Universidad Autónoma Metropolitana Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Ingeniería en Computación

Lenguaje de manipulación y minería de datos

Díaz Jiménez Cristina Alicia 206303660

11-P
14 de septiembre de 2011

Oscar Herrera Alcántara 24709

Índice

OBJETIVO(S) GENERAL(ES)

Definir un lenguaje que permita realizar minería de datos, en datos almacenados en un manejador de base de datos.

OBJETIVO(S) PARTICULAR(ES)

Definir una gramática del lenguaje PANZ de manipulación y minería de datos a partir de SQL.

Implementar una terminal interactiva para probar el lenguaje PANZ.

Probar el lenguaje PANZ con 2 algoritmos de clasificación de datos, uno con enfoque supervisado y el otro sin enfoque supervisado.

ANTECEDENTES

Oracle e IBM, empresas líderes en el ramo de las tecnologías de la información, poseedoras de una visión llamada Business Intelligence, ofrecen productos de minería de datos.

Los productos de estas compañías, se venden con la promesa de que son capaces de ayudar a las organizaciones a encontrar información que no es perceptible de forma directa, como por ejemplo patrones de comportamiento, relaciones, asociaciones, etcétera, que permitan tomar mejores decisiones. A través del análisis del pasado, y aplicando algoritmos, se construyen predicciones que permiten mejorar la eficiencia, consiguiendo una mayor rentabilidad de la actividad de negocio.

La apuesta de estas empresas en este campo, da un panorama de la tendencia actual del mercado, en donde tener datos, no es el objetivo, sino, obtener información. Las bases de datos son el primer escalón en ésta búsqueda, sin embargo ya no son suficientes, pues lo que al mundo le concierne es el uso de esa información para la generación de conocimiento.

Algunas de las aplicaciones de la visión de Business Intelligence los productos de minería de datos son:

- Sacar perfiles de los clientes y entender su comportamiento
- Fidelizar a los clientes ofreciendo lo que ellos esperan
- Mejorar los beneficios y márgenes
- Aumentar la eficacia incrementando la competitividad

Oracle cuenta con conectores y extensiones de la cual el más relevante para este proyecto es Oracle Data Miner. Dichos conectores y extensiones son:

Oracle Data Miner— Es una interface gráfica de usuario, la cual permite ayudar a los clientes a extraer información de su Base de Datos Oracle, y descubrir nuevos conocimientos, patrones e información valiosa y oculta.

Oracle Spreadsheet adicional para un Análisis Predecible—Permite a los usuarios de Excel extraer información de su Base de Datos Oracle utilizando características de análisis Predict and Explain simples.

Oracle Data Mining JDeveloper y SQL Developer Extensions—Permite a los desarrolladores de Java y de base de datos integrar fácilmente el análisis avanzado en sus aplicaciones.

ODM Connector para mySAP BW—Permite a los usuarios de SAP Business Warehouse 3.5 extraer datos sin dificultades utilizando Oracle Data Mining dentro del entorno SAP Data Mining.

Oracle Data Miner

La base de datos de Oracle, en la edición Enterprise, cuenta con la funcionalidad para la minería de datos, la cual está integrada y se encuentra sobre el mismo motor que la parte relacional de dicha base de datos.

La funcionalidad completa de Data Mining se logra con la API Java que se incluye en la misma base de datos, permitiendo explotar las funciones disponibles de las aplicaciones.

La característica más importante es, quizá, la integración con la base de datos. Oracle Data Mining simplifica el proceso de extracción de conclusiones basadas en grandes cantidades de datos, logrando esto mediante la eliminación de los movimientos de datos para el proceso de análisis. Todas las operaciones de preparación, creación de modelos y análisis permanecen en la base de datos lo que se transforma en una considerable mejora de la productividad, automatización e integración.

Oracle Data Mining acepta tablas transaccionales y no transaccionales, es decir, desde resúmenes hasta registro únicos. Oracle Data Mining hace todas las transformaciones necesarias automáticamente de forma interna, reduciendo el trabajo de los usuarios o desarrolladores.

Los dos algoritmos que se usan para el análisis de datos en Oracle Data Mining son:

1) Naive Bayes para clasificaciones y predicciones.
2) Reglas de asociación para encontrar patrones.

Naive Bayes

Es una técnica de clasificación y predicción que construye modelos, los cuales predicen la probabilidad de que se llegue a ciertos posibles resultados. Esto se logra al utilizar datos históricos para encontrar asociaciones y relaciones y hacer predicciones.

Los resultados predichos por este algoritmos pueden ser de dos clases, o bien binarios o bien multiclase. En este caso de problemas, cada registro cumplirá o no, el comportamiento modelado. Un ejemplo de este tipo de problemas y resultados, es construir un modelo para saber si un cliente será fiel o cambiará de proveedor.

En los problemas multiclase, se pueden tener muchos resultados posibles, un ejemplo sería construir un modelo para predecir qué clase de servicio prefiere cada cliente.

Reglas de Asociación

Detectan eventos asociados que no se percibe en la bases de datos. El uso más común de este tipo de análisis es encontrar combinaciones populares de productos.

Las reglas de asociación generan un conjunto de pares A-B con una probabilidad n%. Una vez creados los modelos Naive Bayes, los registros deben pasar por un proceso de puntación, en el cual se predicen resultados, y puede hacerse en modo batch (el algoritmo recorre una tabla y va almacenando las predicciones en otra tabla) o bajo demanda (el algoritmo puntúa un solo registro y devuelve la predicción, que puede utilizarse directamente en la aplicación que haya pedido esta puntuación).

DB2 Intelligent Miner

En el caso de IBM, en su división DB2, especializada en bases de datos, cuenta con DB2 Intelligent Miner, el cual integra las capacidades de minería de datos con los sistemas ya existentes de dicha compañía para aumentar el rendimiento del análisis de predicciones sin tener que migrar a plataformas específicas de minería de datos. Se puede usar SQL, Web Services, o Java para acceder a las capacidades mineras de DB2.

Los conocimientos necesarios para la realización de este proyecto incluyen conocimientos de bases de datos, uso del manejador de base de datos MySql, compiladores e inteligencia artificial.

JUSTIFICACIÓN

Actualmente, existen programas que implementan algoritmos de minería de datos, así mismo, ya se cuenta con aplicaciones comerciales como Oracle Data Miner o Data Miner o DB2 Intelligent Miner, que integran SQL con algoritmos de minería de datos como Naive Bayes y Reglas de asociación, siendo la minería de datos un tema recurrente en el mundo actual y en el comercial.

Así mismo, existen muchos algoritmos de minería de datos, la intensión es crear un lenguaje adaptado de SQL para incluir algoritmos existentes y otros novedosos de investigación que no son necesariamente usados en aplicaciones existentes.

Las aplicaciones que implementan las aplicaciones de minería de datos, están disponibles en línea de comando o con interfaces gráficas de usuario en el mejor caso cliente-servidor con acceso a base de datos lo cual dificulta su integración en ambientes tipo web por mencionar un ejemplo.

DESCRIPCIÓN TÉCNICA

El proyecto consiste en la realización de un lenguaje de manipulación de datos a partir de SQL, el cual permitirá la clasificación de datos, de una base de datos.

Se desarrolló la gramática correspondiente a PANZ para que trás el análisis del la gramática del lenguaje y la ejecución de SQL, se hace la conexión con la base de datos, de donde se obtienen los datos que sirven como materia prima del proyecto.

PANZ hace la llamada a la ejecución correspondiente del algoritmo de clasificación para que los resultados se muestren en una interface.

Figura 1. Clasificador de datos

En la *Figura 1* se esquematiza que con el uso de SQLPANZ se hace uso del clasificador y se obtienen los datos clasificados a partir de una base de datos, la cual contiene la información características sobre vinos.

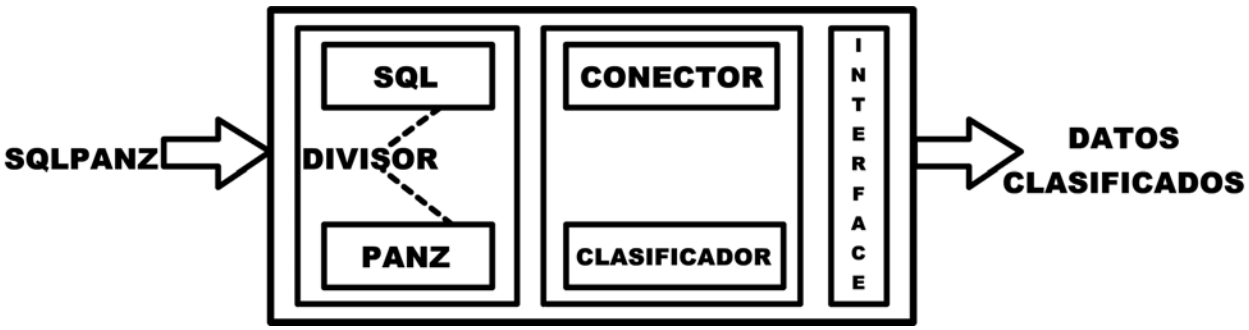


Figura 2. Elementos básicos

SQL y PANZ se separan, para después conectarse a una base de datos, aplicar el algoritmo clasificador correspondiente y mostrar los resultados en una interface que permite la interpretación de los datos arrojados por el clasificador, estos elementos básicos se muestran en la *Figura 2*.

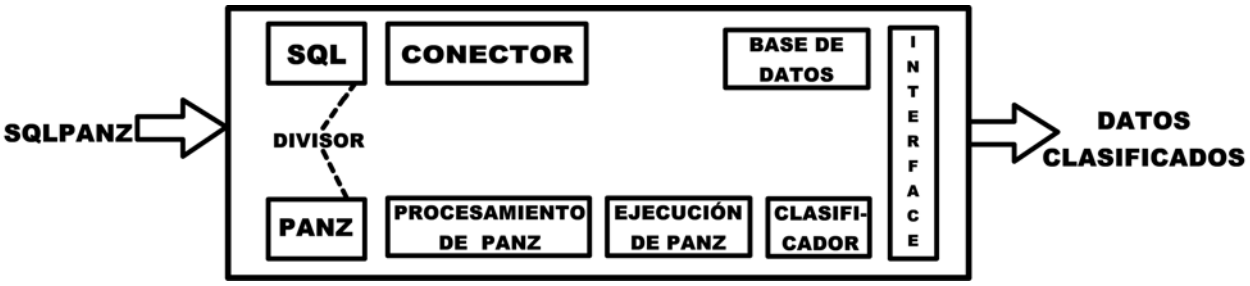


Figura 3. Detalle de los elementos

La ejecución de PANZ, se logra tras un procesamiento que permita verificar que las sentencias introducidas por el usuario son válidas. Se distingue en la *Figura 3* la presencia de la base de datos, la cual debe ser manipulada por un sistema gestos de base de datos.

```
Select Tabla_Campos where condición from Tabla clustering with SOM;
```

Figura 4.1 Sentencia de SQLPANZ usando SOM

```
Select Tabla_Campos where condición from Tabla clustering with MLP;
```

Figura 4.2 Sentencia de SQLPANZ usando MLP

Como se muestra en las figuras, 4.1 y 4.2, esa es la entrada deseada de SQLPANZ, donde condición es establecida por el usuario y es opcional. Se establece por SQL la selección de los campos deseados, y se dice la ubicación de dichos. PANZ, especifica la acción de minería a realizar, es decir la clasificación y el algoritmo que se ha de utilizar.

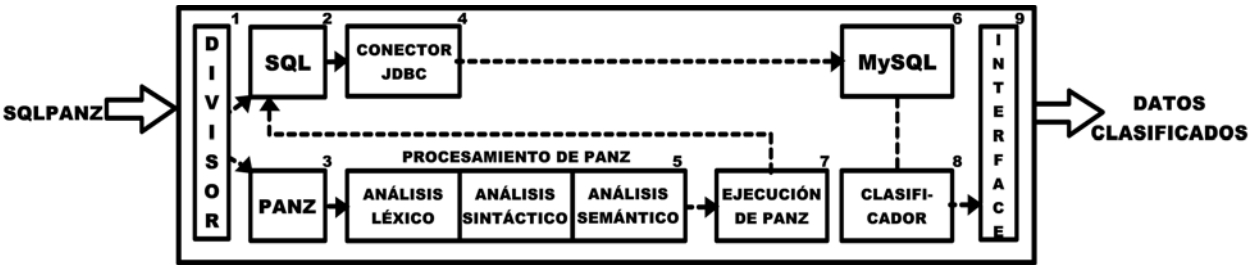


Figura 5.Descripción del proceso para la clasificación de los datos

Ya que PANZ sea ejecutado, se empata con SQL para que se conecte con JDBC y así, ir por los datos al manejador de base de datos MySql, quien proporciona los datos al clasificador *Figura 5*.

Tras la ejecución del clasificador, la interface muestra los resultados obtenidos.

En caso de no resultar correcta el uso de las gramática, se muestran los errores correspondientes donde se indica donde se ubican y de que tipo son, en caso contrario se debe proceder a la conexión de datos con JDBC que actúa sobre MySQL, quien le proporciona los datos al algoritmo clasificador correspondiente.

Los dos algoritmos que se utilizaran para la realización del proyecto son:

- MLP: Multi Layer Perceptron (Perceptrones Multi Capa)
- SOM: Self Organized Maps (Mapas auto organizables)

La implementación de los dos algoritmos se tomó de Weka (*Waikato Environment for Knowledge Analysis, Entorno para Análisis del Conocimiento de la Universidad de Waikato*) el cual es un software que implementa diferentes algoritmos de *machine learning* para minería de datos, desarrollado en Java por *Universidad de Waikato, Nueva Zelanda*. Dicho software es libre y su distribución es bajo licencia GNU-GPL.

Tras la ejecución del algoritmo clasificador correspondiente, se deberán mostrar los resultados en una interface.

ESPECIFICACIONES TÉCNICAS

La base de datos donde se prueba el lenguaje es cargada de un script SQL, donde se crea la base de datos, 2 las tablas(una que incluye sólo las 13 características de los vinos, la segunda es copia de la primera mas la clase a la que pertenecen los vinos), se insertan los valores correspondientes a las caracteristícas de los vinos.

Esta base de datos es tomada del repositorio de machine learning de la UCI (University of California Irvine), estos datos se obtuvieron de hacer un analísis químico a 178 vinos para determinar su origen. Dichos datos ya se encuentran normalizados para comenzar el proceso de minería de datos. Dentro de las características que conforman la base de datos se encuentran:

- Alcohol
- Ácido málico
- Cenizas
- Cantidad de cenizas
- Magnesio
- Fenoles totales
- Flavonoides
- Fenoles no flavenoidales
- Proantocianidinas
- Intensidad del color
- Color

- OD280/OD315 de vinos disueltos
- Prolina
- Más la clase a la que pertenecen según estas 13 características previas

```
drop database if exists minero;
create database minero;
use minero;
drop table if exists tmineroconclases;
drop table if exists tminero;
create table tmineroconclases (
f1 varchar(32),f2 varchar(32),f3 varchar(32),f4 varchar(32),f5 varchar(32),f6 varchar(32),f7 varchar(32),f8
varchar(32),f10 varchar(32),
f11 varchar(32),f12 varchar(32),f13 varchar(32), clase varchar(32)
);
create table tminero (
f1 varchar(32),f2 varchar(32),f3 varchar(32),f4 varchar(32),f5 varchar(32),f6 varchar(32),f7 varchar(32),f8
varchar(32),f10 varchar(32),
f11 varchar(32),f12 varchar(32),f13 varchar(32)
);

Insert into tmineroconclases values
('14.23','1.71','2.43','15.6','127','2.8','3.06','0.28','2.29','5.64','1.04','3.92','1065','1');
Insert into tmineroconclases values
('13.2','1.78','2.14','11.2','100','2.65','2.76','0.26','1.28','4.38','1.05','3.4','1050','1');
Insert into tmineroconclases values
('13.16','2.36','2.67','18.6','101','2.8','3.24','0.3','2.81','5.68','1.03','3.17','1185','1');
```

Una vez cargados los datos en el manejador mediante el script se procede a la interacción con PANZ.

El usuario introduce, en la ventana de PANZ sentencias de leguaje declarativo de bases de datos SQL, especificando que acciones de manipulación desea hacer sobre la base de datos, es decir de donde obtener los datos y bajo que criterios seleccionarlos.

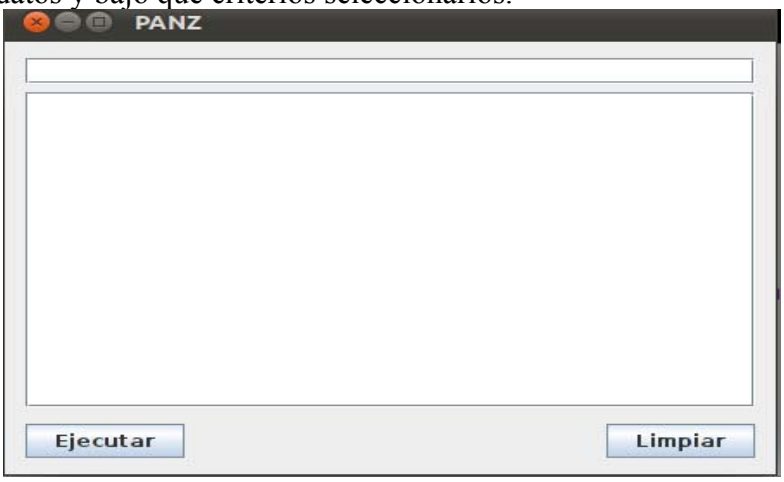


Figura 6. Ventana de PANZ

Después , se concatenan las sentencias de PANZ , especificando el parámetro de entrada correspondiente, que será el algoritmo que se desee utilizar para obtener la clasificación de los datos y se pulsa “Ejecutar”.

Una vez introducidas las sentencias, SQL y PANZ se separan, pasando SQL por el procesamiento ya definido del lenguaje. Así mismo PANZ deberá ser analizado por un programa en Java de la siguiente manera:

- Léxicamente
- Sintácticamente
- Semánticamente:

De haber resultado un error en PANZ, se indica en la pantalla qué error es y dónde ubicarlo, en caso contrario se procederá al proceso de acceder a JDBC mediante SQL.

En JDBC, mediante interfaces de Java, se lleva a cabo las operaciones sobre la base de datos, teniendo como objetivo establecer la conexión con dicha base, enviar las sentencias que se encuentran en SQL para así poder procesar los resultados que han sido solicitados.

JDBC hace la solicitud al manejador de la base de datos, que es MySQL, éste gestiona los datos y ofrece los resultados que se solicitaron, es decir los datos que han sido seleccionados de la base de datos, pues cumplen con las condiciones que se hicieron desde las sentencias de SQL.

Estos resultados son la entrada para los algoritmos de clasificación, que son obtenidos en un archivo ARFF (formato de Weka), sobre los cuales, el algoritmo se ejecuta para obtener una clasificación de dichos datos.

```
% Nombre de la relación que en este caso es el nombre de la tabla
%
@relation 'tmineroconclases'
% Descripción de los campos correspondientes a cada atributo%
@attribute f1 real
@attribute f2 real
@attribute f3 real
@attribute f4 real
@attribute f5 real
@attribute f6 real
@attribute f7 real
@attribute f8 real
@attribute f9 real
@attribute f10 real
@attribute f11 real
@attribute f12 real
@attribute f13 real
@ATTRIBUTE class {uno, dos, tres}
%Conjunto de datos%
@data
13.2,1.78,2.14,11.2,100,2.65,2.76,0.26,1.28,4.38,1.05,3.4,1050,uno

13.16,2.36,2.67,18.6,101,2.8,3.24,0.3,2.81,5.68,1.03,3.17,1185,uno

14.37,1.95,2.5,16.8,113,3.85,3.49,0.24,2.18,7.8,0.86,3.45,1480,uno

13.24,2.59,2.87,21,118,2.8,2.69,0.39,1.82,4.32,1.04,2.93,735,uno
```

Figura 7. Archivo ARFF

Para la ejecución del algoritmo MLP se tiene los siguientes parámetros, tomados por defecto.

-L : Taza de aprendizaje. (defecto 0.3)
-S : Momentum (defecto 0.2) -N : Número de defecto de épocas usadas para entrenamiento. (defecto 500) -V: Porcentaje del set de validación usado para el entrenamiento. (defecto 0 (no usa set de validación y en su lugar ocupa el número de épocas num of epochs is used)
-S : Semilla para generar número aleatorios. (defecto 0)

En el caso del algoritmo SOM sólo se cuenta con:

-H : Determina la altura del mapa auto organizable (3) -W : Determina el ancho del mapa auto organizable(1)

Una vez terminada la ejecución del algoritmo pertinente, la interface presenta una tabla con 14 columnas, donde 13 de ellas son las características de cada uno de los vinos, siendo la última columna la que especifique a que clase pertenece cada vino.

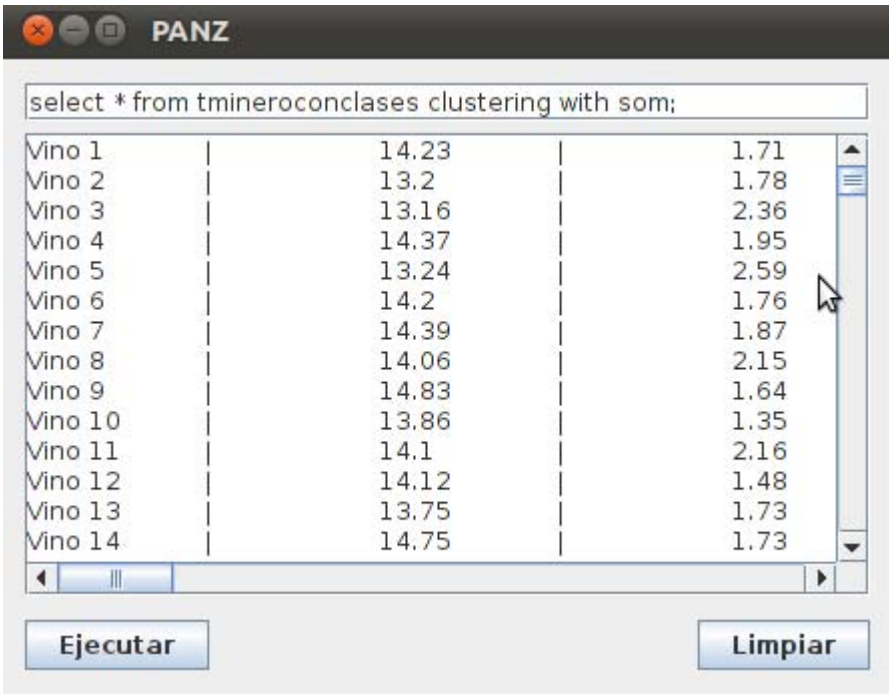


Figura 8. Salida de PANZ

Código Fuente

Clase AccesoDatos

Gestiona la conexión a la base de datos, así como la ejecución de consultas. Dichas consultas son impresas en un archivo ARFF en el caso de PANZ o TXT en caso de ser exclusivamente SQL.

```
/*Codigo que genera el archivo arff */

import java.sql.*;
import java.io.*;



/** @author cristina */
/* Clase que gestiona el acceso a la BD así como las consultas, imprimiendo los archivos correspondientes.
*/
public class AccesoDatos {
   Connection conexion = null;
   public String nombre_tabla;
   private String tipo_columna[];
   private String nombre_columna[];
   private int clase=-1;
   private    Statement st= null;
   private    ResultSet resultados=null;
   public AccesoDatos()
   {   clase=-1;
      st= null;
      resultados=null;
   }



/*        Realiza la conexión
*/
   public Connection getConexion()
   {   try
      {   Class.forName("com.mysql.jdbc.Driver");
//Crear el objeto de conexion a la base de datos
            //this.conexion = DriverManager.getConnection("jdbc:mysql://localhost/minero", "root" , "%mysqlrootpassword");
```

```
            this.conexion = DriverManager.getConnection("jdbc:mysql://localhost/minero", "root" ,
"alfa00");
        }
        catch(ClassNotFoundException e){ System.out.println(e); }
        catch(SQLException e){ System.out.println(e);  }
        catch(Exception e){ System.out.println(e);  }
        return this.conexion;
    }

/*        Ejecuta las consultas de PANZ generando el ARFF correspondiente
*/
    public ResultSet EscrituraConsulta (String consulta)throws SQLException
    {   Connection cn=getConexion();
        st= cn.createStatement();
        resultados = st.executeQuery(consulta);
        ResultSetMetaData metadatos = resultados.getMetaData();
        int cuenta= metadatos.getColumnCount();
        this.nombre_tabla = metadatos.getTableName(1);
        tipo_columna= new String [cuenta];
        nombre_columna= new String [cuenta];
        if (resultados.next())
        {   FileWriter fichero = null;
            PrintWriter pw = null;
            try
            {   fichero = new FileWriter(nombre_tabla+".arff");
                pw = new PrintWriter(fichero);
                pw.println("%");
                pw.println("%");
                pw.println("@relation " +  """+metadatos.getTableName(1)+""");
                for(int i =0; i< metadatos.getColumnCount(); i++)
                {
                    tipo_columna[i] = metadatos.getColumnTypeName(i+1);
                    nombre_columna[i] = metadatos.getColumnName(i+1);
                    if ( tipo_columna[i].equalsIgnoreCase("varchar"))
                    {
                        String dummy=resultados.getString(i+1);
                        if(nombre_columna[i].equalsIgnoreCase("clase"))
                        {   pw.println("@ATTRIBUTE clase {1, 2, 3} ");
                            clase = i;
                        }
                        else
                        {
                             if (dummy.charAt(1) == '-' || dummy.charAt(1) == '.' || (dummy.codePointAt(1)
>= 48 && dummy.codePointAt(1) <= 57))
                            {   pw.println("@attribute " + metadatos.getColumnName(i+1) + " real"); }
                            else
```

```
                {   pw.println("@attribute " + metadatos.getColumnName(i+1) + " string");
                }
            }
        }
        else
                                        {       if  (tipo_columna[i].equalsIgnoreCase("char")  ||
tipo_columna[i].equalsIgnoreCase("text")
                                                ||  tipo_columna[i].equalsIgnoreCase("blob")  ||
tipo_columna[i].equalsIgnoreCase("medium blob") || tipo_columna[i].equalsIgnoreCase("medium
text")
                                            ||  tipo_columna[i].equalsIgnoreCase("long  blob")  ||
tipo_columna[i].equalsIgnoreCase("long text") || tipo_columna[i].equalsIgnoreCase("set")
            || tipo_columna[i].equalsIgnoreCase("set"))
              {       pw.println("@attribute " + metadatos.getColumnName(i+1) + " string"); }
              else
              {
                                                  if  (tipo_columna[i].equalsIgnoreCase("real")  ||
tipo_columna[i].equalsIgnoreCase("TinyInt")
                                                ||  tipo_columna[i].equalsIgnoreCase("SMALLINT")  ||
tipo_columna[i].equalsIgnoreCase("MEDIUMINT") || tipo_columna[i].equalsIgnoreCase("INT")
                                                ||  tipo_columna[i].equalsIgnoreCase("INTEGER")  ||
tipo_columna[i].equalsIgnoreCase("BIGINT") || tipo_columna[i].equalsIgnoreCase("FLOAT")
                                                ||  tipo_columna[i].equalsIgnoreCase("INTEGER")  ||
tipo_columna[i].equalsIgnoreCase("BIGINT") || tipo_columna[i].equalsIgnoreCase("FLOAT")
                                                  ||  tipo_columna[i].equalsIgnoreCase("float")  ||
tipo_columna[i].equalsIgnoreCase("float()") || tipo_columna[i].equalsIgnoreCase("double")
                                            ||  tipo_columna[i].equalsIgnoreCase("double  precision")  ||
tipo_columna[i].equalsIgnoreCase("decimal") || tipo_columna[i].equalsIgnoreCase("numeric"))
                  {   pw.println("@attribute " + metadatos.getColumnName(i+1) + " real");}

                else
                  {   pw.println("@attribute" + metadatos.getColumnName(i+1) + " date");}
              }
          }
      }

      pw.println("@data");
      String contenido_clase=null;
      System.out.println(resultados.getObject(1).toString());
      resultados.beforeFirst();
      while (resultados.next())
      {   for(int k = 0; k<cuenta; k++)
          {   if(k == clase)
              { contenido_clase=(resultados.getObject(k+1).toString());
                if(contenido_clase.equalsIgnoreCase("1"))
                { pw.print("1");
```

```
                    }
                    else
                    {     if(contenido_clase.equalsIgnoreCase("2"))
                          { pw.print("2");
                          }
                          else
                          {   pw.print("3");
                          }
                    }
                   }
                   else
                   {       pw.print(resultados.getString(k + 1));
                   }
                  if ( k <cuenta-1)
                   {pw.print(",");}
                }
                //pw.println(",uno");
                pw.println();
             }


          }
          catch (Exception e)
          { System.out.println("Error al generar " + nombre_tabla+".arff");}
          finally
          {  try
             {  if (null != fichero)
                fichero.close();
             }
          catch (Exception e2) { }
          }

        }
        return resultados;

    }
/*
        Ejecuta las consultas exclusivamente cuando se ingresa condigo SQL puro, genera un archivo de salida.
*/

    public ResultSet EscrituraConsultaSQL (String consulta)throws SQLException
    {   Connection cn=getConexion();
        st= cn.createStatement();
        resultados = st.executeQuery(consulta);
        ResultSetMetaData metadatos = resultados.getMetaData();
```

```
int cuenta= metadatos.getColumnCount();
this.nombre_tabla = metadatos.getTableName(1);
tipo_columna= new String [cuenta];
nombre_columna= new String [cuenta];

if (resultados.next())
{   FileWriter fichero = null;
   PrintWriter pw = null;
   try
   {   fichero = new FileWriter("resultados.txt");
      pw = new PrintWriter(fichero);
      for(int i =0; i< metadatos.getColumnCount(); i++)
      {  pw.print( metadatos.getColumnName(i+1)+ '\t'+ " " + "\t");

      }
      pw.println();
      resultados.beforeFirst();
      while (resultados.next())
      {   for(int k = 0; k<cuenta; k++)
         {   pw.print(resultados.getObject(k+1).toString()+ "\t"+"|"+ "\t");


         }
         pw.println();

      }


  }
  catch (Exception e)
  { System.out.println("Error al generar " + nombre_tabla+".arff");}
  finally
  {  try
    {  if (null != fichero)
       fichero.close();
    }
  catch (Exception e2) { }
  }

}
return resultados;

}


/* Cierra la conexión a la BD
```

```
*/
      public void cerrarconexion()throws SQLException
   {   this.conexion.close();
   }
}
```

Clase Analizador

Analiza las sentencias introducidas por el usuario , distinguiendo entre SQL y PANZ. Si es una sentencia de PANZ se encargará de checar la sintáxis, parsea los parámetros y averigua si se ha de ejecutar SOM o MLP.

```
import java.util.regex.*;
import java.util.StringTokenizer;
/* @author cristina */

/* Clase que analiza la sintáxis de la instrucción dada, ya sea
 * de SQL o de PANZ.

 */
public class Analizador
{   public String cadena, str2 = "", algoritmo, str;
   StringTokenizer tokens;
   boolean accion = false;
/*  Función que verifica la sintáxis de una instrucción de PANZ.
 */
   public boolean Analiza(String entrada)
   { cadena = entrada;
     String s= "som", mi="mlp" ;
     /* Hace uso de expresiones regulares mediante la clase
      * Pattern y Matcher
      */
     String patron= ".+\\s+classify\\s+with\\s+(som|mlp)\\s*;";
     cadena = cadena.toLowerCase();
     Pattern p=Pattern.compile(patron);
     Matcher m=p.matcher(cadena);
     if (!m.matches())
     {   return false;}
     return true;
   }
   /* Función que verifica el algoritmo que
    * PANZ ha de implementar.
    */
   public void Tokens(String entrada)
   {   tokens=new StringTokenizer(entrada);
```

```
        while(tokens.hasMoreTokens())
        {  str= tokens.nextToken();
           if ( str.equalsIgnoreCase("clustering"))
           {    break;          }
           str2 = str2 + " " +str;
         }
        tokens.nextToken();
        algoritmo =(String) tokens.nextToken();
        str2 = str2 + ";";

    }

    /* Función que identifica entre la sentencia de
     * SQL o de PANZ.
     */
    public boolean Identifica(String entrada)
    {   tokens=new StringTokenizer(entrada);
        while(tokens.hasMoreTokens())
        {  str= tokens.nextToken();
           if ( str.equalsIgnoreCase("som;") || str.equalsIgnoreCase("mlp;"))
           {    accion = true;          }

         }
        return accion;
    }

}
```

Clase EntradaWeka

Carga el archivo tipo ARFF a un dataset de Weka

```
import weka.core.Instances;
import weka.core.converters.ArffLoader;
import java.io.*;

/**
 *
 * @author cristina
 */
public class EntradaWeka
{   Instances dataset;
    // Método para cargar los datos para WEKA
    public void CargaArff(String nombre_tabla) throws FileNotFoundException
    {       ArffLoader cargadorARRF = new ArffLoader();
            System.out.println("Cargando DATASET ...");
```

```
        FileInputStream carga;
        //carga = new FileInputStream(nombre_tabla+".arff");
        carga = new FileInputStream(nombre_tabla+".arff");
        try
        {
           cargadorARRF.setSource(carga);
           this.dataset= cargadorARRF.getDataSet();
           dataset = new Instances(
                 new BufferedReader(new FileReader(nombre_tabla+".arff")));
                System.out.println("FFFFFFFFFFFFFFFf "+nombre_tabla+".arff");
        }
        catch (IOException ex)
        { System.out.println("Error al cargar DATASET"); }


   }


}
```

Clase CSOM

Implementa el algoritmo SOM localizado en el archivo weka.classalgos (Plugin de Weka)

```
/*Implementación del algoritmo SOM */

import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;
import java.util.StringTokenizer;
import weka.classifiers.neural.lvq.model.CodebookVector;
import weka.core.Instances;
import weka.core.Utils;
import weka.classifiers.Evaluation;
import weka.classifiers.neural.lvq.Som;
/**
 *
 * @author cristina
 */

/*      Clase que instancia la clase Som (API de WEka)
        Ejecuta el algoritmo y arroja los resultados en un archivo de texto.
*/
public class CSOM {
  Som som = new Som();
```

```
  private FileWriter fichero = null;
  private PrintWriter pw = null;
  private Evaluation eval;

/*        Constructor que iniciaiza los parametros para la escritura del archivo.
*/
  public CSOM() throws IOException
  {   fichero = new FileWriter("resultados.txt");
      pw = new PrintWriter(fichero);
  }

    Instances entrenamiento ;

/*        Función que realiza el entrenamiento del algoritmo.
*/
    public void entrenar(Instances dataset) throws Exception
    {
        entrenamiento = new Instances(dataset);
        entrenamiento.setClassIndex(entrenamiento.numAttributes() -1);
        som.setOptions(Utils.splitOptions("-H 3 -W 1"));
        som.buildClassifier(entrenamiento);
        som.setSupervised(false);
        eval = new Evaluation(entrenamiento);
        eval.evaluateModel(som, entrenamiento);// Evaluación del entrenamiento

    }
/*        Función que etiqueta, previo entrenamiento y arroja los resultados a un archivo d texto;
*/

    public void resultados() throws Exception{

        System.out.println("\n"+som.prepareCodebookVectorReport());
        AccesoDatos ad = new AccesoDatos();
        Connection cn=ad.getConexion();
        Statement st;
        st= cn.createStatement();
        ResultSet resultados = st.executeQuery("Select f1, f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12,f13 from
tminero");
        ResultSetMetaData metadatos = resultados.getMetaData();
        int cuenta= metadatos.getColumnCount();
        String vb = som.prepareCodebookVectorReport();
        String nt;
        double[][] miv = new double[som.getMapHeight()*som.getMapWidth()][cuenta];
                StringTokenizer stok = new StringTokenizer(vb, ",\n");
            nt =stok.nextToken();
            int feature=0;
```

```
    int centroide=0;
    while(stok.hasMoreTokens()){ //feaures
       nt = stok.nextToken();
       double dt1 = Double.parseDouble(nt);
       miv[centroide][feature]=dt1;
       feature++;
       if(feature==cuenta){
          nt = stok.nextToken();
          nt = stok.nextToken();
          System.out.println("");
          feature=0;
          centroide++;
       }
    }
    int vino=0;

try
{    while(resultados.next() )
     {
       // pw.println("comparando vino "+vino);
        double[] error = new double[som.getMapHeight()*som.getMapWidth()];
        for(int e=0; e<som.getMapHeight()*som.getMapWidth(); e++)
        {
           error[e]=0;
           for(int f=0; f<cuenta; f++)
           {
              double tmp = Double.parseDouble(resultados.getObject(f+1).toString());
              error[e] += Math.sqrt(Math.pow(miv[e][f]-tmp,2));
           }
        }
        int clase=0;
        double mineuc=error[0];
        for(int e=0; e<som.getMapHeight()*som.getMapWidth(); e++)
        {
              if(error[e]<mineuc){
                 clase=e;
              }
        }

        pw.print("Vino "+ (vino+1) + "\t" +  "|"+ "\t");
        for(int k = 0; k<cuenta; k++)
        {   pw.print(resultados.getObject(k + 1).toString()+"\t"+"|"+ "\t");

        }
        pw.println(" es de la clase  " + (clase+1));
        vino++;
```

```
            }
            pw.println(vino+" vinos clasificados");
        } catch (Exception e)
        { System.out.println("Error al generar archivo de resultados");}
        finally
        {  try
           {  if (null != fichero)
              fichero.close();
           }
        catch (Exception e2) { }
        }


    }

}
```

Clase MLP

Implementa el algoritmo MLP ubicado en el archivo weka.jar

```
/* Implementación del algoritmo MLP*/

import java.sql.*;
import java.io.*;
import java.util.*;
import weka.classifiers.Classifier;
import weka.core.Instances;
import weka.core.Utils;
import weka.classifiers.Evaluation;
import weka.classifiers.functions.MultilayerPerceptron;

/**
 *
 * @author cristina
 */

/*Clase que instancia la clase MultLayerPerceptron (API) weka
  Ejectua el algoritmo y escribe los resultados en un archivo de texto
*/
public class MLP
{   //MultilayerPerceptron mlp = new MultilayerPerceptron();
Classifier mlp;
    private FileWriter fichero = null;
    private PrintWriter pw = null;
    private Evaluation eval;
    public Instances dataset;
```

```
   public MLP () throws IOException
   {   fichero = new FileWriter("resultados.txt");
       pw = new PrintWriter(fichero);
   }
   Instances entrenamiento ;
   Instances unlabeled;
/* Función que realiza el entrenamiento del la red neuronal.
*/
   public void entrenar(Instances dataset) throws Exception
   {
       this.dataset=dataset;
       entrenamiento = new Instances(dataset);
       entrenamiento.setClassIndex(entrenamiento.numAttributes()-1 );
       unlabeled = new Instances(entrenamiento);

       mlp = new MultilayerPerceptron();
       mlp.buildClassifier(entrenamiento);
       eval = new Evaluation(entrenamiento);
       eval.evaluateModel(mlp, entrenamiento);// Evaluación del entrenamiento

/* Función que lleva a cabo la clasificación y arroja los resultados a un archivo de texto*/
   }
   public void resultados() throws SQLException{


       AccesoDatos ad = new AccesoDatos();
       Connection cn=ad.getConexion();
       Statement st;
       st= cn.createStatement();
       ResultSet resultados = st.executeQuery("Select f1, f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12,f13 from
tminero");
       ResultSetMetaData metadatos = resultados.getMetaData();
       int cuenta= metadatos.getColumnCount();
       int vino=0;
       try
       {      resultados.beforeFirst();
              while(resultados.next() )
              {
                 double[] error=null ; //ops
                 double clsLabel = mlp.classifyInstance(unlabeled.instance(vino));
                 int clase = (int) Math.floor(clsLabel)+1 ;

                 pw.print("Vino "+ (vino+1) + "\t" +  "|"+ "\t");
                 for(int k = 0; k<cuenta; k++)
                 {   pw.print(resultados.getObject(k + 1).toString()+"\t"+"|"+ "\t");
```

```
                }
                pw.println(" es de la clase  " + clase);

                vino++;
              }
              pw.println(vino+" vinos clasificados");
        }catch (Exception e)
         { System.out.println("Error al generar archivo de resultados" +e);}
         finally
         {  try
           {  if (null != fichero)
             fichero.close();
           }
         catch (Exception e2) { }
         }

    }

}
```

Clase principal

Crea la instancia de un Jframe, con el que el usuario interactúa.

```
import java.io.*;
import java.sql.*;
import java.util.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.regex.*;

/**
 *
 * @author cristina
 */
/*  Clase principal, presenta un JFrame como interfaz para interactuar
 * con el usuario
 */
public class PJFrame extends javax.swing.JFrame {

    /** Inicalización de los valores */
    public PJFrame() {
        super ("PANZ");
        initComponents();
        jTextArea1.setEditable(false);
```

```
    }

    /** Este método es generado por el editor de formas, se llama desde el
     * constructor y no debe ser modificado.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jTextField1 = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jTextArea1.setColumns(20);
        jTextArea1.setRows(5);
        jScrollPane1.setViewportView(jTextArea1);

        jButton1.setText("Ejectuar");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Limpiar");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILIN
G)
                        .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 376, Short.MAX_VALUE)
```

```
                        .addComponent(jTextField1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 376, Short.MAX_VALUE)
                                        .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
                    .addComponent(jButton1)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 244,
Short.MAX_VALUE)
                    .addComponent(jButton2)))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(12, 12, 12)
                        .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 231,
Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                    .addComponent(jButton1)
                    .addComponent(jButton2))
                .addContainerGap())
        );

        pack();
    }// </editor-fold>
/* Método que reacciona cuando el usuario teclea una sentencia y pulsa el botón
 * Ejecutar
 */
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      if (!jTextField1.getText().equalsIgnoreCase("")){
      String patron = "\\s*";
      Pattern p=Pattern.compile(patron);
      Matcher m=p.matcher(jTextField1.getText());

          Boolean corresponde;
          Statement st= null;
          ResultSetMetaData metadatos;
          ResultSet tabla;
          Analizador analiza = new Analizador();//analiza la sentencia de Panx
          // cadena de dentrada
          String entrada = null, algoritmo= null;// cadena de dentrada
          jTextField1.getText();
```

```
        AccesoDatos acceso = new AccesoDatos();
        Connection conexion = acceso.getConexion();
        entrada = jTextField1.getText();
        corresponde = analiza.Analiza(entrada);
        if (analiza.Identifica(entrada) == false)
        {   jTextArea1.setText("");
           //hacer la consulta simple de sql
             try
             {   st= acceso.conexion.createStatement();
               tabla = acceso.EscrituraConsultaSQL(entrada);
               metadatos = tabla.getMetaData();
             try {
                Escritura();
             } catch (FileNotFoundException ex) {
                Logger.getLogger(PJFrame.class.getName()).log(Level.SEVERE, null, ex);
             } catch (IOException ex) {
                Logger.getLogger(PJFrame.class.getName()).log(Level.SEVERE, null, ex);
             }
             }
                                                    catch(SQLException   e){   System.out.println(e);
jTextArea1.append("com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException:"); }

           }
           else
           {    System.out.println("--------------------");
              corresponde = analiza.Analiza(entrada);
             // Si no se cumple con la gramática
             if(!corresponde)
              {   jTextArea1.append("Error en la sentencia de PANZ ");}
             else
              {  // division de la cadena
                 analiza.Tokens(entrada);
                 algoritmo = analiza.algoritmo;
                 // ejecución de la consulta
                 try
                 {   st= acceso.conexion.createStatement();
                    tabla = acceso.EscrituraConsulta(analiza.str2);
                    //tabla = acceso.EscrituraConsultaPANZ(analiza.str2, analiza.);//algoritmo;
                    metadatos = tabla.getMetaData();
                 }
                                                    catch(SQLException   e){   System.out.println(e);
jTextArea1.append("com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException:"); }
                 catch(Exception e){ jTextArea1.append("¡Ooops algo falló!"); }
              }
              //Cargar datos para weka
             EntradaWeka dataw = new EntradaWeka();
```

```
            try
            {  dataw.CargaArff(acceso.nombre_tabla);}
                                                        catch (FileNotFoundException ex)
{   Logger.getLogger(PJFrame.class.getName()).log(Level.SEVERE, null, ex);}
            if(analiza.algoritmo.equals("som;"))
            {   jTextArea1.setText("clasificando...");

               try
               {
               CSOM csom = new CSOM();

               try {
                  csom.entrenar(dataw.dataset);
                  csom.resultados();
                  Escritura();
               } catch (Exception ex) {
                  Logger.getLogger(PJFrame.class.getName()).log(Level.SEVERE, null, ex);
               }
               }

catch (IOException ex) { Logger.getLogger(       PJFrame.class.getName()).log(Level.SEVERE,
null, ex);}
               // jTextArea1.setText("");
              }
               else
               {                  try {                  System.out.println("MLP");MLP mlp = new
MLP();
               try {
                  mlp.entrenar(dataw.dataset);
                  mlp.resultados();
                  Escritura();

               } catch (Exception ex) {
                  Logger.getLogger(PJFrame.class.getName()).log(Level.SEVERE, null, ex);
               }
}
                                                                  catch (IOException ex)
{ Logger.getLogger(     PJFrame.class.getName()).log(Level.SEVERE, null, ex);}
               }

         }
      }
   }
/* Método que limpia los campos de texto, se activa al pulsar Limpiar
 */
   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        jTextArea1.setText("");
        jTextField1.setText("");
    }
// Método que escribe sobre el área de texto del JFrame
    public void Escritura () throws FileNotFoundException, IOException
    {   FileReader fr = new FileReader ("resultados.txt");
        BufferedReader br = new BufferedReader(fr);
        String linea = br.readLine();
       //System.out.println(linea);
       while (linea != null)
       {
          jTextArea1.append(linea+"\n");
         // System.out.println(linea);
          linea = br.readLine();
       }
    }
    /**
    * Invocación de la clase JFrame
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new PJFrame().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea jTextArea1;
    private javax.swing.JTextField jTextField1;
    // End of variables declaration

}
```

CONCLUSIÓNES

El trabajo, realizado con software libre (MySql, Weka) presenta un ejemplo pequeño de lo que son los módulos de minería de datos en los manejadores de bases de datos comerciales. Muestra, así también la posiblidad de hacer un lenguaje, que no sea simplemente para manipulación y descripción de datos, si no, ir más allá y generar conocimiento sobre los datos almacenados en una base de datos ordinarias.

Cabe destacar que para que los datos puedas ser procesados, es requerida una limpieza de datos, es decir homogenizar los contendios, en cuanto campos, valores, tipos de dato , contenido y preferentemente una normalización. Dicho proceso provino del conjunto de datos de vinos.

La implementación de los algoritmos corrió por cuenta de Weka, el cual es una plataforma de software que contiene gran cantidad de algoritmos de inteligencia artificial, que pueden ser anexados a este proyecto, mediante patrones de software. Esto es permisible gracias a stu diseño, orientado a objetos.

BIBLIOGRAFÍA

[1] Minería de Datos, “Oracle Data Miner”. Febrero del 2010. [En línea] Disponible en http://www-01.ibm.com/software/data/iminer/

[2] Minería de Datos en DB2, “DB2 Intelligent Miner”. Febrero del 2010. [En línea] Disponible en http://www-01.ibm.com/software/data/iminer/

[3] Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository . Febrero de 2010 [En línea] Disponible en http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA: University of California, School of Information and Computer Science.

[4] O. Herrera. Redes Neuronales y Algoritmos Genéticos. México D.F.: Sometido en Febrero de 2010 al Consejo Editorial de la División CBI de la UAM Azcapotzalco para su posible publicación.

[5] S. Haykin. Neural Networks, a comprehensive foundation. McMaster University, Ontario Canada: Prentice Hall, 1999.

[6] API Weka, “Weka”. Septiembre del 2011. [En línea] Disponible en http://www.cs.waikato.ac.nz/ml/weka/

[7] API Weka Plugin, “Weka Plugin”. Septiembre del 2011. [En línea] Disponible en http://wekaclassalgos.sourceforge.net/api/index.html

[8] Dataset vinos, “UCI machine learning repository”. Febrero del 2010. [En línea] Disponible en http://archive.ics.uci.edu/Wine

[9]C. Díaz, O. Herrera. " Lenguaje de Definición, Manipulación y Minería de daatos . Septiembre 2011.

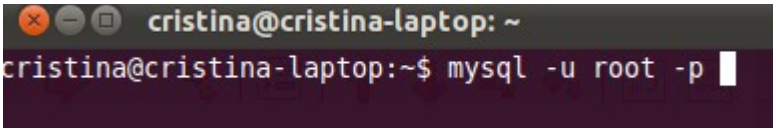
Guía Rápida PANZ

Table of Contents

Instalación.

1. Cargar base de datos.
   Abra una terminal e inicie MySql: mysql -u root -p

   Teclée la contraseña

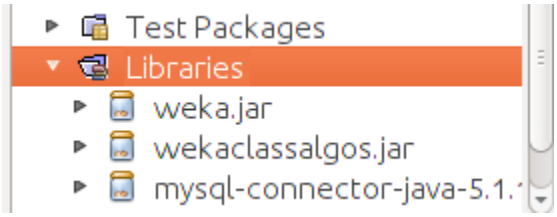


2. Una vez dentro teclee : source “ruta_ubicacion_script”;

```
cristina@cristina-laptop:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 169
Server version: 5.1.54-1ubuntu4 (Ubuntu)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement

mysql> source /home/cristina/Documentos/wineInsert.SQl;
```

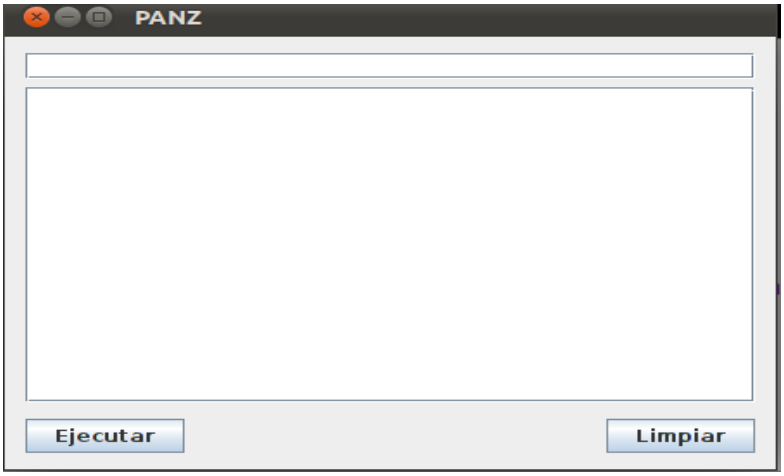
3. Salga de Mysql y de la terminal.
4. Abra Netbeans y abra el proyecto llamado Pframe .Se recomienda el uso de Netbeans en lugar de consola pues es más amigable.

5. Vaya a la sección de librerias dentro del proyecto y busqué 3 archivos jar
   1. weka.jar
   2. wekaalgos.jar
   3. mysql.jar



6. En caso de no existir, agréguelos al proyecto. Los archivos se encuetran en la misma carpeta del proyecto.
7. Corra el proyecto.

Ejecución



- Se desplegará la siguiente pantalla. Introduzca las sentencias en el campo superior.
- Teclée “Ejecutar” para que la sentencia se procese.
- Los resultados se mostraran en el área de texto inferior.
- Siga el procedimiento para ejecutar más sentencia.
- Use el campo “Limpiar” para borrar el contenido en ambos campos.

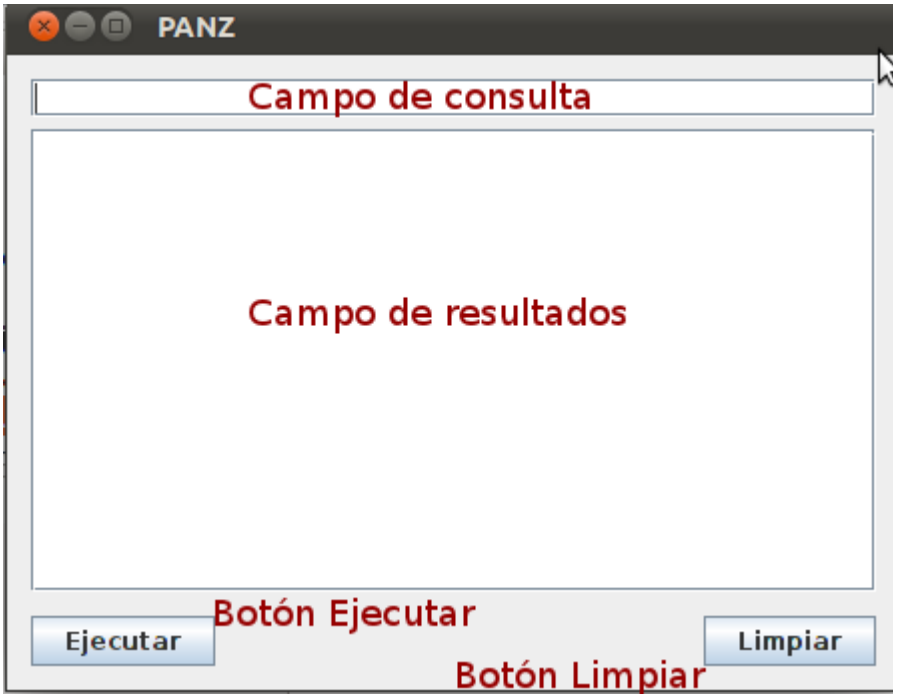
Manual de usuario

Índice

Instalación

Vea la guía rápida.

Ventana



Ejecución

Para ejecutar una sentencia de , situése sobre el campo de consulta y teclée la sentencia. Al terminar presionée el botón “Ejecutar.”

Ejecutar PANZ

Para hacer una consulta y clasificación usando PANZ teclée la instrucción en el campo de consulta con el siguiente formato.

Select *algun criterio del lenguaje SQL* **clustering with** *clasificador;*

SOM

Si desea hacer la clasificación usando usó del algoritmo SOM simplemente ingresée en el campo de consulta :

Select *algun criterio del lenguaje SQL* **clustering with SOM;**

Seguido de esto presionée el botón “Ejecutar”, los resultados se mostrarán en el campo de resultados.

MLP

Si desea hacer la clasificación usando usó del algoritmo MLP simplemente ingresée en el campo consulta:

Select *algun criterio del lenguaje SQL* **clustering with MLP;**

Seguido de esto presione el botón “Ejecutar”, los resultados se mostrarán en el campo de resultados.

Hacer consulta de SQL

En el campo de consulta teclée la sentencia deseada con la sintáxis de SQL. Ejemplo:

Select ** from tmineroconclases;*

Seguido de esto presione el botón “Ejecutar”, los resultados se mostrarán en el campo de resultados.

Limpiar Campos

Cada vez que se realiza una nueva sentencia de SQL o PANZ el campo de resultados se limpiará. Puede borrar los campos de consulta y resultados al mismo tiempo al presionar el botón “Limpiar”.

Salir

Para salir de la ventana de PANZ, simplemente cierre la ventana como cualquier ventana de su sistema operativo.

Maximizar / Minimizar

Utilice los mismos botónes que en su sistema operativo.