Minería de datos usando WEKA

Nombre:

Javier Eduardo Lopez Ontiveros

Materia:

Temas selectos de base de datos

Hora:

11 am a 12pm

**Generación del archivo arff**

try {

// Crear el archivo ARFF

writer = new BufferedWriter(new FileWriter("clientes.arff"));

// Escribir la sección de encabezado

writer.write("@relation Registro\n\n");

writer.write("@attribute Estacion {Primavera, Verano, Otono, Invierno}\n");

writer.write("@attribute year numeric\n");

writer.write("@attribute Edad numeric\n");

writer.write("@attribute NecesitaPermiso {Si, No}\n");

writer.write("@attribute TienePermiso {Si, No}\n");

writer.write("@attribute Genero {Masculino, Femenino}\n");

writer.write("@attribute SeCaso {Si, No}\n");

writer.write("@data\n");

// Escribir la sección de datos

generaDatos();

// Cerrar el archivo

writer.close();

System.out.println("Archivo ARFF generado exitosamente.");

} catch (IOException e) {

e.printStackTrace();

}

Se genera la estructura del archivo arff y se llama al metodo generaDatos para generar los ejemplos ejemplo

**Generación de la tabla de base de datos**

CREATE TABLE Personas (

Estacion VARCHAR(20),

año INT,

Edad INT,

NecesitaPermiso VARCHAR(2),

TienePermiso VARCHAR(2),

Genero VARCHAR(10),

SeCaso VARCHAR(2)

);

se crea la tabla en sql para poder llenarla

try {

con = DriverManager.getConnection(

"jdbc:sqlserver://localhost:1433;encrypt=true;databaseName=Registrocivil;TrustServerCertificate=true;",

user, password);

} catch (SQLException e) {

e.printStackTrace();

}

se establece conexión a la base de datos

**Metodo generaDatos**

private static void generaDatos() {

String[] estaciones = { "Primavera", "Verano", "Otono", "Invierno" };

for (int i = 0; i < 2000; i++) {

StringBuilder ejemplo = new StringBuilder();

double num = Math.random();

int edad = (int) ((Math.random() \* 51) + 9);

ejemplo.append(estaciones[(int) (Math.random() \* 4)] + ", ");

ejemplo.append("" + (int) ((Math.random() \* 3) + 2020) + ", ");

ejemplo.append("" + edad + ", ");

if (num <= 0.95)// se establece el margen de error

ejemplo.append((edad < 18 ? "Si," + (Math.random() >= 0.5 ? "Si" : "No") : "No, No") + ", ");

// si la edad es menor a 18 necesita permiso

else// en caso de que la persona no ocupe permiso se le asigna no tiene permiso

{// caso de falla o corrupcion

ejemplo.append((Math.random() >= 0.5 ? "Si" : "No") + ", ");

ejemplo.append((Math.random() >= 0.5 ? "Si" : "No") + ", ");

}

ejemplo.append((Math.random() >= 0.5 ? "Masculino" : "Femenino") + ", ");

ejemplo.append(Math.random() >= 0.5 ? "Si" : "No");

insertaTabla(ejemplo);

ejemplo.append("\n");

insertaARFF(ejemplo);

}

}

Se utiliza el mismo método para generar los datos del archivo arff y la tabla sql, este metodo crea 2000 ejemplos y en cada iteración se usan un stringbuilder que es la cadena que se va a agregar, se genera aleatoriamente la edad, la estación y el año del registro y se va agregando al stringbuilder, también se establece un rango de error el cual es de 95% correcto, dentro de este if se valida que la edad sea menor a 18, si es menor significa que SI necesita permiso y se usa otra condición para saber si tiene permiso o no, en caso de que la edad sea mayor a 18 se le asigna no ocupa permiso y no tiene permiso,en caso de que sea margen de error se le puede asignar si ocupa permiso y tiene permiso sin importar la edad, al tener la cadena completa se manda al metodo de inserta tabla y inserta arff

**Inserta tabla**

private static void insertaTabla(StringBuilder ejemplo) {

ejemplo.replace(0, ejemplo.length(),

ejemplo.toString().replaceAll("([a-zA-Z]+)", "'$1'"));

try {

Statement stmt = con.createStatement();

stmt.executeUpdate("insert into Personas values(" + ejemplo + ")");

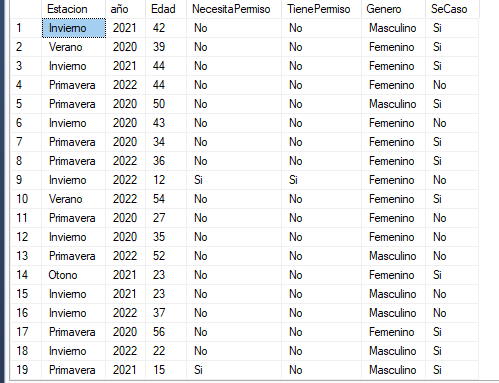
} catch (SQLException e) {

e.printStackTrace();

}

}

se agregan comillas a los datos que lo requieren y se crea un statement con la conexión y se inserta en la tabla personas



**Inserta ARFF**

private static void insertaARFF(StringBuilder ejemplo) {

try {

writer.write("" + ejemplo);

} catch (IOException e) {

e.printStackTrace();

}

}

se escribe en el documento el nuevo ejemplo

@relation Registro

@attribute Estacion {Primavera, Verano, Otono, Invierno}

@attribute year numeric

@attribute Edad numeric

@attribute NecesitaPermiso {Si, No}

@attribute TienePermiso {Si, No}

@attribute Genero {Masculino, Femenino}

@attribute SeCaso {Si, No}

@data

'Invierno', 2021, 42, 'No', 'No', 'Masculino', 'Si'

'Verano', 2020, 39, 'No', 'No', 'Femenino', 'Si'

'Invierno', 2021, 44, 'No', 'No', 'Femenino', 'Si'

'Primavera', 2022, 44, 'No', 'No', 'Femenino', 'No'

'Primavera', 2020, 50, 'No', 'No', 'Masculino', 'Si'

'Invierno', 2020, 43, 'No', 'No', 'Femenino', 'No'

'Primavera', 2020, 34, 'No', 'No', 'Femenino', 'Si'

'Primavera', 2022, 36, 'No', 'No', 'Femenino', 'Si'

'Invierno', 2022, 12, 'Si','Si', 'Femenino', 'No'

'Verano', 2022, 54, 'No', 'No', 'Femenino', 'Si'

'Primavera', 2020, 27, 'No', 'No', 'Femenino', 'No'

'Invierno', 2020, 35, 'No', 'No', 'Femenino', 'No'

'Primavera', 2022, 52, 'No', 'No', 'Masculino', 'No'

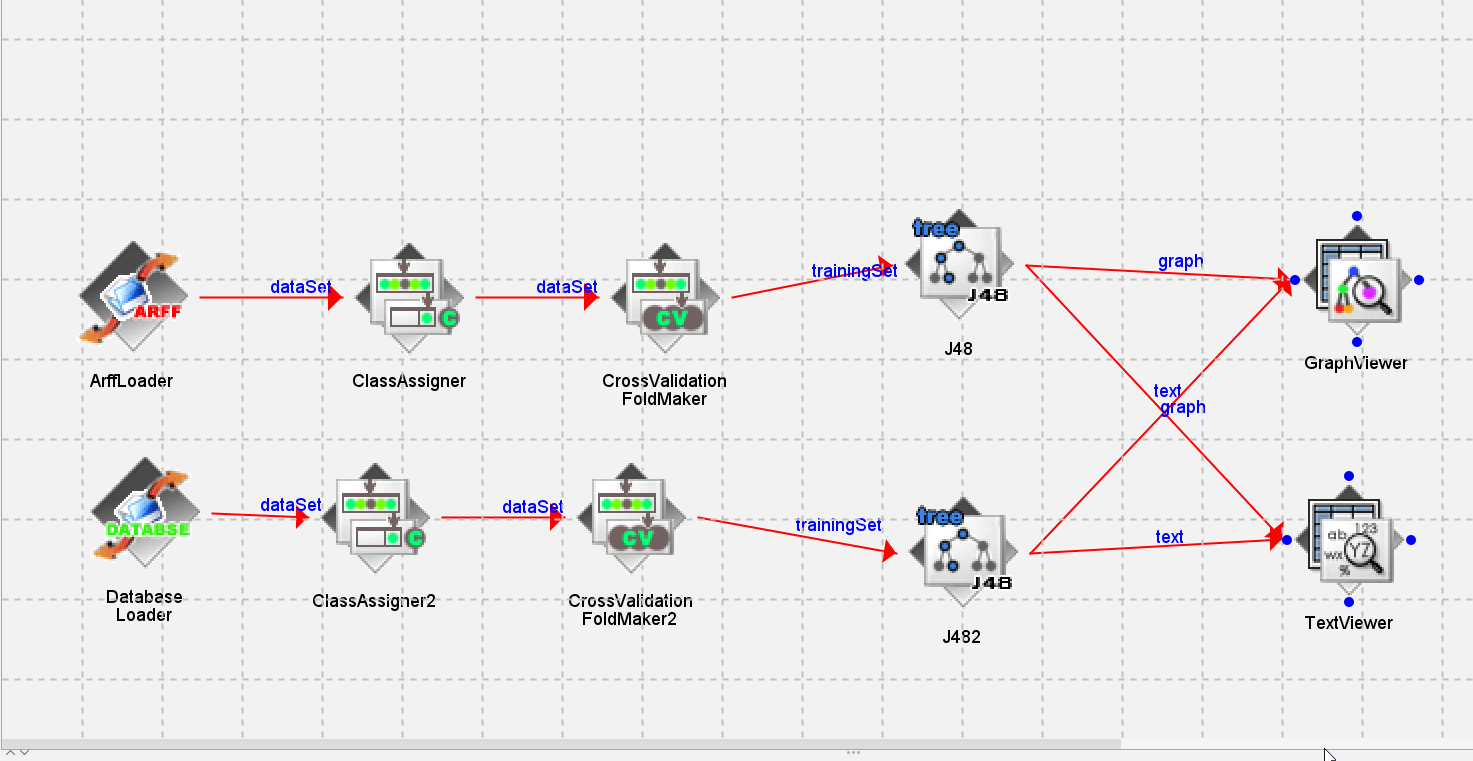
'Otono', 2021, 23, 'No', 'No', 'Femenino', 'Si'

'Invierno', 2021, 23, 'No', 'No', 'Masculino', 'No'

'Invierno', 2022, 37, 'No', 'No', 'Masculino', 'No'

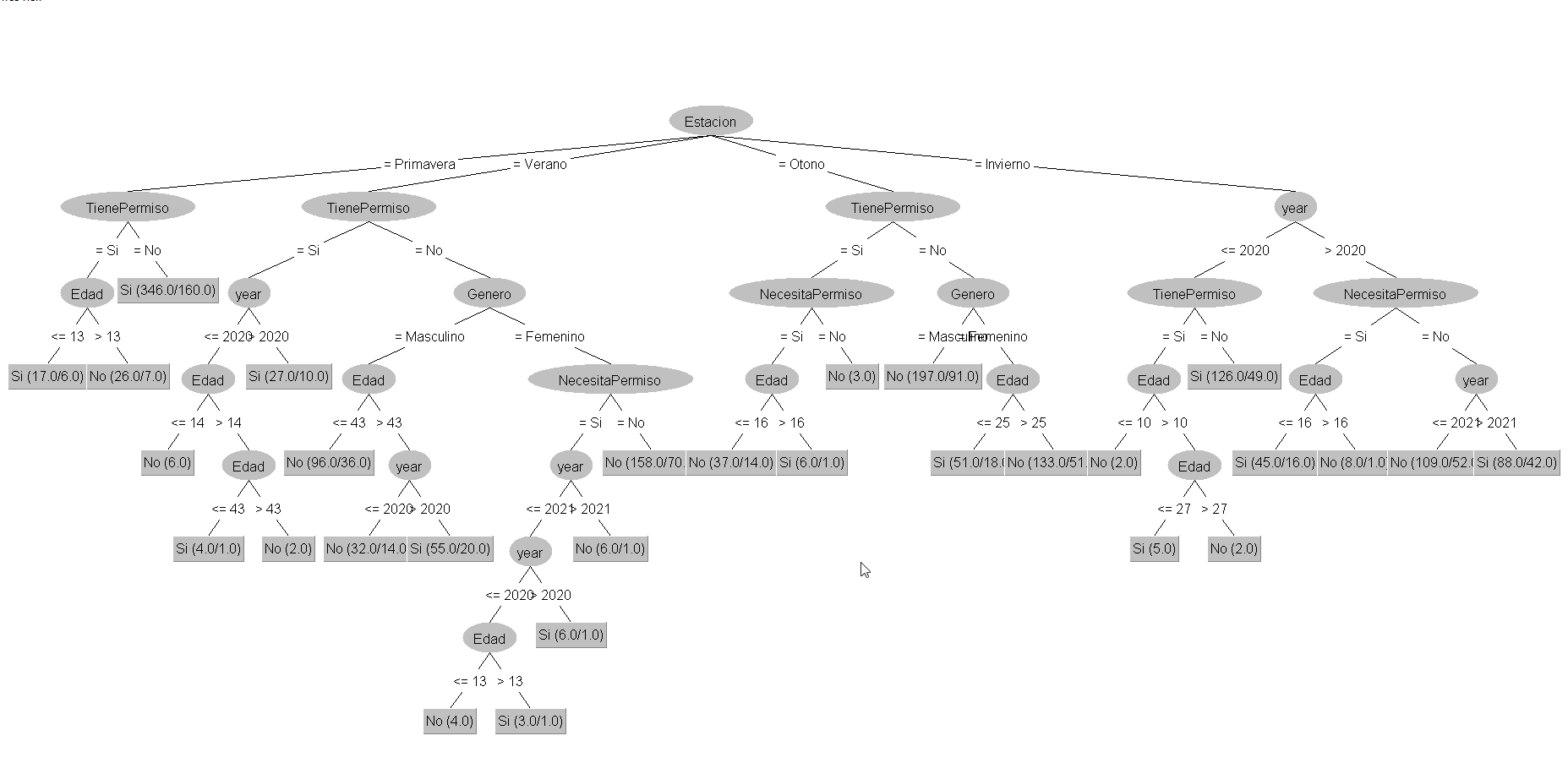
'Primavera', 2020, 56, 'No', 'No', 'Femenino', 'Si'

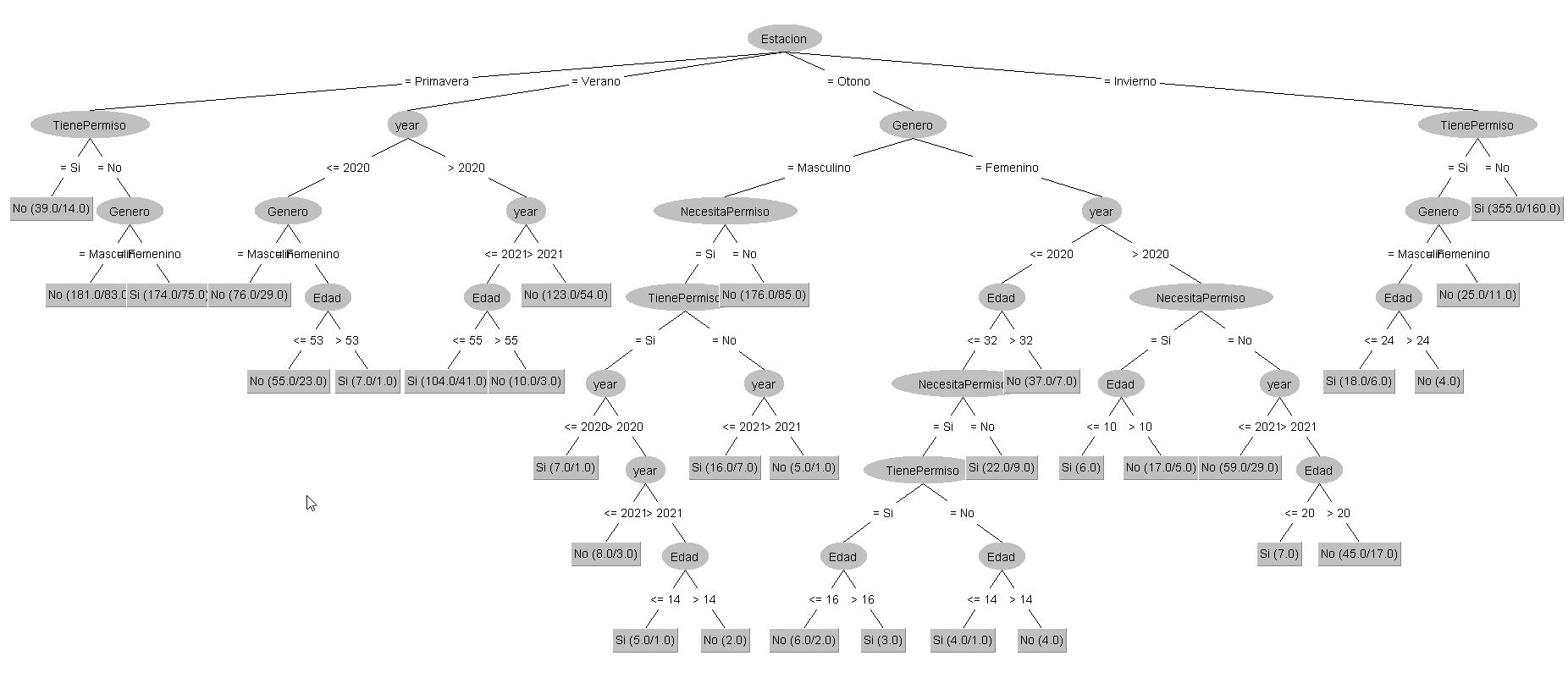
**Flujo de conocimiento**

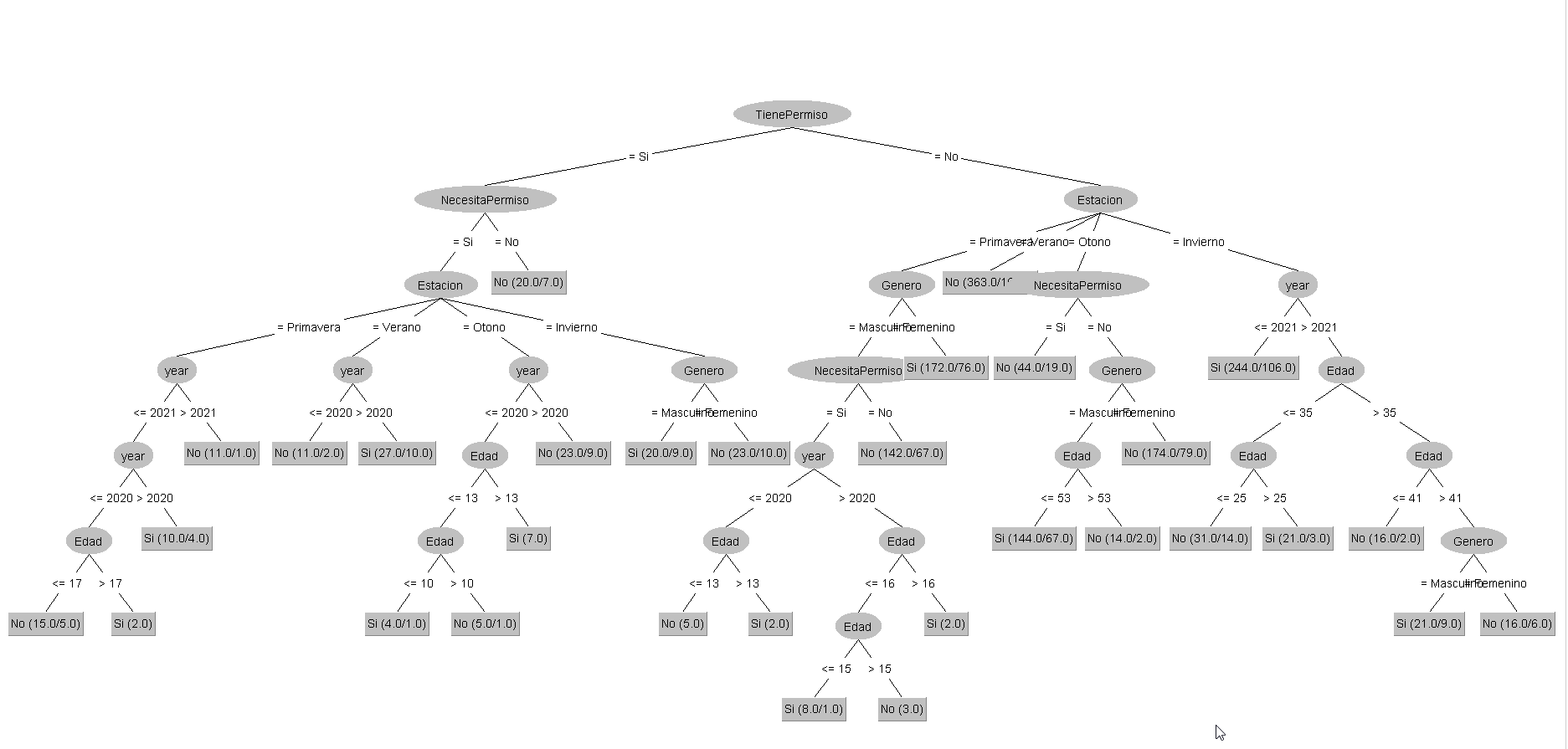


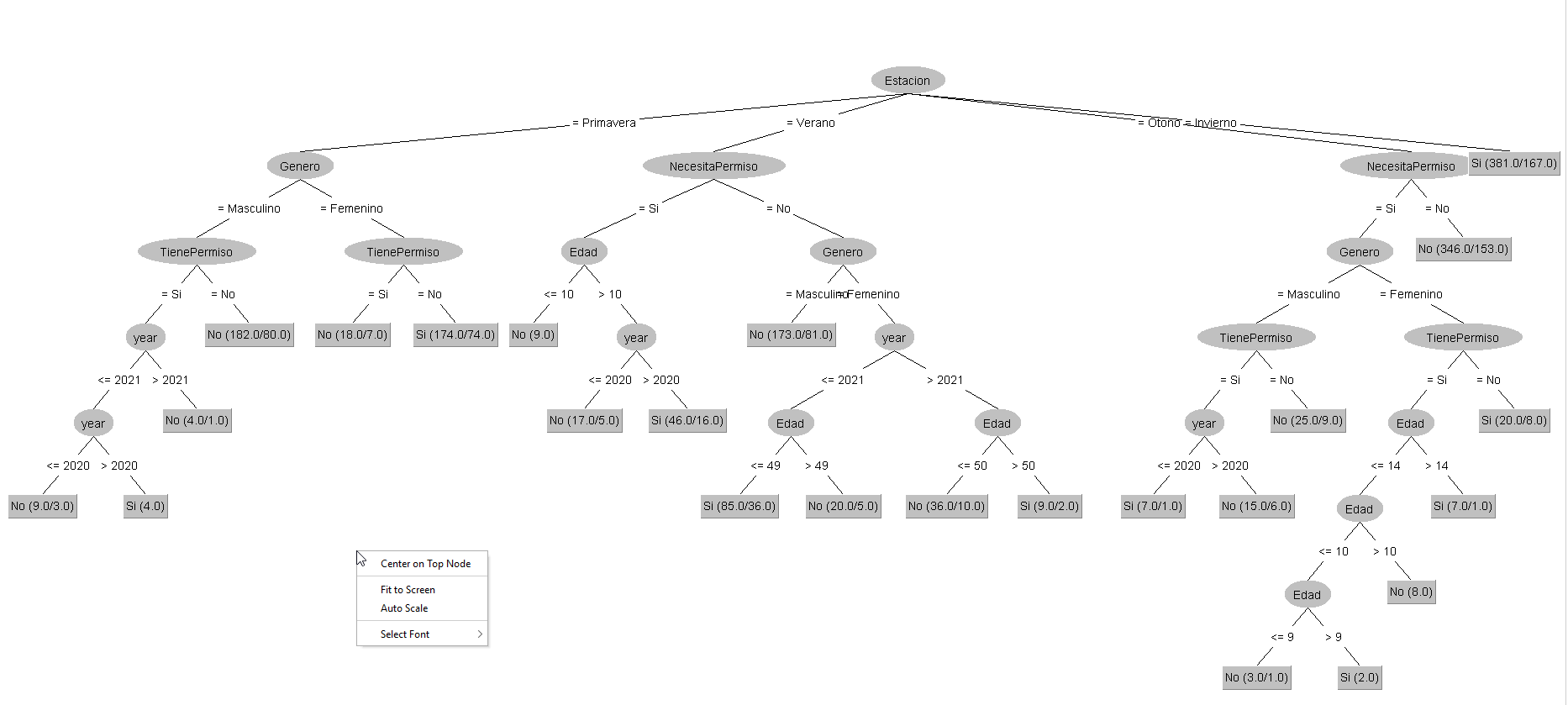
El primer componente es el loader, el cual carga los datos desde la tabla y el archivo arff, después sigue el class asignar este componente lo que hace es determinar en qué atributo va a terminar el árbol, El componente Cross Validation es utilizado para determinar cuántos patrones se van generar, este se asigna al algoritmo J48 que es el que se encarga de generar los patrones, finalmente se muestran los patrones con un Graphviewer y un textView

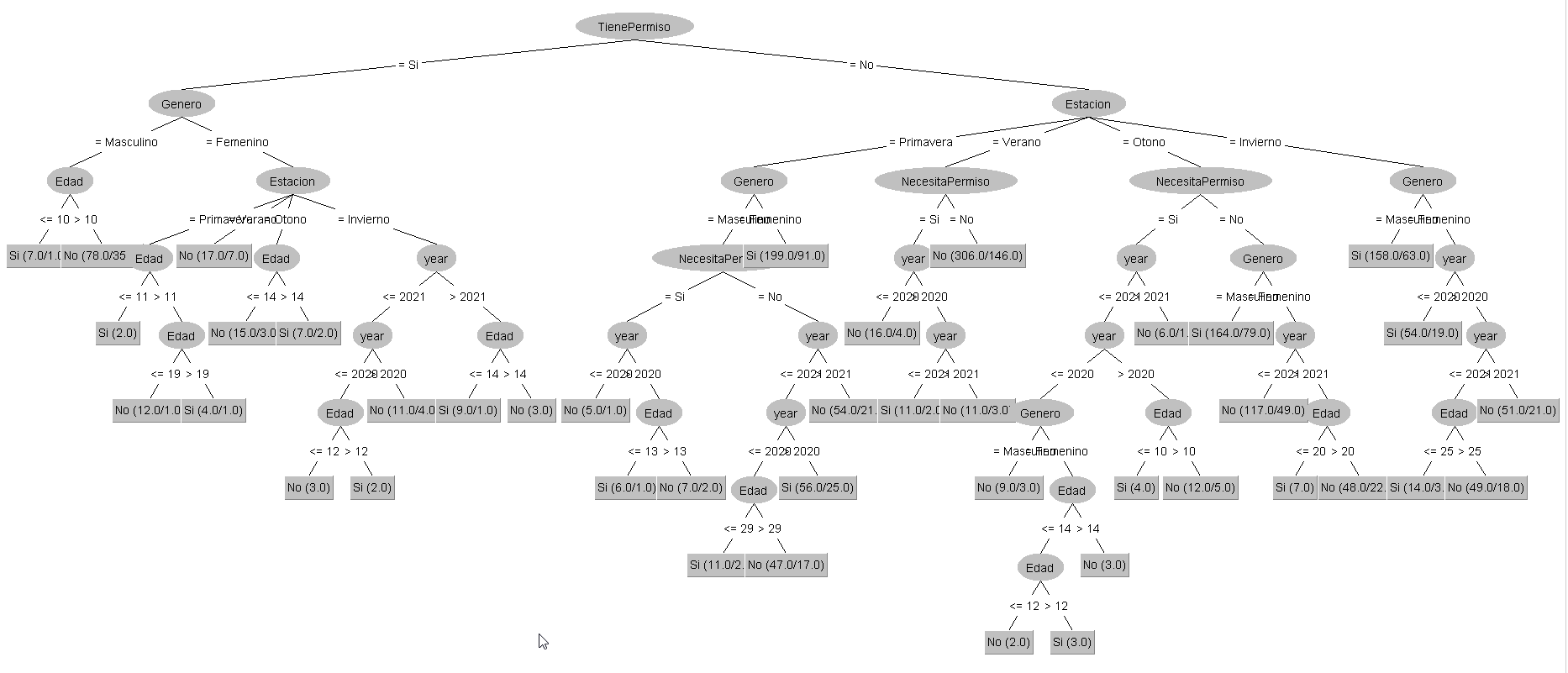
Patrones Generados con el atributo de finalización por “SeCaso”



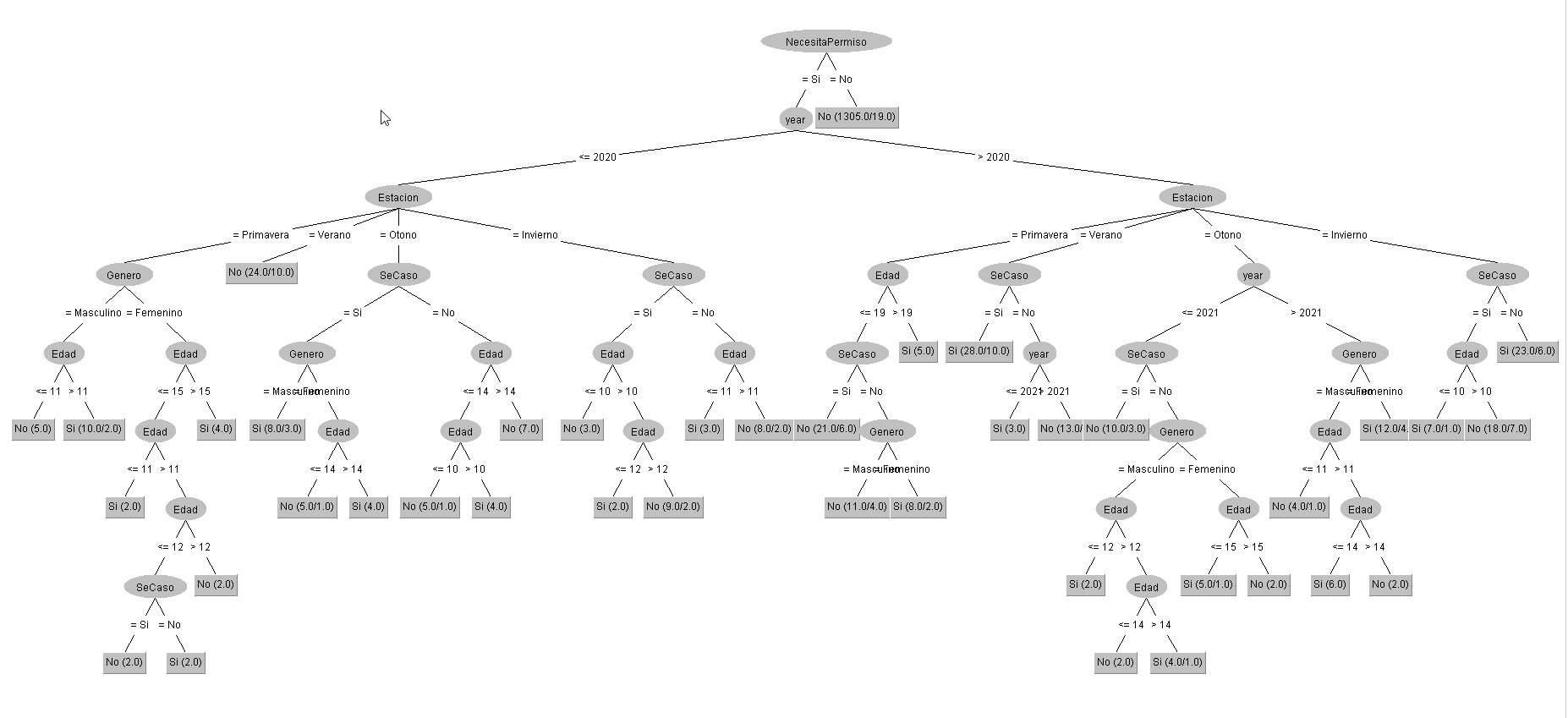


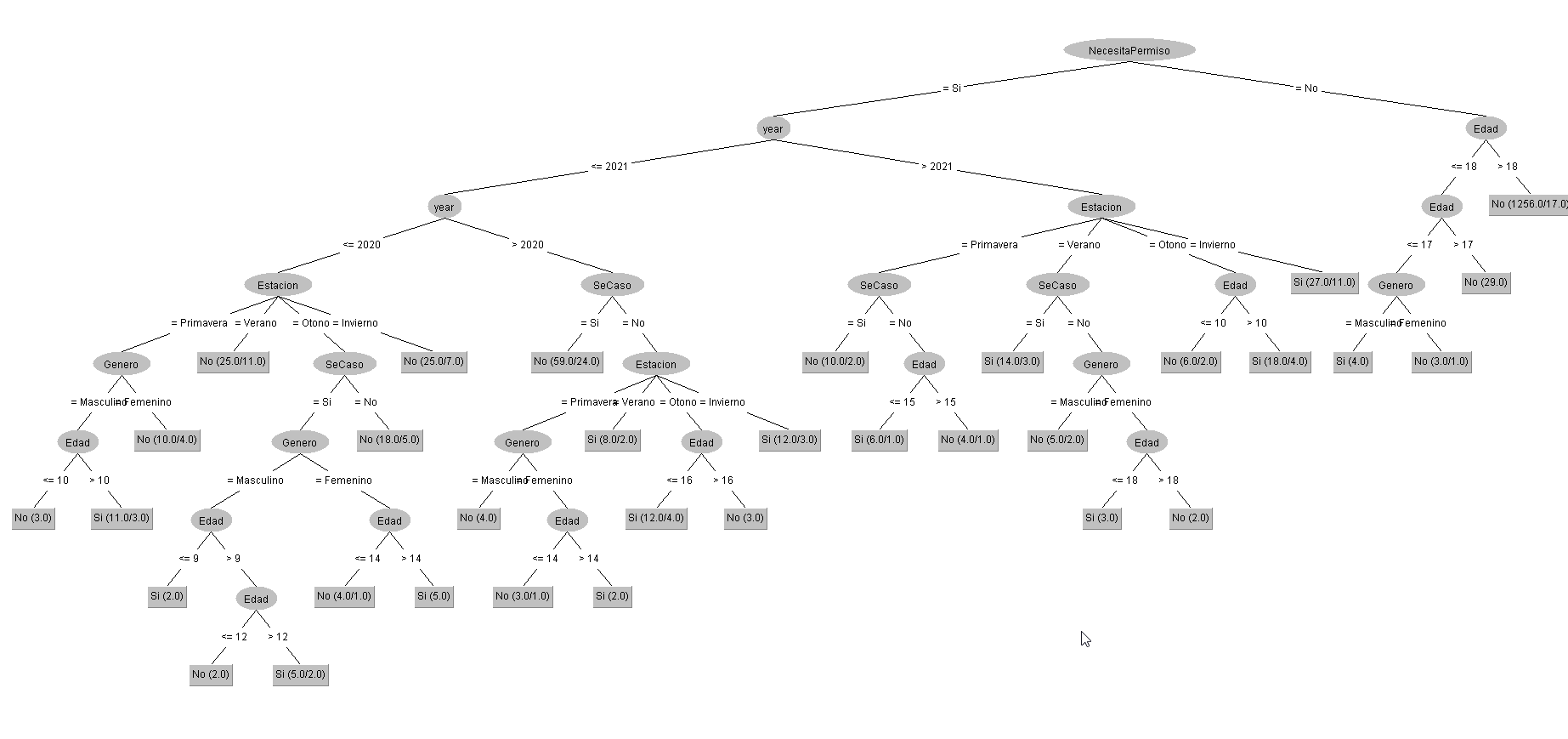


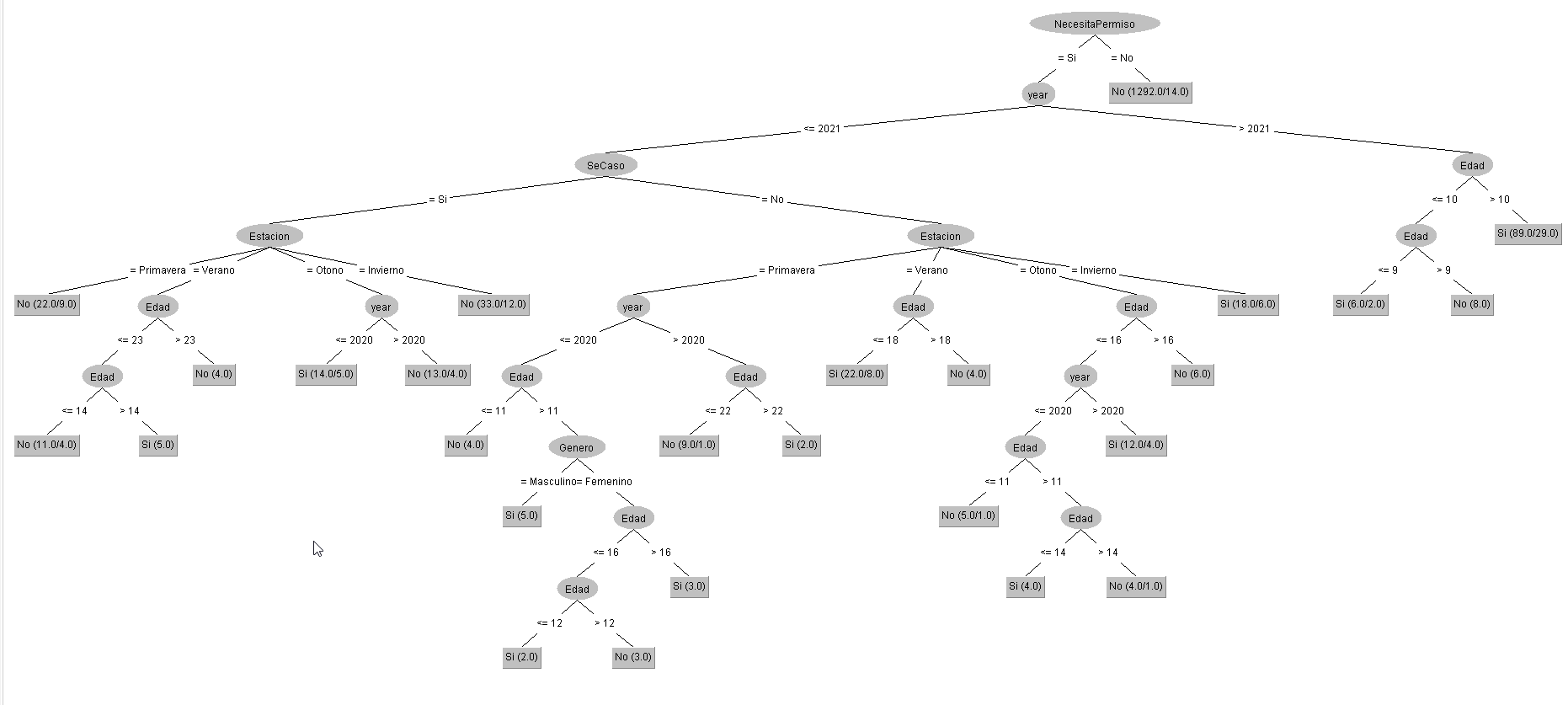


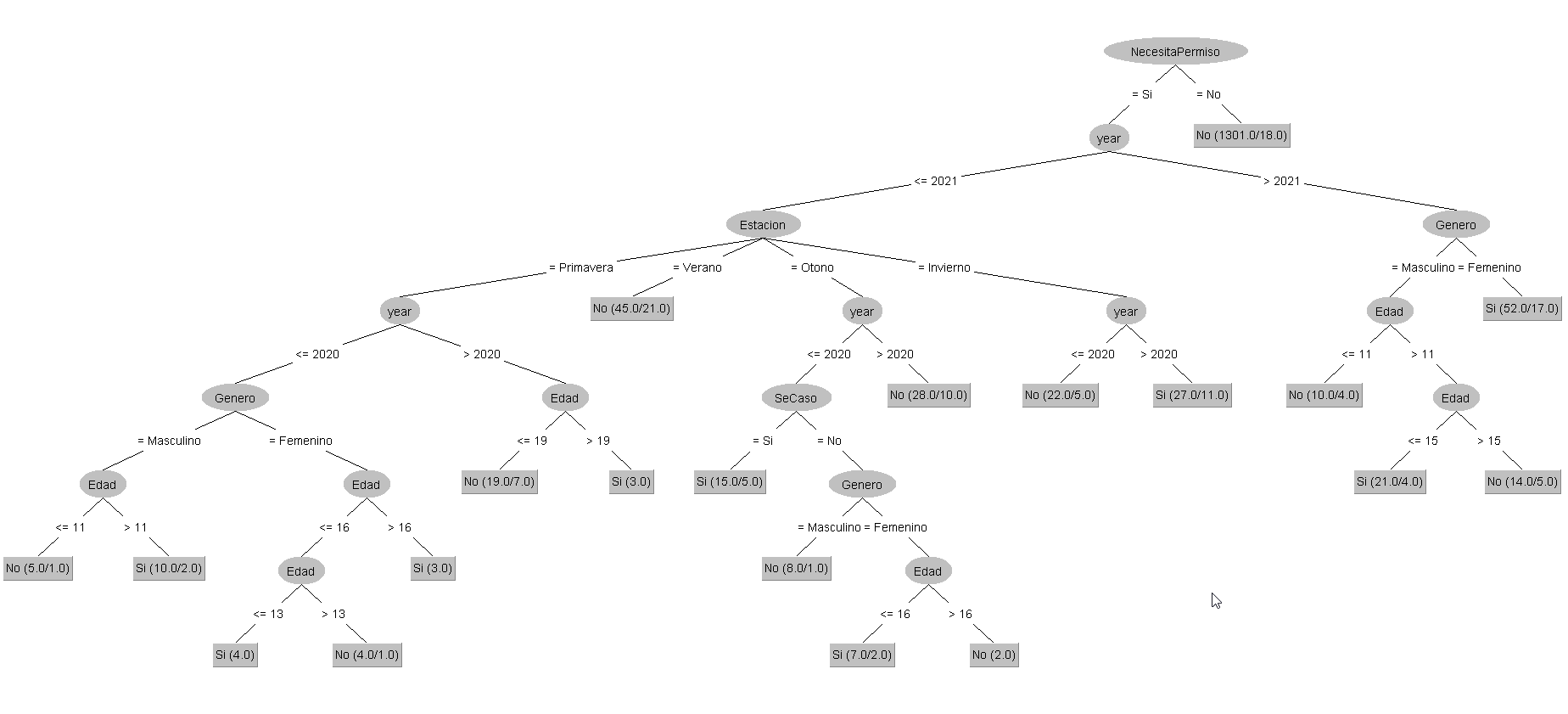


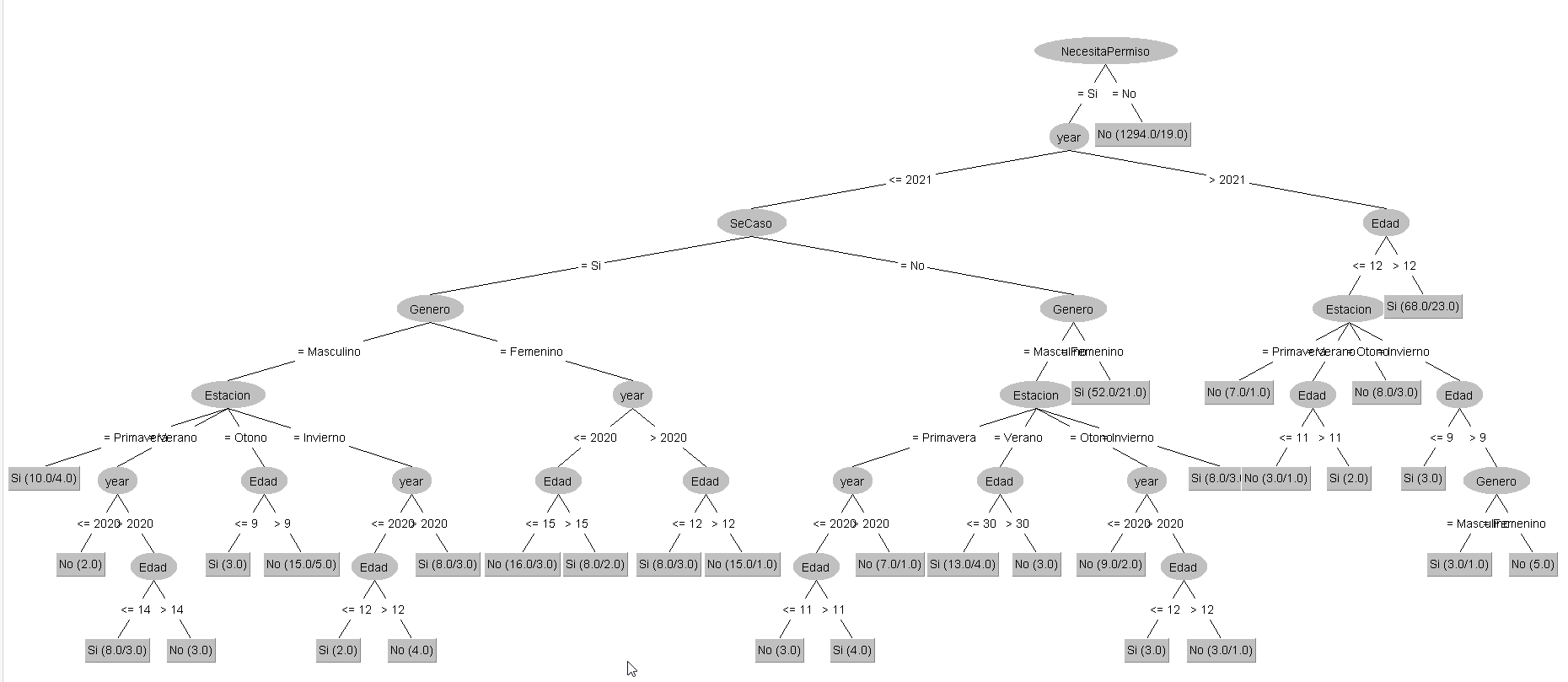
Patrones Generados con el atributo de finalización por “tiene Permiso”



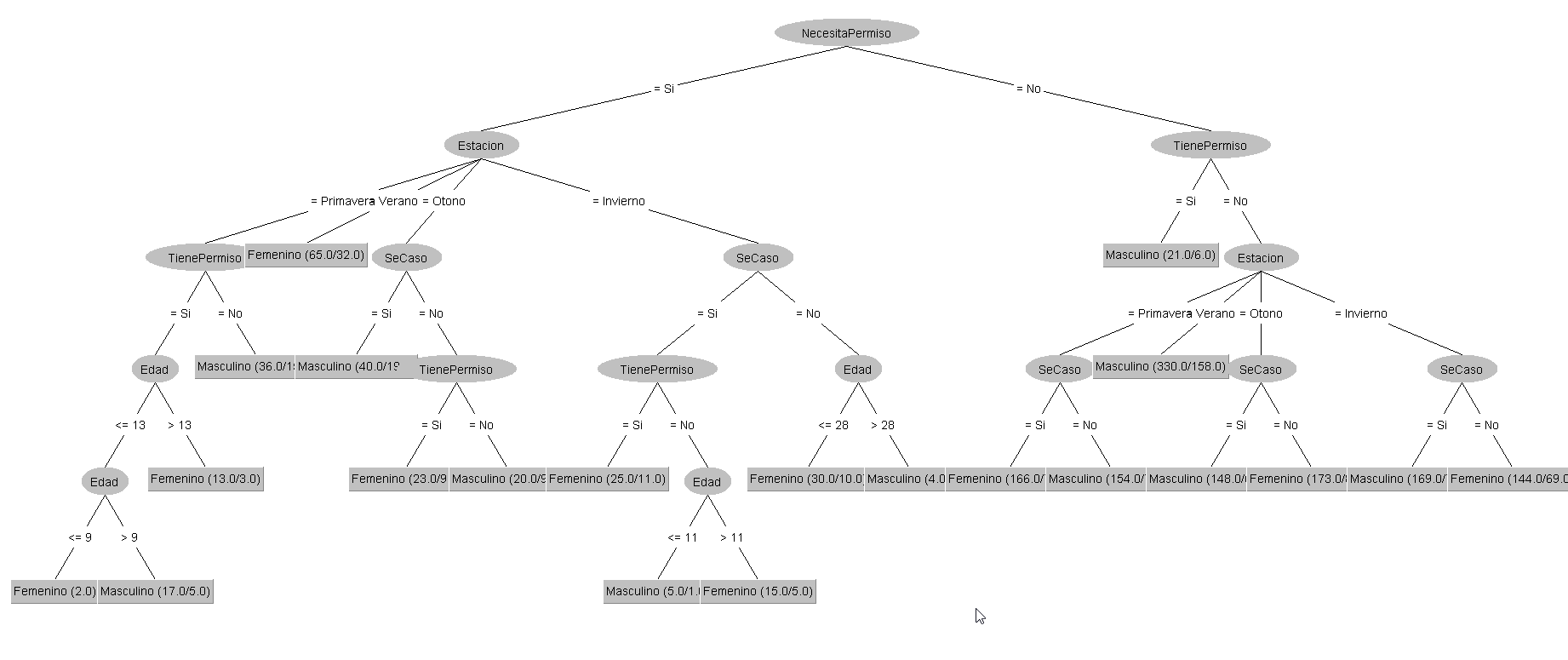


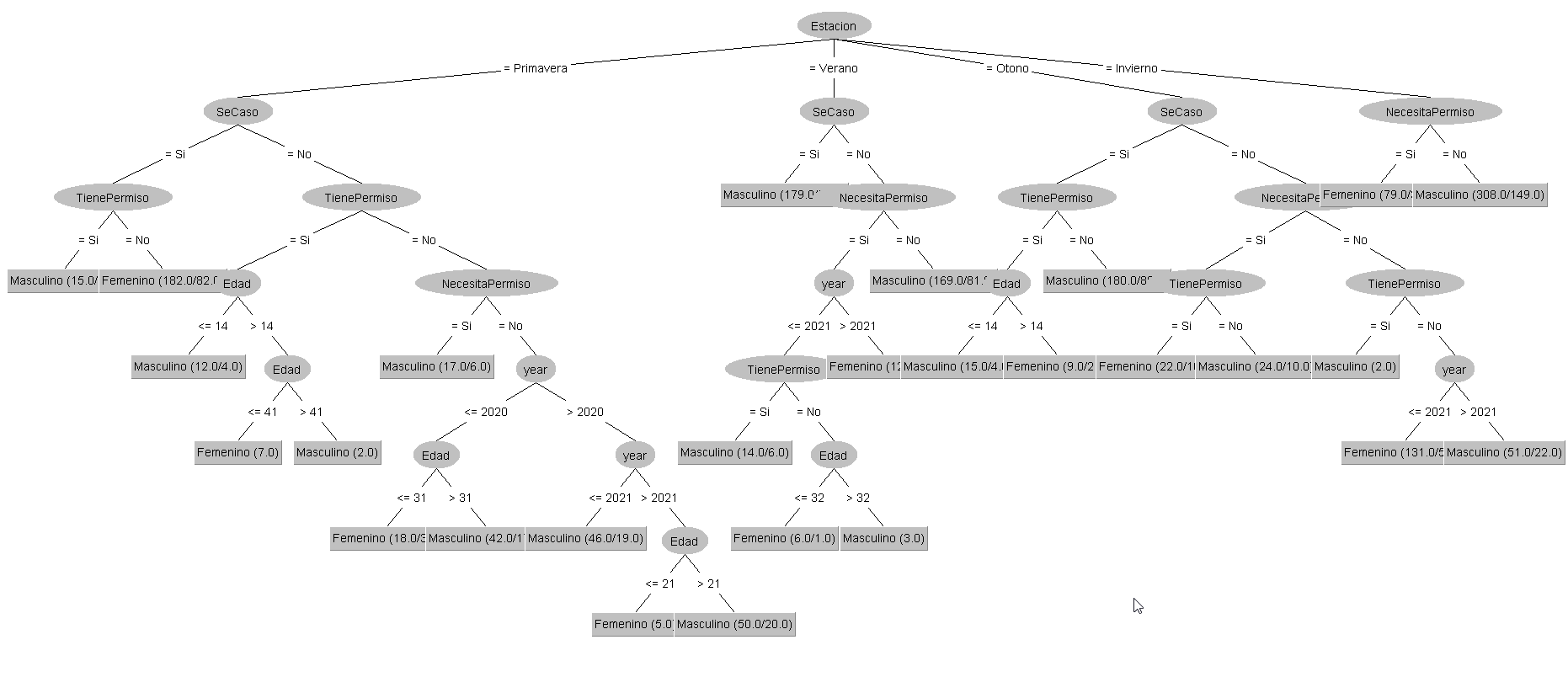


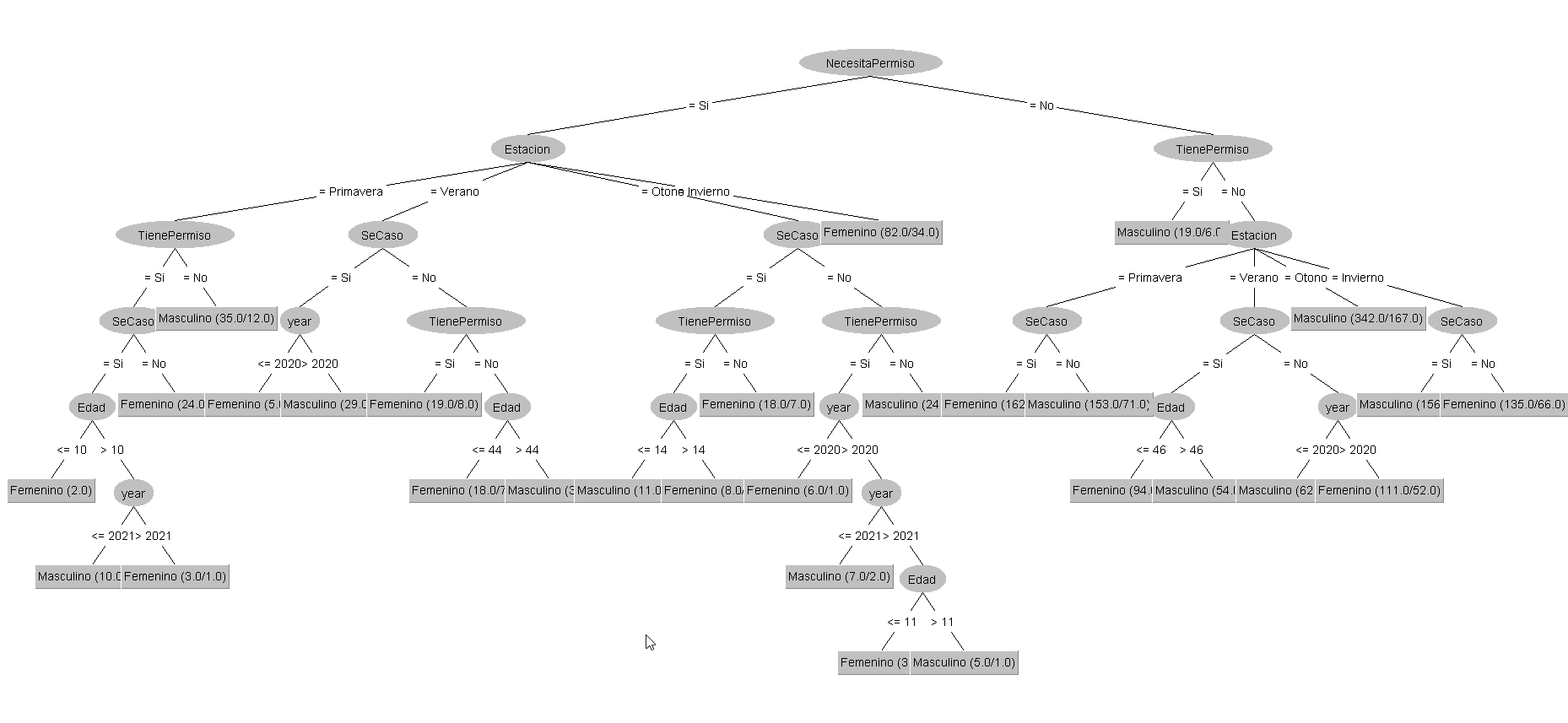


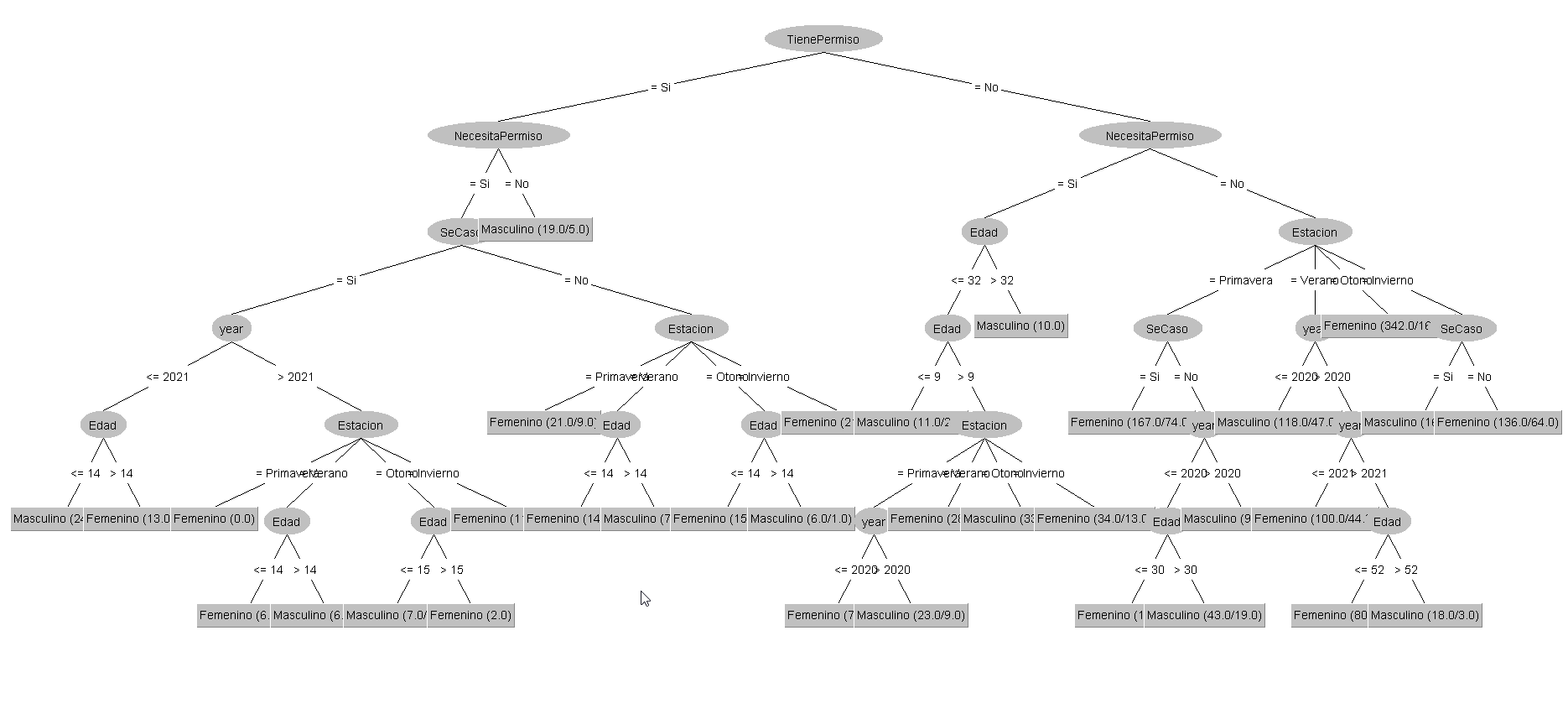


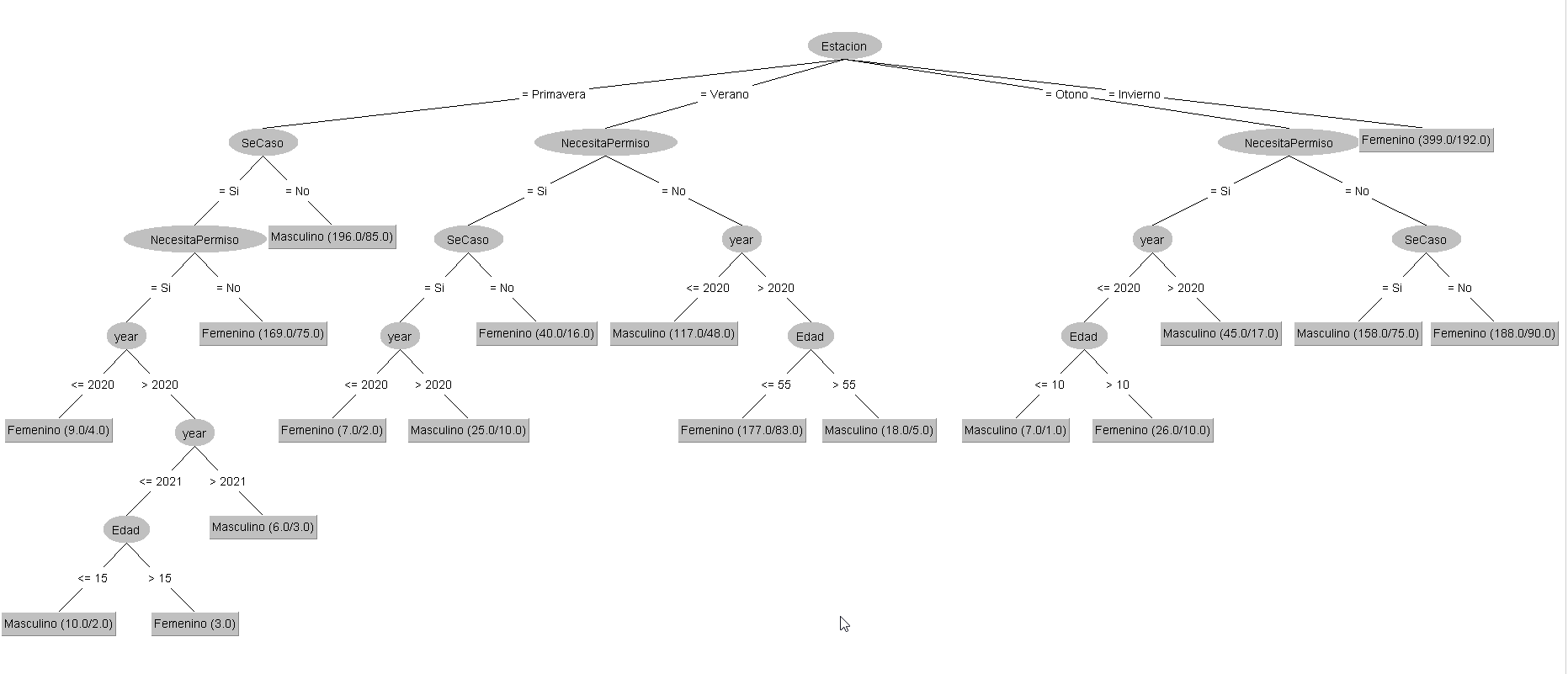
Patrones Generados con el atributo de finalización por “género”



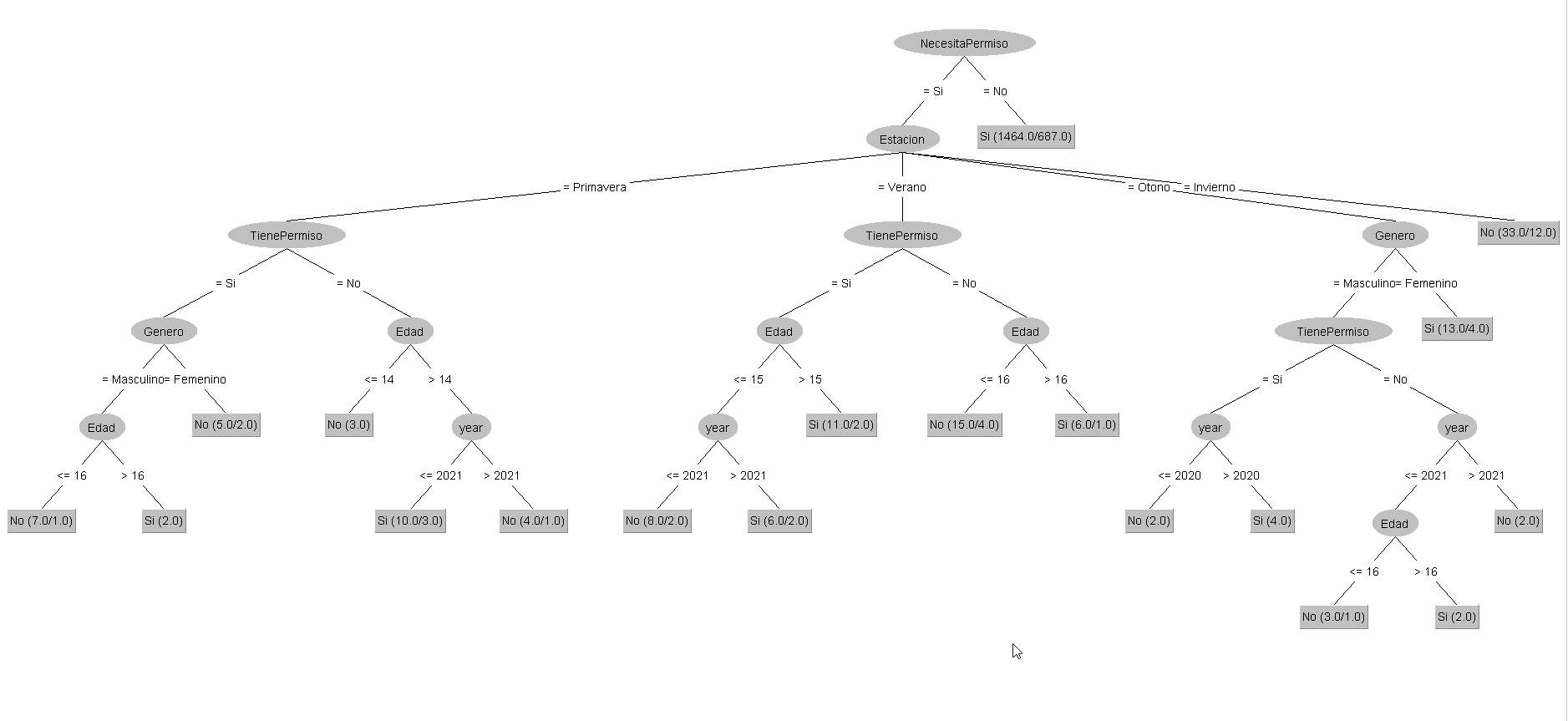




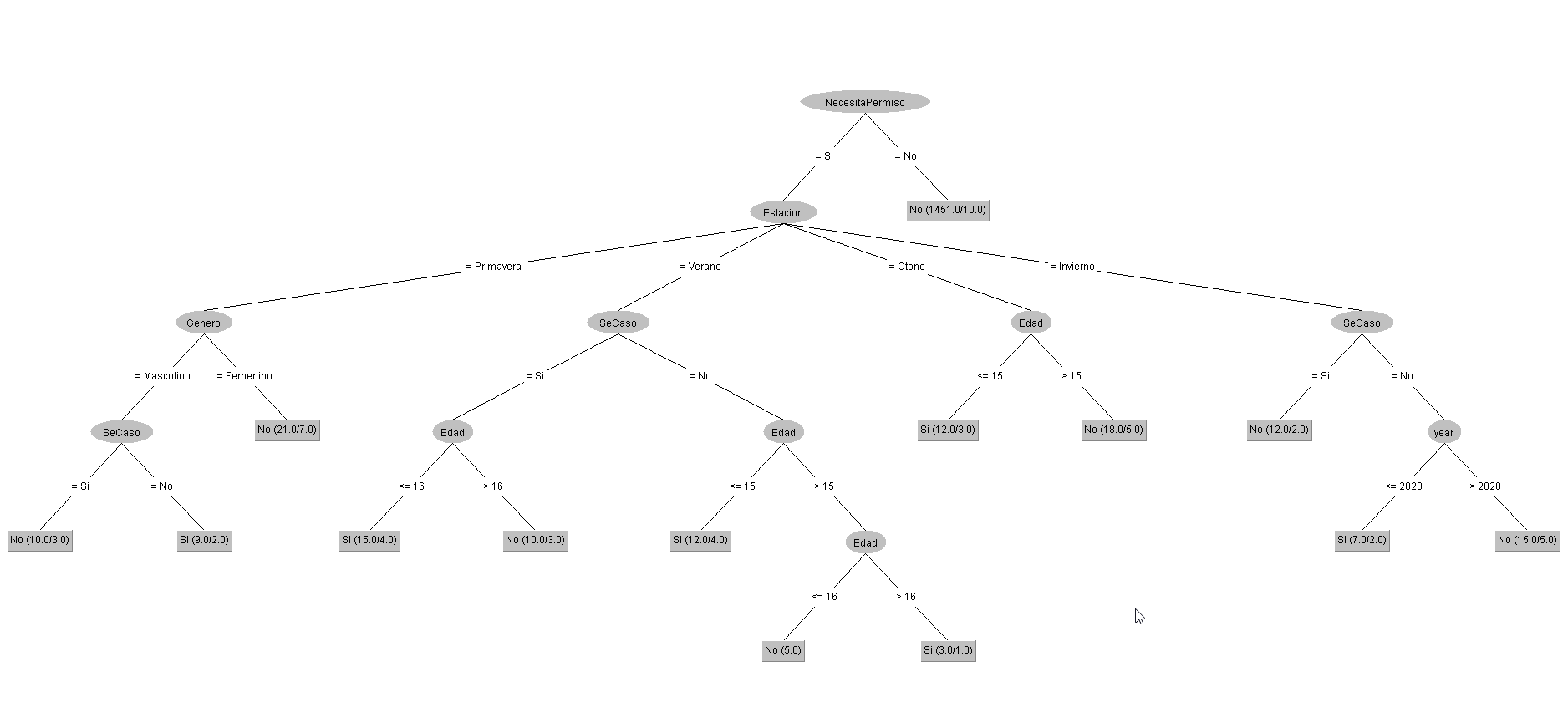




**Interpretando un patrón**

****

Este patrón indica que si una persona no necesita permiso, puede que se case en la mayoría de las veces pero si necesita permiso, se pregunta por la estación del año, en este caso en la rama de invierno no se pudieron casar, por lo que no tiene mucho sentido, otra rama es que sea verano y que la persona tenga permiso y que tenga menos de 15 años y el año sea antes de 2022 entonces la persona no se podrá casar



Este patrón determina que si alguien no necesita permiso, entonces no tiene permiso pero si tiene permiso se pregunta por la estación y si la persona se caso y si su edad es menor a 16 entonces si tiene permiso