KNOWLEDGE DISCOVERY
TUGAS M6
AUTOMATIC CLUSTERING

Nama : Ni Putu Devira Ayu Martini

NRP : 1120800012

Kelas : S2 Elektro 2020

SOAL!

Assignment

1. Convert that data into the numerical values
2. Impute the missing data with the mean values of same attribute in the same class
3. Hide the class label of the supervised data
4. Normalize the data
5. Apply the automatic clustering. How many clusters are created?
6. Compare the clusters and the original classes of the dataset
7. What is your analysis from the experiment?
8. Make the report and upload to the website (M6)

DATASET “HEPATITIS”

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
#	Age	Sex	Steroid	Antivirals	Fatique	Malaise	A0rexia	1r Big	1r Firm	Spleen Palpable	Speiders	Ascites	Varices	Bilirubin	Alk Phosphate	SGOT	Albumin	Protime	Histology	CLASS
1	30	1	0	1	1	1	1	0	1	1	1	1	1	1	85	18	4	0	0	1
2	50	0	0	1	0	1	1	0	1	1	1	1	1	0.9	135	42	3.5	0	0	1
3	78	0	1	1	0	1	1	1	1	1	1	1	1	0.7	96	32	4	0	0	1
4	31	0	0	0	1	1	1	1	1	1	1	1	1	0.7	46	52	4	80	0	1
5	34	0	1	1	1	1	1	1	1	1	1	1	1	1	0	200	4	0	0	1
6	34	0	1	1	1	1	1	1	1	1	1	1	1	0.9	95	28	4	75	0	1
7	51	0	0	1	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0
8	23	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1
9	39	0	1	1	0	1	1	1	0	1	1	1	1	0.7	0	48	4.4	0	0	1
10	30	0	1	1	1	1	1	1	1	1	1	1	1	1	0	120	3.9	0	0	1
11	39	0	0	0	1	1	1	0	0	1	1	1	1	1.3	78	30	4.4	85	0	1
12	32	0	1	0	0	1	1	1	0	1	0	1	1	1	59	249	3.7	54	0	1
13	41	0	1	0	0	1	1	1	0	1	1	1	1	0.9	81	60	3.9	52	0	1
14	30	0	1	1	0	1	1	1	0	1	1	1	1	2.2	57	144	4.9	78	0	1
15	47	0	0	0	1	1	1	1	1	1	1	1	1	0	0	60	0	0	0	1
16	38	0	0	1	0	0	0	1	1	1	1	0	1	2	72	89	2.9	46	0	1
17	66	0	1	1	0	1	1	1	1	1	1	1	1	1.2	102	53	4.3	0	0	1
18	40	0	0	1	0	1	1	1	0	1	1	1	1	0.6	62	166	4	63	0	1
19	38	0	1	1	1	1	1	1	1	1	1	1	1	0.7	53	42	4.1	85	1	1

PERALATAN YANG DIPERLUKAN

1. PC/Komputer
2. Dataset Hepatitis
3. Software Bahasa C

CLUSTERING

PROGRAM

1. Mendeklarasikan input dataset ‘Hepatitis’

```
freopen("taskl.txt","r",stdin);//if You want to notepad , with name input.txt
for( int i = 1 ; i <=155; i++ )
{
    for(j=1; j<=19; j++)
    {
        scanf("%f", &I[i][j]);
    }
    scanf("%f",&kelas[i]);
}
```

2. Melakukan normalisasi dataset ‘Hepatitis’ menggunakan Algoritma Normalisasi Min-Max (1-0)

```
//Data Baru yang sudah diNormalisasi Min-Max(0-1)
//Data Learning
newmax=1;
newmin=0;
for(i=1; i<=155; i++)
{
    I[i][1]=((I[i][1]-7)*(newmax-newmin))/((78-7)+newmin);
    I[i][14]=((I[i][14]-1)*(newmax-newmin))/((8-1)+newmin);
    I[i][15]=((I[i][15]-26)*(newmax-newmin))/((295-26)+newmin);
    I[i][16]=((I[i][16]-14)*(newmax-newmin))/((648-14)+newmin);
    I[i][17]=((I[i][17]-3)*(newmax-newmin))/((5-3)+newmin);
    I[i][18]=((I[i][18]-0)*(newmax-newmin))/((100-0)+newmin);
    //printf("%d = %f %f %f %f %f %f\n",i,Age[i],Bilir[i],Alk[i],Sgot[i],Albumin[i],Protime[i]);
}
```

Rumus:
newdata = (data-min)*(newmax-newmin)/(max-min) + newmin

PROGRAM

3. Merandom nilai Gravity / Centroid

```
for(i=1; i<=cl; i++)
{
    for(j=1; j<=19; j++)
    {
        //c[i][j]=rand()%100/99.1938129;
        //printf("c[%d][%d]= %f\n",i,j,c[i][j]);
    }
}
```

4. Menghitung jarak dari setiap Data dengan setiap Centroid menggunakan perhitungan Rumus Euclidean

```
for(i=1; i<=155; i++)
{
    for(j=1; j<=cl; j++) //CLuster
    {
        jarak[j]=0;
        for(k=1; k<=19; k++) //Penjumlahan jarak
        {
            jarak[j]=jarak[j]+pow(I[i][k]-c[j][k],2);
        }
        jarak[j]=sqrt(jarak[j]);
        //printf("jarak[%d][%d]=%f\n",i,j,jarak[j]);
    }
```

$$d = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

PROGRAM

5. Melakukan pemeringkatan jarak

```
for(j=1; j<=cl; j++)
    {
        rank[j]=1;
        for(b=1; b<=cl; b++)
            {
                if(jarak[j]>jarak[b])
                {
                    rank[j]++;
                }
            }
    }
    //printf("Nilai Ujian\t Rangking\n");

for(j=1; j<=cl; j++)
//printf("%f\t\t %f\n",jarak[j], rank[j]);
```

6. Memilih jarak terdekat

```
for(j=1; j<=cl; j++)
{
    for(k=1; k<=cl; k++)
    {
        if(rank[j]==1 && j==k)
        {
            jumlah[j]=jumlah[j]+1;
            ranking[i][j]=j;
        }
        else if(rank[j]!=1)
        {
            ranking[i][j]=0;
        }
    }
}
```

PROGRAM

7. Mendapatkan centroid yang baru dengan cara mencari mean dari data dengan cluster yang sama.

```
for(i=1; i<=155; i++)
{
    //printf("i= %d   ",i);
    for(j=1; j<=cl; j++)
    {
        for(k=1; k<=cl; k++)
        {
            if(ranking[i][j]==k)
            {
                kelastesting[i]=ranking[i][j];
                //printf("class[%d] = %f\n",i,kelastesting[i]);
                mean[ranking[i][j]][1]=mean[ranking[i][j]][1]+I[i][1];
                mean[ranking[i][j]][2]=mean[ranking[i][j]][2]+I[i][2];
                mean[ranking[i][j]][3]=mean[ranking[i][j]][3]+I[i][3];
                mean[ranking[i][j]][4]=mean[ranking[i][j]][4]+I[i][4];
                mean[ranking[i][j]][5]=mean[ranking[i][j]][5]+I[i][5];
                mean[ranking[i][j]][6]=mean[ranking[i][j]][6]+I[i][6];
                mean[ranking[i][j]][7]=mean[ranking[i][j]][7]+I[i][7];
                mean[ranking[i][j]][8]=mean[ranking[i][j]][8]+I[i][8];
                mean[ranking[i][j]][9]=mean[ranking[i][j]][9]+I[i][9];
                mean[ranking[i][j]][10]=mean[ranking[i][j]][10]+I[i][10];
                mean[ranking[i][j]][11]=mean[ranking[i][j]][11]+I[i][11];
                mean[ranking[i][j]][12]=mean[ranking[i][j]][12]+I[i][12];
                mean[ranking[i][j]][13]=mean[ranking[i][j]][13]+I[i][13];
                mean[ranking[i][j]][14]=mean[ranking[i][j]][14]+I[i][14];
                mean[ranking[i][j]][15]=mean[ranking[i][j]][15]+I[i][15];
                mean[ranking[i][j]][16]=mean[ranking[i][j]][16]+I[i][16];
                mean[ranking[i][j]][17]=mean[ranking[i][j]][17]+I[i][17];
                mean[ranking[i][j]][18]=mean[ranking[i][j]][18]+I[i][18];
                mean[ranking[i][j]][19]=mean[ranking[i][j]][19]+I[i][19];
            }
        }
    }
}
```

```
//printf("\n\n");
for(i=1; i<=cl; i++)
{
    for(k=1; k<=cl; k++)
    {
        if(i==k)
        {
            for(j=1; j<=19; j++)
            {
                if(jumlah[i]==0)
                {
                    c[i][j]=0;
                }
                else if(jumlah[i]!=0)
                {
                    c[i][j]=mean[i][j]/jumlah[i];
                }
                //printf("c[%d][%d] = %f\n",i,j,c[i][j]);
            }
        }
    }
}
```

PROGRAM

8. Melooping proses 4-7 agar didapatkan nilai centroid yang tidak berubah. Yang artinya pengclusteran pada setiap dataset, sudah benar sesuai algoritma Clustering.

```
loop=10;
for(iterasi=1; iterasi<=loop; iterasi++)
{
```

CLUSTER ANALYSIS

RUMUS “VARIANCE”

1. $$v_c^{\,2} = \frac{1}{n_c - 1} \sum_{i=1}^{n_c} \left(d_i - \overline{d}_i \right)^2$$

2. $$v_w = \frac{1}{N - k} \sum_{i=1}^{k} (n_i - 1) \cdot v_i^{\,2}$$

3. $$v_b = \frac{1}{k - 1} \sum_{i=1}^{k} n_i \left(\overline{d}_i - \overline{d} \right)^2$$

4. $$v = \frac{v_w}{v_b}$$

PROGRAM “VARIANCE” (Vc)

9. Menghitung jumlah data pada setiap Cluster

```
//MENGHITUNG PERFORMANCE DENGAN "VARIAN"
//Menghitung nC /Jumlah data pada Cluster C
int nC[200];
nC[1]=0;
nC[2]=0;
for(i=1; i<=155; i++)
{
    for(j=1; j<=cl; j++)
    {
        if(kelastesting[i]==j)
        {
            nC[j]=nC[j]+1;
        }
    }
}
```

10. Menghitung rata-rata

```
//Menghitung dibar atau xi bar atau rata2
float dibar[200];
for(j=1; j<=19; j++)
{
    for(i=1; i<=155; i++)
    {
        for(k=1; k<=cl; k++)
        {
            if(kelastesting[i]==k)
            {
                dibar[k]=dibar[k]+I[i][j];
            }
        }
    }
}
```

```
for(i=1; i<=cl; i++)
{
    if(nC[i]==0)
    {
        dibar[i]=0;
    }
    else if(nC[i]!=0)
    {
        dibar[i]=dibar[i]/nC[i];
    }
    //printf("dibar[%d] = %f\n",i,dibar[i]);
}
```

PROGRAM “VARIANCE” (Vc)

11. Menghitung nilai $\sum(di - \bar{di})^2$ (bagian dari Proses mencari Vc^2)

```
//Menghitung didibar
float didibar[200][25];
for(j=1; j<=19; j++)
{
    //didibar[1][j]=0;
    //didibar[2][j]=0;
    for(i=1; i<=155; i++)
    {
        for(k=1; k<=cl; k++)
        {
            if(kelastesting[i]==k)
            {
                didibar[k][j]=didibar[k][j]+ pow((I[i][j]-dibar[k]),2);
            }
        }
    }
}
```

12. Menghitung nilai Vc

```
//Menghitung Cluster Variance (Vc)
float Vc[200][19];
for(i=1; i<=19; i++)
{
    for(j=1; j<=cl; j++)
    {
        if((nC[j]-1)==0)
        {
            Vc[j][i]=0;
        }
        else if((nC[j]-1)!=0)
        {
            Vc[j][i]=didibar[j][i]/(nC[j]-1);
        }
        //printf("Vc[%d][%d]= %f\n",j,i,Vc[j][i])
    }
}
```

PROGRAM “VARIANCE” (Vw)

13. Menghitung nilai Vw

```
//MENGHITUNG Vw / VARIANCE WITHIN CLUSTER
//Menghitung sigma (nl-1)vl
float nivi[200][20];
for(i=1; i<=19; i++)
{
    for(j=1; j<=cl; j++)
    {
        nivi[j][i]=(nC[j]-1)*Vc[j][i];
        //printf("nivi[%d][%d] = %f\n",j,i,nivi[j][i]);
    }
}
//Menghitung nilai Vw
float Vw[200];
for(i=1; i<=19; i++)
{
    Vw[i]=0;
    for(j=1; j<=cl; j++)
    {
        Vw[i]=Vw[i]+nivi[j][i];
    }
    //printf("Vw[%d] = %f\n",i,Vw[i]);
    Vw[i]= Vw[i]/(155-cl);
    //printf("Vw[%d] = %f\n",i,Vw[i]);
}
```

PROGRAM “VARIANCE” (Vb)

14. Menghitung nilai Vb

```
//MENGHITUNG VB / VARIANCE BETWEEN CLUSTERS
//X bar data semua
float xbardata[200];
for(i=1; i<=19; i++)
{
    xbardata[i]=0;
    for(j=1; j<=155; j++)
    {
        xbardata[i]=xbardata[i]+I[j][i];
    }
    xbardata[i]=xbardata[i]/155;
}
//Mencari nl*(xbar klaster - xbardata)^2
float nxbar[200][20];
for(i=1; i<=19; i++)
{
    for(j=1; j<=cl; j++) //cluster
    {
        nxbar[j][i]=nC[j]*pow((dibar[j]-xbardata[i]),2);
        //printf("nxbar[%d][%d] = %f\n",j,i,nxbar[j][i]);
    }
}
```

```
//Menghitung Vb
float Vb[200];
for(i=1; i<=19; i++)
{
    Vb[i]=0;
    for(j=1; j<=cl; j++) //Cluster
    {
        Vb[i]=Vb[i]+nxbar[j][i];
    }
    Vb[i]=Vb[i]/(cl-1);
    //printf("Vb[%d] = %f\n",i,Vb[i]);
}
```

PROGRAM “VARIANCE”

15. Menghitung nilai Variance

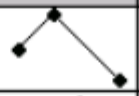
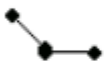
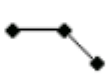
```
//MENGHITUNG V
//Menghitung V, yang belum dirata2 oleh semua attribute
float V[20];
for(i=1; i<=19; i++)
{
    V[i]=Vw[i]/Vb[i];
    //printf("V[%d] = %f\n",i,V[i]);
}
//Menghitung V yang fix, sudah dihitung rata2 perattributenya
float ve;
ve=0;
for(i=1; i<=19; i++)
{
    ve=ve+V[i];
}
//printf("V=%f\n",ve);
```

16. Melooping proses 4-15 untuk mendapatkan nilai Variance saat dibagi menjadi beberapa cluster (Cluster 1-154)

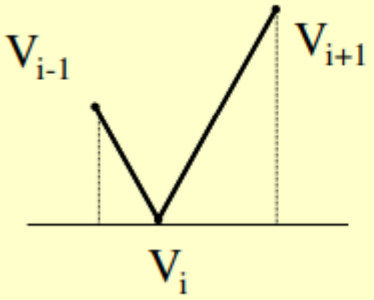
- ```
  for(iterasicl=0; iterasicl<=cl; iterasicl++)
  {
  ```
- ```
  varian[iterasicl]=ve;
  ```

ALGORITMA VALLEY-TRACING UNTUK MENDAPATKAN IDEAL CLUSTER

PERHITUNGAN “VALLEY-TRACING”

Pattern	Possible?	Pattern	Possible?
	√		X
	√		X
	√		X
	X		√
	X		X
	X		X
	X		X
	X		X

1.



$$(V_{i-1} \geq V_i) \cap (V_{i+1} > V_i)$$

2.

$$\partial = (V_{i+1} - V_i) + (V_{i-1} - V_i)$$
$$= (V_{i+1} + V_{i-1}) - (2 \times V_i)$$

PROGRAM

17. Menghitung ideal Cluster yang didapatkan dari pattern of moving Variance.

```
float delta[200];
varian[0]=0;
for(i=1; i<=cl; i++)
{
    if((varian[i-1]>=varian[i])&&varian[i]<varian[i+1])
    {
        delta[i] = (varian[i+1]-varian[i])+(varian[i-1]-varian[i]);
        printf("delta[%d]= %f\n",i,delta[i]);
    }
    else
    {
        delta[i]=0;
        printf("delta[%d]= %f\n",i,delta[i]);
    }
}
```

18. Memilih nilai delta paling besar

```
//RANKING NILAI DELTA=MAX
int peringkat[200];
for(j=1; j<=cl; j++)
        {
            peringkat[j]=1;
            for(b=1; b<=cl; b++)
                {
                    if(delta[j]>delta[b])
                    {
                        peringkat[j]++;
                    }
                }
        }
        printf("Nilai Ujian\t Rangking\n");

    for(j=1; j<=cl; j++)
    printf("%f\t\t %d\n",delta[j], peringkat[j]);
```

PROGRAM

19. Menyimpulkan bahwa ideal cluster adalah nilai delta yang paling besar

```
//Kesimpulan Ideal Cluster
for(i=1; i<=cl; i++)
{
    if(peringkat[i]==cl) //Peringkat dengan delta max
    {
        printf ("IDEAL CLUSTER = %d\n",i);
    }
}
```

20. Menghitung akurasi dari kesimpulan yang dihasilkan dari metode "Valley-Tracing" ini.

```
//Accuracy
float akurasi;
float p,q;
for(i=1; i<=cl; i++)
{
    if(peringkat[i]==cl)
    {
        p=delta[i];
    }
    else if(peringkat[i]==(cl-1))
    {
        q=delta[i];
    }
}
akurasi=p/q;
printf("akurasi = %f\n",akurasi);
```

$$\varphi = \frac{\max(\partial)}{\text{closer value to } \max(\partial)}$$

HASIL

```
IDEAL CLUSTER = 19
akurasi = 7.190645
```

ANALISA

- Praktikum kali ini membahas tentang Clustering dengan menggunakan algoritma K-Means, kemudian untuk Cluster Analysisnya menggunakan perhitungan Variance dan Algoritma untuk mendapatkan Ideal Cluster menggunakan Valley-Tracing.
- Algoritma “Valley-Tracing” adalah algoritma proposed atau algoritma baru, yang digunakan untuk menentukan jumlah Cluster yang ideal. Karena, metode Clustering termasuk data yang tidak mempunyai Class-label/Data Unsupervised. Sehingga kelasnya ditentukan dari perhitungan menggunakan metode. Salah satu metodenya adalah “Valley-Tracing”
- Algoritma “Valley-Tracing” dihitung menggunakan rumus yang sudah dijelaskan dislide sebelumnya, dan yang dipilih adalah nilai delta yang terbesar. Setelah itu, juga dapat memeriksa kinerja kesimpulan jumlah cluster ideal tadi dengan menggunakan rumus Accuracy proposed method, yang mana dalam praktikum ini akurasinya sudah >=2, yang artinya jumlah cluster yang dihasilkan sudah benar.