

**ESCUELA POLITÉCNICA  
SUPERIOR DE CÓRDOBA**  
Universidad de Córdoba



**TRABAJO FIN DE GRADO**

*Grado en Ingeniería Informática  
Mención en Ingeniería del Software*

**Librería de Detección de Anomalías en Python**

**Autor:** Álvaro Pino Mérida  
**Directora:** Amelia Zafra Gómez

*Manual Técnico*

**julio, 2024**



UNIVERSIDAD DE CÓRDOBA



*Dedicado a  
mi familia*



# Agradecimientos

Quisiera expresar mi más sincero agradecimiento a todas las personas que han contribuido de manera significativa a la realización de este Trabajo de Fin de Grado.

En primer lugar, agradezco a mi tutora, Amelia Zafra Gómez, por su invaluable guía, paciencia y apoyo continuo durante todo el proceso. Sus sugerencias y comentarios han sido esenciales para la elaboración de este trabajo.

A mi familia, por su amor incondicional y su apoyo inquebrantable. A mis padres, por creer siempre en mí y proporcionarme las herramientas necesarias para alcanzar mis metas. A mi hermana, por estar siempre a mi lado y celebrar juntos cada logro.

A mis amigos, por todos los momentos compartidos durante estos años, por todo lo que hemos aprendido, todo lo que hemos crecido y por ser una fuente constante de apoyo y alegría.

A todos aquellos que, de una manera u otra, han estado presentes en esta etapa de mi vida, muchas gracias.



# Índice general

|   |           |
|---|-----------|
| Índice de figuras                       | X         |
| Índice de tablas                        | XIII      |
| <b>1. Introducción</b>                  | <b>1</b>  |
| <b>2. Definición del Problema</b>       | <b>3</b>  |
| 2.1. Problema Real . . . . .            | 3         |
| 2.2. Problema Técnico . . . . .         | 4         |
| 2.2.1. Funcionamiento . . . . .         | 4         |
| 2.2.2. Entorno . . . . .                | 5         |
| 2.2.3. Vida Esperada . . . . .          | 6         |
| 2.2.4. Ciclo de Mantenimiento . . . . . | 6         |
| 2.2.5. Competencia . . . . .            | 6         |
| 2.2.6. Aspecto Externo . . . . .        | 7         |
| 2.2.7. Estandarización . . . . .        | 7         |
| 2.2.8. Calidad y Fiabilidad . . . . .   | 7         |
| 2.2.9. Programa de Tareas . . . . .     | 7         |
| 2.2.10. Pruebas . . . . .               | 9         |
| 2.2.11. Seguridad . . . . .             | 9         |
| <b>3. Objetivos</b>                     | <b>10</b> |
| <b>4. Antecedentes</b>                  | <b>13</b> |
| <b>5. Restricciones</b>                 | <b>15</b> |
| 5.1. Factores dato . . . . .            | 15        |
| 5.2. Factores estratégicos . . . . .    | 15        |
| <b>6. Recursos</b>                      | <b>17</b> |
| 6.1. Recursos Humanos . . . . .         | 17        |
| 6.2. Recursos software . . . . .        | 17        |
| 6.3. Recursos hardware . . . . .        | 17        |



|   |           |
|---|-----------|
| <b>7. Especificación de Requisitos</b>                    | <b>18</b> |
| 7.1. Requisitos de Información . . . . .                  | 18        |
| 7.2. Requisitos Funcionales . . . . .                     | 19        |
| 7.3. Requisitos No Funcionales . . . . .                  | 21        |
| <b>8. Análisis funcional</b>                              | <b>22</b> |
| 8.1. Casos de uso . . . . .                               | 22        |
| 8.1.1. Actores . . . . .                                  | 22        |
| 8.1.2. Especificación de Caso de Uso . . . . .            | 23        |
| 8.1.3. Matriz de trazabilidad . . . . .                   | 31        |
| 8.2. Descripción del Comportamiento . . . . .             | 31        |
| <b>9. Diseño del sistema</b>                              | <b>38</b> |
| 9.1. Arquitectura . . . . .                               | 38        |
| 9.1.1. Dominio . . . . .                                  | 40        |
| 9.1.2. Aplicación . . . . .                               | 42        |
| 9.1.3. Infraestructura . . . . .                          | 46        |
| 9.2. Descripción del modelo conceptual de datos . . . . . | 47        |
| 9.3. Descripción de las clases . . . . .                  | 50        |
| 9.3.1. Clase AutoEncoder . . . . .                        | 50        |
| 9.3.2. Clase Cblof . . . . .                              | 50        |
| 9.3.3. Clase Cof . . . . .                                | 51        |
| 9.3.4. Clase Dae . . . . .                                | 51        |
| 9.3.5. Clase Hbos . . . . .                               | 52        |
| 9.3.6. Clase Knn . . . . .                                | 52        |
| 9.3.7. Clase Lof . . . . .                                | 53        |
| 9.3.8. Clase LstmVae . . . . .                            | 53        |
| 9.3.9. Clase CblofTrainer . . . . .                       | 54        |
| 9.3.10. Clase CofTrainer . . . . .                        | 54        |
| 9.3.11. Clase HbosTrainer . . . . .                       | 54        |
| 9.3.12. Clase KnnTrainer . . . . .                        | 54        |
| 9.3.13. Clase LofTrainer . . . . .                        | 54        |
| 9.3.14. Clase DaeTrainer . . . . .                        | 55        |
| 9.3.15. Clase AutoEncoderTrainer . . . . .                | 55        |
| 9.3.16. Clase LstmVaeTrainer . . . . .                    | 55        |
| 9.3.17. Clase Cblof . . . . .                             | 55        |
| 9.3.18. Clase Cof . . . . .                               | 55        |
| 9.3.19. Clase Hbos . . . . .                              | 56        |
| 9.3.20. Clase Knn . . . . .                               | 56        |
| 9.3.21. Clase Lof . . . . .                               | 56        |



|  |            |
|--|------------|
| 9.3.22. Clase Dae . . . . .                                | 56         |
| 9.3.23. Clase AutoEncoder . . . . .                        | 56         |
| 9.3.24. Clase LstmVae . . . . .                            | 57         |
| 9.3.25. Clase AlgorithmFactory . . . . .                   | 57         |
| 9.3.26. Clase AlgorithmCommandFactory . . . . .            | 57         |
| 9.3.27. Clase AlgorithmManager . . . . .                   | 57         |
| 9.3.28. Clase AlgorithmConfigurator . . . . .              | 58         |
| 9.3.29. Clase AlgorithmTrainer . . . . .                   | 58         |
| 9.3.30. Clase AlgorithmEvaluate . . . . .                  | 58         |
| 9.3.31. Clase PyodWrapper . . . . .                        | 58         |
| 9.3.32. Clase FileSystemService . . . . .                  | 59         |
| 9.3.33. Clase TimeEvalWrapper . . . . .                    | 59         |
| 9.3.34. Clase TrainRepository . . . . .                    | 59         |
| 9.3.35. Clase AlgorithmDataProcessor . . . . .             | 59         |
| 9.3.36. Clase BasicReport . . . . .                        | 60         |
| 9.3.37. Clase ReportRepository . . . . .                   | 60         |
| <b>10.Resultados experimentales</b>                        | <b>61</b>  |
| 10.1. Marco experimental . . . . .                         | 62         |
| 10.1.1. Conjunto de datos . . . . .                        | 62         |
| 10.1.2. Estructura de la información . . . . .             | 62         |
| 10.2. Procesamiento del conjunto de datos . . . . .        | 70         |
| 10.3. Análisis de la información almacenada . . . . .      | 73         |
| 10.4. Particionamiento de los datos . . . . .              | 77         |
| 10.5. Configuración de los Algoritmos . . . . .            | 78         |
| 10.6. Resultados de la experimentación . . . . .           | 79         |
| 10.6.1. Comparación parámetros de los algoritmos . . . . . | 79         |
| 10.6.2. Comparación de algoritmos por cursos . . . . .     | 113        |
| <b>11.Pruebas</b>  | <b>119</b> |
| 11.1. Proceso de Pruebas y Validación . . . . .            | 119        |
| 11.1.1. Pruebas Unitarias . . . . .                        | 119        |
| 11.1.2. Pruebas Funcionales . . . . .                      | 120        |
| 11.1.3. Pruebas de Integración . . . . .                   | 120        |
| 11.2. Errores y Soluciones Durante las Pruebas . . . . .   | 121        |
| 11.3. Conclusión de las Pruebas . . . . .                  | 121        |
| <b>12.Conclusiones</b>                                     | <b>122</b> |
| 12.1. Análisis de objetivos . . . . .                      | 122        |
| 12.2. Futuras mejoras . . . . .                            | 124        |

## Bibliografía

125



# Índice de figuras

|     |  |    |
|-----|--|----|
| 1.  | Diagrama de casos de uso del sistema . . . . .                           | 22 |
| 2.  | Matriz de trazabilidad . . . . .   | 31 |
| 3.  | Diagrama de secuencia de CU-1 Entrenar modelo . . . . .                  | 32 |
| 4.  | Diagrama de secuencia de CU-2 Ejecutar modelo . . . . .                  | 33 |
| 5.  | Diagrama de secuencia de CU-3 Crear informe . . . . .                    | 34 |
| 6.  | Diagrama de secuencia de CU-4 Configurar modelo . . . . .                | 34 |
| 7.  | Diagrama de secuencia de CU-5 Particionar datos . . . . .                | 35 |
| 8.  | Diagrama de secuencia de CU-6 Transformar datos . . . . .                | 35 |
| 9.  | Diagrama de secuencia de CU-7 Aplicar modelo . . . . .                   | 36 |
| 10. | Diagrama de secuencia de CU-8 Realizar estudio experimental . . . . .    | 37 |
| 1.  | Diseño de la arquitectura hexagonal . . . . .                            | 38 |
| 2.  | Diagrama de clases completo del sistema . . . . .                        | 39 |
| 3.  | Ejemplo de fichero de configuración . . . . .                            | 45 |
| 4.  | Diagrama de clases referente al patrón comando del sistema . . . . .     | 46 |
| 5.  | Modelo de datos de los algoritmos . . . . .                              | 47 |
| 6.  | Diagrama de clases Cblof . . . . .                                       | 49 |
| 1.  | Estructura del dataset (Figura obtenida de [1]) . . . . .                | 63 |
| 2.  | Extracto del fichero docker-compose . . . . .                            | 72 |
| 3.  | Estudio parámetros del algoritmo Autoencoder para el curso AAA . . . . . | 80 |
| 4.  | Estudio parámetros del algoritmo CBLOF para el curso AAA . . . . .       | 81 |
| 5.  | Estudio parámetros del algoritmo COF para el curso AAA . . . . .         | 82 |
| 6.  | Estudio parámetros del algoritmo DAE para el curso AAA . . . . .         | 82 |
| 7.  | Estudio parámetros del algoritmo HBOS para el curso AAA . . . . .        | 83 |
| 8.  | Estudio parámetros del algoritmo Knn para el curso AAA . . . . .         | 84 |
| 9.  | Estudio parámetros del algoritmo LOF para el curso AAA . . . . .         | 84 |
| 10. | Estudio parámetros del algoritmo Autoencoder para el curso BBB . . . . . | 85 |
| 11. | Estudio parámetros del algoritmo CBLOF para el curso BBB . . . . .       | 86 |
| 12. | Estudio parámetros del algoritmo COF para el curso BBB . . . . .         | 87 |
| 13. | Estudio parámetros del algoritmo DAE para el curso BBB . . . . .         | 87 |
| 14. | Estudio parámetros del algoritmo HBOS para el curso BBB . . . . .        | 88 |
| 15. | Estudio parámetros del algoritmo Knn para el curso BBB . . . . .         | 89 |
| 16. | Estudio parámetros del algoritmo LOF para el curso BBB . . . . .         | 89 |
| 17. | Estudio parámetros del algoritmo Autoencoder para el curso CCC . . . . . | 90 |



|     |  |     |
|-----|--|-----|
| 18. | Estudio parámetros del algoritmo CBLOF para el curso CCC . . . . .                             | 91  |
| 19. | Estudio parámetros del algoritmo COF para el curso CCC . . . . .                               | 91  |
| 20. | Estudio parámetros del algoritmo DAE para el curso CCC . . . . .                               | 92  |
| 21. | Estudio parámetros del algoritmo HBOS para el curso CCC . . . . .                              | 93  |
| 22. | Estudio parámetros del algoritmo Knn para el curso CCC . . . . .                               | 93  |
| 23. | Estudio parámetros del algoritmo LOF para el curso CCC . . . . .                               | 94  |
| 24. | Estudio parámetros del algoritmo Autoencoder para el curso DDD . .                             | 95  |
| 25. | Estudio parámetros del algoritmo CBLOF para el curso DDD . . . . .                             | 96  |
| 26. | Estudio parámetros del algoritmo COF para el curso DDD . . . . .                               | 96  |
| 27. | Estudio parámetros del algoritmo DAE para el curso DDD . . . . .                               | 97  |
| 28. | Estudio parámetros del algoritmo HBOS para el curso DDD . . . . .                              | 98  |
| 29. | Estudio parámetros del algoritmo Knn para el curso DDD . . . . .                               | 98  |
| 30. | Estudio parámetros del algoritmo LOF para el curso DDD . . . . .                               | 99  |
| 31. | Estudio parámetros del algoritmo Autoencoder para el curso EEE . .                             | 100 |
| 32. | Estudio parámetros del algoritmo CBLOF para el curso EEE . . . . .                             | 100 |
| 33. | Estudio parámetros del algoritmo COF para el curso EEE . . . . .                               | 101 |
| 34. | Estudio parámetros del algoritmo DAE para el curso EEE . . . . .                               | 102 |
| 35. | Estudio parámetros del algoritmo HBOS para el curso EEE . . . . .                              | 102 |
| 36. | Estudio parámetros del algoritmo Knn para el curso EEE . . . . .                               | 103 |
| 37. | Estudio parámetros del algoritmo LOF para el curso EEE . . . . .                               | 104 |
| 38. | Estudio parámetros del algoritmo Autoencoder para el curso FFF . .                             | 105 |
| 39. | Estudio parámetros del algoritmo CBLOF para el curso FFF . . . . .                             | 105 |
| 40. | Estudio parámetros del algoritmo COF para el curso FFF . . . . .                               | 106 |
| 41. | Estudio parámetros del algoritmo DAE para el curso FFF . . . . .                               | 107 |
| 42. | Estudio parámetros del algoritmo HBOS para el curso FFF . . . . .                              | 107 |
| 43. | Estudio parámetros del algoritmo Knn para el curso FFF . . . . .                               | 108 |
| 44. | Estudio parámetros del algoritmo LOF para el curso FFF . . . . .                               | 108 |
| 45. | Estudio parámetros del algoritmo Autoencoder para el curso GGG . .                             | 109 |
| 46. | Estudio parámetros del algoritmo CBLOF para el curso GGG . . . . .                             | 110 |
| 47. | Estudio parámetros del algoritmo COF para el curso GGG . . . . .                               | 111 |
| 48. | Estudio parámetros del algoritmo DAE para el curso GGG . . . . .                               | 111 |
| 49. | Estudio parámetros del algoritmo HBOS para el curso GGG . . . . .                              | 112 |
| 50. | Estudio parámetros del algoritmo Knn para el curso GGG . . . . .                               | 113 |
| 51. | Estudio parámetros del algoritmo LOF para el curso GGG . . . . .                               | 113 |
| 52. | Comparativa mejor configuración por algoritmo y curso . . . . .                                | 114 |
| 53. | Distancia crítica del post-test de Shaffer's para los resultados de<br>sensibilidad . . . . .  | 117 |
| 54. | Distancia crítica del post-test de Shaffer's para los resultados de<br>especificidad . . . . . | 118 |



# Índice de tablas

|     |  |     |
|-----|--|-----|
| 1.  | Comparación de Características de Librerías de Detección de Anomalías      | 14  |
| 1.  | CU-1 Entrenar modelo   | 23  |
| 2.  | CU-2 Ejecutar modelo   | 24  |
| 3.  | CU-3 Crear informe   | 25  |
| 4.  | CU-4 Configurar modelo   | 26  |
| 5.  | CU-5 Particionar datos   | 27  |
| 6.  | CU-6 Transformar datos   | 28  |
| 7.  | CU-7 Aplicar modelo  | 29  |
| 8.  | CU-8 Realizar estudio experimental   | 30  |
| 1.  | Tabla de cursos  | 63  |
| 2.  | Tabla de entregas  | 64  |
| 3.  | Tabla de recursos  | 65  |
| 4.  | Tabla de información del estudiante  | 66  |
| 5.  | Tabla de registro  | 67  |
| 6.  | Tabla de resultados en evaluaciones  | 68  |
| 7.  | Tabla de interacción del alumno con los recursos                           | 69  |
| 8.  | Análisis de los datos por módulo y presentación                            | 74  |
| 9.  | Análisis de los datos por módulo   | 74  |
| 10. | Cantidad de Alumnos y Anomalías por Curso                                  | 75  |
| 11. | Resultados medios de sensibilidad para cada algoritmo y conjunto de datos  | 116 |
| 12. | Resultados medios de especificidad para cada algoritmo y conjunto de datos | 117 |
| 1.  | Resultados de las pruebas unitarias  | 119 |
| 2.  | Resultados de las pruebas funcionales                                      | 120 |
| 3.  | Resultados de las pruebas de integración                                   | 120 |







# 1 Introducción

En la era actual, caracterizada por una digitalización masiva y una generación de datos sin precedentes, la necesidad de métodos eficientes para la detección de anomalías ha cobrado una importancia crucial. Esta relevancia se evidencia en sectores tan diversos como el bancario, donde la detección temprana de fraudes es vital [2], o en la seguridad de infraestructuras críticas [3], donde la pronta identificación de ciberataques puede ser determinante. La detección de anomalías, según Douglas M. Hawkins en su obra “Identification of Outliers” [4], implica identificar observaciones que se desvían significativamente de la norma, sugiriendo una generación por mecanismos atípicos.

Detectar tales anomalías es crucial no solo para identificar errores, sino también para prevenir actividades fraudulentas o peligrosas. En el ámbito técnico, la detección de anomalías se aborda mediante una variedad de algoritmos, cada uno con sus propias fortalezas en términos de eficacia y coste computacional. Si bien existe un número representativo de propuestas que intentan resolver el problema [5] y diferentes librerías como PyOd [6] y Anomaly Detection Toolbox [7]. Nos encontramos con un importante desafío: la falta de una convención unificada para las entradas y salidas de datos entre las diferentes librerías y propuestas existentes. Esta diversidad en los métodos de implementación y las librerías complica su uso e integración en estudios experimentales.

Por lo tanto, la propuesta central de este Trabajo Fin de Grado (TFG) es el desarrollo de una librería en Python que unifique diferentes propuestas de detección de anomalías bajo una capa de abstracción común. Esta librería no solo facilitará la ejecución de diversos algoritmos provenientes de distintas fuentes, sino que también proporcionará una estandarización en la salida de datos y en el análisis de resultados. Esto permitirá a los investigadores y profesionales simplificar el proceso de experimentación y análisis de propuestas previas. Para lograr esto, se empleará una metodología basada en la arquitectura limpia [8], garantizando que la integración de diferentes herramientas de detección de anomalías sea sencilla y robusta. Además, se llevará a cabo un tutorial detallado que permita a los usuarios implementar las diferentes metodologías y algoritmos bajo esta nueva librería unificada. Este enfoque integrador y estandarizado no solo simplificará el proceso de implementación de métodos de detección de anomalías, sino que también abrirá nuevas posibilidades para la experimentación y la innovación en este campo crítico y en constante



## CAPÍTULO 1. INTRODUCCIÓN

---

evolución. Se espera que esta librería tenga un impacto significativo en el ámbito académico, facilitando la investigación de los sistemas de detección de anomalías.

Finalmente, se llevará a cabo un estudio experimental con la finalidad de mostrar cómo se puede identificar y manejar las anomalías en un contexto práctico. Para ello, se analizará la resolución de un problema real, abordándolo desde la perspectiva de la detección de anomalías. Este estudio permitirá demostrar de manera concreta la utilidad y efectividad de la librería para llevar a cabo este tipo de estudios, mostrando sus beneficios en la práctica.



## 2 Definición del Problema

### 2.1. Problema Real

La detección de anomalías es un desafío presente en numerosos campos debido al volumen creciente de datos y la necesidad de identificar eventos atípicos que pueden indicar errores, fraudes o amenazas. En sectores como el bancario, la identificación temprana de actividades fraudulentas es crucial para la seguridad financiera de los clientes y las instituciones. Del mismo modo, en la seguridad de infraestructuras críticas, la detección temprana de ciberataques puede prevenir daños significativos. En sectores industriales, la detección de anomalías es fundamental para la optimización de procesos y el mantenimiento predictivo, permitiendo anticipar fallos y mejorar la eficiencia operativa.

Debido a su utilidad en muchos ámbitos, ha habido un crecimiento considerable en los últimos años de desarrollo de nuevos modelos que resuelvan este problema. Sin embargo, estos modelos a menudo se encuentran disponibles en repositorios de forma aislada y las librerías disponibles presentan diferentes limitaciones. Estos hechos dificultan el proceso de llevar a cabo un estudio comparativo entre propuestas previas.

En el presente TFG, se propone el desarrollo de un framework unificado que estandarice tanto las entradas como las salidas de los datos. Además, se incluirán diversas propuestas que abarcan diferentes tipos de modelos de aprendizaje automático. El objetivo principal es crear una librería que permita una comparación eficiente y directa entre los distintos algoritmos de detección de anomalías. Este framework está diseñado para facilitar la evaluación y el análisis de los resultados, proporcionando un entorno coherente y accesible para la implementación y comparación de estos algoritmos.



## 2.2. Problema Técnico

Para el desarrollo del TFG se seguirá la metodología de Desarrollo de Software (PDS [9]), que incluye las etapas de planificación, diseño, implementación, verificación y mantenimiento. Esta metodología garantiza un enfoque estructurado y sistemático para la creación del framework, asegurando su calidad y funcionalidad a lo largo del proyecto.

### 2.2.1. Funcionamiento

El funcionamiento de esta librería se basa en una estructura modular que permite la integración sencilla de distintos algoritmos de detección de anomalías. Los componentes principales incluyen:

- **Módulo de preprocesamiento de datos:** Se encarga de preparar las entradas de los algoritmos, incluyendo la definición del formato de datos de entrada (que será detallado en secciones posteriores del documento) y los métodos de transformación de formatos. Esto permite la adaptación de varios formatos de datos al formato estándar utilizado por la librería.
- **Interfaz común para algoritmos:** Proporciona una interfaz unificada para la implementación de distintos algoritmos de detección de anomalías. Esta interfaz facilita la integración y el intercambio de algoritmos sin modificar la estructura subyacente del sistema.
- **Métodos de detección de anomalías:** Implementa diversos métodos de detección, permitiendo la incorporación de algoritmos provenientes de diferentes librerías. Se han incluido métodos *wrapper* para utilizar algoritmos de dos librerías diferentes, promoviendo la realización de estudios experimentales dentro de un mismo *framework*.
- **Sistema de estandarización de salidas y resultados:** Define un formato estándar para la salida y los resultados de los algoritmos de detección de anomalías, garantizando la coherencia y facilitando la comparación entre distintos métodos.
- **Métodos de validación cruzada:** Incluye métodos para aplicar la validación cruzada en los conjuntos de datos, asegurando la fiabilidad y la robustez de los resultados de los algoritmos.
- **Uso de ficheros de configuración:** Facilita la tarea de ejecución de los algoritmos y los estudios experimentales a través de ficheros de configuración.



personalizables, permitiendo una configuración flexible y específica según las necesidades del usuario.

- **Módulo de informes de resultados:** Genera informes detallados de los resultados obtenidos por los distintos algoritmos, proporcionando una visión comprensiva del rendimiento y las características de cada método.

### 2.2.2. Entorno

El entorno de programación para la implementación de la librería se lleva a cabo utilizando Visual Studio Code [10], con Python como lenguaje de programación principal. Para su correcta configuración, es necesario instalar una serie de librerías y dependencias especificadas en el código fuente.

En cuanto al hardware, la librería puede ejecutarse en cualquier dispositivo con una capacidad computacional básica. Sin embargo, debido a la naturaleza intensiva de las operaciones que realiza, se recomienda utilizar equipos con mayor potencia para reducir los tiempos de ejecución.

Respecto al software, es imprescindible contar con Python instalado para ejecuciones no encapsuladas. Alternativamente, se puede emplear Docker para ejecutar la versión dockerizada de la librería.

El entorno de usuario se puede dividir en dos categorías:

- **Entorno de desarrollo:** Adecuado para usuarios que deseen integrar la librería con otros procesos o personalizar su funcionamiento.
- **Entorno Docker:** Ideal para usuarios que únicamente necesitan generar informes, proporcionando una solución encapsulada y fácil de desplegar.

Este entorno permite una implementación flexible y eficiente, adaptándose a diversas necesidades y capacidades técnicas.

El entorno de ejecución de la librería será cualquier sistema compatible con Docker, ejecutándose en Python aprovechando el extenso ecosistema de librerías científicas y de análisis de datos. Esto nos permite, por un lado, la ventaja del uso de un ecosistema ya usado por personas del entorno y por otro la consistencia de versiones que nos proporciona Docker.



### 2.2.3. Vida Esperada

La librería está diseñada para ser flexible y extensible, permitiendo su adaptación a futuras necesidades y tecnologías emergentes en el campo de la detección de anomalías. Se determina, por tanto, que la vida del proyecto será extensa y debe seguir activa mientras sea de utilidad para su propósito. Además, se plantea la posibilidad de ampliarla con nuevas propuestas que le permitan incluir y representar el estado del arte en el área, asegurando así su relevancia y efectividad continuas en un entorno tecnológico en constante evolución.

### 2.2.4. Ciclo de Mantenimiento

Para mantener la librería en un estado óptimo para su uso, se recogerán las opiniones de los usuarios de la misma y se harán actualizaciones recurrentes. El mantenimiento de la librería incluirá:

- Actualizaciones periódicas para garantizar la compatibilidad con nuevas versiones de Python y librerías dependientes.
- Corrección de errores y mejoras basadas en el feedback de los usuarios.
- Adición de nuevos algoritmos y funcionalidades según sea necesario.

### 2.2.5. Competencia

Existen varias librerías y herramientas dedicadas a la detección de anomalías, como AnomalyDetectionToolbox [7] entre otras que se citarán más adelante en antecedentes. Sin embargo, ninguna ofrece una solución unificada que facilite la ejecución y estandarización de múltiples algoritmos de diferentes fuentes. Esta librería pretende llenar ese vacío, ofreciendo una plataforma robusta.

A diferencia de las propuestas actuales, que se limitan a ofrecer implementaciones específicas, la presente librería se enfoca en la integración y normalización de diversas herramientas de detección de anomalías disponibles en el mercado. Mediante el desarrollo de métodos wrapper, se permite el uso de estas herramientas dentro de un mismo framework, simplificando su aplicación y extendiendo sus capacidades.



### 2.2.6. Aspecto Externo

La comunicación con la persona que utilice la librería se realizará principalmente a través de la línea de comandos. Los usuarios podrán configurar los algoritmos y características de la librería mediante un fichero de configuración, lo cual permitirá utilizarla sin necesidad de modificar el código fuente.

La librería será distribuida a través de GitHub, facilitando su acceso e instalación por parte de los usuarios. Además, se proporcionará documentación detallada y tutoriales para facilitar su uso y adopción.

### 2.2.7. Estandarización

El código seguirá las mejores prácticas de desarrollo en Python, incluyendo PEP 8 (Python Enhancement Proposals - Style Guide for Python Code) [11] para estilo de código y convenciones de nombres. Se utilizarán herramientas de linting y formateo automático para mantener la consistencia y legibilidad del código, en este caso black [12].

### 2.2.8. Calidad y Fiabilidad

La calidad y fiabilidad son dos elementos con mucho peso dentro del desarrollo, es por ello que se ha prestado especial atención mediante unos procedimientos fiables. Para garantizar la calidad y fiabilidad de la librería, se ha tenido especial cuidado en la arquitectura de la misma. Cada elemento adquiere una responsabilidad que debe de cumplir y mediante esta separación de responsabilidades se pueden encapsular estos problemas concretos y tener una trazabilidad de errores mucho más limpia.

### 2.2.9. Programa de Tareas

La metodología SCRUM [13] será empleada para la planificación y desarrollo del proyecto, facilitando iteraciones rápidas y flexibles. Las etapas del proyecto se describen a continuación:

**Fase de Investigación Preliminar:** Inicialmente, se llevará a cabo un estudio preliminar para obtener una comprensión clara del problema y determinar las herramientas necesarias para abordarlo eficientemente.

**Creación de la Biblioteca:** Una vez realizado el estudio inicial, se procederá a desarrollar una biblioteca en Python que apoye la implementación futura del



## CAPÍTULO 2. DEFINICIÓN DEL PROBLEMA

---

proyecto. Esta fase incluirá la configuración del entorno de desarrollo y la instalación de las librerías necesarias.

**Investigación de Datos y Creación del Dataset:** Tras establecer la biblioteca, se iniciará la investigación de datos relevantes y se creará un dataset adecuado. Esta etapa será crucial para asegurar que los datos sean representativos del problema y adecuados para las técnicas de análisis seleccionadas.

**Desarrollo de Scripts y Ejecución del Estudio:** Con la biblioteca y los datos ya preparados, se desarrollarán scripts específicos para procesar y analizar la información. Posteriormente, se ejecutará el estudio, realizando pruebas de las técnicas sobre el dataset creado y comparando los resultados con otras técnicas existentes para evaluar la eficiencia y rendimiento del enfoque propuesto.

**Fase de Diseño:** Una vez completadas las pruebas, se diseñará el comportamiento del algoritmo y de los componentes del sistema, asegurando que cumplan con las condiciones necesarias para su correcto funcionamiento. Lo más importante a tener en cuenta es la facilidad para agregar un nuevo algoritmo.

**Fase de Implementación:** Se implementará la biblioteca en su conjunto, integrando los diversos algoritmos y módulos desarrollados.

**Fase de Pruebas:** Una vez implementado el sistema, se realizarán pruebas para verificar su funcionalidad y eficacia, comparando los resultados obtenidos con los estándares esperados.

**Fase de Documentación:** Finalmente, se documentará detalladamente el proceso y los resultados del proyecto en la Memoria Técnica, el Manual de Usuario y el Manual de Código, documentando concurrentemente a lo largo del desarrollo del proyecto.





### **2.2.10. Pruebas**

Se realizarán pruebas manuales para asegurar el correcto funcionamiento de la librería. La ejecución de cada algoritmo de forma aislada será suficiente para asegurar el correcto funcionamiento de la misma.

### **2.2.11. Seguridad**

Los datos manejados en el proyecto en cuestión son de uso exclusivamente personal y se procesan en un entorno local. Por esta razón, no se ha priorizado la implementación de medidas de seguridad avanzadas, dado que el valor de la información no trasciende más allá del usuario que la maneja.

En cambio, se ha optado por priorizar la eficiencia y la velocidad de ejecución en el desarrollo del software. Estos aspectos son considerados críticos para el cumplimiento de los objetivos del proyecto, enfocándose en optimizar el rendimiento del sistema en el entorno de uso previsto.

## 3 Objetivos

El propósito fundamental de este TFG es la creación de una librería en Python que unifique el formato de entrada y salida para la utilización de algoritmos de detección de anomalías, así como facilitar la realización de estudios comparativos entre distintas metodologías. A continuación, se detallan los objetivos específicos que guiarán el desarrollo de este proyecto:

### 1. Diseño y desarrollo de un módulo de formato de datos común:

Este módulo se centrará en establecer un formato estándar para la carga y manipulación de conjuntos de datos, permitiendo la extracción y análisis de métricas como el número de atributos o el porcentaje de anomalías. La uniformidad en el formato de datos es esencial para garantizar la compatibilidad y la eficiencia en el procesamiento de la información.

Además de definir el formato de datos en la librería, se desarrollarán diversos métodos complementarios que enriquecen las capacidades del módulo. Entre ellos se han incluido, como objetivos adicionales a la carga de datos, métodos de procesamiento de los datos que faciliten su uso en diversas aplicaciones y análisis:

- **Métodos de particionado de datos:** Estas funciones permiten dividir los conjuntos de datos en particiones de validación cruzada, facilitando así su análisis y procesamiento.
- **Métodos de transformación de formato:** Estas funciones permiten convertir datos de formatos diversos al formato estándar definido por el módulo, asegurando la interoperabilidad y la facilidad de integración de datos provenientes de diferentes fuentes.

### 2. Desarrollo de un Módulo para Algoritmos de Detección de Anomalías:

Este componente incluirá una variedad de algoritmos de detección de anomalías, todos diseñados para funcionar bajo un formato de configuración unificado. La inclusión de estos algoritmos en un solo módulo permitirá una aplicación y comparación más eficiente en diferentes escenarios. Cada uno de estos algoritmos estará configurado mediante un archivo de configuración común, lo que permitirá su ejecución sin necesidad de modificar el código fuente de la librería. Los algoritmos que se han incluido son:

#### a) Métodos Basados en Vecinos



## CAPÍTULO 3. OBJETIVOS

---

- 1) K-Nearest Neighbors (KNN) [14]
- 2) Local Outlier Factor (LOF) [15]
- 3) Connectivity-Based Outlier Factor (COF) [16]
- b) **Métodos de Clustering**
  - 1) Clustering-Based Local Outlier Factor (CBLOF) [17]
- c) **Enfoques Basados en Redes Neuronales**
  - 1) Autoencoders [18]
  - 2) Variational Autoencoders (VAE) con LSTM [19]
  - 3) Deep Autoencoders (DAE) [20]
- d) **Métodos Basados en Estadística**
  - 1) Histogram-Based Outlier Score (HBOS) [21]

Todos estos algoritmos estarán integrados bajo un formato de fichero de configuración común, permitiendo su ejecución sin necesidad de modificar el código fuente de la librería.

3. **Diseño y desarrollo de un módulo para Generación de Informes:**

Este módulo se encargará de configurar y generar informes estandarizados para cualquier algoritmo incluido en la librería. La estandarización de los informes permitirá una interpretación y comparación más sencilla y directa de los resultados obtenidos.
4. **Diseño y desarrollo de un módulo para Estudios Experimentales Comparativos:**

Este módulo proporcionará las herramientas necesarias para realizar análisis comparativos de diferentes algoritmos, conjuntos de datos y métodos, utilizando un formato de configuración uniforme. Facilitará la evaluación comparativa de distintas propuestas bajo condiciones controladas. Este módulo facilitará la ejecución de algoritmos en tres entornos distintos:

  - Entrenamiento de modelos: Permite entrenar algoritmos con diferentes conjuntos de datos para obtener los modelos correspondientes.
  - Ejecución de modelos entrenados: Permite utilizar modelos previamente entrenados para realizar detecciones de anomalías sobre nuevos datos.
  - Proceso entrenamiento y prueba: Permite realizar el entrenamiento de los modelos y su posterior evaluación en un solo flujo de trabajo.

Además, el módulo permitirá generar un experimento completo que implique diferentes conjuntos de datos, algoritmos y configuraciones en un mismo fichero de configuración, facilitando así la replicabilidad y la comparación de los resultados obtenidos bajo distintas condiciones experimentales.



### CAPÍTULO 3. OBJETIVOS

---

5. **Diseño y desarrollo de un módulo Tutorial:** Se diseñará un módulo educativo que presentará ejemplos prácticos de las funcionalidades principales de la librería. Este módulo será una guía esencial para usuarios nuevos, ofreciendo una visión clara de cómo utilizar la librería eficazmente.
6. **Extensibilidad y Adaptabilidad de la Librería:** Finalmente, la librería será diseñada con una arquitectura que permita su fácil expansión y adaptación. Esto asegurará que la librería pueda evolucionar y adaptarse a nuevas técnicas y enfoques en el campo de la detección de anomalías.

Como elemento adicional a los objetivos propuestos para la librería, se va a desarrollar un estudio experimental. Este estudio tendrá como finalidad demostrar cómo se puede identificar y manejar las anomalías en un contexto práctico. Se analizará la resolución de un problema real, abordándolo desde la perspectiva de la detección de anomalías. Este análisis práctico permitirá demostrar de manera concreta la utilidad y efectividad de la librería para llevar a cabo este tipo de estudios, mostrando sus beneficios en la práctica. A través de este análisis, se destacarán las ventajas de contar con una herramienta unificada para la detección de anomalías, proporcionando evidencia empírica de su impacto positivo tanto en términos de precisión como de eficiencia.

## 4 Antecedentes

En el ámbito de la detección de anomalías, la diversidad de librerías ofrece diferentes enfoques y metodologías. Cada librería tiene sus propios formatos de entrada y salida, lo que resalta la necesidad de una estandarización para facilitar comparativas y estudios integrados.

- PyOD [6] se destaca por su enfoque orientado a objetos, donde los algoritmos se implementan como clases. Su formato de entrada generalmente incluye arreglos de NumPy o DataFrames de Pandas, y como salida, devuelve las puntuaciones de anomalía o etiquetas binarias en arreglos de NumPy, lo que facilita su uso en Python pero puede limitar la interoperabilidad con sistemas que no utilizan estos formatos.
- TimeEval-algorithms [22] adopta un enfoque basado en Docker para la ejecución de algoritmos, requiriendo que los datos estén en formatos de archivo específicos como CSV para su procesamiento dentro de los contenedores Docker. Cada algoritmo tiene su propio contenedor Docker específico y requiere una configuración particular, lo que complica la tarea de realizar estudios experimentales de manera uniforme. Los resultados se generan en archivos, comúnmente en formatos CSV o JSON, almacenados en un volumen de Docker, lo que permite un acceso fácil a los datos, pero introduce complejidades en la integración directa con otras herramientas de Python.
- AnomalyDetectionToolbox [7] es un paquete de Matlab que implementa los algoritmos más populares para la detección de anomalías. Es especialmente útil para datos multivariados y contiene 10 algoritmos diferentes de detección de anomalías. La entrada generalmente involucra datos multivariados procesables en Matlab, y las salidas son típicamente puntuaciones de anomalía o clasificaciones en formatos compatibles con Matlab. Esta herramienta es ideal para usuarios que trabajan principalmente en el entorno de Matlab, pero puede presentar desafíos para la interoperabilidad con otros lenguajes de programación.
- ELKI (Environment for Developing KDD-Applications Supported by Index-Structures) [23] es un software de minería de datos de código abierto escrito en Java, enfocado en la investigación de algoritmos, con énfasis en métodos no supervisados en análisis de clusters y detección de outliers. Implementa estructuras de índices de datos como uno de sus componentes centrales para



## CAPÍTULO 4. ANTECEDENTES

mejorar el rendimiento. Acepta datos en varios formatos comunes utilizados en la minería de datos y genera resultados en formatos que son estándares en Java. Es una excelente opción para investigadores y profesionales que buscan explorar algoritmos avanzados y personalizados, pero puede requerir conocimientos de Java para su máximo aprovechamiento.

En la tabla 1 se comparan las diferentes propuestas, destacando que aunque cada librería comparte un formato de entrada, este varía entre las diferentes librerías. Además, cada una incluye diferentes tipos de algoritmos. La idea es unificar las propuestas utilizando interfaces que permitan emplear algoritmos de las diferentes librerías en un mismo framework para realizar estudios experimentales. De momento, se han adaptado las librerías que están en el mismo lenguaje(Pyod [6] y TimeEval [22]).

| Características                       | PyOD | TimeEval | AnomalyDetectionToolbox | ELKI | Librería desarrollada |
|---------------------------------------|------|----------|-------------------------|------|-----------------------|
| Formato de datos estandarizado        | ✓    | ✓        | ✓                       | ✓    | ✓                     |
| Facilidad de integración              | ✓    |          |                         |      | ✓                     |
| Soporte para múltiples algoritmos     | ✓    | ✓        | ✓                       | ✓    | ✓                     |
| Generación de informes estandarizados |      |          |                         |      | ✓                     |
| Estudios experimentales comparativos  |      |          |                         |      | ✓                     |

Tabla 1: Comparación de Características de Librerías de Detección de Anomalías

## 5 Restricciones

En este capítulo se expondrán todas aquellas restricciones presentes en el diseño y que condicionan las decisiones a tomar con respecto al proyecto.

### 5.1. Factores dato

Son aquellas limitaciones intrínsecas al proyecto, no son elegidas, sino que vienen impuestas por la naturaleza del problema y se debe de cumplir con ellas obligatoriamente. En este proyecto se tienen:

- El proyecto no debe superar las 300 horas debido a las directrices impuestas por la Universidad De córdoba acerca de los Trabajos de Fin de Grado.
- La entrada y salida de datos debe de ser agnóstica al lenguaje de programación que se use.
- No se disponen de recursos económicos para la realización del Trabajo de Fin de Grado.
- Los recursos disponibles tanto para la realización de la librería como el estudio aportado se deben de desarrollar con recursos propios y con el hardware propio.
- El lenguaje de programación elegido es Python, debido a su amplia variedad de librerías para el procesamiento de datos y algoritmos, lo cual facilita el desarrollo y la integración de nuevas funcionalidades.

### 5.2. Factores estratégicos

Los factores estratégicos son seleccionados por el ingeniero después de un estudio detallado para elegir la alternativa más adecuada. En este contexto, se han considerado los siguientes puntos clave:

- Para demostrar la versatilidad de la librería de detección de anomalías, se han establecido modelos wrapper que permiten utilizar métodos de Pyod [6] y TimeEval [22]. Estas librerías se seleccionaron porque están implementadas en Python y abarcan una gran variedad de métodos relevantes en sus respectivos dominios. Aunque poseen entornos de ejecución y entradas de datos distintas,



## CAPÍTULO 5. RESTRICCIONES

---

el estudio de su adaptación permite cubrir con los objetivos planteados. En los antecedentes, se consideraron otras alternativas, pero se optó por estas dos debido a su clara diferenciación en términos de ejecución, entrada y salida de datos.

- Se ha priorizado una arquitectura sencilla de entender y extender, de modo que el usuario común, especializado en datos y no en arquitectura de software, pueda integrar nuevos algoritmos con facilidad. Esta decisión busca hacer la herramienta accesible a un público más amplio y menos especializado en detalles técnicos de software.
- Se ha decidido que la ejecución de la librería sea a través de Docker, aprovechando la interoperabilidad que esta herramienta ofrece, lo cual facilita la distribución y el despliegue en diversos entornos sin necesidad de configuración adicional. Se consideró el uso de entornos virtuales tradicionales y herramientas como Conda, pero Docker fue preferido por su capacidad de aislar completamente las dependencias y garantizar la reproducibilidad del entorno de ejecución.
- Para el desarrollo y edición del código, se ha decidido utilizar Visual Studio Code [10] (VSCode). Esta herramienta ofrece una amplia gama de extensiones y funcionalidades que mejoran la productividad y facilitan la colaboración entre los desarrolladores. Su capacidad para integrarse con Docker y otros entornos de desarrollo lo convierte en una elección ideal para este proyecto. Se ha seleccionado sobre otras herramientas como pueden ser SublimeText por la familiaridad que ya se tenía con la herramienta y su establecimiento como estándar en la industria.
- Para la documentación se ha optado por Overleaf [24]. Esta plataforma permite la edición colaborativa de documentos en  $\text{\LaTeX}$ , lo que facilita la creación de textos bien estructurados y la inclusión de fórmulas y gráficos de manera eficiente. Además, Overleaf proporciona un entorno accesible para compartir, revisar documentos y dar feedback mediante comentarios. La alternativa a su uso eran editores de ficheros como pueden ser Word o OpenOffice que se quedan cortos ante las facilidades que ofrece  $\text{\LaTeX}$ , la facilidad de citas, enumeraciones, índices y demás funcionalidades ha hecho que se decante por esta herramienta por encima de las demás.



## 6 Recursos

### 6.1. Recursos Humanos

El autor del proyecto encargado de realizar la solución al problema propuesto, junto con el análisis de los requisitos, diseño, codificación, pruebas y generación de documentación, corresponde al alumno del Grado de Ingeniería Informática, Álvaro Pino Mérida.

La directora encargada de la coordinación y supervisión del proyecto es Amelia Zafra Gómez: Investigadora en ciencia de la Computación e Inteligencia Artificial.

### 6.2. Recursos software

Para la realización del proyecto se hará uso de los siguientes recursos software:

- Visual studio code [10] : Editor de código desde el que se desarrollará la librería.
- Git [25] : Sistema de control de versiones para el seguimiento de los cambios en el código.
- Overleaf [24] : Editor online para la escritura en LaTeX para la documentación.
- Python [26]: Lenguaje de programación de alto nivel.
- Docker [27]: Plataforma de contenedores que permite empaquetar aplicaciones y sus dependencias en un contenedor virtual, asegurando la consistencia del entorno de desarrollo y producción.

### 6.3. Recursos hardware

Se hará uso del ordenador personal con las siguientes características:

- Procesador i7 de 11 generación
- 32 GB de RAM
- gráfica Nvidia 3070 ti
- 1TB de almacenamiento SSD

## 7 Especificación de Requisitos

En este capítulo se procederá a analizar los requisitos de nuestro sistema con mayor profundidad y desde un punto de vista técnico.

### 7.1. Requisitos de Información

RI:1 El informe de salida debe contener la siguiente información:

- Algoritmo utilizado para el entrenamiento y prueba.
- Modelo generado o utilizado durante las pruebas.
- Conjunto de Datos empleado en cada etapa del proceso.
- Número de Muestras disponibles y utilizadas.
- Dimensiones de los datos, especificando número de características.
- Tasa de Anomalía observada en los datos.
- Tiempo total invertido en el entrenamiento y la prueba.
- Sensibilidad (capacidad para identificar verdaderos positivos).
- Especificidad (capacidad para identificar verdaderos negativos).
- Precisión (capacidad para identificar la proporción de positivos verdaderos).
- Curva ROC para evaluar el rendimiento del modelo.

Estas métricas están descritas en la sección 9.1.1.2.

RI:2 Los datos guardados de un modelo entrenado deben incluir:

- Nombre del archivo de datos.
- Nombre del modelo.
- Parámetros utilizados.

RI:3 Cada algoritmo debe ser configurado de acuerdo con las especificaciones definidas en el archivo de configuración (especificado en la sección 9.2):

- Nombre del algoritmo.



- Archivo de reporte donde se almacenarán los resultados.
- Conjunto de datos para el entrenamiento y prueba.
- Parámetros específicos del algoritmo para el entrenamiento.
- Modelos generados durante el entrenamiento y utilizados para la prueba.
- Semilla a utilizar.

RI:4 Las acciones deben estar separadas entre entrenamiento y prueba:

- Realizar el entrenamiento de modelos configurados con los datos y parámetros definidos.
- Realizar la ejecución de un modelo entrenado para aplicarlo a nuevos datos..

## 7.2. Requisitos Funcionales

RF:1 El sistema debe ejecutar en tres modos: **train**, **test** y **train-test**.

- **train**: Se pueden definir tantos entrenamientos como se desee, siempre que se especifiquen uno tras otro dentro de la sección **train**. Cada entrenamiento debe contener:
  - El nombre del fichero de datos con el que se va a entrenar.
  - El nombre del modelo a generar.
  - Los parámetros específicos del algoritmo.
- **test**: Se pueden definir tantas ejecuciones como se desee, siempre que se especifiquen una tras otra dentro de la sección **test**. Cada ejecución debe contener:
  - El nombre del fichero de datos con el que se va a ejecutar.
  - El modelo de entrenamiento previamente generado.
- **train-test**: Combina las secciones **train** y **test** para simplificar el fichero, siendo una especie de *syntactic sugar*(no añade nuevas funcionalidades, pero hacen el fichero más fácil de leer y escribir). Para ejecutar este tipo de configuración, se requiere:
  - El nombre del fichero de datos de entrenamiento.
  - El nombre del fichero de datos de ejecución.
  - El nombre del modelo a guardar.
  - Los parámetros de entrenamiento.



## CAPÍTULO 7. ESPECIFICACIÓN DE REQUISITOS

---

- RF:2 El sistema debe generar modelos entrenados.
- RF:3 El sistema debe generar reportes de evaluación.
- RF:4 El sistema debe procesar un conjunto de datos en el formato establecido.
- RF:5 El sistema debe permitir configurar el formato de salida de los informes.
- RF:6 El sistema debe ejecutar un estudio experimental de acuerdo con las características dadas en cuanto a datos y algoritmos.
- RF:7 El sistema debe transformar un conjunto de datos al formato establecido.
- RF:8 El sistema debe permitir configurar los algoritmos mediante un fichero de configuración.



### 7.3. Requisitos No Funcionales

- RNF:1 El sistema debe asegurar la mantenibilidad del código mediante una arquitectura desacoplada.
- RNF:2 El sistema debe implementar la paralelización de tareas para mejorar la eficiencia.
- RNF:3 El sistema debe mantener una estructura clara y modular del proyecto para facilitar futuras extensiones.
- RNF:4 El sistema debe optimizar el tiempo de ejecución de los algoritmos de detección de anomalías.
- RNF:5 El sistema debe asegurar la interoperabilidad con otras herramientas de análisis de datos.
- RNF:6 El sistema debe mantener un código modular y seguir buenas prácticas de desarrollo para facilitar su mantenimiento y actualización.
- RNF:7 El sistema debe garantizar la compatibilidad y eficiencia en el procesamiento de la información.
- RNF:8 El sistema debe asegurar la evolución y adaptación de la librería a nuevas técnicas y enfoques en el campo de la detección de anomalías.
- RNF:9 El sistema debe utilizar un formato de configuración uniforme para facilitar la evaluación comparativa bajo condiciones controladas.
- RNF:10 El sistema debe seguir un formato estándar para la carga y manipulación de conjuntos de datos.

## 8 Análisis funcional

En este capítulo se llevará a cabo un análisis de las funcionalidades del sistema que se pretende desarrollar en este Trabajo de Fin de Grado. Este análisis se abordará desde dos perspectivas diferentes:

La primera perspectiva implicará la especificación de las funcionalidades tal como serán percibidas por el usuario final, utilizando para ello los casos de uso.

La segunda perspectiva consistirá en describir el comportamiento del sistema, detallando las directrices internas que sigue la librería. Para lograr una descripción más exhaustiva, se emplearán diagramas de secuencia.

### 8.1. Casos de uso

El modelado (figura 1) ha sido realizado siguiendo las pautas de diseño de casos de uso recomendadas de IBM [28].

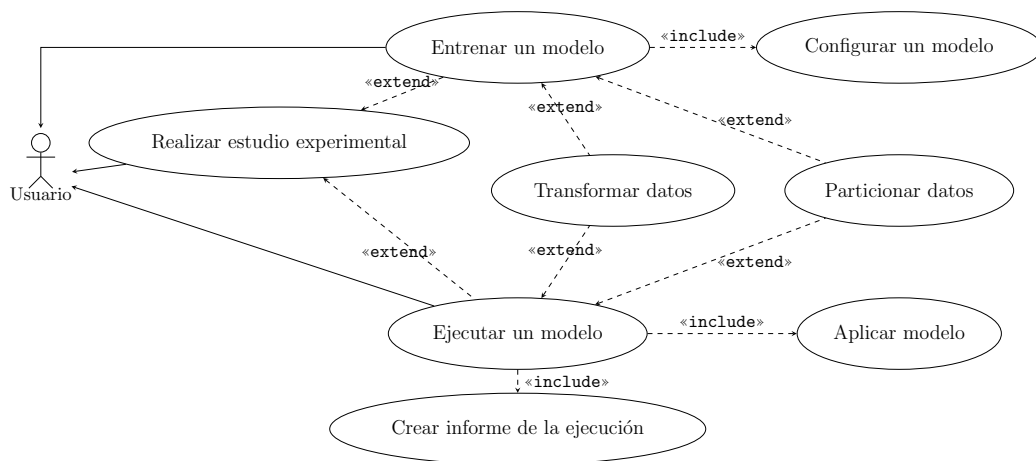


Figura 1: Diagrama de casos de uso del sistema

#### 8.1.1. Actores

**Usuario:** Un usuario que interactúa con el sistema para ejecutar modelos y generar informes. Este actor puede ser cualquier persona con acceso autorizado



al sistema, como un analista de datos o un científico de datos. Sus interacciones incluyen entrenar modelos, seleccionar modelos para ejecución, proporcionar datos de entrada, y generar informes basados en los resultados de las ejecuciones.

### 8.1.2. Especificación de Caso de Uso

A continuación se detalla la especificación de los casos de uso para los 8 casos presentes en la librería (Tablas 1, 2, 3, 4, 5, 6, 7 y 8).

#### CU-1 Entrenar modelo

|                   |   |
|-------------------|---|
| Nombre            | Entrenar_modelo   |
| Identificador     | CU-1  |
| Descripción       | El objetivo del caso de uso es entrenar un algoritmo utilizando datos específicos.  |
| Actor principal   | Usuario   |
| Actor secundario  | Ninguno   |
| Precondiciones    | Ninguno   |
| Flujo principal   | <ol style="list-style-type: none"><li>1. El caso de uso comienza cuando el Usuario selecciona un algoritmo para entrenar.</li><li>2. El sistema solicita los datos de entrenamiento necesarios.</li><li>3. include(ConfigurarModelo)<br/>    extension point: transformarDatos.<br/>    extension point: particionarDatos.</li><li>4. El sistema entrena el algoritmo con los datos proporcionados.</li><li>5. El sistema devuelve el modelo entrenado.</li></ol> |
| Flujo alternativo | <ol style="list-style-type: none"><li>1. <b>CU-1A: Datos de Entrenamiento Insuficientes</b><ol style="list-style-type: none"><li>1 Si los datos de entrenamiento no son suficientes, el sistema muestra un mensaje de error indicando los problemas.</li><li>2 El flujo retorna al paso 2 del flujo principal.</li></ol></li></ol>  |
| Postcondición     | El modelo entrenado se ha guardado correctamente.   |

Tabla 1: CU-1 Entrenar modelo

**CU-2 Ejecutar modelo**

|                   |   |
|-------------------|---|
| Nombre            | Ejecutar_modelo   |
| Identificador     | CU-2  |
| Descripción       | El objetivo del caso de uso es ejecutar un modelo previamente entrenado para hacer predicciones.  |
| Actor principal   | Usuario   |
| Actor secundario  | Ninguno   |
| Precondiciones    | El modelo debe estar entrenado y guardado en el sistema.  |
| Flujo principal   | <ol style="list-style-type: none"> <li>1. El caso de uso comienza cuando el usuario selecciona un modelo para ejecutar.</li> <li>2. El sistema solicita los datos de entrada necesarios para la ejecución.<br/>extension point: transformarDatos.<br/>extension point: particionarDatos.</li> <li>3. El usuario proporciona los datos de entrada.</li> <li>4. include(AplicarModelo)</li> <li>5. El sistema ejecuta el modelo con los datos proporcionados.</li> <li>6. El sistema muestra los resultados de la ejecución.</li> <li>7. include(CrearInforme)</li> </ol> |
| Flujo alternativo | No hay  |
| Postcondición     | Los resultados de la ejecución del modelo se guardan.   |

Tabla 2: CU-2 Ejecutar modelo



**CU-3 Crear informe**

|                   |  |
|-------------------|--|
| Nombre            | Crear_informe  |
| Identificador     | CU-3   |
| Descripción       | El objetivo del caso de uso es generar un informe detallado de los resultados de la ejecución de un modelo.  |
| Actor principal   | Usuario  |
| Actor secundario  | Ninguno  |
| Precondiciones    | El modelo debe haber sido ejecutado y debe haber resultados disponibles.   |
| Flujo principal   | <ol style="list-style-type: none"><li>1. El caso de uso comienza cuando el modelo termina su ejecución.</li><li>2. El sistema recopila los resultados de la ejecución del modelo.</li><li>3. El sistema genera un informe detallado con los resultados y otros datos relevantes.</li><li>4. El sistema guarda el informe generado.</li></ol> |
| Flujo alternativo | Ninguno  |
| Postcondición     | El informe de la ejecución se ha generado y mostrado correctamente.  |

Tabla 3: CU-3 Crear informe

**CU-4 Configurar modelo**

|                   |  |
|-------------------|--|
| Nombre            | Configurar_modelo  |
| Identificador     | CU-4   |
| Descripción       | El objetivo del caso de uso es permitir al Usuario configurar los parámetros de un modelo antes de su entrenamiento.   |
| Actor principal   | Usuario  |
| Actor secundario  | Ninguno  |
| Precondiciones    | El sistema debe permitir la modificación de los parámetros del modelo.   |
| Flujo principal   | <ol style="list-style-type: none"><li>1. El caso de uso comienza cuando el Usuario selecciona un modelo para configurar.</li><li>2. El sistema muestra las opciones de configuración disponibles según el algoritmo y wrapper de librería utilizado.</li><li>3. El Usuario modifica los parámetros del modelo según sea necesario.</li><li>4. El sistema guarda la configuración del modelo.</li></ol> |
| Flujo alternativo | Ninguno  |
| Postcondición     | La configuración del modelo se ha guardado correctamente.  |

Tabla 4: CU-4 Configurar modelo

**CU-5 Particionar datos**

|                   |   |
|-------------------|---|
| Nombre            | Particionar_datos   |
| Identificador     | CU-5  |
| Descripción       | El objetivo del caso de uso es dividir los datos en conjuntos de entrenamiento, validación y prueba.  |
| Actor principal   | Usuario   |
| Actor secundario  | Ninguno   |
| Precondiciones    | El sistema debe tener datos cargados para particionar.  |
| Flujo principal   | <ol style="list-style-type: none"> <li>1. El caso de uso comienza cuando el Usuario selecciona la opción de particionar datos.</li> <li>2. El sistema solicita los parámetros de partición (número de divisiones de los datos).</li> <li>3. El Usuario proporciona los parámetros de partición.</li> <li>4. El sistema particiona los datos según los parámetros proporcionados.</li> <li>5. El sistema guarda los conjuntos de datos particionados.</li> </ol> |
| Flujo alternativo | <ol style="list-style-type: none"> <li>1. <b>CU-5A: Parámetros de partición inválidos</b> <ol style="list-style-type: none"> <li>1 Si los parámetros de partición son inválidos, el sistema muestra un mensaje de error indicando los problemas.</li> <li>2 El flujo retorna al paso 2 del flujo principal.</li> </ol> </li> </ol>  |
| Postcondición     | Los datos se han particionado y guardado correctamente.   |

Tabla 5: CU-5 Particionar datos

**CU-6 Transformar datos**

|                   |  |
|-------------------|--|
| Nombre            | Transformar_datos  |
| Identificador     | CU-6   |
| Descripción       | El objetivo del caso de uso es transformar los datos para su uso en el entrenamiento y ejecución de modelos.   |
| Actor principal   | Usuario  |
| Actor secundario  | Ninguno  |
| Precondiciones    | El sistema debe tener acceso a los datos para transformar.   |
| Flujo principal   | <ol style="list-style-type: none"> <li>1. El caso de uso comienza cuando el Usuario selecciona la opción de transformar datos.</li> <li>2. El sistema solicita los parámetros de transformación.</li> <li>3. El Usuario proporciona los parámetros de transformación.</li> <li>4. El sistema transforma los datos según los parámetros proporcionados.</li> <li>5. El sistema guarda los datos transformados.</li> </ol> |
| Flujo alternativo | <ol style="list-style-type: none"> <li>1. <b>CU-6A: Parámetros de transformación inválidos</b> <ol style="list-style-type: none"> <li>1 Si los parámetros de transformación son inválidos, el sistema muestra un mensaje de error indicando los problemas.</li> <li>2 El flujo retorna al paso 2 del flujo principal.</li> </ol> </li> </ol>   |
| Postcondición     | Los datos se han transformado y guardado correctamente.  |

Tabla 6: CU-6 Transformar datos

**CU-7 Aplicar modelo**

|                   |  |
|-------------------|--|
| Nombre            | Aplicar_modelo   |
| Identificador     | CU-7   |
| Descripción       | El objetivo del caso de uso es aplicar un modelo entrenado a nuevos datos para obtener predicciones.   |
| Actor principal   | Usuario  |
| Actor secundario  | Ninguno  |
| Precondiciones    | El modelo debe estar entrenado y guardado en el sistema.   |
| Flujo principal   | <ol style="list-style-type: none"> <li>1. El caso de uso comienza cuando el Usuario selecciona un modelo para aplicar.</li> <li>2. El sistema solicita los nuevos datos para detectar anomalías.</li> <li>3. El Usuario proporciona los datos nuevos.</li> <li>4. El sistema aplica el modelo a los datos proporcionados.</li> <li>5. El sistema muestra las medidas de rendimiento obtenidas por el algoritmo sobre los datos indicados.</li> </ol> |
| Flujo alternativo | <ol style="list-style-type: none"> <li>1. <b>CU-7A: Datos de entrada insuficientes o inválidos</b> <ol style="list-style-type: none"> <li>1 Si los datos de entrada son insuficientes o inválidos, el sistema muestra un mensaje de error indicando los problemas.</li> <li>2 El flujo retorna al paso 2 del flujo principal.</li> </ol> </li> </ol>   |
| Postcondición     | Las predicciones se han generado y mostrado correctamente.   |

Tabla 7: CU-7 Aplicar modelo

**CU-8 Realizar estudio experimental**

|                   |  |
|-------------------|--|
| Nombre            | Realizar_estudio_experimental  |
| Identificador     | CU-8   |
| Descripción       | El objetivo del caso de uso es llevar a cabo un estudio experimental con múltiples modelos y conjuntos de datos para evaluar su desempeño.   |
| Actor principal   | Usuario  |
| Actor secundario  | Ninguno  |
| Precondiciones    | El sistema debe tener acceso a múltiples modelos y conjuntos de datos.   |
| Flujo principal   | <ol style="list-style-type: none"> <li>1. El caso de uso comienza cuando el Usuario desea realizar un estudio experimental.</li> <li>2. El sistema solicita los parámetros del estudio (modelos a evaluar, conjuntos de datos, métricas de evaluación, etc.).</li> <li>3. El Usuario proporciona los parámetros del estudio.</li> <li>4. El sistema ejecuta los modelos seleccionados con los conjuntos de datos proporcionados.</li> <li>5. El sistema recopila y muestra los resultados del estudio experimental.</li> </ol> |
| Flujo alternativo | <ol style="list-style-type: none"> <li>1. <b>CU-8A: Parámetros del estudio inválidos</b> <ol style="list-style-type: none"> <li>1 Si los parámetros del estudio son inválidos, el sistema muestra un mensaje de error indicando los problemas.</li> <li>2 El flujo retorna al paso 2 del flujo principal.</li> </ol> </li> </ol>   |
| Postcondición     | Los resultados del estudio experimental se han recopilado y mostrado correctamente.  |

Tabla 8: CU-8 Realizar estudio experimental



### 8.1.3. Matriz de trazabilidad

A continuación se presenta la matriz de trazabilidad (Figura 2) entre casos de uso y requisitos funcionales:

| CASOS DE USO           |     |     |     |     |     |     |     |     |     |
|------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| REQUISITOS FUNCIONALES |     | CU1 | CU2 | CU3 | CU4 | CU5 | CU6 | CU7 | CU8 |
|                        | RF1 | X   | X   |     | X   |     |     |     | X   |
|                        | RF2 | X   |     |     |     |     |     |     | X   |
|                        | RF3 |     | X   | X   |     |     |     |     | X   |
|                        | RF4 | X   | X   |     |     |     |     |     | X   |
|                        | RF5 | X   |     | X   |     |     |     |     | X   |
|                        | RF6 |     |     | X   |     |     |     |     | X   |
|                        | RF7 |     |     |     |     | X   | X   | X   | X   |
|                        | RF8 |     |     | X   |     |     |     |     | X   |

Figura 2: Matriz de trazabilidad

## 8.2. Descripción del Comportamiento

Esta sección se encarga de mostrar el comportamiento del sistema detallando las directrices internas que sigue la librería. Para lograr una descripción más exhaustiva, se emplearán diagramas de secuencia (Figuras 3, 4, 5, 6, 7, 8, 9 y 10. En el siguiente capítulo se explicarán más detenidamente las clases que componen la librería.

## CU-1 Entrenar modelo

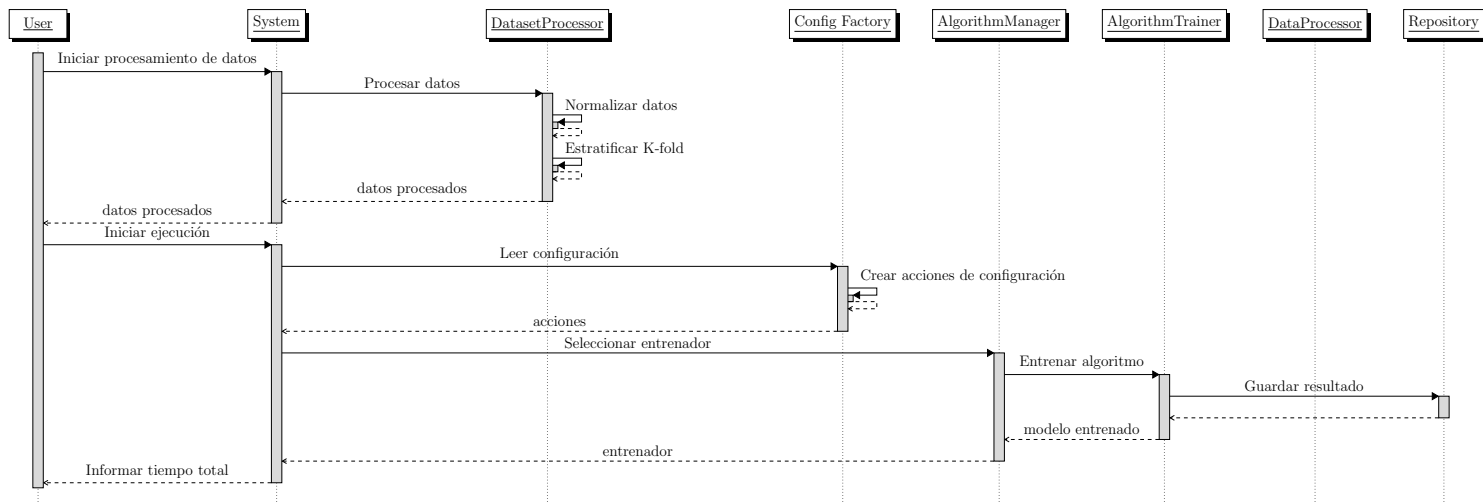


Figura 3: Diagrama de secuencia de CU-1 Entrenar modelo



## CU-2 Ejecutar modelo

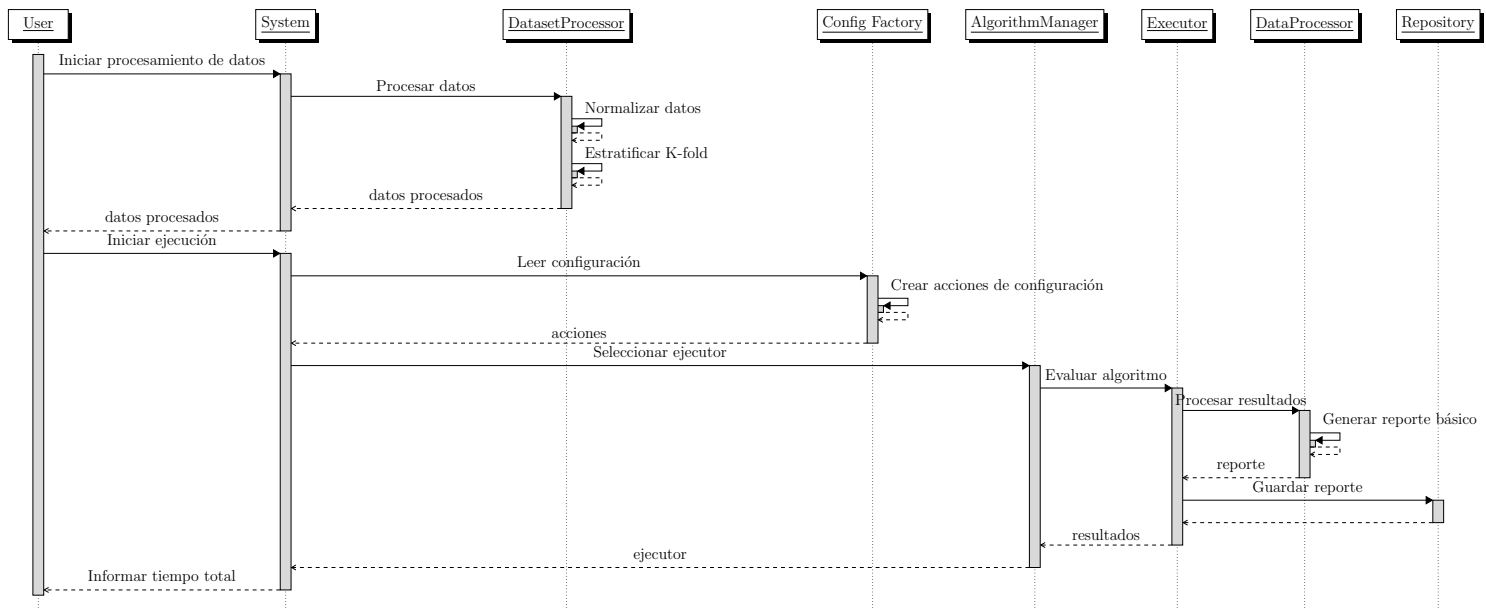


Figura 4: Diagrama de secuencia de CU-2 Ejecutar modelo

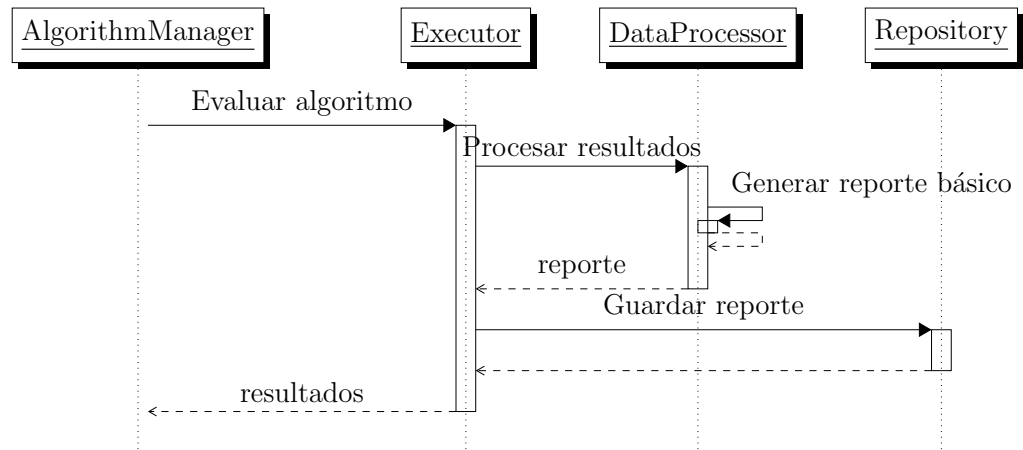
**CU-3 Crear informe**

Figura 5: Diagrama de secuencia de CU-3 Crear informe

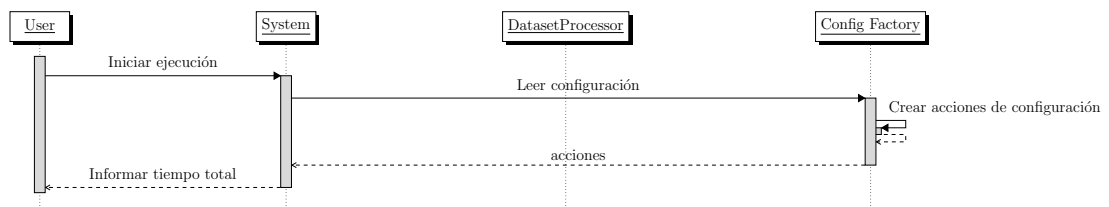
**CU-4 Configurar modelo**

Figura 6: Diagrama de secuencia de CU-4 Configurar modelo

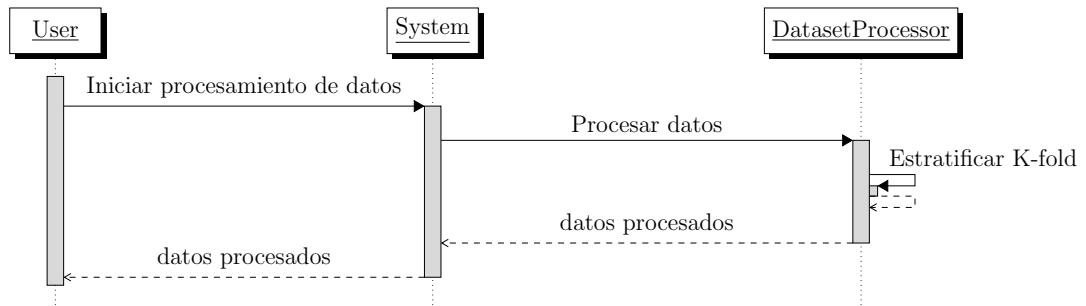
**CU-5 Particionar datos**

Figura 7: Diagrama de secuencia de CU-5 Particionar datos

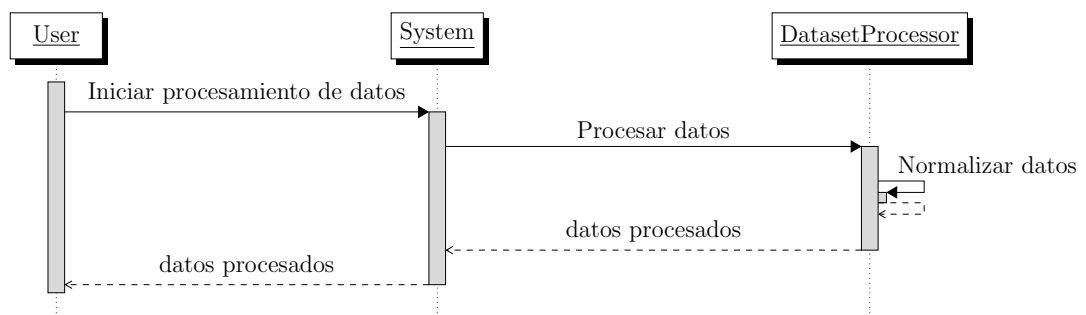
**CU-6 Transformar datos**

Figura 8: Diagrama de secuencia de CU-6 Transformar datos

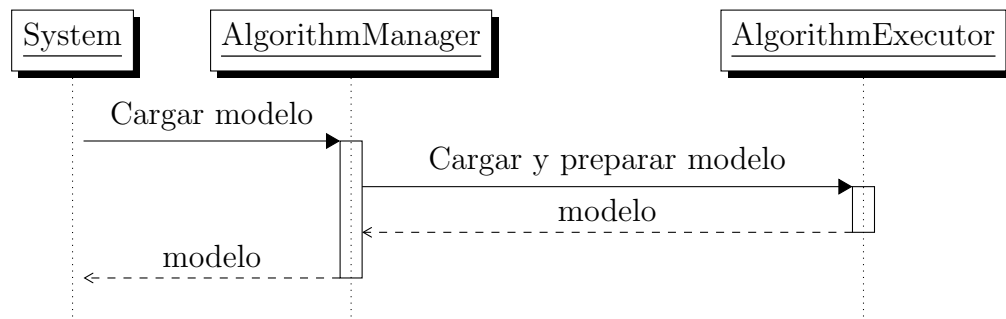
**CU-7 Aplicar modelo**

Figura 9: Diagrama de secuencia de CU-7 Aplicar modelo

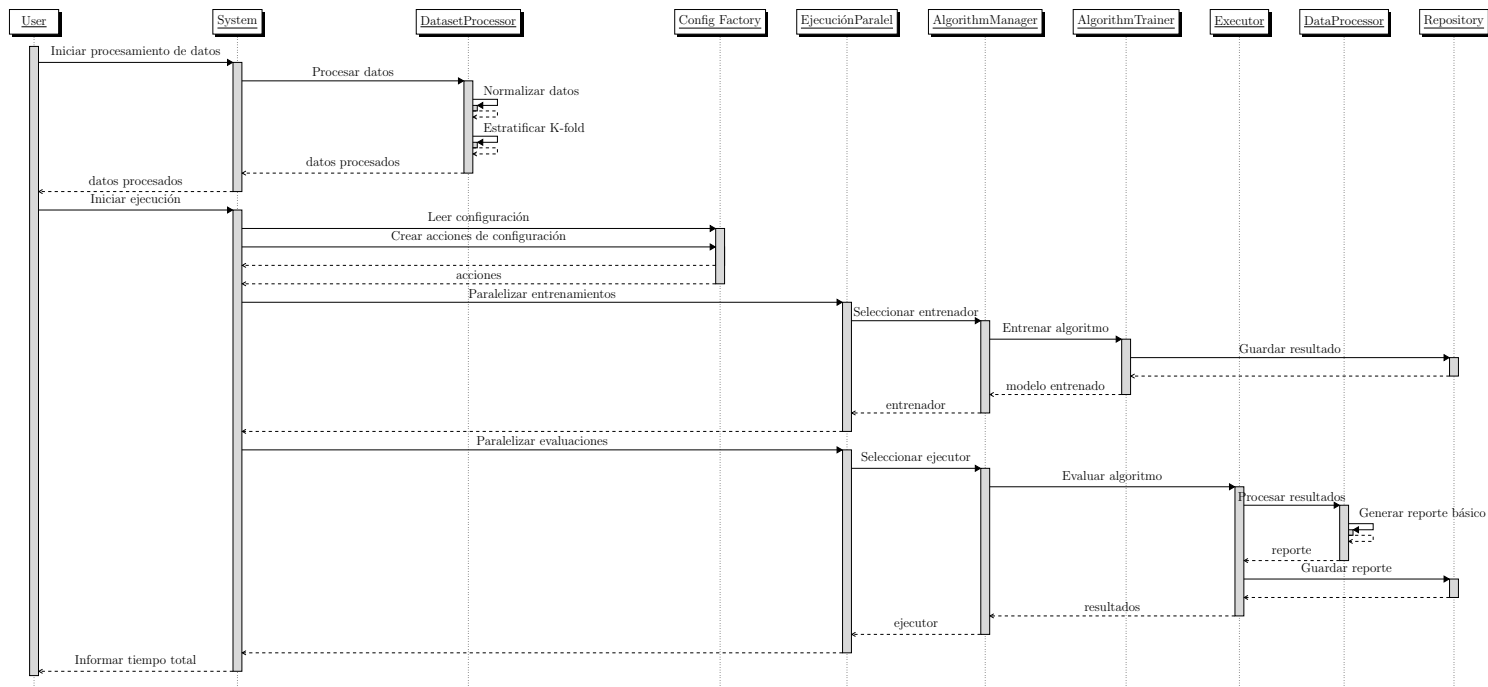


Figura 10: Diagrama de secuencia de CU-8 Realizar estudio experimental

## 9 Diseño del sistema

En este capítulo se explicará el diseño del sistema y se justificará por qué se ha llevado a cabo de esta forma. Como ya se ha comentado con anterioridad, este diseño se ha centrado en la mantenibilidad, la facilidad de uso y la capacidad de expansión a nuevos algoritmos, dado el perfil del usuario y la naturaleza del problema a resolver.

### 9.1. Arquitectura

Para lograr los objetivos de diseño, se ha seguido una arquitectura hexagonal. La arquitectura hexagonal, también conocida como Arquitectura de Puertos y Adaptadores, es un estilo de diseño de software que busca separar la lógica del negocio de un sistema de sus mecanismos de entrada y salida. Fue introducida por Alistair Cockburn en 2005 [29] con el objetivo de crear sistemas más flexibles y mantenibles. Esto es así, ya que se aíslan las capas interiores de las exteriores, haciendo que cambios externos no afecten a lógica interna. Es por ello que se ha separado el código en dominio, aplicación e infraestructura (Figura 1).

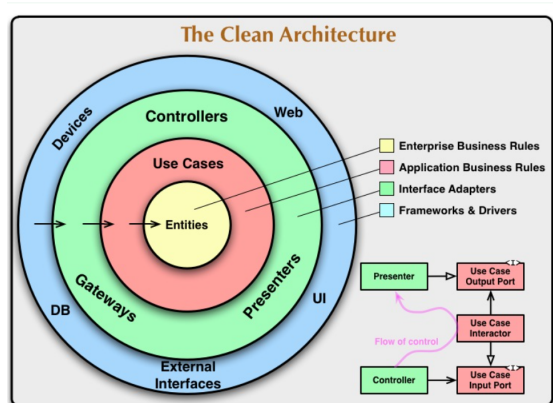


Figura 1: Diseño de la arquitectura hexagonal



## CAPÍTULO 9. DISEÑO DEL SISTEMA

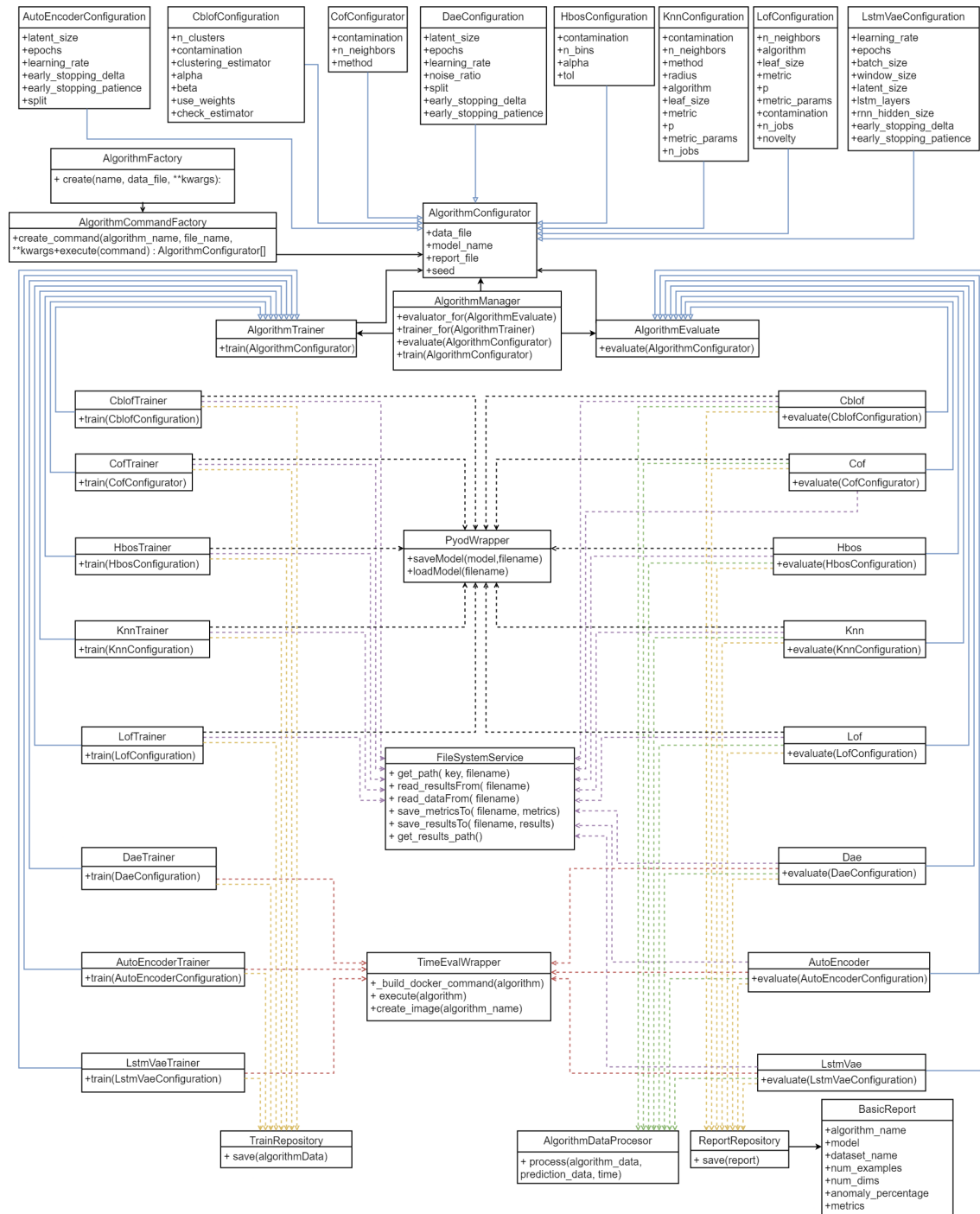


Figura 2: Diagrama de clases completo del sistema



En la figura 2 se puede observar el diagrama de clases del sistema y las relaciones entre ellas.

### 9.1.1. Dominio

En el dominio, se definen las interfaces clave que permiten la extensión y la implementación de diferentes algoritmos. Los objetos de dominio no deben de tener ninguna dependencia externa de las otras capas. Las interfaces principales son:

- **AlgorithmConfigurator**: Define los parámetros necesarios para configurar un algoritmo.
- **AlgorithmEvaluate**: Proporciona un método abstracto para evaluar algoritmos.
- **AlgorithmTrainer**: Proporciona un método abstracto para entrenar algoritmos.
- **ReportInterface**: Define un método para guardar los informes generados.
- **TrainRepository**: Define un método para guardar la configuración del algoritmo.

#### 9.1.1.1. Modelo

El modelo central del sistema incluye la clase **BasicReport**, que encapsula la información sobre el rendimiento de un algoritmo específico. Esta clase contiene atributos como el nombre del algoritmo, el modelo, el conjunto de datos utilizado, el número de ejemplos, el número de dimensiones, el porcentaje de anomalías y las métricas de rendimiento.

#### 9.1.1.2. Servicios de Dominio

En la arquitectura hexagonal, los servicios de dominio encapsulan la lógica de negocio, estando ubicados en el núcleo de la aplicación y permaneciendo independientes de las infraestructuras externas, como bases de datos o interfaces de usuario; su función principal es definir y gestionar las reglas de negocio, asegurando que estas sean el centro del desarrollo y puedan mantenerse inalteradas frente a cambios técnicos externos, permitiendo así una mayor flexibilidad y mantenibilidad del software.

Se han implementado varios servicios de dominio para calcular las métricas de rendimiento y extraer información de los archivos de datos. Estos servicios proporcionan funcionalidad clave para el análisis y evaluación del rendimiento de modelos de clasificación.





## CAPÍTULO 9. DISEÑO DEL SISTEMA

---

Para el cálculo de estas métricas, es necesario el uso de dos conceptos fundamentales: valores positivos y valores negativos.

- **Verdaderos positivos:** Casos en los que el sistema de detección identifica correctamente una anomalía.
- **Verdaderos negativos:** Casos en los que el sistema de detección identifica correctamente una instancia normal.

A continuación, se definen las métricas utilizadas:

- **Sensibilidad (Sensibilidad):** Capacidad del modelo para identificar correctamente los verdaderos positivos. Se calcula como la proporción de verdaderos positivos entre el total de casos positivos reales.

$$\text{Sensibilidad} = \frac{\text{Verdaderos Positivos}}{\text{Verdaderos Positivos} + \text{Falsos Negativos}} \quad (9.1)$$

- **Especificidad (Especificidad):** Capacidad del modelo para identificar correctamente los verdaderos negativos. Se calcula como la proporción de verdaderos negativos entre el total de casos negativos reales.

$$\text{Especificidad} = \frac{\text{Verdaderos Negativos}}{\text{Verdaderos Negativos} + \text{Falsos Positivos}} \quad (9.2)$$

- **Precisión (Precisión):** Capacidad del modelo para identificar correctamente los positivos predichos. Se calcula como la proporción de verdaderos positivos entre el total de casos predichos como positivos.

$$\text{Precisión} = \frac{\text{Verdaderos Positivos}}{\text{Verdaderos Positivos} + \text{Falsos Positivos}} \quad (9.3)$$

- **Curva ROC (Receiver Operating Characteristic):** Permite evaluar el rendimiento de un modelo de detección de anomalías a través de diferentes umbrales de decisión. La curva ROC traza la tasa de verdaderos positivos frente a la tasa de falsos positivos. El área bajo la curva ROC (AUC-ROC) proporciona una medida de la capacidad del modelo para distinguir entre clases.



### 9.1.2. Aplicación

En la capa de aplicación, se encuentran los algoritmos. Cada algoritmo debe implementar las interfaces de configuración, entrenamiento y evaluación. Para facilitar la mantenibilidad y la extensión, se han seguido las prácticas SOLID [30] (*Single Responsibility Principle, Open/Closed Principle, Liskov Substitution Principle, Interface Segregation Principle, Dependency Inversion Principle*), incluyendo la inyección de dependencias y la creación de un contenedor de dependencias para no exponer desde aplicación clases que implementen un contrato de dominio. Esto permite un cambio en cualquiera de las implementaciones de las clases que se inyectan, como puede ser TrainRepository, sin afectar a las capas inferiores.

#### 9.1.2.1. Servicios de Aplicación

Los servicios de aplicación en una arquitectura hexagonal representan la capa que orquesta las interacciones entre el dominio y los interfaces externos. Esta capa no contiene lógica de negocio propia, sino que actúa como un mediador que coordina las operaciones del dominio y las expone a través de puertos para ser utilizadas por adaptadores externos (como interfaces de usuario o APIs). Los servicios de aplicación se aseguran de que las reglas de negocio se cumplan al gestionar las transacciones y las reglas de seguridad, permitiendo una separación clara entre la lógica de negocio pura y los detalles de implementación técnica.

- **AlgorithmDataProcessor:** Este servicio se encarga del procesamiento de datos posterior a la ejecución de los algoritmos. Incluye extracción de información de los datasets y métricas de ejecución.
- **AlgorithmManager:** Este servicio gestiona los algoritmos disponibles en el sistema. Proporciona métodos para registrar nuevos algoritmos, obtener configuradores, entrenadores y evaluadores de algoritmos. Esto permite añadir nuevos algoritmos sin modificar significativamente el código existente.
- **DockerExecutor:** Este servicio se utiliza para ejecutar los algoritmos en contenedores Docker como es el caso de TimeEval, del cual esta clase es dependencia directa de su wrapper.
- **FileSystemService:** Este servicio maneja las operaciones de entrada/salida de archivos en el sistema. Proporciona métodos para leer y escribir configuraciones de algoritmos, datos de entrenamiento y resultados de evaluaciones en el sistema de archivos.



## CAPÍTULO 9. DISEÑO DEL SISTEMA

---

- PyodWrapper: Este servicio actúa como un adaptador para los algoritmos de detección de anomalías disponibles en la biblioteca PyOD. Permite integrar fácilmente algoritmos de PyOD dentro del sistema, adaptando su interfaz a las necesidades del sistema.
- TimeEvalWrapper: Este servicio envuelve la funcionalidad de evaluación temporal de algoritmos. Proporciona métodos para evaluar algoritmos en series temporales, calculando métricas específicas para datos secuenciales y asegurando que los algoritmos puedan manejar este tipo de datos adecuadamente.
- ReportInterface: Este servicio se encarga de la persistencia de los informes de salida que se generen.
- TrainRepository: Este servicio se encarga de la persistencia de los entrenamientos guardando los parámetros asociados a un modelo entrenado una vez hecha la fase de entrenamiento del mismo.



### 9.1.2.2. Gestión de Algoritmos

Para facilitar la incorporación de nuevos algoritmos sin necesidad de modificar el código existente, excepto en los archivos específicos del nuevo algoritmo, se han desarrollado dos clases: ‘AlgorithmManager’ y ‘AlgorithmFactory’. La clase ‘AlgorithmManager’ contiene las funciones ‘evaluator\_for’ y ‘trainer\_for’, mientras que ‘AlgorithmFactory’ incluye la función ‘configurator\_for’. Estas funciones se utilizan como decoradores sobre los algoritmos, permitiendo que el registro de nuevos algoritmos se realice en los mismos archivos donde se definen, sin necesidad de modificar archivos externos.

### 9.1.2.3. Fabricación de Algoritmos

Se han implementado varias factorías para la creación de configuraciones de algoritmos:

- **AlgorithmFactory:** Gestiona la creación de configuraciones de algoritmos a partir de un nombre y parámetros adicionales.
- **AlgorithmFileConfigurationFactory:** Carga configuraciones desde un archivo y crea las instancias correspondientes de configuraciones de algoritmos.

Respecto al fichero de entrada (figura 3), se utiliza el siguiente formato YAML para definir las configuraciones de los algoritmos:

Como se puede observar, el fichero debe comenzar definiendo el apartado **algorithms**. Dentro de este, se especifica el nombre del algoritmo, el cual debe coincidir con el indicado en el configurador del algoritmo (indicado en **name**). Además, se puede determinar el nombre del informe como un parámetro del algoritmo. Paralelamente, se presentan tres secciones: **train**, **train-test** y **test**. Estas secciones contienen información similar, aunque varían ligeramente entre sí:

- **train:** Es una subsección de **name**, donde se pueden definir tantos entrenamientos como se desee, siempre que se especifiquen uno tras otro dentro de la sección **train**. Cada entrenamiento debe contener el nombre del fichero de datos con el que se va a entrenar, el nombre del modelo a generar y los parámetros específicos del algoritmo.
- **test:** Es otra subsección de **name**, en la que se pueden definir tantas ejecuciones como se desee, siempre que se especifiquen una tras otra dentro de la sección **test**. Cada ejecución debe contener el nombre del fichero de datos con el que se va a ejecutar y el modelo de entrenamiento previamente generado.

```
algorithms:
- name: autoencoder
  reportFile: "allReports.csv"
  train:
    - data_file: "dataset.csv"
      model_output: "au_model.zip"
      parameters:
        latent_size: 5
        epochs: 2
  train-test:
    - data_file_test: "dataset.csv"
      data_file_train: "dataset.csv"
      model: "au_model.zip"
      parameters:
        latent_size: 10
  test:
    - data_file: "dataset.csv"
      model_input: "au_model.zip"
```

Figura 3: Ejemplo de fichero de configuración

- **train-test:** Es una subsección que combina las secciones **train** y **test** para simplificar el fichero, siendo una especie de *syntactic sugar*. Para ejecutar este tipo de configuración, se requiere el nombre del fichero de datos de entrenamiento, el nombre del fichero de datos de ejecución, el nombre del modelo a guardar y los parámetros de entrenamiento.

La ejecución en la fase de **train** produce un fichero que contiene el modelo entrenado. En la fase de **test**, la salida incluye tanto las estimaciones realizadas sobre los datos como el informe correspondiente, todo ello almacenado en el fichero especificado.

Cabe mencionar que todos los algoritmos cuentan con parámetros por defecto, por lo que no es necesario definir todos los parámetros en el fichero de configuración si los parámetros por defecto parecen acertados para el caso. Estos parámetros están definidos más adelante en la sección 9.2.

#### 9.1.2.4. Ejecución de Algoritmos

Una vez cargadas las configuraciones, se utiliza el patrón Command para ejecutar las tareas de entrenamiento y evaluación en paralelo. El patrón Command (como se puede observar en la figura 4) nos permite encontrar la clase ejecutora o entrenadora mediante el fichero de configuración, esto ayuda al desacople que se busca con la librería y lo hace robusto a cambios posteriores. La paralelización permite la ejecución de múltiples tareas, mejorando la eficiencia y reduciendo el tiempo de procesamiento.

Este diseño modular y extensible asegura que el sistema pueda mantenerse y expandirse fácilmente para incluir nuevos algoritmos de detección de anomalías sin afectar el funcionamiento de los componentes existentes.

Para la ejecución de los algoritmos se hace uso de los servicios mencionados en la sección 9.1.2.1.

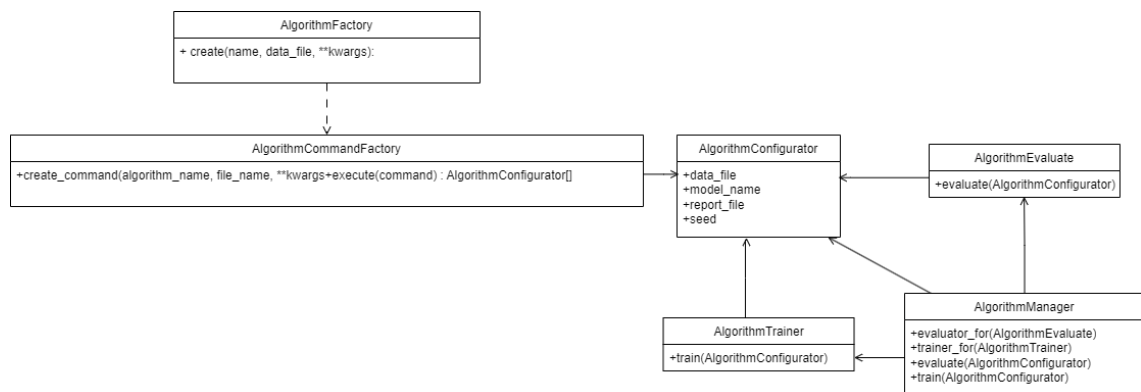


Figura 4: Diagrama de clases referente al patrón comando del sistema

#### 9.1.3. Infraestructura

Esta capa se encarga de implementar las interfaces del dominio, incluyendo la implementación de los repositorios de **Train** y **Report**, los cuales persisten los datos en formato CSV. Se ha elegido este formato porque es probable que los usuarios de la librería deseen realizar un postprocesamiento de los resultados de la ejecución. Dado que los datos ya se manejan en ficheros durante la carga, resulta natural que los resultados también se almacenen en el mismo formato.



El formato de salida para ambos repositorios se define en las siguientes clases:

- **Modelos de entrenamiento:** Cada registro incluye el nombre del algoritmo, el fichero de datos utilizado para el entrenamiento, el nombre del modelo y los parámetros de configuración.
- **Informes de ejecución:** Cada registro incluye el nombre del algoritmo, el modelo ejecutado, el fichero de datos utilizado para la ejecución, ejemplos de datos, dimensiones, porcentaje de anomalías, tiempo de ejecución, valores de SE, SP, P y ROC.

## 9.2. Descripción del modelo conceptual de datos

Dentro de la librería se soportan los algoritmos expuestos en la Figura 5:

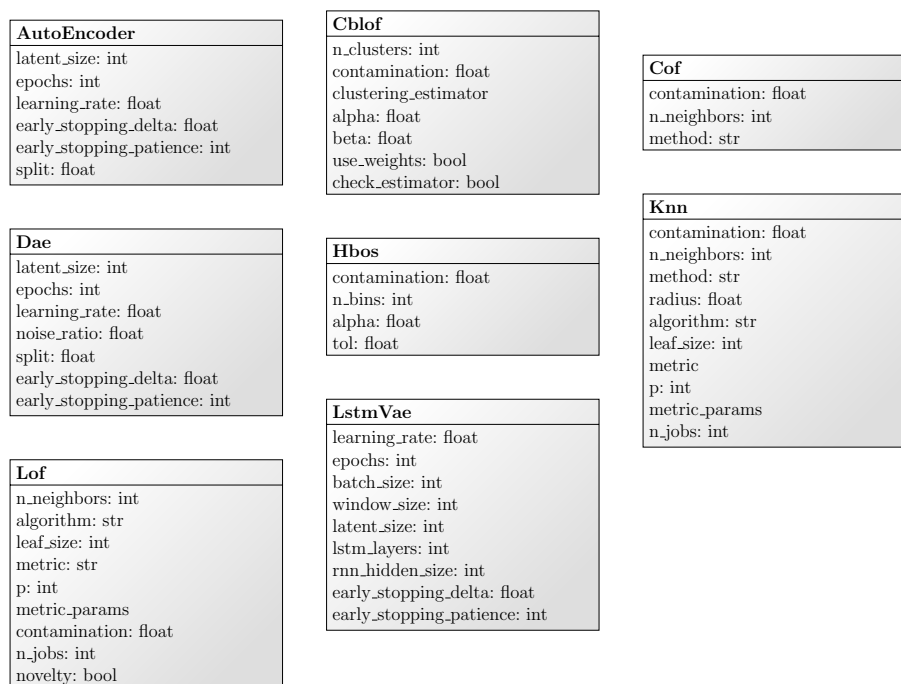


Figura 5: Modelo de datos de los algoritmos

Los algoritmos desarrollados se han incorporado a la librería mediante el uso de *wrappers* específicos. En primer lugar, utilizando el *wrapper* de Pyod, se ha dado soporte a los siguientes algoritmos:



- **Cblof (Clustering-Based Local Outlier Factor):** Este algoritmo se basa en la agrupación, identificando *outliers* en función de su pertenencia a *clusters* pequeños y distantes.
- **HBos (Histogram-based Outlier Score):** Utiliza histogramas para modelar la distribución de los datos y detecta *outliers* basándose en las frecuencias observadas.
- **Lof (Local Outlier Factor):** Identifica *outliers* evaluando la densidad local de cada punto, comparando la densidad de un punto con la de sus vecinos más cercanos.
- **Cof (Connectivity-Based Outlier Factor):** Basa su detección de anomalías en la conectividad de los datos, considerando puntos con baja conectividad como *outliers*.
- **Knn (K-Nearest Neighbors):** Determina *outliers* evaluando la distancia de un punto a sus *k* vecinos más cercanos, considerando aquellos puntos que están alejados de sus vecinos como anomalías.

Por otro lado, mediante el uso del *wrapper* de TimeEval, se han integrado los siguientes algoritmos:

- **LstmVae (Long Short-Term Memory Variational Autoencoder):** Utiliza una combinación de redes neuronales recurrentes (LSTM) y *autoencoders* variacionales para detectar anomalías en secuencias temporales, basándose en técnicas de *deep learning*.
- **Autoencoder:** Es un tipo de red neuronal que aprende a reconstruir los datos de entrada, identificando como *outliers* aquellos datos que no pueden ser reconstruidos de manera efectiva.
- **Dae (Denoising Autoencoder):** Variante del *autoencoder* que entrena la red para reconstruir datos a partir de versiones ruidosas, mejorando la robustez en la detección de anomalías al aprender características importantes de los datos.

Estos algoritmos permiten una detección de anomalías robusta y eficiente en diversas aplicaciones.



## CAPÍTULO 9. DISEÑO DEL SISTEMA

Se ha decidido diseñar de este modo debido a las prácticas comunes de desarrollo por librería, hace el código más limpio y mantenible. Estos wrappers nos brindan la ejecución correcta del algoritmo, aunque es responsabilidad de la clase del algoritmo (ya sea el entrenador o el ejecutor) mantener el formato estándar de entrada y salida de la librería. En el diagrama de clases (figura 6) se puede ver como se relacionan las clases entre sí, se ha escogido Cblof de forma arbitraria, cualquier otro algoritmo de Pyod tiene la misma estructura.

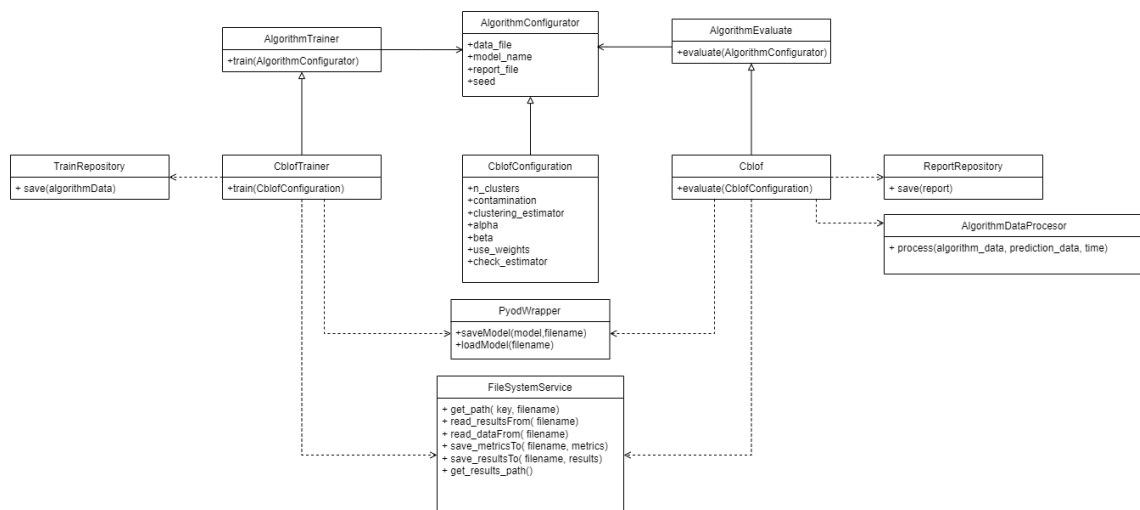


Figura 6: Diagrama de clases Cblof

q

Para incluir algoritmos de la librería Pyod, de acuerdo con el diseño realizado, se debe realizar los siguientes pasos:

1. Se crea una clase configuradora que contenga los parámetros específicos del algoritmo a implementar. Esta clase debe heredar de una clase configuradora general que incluya parámetros comunes a todos los algoritmos, tales como el archivo de datos, el nombre del modelo, el archivo de reporte y la semilla.
2. Se crea una clase entrenadora que se encargue de entrenar el algoritmo utilizando los parámetros especificados en la clase configuradora. Esta clase debe heredar de una clase entrenadora general.
3. Se crea una clase ejecutora que evalúe el rendimiento del algoritmo utilizando los parámetros especificados en la clase configuradora. Esta clase debe heredar de una clase ejecutora general.



4. Utilizar el wrapper de Pyod para cargar y guardar el modelo, encapsulando la lógica concreta de la librería.

Si se desea implementar un algoritmo utilizando el wrapper de TimeEval, el procedimiento es similar, cambiando el wrapper de Pyod por el de TimeEval en el paso 4. Los demás pasos permanecen inalterados, garantizando así una implementación consistente y mantenible.

## 9.3. Descripción de las clases

A continuación se describe con mayor detalle las clases de cada algoritmo incluido en la librería:

### 9.3.1. Clase **AutoEncoder**

La clase **AutoEncoder** representa un modelo de autoencoder utilizado para reducción de dimensionalidad y detección de anomalías en los datos. Esta clase es responsable de almacenar la configuración y los parámetros necesarios para entrenar el modelo.

- `latent_size: int` - El tamaño del espacio latente.
- `epochs: int` - Número de épocas para el entrenamiento.
- `learning_rate: float` - Tasa de aprendizaje.
- `early_stopping_delta: float` - Delta para el criterio de parada temprana.
- `early_stopping_patience: int` - Paciencia para el criterio de parada temprana.
- `split: float` - Proporción de datos de entrenamiento y prueba.

### 9.3.2. Clase **Cblof**

La clase **Cblof** implementa el algoritmo Clustering-Based Local Outlier Factor para la detección de anomalías basado en la agrupación de datos. Esta clase almacena los parámetros y la configuración del algoritmo.

- `n_clusters: int` - Número de clústeres.
- `contamination: float` - Proporción de contaminación en los datos.



- `clustering_estimator` - Estimador de agrupamiento.
- `alpha: float` - Parámetro alfa del algoritmo.
- `beta: float` - Parámetro beta del algoritmo.
- `use_weights: bool` - Indicador de uso de pesos.
- `check_estimator: bool` - Indicador de verificación del estimador.

### 9.3.3. Clase Cof

La clase **Cof** representa el algoritmo Connectivity-Based Outlier Factor (COF) utilizado para la detección de anomalías considerando la conectividad de los puntos en el espacio de características.

- `contamination: float` - Proporción de contaminación en los datos.
- `n_neighbors: int` - Número de vecinos.
- `method: str` - Método utilizado en el cálculo del COF.

### 9.3.4. Clase Dae

La clase **Dae** representa un autoencoder denoising (Denoising AutoEncoder) utilizado para la reducción de ruido en los datos y la detección de anomalías. Esta clase almacena la configuración y los parámetros necesarios para entrenar el modelo.

- `latent_size: int` - El tamaño del espacio latente.
- `epochs: int` - Número de épocas para el entrenamiento.
- `learning_rate: float` - Tasa de aprendizaje.
- `noise_ratio: float` - Proporción de ruido en los datos.
- `split: float` - Proporción de datos de entrenamiento y prueba.
- `early_stopping_delta: float` - Delta para el criterio de parada temprana.
- `early_stopping_patience: int` - Paciencia para el criterio de parada temprana.



### 9.3.5. Clase Hbos

La clase **Hbos** implementa el algoritmo Histogram-Based Outlier Score (HBOS) para la detección de anomalías basado en histogramas.

- `contamination: float` - Proporción de contaminación en los datos.
- `n_bins: int` - Número de bins en el histograma.
- `alpha: float` - Parámetro alfa del algoritmo.
- `tol: float` - Tolerancia del algoritmo.

### 9.3.6. Clase Knn

La clase **Knn** representa el algoritmo K-Nearest Neighbors (KNN) utilizado para la detección de anomalías mediante la evaluación de la proximidad de los puntos en el espacio de características.

- `contamination: float` - Proporción de contaminación en los datos.
- `n_neighbors: int` - Número de vecinos.
- `method: str` - Método utilizado en el cálculo del KNN.
- `radius: float` - Radio de búsqueda para los vecinos.
- `algorithm: str` - Algoritmo utilizado para la búsqueda de vecinos.
- `leaf_size: int` - Tamaño de las hojas en el árbol.
- `metric` - Métrica utilizada para la distancia.
- `p: int` - Parámetro p para la métrica de Minkowski.
- `metric_params` - Parámetros adicionales para la métrica.
- `n_jobs: int` - Número de trabajos paralelos.



### 9.3.7. Clase Lof

La clase **Lof** implementa el algoritmo Local Outlier Factor (LOF) para la detección de anomalías basado en la densidad local de los puntos.

- `n_neighbors: int` - Número de vecinos.
- `algorithm: str` - Algoritmo utilizado para la búsqueda de vecinos.
- `leaf_size: int` - Tamaño de las hojas en el árbol.
- `metric: str` - Métrica utilizada para la distancia.
- `p: int` - Parámetro p para la métrica de Minkowski.
- `metric_params` - Parámetros adicionales para la métrica.
- `contamination: float` - Proporción de contaminación en los datos.
- `n_jobs: int` - Número de trabajos paralelos.
- `novelty: bool` - Indicador de detección de novedades.

### 9.3.8. Clase LstmVae

La clase **LstmVae** representa un modelo de autoencoder variacional basado en LSTM (LSTM-VAE) utilizado para la detección de anomalías en secuencias temporales.

- `learning_rate: float` - Tasa de aprendizaje.
- `epochs: int` - Número de épocas para el entrenamiento.
- `batch_size: int` - Tamaño del lote.
- `window_size: int` - Tamaño de la ventana temporal.
- `latent_size: int` - Tamaño del espacio latente.
- `lstm_layers: int` - Número de capas LSTM.
- `rnn_hidden_size: int` - Tamaño oculto del RNN.
- `early_stopping_delta: float` - Delta para el criterio de parada temprana.
- `early_stopping_patience: int` - Paciencia para el criterio de parada temprana.



### 9.3.9. Clase CblofTrainer

La clase **CblofTrainer** es responsable de entrenar el algoritmo Clustering-Based Local Outlier Factor (CBLOF).

- `train(CblofConfiguration)` - Método para entrenar una configuración de CBLOF.

### 9.3.10. Clase CofTrainer

La clase **CofTrainer** es responsable de entrenar el algoritmo Connectivity-Based Outlier Factor (COF).

- `train(CofConfiguration)` - Método para entrenar una configuración de COF.

### 9.3.11. Clase HbosTrainer

La clase **HbosTrainer** es responsable de entrenar el algoritmo Histogram-Based Outlier Score (HBOS).

- `train(HbosConfiguration)` - Método para entrenar una configuración de HBOS.

### 9.3.12. Clase KnnTrainer

La clase **KnnTrainer** es responsable de entrenar el algoritmo K-Nearest Neighbors (KNN).

- `train(KnnConfiguration)` - Método para entrenar una configuración de KNN.

### 9.3.13. Clase LofTrainer

La clase **LofTrainer** es responsable de entrenar el algoritmo Local Outlier Factor (LOF).

- `train(LofConfiguration)` - Método para entrenar una configuración de LOF.



### 9.3.14. Clase DaeTrainer

La clase **DaeTrainer** es responsable de entrenar el modelo de autoencoder denoising (Denoising AutoEncoder).

- **train(DaeConfiguration)** - Método para entrenar una configuración de DAE.

### 9.3.15. Clase AutoEncoderTrainer

La clase **AutoEncoderTrainer** es responsable de entrenar el modelo de autoencoder.

- **train(AutoEncoderConfiguration)** - Método para entrenar una configuración de autoencoder.

### 9.3.16. Clase LstmVaeTrainer

La clase **LstmVaeTrainer** es responsable de entrenar el modelo de autoencoder variacional basado en LSTM (LSTM-VAE).

- **train(LstmVaeConfiguration)** - Método para entrenar una configuración de LSTM-VAE.

### 9.3.17. Clase Cblof

La clase **Cblof** implementa el algoritmo Clustering-Based Local Outlier Factor para la detección de anomalías basado en la agrupación de datos.

- **evaluate(CblofConfiguration)** - Método para evaluar una configuración de CBLOF.

### 9.3.18. Clase Cof

La clase **Cof** representa el algoritmo Connectivity-Based Outlier Factor (COF) utilizado para la detección de anomalías considerando la conectividad de los puntos en el espacio de características.

- **evaluate(CofConfiguration)** - Método para evaluar una configuración de COF.



### 9.3.19. Clase Hbos

La clase **Hbos** implementa el algoritmo Histogram-Based Outlier Score (HBOS) para la detección de anomalías basado en histogramas.

- `evaluate(HbosConfiguration)` - Método para evaluar una configuración de HBOS.

### 9.3.20. Clase Knn

La clase **Knn** representa el algoritmo K-Nearest Neighbors (KNN) utilizado para la detección de anomalías mediante la evaluación de la proximidad de los puntos en el espacio de características.

- `evaluate(KnnConfiguration)` - Método para evaluar una configuración de KNN.

### 9.3.21. Clase Lof

La clase **Lof** implementa el algoritmo Local Outlier Factor (LOF) para la detección de anomalías basado en la densidad local de los puntos.

- `evaluate(LofConfiguration)` - Método para evaluar una configuración de LOF.

### 9.3.22. Clase Dae

La clase **Dae** representa un autoencoder denoising (Denoising AutoEncoder) utilizado para la reducción de ruido en los datos y la detección de anomalías.

- `evaluate(DaeConfiguration)` - Método para evaluar una configuración de DAE.

### 9.3.23. Clase AutoEncoder

La clase **AutoEncoder** representa un modelo de autoencoder utilizado para reducción de dimensionalidad y detección de anomalías en los datos.

- `evaluate(AutoEncoderConfiguration)` - Método para evaluar una configuración de autoencoder.





### 9.3.24. Clase **LstmVae**

La clase **LstmVae** representa un modelo de autoencoder variacional basado en LSTM (LSTM-VAE) utilizado para la detección de anomalías en secuencias temporales.

- `evaluate(LstmVaeConfiguration)` - Método para evaluar una configuración de LSTM-VAE.

### 9.3.25. Clase **AlgorithmFactory**

La clase **AlgorithmFactory** es responsable de la creación de configuraciones de algoritmos basados en los datos y parámetros proporcionados.

- `create(name, data_file, **kwargs)` - Método para crear una instancia de configuración de algoritmo.

### 9.3.26. Clase **AlgorithmCommandFactory**

La clase **AlgorithmCommandFactory** es responsable de la creación de configuraciones de algoritmos basados un archivo de configuración.

- `create_command(algorithm_name, file_name, **kwargs)` - Método para crear un comando de algoritmo.
- `execute(command)` - Método para ejecutar un comando.

### 9.3.27. Clase **AlgorithmManager**

La clase **AlgorithmManager** maneja la evaluación y entrenamiento de los algoritmos.

- `evaluate(AlgorithmConfigurator)` - Método para evaluar un configurador de algoritmo.
- `train(AlgorithmConfigurator)` - Método para entrenar un configurador de algoritmo.



### 9.3.28. Clase **AlgorithmConfigurator**

La clase **AlgorithmConfigurator** es responsable de la configuración de los algoritmos y de la que heredan todos los algoritmos con tal de funcionar en la librería.

- `data_file: str` - Archivo de datos.
- `model_name: str` - Nombre del modelo.
- `report: str` - Informe de resultados.
- `seed: int` - Semilla para la aleatoriedad.

### 9.3.29. Clase **AlgorithmTrainer**

La clase **AlgorithmTrainer** es una clase base para los entrenadores específicos de algoritmos.

- `train(AlgorithmConfigurator)` - Método para entrenar un configurador de algoritmo.

### 9.3.30. Clase **AlgorithmEvaluate**

La clase **AlgorithmEvaluate** es una clase base para las ejecuciones específicas de algoritmos.

- `train(AlgorithmConfigurator)` - Método para entrenar un configurador de algoritmo.

### 9.3.31. Clase **PyodWrapper**

La clase **PyodWrapper** proporciona métodos para guardar y cargar modelos.

- `saveModel(model, filename)` - Método para guardar un modelo.
- `loadModel(filename)` - Método para cargar un modelo.



### 9.3.32. Clase **FileSystemService**

La clase **FileSystemService** maneja las operaciones de lectura y escritura en el sistema de archivos.

- `get_path(key, filename)` - Método para obtener la ruta de un archivo.
- `read_results_from(filename)` - Método para leer resultados desde un archivo.
- `read_data_from(filename)` - Método para leer datos desde un archivo.
- `save_metrics_to(filename, metrics)` - Método para guardar métricas en un archivo.
- `save_results_to(filename, results)` - Método para guardar resultados en un archivo.
- `get_results_path()` - Método para obtener la ruta de los resultados.

### 9.3.33. Clase **TimeEvalWrapper**

La clase **TimeEvalWrapper** proporciona métodos para ejecutar y evaluar algoritmos de TimeEval en un entorno de contenedor Docker.

- `build_docker_command(algorithm)` - Método para construir un comando Docker.
- `execute(algorithm)` - Método para ejecutar un algoritmo en Docker.
- `create_image(algorithm_name)` - Método para crear una imagen Docker.

### 9.3.34. Clase **TrainRepository**

La clase **TrainRepository** almacena los datos de entrenamiento de los algoritmos.

- `save(algorithmData)` - Método para guardar datos de entrenamiento.

### 9.3.35. Clase **AlgorithmDataProcessor**

La clase **AlgorithmDataProcessor** procesa los datos del algoritmo, incluyendo datos de predicción y tiempo de ejecución.

- `process(algorithm_data, prediction_data, time)` - Método para procesar datos del algoritmo.



### 9.3.36. Clase **BasicReport**

La clase **BasicReport** representa un informe básico de los resultados del algoritmo.

- `algorithm_name: str` - Nombre del algoritmo.
- `model: str` - Modelo utilizado.
- `dataset_name: str` - Nombre del conjunto de datos.
- `num_examples: int` - Número de ejemplos.
- `num_dims: int` - Número de dimensiones.
- `anomaly_percentage: float` - Porcentaje de anomalías.
- `metrics` - Métricas del modelo.

### 9.3.37. Clase **ReportRepository**

La clase **ReportRepository** almacena los informes generados por los algoritmos.

- `save(report)` - Método para guardar un informe.



## 10 Resultados experimentales

Haciendo uso de la librería desarrollada, en esta sección se llevará a cabo un estudio experimental aprovechando toda la funcionalidad de la librería desarrollada. En este estudio se presentarán y estudiarán los resultados obtenidos por los diferentes algoritmos incluidos en la librería en la resolución de un mismo problema. Para llevar a cabo este estudio, se ha seleccionado un problema real: la identificación de los estudiantes que no superan un curso online.

La detección de estudiantes que están en riesgo de no superar curso es de suma importancia, especialmente en el contexto de la educación online. Con la creciente popularidad de los cursos en línea, la capacidad de identificar a los estudiantes que necesitan apoyo adicional puede tener un impacto significativo en sus tasas de retención y éxito. Tradicionalmente, este problema ha sido abordado desde muchas perspectivas, principalmente mediante técnicas de clasificación y regresión, las cuales intentan predecir si un estudiante aprobará o no basado en ciertos indicadores. Sin embargo, consideramos que abordar este problema desde la perspectiva de la detección de anomalías puede ofrecer una visión más efectiva. La detección de anomalías se centra en identificar comportamientos que se desvían de lo que se considera normal, lo que puede ser especialmente útil en contextos educativos donde los patrones de comportamiento de los estudiantes que están en riesgo pueden ser sutiles y difíciles de capturar con métodos tradicionales. Esta forma de resolver este problema ha sido menos explorado en la educación, pero por el funcionamiento de estos modelos, puede resultar beneficioso. Los algoritmos de detección de anomalías pueden detectar patrones de comportamiento atípicos en estudiantes que están fallando, lo que puede permitir intervenciones tempranas y más efectivas para ayudarlos a mejorar su rendimiento académico. De este modo, explorar la detección de estudiantes en riesgo desde la perspectiva de anomalías no solo amplía el tipo de técnicas habituales que se han utilizado, sino que también puede ser una metodología que permita mejorar significativamente los resultados educativos en entornos de aprendizaje online.

Con la finalidad de mostrar la utilidad de la librería, se describirán todos los pasos seguidos en el estudio experimental utilizando la librería para mostrar la facilidad de llevar a cabo un estudio experimental completo que permita comparar diferentes propuestas desde el ajuste de parámetro de los algoritmos hasta la obtención de los resultados utilizando un método de validación cruzada. A continuación se



detallará el marco experimental para mostrar toda la configuración llevada a cabo y favorecer la reproducción del estudio que tendrá disponible tanto el código (la librería desarrollada), como todo el preprocesamiento de los datos que se ha tenido que realizar, como los ficheros de configuración necesarios.

## 10.1. Marco experimental

Este marco muestra todos aquellos componentes necesarios para poder llevar a cabo los experimentos y que servirán como base para poder reproducirlos. Además, se explicará todo el procesamiento realizado al conjunto de datos, el cual ha requerido un gran esfuerzo para poder utilizarlo. Este esfuerzo ha implicado el desarrollo de una base de datos y módulos que permiten obtener los conjuntos de datos con la representación y el formato necesarios para su uso en la librería.

### 10.1.1. Conjunto de datos

El dataset seleccionado para este estudio es el Open University Learning Analytics Dataset (OULAD) [1]. Este dataset proporciona información académica detallada sobre cursos, estudiantes y sus interacciones con el Entorno Virtual de Aprendizaje (VLE) en siete cursos seleccionados (denominados módulos).

El dataset se compone de varias tablas interconectadas mediante identificadores únicos. Todas las tablas se encuentran en formato CSV, lo que facilita su manejo y análisis con herramientas de procesamiento de datos.

### 10.1.2. Estructura de la información

En la figura 1 se muestra de forma visual la estructura de los datos, se encuentra de forma detallada en OULAD[1]. A continuación se indican los ficheros que tiene disponible la web junto con una descripción de la información que contiene cada fichero.

#### 1. Fichero `courses.csv`

Archivo que contiene la lista de todos los módulos disponibles y sus presentaciones. En la Tabla 1 se muestra la descripción de las columnas:

La estructura de las presentaciones B y J puede diferir y, por lo tanto, es una buena práctica analizar las presentaciones B y J por separado. No obstante, para algunas presentaciones no existe la correspondiente presentación previa B/J y, por lo

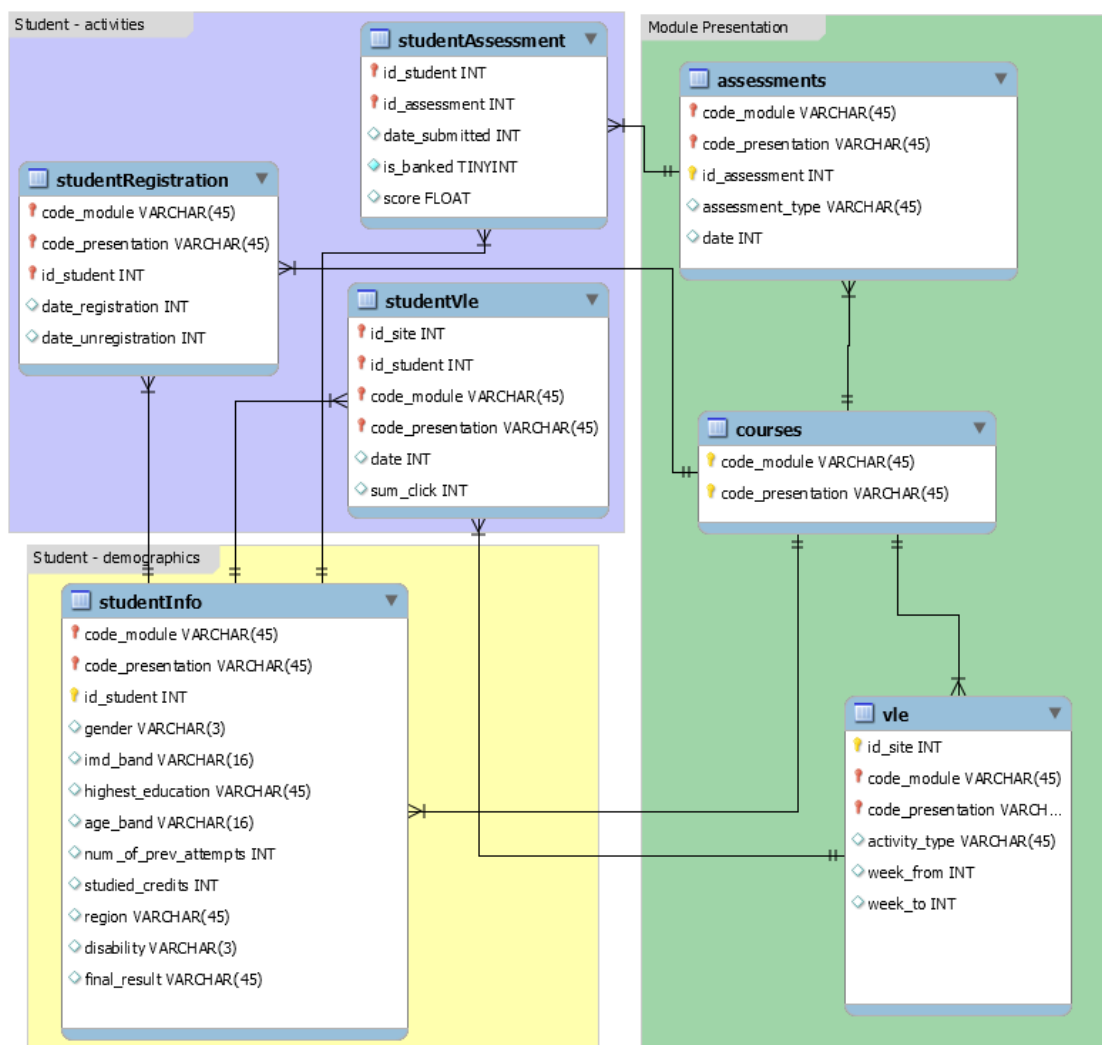


Figura 1: Estructura del dataset (Figura obtenida de [1])

| Columna           | Descripción   |
|-------------------|---|
| code_module       | Nombre en clave del módulo, que sirve de identificador.   |
| code_presentation | Nombre en clave de la presentación. Consiste en el año y "B" para la presentación que comienza en febrero y "J" para la presentación que comienza en octubre. |
| length            | Duración del módulo-presentación en días.   |

Tabla 1: Tabla de cursos



tanto, la presentación J debe utilizarse para informar la presentación B o viceversa. En el conjunto de datos este es el caso de los módulos CCC, EEE y GGG.

## 2. Fichero assessments.csv

Este archivo contiene información sobre las evaluaciones de las presentaciones de los módulos. Normalmente, cada presentación tiene una serie de evaluaciones seguidas del examen final. En la Tabla 2 se muestra la descripción de las columnas:

| Columna           | Descripción   |
|-------------------|---|
| code_module       | Código de identificación del módulo al que pertenece la evaluación.   |
| code_presentation | Código de identificación de la presentación a la que pertenece la evaluación.   |
| id_assessment     | Número de identificación de la evaluación.  |
| assessment_type   | Tipo de evaluación. Existen tres tipos de evaluaciones: Tutor Marked Assessment (TMA), Computer Marked Assessment (CMA) y Final Exam (Exam).  |
| date              | Información sobre la fecha de presentación final de la evaluación calculada como el número de días transcurridos desde el inicio de la presentación del módulo. La fecha de inicio de la presentación tiene el número 0 (cero). |
| weight            | Peso de la evaluación en %. Normalmente, los exámenes se tratan por separado y tienen el peso 100 %; la suma de todas las demás evaluaciones es 100 %.  |

Tabla 2: Tabla de entregas

Si falta la información sobre la fecha del examen final, es al final de la última semana de presentación.





### 3. Fichero vle.csv

El archivo csv contiene información sobre los materiales disponibles en la plataforma educativa. Normalmente se trata de páginas html, archivos pdf, etc. Los estudiantes tienen acceso a estos materiales en línea y se registran sus interacciones con los materiales. En la Tabla 3 se muestra la descripción de las columnas:

| Columna           | Descripción  |
|-------------------|--|
| id_site           | Un número de identificación del material.                      |
| code_module       | Un código de identificación del módulo.                        |
| code_presentation | Código de identificación de la presentación.                   |
| activity_type     | El rol asociado al material del módulo.                        |
| week_from         | Semana a partir de la cual está previsto utilizar el material. |
| week_to           | Semana hasta la que está previsto utilizar el material.        |

Tabla 3: Tabla de recursos



#### 4. Fichero studentInfo.csv

Este fichero contiene información demográfica sobre los alumnos junto con sus resultados. En la Tabla 4 se muestra la descripción de las columnas:

| Columna              | Descripción  |
|----------------------|--|
| code_module          | Código de identificación del módulo en el que está matriculado el alumno.  |
| code_presentation    | Código de identificación de la presentación en la que el estudiante está inscrito en el módulo.                                |
| id_student           | Un número de identificación único para el estudiante.  |
| gender               | El género del estudiante.  |
| region               | Identifica la región geográfica en la que vivía el estudiante mientras cursaba el módulo-presentación.                         |
| highest_education    | Nivel educativo más alto del estudiante al entrar en la presentación del módulo.   |
| imd_band             | Especifica la banda del Índice de Depravación Múltiple del lugar donde vivía el estudiante durante la presentación del módulo. |
| age_band             | Banda de la edad del alumno.   |
| num_of_prev_attempts | El número de veces que el alumno ha intentado este módulo.   |
| studied_credits      | El número total de créditos de los módulos que el estudiante está cursando actualmente.  |
| disability           | Indica si el estudiante ha declarado una discapacidad.   |
| final_result         | Resultado final del alumno en la presentación del módulo.  |

Tabla 4: Tabla de información del estudiante



### 5. Fichero studentRegistration.csv

Este fichero contiene información sobre el momento en que el estudiante se matriculó en la presentación del módulo. En el caso de los estudiantes que se dieron de baja, también se registra la fecha de baja. En la Tabla 5 se muestra la descripción de las columnas:

| Columna             | Descripción   |
|---------------------|---|
| code_module         | Código de identificación de un módulo.  |
| code_presentation   | Código de identificación de la presentación.  |
| id_student          | Un número de identificación único para el estudiante.   |
| date_registration   | La fecha de registro del estudiante en la presentación del módulo, es el número de días medidos en relación con el inicio de la presentación del módulo (por ejemplo, el valor negativo -30 significa que el estudiante se registró en la presentación del módulo 30 días antes de que comenzara).  |
| date_unregistration | Fecha en la que el estudiante se desmatriculó de la presentación del módulo, es el número de días medidos con respecto al inicio de la presentación del módulo. Los estudiantes que han completado el curso tienen este campo vacío. Los estudiantes que se han dado de baja tienen como valor de la columna final_resultado en el archivo studentInfo.csv la palabra Withdrawal. |

Tabla 5: Tabla de registro



## 6. Fichero `studentAssessment.csv`

Este fichero contiene los resultados de las evaluaciones de los alumnos. Si el alumno no presenta la evaluación, no se registra ningún resultado. Si el resultado de las evaluaciones no se almacena en el sistema, no se registra la presentación del examen final. En la Tabla 6 se muestra la descripción de las columnas:

| Columna                     | Descripción   |
|-----------------------------|---|
| <code>id_assessment</code>  | Número de identificación de la evaluación.  |
| <code>id_student</code>     | Número de identificación único del estudiante.  |
| <code>date_submitted</code> | La fecha de presentación del estudiante, medida como el número de días desde el inicio de la presentación del módulo.   |
| <code>is_banked</code>      | Una bandera de estado que indica que el resultado de la evaluación ha sido transferido de una presentación anterior.  |
| <code>score</code>          | La puntuación del estudiante en esta evaluación. El rango es de 0 a 100. La puntuación inferior a 40 se interpreta como Suspenso. Las notas están en el rango de 0 a 100. |

Tabla 6: Tabla de resultados en evaluaciones



### 7. Fichero studentVle.csv

El archivo studentVle.csv contiene información sobre las interacciones de cada estudiante con los materiales del EVE. En la Tabla 7 se muestra la descripción de las columnas:

| Columna           | Descripción   |
|-------------------|---|
| code_module       | Código de identificación de un módulo.  |
| code_presentation | Código de identificación de la presentación del módulo.   |
| id_student        | Un número de identificación único para el estudiante.   |
| id_site           | Número de identificación del material del EVE.  |
| date              | La fecha de la interacción del estudiante con el material, medida como el número de días desde el inicio de la presentación del módulo. |
| sum_click         | El número de veces que un estudiante interactúa con el material en ese día.   |

Tabla 7: Tabla de interacción del alumno con los recursos



## 10.2. Procesamiento del conjunto de datos

El formato de los ficheros del dataset descrito en la sección anterior presenta una estructura ideal para cargar a base de datos, pero no es el caso para su análisis directo. Es por ello que se ha tenido que llevar a cabo un preprocesamiento de los datos.

### Diseño de la base de datos

Debido a la cantidad de tuplas que presenta el dataset (unas 10 millones) y las relaciones entre ficheros, es necesario el traspaso de los ficheros a una base de datos relacional. Esto nos permite, por un lado, acceder a la información de forma más sencilla y, por otro lado, formatearla de una manera mucho más rápida.

El formato de ficheros del dataset ya presentaba un formato de base de datos relacional, es por ello que después de la viabilidad de su traspaso se recreó ese mismo diseño que nos aportan en su web [1] y que se muestra en la figura 1. Se puede ver el script de migración de fichero a base de datos en `/database/init.sql/init.sql`.

La base de datos diseñada consta de siete tablas principales, que corresponden a las diferentes entidades y sus relaciones dentro del dataset. A continuación, se describen cada una de las tablas:

- **Courses:** Esta tabla contiene información sobre los cursos disponibles. Cada curso se identifica mediante un código de módulo (`code_module`) y un código de presentación (`code_presentation`). Además, se incluye la duración del curso (`length`).
- **Assessments:** Esta tabla almacena las evaluaciones asociadas a los cursos. Cada evaluación se identifica mediante un `id_assessment` único y se relaciona con los cursos a través de `code_module` y `code_presentation`. Se incluye el tipo de evaluación (`assessment_type`), la fecha (`date`) y el peso (`weight`) de la misma.
- **Vle:** En esta tabla se registra la información relacionada con los entornos virtuales de aprendizaje (VLEs). Cada sitio (`id_site`) está vinculado a un curso y una presentación específicos. Se incluye el tipo de actividad (`activity_type`) y el rango de semanas (`week_from` y `week_to`) en el que está disponible.
- **StudentInfo:** Esta tabla contiene información sobre los estudiantes, identificados por un `id_student` único. Además de los datos demográficos



---

CAPÍTULO 10. RESULTADOS EXPERIMENTALES

---

(género, región, educación más alta, etc.), se incluye información académica como el número de intentos previos (`num_of_prev_attempts`) y el resultado final (`final_result`).

- **StudentRegistration:** Registra las fechas de registro y desregistro de los estudiantes en los cursos. Cada registro está vinculado a un estudiante específico y a un curso.
- **StudentAssessment:** Almacena las evaluaciones de los estudiantes. Cada entrada se identifica por un `id_assessment` y un `id_student`. Se registra la fecha de envío (`date_submitted`), si está en banca (`is_banked`) y la puntuación obtenida (`score`).
- **StudentVle:** Esta tabla recoge la interacción de los estudiantes con los VLEs. Se almacena el número de clics (`sum_click`) que un estudiante realiza en un sitio específico dentro de un curso en una fecha determinada.

Las relaciones entre las entidades se representan mediante claves foráneas. Por ejemplo, la tabla **Assessments** tiene claves foráneas que refieren a la tabla **Courses**. De manera similar, la tabla **Vle** se relaciona con **Courses** a través de claves foráneas. La tabla **StudentInfo** también se vincula a **Courses**. Además, **StudentRegistration** se relaciona con **StudentInfo**. La tabla **StudentAssessment** se relaciona tanto con **Assessments** como con **StudentInfo**. Finalmente, **StudentVle** se relaciona con **StudentInfo**, **Vle**, y **Courses**.



## Importación de ficheros

Para la carga de los datos y su posterior uso se ha hecho uso de la herramienta `docker-compose`. Mediante el uso de la imagen de `docker mysql:8.0`, se puede importar la información necesaria, primero se crea la base de datos y una vez creada se cargan los datos haciendo uso del comando `LOAD DATA`. Se muestra a continuación la parte relacionada con los datos del fichero de configuración de `docker compose` en la figura 2.

```
db:
  image: mysql:8.0
  volumes:
    - ./database/init_sql:/docker-entrypoint-initdb.d
    - ./database/init_csv:/var/lib/mysql-files/init_csv
  ports:
    - "3306:3306"
  environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
phpmyadmin:
  image: phpmyadmin/phpmyadmin
  ports:
    - "8080:80"
  environment:
    PMA_HOST: db
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
  depends_on:
    - db
```

Figura 2: Extracto del fichero `docker-compose`





## Limpieza de datos

El conjunto de datos presenta duplicados que han sido tratados durante el proceso de formateo. En particular, en la tabla *StudentVle* se observan múltiples tuplas con los mismos valores para *code\_module*, *code\_presentation*, *id\_student*, *id\_site* y *date*.

Con respecto a la fecha, está formateada como entero relativo de días desde el inicio de curso, por lo que nos podemos encontrar hasta fecha -24, es fundamental tener en consideración que el rango de fechas no es únicamente positivo.

Otro aspecto relevante para obtener información coherente es que no todas las fechas dentro del rango del curso están presentes para todos los estudiantes; solo se registran aquellas fechas en las que los estudiantes realizaron clics. Al homogeneizar los datos por estudiante, es fundamental considerar este aspecto y rellenar las fechas faltantes con un valor de 0.

### 10.3. Análisis de la información almacenada

La base de datos contiene información sobre un total de 22 cursos. Cada curso tiene un número variable de tareas asociadas, con un rango que varía entre 5 y 14 tareas por curso. Además, la actividad en la plataforma virtual de aprendizaje (VLE) se mide en clics, con una media de clics por curso que oscila entre 2.4942 y 4.4573.

Como se puede observar en la tabla 8, los cursos presentan diferencias notables entre ellos, mientras que dichas diferencias no son significativas en sus respectivas divisiones por presentaciones. Además, los cursos presentan distintos recursos, por lo que ante un análisis de datos general induce a una gran cantidad de celdas vacías o con valor 0. Por esta razón, se ha decidido estratificar la información por cursos (tabla 9) en lugar de por presentaciones.



| Código del Módulo | Presentación | Duración (días) | Número de Tareas | Fecha de Inicio | Fecha de Fin | Media de Clics |
|-------------------|--------------|-----------------|------------------|-----------------|--------------|----------------|
| AAA               | 2013J        | 268             | 6                | -198            | 268          | 3.5832         |
| AAA               | 2014J        | 269             | 6                | -156            | 269          | 3.5328         |
| BBB               | 2013B        | 240             | 12               | -276            | 240          | 3.3425         |
| BBB               | 2013J        | 268             | 12               | -198            | 268          | 3.0458         |
| BBB               | 2014B        | 234             | 12               | -322            | 234          | 3.0518         |
| BBB               | 2014J        | 262             | 6                | -163            | 262          | 3.8176         |
| CCC               | 2014B        | 241             | 10               | -312            | 241          | 3.8074         |
| CCC               | 2014J        | 269             | 10               | -156            | 269          | 3.9247         |
| DDD               | 2013B        | 240             | 14               | -310            | 240          | 2.5845         |
| DDD               | 2013J        | 261             | 7                | -205            | 261          | 2.5811         |
| DDD               | 2014B        | 241             | 7                | -304            | 241          | 2.4942         |
| DDD               | 2014J        | 262             | 7                | -163            | 262          | 2.5272         |
| EEE               | 2013J        | 268             | 5                | -195            | 268          | 4.3113         |
| EEE               | 2014B        | 241             | 5                | -275            | 241          | 4.1144         |
| EEE               | 2014J        | 269             | 5                | -156            | 269          | 4.0110         |
| FFF               | 2013B        | 240             | 13               | -305            | 240          | 4.4573         |
| FFF               | 2013J        | 268             | 13               | -198            | 444          | 4.3652         |
| FFF               | 2014B        | 241             | 13               | -365            | 241          | 4.3422         |
| FFF               | 2014J        | 269             | 13               | -156            | 269          | 4.3638         |
| GGG               | 2013J        | 261             | 10               | -190            | 261          | 3.4870         |
| GGG               | 2014B        | 241             | 10               | -214            | 241          | 3.4477         |
| GGG               | 2014J        | 269             | 10               | -156            | 269          | 3.3907         |

Tabla 8: Análisis de los datos por módulo y presentación

| Código del Módulo | Duración (días) | Número de Tareas | Fecha de Inicio | Fecha de Fin | Media de Clics |
|-------------------|-----------------|------------------|-----------------|--------------|----------------|
| AAA               | 269             | 12               | -198            | 269          | 3.5588         |
| BBB               | 268             | 42               | -322            | 268          | 3.3391         |
| CCC               | 269             | 20               | -312            | 269          | 3.8765         |
| DDD               | 262             | 35               | -310            | 262          | 2.5526         |
| EEE               | 269             | 15               | -275            | 269          | 4.1441         |
| FFF               | 269             | 52               | -365            | 444          | 4.3826         |
| GGG               | 269             | 30               | -214            | 269          | 3.4452         |

Tabla 9: Análisis de los datos por módulo

En el contexto del análisis de los resultados de los estudiantes, se ha unificado en la consideración de dos clases los que no superan el curso combinando los valores de 'Fail' o 'Withdrawn' y los que sí lo superan combinando los valores de 'Pass' y 'Distinction'. Para adaptarlo a un problema de detección de anomalías se va a considerar que no superar el curso sería la anomalía y superarlo sería la clase normal. De acuerdo con la tabla 10, se puede apreciar que esta consideración hace que la clase que queremos que se trate como anomalía es muy elevada en algunos cursos, lo que puede conllevar a que su resolución no sea muy acertada utilizando estos métodos. Este estudio, cuyo principal interés es demostrar el funcionamiento de librería para el desarrollo de estudios experimentales, también nos permitirá determinar el funcionamiento de estos métodos en estos entornos y analizar su comportamiento.

| Código del Módulo | Número de Estudiantes | Número de Anomalías | Porcentaje de Anomalías |
|-------------------|-----------------------|---------------------|-------------------------|
| AAA               | 748                   | 217                 | 29.01 %                 |
| BBB               | 7909                  | 4155                | 52.54 %                 |
| CCC               | 4434                  | 2756                | 62.16 %                 |
| DDD               | 6272                  | 3662                | 58.39 %                 |
| EEE               | 2934                  | 1284                | 43.76 %                 |
| FFF               | 7762                  | 4114                | 53.00 %                 |
| GGG               | 2534                  | 1020                | 40.25 %                 |

Tabla 10: Cantidad de Alumnos y Anomalías por Curso



## Generación de los Conjuntos de Datos

Para el análisis y entrenamiento de los modelos, se ha definido el concepto de anomalía basándose en los resultados finales de los estudiantes como se ha comentado en el apartado anterior. La identificación de estos casos permite tratar el conjunto de datos con dos clases (la normal y la anomalía), facilitando el uso de técnicas de aprendizaje supervisado para detectar patrones de comportamiento anómalo.

La información se ha organizado por cursos, permitiendo una mejor agrupación y análisis de los datos. Esta estructura proporciona un punto de partida claro para comprender la información y facilita la comparación entre diferentes cursos. La división del conjunto de datos por cursos se ha implementado tanto para el entrenamiento como para la ejecución de los modelos. Las pruebas iniciales se realizaron sobre el curso AAA.

En el proceso de formateo de datos, se exploraron diversas estrategias, descritas a continuación, aunque algunas no resultaron exitosas:

Como primer intento, se fusionaron los datos de *StudentInfo*, *StudentVle* y *StudentAssessment* en un único DataFrame, estructurando la información agrupada por alumno y fecha. Esta estrategia permitía obtener una visión detallada del comportamiento del estudiante en fechas específicas, pero resultó en un DataFrame extremadamente grande y difícil de manejar debido a la granularidad de los datos. Este proceso se implementó en los ficheros `1-changeDataset.py` y `1-GroupByCourse.py`.

En un segundo intento, se fusionaron únicamente los datos de *StudentInfo* y *StudentVle*, excluyendo *StudentAssessment*, en un único DataFrame, también agrupado por alumno y fecha. Sin embargo, esta estrategia resultó en un DataFrame poco informativo, con muchas tuplas repletas de ceros, incluso tras eliminar las evaluaciones. Este proceso se implementó en el fichero `3-activityTypeDataset.py`.

Para combatir estos problemas, se eliminó la agrupación por fecha y se optó por agrupar por alumno y tipo de recurso. De esta idea surgieron dos conjuntos de datos que dieron mejores resultados: uno agrupado por suma (`4-sumActivityTypeDataset.py`) y otro por media de clics (`4-averageActivityTypeDataset.py`). Finalmente, la representación usada en el estudio experimental se compone de la suma de actividades, la cual contiene toda la información relativa al estudiante (número de intentos previos, créditos de



estudio) y la información relativa al curso en relación con el estudiante (fechas de entrega, notas y clics en los recursos).

Los resultados insatisfactorios iniciales parecían deberse a la gran cantidad de ceros presentes en el conjunto de datos. Al dejar de agrupar por fecha y agrupar los recursos únicamente por alumno, se redujo drásticamente la cantidad de tuplas con valores nulos. Esta estrategia produjo una única tupla por estudiante con toda su información, reduciendo significativamente la cantidad de ceros presentes. Además, se observó una mejoría al normalizar los datos.

## 10.4. Particionamiento de los datos

La división de los datos utilizando validación cruzada es muy importante a la hora de hacer cualquier estudio experimental. En este caso se decidió hacer uso de validación cruzada 5-fold separando entre conjuntos de prueba y entrenamiento. Además, al existir algoritmos entrenados únicamente sobre datos normales, sin la presencia de anomalías, se prepararon particiones específicos para este caso específico también. En el desarrollo de las diferentes particiones, se mantuvo el porcentaje de anomalías presente en el conjunto de datos original.



## 10.5. Configuración de los Algoritmos

La configuración de los algoritmos es una parte esencial del proceso de experimentación. En este proyecto, se utilizaron diversos algoritmos de detección de anomalías, cada uno con sus respectivos parámetros ajustables incluidos en la librería: K-Nearest Neighbors (KNN), Local Outlier Factor (LOF), Connectivity-Based Outlier Factor (COF), Clustering-Based Local Outlier Factor (CBLOF), Histogram-Based Outlier Score (HBOS), Autoencoder y Denoising Autoencoder (DAE).

Debido a que los algoritmos necesitan una configuración apropiada de sus parámetros para resolver un problema concreto. Este estudio también incluye un estudio de los parámetros más relevantes de cada algoritmo, para estudiar cómo influyen en este problema y principalmente, para mostrar la facilidad de llevar a cabo este proceso a partir de los ficheros de configuración de la librería desarrollada. Se ejecutó con semillas diferentes para garantizar la reproducibilidad y evaluar la robustez de los resultados. Los parámetros configurables y sus valores correspondientes se detallan a continuación:

- **KNN:**  $n\_neighbors = [5, 10, 20, 30, 40]$
- **LOF:**  $n\_neighbors = [5, 10, 20, 30, 40]$
- **COF:**  $n\_neighbors = [5, 10, 20, 30, 40]$
- **CBLOF:**  $n\_clusters = [6, 7, 8, 9, 10]$
- **HBOS:**  $n\_bins = [5, 10, 15, 20, 25]$
- **Autoencoder:**  $latent\_size = [16, 32, 64, 128, 256]$
- **DAE:**  $latent\_size = [16, 32, 64, 128, 256]$

Se emplearon diez semillas diferentes: 114, 99, 25, 76, 9, 160, 53, 60, 30 y 170, y los módulos de código correspondientes fueron *AAA*, *BBB*, *CCC*, *DDD*, *EEE*, *FFF*, *GGG*.

Para garantizar una evaluación exhaustiva de los resultados, se aplicó validación cruzada 5-fold, utilizando la funcionalidad de la librería. La validación cruzada es esencial en la experimentación para asegurar que los resultados obtenidos sean robustos y generalizables. Al dividir los datos en cinco subconjuntos (*folds*), se entrena el modelo en cuatro de ellos y se prueba en el quinto, repitiendo este proceso



cinco veces para que cada subconjunto sea utilizado como conjunto de prueba una vez. Esto no solo maximiza el uso de los datos disponibles, sino que también proporciona una medida más confiable del rendimiento del modelo al promediar los resultados de las cinco iteraciones.

Además, cada *fold* y configuración del algoritmo se ejecutó con las 10 semillas diferentes en los algoritmos estocásticos para garantizar una evaluación apropiada de los resultados. Los algoritmos estocásticos, que dependen de la aleatoriedad en su proceso de entrenamiento, pueden producir resultados diferentes en cada ejecución. Al utilizar múltiples semillas, se puede mitigar el efecto de esta variabilidad y obtener una estimación más precisa y estable del rendimiento del modelo. Esto es crucial para asegurar que los resultados no sean producto del azar, sino que reflejen el verdadero comportamiento del algoritmo bajo distintas condiciones.

La librería ha sido diseñada para que estos procesos puedan ser indicados fácilmente en los propios ficheros de configuración, facilitando la tarea de llevar a cabo estudios experimentales sólidos y aplicables en situaciones reales. Esto contribuye a la validez y reproducibilidad de los experimentos.

## 10.6. Resultados de la experimentación

En esta sección se presentan los resultados obtenidos de la experimentación llevada a cabo.

Primero, se analizará algunos de los principales parámetros de cada algoritmo, tal y como se ha detallado en la sección 10.5 de configuración de los algoritmos considerando su influencia en los diferentes conjuntos de datos. A continuación, se comparará los mejores resultados obtenidos por cada algoritmo con el resto de algoritmos para cada uno de los cursos y se discutirán los resultados. Finalmente, se analizarán los resultados de los algoritmos considerando todos los cursos en general y se aplicarán test estadísticos que nos determinen si existen diferencias significativas en los resultados de las diferentes propuestas.

### 10.6.1. Comparación parámetros de los algoritmos

En esta sección, se estudiarán los resultados obtenidos por cada uno de los algoritmos en cada uno de los cursos, analizando los diferentes valores de los parámetros que se han determinado en la configuración de cada algoritmo (sección 10.5). Todos los resultados mostrados en las tablas se corresponden con los resultados



medios considerando las diferentes ejecuciones y particiones de cada conjunto de datos.

### 10.6.1.1. Curso AAA

En esta sección se van a mostrar los resultados para el curso AAA por los diferentes algoritmos con las diferentes configuraciones consideradas.

#### Autoencoder

En la tabla 3 se muestran los resultados obtenidos por el algoritmo autoencoder tanto para test como para train con los diferentes valores de dimensión de la capa latente considerados que eran 16, 32, 64, 128, 256. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. En la tabla se muestra las diferentes opciones con respecto al valor concreto que se ha usado para determinar la dimensión de la capa latente, disminuir este valor puede ser útil para reducir la dimensionalidad y eliminar el ruido, pero también puede resultar en una pérdida de información. Este algoritmo, al utilizar únicamente valores normales durante la fase de entrenamiento, no permite calcular varias de las métricas comúnmente utilizadas en esta etapa. Sin embargo, al analizar los resultados en la fase de prueba, se observa que los diferentes valores de los parámetros producen resultados similares. No obstante, los valores más reducidos tienden a mostrar un mejor rendimiento. En particular, un valor que utiliza una representación de 16 dimensiones demuestra obtener los mejores resultados. En general, se produce un error mayor en la detección de las anomalías, siendo los valores de sensibilidad más reducidos, indicando que existen anomalías que no es capaz de identificar correctamente. Con respecto a la desviación típica se muestra reducida, con lo que sus resultados son estables en las diferentes ejecuciones.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 16    | test  | 149.6   | 31.0 | 29.0103     | 0.5779±0.0432 | 0.9527±0.0173 | 0.8343±0.0609 | 0.7653±0.0302 |
| 32    | test  | 149.6   | 31.0 | 29.0103     | 0.571±0.0332  | 0.9498±0.0137 | 0.8244±0.0478 | 0.7604±0.0232 |
| 64    | test  | 149.6   | 31.0 | 29.0103     | 0.5638±0.0376 | 0.9464±0.0156 | 0.8128±0.0533 | 0.7551±0.0264 |
| 128   | test  | 149.6   | 31.0 | 29.0103     | 0.5533±0.0435 | 0.9422±0.0177 | 0.7981±0.0606 | 0.7478±0.0304 |
| 256   | test  | 149.6   | 31.0 | 29.0103     | 0.5496±0.0433 | 0.9407±0.0164 | 0.7926±0.057  | 0.7452±0.0297 |
| 16    | train | 424.8   | 31.0 | 0.0         | 0.0±0.0       | 0.7994±0.0002 | 0.0±0.0       | nan±nan       |
| 32    | train | 424.8   | 31.0 | 0.0         | 0.0±0.0       | 0.7994±0.0002 | 0.0±0.0       | nan±nan       |
| 64    | train | 424.8   | 31.0 | 0.0         | 0.0±0.0       | 0.7994±0.0002 | 0.0±0.0       | nan±nan       |
| 128   | train | 424.8   | 31.0 | 0.0         | 0.0±0.0       | 0.7994±0.0002 | 0.0±0.0       | nan±nan       |
| 256   | train | 424.8   | 31.0 | 0.0         | 0.0±0.0       | 0.7994±0.0002 | 0.0±0.0       | nan±nan       |

Figura 3: Estudio parámetros del algoritmo Autoencoder para el curso AAA



**Cblof**

En la tabla 4 se muestran los resultados obtenidos por el algoritmo cblof tanto para test como para train con los diferentes números de clústeres considerados, que eran 6, 7, 8, 9 y 10. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar es muy reducido, siendo el valor de sensibilidad solamente entre un 18 % y 19 %. Los diferentes valores de clústeres produce resultados parecidos, obteniendo ligeramente mejores valores cuanto más reducido es el número de clústeres.

| value | type  | samples  | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|----------|------|-------------|---------------|---------------|---------------|---------------|
| 6     | test  | 149.5833 | 31.0 | 29.0107     | 0.1886±0.0732 | 0.9355±0.0189 | 0.5352±0.087  | 0.562±0.0329  |
| 7     | test  | 149.5833 | 31.0 | 29.0107     | 0.1871±0.072  | 0.9255±0.0195 | 0.4954±0.1046 | 0.5563±0.035  |
| 8     | test  | 149.5833 | 31.0 | 29.0107     | 0.1819±0.0738 | 0.9233±0.0203 | 0.4791±0.1036 | 0.5526±0.0364 |
| 9     | test  | 149.5918 | 31.0 | 29.0173     | 0.1914±0.0764 | 0.9255±0.0173 | 0.4964±0.1086 | 0.5584±0.0385 |
| 10    | test  | 149.5918 | 31.0 | 29.0173     | 0.1944±0.0735 | 0.9297±0.0123 | 0.515±0.0975  | 0.562±0.0362  |
| 6     | train | 598.4167 | 31.0 | 29.0106     | 0.1846±0.0099 | 0.9342±0.0041 | 0.534±0.0296  | 0.5594±0.007  |
| 7     | train | 598.4082 | 31.0 | 29.0123     | 0.1732±0.0138 | 0.9295±0.0056 | 0.501±0.0396  | 0.5513±0.0097 |
| 8     | train | 598.4167 | 31.0 | 29.0106     | 0.1673±0.0191 | 0.9271±0.0078 | 0.484±0.0549  | 0.5472±0.0134 |
| 9     | train | 598.4082 | 31.0 | 29.0089     | 0.1744±0.0152 | 0.93±0.0062   | 0.5044±0.0439 | 0.5522±0.0107 |
| 10    | train | 598.4082 | 31.0 | 29.0089     | 0.1803±0.0156 | 0.9325±0.0064 | 0.5218±0.0457 | 0.5564±0.011  |

Figura 4: Estudio parámetros del algoritmo CBLOF para el curso AAA

**Cof**

En la tabla 5 se muestran los resultados obtenidos por el algoritmo cof tanto para test como para train con los diferentes números de vecinos considerados que eran 5, 10, 20, 30 y 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar supera el 30 % manteniendo una alta especificidad. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 30 y 40 vecinos consigue mejores resultados frente a las otras alternativas.

**Dae**



| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 149.6   | 31.0 | 29.0103     | 0.253±0.0765  | 0.8983±0.0258 | 0.4992±0.1183 | 0.5756±0.0422 |
| 10    | test  | 149.6   | 31.0 | 29.0103     | 0.2394±0.0783 | 0.9133±0.0279 | 0.5237±0.1242 | 0.5764±0.0452 |
| 20    | test  | 149.6   | 31.0 | 29.0103     | 0.3356±0.0719 | 0.9021±0.0045 | 0.5774±0.0545 | 0.6189±0.0357 |
| 30    | test  | 149.6   | 31.0 | 29.0103     | 0.3216±0.0948 | 0.9266±0.0141 | 0.6337±0.0583 | 0.624±0.0439  |
| 40    | test  | 149.6   | 31.0 | 29.0103     | 0.3266±0.1409 | 0.9266±0.0225 | 0.6287±0.0894 | 0.6266±0.063  |
| 5     | train | 598.4   | 31.0 | 29.0107     | 0.2004±0.014  | 0.9407±0.0059 | 0.58±0.0418   | 0.5706±0.01   |
| 10    | train | 598.4   | 31.0 | 29.0107     | 0.2074±0.0062 | 0.9435±0.0026 | 0.6±0.0184    | 0.5754±0.0044 |
| 20    | train | 598.4   | 31.0 | 29.0107     | 0.1959±0.0215 | 0.9388±0.0087 | 0.5667±0.0621 | 0.5673±0.0151 |
| 30    | train | 598.4   | 31.0 | 29.0107     | 0.1912±0.0169 | 0.9369±0.0069 | 0.5533±0.0492 | 0.5641±0.0119 |
| 40    | train | 598.4   | 31.0 | 29.0107     | 0.1947±0.0071 | 0.9383±0.0028 | 0.5633±0.0196 | 0.5665±0.0049 |

Figura 5: Estudio parámetros del algoritmo COF para el curso AAA

En la tabla 6 se muestran los resultados obtenidos por el algoritmo dae tanto para test como para train con los diferentes valores de dimensión de la capa latente considerados que eran 16, 32, 64, 128, 256. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test en este caso se presentan diferentes, debido a que este algoritmo se entrena solamente con valores normales, por tanto, durante la fase de train nos encontramos con métricas que no pueden calcularse. Con respecto a la desviación típica se muestra reducida, con lo que sus resultados son estables. En este caso, el número de anomalías que es capaz de detectar supera el 60 % manteniendo una alta especificidad. Se aprecia que resultados similares son obtenidos a partir del uso de un tamaño de dimensión igual a 32, no afectando mucho el valor de este parámetro en los resultados.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 16    | test  | 149.6   | 31.0 | 29.0103     | 0.641±0.0398  | 0.9788±0.0165 | 0.9255±0.0581 | 0.8099±0.028  |
| 32    | test  | 149.6   | 31.0 | 29.0103     | 0.6489±0.0376 | 0.9822±0.0157 | 0.9374±0.0552 | 0.8155±0.0265 |
| 64    | test  | 149.6   | 31.0 | 29.0103     | 0.6475±0.0369 | 0.9818±0.0154 | 0.9361±0.0541 | 0.8146±0.026  |
| 128   | test  | 149.6   | 31.0 | 29.0103     | 0.6475±0.0361 | 0.9816±0.0151 | 0.9355±0.0531 | 0.8145±0.0255 |
| 256   | test  | 149.6   | 31.0 | 29.0103     | 0.648±0.0333  | 0.9814±0.0144 | 0.935±0.0504  | 0.8147±0.0236 |
| 16    | train | 424.8   | 31.0 | 0.0         | 0.0±0.0       | 0.7994±0.0002 | 0.0±0.0       | nan±nan       |
| 32    | train | 424.8   | 31.0 | 0.0         | 0.0±0.0       | 0.7994±0.0002 | 0.0±0.0       | nan±nan       |
| 64    | train | 424.8   | 31.0 | 0.0         | 0.0±0.0       | 0.7994±0.0002 | 0.0±0.0       | nan±nan       |
| 128   | train | 424.8   | 31.0 | 0.0         | 0.0±0.0       | 0.7994±0.0002 | 0.0±0.0       | nan±nan       |
| 256   | train | 424.8   | 31.0 | 0.0         | 0.0±0.0       | 0.7994±0.0002 | 0.0±0.0       | nan±nan       |

Figura 6: Estudio parámetros del algoritmo DAE para el curso AAA

## Hbos

En la tabla 7 se muestran los resultados obtenidos por el algoritmo hbos tanto para test como para train con los diferentes tamaños de los intervalos considerados que



eran 5, 10, 15, 20, 25. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar no alcanza ni el 10 % siendo también la especificidad más reducida que en las anteriores, no alcanzando el 90 %. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de intervalos considerados, aunque se trata de resultados con un rendimiento muy bajo para detectar las anomalías.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 149.6   | 31.0 | 29.0103     | 0.0692±0.0392 | 0.8851±0.0359 | 0.1917±0.087  | 0.4772±0.0225 |
| 10    | test  | 149.6   | 31.0 | 29.0103     | 0.0691±0.0295 | 0.8775±0.0468 | 0.2024±0.0842 | 0.4733±0.028  |
| 15    | test  | 149.6   | 31.0 | 29.0103     | 0.0738±0.0345 | 0.8795±0.0322 | 0.2009±0.0777 | 0.4766±0.0215 |
| 20    | test  | 149.6   | 31.0 | 29.0103     | 0.0784±0.041  | 0.8719±0.0394 | 0.2045±0.0885 | 0.4752±0.0288 |
| 25    | test  | 149.6   | 31.0 | 29.0103     | 0.0921±0.039  | 0.8833±0.0343 | 0.2567±0.1294 | 0.4877±0.0301 |
| 5     | train | 598.4   | 31.0 | 29.0107     | 0.0714±0.0114 | 0.8879±0.0047 | 0.2067±0.033  | 0.4797±0.008  |
| 10    | train | 598.4   | 31.0 | 29.0107     | 0.0691±0.0074 | 0.887±0.0031  | 0.2±0.0213    | 0.4781±0.0052 |
| 15    | train | 598.4   | 31.0 | 29.0107     | 0.0726±0.007  | 0.8884±0.0029 | 0.21±0.0202   | 0.4805±0.005  |
| 20    | train | 598.4   | 31.0 | 29.0107     | 0.0703±0.0086 | 0.8875±0.0036 | 0.2033±0.0247 | 0.4789±0.0061 |
| 25    | train | 598.4   | 31.0 | 29.0107     | 0.0737±0.0077 | 0.8889±0.0032 | 0.2133±0.0223 | 0.4813±0.0055 |

Figura 7: Estudio parámetros del algoritmo HBOS para el curso AAA

## Knn

En la tabla 8 se muestran los resultados obtenidos por el algoritmo knn tanto para test como para train con los diferentes números de vecinos considerados que eran 5, 10, 20, 30 y 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar es más elevado que la detección de caso normales, alcanzando el 100 % de anomalías detectadas, aunque solamente se alcanza valores cercanos al 70 % de identificación de casos normales. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 40 vecinos consigue mejores resultados frente a las otras alternativas. Se resalta la capacidad de detectar anomalías frente a las otras propuestas.

## Lof



| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 149.6   | 31.0 | 29.0103     | 0.8574±0.0467 | 0.6477±0.0916 | 0.5066±0.0643 | 0.7525±0.0468 |
| 10    | test  | 149.6   | 31.0 | 29.0103     | 0.9539±0.0208 | 0.6704±0.064  | 0.546±0.0506  | 0.8122±0.0332 |
| 20    | test  | 149.6   | 31.0 | 29.0103     | 1.0±0.0       | 0.6797±0.0308 | 0.5616±0.0244 | 0.8399±0.0154 |
| 30    | test  | 149.6   | 31.0 | 29.0103     | 1.0±0.0       | 0.7042±0.0401 | 0.582±0.033   | 0.8521±0.02   |
| 40    | test  | 149.6   | 31.0 | 29.0103     | 1.0±0.0       | 0.7363±0.0937 | 0.6184±0.0785 | 0.8681±0.0468 |
| 5     | train | 598.4   | 31.0 | 29.0107     | 0.8307±0.0128 | 0.6784±0.008  | 0.5136±0.0086 | 0.7546±0.0088 |
| 10    | train | 598.4   | 31.0 | 29.0107     | 0.9113±0.0301 | 0.6958±0.0209 | 0.5508±0.0231 | 0.8036±0.0242 |
| 20    | train | 598.4   | 31.0 | 29.0107     | 1.0±0.0       | 0.7114±0.0022 | 0.5861±0.0026 | 0.8557±0.0011 |
| 30    | train | 598.4   | 31.0 | 29.0107     | 1.0±0.0       | 0.7194±0.0114 | 0.5931±0.0097 | 0.8597±0.0057 |
| 40    | train | 598.4   | 31.0 | 29.0107     | 1.0±0.0       | 0.7425±0.0261 | 0.6143±0.0238 | 0.8712±0.013  |

Figura 8: Estudio parámetros del algoritmo Knn para el curso AAA

En la tabla 9 se muestran los resultados obtenidos por el algoritmo lof tanto para test como para train con los diferentes parámetros considerados que eran 5, 10, 20, 30, 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar está entorno al 20% manteniendo una alta especificidad. Se aprecia que mejores resultados son alcanzados conforme disminuye el número de vecinos considerados, en este caso el uso de 5 vecinos consigue mejores resultados frente a las otras alternativas.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 149.6   | 31.0 | 29.0103     | 0.2402±0.0636 | 0.9398±0.0222 | 0.6202±0.1174 | 0.59±0.0337   |
| 10    | test  | 149.6   | 31.0 | 29.0103     | 0.199±0.0933  | 0.9454±0.0185 | 0.5839±0.0711 | 0.5722±0.0385 |
| 20    | test  | 149.6   | 31.0 | 29.0103     | 0.1569±0.0621 | 0.936±0.0284  | 0.5088±0.1723 | 0.5464±0.0367 |
| 30    | test  | 149.6   | 31.0 | 29.0103     | 0.1793±0.0747 | 0.9229±0.0256 | 0.4787±0.1802 | 0.5511±0.0424 |
| 40    | test  | 149.6   | 31.0 | 29.0103     | 0.17±0.0669   | 0.9266±0.02   | 0.4773±0.1351 | 0.5483±0.0353 |
| 5     | train | 598.4   | 31.0 | 29.0107     | 0.1532±0.0117 | 0.9576±0.0015 | 0.5956±0.0238 | 0.5554±0.0062 |
| 10    | train | 598.4   | 31.0 | 29.0107     | 0.1659±0.0206 | 0.952±0.017   | 0.5974±0.0986 | 0.5589±0.0137 |
| 20    | train | 598.4   | 31.0 | 29.0107     | 0.1451±0.0101 | 0.9355±0.0076 | 0.4805±0.0391 | 0.5403±0.0072 |
| 30    | train | 598.4   | 31.0 | 29.0107     | 0.1555±0.0096 | 0.9351±0.0063 | 0.4953±0.037  | 0.5453±0.0072 |
| 40    | train | 598.4   | 31.0 | 29.0107     | 0.1659±0.0115 | 0.9322±0.0038 | 0.4998±0.0301 | 0.549±0.0073  |

Figura 9: Estudio parámetros del algoritmo LOF para el curso AAA

### 10.6.1.2. Curso BBB

En esta sección se van a mostrar los resultados para el curso BBB por los diferentes algoritmos con las diferentes configuraciones consideradas.

#### Autoencoder



## CAPÍTULO 10. RESULTADOS EXPERIMENTALES

En la tabla 10 se muestran los resultados obtenidos por el algoritmo autoencoder tanto para test como para train con los diferentes valores de dimensión de la capa latente considerados que eran 16, 32, 64, 128, 256. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. En la tabla se muestra las diferentes opciones con respecto al valor concreto que se ha usado para determinar la dimensión de la capa latente, disminuir este valor puede ser útil para reducir la dimensionalidad y eliminar el ruido, pero también puede resultar en una pérdida de información. Este algoritmo, al utilizar únicamente valores normales durante la fase de entrenamiento, no permite calcular varias de las métricas comúnmente utilizadas en esta etapa. Sin embargo, al analizar los resultados en la fase de prueba, se observa que los diferentes valores de los parámetros producen resultados similares. No obstante, los valores más reducidos tienden a mostrar un mejor rendimiento. En particular, un valor que utiliza una representación de 16 dimensiones demuestra obtener los mejores resultados. En general, se produce un error mayor en la detección de las anomalías, siendo los valores de sensibilidad más reducidos, indicando que existen anomalías que no es capaz de identificar correctamente. Con respecto a la desviación típica se muestra reducida, con lo que sus resultados son estables en las diferentes ejecuciones.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 16    | test  | 1581.8  | 90.0 | 52.5351     | 0.3458±0.0389 | 0.949±0.0351  | 0.8818±0.0833 | 0.6474±0.0356 |
| 32    | test  | 1581.8  | 90.0 | 52.5351     | 0.3333±0.0296 | 0.9359±0.0228 | 0.8515±0.0538 | 0.6346±0.0242 |
| 64    | test  | 1581.8  | 90.0 | 52.5351     | 0.2975±0.0316 | 0.9013±0.0283 | 0.7687±0.0676 | 0.5994±0.0287 |
| 128   | test  | 1581.8  | 90.0 | 52.5351     | 0.2856±0.0361 | 0.8845±0.028  | 0.7311±0.0684 | 0.5851±0.03   |
| 256   | test  | 1581.8  | 90.0 | 52.5351     | 0.2814±0.0231 | 0.8852±0.0295 | 0.7312±0.064  | 0.5833±0.0254 |
| 16    | train | 3003.2  | 90.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 32    | train | 3003.2  | 90.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 64    | train | 3003.2  | 90.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 128   | train | 3003.2  | 90.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 256   | train | 3003.2  | 90.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |

Figura 10: Estudio parámetros del algoritmo Autoencoder para el curso BBB

## Cblof

En la tabla 11 se muestran los resultados obtenidos por el algoritmo cblof tanto para test como para train con los diferentes números de clústeres considerados, que eran 6, 7, 8, 9 y 10. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de



## CAPÍTULO 10. RESULTADOS EXPERIMENTALES

anomalías que es capaz de detectar es muy reducido, siendo el valor de sensibilidad solamente entre un 8 % y 12 %. Los diferentes valores de clústeres produce resultados parecidos, obteniendo ligeramente mejores valores cuanto más reducido es el número de clústeres.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 6     | test  | 1581.8  | 90.0 | 52.5351     | 0.1206±0.06   | 0.9065±0.0619 | 0.6024±0.0796 | 0.5136±0.0147 |
| 7     | test  | 1581.8  | 90.0 | 52.5351     | 0.1196±0.0632 | 0.894±0.0679  | 0.5672±0.154  | 0.5068±0.033  |
| 8     | test  | 1581.8  | 90.0 | 52.5351     | 0.111±0.0565  | 0.8738±0.0849 | 0.5212±0.1915 | 0.4924±0.0484 |
| 9     | test  | 1581.8  | 90.0 | 52.5351     | 0.0961±0.0558 | 0.8228±0.1094 | 0.4177±0.2359 | 0.4594±0.0691 |
| 10    | test  | 1581.8  | 90.0 | 52.5351     | 0.082±0.0562  | 0.7946±0.1094 | 0.3343±0.2158 | 0.4383±0.0683 |
| 6     | train | 6327.2  | 90.0 | 52.535      | 0.1165±0.0085 | 0.9182±0.0095 | 0.612±0.0448  | 0.5174±0.009  |
| 7     | train | 6327.2  | 90.0 | 52.535      | 0.1127±0.0153 | 0.9141±0.017  | 0.5923±0.0807 | 0.5134±0.0162 |
| 8     | train | 6327.2  | 90.0 | 52.535      | 0.1046±0.0256 | 0.9051±0.0284 | 0.5496±0.1348 | 0.5049±0.027  |
| 9     | train | 6327.2  | 90.0 | 52.535      | 0.0883±0.0331 | 0.8871±0.0366 | 0.464±0.174   | 0.4877±0.0349 |
| 10    | train | 6327.2  | 90.0 | 52.535      | 0.0737±0.0325 | 0.8709±0.036  | 0.3873±0.1709 | 0.4723±0.0343 |

Figura 11: Estudio parámetros del algoritmo CBLOF para el curso BBB

## Cof

En la tabla 12 se muestran los resultados obtenidos por el algoritmo cof tanto para test como para train con los diferentes números de vecinos considerados que eran 5, 10, 20, 30 y 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar supera el 25 % manteniendo una alta especificidad. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 30 y 40 vecinos consigue mejores resultados frente a las otras alternativas.

## Dae

En la tabla 13 se muestran los resultados obtenidos por el algoritmo dae tanto para test como para train con los diferentes valores de dimensión de la capa latente considerados que eran 16, 32, 64, 128, 256. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test en este caso se presentan diferentes, debido a que este algoritmo se entrena solamente con valores



| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 1581.8  | 90.0 | 52.5351     | 0.1921±0.0458 | 0.9241±0.0195 | 0.7371±0.016  | 0.5581±0.0139 |
| 10    | test  | 1581.8  | 90.0 | 52.5351     | 0.213±0.0544  | 0.9339±0.0161 | 0.7798±0.0233 | 0.5735±0.0201 |
| 20    | test  | 1581.8  | 90.0 | 52.5351     | 0.2274±0.0646 | 0.9539±0.0139 | 0.8452±0.0213 | 0.5907±0.0265 |
| 30    | test  | 1581.8  | 90.0 | 52.5351     | 0.2397±0.0502 | 0.9675±0.0116 | 0.8939±0.0145 | 0.6036±0.0194 |
| 40    | test  | 1581.8  | 90.0 | 52.5351     | 0.2696±0.0368 | 0.9664±0.0174 | 0.9035±0.0382 | 0.618±0.0134  |
| 5     | train | 6327.2  | 90.0 | 52.535      | 0.1533±0.0022 | 0.9589±0.0024 | 0.805±0.0113  | 0.5561±0.0023 |
| 10    | train | 6327.2  | 90.0 | 52.535      | 0.1611±0.0023 | 0.9676±0.0026 | 0.8461±0.0121 | 0.5643±0.0024 |
| 20    | train | 6327.2  | 90.0 | 52.535      | 0.1697±0.0026 | 0.9773±0.0028 | 0.8921±0.0134 | 0.5735±0.0027 |
| 30    | train | 6327.2  | 90.0 | 52.535      | 0.1688±0.0024 | 0.976±0.0026  | 0.8862±0.0125 | 0.5724±0.0025 |
| 40    | train | 6327.2  | 90.0 | 52.535      | 0.165±0.0033  | 0.9718±0.0036 | 0.8664±0.0173 | 0.5684±0.0035 |

Figura 12: Estudio parámetros del algoritmo COF para el curso BBB

normales, por tanto, durante la fase de train nos encontramos con métricas que no pueden calcularse. Con respecto a la desviación típica se muestra reducida, con lo que sus resultados son estables. En este caso, el número de anomalías que es capaz de detectar se queda en el 41 % manteniendo una alta especificidad. Se puede suponer que es debido al número de anomalías que se presenta en este curso. Se aprecia que resultados similares son obtenidos a partir del uso de un tamaño de dimensión igual a 32, no afectando mucho el valor de este parámetro en los resultados.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 16    | test  | 1581.8  | 90.0 | 52.5351     | 0.3994±0.0249 | 0.9988±0.0013 | 0.9972±0.0031 | 0.6991±0.0126 |
| 32    | test  | 1581.8  | 90.0 | 52.5351     | 0.4023±0.0272 | 0.9988±0.0013 | 0.9973±0.003  | 0.7006±0.0137 |
| 64    | test  | 1581.8  | 90.0 | 52.5351     | 0.4066±0.0279 | 0.9988±0.0015 | 0.9972±0.0036 | 0.7027±0.0142 |
| 128   | test  | 1581.8  | 90.0 | 52.5351     | 0.4081±0.0261 | 0.9988±0.0015 | 0.9973±0.0036 | 0.7035±0.0133 |
| 256   | test  | 1581.8  | 90.0 | 52.5351     | 0.4103±0.0267 | 0.9989±0.0013 | 0.9975±0.003  | 0.7046±0.0136 |
| 16    | train | 3003.2  | 90.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 32    | train | 3003.2  | 90.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 64    | train | 3003.2  | 90.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 128   | train | 3003.2  | 90.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 256   | train | 3003.2  | 90.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |

Figura 13: Estudio parámetros del algoritmo DAE para el curso BBB

## Hbos

En la tabla 14 se muestran los resultados obtenidos por el algoritmo hbos tanto para test como para train con los diferentes tamaños de los intervalos considerados que eran 5, 10, 15, 20, 25. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar no alcanza ni el 5 % siendo





## CAPÍTULO 10. RESULTADOS EXPERIMENTALES

también la especificidad más reducida que en las anteriores, no alcanzando el 80 %. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de intervalos considerados, aunque se trata de resultados con un rendimiento muy bajo para detectar las anomalías.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 1581.8  | 90.0 | 52.5351     | 0.0479±0.0291 | 0.6849±0.1626 | 0.1458±0.0315 | 0.3664±0.0676 |
| 10    | test  | 1581.8  | 90.0 | 52.5351     | 0.046±0.0261  | 0.7039±0.1486 | 0.1475±0.0256 | 0.3749±0.0619 |
| 15    | test  | 1581.8  | 90.0 | 52.5351     | 0.0467±0.0231 | 0.7052±0.1454 | 0.1535±0.0272 | 0.376±0.0617  |
| 20    | test  | 1581.8  | 90.0 | 52.5351     | 0.045±0.0216  | 0.7126±0.1379 | 0.1501±0.0166 | 0.3788±0.0583 |
| 25    | test  | 1581.8  | 90.0 | 52.5351     | 0.0464±0.0226 | 0.7102±0.1405 | 0.1536±0.0248 | 0.3784±0.0595 |
| 5     | train | 6327.2  | 90.0 | 52.535      | 0.0348±0.0019 | 0.8278±0.0021 | 0.1829±0.0099 | 0.4313±0.002  |
| 10    | train | 6327.2  | 90.0 | 52.535      | 0.0328±0.0011 | 0.8255±0.0012 | 0.1722±0.0057 | 0.4291±0.0012 |
| 15    | train | 6327.2  | 90.0 | 52.535      | 0.0335±0.002  | 0.8262±0.0023 | 0.1757±0.0108 | 0.4299±0.0022 |
| 20    | train | 6327.2  | 90.0 | 52.535      | 0.0319±0.0023 | 0.8245±0.0026 | 0.1675±0.0123 | 0.4282±0.0025 |
| 25    | train | 6327.2  | 90.0 | 52.535      | 0.0321±0.0023 | 0.8247±0.0026 | 0.1684±0.0124 | 0.4284±0.0025 |

Figura 14: Estudio parámetros del algoritmo HBOS para el curso BBB

## Knn

En la tabla 15 se muestran los resultados obtenidos por el algoritmo knn tanto para test como para train con los diferentes números de vecinos considerados que eran 5, 10, 20, 30 y 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar es similar a la detección de caso normales, alcanzando el 56 % de anomalías detectadas y valores cercanos al 52 % de identificación de casos normales. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 40 vecinos consigue mejores resultados frente a las otras alternativas.

## Lof

En la tabla 16 se muestran los resultados obtenidos por el algoritmo lof tanto para test como para train con los diferentes parámetros considerados que eran 5, 10, 20, 30, 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son





| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 1581.8  | 90.0 | 52.5351     | 0.4171±0.103  | 0.3671±0.1567 | 0.4239±0.0356 | 0.3921±0.0434 |
| 10    | test  | 1581.8  | 90.0 | 52.5351     | 0.4568±0.1004 | 0.4004±0.1679 | 0.4639±0.0518 | 0.4286±0.0527 |
| 20    | test  | 1581.8  | 90.0 | 52.5351     | 0.5133±0.0831 | 0.446±0.1639  | 0.5154±0.0585 | 0.4797±0.0535 |
| 30    | test  | 1581.8  | 90.0 | 52.5351     | 0.5512±0.0752 | 0.4892±0.1587 | 0.5541±0.0662 | 0.5201±0.0574 |
| 40    | test  | 1581.8  | 90.0 | 52.5351     | 0.5603±0.0836 | 0.5224±0.1527 | 0.5744±0.0623 | 0.5414±0.0489 |
| 5     | train | 6327.2  | 90.0 | 52.535      | 0.3818±0.0138 | 0.407±0.0164  | 0.4162±0.0155 | 0.3945±0.015  |
| 10    | train | 6327.2  | 90.0 | 52.535      | 0.4211±0.0183 | 0.4358±0.0226 | 0.4524±0.0206 | 0.4284±0.0204 |
| 20    | train | 6327.2  | 90.0 | 52.535      | 0.4719±0.0211 | 0.4778±0.0227 | 0.5±0.022     | 0.4749±0.0219 |
| 30    | train | 6327.2  | 90.0 | 52.535      | 0.5093±0.0178 | 0.5252±0.0182 | 0.5428±0.018  | 0.5172±0.0178 |
| 40    | train | 6327.2  | 90.0 | 52.535      | 0.5403±0.0125 | 0.5506±0.0148 | 0.571±0.0137  | 0.5454±0.0136 |

Figura 15: Estudio parámetros del algoritmo Knn para el curso BBB

reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar está entorno al 20% manteniendo una alta especificidad. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 40 vecinos consigue mejores resultados frente a las otras alternativas.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 1581.8  | 90.0 | 52.5351     | 0.1742±0.0442 | 0.9374±0.0263 | 0.7615±0.0288 | 0.5558±0.0107 |
| 10    | test  | 1581.8  | 90.0 | 52.5351     | 0.1764±0.0578 | 0.9427±0.0291 | 0.7804±0.045  | 0.5596±0.0171 |
| 20    | test  | 1581.8  | 90.0 | 52.5351     | 0.1959±0.0521 | 0.9515±0.0268 | 0.8279±0.0523 | 0.5737±0.0154 |
| 30    | test  | 1581.8  | 90.0 | 52.5351     | 0.1998±0.0448 | 0.9638±0.0342 | 0.8801±0.0688 | 0.5818±0.0073 |
| 40    | test  | 1581.8  | 90.0 | 52.5351     | 0.2115±0.0435 | 0.9648±0.0348 | 0.8903±0.0725 | 0.5882±0.0086 |
| 5     | train | 6327.2  | 90.0 | 52.535      | 0.1132±0.0043 | 0.9792±0.0018 | 0.8572±0.0139 | 0.5462±0.0029 |
| 10    | train | 6327.2  | 90.0 | 52.535      | 0.1337±0.0007 | 0.9773±0.0051 | 0.8677±0.0251 | 0.5555±0.0025 |
| 20    | train | 6327.2  | 90.0 | 52.535      | 0.1516±0.0031 | 0.9755±0.004  | 0.8727±0.0194 | 0.5636±0.0033 |
| 30    | train | 6327.2  | 90.0 | 52.535      | 0.1604±0.0045 | 0.9783±0.0048 | 0.891±0.024   | 0.5694±0.0046 |
| 40    | train | 6327.2  | 90.0 | 52.535      | 0.1632±0.0043 | 0.9769±0.0056 | 0.8867±0.0266 | 0.57±0.0048   |

Figura 16: Estudio parámetros del algoritmo LOF para el curso BBB

### 10.6.1.3. Curso CCC

En esta sección se van a mostrar los resultados para el curso CCC por los diferentes algoritmos con las diferentes configuraciones consideradas.

#### Autoencoder

En la tabla 17 se muestran los resultados obtenidos por el algoritmo autoencoder tanto para test como para train con los diferentes valores de dimensión de la capa latente considerados que eran 16, 32, 64, 128, 256. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se



muestran las tablas completas por fold y semilla. En la tabla se muestra las diferentes opciones con respecto al valor concreto que se ha usado para determinar la dimensión de la capa latente, disminuir este valor puede ser útil para reducir la dimensionalidad y eliminar el ruido, pero también puede resultar en una pérdida de información. Este algoritmo, al utilizar únicamente valores normales durante la fase de entrenamiento, no permite calcular varias de las métricas comúnmente utilizadas en esta etapa. Sin embargo, al analizar los resultados en la fase de prueba, se observa que los diferentes valores de los parámetros producen resultados similares. No obstante, los valores más reducidos tienden a mostrar un mejor rendimiento. En particular, un valor que utiliza una representación de 16 dimensiones demuestra obtener los mejores resultados. En general, se produce un error mayor en la detección de las anomalías, siendo los valores de sensibilidad más reducidos, indicando que existen anomalías que no es capaz de identificar correctamente. Con respecto a la desviación típica se muestra reducida, con lo que sus resultados son estables en las diferentes ejecuciones.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 16    | test  | 886.8   | 47.0 | 62.1561     | 0.3031±0.0168 | 0.964±0.0231  | 0.9323±0.0435 | 0.6335±0.0194 |
| 32    | test  | 886.8   | 47.0 | 62.1561     | 0.2831±0.0148 | 0.9344±0.0247 | 0.8764±0.0465 | 0.6088±0.0197 |
| 64    | test  | 886.8   | 47.0 | 62.1561     | 0.2803±0.0118 | 0.9298±0.0199 | 0.8676±0.0374 | 0.605±0.0158  |
| 128   | test  | 886.8   | 47.0 | 62.1561     | 0.2764±0.0127 | 0.9235±0.0215 | 0.8557±0.0404 | 0.6±0.0171    |
| 256   | test  | 886.8   | 47.0 | 62.1561     | 0.2746±0.014  | 0.9202±0.0234 | 0.8496±0.0441 | 0.5974±0.0187 |
| 16    | train | 1342.4  | 47.0 | 0.0         | 0.0±0.0       | 0.7995±0.0001 | 0.0±0.0       | nan±nan       |
| 32    | train | 1342.4  | 47.0 | 0.0         | 0.0±0.0       | 0.7995±0.0001 | 0.0±0.0       | nan±nan       |
| 64    | train | 1342.4  | 47.0 | 0.0         | 0.0±0.0       | 0.7995±0.0001 | 0.0±0.0       | nan±nan       |
| 128   | train | 1342.4  | 47.0 | 0.0         | 0.0±0.0       | 0.7995±0.0001 | 0.0±0.0       | nan±nan       |
| 256   | train | 1342.4  | 47.0 | 0.0         | 0.0±0.0       | 0.7995±0.0001 | 0.0±0.0       | nan±nan       |

Figura 17: Estudio parámetros del algoritmo Autoencoder para el curso CCC

## Cblof

En la tabla 18 se muestran los resultados obtenidos por el algoritmo cblof tanto para test como para train con los diferentes números de clústeres considerados, que eran 6, 7, 8, 9 y 10. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar es muy reducido, siendo el valor de sensibilidad solamente entre un 5 % y 10 %. Los diferentes valores de clústeres produce resultados parecidos, obteniendo ligeramente mejores valores cuanto más reducido es el número de clústeres.



| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 6     | test  | 886.8   | 47.0 | 62.1561     | 0.1059±0.0401 | 0.9052±0.0512 | 0.6625±0.0608 | 0.5055±0.0102 |
| 7     | test  | 886.8   | 47.0 | 62.1561     | 0.097±0.0432  | 0.8847±0.0611 | 0.5873±0.1432 | 0.4908±0.029  |
| 8     | test  | 886.8   | 47.0 | 62.1561     | 0.0715±0.037  | 0.8513±0.0673 | 0.448±0.1825  | 0.4614±0.0356 |
| 9     | test  | 886.8   | 47.0 | 62.1561     | 0.0557±0.0372 | 0.8157±0.0611 | 0.3188±0.1568 | 0.4357±0.0328 |
| 10    | test  | 886.8   | 47.0 | 62.1561     | 0.0502±0.0388 | 0.7971±0.065  | 0.2744±0.1688 | 0.4237±0.0383 |
| 6     | train | 3547.2  | 47.0 | 62.1561     | 0.1052±0.0094 | 0.9084±0.0154 | 0.6536±0.0583 | 0.5068±0.0124 |
| 7     | train | 3547.2  | 47.0 | 62.1561     | 0.0951±0.0186 | 0.8918±0.0306 | 0.5907±0.1157 | 0.4934±0.0246 |
| 8     | train | 3547.2  | 47.0 | 62.1561     | 0.0737±0.0262 | 0.8566±0.0431 | 0.4576±0.1628 | 0.4651±0.0346 |
| 9     | train | 3547.2  | 47.0 | 62.1561     | 0.0562±0.0192 | 0.8278±0.0316 | 0.3489±0.1195 | 0.442±0.0254  |
| 10    | train | 3547.2  | 47.0 | 62.1561     | 0.0482±0.022  | 0.8147±0.0361 | 0.2994±0.1364 | 0.4315±0.029  |

Figura 18: Estudio parámetros del algoritmo CBLOF para el curso CCC

### Cof

En la tabla 19 se muestran los resultados obtenidos por el algoritmo cof tanto para test como para train con los diferentes números de vecinos considerados que eran 5, 10, 20, 30 y 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar está entre un 20 % al 25 % manteniendo una alta especificidad. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 30 y 40 vecinos consigue mejores resultados frente a las otras alternativas.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 886.8   | 47.0 | 62.1561     | 0.2087±0.0246 | 0.9052±0.0184 | 0.7842±0.0251 | 0.5569±0.009  |
| 10    | test  | 886.8   | 47.0 | 62.1561     | 0.2351±0.0231 | 0.9374±0.0162 | 0.862±0.0286  | 0.5863±0.0092 |
| 20    | test  | 886.8   | 47.0 | 62.1561     | 0.2649±0.0299 | 0.9571±0.0211 | 0.9135±0.0366 | 0.611±0.0112  |
| 30    | test  | 886.8   | 47.0 | 62.1561     | 0.2638±0.0331 | 0.9696±0.0173 | 0.9363±0.0339 | 0.6167±0.0161 |
| 40    | test  | 886.8   | 47.0 | 62.1561     | 0.2434±0.0423 | 0.9779±0.0168 | 0.9509±0.0354 | 0.6107±0.019  |
| 5     | train | 3547.2  | 47.0 | 62.1561     | 0.1286±0.0019 | 0.9468±0.0032 | 0.7989±0.012  | 0.5377±0.0025 |
| 10    | train | 3547.2  | 47.0 | 62.1561     | 0.1345±0.0025 | 0.9565±0.004  | 0.8355±0.0152 | 0.5455±0.0033 |
| 20    | train | 3547.2  | 47.0 | 62.1561     | 0.1437±0.0035 | 0.9715±0.0058 | 0.8924±0.0219 | 0.5576±0.0047 |
| 30    | train | 3547.2  | 47.0 | 62.1561     | 0.1485±0.0024 | 0.9794±0.0039 | 0.9222±0.0147 | 0.564±0.0031  |
| 40    | train | 3547.2  | 47.0 | 62.1561     | 0.1471±0.0016 | 0.9772±0.0027 | 0.9138±0.0101 | 0.5622±0.0022 |

Figura 19: Estudio parámetros del algoritmo COF para el curso CCC

### Dae

En la tabla 20 se muestran los resultados obtenidos por el algoritmo dae tanto para test como para train con los diferentes valores de dimensión de la capa latente



considerados que eran 16, 32, 64, 128, 256. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test en este caso se presentan diferentes, debido a que este algoritmo se entrena solamente con valores normales, por tanto, durante la fase de train nos encontramos con métricas que no pueden calcularse. Con respecto a la desviación típica se muestra reducida, con lo que sus resultados son estables. En este caso, el número de anomalías que es capaz de detectar no llega ni al 40 % manteniendo una alta especificidad. Se aprecia que resultados similares son obtenidos a partir del uso de un tamaño de dimensión igual a 64, no afectando mucho el valor de este parámetro en los resultados.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 16    | test  | 886.8   | 47.0 | 62.1561     | 0.3242±0.0048 | 0.9994±0.0012 | 0.9989±0.0023 | 0.6618±0.0027 |
| 32    | test  | 886.8   | 47.0 | 62.1561     | 0.3249±0.0063 | 0.9994±0.0012 | 0.9989±0.0023 | 0.6622±0.0035 |
| 64    | test  | 886.8   | 47.0 | 62.1561     | 0.3256±0.0067 | 0.9994±0.0012 | 0.9989±0.0023 | 0.6625±0.0036 |
| 128   | test  | 886.8   | 47.0 | 62.1561     | 0.3262±0.0083 | 0.9994±0.0012 | 0.9989±0.0022 | 0.6628±0.0043 |
| 256   | test  | 886.8   | 47.0 | 62.1561     | 0.3284±0.0099 | 0.9995±0.0012 | 0.999±0.0021  | 0.6639±0.005  |
| 16    | train | 1342.4  | 47.0 | 0.0         | 0.0±0.0       | 0.7995±0.0001 | 0.0±0.0       | nan±nan       |
| 32    | train | 1342.4  | 47.0 | 0.0         | 0.0±0.0       | 0.7995±0.0001 | 0.0±0.0       | nan±nan       |
| 64    | train | 1342.4  | 47.0 | 0.0         | 0.0±0.0       | 0.7995±0.0001 | 0.0±0.0       | nan±nan       |
| 128   | train | 1342.4  | 47.0 | 0.0         | 0.0±0.0       | 0.7995±0.0001 | 0.0±0.0       | nan±nan       |
| 256   | train | 1342.4  | 47.0 | 0.0         | 0.0±0.0       | 0.7995±0.0001 | 0.0±0.0       | nan±nan       |

Figura 20: Estudio parámetros del algoritmo DAE para el curso CCC

## Hbos

En la tabla 21 se muestran los resultados obtenidos por el algoritmo hbos tanto para test como para train con los diferentes tamaños de los intervalos considerados que eran 5, 10, 15, 20, 25. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar no alcanza ni el 5 % siendo también la especificidad más reducida que en las anteriores, no alcanzando el 80 %. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de intervalos considerados, aunque se trata de resultados con un rendimiento muy bajo para detectar las anomalías.

## Knn



| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 886.8   | 47.0 | 62.1561     | 0.0294±0.012  | 0.7826±0.0896 | 0.1827±0.0611 | 0.406±0.0398  |
| 10    | test  | 886.8   | 47.0 | 62.1561     | 0.0348±0.0156 | 0.7772±0.0641 | 0.1975±0.0553 | 0.406±0.026   |
| 15    | test  | 886.8   | 47.0 | 62.1561     | 0.0341±0.0163 | 0.77±0.0558   | 0.1837±0.0609 | 0.4021±0.0211 |
| 20    | test  | 886.8   | 47.0 | 62.1561     | 0.0323±0.0142 | 0.7706±0.0498 | 0.1793±0.0453 | 0.4015±0.019  |
| 25    | test  | 886.8   | 47.0 | 62.1561     | 0.0338±0.0162 | 0.7694±0.0502 | 0.1826±0.0577 | 0.4016±0.0185 |
| 5     | train | 3547.2  | 47.0 | 62.1561     | 0.0316±0.0039 | 0.7901±0.0049 | 0.1979±0.023  | 0.4108±0.0043 |
| 10    | train | 3547.2  | 47.0 | 62.1561     | 0.0345±0.0034 | 0.7922±0.0057 | 0.2141±0.0213 | 0.4133±0.0045 |
| 15    | train | 3547.2  | 47.0 | 62.1561     | 0.0326±0.0019 | 0.789±0.0032  | 0.2022±0.012  | 0.4108±0.0026 |
| 20    | train | 3547.2  | 47.0 | 62.1561     | 0.0316±0.0023 | 0.7874±0.0038 | 0.1961±0.014  | 0.4095±0.003  |
| 25    | train | 3547.2  | 47.0 | 62.1561     | 0.0305±0.0014 | 0.7856±0.0023 | 0.1893±0.0087 | 0.4081±0.0019 |

Figura 21: Estudio parámetros del algoritmo HBOS para el curso CCC

En la tabla 22 se muestran los resultados obtenidos por el algoritmo knn tanto para test como para train con los diferentes números de vecinos considerados que eran 5, 10, 20, 30 y 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar es similar a la detección de caso normales, alcanzando el 50 % de anomalías detectadas y valores cercanos al 50 % de identificación de casos normales. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 40 vecinos consigue mejores resultados frente a las otras alternativas.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 886.8   | 47.0 | 62.1561     | 0.4108±0.0469 | 0.3547±0.06   | 0.5106±0.0139 | 0.3827±0.0152 |
| 10    | test  | 886.8   | 47.0 | 62.1561     | 0.4489±0.0523 | 0.4017±0.0601 | 0.5515±0.0178 | 0.4253±0.0175 |
| 20    | test  | 886.8   | 47.0 | 62.1561     | 0.4917±0.0489 | 0.472±0.0381  | 0.604±0.0159  | 0.4819±0.0157 |
| 30    | test  | 886.8   | 47.0 | 62.1561     | 0.5098±0.0553 | 0.5±0.0391    | 0.6253±0.0161 | 0.5049±0.017  |
| 40    | test  | 886.8   | 47.0 | 62.1561     | 0.5011±0.0595 | 0.4989±0.0429 | 0.6205±0.0232 | 0.5±0.0237    |
| 5     | train | 3547.2  | 47.0 | 62.1561     | 0.3924±0.0105 | 0.3763±0.014  | 0.5082±0.012  | 0.3844±0.0119 |
| 10    | train | 3547.2  | 47.0 | 62.1561     | 0.4323±0.0088 | 0.4185±0.015  | 0.5498±0.0109 | 0.4254±0.0115 |
| 20    | train | 3547.2  | 47.0 | 62.1561     | 0.4812±0.0048 | 0.4891±0.01   | 0.6074±0.0066 | 0.4852±0.007  |
| 30    | train | 3547.2  | 47.0 | 62.1561     | 0.501±0.0019  | 0.5156±0.0067 | 0.6295±0.004  | 0.5083±0.0043 |
| 40    | train | 3547.2  | 47.0 | 62.1561     | 0.4974±0.005  | 0.5107±0.0064 | 0.6254±0.0047 | 0.504±0.0051  |

Figura 22: Estudio parámetros del algoritmo Knn para el curso CCC

## Lof

En la tabla 23 se muestran los resultados obtenidos por el algoritmo lof tanto para test como para train con los diferentes parámetros considerados que eran 5, 10, 20, 30, 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad,



Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar está entorno al 20% manteniendo una alta especificidad. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, aunque la diferencia es mínima entre los valores, en este caso de uso de 40 vecinos consigue mejores resultados frente a otras alternativas.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 886.8   | 47.0 | 62.1561     | 0.1423±0.0232 | 0.9464±0.0107 | 0.8114±0.0345 | 0.5443±0.0115 |
| 10    | test  | 886.8   | 47.0 | 62.1561     | 0.1328±0.0172 | 0.9434±0.0234 | 0.7968±0.0655 | 0.5381±0.0138 |
| 20    | test  | 886.8   | 47.0 | 62.1561     | 0.147±0.0247  | 0.9666±0.0262 | 0.8893±0.0697 | 0.5568±0.0103 |
| 30    | test  | 886.8   | 47.0 | 62.1561     | 0.1481±0.0232 | 0.975±0.0235  | 0.9196±0.0617 | 0.5615±0.0037 |
| 40    | test  | 886.8   | 47.0 | 62.1561     | 0.1578±0.0181 | 0.9785±0.0176 | 0.9291±0.0541 | 0.5682±0.0092 |
| 5     | train | 3547.2  | 47.0 | 62.1561     | 0.0862±0.0034 | 0.9699±0.0036 | 0.8246±0.0217 | 0.5281±0.0031 |
| 10    | train | 3547.2  | 47.0 | 62.1561     | 0.1057±0.0034 | 0.9605±0.0029 | 0.8146±0.0137 | 0.5331±0.0026 |
| 20    | train | 3547.2  | 47.0 | 62.1561     | 0.1337±0.0038 | 0.975±0.0051  | 0.8977±0.0212 | 0.5543±0.0043 |
| 30    | train | 3547.2  | 47.0 | 62.1561     | 0.1397±0.0025 | 0.9781±0.0025 | 0.9128±0.0103 | 0.5589±0.0023 |
| 40    | train | 3547.2  | 47.0 | 62.1561     | 0.1485±0.0021 | 0.9863±0.0028 | 0.9468±0.0109 | 0.5674±0.0024 |

Figura 23: Estudio parámetros del algoritmo LOF para el curso CCC

#### 10.6.1.4. Curso DDD

En esta sección se van a mostrar los resultados para el curso DDD por los diferentes algoritmos con las diferentes configuraciones consideradas.

#### Autoencoder

En la tabla 24 se muestran los resultados obtenidos por el algoritmo autoencoder tanto para test como para train con los diferentes valores de dimensión de la capa latente considerados que eran 16, 32, 64, 128, 256. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. En la tabla se muestra las diferentes opciones con respecto al valor concreto que se ha usado para determinar la dimensión de la capa latente, disminuir este valor puede ser útil para reducir la dimensionalidad y eliminar el ruido, pero también puede resultar en una pérdida de información. Este algoritmo, al utilizar únicamente valores normales durante la fase de entrenamiento, no permite calcular varias de las métricas comúnmente utilizadas en esta etapa. Sin embargo, al analizar los resultados en la fase de prueba, se observa que los diferentes valores de los parámetros producen resultados similares. No obstante, los valores



más reducidos tienden a mostrar un mejor rendimiento. En particular, un valor que utiliza una representación de 16 dimensiones demuestra obtener los mejores resultados. En general, se produce un error mayor en la detección de las anomalías, siendo los valores de sensibilidad más reducidos, indicando que existen anomalías que no es capaz de identificar correctamente. Con respecto a la desviación típica se muestra reducida, con lo que sus resultados son estables en las diferentes ejecuciones.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 16    | test  | 1254.4  | 84.0 | 58.3865     | 0.3252±0.0152 | 0.9708±0.0186 | 0.9397±0.0385 | 0.648±0.0164  |
| 32    | test  | 1254.4  | 84.0 | 58.3865     | 0.3177±0.0151 | 0.9615±0.0186 | 0.9205±0.0383 | 0.6396±0.0162 |
| 64    | test  | 1254.4  | 84.0 | 58.3865     | 0.3051±0.0133 | 0.9461±0.0187 | 0.888±0.0388  | 0.6256±0.016  |
| 128   | test  | 1254.4  | 84.0 | 58.3865     | 0.2972±0.0118 | 0.9353±0.0162 | 0.8656±0.0337 | 0.6163±0.014  |
| 256   | test  | 1254.4  | 84.0 | 58.3865     | 0.3043±0.0124 | 0.9454±0.0171 | 0.8864±0.0356 | 0.6248±0.0148 |
| 16    | train | 2088.0  | 84.0 | 0.0         | 0.0±0.0       | 0.7997±0.0    | 0.0±0.0       | nan±nan       |
| 32    | train | 2088.0  | 84.0 | 0.0         | 0.0±0.0       | 0.7997±0.0    | 0.0±0.0       | nan±nan       |
| 64    | train | 2088.0  | 84.0 | 0.0         | 0.0±0.0       | 0.7997±0.0    | 0.0±0.0       | nan±nan       |
| 128   | train | 2088.0  | 84.0 | 0.0         | 0.0±0.0       | 0.7997±0.0    | 0.0±0.0       | nan±nan       |
| 256   | train | 2088.0  | 84.0 | 0.0         | 0.0±0.0       | 0.7997±0.0    | 0.0±0.0       | nan±nan       |

Figura 24: Estudio parámetros del algoritmo Autoencoder para el curso DDD

## Cblof

En la tabla 25 se muestran los resultados obtenidos por el algoritmo cblof tanto para test como para train con los diferentes números de clústeres considerados, que eran 6, 7, 8, 9 y 10. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar es muy reducido, siendo el valor de sensibilidad solamente entre un 7 % y 12 %. Los diferentes valores de clústeres produce resultados parecidos, obteniendo ligeramente mejores valores cuanto más reducido es el número de clústeres.

## Cof

En la tabla 26 se muestran los resultados obtenidos por el algoritmo cof tanto para test como para train con los diferentes números de vecinos considerados que eran 5, 10, 20, 30 y 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas





| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 6     | test  | 1254.4  | 84.0 | 58.3865     | 0.1168±0.036  | 0.9124±0.0533 | 0.6765±0.0936 | 0.5146±0.0201 |
| 7     | test  | 1254.4  | 84.0 | 58.3865     | 0.1069±0.0403 | 0.8971±0.079  | 0.6309±0.1384 | 0.502±0.0378  |
| 8     | test  | 1254.4  | 84.0 | 58.3865     | 0.0947±0.0416 | 0.8661±0.1135 | 0.5619±0.1795 | 0.4804±0.056  |
| 9     | test  | 1254.4  | 84.0 | 58.3865     | 0.081±0.0516  | 0.8576±0.1247 | 0.4896±0.159  | 0.4693±0.0603 |
| 10    | test  | 1254.4  | 84.0 | 58.3865     | 0.0747±0.0519 | 0.8435±0.1361 | 0.4416±0.1538 | 0.4591±0.0638 |
| 6     | train | 5017.6  | 84.0 | 58.3865     | 0.1131±0.0091 | 0.9182±0.0127 | 0.6598±0.0529 | 0.5156±0.0109 |
| 7     | train | 5017.6  | 84.0 | 58.3865     | 0.1032±0.0193 | 0.9043±0.0271 | 0.602±0.1126  | 0.5037±0.0232 |
| 8     | train | 5017.6  | 84.0 | 58.3865     | 0.091±0.0253  | 0.8872±0.0354 | 0.5308±0.1473 | 0.4891±0.0303 |
| 9     | train | 5017.6  | 84.0 | 58.3865     | 0.0761±0.0265 | 0.8664±0.0372 | 0.4443±0.1547 | 0.4713±0.0319 |
| 10    | train | 5017.6  | 84.0 | 58.3865     | 0.0699±0.0261 | 0.8577±0.0367 | 0.408±0.1525  | 0.4638±0.0314 |

Figura 25: Estudio parámetros del algoritmo CBLOF para el curso DDD

por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar ronda entre 21 % y 26 % manteniendo una alta especificidad. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 30 y 40 vecinos consigue mejores resultados frente a las otras alternativas.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 1254.4  | 84.0 | 58.3865     | 0.2116±0.0383 | 0.9483±0.0109 | 0.8501±0.0321 | 0.58±0.0189   |
| 10    | test  | 1254.4  | 84.0 | 58.3865     | 0.2217±0.0501 | 0.9621±0.0083 | 0.8908±0.01   | 0.5919±0.0213 |
| 20    | test  | 1254.4  | 84.0 | 58.3865     | 0.2381±0.0652 | 0.9674±0.0116 | 0.9085±0.037  | 0.6028±0.0322 |
| 30    | test  | 1254.4  | 84.0 | 58.3865     | 0.2616±0.0548 | 0.9682±0.0157 | 0.9204±0.0373 | 0.6149±0.0271 |
| 40    | test  | 1254.4  | 84.0 | 58.3865     | 0.2603±0.051  | 0.9694±0.0225 | 0.9249±0.0434 | 0.6148±0.0251 |
| 5     | train | 5017.6  | 84.0 | 58.3865     | 0.1491±0.002  | 0.9688±0.0029 | 0.8701±0.0119 | 0.559±0.0024  |
| 10    | train | 5017.6  | 84.0 | 58.3865     | 0.1553±0.0017 | 0.9775±0.0025 | 0.9064±0.0102 | 0.5664±0.0021 |
| 20    | train | 5017.6  | 84.0 | 58.3865     | 0.1555±0.0035 | 0.9778±0.005  | 0.9076±0.0208 | 0.5666±0.0043 |
| 30    | train | 5017.6  | 84.0 | 58.3865     | 0.1553±0.0026 | 0.9774±0.0037 | 0.906±0.0154  | 0.5663±0.0032 |
| 40    | train | 5017.6  | 84.0 | 58.3865     | 0.1555±0.0018 | 0.9778±0.0025 | 0.9076±0.0104 | 0.5667±0.0021 |

Figura 26: Estudio parámetros del algoritmo COF para el curso DDD

## Dae

En la tabla 27 se muestran los resultados obtenidos por el algoritmo dae tanto para test como para train con los diferentes valores de dimensión de la capa latente considerados que eran 16, 32, 64, 128, 256. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test en este caso se presentan diferentes, debido a que este algoritmo se entrena solamente con valores normales, por tanto, durante la fase de train nos encontramos con métricas que no pueden calcularse. Con respecto a la desviación típica se muestra reducida, con lo





que sus resultados son estables. En este caso, el número de anomalías que es capaz de detectar se mantiene sobre el 34 % manteniendo una alta especificidad. Se aprecia que resultados similares son obtenidos durante todos los valores, siendo ligeramente superior con 256, no afectando mucho el valor de este parámetro en los resultados.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 16    | test  | 1254.4  | 84.0 | 58.3865     | 0.3448±0.0055 | 0.9995±0.0009 | 0.9989±0.0018 | 0.6721±0.0027 |
| 32    | test  | 1254.4  | 84.0 | 58.3865     | 0.3482±0.0114 | 0.9997±0.0007 | 0.9993±0.0015 | 0.6739±0.0057 |
| 64    | test  | 1254.4  | 84.0 | 58.3865     | 0.3466±0.0098 | 0.9996±0.0008 | 0.9991±0.0017 | 0.6731±0.0049 |
| 128   | test  | 1254.4  | 84.0 | 58.3865     | 0.3461±0.0075 | 0.9997±0.0007 | 0.9995±0.0014 | 0.6729±0.0037 |
| 256   | test  | 1254.4  | 84.0 | 58.3865     | 0.3469±0.0086 | 0.9997±0.0007 | 0.9994±0.0014 | 0.6733±0.0044 |
| 16    | train | 2088.0  | 84.0 | 0.0         | 0.0±0.0       | 0.7997±0.0    | 0.0±0.0       | nan±nan       |
| 32    | train | 2088.0  | 84.0 | 0.0         | 0.0±0.0       | 0.7997±0.0    | 0.0±0.0       | nan±nan       |
| 64    | train | 2088.0  | 84.0 | 0.0         | 0.0±0.0       | 0.7997±0.0    | 0.0±0.0       | nan±nan       |
| 128   | train | 2088.0  | 84.0 | 0.0         | 0.0±0.0       | 0.7997±0.0    | 0.0±0.0       | nan±nan       |
| 256   | train | 2088.0  | 84.0 | 0.0         | 0.0±0.0       | 0.7997±0.0    | 0.0±0.0       | nan±nan       |

Figura 27: Estudio parámetros del algoritmo DAE para el curso DDD

## Hbos

En la tabla 28 se muestran los resultados obtenidos por el algoritmo hbos tanto para test como para train con los diferentes tamaños de los intervalos considerados que eran 5, 10, 15, 20, 25. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar no alcanza ni el 10 % siendo también la especificidad más reducida que en las anteriores, no alcanzando el 90 %. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de intervalos considerados, aunque se trata de resultados con un rendimiento muy bajo para detectar las anomalías.

## Knn

En la tabla 29 se muestran los resultados obtenidos por el algoritmo knn tanto para test como para train con los diferentes números de vecinos considerados que eran 5, 10, 20, 30 y 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas



| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 1254.4  | 84.0 | 58.3865     | 0.0516±0.0357 | 0.805±0.179   | 0.4024±0.2248 | 0.4283±0.0742 |
| 10    | test  | 1254.4  | 84.0 | 58.3865     | 0.0546±0.0335 | 0.8058±0.1773 | 0.4158±0.2187 | 0.4302±0.0752 |
| 15    | test  | 1254.4  | 84.0 | 58.3865     | 0.0535±0.0323 | 0.8027±0.1796 | 0.4106±0.2221 | 0.4281±0.0773 |
| 20    | test  | 1254.4  | 84.0 | 58.3865     | 0.0535±0.032  | 0.8065±0.1777 | 0.4161±0.2227 | 0.43±0.0766   |
| 25    | test  | 1254.4  | 84.0 | 58.3865     | 0.054±0.0318  | 0.8073±0.1778 | 0.4203±0.2206 | 0.4307±0.0767 |
| 5     | train | 5017.6  | 84.0 | 58.3865     | 0.0451±0.0121 | 0.8229±0.0169 | 0.2631±0.0704 | 0.434±0.0145  |
| 10    | train | 5017.6  | 84.0 | 58.3865     | 0.046±0.0103  | 0.8241±0.0144 | 0.2685±0.0599 | 0.4351±0.0123 |
| 15    | train | 5017.6  | 84.0 | 58.3865     | 0.0467±0.0093 | 0.8251±0.013  | 0.2725±0.0542 | 0.4359±0.0112 |
| 20    | train | 5017.6  | 84.0 | 58.3865     | 0.0466±0.0097 | 0.825±0.0136  | 0.2721±0.0568 | 0.4358±0.0117 |
| 25    | train | 5017.6  | 84.0 | 58.3865     | 0.0469±0.0097 | 0.8254±0.0136 | 0.2737±0.0567 | 0.4362±0.0117 |

Figura 28: Estudio parámetros del algoritmo HBOS para el curso DDD

son reducidas con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar es similar a la detección de caso normales, alcanzando valores entre el 50 % y el 58 % de anomalías detectadas y valores medios sobre al 57 % de identificación de casos normales. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 30 vecinos consigue mejores resultados frente a las otras alternativas.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 1254.4  | 84.0 | 58.3865     | 0.5043±0.0949 | 0.4782±0.1563 | 0.582±0.0341  | 0.4912±0.0361 |
| 10    | test  | 1254.4  | 84.0 | 58.3865     | 0.5382±0.0962 | 0.5096±0.1473 | 0.6131±0.0355 | 0.5239±0.0307 |
| 20    | test  | 1254.4  | 84.0 | 58.3865     | 0.5704±0.0915 | 0.5724±0.1409 | 0.6606±0.0425 | 0.5714±0.0268 |
| 30    | test  | 1254.4  | 84.0 | 58.3865     | 0.58±0.095    | 0.5751±0.1211 | 0.6628±0.031  | 0.5775±0.0173 |
| 40    | test  | 1254.4  | 84.0 | 58.3865     | 0.566±0.1023  | 0.5567±0.1024 | 0.6444±0.0156 | 0.5614±0.0075 |
| 5     | train | 5017.6  | 84.0 | 58.3865     | 0.4737±0.0098 | 0.5205±0.0116 | 0.5809±0.0108 | 0.4971±0.0106 |
| 10    | train | 5017.6  | 84.0 | 58.3865     | 0.511±0.0075  | 0.5445±0.0132 | 0.6115±0.01   | 0.5277±0.01   |
| 20    | train | 5017.6  | 84.0 | 58.3865     | 0.5477±0.0103 | 0.5893±0.0156 | 0.6517±0.0128 | 0.5685±0.0128 |
| 30    | train | 5017.6  | 84.0 | 58.3865     | 0.5587±0.0088 | 0.5906±0.0136 | 0.6569±0.011  | 0.5746±0.0112 |
| 40    | train | 5017.6  | 84.0 | 58.3865     | 0.5549±0.0148 | 0.5852±0.0204 | 0.6524±0.0172 | 0.5701±0.0176 |

Figura 29: Estudio parámetros del algoritmo Knn para el curso DDD

## Lof

En la tabla 30 se muestran los resultados obtenidos por el algoritmo lof tanto para test como para train con los diferentes parámetros considerados que eran 5, 10, 20, 30, 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar está entorno al 20 % manteniendo una alta especificidad. Se aprecia que mejores resultados son alcanzados conforme aumenta



el número de vecinos considerados, en este caso el uso de 30 vecinos consigue mejores resultados frente a las otras alternativas.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 1254.4  | 84.0 | 58.3865     | 0.1622±0.0303 | 0.9628±0.0202 | 0.8658±0.0512 | 0.5625±0.0122 |
| 10    | test  | 1254.4  | 84.0 | 58.3865     | 0.1589±0.0318 | 0.9674±0.0152 | 0.8751±0.041  | 0.5632±0.0144 |
| 20    | test  | 1254.4  | 84.0 | 58.3865     | 0.1808±0.0339 | 0.969±0.0168  | 0.894±0.0412  | 0.5748±0.0154 |
| 30    | test  | 1254.4  | 84.0 | 58.3865     | 0.1821±0.0373 | 0.9805±0.0063 | 0.9272±0.024  | 0.5813±0.0188 |
| 40    | test  | 1254.4  | 84.0 | 58.3865     | 0.1731±0.0141 | 0.9793±0.0141 | 0.9238±0.0505 | 0.5762±0.0099 |
| 5     | train | 5017.6  | 84.0 | 58.3865     | 0.1055±0.0049 | 0.9817±0.0037 | 0.8897±0.024  | 0.5436±0.0041 |
| 10    | train | 5017.6  | 84.0 | 58.3865     | 0.131±0.0059  | 0.9798±0.0045 | 0.9006±0.0236 | 0.5554±0.005  |
| 20    | train | 5017.6  | 84.0 | 58.3865     | 0.1457±0.002  | 0.9792±0.005  | 0.908±0.0209  | 0.5625±0.0032 |
| 30    | train | 5017.6  | 84.0 | 58.3865     | 0.1527±0.0017 | 0.9869±0.0036 | 0.9425±0.0151 | 0.5698±0.0019 |
| 40    | train | 5017.6  | 84.0 | 58.3865     | 0.158±0.0064  | 0.9886±0.0081 | 0.951±0.0353  | 0.5733±0.0072 |

Figura 30: Estudio parámetros del algoritmo LOF para el curso DDD

#### 10.6.1.5. Curso EEE

En esta sección se van a mostrar los resultados para el curso EEE por los diferentes algoritmos con las diferentes configuraciones consideradas.

#### Autoencoder

En la tabla 31 se muestran los resultados obtenidos por el algoritmo autoencoder tanto para test como para train con los diferentes valores de dimensión de la capa latente considerados que eran 16, 32, 64, 128, 256. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. En la tabla se muestra las diferentes opciones con respecto al valor concreto que se ha usado para determinar la dimensión de la capa latente, disminuir este valor puede ser útil para reducir la dimensionalidad y eliminar el ruido, pero también puede resultar en una pérdida de información. Este algoritmo, al utilizar únicamente valores normales durante la fase de entrenamiento, no permite calcular varias de las métricas comúnmente utilizadas en esta etapa. Sin embargo, al analizar los resultados en la fase de prueba, se observa que los diferentes valores de los parámetros producen resultados similares. No obstante, los valores más reducidos tienden a mostrar un mejor rendimiento. En particular, un valor que utiliza una representación de 16 dimensiones demuestra obtener los mejores resultados. En general, se produce un error mayor en la detección de las anomalías, siendo los valores de sensibilidad más reducidos, indicando que existen anomalías que no es capaz de identificar correctamente. Con respecto a la desviación típica se muestra reducida, con lo que sus resultados son estables en las diferentes ejecuciones.



| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 16    | test  | 586.8   | 37.0 | 43.7627     | 0.4262±0.0511 | 0.9672±0.0341 | 0.9093±0.0957 | 0.6967±0.0417 |
| 32    | test  | 586.8   | 37.0 | 43.7627     | 0.3981±0.0244 | 0.9508±0.0174 | 0.8627±0.0488 | 0.6744±0.0207 |
| 64    | test  | 586.8   | 37.0 | 43.7627     | 0.3811±0.0158 | 0.9389±0.0122 | 0.8293±0.0339 | 0.66±0.014    |
| 128   | test  | 586.8   | 37.0 | 43.7627     | 0.3737±0.0167 | 0.9333±0.0131 | 0.8135±0.0364 | 0.6535±0.0149 |
| 256   | test  | 586.8   | 37.0 | 43.7627     | 0.3725±0.017  | 0.9323±0.0134 | 0.8108±0.0372 | 0.6524±0.0152 |
| 16    | train | 1320.0  | 37.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 32    | train | 1320.0  | 37.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 64    | train | 1320.0  | 37.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 128   | train | 1320.0  | 37.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 256   | train | 1320.0  | 37.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |

Figura 31: Estudio parámetros del algoritmo Autoencoder para el curso EEE

### Cblof

En la tabla 32 se muestran los resultados obtenidos por el algoritmo cblof tanto para test como para train con los diferentes números de clústeres considerados, que eran 6, 7, 8, 9 y 10. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar es muy reducido, siendo el valor de sensibilidad menor al 10 %. Los diferentes valores de clústeres produce resultados parecidos, obteniendo ligeramente mejores valores cuanto más reducido es el número de clústeres.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 6     | test  | 586.8   | 37.0 | 43.7627     | 0.076±0.0232  | 0.8751±0.0421 | 0.328±0.0787  | 0.4755±0.0206 |
| 7     | test  | 586.8   | 37.0 | 43.7627     | 0.0732±0.0202 | 0.8712±0.0412 | 0.314±0.0681  | 0.4722±0.0193 |
| 8     | test  | 586.8   | 37.0 | 43.7627     | 0.0623±0.0186 | 0.8579±0.0448 | 0.2623±0.0616 | 0.4601±0.0201 |
| 9     | test  | 586.8   | 37.0 | 43.7627     | 0.0595±0.0187 | 0.8479±0.0504 | 0.2437±0.0682 | 0.4537±0.0239 |
| 10    | test  | 586.8   | 37.0 | 43.7627     | 0.0628±0.0202 | 0.8421±0.0524 | 0.2466±0.0682 | 0.4525±0.0246 |
| 6     | train | 2347.2  | 37.0 | 43.7628     | 0.0739±0.0062 | 0.8796±0.005  | 0.3234±0.0275 | 0.4768±0.0056 |
| 7     | train | 2347.2  | 37.0 | 43.7628     | 0.0704±0.0069 | 0.8768±0.0054 | 0.3079±0.0303 | 0.4736±0.0062 |
| 8     | train | 2347.2  | 37.0 | 43.7628     | 0.0607±0.0108 | 0.8693±0.0084 | 0.2655±0.0473 | 0.465±0.0096  |
| 9     | train | 2347.2  | 37.0 | 43.7628     | 0.0578±0.0117 | 0.867±0.0091  | 0.2528±0.0511 | 0.4624±0.0104 |
| 10    | train | 2347.2  | 37.0 | 43.7628     | 0.0581±0.0133 | 0.8672±0.0104 | 0.2538±0.0582 | 0.4626±0.0118 |

Figura 32: Estudio parámetros del algoritmo CBLOF para el curso EEE

### Cof

En la tabla 33 se muestran los resultados obtenidos por el algoritmo cof tanto para test como para train con los diferentes números de vecinos considerados que eran



5, 10, 20, 30 y 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar supera el 30 % manteniendo una alta especificidad. Se aprecia que mejores resultados son alcanzados conforme disminuye el número de vecinos considerados, en este caso el uso de 10 vecinos consigue mejores resultados frente a las otras alternativas.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 586.8   | 37.0 | 43.7627     | 0.3084±0.0383 | 0.9236±0.0308 | 0.7673±0.056  | 0.616±0.0126  |
| 10    | test  | 586.8   | 37.0 | 43.7627     | 0.3505±0.0508 | 0.9436±0.0204 | 0.8336±0.0398 | 0.6471±0.0184 |
| 20    | test  | 586.8   | 37.0 | 43.7627     | 0.2936±0.0493 | 0.9624±0.0079 | 0.8566±0.0302 | 0.628±0.0253  |
| 30    | test  | 586.8   | 37.0 | 43.7627     | 0.2149±0.0514 | 0.9637±0.0175 | 0.8206±0.0658 | 0.5893±0.0276 |
| 40    | test  | 586.8   | 37.0 | 43.7627     | 0.1752±0.0383 | 0.983±0.0138  | 0.8824±0.0951 | 0.5791±0.0249 |
| 5     | train | 2347.2  | 37.0 | 43.7628     | 0.1714±0.0077 | 0.9553±0.006  | 0.7489±0.0334 | 0.5633±0.0068 |
| 10    | train | 2347.2  | 37.0 | 43.7628     | 0.1842±0.0027 | 0.9653±0.0021 | 0.8051±0.0116 | 0.5748±0.0024 |
| 20    | train | 2347.2  | 37.0 | 43.7628     | 0.2019±0.0024 | 0.9791±0.0018 | 0.8826±0.0104 | 0.5905±0.0021 |
| 30    | train | 2347.2  | 37.0 | 43.7628     | 0.2079±0.0022 | 0.9838±0.0017 | 0.9089±0.0097 | 0.5959±0.0019 |
| 40    | train | 2347.2  | 37.0 | 43.7628     | 0.2081±0.0042 | 0.984±0.0033  | 0.9098±0.0187 | 0.5961±0.0038 |

Figura 33: Estudio parámetros del algoritmo COF para el curso EEE

## Dae

En la tabla 34 se muestran los resultados obtenidos por el algoritmo dae tanto para test como para train con los diferentes valores de dimensión de la capa latente considerados que eran 16, 32, 64, 128, 256. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test en este caso se presentan diferentes, debido a que este algoritmo se entrena solamente con valores normales, por tanto, durante la fase de train nos encontramos con métricas que no pueden calcularse. Con respecto a la desviación típica se muestra reducida, con lo que sus resultados son estables. En este caso, el número de anomalías que es capaz de detectar se mantiene en un 45 % manteniendo una alta especificidad. Se aprecia que resultados similares son obtenidos a partir del uso de un tamaño de dimensión igual a 32, no afectando mucho el valor de este parámetro en los resultados.

## Hbos



| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 16    | test  | 586.8   | 37.0 | 43.7627     | 0.4588±0.0026 | 0.999±0.0014  | 0.9971±0.0041 | 0.7289±0.002  |
| 32    | test  | 586.8   | 37.0 | 43.7627     | 0.459±0.0038  | 0.9989±0.0015 | 0.9968±0.0042 | 0.7289±0.0025 |
| 64    | test  | 586.8   | 37.0 | 43.7627     | 0.4587±0.0036 | 0.9989±0.0015 | 0.9968±0.0042 | 0.7288±0.0025 |
| 128   | test  | 586.8   | 37.0 | 43.7627     | 0.459±0.0033  | 0.9989±0.0015 | 0.9969±0.0041 | 0.729±0.0022  |
| 256   | test  | 586.8   | 37.0 | 43.7627     | 0.459±0.0033  | 0.9989±0.0015 | 0.9969±0.0041 | 0.729±0.0023  |
| 16    | train | 1320.0  | 37.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 32    | train | 1320.0  | 37.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 64    | train | 1320.0  | 37.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 128   | train | 1320.0  | 37.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 256   | train | 1320.0  | 37.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |

Figura 34: Estudio parámetros del algoritmo DAE para el curso EEE

En la tabla 35 se muestran los resultados obtenidos por el algoritmo hbos tanto para test como para train con los diferentes tamaños de los intervalos considerados que eran 5, 10, 15, 20, 25. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar no alcanza ni el 3% siendo también la especificidad más reducida que en las anteriores, rozando el 80%. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de intervalos considerados, aunque se trata de resultados con un rendimiento muy bajo para detectar las anomalías.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 586.8   | 37.0 | 43.7627     | 0.0296±0.015  | 0.8157±0.0289 | 0.1091±0.0528 | 0.4227±0.0151 |
| 10    | test  | 586.8   | 37.0 | 43.7627     | 0.0265±0.0144 | 0.8097±0.0387 | 0.1007±0.0559 | 0.4181±0.0229 |
| 15    | test  | 586.8   | 37.0 | 43.7627     | 0.0272±0.0165 | 0.8054±0.0364 | 0.1026±0.0631 | 0.4163±0.024  |
| 20    | test  | 586.8   | 37.0 | 43.7627     | 0.028±0.0229  | 0.8079±0.0344 | 0.1021±0.0761 | 0.4179±0.0237 |
| 25    | test  | 586.8   | 37.0 | 43.7627     | 0.0296±0.0216 | 0.8006±0.0333 | 0.1041±0.0735 | 0.4151±0.0233 |
| 5     | train | 2347.2  | 37.0 | 43.7628     | 0.0263±0.0041 | 0.8424±0.0032 | 0.1149±0.0178 | 0.4344±0.0036 |
| 10    | train | 2347.2  | 37.0 | 43.7628     | 0.0257±0.0042 | 0.842±0.0033  | 0.1124±0.0186 | 0.4338±0.0038 |
| 15    | train | 2347.2  | 37.0 | 43.7628     | 0.0226±0.004  | 0.8395±0.0032 | 0.0987±0.0177 | 0.4311±0.0036 |
| 20    | train | 2347.2  | 37.0 | 43.7628     | 0.0238±0.0036 | 0.8405±0.0028 | 0.1038±0.0157 | 0.4321±0.0032 |
| 25    | train | 2347.2  | 37.0 | 43.7628     | 0.0238±0.0027 | 0.8405±0.0021 | 0.1038±0.0117 | 0.4321±0.0024 |

Figura 35: Estudio parámetros del algoritmo HBOS para el curso EEE

## Knn

En la tabla 36 se muestran los resultados obtenidos por el algoritmo knn tanto para test como para train con los diferentes números de vecinos considerados que eran 5, 10, 20, 30 y 40. Los resultados que se muestran son los valores de Sensibilidad,



## CAPÍTULO 10. RESULTADOS EXPERIMENTALES

Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar es similar a la detección de caso normales, alcanzando el 58 % de anomalías detectadas y valores cercanos al 52 % de identificación de casos normales. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 40 vecinos consigue mejores resultados frente a las otras alternativas.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 586.8   | 37.0 | 43.7627     | 0.4198±0.0287 | 0.4031±0.0489 | 0.3541±0.0177 | 0.4114±0.0219 |
| 10    | test  | 586.8   | 37.0 | 43.7627     | 0.4876±0.0237 | 0.4564±0.0552 | 0.4121±0.0186 | 0.4719±0.0214 |
| 20    | test  | 586.8   | 37.0 | 43.7627     | 0.5358±0.0135 | 0.503±0.0503  | 0.4576±0.0277 | 0.5194±0.0267 |
| 30    | test  | 586.8   | 37.0 | 43.7627     | 0.574±0.0193  | 0.5291±0.0535 | 0.4883±0.0265 | 0.5515±0.0243 |
| 40    | test  | 586.8   | 37.0 | 43.7627     | 0.5794±0.0185 | 0.5382±0.06   | 0.496±0.03    | 0.5588±0.0263 |
| 5     | train | 2347.2  | 37.0 | 43.7628     | 0.3705±0.0112 | 0.4543±0.0063 | 0.3457±0.0092 | 0.4124±0.0084 |
| 10    | train | 2347.2  | 37.0 | 43.7628     | 0.448±0.0047  | 0.4909±0.0033 | 0.4065±0.0028 | 0.4695±0.0027 |
| 20    | train | 2347.2  | 37.0 | 43.7628     | 0.5177±0.0108 | 0.5282±0.0087 | 0.4606±0.0096 | 0.5229±0.0095 |
| 30    | train | 2347.2  | 37.0 | 43.7628     | 0.5576±0.0076 | 0.5582±0.0062 | 0.4955±0.0064 | 0.5579±0.0065 |
| 40    | train | 2347.2  | 37.0 | 43.7628     | 0.579±0.0026  | 0.5667±0.0029 | 0.5098±0.0028 | 0.5729±0.0027 |

Figura 36: Estudio parámetros del algoritmo Knn para el curso EEE

## Lof

En la tabla 37 se muestran los resultados obtenidos por el algoritmo lof tanto para test como para train con los diferentes parámetros considerados que eran 5, 10, 20, 30, 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar está entorno al 20 % manteniendo una alta especificidad. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 30 vecinos consigue mejores resultados frente a las otras alternativas.

### 10.6.1.6. Curso FFF

En esta sección se van a mostrar los resultados para el curso FFF por los diferentes algoritmos con las diferentes configuraciones consideradas.





| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 586.8   | 37.0 | 43.7627     | 0.1838±0.0412 | 0.9564±0.025  | 0.7836±0.0728 | 0.5701±0.0143 |
| 10    | test  | 586.8   | 37.0 | 43.7627     | 0.1877±0.0283 | 0.9491±0.0397 | 0.7657±0.1281 | 0.5684±0.0241 |
| 20    | test  | 586.8   | 37.0 | 43.7627     | 0.2056±0.0271 | 0.9703±0.0299 | 0.8672±0.1163 | 0.588±0.0153  |
| 30    | test  | 586.8   | 37.0 | 43.7627     | 0.2118±0.0291 | 0.9745±0.0263 | 0.8852±0.0983 | 0.5932±0.0144 |
| 40    | test  | 586.8   | 37.0 | 43.7627     | 0.1986±0.0371 | 0.9685±0.0293 | 0.859±0.0963  | 0.5835±0.008  |
| 5     | train | 2347.2  | 37.0 | 43.7628     | 0.1264±0.0094 | 0.9771±0.0043 | 0.8111±0.0339 | 0.5517±0.0058 |
| 10    | train | 2347.2  | 37.0 | 43.7628     | 0.1437±0.0055 | 0.9711±0.0024 | 0.7944±0.0158 | 0.5574±0.0031 |
| 20    | train | 2347.2  | 37.0 | 43.7628     | 0.1857±0.0053 | 0.9791±0.003  | 0.8738±0.017  | 0.5824±0.0033 |
| 30    | train | 2347.2  | 37.0 | 43.7628     | 0.1933±0.005  | 0.9806±0.0033 | 0.8858±0.0194 | 0.587±0.004   |
| 40    | train | 2347.2  | 37.0 | 43.7628     | 0.1805±0.012  | 0.9742±0.0042 | 0.8442±0.03   | 0.5774±0.008  |

Figura 37: Estudio parámetros del algoritmo LOF para el curso EEE

## Autoencoder

En la tabla 38 se muestran los resultados obtenidos por el algoritmo autoencoder tanto para test como para train con los diferentes valores de dimensión de la capa latente considerados que eran 16, 32, 64, 128, 256. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. En la tabla se muestra las diferentes opciones con respecto al valor concreto que se ha usado para determinar la dimensión de la capa latente, disminuir este valor puede ser útil para reducir la dimensionalidad y eliminar el ruido, pero también puede resultar en una pérdida de información. Este algoritmo, al utilizar únicamente valores normales durante la fase de entrenamiento, no permite calcular varias de las métricas comúnmente utilizadas en esta etapa. Sin embargo, al analizar los resultados en la fase de prueba, se observa que los diferentes valores de los parámetros producen resultados similares. No obstante, los valores más reducidos tienden a mostrar un mejor rendimiento. En particular, un valor que utiliza una representación de 16 dimensiones demuestra obtener los mejores resultados. En general, se produce un error mayor en la detección de las anomalías, siendo los valores de sensibilidad más reducidos, indicando que existen anomalías que no es capaz de identificar correctamente. Con respecto a la desviación típica se muestra reducida, con lo que sus resultados son estables en las diferentes ejecuciones.

## Cblof

En la tabla 39 se muestran los resultados obtenidos por el algoritmo cblof tanto para test como para train con los diferentes números de clústeres considerados, que eran 6, 7, 8, 9 y 10. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas





| value | type  | samples | dims  | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|-------|-------------|---------------|---------------|---------------|---------------|
| 16    | test  | 1552.4  | 116.0 | 53.0018     | 0.3714±0.0106 | 0.9892±0.0083 | 0.9749±0.0193 | 0.6803±0.0085 |
| 32    | test  | 1552.4  | 116.0 | 53.0018     | 0.3581±0.0131 | 0.9755±0.0111 | 0.9427±0.026  | 0.6668±0.0116 |
| 64    | test  | 1552.4  | 116.0 | 53.0018     | 0.3393±0.013  | 0.9557±0.0146 | 0.8962±0.0342 | 0.6475±0.0138 |
| 128   | test  | 1552.4  | 116.0 | 53.0018     | 0.3312±0.0128 | 0.9468±0.0143 | 0.8752±0.0336 | 0.639±0.0135  |
| 256   | test  | 1552.4  | 116.0 | 53.0018     | 0.3348±0.0078 | 0.9508±0.0091 | 0.8847±0.0211 | 0.6428±0.0084 |
| 16    | train | 2918.4  | 116.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 32    | train | 2918.4  | 116.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 64    | train | 2918.4  | 116.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 128   | train | 2918.4  | 116.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 256   | train | 2918.4  | 116.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |

Figura 38: Estudio parámetros del algoritmo Autoencoder para el curso FFF

por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar es muy reducido, siendo el valor de sensibilidad solamente entre un 7 % y 13 %. Los diferentes valores de clústeres produce resultados parecidos, obteniendo ligeramente mejores valores cuanto más reducido es el número de clústeres.

| value | type  | samples | dims  | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|-------|-------------|---------------|---------------|---------------|---------------|
| 6     | test  | 1552.4  | 116.0 | 53.0018     | 0.1232±0.0423 | 0.9143±0.0433 | 0.6339±0.0686 | 0.5188±0.0135 |
| 7     | test  | 1552.4  | 116.0 | 53.0018     | 0.1024±0.0411 | 0.8617±0.0812 | 0.4818±0.1731 | 0.4821±0.0466 |
| 8     | test  | 1552.4  | 116.0 | 53.0018     | 0.0982±0.0513 | 0.8485±0.0905 | 0.4469±0.2042 | 0.4733±0.059  |
| 9     | test  | 1552.4  | 116.0 | 53.0018     | 0.0798±0.0534 | 0.8012±0.11   | 0.3477±0.2311 | 0.4405±0.0716 |
| 10    | test  | 1552.4  | 116.0 | 53.0018     | 0.0778±0.0513 | 0.805±0.1125  | 0.3547±0.246  | 0.4414±0.0736 |
| 6     | train | 6209.6  | 116.0 | 53.0018     | 0.1202±0.0076 | 0.9228±0.0085 | 0.6372±0.0401 | 0.5215±0.008  |
| 7     | train | 6209.6  | 116.0 | 53.0018     | 0.1001±0.0261 | 0.9002±0.0294 | 0.5307±0.1384 | 0.5001±0.0278 |
| 8     | train | 6209.6  | 116.0 | 53.0018     | 0.0966±0.0319 | 0.8961±0.0359 | 0.5119±0.1689 | 0.4964±0.0339 |
| 9     | train | 6209.6  | 116.0 | 53.0018     | 0.0775±0.038  | 0.8747±0.0428 | 0.411±0.2013  | 0.4761±0.0404 |
| 10    | train | 6209.6  | 116.0 | 53.0018     | 0.076±0.0402  | 0.8729±0.0453 | 0.4029±0.2131 | 0.4745±0.0428 |

Figura 39: Estudio parámetros del algoritmo CBLOF para el curso FFF

## Cof

En la tabla 40 se muestran los resultados obtenidos por el algoritmo cof tanto para test como para train con los diferentes números de vecinos considerados que eran 5, 10, 20, 30 y 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar llega hasta el 30 % manteniendo una alta especificidad. Se aprecia que mejores resultados son alcanzados conforme aumenta



## CAPÍTULO 10. RESULTADOS EXPERIMENTALES

el número de vecinos considerados, en este caso el uso de 40 vecinos consigue mejores resultados frente a las otras alternativas.

| value | type  | samples | dims  | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|-------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 1552.4  | 116.0 | 53.0018     | 0.2265±0.045  | 0.9613±0.0162 | 0.8705±0.0375 | 0.594±0.0202  |
| 10    | test  | 1552.4  | 116.0 | 53.0018     | 0.2533±0.0455 | 0.969±0.0141  | 0.9043±0.0315 | 0.6111±0.0204 |
| 20    | test  | 1552.4  | 116.0 | 53.0018     | 0.2693±0.0479 | 0.972±0.0162  | 0.921±0.0411  | 0.6207±0.0199 |
| 30    | test  | 1552.4  | 116.0 | 53.0018     | 0.2895±0.0316 | 0.9688±0.0152 | 0.9154±0.0366 | 0.6291±0.0135 |
| 40    | test  | 1552.4  | 116.0 | 53.0018     | 0.3014±0.0288 | 0.9605±0.0266 | 0.8994±0.056  | 0.631±0.0187  |
| 5     | train | 6209.6  | 116.0 | 53.0018     | 0.1665±0.0024 | 0.975±0.0027  | 0.8828±0.0127 | 0.5708±0.0025 |
| 10    | train | 6209.6  | 116.0 | 53.0018     | 0.1743±0.0018 | 0.9839±0.002  | 0.9243±0.0094 | 0.5791±0.0019 |
| 20    | train | 6209.6  | 116.0 | 53.0018     | 0.1748±0.0018 | 0.9844±0.002  | 0.9266±0.0096 | 0.5796±0.0019 |
| 30    | train | 6209.6  | 116.0 | 53.0018     | 0.1756±0.0026 | 0.9852±0.0029 | 0.9304±0.0135 | 0.5804±0.0027 |
| 40    | train | 6209.6  | 116.0 | 53.0018     | 0.1738±0.0023 | 0.9833±0.0026 | 0.9214±0.0123 | 0.5786±0.0025 |

Figura 40: Estudio parámetros del algoritmo COF para el curso FFF

## Dae

En la tabla 41 se muestran los resultados obtenidos por el algoritmo dae tanto para test como para train con los diferentes valores de dimensión de la capa latente considerados que eran 16, 32, 64, 128, 256. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test en este caso se presentan diferentes, debido a que este algoritmo se entrena solamente con valores normales, por tanto, durante la fase de train nos encontramos con métricas que no pueden calcularse. Con respecto a la desviación típica se muestra reducida, con lo que sus resultados son estables. En este caso, el número de anomalías que es capaz de detectar se mantiene estable en un 37 % manteniendo una alta especificidad. Se puede suponer que es debido al número de anomalías que se presenta en este curso. Se aprecia que resultados similares son obtenidos a partir del uso de un tamaño de dimensión igual a 32, no afectando mucho el valor de este parámetro en los resultados.

## Hbos

En la tabla 42 se muestran los resultados obtenidos por el algoritmo hbos tanto para test como para train con los diferentes tamaños de los intervalos considerados que eran 5, 10, 15, 20, 25. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las



| value | type  | samples | dims  | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|-------|-------------|---------------|---------------|---------------|---------------|
| 16    | test  | 1552.4  | 116.0 | 53.0018     | 0.378±0.001   | 0.9995±0.0007 | 0.9988±0.0016 | 0.6888±0.0007 |
| 32    | test  | 1552.4  | 116.0 | 53.0018     | 0.378±0.0007  | 0.9995±0.0007 | 0.9989±0.0015 | 0.6888±0.0006 |
| 64    | test  | 1552.4  | 116.0 | 53.0018     | 0.378±0.0008  | 0.9996±0.0007 | 0.999±0.0015  | 0.6888±0.0006 |
| 128   | test  | 1552.4  | 116.0 | 53.0018     | 0.3783±0.0008 | 0.9997±0.0006 | 0.9993±0.0013 | 0.689±0.0006  |
| 256   | test  | 1552.4  | 116.0 | 53.0018     | 0.378±0.0006  | 0.9996±0.0006 | 0.999±0.0015  | 0.6888±0.0006 |
| 16    | train | 2918.4  | 116.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 32    | train | 2918.4  | 116.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 64    | train | 2918.4  | 116.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 128   | train | 2918.4  | 116.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |
| 256   | train | 2918.4  | 116.0 | 0.0         | 0.0±0.0       | 0.7998±0.0    | 0.0±0.0       | nan±nan       |

Figura 41: Estudio parámetros del algoritmo DAE para el curso FFF

tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar no alcanza ni el 7 % siendo también la especificidad más reducida que en las anteriores, alcanzando un 70 %. No se aprecia una tendencia de mejora de resultados en base al alza o baja del valor, el mejor valor es 5.

| value | type  | samples | dims  | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|-------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 1552.4  | 116.0 | 53.0018     | 0.0693±0.0302 | 0.665±0.1365  | 0.1944±0.0551 | 0.3671±0.0575 |
| 10    | test  | 1552.4  | 116.0 | 53.0018     | 0.0513±0.0295 | 0.7034±0.113  | 0.1593±0.0525 | 0.3774±0.0455 |
| 15    | test  | 1552.4  | 116.0 | 53.0018     | 0.0476±0.0273 | 0.6969±0.1088 | 0.146±0.0465  | 0.3723±0.0435 |
| 20    | test  | 1552.4  | 116.0 | 53.0018     | 0.0452±0.0278 | 0.7026±0.1106 | 0.1411±0.0488 | 0.3739±0.0444 |
| 25    | test  | 1552.4  | 116.0 | 53.0018     | 0.0447±0.0279 | 0.6952±0.1016 | 0.1359±0.0497 | 0.3699±0.0405 |
| 5     | train | 6209.6  | 116.0 | 53.0018     | 0.0451±0.0044 | 0.8381±0.005  | 0.2393±0.0235 | 0.4416±0.0047 |
| 10    | train | 6209.6  | 116.0 | 53.0018     | 0.037±0.0043  | 0.8289±0.0049 | 0.1958±0.0227 | 0.4329±0.0046 |
| 15    | train | 6209.6  | 116.0 | 53.0018     | 0.0353±0.0052 | 0.827±0.0058  | 0.1871±0.0273 | 0.4312±0.0055 |
| 20    | train | 6209.6  | 116.0 | 53.0018     | 0.0349±0.0056 | 0.8266±0.0063 | 0.1849±0.0295 | 0.4307±0.0059 |
| 25    | train | 6209.6  | 116.0 | 53.0018     | 0.0343±0.0049 | 0.8259±0.0055 | 0.182±0.026   | 0.4301±0.0052 |

Figura 42: Estudio parámetros del algoritmo HBOS para el curso FFF

## Knn

En la tabla 43 se muestran los resultados obtenidos por el algoritmo knn tanto para test como para train con los diferentes números de vecinos considerados que eran 5, 10, 20, 30 y 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar es similar a la detección de caso normales, alcanzando el 60 % de anomalías detectadas y valores cercanos al 53 % de



identificación de casos normales. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 40 vecinos consigue mejores resultados frente a las otras alternativas.

| value | type  | samples | dims  | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|-------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 1552.4  | 116.0 | 53.0018     | 0.483±0.0916  | 0.3819±0.0246 | 0.4644±0.0452 | 0.4324±0.0421 |
| 10    | test  | 1552.4  | 116.0 | 53.0018     | 0.5294±0.0827 | 0.4381±0.0522 | 0.5132±0.0377 | 0.4837±0.0386 |
| 20    | test  | 1552.4  | 116.0 | 53.0018     | 0.5885±0.0902 | 0.5134±0.076  | 0.5767±0.0414 | 0.551±0.043   |
| 30    | test  | 1552.4  | 116.0 | 53.0018     | 0.6021±0.093  | 0.5346±0.0724 | 0.5927±0.0431 | 0.5683±0.0455 |
| 40    | test  | 1552.4  | 116.0 | 53.0018     | 0.6084±0.0881 | 0.5378±0.0674 | 0.5971±0.0373 | 0.5731±0.0402 |
| 5     | train | 6209.6  | 116.0 | 53.0018     | 0.424±0.0194  | 0.4765±0.0204 | 0.4774±0.021  | 0.4503±0.0199 |
| 10    | train | 6209.6  | 116.0 | 53.0018     | 0.4806±0.019  | 0.513±0.0161  | 0.5266±0.0179 | 0.4968±0.0174 |
| 20    | train | 6209.6  | 116.0 | 53.0018     | 0.5551±0.0164 | 0.5749±0.0153 | 0.5955±0.0157 | 0.565±0.0158  |
| 30    | train | 6209.6  | 116.0 | 53.0018     | 0.5766±0.0105 | 0.5949±0.0129 | 0.6162±0.0118 | 0.5858±0.0117 |
| 40    | train | 6209.6  | 116.0 | 53.0018     | 0.5786±0.0113 | 0.6002±0.0112 | 0.6201±0.011  | 0.5894±0.0111 |

Figura 43: Estudio parámetros del algoritmo Knn para el curso FFF

## Lof

En la tabla 44 se muestran los resultados obtenidos por el algoritmo lof tanto para test como para train con los diferentes parámetros considerados que eran 5, 10, 20, 30, 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar está entorno al 21 % manteniendo una alta especificidad. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 40 vecinos consigue mejores resultados frente a las otras alternativas.

| value | type  | samples | dims  | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|-------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 1552.4  | 116.0 | 53.0018     | 0.186±0.0341  | 0.9581±0.0185 | 0.8388±0.0539 | 0.572±0.015   |
| 10    | test  | 1552.4  | 116.0 | 53.0018     | 0.1901±0.0433 | 0.9652±0.0223 | 0.8714±0.0693 | 0.5777±0.0178 |
| 20    | test  | 1552.4  | 116.0 | 53.0018     | 0.2103±0.0387 | 0.9731±0.018  | 0.9052±0.0609 | 0.5917±0.0176 |
| 30    | test  | 1552.4  | 116.0 | 53.0018     | 0.2195±0.0423 | 0.9756±0.0186 | 0.9104±0.0713 | 0.5976±0.025  |
| 40    | test  | 1552.4  | 116.0 | 53.0018     | 0.2197±0.0255 | 0.98±0.0154   | 0.9254±0.0591 | 0.5999±0.0172 |
| 5     | train | 6209.6  | 116.0 | 53.0018     | 0.1152±0.0037 | 0.9827±0.0015 | 0.8826±0.011  | 0.549±0.0024  |
| 10    | train | 6209.6  | 116.0 | 53.0018     | 0.1429±0.0026 | 0.9823±0.0034 | 0.9013±0.0181 | 0.5626±0.0026 |
| 20    | train | 6209.6  | 116.0 | 53.0018     | 0.1592±0.0022 | 0.9842±0.003  | 0.9194±0.0151 | 0.5717±0.0025 |
| 30    | train | 6209.6  | 116.0 | 53.0018     | 0.1639±0.0019 | 0.984±0.0023  | 0.9202±0.0112 | 0.5739±0.002  |
| 40    | train | 6209.6  | 116.0 | 53.0018     | 0.1705±0.0037 | 0.9851±0.0038 | 0.9282±0.0182 | 0.5778±0.0036 |

Figura 44: Estudio parámetros del algoritmo LOF para el curso FFF

**10.6.1.7. Curso GGG**

En esta sección se van a mostrar los resultados para el curso GGG por los diferentes algoritmos con las diferentes configuraciones consideradas.

**Autoencoder**

En la tabla 45 se muestran los resultados obtenidos por el algoritmo autoencoder tanto para test como para train con los diferentes valores de dimensión de la capa latente considerados que eran 16, 32, 64, 128, 256. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. En la tabla se muestra las diferentes opciones con respecto al valor concreto que se ha usado para determinar la dimensión de la capa latente, disminuir este valor puede ser útil para reducir la dimensionalidad y eliminar el ruido, pero también puede resultar en una pérdida de información. Este algoritmo, al utilizar únicamente valores normales durante la fase de entrenamiento, no permite calcular varias de las métricas comúnmente utilizadas en esta etapa. Sin embargo, al analizar los resultados en la fase de prueba, se observa que los diferentes valores de los parámetros producen resultados similares. No obstante, los valores más reducidos tienden a mostrar un mejor rendimiento. En particular, un valor que utiliza una representación de 16 dimensiones demuestra obtener los mejores resultados. En general, se produce un error mayor en la detección de las anomalías, siendo los valores de sensibilidad más reducidos, indicando que existen anomalías que no es capaz de identificar correctamente. Con respecto a la desviación típica se muestra reducida, con lo que sus resultados son estables en las diferentes ejecuciones.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 16    | test  | 506.8   | 63.0 | 40.2526     | 0.4708±0.057  | 0.9715±0.025  | 0.9164±0.0745 | 0.7212±0.0393 |
| 32    | test  | 506.8   | 63.0 | 40.2526     | 0.3729±0.0256 | 0.9148±0.017  | 0.7473±0.0506 | 0.6439±0.0213 |
| 64    | test  | 506.8   | 63.0 | 40.2526     | 0.3578±0.0266 | 0.9044±0.0174 | 0.7167±0.0521 | 0.6311±0.022  |
| 128   | test  | 506.8   | 63.0 | 40.2526     | 0.3587±0.0199 | 0.905±0.0132  | 0.7186±0.0392 | 0.6319±0.0165 |
| 256   | test  | 506.8   | 63.0 | 40.2526     | 0.3622±0.0288 | 0.9074±0.0191 | 0.7255±0.0569 | 0.6348±0.0239 |
| 16    | train | 1211.2  | 63.0 | 0.0         | 0.0±0.0       | 0.7999±0.0003 | 0.0±0.0       | nan±nan       |
| 32    | train | 1211.2  | 63.0 | 0.0         | 0.0±0.0       | 0.7999±0.0003 | 0.0±0.0       | nan±nan       |
| 64    | train | 1211.2  | 63.0 | 0.0         | 0.0±0.0       | 0.7999±0.0003 | 0.0±0.0       | nan±nan       |
| 128   | train | 1211.2  | 63.0 | 0.0         | 0.0±0.0       | 0.7999±0.0003 | 0.0±0.0       | nan±nan       |
| 256   | train | 1211.2  | 63.0 | 0.0         | 0.0±0.0       | 0.7999±0.0003 | 0.0±0.0       | nan±nan       |

Figura 45: Estudio parámetros del algoritmo Autoencoder para el curso GGG

**Cblof**



## CAPÍTULO 10. RESULTADOS EXPERIMENTALES

En la tabla 46 se muestran los resultados obtenidos por el algoritmo cblof tanto para test como para train con los diferentes números de clústeres considerados, que eran 6, 7, 8, 9 y 10. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar es muy reducido, siendo el valor de sensibilidad solamente entre un 8 % y 12 %. Los diferentes valores de clústeres produce resultados parecidos, obteniendo ligeramente mejores valores cuanto más alto es el número de clústeres.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 6     | test  | 506.8   | 63.0 | 40.2526     | 0.0812±0.0487 | 0.802±0.1125  | 0.2674±0.1869 | 0.4416±0.0707 |
| 7     | test  | 506.8   | 63.0 | 40.2526     | 0.0712±0.0507 | 0.773±0.1176  | 0.2153±0.1731 | 0.4221±0.075  |
| 8     | test  | 506.8   | 63.0 | 40.2526     | 0.0812±0.0714 | 0.7663±0.1241 | 0.2231±0.1928 | 0.4237±0.0865 |
| 9     | test  | 506.8   | 63.0 | 40.2526     | 0.0866±0.0716 | 0.7718±0.1541 | 0.264±0.2146  | 0.4292±0.1017 |
| 10    | test  | 506.8   | 63.0 | 40.2526     | 0.1273±0.0554 | 0.8676±0.1208 | 0.4588±0.1768 | 0.4974±0.0758 |
| 6     | train | 2027.2  | 63.0 | 40.2525     | 0.0718±0.0418 | 0.8807±0.0282 | 0.2885±0.168  | 0.4762±0.035  |
| 7     | train | 2027.2  | 63.0 | 40.2525     | 0.0602±0.0407 | 0.8729±0.0274 | 0.2419±0.1635 | 0.4665±0.0341 |
| 8     | train | 2027.2  | 63.0 | 40.2525     | 0.0645±0.046  | 0.8759±0.031  | 0.2593±0.1847 | 0.4702±0.0385 |
| 9     | train | 2027.2  | 63.0 | 40.2525     | 0.0734±0.0495 | 0.8819±0.0333 | 0.2951±0.1988 | 0.4776±0.0414 |
| 10    | train | 2027.2  | 63.0 | 40.2525     | 0.1095±0.0358 | 0.9061±0.0241 | 0.44±0.1439   | 0.5078±0.03   |

Figura 46: Estudio parámetros del algoritmo CBLOF para el curso GGG

## Cof

En la tabla 47 se muestran los resultados obtenidos por el algoritmo cof tanto para test como para train con los diferentes números de vecinos considerados que eran 5, 10, 20, 30 y 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías se encuentra entre un 23 % hasta un 46 % manteniendo una alta especificidad. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 40 vecinos consigue mejores resultados frente a las otras alternativas.

## Dae



| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 506.8   | 63.0 | 40.2526     | 0.2324±0.0446 | 0.9194±0.0188 | 0.6563±0.0902 | 0.5759±0.03   |
| 10    | test  | 506.8   | 63.0 | 40.2526     | 0.2324±0.0263 | 0.9399±0.0102 | 0.7219±0.0418 | 0.5861±0.0141 |
| 20    | test  | 506.8   | 63.0 | 40.2526     | 0.3186±0.035  | 0.9584±0.0177 | 0.8382±0.0648 | 0.6385±0.0232 |
| 30    | test  | 506.8   | 63.0 | 40.2526     | 0.3774±0.078  | 0.9604±0.0246 | 0.8644±0.0901 | 0.6689±0.0428 |
| 40    | test  | 506.8   | 63.0 | 40.2526     | 0.4686±0.083  | 0.9293±0.0308 | 0.8127±0.0853 | 0.699±0.0554  |
| 5     | train | 2027.2  | 63.0 | 40.2525     | 0.1843±0.0023 | 0.9566±0.0015 | 0.7409±0.0092 | 0.5704±0.0019 |
| 10    | train | 2027.2  | 63.0 | 40.2525     | 0.1809±0.0067 | 0.9542±0.0045 | 0.7271±0.0268 | 0.5676±0.0056 |
| 20    | train | 2027.2  | 63.0 | 40.2525     | 0.1706±0.0066 | 0.9473±0.0045 | 0.6857±0.0266 | 0.559±0.0055  |
| 30    | train | 2027.2  | 63.0 | 40.2525     | 0.1659±0.0085 | 0.9442±0.0057 | 0.667±0.034   | 0.555±0.0071  |
| 40    | train | 2027.2  | 63.0 | 40.2525     | 0.186±0.0053  | 0.9577±0.0035 | 0.7478±0.0212 | 0.5719±0.0044 |

Figura 47: Estudio parámetros del algoritmo COF para el curso GGG

En la tabla 48 se muestran los resultados obtenidos por el algoritmo dae tanto para test como para train con los diferentes valores de dimensión de la capa latente considerados que eran 16, 32, 64, 128, 256. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test en este caso se presentan diferentes, debido a que este algoritmo se entrena solamente con valores normales, por tanto, durante la fase de train nos encontramos con métricas que no pueden calcularse. Con respecto a la desviación típica se muestra reducida, con lo que sus resultados son estables. En este caso, el número de anomalías que es capaz de detectar se queda en el 50 % manteniendo una especificidad perfecta. Se puede suponer que es debido al número de anomalías que se presenta en este curso. Se aprecia que resultados similares son obtenidos a partir del uso de un tamaño de dimensión igual a 32, no afectando mucho el valor de este parámetro en los resultados.

| value | type  | samples | dims | anomalyRate | se            | sp            | p       | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------|---------------|
| 16    | test  | 506.8   | 63.0 | 40.2526     | 0.4993±0.0026 | 1.0±0.0       | 1.0±0.0 | 0.7496±0.0013 |
| 32    | test  | 506.8   | 63.0 | 40.2526     | 0.4993±0.0026 | 1.0±0.0       | 1.0±0.0 | 0.7496±0.0013 |
| 64    | test  | 506.8   | 63.0 | 40.2526     | 0.4994±0.0029 | 1.0±0.0       | 1.0±0.0 | 0.7497±0.0015 |
| 128   | test  | 506.8   | 63.0 | 40.2526     | 0.4995±0.003  | 1.0±0.0       | 1.0±0.0 | 0.7497±0.0015 |
| 256   | test  | 506.8   | 63.0 | 40.2526     | 0.4995±0.003  | 1.0±0.0       | 1.0±0.0 | 0.7497±0.0015 |
| 16    | train | 1211.2  | 63.0 | 0.0         | 0.0±0.0       | 0.7999±0.0003 | 0.0±0.0 | nan±nan       |
| 32    | train | 1211.2  | 63.0 | 0.0         | 0.0±0.0       | 0.7999±0.0003 | 0.0±0.0 | nan±nan       |
| 64    | train | 1211.2  | 63.0 | 0.0         | 0.0±0.0       | 0.7999±0.0003 | 0.0±0.0 | nan±nan       |
| 128   | train | 1211.2  | 63.0 | 0.0         | 0.0±0.0       | 0.7999±0.0003 | 0.0±0.0 | nan±nan       |
| 256   | train | 1211.2  | 63.0 | 0.0         | 0.0±0.0       | 0.7999±0.0003 | 0.0±0.0 | nan±nan       |

Figura 48: Estudio parámetros del algoritmo DAE para el curso GGG

## Hbos



## CAPÍTULO 10. RESULTADOS EXPERIMENTALES

En la tabla 49 se muestran los resultados obtenidos por el algoritmo hbos tanto para test como para train con los diferentes tamaños de los intervalos considerados que eran 5, 10, 15, 20, 25. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar no alcanza ni el 8 % siendo también la especificidad más reducida que en las anteriores, estable sobre un 71 %. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de intervalos considerados, aunque se trata de resultados con un rendimiento muy bajo para detectar las anomalías.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 506.8   | 63.0 | 40.2526     | 0.0725±0.0263 | 0.7072±0.154  | 0.158±0.0629  | 0.3899±0.0763 |
| 10    | test  | 506.8   | 63.0 | 40.2526     | 0.0588±0.0166 | 0.7198±0.1449 | 0.1367±0.0491 | 0.3893±0.0699 |
| 15    | test  | 506.8   | 63.0 | 40.2526     | 0.0559±0.0152 | 0.7284±0.1445 | 0.1344±0.0403 | 0.3921±0.0681 |
| 20    | test  | 506.8   | 63.0 | 40.2526     | 0.0559±0.0142 | 0.7218±0.1384 | 0.1309±0.041  | 0.3888±0.0663 |
| 25    | test  | 506.8   | 63.0 | 40.2526     | 0.0568±0.0167 | 0.7211±0.1412 | 0.1326±0.0478 | 0.389±0.0677  |
| 5     | train | 2027.2  | 63.0 | 40.2525     | 0.0424±0.008  | 0.861±0.0054  | 0.1705±0.0322 | 0.4517±0.0067 |
| 10    | train | 2027.2  | 63.0 | 40.2525     | 0.0351±0.005  | 0.856±0.0034  | 0.1409±0.0203 | 0.4455±0.0042 |
| 15    | train | 2027.2  | 63.0 | 40.2525     | 0.0353±0.0055 | 0.8562±0.0037 | 0.1419±0.0221 | 0.4458±0.0046 |
| 20    | train | 2027.2  | 63.0 | 40.2525     | 0.0341±0.005  | 0.8553±0.0033 | 0.1369±0.02   | 0.4447±0.0042 |
| 25    | train | 2027.2  | 63.0 | 40.2525     | 0.0343±0.0057 | 0.8555±0.0039 | 0.1379±0.0229 | 0.4449±0.0048 |

Figura 49: Estudio parámetros del algoritmo HBOS para el curso GGG

## Knn

En la tabla 50 se muestran los resultados obtenidos por el algoritmo knn tanto para test como para train con los diferentes números de vecinos considerados que eran 5, 10, 20, 30 y 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar es similar a la detección de caso normales, alcanzando el 58 % de anomalías detectadas y valores cercanos al 46 % de identificación de casos normales. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 40 vecinos consigue mejores resultados frente a las otras alternativas.

## Lof





| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 506.8   | 63.0 | 40.2526     | 0.3657±0.0439 | 0.3672±0.0521 | 0.2809±0.0363 | 0.3665±0.0416 |
| 10    | test  | 506.8   | 63.0 | 40.2526     | 0.4108±0.0609 | 0.3903±0.0615 | 0.313±0.0493  | 0.4006±0.0553 |
| 20    | test  | 506.8   | 63.0 | 40.2526     | 0.5078±0.0431 | 0.4286±0.0351 | 0.3743±0.0186 | 0.4682±0.0213 |
| 30    | test  | 506.8   | 63.0 | 40.2526     | 0.5686±0.0681 | 0.4537±0.0367 | 0.4116±0.0403 | 0.5112±0.0453 |
| 40    | test  | 506.8   | 63.0 | 40.2526     | 0.5863±0.0448 | 0.4669±0.0312 | 0.4256±0.0285 | 0.5266±0.0321 |
| 5     | train | 2027.2  | 63.0 | 40.2525     | 0.3314±0.0083 | 0.4296±0.0046 | 0.2813±0.0062 | 0.3805±0.0059 |
| 10    | train | 2027.2  | 63.0 | 40.2525     | 0.3831±0.0139 | 0.4435±0.0046 | 0.3168±0.0094 | 0.4133±0.0091 |
| 20    | train | 2027.2  | 63.0 | 40.2525     | 0.4721±0.0269 | 0.5008±0.0185 | 0.3892±0.0218 | 0.4864±0.0223 |
| 30    | train | 2027.2  | 63.0 | 40.2525     | 0.5194±0.0211 | 0.5236±0.0135 | 0.4235±0.0167 | 0.5215±0.0172 |
| 40    | train | 2027.2  | 63.0 | 40.2525     | 0.5348±0.0237 | 0.5281±0.0176 | 0.433±0.0201  | 0.5314±0.0206 |

Figura 50: Estudio parámetros del algoritmo Knn para el curso GGG

En la tabla 51 se muestran los resultados obtenidos por el algoritmo lof tanto para test como para train con los diferentes parámetros considerados que eran 5, 10, 20, 30, 40. Los resultados que se muestran son los valores de Sensibilidad, Especificidad, Precisión y Roc medios para los 5 fold y las 10 semillas consideradas por fold junto con la desviación típica, en el anexo se muestran las tablas completas por fold y semilla. Resultados de train y test son similares y las desviaciones típicas son reducidas, con lo que se aprecia resultados estables. En este caso, el número de anomalías que es capaz de detectar está entorno al 25 % manteniendo una alta especificidad. Se aprecia que mejores resultados son alcanzados conforme aumenta el número de vecinos considerados, en este caso el uso de 40 vecinos consigue mejores resultados frente a las otras alternativas.

| value | type  | samples | dims | anomalyRate | se            | sp            | p             | roc           |
|-------|-------|---------|------|-------------|---------------|---------------|---------------|---------------|
| 5     | test  | 506.8   | 63.0 | 40.2526     | 0.1834±0.0131 | 0.9419±0.0199 | 0.687±0.0733  | 0.5626±0.0124 |
| 10    | test  | 506.8   | 63.0 | 40.2526     | 0.1814±0.015  | 0.9287±0.031  | 0.6472±0.0931 | 0.555±0.0148  |
| 20    | test  | 506.8   | 63.0 | 40.2526     | 0.1422±0.0241 | 0.9293±0.0373 | 0.5978±0.1347 | 0.5357±0.0243 |
| 30    | test  | 506.8   | 63.0 | 40.2526     | 0.1951±0.0407 | 0.9511±0.0285 | 0.7368±0.1164 | 0.5731±0.0279 |
| 40    | test  | 506.8   | 63.0 | 40.2526     | 0.25±0.0619   | 0.9577±0.0261 | 0.795±0.1307  | 0.6039±0.0411 |
| 5     | train | 2027.2  | 63.0 | 40.2525     | 0.1287±0.0026 | 0.9737±0.0019 | 0.7677±0.0121 | 0.5512±0.0013 |
| 10    | train | 2027.2  | 63.0 | 40.2525     | 0.1358±0.008  | 0.9574±0.0052 | 0.6833±0.016  | 0.5466±0.0019 |
| 20    | train | 2027.2  | 63.0 | 40.2525     | 0.1302±0.0069 | 0.9412±0.0051 | 0.5988±0.0309 | 0.5357±0.0056 |
| 30    | train | 2027.2  | 63.0 | 40.2525     | 0.1615±0.0028 | 0.9521±0.0044 | 0.6948±0.0233 | 0.5568±0.0035 |
| 40    | train | 2027.2  | 63.0 | 40.2525     | 0.1708±0.0102 | 0.9571±0.0065 | 0.7284±0.0386 | 0.5639±0.0078 |

Figura 51: Estudio parámetros del algoritmo LOF para el curso GGG

### 10.6.2. Comparación de algoritmos por cursos

En esta sección, se estudiarán los resultados obtenidos por cada uno de los algoritmos utilizando la mejor configuración determinada en la sección anterior. Los resultados presentados en este estudio corresponden a los promedios de las pruebas realizadas para cada propuesta y curso.



| course | algorithm   | value | type | samples  | dims  | anomalyRate | se            | sp            | p             | roc           |
|--------|-------------|-------|------|----------|-------|-------------|---------------|---------------|---------------|---------------|
| AAA    | autoencoder | 16    | test | 149.6    | 31.0  | 29.0103     | 0.5779±0.0432 | 0.9527±0.0173 | 0.8343±0.0609 | 0.7653±0.0302 |
| AAA    | cblof       | 6     | test | 149.5833 | 31.0  | 29.0107     | 0.1886±0.0732 | 0.9355±0.0189 | 0.5352±0.087  | 0.562±0.0329  |
| AAA    | cof         | 30    | test | 149.6    | 31.0  | 29.0103     | 0.3216±0.0948 | 0.9266±0.0141 | 0.6337±0.0583 | 0.624±0.0439  |
| AAA    | dae         | 32    | test | 149.6    | 31.0  | 29.0103     | 0.6489±0.0376 | 0.9822±0.0157 | 0.9374±0.0552 | 0.8155±0.0265 |
| AAA    | hbos        | 25    | test | 149.6    | 31.0  | 29.0103     | 0.0921±0.039  | 0.8833±0.0343 | 0.2567±0.1294 | 0.4877±0.0301 |
| AAA    | knn         | 40    | test | 149.6    | 31.0  | 29.0103     | 1.0±0.0       | 0.7363±0.0937 | 0.6184±0.0785 | 0.8681±0.0468 |
| AAA    | lof         | 5     | test | 149.6    | 31.0  | 29.0103     | 0.2402±0.0636 | 0.9398±0.0222 | 0.6202±0.1174 | 0.59±0.0337   |
| BBB    | autoencoder | 16    | test | 1581.8   | 90.0  | 52.5351     | 0.3458±0.0389 | 0.949±0.0351  | 0.8818±0.0833 | 0.6474±0.0356 |
| BBB    | cblof       | 6     | test | 1581.8   | 90.0  | 52.5351     | 0.1206±0.06   | 0.9065±0.0619 | 0.6024±0.0796 | 0.5136±0.0147 |
| BBB    | cof         | 40    | test | 1581.8   | 90.0  | 52.5351     | 0.2696±0.0368 | 0.9664±0.0174 | 0.9035±0.0382 | 0.618±0.0134  |
| BBB    | dae         | 256   | test | 1581.8   | 90.0  | 52.5351     | 0.4103±0.0267 | 0.9989±0.0013 | 0.9975±0.003  | 0.7046±0.0136 |
| BBB    | hbos        | 25    | test | 1581.8   | 90.0  | 52.5351     | 0.0464±0.0226 | 0.7102±0.1405 | 0.1536±0.0248 | 0.3784±0.0595 |
| BBB    | knn         | 40    | test | 1581.8   | 90.0  | 52.5351     | 0.5603±0.0836 | 0.5224±0.1527 | 0.5744±0.0623 | 0.5414±0.0489 |
| BBB    | lof         | 40    | test | 1581.8   | 90.0  | 52.5351     | 0.2115±0.0435 | 0.9648±0.0348 | 0.8903±0.0725 | 0.5882±0.0086 |
| CCC    | autoencoder | 16    | test | 886.8    | 47.0  | 62.1561     | 0.3031±0.0168 | 0.964±0.0231  | 0.9323±0.0435 | 0.6335±0.0194 |
| CCC    | cblof       | 6     | test | 886.8    | 47.0  | 62.1561     | 0.1059±0.0401 | 0.9052±0.0512 | 0.6625±0.0608 | 0.5055±0.0102 |
| CCC    | cof         | 30    | test | 886.8    | 47.0  | 62.1561     | 0.2638±0.0331 | 0.9696±0.0173 | 0.9363±0.0339 | 0.6167±0.0161 |
| CCC    | dae         | 64    | test | 886.8    | 47.0  | 62.1561     | 0.3256±0.0067 | 0.9994±0.0012 | 0.9989±0.0023 | 0.6625±0.0036 |
| CCC    | hbos        | 25    | test | 886.8    | 47.0  | 62.1561     | 0.0338±0.0162 | 0.7694±0.0502 | 0.1826±0.0577 | 0.4016±0.0185 |
| CCC    | knn         | 40    | test | 886.8    | 47.0  | 62.1561     | 0.5011±0.0595 | 0.4989±0.0429 | 0.6205±0.0232 | 0.5±0.0237    |
| CCC    | lof         | 40    | test | 886.8    | 47.0  | 62.1561     | 0.1578±0.0181 | 0.9785±0.0176 | 0.9291±0.0541 | 0.5682±0.0092 |
| DDD    | autoencoder | 16    | test | 1254.4   | 84.0  | 58.3865     | 0.3252±0.0152 | 0.9708±0.0186 | 0.9397±0.0385 | 0.648±0.0164  |
| DDD    | cblof       | 6     | test | 1254.4   | 84.0  | 58.3865     | 0.1168±0.036  | 0.9124±0.0533 | 0.6765±0.0936 | 0.5146±0.0201 |
| DDD    | cof         | 40    | test | 1254.4   | 84.0  | 58.3865     | 0.2603±0.051  | 0.9694±0.0225 | 0.9249±0.0434 | 0.6148±0.0251 |
| DDD    | dae         | 256   | test | 1254.4   | 84.0  | 58.3865     | 0.3469±0.0086 | 0.9997±0.0007 | 0.9994±0.0014 | 0.6733±0.0044 |
| DDD    | hbos        | 25    | test | 1254.4   | 84.0  | 58.3865     | 0.054±0.0318  | 0.8073±0.1778 | 0.4203±0.2206 | 0.4307±0.0767 |
| DDD    | knn         | 30    | test | 1254.4   | 84.0  | 58.3865     | 0.58±0.095    | 0.5751±0.1211 | 0.6628±0.031  | 0.5775±0.0173 |
| DDD    | lof         | 30    | test | 1254.4   | 84.0  | 58.3865     | 0.1821±0.0373 | 0.9805±0.0063 | 0.9272±0.024  | 0.5813±0.0188 |
| EEE    | autoencoder | 16    | test | 586.8    | 37.0  | 43.7627     | 0.4262±0.0511 | 0.9672±0.0341 | 0.9093±0.0957 | 0.6967±0.0417 |
| EEE    | cblof       | 6     | test | 586.8    | 37.0  | 43.7627     | 0.076±0.0232  | 0.8751±0.0421 | 0.328±0.0787  | 0.4755±0.0206 |
| EEE    | cof         | 10    | test | 586.8    | 37.0  | 43.7627     | 0.3505±0.0508 | 0.9436±0.0204 | 0.8336±0.0398 | 0.6471±0.0184 |
| EEE    | dae         | 32    | test | 586.8    | 37.0  | 43.7627     | 0.459±0.0038  | 0.9989±0.0015 | 0.9968±0.0042 | 0.7289±0.0025 |
| EEE    | hbos        | 25    | test | 586.8    | 37.0  | 43.7627     | 0.0296±0.0216 | 0.8006±0.0333 | 0.1041±0.0735 | 0.4151±0.0233 |
| EEE    | knn         | 40    | test | 586.8    | 37.0  | 43.7627     | 0.5794±0.0185 | 0.5382±0.06   | 0.496±0.03    | 0.5588±0.0263 |
| EEE    | lof         | 30    | test | 586.8    | 37.0  | 43.7627     | 0.2118±0.0291 | 0.9745±0.0263 | 0.8852±0.0983 | 0.5932±0.0144 |
| FFF    | autoencoder | 16    | test | 1552.4   | 116.0 | 53.0018     | 0.3714±0.0106 | 0.9892±0.0083 | 0.9749±0.0193 | 0.6803±0.0085 |
| FFF    | cblof       | 6     | test | 1552.4   | 116.0 | 53.0018     | 0.1232±0.0423 | 0.9143±0.0433 | 0.6339±0.0686 | 0.5188±0.0135 |
| FFF    | cof         | 40    | test | 1552.4   | 116.0 | 53.0018     | 0.3014±0.0288 | 0.9605±0.0266 | 0.8994±0.056  | 0.631±0.0187  |
| FFF    | dae         | 32    | test | 1552.4   | 116.0 | 53.0018     | 0.378±0.0007  | 0.9995±0.0007 | 0.9989±0.0015 | 0.6888±0.0006 |
| FFF    | hbos        | 5     | test | 1552.4   | 116.0 | 53.0018     | 0.0693±0.0302 | 0.665±0.1365  | 0.1944±0.0551 | 0.3671±0.0575 |
| FFF    | knn         | 40    | test | 1552.4   | 116.0 | 53.0018     | 0.6084±0.0881 | 0.5378±0.0674 | 0.5971±0.0373 | 0.5731±0.0402 |
| FFF    | lof         | 40    | test | 1552.4   | 116.0 | 53.0018     | 0.2197±0.0255 | 0.98±0.0154   | 0.9254±0.0591 | 0.5999±0.0172 |
| GGG    | autoencoder | 16    | test | 506.8    | 63.0  | 40.2526     | 0.4708±0.057  | 0.9715±0.025  | 0.9164±0.0745 | 0.7212±0.0393 |
| GGG    | cblof       | 10    | test | 506.8    | 63.0  | 40.2526     | 0.1273±0.0554 | 0.8676±0.1208 | 0.4588±0.1768 | 0.4974±0.0758 |
| GGG    | cof         | 40    | test | 506.8    | 63.0  | 40.2526     | 0.4686±0.083  | 0.9293±0.0308 | 0.8127±0.0853 | 0.699±0.0554  |
| GGG    | dae         | 32    | test | 506.8    | 63.0  | 40.2526     | 0.4993±0.0026 | 1.0±0.0       | 1.0±0.0       | 0.7496±0.0013 |
| GGG    | hbos        | 25    | test | 506.8    | 63.0  | 40.2526     | 0.0568±0.0167 | 0.7211±0.1412 | 0.1326±0.0478 | 0.389±0.0677  |
| GGG    | knn         | 40    | test | 506.8    | 63.0  | 40.2526     | 0.5863±0.0448 | 0.4669±0.0312 | 0.4256±0.0285 | 0.5266±0.0321 |
| GGG    | lof         | 40    | test | 506.8    | 63.0  | 40.2526     | 0.25±0.0619   | 0.9577±0.0261 | 0.795±0.1307  | 0.6039±0.0411 |

Figura 52: Comparativa mejor configuración por algoritmo y curso



En la Tabla 52, podemos ver que, para el curso AAA, el algoritmo DAE muestra el mejor equilibrio entre las medidas de sensibilidad y especificidad. Esto significa que DAE logra una detección efectiva de los estudiantes que abandonan o no superan las asignaturas comparado con el resto de propuestas sin sacrificar la identificación de los estudiantes que superar el curso. De este modo, proporciona una evaluación más equilibrada en términos generales. A continuación, el método de Autoencoder también muestra un rendimiento competitivo, aunque ligeramente inferior en cuanto al equilibrio entre sensibilidad y especificidad.

En el caso del algoritmo KNN, este algoritmo se distingue por su capacidad para identificar todos los alumnos que suspenden o abandonan el curso, lo que optimiza la sensibilidad. Sin embargo, esta alta sensibilidad viene a costa de una reducción significativa en los valores de especificidad y precisión. Como resultado, aunque el KNN es eficaz para detectar casos positivos, sus limitaciones en especificidad y precisión hacen que no sea una opción adecuada para este contexto. Estos mismos resultados también se reflejan en el resto de los cursos, indicando que DAE y Autoencoder son las propuestas más robustas para la detección en distintos contextos educativos. Sin embargo, es importante destacar que, debido al alto nivel de anomalías presentes en los cursos, incluso estos métodos avanzados no logran una optimización completa de los resultados. En términos generales, las mejores propuestas permiten una detección de casos entre el 40 % y el 60 %. Para mejorar los resultados, podría ser beneficioso explorar métodos más especializados que estén adaptados específicamente al contexto del problema en cuestión.

Finalmente, se ha realizado un estudio estadístico para comparar los algoritmos utilizando todos los conjuntos de datos. Este análisis tiene como objetivo determinar si existen diferencias significativas en el rendimiento de los distintos algoritmos en la detección de anomalías en todos los cursos. Para ello, se ha aplicado el test de Friedman [31], una prueba estadística no paramétrica que evalúa si hay diferencias significativas entre varios algoritmos en varios conjuntos de datos. Este test es adecuado en este contexto porque permite comparar múltiples métodos en términos de sus resultados de rendimiento, controlando al mismo tiempo las variaciones entre los conjuntos de datos. De este modo podemos obtener una visión más precisa de las diferencias observadas en los resultados y si éstas son significativas o si podrían ser atribuibles al azar o a las particularidades de los conjuntos de datos específicos.

Tabla 11 muestra los valores medios de sensibilidad de cada algoritmo en cada conjunto de datos. Estos valores determinan el porcentaje de anomalías identificadas correctamente, que en nuestro problema corresponde a los estudiantes que no superan el curso. En la fila de Ranking Medio se presenta el ranking obtenido mediante el test de Friedman. Este ranking se calcula asignando rangos a las puntuaciones de cada algoritmo en cada conjunto de datos. Para cada conjunto de datos, se asignan rangos a las puntuaciones de los algoritmos, donde el algoritmo

## CAPÍTULO 10. RESULTADOS EXPERIMENTALES

| dataset  | AE    | CBLOF | COF   | DAE   | HBOS  | KNN          | LOF   |
|--|-------|-------|-------|-------|-------|--------------|-------|
| AAA  | 0.578 | 0.189 | 0.322 | 0.649 | 0.092 | <b>1.000</b> | 0.240 |
| BBB  | 0.346 | 0.121 | 0.270 | 0.410 | 0.046 | <b>0.560</b> | 0.211 |
| CCC  | 0.303 | 0.106 | 0.264 | 0.326 | 0.034 | <b>0.501</b> | 0.158 |
| DDD  | 0.325 | 0.117 | 0.260 | 0.347 | 0.054 | <b>0.580</b> | 0.182 |
| EEE  | 0.426 | 0.076 | 0.350 | 0.459 | 0.030 | <b>0.579</b> | 0.212 |
| FFF  | 0.371 | 0.123 | 0.301 | 0.378 | 0.069 | <b>0.608</b> | 0.220 |
| GGG  | 0.471 | 0.127 | 0.469 | 0.499 | 0.057 | <b>0.586</b> | 0.250 |
| Ranking Medio  | 3     | 6     | 4     | 2     | 7     | 1            | 5     |
| Friedman $\chi^2 = 42$ , df = 6, p-value = 1.839e-07 |       |       |       |       |       |              |       |

Tabla 11: Resultados medios de sensibilidad para cada algoritmo y conjunto de datos

con el mejor rendimiento recibe el rango 1, el segundo mejor recibe el rango 2, y así sucesivamente. Luego, se calculan los rangos promedio para cada algoritmo a través de todos los conjuntos de datos. Estos valores reflejan la posición relativa de cada algoritmo en comparación con los demás y proporcionan una visión clara de su desempeño en términos de sensibilidad media. Un rango más bajo indica un mejor desempeño relativo del algoritmo en los conjuntos de datos analizados. Para esta medida, se observa que el algoritmo kNN obtiene un rango medio de 1, indicando que tiene los mejores valores en todos los cursos. El test de Friedman rechaza la hipótesis nula con un nivel de confianza del 99 %, lo que sugiere que existen diferencias significativas en los resultados alcanzados por los distintos algoritmos. Para identificar cuáles algoritmos tienen un rendimiento significativamente peor, se ha aplicado el post-test de Shaffer [31]. Los resultados de este post-test se muestran en la Figura 53, donde los algoritmos LOF, CBLOF y HBOS son considerados como las peores propuestas con una confianza del 99 %.

Un estudio similar es llevado a cabo para la medida de especificidad, Tabla 12 muestra los valores medios de especificidad de cada algoritmo en cada conjunto de datos. Estos valores determinan el porcentaje de estudiantes que superan el curso y son identificados correctamente. De nuevo, el ranking medio proporcionado por el test de Friedman indica que el algoritmo DAE obtiene un valor de 1, lo que significa que optimiza esta métrica en comparación con las demás propuestas para todos los cursos analizados. El test de Friedman rechaza la hipótesis nula con un 99 % de confianza, y los resultados del post-test de Shaffer se muestran en la Figura 54, donde se indica que los algoritmos CBLOF, HBOS y KNN son considerados las peores propuestas. Esto sugiere que DAE, AE y COF son las únicas propuestas para las que no se han detectado diferencias significativas en sus resultados tanto en las medidas de sensibilidad como en especificidad, lo cual es importante ya que ambas métricas deben ser optimizadas. De hecho, DAE obtiene el mejor ranking para la medida de especificidad y el segundo mejor ranking para la medida de sensibilidad. Como se ha observado en el estudio, DAE se puede considerar la propuesta más robusta para resolver este problema en los diferentes cursos.

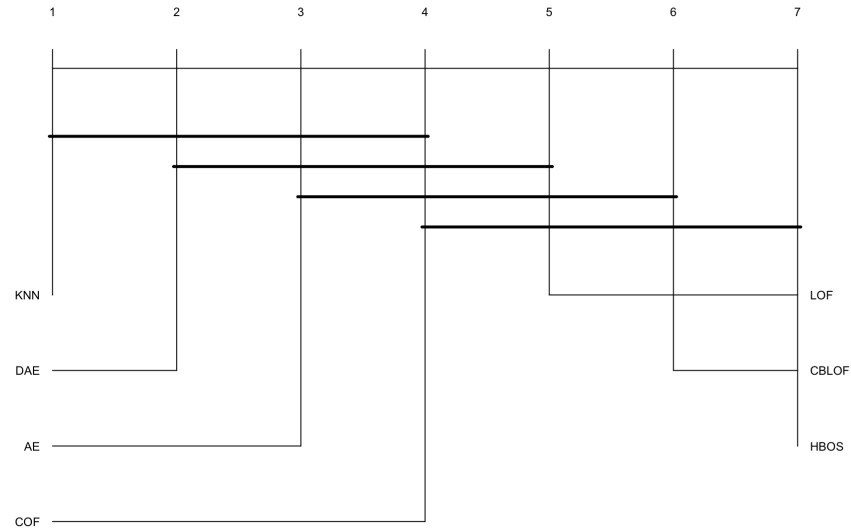


Figura 53: Distancia crítica del post-test de Shaffer's para los resultados de sensibilidad

| dataset  | AE    | CBLOF | COF   | DAE          | HBOS  | KNN   | LOF   |
|--|-------|-------|-------|--------------|-------|-------|-------|
| AAA  | 0.953 | 0.935 | 0.927 | <b>0.982</b> | 0.883 | 0.736 | 0.940 |
| BBB  | 0.949 | 0.906 | 0.966 | <b>0.999</b> | 0.710 | 0.522 | 0.965 |
| CCC  | 0.964 | 0.905 | 0.970 | <b>0.999</b> | 0.769 | 0.499 | 0.979 |
| DDD  | 0.971 | 0.912 | 0.969 | <b>1.000</b> | 0.807 | 0.575 | 0.981 |
| EEE  | 0.967 | 0.875 | 0.944 | <b>0.999</b> | 0.801 | 0.538 | 0.975 |
| FFF  | 0.989 | 0.914 | 0.961 | <b>1.000</b> | 0.665 | 0.538 | 0.980 |
| GGG  | 0.972 | 0.868 | 0.929 | <b>1.000</b> | 0.721 | 0.467 | 0.958 |
| Ranking Medio  | 2.857 | 4.857 | 3.714 | 1.000        | 6.000 | 7.000 | 2.571 |
| Friedman $\chi^2 = 39.245$ , df = 6, p-value = 6.408e-07 |       |       |       |              |       |       |       |

Tabla 12: Resultados medios de especificidad para cada algoritmo y conjunto de datos

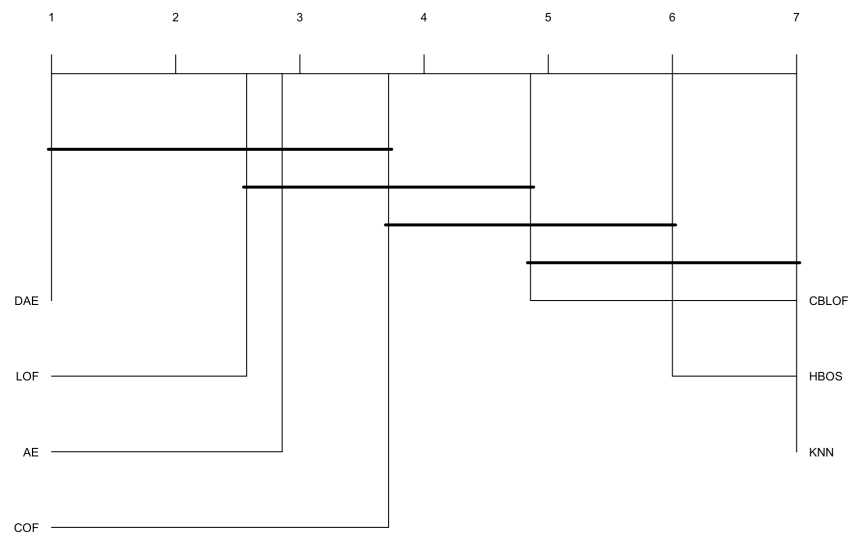


Figura 54: Distancia crítica del post-test de Shaffer's para los resultados de especificidad

# 11 Pruebas

## 11.1. Proceso de Pruebas y Validación

Durante el desarrollo de este proyecto, se ha seguido una metodología rigurosa de pruebas y validación para asegurar la correcta implementación de los algoritmos y la funcionalidad general del sistema. Las pruebas realizadas se clasifican en tres categorías principales: pruebas unitarias, pruebas funcionales y pruebas de integración.

### 11.1.1. Pruebas Unitarias

Las pruebas unitarias se realizaron para verificar que cada función, método y clase del sistema funciona correctamente de manera aislada. La tabla 1 resume los resultados de las pruebas unitarias realizadas:

| Componente                              | Pasó test | Observaciones |
|---|-----------|---------------|
| Inyección de dependencias               | Sí        | -             |
| Contenedor de dependencias              | Sí        | -             |
| Servicio de cálculo de métricas         | Sí        | -             |
| Servicio de entrenamiento de algoritmos | Sí        | -             |
| Servicio de ejecución en Docker         | Sí        | -             |
| Factoría de algoritmos                  | Sí        | -             |
| Repositorio de ejecuciones              | Sí        | -             |
| Repositorio de entrenamientos           | Sí        | -             |

Tabla 1: Resultados de las pruebas unitarias



### 11.1.2. Pruebas Funcionales

Las pruebas funcionales verifican que las funcionalidades de la aplicación cumplen con los requisitos especificados. Se llevaron a cabo pruebas para cada caso de uso, asegurando que todas las funcionalidades implementadas operen correctamente. A continuación se presentan los resultados de las pruebas funcionales:

| Funcionalidad                 | Pasó test | Observaciones |
|-------------------------------|-----------|---------------|
| Entrenar modelo               | Sí        | -             |
| Ejecutar modelo               | Sí        | -             |
| Crear informe                 | Sí        | -             |
| Configurar modelo             | Sí        | -             |
| Particionar datos             | Sí        | -             |
| Transformar datos             | Sí        | -             |
| Aplicar modelo                | Sí        | -             |
| Realizar estudio experimental | Sí        | -             |

Tabla 2: Resultados de las pruebas funcionales

### 11.1.3. Pruebas de Integración

Las pruebas de integración se realizaron para verificar la correcta interacción entre los diferentes componentes del sistema. Se prestó especial atención a la integración de los algoritmos de detección de anomalías y su correcto funcionamiento dentro de la librería. Los resultados de las pruebas de integración se detallan a continuación:

| Componente Integrado                | Pasó test | Observaciones                   |
|-------------------------------------|-----------|---------------------------------|
| Integración de AutoEncoder          | No        | Problema detallado en 11.2.0.1. |
| Integración de AutoEncoder          | Sí        | -                               |
| Integración de CBLOF                | Sí        | -                               |
| Integración de COF                  | Sí        | -                               |
| Integración de DAE                  | Sí        | -                               |
| Integración de HBOS                 | Sí        | -                               |
| Integración de KNN                  | Sí        | -                               |
| Integración de LOF                  | Sí        | -                               |
| Integración de LSTM-VAE             | Sí        | -                               |
| Integración con TimeEval            | Sí        | -                               |
| Integración con servicios de Docker | Sí        | -                               |

Tabla 3: Resultados de las pruebas de integración



## **11.2. Errores y Soluciones Durante las Pruebas**

### **11.2.0.1. Cálculo del ROC**

Durante las pruebas de los algoritmos entrenados exclusivamente con datos normales, se encontró un error al intentar calcular la curva ROC (Receiver Operating Characteristic). Este problema surge porque la curva ROC requiere tanto verdaderos positivos como falsos positivos para su cálculo, y con datos únicamente normales, estos valores no pueden generarse. La solución implementada fue ajustar el procedimiento para omitir el cálculo de la curva ROC en estos casos, permitiendo así la evaluación correcta de otros parámetros.

### **11.2.0.2. Cambios en el Funcionamiento de TimeEval Durante el Desarrollo**

Durante el desarrollo, al realizar un test para comprobar que la instalación de la librería era transparente y estaba actualizada, TimeEval cambió su método de instalación de algoritmos. Gracias a la forma en que se modeló el sistema, no se produjeron cambios en su estructura ni en su código. Sin embargo, fue necesario actualizar el método de instalación para adaptarse a esta modificación.

## **11.3. Conclusión de las Pruebas**

Las pruebas realizadas han permitido validar la correcta implementación y funcionamiento de los diferentes componentes del sistema. La metodología seguida ha garantizado que cada parte del sistema, desde unidades individuales hasta la integración completa, opere según lo esperado, asegurando así la robustez y fiabilidad del sistema en su conjunto.

## 12 Conclusiones

Una vez acabadas las fases de desarrollo de la librería se expondrán las conclusiones que se han llegado durante el desarrollo.

### 12.1. Análisis de objetivos

#### 1. Diseño y desarrollo de un módulo de formato de datos común:

Se ha logrado establecer un formato estándar para la carga y manipulación de conjuntos de datos. La librería incluye métodos robustos para el particionado y transformación.

#### 2. Creación de un Módulo de Algoritmos de Detección de Anomalías:

El módulo desarrollado integra diversos algoritmos de detección de anomalías, todos adaptados a un formato común de configuración. Entre los algoritmos incluidos se encuentran k-nearest neighbors (knn), local outlier factor (lof), clustering-based local outlier factor (cblof), autoencoders, variational autoencoders (vae) con lstm, deep autoencoders (dae), histogram-based outlier score (hbos) y connectivity-based outlier factor (cof).

#### 3. Desarrollo de un módulo para Generación de Informes:

Se ha implementado un módulo que permite la generación de informes estandarizados para cualquier algoritmo incluido en la librería. Esta estandarización facilita la interpretación y comparación de los resultados obtenidos, proporcionando una herramienta útil para el análisis y la presentación de los datos.

#### 4. Implementación de un módulo para Estudios Experimentales Comparativos:



El módulo diseñado proporciona herramientas para realizar análisis comparativos de diferentes algoritmos, conjuntos de datos y métodos, utilizando un formato de configuración uniforme. Este módulo ha demostrado ser efectivo en la evaluación comparativa de distintas propuestas, permitiendo realizar estudios bajo condiciones controladas y ofreciendo una plataforma sólida para la investigación experimental.

#### **5. Creación de un módulo Tutorial:**

Se ha desarrollado un módulo educativo que incluye ejemplos prácticos de las funcionalidades principales de la librería. Este módulo sirve como guía esencial para nuevos usuarios, proporcionando una visión clara de cómo utilizar la librería de manera eficaz y maximizar sus capacidades.

#### **6. Extensibilidad y Adaptabilidad de la Librería:**

La arquitectura de la librería ha sido diseñada para permitir su fácil expansión y adaptación. Esta característica asegura que la librería pueda evolucionar y adaptarse a nuevas técnicas y enfoques en el campo de la detección de anomalías, garantizando su relevancia y utilidad a largo plazo.

Adicionalmente, se ha llevado a cabo un caso de estudio de un problema real para demostrar la efectividad de los módulos desarrollados. Este caso de estudio ha permitido validar las funcionalidades y el rendimiento de la librería en un entorno práctico, asegurando que cumple con los objetivos planteados y proporcionando una base sólida para futuras extensiones y mejoras.



## 12.2. Futuras mejoras

Aunque se han cumplido todos los objetivos planteados en un inicio, existen posibles mejoras para la aplicación que se pueden implementar en el futuro. Dichas mejoras son las siguientes:

- Módulo de vista interactiva para la creación de estudios: Considerando que actualmente se realiza a través de un fichero de configuración, el siguiente paso sería implementar una vista más interactiva que facilite la creación de experimentos.
- Incorporación de más algoritmos: La librería ha sido diseñada con la flexibilidad necesaria para permitir la adición de nuevos algoritmos. Esto asegurará que, a medida que el usuario requiera algoritmos no soportados en la versión actual, pueda integrarlos y ejecutarlos fácilmente desde la librería.

# Bibliografía

- [1] Jakub Kuzilek, Martin Hlosta, and Zdenek Zdrahal. Open university learning analytics dataset, 2017. URL [https://analyse.kmi.open.ac.uk/open\\_dataset](https://analyse.kmi.open.ac.uk/open_dataset). Consultado el: 2024-06-09.
- [2] Bogdan Dumitrescu, Andra Băltoiu, and Ștefania Budulan. Anomaly detection in graphs of bank transactions for anti money laundering applications. *IEEE Access*, 10:47699–47714, 2022.
- [3] Bedeuro Kim, Mohsen Ali Alawami, Eunsoo Kim, Sanghak Oh, Jeongyong Park, and Hyounghick Kim. A comparative study of time series anomaly detection models for industrial control systems. *Sensors*, 23(3):1310, 2023.
- [4] D.M. Hawkins. *Identification of Outliers*. Chapman and Hall, 1980.
- [5] Véronne Yepmo, Grégory Smits, and Olivier Pivert. Anomaly explanation: A review. *Data & Knowledge Engineering*, 137:101946, 2022.
- [6] Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection, 2020. URL <https://pyod.readthedocs.io/en/latest/index.html>. Consultado el 2023-12-10.
- [7] dsmi-lab ntust. Anomaly detection toolbox, 2023. URL <http://dsmi-lab-ntust.github.io/AnomalyDetectionToolbox/>. Consultado el 2023-12-10.
- [8] Domain Logic. Clean architecture: Qué es, importancia y beneficios para tu empresa. *Domain Logic Blog*, 2020. URL <https://domainlogic.io/clean-architecture-que-es-importancia-y-beneficios-para-tu-empresa/>. Consultado el 2023-12-10.
- [9] Duncan Geddes. What is a product design specification (pds)?, 2022. URL <https://technicalfoamservices.co.uk/blog/what-is-a-product-design-specification-pds/>. Accessed: 2024-07-12.
- [10] Microsoft. Visual studio code, 2023. URL <https://code.visualstudio.com/>. Consultado el 2023-12-10.



- [11] Python Software Foundation. Pep 8 – style guide for python code, 2024. URL <https://www.python.org/dev/peps/pep-0008/>. Consultado el 2024-04-05.
- [12] Python Software Foundation. Black: The uncompromising code formatter, 2024. URL <https://pypi.org/project/black/>. Consultado el 2024-01-05.
- [13] Ken Schwaber and Jeff Sutherland. The scrum guide, 2022. URL <https://www.scrum.org/resources/scrum-guide>. Accessed: 2024-07-12.
- [14] N.S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 1992.
- [15] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. *ACM SIGMOD Record*, 2000.
- [16] J. Tang, Z. Chen, A. W.-C. Fu, and D. Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *PAKDD*, 2002.
- [17] Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 2003.
- [18] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 2006.
- [19] D.P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [20] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 2010.
- [21] M. Goldstein and A. Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: Poster and Demo Track*, 2012.
- [22] TimeEval. Timeeval algorithms, 2024. URL <https://github.com/TimeEval/TimeEval-algorithms/tree/main>. Consultado el: 2024-06-06.
- [23] Arthur Zimek Erich Schubert. Elki: Environment for developing kdd-applications supported by index-structures, 2023. URL <https://elki-project.github.io/>. Consultado el 2023-12-10.
- [24] Writelatex Limited. Overleaf: Collaborative writing and publishing, 2023. URL <https://www.overleaf.com/>. Consultado el 2023-12-10.

BIBLIOGRAFÍA

---

- [25] Software Freedom Conservancy. Git, 2023. URL <https://git-scm.com/>. Consultado el 2023-12-10.
- [26] Python Software Foundation. Python: A dynamic, open source programming language, 2023. URL <https://www.python.org/>. Consultado el 2023-12-10.
- [27] Inc. Docker. Docker: Empowering app development for developers, 2023. URL <https://www.docker.com/>. Consultado el 2023-12-10.
- [28] IBM. Casos de uso, 2008. URL <https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/lifecycle-management/6.0.1?topic=cases-use>. Consultado el 2024-06-06.
- [29] Alistair Cockburn. Hexagonal architecture, 2005. URL <https://alistair.cockburn.us/hexagonal-architecture/>. Consultado el: 2024-06-09.
- [30] Robert C. Martin. Design principles and design patterns, 2000. URL <https://www.goodreads.com/series/160070-object-mentor-solid-design-papers>. Consultado el : 2024-06-09.
- [31] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006. ISSN 1533-7928.