Comparación de Clasificadores

Ismael Cortés Castillo
Jesús Abelardo Dávila Mauricio

November 30, 2020

Introducción

Para el desarrollo de la practica se tiene como objetivo comparar la eficacia de los clasificadores Bayes, Miníma Distancia y Knn. Cada uno de estos tiene su propio medidor de eficacia y al finalizar arroja un total de cuantas clases fueron clasificadas correctamente.

Desarrollo Dataset: Iris

Dataset con 150 estancias de la forma:

5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa

Como resultado tenemos

```
Eficacia de Bayes=96%
144 de 150


Eficacia de Knn= 100%
150 de 150


Eficacia de Minima Distancia= 92.0%
139 de 150
BUILD SUCCESSFUL (total time: 5 seconds)
```

Dataset: Glass

Dataset con 214 estancias de la forma:

152.101	13.64	4.49	1.1	71.78	0.06	8.75	0	0	1
151.761	13.89	3.6	1.36	72.73	0.48	7.83	0	0	1
151.618	13.53	3.55	1.54	72.99	0.39	7.78	0	0	1
151.766	13.21	3.69	1.29	72.61	0.57	8.22	0	0	1
151.742	13.27	3.62	1.24	73.08	0.55	8.07	0	0	1

Resultados

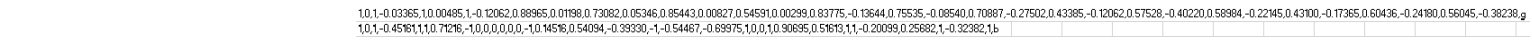
```
Eficacia de Bayes=50%
109 de 214


Eficacia de Knn= 75%
161 de 214


Eficacia de Minima Distancia= 48.0%
103 de 214
BUILD SUCCESSFUL (total time: 5 seconds)
```

Dataset: Inosphere

Dataset con 351 estancias de la forma:



Resultados

```
Eficacia de Bayes=0%
0 de 351


Eficacia de Knn= 87%
307 de 351


Eficacia de Minima Distancia= 72.0%
256 de 351
Exception while removing reference.
BUILD SUCCESSFUL (total time: 2 minutes 5 seconds)
```

Dataset:Vino

Dataset con 178 estancias de la forma:

14.23	1.71	2.43	15.6	127	2.8	3.06	0.28	2.29	5.64	1.04	3.92
13.2	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.4
13.16	2.36	2.67	18.6	101	2.8	3.24	0.3	2.81	5.68	1.03	3.17
14.37	1.95	2.5	16.8	113	3.85	3.49	0.24	2.18	7.8	0.86	3.45

Resultados

```
Eficacia de Bayes=0%
0 de 178
Eficacia de Knn= 100%
178 de 178
Eficacia de Minima Distancia= 53.0%
96 de 178
BUILD SUCCESSFUL (total time: 11 seconds)
```

Dataset: Chicles

Dataset con 49 estancias de la forma:

3,6	15	Bubbaloo Tutti-Frutti
3,6	15	Bubbaloo Tutti-Frutti
3,6	15	Bubbaloo Pik Piña

Resultados

```
Eficacia de Bayes=0%
0 de 49

Eficacia de Knn= 97%
48 de 49

Eficacia de Minima Distancia= 79.0%
39 de 49
BUILD SUCCESSFUL (total time: 14 seconds)
```

Código Clasificador Bayes

```
public class Bayes implements ClasificadorSupervisado {


    ArrayList<Patron> promedio,varianza,desviacion;
    ArrayList<String> nombre;
    ArrayList<Pair<String,Double>>  numPatronClase = new ArrayList<>();

    Patron promedios, varianzas, desviaciones;
    double[] auxiliar,var, des,porcentaje;
    public Bayes() {
        this.promedio = new ArrayList<>();
        this.varianza = new ArrayList<>();
        this.desviacion = new ArrayList<>();
        this.nombre = new ArrayList();
        this.numPatronClase = new ArrayList();

    }
@Override
  public void entrenar(ArrayList<Patron> instancias) {


     for(int x=0; x<instancias.size(); x++){
             this.nombre.add(instancias.get(x).getClase());
     }
    Set<String> hs = new HashSet<String>(this.nombre);
    this.nombre.clear();
    this.nombre.addAll(hs);
  //Calculamos el promedio o media
  calcularVectorPromedio(this.nombre,instancias);
  //Calculamos la varianza
  calcularVectorVarianza(this.nombre,instancias);
  //Calculamos la desviación estandar
  calcularVectorDesviacion(this.nombre,instancias);
  //Calculamos los patrones por clase y su porcentaje
  porcentaje = new double[nombre.size()];
  for(int w=0; w<nombre.size(); w++){

      double contador = 0;
     for(int x=0; x<instancias.size(); x++){
         if(nombre.get(w).equals(instancias.get(x).getClase()))
         {
            contador++;
         }
     }
    porcentaje[w]=(contador/instancias.size());
    numPatronClase.add(new Pair<>(nombre.get(w),contador));
  }
}
```

```
    @Override
    public void clasificar(ArrayList<Patron> instancias) {
        int cont=0,eficacia;
        for(int x =0; x<instancias.size(); x++){
            clasificarPatron(instancias.get(x));
//              System.out.println("Clase "+x+" = "+instancias.get(x).getClase()+" -> Clase Resul
            if(instancias.get(x).getClase().equals(instancias.get(x).getClaseResultante())){

             cont++;
            }
        }
        eficacia=(cont*100)/instancias.size();
        System.out.println("Eficacia de Bayes="+eficacia+"%");
        System.out.println(cont+" de "+instancias.size());


    }

    private void calcularVectorVarianza(ArrayList<String> aux,ArrayList<Patron> instancias){
        int contInsClas;
        for(int w=0; w<aux.size();w++){
         this.var = new double[instancias.get(0).getVectorC().length];
         contInsClas=0;
        for(int x=0; x<instancias.size();x++){
          if(aux.get(w).equals(instancias.get(x).getClase())){
              contInsClas+=1;
            for(int y=0; y<instancias.get(x).getVectorC().length;y++){
                  var[y]=var[y]+Math.pow(instancias.get(x).getVectorC()[y]-this.promedio.get(w).getVectorC()[y], 2);
                 }
            }
         }
        for(int x=0; x<this.var.length; x++){
          this.var[x]=this.var[x]/(contInsClas-1);
        }
        this.varianzas = new Patron(aux.get(w),"",this.var);
        this.varianza.add(this.varianzas);
     }
//      System.out.println("Vectores Varianza");
//      imprimirMatrices(this.varianza);
    }
    private void calcularVectorPromedio(ArrayList<String> aux,ArrayList<Patron> instancias){
        int contInsClas;
     for(int w=0; w<aux.size();w++){
         this.auxiliar = new double[instancias.get(0).getVectorC().length];
         contInsClas=0;
        for(int x=0; x<instancias.size();x++){
          if(aux.get(w).equals(instancias.get(x).getClase())){
              contInsClas+=1;
            for(int y=0; y<instancias.get(x).getVectorC().length;y++){
               auxiliar[y]=auxiliar[y]+instancias.get(x).getVectorC()[y];

                }
            }
         }
        for(int x=0; x<this.auxiliar.length; x++){
           this.auxiliar[x]=this.auxiliar[x]/contInsClas;
        }
        this.promedios = new Patron(aux.get(w),"",this.auxiliar);
        this.promedio.add(this.promedios);
     }
//       System.out.println("Vectores Promedio");
//      imprimirMatrices(this.promedio);
    }
```

```
    private void calcularVectorDesviacion(ArrayList<String> aux,ArrayList<Patron> instancias){
        for(int w=0; w<aux.size();w++){
        this.des = new double[instancias.get(0).getVectorC().length];
        for(int x=0; x<varianza.size();x++){
          if(aux.get(w).equals(varianza.get(x).getClase())){
            for(int y=0; y<instancias.get(x).getVectorC().length;y++){
                  this.des[y]= Math.sqrt(this.varianza.get(x).getVectorC()[y]);
                 }
             }
         }

        this.desviaciones = new Patron(aux.get(w),"",this.des);
        this.desviacion.add(this.desviaciones);
     }
//      System.out.println("Vectores Desviacion");
//      imprimirMatrices(this.desviacion);

    }
    private void clasificarPatron(Patron p){
     ArrayList<Patron> distribucion = new ArrayList<>();
     ArrayList<Pair<String,Double>>posteriori = new ArrayList<>();
     ArrayList<Double> promXClase = new ArrayList<>();
     Double evidencia=0.0;
     for(int x=0; x<desviacion.size();x++){
         double[] norm=new double[desviacion.get(x).getVectorC().length];
         for(int y=0; y<desviacion.get(x).getVectorC().length;y++){
             double raiz=2*Math.PI*varianza.get(x).getVectorC()[y];
             double factor1=1/Math.sqrt(raiz);
             double exp=-1*((Math.pow(p.getVectorC()[y]-promedio.get(x).getVectorC()[y], 2)/(2*varianza.get(x).getVectorC()[y])));
             double factor2=Math.pow(Math.E, exp);
             norm[y]=factor1*factor2; }
         Patron n = new Patron("",desviacion.get(x).getClaseResultante(),norm);
         distribucion.add(n); }
     for(int x=0; x<numPatronClase.size();x++){
         double aux=porcentaje[x];
         for(int y=0; y<desviacion.get(x).getVectorC().length;y++){
             aux*=desviacion.get(x).getVectorC()[y]; }
       evidencia+=aux; }
     for(int x=0; x<numPatronClase.size();x++){
         double aux=porcentaje[x];
         for(int y=0; y<distribucion.get(x).getVectorC().length;y++){
             aux*=distribucion.get(x).getVectorC()[y]; }
       posteriori.add(new Pair<>(numPatronClase.get(x).getKey(),aux/evidencia));  }
     double mayor=0;
     for(int x=0; x<numPatronClase.size();x++){
        if(mayor<posteriori.get(x).getValue()){
            p.setClaseResultante(posteriori.get(x).getKey());
            mayor=posteriori.get(x).getValue();
            }
    private void imprimirMatrices(ArrayList<Patron> instancias){
        for(int x=0; x<instancias.size(); x++){
            for(int y=0; y<instancias.get(x).getVectorC().length; y++){
                System.out.print("[ "+instancias.get(x).getVectorC()[y]+" ]");
            }
            System.out.print(" -> "+instancias.get(x).getClase()+"\n");
        }
    }
```