



UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

DESARROLLO DE UN FRAMEWORK PARA VISUALIZAR EL CONOCIMIENTO DE MODELOS DESCRIPTIVOS SUPERVISADOS

Alumno: María Asunción Padilla Rascón

Tutor: Prof. D. Cristóbal José Carmona del Jesus
Dpto: Departamento de Informática

Julio, 2023



Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Cristóbal José Carmona del Jesus, tutor del Proyecto Fin de Carrera titulado:
Desarrollo de un framework para visualizar el conocimiento de modelos descriptivos
supervisados, que presenta María Asunción Padilla Rascón, autoriza su
presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, julio de 2023

El alumno:

**PADILLA
RASCON
MARIA
ASUNCION -
26535092S**

Firmado
digitalmente por
PADILLA RASCON
MARIA ASUNCION
- 26535092S
Fecha: 2023.07.03
12:14:05 +02'00'

María Asunción Padilla Rascón

El tutor:

**CARMONA DEL
JESUS
CRISTOBAL
JOSE -
75102603J**

Firmado digitalmente
por CARMONA DEL
JESUS CRISTOBAL
JOSE - 75102603J
Fecha: 2023.07.03
11:50:43 +02'00'

Cristóbal José Carmona del Jesus

TABLA DE CONTENIDOS

CAPÍTULO I. INTRODUCCIÓN	11
1.1. INTRODUCCIÓN	12
1.2. RESUMEN.....	13
1.3. OBJETIVOS	14
1.4. PLANIFICACIÓN Y COSTES	15
1.4.1. <i>Temporal</i>	15
1.4.2. <i>Estimación de costes</i>	17
CAPÍTULO II. ESTADO DEL ARTE	18
2.1. CIENCIA DE DATOS (<i>DATA SCIENCE</i>)	19
2.1.1. <i>Descubrimiento de Reglas con aprendizaje supervisado (SDRD)</i>	20
2.1.2. <i>Medidas de calidad de los algoritmos SDRD</i>	22
2.2. HERRAMIENTAS EXISTENTES DE DIFUSIÓN Y USO DE ALGORITMOS SDRD	24
2.2.1. <i>KEEL</i>	24
2.2.2. <i>Weka</i>	25
2.2.3. <i>Orange</i>	26
2.3. HERRAMIENTAS PARA LA CREACIÓN DE LA APLICACIÓN	29
2.3.1. <i>Librerías para la creación de interfaces gráficas</i>	29
2.3.2. <i>Lenguaje de programación</i>	32
2.3.3. <i>Bibliotecas utilizadas</i>	32
CAPÍTULO III. ANÁLISIS Y DISEÑO	34
3.1. REQUISITOS FUNCIONALES Y NO FUNCIONALES	35
3.2. ANÁLISIS DE CASOS DE USO	36
3.2.1. <i>Sistema en general</i>	36
3.2.2. <i>Generar visualización de las reglas</i>	37
3.2.3. <i>Visualizar resultados globales</i>	38
3.2.4. <i>Visualizar resultados de una regla</i>	39
3.2.5. <i>Exportar</i>	40
3.3. DIAGRAMAS DE SECUENCIA	41
3.3.1. <i>Generar visualización de las reglas</i>	42
3.3.2. <i>Visualizar resultados globales</i>	43
3.3.3. <i>Visualizar resultados de una regla</i>	44
3.3.4. <i>Exportar</i>	45
3.4. ARQUITECTURA DEL SISTEMA	45
3.5. DISEÑO DE LA INTERFAZ	46
3.5.1. <i>Vistas necesarias</i>	47
3.5.2. <i>Storyboard</i>	48

CAPÍTULO IV. IMPLEMENTACIÓN	55
4.1. BACK-END	56
4.1.1. <i>Base de la aplicación</i>	56
4.1.2. <i>Importación de los ficheros</i>	58
4.1.3. <i>Evaluación de las reglas</i>	64
4.1.4. <i>Exportación de la información</i>	67
4.1.5. <i>Difusión de la aplicación</i>	71
4.2. FRONT-END	72
4.2.1. <i>Importar la información</i>	72
4.2.2. <i>Información general</i>	75
4.2.3. <i>Información de una regla individual</i>	83
CAPÍTULO V. CASOS DE PRUEBA	85
5.1. CASO 1: ALGORITMO DIFUSO. CONJUNTO DE DATOS: ECOLI. ALGORITMO: MESDIF86	
5.1.1. <i>Fichero de datos</i>	87
5.1.2. <i>Fichero de reglas</i>	88
5.1.3. <i>Visualización general en la aplicación</i>	89
5.1.4. <i>Información de una de las reglas</i>	92
5.2. CASO 2: ALGORITMO NO DIFUSO O NÍTIDO. CONJUNTO DE DATOS: ECOLI.	
ALGORITMO: CN2	94
5.2.1. <i>Fichero de datos</i>	94
5.2.2. <i>Fichero de reglas</i>	95
5.2.3. <i>Visualización general en la aplicación</i>	96
5.2.4. <i>Información de una de las reglas</i>	98
CAPÍTULO VI. CONCLUSIONES	100
6.1. CONCLUSIÓN	101
6.2. CONOCIMIENTOS ADQUIRIDOS	102
6.3. FUTUROS TRABAJOS	103
ANEXO I. MANUAL DE INSTALACIÓN	104
ANEXO II. MANUAL DE USUARIO	107
BIBLIOGRAFÍA.....	114

LISTADO DE FIGURAS

Figura 1. Fases de la Ciencia de Datos. Fuente: Elaboración propia.....	13
Figura 2. Diagrama de Gantt. Planificación temporal. Elaboración propia	16
Figura 3. Icono KEEL. Fuente: (<i>KEEL web</i> , s. f.).....	25
Figura 4. Icono de Weka. Fuente: («Weka web», s. f.).....	26
Figura 5. Visualización en Weka. Fuente: («Weka web», s. f.).....	26
Figura 6. Icono de Orange. Fuente: (<i>Orange web</i> , s. f.).....	27
Figura 7. Visualización de la información en Orange. Fuente: (<i>Orange web</i> , s. f.)...	27
Figura 8. Librería Tkinter. Fuente: (Vương Quý, 2022)	29
Figura 9. Icono Qt. Fuente: (<i>Qt web</i> , s. f.)	30
Figura 10. Icono JavaFX. Fuente: (<i>JavaFX web</i> , s. f.).....	30
Figura 11. Icono de Python. Fuente: (<i>Python web</i> , s. f.).....	32
Figura 12. Icono Matplotlib. Fuente: (<i>Matplotlib web</i> , s. f.)	32
Figura 13. Icono Numpy. Fuente: (<i>Numpy web</i> , s. f.)	33
Figura 14. Casos de usos de la aplicación. Fuente: Elaboración propia	37
Figura 15. Diagrama de casos de uso de Generar visualización de las reglas. Fuente: Elaboración propia.....	38
Figura 16. Diagrama de casos de uso de Visualizar resultados globales. Fuente: Elaboración propia.....	39
Figura 17. Diagrama de casos de uso de Visualizar resultados de una regla. Fuente: Elaboración propia.....	40
Figura 18. Diagrama de casos de uso de Exportar. Fuente: Elaboración propia.....	41
Figura 19. Diagrama de secuencia de Generar visualización de las reglas. Fuente: Elaboración propia.....	42
Figura 20. Diagrama de secuencia de Visualizar resultados globales. Fuente: Elaboración propia.....	43
Figura 21. Diagrama de secuencia de Visualizar resultados de una regla. Fuente: Elaboración propia.....	44

Figura 22. Diagrama de secuencia de Exportar. Fuente: Elaboración propia	45
Figura 23. Arquitectura del sistema. Fuente: Elaboración propia	46
Figura 24. Vista 1. Vista de la pantalla principal. Fuente: Elaboración propia	48
Figura 25. Vista 2. Vista de la pantalla de importar ficheros. Fuente: Elaboración propia	49
Figura 26. Vista de la ventana de error al importar ficheros. Fuente: Elaboración propia	50
Figura 27. Vista 3. Vista de la información general. Fuente: Elaboración propia.....	51
Figura 28. Vista de la ventana emergente después de exportar. Fuente: Elaboración propia	52
Figura 29. Vista 3.1. Vista de la ventana de la información del dataset. Fuente: Elaboración propia.....	53
Figura 30. Vista 4. Vista de una regla específica. Fuente: Elaboración propia.....	54
Figura 34. Cabecera de un fichero de dataset. Fuente: Elaboración propia.....	58
Figura 35. Cabecera de un fichero de reglas. Fuente: Elaboración propia.....	59
Figura 36. Cabecera de un archivo de reglas difusas. Fuente: Elaboración propia..	59
Figura 37. Diagrama UML del patrón Strategy aplicado en el proyecto. Fuente: Elaboración propia.....	61
Figura 38. Cabecera de un dataset no discretizado. Fuente: Elaboración propia	64
Figura 43. Ventana emergente de selección de fichero a importar. Fuente: Elaboración propia.....	72
Figura 44. Visualización de showinfo(). Fuente:	73
Figura 45. Visualización de showwarning(). Fuente:	73
Figura 46. Visualización de showerror(). Fuente:	73
Figura 47. Visualización de askquestion(). Fuente:	74
Figura 48. Visualización del mensaje del error04 al importar ficheros. Fuente: Elaboración propia.....	74
Figura 49. Visualización del warning01 al importar ficheros. Fuente: Elaboración propia	74

Figura 51. Ventana que muestra la cabecera del conjunto de datos. Fuente: Elaboración propia.....	75
Figura 52. Ventana emergente indicando que ha finalizado la exportación. Fuente: Elaboración propia.....	76
Figura 53. Ventana emergente para exportar la información. Fuente: Elaboración propia	77
Figura 54. Gráfico de puntos generado con Matplotlib. Fuente: Elaboración propia	78
Figura 55. Gráfico de pirámide generado con Matplotlib. Fuente: Elaboración propia	79
Figura 56. Visualización de las gráficas en la pantalla de información general. Fuente: Elaboración propia.....	80
Figura 57. Visualización de la tabla de las reglas que cubren a cada dato con reglas no difusas. Fuente: Elaboración propia	81
Figura 58. Visualización de la tabla de las reglas que cubren a cada dato con reglas difusas. Fuente: Elaboración propia	81
Figura 59. Visualización del tooltip en la pantalla de la información general. Fuente: Elaboración propia.....	82
Figura 60. Visualización de las tablas de contingencias y de medidas. Fuente: Elaboración propia.....	83
Figura 61. Visualización del tooltip en la pantalla de una regla específica. Fuente: Elaboración propia.....	84
Figura 62. Caso 1: Vista general de la información general de las reglas. Fuente: Elaboración propia.....	91
Figura 63. Información concreta de la regla 5. Fuente: Elaboración propia	93
Figura 64. Caso 2: Vista general de la información general de las reglas. Fuente: Elaboración propia.....	97
Figura 65. Información concreta de la regla 17. Fuente: Elaboración propia	98
Figura 66. Mockup del árbol jerárquico de reglas. Fuente: Elaboración propia	103
Figura 67. Pantalla inicial. Fuente: Elaboración propia.....	108
Figura 68. Importación de ficheros inicialmente. Fuente: Elaboración propia	109

Figura 69. Selección de fichero del conjunto de datos. Fuente: Elaboración propia	109
Figura 70. Mensaje de error por no corresponder dataset con reglas. Fuente: Elaboración propia.....	110
Figura 71. Selección de ficheros realizada. Fuente: Elaboración propia	110
Figura 72. Vista inicial de la información general. Fuente: Elaboración propia.....	111
Figura 73. Visualización de la cabecera del conjunto de datos. Fuente: Elaboración propia	111
Figura 74. Vista inicial de una única regla. Fuente: Elaboración propia	112

LISTA DE TABLAS

Tabla 1. Planificación y costes del proyecto. Fuente: Elaboración propia	17
Tabla 2. Tabla de contingencias.....	22
Tabla 3. Resumen de las características de las herramientas de difusión y uso de algoritmos SDRD	28
Tabla 4. Análisis textual general.....	36
Tabla 5. Análisis textual de Generar visualización de las reglas	37
Tabla 6. Análisis textual de Visualizar resultados globales	38
Tabla 7. Análisis textual de Visualizar resultado de una regla.....	39
Tabla 8. Análisis textual de Exportar	40
Tabla 9. Descripción de las vistas	47
Tabla 10. Tabla de los posibles errores que se pueden producir al importar un fichero	63
Tabla 11. Cabecera de la tabla de las reglas que cubren a los datos. Fichero LaTeX	70
Tabla 12. Cabecera de la tabla de las reglas que cubren a cada dato	80

LISTA DE ECUACIONES

Ecuación 1. Cálculo del TPr	23
Ecuación 2. Cálculo del FPr	23
Ecuación 3. Cálculo de la Confianza de una regla	23
Ecuación 4. Cálculo del WRAcc de una regla	24
Ecuación 5. Cálculo del WRAcc normalizado. Fuente: (Carmona et al., 2018)	24

LISTA DE CÓDIGOS

Código 1. Iniciación de la aplicación.....	56
Código 2. Creación de las pantallas de la aplicación	57
Código 3. Cambio de pantallas en la aplicación	57
Código 4. Subdivisión del intervalo en intervalos más pequeños	66
Código 5. Cálculo del valor de pertenencia y etiqueta de un atributo.....	66
Código 6. Exportación de los gráficos a SVG.....	69
Código 7. Cabecera fichero LaTeX.	70

Capítulo I

INTRODUCCIÓN

Este capítulo comenzará con una introducción que justifica la realización del proyecto y explicará su importancia. A continuación, se describirán los objetivos específicos del proyecto y se presentará un resumen de toda la memoria. Finalmente, se incluirá la planificación temporal del proyecto y una estimación de sus costes, con el fin de proporcionar una visión completa del proyecto y sentar las bases para su desarrollo y éxito.

1.1. Introducción

En la actualidad, la disciplina de la informática de la Ciencia de Datos (*Data Science*) está en pleno auge, puesto que la cantidad de información recogida digitalmente está en continuo crecimiento y, gracias a las técnicas de Ciencia de Datos, se puede extraer conocimiento relevante de dicha información para la toma de decisiones (Hernández Orallo et al., 2010).

La utilización de la Ciencia de Datos conlleva varias fases (Fayyad et al., 1996):

1. Recopilación de los datos. En esta fase se agrupa toda la información de diferentes fuentes de la que se quiera obtener conocimiento.
2. Selección, limpieza y transformación. En esta fase los datos recogidos en el anterior paso pasan por un proceso de filtrado donde se seleccionan las variables que se quieren estudiar, se eliminan datos incompletos o erróneos y se transforman de manera que la técnica de Ciencia de Datos pueda tratar esta información.
3. Minería de datos. En esta fase se decide qué algoritmo se va a aplicar sobre el conjunto de datos en función de la técnica que se quiera aplicar (aprendizaje supervisado o no supervisado, descriptivo o predictivo...) y la forma en la que se quiere obtener los resultados (conjunto de reglas, árboles de decisión, etc).
4. Evaluación. Los resultados obtenidos en la anterior fase son evaluados por expertos para conocer la calidad del modelo resultante y si es necesario repetir el experimento modificando ciertos parámetros para mejorarlo u observar otros aspectos.
5. Difusión y uso. Una vez se han obtenido los modelos deseados y el conocimiento implícito en la información recopilada, es necesario presentar esta información de manera amigable y fácil de entender para el usuario.

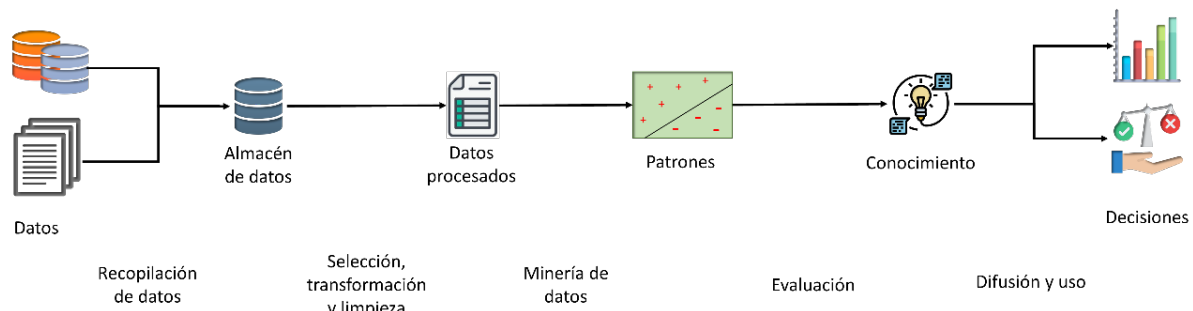


Figura 1. Fases de la Ciencia de Datos. Fuente: Elaboración propia

Los científicos de datos se han ido especializando en cada una de las etapas descritas anteriormente para que la Ciencia de Datos esté lo más optimizada posible y se encuentren los mejores resultados posibles. Pero, existe una carencia en la quinta fase debido a que no se ha invertido igual de trabajo y tiempo en realizar interfaces que sean usables y amigables para que cualquier tipo de usuario sea capaz de interpretar los resultados sin necesidad de que sea o no un experto en la materia.

En definitiva, este trabajo intentará cubrir esa carencia que existe en la disciplina de las Ciencia de Datos, elaborando una interfaz gráfica (GUI) usable que permita a cualquier usuario consultar los resultados obtenidos de las ejecuciones de manera gráfica y tabular.

1.2. Resumen

A continuación, se indicará un breve resumen de lo que se encuentra recogido en esta memoria.

En el capítulo 2 se exponen algunos aspectos extraídos de la investigación realizada sobre la disciplina de la informática Ciencia de Datos y los algoritmos de descubrimiento de reglas descriptivas con aprendizaje supervisado (SDRD), así como la investigación de diferentes frameworks de elaboración de interfaces gráficas y las tecnologías finalmente escogidas. En el capítulo 3 quedan recogidos todos los aspectos relacionados con la ingeniería del software del proyecto. En el capítulo 4 se describe cómo se ha realizado la implementación de los elementos principales de la aplicación, detallando los problemas surgidos durante la misma y cómo se han solventado. En el capítulo 5 se han descrito un par de casos de prueba de los innumerables que se han realizado para comprobar el funcionamiento y la calidad de

la aplicación. En el capítulo 6 se indican las conclusiones extraídas de la elaboración de este proyecto y algunas ideas de mejora para el futuro. Finalmente, se han incluido dos anexos que le indican al usuario cómo se realiza la instalación y puesta en marcha de la aplicación, y cómo se utiliza.

1.3. Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación que permita al usuario visualizar y analizar de manera eficiente la información relativa a las reglas descriptivas supervisadas obtenidas de los algoritmos SDRD sin que sea necesario modificar la implementación de éstos para adecuarse a la aplicación. La aplicación se enfocará en brindar información gráfica y tabular de manera clara y concisa para facilitar la comprensión y la identificación de patrones y tendencias en los datos. Además de la visualización de la información de las reglas, la aplicación también permitirá al usuario acceder a la información del conjunto de datos de manera sencilla, así como exportar la información que está visualizando en la aplicación. La posibilidad de exportar la información es esencial para permitir el análisis posterior y la integración con otras herramientas y sistemas.

La aplicación debe incluir:

- Visualización de los resultados obtenidos de los algoritmos de descubrimiento de reglas descriptivas con aprendizaje supervisado SDRD.
- Que sea usable, amigable y fácil de usar para usuarios expertos y no expertos.
- Que sea modular, y que, de manera sencilla, permita a los expertos la inclusión de nuevos algoritmos en la aplicación.
- Que sea capaz de procesar los resultados de todos los algoritmos implementados en ella para mostrar al usuario final los resultados gráficamente de forma intuitiva.

Para llevar a cabo esta interfaz será necesario:

- Realizar una revisión del estado del arte dentro del descubrimiento de reglas descriptivas con aprendizaje supervisado.
- Evaluar la calidad de algoritmos en entornos complejos como *Data Streaming* y *Big Data*, mediante el análisis de los ficheros de cualquier conjunto de reglas y su lote de datos de datos asociado.

1.4. Planificación y costes

Tras haber presentado la motivación detrás de este proyecto y los objetivos principales que se deben alcanzar, es importante establecer una planificación clara del mismo. Esta planificación incluye la subdivisión en tareas y los costes asociados a su desarrollo.

La subdivisión en tareas permitirá una gestión eficiente del proyecto, ya que permite establecer plazos y responsabilidades específicas para cada una de ellas. Además, permite controlar el progreso del proyecto y hacer ajustes en caso de que sea necesario.

1.4.1. Temporal

En la Figura 2 se muestra la distribución de tareas específicas para este proyecto junto con el tiempo estimado que se necesitaría para completarlas. Cada celda en la tabla de la figura representa una semana, y el tiempo estimado se indica en días. Es importante destacar que el reparto de tareas se ha realizado considerando una jornada laboral de 25 horas semanales con una tarifa horaria de 13,85€ («talent.com», 2023).



Figura 2. Diagrama de Gantt. Planificación temporal. Elaboración propia

1.4.2. Estimación de costes

A continuación, se presenta una tabla que detalla la duración y el costo estimado para cada una de las tareas del proyecto, de acuerdo con las condiciones previamente mencionadas y representadas en el diagrama de Gantt (Figura 2).

Tareas	Duración (días)	Coste (€)
Investigar herramientas de representación de información en data science	8	554
Planificación del proyecto	3	207,75
Estudio ingeniería de software	4	277
Estudio documentación librerías de interfaces gráficas	8	554
Estudio documentación Tkinter, Matplotlib, Numpy, Python	18	1246,5
Implementación pantallas principales sin elementos gráficos	6	415,5
Implementación de la importación de ficheros y el patrón Strategy	18	1246,5
Implementación de la evaluación de las reglas	14	969,5
Inclusión de elementos gráficos en la interfaz	12	831
Implementación de la exportación de la información	10	692,5
Puesta a prueba de la aplicación	3	207,75
Despliegue de la aplicación	5	346,25
Documentación de la aplicación	16	1108
TOTAL	125	8656,25

Tabla 1. Planificación y costes del proyecto. Fuente: Elaboración propia

Capítulo II

ESTADO DEL ARTE

En esta sección, se profundizará en la investigación llevada a cabo sobre los algoritmos de descubrimiento de reglas descriptivas con aprendizaje supervisado y su relevancia en la Ciencia de Datos. Se abordarán los conceptos fundamentales que sustentan esta rama de investigación, así como las técnicas y herramientas más relevantes para su aplicación. Así mismo, se presentará una revisión detallada de las librerías estudiadas para llevar a cabo la aplicación y las herramientas utilizadas en la implementación. Se explicará cómo se seleccionaron y evaluaron las librerías, así como las características y funcionalidades más relevantes de cada una.

2.1. Ciencia de Datos (*data science*)

La Ciencia de Datos como toda disciplina informática surgió para cubrir una necesidad, la necesidad de extraer conocimiento a partir de los datos almacenados y utilizarlo para la toma de decisiones (Cao, 2017).

Esta necesidad surgió y persiste debido al aumento del uso y aplicación de las nuevas tecnologías en cada vez más ámbitos de nuestra vida diaria, provocando a su vez el crecimiento de datos que se registran en bases y almacenes de datos, siendo imposible de gestionar para los humanos (Dhar, 2013).

El estudio manual y las herramientas SQL y OLPT eran las herramientas previas a la aparición de la Ciencia de Datos, y éstas únicamente obtenían diferentes representaciones de los datos almacenados según los modelos establecidos previamente por los expertos. En cambio, la Ciencia de Datos extrae de los datos conocimiento que permite generar modelos completamente nuevos.

El resultado de la aplicación de la Ciencia de Datos es un conjunto de reglas, árboles de decisión, redes neuronales, grafos probabilísticos, etc; que permiten resolver cuestiones que se plantea el usuario como *¿existe un grupo de clientes que se comporta de manera diferenciada?*, *¿qué secuenciación de tratamientos puede ser más efectiva para este nuevo síndrome?*, etc. (Dhar, 2013).

Existen dos enfoques principales en la minería de datos, normalmente se decide aplicar uno u otro (Cao, 2017):

- Aprendizaje supervisado: Se basa en un conjunto de datos etiquetados, es decir, se tienen ejemplos de entrada junto con sus correspondientes etiquetas de salida. El objetivo es aprender una función que pueda mapear las entradas a las salidas correctas. El algoritmo recibe retroalimentación (*feedback*) durante el proceso de entrenamiento para ajustar sus parámetros y mejorar la precisión de las predicciones.
- Aprendizaje no supervisado: Se basa en un conjunto de datos no etiquetados. En este enfoque, el algoritmo debe identificar patrones o estructuras en los datos sin tener información previa sobre las etiquetas

de salida. El objetivo principal es descubrir la estructura oculta en los datos, agruparlos o encontrar relaciones entre las variables.

Por otro lado, se pueden encontrar dos tipos de modelos al aplicar la Ciencia de Datos (Han et al., 2011).

- **Modelo predictivo:** Estos modelos intentan obtener valores futuros o desconocidos de las variables de interés u objetivo, a partir de otras variables almacenadas en la base datos, llamadas variables independientes.
- **Modelo descriptivo:** Estos modelos identifican patrones en los datos para obtener información de las relaciones obtenidas, no se utilizan para realizar predicciones.

Comúnmente, en la literatura a los modelos predictivos también se les denomina como aprendizaje supervisado y a los modelos descriptivos como aprendizaje no supervisado; aunque no son términos cien por cien iguales, ya que el aprendizaje supervisado se centra sobre todo en la clasificación y el aprendizaje no supervisado se ajusta sobre todo al agrupamiento (Hernández Orallo et al., 2010).

A pesar de que se distingan dos tipos de modelos diferentes, existe un modelo que se encuentra entre los dos, el descubrimiento de reglas descriptivas con aprendizaje supervisado.

2.1.1. Descubrimiento de Reglas con aprendizaje supervisado (SDRD)

El descubrimiento de reglas descriptivas con aprendizaje supervisado, como se ha mencionado anteriormente, se encuentra entre el modelo predictivo y el modelo descriptivo, ya que hace uso de reglas y un modelo descriptivo para tratar de obtener conocimiento mediante aprendizaje supervisado que sea inusual y de interés para el usuario (Carmona et al., 2018).

Dentro de esta vertiente de la Ciencia de Datos se encuentran tres técnicas:

- Descubrimiento de subgrupos (SD): A partir de una variable de interés indicada por el usuario, describe relaciones inusuales e interesantes en los datos (Klösgen, 1996). Estas relaciones son reglas que se denominan subgrupos (Wrobel, 1997).
- Minería de patrones emergentes (EP): Tiene dos vertientes, respecto a una variable de tiempo describen tendencias emergentes en los datos o bien, buscan diferentes características teniendo como referencia una variable de interés (García-Vico et al., 2020). La idea principal es obtener patrones muy precisos independientemente del número de ejemplos positivos cubiertos.
- Minería de conjuntos de contraste (CS): Consiste en obtener grupos que describen relaciones inusuales entre las variables. La idea principal es que los grupos obtenidos cubran un gran número de ejemplos (Bay & Pazzani, 2001).

La diferencia entre estas técnicas reside en que los algoritmos SD tratan de describir situaciones inusuales en el espacio de búsqueda teniendo en cuenta un valor concreto de la variable objetivo (Herrera et al., 2011); mientras que los algoritmos EP intentan encontrar patrones (reglas) con una tasa de crecimiento alta, es decir, tasa de ejemplos bien cubiertos frente a los mal cubiertos, por eso describen patrones emergentes en los datos (García-Vico et al., 2018). Por otro lado, los algoritmos de CS intentan buscar ese mismo concepto, pero analizando la diferencia entre los ejemplos cubiertos correctamente y los ejemplos cubiertos erróneamente (Bay & Pazzani, 2001).

La representación del conocimiento más empleada en las técnicas de SDRD se define mediante una regla del tipo:

$$R: Cond \rightarrow Valor\ objetivo$$

Donde *Cond* es un conjunto de pares atributo-valor y *Valor objetivo* es el valor analizado para la variable objetivo (Carmona et al., 2018).

2.1.2. Medidas de calidad de los algoritmos SDRD

Cuando se ejecuta un algoritmo SDRD sobre un conjunto de datos, se generan reglas descriptivas supervisadas que permiten describir o predecir las características del conjunto de datos. A continuación, se van a describir las medidas de calidad que normalmente se les aplica a estas reglas para medir su calidad y valorar la eficacia del algoritmo.

La representación de las reglas descriptivas supervisadas obtenidas siempre tiene la misma forma. Esta característica permite definir una misma tabla de contingencias (Tabla 2) para cualquier regla R .

	Positivos (Ejemplos de la clase)	Negativos (Ejemplos de la no clase)	
Cubiertos	$p = tp$	$n = fp$	$p + n$
No cubiertos	$\bar{p} = fn$	$\bar{n} = tn$	$\bar{p} + \bar{n}$
	$p + \bar{p} = P$	$n + \bar{n} = N$	$P + N = T$

Tabla 2. Tabla de contingencias

Donde:

- p y tp (verdaderos positivos) es el número de ejemplos que han sido cubiertos correctamente,
- n y fp (falsos positivos) es el número de ejemplos que han sido erróneamente cubiertos porque no pertenecen a la clase,
- $\bar{p}$ y fn (falsos negativos) es el número de ejemplos que pertenecen a la clase pero no han sido cubiertos, y
- $\bar{n}$ y tn (verdaderos negativos) es el número de ejemplos que no pertenecen a la clase y no han sido cubiertos.
- $p + n$ es el número total de ejemplos cubiertos por la regla.
- $\bar{p} + \bar{n}$ es el número total de ejemplos que no han sido cubiertos por la regla.
- P es el número total de ejemplos que pertenecen a la clase.
- N es el número total de ejemplos que no pertenecen a la clase.
- T es el número total de ejemplos del conjunto de datos.

A partir de la tabla de contingencias, que compara las predicciones de la regla generada por el algoritmo SDRD con los valores reales del conjunto de datos, se pueden obtener diversas medidas de calidad que permiten evaluar la eficacia y utilidad de la regla. Las medidas más utilizadas en la literatura para medir la calidad de las reglas descriptivas supervisadas son (García-Vico et al., 2018):

- TPr (*True Positive rate*): Es el ratio de verdaderos positivos respecto al total de los ejemplos que pertenecen a la clase. Una regla se considera de buena calidad si el número de verdaderos positivos (tp) se acerca lo máximo posible al número de datos que pertenecen a la clase, es decir, que su TPr sea lo más cercano a 1.

$$TPr = \frac{tp}{tp + fn}$$

Ecuación 1. Cálculo del TPr

- FPr (*False Positive rate*): Es la proporción de falsos positivos respecto al número de los ejemplos que no pertenecen a la clase. Una regla se considera de buena calidad si el número de falsos positivos (fp) es cercano a 0, es decir, que su FPr sea lo más cercano a 0 posible.

$$FPr = \frac{fp}{fp + tn}$$

Ecuación 2. Cálculo del FPr

- Conf (Confianza): Mide la precisión de la regla con respecto a los ejemplos que cubre. Su valor se mide entre 0 y 1 y el objetivo es maximizar esta medida. Se calcula como:

$$Conf(R) = \frac{tp}{tp + fp}$$

Ecuación 3. Cálculo de la Confianza de una regla

- WRAcc (Exactitud relativa ponderada): Mide el equilibrio entre la generalidad y la precisión, también es conocida como *unusualness* y dota al problema de generalidad, precisión e interés. El objetivo es maximizar esta medida. Se calcula como:

$$WRAcc(R) = \frac{tp + fp}{tp + fp + tn + fn} \left(\frac{tp}{tp + fp} - \frac{tp + fn}{tp + fn + tn + fp} \right)$$

Ecuación 4. Cálculo del WRAcc de una regla

No obstante, resulta crucial emplear una versión normalizada de esta medida, ya que su alcance está condicionado por la distribución de ejemplos de la clase que se esté analizando (Carmona et al., 2018), normalizando su valor en un intervalo dependiente del problema al valor [0, 1] donde reglas buenas deben tener valores superiores a 0,55. Se calcula de la siguiente manera:

$$LB_{WRAcc} = \left(1 - \frac{P}{T}\right) * \left(0 - \frac{P}{T}\right)$$

$$UB_{WRAcc} = \frac{P}{T} * \frac{N}{T}$$

$$WRAccN(R) = \frac{WRAcc - LB_{WRAcc}}{UB_{WRAcc} - LB_{WRAcc}}$$

Ecuación 5. Cálculo del WRAcc normalizado. Fuente: (Carmona et al., 2018)

2.2. Herramientas existentes de difusión y uso de algoritmos SDRD

En esta subsección se comentarán las herramientas que existen actualmente para el uso y visualización de los algoritmos de descubrimiento de reglas descriptivas con aprendizaje supervisado (SDRD).

2.2.1. KEEL

KEEL¹ es una herramienta de software libre desarrollada en lenguaje JAVA que cuenta con diversos algoritmos de aprendizaje automático como clasificación, regresión y clustering.

Aunque KEEL no posee ninguna funcionalidad de visualización de algoritmos, permite realizar ejecuciones de los algoritmos integrados utilizando las bases de datos incluidas en la aplicación. De esta manera, KEEL es una herramienta útil para obtener

¹ <https://sci2s.ugr.es/keel/description.php>

los ficheros de datos en el formato compatible con la aplicación del proyecto y para realizar las ejecuciones de los algoritmos y obtener las reglas que posteriormente serán evaluadas.

Además, KEEL cuenta con herramientas para preprocesamiento de datos, selección de características y validación de modelos, lo que facilita el proceso de aprendizaje automático y mejora la calidad de las reglas obtenidas.



Figura 3. Icono KEEL. Fuente: (KEEL web, s. f.)

2.2.2. Weka

Weka² es una herramienta de software libre desarrollada en la Universidad de Waikato en Nueva Zelanda que proporciona una amplia variedad de algoritmos de aprendizaje automático y minería de datos. La aplicación es muy popular entre los investigadores y profesionales de la Ciencia de Datos debido a su amplia gama de funcionalidades, su facilidad de uso y su capacidad para trabajar con una amplia variedad de formatos de datos.

Weka permite a los usuarios cargar conjuntos de datos desde una variedad de fuentes, preprocesar los datos y aplicar una variedad de algoritmos de aprendizaje automático y análisis de datos para generar modelos predictivos y reglas descriptivas.

Weka no tiene algoritmos de reglas descriptivas tipo SDRD como tal. Únicamente tiene algoritmos de aprendizaje automático supervisado basados en reglas como JRip y PART, que son capaces de generar reglas descriptivas precisas y útiles a partir de un conjunto de datos etiquetados, pero estos no son algoritmos de reglas descriptivas tipo SDRD.

² <https://www.cs.waikato.ac.nz/ml/weka/>



Figura 4. Icono de Weka. Fuente: («Weka web», s. f.)

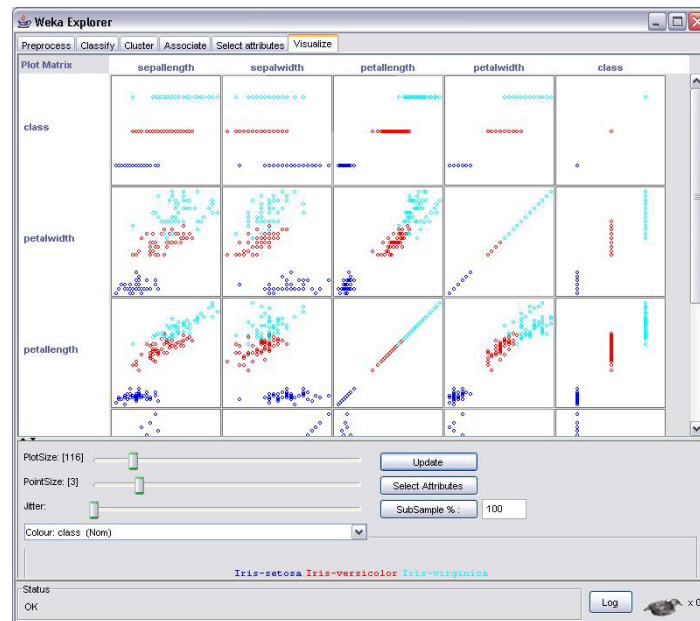


Figura 5. Visualización en Weka. Fuente: («Weka web», s. f.)

2.2.3. Orange

Orange³ es una herramienta visual de minería de datos y análisis de datos de código abierto que proporciona una amplia gama de algoritmos de aprendizaje automático y minería de datos para realizar análisis predictivos y descriptivos.

Orange utiliza una interfaz gráfica de usuario intuitiva que permite a los usuarios construir flujos de trabajo de datos mediante la combinación de widgets de análisis de datos predefinidos. Orange es muy popular entre los investigadores y profesionales de la Ciencia de Datos debido a su amplia gama de funcionalidades, su facilidad de uso y su capacidad para trabajar con una amplia variedad de formatos de datos.

Los algoritmos de reglas descriptivas supervisadas (SDRD) que se encuentran en Orange son los algoritmos CN2-SD y AprioriSD, los cuales han sido incluidos

³ <https://orangedatamining.com/>

mediante un plugin adicional creado por investigadores⁴. Y, a pesar de su versatilidad para el formato de los datos, Orange no permite el análisis de los datos, únicamente permite el análisis de las reglas obtenidas.

En cuanto a la visualización de los resultados en Orange, esta herramienta ofrece una amplia variedad de widgets de visualización que permiten a los usuarios explorar y visualizar los resultados de sus análisis de datos. Estos widgets incluyen gráficos de barras, gráficos de dispersión, diagramas de cajas y bigotes, mapas de calor, tablas y mucho más. Los usuarios pueden personalizar la visualización de los resultados para adaptarse a sus necesidades específicas, así como ajustar los colores, las etiquetas y las escalas de los ejes.

Además, Orange permite a los usuarios agregar y personalizar widgets de visualización utilizando el lenguaje de programación Python, lo que ofrece una gran flexibilidad para la visualización de los resultados y la creación de nuevos widgets. Sin embargo, esta característica presenta la limitación de que es necesario adaptar los algoritmos y la salida de los algoritmos a la arquitectura que indica Orange.



Figura 6. Icono de Orange. Fuente: (Orange web, s. f.)

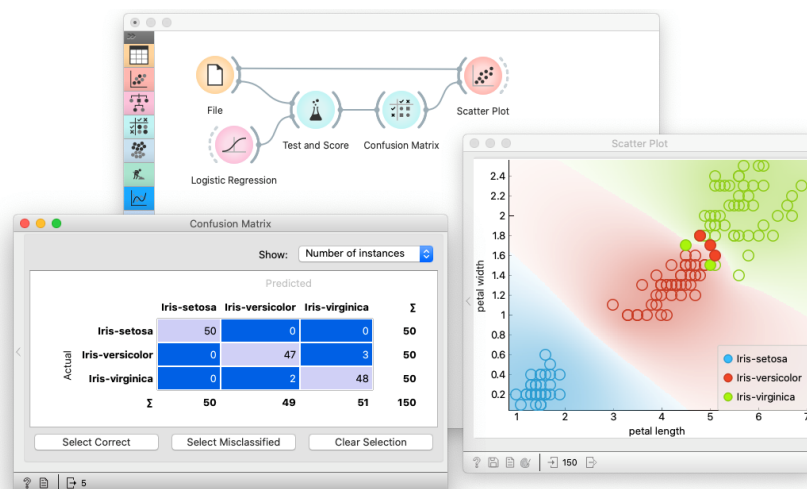


Figura 7. Visualización de la información en Orange. Fuente: (Orange web, s. f.)

⁴ https://kt.ijs.si/petra_kralj/SubgroupDiscovery/

Características	KEEL	Weka	Orange
Algoritmos de aprendizaje automático	Sí	Sí	Sí
Algoritmos de reglas descriptivas tipo SDRD	Sí	No	Sí
Herramientas de preprocesamiento de datos	Sí	Sí	Sí
Herramientas de selección de características	Sí	Sí	Sí
Herramientas de validación de modelos	Sí	Sí	Sí
Interfaz gráfica de usuario intuitiva	No	Sí	Sí
Amplia variedad de widgets de visualización	No	Sí	Sí
Permite personalizar visualizaciones y agregar widgets con Python	No	No	Sí

Tabla 3. Resumen de las características de las herramientas de difusión y uso de algoritmos SDRD

2.3. Herramientas para la creación de la aplicación

En este apartado se comentará brevemente diferentes herramientas que hay disponibles para llevar a cabo los objetivos planteados, así como, la herramienta que presenta características de mayor interés.

2.3.1. Librerías para la creación de interfaces gráficas

Existen diversas librerías en los diferentes lenguajes de programación que permiten el desarrollo de interfaces gráficas (GUI). Por ejemplo,

- Tkinter⁵ es una librería estándar de Python que proporciona una interfaz para la librería Tk, la cual es una herramienta de desarrollo de interfaces gráficas de usuario en Tcl (*Tool Command Language*). Al ser Python el lenguaje de programación de Tkinter, es compatible con una amplia variedad de librerías de Ciencia de Datos incluidas en Python para procesamiento y visualización de datos, tales como NumPy, Pandas, Matplotlib, Seaborn, entre otras.

Además, tiene el estilo nativo de los diferentes sistemas operativos, esto es, los componentes visuales se ajustan automáticamente a los estándares de diseño de cada sistema operativo en el que se ejecuta la aplicación, por lo que la aplicación tendrá una apariencia y comportamiento similar al de otras aplicaciones nativas del sistema operativo, lo que hace que los usuarios se sientan más cómodos y familiarizados con la aplicación y puedan utilizarla de manera más intuitiva.



Figura 8. Librería Tkinter. Fuente: (Vương Quý, 2022)

⁵ <https://docs.python.org/es/3/library/tkinter.html>

- Qt⁶: Es un *framework* multiplataforma de desarrollo de software cuya principal finalidad es la creación de interfaces de usuario, ofreciendo diferentes herramientas y personalización. Qt se programa principalmente en C o C++, aunque también puede ser utilizado en otros lenguajes como Python.



Figura 9. Icono Qt. Fuente: (Qt web, s. f.)

- JavaFX⁷: Es una librería multiplataforma, que se programa únicamente para Java. JavaFX Tiene soporte para hardware acelerado, lo que significa que puede aprovechar las capacidades de hardware del sistema para mejorar el rendimiento de la aplicación.



Figura 10. Icono JavaFX. Fuente: (JavaFX web, s. f.)

Finalmente, de estas tres opciones más comunes se decidió utilizar Tkinter porque:

- La herramienta Qt, a pesar de ser más completa, puede tener problemas de rendimiento, no hay tutoriales ni documentación clara y es necesaria su instalación para utilizar la aplicación. Además, ciertas funcionalidades están limitadas ya que tiene diferentes licencias y la gratuita solo permite realizar interfaces simples y básicas.

⁶ <https://www.qt.io/>

⁷ <https://openjfx.io/>

- La herramienta JavaFx puede presentar problemas de incompatibilidad entre las diferentes versiones de Java. Además, es obligatorio tener instalado Java para su uso ya que se ejecuta en la máquina virtual de Java.

Característica	Tkinter	Qt	JavaFX
Multiplataforma	Sí	Sí	Sí
Lenguajes de programación compatibles	Python	C++, Python, y otros	Java
Integración con otras librerías	Sí	Sí	Sí
Documentación clara y abundante	Sí	No	No
Fácil de usar y aprender	Sí	-	-
Requerimientos de instalación adicionales	No	Sí	Sí (Java)
Problemas de rendimiento	No	Posible	No
Soporte para hardware acelerado	No	Posible	Sí
Licencias limitantes	No	Posible	No
Estilo nativo en diferentes sistemas operativos	Sí	No	No

Por lo tanto, como la GUI a desarrollar es simple, las herramientas que ofrece Tkinter son suficientes y se complementan con las principales librerías de tratamiento de datos. También, al ser una librería incluida en la instalación base de Python, es *open source* y no es necesario instalar ningún software adicional.

2.3.2. Lenguaje de programación

Siguiendo el análisis anterior, se considera más relevante para desarrollar la aplicación emplear Tkinter, la cual utiliza el lenguaje de programación Python.

Python⁸ es un lenguaje de programación multiplataforma de alto nivel, interpretado, orientado a objetos y de propósito general. Tiene una sintaxis clara y legible que permite a los desarrolladores escribir código de forma más eficiente y fácil de entender. Además, es un lenguaje interpretado, es decir, no requiere de una compilación previa; y permite el tratamiento de ficheros y cadena de caracteres de manera muy sencilla, lo que lo hace más ideal para este proyecto. (*Python web*, s. f.)



Figura 11. Icono de Python. Fuente: (*Python web*, s. f.)

2.3.3. Bibliotecas utilizadas

Para el tratamiento de información las principales bibliotecas de Python son:

- Matplotlib⁹: Es una biblioteca de visualización de datos en Python que se utiliza para crear gráficos y visualizaciones en 2D y 3D. Ofrece una gran variedad de tipos de gráficos y permite personalizar gran cantidad de aspectos de éstos. Incluso, permite exportar los gráficos generados (*Matplotlib web*, s. f.).



Figura 12. Icono Matplotlib. Fuente: (*Matplotlib web*, s. f.)

⁸ <https://www.python.org/>

⁹ <https://matplotlib.org/>

- NumPy¹⁰: Es una biblioteca de Python empleada principalmente para el procesamiento numérico y científico de datos. Proporciona una amplia gama de funciones matemáticas y estadísticas para trabajar con arrays de manera sencilla y eficiente (*Numpy web*, s. f.).



Figura 13. Icono Numpy. Fuente: (*Numpy web*, s. f.)

- Re¹¹: Es una biblioteca de Python utilizada para realizar operaciones de expresiones regulares, es decir, permite encontrar coincidencias dentro de una cadena de texto, extraer información de una cadena de texto y, en general, operar con cadenas de caracteres fácilmente (*Re web*, s. f.).
- Zipfile¹²: La librería Zipfile en Python proporciona una manera de trabajar con archivos comprimidos en el formato ZIP. Permite crear, extraer, modificar y manipular archivos ZIP utilizando métodos simples y convenientes. Se pueden comprimir varios archivos y directorios en un único archivo ZIP, extraer archivos de un archivo ZIP existente, agregar archivos a un archivo ZIP y mucho más (*Zipfile web*, s. f.).

¹⁰ <https://numpy.org/>

¹¹ <https://docs.python.org/3/library/re.html>

¹² <https://docs.python.org/3/library/zipfile.html>

Capítulo III

ANÁLISIS Y DISEÑO

En esta sección, se llevará a cabo un análisis y una especificación exhaustiva de los requisitos que debe cumplir la aplicación, tanto funcionales como no funcionales. Se describirá detalladamente los casos de uso y los diagramas correspondientes de cada funcionalidad implementada en la aplicación. Además, se incluirá la arquitectura del sistema y se abordará el diseño de la interfaz de usuario, teniendo en cuenta los requisitos de usabilidad y accesibilidad.

Es importante tener en cuenta a lo largo de esta sección, que se ha diseñado la aplicación con un único rol de usuario: el Usuario. Este Usuario corresponde con un investigador experto capaz de interpretar los datos que se visualizan, aunque la interfaz es sencilla y accesible para cualquier persona sea experta o no.

Sin embargo, es relevante destacar que la información que se muestra en pantalla requiere de conocimientos especializados para su correcta comprensión. Por lo tanto, la aplicación se ha enfocado en proporcionar una experiencia de usuario óptima para este rol específico, con el objetivo de facilitar la tarea de análisis y toma de decisiones del investigador.

3.1. Requisitos funcionales y no funcionales

La principal finalidad de la aplicación es que el usuario pueda visualizar gráficamente el análisis de las reglas descriptivas supervisadas resultantes de la ejecución de un algoritmo SDRD sobre un conjunto de datos. De este objetivo principal surgen los siguientes requisitos funcionales que deben ser cubiertos correctamente en el proyecto:

1. REQF01: Visualizar gráficamente el análisis general de todas las reglas.
2. REQF02: Visualizar la tabla de contingencias de cada una de las reglas.
3. REQF03: Visualizar gráficos que muestren los resultados registrados en la tabla de contingencias.
4. REQF04: Visualizar la tabla de medidas (Confianza, WRAccN, TPr y FPr) de cada una de las reglas.
5. REQF05: Exportar a PDF la información mostrada en pantalla.
6. REQF06: Mostrar las reglas que cubren a cada dato.
7. REQF07: Mostrar los datos que cubre una regla.

Por otro lado, los requisitos no funcionales que deben cumplirse son:

1. REQNF01: Facilidad a la hora de cargar cualquier fichero de reglas y de conjunto de datos sin necesidad de realizar ninguna modificación para adaptarlos al formato de la aplicación.

2. REQNF02: Facilidad a la hora de instalar y utilizar la aplicación por parte de los usuarios finales.
3. REQNF03: Mostrar la información lo más clara posible para que resulte fácil de entender para el usuario.

3.2. Análisis de casos de uso

Tras la especificación de los requisitos principales que debe tener la aplicación, se pueden extraer los siguientes casos de uso con sus respectivos análisis textuales.

3.2.1. Sistema en general

Actor	Use case	Business
Soy	Quiero	Para
Usuario	Generar una nueva visualización	Visualizar gráficamente las reglas obtenidas
Usuario	Visualizar la información de cada una de las reglas	Ver los resultados obtenidos
Usuario	Visualizar resultados de todas las reglas	Ver los resultados globales de la ejecución
Usuario	Exportar la información mostrada en pantalla	Almacenar toda la información en un fichero PDF

Tabla 4. Análisis textual general

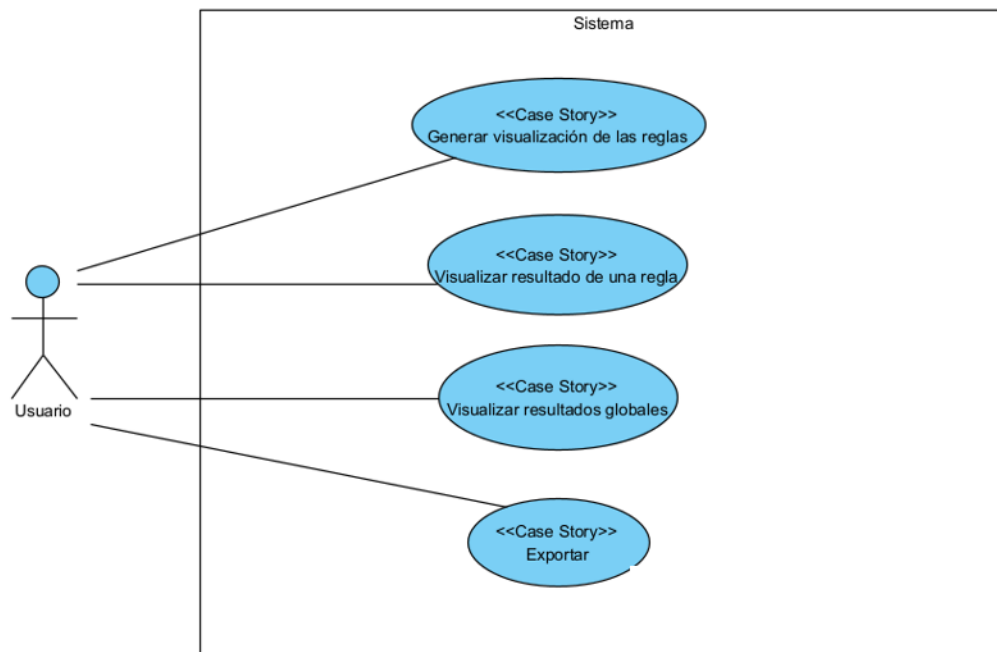


Figura 14. Casos de usos de la aplicación. Fuente: Elaboración propia

3.2.2. Generar visualización de las reglas

Caso de uso	Generar visualización de las reglas
Actor primario	Usuario
Sistema	Aplicación
Participantes	Usuario
Nivel	Objetivo de usuario
Precondición	Existencia del fichero de reglas a visualizar y el fichero del conjunto de datos
Flujo básico	
	1 Pulsar el botón "Nueva visualización".
	2 Seleccionar el fichero con el conjunto de reglas.
	3 Seleccionar el fichero con el conjunto de datos.
	4 Pulsar el botón "Visualizar"
Flujo alternativo	
	4.a Si no quiere visualizar los datos, cerrar aplicación

Tabla 5. Análisis textual de Generar visualización de las reglas

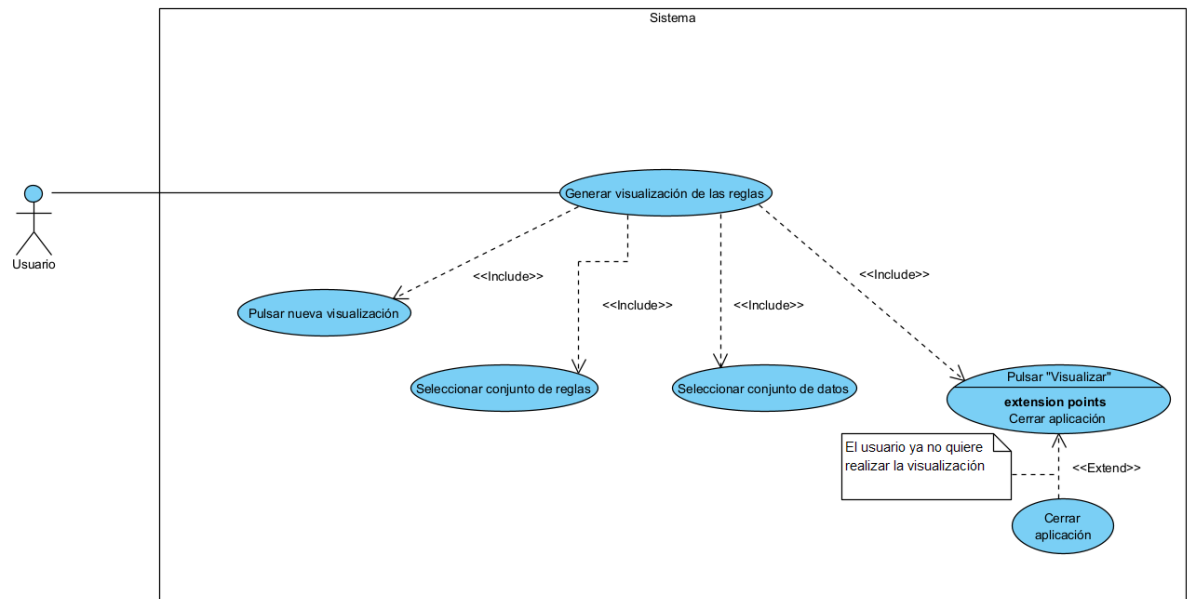


Figura 15. Diagrama de casos de uso de Generar visualización de las reglas. Fuente: Elaboración propia

3.2.3. Visualizar resultados globales

Caso de uso	Visualizar resultados globales
Actor primario	Usuario
Sistema	Aplicación
Participantes	Usuario
Nivel	Objetivo del usuario
Precondición	Selección del conjunto de reglas y del conjunto de datos
Flujo básico	
	1 Ver el solapamiento de reglas.
	2 Visualizar gráficas de los resultados de todas las reglas.
	3 Ver las reglas que cubre cada dato
	4 Atrás para seleccionar otros ficheros
Flujo alternativo	
	4.b Cerrar la aplicación

Tabla 6. Análisis textual de Visualizar resultados globales

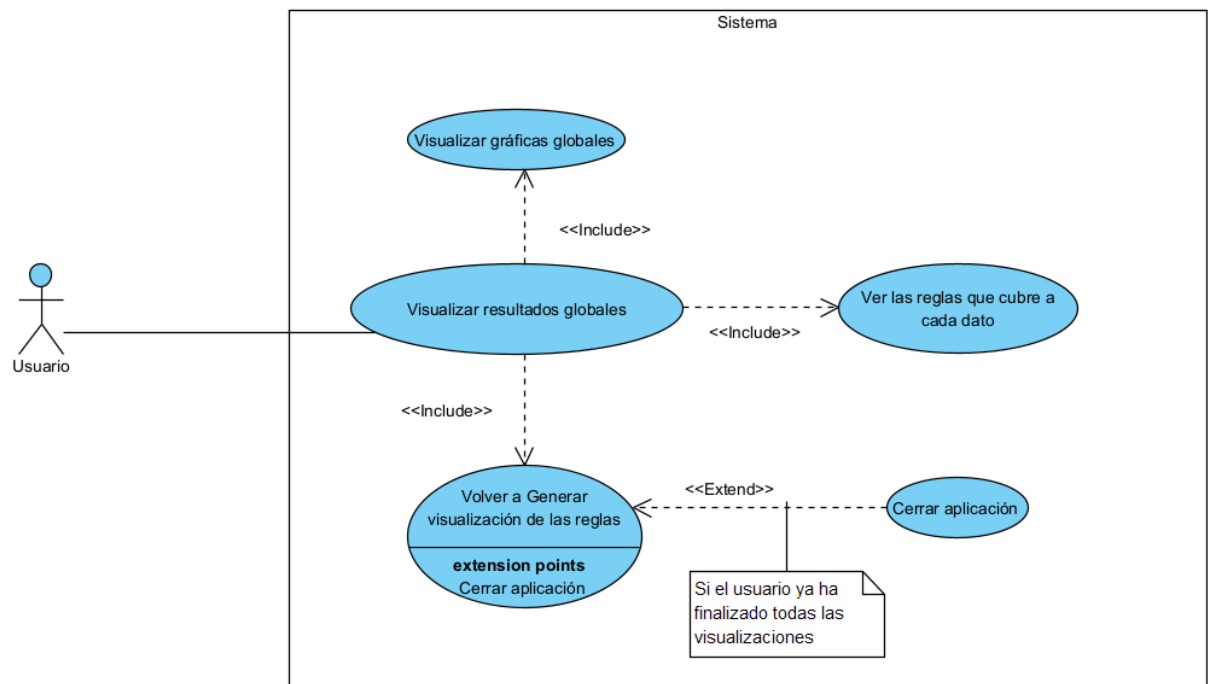


Figura 16. Diagrama de casos de uso de Visualizar resultados globales. Fuente: Elaboración propia

3.2.4. Visualizar resultados de una regla

Caso de uso	Visualizar resultado de una regla
Actor primario	Usuario
Sistema	Aplicación
Participantes	Usuario
Nivel	Objetivo de usuario
Precondición	Selección del conjunto de reglas y selección del conjunto de datos
Flujo básico	
	1 Seleccionar una regla.
	2 Visualizar la tabla de contingencias.
	3 Ver la representación gráfica.
	4 Ver los datos que cubre la regla
	5 Volver atrás.
Flujo alternativo	
	5.b Cerrar la aplicación

Tabla 7. Análisis textual de Visualizar resultado de una regla

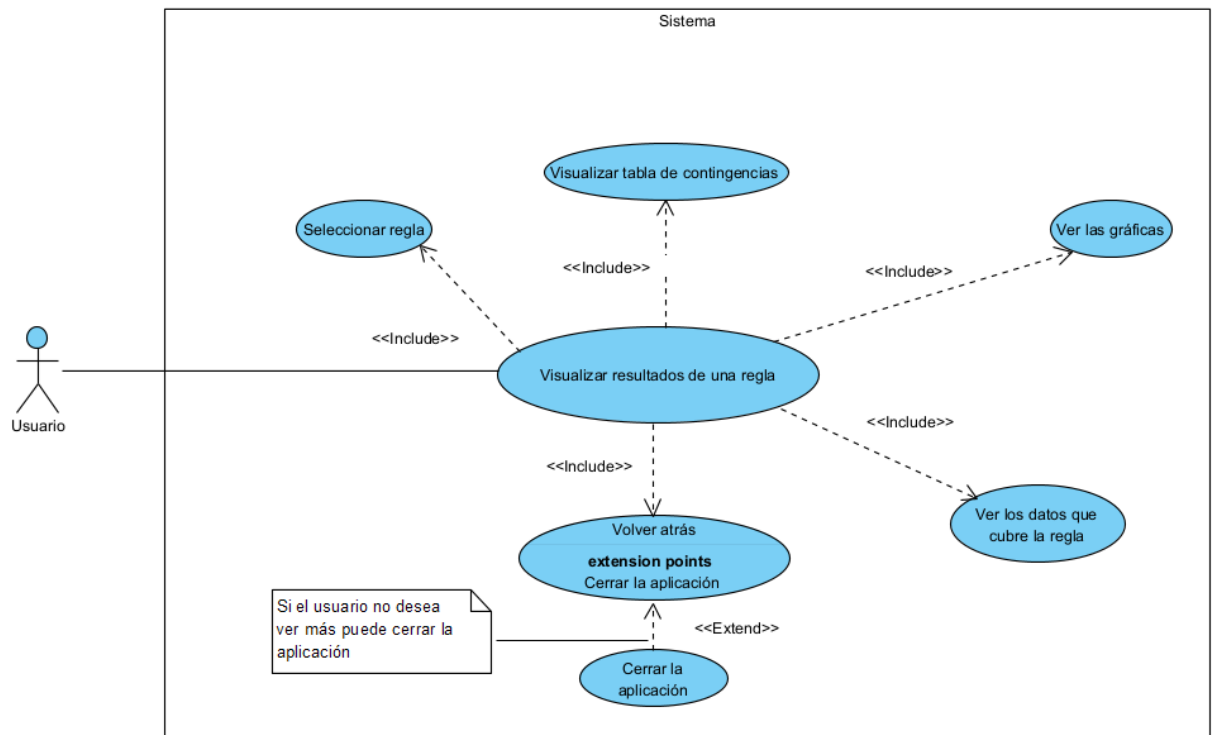


Figura 17. Diagrama de casos de uso de Visualizar resultados de una regla. Fuente: Elaboración propia

3.2.5. Exportar

Caso de uso	Exportar
Actor primario	Usuario
Sistema	Aplicación
Participantes	Usuario
Nivel	Objetivo del usuario
Precondición	Haber importado el dataset y el fichero de reglas
Flujo básico	
	1 Indicar el nombre del fichero
	2 Indicar la ubicación donde se va a guardar
	3 Realizar la exportación
Flujo alternativo	
	3.b Cancelar la exportación

Tabla 8. Análisis textual de Exportar

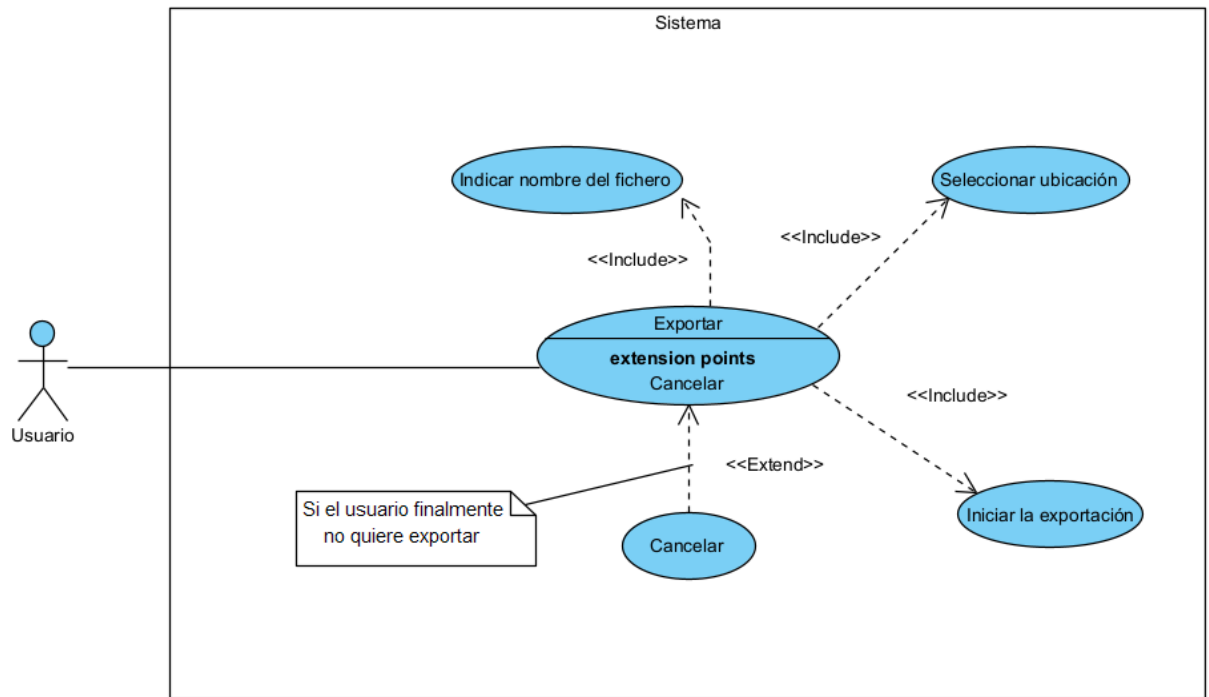


Figura 18. Diagrama de casos de uso de Exportar. Fuente: Elaboración propia

3.3. Diagramas de secuencia

A continuación, se muestran los diagramas de secuencia donde se puede ver la relación temporal y la interacción entre los elementos del sistema y el usuario.

3.3.1. Generar visualización de las reglas

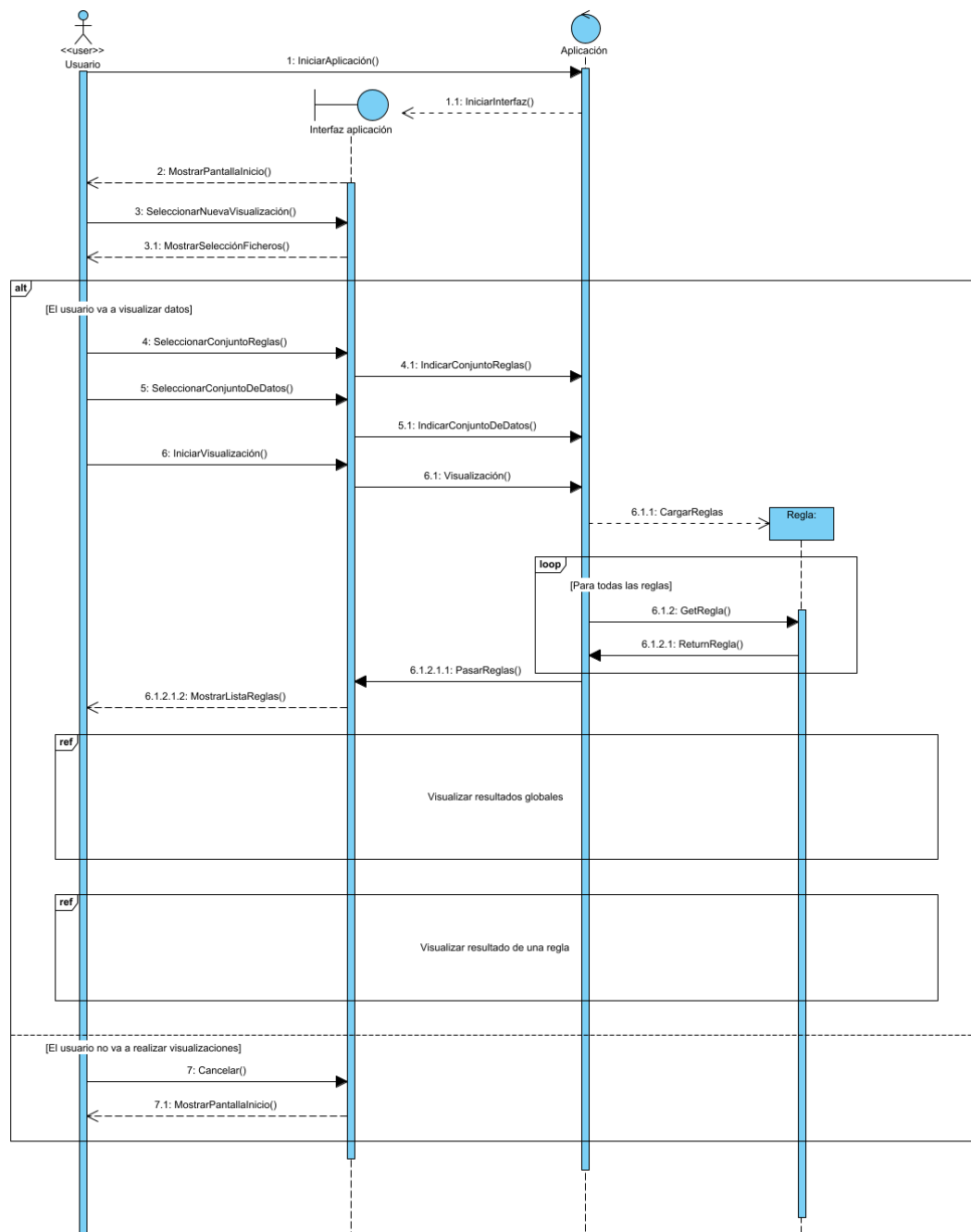


Figura 19. Diagrama de secuencia de Generar visualización de las reglas. Fuente: Elaboración propia

3.3.2. Visualizar resultados globales

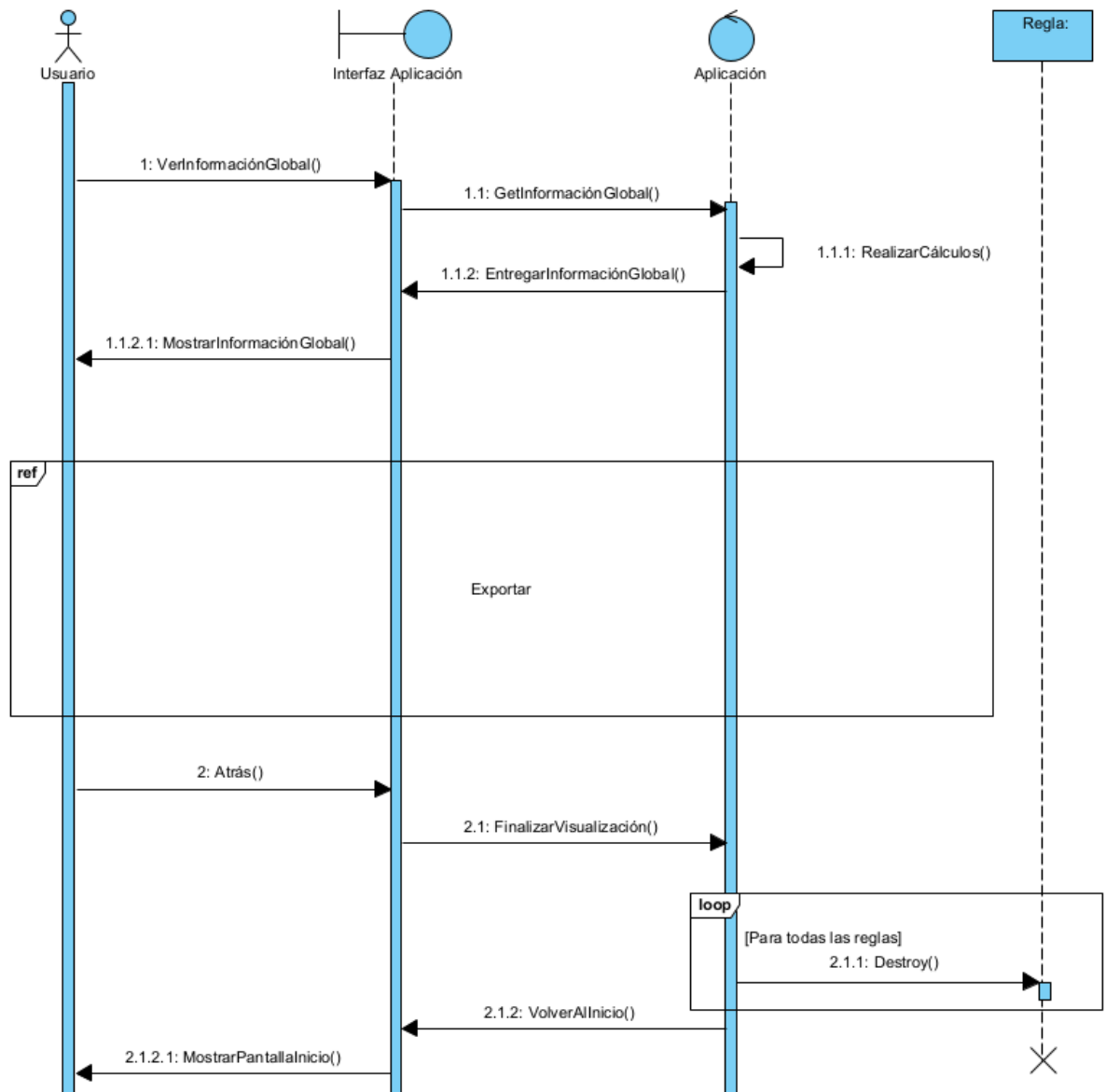


Figura 20. Diagrama de secuencia de Visualizar resultados globales. Fuente: Elaboración propia

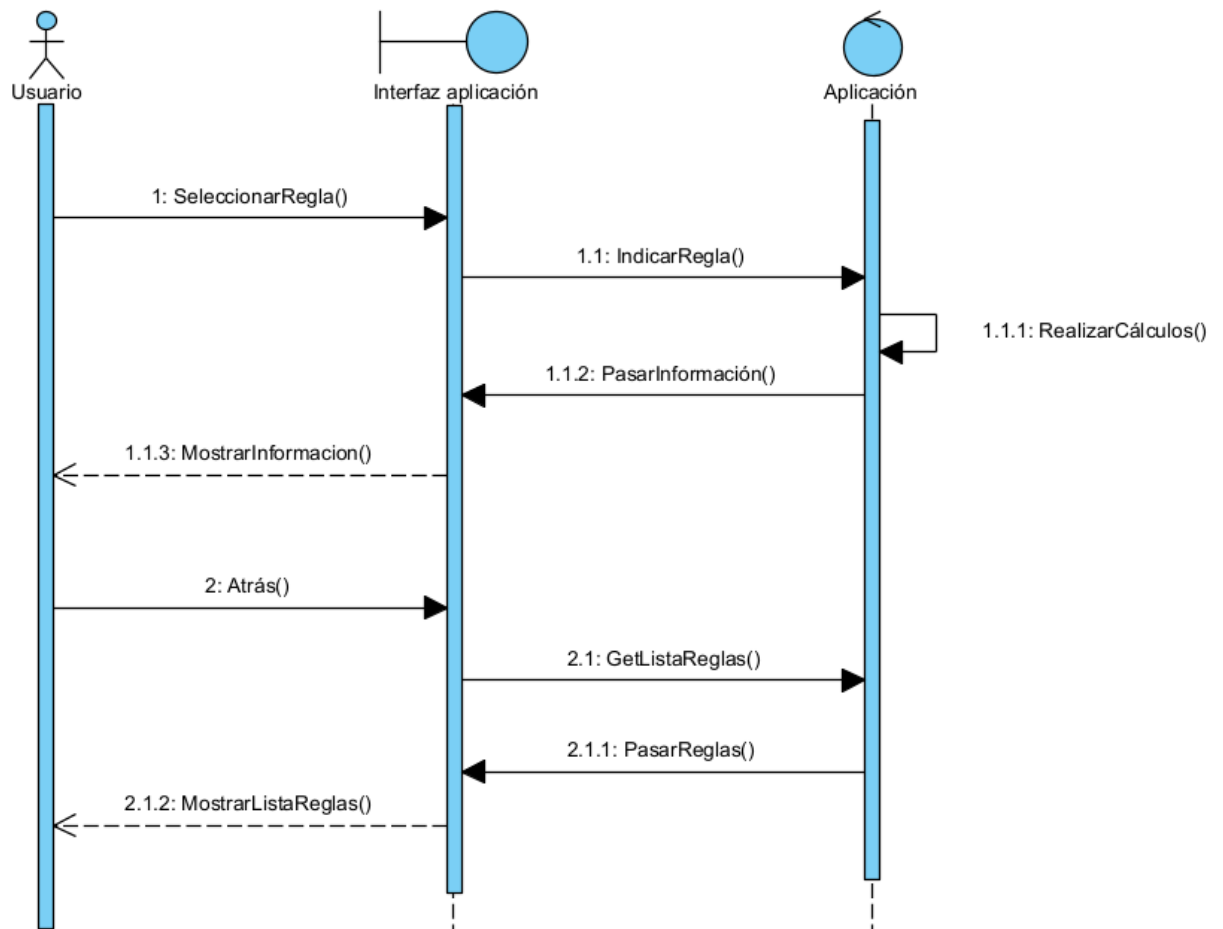
3.3.3. Visualizar resultados de una regla

Figura 21. Diagrama de secuencia de Visualizar resultados de una regla. Fuente: Elaboración propia

3.3.4. Exportar

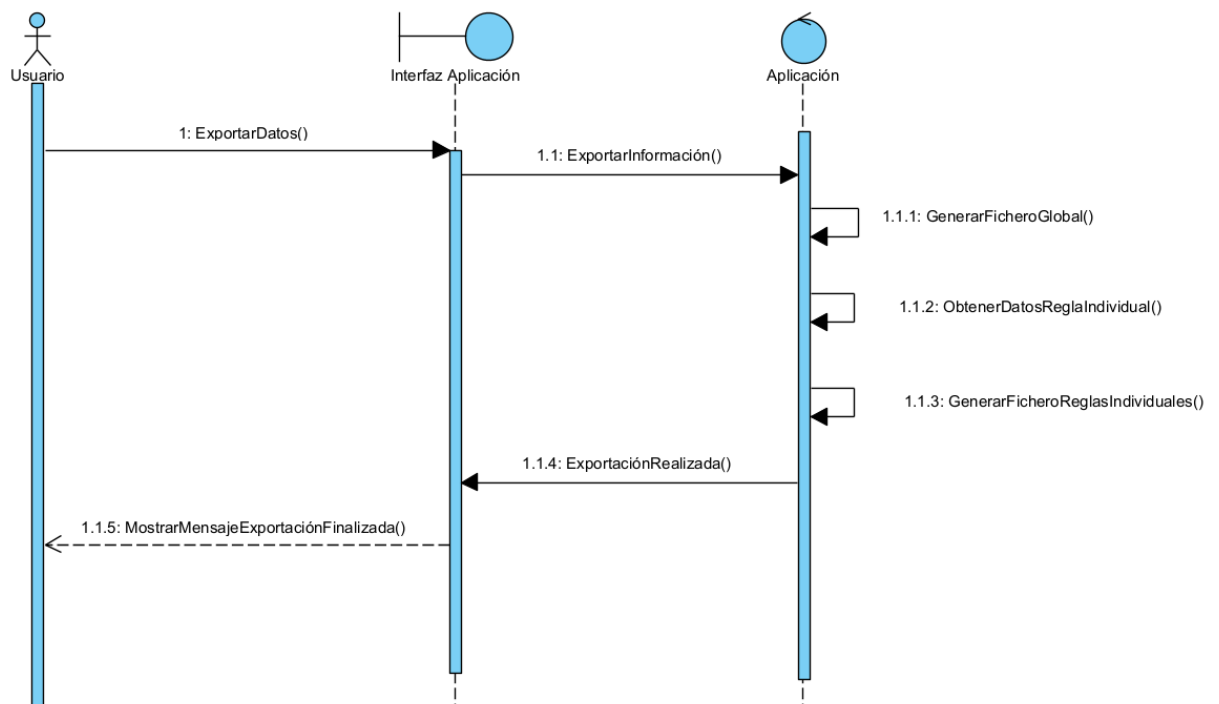


Figura 22. Diagrama de secuencia de Exportar. Fuente: Elaboración propia

3.4. Arquitectura del sistema

Finalmente, se incluye un esquema con la arquitectura del sistema con las entradas, salidas y operaciones que se realizan internamente.

Para utilizar el sistema, el usuario necesitará realizar dos pasos básicos: importar el fichero de datos y el fichero de reglas. El fichero de datos debe contener el conjunto de datos que se utilizará para la evaluación de las reglas. El usuario puede importar el fichero completo, que contendrá tanto el conjunto de entrenamiento como el conjunto de prueba, o bien importar el fichero de entrenamiento o el fichero de prueba.

Una vez importado el fichero de datos, el usuario deberá importar el fichero de reglas que se desea evaluar. Este fichero debe contener las reglas que se aplicarán al conjunto de datos para obtener los resultados deseados.

Es importante destacar que los formatos de los ficheros de datos y reglas deben ser compatibles con el utilizado por la aplicación para que funcione correctamente; y que es indiferente el orden de importación de los ficheros, es decir, el usuario puede importar primero el fichero de datos y luego el de reglas o viceversa.

A partir de esta información, el sistema internamente extrae los datos necesarios y realiza los cálculos correspondientes para evaluar las reglas. Una vez procesados los datos, el sistema genera una representación gráfica de los resultados para que el usuario pueda visualizarlos de forma clara. Si el usuario lo desea, también puede solicitar que el sistema genere un archivo de exportación que contenga la información mostrada en la aplicación para su posterior uso.

En resumen, el sistema se encarga internamente de todo el procesamiento de los datos, los cálculos y su visualización, permitiendo al usuario centrarse en la interpretación de los resultados y la toma de decisiones basadas en ellos.

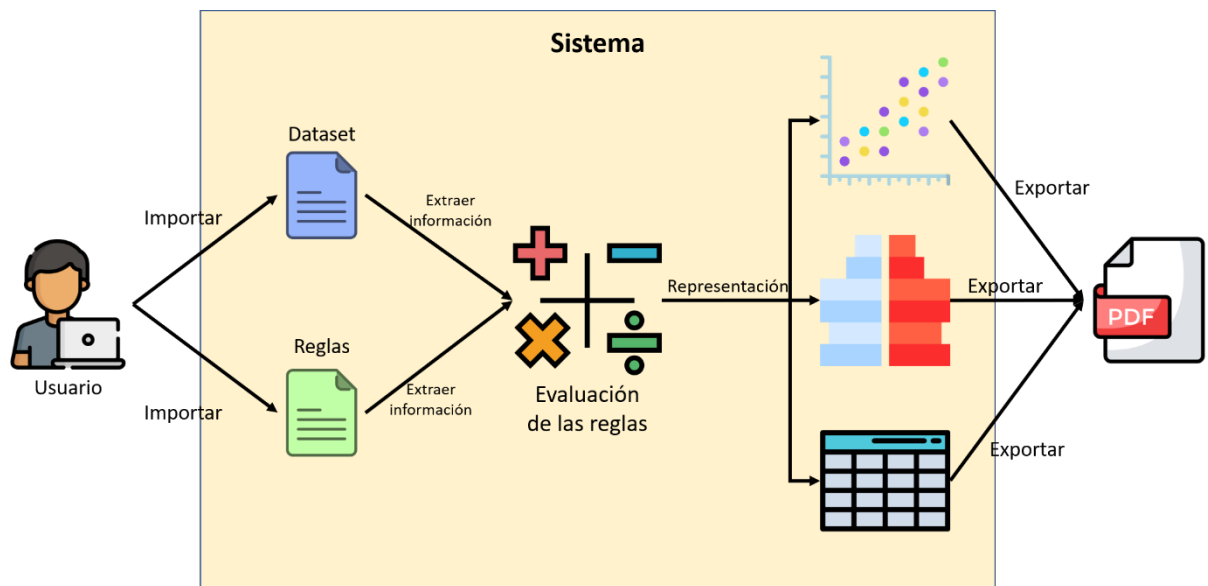


Figura 23. Arquitectura del sistema. Fuente: Elaboración propia

3.5. Diseño de la interfaz

En esta subsección se presentarán las vistas principales que deben estar incluidas en la aplicación para asegurar que se cumplen los requisitos especificados respetando los criterios de usabilidad. Además, se describirá el storyboard que servirá como base para la implementación de la aplicación. Este storyboard es esencial para visualizar cómo se van a conectar las diferentes vistas y cómo se va a desarrollar la navegación dentro de la aplicación.

3.5.1. Vistas necesarias

La aplicación de escritorio requerirá de varias vistas importantes que serán necesarias para brindar una experiencia de usuario satisfactoria. Estas vistas se presentan en la Tabla 9, donde se especifica el nombre de cada vista y su propósito en la aplicación.

Número de vista	Vista	Finalidad
1	Inicio	Presentar la aplicación y permitir que se inicie su uso.
2	Importar ficheros	Proporciona al usuario la capacidad de importar los dos ficheros indispensables para realizar los cálculos: el fichero de reglas y el fichero de dataset.
3	Información general de las reglas	Ofrece al usuario una evaluación general de las reglas y la relación con los datos, además de acceso a la información detallada de las reglas específicas y la posibilidad de exportación para guardar los resultados obtenidos.
4	Información específica de una regla	Proporciona al usuario una evaluación detallada de una regla en particular mostrando las medidas de calidad apoyándose de gráficas y tablas.

Tabla 9. Descripción de las vistas

3.5.2. Storyboard

A continuación, se incluyen los mockups que constituyen el storyboard de la aplicación, así como el flujo de interacción entre ellos.

Los números que se muestran en las figuras hacen referencia a la vista que se muestra cuando se pulsa el botón.

Vista 1) Inicio



Figura 24. Vista 1. Vista de la pantalla principal. Fuente: Elaboración propia

Esta es la vista inicial que el usuario visualizaría al ejecutar la aplicación.

Vista 2) Importar ficheros

Figura 25. Vista 2. Vista de la pantalla de importar ficheros. Fuente: Elaboración propia

En esta vista, cuando el usuario pulsa “Elija dataset” y “Elija reglas” se abrirá la ventana para que pueda navegar por los ficheros de su dispositivo y pueda seleccionar el fichero a cargar. El botón “Siguiente” estará inactivo hasta que no se hayan seleccionado los dos ficheros con formato válido.

Además, se contemplan errores relacionados con el formato de los ficheros. Si se produce un error se muestra una pantalla emergente como la siguiente indicando el error.



Figura 26. Vista de la ventana de error al importar ficheros. Fuente: Elaboración propia

Cuando el usuario cierre la ventana volverá a la vista anterior, pero con el botón “Siguiente” desactivado, pues los ficheros no son correctos.

Vista 3) Información general de las reglas

Visualizador de algoritmos SDRD

Selecciona una regla

Atrás 3.1 Info del dataset Exportar

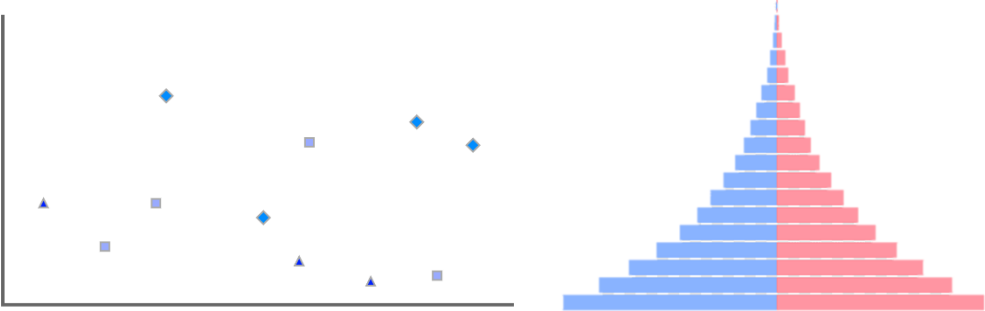
Rule 1: [...] 4

Rule 2: [...] 4

Rule 3: [...] 4

⋮

Rule n: [...] 4



Reglas que cubren a cada dato

Ejemplo 1: [...]	Rule 1, Rule 2	Rule 5
Ejemplo 2: [...]	Rule 2, Rule 3	Rule 1, Rule 6
Ejemplo n: [...]	Rule 5	Rule 8

Figura 27. Vista 3. Vista de la información general. Fuente: Elaboración propia

Cuando el usuario pulse el botón “Exportar” se abrirá la ventana para que pueda navegar por los ficheros de su dispositivo y seleccione el directorio donde va a almacenar la información y el nombre que le quiere poner.

Una vez finalice la exportación se mostrará durante unos segundos una ventana emergente que le indicará al usuario que ha finalizado la exportación y tiene disponible la información exportada.



Figura 28. Vista de la ventana emergente después de exportar. Fuente: Elaboración propia

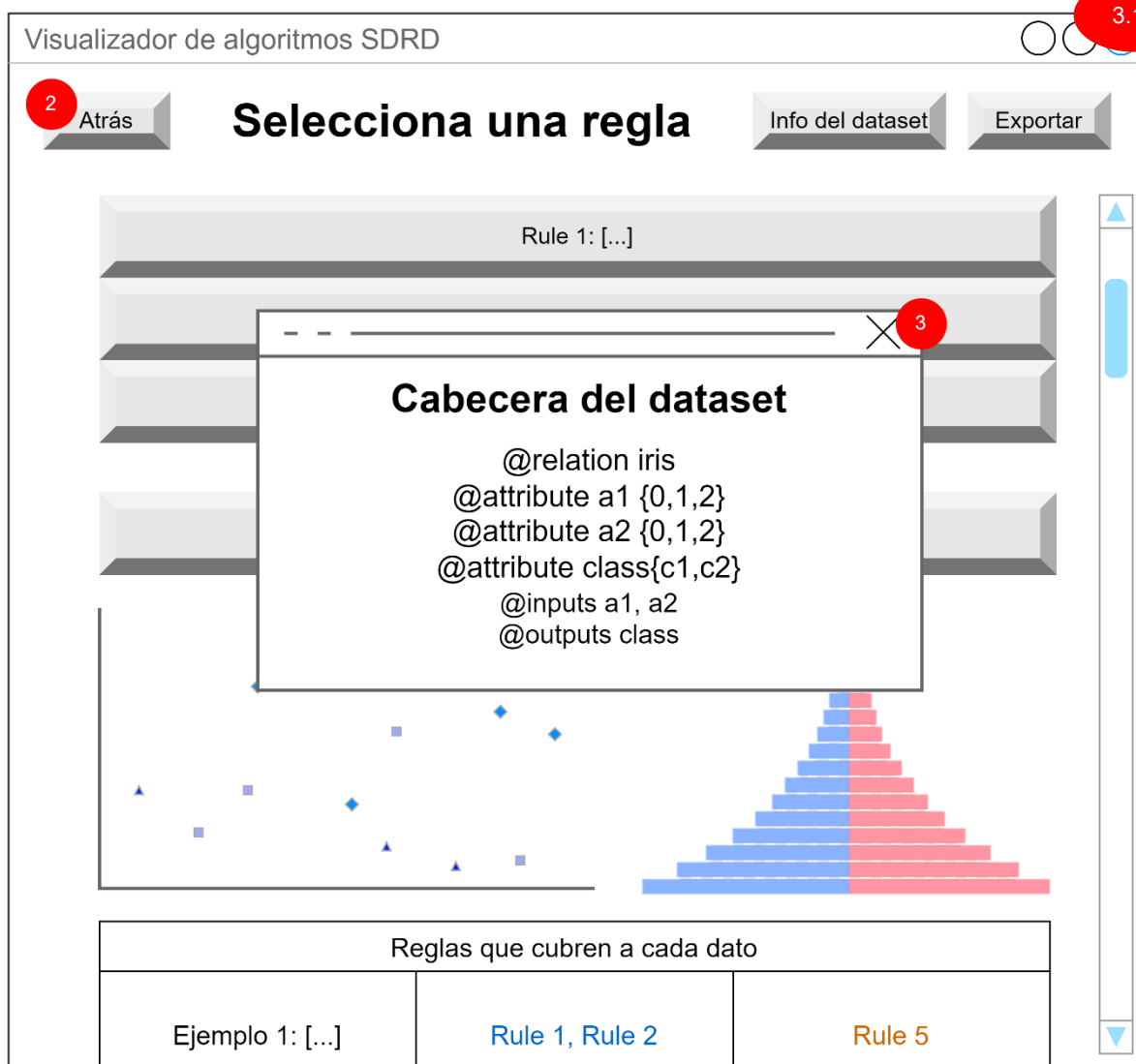
Vista 3.1.)

Figura 29. Vista 3.1. Vista de la ventana de la información del dataset. Fuente: Elaboración propia

Se muestra una ventana emergente con la cabecera del dataset cuando el usuario pulsa el botón "Info del dataset". Cuando la cierre volverá a la vista anterior.

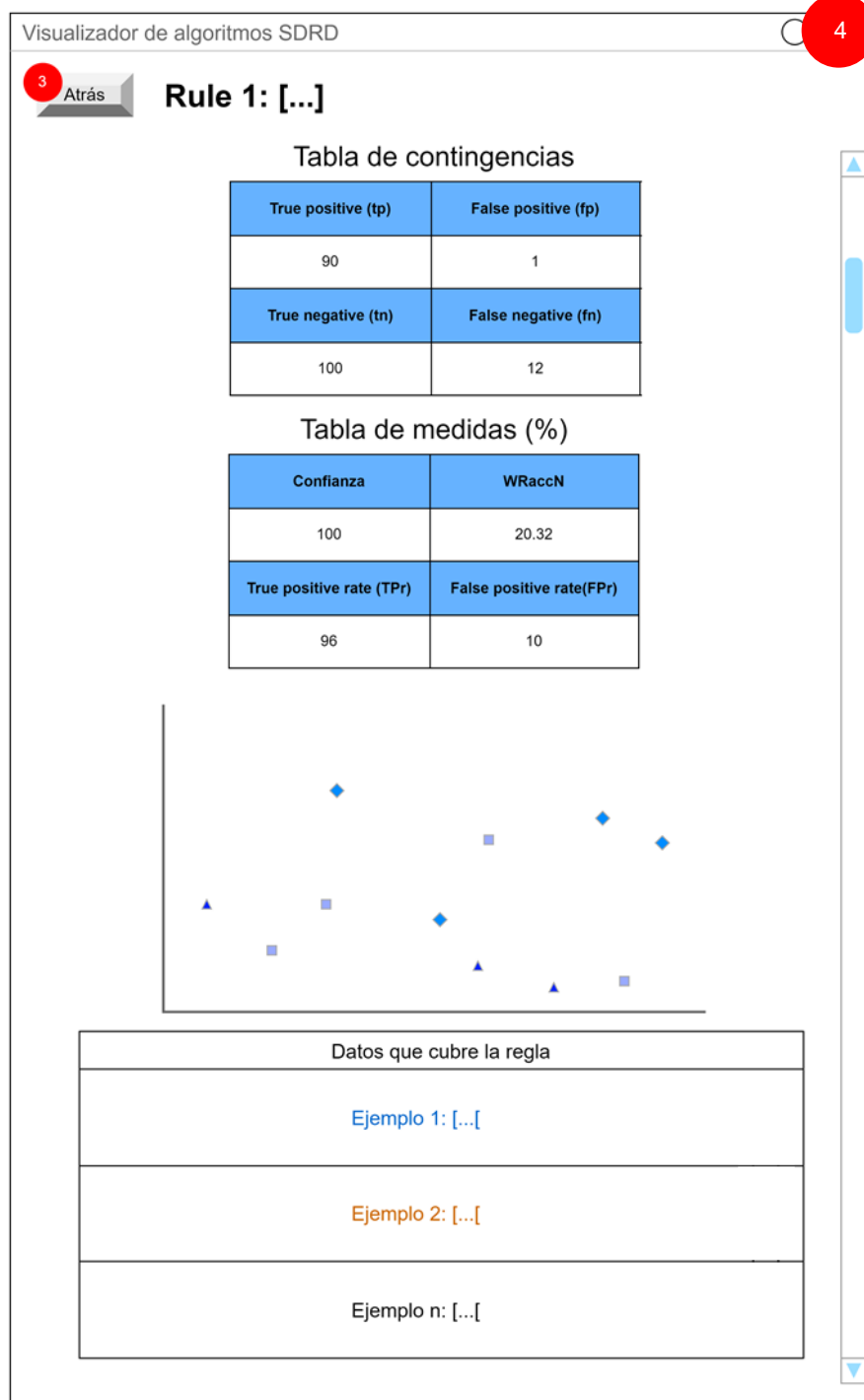
Vista 4) Información específica de una regla

Figura 30. Vista 4. Vista de una regla específica. Fuente: Elaboración propia

Capítulo IV

IMPLEMENTACIÓN

En este capítulo, se abordarán aspectos específicos relacionados con la implementación de la aplicación. Se detallará cómo se lograron implementar ciertos mecanismos concretos del sistema, y cómo se emplearon las diversas tecnologías mencionadas previamente en la sección 2.2 Herramientas para la creación de la aplicación.

4.1. Back-end

En esta sección se detallará la implementación del back-end de la aplicación, y se abordarán aspectos importantes que permitirán su correcto funcionamiento. Se describirá el proceso desde la ejecución de la aplicación hasta cómo se extrae la información de los archivos, teniendo en cuenta aspectos como la estructura y el formato de los mismos. Además, se explicará en detalle los cálculos que se realizan para evaluar las reglas, y cómo estos se integran en la aplicación. Así mismo, se abordará el proceso de exportación de la información, detallando los formatos de salida. Finalmente, se explicará cómo se realiza el despliegue de la aplicación.

4.1.1. Base de la aplicación

La base de la aplicación la componen dos ficheros: el fichero `main.py` y el fichero `Manager.py`.

main.py

Este fichero es muy simple y a la vez muy importante, ya que se encarga de lanzar la aplicación. A partir de éste se obtienen los ejecutables para utilizar la aplicación en diferentes sistemas operativos e iniciar la aplicación desde consola.

Básicamente, crea un objeto de tipo *Manager* e inicia la aplicación con la función de Tkinter dedicada a ello.

```
app = Manager()  
app.mainloop()
```

Código 1. Iniciación de la aplicación

Manager.py

Este archivo contiene la clase *Manager*. Es una clase hijo de la clase Tkinter y cumple un papel fundamental en la aplicación. Es la responsable de crear la interfaz principal, facilitar el intercambio de vistas y almacenar valores globales entre ellas.

Al heredar de la clase *tkinter.Tk*, el *Manager* obtiene todas las funcionalidades necesarias para crear y gestionar la interfaz gráfica de usuario. Es decir, se pueden utilizar todos los elementos y métodos proporcionados por Tkinter para diseñar y controlar la apariencia y el comportamiento de la interfaz.

Una de las principales tareas del *Manager* es crear el contenedor principal de la interfaz. Este contenedor se utiliza como un marco sobre el que se colocarán las diferentes vistas o pantallas de la aplicación. Éste se crea y se configura en el constructor de la clase.

Además, el *Manager* cuenta con un diccionario de clases, que se emplea para gestionar las diferentes vistas de la aplicación. Cada vista corresponde con una clase que hereda de la clase *Frame* y debe ser añadida a mano para que se cree.

Al inicio del programa, se crea cada una de las vistas y se almacenan en el array. Este enfoque permite una gestión dinámica y eficiente de las vistas, ya que facilita el intercambio de vistas según el flujo de interacción del usuario y solo existe una instancia de cada vista durante la ejecución de la aplicación que se actualiza cuando se produce un cambio en ella.

```
1         self.frames = {} #Diccionario de clases
2         for F in (Home, Importar, VisualizarReglas,
3                   VisualizarInfoRegla):#!Modificar aquí para añadir vistas
4             frame = F(self.container, self)
5             self.frames[F] = frame
6             frame.grid(row=0, column=0, sticky=tk.NSEW)
```

Código 2. Creación de las pantallas de la aplicación

Cuando se realiza un cambio de vista, el *Manager* recupera el objeto correspondiente de la vista deseada a partir del vector y lo muestra en el contenedor principal de la interfaz.

```
1 def show_frame(self, container, hecho):
2     frame = self.frames[container]
3     if(not hecho):
4         frame.init_widgets()
5     frame.tkraise()
```

Código 3. Cambio de pantallas en la aplicación

Además de gestionar el intercambio de vistas, el *Manager* también es responsable de almacenar valores globales que pueden ser compartidos entre diferentes vistas. Esto permite mantener la coherencia y la persistencia de datos importantes a lo largo de la aplicación. Las variables son:

- Reglas: Array que contendrá todas las reglas extraídas del fichero de reglas.
- Dataset: Es un objeto que incluye toda la información referente al dataset que ha importado el usuario.
- ReglaSeleccionada: Regla que el usuario ha seleccionado para ver su información más detallada.

En resumen, la clase *Manager* desempeña un papel crucial en la implementación de la aplicación, ya que, gracias a su funcionalidad, permite diseñar una interfaz intuitiva y fluida, ofreciendo una experiencia de usuario agradable y coherente y permitiendo añadir nuevas vistas a la aplicación de manera sencilla.

4.1.2. Importación de los ficheros

La funcionalidad principal de esta aplicación radica en la importación de ficheros, la cual desempeña un papel crucial en su funcionamiento, ya que los ficheros que aportará el usuario son imprescindibles para la evaluación de las reglas y su respectiva representación.

Es importante destacar que el orden en que el usuario seleccione los ficheros no afecta al funcionamiento de la aplicación, pero sí es obligatorio que se aporten ambos porque de otra manera el botón de “Siguiente” no se activará para pasar a la siguiente vista. Los ficheros necesarios son:

- **Dataset**

El fichero del conjunto de datos debe tener el formato de la aplicación KEEL para ser aceptado y leído correctamente por la aplicación. El formato es el siguiente:

```
@relation iris
@attribute sepallength {0,1,2}
@attribute sepalwidth {0,1}
@attribute petallength {0,1,2}
@attribute petalwidth {0,1,2}
@attribute class{Iris-setosa,Iris-versicolor,Iris-virginica}
@inputs sepallength,sepalwidth,petallength,petalwidth
@outputs class
@data
0,1,0,0,Iris-setosa
0,0,0,0,Iris-setosa
```

Figura 31. Cabecera de un fichero de dataset. Fuente: Elaboración propia

donde:

- @relation: Hace referencia al nombre del dataset.
- @attribute: Se refiere al nombre del atributo y los posibles valores que puede tener. El último de la lista alude a las distintas clases que puede ser cada uno de los datos.
- @inputs: Indica los atributos y en qué orden se les da valor en los datos.
- @outputs: El resultado de que un dato tenga asignados ciertos valores a los atributos, concretamente es una clase.
- @data: Indica que a partir de esta línea se encuentran los datos.

Si el formato del fichero no es correcto, la aplicación le mostrará al usuario un mensaje error de que el formato es erróneo y no le permitirá avanzar a la siguiente pantalla.

• Reglas

El fichero de reglas debe indicar qué algoritmo se utilizó para generarlo, ya que cada algoritmo presenta la salida de manera diferente, por lo que el fichero de reglas debe ser tratado de forma específica según el algoritmo que ha obtenido las reglas.

```
@algorithm apriori
Rule 1: IF petalWidth = 0 THEN class -> Iris-setosa --
Support: 45 [ 45 0 0]
```

Figura 32. Cabecera de un fichero de reglas. Fuente: Elaboración propia

donde en @algorithm se debe indicar el algoritmo.

Si el algoritmo es capaz de trabajar con un conjunto de datos que no está discretizado, también es necesario indicar en el fichero de reglas cuántas etiquetas ha generado.

```
@algorithm nmeef
Number of labels: 3
GENERATED RULE 0
Antecedent
    Variable petalLength = Label 0 (-1.95 1.0 3.95)
Consequent: Iris-setosa
```

Figura 33. Cabecera de un archivo de reglas difusas. Fuente: Elaboración propia

Con el objetivo de solucionar la problemática derivada de los diversos formatos utilizados por los diferentes algoritmos y que además fuera sencillo añadir nuevos algoritmos sin modificar en exceso el código original, se ha implementado el patrón Strategy en la aplicación.

El patrón Strategy permite encapsular diferentes funcionalidades en clases separadas, promoviendo la flexibilidad, la reutilización y la fácil sustitución de las funcionalidades en tiempo de ejecución. Consta de tres elementos principales:

- El contexto es el objeto que utiliza las estrategias.
- Las estrategias son las clases que encapsulan cada una de las funcionalidades, las cuales se deben implementar en una interfaz común.
- La interfaz común define los métodos que deben implementar todas las estrategias.

Este patrón se ha aplicado en la aplicación de manera que:

- El contexto es la clase que se encarga de crear la vista de importar la información (*Importar*), ya que en ella también se hace la gestión y comprobaciones del formato de ficheros.
- Las estrategias son las diferentes clases que hacen referencia a los algoritmos cuyos resultados, las reglas, es capaz de procesar la aplicación.
- La interfaz común corresponde con una clase denominada *lecturaFicheroReglas*. Está compuesta por un solo método *lecturaFichero* que recibe como parámetro el nombre del fichero de reglas y la información obtenida del fichero del dataset.

Esta estructura aporta modularidad a la aplicación, facilitando la expansión de ésta, ya que permite añadir y extraer los resultados de nuevos algoritmos que no estén incluidos en la aplicación de forma sencilla, eficiente y sin tener que modificar prácticamente nada del código original.

La lectura de ficheros de reglas que hay implementados en la aplicación actualmente son los algoritmos:

- APRIORI (Kavšek & Lavrač, 2006),
- SD-MAP (Atzmueller & Puppe, 2006),
- SD (Gamberger & Lavrac, 2002),
- CN2-SD (Lavrac et al., 2004),
- Algoritmos evolutivos:
 - MESDIF (Berlanga et al., 2006),
 - NMEEF (Carmona et al., 2010),
 - SDIGA (Jesus et al., 2007)).

Los tres algoritmos evolutivos están implementados en una misma clase, pues el fichero de reglas que generan tiene el mismo formato.

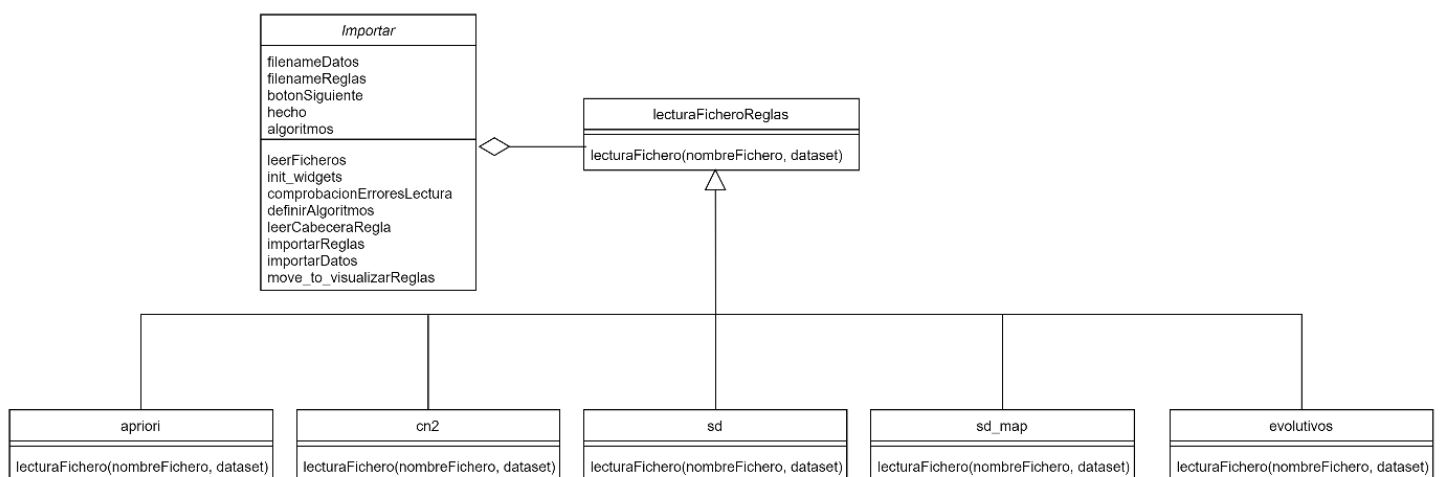


Figura 34. Diagrama UML del patrón Strategy aplicado en el proyecto. Fuente: Elaboración propia

Para la inclusión de un nuevo algoritmo es necesario realizar modificaciones en los ficheros:

- constantes.py: Aquí se debe clasificar el algoritmo entre algoritmos que necesitan tener el conjunto de datos discretizado (ALGORITMOS_NO_CONTINUOS) y los que no (ALGORITMOS_EVOLUTIVOS). Posteriormente, comentaré por qué es necesaria esta distinción.
- importar_info:
 - En el método *definirAlgoritmos*. Aquí se crea el diccionario de clases donde están recogidas las clases que representan a los algoritmos.

- En el método *leerFicheros* para llamar el método *lecturaFichero* del algoritmo concreto que esté indicado en el fichero de reglas.

4.1.2.1. Detección de errores

Una vez que el usuario ha importado ambos ficheros, la aplicación, antes de procesar la información y hacer los cálculos, se encarga de comprobar que el formato de ambos ficheros es correcto y que es posible realizar los cálculos.

Si se produce un error la aplicación mostrará una ventana emergente indicando el error e impedirá al usuario pasar a la vista con la evaluación de las reglas.

Los errores que se han tenido en cuenta y cómo se detectan son:

Código error	Mensaje	Error	Detección
error01	El fichero de reglas no es correcto	Formato del fichero de reglas erróneo	En el fichero de reglas no aparece en todo el texto la palabra clave Rule en todas sus formas (rule o Rule o RULE).
error02	El formato del contenido del archivo es incorrecto	Formato del fichero del conjunto de datos erróneo	En el archivo del dataset no aparece la palabra clave @data. Sin esta expresión la aplicación es incapaz de detectar donde termina la descripción del dataset y donde empiezan los datos
error03	El algoritmo no está registrado en la aplicación	Algoritmo no registrado en la aplicación	Cuando el algoritmo indicado en el fichero de reglas tras la expresión @algorithm no está registrado en el array constante <i>ALGORITMOS_VALIDOS</i> del fichero <i>constantes.py</i>

error04	El algoritmo necesita el conjunto de datos discretizado	El algoritmo no funciona con datos cuyos valores de los atributos se encuentran en el entorno de los números reales y se ha importado un dataset con el conjunto de datos sin discretizar	El fichero de datos indica en la cabecera que el valor del atributo está dentro de un intervalo (Figura 38) y el algoritmo está registrado en la aplicación como <i>ALGORITMO_NO_CONTINUO</i> en el fichero de <i>constantes.py</i>
warning01	El conjunto de datos está discretizado y el algoritmo no lo necesita	El algoritmo funciona con datos cuyos valores de los atributos se encuentran en el entorno de los números reales y se ha importado un conjunto de datos discretizado. Se detecta como error porque el algoritmo define sus propias etiquetas basándose en los intervalos indicados en el dataset, por lo tanto, las reglas se obtienen en base a esas etiquetas. En cambio, el dataset discretizado da directamente las etiquetas y no los intervalos.	Se detecta cuando el conjunto de datos tiene definido los distintos valores que puede tomar cada atributo en una lista de etiquetas y no en un intervalo de números reales o enteros, y el fichero de reglas indica cuántas etiquetas ha generado para obtener las reglas

Tabla 10. Tabla de los posibles errores que se pueden producir al importar un fichero

```
@relation iris  
@attribute sepalLength real [4.3, 7.9]  
@attribute sepalWidth real [2.0, 4.4]
```

Figura 35. Cabecera de un dataset no discretizado. Fuente: Elaboración propia

4.1.2.2. Información extraída de los ficheros

A continuación, se detalla qué información se registra de cada uno de los ficheros aportados por el usuario. Este proceso se lleva a cabo una vez se ha verificado que los ficheros cuentan con el formato adecuado.

- **Dataset**

- La cabecera del fichero tal cual viene indicada en el fichero para extraer la siguiente información:
 - Los atributos en el orden que están indicados en los datos.
 - Las diferentes clases en las que se clasifican los datos.
 - En el caso de dataset no discretizado, se extrae por cada atributo el intervalo de valores que puede tomar ese atributo.
- Cada uno de los datos con los valores de los atributos asignado en el orden indicado en la cabecera y la clase que corresponde a ese dato.

- **Reglas**

- El algoritmo que ha generado la regla.
- En el caso de un algoritmo difuso, el número de etiquetas en el que ha descompuesto los intervalos continuos.
- El nombre de cada una de las reglas del cual se extrae:
 - Los atributos a los que les asigna un valor.
 - El valor que les asigna a dichos atributos.
 - La clase que corresponde según los valores asignados.

4.1.3. Evaluación de las reglas

Una vez que se ha obtenido la información requerida de los archivos, se realiza el proceso de evaluación de las reglas. A diferencia de la lectura de archivos de reglas, solo se realizan dos evaluaciones distintas: una para el conjunto de datos continuos, es decir, reglas difusas; y otra para aquellos que no son difusos.

La principal diferencia entre estos dos tipos de evaluaciones radica en la forma en que se abordan las reglas difusas.

4.1.3.1. Evaluación general de las reglas

Ambas evaluaciones (reglas difusas y no difusas) son bastantes similares. Para cada regla, se extraen los atributos a los que les ha asignado un valor y el valor asignado. Luego, se revisa cada dato para calcular la tabla de contingencias de la regla.

La tabla de contingencias se calcula de la siguiente manera:

- Se considera un verdadero positivo (tp) cuando la regla ha asignado correctamente las etiquetas de todos los atributos definidos, según el dato con el que se está comparando, y la clase que indica la regla coincide con la clase que indica el dato. Se registra que el dato está bien cubierto.
- En el caso anterior, si la clase no coincide con la del dato, se considera un falso positivo (fp) y se registra como un dato mal cubierto.
- Si la regla no ha asignado correctamente una etiqueta, pero tiene la misma clase que el dato, se considera un falso negativo (fn).
- En el caso anterior, si la clase es distinta a la del dato, se considera un verdadero negativo (tn).

Una vez que se ha calculado la tabla de contingencias, se llama a un método de la regla que se encarga de calcular la tabla de valores (TPr, FPr, Confianza y WRAccN), de acuerdo con lo que se ha comentado previamente en Medidas de calidad de las técnicas de SDRD.

Este proceso permite evaluar la efectividad y precisión de las reglas generadas, así como su capacidad para clasificar adecuadamente los datos según los atributos y las clases definidas.

Una vez concluido este procedimiento, se cuenta con todos los elementos necesarios para mostrar la información relativa al conjunto de datos y la evaluación de las reglas de una manera atractiva y fácilmente comprensible para el usuario mediante gráficos y tablas.

4.1.3.2. *Evaluación reglas difusas*

En el caso de las reglas difusas, hay que realizar un paso previo antes de calcular la tabla de contingencias y la tabla de medidas. Este paso es la división en subintervalos de los intervalos continuos de cada atributo, de manera que habrá tantos subintervalos como etiquetas estén indicadas en el fichero de reglas. A partir de esto, según el valor que tenga cada dato en ese atributo se le asigna la etiqueta correspondiente.

```

1 def dividirIntervalos(self, intervalos):
2     for interval in self.dataset.intervalos:
3         interval = interval.replace('[', '').replace(']', '').split(',')
4         inicio = float(interval[0])
5         fin = float(interval[1])
6         intervalo = np.linspace(inicio, fin, self.numEtiquetas)

```

Código 4. Subdivisión del intervalo en intervalos más pequeños

La subdivisión en intervalos la realiza una función de la librería de Numpy, la cual hay que indicarle el inicio del intervalo, el final del intervalo y el número de subdivisiones.

```

1 def funcionPertenencia(self, intervalo, valor):
2     etiqueta = -1
3     pertenencia = -1.0
4     y = 1.0
5     calculo = -1.0
6     valor = float(valor)
7     y = float(y)
8     for i in range(self.numEtiquetas):
9         x0 = float(intervalo[i])
10        x1 = float(intervalo[i+1])
11        x2 = float(intervalo[i+2])
12        if (valor < x1):
13            calculo = ((valor-x0)*(y/(x1-x0)))
14        if (valor > x1):
15            calculo = ((x2-valor)*(y/(x2-x1)))
16        if (calculo > pertenencia):
17            pertenencia = calculo
18            etiqueta = i
19            calculo = -1.0
20
21        return etiqueta, pertenencia

```

Código 5. Cálculo del valor de pertenencia y etiqueta de un atributo

La asignación de la etiqueta se realiza calculando el grado de pertenencia del valor del dato respecto a cada etiqueta lingüística. La función itera sobre cada etiqueta y calcula el valor de pertenencia utilizando las fórmulas de interpolación lineal, ya que se emplean etiquetas lingüísticas triangulares uniformes. Si el valor de pertenencia calculado es mayor que el valor de pertenencia anterior, se actualiza el valor de pertenencia y la etiqueta. Al final, la función devuelve la etiqueta y el valor con el mayor valor de pertenencia.

Finalmente, para el cálculo del grado de pertenencia de la regla para un dato concreto se emplea el operador T-norma, ya que se está empleando una representación canónica de las reglas. En resumen, el grado de pertenencia de la regla se corresponde con el mínimo de los grados de pertenencia calculados de todas las variables activas de la regla.

4.1.4. Exportación de la información

La exportación de ficheros se encargará de almacenar la información que el usuario visualiza en la aplicación de la siguiente manera:

1. Imagen en formato SVG del gráfico de puntos: Esta imagen representa los valores TPr (Tasa de Verdaderos Positivos) y FPr (Tasa de Falsos Positivos) de todas las reglas. Se ha elegido el formato SVG debido a que las imágenes se almacenan sin compresión, lo que permite al usuario utilizarlas en cualquier contexto sin perder calidad. Nombre del fichero: graficoPuntos.svg.
2. Imagen en formato SVG del gráfico de pirámide: Esta imagen representa los valores TPr y FPr de todas las reglas utilizando un gráfico de pirámide. Al igual que la imagen anterior, se utiliza el formato SVG para preservar la calidad y flexibilidad de uso. Nombre del fichero: graficoPiramide.svg
3. Fichero LaTeX con información general de las reglas: Este fichero almacena la información general sobre las reglas. Incluye los gráficos mencionados anteriormente, información del dataset y una tabla con todos los datos, donde se indica el número de ejemplo, los primeros atributos, la clase, las reglas que lo cubren correctamente (con su

respectivo grado de pertenencia en reglas difusas) y las reglas que no lo cubren adecuadamente. Nombre del fichero: `infoGeneral.tex`

4. Fichero LaTeX con información detallada de cada regla: Este fichero contiene información específica de cada una de las reglas. Incluye el nombre de la regla, tablas de contingencias, tablas de medidas, así como la tabla de los datos que la regla cubre correctamente y aquellos que no cubre adecuadamente. Nombre del fichero: `infoReglas.tex`

La elección del formato SVG para las imágenes y LaTeX para los ficheros se basa en el perfil de usuario, quien se espera que sea un investigador. El formato SVG permite almacenar las imágenes sin pérdida de calidad, lo que facilita su uso en cualquier contexto. Por otro lado, LaTeX proporciona la posibilidad de exportar los ficheros a formato PDF si se desea, o tomar directamente la información para añadirla a otros documentos. En cambio, si se hubiese utilizado directamente el formato PDF sería más estático para el usuario y dificultaría la extracción de la información de manera flexible.

Además de la estructura descrita anteriormente, se comprimirá toda la información en un archivo ZIP utilizando la librería Zipfile de Python. Esto permitirá agrupar todos los archivos generados en una sola entidad, facilitando su manejo y distribución.

El archivo ZIP contendrá los siguientes elementos:

- Imagen en formato SVG del gráfico de puntos.
- Imagen en formato SVG del gráfico de pirámide.
- Fichero LaTeX con la información general de las reglas.
- Fichero LaTeX con la información detallada de cada regla.

Al utilizar la biblioteca Zipfile de Python, se genera fácilmente el archivo ZIP que contiene todos estos elementos, proporcionando una forma útil de almacenar y compartir la información generada por la aplicación.

El archivo ZIP que se genera tendrá un nombre predeterminado, que es `info_general_nombreAlgoritmo.zip`, donde `nombreAlgoritmo` es el nombre del algoritmo que ha generado las reglas. Sin embargo, el usuario tendrá la opción de

asignar un nombre personalizado al archivo ZIP si así lo desea. De esta manera, podrán especificar un nombre más relevante y adecuado para su contexto específico.

4.1.4.1. Generación de las imágenes SVG

Para exportar los gráficos en formato SVG (Scalable Vector Graphics) se ha utilizado el método `savefig`, una función integrada en Matplotlib que permite guardar un gráfico en un archivo específico. En este caso, se utiliza para guardar los gráficos en formato SVG.

```
1 self.figGraf1.savefig('graficaPuntos.svg', format='svg')
2 self.figGraf2.savefig('graficaPiramide.svg', format='svg')
```

Código 6. Exportación de los gráficos a SVG

En este código, `self.figGraf1` y `self.figGraf2` representan las figuras o gráficos generados previamente en la aplicación. Utilizando el método `savefig` de estos objetos, se guarda cada gráfico en un archivo SVG.

4.1.4.2. Generación de los ficheros LaTeX

El archivo LaTeX se genera construyendo y concatenando cadenas de texto en Python, utilizando variables, expresiones y métodos para generar el contenido deseado. Luego, el resultado final se guarda en un archivo LaTeX.

Primero se define el tipo de documento que se va a generar y se incluyen los paquetes que son necesarios para generar correctamente el archivo PDF. Los paquetes que se utilizan son:

- `\usepackage{graphicx}`: Este paquete permite la inclusión de imágenes en el documento LaTeX. Complementado con `\usepackage{svg}` es posible insertar las imágenes SVG generadas anteriormente en el archivo.
- `\usepackage{longtable}`: Este paquete proporciona el entorno `longtable`, que permite crear tablas que se pueden dividir en varias páginas. Es utilizado en la sección de la tabla de reglas que cubren los datos y los datos que cubre cada regla, para asegurar que la tabla se ajuste correctamente en caso de que sea demasiado grande para una sola página.

- `\usepackage{tabularx}`: Este paquete proporciona el entorno `tabularx`, que es una variante del entorno `tabular` con capacidad para ajustar automáticamente el ancho de las columnas.
- `\usepackage{geometry}`: Este paquete permite personalizar los márgenes y el tamaño de página del documento LaTeX. En el código se utiliza el comando `\geometry` para establecer un margen izquierdo de 2 cm.

Finalmente, se indica que da comienzo el documento LaTeX.

```

1 latex_code = "\\documentclass{article}\\n"
2 latex_code += "\\usepackage{graphicx}\\n"
3 latex_code += "\\usepackage{longtable}\\n"
4 latex_code += "\\usepackage{tabularx}\\n"
5 latex_code += "\\usepackage{geometry}\\n"
6 latex_code += "\\geometry{left=2cm}\\n"
7 latex_code += "\\begin{document}\\n"

```

Código 7. Cabecera fichero LaTeX.

Una vez se definen los paquetes necesarios y se establece el formato del PDF resultante, se incluye la información de la aplicación de la siguiente manera:

- Para el fichero de información general *infoGeneral.tex*:
 - Se crea una sección denominada *Cabecera del dataset*. Este texto contiene información descriptiva sobre los atributos del dataset y corresponde con la cabecera del fichero del dataset.
 - Una segunda sección titulada *Gráficas*. En este caso, se insertan las dos gráficas en formato SVG (*graficaPuntos.svg* y *graficaPiramide.svg*) en el documento LaTeX. Después, se inserta un salto de página.
 - La última sección titulada *Tabla de las reglas que cubren a los datos*. Se crea una tabla en LaTeX con siete columnas y tantas filas como datos haya en el dataset. Las columnas corresponden con:

Nº ejemplo	Atributo_1	Atributo_2	...	Clase	Reglas cubren bien	Reglas cubren mal
------------	------------	------------	-----	-------	--------------------	-------------------

Tabla 11. Cabecera de la tabla de las reglas que cubren a los datos. Fichero LaTeX

- Para el fichero con la información de cada una de las reglas *infoReglas.tex*:

Para cada regla se crea una sección titulada con el nombre de la regla. Dentro de cada sección, se agregan subsecciones que contienen la información sobre dicha regla:

- La subsección titulada *Tabla de contingencias*, donde se genera una tabla en LaTeX con dos columnas y cuatro filas. Se agregan los valores true positive (tp), false positive (fp), true negative (tn) y false negative (fn) a la tabla.
- Seguidamente, la subsección denominada *Tabla de medidas (%)*. Se genera otra tabla en LaTeX con dos columnas y cuatro filas, donde se agregan los valores de Confianza, WRAccN, true positive rate (TPR) y false positive (FPr).
- Después, se crea una subsección titulada *Tabla de los datos que cubre*. Se genera una tabla en LaTeX con una columna y se agregan los datos cubiertos por la regla. Finalmente, se inserta un salto de página para la siguiente regla.

4.1.5. Difusión de la aplicación

Uno de los métodos de inicio de la aplicación va a ser mediante ejecutable. De esta manera, el usuario no necesitará tener acceso al código de la aplicación, ni descargar Python y las librerías, ni hacer uso de la consola.

Para generar el archivo ejecutable, se deben seguir los siguientes pasos:

1. Instalar la librería de Python *pyinstaller*, que se encarga de generar ejecutables a partir de archivos .py. Se puede instalar esta librería utilizando el siguiente comando en la consola:

```
pip install pyinstaller
```

2. Dentro de la carpeta del proyecto, ejecutar el siguiente comando para generar el archivo ejecutable:

```
pyinstaller --add-data "assets/*;assets/" main.py
```

Este comando indica que se incluya toda la carpeta "assets" y su contenido en el archivo ejecutable y que se guarde en la misma ubicación con el nombre "assets". Esta carpeta alberga el icono de la aplicación y el logo de la universidad presente en la primera pantalla.

Es importante destacar que el formato del archivo ejecutable (.exe) variará dependiendo del sistema operativo donde se haya generado el ejecutable. Se ha generado archivo ejecutable para Windows.

4.2. Front-end

Esta sección se enfocará en explicar cómo se han implementado los aspectos más destacables de la interfaz y las razones detrás de las decisiones tomadas al respecto, ya que se comentó el storyboard en la sección 3.5. Diseño de la interfaz.

4.2.1. Importar la información

Al pulsar el botón correspondiente para importar ficheros, se hará uso de la pantalla emergente proporcionada por la librería *filedialog* de Tkinter para que el usuario pueda seleccionar el archivo deseado.

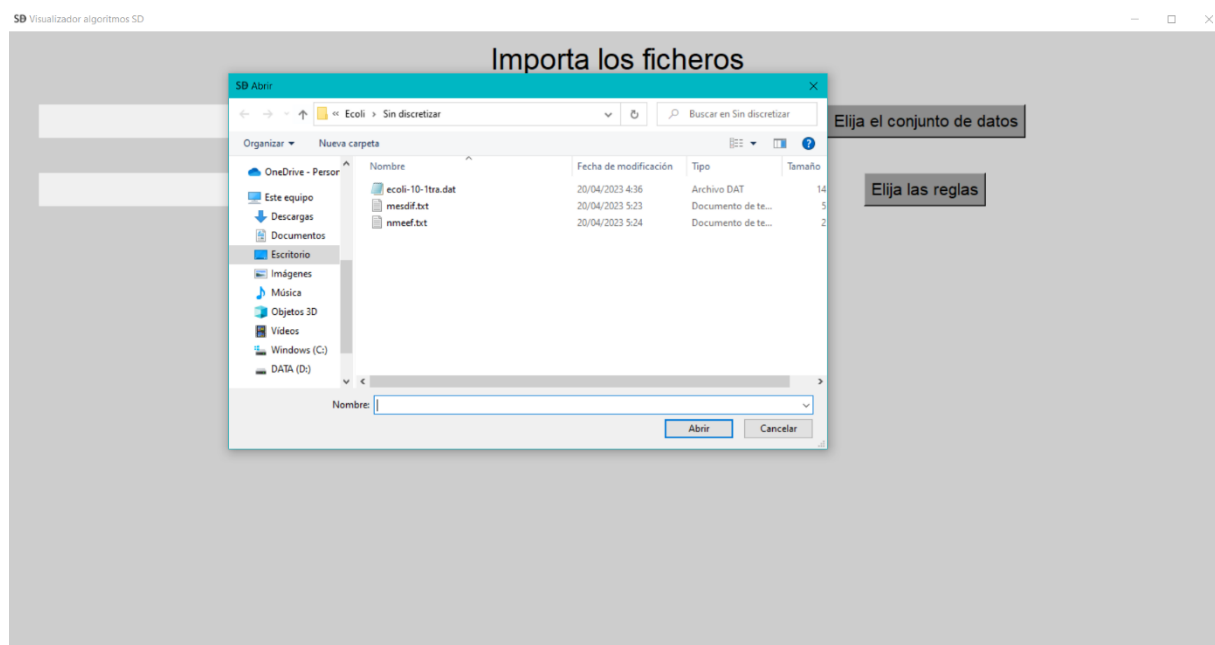


Figura 36. Ventana emergente de selección de fichero a importar. Fuente: Elaboración propia

En caso de que ocurra un error durante la lectura o el formato de los ficheros, se mostrará una ventana emergente utilizando la librería *messagebox* de Tkinter para informar al usuario del problema.

La librería *messagebox* de Tkinter es una herramienta que permite mostrar una ventana emergente para mostrar mensajes al usuario de la aplicación. Ofrece varias funciones para mostrar diferentes tipos de mensajes, como *showinfo* para mostrar información, *showwarning* para mostrar una advertencia, *showerror* para mostrar un mensaje de error, *askquestion* para hacer una pregunta.

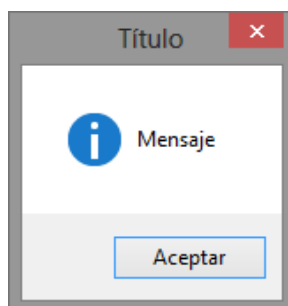


Figura 37. Visualización de *showinfo()*. Fuente:

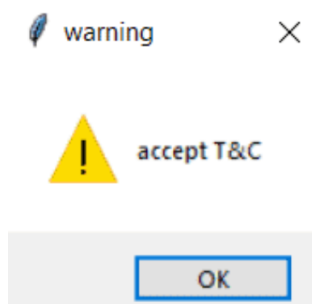


Figura 38. Visualización de *showwarning()*. Fuente:

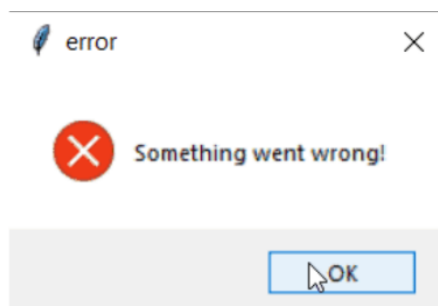


Figura 39. Visualización de *showerror()*. Fuente:

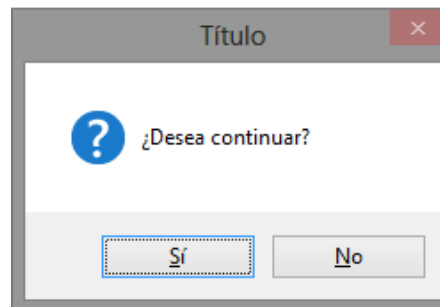


Figura 40. Visualización de askquestion(). Fuente:

En este proyecto se han utilizado las ventanas emergentes del tipo *showerror* y *showwarning* para informar al usuario sobre el problema que ha surgido a la hora de importar los ficheros. Esta ventana emergente le indica al usuario la naturaleza del problema y le impide avanzar en la aplicación hasta que lo haya solucionado. De esta manera, se pretende garantizar que el usuario tenga una experiencia de usuario óptima, evitando situaciones indeseadas.

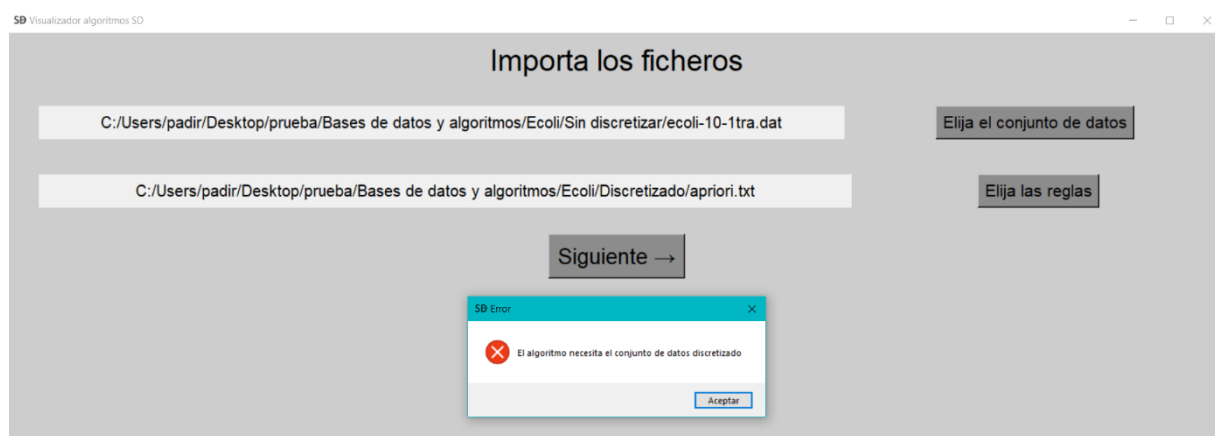


Figura 42. Visualización del mensaje del error04 al importar ficheros. Fuente: Elaboración propia

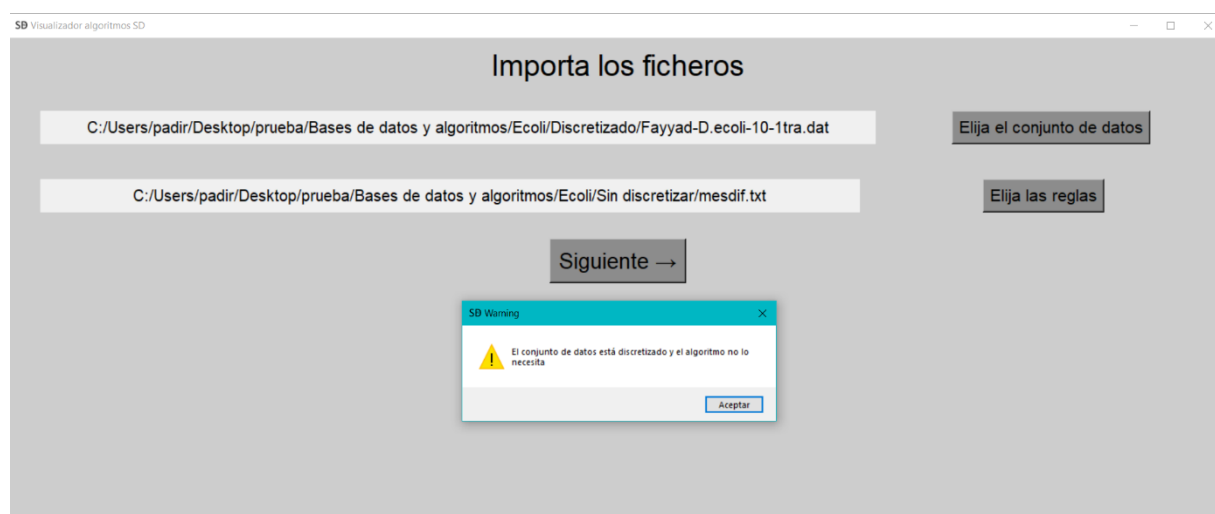


Figura 41. Visualización del warning01 al importar ficheros. Fuente: Elaboración propia

4.2.2. Información general

En esta pantalla se muestra la información general de todas las reglas e información del dataset

4.2.2.1. Información de la cabecera del dataset

En este proyecto se utiliza una ventana emergente del tipo *Toplevel* proporcionada por la librería Tkinter para mostrar la información de la cabecera del dataset. Dentro de esta ventana emergente, la información del dataset se muestra en una etiqueta creada con la función *Label* de Tkinter. De esta forma, se pretende que el usuario pueda visualizar la información de manera clara y accesible.

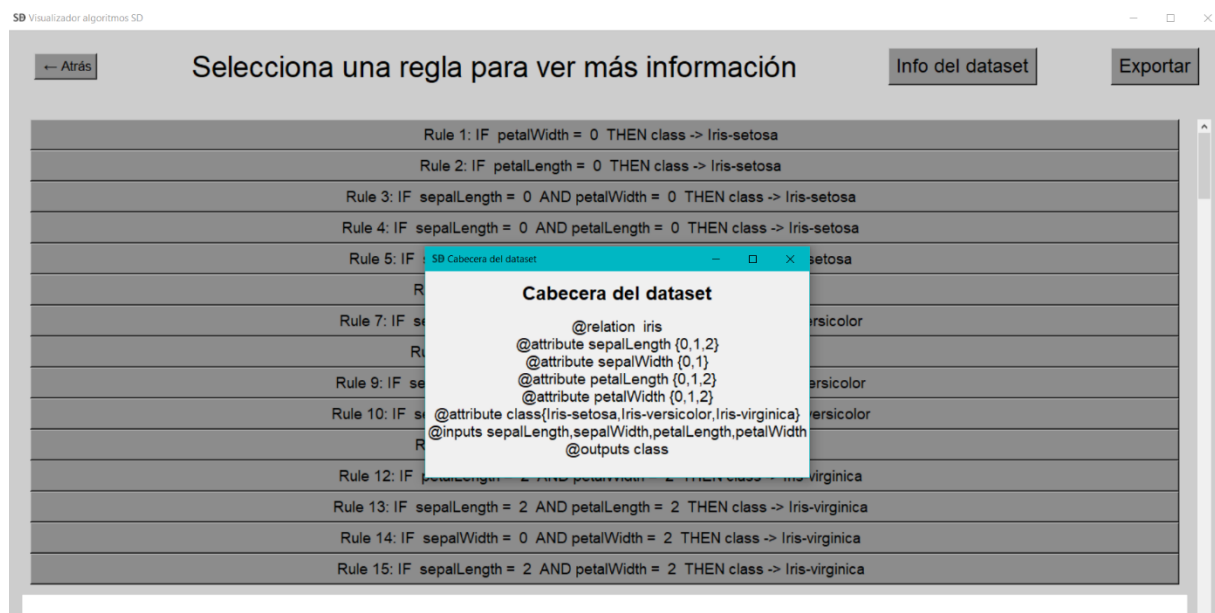


Figura 43. Ventana que muestra la cabecera del conjunto de datos. Fuente: Elaboración propia

4.2.2.2. Exportación

Así mismo, también se ha utilizado esta ventana emergente para indicar al usuario que ha finalizado la exportación. A diferencia de la ventana anterior, esta ventana emergente está programada para cerrarse automáticamente después de tres segundos, ya que solo se trata de un aviso puntual para el usuario.

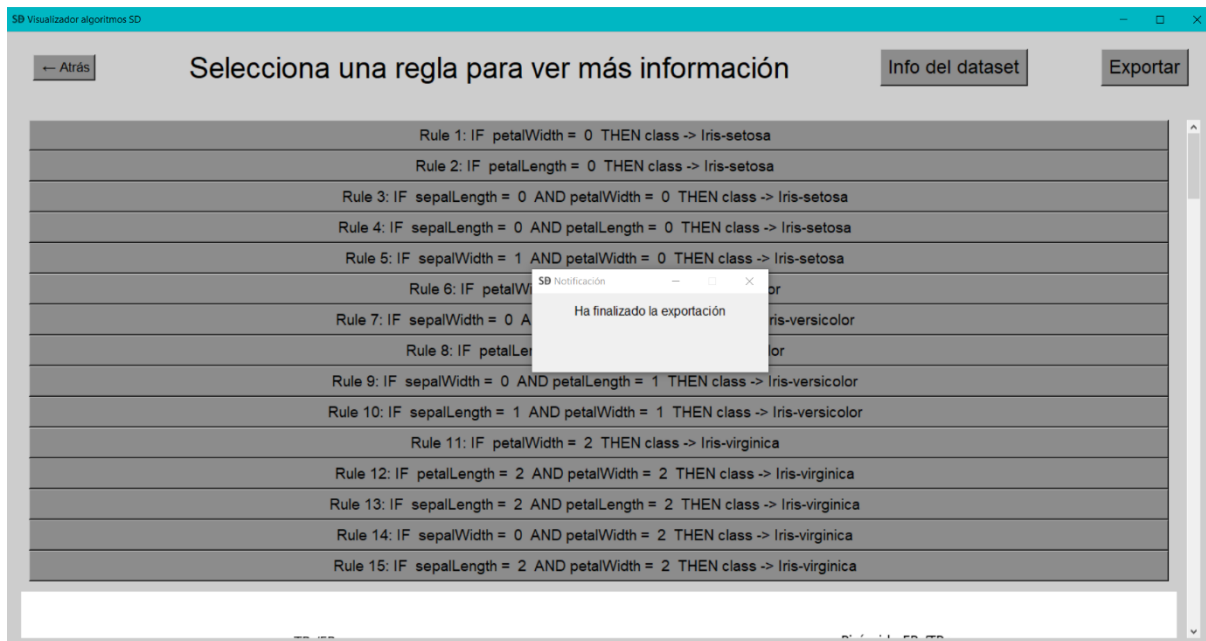


Figura 44. Ventana emergente indicando que ha finalizado la exportación. Fuente: Elaboración propia

Por otro lado, se ha utilizado en esta pantalla la librería *filedialog* de Tkinter, mencionada anteriormente, para permitir al usuario elegir la ubicación donde se almacenará la exportación de los datos. Esta librería proporciona una ventana emergente de navegación para que el usuario pueda seleccionar la ruta deseada y crear nuevas carpetas. También, se encarga de manejar los casos en los que ya existen archivos con el mismo nombre, permitiendo al usuario reemplazarlos.

Además de la funcionalidad mencionada, esta librería también permite personalizar la ventana de navegación para incluir filtros de archivos, nombre predeterminado del archivo y mucho más. En este caso, se filtra por ficheros de tipo ZIP y se pone un nombre genérico al archivo que se va a generar.

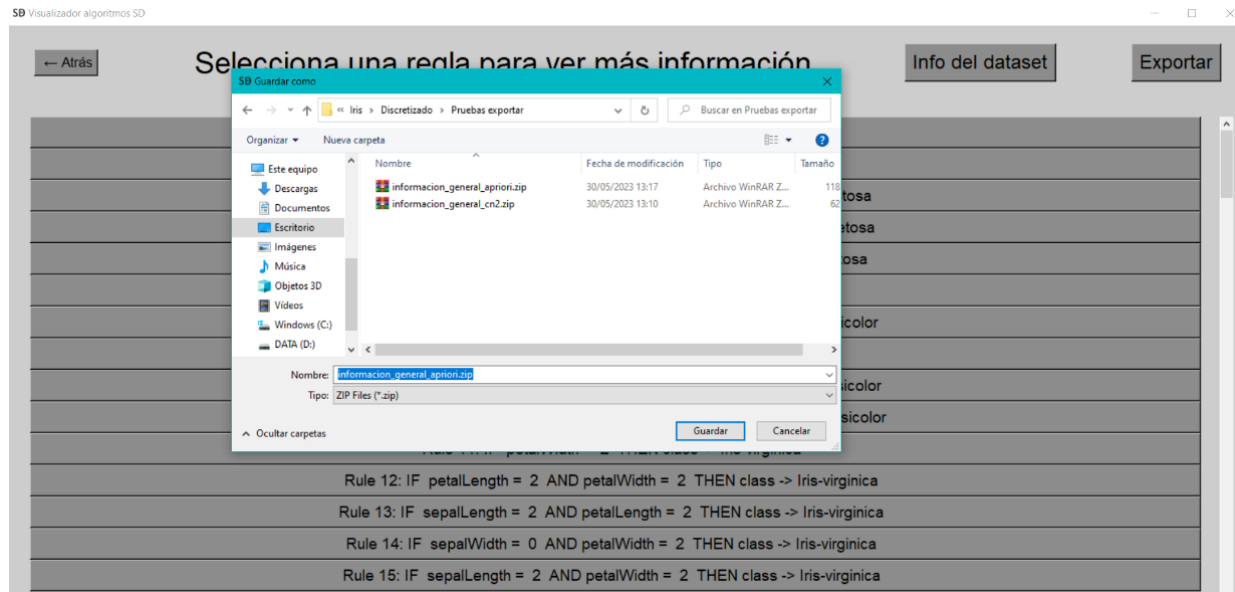


Figura 45. Ventana emergente para exportar la información. Fuente: Elaboración propia

4.2.2.3. Gráficas

Para crear las gráficas se ha utilizado la librería Matplotlib, ya que ofrece una gran variedad de tipos de gráficas y permite generarlas simplemente indicándole los valores adecuados. En este caso, se ha utilizado una gráfica de puntos y una gráfica de barras horizontales, ambas expresadas en porcentaje.

En la gráfica de puntos, el eje x corresponde al FPr (false positive rate) y el eje y al TPr (true positive rate), y cada punto representa una regla. Para destacar las reglas de peor calidad, se ha dibujado un área roja en la línea $x=y$, ya que las reglas que se encuentran en esa área tienen un TPr más bajo que el FPr, lo cual indica que las reglas son de muy mala calidad. En este tipo de gráfica, una regla será mejor cuanto más cerca esté de la coordenada (0,100), es decir, cuanto más cerca esté de la esquina superior izquierda.

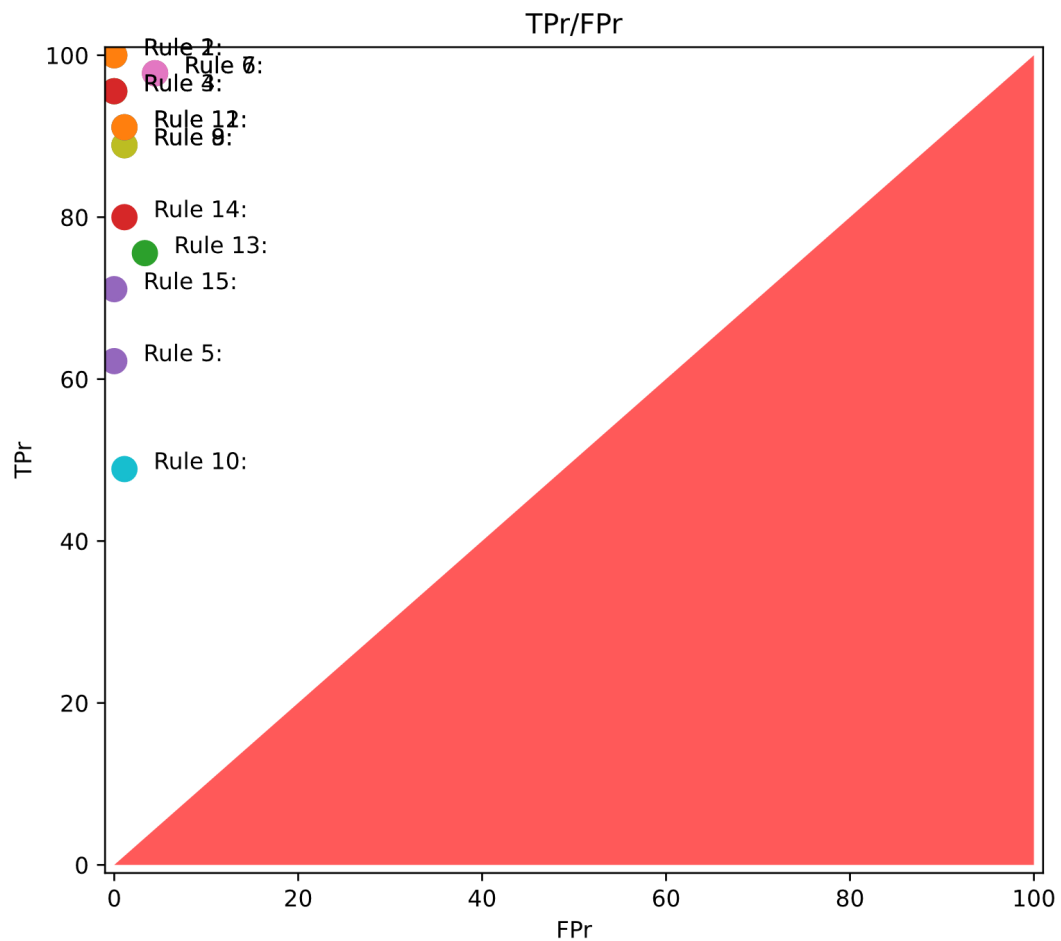


Figura 46. Gráfico de puntos generado con Matplotlib. Fuente: Elaboración propia

En cuanto al gráfico de barras horizontales, se ha tenido que modificar para que el eje x empiece en el valor -100 y termine en el 100. De esta manera, el valor de FPr se indica en negativo, haciendo que la barra vaya de derecha a izquierda, mientras que el TPr se indica en positivo, haciendo que la barra vaya de izquierda a derecha. Además, para evitar confusiones de color para los daltónicos, se ha marcado el FPr con el color naranja y el TPr con el color azul. Cada par de barras representa una regla.

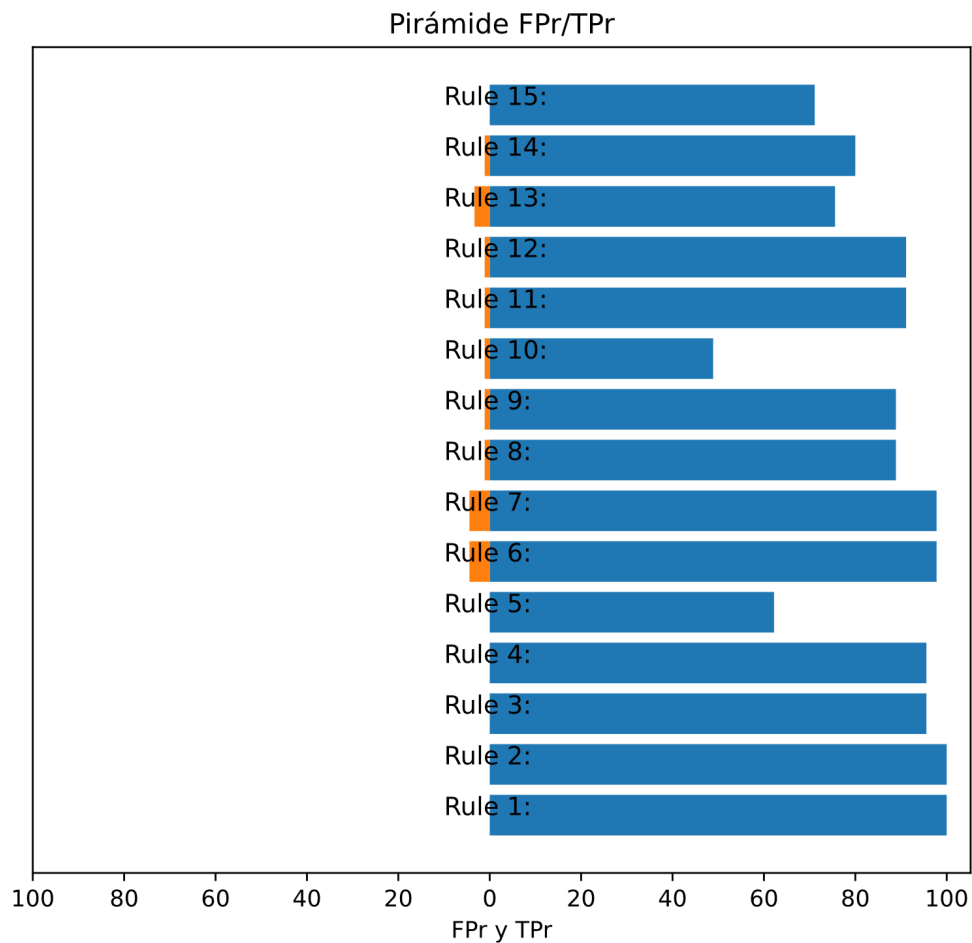


Figura 47. Gráfico de pirámide generado con Matplotlib. Fuente: Elaboración propia

Por último, para insertar estas gráficas en la aplicación se ha utilizado la librería *FigureCanvasTkAgg* de Matplotlib, que permite mostrar las gráficas en una ventana de Tkinter de manera sencilla y eficiente. De esta forma, se logra una integración perfecta entre la visualización de los datos y la interfaz de usuario de la aplicación.

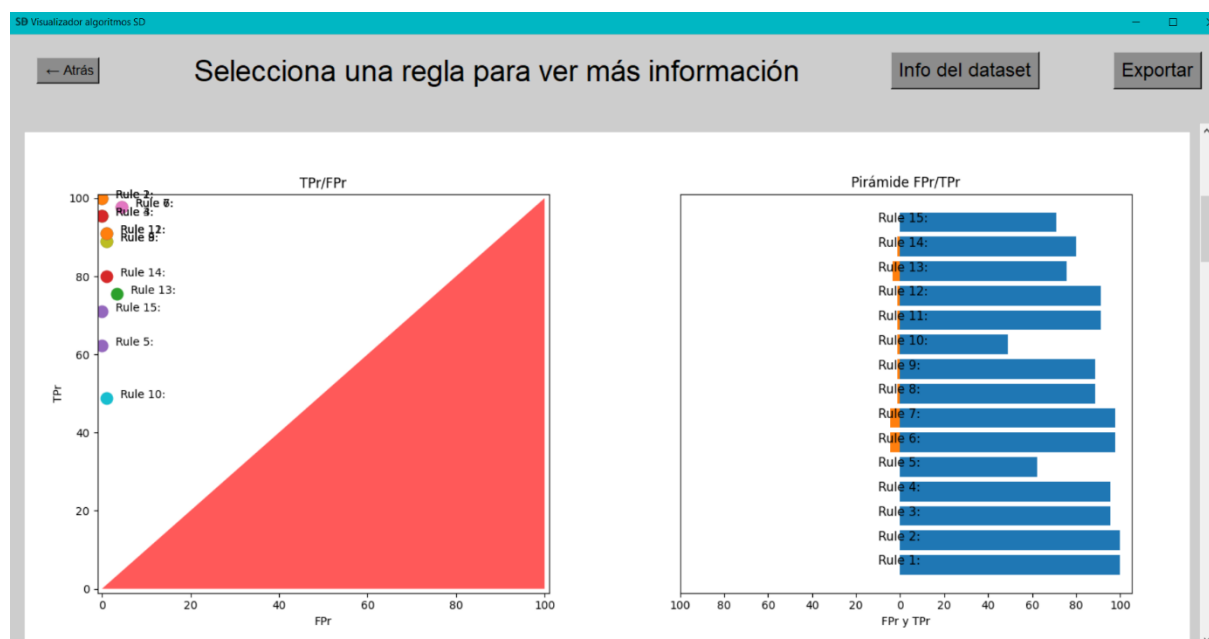


Figura 48. Visualización de las gráficas en la pantalla de información general. Fuente: Elaboración propia

4.2.2.4. Tabla de las reglas que cubre cada dato

La tabla de las reglas que cubre cada dato está compuesta por varios *Entry*, un objeto de Tkinter de introducción de texto pero que en este caso está deshabilitado. Cada celda corresponde con un *Entry*, cada fila corresponde con un dato y cada columna representa:

Información del dato	Reglas que lo cubren bien	Reglas que lo cubren mal
----------------------	---------------------------	--------------------------

Tabla 12. Cabecera de la tabla de las reglas que cubren a cada dato

Es importante mencionar que la información del texto se encuentra acortada, es decir, se han eliminado algunos atributos, manteniendo solamente el número de ejemplo, los tres primeros atributos y la clase. Esta reducción de contenido se ha realizado para evitar que la información no quepa en la celda y se corte, ya que algunos datos pueden tener muchos atributos o sus valores son extremadamente largos.

En cuanto a los colores de las reglas, se han mantenido respecto a las gráficas comentadas anteriormente. Es decir, las reglas que cubren correctamente el dato se muestran en color azul, mientras que las reglas que lo cubren incorrectamente aparecen en color naranja.

En el caso de las reglas, solo se muestra el número de la regla debido a la razón anteriormente mencionada, ya que si se incluyera el texto completo de cada regla ("Rule 1, Rule 2, ..., Rule n"), podría no caber en la celda correspondiente si hay muchas reglas. Finalmente, si se están evaluando reglas difusas, se muestra entre paréntesis el valor de pertenencia al dato.

Reglas que cubren a cada dato		
Ejemplo 1: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 2: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 3: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 4: 1, 0, 1, ..., cp	1, 3	29, 30, 31, 32
Ejemplo 5: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 6: 1, 0, 1, ..., cp	1, 3	29, 30, 31, 32
Ejemplo 7: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 8: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 9: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 10: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 11: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 12: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 13: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 14: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 15: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 16: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 17: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 18: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 19: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 20: 0, 0, 1, ..., cp	1, 2, 3, 4	9, 10, 11, 12
Ejemplo 21: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 22: 0, 0, 1, ..., cp	1, 2, 3, 4	
Ejemplo 23: 0, 0, 1, ..., cp	1, 2, 3, 4	

Figura 49. Visualización de la tabla de las reglas que cubren a cada dato con reglas no difusas. Fuente: Elaboración propia

Reglas que cubren a cada dato		
Ejemplo 1: 49.0, 29.0, 48.0, ..., cp		5(0.64)
Ejemplo 2: 7.0, 4.0, 48.0, ..., cp		
Ejemplo 3: 56.0, 4.0, 48.0, ..., cp		
Ejemplo 4: 59.0, 49.0, 48.0, ..., cp	1(0.89)	5(0.9)
Ejemplo 5: 23.0, 32.0, 48.0, ..., cp	1(0.66)	5(0.71)
Ejemplo 6: 67.0, 39.0, 48.0, ..., cp	1(0.83)	5(0.87)
Ejemplo 7: 29.0, 28.0, 48.0, ..., cp		5(0.62)
Ejemplo 8: 2.0, 44.0, 48.0, ..., cp	1(0.96)	5(0.99)
Ejemplo 9: 42.0, 4.0, 48.0, ..., cp		
Ejemplo 10: 42.0, 24.0, 48.0, ..., cp	1(0.6)	5(0.53)
Ejemplo 11: 25.0, 48.0, 48.0, ..., cp		5(0.92)
Ejemplo 12: 39.0, 32.0, 48.0, ..., cp		5(0.71)
Ejemplo 13: 51.0, 5.0, 48.0, ..., cp		
Ejemplo 14: 25.0, 4.0, 48.0, ..., cp		
Ejemplo 15: 34.0, 45.0, 48.0, ..., cp		5(0.99)
Ejemplo 16: 44.0, 27.0, 48.0, ..., cp	1(0.75)	5(0.6)
Ejemplo 17: 23.0, 4.0, 48.0, ..., cp		
Ejemplo 18: 41.0, 57.0, 48.0, ..., cp		5(0.71)
Ejemplo 19: 4.0, 45.0, 48.0, ..., cp		5(0.99)
Ejemplo 20: 31.0, 23.0, 48.0, ..., cp		5(0.51)
Ejemplo 21: 51.0, 54.0, 48.0, ..., cp	1(0.81)	5(0.78)
Ejemplo 22: 3.0, 16.0, 48.0, ..., cp	3(0.75)	
Ejemplo 23: 36.0, 30.0, 48.0, ..., cp	2(0.72)	5(0.87)

Figura 50. Visualización de la tabla de las reglas que cubren a cada dato con reglas difusas. Fuente: Elaboración propia

Tooltip

Con el fin de solucionar el problema anteriormente mencionado, en el que se pierde información en la tabla debido a que no cabe en la celda o se ha acortado para que quepa, se ha implementado una herramienta muy útil conocida como *tooltip*.

En Tkinter, un *tooltip* es una descripción emergente que aparece cuando el usuario posiciona el cursor del ratón sobre un objeto durante un cierto período de tiempo. Esta descripción suele contener información adicional sobre el objeto.

En este proyecto, se ha utilizado un *tooltip* para mostrar la información completa del dato. De esta manera, el usuario puede obtener información detallada simplemente posicionando el cursor del ratón sobre la celda correspondiente durante un cierto período de tiempo.

En Tkinter, una de las clases principales para crear *tooltips* es *Hovertip*. Sin embargo, el formato predeterminado de *Hovertip* presenta limitaciones en cuanto a la personalización de la apariencia y el contenido de la información que se muestra. Por eso que se ha creado una clase personalizada llamada *CustomHovertip*, la cual hereda de *Hovertip* y permite modificar los parámetros necesarios para adaptar el tooltip a las necesidades específicas. Al heredar de *Hovertip*, *CustomHovertip* mantiene todas las funcionalidades de la clase principal, pero añade la capacidad de personalizar aspectos como el tamaño de letra, la posición de la ventana y el texto que se muestra.



Figura 51. Visualización del tooltip en la pantalla de la información general. Fuente: Elaboración propia

4.2.3. Información de una regla individual

A continuación, se describen los elementos visuales implementados para mostrar la información relativa a cada una de las reglas.

4.2.3.1. Tabla de contingencias y tabla de medidas

Las dos tablas a las que se hace referencia tienen cuatro filas y dos columnas cada una. Cada celda de ambas tablas está configurada como un objeto *Entry* de Tkinter, un widget de entrada de datos que permite al usuario ingresar texto o números.

En este caso, las celdas de las tablas se han inhabilitado para que solo se pueda leer lo que está escrito en ellas, lo que significa que no es posible editar el contenido de las celdas. De esta manera, se garantiza que la información en las tablas permanezca intacta y no se puedan realizar cambios no deseados.

Tabla de contingencias	
True positive (tp)	False positive (fp)
44	4
True negative (tn)	False negative (fn)
86	1
Tabla de medidas (%)	
Conf	WRAccN
91.67	32.48
True positive rate (TPR)	False negative rate (FPr)
97.78	4.44

Figura 52. Visualización de las tablas de contingencias y de medidas. Fuente: Elaboración propia

4.2.3.2. Tabla de los datos que cubre la regla

La tabla consta de una sola columna y un número variable de filas, que corresponde al número de datos que se quieren representar. Cada fila de la tabla contiene un único dato. Al igual que en la tabla de la pantalla anterior, los datos que la regla cubre correctamente se representan en color azul, mientras que los datos que están cubiertos erróneamente se representan en color naranja.

Es importante destacar que esta tabla también puede presentar el mismo problema que se mencionó anteriormente, es decir, que la información del dato no quepa dentro de la celda. Para solucionar este problema, se ha incluido el *tooltip* comentado anteriormente. De esta manera, se garantiza que el usuario pueda acceder a toda la información del dato.

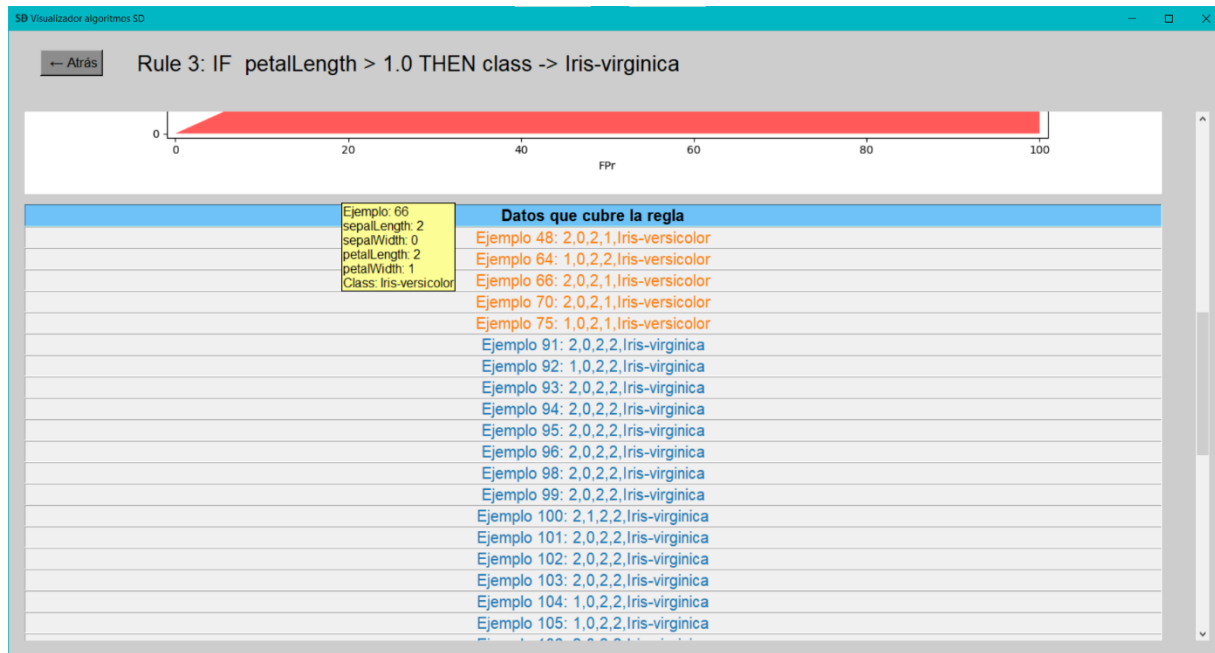


Figura 53. Visualización del tooltip en la pantalla de una regla específica. Fuente: Elaboración propia

Capítulo V

CASOS DE PRUEBA

En este capítulo, se presentará la puesta a prueba llevada a cabo en este trabajo para evaluar el rendimiento de la aplicación. Se mostrará cómo se visualiza la información a través de un par de ejemplos, considerando diferentes algoritmos SDRD incluidos en la aplicación. El objetivo es analizar el comportamiento de la aplicación en escenarios diversos y determinar su capacidad para manejar diferentes situaciones y volúmenes de datos.

Se presentarán dos casos de prueba diferentes, uno referente a un algoritmo difuso y otro a un algoritmo no difuso o nítido. Se explicará detalladamente la metodología utilizada en cada caso de prueba, así como los resultados obtenidos y su interpretación.

La principal diferencia entre ambos algoritmos radica en la evaluación interna que se realiza de las reglas. En el caso del algoritmo difuso, es necesario calcular un grado de pertenencia para cada dato, a fin de determinar si una regla cubre al dato y en qué proporción. Este cálculo se realiza mediante técnicas de lógica difusa, que permiten modelar la incertidumbre y la imprecisión en los datos.

Por otro lado, en el caso del algoritmo no difuso o nítido, se utiliza una evaluación binaria de las reglas, es decir, una regla se considera que cubre un dato si cumple con todas sus condiciones. En este caso, no es necesario calcular el grado de pertenencia de cada dato, lo que simplifica en cierta medida el proceso de evaluación.

En cuanto a la visualización de los resultados, en el caso del algoritmo difuso se muestra el grado de pertenencia en la lista de reglas que cubren a los datos, lo que permite comprender mejor el nivel de incertidumbre en los datos y la precisión del modelo. En cambio, en el caso del algoritmo no difuso, se muestra simplemente la lista de reglas que cubren a los datos, sin información adicional sobre la precisión del modelo.

5.1. Caso 1: Algoritmo difuso.

Conjunto de datos: Ecoli. Algoritmo: mesdif

En este caso de estudio se analizará la visualización de las reglas descriptivas supervisadas obtenidas en la ejecución del algoritmo MESDIF (Berlanga et al., 2006) sobre el conjunto de datos ecoli 10-1 tra. Este conjunto de datos corresponde al primer fichero de entrenamiento de la base de datos Ecoli, la cual es ampliamente utilizada en el campo de la Ciencia de Datos para la evaluación de algoritmos de aprendizaje automático.

El algoritmo MESDIF es un algoritmo de aprendizaje automático que se utiliza para la construcción de reglas descriptivas supervisadas. Su funcionamiento se basa en la detección de patrones relevantes en los datos y la generación de reglas que permitan describir dichos patrones de manera clara y concisa. Este algoritmo ha sido ampliamente estudiado y utilizado en la literatura debido a su efectividad y eficiencia en la construcción de reglas descriptivas supervisadas.

Por otro lado, es importante destacar que el fichero de datos de entrenamiento se utiliza para alimentar al algoritmo de aprendizaje automático durante la fase de entrenamiento. En este proceso, el algoritmo utiliza esta información para aprender y construir un modelo que permita la clasificación o descripción de nuevos datos. En este caso, el fichero de datos de entrenamiento del conjunto de datos ecoli 10-1 tra se utiliza para la construcción de las reglas descriptivas supervisadas mediante el algoritmo MESDIF y posteriormente, se realiza un análisis de la visualización de estas reglas con el fin de evaluar su efectividad e interpretabilidad.

5.1.1. Fichero de datos

Nombre: ecoli-10-1-tra.dat

Cabecera del dataset:

```
@relation ecoli
@attribute a1 real [0.0, 89.0]
@attribute a2 real [1.0, 88.0]
@attribute a3 real [1.0, 48.0]
@attribute a4 real [1.0, 5.0]
@attribute a5 real [0.0, 88.0]
@attribute a6 real [1.0, 94.0]
@attribute a7 real [0.0, 99.0]
@attribute class {cp, im, imS, imL, imU, om, omL,
pp}
@inputs a1, a2, a3, a4, a5, a6, a7
@outputs class
```

La cabecera del dataset proporciona información importante acerca del conjunto de datos Ecoli utilizado en este caso de estudio. Los datos están compuestos de siete atributos diferentes más la clase correspondiente según el valor de los atributos. Es importante tener en cuenta que, debido a la naturaleza de los atributos, este fichero

de datos solo es adecuado para el entrenamiento de algoritmos con lógica difusa que sean capaces de subdividir el intervalo de valores continuos en varios subintervalos y asignar a cada parte una etiqueta correspondiente. Esta etiqueta posteriormente sustituirá a los valores del atributo en los datos para poder obtener las reglas descriptivas supervisadas. Cabe destacar que los datos pueden clasificarse en hasta ocho clases diferentes.

Ejemplo de datos:

```
49.0, 29.0, 48.0, 5.0, 56.0, 24.0, 35.0, cp
7.0, 4.0, 48.0, 5.0, 54.0, 35.0, 44.0, cp
```

5.1.2. Fichero de reglas

El fichero incluye 24 reglas, pero solo se han mostrado dos en esta memoria.

```
@algorithm mesdif
```

```
Number of labels: 3
```

```
GENERATED RULE 0
```

```
Antecedent
```

```
Variable a2 = Label 1      (1.0 44.5 88.0)
```

```
Variable a5 = Label 1      (0.0 44.0 88.0)
```

```
Variable a6 = Label 1      (1.0 47.5 94.0)
```

```
Consequent: cp
```

```
GENERATED RULE 1
```

```
Antecedent
```

```
Variable a2 = Label 1      (1.0 44.5 88.0)
```

```
Variable a5 = Label 1      (0.0 44.0 88.0)
```

```
Variable a6 = Label 0      (-45.5 1.0 47.5)
```

```
Variable a7 = Label 0      (-49.5 0.0 49.5)
```

```
Consequent: cp
```

```
GENERATED RULE 2
```

```
Antecedent
```

```
Variable a1 = Label 0      (-44.5 0.0 44.5)
```

```
Variable a5 = Label 1      (0.0 44.0 88.0)
```

```
Variable a6 = Label 0      (-45.5 1.0 47.5)
```

```
Variable a7 = Label 0      (-49.5 0.0 49.5)
```

```
Consequent: cp
```

[...]

La información presentada en el fichero de reglas es esencial para comprender el proceso de generación de reglas descriptivas supervisadas a partir del algoritmo MESDIF utilizado en este caso de estudio.

En primer lugar, se indica el nombre del algoritmo que ha generado las reglas, lo que permite a la aplicación identificar la estructura del fichero. Además, se especifica el número de etiquetas en las que el algoritmo ha subdividido los intervalos de los atributos, lo que es importante para entender cómo se han clasificado los datos y permite realizar internamente las mismas subdivisiones.

Seguidamente, se presenta la lista de todas las reglas generadas por el algoritmo, donde cada regla se compone de dos partes: el *Antecedent* y el *Consequent*. En la parte de *Antecedent* se especifica la etiqueta que el algoritmo ha asignado a cada atributo. Cada línea entre *Antecedent* y *Consequent* corresponde con un atributo, lo que facilita la comprensión de las reglas generadas. En la parte de *Consequent*, se indica la clase según los valores de los atributos.

5.1.3. Visualización general en la aplicación

En esta sección se puede observar el resultado obtenido para el conjunto de datos estudiado en este caso, donde se puede observar una pantalla con tres partes bien diferenciadas.

En la primera parte, se muestran todas las reglas contenidas en el fichero de reglas, lo que permite al usuario consultar individualmente la información de cada una de ellas. Esta sección es fundamental para analizar la eficacia de cada una de las reglas en la clasificación de los datos.

La segunda parte se compone de dos gráficas. La de la izquierda es una gráfica de puntos que indica la dispersión generalizada de las reglas según sus TPr y sus FPr, mientras que la de la derecha es una gráfica piramidal que muestra la distribución de TPr y FPr para cada una de las reglas. Las reglas que no presentan ninguna barra en la gráfica piramidal o se encuentran en la esquina inferior izquierda del gráfico de puntos son consideradas como reglas ineficaces, ya que no han sido capaces de asignar correctamente los valores de los atributos para que estén relacionados con la clase asignada.

Finalmente, la tercera parte de la imagen está compuesta por una tabla con tres columnas. En la primera columna se indica el dato correspondiente, en la segunda columna de color azul se muestran las reglas que han cubierto correctamente ese dato y en la tercera columna se muestran de color naranja las reglas que han cubierto erróneamente a ese dato. Todas las reglas tienen indicado a su derecha entre paréntesis el grado de pertenencia al dato. Esta parte es especialmente útil para comprender cómo las reglas están clasificando los datos y para identificar patrones y tendencias en la clasificación.

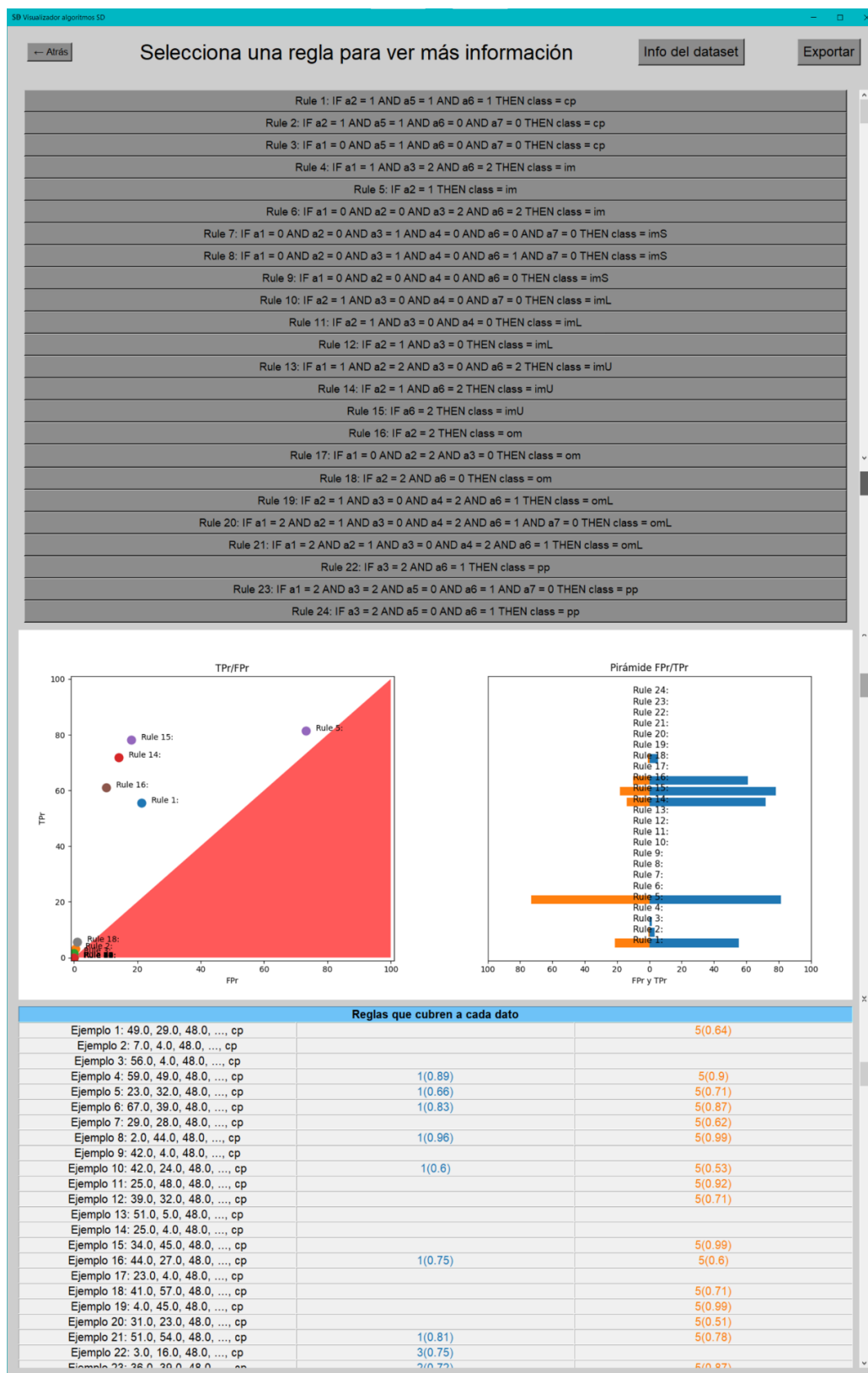


Figura 54. Caso 1: Vista general de la información general de las reglas. Fuente: Elaboración propia

5.1.4. Información de una de las reglas

En esta sección se puede observar el resultado obtenido para una regla descriptiva supervisada en concreto, donde se puede observar una pantalla con tres partes bien diferenciadas.

En la primera parte se encuentran las tablas con las medidas de calidad de la regla:

- La tabla de contingencias indica cuántos ejemplos ha cubierto la regla y cuántos no, así como cuántos ha cubierto correctamente (en azul) y cuántos ha cubierto incorrectamente (en naranja).
- La tabla de medidas representa las medidas de evaluación, donde se incluyen la Confianza, el WRAccN, la tasa de tp (TPr) y la tasa de fp (FPr).

Estas medidas son fundamentales para comprender el rendimiento de la regla y determinar su eficacia en la clasificación de los datos.

En la segunda sección se presenta una gráfica de puntos que permite visualizar la dispersión de la regla a partir de los valores de TPr (tasa de verdaderos positivos) y FPr (tasa de falsos positivos). Esta gráfica es especialmente útil para analizar el rendimiento de la regla en la clasificación de los datos y para identificar patrones y tendencias.

En la tercera sección se encuentra una tabla que recoge todos los datos que la regla ha cubierto, indicando si los ha cubierto correctamente o no. Los datos cubiertos correctamente se muestran en color azul, mientras que los datos cubiertos incorrectamente se muestran en color naranja. Esta tabla proporciona información detallada sobre cómo la regla está clasificando los datos y permite identificar posibles errores en el proceso de clasificación.

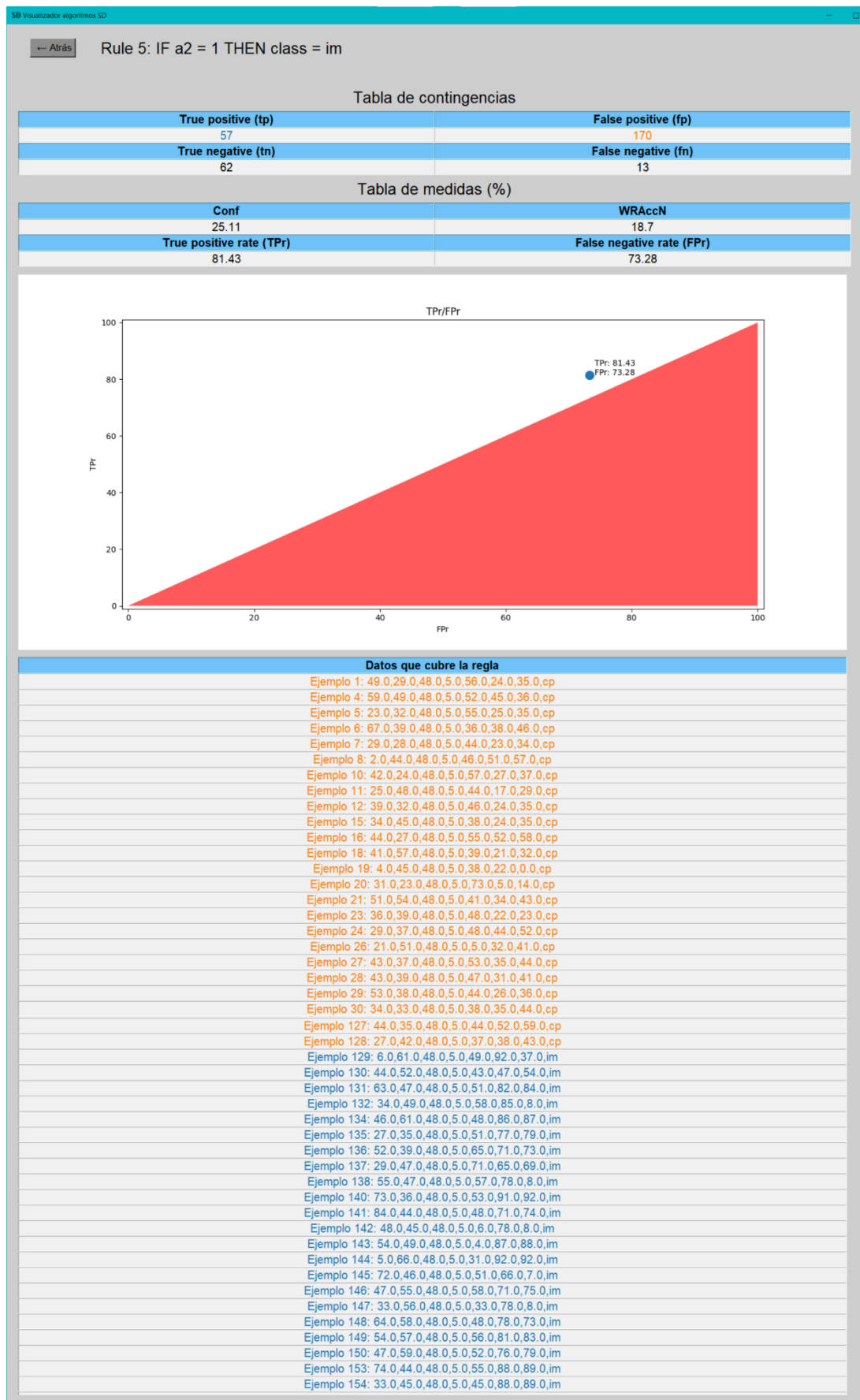


Figura 55. Información concreta de la regla 5. Fuente: Elaboración propia

5.2. Caso 2: Algoritmo no difuso o nítido. Conjunto de datos: Ecoli. Algoritmo: cn2

En este caso de estudio se analizará la visualización de las reglas descriptivas supervisadas obtenidas en la ejecución del algoritmo CN2-SD (Lavraç et al., 2004) sobre el conjunto de datos ecoli 10-1 tra. Este conjunto de datos corresponde al primer fichero de entrenamiento de la base de datos Ecoli, la cual es ampliamente utilizada en el campo de la Ciencia de Datos para la evaluación de algoritmos de aprendizaje automático.

El algoritmo CN2-SD es un algoritmo utilizado para descubrir subgrupos interesantes en un conjunto de datos. El objetivo del algoritmo es encontrar subgrupos que tengan una alta tasa de ganancia de información y que sean lo suficientemente grandes como para ser significativos. El algoritmo CN2-SD comienza dividiendo el conjunto de datos en subconjuntos más pequeños y aplicando técnicas de minería de datos para descubrir reglas que describan subgrupos interesantes.

5.2.1. Fichero de datos

Nombre: Fayyad-D.ecoli-10-1tra.dat

El fichero de datos corresponde con el expuesto anteriormente, con la salvedad de que éste ha pasado por un proceso de discretización antes de que el algoritmo trabajara con él.

Cabecera del dataset:

```
@relation ecoli
@attribute a1 {0,1,2}
@attribute a2 {0,1}
@attribute a3 {0,1}
@attribute a4 {0}
@attribute a5 {0,1,2}
@attribute a6 {0,1,2,3,4}
@attribute a7 {0,1}
@attribute class{cp,im,imS,imL,imU,om,omL,pp}
@inputs a1,a2,a3,a4,a5,a6,a7
@outputs class
```

La cabecera del dataset proporciona información detallada sobre la estructura del conjunto de datos Ecoli utilizado en este caso de estudio. El conjunto de datos está compuesto por siete atributos diferentes, cada uno con valores discretos, y una clase correspondiente que se determina en función de los valores de los atributos. Es importante destacar que, debido a la naturaleza de los atributos, este conjunto de datos es adecuado para el entrenamiento de algoritmos nítidos, ya que los valores de los atributos son directamente las etiquetas que se han generado por el preprocesamiento de los datos aplicándole el algoritmo de Fayyad (Fayyad et al., 1996). Esto significa que el valor de los atributos en los datos corresponde con una etiqueta, lo que facilita el proceso de clasificación.

Cada atributo del conjunto de datos tiene un conjunto finito de valores posibles. El atributo a1 tiene tres valores posibles (0, 1, 2), el atributo a2 tiene dos valores posibles (0, 1), el atributo a3 tiene dos valores posibles (0, 1), el atributo a4 tiene un solo valor posible (0), el atributo a5 tiene tres valores posibles (0, 1, 2), el atributo a6 tiene cinco valores posibles (0, 1, 2, 3, 4), y el atributo a7 tiene dos valores posibles (0, 1).

La clase correspondiente a cada instancia del conjunto de datos puede ser una de ocho posibles clases diferentes: cp, im, imS, imL, imU, om, omL, pp. Estas clases representan diferentes tipos de proteínas bacterianas y son importantes para la clasificación de las instancias del conjunto de datos.

Ejemplo de datos:

0,0,1,0,0,2,0,cp
0,0,1,0,0,2,0,cp

5.2.2. Fichero de reglas

El fichero incluye 17 reglas, pero solo se han mostrado dos en esta memoria.

@algorithm cn2

Rule 1: IF a6 <= 3.0 AND a1 = 0.0 AND a2 = 0.0 AND a7 = 0.0 THEN
class -> cp [0.952755905511811 0.023622047244094488 0.0 0.0 0.0
0.0 0.0 0.023622047244094488]

Rule 2: IF a6 <= 3.0 AND a2 = 0.0 AND a7 = 0.0 AND a3 = 1.0 AND a1
<= 1.0 THEN class -> cp [0.8639455782312925
0.027210884353741496 0.0 0.0 0.006802721088435374
0.02040816326530612 0.0 0.08163265306122448]

[...]

La información presentada en el fichero de reglas es esencial para comprender el proceso de generación de reglas descriptivas supervisadas a partir del algoritmo CN2-SD utilizado en este caso de estudio.

En primer lugar, se indica el nombre del algoritmo que ha generado las reglas, lo que permite a la aplicación identificar la estructura del fichero.

A continuación, se presenta una lista de todas las reglas generadas por el algoritmo, cada línea corresponde con una regla diferente. Cada regla indica para los atributos el valor de la etiqueta que ha asignado para dar como consecuente la clase. Este algoritmo no les asigna a los atributos una etiqueta como tal, sino que emplea diferentes operadores como son: distinto ($\neq$), mayor que ($>$), menor que ($<$), mayor o igual que ($\geq$) o menor o igual que ($\leq$). Los valores indicados después de la regla son las medidas de calidad de la regla.

5.2.3. Visualización general en la aplicación

Al igual que en el caso anterior, en esta sección se puede observar el resultado obtenido para el conjunto de datos estudiado en este caso, donde se puede observar una pantalla con tres partes bien diferenciadas.

Las dos primeras partes son exactamente iguales a las descritas en el caso anterior. Solo hay una pequeña diferencia en la tercera parte, y es que no se muestra el grado de pertenencia de la regla, ya que en los algoritmos nítidos no existe este concepto ya que la regla cubre (1) o no cubre (0) al dato.



5.2.4. Información de una de las reglas

Esta sección es exactamente igual que la del caso anterior.

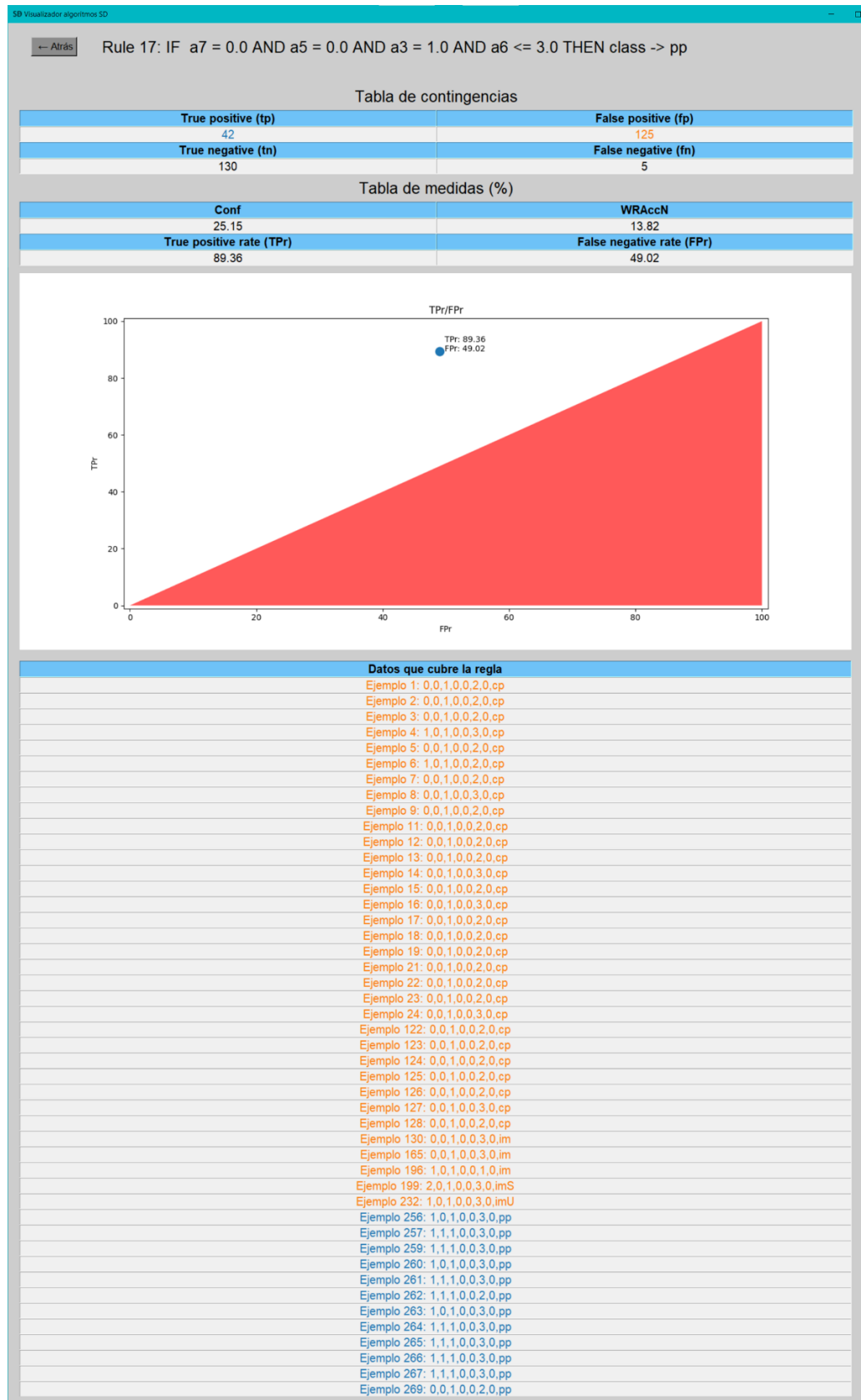


Figura 57. Información concreta de la regla 17. Fuente: Elaboración propia

En las demostraciones realizadas, se han podido observar no solo las diferencias significativas en la evaluación y representación de la información comentadas anteriormente, sino también en los ficheros de reglas y en los ficheros de datos utilizados en cada algoritmo.

En los ficheros de reglas la diferencia más notoria es la gran diferencia de formatos que presentan los archivos dependiendo del algoritmo, por esto, ha sido necesario implementar el patrón Strategy comentado anteriormente. Además, los ficheros de reglas de algoritmos difusos es necesario que indiquen el número de etiquetas que el algoritmo ha empleado para generar las reglas, sin esta información es imposible realizar la evaluación de las reglas.

Por otro lado, en los ficheros de datos la diferencia es el valor que pueden tomar los atributos. En el caso de los algoritmos difusos, el valor que puede tomar un atributo es un valor dado en un intervalo en el dominio de los reales, mientras que los algoritmos nítidos el valor que toma un atributo viene dado por una lista de etiquetas generada por el algoritmo de discretización empleado antes de aplicar el algoritmo SDRD.

Es importante mencionar que, si bien los casos de prueba se han realizado con ficheros de datos de entrenamiento, la aplicación también es capaz de trabajar con ficheros de datos de tests. Esto permite al usuario realizar visualizaciones y análisis no solo en la fase de entrenamiento, sino también durante la evaluación de los resultados. De esta manera, la aplicación no limita al usuario en términos de la variedad de datos que puede emplear para la ejecución de los algoritmos de descubrimiento de reglas descriptivas con aprendizaje supervisado SDRD, lo que aumenta su utilidad y capacidad de adaptación.

Es importante resaltar que, aunque los casos de prueba se han realizado utilizando conjuntos de datos estáticos, la aplicación también tiene la capacidad de trabajar con datos de *streaming* que han sido recolectados en tiempo real.

Capítulo VI

CONCLUSIONES

Para finalizar, se presentarán las conclusiones generales obtenidas a partir del desarrollo del proyecto y los conocimientos adquiridos durante el desarrollo del mismo. Además, se propondrán ideas para futuros trabajos a partir de la versión actual del sistema.

6.1. Conclusión

La aplicación de escritorio desarrollada en este proyecto es innovadora, la cual destaca por su capacidad de representar de manera gráfica y tabular la evaluación de las reglas descriptivas supervisadas resultantes de la ejecución de un algoritmo SDRD. Esta capacidad de visualización es especialmente provechosa para los investigadores que necesitan analizar los resultados obtenidos a partir de un algoritmo implementado. Aunque es destacable la funcionalidad de la herramienta resultante, no es única en su campo ya que existen otras herramientas como Orange cuya finalidad es similar a la de la aplicación de este proyecto. No obstante, la aplicación desarrollada se distingue por contar con diversos algoritmos SDRD, lo que la hace especialmente útil para el análisis de reglas descriptivas supervisadas obtenidas por estos algoritmos. También, otro aspecto que la hace destacar de las demás herramientas es que no solo permite analizar los resultados de los algoritmos, sino que también ofrece al usuario un análisis de los datos mostrándole la información relevante del conjunto de datos y la relación entre las reglas y los datos.

Además de las características ya mencionadas, es importante destacar que la aplicación se adapta a cualquier algoritmo sin necesidad de que el investigador tenga que adaptarlo para que la aplicación lo detecte, ya que la aplicación es capaz de analizar y evaluar los resultados de múltiples técnicas de descubrimiento de reglas descriptivas con aprendizaje supervisado, lo que la hace especialmente útil para abordar diversas situaciones en el campo de la Ciencia de Datos. Así mismo, su versatilidad se extiende a entornos complejos como *Big Data* y *Data Streaming*, lo que permite a los usuarios trabajar con grandes volúmenes de datos y realizar análisis de datos obtenidos en tiempo real. Esta capacidad de la aplicación es de gran importancia en un contexto en el que el procesamiento de grandes cantidades de información se ha vuelto cada vez más común en distintas áreas de investigación y en la vida cotidiana.

Además de su capacidad para representar gráficamente los resultados, la aplicación también ofrece la posibilidad de exportar la información a diferentes formatos, lo que permite al usuario extraer la información que necesite, facilitando el análisis y divulgación de los resultados obtenidos por el algoritmo.

La solución planteada ha logrado cumplir satisfactoriamente con los objetivos y requisitos establecidos al inicio de la memoria. Se ha conseguido una interfaz accesible, usable y amigable para el usuario, lo que facilita su uso y reduce la curva de aprendizaje para los usuarios finales. Además, el proyecto software desarrollado es escalable y fácil de ampliar para añadir nuevos algoritmos, gracias al uso del patrón Strategy. Esto asegura que la aplicación pueda evolucionar y adaptarse a las necesidades de los usuarios.

6.2. Conocimientos adquiridos

Gracias a la elaboración de este proyecto, he adquirido nuevos conocimientos y habilidades, como el uso de Python, el cual está en pleno auge actualmente en el mundo de la informática. Además, he tenido la oportunidad de poner en práctica todo lo aprendido en el grado sobre interfaces gráficas, prestando especial atención a la usabilidad como factor clave para el éxito de cualquier aplicación.

Otro aspecto que he aprendido durante este proyecto es sobre los algoritmos SDRD, una disciplina de la Ciencia de Datos en vías de desarrollo e investigación y de gran interés en la comunidad, ya que obtiene conocimiento descriptivo mediante aprendizaje supervisado muy interpretable, confiable y explicable que en campos como medicina o desastres naturales es de especial interés. He adquirido nuevos conocimientos sobre la evaluación de los algoritmos SDRD y he profundizado en el funcionamiento de estos algoritmos para conocer, tratar y mostrar la información de la manera más adecuada.

Por otro lado, he podido poner en práctica ciertos conocimientos adquiridos en la asignatura de Programación de Aplicaciones Gráficas, en la cual, aprendí sobre las diferentes herramientas y técnicas que se utilizan para crear aplicaciones gráficas complejas y de alto rendimiento. En particular, he aplicado estos conocimientos para crear una aplicación eficiente y optimizada que pudiera manejar grandes cantidades de datos y visualizarlos de manera rápida y precisa. Gracias a esto, también ha sido posible crear una aplicación escalable y modular que permite agregar nuevas funcionalidades y características con facilidad y eficiencia.

6.3. Futuros trabajos

A pesar de que la aplicación actualmente es funcional y cumple con los requisitos establecidos, siempre hay margen para la mejora y la innovación. En este apartado, se proporcionarán una lista de futuros trabajos que podrían ser implementados en la aplicación para mejorar su funcionalidad y experiencia de usuario:

- Esta aplicación está en constante crecimiento, ya que es ampliable a incluir nuevos algoritmos para visualizar sus resultados. Actualmente, la aplicación cuenta principalmente con algoritmos de descubrimiento de subgrupos, pero se puede considerar la posibilidad de incluir algoritmos de minería de patrones emergentes y minería de conjuntos de contraste, ya que está preparada para incluir cualquier algoritmo SDRD. De esta manera, se ampliaría aún más su funcionalidad y su capacidad de análisis.
- Para ampliar el público que puede utilizar la aplicación de manera sencilla, se pueden realizar ejecutables de la aplicación también para los demás sistemas operativos, Linux y MacOS.
- Para mejorar la capacidad de análisis y visualización de los resultados de los algoritmos SDRD en la aplicación, se puede considerar la inclusión de un nuevo elemento: un árbol jerárquico que represente la relación entre las reglas y las muestre de manera progresiva, de más a menos generales, a medida que se va desglosando el árbol. Este nuevo elemento permitiría a los usuarios explorar y comprender la estructura de las reglas de manera más intuitiva y clara, lo que facilitaría la identificación de patrones y relaciones en los datos. Además, el árbol jerárquico proporcionaría una visualización más detallada de la relación entre las reglas, lo que podría ser de gran utilidad para los usuarios que requieren un mayor nivel de detalle en su análisis.

Rule 1: IF petalWidth = 0 THEN class -> Iris-setosa ☐
Rule 3: IF sepalLength = 0 AND petalWidth = 0 THEN class -> Iris-setosa
Rule 5: IF sepalWidth = 1 AND petalWidth = 0 THEN class -> Iris-setosa

Rule 2: IF petalLength = 0 THEN class -> Iris-setosa ☐

Figura 58. Mockup del árbol jerárquico de reglas. Fuente: Elaboración propia

Anexo I

MANUAL DE INSTALACIÓN

Antes de explicar las distintas maneras de utilizar la aplicación, es necesario obtener el archivo de la misma del repositorio¹³.

Una vez obtenido el archivo, se pueden iniciar la aplicación de dos maneras distintas, las cuales se describen a continuación:

1. Consola:

Para iniciar la aplicación desde la consola, es necesario seguir los siguientes pasos:

1.1. Verificar que se tenga instalado Python versión 3.x.x en el equipo. En caso de no tenerlo instalado, se puede descargar de la página oficial de Python¹⁴ y seleccionar la versión recomendada 3.11.0, ya que es la versión utilizada en la implementación de la aplicación.

1.2. Verificar que se tenga instalada la librería Matplotlib. En caso de no tenerla instalada, se puede instalar utilizando el siguiente comando en la consola:

```
pip install matplotlib
```

1.3. Verificar que se tenga instalada la librería Numpy. En caso de no tenerla instalada, se puede instalar utilizando el siguiente comando en la consola:

```
pip install numpy
```

1.4. Una vez verificado que se tienen instalados Python, Matplotlib y Numpy, se puede iniciar la aplicación desde la consola utilizando el siguiente comando:

```
python [ruta del proyecto]\main.py
```

Donde "ruta del proyecto" es la ruta donde se encuentra la carpeta del proyecto.

Es importante destacar que, para utilizar la aplicación desde la consola, es necesario tener conocimientos básicos de programación y manejo de la consola. Si

¹³ <https://github.com/mariasun-pr/Proyecto>

¹⁴ <https://www.python.org/downloads/>

no se cuenta con estos conocimientos, se recomienda utilizar la otra forma de utilizar la aplicación.

2. Ejecutable

Se ha creado un ejecutable utilizando *pyinstaller* que permite a los usuarios ejecutar una aplicación de escritorio en su sistema operativo Windows sin necesidad de descargar Python o cualquier otra biblioteca adicional.

Para ejecutar la aplicación, el usuario debe seleccionar el directorio correspondiente a su sistema operativo y abrirlo. Seguidamente, debe buscar la carpeta "dist" y abrir el directorio "main". Dentro de este directorio, encontrará un archivo denominado "main.exe" con el icono de la aplicación. Al hacer doble clic sobre este archivo, se iniciará la aplicación.

Cabe destacar que se ha generado un único ejecutable, lo que significa que los usuarios no necesitan preocuparse por la compatibilidad de la biblioteca o la versión de Python. Además, se ha incluido toda la información necesaria en el ejecutable, lo que elimina la necesidad de instalar cualquier biblioteca adicional o realizar configuraciones especiales.

Anexo II

MANUAL DE USUARIO

Este manual se ha creado con el fin de guiar al usuario en el uso de la aplicación de manera clara y detallada, permitiéndole aprovechar todas las funcionalidades que se han diseñado.

Una vez que el usuario haya iniciado la aplicación, a través de cualquiera de las opciones previamente mencionadas. Desde aquí, podrá comenzar a explorar todas las posibilidades que la aplicación le ofrece.

Primero aparecerá la pantalla de inicio. Para acceder a la siguiente pantalla, el usuario deberá pulsar el botón de *nueva visualización*.



Figura 59. Pantalla inicial. Fuente: Elaboración propia

En la siguiente pantalla, el usuario tendrá la posibilidad de seleccionar el conjunto de datos y el fichero de reglas correspondiente a su dispositivo. Para ello, es necesario que el usuario pulse en el botón correspondiente y, automáticamente, se abrirá una ventana emergente de selección de ficheros.



Figura 60. Importación de ficheros inicialmente. Fuente: Elaboración propia

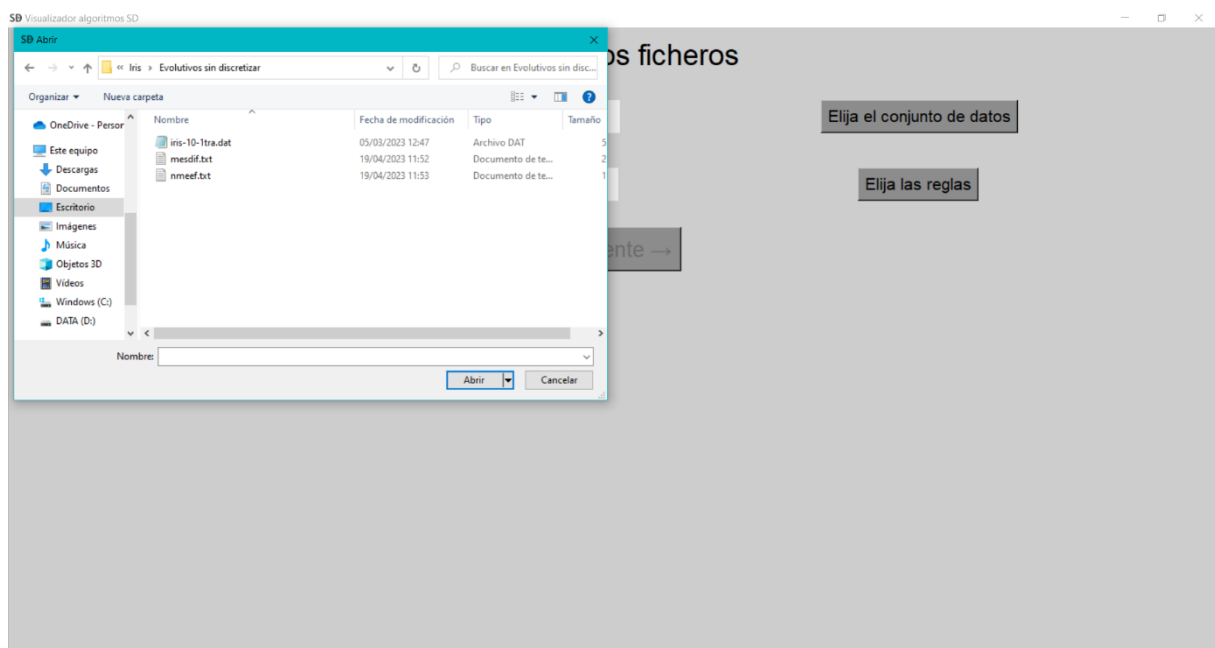


Figura 61. Selección de fichero del conjunto de datos. Fuente: Elaboración propia

Es importante destacar que, en caso de que se presenten errores de formato en los ficheros seleccionados, el usuario recibirá una notificación correspondiente. En este caso, se recomienda revisar los ficheros seleccionados y corregir cualquier problema ya que no se podrá continuar con el proceso.

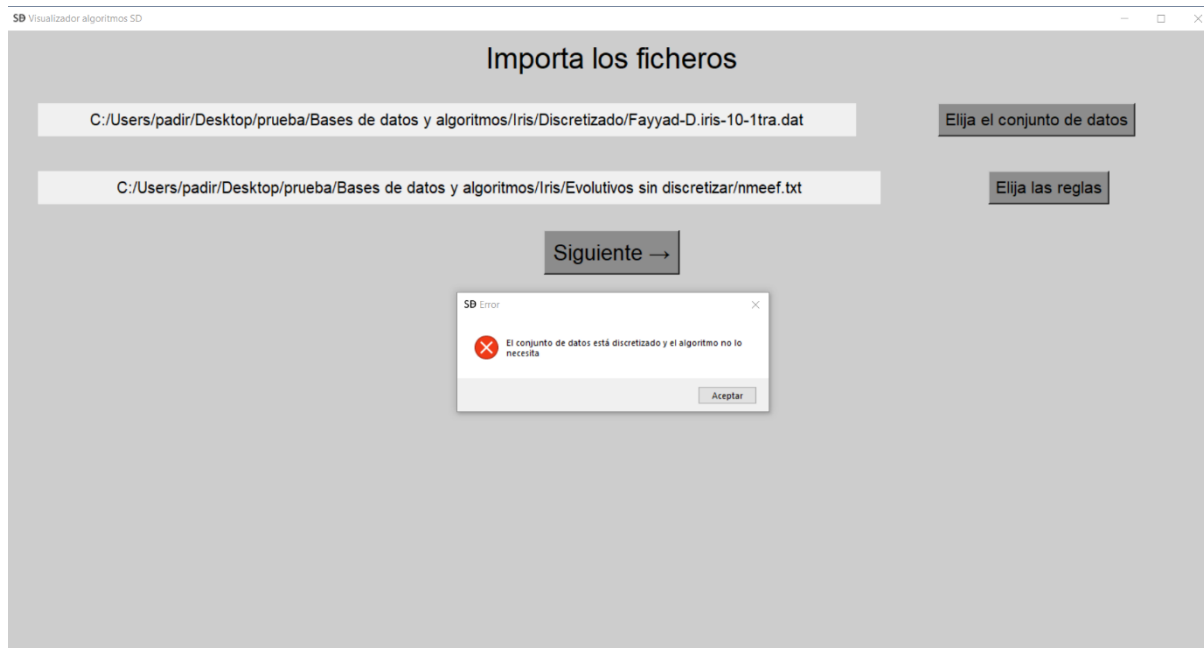


Figura 62. Mensaje de error por no corresponder dataset con reglas. Fuente: Elaboración propia

Si los ficheros seleccionados no presentan errores de formato, el botón "Siguiente" se desbloqueará, permitiendo al usuario avanzar a la siguiente pantalla donde se encuentra la evaluación general de las reglas.



Figura 63. Selección de ficheros realizada. Fuente: Elaboración propia

En la pantalla de evaluación de reglas, el usuario tendrá a su disposición varias opciones para interactuar con la información presentada. Seguidamente, se describen las opciones disponibles:

- Realizar scroll: el usuario podrá desplazarse por toda la información contenida en la pantalla utilizando la barra de desplazamiento.

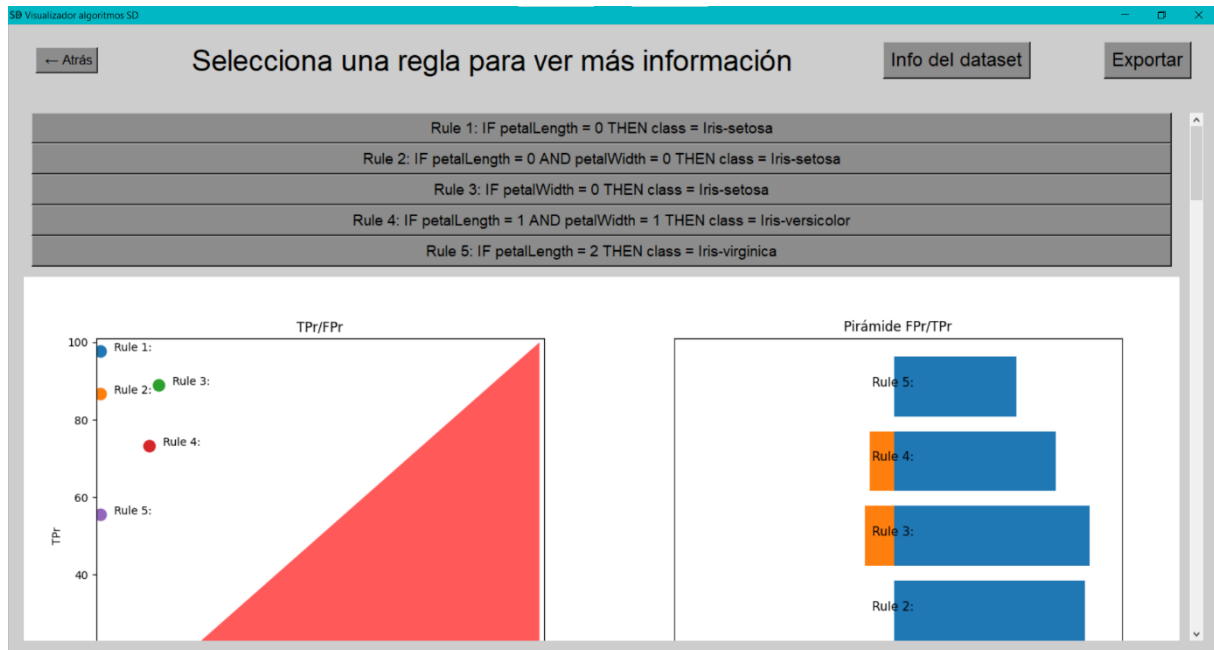


Figura 64. Vista inicial de la información general. Fuente: Elaboración propia

- Ver información del conjunto de datos: pulsando el botón de *Info del dataset*, el usuario podrá acceder a la información de la cabecera del fichero de dataset.

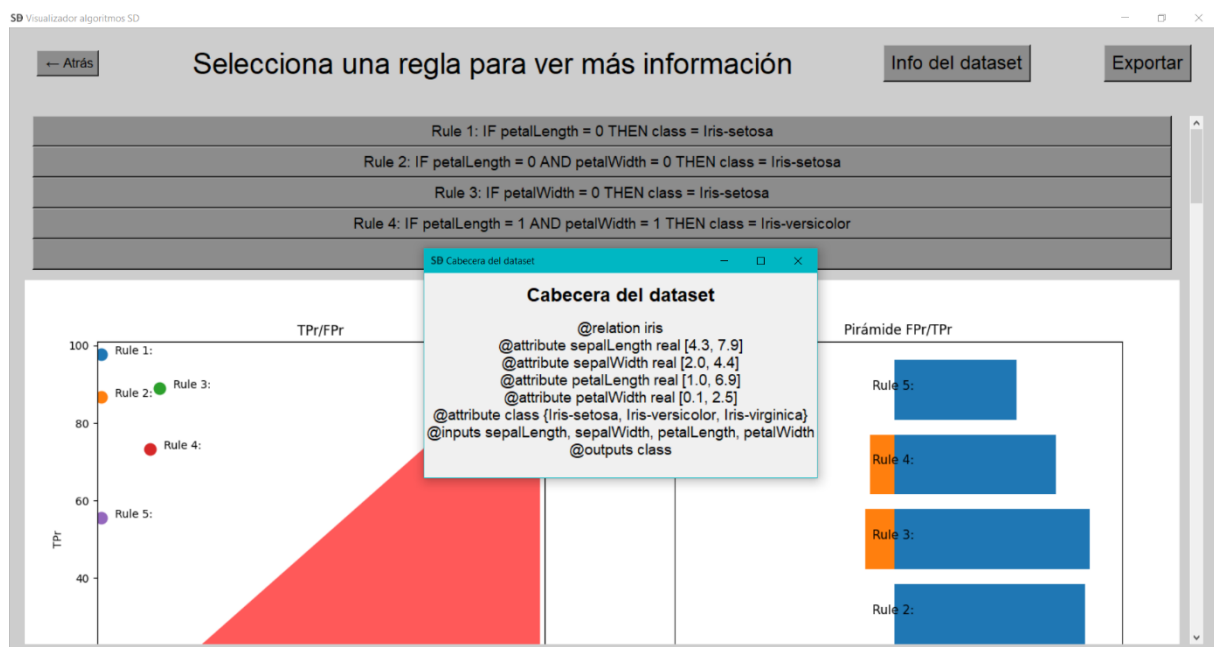


Figura 65. Visualización de la cabecera del conjunto de datos. Fuente: Elaboración propia

- Exportar información: mediante el botón *Exportar*, el usuario tendrá la posibilidad de exportar tanto la información general de la pantalla como la información de cada regla individual en formato ZIP. Al pulsar este botón, se abrirá una ventana emergente donde el usuario podrá seleccionar la ubicación donde desea guardar el archivo ZIP y asignarle un nombre.
- Volver a la pantalla de importación de ficheros: en caso de que el usuario desee volver a la pantalla anterior, podrá hacerlo pulsando el botón *Atrás*.
- Visualizar información de una regla en particular: haciendo clic en una regla específica de la lista de reglas, el usuario podrá acceder a toda la información relacionada con dicha regla.

Al visualizar la información de una regla concreta, el usuario tendrá únicamente las opciones de:

- Hacer scroll de la pantalla para visualizar toda la información relacionada con esa regla.
- Volver a la pantalla anterior.

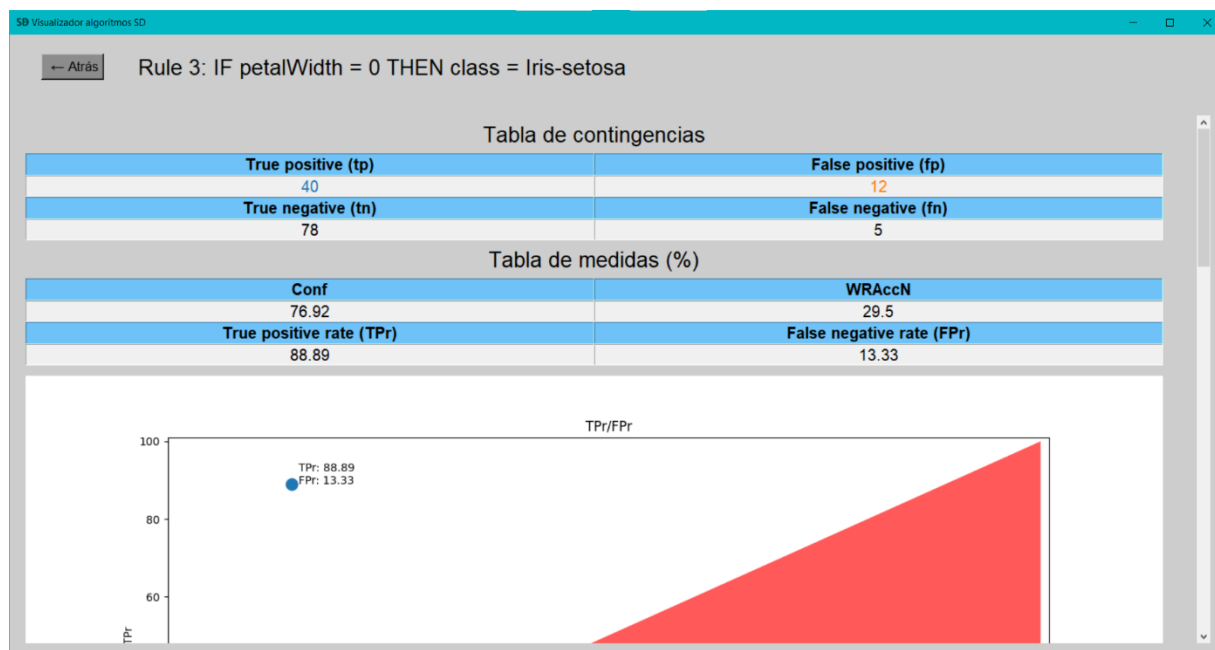


Figura 66. Vista inicial de una única regla. Fuente: Elaboración propia

Cabe destacar que la interfaz de la aplicación ha sido diseñada pensando en el usuario, ofreciendo una experiencia intuitiva y sencilla para que pueda realizar sus tareas de manera eficiente y efectiva.

En cada paso del proceso, la aplicación indica claramente qué es lo que el usuario debe realizar y proporciona un *feedback* inmediato de cada acción que realiza. Por ejemplo, cuando finaliza la exportación de la información, el usuario recibirá una notificación correspondiente. De igual manera, si se presenta algún error en el formato de los ficheros a importar, el usuario será informado de manera oportuna para que pueda corregir el problema. Y, cuando los ficheros seleccionados son correctos, se le indicará al usuario que puede avanzar a la siguiente pantalla.

De este modo, se ha buscado facilitarle al usuario la realización de sus tareas, para que pueda aprovechar al máximo todas las funcionalidades que la herramienta ofrece.

Por último, indicar que también se ha realizado un video tutorial de demostración y uso de la aplicación¹⁵.

¹⁵ <https://drive.google.com/drive/folders/1EFHl8YOmQWXugGCBpKFJpLKjTZxuGKkH?usp=sharing>

Bibliografía

- Atzmueller, M., & Puppe, F. (2006). SD-Map – a Fast Algorithm for Exhaustive Subgroup Discovery. *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 6-17.
- Bay, S. D., & Pazzani, M. J. (2001). Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, 5(3), 213-246. Scopus.
<https://doi.org/10.1023/A:1011429418057>
- Berlanga, F., Del Jesus, M. J., González, P., Herrera, F., & Mesonero, M. (2006). Multiobjective Evolutionary Induction of Subgroup Discovery Fuzzy Rules: A Case Study in Marketing. En P. Perner (Ed.), *Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining* (Vol. 4065, pp. 337-349). Springer Berlin Heidelberg. https://doi.org/10.1007/11790853_27
- Cao, L. (2017). Data Science: A Comprehensive Overview. *ACM Comput. Surv.*, 50(3).
<https://doi.org/10.1145/3076253>
- Carmona, C. J., del Jesus, M. J., & Herrera, F. (2018). A unifying analysis for the supervised descriptive rule discovery via the weighted relative accuracy. *Knowledge-Based Systems*, 139, 89-100. <https://doi.org/10.1016/j.knosys.2017.10.015>
- Carmona, C. J., González, P., Jesus, M. J. D., & Herrera, F. (2010). NMEEF-SD: Non-dominated multiobjective evolutionary algorithm for extracting fuzzy rules in subgroup discovery. *IEEE Transactions on Fuzzy Systems*, 18(5), 958-970. Scopus.
<https://doi.org/10.1109/TFUZZ.2010.2060200>
- Dhar, V. (2013). Data Science and Prediction. *Commun. ACM*, 56(12), 64-73.
<https://doi.org/10.1145/2500499>
- Fayyad, U. M., Piatetsky-Shapiro, G., & Smyth, P. (1996). From Data Mining to Knowledge Discovery: An Overview. En *Advances in Knowledge Discovery and Data Mining* (pp. 1-34). American Association for Artificial Intelligence.
- Gamberger, D., & Lavrac, N. (2002). Expert-Guided Subgroup Discovery: Methodology and Application. *J. Artif. Int. Res.*, 17(1), 501-527.

García-Vico, A. M., Carmona, C. J., Gonzalez, P., Seker, H., & del Jesus, M. J. (2020).

FEPDS: A Proposal for the Extraction of Fuzzy Emerging Patterns in Data Streams.

IEEE Transactions on Fuzzy Systems, 28(12), 3193-3203.

<https://doi.org/10.1109/TFUZZ.2020.2992849>

García-Vico, A. M., Carmona, C. J., Martín, D., García-Borroto, M., & del Jesus, M. J. (2018).

An overview of emerging pattern mining in supervised descriptive rule discovery:

Taxonomy, empirical study, trends, and prospects: Emerging pattern mining. *Wiley*

Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(1), e1231.

<https://doi.org/10.1002/widm.1231>

Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques (The Morgan*

Kaufmann Series in Data Management Systems) (3.^a ed.).

[https://www.amazon.com/Data-Mining-Concepts-Techniques-](https://www.amazon.com/Data-Mining-Concepts-Techniques-Management/dp/0123814790)

[Management/dp/0123814790](https://www.amazon.com/Data-Mining-Concepts-Techniques-Management/dp/0123814790)

Hernández Orallo, J., Ramírez Quintana, M. J., & Ramírez Ferri, C. (2010). *Introducción a la*

minería de datos (Reimp). Pearson.

Herrera, F., Carmona, C. J., González, P., & del Jesus, M. J. (2011). An overview on

subgroup discovery: Foundations and applications. *Knowledge and Information*

Systems, 29(3), 495-525. <https://doi.org/10.1007/s10115-010-0356-2>

JavaFX web. (s. f.). <https://openjfx.io/>

Jesus, M. J. del, González, P., Herrera, F., & Mesonero, M. (2007). Evolutionary fuzzy rule

induction process for subgroup discovery: A case study in marketing. *IEEE*

Transactions on Fuzzy Systems, 15(4), 578-592.

Kavšek, B., & Lavrač, N. (2006). APRIORI-SD: ADAPTING ASSOCIATION RULE

LEARNING TO SUBGROUP DISCOVERY. *Applied Artificial Intelligence*, 20(7), 543-

583. <https://doi.org/10.1080/08839510600779688>

KEEL web. (s. f.). <https://sci2s.ugr.es/keel/description.php>

Klösgen, W. (1996). Explora: A Multipattern and Multistrategy Discovery Assistant. En
Advances in Knowledge Discovery and Data Mining (pp. 249-271). American
Association for Artificial Intelligence.

Lavrac, N., Kavšek, B., Flach, P., & Todorovski, L. (2004). Subgroup discovery with CN2-SD.
Journal of Machine Learning Research, 5, 153-188.

Matplotlib web. (s. f.). <https://pypi.org/project/matplotlib/>

Numpy web. (s. f.). <https://numpy.org/>

Orange web. (s. f.). <https://orangedatamining.com/>

Python web. (s. f.). [Python]. <https://www.python.org/>

Qt web. (s. f.). <https://www.qt.io/>

Re web. (s. f.). <https://docs.python.org/3/library/re.html>

Salario medio para Ingeniero informático en España 2023. (2023, junio). *talent.com*.

<https://es.talent.com/salary?job=ingeniero%2Binform%C3%A1tico>

Vương Quý, P. (2022, junio 22). Foto tkinter. *Zendvn*. <https://zendvn.com/khoa-hoc-lap-trinh-phan-mem-voi-tkinter-python>

Weka web. (s. f.). *Weka*. <https://www.cs.waikato.ac.nz/ml/weka/>

Wrobel, S. (1997). An Algorithm for Multi-Relational Discovery of Subgroups. *Proceedings of
the First European Symposium on Principles of Data Mining and Knowledge
Discovery*, 78-87.

Zipfile web. (s. f.). <https://docs.python.org/3/library/zipfile.html>