db\_min\_dis)/(db\_max\_dis-db\_min\_dis));

//System.out.println("Attribute "+(row+1) +" Rank\_difference "+ Rank\_diff\_abs[row][1] );

}

**for**(row=0;row<NumAtt;row++)

{

*Data\_Array*[row][0]=row+1;

*Data\_Array*[row][1]=*F*[row][3];

}

*sortbyColumn*(*Data\_Array*, 1);

**int** Row;

System.***out***.println("\nWRD-Ensemble \n");

**for**(row=0;row<NumAtt;row++)

{

Row=(**int**)*Data\_Array*[row][0];

System.***out***.println("Attribute "+*Data\_Array*[row][0] +" Mean Weight "+*Data\_Array*[row][1]);

}

}

**protected** **static** **void** RemoveAttr\_for\_Weight\_threshold(Instances data, **int** numAtt) **throws** Exception {

/\*Remove Attribute for a particular threshold value of mean weight \*/

**int** row,i=0;

**double** score;

**for**(row =0;row<numAtt;row++)

{

score=*F*[row][3];

//the user has to put value of weight threshold here, not from prompt)

**if**(score<=0.10623967633558795)

{

*indices*[i]=row+1;

System.***out***.println("Attributes "+*indices*[i]+" Weight "+score);

i++;

}

}

System.***out***.println("\n");

System.***out***.println("Attribute that has weight <= thresold(0.106239676335587953) will be deleted from dataset" );

**for**(row=0;row<i;row++) {

System.***out***.println("Attributes "+*indices*[row]+" Score "+*F*[*indices*[row]-1][3]);

}

System.***out***.println("\n");

}

**protected** **static** **void** RemoveAttr\_for\_Rank\_threshold(Instances data, **int** numAtt) **throws** Exception {

/\*Remove Attribute if it's rank difference is greater than threshold value and do not have a rank equal to the last one-fourth (for CKD:19 to 24 rank) of the rank in any three rank lists. \*/

**int** row,i=0;

**double** Abs\_Rank\_diff;

**for**(row =0;row<numAtt;row++)

{

Abs\_Rank\_diff=*Rank\_diff\_abs*[row][1];

//the user has to put value of absolute rank difference has to here, not from prompt)

**if**(Abs\_Rank\_diff>=1.0)

{

*indices*[i]=row+1;

System.***out***.println("Attributes "+*indices*[i]+" has Rank\_differnce "+ Abs\_Rank\_diff);

i++;

}

}

System.***out***.println("\n");

System.***out***.println("Attribute(s) that has Absoulte Rank diffrence >= threshold(1.0) are candidate to delete from dataset" );

**for**(row=0;row<i;row++) {

System.***out***.println("Attribute "+*indices*[row]+ " Rank in GR FS Method is: "+*Rank*[*indices*[row]-1][0]);

System.***out***.println("Attribute "+*indices*[row]+ " Rank in ReliefF FS Method is: "+*Rank*[*indices*[row]-1][1]);

System.***out***.println("Attribute "+*indices*[row]+ " Rank in PCC FS Method is: "+*Rank*[*indices*[row]-1][2]);

}

}

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

CSVLoader loader = **new** CSVLoader();

// load dataset from local disk

loader.setSource(**new** File("F:\\BUET\\chronic\_kidney\_disease.csv ")); // show the path of dataset

//loader.setSource(new File("F:\\BUET\\lung cancer.csv "));

Instances D = loader.getDataSet();

*W* = **new** **double**[D.numAttributes()][2];

**if** (D.classIndex() == -1)

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

//handling missing value if there is any, otherwise this section is not necessary

Filter impute= **new** ReplaceMissingValues();

impute.setInputFormat(D);

D=Filter.*useFilter*(D, impute);

// end of missing value handling code

System.***out***.println("Feature Ranking using GR FS Method\n");

*GainRatio*(D);

System.***out***.println("Feature Ranking using ReleifF FS Method\n");

*RelF*(D);

System.***out***.println("Feature Ranking using PCC FS Method\n");

*Corr*(D);

**int** numAtt;

numAtt=D.numAttributes()-1;

*WRD\_Ensemble*(D,numAtt);

*RemoveAttr\_for\_Weight\_threshold*(D,numAtt);

*RemoveAttr\_for\_Rank\_threshold*(D,numAtt);

}

}