UNIVERSIDAD DE MURCIA

ESCUELA INTERNACIONAL DE DOCTORADO

TESIS DOCTORAL

Clustering and subgroup discovery for patient phenotyping

Agrupamiento y descubrimiento de subgrupos para fenotipado de pacientes

D. Antonio López Martínez-Carrasco

2023

UNIVERSIDAD DE MURCIA

ESCUELA INTERNACIONAL DE DOCTORADO

TESIS DOCTORAL

Clustering and subgroup discovery for patient phenotyping

Agrupamiento y descubrimiento de subgrupos para fenotipado de pacientes

Autor: D. Antonio López Martínez-Carrasco

Directores: D. Manuel Campos Martínez y D. José Manuel Juárez Herrero

UNIVERSIDAD DE MURCIA

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR

Aprobado por la Comisión General de Doctorado el 19-10-2022

D./Dña. Antonio López Martínez-Carrasco

doctorando del Programa de Doctorado en

Informática

de la Escuela Internacional de Doctorado de la Universidad Murcia, como autor/a de la tesis presentada para la obtención del título de Doctor y titulada:

Clustering and subgroup discovery for patient phenotyping / Agrupamiento y descubrimiento de subgrupos para fenotipado de pacientes

y dirigida por,

D./Dña. Manuel Campos Martínez

D./Dña. José Manuel Juárez Herrero

D./Dña.

DECLARO QUE:

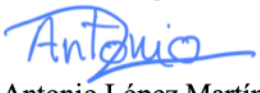
La tesis es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, de acuerdo con el ordenamiento jurídico vigente, en particular, la Ley de Propiedad Intelectual (R.D. legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, modificado por la Ley 2/2019, de 1 de marzo, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), en particular, las disposiciones referidas al derecho de cita, cuando se han utilizado sus resultados o publicaciones.

Si la tesis hubiera sido autorizada como tesis por compendio de publicaciones o incluyese 1 o 2 publicaciones (como prevé el artículo 29.8 del reglamento), declarar que cuenta con:

- *La aceptación por escrito de los coautores de las publicaciones de que el doctorando las presente como parte de la tesis.*
- *En su caso, la renuncia por escrito de los coautores no doctores de dichos trabajos a presentarlos como parte de otras tesis doctorales en la Universidad de Murcia o en cualquier otra universidad.*

Del mismo modo, asumo ante la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad del contenido de la tesis presentada, en caso de plagio, de conformidad con el ordenamiento jurídico vigente.

En Murcia, a 18 de julio de 2023



Fdo.: Antonio López Martínez-Carrasco

Esta DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD debe ser insertada en la primera página de la tesis presentada para la obtención del título de Doctor.

Información básica sobre protección de sus datos personales aportados	
Responsable:	Universidad de Murcia. Avenida teniente Flomesta, 5. Edificio de la Convalecencia. 30003; Murcia. Delegado de Protección de Datos: dpd@um.es
Legitimación:	La Universidad de Murcia se encuentra legitimada para el tratamiento de sus datos por ser necesario para el cumplimiento de una obligación legal aplicable al responsable del tratamiento. art. 6.1.c) del Reglamento General de Protección de Datos
Finalidad:	Gestionar su declaración de autoría y originalidad
Destinatarios:	No se prevén comunicaciones de datos
Derechos:	Los interesados pueden ejercer sus derechos de acceso, rectificación, cancelación, oposición, limitación del tratamiento, olvido y portabilidad a través del procedimiento establecido a tal efecto en el Registro Electrónico o mediante la presentación de la correspondiente solicitud en las Oficinas de Asistencia en Materia de Registro de la Universidad de Murcia

Agradecimientos

Quiero agradecer a todas las personas que han hecho posible la realización de esta tesis doctoral y cuyo apoyo ha sido crucial en cada etapa de este proceso.

En primer lugar, deseo expresar mi profundo agradecimiento a mis padres, por su apoyo incondicional y constante respaldo durante todos estos años. Sin ellos, nada de esto habría sido posible. Del mismo modo, también al resto de mi familia.

En segundo lugar, a mis directores de tesis, cuya guía y orientación han sido fundamentales para el desarrollo de esta tesis y cuyo conocimiento y experiencia han permitido poder llevar a cabo esta investigación.

Asimismo, a mis amigos, quienes han estado a mi lado todo este tiempo y me han acompañado durante este viaje.

De igual manera, a la Universidad de Murcia y a la Escuela Internacional de Doctorado, que me han brindado la oportunidad de llevar a cabo esta investigación. También a todos los profesores y resto de personal, que han contribuido de diversas formas a mi formación y crecimiento académico.

Por último, quiero expresar mi gratitud de manera general a todas aquellas personas que han dejado una huella en mi vida y que, de una forma u otra, han formado parte de mi trayectoria personal, académica y profesional.

Funding

This work was funded by a national PhD grant (Ref: FPU18/02220) provided by the Spanish Ministry of Science, Innovation and Universities (MCIU). Moreover, this research was developed within the framework of the following research projects: (a) the SITSUS project (Ref: RTI2018-094832-B-I00), funded by the Spanish Ministry of Science, Innovation and Universities (MCIU), the Spanish Agency for Research (AEI), and the European Fund for Regional Development (FEDER), (b) the CONFAINCE project (Ref: PID2021-122194OB-I00), supported by the Spanish Ministry of Science and Innovation, the Spanish Agency for Research (MCIN/AEI/10.13039/501100011033) and ERDF A way of making Europe, and (c) the GRALENIA project (Ref: 2021/C005/00150055) supported by the Spanish Ministry of Economic Affairs and Digital Transformation, the Spanish Secretariat of State for Digitization and Artificial Intelligence, Red.es and NextGenerationEU funding. This research was also partially funded by a mobility grant (Ref: R-933/2021) provided by the University of Murcia.

Abstract

According to principal healthcare organisations, antimicrobial resistance (AMR) is a significant danger to human health worldwide, and is a critical issue in the medical field. AMR occurs when microorganisms become resistant to antimicrobial treatments, making the latter unable to combat infections effectively. Some of the principal causes of AMR are the inappropriate use of antimicrobials and the transfer of resistant microorganisms between humans, animals or the environment. This means that, despite the use of antimicrobial drugs to treat patients infected with resistant microorganisms, their excessive use and inadequate regulation promote the spread of these resistant microorganisms.

On the one hand, from the health and hospital point of view, it is essential to have resources, tools and procedures with which to monitor, detect and control possible cases of AMR, in addition to eradicating all potential threats to both patients and the rest of society. On the other hand, many efforts have been made in the clinical research field to address the AMR problem and to mitigate the effects and problems that it causes. In this context, finding sets of patients with interesting characteristics has become a core issue. This task is denominated as the patient phenotyping process, and these patient characteristics are denominated as phenotypes.

Machine Learning (ML) is a promising area in the field of computer science, since it provides a mechanism with which to research and develop new solutions when confronting certain problems such as that described in this work. More precisely, ML can be used for the automatic generation of patient phenotypes.

The hypothesis of this PhD thesis is that clustering and subgroup discovery (SD), which are two ML techniques, are effective as regards supporting the patient phenotyping process in the clinical context of antibiotic resistance. We hypothesize that refined and adapted versions of such techniques can generate phenotypes that are helpful and understandable for clinicians. In order to prove this hypothesis, we therefore establish the following objectives: (1) the use of clustering or SD as the basis on which to propose ML techniques for phenotyping whose results would be useful for clinical experts and

easy for them to understand; (2) the generation of patient phenotypes by designing a new unsupervised ML technique based on clustering; (3) the identification of patient phenotypes by proposing a new methodology that would allow clinical experts to become involved in the process; (4) the extraction of phenotypes by creating a new and efficient SD algorithm; (5) the definition of patient phenotypes by proposing the new problem of mining diverse top-k subgroup lists; (6) the facilitation of the use of all the SD algorithms developed in this research, along with others already existing in literature, by developing a public, accessible and open-source Python library, and (7) a guarantee of the reproducibility of the research by extracting and using clinical data related to the antibiotic resistance problem from a public repository.

Finally, the main conclusions of this PhD thesis in relation to the objectives proposed are that: (1) the new ML techniques created in this work can be successfully applied to the antibiotic resistance problem and their results are easy for clinicians to interpret; (2) the Trace-based clustering technique generates patient phenotypes; (3) the new 5-step methodology provides a straightforward guide with which to identify and rank patient phenotypes, and allows clinical experts to be involved in the discovery process; (4) the VLSD algorithm can be used either to directly extract patient phenotypes or as part of other phenotyping techniques; (5) the new problem of mining diverse top-k subgroup lists provides a new approach for patient phenotyping; (6) the 'subgroups' library can be easily accessed, since it is available on GitHub and PyPI and can be used by data scientists, ML researchers and end-users for tasks such as phenotyping, and (7) the MIMIC-III database is an excellent data source that provides rich data concerning the antibiotic resistance problem, helps researchers in this field, and ensures the reproducibility of research.

Resumen extendido

La resistencia antimicrobiana (antimicrobial resistance o AMR, por sus siglas en inglés) es, según las principales organizaciones sanitarias, una de las principales amenazas mundiales para la salud humana y uno de los problemas más alarmantes en el ámbito clínico. La resistencia antimicrobiana se produce cuando los microorganismos se vuelven resistentes a los antimicrobianos, lo que significa que estos pierden su capacidad para combatir las infecciones por microorganismos.

Algunas de las principales causas de la AMR son la utilización inadecuada de antimicrobianos y la propagación de microorganismos resistentes entre humanos, animales o el medio ambiente. Por este motivo, aunque la utilización de antimicrobianos es útil para tratar a pacientes infectados con microorganismos resistentes, su uso excesivo y un bajo control de este tipo de fármacos favorecen la propagación de estos microorganismos resistentes. Por ello, los Sistemas Nacionales de Salud instan a implementar marcos de control de los antimicrobianos y a utilizarlos de forma responsable para dar una respuesta eficaz a la AMR. Además, la OMS insta a realizar esfuerzos a todos los niveles y a llevar a cabo una colaboración internacional.

Por un lado, desde el punto de vista sanitario y hospitalario, es imprescindible disponer de recursos, herramientas y procedimientos para monitorizar, detectar y controlar los posibles casos de AMR, así como para erradicar todas las amenazas potenciales tanto para los pacientes como para el resto de la sociedad. Por otro lado, además de las acciones sanitarias y hospitalarias, se han realizado muchos esfuerzos en el campo de la investigación clínica para abordar el problema de la AMR y mitigar los efectos y problemas que causa. En este contexto, la búsqueda de conjuntos de pacientes con características interesantes se ha convertido en una cuestión central. Esta tarea se denomina fenotipado de pacientes y estas características de los pacientes se denominan fenotipos. Por tanto, el objetivo es descubrir fenotipos comunes y novedosos relativos al problema concreto estudiado, siendo útiles, por ejemplo, para revisar protocolos de administración específicos en hospitales o para apoyar la toma de decisiones de los clínicos, entre otros.

La inteligencia artificial (IA) es un campo multidisciplinar dedicado al desarrollo de

sistemas informáticos capaces de realizar tareas que normalmente requieren inteligencia humana como, por ejemplo, la percepción visual, la toma de decisiones, el diagnóstico o el procesamiento del lenguaje. Una de las ramas más importantes de la IA que ha progresado enormemente en los últimos tiempos es el aprendizaje computacional (machine learning o ML, por sus siglas en inglés). El objetivo principal del ML es desarrollar sistemas y métodos que aprendan, es decir, que mejoren automáticamente a través de la experiencia a la hora de realizar algunas tareas, mejorando así su rendimiento y precisión. Hoy en día se pueden encontrar diversas aplicaciones en el ámbito del ML como, por ejemplo, robots autónomos que reconocen el entorno y navegan sin intervención humana, sistemas que generan nuevas imágenes a partir de texto, sistemas de apoyo a la toma de decisiones utilizados en dominios complejos como la medicina o la economía, o modelos de lenguaje que pueden hablar de cualquier tema como lo haría un humano. En definitiva, el ML es un área prometedora, ya que proporciona un mecanismo para investigar y desarrollar nuevas soluciones ante determinados problemas como el descrito en este trabajo.

Uno de los aspectos clave de cualquier técnica de ML es la salida final que genera. Esta salida se denomina modelo y determina lo que la técnica de ML ha aprendido. En este contexto, un modelo es una representación matemática/estadística de una población definida por unos datos de entrada dados. En consecuencia, dados estos datos de entrada, el objetivo principal de un determinado algoritmo de ML es ajustar un modelo, que representa y contiene el conocimiento que este algoritmo de ML ha aprendido de los datos de entrada. Esto significa que el uso y desarrollo de un algoritmo de ML vendrá determinado por el tipo de modelo a obtener y sus características. Así, se pueden definir diferentes tipos de modelos en función de ciertos criterios como el número de individuos de la población que cubre (es decir, de qué parte de los datos procede el conocimiento aprendido) o la finalidad del modelo (es decir, para qué se utiliza), entre otros.

En primer lugar, un modelo puede ser global o local en función del número de individuos de la población que abarque. Por un lado, los modelos globales son los que cubren todos los datos de entrada, es decir, en los que el conocimiento aprendido se refiere a toda la población. Por otro lado, los modelos locales son los que cubren sólo una parte de los datos de entrada, es decir, en los que el conocimiento aprendido se refiere sólo a un subconjunto de la población.

Además, un modelo puede ser predictivo o descriptivo según su finalidad. Por un lado, los modelos predictivos son aquellos obtenidos por algoritmos de ML predictivos y utilizados para predecir o estimar resultados futuros de nuevos datos no utilizados al

ajustar el modelo. Por otro lado, los modelos descriptivos son los que se obtienen mediante algoritmos de ML descriptivos y se utilizan para describir o explicar los datos actuales utilizados al ajustar el modelo. Tanto los modelos predictivos como los descriptivos permiten descubrir automáticamente nuevo conocimiento que había quedado oculto en los datos disponibles.

Centrándonos en el proceso de fenotipado de pacientes, el ML puede utilizarse para generar fenotipos de pacientes de forma automática. De este modo, desde la perspectiva del ML, el proceso de fenotipado de pacientes puede interpretarse como el proceso de generación automática de modelos descriptivos que caracterizan o bien a todos los individuos de una población (mediante el uso de un modelo global) o bien a un subconjunto de ellos (mediante el uso de un modelo local).

Podemos destacar los siguientes tipos de técnicas de ML: no supervisadas, supervisadas y por refuerzo. En primer lugar, las técnicas de ML no supervisadas son aquellas que utilizan datos de entrada no etiquetados a la hora de ajustar el modelo (es decir, datos en los que no hay ningún atributo de especial interés sobre el resto) y cuyo objetivo es encontrar una estructura o alguna regularidad en los datos de entrada mediante el uso de ciertas estrategias como, por ejemplo, la medición de distancias entre los individuos o el conteo de patrones que ocurren más a menudo que otros. En segundo lugar, las técnicas de ML supervisadas son aquellas que utilizan datos de entrada etiquetados a la hora de ajustar el modelo (es decir, datos en los que se utiliza un atributo de interés como referencia) y cuyo objetivo es generar un mapeo entre los atributos de los datos y el atributo de interés utilizado como referencia (también llamado clase). La técnica de clustering es un ejemplo de técnica de ML no supervisada, mientras que la técnica de descubrimiento de subgrupos (subgroup discovery o SD, por sus siglas en inglés) es un ejemplo de técnica de ML supervisada. Además, el aprendizaje por refuerzo, a diferencia del resto de enfoques, se basa en la noción de recompensa acumulativa. Este tipo de aprendizaje se centra principalmente en resolver problemas relacionados con la maximización de las recompensas de las acciones realizadas por un agente en un entorno desconocido.

Volviendo al ámbito médico, a pesar de la amplia gama de técnicas y algoritmos de ML disponibles para generar fenotipos de pacientes de forma automática, su aplicación práctica en la investigación médica supone todo un reto. Uno de estos retos consiste en la selección de los valores iniciales de los parámetros y los atributos relevantes para llevar a cabo los experimentos, a lo que se suma la limitada disponibilidad de datos hospitalarios. Estos factores contribuyen a la obtención de resultados deficientes y

producen un segundo reto para los clínicos, que consiste en la realización de un proceso de evaluación posterior para revisar los historiales de los pacientes y valorar su importancia clínica. En consecuencia, no todos los fenotipos de pacientes generados automáticamente son valiosos desde el punto de vista clínico, lo que pone de relieve la necesidad de explorar perspectivas o interpretaciones alternativas de los mismos datos. Otro obstáculo es la falta de confianza de la comunidad médica en las técnicas de ML. Por lo tanto, es crucial garantizar la trazabilidad de las técnicas utilizadas, la transparencia de los modelos obtenidos y la participación activa de los expertos clínicos en el proceso.

La hipótesis de esta tesis doctoral es que las técnicas de clustering y SD son eficaces para apoyar el proceso de fenotipado de pacientes en el contexto clínico de la resistencia a los antibióticos. Nuestra hipótesis es que las versiones refinadas y adaptadas de dichas técnicas pueden generar fenotipos útiles y legibles para los clínicos. Por lo tanto, para probar esta hipótesis, proponemos los siguientes objetivos: (1) utilización de las técnicas de clustering o SD como base para proponer técnicas de ML para el fenotipado cuyos resultados sean útiles y fácilmente legibles por los expertos clínicos, (2) generación de fenotipos de pacientes mediante el diseño de una nueva técnica de ML no supervisada basada en clustering, (3) identificación de fenotipos de pacientes mediante la propuesta de una nueva metodología que permita involucrar a expertos clínicos en el proceso, (4) extracción de fenotipos mediante la creación de un nuevo y eficiente algoritmo de SD, (5) definición de fenotipos de pacientes con la propuesta del nuevo problema de minado de las top-k listas de subgrupos diversas, (6) facilitación del uso de todos los algoritmos de SD desarrollados en esta investigación, junto con otros ya existentes en la literatura, mediante el desarrollo de una biblioteca Python pública, accesible y de código abierto y (7) garantía de la reproducibilidad de la investigación mediante la extracción y el uso de datos clínicos relacionados con el problema de la resistencia a los antibióticos a partir de un repositorio público.

La organización de esta tesis doctoral se describe a continuación.

En primer lugar, extraemos datos clínicos de la base de datos MIMIC-III, la cual se utilizará a lo largo de toda la tesis doctoral. Esta base de datos es un repositorio de acceso público que contiene información relacionada con la salud de más de 45,000 pacientes que recibieron tratamiento en unidades de cuidados intensivos y alrededor de 60,000 ingresos ocurridos entre 2001 y 2012 en el Beth Israel Deaconess Medical Center (Estados Unidos). Esta base de datos incluye información procedente de diversas fuentes como, por ejemplo, datos demográficos, resultados de pruebas de laboratorio, mediciones de signos vitales (presión arterial, frecuencia cardíaca, saturación de oxígeno, etc.) e

información relativa a los medicamentos administrados.

A continuación, presentamos y explicamos la técnica de clustering basado en trazas (o Trace-based clustering), que es una técnica no supervisada basada en clustering cuyo objetivo es encontrar fenotipos de pacientes evaluando el solapamiento entre clusters de diferentes particiones. Una ventaja de esta técnica es que sus resultados son altamente legibles por expertos clínicos. Al mismo tiempo, también definimos y explicamos una metodología de 5 pasos basada en la técnica de clustering basado en trazas. Un aspecto clave de esta metodología es la participación de expertos clínicos en el proceso.

Seguidamente, definimos formalmente la técnica de SD y describimos otra de nuestras propuestas: un nuevo y eficiente algoritmo de SD y una novedosa estructura de datos utilizada para implementarlo. El SD es una técnica de ML cuyo propósito es identificar un conjunto de relaciones entre atributos de un conjunto de datos (denominadas subgrupos) con respecto a un atributo objetivo de interés. Además, sus resultados son también muy legibles para los expertos, ya que se trata de descripciones fácilmente comprensibles.

Tras lo anterior, definimos formalmente el nuevo problema de minado de las top-k listas de subgrupos diversas y lo aplicamos al fenotipado de pacientes. Este nuevo enfoque se basa en la técnica de SD, el modelo de lista de subgrupos y el principio de longitud de descripción mínima (minimum description length o MDL, por sus siglas en inglés). Además, los resultados generados por esta técnica son muy legibles para los expertos clínicos, ya que se trata de múltiples listas de subgrupos, cada una de ellas formada por una colección de subgrupos.

La última parte de esta tesis consiste en presentar y describir la librería 'subgroups', que es una librería pública, accesible y de código abierto para trabajar con la técnica SD. Esta librería está implementada en Python, incluye los componentes necesarios relacionados con la técnica SD y contiene una colección de algoritmos SD (todos los desarrollados en esta investigación, junto con otros ya existentes en la literatura).

Por último, a continuación se describen las conclusiones de esta tesis doctoral en relación con los objetivos iniciales propuestos.

Con respecto al primer objetivo, (1) tanto clustering como SD pueden servir de base para diseñar nuevas técnicas de ML para fenotipado cuyos resultados sean útiles y fácilmente legibles por los clínicos, (2) las nuevas técnicas de ML creadas en este trabajo a partir de clustering o SD pueden aplicarse con éxito al problema de la resistencia a los antibióticos y (3) tanto los modelos locales como los globales pueden servir para el fenotipado de pacientes.

En cuanto al segundo objetivo, (1) la técnica de clustering basado en trazas genera

fenotipos de pacientes y sus resultados son fácilmente legibles por los clínicos, (2) la generación de diferentes particiones a partir de un mismo conjunto de datos para evaluar su solapamiento reduce la dependencia de la aleatoriedad de las técnicas de clustering tradicionales, (3) el concepto de estabilidad permite identificar conjuntos interesantes de pacientes y es genérico, transversal y no dependiente de la técnica específica de ML utilizada, (4) una representación basada en un mapa de calor (o heat-map) ayuda a visualizar fácilmente el solapamiento entre una gran cantidad de pares de clusters y (5) una métrica estadística como la media permite hacer un ranking y filtrar los resultados obtenidos por la técnica de clustering basado en trazas.

Con relación al tercer objetivo, (1) la nueva metodología de 5 pasos proporciona una guía sencilla para identificar y hacer un ranking de los fenotipos de los pacientes y permite que los expertos clínicos participen en el proceso de descubrimiento, (2) el estadístico de Hopkins es una alternativa interesante a considerar para determinar los mejores valores del hiperparámetro k cuando el método del codo no es concluyente, (3) las técnicas de ranking de clusters acompañadas de métodos de visualización permiten a los usuarios seleccionar y analizar fácilmente un número reducido de clusters, (4) una evaluación basada en clasificación puede ser una alternativa cuando los historiales personales de los pacientes no pueden ser examinados por un experto clínico, (5) la alta precisión obtenida en la evaluación basada en clasificación proporciona pruebas objetivas de la validez de nuestra metodología y (6) la definición de conceptos genéricos como función de agrupamiento o función de emparejamiento permite aumentar la versatilidad de nuestra metodología.

En lo que respecta al cuarto objetivo, (1) el algoritmo VLSD puede utilizarse para extraer directamente fenotipos de pacientes o como parte de otras técnicas de fenotipado, (2) los resultados obtenidos por el algoritmo VLSD son fácilmente legibles por los usuarios, (3) el algoritmo VLSD tiene un mayor rendimiento en términos de tiempo de ejecución, uso máximo de memoria y nodos visitados con respecto al resto de algoritmos SD del estado del arte considerados debido al uso combinado de la estrategia de exploración de clases de equivalencia, una poda basada en estimación optimista y una poda basada en la matriz $\mathcal{M}$, (4) la utilización de técnicas de poda como las basadas en estimación optimista mejora el rendimiento general de un algoritmo de SD, (5) la utilización de valores umbral más altos al implementar una poda basada en estimación optimista permite que el algoritmo visite menos nodos con respecto a otros algoritmos que no implementan esta poda, (6) la poda basada en la matriz $\mathcal{M}$ permite al algoritmo visitar menos nodos con respecto a otros algoritmos que no implementan esta poda y (7) la estructura de datos

propuesta permite generar eficientemente refinamientos de subgrupos y calcular todas las medidas de calidad.

En cuanto al quinto objetivo, (1) el nuevo problema de minado de las top-k listas de subgrupos diversas proporciona un nuevo enfoque para el fenotipado de pacientes, (2) los algoritmos GMSL y DSLM pueden generar los top-k fenotipos diversos en forma de listas de subgrupos y sus resultados son útiles y fácilmente legibles por los expertos, (3) el uso de un factor de solapamiento garantiza una mayor diversidad en términos de cobertura, (4) el uso de un mecanismo de eliminación de refinamientos aumenta la diversidad en términos de descripciones, (5) la capacidad predictiva es una métrica útil para evaluar fenotipos desde una perspectiva objetiva, (6) la combinación del principio MDL con los algoritmos de ML existentes es un enfoque prometedor para resolver el nuevo problema definido y proporcionar una base sólida para los modelos y (7) la combinación del principio MDL y la técnica de SD es un enfoque bien fundamentado desde un punto de vista teórico y práctico y resuelve el problema descrito en esta investigación.

Con respecto al sexto objetivo, (1) la librería 'subgroups' es fácilmente comprensible por los científicos de datos ya que sigue una interfaz similar a *scikit-learn*, (2) la librería 'subgroups' ha sido diseñada para ser extensible, ya que la extensibilidad es una propiedad clave que permite a los usuarios contribuir fácilmente a una librería, (3) la librería 'subgroups' es fácilmente accesible ya que está disponible en GitHub y PyPI, (4) la librería 'subgroups' ya implementa varias medidas de calidad y algoritmos de SD, lo cual permite actualmente su uso por parte de un usuario final para tareas como el fenotipado y (5) la librería 'subgroups' dispone de tests y métricas de ejecución para validar y comparar las nuevas implementaciones de algoritmos, lo cual permite su uso por parte de investigadores en ML.

En lo que se refiere al séptimo objetivo, (1) la base de datos MIMIC-III es una excelente fuente de datos que proporciona abundante información sobre el problema de la resistencia a los antibióticos, ayuda a los investigadores en este campo y garantiza la reproducibilidad de la investigación y (2) las bases de datos clínicas públicas como MIMIC-III pueden utilizarse tanto para reproducir investigaciones existentes como para llevar a cabo nuevas investigaciones cuando no se dispone de otras fuentes de datos.

Contents

List of Figures

Figure **Page**

List of Tables

CHAPTER 1

Introduction

This chapter shows the motivation behind and the background fundamentals of this PhD thesis. Section 1.1 provides an introduction to the clinical context of this research into the importance of antibiotic resistance and patient phenotyping. Section 1.2 shows why Machine Learning (ML) is helpful as regards tackling these clinical problems, and this section focuses particularly on the two ML techniques that will be the pillars of the research: clustering and subgroup discovery (SD). Section 1.3 describes the hypothesis and objectives, and finally, Section 1.4 presents the structure of this document and describes the links between chapters and objectives.

1.1 The antibiotic resistance problem

Antimicrobial resistance (AMR) is, according to principal healthcare organisations, one of the major global threats to human health and one of the most alarming problems in the clinical field. It occurs when microorganisms become resistant to antimicrobials, meaning that antimicrobials lose their ability to combat microorganism infections.

The "Antimicrobial resistance surveillance in Europe" report (WHO Regional Office for Europe & European Centre for Disease Prevention and Control, 2022) already informs and warns about this alarming situation. According to this report, the consequences of AMR can be critical, since antibiotics are currently the most effective solution in the fight against microorganisms, and reduce their risks. Some of the immediate clinical consequences are a longer duration of an illness and an increased risk of death. It is similarly estimated that more than 670,000 infections that occur each year in the

EU/EEA are attributable to antibiotic-resistant bacteria, and that approximately 33,000 people die as a result of these infections, implying a cost of around 1,100 million Euros.

As explained in the aforementioned report, AMR can occur in different types of microorganisms, such as viruses, parasites, fungi or bacteria. Focusing on bacterial microorganisms, there are various reasons for the acquisition of resistance, such as DNA mutations or the acquisition of exogenous genes (external genes originating from other already resistant bacteria). This produces a situation in which a single bacterium can be simultaneously resistant to multiple antimicrobial agents. This is extremely problematic because it could reduce the number of alternative treatments available for this bacterium.

Some of the principal causes of AMR are the inappropriate use of antimicrobials and the spread of resistant microorganisms between humans, animals or the environment. It is for this reason that, although the utilisation of antimicrobials is useful as regards treating patients infected with resistant microorganisms, its excessive utilisation and the poor control of these types of drugs favour the spread of these resistant microorganisms. As a consequence, National Health Systems urge the implementation of frameworks with which to control antimicrobial agents and their responsible use in order to provide an effective response to AMR. Moreover, the WHO urges that efforts be made at all levels, along with international collaboration.

AMR also leads to a rise in overall healthcare expenses and an increase in spending on research into the development of new medications. Furthermore, it results in patients requiring extended hospital care, thus implying a longer duration of admissions to hospitals. This situation leads to a reduction in the availability of hospital beds and an increase in the number of people requiring hospitalisation.

From the health and hospital point of view, it is essential to have resources, tools and procedures with which to monitor, detect and control possible cases of AMR, in addition to eradicating all potential threats to both patients and the rest of society. One example of a global solution that has been proposed in order to control this problem is the development of Antimicrobial Stewardship Programmes (ASPs) in hospitals (Doron & Davidson, 2011). The objective of these programmes is to achieve the selection of the ideal combination of antibiotics, dose and treatment duration so as to attain the best clinical outcome that minimises the toxicity and future resistance to the antibiotic used. One of the most necessary aspects of these programmes is the constitution of a stewardship team in order to improve the communication and collaboration among the different hospital services so as to use antibiotics in a more rational manner. This

team is multidisciplinary, since it is composed of microbiologists, pharmacists, managers, epidemiologists and physicians. Recent efforts to develop systems with which to support ASP teams through data integration and alerts are based on Artificial Intelligence (AI), such as the WASPSS framework (Segura, Morales, Juarez, Campos, & Palacios, 2020).

In addition to the aforementioned health and hospital actions, many efforts have been made in the clinical research field to address the AMR problem and to mitigate the effects and problems that it causes. In this context, finding sets of patients with interesting characteristics has become a core issue. This task is denominated as the patient phenotyping process, and these patient characteristics are denominated as phenotypes (Wojczynski & Tiwari, 2008). The goal of this process is to discover common and innovative phenotypes concerning the specific problem studied that will be useful to, for example, review specific administration protocols in hospitals or support clinicians' decision-making, among others.

1.2 Machine Learning for patient phenotyping

AI is a multidisciplinary field devoted to the development of computer systems able to perform tasks that normally require human intelligence, such as visual perception, decision making, diagnosis or language processing (Nilsson, 1998; Russell & Norvig, 2020). One of the most important branches of AI that has progressed enormously in recent times is ML. The main objective of ML is to develop systems and methods that learn, i.e. that are automatically enhanced through experience when carrying out some tasks, thus improving their performance and accuracy (Alpaydin, 2004; Mitchell, 1997; Russell & Norvig, 2020). A variety of ML applications are currently available, such as autonomous robots that recognise the environment and navigate without human intervention, systems that generate new images from text, systems that support decision-making in complex domains such as medicine or the economy, or language models that can talk about any topic just like a human would. ML is, definitely, a relevant and promising area, since it provides a mechanism with which to research and develop new solutions when confronting certain problems such as that described in this work.

One of the key aspects of any ML technique is the final output that it generates. This output is denominated as a model and determines what the ML technique has learned. In this context, a model is a mathematical/statistical representation of a population defined by a given piece of input data. Given this input data, the main objective of a particular ML algorithm is consequently to fit a model, which represents and contains

the knowledge that this ML algorithm has learned from the input data. This means that the use and development of an ML algorithm will be driven by the type of model to be obtained and its characteristics. Furthermore, different types of models can be defined according to certain criteria, such as the number of individuals from the population that it covers (i.e. from which part of data the knowledge learned originates) or the purpose of the model (i.e. what it is used for), among others.

In the first place, a model can be global or local depending on the number of individuals from the population that it covers. On the one hand, global models are those that cover all input data, meaning that the knowledge learned refers to the whole population. On the other hand, local models are those that cover only a part of the input data, meaning that the knowledge learned refers only to a subset of the population.

In the second place, a model can be predictive or descriptive according to its purpose. On the one hand, predictive models are those obtained by predictive ML algorithms and used to predict or estimate future outcomes of new data not used when fitting the model. On the other hand, descriptive models are those obtained by descriptive ML algorithms and used to describe or explain the current data used when fitting the model. Both predictive and descriptive models make it possible to automatically discover new knowledge that had been hidden in the data available.

Focusing on the objectives of this thesis, ML can be used to generate patient phenotypes automatically. In this manner, from the ML perspective, the patient phenotyping process can be interpreted as the process of automatically generating descriptive models that characterise either all the individuals from a population (by using a global model) or a subset of them (by using a local model).

Some notable types of ML techniques can be highlighted: unsupervised, supervised and reinforcement (Alpaydin, 2004; Russell & Norvig, 2020). First, unsupervised ML techniques are those that use unlabelled input data when fitting the model (that is, data in which there is no attribute of special interest when compared to the rest) and whose objective is to find a structure or some type of regularity in the input data by using certain strategies such as measuring distances between the individuals or counting patterns that occur more often than others. Second, supervised ML techniques are those that use labelled input data when fitting the model (that is, data in which an attribute of interest is used as a reference) and whose objective is to generate a mapping between the data attributes and the attribute of interest used as a reference (also called a class). Additionally, reinforcement learning is, unlike the other approaches, based on the notion of accumulative reward. It is focused mainly on solving problems related to maximising

the rewards of the actions taken by an agent in an unknown environment.

A short introduction to clustering and SD techniques is now provided, since they are the basis of all the proposals made in this thesis.

Clustering is an unsupervised ML technique whose goal is to divide a given dataset into disjoint groups known as clusters. A clustering technique identifies these disjoint clusters by employing a measure in order to assess the similarities between individuals in the dataset. Various classical distances can be used, such as the Euclidean, Manhattan or cosine (Saxena et al., 2017).

There are several types of clustering methods, e.g., hierarchical clustering or partitional clustering. Hierarchical clustering establishes a cluster hierarchy in which clusters at level n are divided into multiple clusters at level $n+1$, preserving the same elements as their parent cluster. Conversely, partitional clustering methods directly partition the input dataset in order to create a set of k disjoint clusters.

According to the above, a clustering algorithm learns the cluster to which each individual from the input dataset belongs and generates a global model that represents/stores this knowledge.

SD is a supervised ML technique that is utilised for the descriptive and exploratory analysis of data. Its main objective is to discover a collection of relationships, referred to as subgroups, among attributes in a dataset with respect to a target attribute. SD is valuable as regards generating hypotheses, extracting general patterns, and facilitating data analysis and exploration (Atzmueller, 2015).

One key aspect of this technique is that of computing the quality of a subgroup obtained using an SD algorithm. Various quality measures are available for this purpose. A quality measure is generally a function that assigns a numerical value to a subgroup on the basis of specific properties. These quality measures can be divided into two groups: (1) quality measures for nominal target attributes, and (2) quality measures for numeric target attributes. Some examples of quality measures are Sensitivity, Specificity, Weighted Relative Accuracy (WRAcc), and Information Gain, among others. It is possible to adapt certain popular quality measures, such as those mentioned earlier, for their application to both nominal and numeric attributes.

According to the above, an SD algorithm learns a set of subgroups of a subset of individuals from a dataset along with their quality values that are computed by a certain quality measure, and generates a model that represents/stores this knowledge. A model obtained with an SD algorithm is, in general, local, since subgroups from the set of subgroups discovered might not describe all individuals from the input dataset. However,

note that a model generated by an SD algorithm will be global only in the specific case that subgroups from the set of subgroups discovered describe all individuals from the input dataset.

Finally, with regard to the medical field, despite the wide range of ML techniques and algorithms for the automatic generation of patient phenotypes that are available, their practical implementation in medical research is challenging. One challenge involves the selection of initial parameter values and relevant attributes for the realisation of experiments, which is compounded by the limited availability of hospital data. These factors contribute to the attainment of deficient results and produce a second challenge for clinicians, which consists of performing a post-evaluation process in order to review patients' records and assess their clinical significance. Not all automatically generated patient phenotypes are, therefore, clinically valuable, highlighting the need to explore alternative perspectives or interpretations of the same data. Another hurdle is the medical community's lack of confidence in ML techniques. In order to confront this challenge, some aspects of the utmost importance (Mühlbacher, Piringer, et al., 2014) are ensuring the traceability of the techniques employed, the legibility of the models obtained, and the active involvement of clinicians in the process.

1.3 Hypothesis and Objectives

The hypothesis of this PhD thesis is that clustering and SD techniques are effective as regards supporting the patient phenotyping process in the clinical context of antibiotic resistance. We hypothesize that refined and adapted versions of such techniques can generate helpful and readable phenotypes for clinicians.

In order to prove this hypothesis, we established the following objectives:

Objective 1: the use of clustering or SD as the basis on which to propose ML techniques for phenotyping whose results would be useful for clinical experts and easy for them to interpret.

Objective 2: the generation of patient phenotypes by designing a new unsupervised ML technique based on clustering.

Objective 3: the identification of patient phenotypes by proposing a new methodology that would allow clinical experts to become involved in the process.

Objective 4: the extraction of phenotypes by creating a new and efficient SD algorithm.

Objective 5: the definition of patient phenotypes by proposing the new problem of mining diverse top-k subgroup lists.

Objective 6: the facilitation of the use of all the SD algorithms developed in this research, along with others already existing in literature, by developing a public, accessible and open-source Python library.

Objective 7: a guarantee of the reproducibility of the research by extracting and using clinical data related to the antibiotic resistance problem from a public repository.

Objective 1 is a transversal objective, since the usefulness and readability of the phenotypes obtained are two necessary characteristics that all the proposals made in this thesis need to have. Objectives 2 and 3 are covered in Chapter 2, which proposes a new unsupervised ML technique and a new methodology for the generation of patient phenotypes that is based on it. Objective 4 is covered in Chapter 3, which formalises the SD technique by further defining a new SD algorithm with a new data structure that is used to implement it. Objective 5 is covered in Chapter 4, which defines the new problem of mining diverse top-k subgroup lists and proposes different algorithms based on the SD technique, the subgroup list model and the Minimum Description Length (MDL) principle for the extraction of diverse top-k patient phenotypes. Objective 6 is a transversal objective, since all of the SD algorithms developed in the different parts of this thesis are, along with others already existing in literature, implemented and packed together in a public, accessible and open-source Python library. More details on this library and its characteristics are provided in Chapter 5. Objective 7 is a transversal objective, since the data extracted from the public clinical repository is used in different parts of this PhD thesis.

This PhD thesis has, from the outset, been designed to follow the open-science principles, as shown in Objectives 6 and 7, which promote the use of open code, accessible resources, and public datasets in order to make our research reproducible.

1.4 Thesis structure

This thesis consists of several chapters, which deal with the objectives defined in Section 1.3. These chapters are specifically the following:

Chapter 1: this chapter introduces the fundamentals of the different parts of which this research is formed, in addition to putting all the work carried out into context.

In order to achieve the latter, this chapter shows the hypothesis and the objectives to be achieved in this thesis, along with its structure.

Chapter 2: this chapter presents the Trace-based clustering technique and proposes a 5-step methodology that applies this technique in order to identify patient phenotypes. This trace-based clustering technique is based on the clustering technique, and its purpose is to discover patient phenotypes in the form of patient sets by evaluating the overlap between clusters from different partitions. The results obtained when employing the Trace-based clustering technique are highly legible, since they are patient sets with all their attributes, and the methodology defined allows clinical experts to become involved in the process.

Chapter 3: this chapter formalises the SD technique and presents a new and efficient SD algorithm denominated as VLSD (Vertical List Subgroup Discovery) that combines an equivalence class exploration strategy and a pruning strategy based on an optimistic estimate. It also describes a new data structure that is used to implement the algorithm.

Chapter 4: this chapter defines the new problem of mining diverse top-k subgroup lists, which extends the SD technique by introducing the Subgroup List model and the MDL principle. New different algorithms with which to mine diverse top-k patient phenotypes are also presented. In this case, the results obtained are also highly readable, since they are multiple subgroups lists, each formed of a collection of subgroups.

Chapter 5: this chapter describes the 'subgroups' library, which is a public, accessible and open-source Python library that implements the components required in order to work with the SD technique and which contains a collection of SD algorithms (all of the algorithms developed in this research, along with others already existing in literature).

Chapter 6: this chapter sets out the final conclusions of this research, future work and the published papers that resulted from this thesis.

Finally, Table 1.1 provides s summary of the relation between the proposed objectives and the different chapters into which this thesis is structured.

	Chapter 2	**Chapter 3**	**Chapter 4**	**Chapter 5**
Objective 1	Transversal			
Objective 2	X			
Objective 3	X			
Objective 4		X		
Objective 5			X	
Objective 6	Transversal			
Objective 7	Transversal			

Table 1.1: Relation between the proposed objectives and the chapters of this thesis.

CHAPTER 2

Trace-based clustering and Methodology

In this chapter, we confront the problem of obtaining high-quality phenotypes using the clustering technique. We approach the problem by evaluating the overlap between clusters from different partitions, ensuring the readability of the results and involving the clinicians in the process. We, therefore, propose the Trace-based clustering technique and define a 5-step methodology based on this technique.

2.1 Motivation

Clustering is one of the most traditionally used unsupervised techniques in Machine Learning (ML). This technique is employed in a wide variety of fields, such as the medical field, to carry out data exploratory analysis. When using this technique, the data is partitioned in order to extract valuable knowledge from it. The proposal of a new technique based on clustering is, therefore, a convenient starting point for this PhD thesis. Moreover, this technique allows us to prove our hypothesis, since it is useful for generating legible patient phenotypes in the context of the antibiotic resistance problem.

In this research, we focus on partitional clustering to design our technique. One characteristic of this type of clustering is that it uses a random seed passed initially as a parameter. This means that these methods are conditioned by the random numbers generated from this seed, thus affecting the results generated by them. This is, in some cases, problematic since it makes the results obtained dependent on randomness, and it is challenging to know whether some individuals from the same cluster have been grouped either because they have common characteristics or because of the specific

random seed chosen. The technique proposed in this first chapter tackles this difficulty by obtaining several partitions from the same dataset and evaluating/tracing the overlap between their clusters.

In general, the essential elements of the proposal carried out and explained in this chapter are the following:

- We describe a new technique that allows the identification of sets of patients (patient phenotypes) by adapting and using traditional clustering algorithms.
- We define a new approach consisting of finding patient phenotypes by evaluating the overlap between clusters from different partitions generated by a given clustering method.
- We propose a new phenotyping methodology based on patient traceability, thus allowing the involvement of clinical experts in the process.

The remainder of this chapter is structured as follows: Section 2.2 provides a background to the clustering technique and introduces related works, while Section 2.3 shows the clinical problem studied in this work, along with the dataset extracted from a public repository and used in both this and other chapters of this PhD thesis. Section 2.4 shows and describes our proposal: the Trace-based clustering technique (Section 2.4.1) and the 5-step methodology for patient phenotyping (Section 2.4.2). Section 2.5 describes the configuration of the experiments for each step of the methodology and Section 2.6 provides a discussion of the results of the experiments and the applicability of the methodology. Finally, Section 2.7 explains the conclusions reached after carrying out the research.

2.2 Background

As explained in Chapter 1, there are different types of clustering methods, such as hierarchical clustering or partitional clustering. Each one works in a different way and has specific characteristics.

Focusing on partitional clustering, one algorithm that is of significant importance is K-Means. This algorithm uses the following elements as input: (1) a dataset, and (2) a specific value for k, which determines the number of clusters to be obtained. The algorithm partitions the elements of the input dataset into k clusters and returns them all. The K-Means algorithm initially selects random k elements (centroids) from the input dataset. It subsequently assigns the remaining elements to the nearest centroid

on the basis of a distance function, and updates the centroids for each cluster. These two phases are iteratively repeated until none of the centroids undergo any further changes.

Another clustering technique is that of clustering ensembles, which is also referred to as clustering aggregation (Boongoen & Iam-On, 2018; Vega-Pons & Ruiz-Shulcloper, 2011). This approach involves generating a collection of partitions by employing either the same algorithm with diverse initialisations or different clustering algorithms on the same input dataset. All the partitions (i.e. all their clusters) are subsequently combined to yield a final partition (that is, a final set of clusters). This technique typically comprises two primary phases: (1) the Generation phase, in which a set of partitions is created, and (2) the Consensus phase, in which the original partitions are integrated to obtain a new and unified partition.

Evaluating the partitions obtained and their clusters is a crucial task when executing a clustering algorithm. This evaluation is performed by using Cluster Validity Indexes such as the Silhouette index or the Rand index (Kim, Lee, & Kang, 2018; Lei et al., 2017). This evaluation process encompasses multiple criteria. As outlined by Theodoridis and Koutroumbas (2008), there are three distinct approaches to cluster validity: external, internal, and relative. The external approach involves assessing the dataset on the basis of its structure, regardless of the clustering algorithm employed. The internal approach involves evaluating the clusters within a partition, using the internal properties of their elements as a basis. The relative approach entails evaluating the clusters of a partition by comparing them with clusters from another partition. Nevertheless, the most common approach often involves the definition of a function with which to assess a cluster by utilising diverse metrics. These metrics encompass proximity metrics, which measure the closeness between elements within clusters, in addition to separation metrics, which quantify the extent to which elements are separated from the other clusters. One of the separation metrics most frequently used is the Jaccard coefficient (Halkidi, Batistakis, & Vazirgiannis, 2001): $Jaccard(C_{xi}, C_{yj}) = \frac{|C_{xi} \cap C_{yj}|}{|C_{xi} \cup C_{yj}|}$. Another example of a separation metric is the Dice coefficient (Dice, 1945): $Dice(C_{xi}, C_{yj}) = \frac{2 * |C_{xi} \cap C_{yj}|}{|C_{xi}| + |C_{yj}|}$.

Before executing a clustering algorithm using a dataset, it is necessary to determine whether the data has meaningful clusters or whether it has a random structure. This assessment process is known as clustering tendency evaluation. The Hopkins statistic (Banerjee & Dave, 2004) can be employed to assess the clustering tendency of a dataset. This statistic compares the nearest-neighbour distribution of randomly selected samples positioned at random within a d-dimensional sampling window (which may not be part of the dataset) with the nearest-neighbour distribution of randomly selected elements from

the dataset itself. When the Hopkins statistic is calculated for an entire dataset, there may be two outcomes: (a) if the individuals from the dataset are randomly placed (i.e. not clustered), the Hopkins statistic is around 0.5, and (b) if the individuals from the dataset form distinct clusters, the Hopkins statistic exceeds 0.5 and approaches 1.0 for highly defined clustered data. When investigating whether a dataset has clustered structures, an ideal scenario is when the Hopkins statistic falls between 0.7 and 0.99. The Hopkins statistic can also assist in determining the optimal value for the hyperparameter k (which represents the desired number of clusters) when using a partitional clustering algorithm. In order to achieve this, it is necessary to create a partition with k clusters, after which the Hopkins statistic is computed separately for each cluster, and the mean of these values is subsequently calculated. In this case, the preferred outcome is a final Hopkins statistic close to 0.5, indicating that the clusters have a random structure and that further division into additional clusters is not advisable. In summary, the Hopkins statistic serves two purposes: (1) that of evaluating the clustering tendency of a dataset and (2) that of identifying the optimal value of the hyperparameter k for a partitional clustering algorithm in a dataset. An implementation of the Hopkins statistic is available at [1].

For a general understanding of the clustering technique, we refer the reader to Saxena et al. (2017). With regard to the use of clustering in the clinical domain, Elbattah and Molloy (2017); Hielscher et al. (2018); Liao, Li, Kianifard, Obi, and Arcona (2016); Silitonga (2018) demonstrate the utilisation of different clustering techniques in clinical scenarios related to the identification of sets of patients of special interest. Moreover, unsupervised ML models are extensively utilised to discover potential phenotypes, which serve as hypotheses for clinical research (Hoeper et al., 2020; Salmanpour et al., 2021; Wang et al., 2020). Additionally, numerous papers employ alternative ML techniques to identify patients with particular characteristics, such as those of Nannings, Abu-Hanna, and Jonge (2008); Stiglic and Kokol (2012); Umek et al. (2009).

In this context, the use of precision medicine terminology has recently appeared. Precision medicine refers to an approach that focuses on designing and refining the pathway for diagnosis, therapeutic intervention, and prognosis. It achieves this by leveraging extensive multidimensional biological datasets that encompass individual variations in genes, function, and environment (Uddin, Wang, & Woodbury-Smith, 2019). Several works demonstrate this process and propose design methodologies with which

[1] https://github.com/antoniolopezmc/A-methodology-based-on-Trace-based-clustering-for-patient-phenotyping

	Clustering	**Trace-based clustering**	**Subgroup discovery**	**Classification**
Learning	Unsupervised	Unsupervised	Supervised	Supervised
Goal	Partition by distance	Subsets by clustering intersection	Subsets by target attribute	Accurate model to assign class
Approach	Descriptive	Descriptive	Descriptive / Predictive	Predictive
Evaluation measure	Rand, Silhouette, Jaccard, Dice, etc.	Jaccard, Dice, etc.	Support, Precision, Binomial Test, WRAcc, Sensitivity, etc.	Precision, Recall, etc.
Outcome	Partition of a dataset	Non disjoint subsets of elements	Model (rules) describing non disjoint subsets	Model for classification

Table 2.1: Comparison of existing ML techniques and our proposal.

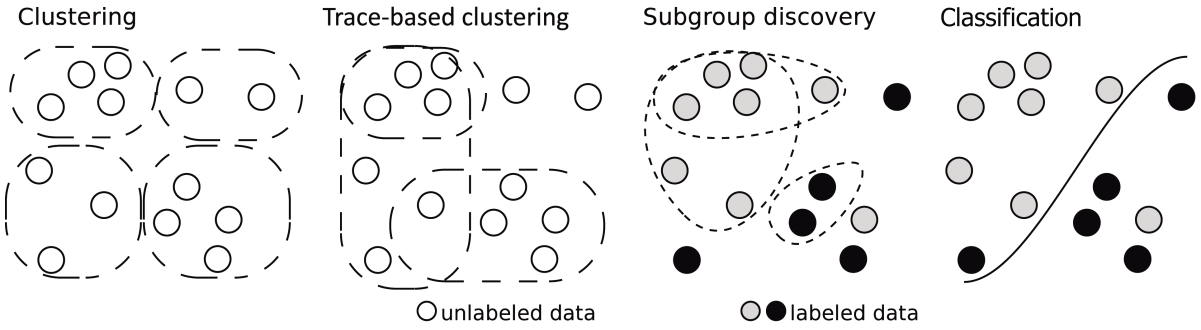


Figure 2.1: Visual comparison of existing ML techniques and our proposal.

to implement this approach, such as those by Chen et al. (2019); Fröhlich et al. (2018); Uddin et al. (2019).

On the one hand, the general approaches used in ML in order to identify groups and classes, including our own approach, are presented in Table 2.1. On the other, Figure 2.1 visually compares these approaches with ours, extending the diagram provided by Ventura and Luna (2018).

2.3 Clinical problem and dataset

This chapter shows the potential of our proposal by analysing the specific problem of identifying and characterising groups of hospitalised patients who have a Gram-positive bacterium, such as MRSA (Methicillin-Resistant Staphylococcus Aureus) or Enterococcus

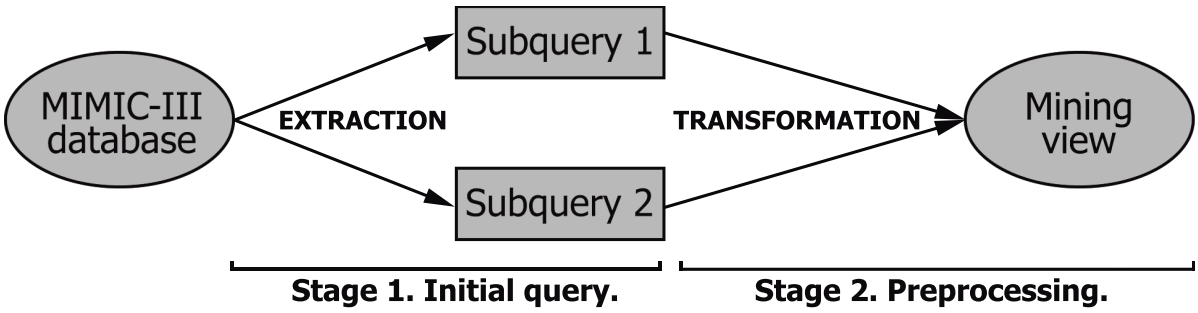


Figure 2.2: Proposed transformation pipeline.

Faecium, that showed resistance to the Vancomycin treatment. This analysis is accomplished by examining susceptibility tests of the patients, which assess the bacteria's sensitivity to one or multiple antibiotics.

One of the objectives of this PhD thesis is the use of open data, although very few public repositories with real clinical data are available for scientific research. Fortunately, the MIMIC-III database (Johnson et al., 2016) provides high-quality information with which to test our experiments. MIMIC-III is a publicly accessible repository comprising health-related information from over 45,000 patients who have received treatment in intensive care units. It encompasses approximately 60,000 admissions that occurred between 2001 and 2012. This extensive database incorporates a diverse range of data sources, including demographic details, laboratory test results, vital sign measurements (such as blood pressure, heart rate, oxygen saturation, etc.), and information regarding the medications administered.

Our experiments involve following the transformation pipeline illustrated in Figure 2.2. The stages required in order to reproduce them are described in Sections 2.3.1 and 2.3.2.

2.3.1 Stage 1: Initial query

The first stage of the transformation pipeline is initiated by executing an initial SQL query in the MIMIC-III database, which has been previously stored in a PostgreSQL database. This query consists of two subqueries, meaning that we will obtain two separate datasets.

The first subquery (accessible at [2]) retrieves a dataset in which each instance represents a strain of a microorganism population obtained from a patient's culture (lab-

[2] https://github.com/antoniolopezmc/A-methodology-based-on-Trace-based-clustering-for-patient-phenotyping

oratory test) during one of their admissions. It does this by using the following tables: PATIENTS (which represents patients treated in intensive care units), ADMISSIONS (which captures patients' hospital admissions), MICROBIOLOGYEVENTS (which provides microbiology details such as cultures and associated sensitivities), ICUSTAYS (which records patients' stays in intensive care units), SERVICES (which represents hospital services to which patients were admitted or transferred), and TRANSFERS (which indicates the patients' physical locations throughout their hospital stays). By applying filters for microorganism name (specifically 'ENTEROCOCCUS FAECIUM' or 'POSITIVE FOR METHICILLIN RESISTANT STAPH AUREUS') and antibiotic name (specifically 'VANCOMYCIN'), this subquery produces a dataset comprising 549 instances and 23 attributes.

It is relevant to note that although the MIMIC-III database contains longitudinal records for patients, we follow the standard M39-A4 defined by the Clinical and Laboratory Standards Institute (CLSI[3]). According to this standard, only the first isolate (microorganism) of a particular species from an individual patient should be considered in order to avoid duplicate data that could introduce bias into the results.

The second subquery (accessible at [4]) retrieves a dataset in which each instance represents a medication administered to a patient during one of their admissions. It does this by employing the INPUTEVENTS_MV table. By applying filters based on the `item_id` (225798, corresponding with 'Vancomycin') and `subject_id` (the same patients retrieved in the first subquery), this subquery produces a dataset with 1934 instances and 5 attributes.

2.3.2 Stage 2: Preprocessing

During the second stage of the transformation pipeline, we combine the two previously acquired datasets using the following process: (1) the removal of duplicate instances and attributes, (2) the removal of empty attributes or those with just one value, (3) the transformation and creation of attributes, (4) by changing attribute types, and (5) the treatment of missing values. Moreover, we do not include identification attributes such as `patient_id` or `admission_id`, among others. These attributes serve only for data traceability and are not necessary for ML algorithms.

[3] `https://clsi.org/`

[4] `https://github.com/antoniolopezmc/A-methodology-based-on-Trace-based-clustering-for-patient-phenotyping`

At this point, our dataset consists of 531 instances and 19 attributes. The next step involves examining the presence of correlations among these attributes. This is achieved by calculating the correlation matrix using the Pearson coefficient. We subsequently eliminate any attribute that has a correlation value equal to or exceeding 0.8, or equal to or less than -0.8 in relation to another attribute.

The mining view obtained has 531 instances and 17 attributes, meaning that we have removed 2 correlated attributes. In this mining view, each instance represents a strain of a microorganism population obtained from a patient's culture (laboratory test) during one of their admissions. Table 2.2 provides an explanation of the characteristics associated with the mining view.

Python 3.7.4 was employed in order to implement the transformation pipeline, along with several libraries: *pandas* (version 1.1.3), *scikit-learn* (version 0.21.3), *matplotlib* (version 3.1.1), and *numpy* (version 1.16.5). These libraries were chosen because they are a reference in the ML field and they are four of those most frequently used and tested by the community.

Although we already have a mining view, it is necessary to transform all the attributes into numeric types since we have applied the clustering technique. If an attribute has only two values, we map them onto 0s and 1s. For attributes with more than two values, we employ the One Hot Encoding technique, which is also included in *scikit-learn*.

At this point, our dataset comprises 531 instances and 69 attributes. Since new attributes are generated during the process of transforming the attributes into numeric types, we apply the same procedure as that employed previously in order to remove correlations.

The resulting numerical mining view consists of 531 instances and 65 attributes after removing four correlated attributes.

Finally, clustering methods require data normalisation, and in this case, we employ the min-max normalisation technique with the numerical mining view. This is implemented using the MinMaxScaler class from the *scikit-learn* library.

2.4 Proposal

The proposal shown in this chapter consists of two parts: (1) a new ML technique denominated as Trace-based clustering that allows the attainment of groups of patients with interesting characteristics, known as patient phenotypes, in unlabelled data, and (2) a new methodology based on the Trace-based clustering technique that makes it possible

Attribute name	Description
patient_gender	The gender of the patient
exitus	Whether or not the patient died
admission_type	Possible values: 'Elective' or 'Emergency'
admission_location	Location of the patient before their arrival
discharge_location	Location of the patient after their discharge
culture_specimen_type_description	Description of the specimen which is tested in the culture for bacterial growth
culture_microorganism_name	Name of the microorganism, if any, that grows in the culture
culture_susceptibility	Susceptibility: sensitive or resistant
service_when_culture	Service in which the patient resided when the culture was carried out
icu_when_culture	Intensive care unit in which the patient resided when the culture was carried out
patient_age	The patient's age
days_since_last_admission	Days between the last admission (if it exists) and this one
days_between_admission_and_first_ICU	Days between the admission to the hospital and the admission to the first ICU
days_between_last_vancomycin_treatment _and_culture__ALL_ADMISSIONS	Days between the last vancomycin treatment (in any admission) and the culture of this admission
duration_of_last_vancomycin _treatment__ALL_ADMISSIONS	Duration of the last vancomycin treatment (in any admission)
number_of_last_vancomycin_ treatments__ALL_ADMISSIONS	Number of Vancomycin treatments administered to the patient (in any admission)
culture_month	The month in which the culture was carried out

Table 2.2: Mining view details.

to choose the best clusters from the clinical point of view. Moreover, this methodology enables the participation of clinicians in the selection process.

In order to address the challenge of identifying and evaluating groups of patients in unlabeled data, our proposal consists of executing, combining and comparing multiple executions of the same clustering algorithm with different initialisations (i.e. varying the k hyperparameter and obtaining different partitions from the original dataset). In this study, we introduce the concept of an "interesting set of elements" for phenotyping analysis within a population. This concept is based on the observation that elements which consistently appear together in various clusters across different partitions (gen-

erated through iterative executions of the clustering algorithm with varying k values) are considered noteworthy. These sets of elements have high stability by remaining together in different clusters. We propose to use stability as the factor that determines the significance of an interesting set of elements.

Our proposal clearly differs from the clustering ensemble technique. Firstly, our proposal utilises the same clustering algorithm, although with varying initialisations, to generate different partitions, while the clustering ensemble technique employs different clustering algorithms during the Generation phase. Secondly, our proposal ensures that the results always consist of clusters contained within a partition initially obtained by the clustering algorithm, while the clustering ensemble technique could obtain a new partition that differs from the original ones during the Consensus phase.

2.4.1 Trace-based clustering

This section presents the underlying principles of the Trace-based clustering technique.

We start by defining $C \in \mathbb{C}$ as a particular dataset within the space of datasets.

Definition 2.1 (Partition (C_x))**.** Given a dataset C and a positive integer value x, C_x is a partition of C if $C_x \subseteq \mathcal{P}(C)$ with $|C_x| = x$ where $C_x = \{C_{x1}, \ldots, C_{xx}\}$, $C_{x1} \cup \ldots \cup C_{xx} = C$ and $C_{x,i} \neq \emptyset$.

Definition 2.2 (Cluster (C_{xi}))**.** Given two positive integer values i and j, the elements of partition C_x are denominated as clusters, meaning that $\forall C_{xi}, C_{xj} \in C_x, C_{xi} \cap C_{xj} = \emptyset$.

Using the given terminology, when $x \neq y$, C_{xi} and C_{yi} are two clusters of different partitions, C_x and C_y, respectively.

We denote $\mathcal{P}(C)_k$ as the set of all possible partitions of C with k clusters.

Definition 2.3 (Clustering Function)**.** Given a dataset C and a positive integer value of k, the clustering function obtains a partition of C with k clusters, expressed as follows:

$$Clustering : C \times \mathbb{Z}^+ \rightarrow \mathcal{P}(C), \tag{2.1}$$

where $Clustering(C,k) \in \mathcal{P}(C)_k$.

In this study, we employ traditional partitional clustering algorithms whose objective is to divide a dataset into k clusters, in which the value of the hyperparameter k is established a-priori. The partition resulting from these clustering algorithms consequently has the following characteristics: (1) none of the clusters are empty, (2) there is no overlap

between their clusters, and (3) each element within the input dataset belongs to only one cluster.

In Section 2.2, we discussed numerous metrics with which to evaluate partitions and their clusters. In this study, we introduce a matching function that serves as a generalised metric for this purpose.

Definition 2.4 (Matching Function (M))**.** Given two clusters, $C_{xi} \in C_x$ and $C_{yj} \in C_y$, the matching function M measures the similarity between these clusters in terms of the elements they contain. Formally:

$$M : C_x \times C_y \rightarrow [0,1] \tag{2.2}$$

This function, in the codomain $[0,1]$, has two properties:

$$M(C_{xi}, C_{yj}) = 1 \iff C_{xi} = C_{yj}. \tag{2.3}$$

$$M(C_{xi}, C_{yj}) = 0 \iff C_{xi} \cap C_{yj} = \emptyset. \tag{2.4}$$

Let us now present the notion of trace. The underlying idea behind trace is to track the elements within a cluster that continue to stay grouped together in the clusters of other partitions.

Definition 2.5 (Trace)**.** Let C be a dataset and $\{C_2, \ldots, C_k\}$ be a set of partitions (as a result of iteratively compiling $Clustering(C, i), i \in \{2, \ldots, k\}$). Given a cluster C_{ki} from the partition C_k, the trace of this cluster is the set of clusters of each partition $(C_2, \ldots, C_{k-1})$ that maximise the matching function M in relation to the cluster C_{ki}.

Definition 2.6 (Trace Function)**.** The trace function calculates the trace of a cluster C_{ki}, given a set of partitions $\{C_2, \ldots, C_{k-1}\}$ and by using a matching function M, as follows:

$$Trace : C_k \times \{\mathcal{P}(C)_2, \ldots, \mathcal{P}(C)_{k-1}\} \times M \rightarrow C_2 \times \ldots \times C_{k-1} \tag{2.5}$$

$$\begin{aligned} &Trace(C_{ki}, \{C_2, \ldots, C_{k-1}\}, M) = \\ &< argmax_{C_{2j} \in C_2} M(C_{2j}, C_{ki}), \ldots, argmax_{C_{k-1j} \in C_{k-1}} M(C_{k-1j}, C_{ki}) > \end{aligned} \tag{2.6}$$

Definition 2.7 (MTraces Function)**.** Given a dataset C, a positive integer value (k) and a matching function M, the $MTraces$ function obtains a matrix of traces that considers

the partitions $C_2,\ldots,C_k$ and computes the corresponding vectors by using the $Trace$ function for each cluster C_{ki} in the C_k partition.

$$MTraces : \mathbb{C} \times k \times M \rightarrow (C_2 \times \ldots \times C_{k-1})^k \tag{2.7}$$

$$MTraces(C,k,M) = \begin{bmatrix} Trace(C_{k1},\{C_2,\ldots,C_{k-1}\},M) & \cdots & Trace(C_{kk},\{C_2,\ldots,C_{k-1}\},M) \end{bmatrix} \tag{2.8}$$

We now present Algorithm 1, which outlines a method with which to implement the $Trace$ function. The algorithm employs a matching function M (defined in equation 2.2) and a set T to store the selected clusters that trace C_{ki} (the input cluster). Note that, although the dataset C is initially divided into $k-1$ partitions $(C_2,\ldots,C_k)$, we consider only $k-2$ partitions, specifically from C_2 to C_{k-1}. This is for two reasons: (1) C_1 is not consider, since it is a partition with only one cluster, signifying that $C_{ki} \subseteq C_{11}$ and $C_1 = C$, and (2) C_{ki} represents a cluster within C_k and, by definition, $C_{ki} \cap C_{kj} = \emptyset$.

We also present a specific method with which to implement the $MTraces$ function, which is described in Algorithm 2.

Guiding example:

The Trace-based clustering technique is illustrated by means of a comprehensive example involving dataset C, a specific matching function M, and a value of $k = 5$. The process is shown in Figure 2.3.

The aim of this example is to acquire the trace for each cluster within the partition C_5. These traces are calculated using the $Trace$ function, as depicted in Table 2.3. We then compute $MTraces(C,5,M) = \mathcal{T}$, which is a 3×5 matrix (a matrix consisting of 3 rows and 5 columns), as shown in Table 2.4. It should be interpreted in a column-wise manner. Each column i of the matrix $\mathcal{T}$ represents the trace of cluster C_{5i} in relation to the partitions C_2, C_3 and C_4.

Algorithm 1 Trace.

Input: C_{ki} { cluster } ; $\{C_2,\ldots,C_{k-1}\}$ { set of partitions } ; M { matching function }
Output: T : vector of selected clusters

```
1:  T := ∅
2:  for w = k − 1...2 do
3:      selected_cluster := C_w1
4:      for j = 1...w do
5:          if M(C_ki, C_wj) > M(C_ki, selected_cluster) then
6:              selected_cluster := C_wj
7:          end if
8:      end for
9:      T_w := selected_cluster
10: end for
11: return T
```

Algorithm 2 MTraces: Matrix of traces.

Input: C { dataset } ; $k \in Z^+$; M { matching function }
Output: $\mathcal{T}$: matrix of selected clusters

```
1:  S := ∅
2:  T := ∅
3:  for i = 2...k do
4:      C_i := Clustering(C, i)
5:      S := S ∪ {C_i}
6:  end for
7:  for i = 1...k do
8:      T_*i := Trace(C_ki, S, M)
9:  end for
10: return T
```

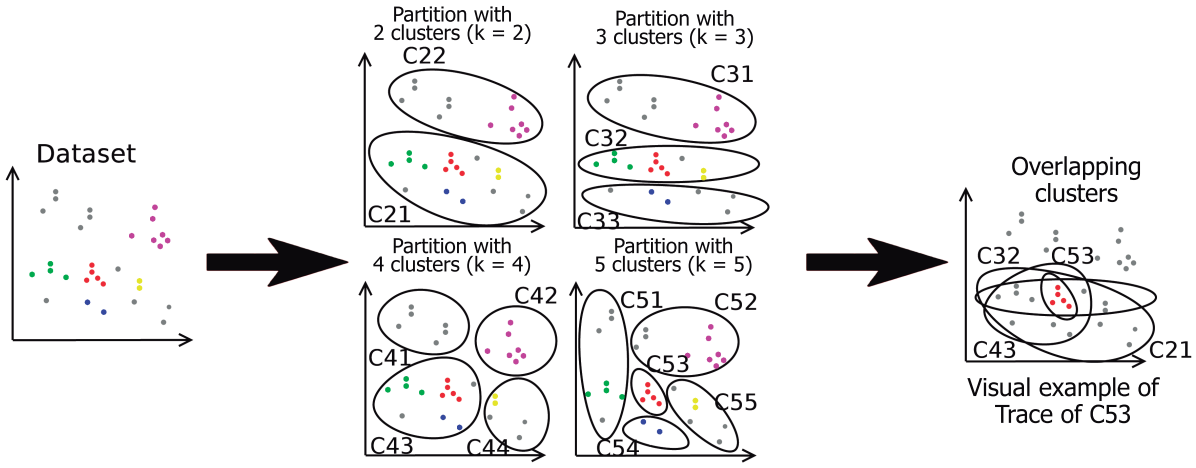


Figure 2.3: Example of the Trace-based clustering approach with $k = 5$.

Cluster	Result
$C_{5,1}$	$Trace(C_{5,1},\{C_2,C_3,C_4\},M) =< C_{2,1},C_{3,2},C_{4,3} >$
$C_{5,2}$	$Trace(C_{5,2},\{C_2,C_3,C_4\},M) =< C_{2,2},C_{3,1},C_{4,2} >$
$C_{5,3}$	$Trace(C_{5,3},\{C_2,C_3,C_4\},M) =< C_{2,1},C_{3,2},C_{4,3} >$
$C_{5,4}$	$Trace(C_{5,4},\{C_2,C_3,C_4\},M) =< C_{2,1},C_{3,3},C_{4,3} >$
$C_{5,5}$	$Trace(C_{5,5},\{C_2,C_3,C_4\},M) =< C_{2,1},C_{3,2},C_{4,4} >$

Table 2.3: Results obtained after applying the $Trace$ function to the 5 clusters from partition C_5.

	1	2	3	4	5
2	$C_{2,1}$	$C_{2,2}$	$C_{2,1}$	$C_{2,1}$	$C_{2,1}$
3	$C_{3,2}$	$C_{3,1}$	$C_{3,2}$	$C_{3,3}$	$C_{3,2}$
4	$C_{4,3}$	$C_{4,2}$	$C_{4,3}$	$C_{4,3}$	$C_{4,4}$

Table 2.4: Matrix $\mathcal{T}$ after applying $MTraces(C,5,M)$.

2.4.2 Methodology

In this chapter, we also propose a methodology based on the Trace-based clustering technique (introduced in Section 2.4.1). The objective of this methodology is to identify patient phenotypes and consists of the following steps:

- Step 1. Extraction and transformation of data and analysis of clustering tendency.
- Step 2. Selection of clustering algorithm and hyperparameters.
- Step 3. Automatic generation of candidate clusters.
- Step 4. Visual support for selection of candidate clusters.
- Step 5. Evaluation by clinical experts.

The first step involves extracting and converting data from clinical sources. This step comprises techniques such as data cleaning and transformation, which encompass tasks such as modifying attributes and handling missing values. These techniques are necessary in order to derive the desired set of attributes, referred to as the mining view, that align with the clinical objectives of the study. The resulting dataset is denoted as C. This mining view must contain only numeric attributes that are normalised.

It is also necessary to evaluate the clustering tendency of the mining view in order to determine the feasibility of applying the technique. In this study, we employ the Hopkins statistic, although alternative methods are also possible.

The second step involves choosing the clustering function (see equation 2.1) and determining the maximum number of expected clusters, known as the k hyperparameter. These decisions are based on the specific clinical problem and the target attributes of the problem studied. It is necessary to estimate k for two reasons: first, if the value is too low, the resulting candidate clusters will be too broad and contain a large number of elements, thus making the task of identification and characterisation more challenging; and second, if the value is too high, the candidate clusters will be too fragmented, with only a few elements in each cluster, leading to increased difficulty in the identification and characterisation process. There are various methods with which to determine the maximum number of expected clusters (referred to as the k hyperparameter), such as the elbow method or the Hopkins statistic.

In the third step, our Trace-based clustering method is employed in order to identify all the candidate clusters that could be selected as interesting sets of patients. To this end, we compute $MTraces(C,k,M) = \mathcal{T}$. Note that these candidate clusters belong exclusively to partition C_k (more precisely, from C_{k1} to C_{kk}).

In the fourth step, clinicians participate actively in the final selection of candidate clusters by considering all the clusters created in the previous step. This is a semi-automatic process in which visualisation techniques aid the clinicians' decision-making process by presenting the trace of the candidate clusters in a suitable manner.

More specifically, we generate a numerical representation of the matrix $\mathcal{T}$, which is the matrix of traces obtained using the $MTraces$ function. Within this matrix, an element $\mathcal{T}xi$ in the column i and in the row x of the matrix $\mathcal{T}$ is the most similar cluster (according to a matching function M) to C_{ki} when C is partitioned with x clusters. Each column i in $\mathcal{T}$ represents the trace of cluster C_{ki} in relation to the previous partitions, i.e. $C_2,\ldots,C_{k-1}$. The matrix $\mathcal{J}$, in which $\mathcal{J}_{xi} = M(C_{ki},\mathcal{T}_{xi})$, contains the values of the matching function M between the candidate clusters and the most similar clusters from previous partitions.

After computing the matrix $\mathcal{J}$, we generate a visual representation in order to facilitate the selection of candidate clusters. There are various techniques available for the visualisation of matrices, but the use of a heat-map is effective as regards identifying clusters of values, especially when there are significant variations between adjacent values (represented with colours). The usefulness of a heat-map for identifying data partitions was demonstrated by Mühlbacher and Piringer (2013). Figure 2.4 illustrates the creation of the visual support.

Despite having a visual support for the final selection of the candidate clusters (that

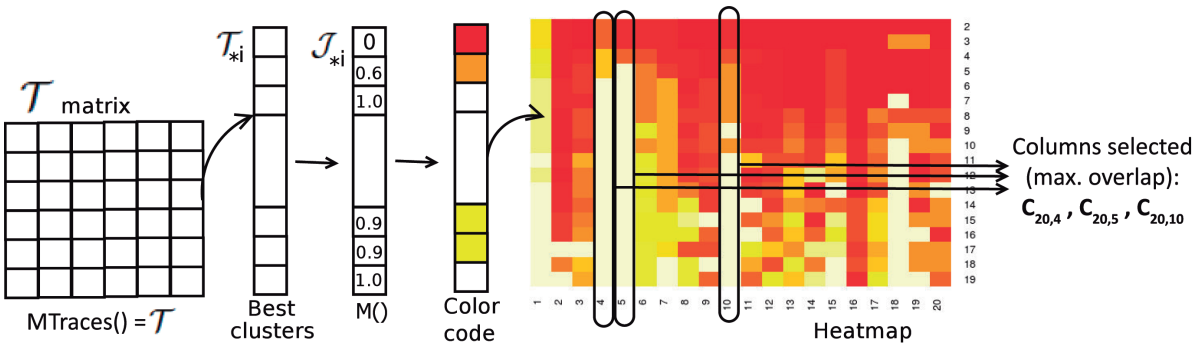


Figure 2.4: Visual support creation process.

is, a visual representation of the matrix $\mathcal{T}$), using a high value for the hyperparameter k could result in a large number of candidate clusters. In order to address this issue, it is possible to establish a ranking system that prioritises candidate clusters for analysis by the expert. Statistical metrics such as mean or median can be employed for this purpose. A post-filtering process can consequently be implemented by computing these statistical metrics within the columns of the matrix $\mathcal{J}$ and selecting only those candidate clusters that surpass a specified filtering threshold. This approach allows the user to control the number of final candidate clusters that require analysis.

The last step is the validation of the candidate clusters chosen. The patients belonging to these chosen clusters will be carefully assessed by doctors, who will extensively review their individual records and characteristics, known as phenotypes.

Please recall that the candidate clusters selected belong to partition C_k, meaning that they are a subset of the clusters from C_{k1} to C_{kk}. For instance, Figure 2.4 illustrates the selection of three candidate clusters from partition C_{20}, which has a higher value of k in this case. Moreover, our methodology highlights, in each column i of the matrix $\mathcal{T}$, the clusters from previous partitions that maximise the number of overlapping elements with cluster C_{ki} based on a matching function M (i.e. the trace of cluster C_{ki}). This represents one of the enhancements of our method when compared to conventional clustering approaches: patient sets are evaluated by measuring the extent of overlapping elements within clusters from previous partitions.

2.5 Experiments and results

The purpose of these experiments is to evaluate the applicability of the Trace-based clustering technique and the 5-step methodology with which to identify groups of patients

who share a common response to antibiotics. More precisely, we focus on the clinical problem defined in Section 2.3 and use the data described in that section.

The research conducted in this study follows the 5-step methodology. In these steps, we examine the configuration of various hyperparameters using different fixed random seeds. This section not only details and describes the experiments performed in this research but also presents the results obtained in each step of our methodology.

2.5.1 Step 1: Extraction and transformation of data and analysis of clustering tendency

The steps outlined in Section 2.3 were followed in order to proceed with the stages of the transformation pipeline. Upon completing this process, we acquired a dataset consisting entirely of numeric attributes. There were no correlations among these attributes and they were normalised, resulting in the creation of a numerical mining view. The numerical mining view encompassed 531 instances and 65 attributes.

After completing the preparation of the mining view, we employed the Hopkins statistic to assess the clustering tendency. The objective was to confirm the appropriateness of the clustering methods and to determine the presence of a non-random structure in the data. By calculating the Hopkins statistic using our numerical mining view, we derived a value of 0.82. Upon interpreting this statistic, it was discovered to fall within the range of 0.7 to 0.99, indicating that the elements of the numerical mining view were statistically aggregated into distinct groups or clusters (i.e. well-defined clustered data).

2.5.2 Step 2: Selection of clustering algorithm and hyperparameters

In this section, we show how experiments were conducted by utilising the K-Means algorithm as a clustering function (equation 2.1). We opted for this algorithm because of its substantiated effectiveness in clinical literature and its ability to enhance the comprehensibility of the process for the clinical expert. The experiments were conducted using the implementation provided in *scikit-learn*.

Determining the appropriate value of hyperparameter k is of significant importance in clustering techniques. In order to identify the most suitable value, we examined two widely recognised techniques for the estimation of k from existing research: the elbow method and the Hopkins statistic.

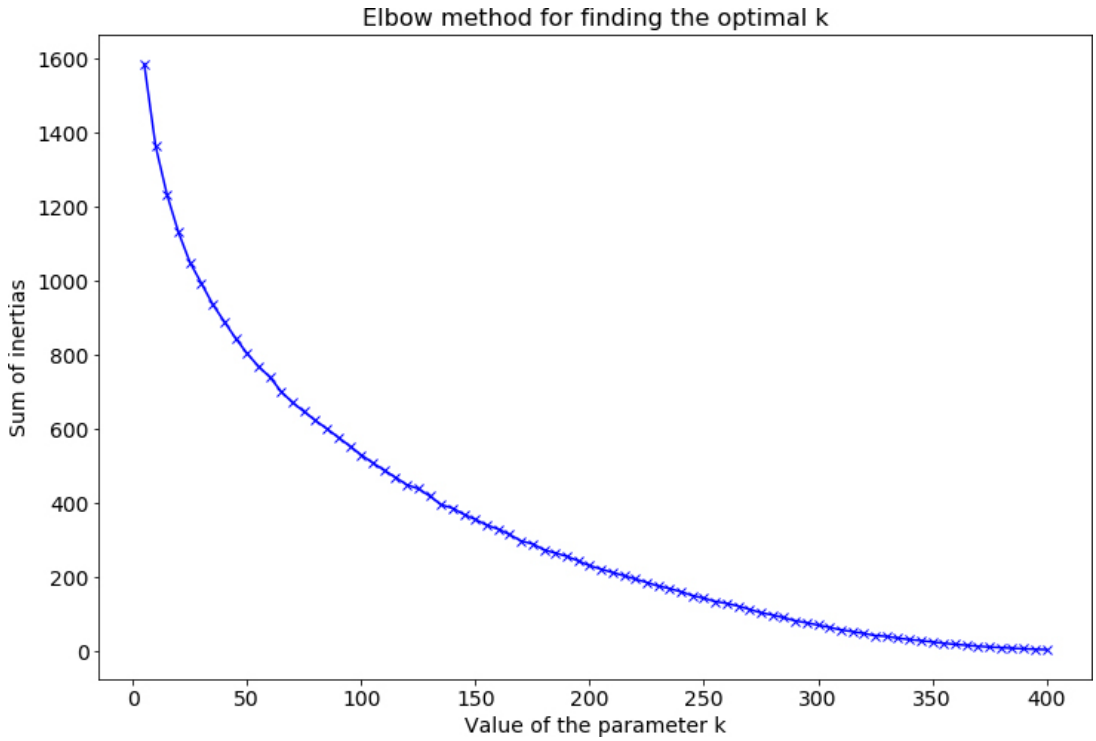


Figure 2.5: Step 2: estimating the best k using the elbow method (up to $k = 400$).

k	Hopkins statistic
150	0.556
105	0.571
115	0.572
110	0.574
175	0.578

Table 2.5: Step 2: the five best values of k using the Hopkins statistic.

The outcomes achieved after employing the elbow method (see Figure 2.5) did not yield definitive results, necessitating the use of alternative methods to estimate the hyperparameter k. Note that when using the elbow method we display the total inertia of the clusters in each partition k (that is, the sum of the inertia of each cluster), in which inertia refers to the sum of squared distances between the elements within a cluster and its centroid. Furthermore, we use the maximum value of 400 for an adequate visualisation of the curve.

In our experiments, it would appear that the Hopkins statistic provides valuable information. Figure 2.6 displays all the values of the hyperparameter k considered and their respective Hopkins statistic values. Furthermore, Table 2.5 presents the 5 best

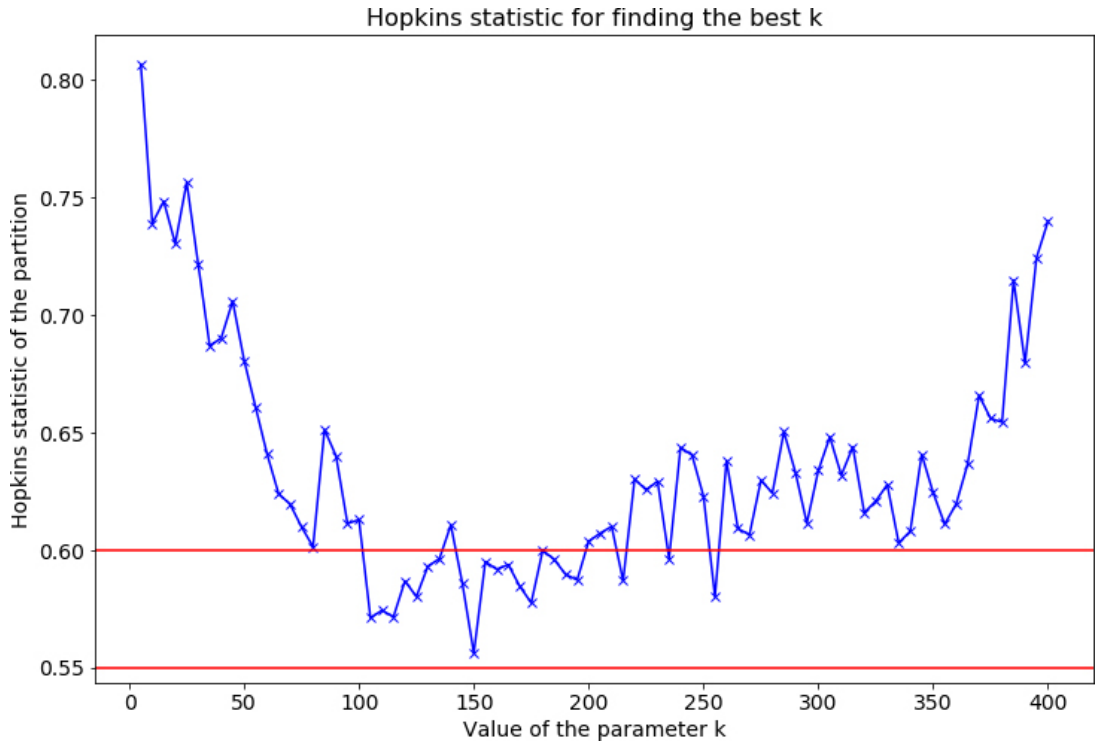


Figure 2.6: Step 2: estimating the best k using the Hopkins statistic (from $k = 1$ to 400).

values of the hyperparameter k (whose Hopkins statistic value was closer to 0.5).

In our experiments, we employed the top three values of the hyperparameter k based on the Hopkins statistic:

- Experiment 1: $k = 150$.
- Experiment 2: $k = 105$.
- Experiment 3: $k = 115$.

This means that a total of 370 candidate clusters were automatically identified and subsequently analysed in the three experiments.

2.5.3 Step 3: Automatic generation of candidate clusters

One essential factor in this step involves adopting a particular matching function denoted as M (equation 2.2). In our study, we used the Dice coefficient. By definition and based on its properties, the Dice function consistently yields values ranging from 0 to 1, inclusive.

In the first experiment, the $MTraces$ algorithm was executed, carrying out the following actions: (1) the K-Means algorithm was utilised across a range of values from

$k = 2$ to $k = 150$, resulting in a total of 149 partitions ($\{C_2, C_3, \ldots, C_{149}, C_{150}\}$), and (2) the $Trace$ algorithm was utilised on $C_{150,1}, C_{150,2}, \ldots, C_{150,149}, C_{150,150}$ and all the partitions retrieved in the previous phase (excluding C_{150}). As a result, we obtained 150 candidate clusters corresponding to all the clusters in partition C_{150}. We then obtained a matrix $\mathcal{T}$ of size 148×150 that corresponded to this particular experiment.

In the second experiment, the $MTraces$ algorithm was executed, carrying out the following actions: (1) the K-Means algorithm was utilised across a range of values from $k = 2$ to $k = 105$, resulting in a total of 104 partitions ($\{C_2, C_3, \ldots, C_{104}, C_{105}\}$), and (2) the $Trace$ algorithm was utilised on $C_{105,1}, C_{105,2}, \ldots, C_{105,104}, C_{105,105}$ and all the partitions retrieved in the previous phase (excluding C_{105}). As a result, we obtained 105 candidate clusters corresponding to all the clusters in partition C_{105}. We subsequently obtained a matrix $\mathcal{T}$ of size 103×105 that corresponded to this particular experiment.

In the third experiment, the $MTraces$ algorithm was executed, carrying out the following actions: (1) the K-Means algorithm was utilised across a range of values from $k = 2$ to $k = 115$, resulting in a total of 114 partitions ($\{C_2, C_3, \ldots, C_{114}, C_{115}\}$), and (2) the $Trace$ algorithm was utilised on $C_{115,1}, C_{115,2}, \ldots, C_{115,114}, C_{115,115}$ and all the partitions retrieved in the previous phase (excluding C_{115}). As a result, we obtained 115 candidate clusters corresponding to all the clusters in partition C_{115}. We subsequently obtained a matrix $\mathcal{T}$ of size 113×115 that corresponded to this particular experiment.

2.5.4 Step 4: Visual support for selection of candidate clusters

We calculated the subsequent $\mathcal{J}$ matrices for the three experiments, which include the matching function M values (specifically, the Dice coefficient) between the candidate clusters and the most similar clusters from each previous partition:

- Experiment 1: a $\mathcal{J}$ matrix of size 148×150. The visualisation of this matrix in the form of a heat-map, as depicted in Figure 2.7, provided a concise summary of experiment 1.

- Experiment 2: a $\mathcal{J}$ matrix of size 103×105. The visualisation of this matrix in the form of a heat-map, as depicted in Figure 2.8, provided a concise summary of experiment 2.

- Experiment 3: a $\mathcal{J}$ matrix of size 113×115. The visualisation of this matrix in the form of a heat-map, as depicted in Figure 2.9, provided a concise summary of experiment 3.

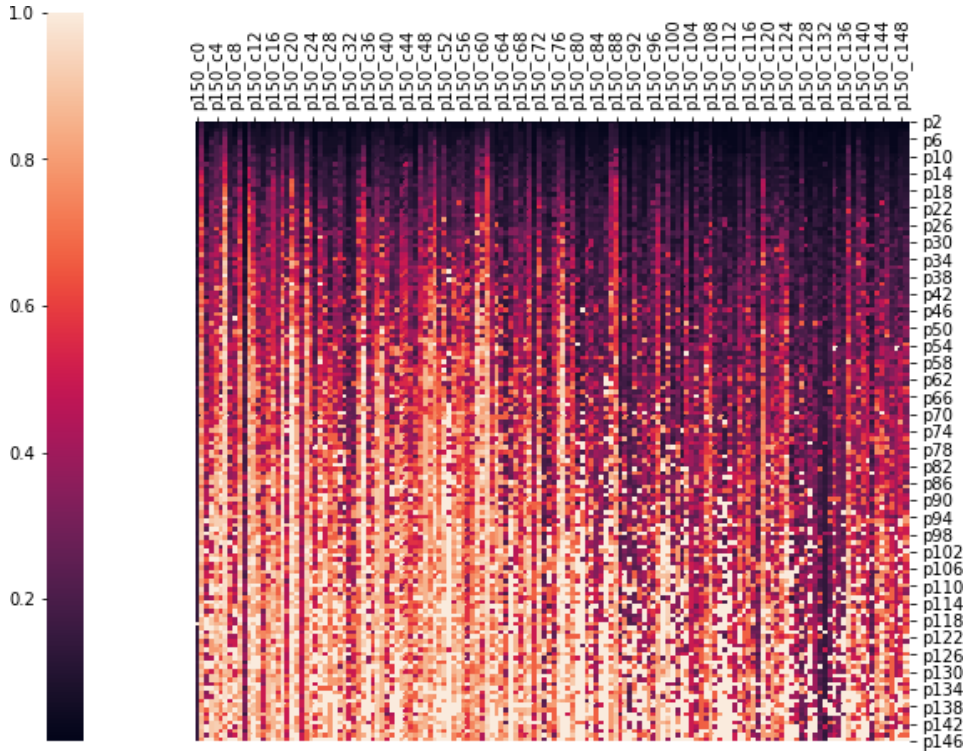


Figure 2.7: Step 4: Heat-map ($\mathcal{J}$ matrix visualisation) for experiment 1 ($k = 150$).

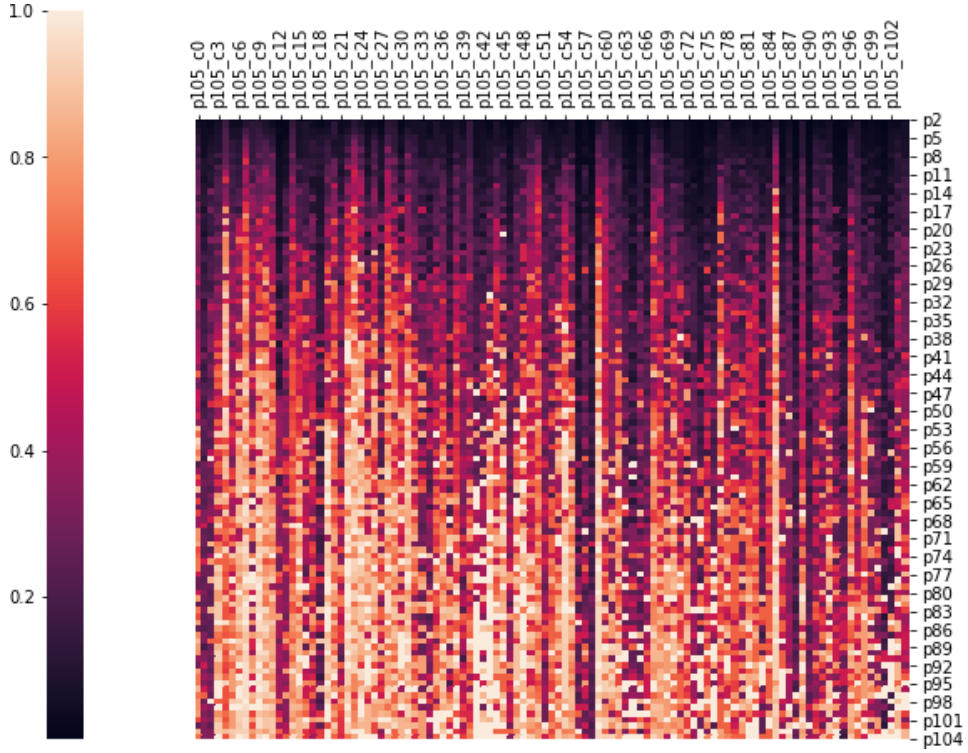


Figure 2.8: Step 4: Heat-map ($\mathcal{J}$ matrix visualisation) for experiment 2 ($k = 105$).

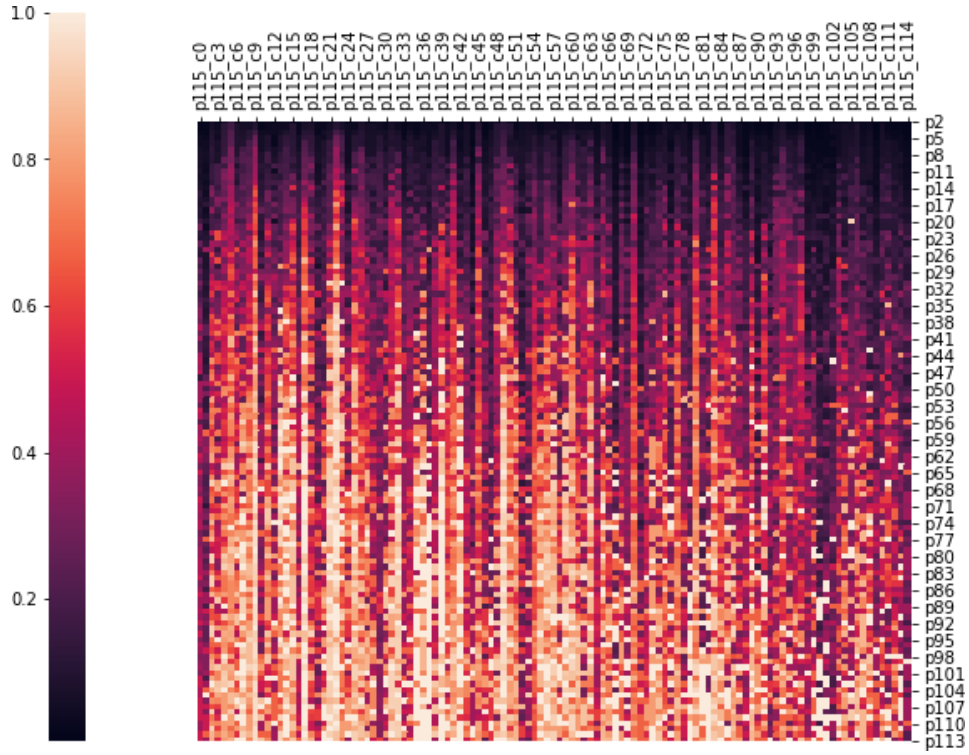


Figure 2.9: Step 4: Heat-map ($\mathcal{J}$ matrix visualisation) for experiment 3 ($k = 115$).

The number of candidate clusters, amounting to 370, is relatively high for manual intervention. Indeed, many of these clusters are of low quality and lack relevance. In order to reduce the number of candidate clusters to be explored, we executed a post-filtering procedure so as delete the candidate clusters whose mean M value was below a threshold, which could be adjusted by the user. The mean M value is computed as the mean of the values from the corresponding column from the $\mathcal{J}$ matrix. Figures 2.10, 2.11 and 2.12 show the number of candidate clusters selected when we varied the filtering threshold.

In experiment 1, we focused on clusters (columns i of matrix $\mathcal{J}$) in which the mean of each column i was 0.75 or higher, resulting in 8 candidate clusters. In experiment 2, we focused on clusters in which the mean of each column i was 0.70 or higher, giving us 6 candidate clusters. In experiment 3, we focused on clusters in which the mean of each column i was 0.70 or higher, resulting in 5 candidate clusters. This filtering process eliminated approximately 95% of the initial candidate clusters (from 370 to 19), thus saving the expert the task of having to manually study numerous irrelevant clusters. The heatmaps representing the matrices $\mathcal{J}$ of only these filtered candidate clusters can be seen in Figures 2.13, 2.14, and 2.15.

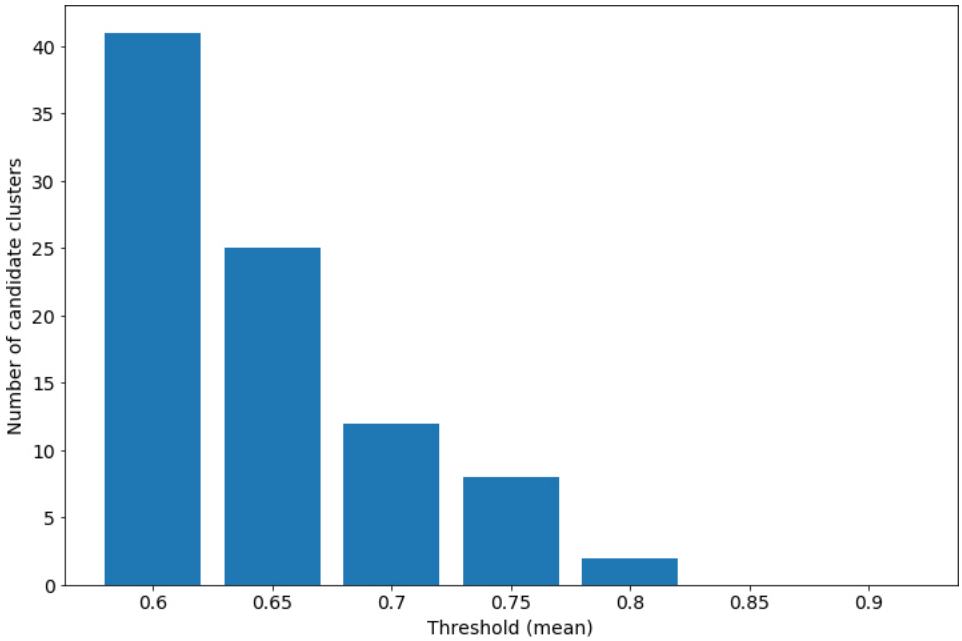


Figure 2.10: Step 4: number of candidate clusters with respect to the filtering threshold for experiment 1 ($k = 150$).

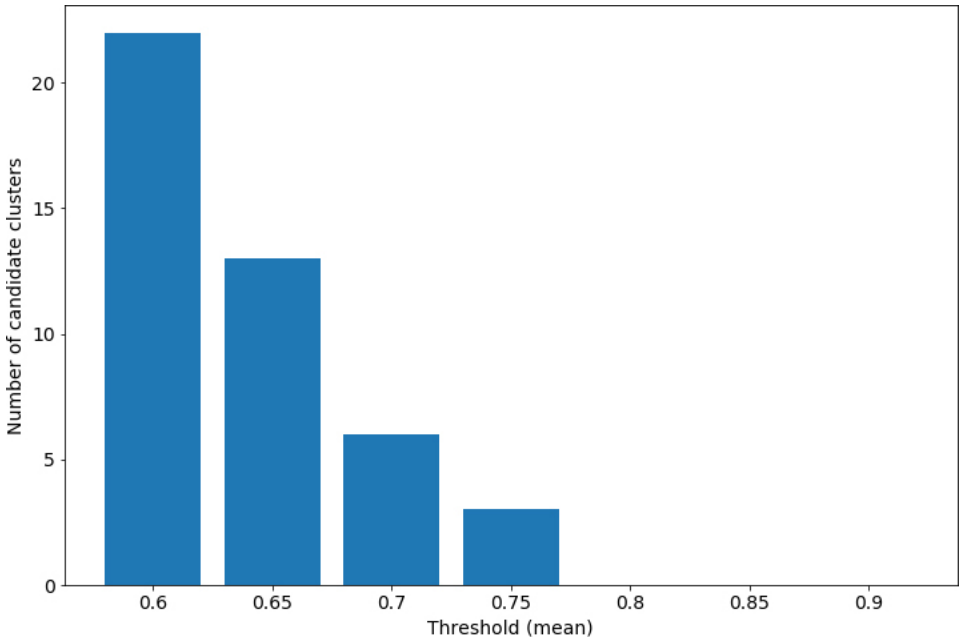


Figure 2.11: Step 4: number of candidate clusters with respect to the filtering threshold for experiment 2 ($k = 105$).

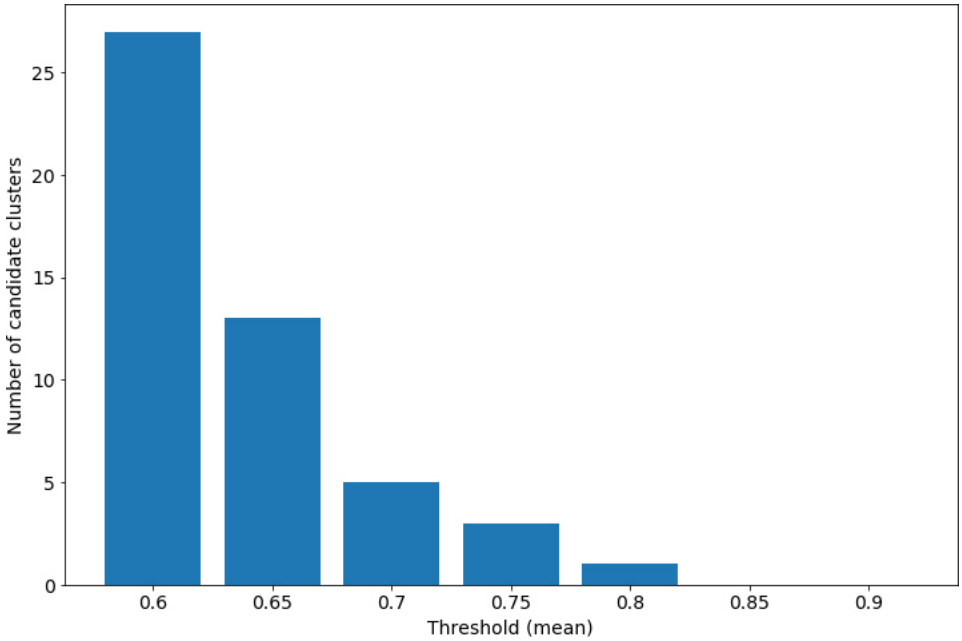


Figure 2.12: Step 4: number of candidate clusters with respect to the filtering threshold for experiment 3 ($k = 115$).

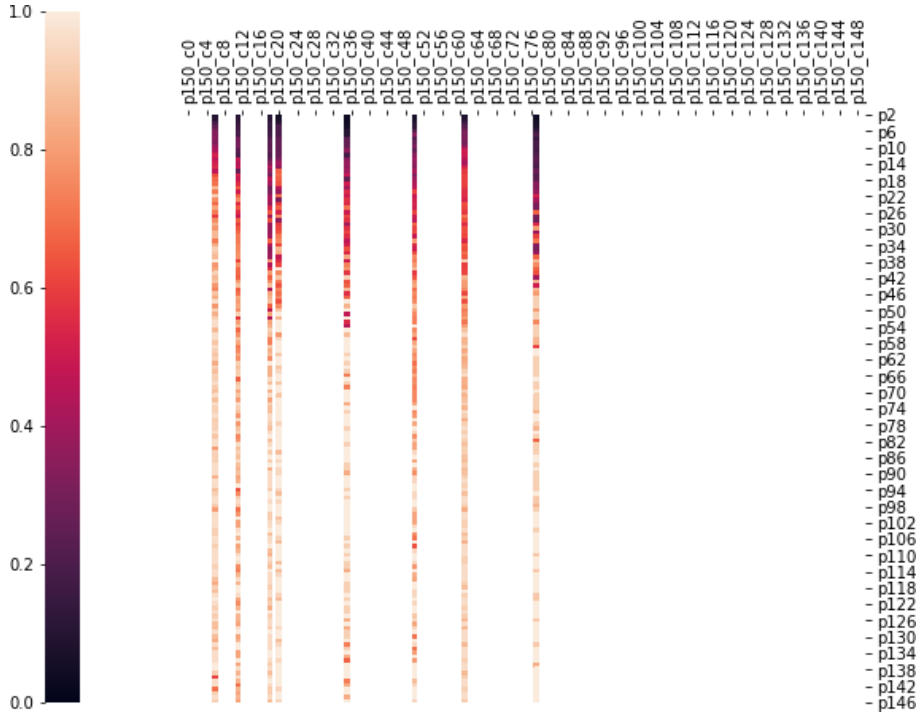


Figure 2.13: Step 4: post-filtering Heat-map ($\mathcal{J}$ matrix visualisation) (approx. 95% reduction) for experiment 1 ($k = 150$).

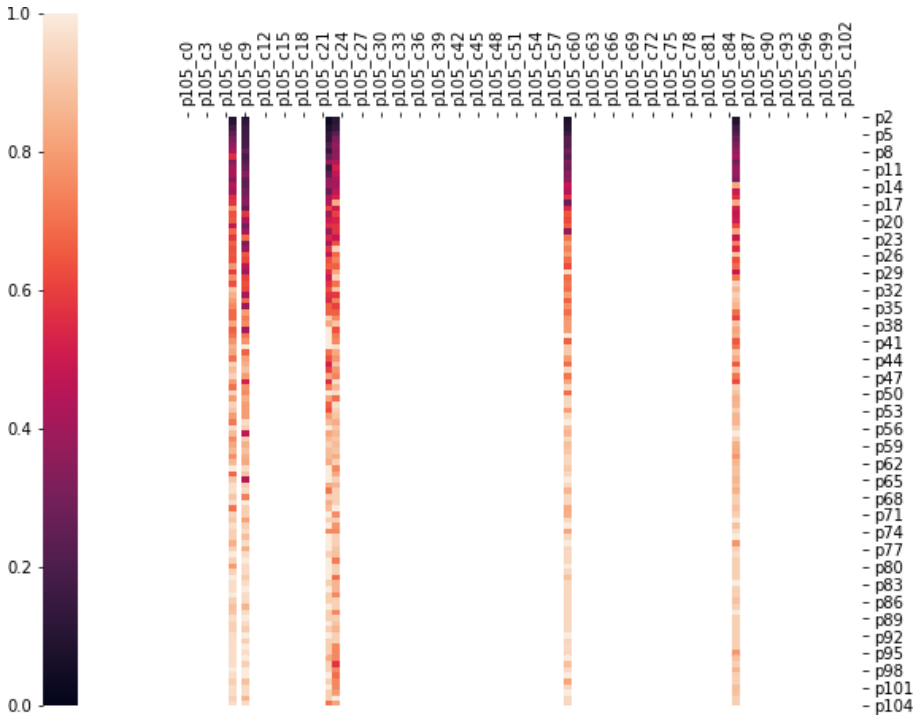


Figure 2.14: Step 4: post-filtering Heat-map ($\mathcal{J}$ matrix visualisation) (approx. 95% reduction) for experiment 2 ($k = 105$).

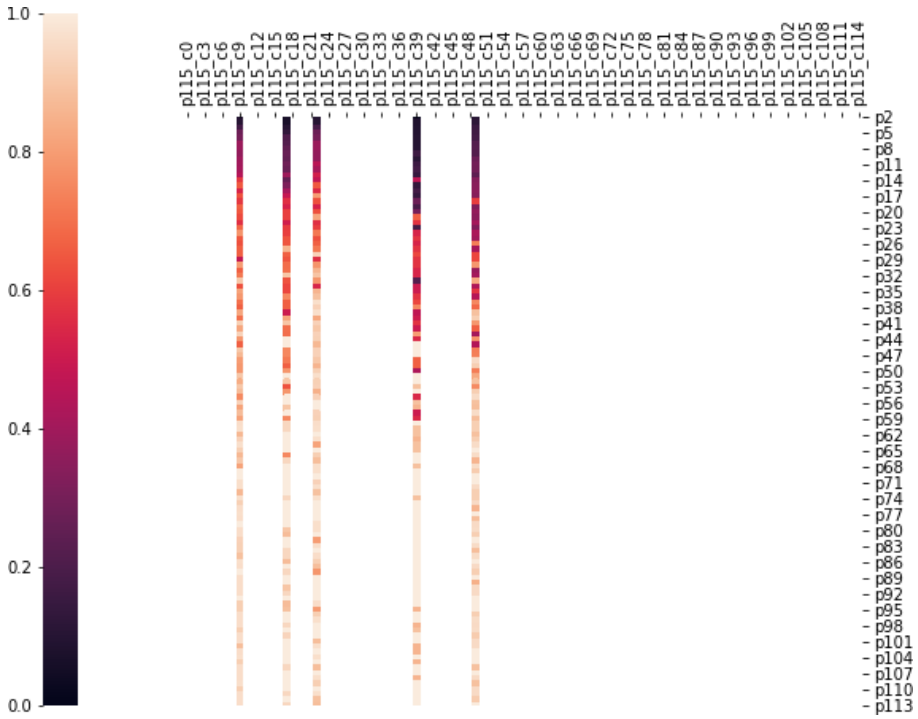


Figure 2.15: Step 4: post-filtering Heat-map ($\mathcal{J}$ matrix visualisation) (approx. 95% reduction) for experiment 3 ($k = 115$).

Cluster name	**Cluster size**	**Mean of the column in matrix $\mathcal{J}$**	**Median of the column in matrix $\mathcal{J}$**	**Number of resistant elements**
Experiment 1 (k = 150)				
p150_c6	13	0.842	0.923	12 (*92%*)
p150_c11	12	0.766	0.818	9 (*75%*)
p150_c18	11	0.761	0.880	11 (*100%*)
p150_c20	9	0.829	0.947	9 (*100%*)
p150_c35	7	0.790	0.923	3 (*43%*)
p150_c50	9	0.759	0.818	8 (*89%*)
p150_c61	12	0.797	0.880	11 (*92%*)
p150_c77	6	0.781	0.923	6 (*100%*)
Experiment 2 (k = 105)				
p105_c7	14	0.771	0.846	13 (*93%*)
p105_c9	12	0.704	0.815	12 (*100%*)
p105_c22	7	0.725	0.833	6 (*86%*)
p105_c23	15	0.741	0.786	14 (*93%*)
p105_c59	10	0.773	0.889	10 (*100%*)
p105_c85	12	0.767	0.857	10 (*83%*)
Experiment 3 (k = 115)				
p115_c9	14	0.797	0.889	13 (*93%*)
p115_c17	9	0.781	0.900	9 (*100%*)
p115_c22	14	0.828	0.929	12 (*86%*)
p115_c39	4	0.719	0.857	0 (*0%*)
p115_c49	12	0.742	0.889	12 (*100%*)

Table 2.6: Step 4: results of the three experiments with the Trace-based clustering technique.

The filtering threshold for each experiment (the mean) was determined empirically. This threshold signifies that, intuitively speaking, the elements in the clusters obtained (C_{ki}) are considered stable and remain grouped throughout the other partitions. Additionally, please recall that users have the flexibility to adjust the filtering threshold on the basis of the number of clusters that they wish to validate, thus allowing them to increase or decrease the threshold accordingly.

Table 2.6 displays the 19 candidate clusters obtained in all the experiments after incorporating visual support and employing the post-filtering procedure. Moreover, for each candidate cluster, we present the count of strains in a bacterial population that are resistant to Vancomycin.

During the execution of the step 4 in our methodology, it is uncertain whether the candidate clusters obtained will be related to Vancomycin resistance. This uncertainty

True class \ Predicted class	a	b	c	d	e	f	g	h	i
a. other_cluster	451	0	0	0	0	0	0	1	0
b. p150_c6	3	8	0	0	0	0	0	2	0
c. p150_c11	8	0	4	0	0	0	0	0	0
d. p150_c18	3	0	0	8	0	0	0	0	0
e. p150_c20	3	0	0	0	6	0	0	0	0
f. p150_c35	3	0	0	0	0	4	0	0	0
g. p150_c50	3	0	0	0	0	0	6	0	0
h. p150_c61	2	0	0	0	0	0	0	10	0
i. p150_c77	1	0	0	0	0	0	0	0	5

Table 2.7: Step 5: evaluation of clusters using the confusion matrix of the Random Forest and stratified 10-fold Cross Validation for experiment 1.

True class \ Predicted class	a	b	c	d	e	f	g
a. other_cluster	459	2	0	0	0	0	0
b. p105_c7	3	11	0	0	0	0	0
c. p105_c9	4	0	8	0	0	0	0
d. p105_c22	2	0	0	5	0	0	0
e. p105_c23	3	0	0	0	12	0	0
f. p105_c59	4	0	0	0	0	6	0
g. p105_c85	4	0	0	0	0	0	8

Table 2.8: Step 5: evaluation of clusters using the confusion matrix of the Random Forest and stratified 10-fold Cross Validation for experiment 2.

will be addressed in the subsequent step of the methodology, in which these candidate clusters will be thoroughly characterised and explored. If any of these clusters contain resistant individuals, they will undergo a detailed analysis in order to identify key characteristics associated with these resistances. Nevertheless, the stability property of the interesting sets of elements makes it possible to state that the presence of resistant individuals in the candidate clusters would not be attributed to mere chance, but would rather suggest genuine significance.

2.5.5 Step 5: Evaluation by clinical experts

In our methodology, once the candidate clusters have been obtained, we propose that clinicians conduct a thorough review of these clusters. They should carefully examine the personal records associated with the patients and their characteristics (phenotypes)

		Predicted class					
		a	**b**	**c**	**d**	**e**	**f**
True class	**a. other_cluster**	478	0	0	0	0	0
	b. p115_c9	3	11	0	0	0	0
	c. p115_c17	3	0	6	0	0	0
	d. p115_c22	3	2	0	9	0	0
	e. p115_c39	2	0	0	0	2	0
	f. p115_c49	6	0	0	0	0	6

Table 2.9: Step 5: evaluation of clusters using the confusion matrix of the Random Forest and stratified 10-fold Cross Validation for experiment 3.

in order to assess the clinical significance of the patient sets extracted. However, it is necessary to acknowledge the limitations of our experiments. The dataset used originates from a public medical database, and the patients' personal records are not, therefore, available. This lack of access makes it challenging for clinical experts to perform manual validation if they are not familiar with the specific patient context represented in the clusters. In order to address this issue, we have employed a classification-based cluster evaluation process with which to estimate the quality of the candidate clusters.

In each experiment, we labelled each row (element) in the dataset in the following manner: (1) if the element belonged to a candidate cluster that was eventually selected, we assigned the label as the name of that candidate cluster, and (2) if the element did not belong to any of the candidate clusters eventually selected, we labelled it as "other_cluster". The purpose of this labelling approach was to verify the potential to distinguish the elements in the candidate clusters from the remaining population.

The RandomForest algorithm was employed for classification using the previously specified label as the class. The implementation provided by *scikit-learn* was employed with its default parameters [5], and only two changes were made: `n_estimators=100` and `random_state=50`. We additionally employed stratified 10-fold Cross-Validation using the StratifiedKFold class from *scikit-learn*. The confusion matrices for the three experiments are presented in Tables 2.7, 2.8, and 2.9. Each confusion matrix is calculated as the sum of the 10 confusion matrices obtained, one for each fold. Furthermore, experiment 1 yielded a validation accuracy of 0.95, experiment 2 resulted in a validation accuracy of 0.96, and experiment 3 achieved a validation accuracy of 0.96.

It is necessary to highlight that RandomForest was used since it is a representative technique from the state of the art. However, evaluating and comparing the different

[5] `https://scikit-learn.org/0.21/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

classification techniques from literature are not objectives of this work.

With regard to the confusion matrices, the following observations can be made: (1) only a small number of elements from the candidate clusters were misclassified in the "other_cluster" category, and (2) only 4 elements from the candidate clusters were misclassified within another candidate cluster (2 in experiment 1 and 2 in experiment 3). A more detailed explanation of the confusion matrices is provided as follows. In experiment 1 (Table 2.7), two elements from the candidate cluster "p150_c6" (letter b) were misclassified in the candidate cluster "p150_c61" (letter h). The remaining misclassified elements either belonged to "other_cluster" or were originally from "other_cluster". In experiment 2 (Table 2.8), no elements from a candidate cluster were misclassified in another candidate cluster. The remaining misclassified elements either belonged to "other_cluster" or were originally from "other_cluster". In experiment 3 (Table 2.9), two elements from the candidate cluster "p115_c22" (letter d) were misclassified in the candidate cluster "p115_c9" (letter b). The remaining misclassified elements either belonged to the "other_cluster" or were originally from the "other_cluster".

In order to ensure the consistency of our findings, we conducted a second classification-based cluster evaluation. This validation method employed the same classifier setup described earlier, and the following classes were used: (1) a unified class encompassing all the elements within the candidate clusters (referred to as "candidate_cluster") and (2) a unified class encompassing all the elements not present in the candidate clusters (referred to as "other_cluster"). Notably, during this evaluation, numerous elements from the candidate clusters were erroneously classified in the "other_cluster" category.

In the second validation, in which three experiments were conducted with varying values of k, we achieved accuracies of 0.90, 0.92 and 0.95, respectively. In comparison, the first validation yielded accuracies of 0.95, 0.96 and 0.96, respectively. These results make it possible to deduce that when employing the same class for all elements within our candidate clusters, distinguishing them from other elements becomes more challenging. This outcome supports the notion that our candidate clusters are separable and easily distinguishable from each other.

Finally, the importance of the features in the first classification process (i.e. with one label for each candidate cluster and another label for "other_cluster") is illustrated in Figures 2.16, 2.17, and 2.18. This was done by employing the permutation importance technique through the use of the `permutation_importance` method [6].

[6] `https://scikit-learn.org/stable/auto_examples/inspection/plot_permutation_importance.html` (implemented in version 0.23.2 of *scikit-learn*)

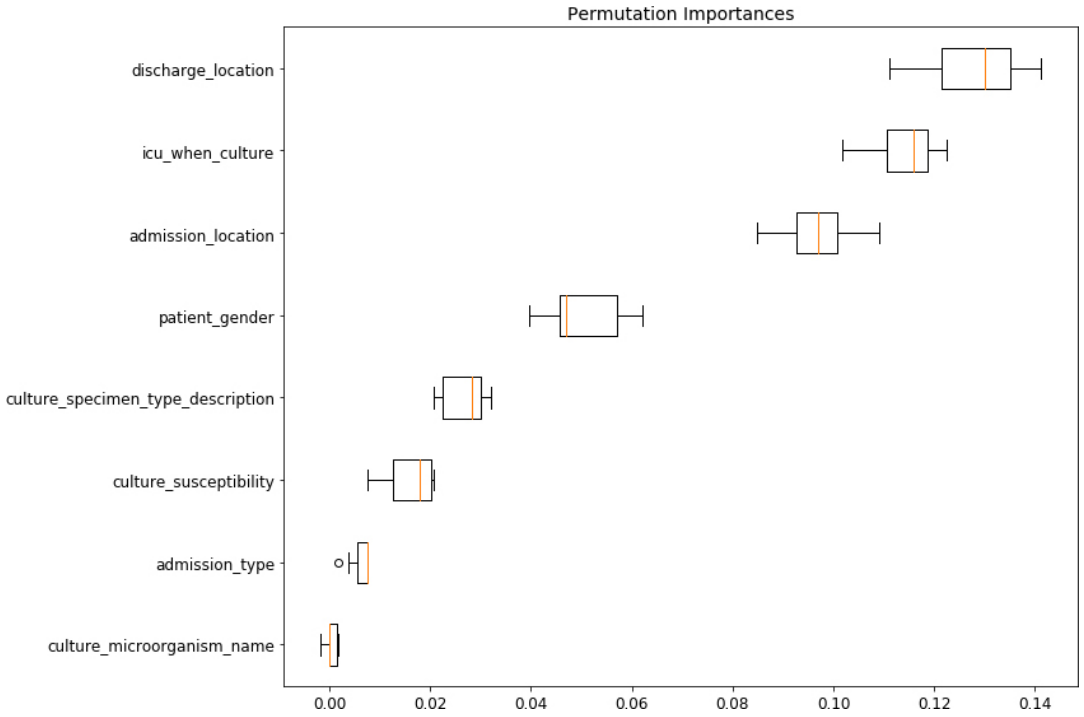


Figure 2.16: Step 5: importance of the features when using the permutation importance technique for experiment 1 ($k = 150$).

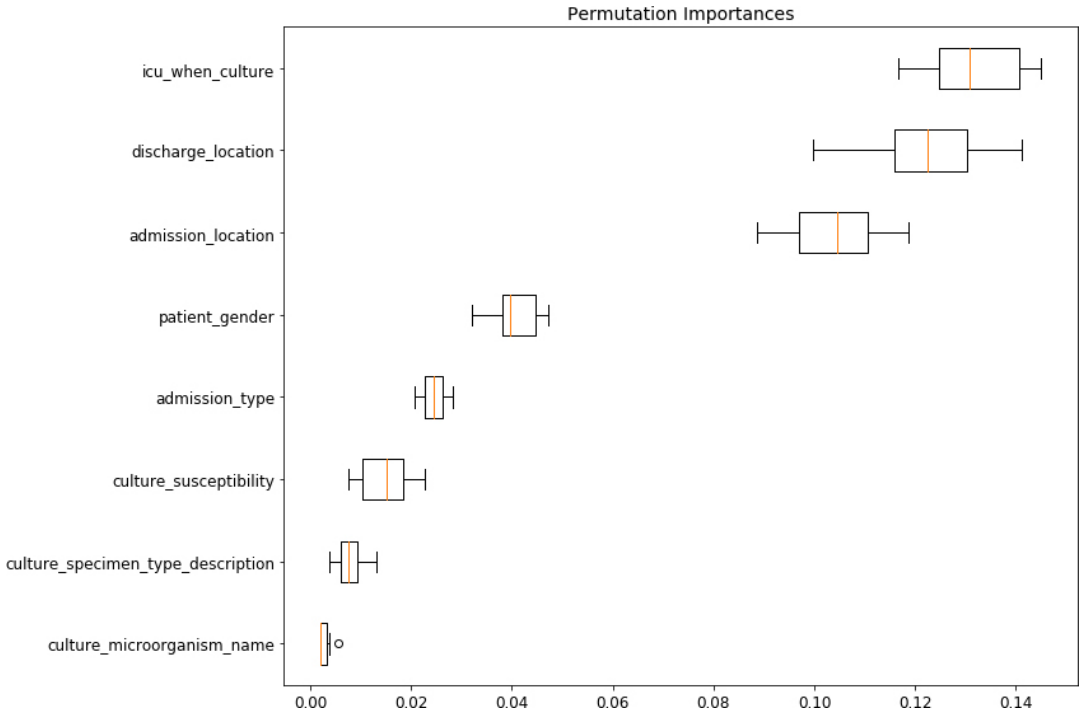


Figure 2.17: Step 5: importance of the features when using the permutation importance technique for experiment 2 ($k = 105$).

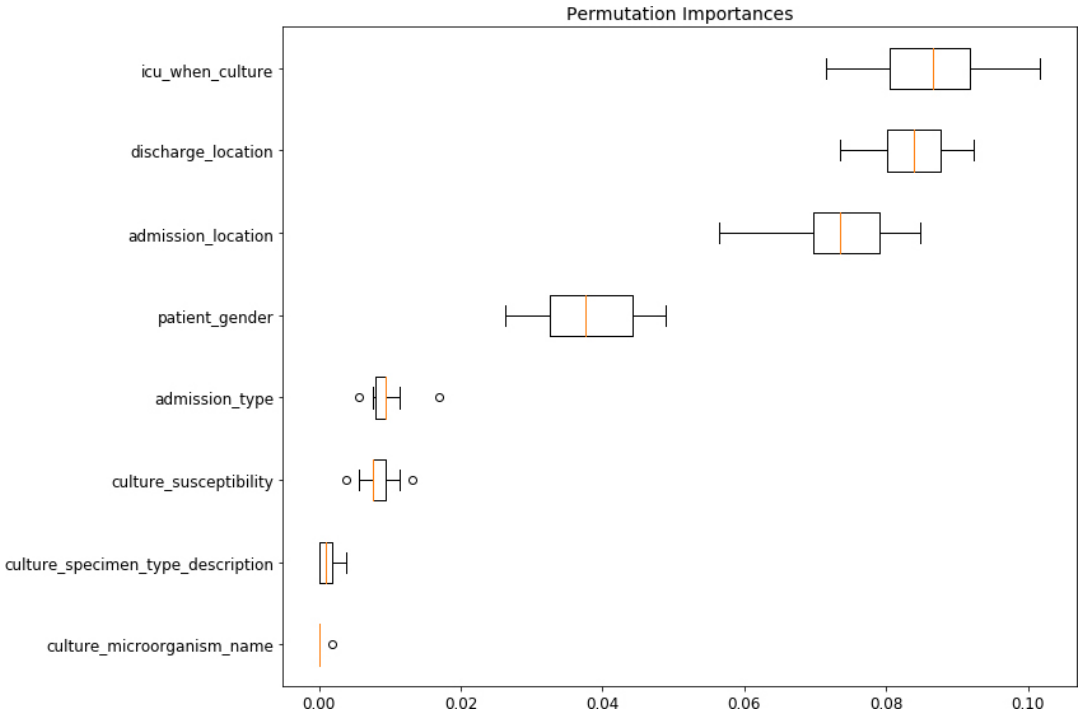


Figure 2.18: Step 5: importance of the features when using the permutation importance technique for experiment 3 ($k = 115$).

In all three experiments it will be noted that the classification process prioritises the same subset of features. The description of the candidate clusters from experiment 1, focusing solely on that particular subset of features, is presented in Table 2.10.

2.5.6 Comparison with a traditional clustering

In this section, we assess our methodology by comparing it with a conventional clustering approach in terms of stability. We specifically utilise the 19 stable candidate clusters obtained in Section 2.5.4 as a reference. For each candidate cluster, we determine the extent to which its elements are grouped together in a cluster generated by a traditional clustering algorithm. This cluster is automatically selected as the most overlapped cluster with the stable candidate cluster that we are considering. We additionally incorporate our own technique into this evaluation process. This comparison is performed by employing the following values for the k hyperparameter: 150, 105, and 115 (the same three values used in the experiments). We also utilise the following traditional clustering methods provided by *scikit-learn*: K-Means, Spectral Clustering, and Mini Batch K-Means.

We perform 200 iterations of each traditional algorithm in addition to our own

Cluster	gender (count)	admission type (count)	admission location (count)	discharge location (count)	culture specimen type description (count)	culture microorganism name (count)	culture susceptibility (count)	icu_ when_ culture (count)
p150_c6 *Size: 13*	M:10 F:3	Elective:11 Emergency:2	Phys referral/ normal deli:13	Rehab/distinct part hosp:13	Blood culture:11 Peritioneal Fluid:1 Pleural Fluid:1	E. Faecium:13	R:12 S:1	SICU:12 TSICU:1
p150_c11 *Size: 12*	F:9 M:3	Emergency:12	Emergency room admit:11 Transfer from other healt:1	Dead/Expired:12	Blood culture:12	E. Faecium:12	R:9 S:3	NO ICU:12
p150_c18 *Size: 11*	F:11	Emergency:11	Emergency room admit:11	Dead/Expired:11	Blood culture:11	E. Faecium:11	R:11	MICU:11
p150_c20 *Size: 9*	M:7 F:2	Emergency:9	Transfer from hosp/extram:8 Clinic referral/ premature:1	Dead/Expired:9	Blood culture:8 Tissue1	E. Faecium:9	R:9	CSRU:9
p150_c35 *Size: 7*	M:6 F:1	Emergency:7	Emergency room admit:6 Clinic referral/ premature:1	Rehab/distinct part hosp:7	Blood culture:6 CSF Spinal Fluid:1	E. Faecium:7	S:4 R:3	TSICU:6 CSRU:1
p150_c50 *Size: 9*	M:9	Emergency:9	Clinic referral/ premature:9	Dead/Expired:8 SNF:1	Blood culture:9	E. Faecium:9	R:8 S:1	NO ICU:9
p150_c61 *Size: 12*	M:12	Emergency:11 Elective:1	Phys referral/ normal deli:12	Rehab/distinct part hosp:11 Home:1	Blood culture:10 Fluid received in blood culture bottle:1 Bile:1	E. Faecium:12	R:11 S:1	NO ICU:12
p150_c77 *Size: 6*	F:6	Emergency:6	Clinic referral/ premature:6	Long term care hospital:6	Blood culture:6	E. Faecium:6	R:6	NO ICU:6

Table 2.10: Step 5: characterisation of the candidate clusters considering only the important features for experiment 1.

	KMeans	Spectral Clustering	Mini Batch K-Means	Trace-based clustering
Experiment 1 (k = 150)				
p150_c6	90%	82%	60%	89%
p150_c11	81%	81%	83%	80%
p150_c18	100%	97%	99%	100%
p150_c20	95%	82%	62%	97%
p150_c35	87%	72%	45%	85%
p150_c50	96%	89%	82%	96%
p150_c61	97%	91%	83%	98%
p150_c77	98%	97%	97%	98%
Experiment 2 (k = 105)				
p105_c7	92%	90%	93%	92%
p105_c9	97%	96%	99%	97%
p105_c22	85%	85%	72%	84%
p105_c23	68%	73%	81%	69%
p105_c59	89%	86%	73%	90%
p105_c85	95%	90%	98%	95%
Experiment 3 (k = 115)				
p115_c9	92%	90%	92%	91%
p115_c17	97%	87%	80%	97%
p115_c22	91%	86%	66%	92%
p115_c39	94%	84%	77%	94%
p115_c49	96%	95%	99%	98%

Table 2.11: Average percentage of elements of our stable candidate clusters, which are grouped together in the same cluster throughout 200 executions.

technique, using the appropriate k hyperparameter and changing the random seed. We calculate the overlapping percentage for each of the 19 stable candidate clusters in each iteration. Lastly, we determine the average percentage across the 200 executions for each stable candidate cluster. The results of this comparison are illustrated in Table 2.11.

A traditional clustering approach typically consists of three main steps. First, a technique such as the elbow method is employed to determine the optimal number of clusters (referred to as the k hyperparameter). Second, an algorithm such as K-Means is utilised to create the clusters. Lastly, the quality of the clustering result is evaluated using a measure such as the Silhouette score.

By comparing this traditional clustering methodology with a focus on the stability property, Table 2.11 demonstrates that our methodology achieves similar results to the best traditional algorithm. Specifically, a significant proportion of elements in the

candidate clusters obtained from our approach are also grouped together in the clusters generated by both traditional algorithms and our proposal across 200 executions.

We believe that our methodology has two significant advantages when compared to the conventional clustering approach. Firstly, our methodology enables the application of a post-filtering stage with which to reduce the number of clusters obtained, even when the value of the hyperparameter k is high (as demonstrated by a reduction of approximately 95% in Section 2.5.4). However, a traditional clustering algorithm strictly adheres to the specified value of the hyperparameter k, making it impractical to obtain a reduced set of clusters in a database like that presented here. In these cases, the manual examination of a large number of clusters would, according to the elbow technique and the Hopkins statistic, be unrealistic (see Section 2.5.2 for more details). Secondly, our proposal introduces the concept of a trace, which guarantees the existence of a group of elements that remain together throughout the iterative executions, thus ensuring their stability. This guarantee cannot be provided by a traditional clustering algorithm.

2.6 Discussion

This section provides a discussion of our 5-step approach and its practical implementation in the MIMIC-III database, focusing on the experiments carried out.

With regard to the first step of our methodology, it is highly dependent on the problem. In the present scenario, which is detailed in Section 2.3, we present a systematic procedure consisting of standard stages meticulously crafted in order to appropriately preprocess the dataset for the subsequent utilisation of clustering algorithms. Our initial dataset comprised 531 instances and 19 attributes. However, various operations such as attribute type transformation, correlation exploration, and normalisation made it possible to successfully derive a numerical representation of the dataset containing 531 instances and 65 attributes.

Given the specialised nature of our field, it would be advisable to have an expert with domain knowledge. This would enable us to perform tasks such as feature engineering and feature unification, ultimately enhancing the interpretability of our results.

With regard to the second step of our methodology, the choice of the clustering algorithm (referred to as the clustering function in equation 2.1) and the determination of the maximum expected number of clusters (hyperparameter k) rely greatly on the specific problem. In this instance, we opted for K-Means as our clustering algorithm, which is widely used, efficient, and provides a straightforward interpretation owing to

the centroids that effectively represent the data within clusters. Additionally, since the k hyperparameter value plays a crucial role in clustering methods, we employed the Hopkins statistic and the elbow method to identify the appropriate value.

In terms of the clustering function, although the K-Means algorithm is employed in this study, please recall that we have purposely defined this function to be generic. This design choice enables the utilisation of other algorithms and enhances the overall versatility of our approach.

In our study, we conducted experiments with three different values for the hyperparameter k: 150, 105, and 115. These values may be considered high when dealing with smaller datasets. While the elbow method typically suggests choosing low values of k to prevent overfitting (which means having partitions with a low number of clusters and clusters with a high number of elements that explain most of the variability), our methodology focuses on identifying stable candidate clusters for expert review and analysis. We considered that a cluster from the last partition C_k is more stable the higher the value of the matching function M with respect to all of the previous partitions $\{C_2,\ldots,C_{k-1}\}$). It is consequently preferable to select a less conservative value of k when compared to that which the elbow method would suggest.

In relation to the third step of our methodology, the automatic generation of candidate clusters does not involve significant computational costs. In experiment 1, in which a value of $k = 150$ was employed (i.e. the highest value considered), we had to compute a matrix of size 148×150. The number of cluster comparisons using the matching function was consequently $150 * \sum_{i=2}^{i=149} i$, which we were able to compute using an arithmetic progression: $150 * \frac{148*(2+149)}{2}$. This resulted in 1,676,100 comparisons among clusters. In experiment 2, in which a value of $k = 105$ was employed, we had to compute a matrix of size 103×105, meaning that the number of cluster comparisons was $105 * \sum_{i=2}^{i=104} i$, which we were able to compute using an arithmetic progression: $105 * \frac{103*(2+104)}{2}$. This resulted in 573,195 comparisons among clusters. In experiment 3 (with $k = 115$), we had to compute a matrix of size 113×115. The number of cluster comparisons was consequently $115 * \sum_{i=2}^{i=114} i$, which we were able to compute using an arithmetic progression: $115 * \frac{113*(2+114)}{2}$. This resulted in 753,710 comparisons among clusters.

While the current implementation in this study is straightforward, there is potential for optimisation in order to minimise the number of comparisons. An alternative approach would involve obtaining the trace of the clusters in partition C_k with not all the previous partitions by, for example, using only $\{C_2, C_7, \ldots, C_{k-6}, C_{k-1}\}$. This modification would considerably decrease the number of comparisons among clusters and the frequency of

executing the K-Means algorithm. However, this would also lead to less stability in the final candidate clusters, as not all previous partitions, clusters, and elements would be taken into account.

In relation to the matching function M, we suggest employing the Dice coefficient for two reasons. This choice is based on its ability to represent the normalised matching ratio between elements as a measure of overlapping. Furthermore, the Dice coefficient is comparable to the F-Score utilised in classification tasks and provides a metric that combines the values of PPV and sensitivity.

With respect to the fourth step of our methodology, we opted to use a heat-map to present the matrix $\mathcal{J}$ visually. This choice was made because a heat-map facilitates the detection of groups of values when there are changes in contiguous values. Moreover, as illustrated in Figures 2.7, 2.8 and 2.9, despite having over 100 clusters, it is evident that certain rows contain a notable presence of high values in the matching function M.

It should be noted that the matching function M may not generate monotonically increasing values for previous partitions. This can be observed in the heat-maps, in which lighter colours are interspersed between darker colours (and vice versa). The reason for this is that the clustering algorithm undergoes a random initialisation in each partition, leading to potential variations in the clusters from one partition to another. This effect would not be noticeable if a hierarchical clustering approach were employed.

With regard to our proposal, we highlight the significance of the filtering threshold employed in the optional post-filtering procedure. When the threshold is increased, the user will obtain a reduced number of candidate clusters, but these clusters will be of a superior quality. This implies that a significant number of elements within the clusters will remain grouped together when compared to the previous partitions. Conversely, decreasing the threshold will result in a larger number of candidate clusters, but some of them may be of a lower quality. In other words, fewer elements in their trace will remain grouped together when compared to the previous partitions.

Note that any candidate clusters not chosen during the post-filtering process might represent false negatives that potentially have clinical significance. The filtering threshold therefore has a direct correlation with false negatives. Raising the threshold could lead to an increase in false negatives, while lowering the threshold could result in a decrease in false negatives.

The mean value of the Dice coefficient for each partition (i.e. for each row of the matrix $\mathcal{J}$) in experiment 1 is illustrated in Figure 2.19. It is expected that the Dice coefficient will be low when comparing clusters from a partition with a small number

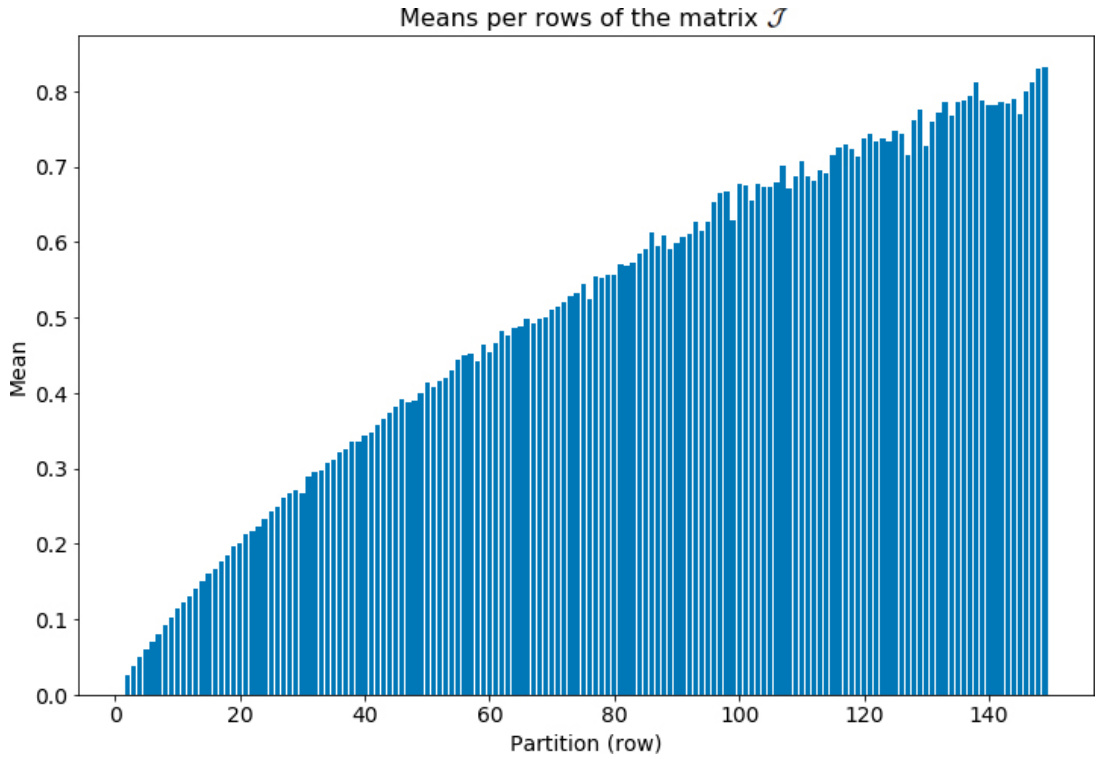


Figure 2.19: Experiment 1: means per row of the matrix $\mathcal{J}$.

of clusters (e.g., the extreme case of 2 clusters) to clusters from a partition with a high number of clusters (in this case, partition C_{150}). This is because the denominator takes the sizes of the two input clusters into account, and clusters in the partition with 2 clusters generally have many more elements than clusters in partition C_{150}.

While the mean is utilised to rank the candidate clusters (columns in matrix $\mathcal{J}$), alternative statistics can be taken into account in order to address domain-specific requirements. For our particular clinical issue, we examine the average values of the Dice coefficient for the columns of matrix $\mathcal{J}$. As depicted in Figure 2.20, the histogram has an approximately normal distribution, leading us to assume that the mean can serve as a representative measure with which to establish a ranking and prioritise the candidate clusters.

For the sake of simplicity, the candidate clusters chosen in the fourth step are a subset of only the clusters within partition C_k (specifically, from C_{k1} to C_{kk}). Nevertheless, it is possible to extend the proposed approach in order to allow expert users to choose clusters from previous partitions. As a result of this selection, the candidate clusters chosen may share some patients, since the clustering algorithm for each partition was initialised in a different manner. The trace of a cluster therefore differs from a branch in hierarchical

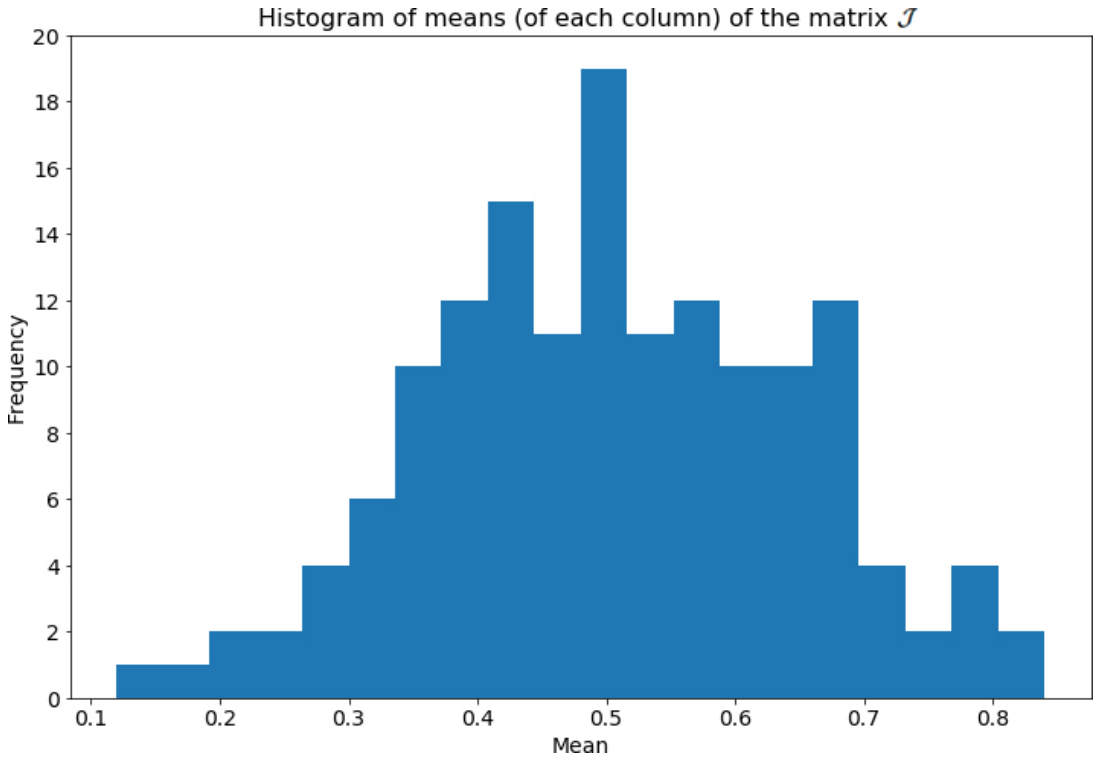


Figure 2.20: Experiment 1: histogram of means (of each column) of the matrix $\mathcal{J}$.

clustering methods.

With respect to the fifth step of our methodology, it is relevant to emphasise that despite not having access to the patients' personal records or the expertise of a clinician to examine them, we were able to assess the consistency of the candidates by employing a classification method.

The results of this process suggest the following: (1) there is a significant level of distinctiveness among the elements within each candidate cluster, as the majority of cases (excluding only 4 cases) accurately classified all the elements within their respective clusters, and (2) despite being an imbalanced problem with numerous minority classes, the elements within the candidate clusters can be clearly differentiated from the remaining elements labelled as "other_cluster".

The preliminary evidence is reinforced by the validation accuracies obtained, which consistently exceeded 90% in all cases. These values signify that at least 90% of the elements were accurately classified in the experiments conducted. The candidate clusters generated using our methodology consequently had a high level of consistency, separability, and distinctiveness from each other and from the remaining elements. We can, therefore, confidently conclude that the characterisation presented in Table 2.10,

which shows interesting candidate clusters associated with Vancomycin resistance and mortality, is also highly representative and coherent.

Furthermore, as indicated in Table 2.10, there are candidate clusters that consistently have a shared phenotype of resistance and death, thus distinguishing them from the remaining candidate clusters. It is plausible that an expert in the field could assign clinically meaningful labels to these candidates.

Achieving stability in the results obtained was a significant concern during these experiments. This stability can be verifying by examining whether employing higher values of k, which correspond to a larger number of candidate clusters, leads to increased knowledge. It is also possible to investigate whether there are candidate clusters with identical elements across different experiments.

One key factor is that the candidate clusters are significantly different from each other and have interesting variables that allow users to differentiate them from the other validated and demonstrated candidate clusters.

The three experiments depicted in Figures 2.16, 2.17 and 2.18 reveal that the key attributes that differentiate the candidate clusters chosen in step 4 from each other and from the remaining candidate clusters are the same. We can consequently infer that these selected candidate clusters have both coherence and distinctiveness.

When examining whether there are candidate clusters throughout the three experiments that contain the same elements, it will be observed that: (1) cluster p150_c20 in experiment 1 shares the same elements as cluster p115_c17 in experiment 3, (2) cluster p105_c7 in experiment 2 shares the same elements as p115_c9 in experiment 3, and (3) cluster p105_c9 in experiment 2 shares the same elements as cluster p115_c49 in experiment 3.

With regard to the application of hierarchical clustering as opposed to Trace-based clustering, the possibility of utilising hierarchical clustering by selectively choosing branches at a suitable depth could be considered. Nonetheless, in both the bottom-up and top-down hierarchical clustering approaches, each level is directly influenced by the next. However, in Trace-based clustering, each previous partition that is used to calculate the trace of a cluster in partition C_k is initialised with a random seed. The clusters in these partitions are consequently independent of each other. This makes Trace-based clustering a more robust technique and less susceptible to being affected by dependencies among partitions.

Finally, in this study, we do not discuss the practical application of the results for decision support. Our focus is rather on using MIMIC-III solely to validate the

proposed methodology. For example, we do not discuss whether the definition of patient clusters has clinical significance. Nevertheless, we have obtained results that could be potentially interesting, as demonstrated in experiment 1 (Table 2.10). Specifically, it is worth mentioning that three of the clusters exclusively identify patients with a resistant bacterial strain or a high percentage of patient mortality.

2.7 Conclusions

The purpose of this study was to provide patient phenotyping with support by introducing a new unsupervised ML technique called Trace-based clustering, along with a 5-step methodology.

Conventional clustering or subgroup discovery approaches have been used in previous research on ML-based phenotyping. While clustering strategies identify separate groups within a population, subgroup discovery can identify overlapping subsets of patients but requires a-priori information. Our approach offers a fresh viewpoint by using clustering techniques to find patient sets and evaluating them on the basis of the overlap of clusters from previous partitions.

The objective of the proposed methodology is to enhance the consistency and interpretability of the pipeline, thus enabling clinical experts to actively trace and control the automatic phenotyping identification process. Our experiments confirm the effectiveness of utilising centroid-based clustering algorithms and visual analysis to help clinicians to interpret the intermediate results of the process. We have demonstrated the superior benefits of our proposal by comparing it to a conventional clustering approach. These benefits include: (1) enabling the application of a post-filtering stage in order to reduce the number of clusters obtained, (2) introducing the concept of a trace, which guarantees the existence of a group of elements that remain together throughout the iterative executions, thus ensuring their stability, and (3) defining different generic concepts such as clustering function or matching function, which enhance the overall versatility.

Our proposal has been assessed by examining a scenario involving antimicrobial resistance, focusing specifically on patients with infections caused by Methicillin-resistant Staphylococcus Aureus and Enterococcus Faecium who were treated with Vancomycin. The MIMIC-III open-access database was employed for our experiments, thus ensuring transparency and reproducibility at every stage of the study.

The results obtained suggest that a small number of candidate clusters can yield specific mortality and antibiotic resistance traits. Further assessment using a classification

model validated the reliability of these clusters. Although the phenotypes cannot be generalised owing to dataset constraints, it can be inferred that employing this methodology enables experts to clinically interpret and trace the phenotypes throughout the entire process.

We are confident that our robust and comprehensive approach will facilitate phenotype identification. Our future research will focus on two main aspects. Firstly, we aim to enhance the versatility of our methodology by investigating various clustering algorithms, which may necessitate optimising hyperparameters (in addition to k) and assessing their stability. Secondly, we intend to explore new phenotype problems by examining diverse microorganisms and exploring data sources derived from clinical environments.

CHAPTER 3

Subgroup Discovery with VLSD algorithm

This chapter shows how patient phenotyping was confronted by interpreting it as a subgroup discovery (SD) problem. Both a formal introduction to the SD problem and its key components are provided. We contribute to the field by designing VLSD, which is a highly efficient SD algorithm that is able to treat the complexity of phenotyping problems, and by defining an innovate data structure that is used to implement it. One desirable property of this technique is that its results are highly readable, since they are simple conjunctions of descriptions for experts.

3.1 Motivation

SD is a relatively new supervised technique employed in ML. This technique is used in a wide variety of fields, such as that of medicine, to carry out data exploratory analysis, and has obtained remarkable results (Atzmüller, Puppe, & Buscher, 2005; Gamberger & Lavrac, 2002; Jorge, Pereira, & Azevedo, 2006). This technique allows the exploration of the search space of a dataset so as to extract valuable knowledge from it in the form of easily readable descriptions. The use of this technique and the proposal of a new algorithm is consequently a suitable subsequent step for this PhD thesis.

It is necessary to highlight from the beginning that this chapter evaluates our proposal solely from the performance point of view by using a variety of well-known datasets. The proposed SD algorithm is, therefore, used to extract patient phenotypes along with other techniques in Chapter 4.

This chapter makes the following main contributions:

- First, Section 3.4.1 proposes an efficient new algorithm for SD that employs the exploration strategy of equivalence classes and incorporates pruning based on an optimistic estimate.
- Second, Section 3.4.2 describes a new data structure called Vertical List, that is based on the work developed by Zaki, Parthasarathy, Ogihara, and Li (1997) and is employed in our algorithm to allow the efficient computation of subgroup refinements.

The remainder of this chapter is structured as follows: Section 3.2 provides a background to the SD technique and to different existing SD algorithms, and introduces related work, while Section 3.3 defines the formal aspects of the SD technique. Section 3.4 explains and describes our proposal: the VLSD algorithm (Section 3.4.1) and vertical list data structure (Section 3.4.2). Section 3.5 shows the definition of the configuration of the experiments carried out in order to compare our proposal with other state-of-the-art SD algorithms, the results obtained after this comparison stage and a discussion of those results. Finally, Section 3.6 provides the conclusions reached after carrying out this research.

3.2 Background

Before presenting the current state of the art of the SD technique, we would like to emphasize the distinctions between SD and other techniques, such as clustering, classification, or pattern mining. Firstly, the objective of classification algorithms is to generate a global model for the entire population, predicting the outcome of new observations. In contrast, SD algorithms produce local descriptive models that consider statistically significant subpopulations related to a single value of the target attribute. Moreover, the subpopulations covered by different subgroups in SD algorithms may overlap, while a classification model does not allow such an overlap. Secondly, clustering and pattern mining algorithms are unsupervised and do not rely on an output attribute or class, while SD algorithms are supervised and generate relations, referred to as subgroups, based on a target value.

SD algorithms have various characteristics that distinguish them from each other, necessitating consideration depending on the problem and input data under analysis. Noteworthy aspects include: (1) the exploration strategy employed by the SD algorithm within the search space of the problem, which can be either exhaustive or heuristic;

(2) the number of subgroups returned by the SD algorithm (all subgroups explored versus top-k subgroups); (3) whether the SD algorithm incorporates additional pruning techniques in order to avoid exploring lower-quality regions of the search space, such as pruning based on optimistic estimates, and (4) the specific data structure utilised by the SD algorithm, such as FPTree, TID List, or Bitset.

There are two types of SD algorithms depending on the exploration strategy: exhaustive and heuristic algorithms. Exhaustive algorithms explore the entire search space of the problem, while heuristic algorithms use a heuristic function to guide the exploration of the search space. Exhaustive algorithms ensure the discovery of the best subgroups, but they may not be feasible when the search space of the problem is too large. In these cases, heuristic algorithms are employed as an alternative. These algorithms are more efficient and help to reduce the number of subgroups that need to be explored. However, they do not guarantee that the best subgroups will be found (Atzmueller, 2015; Herrera, Carmona, González, & Del Jesus, 2011).

When an SD algorithm (either exhaustive or heuristic) is executed along with a predetermined quality measure and threshold, it can yield either the complete set of explored subgroups or only the top-k best subgroups. Opting for the top-k approach provides the benefit of reducing memory usage in the SD algorithm, as there is no need to store all the subgroups explored (Atzmueller, 2015).

Additional pruning techniques are often incorporated into many SD algorithms so as to enhance efficiency and eliminate the need to explore low-quality regions of the search space. One such technique is pruning based on an optimistic estimate. An optimistic estimate is a quality measure that, for a certain subgroup, provides a quality upper bound for all its refinements (Grosskreutz, Rüping, & Wrobel, 2008). This upper bound represents a value that no refinement of the subgroup can surpass. If the optimistic estimate is lower than the predefined quality threshold, this consequently indicates that refining the current subgroup will not generate any suitable subgroups, thus allowing its elimination. The use of this pruning technique makes it unnecessary to explore entire regions of the search space that fall below the established quality threshold after analysing only one subgroup.

One disadvantage of the SD technique is that it may potentially generate a vast number of subgroups, which is known as pattern explosion. This becomes particularly significant when dealing with input datasets that have a high number of attributes. This issue can be addressed by employing an optimistic estimate. If a quality measure threshold is established, it is possible to avoid exploring a substantial portion of the

search space.

It is relevant to note that the conventional quality measures for SD, such as Sensitivity, Specificity, WRAcc, or Information Gain, are neither optimistic estimates nor are they monotonic. This implies that, when employing these standard measures, refined subgroups may have higher quality measures than their parents, thus necessitating exploration of the entire search space. However, optimistic estimate quality measures, which are inherently monotonic, can be utilised to prune and reduce the search space of the problem (Grosskreutz et al., 2008).

SD algorithms are often based on non-SD algorithms. Numerous SD algorithms are actually modified versions of classification algorithms or frequent pattern mining algorithms, among others. In such cases, adjustments are made to the data structures and algorithmic schemes in order to achieve the goal of identifying subgroups.

The following are examples of SD algorithms based on existing classification algorithms: EXPLORA (Klösgen, 1996), MIDOS (Wrobel, 1997), PRIM (Friedman & Fisher, 1999), SubgroupMiner (Klösgen & May, 2002), RSD (Lavrac, Železný, & Flach, 2003), CN2-SD (Lavrac, Kavsek, Flach, & Todorovski, 2004) or SD (Lavrac & Gamberger, 2004), among others. The following are examples of SD algorithms based on existing frequent pattern mining algorithms: Apriori-SD (Kavšek, Lavrac, & Jovanoski, 2008), DpSubgroups (Grosskreutz et al., 2008), SD4TS (Mueller et al., 2009) or SD-Map* (Lemmerich, Atzmüller, & Puppe, 2015), among others.

This section also provides an explanation of SD-Map (Atzmueller & Puppe, 2006) and BSD (Lemmerich, Rohlfs, & Atzmüller, 2010) algorithms, as they serve as notable examples of exhaustive SD algorithms. These algorithms are built upon existing frequent pattern mining algorithms.

SD-Map is an exhaustive SD algorithm based on the FP-Growth (Han, Pei, & Yin, 2000) algorithm, which is a well-known algorithm for frequent pattern mining. The SD-Map algorithm employs the FPTree data structure in order to effectively represent the entire dataset and perform subgroup mining through the use of a two-step process. A complete FPTree is initially constructed using the input dataset, followed by the recursive creation of successive conditional FPTrees to extract subgroups.

BSD is an exhaustive SD algorithm that employs the Bitset data structure and the depth-first search strategy. Each subgroup is associated with a Bitset data structure that keeps track of the instances covered and not covered by that subgroup by employing bits. This data structure has several benefits: (1) it reduces memory usage by employing a bitset-based representation for coverage information, (2) subgroup refinements can be

efficiently obtained by means of logical AND operations, and (3) it can be implemented with high efficiency in terms of time and memory in most programming languages. The mining of subgroups using this data structure involves two steps: a Bitset data structure is first built for each single selector involved in the SD process, and all possible refinements are then derived recursively.

In a specific SD algorithm it is sometimes possible for two subgroups to be redundant, as they essentially represent and explain the same portion of data in a dataset. When redundancy occurs, one of the subgroups is considered dominant while the other is considered dominated in terms of its coverage. The dominated subgroup can consequently be eliminated. In this context, it is possible to highlight two types of dominance relations: closed dominance (Garriga, Kralj Novak, & Lavrac, 2006) and closed-on-the-positives dominance (Lemmerich et al., 2010). On the one hand, two subgroups have a closed dominance relation if the instances covered by both subgroup descriptions (regardless of the target value) are identical. In this case, the most specific subgroup is dominant, while the most general subgroup is dominated. On the other hand, two subgroups have a closed-on-the-positives dominance relation if the positive instances (instances with a positive target value) covered by both subgroup descriptions are the same. Here, the most general subgroup is dominant, while the most specific subgroup is dominated.

The aforementioned algorithms can be adjusted and customised in order to to exclusively identify closed subgroups or solely identify closed-on-the-positives subgroups.

Another strategy has also been employed in addition to the aforementioned exploration strategies: the equivalence class strategy, which has been used in frequent pattern mining. This particular strategy was proposed by Zaki et al. (1997), and it has not, to the best of our knowledge, been used in any SD algorithm.

There are other approaches related to pattern mining that aim to achieve similar objectives. One such technique is utility pattern mining, which is extensively discussed and utilised in literature. It involves the discovery of patterns that have significant relevance based on a numeric utility function. This function not only evaluates the quality or importance of a pattern within a specific dataset but also takes into account additional criteria beyond the database itself (Nouioua, Fournier Viger, Wu, Lin, & Gan, 2021; Qu, Fournier-Viger, Liu, Hang, & Wang, 2020). It should be noted that while these algorithms employ upper bound measures to reduce the search space, which may not be monotonic, we utilise optimistic estimate quality measures that are inherently monotonic. Alternative methods have also recently been introduced for the mining of patterns in scenarios in which the amount of data available is limited. For instance, Le

et al. (2021) presented an algorithm designed to mine colossal patterns, which refers to patterns extracted from databases with numerous attributes and values but with few instances.

Finally, for a general understanding of the SD technique, we refer the reader to Atzmueller (2015); Herrera et al. (2011).

3.3 Definition of problem

The fundamental concepts of the SD technique are provided as follows:

Definition 3.1 (Attribute a)**.** An attribute a is a unique characteristic of an object, which has an associated value. An example of an attribute is $a = age : 30$.

Definition 3.2 (Domain of an attribute a)**.** The domain of an attribute a (denoted as $dom(a)$) is the set of all the unique values that said attribute can take. An attribute can be nominal or numeric, depending on its domain.

Definition 3.3 (Instance i)**.** An instance i is a tuple $i = (a_1, \ldots, a_M)$ of attributes. Given the attributes $a_1 = age : 25$, $a_2 = headache : no$ and $a_3 = fever : yes$, an example of an instance is $i = (age : 25, headache : no, fever : yes)$.

Definition 3.4 (Dataset d)**.** A dataset d is a tuple $d = (i_1, \ldots, i_N)$ of instances. Given the instances $i_1 = (age : 25, headache : no, fever : no)$ and $i_2 = (age : 30, headache : yes, fever : yes)$, an example of a dataset is $d = ((age : 25, headache : no, fever : no), (age : 30, headache : yes, fever : yes))$.

We denote $\mathcal{D}$ as the dataset space.

In a dataset d, each value can be accessed using two integers, x and y. We denote the value of the x-th instance i_x and of the y-th attribute a_y in d as $v_{x,y}$.

Definition 3.5 (Selector e)**.** Given an attribute a_y from a dataset d, a binary $operator \in \{=, \neq, <, >, \leq, \geq\}$ and a value $w \in dom(a_y)$, a selector e is a 3-tuple of the form $(a_y.characteristic, operator, w)$. Note that when an attribute a_y is nominal, only the $=$ and $\neq$ operators are permitted. Some examples of selectors are $e_1 = (headache, =, yes)$ and $e_2 = (fever, =, no)$.

A selector can be informally understood as a binary relation between an attribute from a dataset and a value in the domain of that attribute. This relation represents a property possessed by a particular subset of instances within the dataset.

It is necessary to remark that the first element of a selector only relates to the attribute name, i.e. the characteristic, and not the complete attribute.

Definition 3.6 (Selector covering)**.** Given an instance i_x and an attribute a_y from a dataset d, and a selector $e = (a_y.characteristic, operator, w \in dom(a_y))$, then i_x is covered by e (denoted as $i_x \sqsubset e$) if the binary expression "$v_{x,y}\ operator\ w$" holds "true". Otherwise, we say that it is not covered by e (denoted as $i_x \sqsubset\!\!\!\!+\ e$).

For example, given the instance $i_1 = (headache : no, fever : yes)$ and the selectors $e_1 = (headache, =, no)$ and $e_2 = (fever, =, no)$, it will be noted that $i_1 \sqsubset e_1$ and $i_1 \sqsubset\!\!\!\!+\ e_2$.

Definition 3.7 (Pattern p)**.** A pattern p is a list of selectors $< e_1, \ldots, e_j >$ in which all attributes of the selectors are different. Moreover, its size (denoted as $|p|$) is defined as the number of selectors that it contains.

A pattern is generally understood as a list of selectors (in the form of a conjunction) which represents a list of properties of a specific subset of instances within a dataset.

Definition 3.8 (Pattern covering)**.** Given an instance i_x from a dataset d and a pattern p, then i_x is covered by p (denoted as $i_x \sqsubset p$) if $\forall e \in p, i_x \sqsubset e$. Otherwise, we say that it is not covered by p (denoted as $i_x \sqsubset\!\!\!\!+\ p$).

Definition 3.9 (Subgroup s)**.** A subgroup s is a pair (pattern, selector) in which the pattern is denoted as $s.description$ and the selector is denoted as $s.target$. Given the dataset $d = ((headache : no, fever : yes, flu : yes), (headache : yes, fever : no, flu : no))$, an example of a subgroup is $s = (< (headache, =, no), (fever, =, yes) >, (flu, =, no))$.

We denote $\mathcal{S}$ as the subgroup space.

Definition 3.10 (Subgroup refinement s')**.** Given a subgroup s, each of its refinements s' (denoted as $s \prec s'$) is a subgroup with the same target, $s'.target = s.target$, and with an extended description, $s'.description = concat(s.description, < e_1, \ldots, e_j >)$.

Definition 3.11 (Refine operator)**.** Given two subgroups, s_x and s_y, the $refine$ operator generates a refinement $s_{x,y}$ of s_x, extending its description with the non-common suffix of s_y. For example, if $s_x.description =< e_1 >$ and $s_y.description =< e_2 >$, then $s_{x,y}.description =< e_1, e_2 >$; and if $s_x.description =< e_1, e_2, e_3 >$ and $s_y.description =< e_1, e_2, e_4 >$, then $s_{x,y}.description =< e_1, e_2, e_3, e_4 >$. Formally:

$$refine : \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{S} \tag{3.1}$$

This signifies that the $refine$ operator accepts two subgroups as input and produces one subgroup as output.

Definition 3.12 (Quality Measure q)**.** Given a subgroup s and a dataset d, a quality measure q is a function that computes one numeric value according to that subgroup s and to certain characteristics from that dataset d. Formally:

$$q : \mathcal{S} \times \mathcal{D} \rightarrow \mathbb{R} \tag{3.2}$$

$$q(s,d) \in \mathbb{R} \tag{3.3}$$

Definition 3.13 (Optimistic Estimate oe)**.** Given a quality measure q and a dataset d, an optimistic estimate oe of q is a quality measure that satisfies the following condition:

$$\forall s, s' \, , \, s \prec s' \Rightarrow oe(s,d) \geq q(s',d) \tag{3.4}$$

Informally, an optimistic estimate refers to a quality measure that, for a certain subgroup, establishes a quality upper bound for all its refinements (Grosskreutz et al., 2008).

The subsequent functions can be defined by concentrating on a particular subgroup denoted as s and a specific dataset identified as d:

Definition 3.14 (Function tp (true positives))**.** The function tp is defined as the number of instances i_x from the dataset d that are covered by the subgroup description $s.description$ and by the subgroup target $s.target$. Formally:

$$tp : \mathcal{S} \times \mathcal{D} \rightarrow \mathbb{N} \tag{3.5}$$

$$tp(s,d) = |\{i_x \in d : i_x \sqsubset s.description \wedge i_x \sqsubset s.target\}| \tag{3.6}$$

Definition 3.15 (Function fp (false positives))**.** The function fp is defined as the number of instances i_x from the dataset d that are covered by the subgroup description $s.description$, but not by the subgroup target $s.target$. Formally:

$$fp : \mathcal{S} \times \mathcal{D} \rightarrow \mathbb{N} \tag{3.7}$$

$$fp(s,d) = |\{i_x \in d : i_x \sqsubset s.description \wedge i_x \sqsupset\!\!\!\!\cdot \, s.target\}| \tag{3.8}$$

Definition 3.16 (Function TP (true population))**.** The function TP is defined as the number of instances i_x from the dataset d that are covered by the subgroup target $s.target$. Formally:

$$TP : \mathcal{S} \times \mathcal{D} \rightarrow \mathbb{N} \tag{3.9}$$

$$TP(s,d) = |\{i_x \in d : i_x \sqsubset s.target\}| \tag{3.10}$$

Definition 3.17 (Function FP (false population))**.** The function FP is defined as the number of instances i_x from the dataset d that are not covered by the subgroup target $s.target$. Formally:

$$FP : \mathcal{S} \times \mathcal{D} \rightarrow \mathbb{N} \tag{3.11}$$

$$FP(s,d) = |\{i_x \in d : i_x \dot{\sqcap} s.target\}| \tag{3.12}$$

The four previous functions, therefore, enable a formal redefinition of a quality measure q in the following manner:

Definition 3.18 (Quality Measure q)**.** Given a subgroup s and a dataset d, a quality measure q is a function that computes one numeric value according to the functions tp, fp, TP and FP. Formally:

$$q : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{R} \tag{3.13}$$

$$q(tp(s,d), fp(s,d), TP(s,d), FP(s,d)) \in \mathbb{R} \tag{3.14}$$

The four functions mentioned above are sufficiently expressive to compute any quality measure. Nevertheless, the following functions are also employed in literature:

Definition 3.19 (Function n)**.** The function n is defined as the number of instances i_x from a dataset d that are covered by the subgroup description $s.description$. Formally:

$$n : \mathcal{S} \times \mathcal{D} \rightarrow \mathbb{N} \tag{3.15}$$

$$n(s,d) = |\{i_x \in d : i_x \sqsubset s.description\}| \tag{3.16}$$

Definition 3.20 (Function N)**.** The function N is defined as the number of instances i_x from the dataset d. Formally:

$$N : \mathcal{S} \times \mathcal{D} \rightarrow \mathbb{N} \tag{3.17}$$

$$N(s,d) = |\{i_x \in d : i_x\}| \tag{3.18}$$

Definition 3.21 (Function p)**.** The function p is defined as the distribution of the subgroup target $s.target$ with respect to the instances i_x from a dataset d covered by the subgroup description $s.description$. Formally:

$$p : \mathcal{S} \times \mathcal{D} \rightarrow \mathbb{N} \tag{3.19}$$

$$p(s,d) = \frac{|\{i_x \in d : i_x \sqsubset s.description \wedge i_x \sqsubset s.target\}|}{|\{i_x \in d : i_x \sqsubset s.description\}|} \tag{3.20}$$

		s.target		
		true	false	
s.description	true	tp	fp	n = tp + fp
	false	fn = TP - tp	tn = FP - fp	TP + FP - tp - fp
		TP = tp + fn	FP = fp + tn	N = TP + FP

Table 3.1: Confusion matrix of a subgroup s with respect to a dataset d.

Definition 3.22 (Function p_0)**.** The function p_0 is defined as the distribution of the subgroup target $s.target$ with respect to all instances i_x from a dataset d. Formally:

$$p_0 : \mathcal{S} \times \mathcal{D} \to \mathbb{N} \tag{3.21}$$

$$p_0(s,d) = \frac{|\{i_x \in d : i_x \sqsubset s.target\}|}{|\{i_x \in d : i_x\}|} \tag{3.22}$$

Definition 3.23 (Function tn (true negatives))**.** The function tn is defined as the number of instances i_x from the dataset d that are covered by neither the subgroup description $s.description$ nor the subgroup target $s.target$. Formally:

$$tn : \mathcal{S} \times \mathcal{D} \to \mathbb{N} \tag{3.23}$$

$$tn(s,d) = |\{i_x \in d : i_x \sqsupset\!\!\!\!\!\!+\ s.description \wedge i_x \sqsupset\!\!\!\!\!\!+\ s.target\}| \tag{3.24}$$

Definition 3.24 (Function fn (false negatives))**.** The function fn is defined as the number of instances i_x from the dataset d that are not covered by the subgroup description $s.description$, but are covered by the subgroup target $s.target$. Formally:

$$fn : \mathcal{S} \times \mathcal{D} \to \mathbb{N} \tag{3.25}$$

$$fn(s,d) = |\{i_x \in d : i_x \sqsupset\!\!\!\!\!\!+\ s.description \wedge i_x \sqsubset s.target\}| \tag{3.26}$$

The functions described above are summarised in the confusion matrix of a subgroup s relative to a dataset d shown in Table 3.1.

In relation to the previously defined functions, the following equivalences can be highlighted:

$$p(s,d) = \frac{tp(s,d)}{tp(s,d) + fp(s,d)} = \frac{tp(s,d)}{n(s,d)} \tag{3.27}$$

$$p_0(s,d) = \frac{TP(s,d)}{TP(s,d) + FP(s,d)} = \frac{TP(s,d)}{N(s,d)} \tag{3.28}$$

After describing the four functions used to compute quality measures, it is possible to rewrite some well-known quality measures for SD from literature as follows:

$$Sensitivity = \frac{tp}{TP} \tag{3.29}$$

$$Specificity = \frac{FP - fp}{FP} \tag{3.30}$$

$$Piatetsky\,Shapiro = (tp + fp) \cdot (\frac{tp}{tp + fp} - \frac{TP}{TP + FP}) \tag{3.31}$$

$$WRAcc = \frac{tp + fp}{TP + FP} \cdot (\frac{tp}{tp + fp} - \frac{TP}{TP + FP}) \tag{3.32}$$

The WRAcc quality measure ranges from -1 to 1, inclusive. It is also possible to express an optimistic estimate of this quality measure (Grosskreutz et al., 2008) in the following manner:

$$WRAcc\,optimistic\,estimate = \frac{tp^2}{tp + fp} \cdot (1 - \frac{TP}{TP + FP}) \tag{3.33}$$

For the sake of brevity and space constraints, the parameters of the functions have, in this case, been omitted.

It is crucial to recall from the outset that while this research uses solely the WRAcc quality measure and its optimistic estimate, these are only an example. Any quality measures with an optimistic estimate could, therefore, be employed.

Definition 3.25 (Subgroup Discovery problem)**.** Given a dataset d, a quality measure q and a numeric value $quality\_threshold$, the SD problem consists of exploring the search space of d in order to enumerate the subgroups that have a quality measure value above the selected threshold. Formally:

$$\mathcal{R} = \{(s, q(s,d)) | q(s,d) \geq quality\_threshold\} \tag{3.34}$$

The search space of a problem (i.e. of a dataset d) can be represented using a lattice (Zaki et al., 1997) (see Figure 3.1). When using this analogy, the first level of the search space comprises subgroups s with a description size of one (meaning that $|s.description|$ is equal to one). The second level similarly encompasses subgroups s with a description size of two (meaning that $|s.description|$ is equal to two). Level n of the search space generally includes subgroups s with a description size of n (meaning that $|s.description|$ is equal to n).

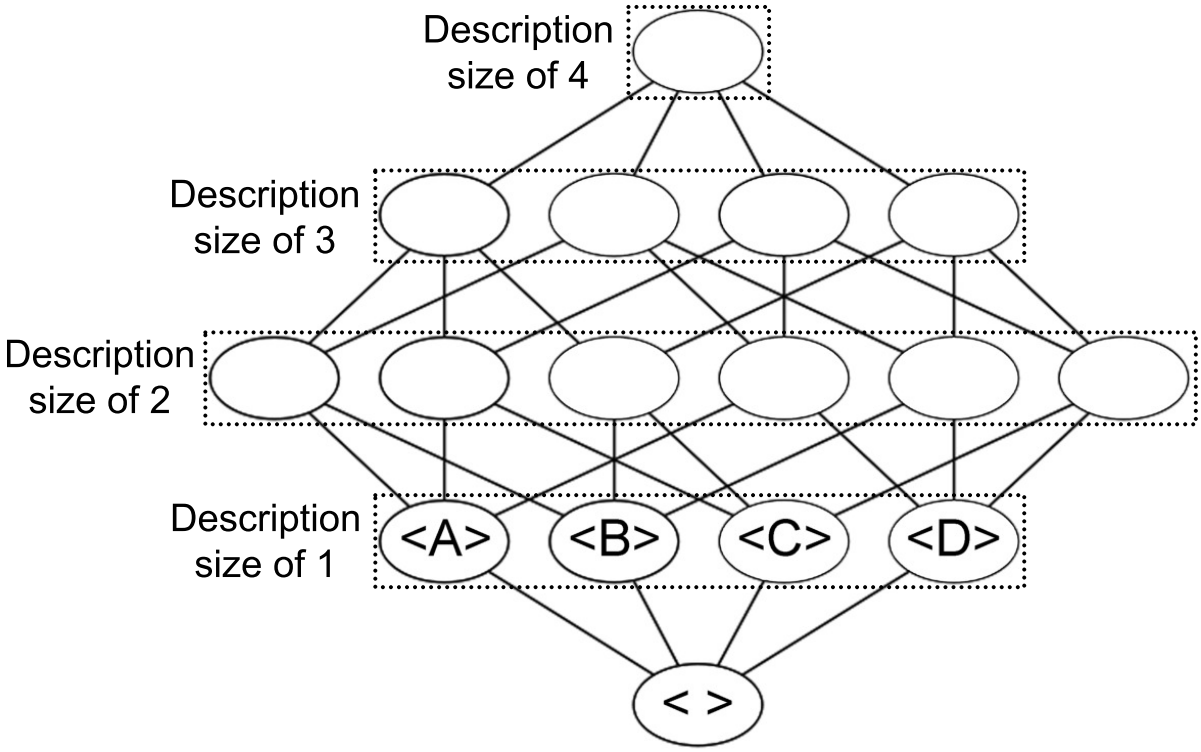


Figure 3.1: Search space of a problem visually illustrated as a lattice.

3.4 Proposal

In this chapter, we propose an efficient new SD algorithm denominated as VLSD (Vertical List Subgroup Discovery), which is defined and explained in Section 3.4.1. We also propose a new data structure denominated as Vertical List, which is used to implement this algorithm in order to compute subgroup refinements efficiently and easily. This is described in Section 3.4.2.

3.4.1 VLSD algorithm

The initial part of our proposal introduces VLSD, which is an efficient new SD algorithm. VLSD comprises an equivalence class exploration strategy (Zaki et al., 1997) and a pruning strategy that relies on optimistic estimation (Grosskreutz et al., 2008). The basis for the execution of this proposal, which makes it easily parallelisable, is the vertical list data structure based on the work developed by Zaki et al. (1997).

The pruning process based on an optimistic estimate involves calculating and comparing the optimistic estimate values of all the nodes (subgroups) generated with a threshold. This determines whether the nodes should be pruned, signifying that their

refinements do not need to be explored, or whether their refinements (the next depth level) should be examined in greater depth.

The VLSD function (Algorithm 3) forms the basis of our proposal and is supported by two functions: one for the generation of subgroups with a description size of one (`GENERATE_SUBGROUPS_S1` function, detailed in Algorithm 4), and another with which to explore the search space and compute the pruning (`SEARCH` function, outlined in Algorithm 5).

The input used for VLSD function (Algorithm 3) comprises the following parameters: a dataset d, a target attribute (a selector) referred to as $target$, a quality measure denoted as q, a threshold value $q\_threshold$ for the quality measure q, an optimistic estimate oe of q, a threshold value $oe\_threshold$ for the optimistic estimate oe, a sorting criterion used for subgroups with a description size of one, and another sorting criterion employed for subgroups with descriptions of sizes greater than one. These sort criteria can, for example, ascend by quality measure value, descend by quality measure value, ascend by description size, have no reordering, etc. Lastly, the function returns a list $\mathcal{F}$ containing the subgroups.

The VLSD function is a constructive function that carries out the following process. First, an empty list called $\mathcal{F}$ is created in order to store subgroups, and the true population TP and false population FP are computed (lines 1 - 2). Next, subgroups with a description size of one (see Figure 3.1) are generated, evaluated and appended to $\mathcal{F}$ after being sorted by a specific criterion (lines 3 - 9), after which a triangular matrix called $\mathcal{M}$ is generated and initialised (lines 10 - 17). This matrix contains subgroups with a description size of two (see Figure 3.1). The two indices i and j of this matrix are selectors, containing $\mathcal{M}[i][j]$, which is the subgroup whose description has two such selectors (or NULL if that subgroup has been pruned). Additionally, $\mathcal{M}[i]$ denotes all subgroups with a description size of two and that start with the selector i. Then, for each selector $selector\_i$ (lines 19 - 20), the subgroups from $\mathcal{M}$ with a description size of two and starting with $selector\_i$ are obtained. These subgroups are subsequently evaluated, added to $\mathcal{F}$, and recursively explored (lines 18 - 31).

The use of matrix $\mathcal{M}$ allows the algorithm to achieve high efficiency. It stores subgroups with a description size of two, thus allowing the quick and effortless pruning of the remaining search space with larger cardinalities, i.e. refinements of those subgroups with a description size of two (Fournier-Viger, Gomariz, Campos, & Thomas, 2014).

The input used by the `GENERATE_SUBGROUPS_S1` function (Algorithm 4) are the following parameters: a dataset denoted as d, a target attribute $target$ (which is a selector), an

Algorithm 3 VLSD function.

Input: d { dataset }, $target$ { selector }, q { quality measure }, $q\_threshold$ { $\mathbb{R}$ }, oe { optimistic estimate of q }, $oe\_threshold$ { $\mathbb{R}$ }, $sort\_criterion\_in\_\mathcal{S}1$ { criterion }, $sort\_criterion\_in\_other\_sizes$ { criterion }

Output: $\mathcal{F}$: list of subgroups.

1: $\mathcal{F}$:=<>
2: $TP := TP((<>,target),d)$; $FP := FP((<>,target),d)$
3: $\mathcal{S}1$:= GENERATE_SUBGROUPS_S1($d,target,oe,oe\_threshold,$ $sort\_criterion\_in\_\mathcal{S}1,TP,FP$)
4: **for each** subgroup $s \in \mathcal{S}1$ **do**
5: $q\_value := q(tp(s,d),fp(s,d),TP,FP)$
6: **if** $q\_value \geq q\_threshold$ **then**
7: $\mathcal{F}.add(s)$
8: **end if**
9: **end for**
10: $\mathcal{M}$:= 2-dimensional $|\mathcal{S}1| \times |\mathcal{S}1|$ triangular matrix, initialized $\mathcal{M}[i,j] = NULL$, in which $\mathcal{M}[i,j]$ is a subgroup (i and j selectors acting as indices).
11: **for each** s_x, s_y in $\mathcal{S}1 =< s_1, s_2, \ldots, s_n >$, being $x < y$ **do**
12: $s_{xy} := refine(s_x, s_y)$
13: $oe\_quality := oe(tp(s_{xy},d),fp(s_{xy},d),TP,FP)$
14: **if** $(tp(s_{xy},d) + fp(s_{xy},d) > 0)$ AND $(oe\_quality \geq oe\_threshold)$ **then**
15: $\mathcal{M}[last(s_x.description)][last(s_y.description)] := s_{xy}$
16: **end if**
17: **end for**
18: **if** $|\mathcal{S}1| \geq 2$ **then**
19: **for** $i := 0$ **to** $(|\mathcal{S}1| - 2)$ **do**
20: $selector\_i := last(\mathcal{S}1[i].description)$
21: $\mathcal{P} := \mathcal{M}[selector\_i]$ { All subgroups whose descriptions are size two and start with $selector\_i$ }
22: $\mathcal{P} := \mathcal{P}.sort(sort\_criterion\_in\_other\_sizes)$
23: **for each** subgroup $s \in \mathcal{P}$ **do**
24: $q\_value := q(tp(s,d),fp(s,d),TP,FP)$
25: **if** $q\_value \geq q\_threshold$ **then**
26: $\mathcal{F}.add(s)$
27: **end if**
28: **end for**
29: $\mathcal{F}.add\_all(SEARCH(d,\mathcal{P},\mathcal{M},q,q\_threshold,oe,oe\_threshold,$ $sort\_criterion\_in\_other\_sizes,TP,FP))$
30: **end for**
31: **end if**
32: **return** $\mathcal{F}$

Algorithm 4 `GENERATE_SUBGROUPS_S1` function.

Input: d { dataset }, $target$ { selector }, oe { optimistic estimate }, $oe\_threshold$ { $\mathbb{R}$ }, $sort\_criterion\_in\_\mathcal{S}1$ { criterion }, TP { $\mathbb{N}$ }, FP { $\mathbb{N}$ }

Output: $\mathcal{S}1$: list of subgroups whose descriptions are size one.

1: $\mathcal{S}1$:=<>
2: $\mathcal{E}$:= scan d (except the target attribute) to generate the selector list.
3: **for each** selector $e \in \mathcal{E}$ **do**
4: $s := (< e >, target)$
5: $oe\_quality := oe(tp(s,d), fp(s,d), TP, FP)$
6: **if** $oe\_quality \geq oe\_threshold$ **then**
7: $\mathcal{S}1.add(s)$
8: **end if**
9: **end for**
10: $\mathcal{S}1 := sort(\mathcal{S}1, sort\_criterion\_in\_\mathcal{S}1)$
11: **return** $\mathcal{S}1$

optimistic estimate denoted as oe, an optimistic estimate threshold called $oe\_threshold$ for oe, a sorting criterion that is employed to sort subgroups with a description size of one, and the TP and FP values extracted from the dataset d. The latter two values are passed as parameters in order to avoid multiple computations of the same elements. Lastly, the function returns a list $\mathcal{S}1$ containing subgroups with a description size of one.

This function initiates by creating an empty list called $\mathcal{S}1$, which will serve as storage for the subgroups (line 1). Next, a selector list $\mathcal{E}$ that is based on the dataset d is generated (line 2). A subgroup is subsequently formed for each selector in the list, evaluated, and appended to $\mathcal{S}1$ (lines 3 - 9). Finally, the list of subgroups $\mathcal{S}1$ is sorted (line 10).

The input used by the `SEARCH` function (Algorithm 5) are the following parameters: a dataset d, a list of subgroups $\mathcal{P}$, a triangular matrix $\mathcal{M}$, a quality measure q, a threshold $q\_threshold$ for the quality measure q, an optimistic estimate oe of q, a threshold $oe\_threshold$ for the optimistic estimate oe, a sort criterion that is used to sort subgroups with descriptions of sizes greater than one, and the TP and FP values obtained from the dataset d (these values are passed as parameters in order to avoid multiple computations of the same elements). Lastly, the function returns a list $\mathcal{F}$ containing subgroups.

This function initiates by creating an empty list called $\mathcal{F}$ in which to store the subgroups (line 1). It then performs a double iteration by using the list of subgroups $\mathcal{P}$ (loops on lines 2 and 5). This results in the generation of new subgroup refinements, which are added to the list of subgroups $\mathcal{L}$ (lines 9 - 18). These subgroups are evaluated

Algorithm 5 SEARCH function.

Input: d { dataset }, $\mathcal{P}$ { list of subgroups }, $\mathcal{M}$ { matrix }, q { quality measure }, $q\_threshold$ { $\mathbb{R}$ }, oe { optimistic estimate of q }, $oe\_threshold$ { $\mathbb{R}$ }, $sort\_criterion\_in\_other\_sizes$ { criterion }, TP { $\mathbb{N}$ }, FP { $\mathbb{N}$ }

Output: $\mathcal{F}$: list of subgroups.

1: $\mathcal{F}$:=<>
2: **while** $|\mathcal{P}| > 1$ **do**
3: $s_x := pop\_first(\mathcal{P})$
4: $\mathcal{L}$:=<> { List of subgroups }
5: **for each** subgroup $s_y \in \mathcal{P}$ **do**
6: $s_{\mathcal{M}} := get(\mathcal{M}, last(s_x.description), last(s_y.description))$
7: $oe\_quality := oe(tp(s_{\mathcal{M}}, d), fp(s_{\mathcal{M}}, d), TP, FP)$
8: **if** $(s_{\mathcal{M}} \neq NULL)$ AND $(oe\_quality \geq oe\_threshold)$ **then**
9: $s_{xy} := refine(s_x, s_y)$
10: $oe\_quality := oe(tp(s_{xy}, d), fp(s_{xy}, d), TP, FP)$
11: **if** $(tp(s_{xy}, d) + fp(s_{xy}, d) > 0)$ AND $(oe\_quality \geq oe\_threshold)$ **then**
12: $\mathcal{L}.add(s_{xy})$
13: $q\_value := q(tp(s_{xy}, d), fp(s_{xy}, d), TP, FP)$
14: **if** $q\_value \geq q\_threshold$ **then**
15: $\mathcal{F}.add(s_{xy})$
16: **end if**
17: **end if**
18: **end if**
19: **end for**
20: **if** $\mathcal{L} \neq$<> **then**
21: $\mathcal{L} := sort(\mathcal{L}, sort\_criterion\_in\_other\_sizes)$
22: $\mathcal{F}.add\_all(SEARCH(d, \mathcal{L}, \mathcal{M}, q, q\_threshold, oe, oe\_threshold, sort\_criterion\_in\_other\_sizes, TP, FP))$
23: **end if**
24: **end while**
25: **return** $\mathcal{F}$

and appended to the list $\mathcal{F}$ (lines 13 - 16). It is essential to emphasize that, as shown in lines 6 - 8, the matrix $\mathcal{M}$ is utilised to prevent the unnecessary generation of subgroup refinements, effectively reducing the search space (Fournier-Viger et al., 2014). Finally, the list of subgroups $\mathcal{L}$ is sorted, and the function is recursively called (lines 20 - 23).

3.4.2 Vertical List data structure

The second part of our proposal consists of a data structure denominated as a vertical list. The idea behind this data structure was initially conceived by Zaki et al. (1997), who proposed a new means of representing a transactional dataset (called a Vertical Data Layout) in which that dataset consists of a list of items in which each item is followed by a list of transaction IDs (denominated as the TID List) in which that item appears. This idea has, therefore, been improved and adapted to the SD technique, achieving a data structure that facilitates the computation of subgroup refinements in an easy and efficient manner through list concatenations and set intersections. It simultaneously ensures the storage of all necessary elements in order to both compute any quality measure and eliminate the need for redundant recalculations.

After receiving a dataset d and identifying a subgroup s, a vertical list, denoted as vl, is created comprising the subsequent components:

- The subgroup description, which is denoted as $vl.description$.
- The set of IDs of the instances counted in $fp(s,d)$, which is denoted as $vl.set\_fp$.
- The set of IDs of the instances counted in $tp(s,d)$, which is denoted as $vl.set\_tp$.

Figure 3.2 provides an example of a vertical list data structure and an adapted $refine$ operator designed for it. In this scenario, the $refine$ operator is employed on vertical lists rather than subgroups. The operator is initially utilised on $vl1$ and $vl2$ to produce $vl3$, followed by its application to $vl3$ and $vl4$ in order to derive $vl5$.

Note that both sets of IDs are implemented using bitsets, thus further enhancing efficiency.

3.5 Experiments and Discussion

We conducted performance tests on the VLSD algorithm in order to compare it with other well-known state-of-the-art SD algorithms. The experiments were carried out on a

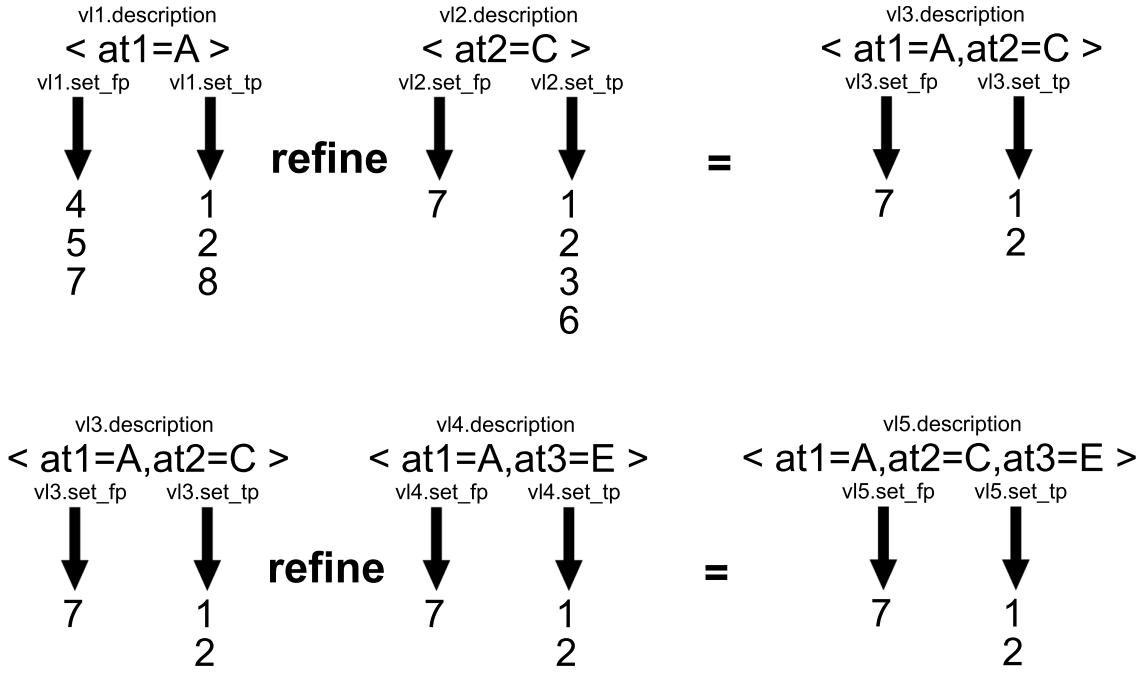


Figure 3.2: Examples of vertical list data structure and adapted *refine* operator.

computer equipped with an Intel Core i7-8700 3.20GHz CPU, 32GB of RAM, running Windows 10 and Anaconda3-2021.11 (x86_64). The implementation was carried out using Python 3.9.7 (64 bits) along with the pandas v1.3.4, numpy v1.20.3, and matplotlib v3.4.3 libraries. We chose these specific libraries because they are a reference in the ML field and have frequently been used and tested by the community. Our proposal was additionally integrated into the `subgroups` python library[1], which will be introduced in Chapter 5.

We employed a set of widely recognised and popular datasets from the existing literature to assess performance. The datasets used in the experiments are presented in Table 3.2, along with their main characteristics. The following preprocessing pipeline was also applied to these datasets: (1) the transformation to nominal type of those attributes that are actually nominal but are represented numerically, (2) the handling of missing values by replacing them with the most frequent value for nominal attributes and the mean value for numerical attributes, and (3) the discretisation of numerical values using the Entropy-based method (Fayyad & Irani, 1993). Additionally, Table 3.3 displays the algorithms used for the performance evaluation, also implemented in the `subgroups` library, along with their respective settings.

[1]Source code available at: `https://github.com/antoniolopezmc/subgroups`

Name	Instances	Attributes	Selectors	Target
car-evaluation	1728	6	21	class = acc
tic-tac-toe	958	10	29	class = positive
heart-disease	918	12	29	HeartDisease = yes
income	899	13	95	workclass = Private
vote	435	17	34	class = republican
lymph	148	19	54	class = malign_lymph
credit-g	1000	21	70	class = good
mushroom	8124	22	118	class = p

Table 3.2: Datasets used and their characteristics

Algorithm	Quality measure	Optimistic estimate	Parameters
VLSD	WRAcc	Expression 3.33	q_threshold and oe_threshold = -1, -0.25, 0, 0.25 both sort criteria = no reorder
SD-Map	WRAcc	-	threshold = -1, -0.25, 0, 0.25 min_support = 0
BSD	WRAcc	Expression 3.33	top-k = 25, 50, 100, 250 min_support = 0 max_depth = maximum
Closed-BSD	WRAcc	Expression 3.33	top-k = 25, 50, 100, 250 min_support = 0 max_depth = maximum
Closed-on-the-positives-BSD	WRAcc	Expression 3.33	top-k = 25, 50, 100, 250 min_support = 0 max_depth = maximum

Table 3.3: Algorithms and settings

After executing each algorithm with each parameter combination with each dataset once, the following metrics were measured: runtime, max memory usage, subgroups selected and nodes visited. The results obtained are shown and explained in this section.

Please recall that the search space of a problem/dataset can be represented visually as a lattice. The levels of this lattice generally correspond to the number of attributes in the dataset, while the nodes at each level represent the unique selectors obtained from the dataset. Two key observations can consequently be made: (1) a larger number of attributes in the dataset leads to a deeper lattice, and (2) a larger number of values in the attributes leads to a greater number of selectors and, therefore, a wider lattice. Moreover, there is a fundamental difference between algorithms that employ pruning based on optimistic estimates and those that do not: while the former may not explore

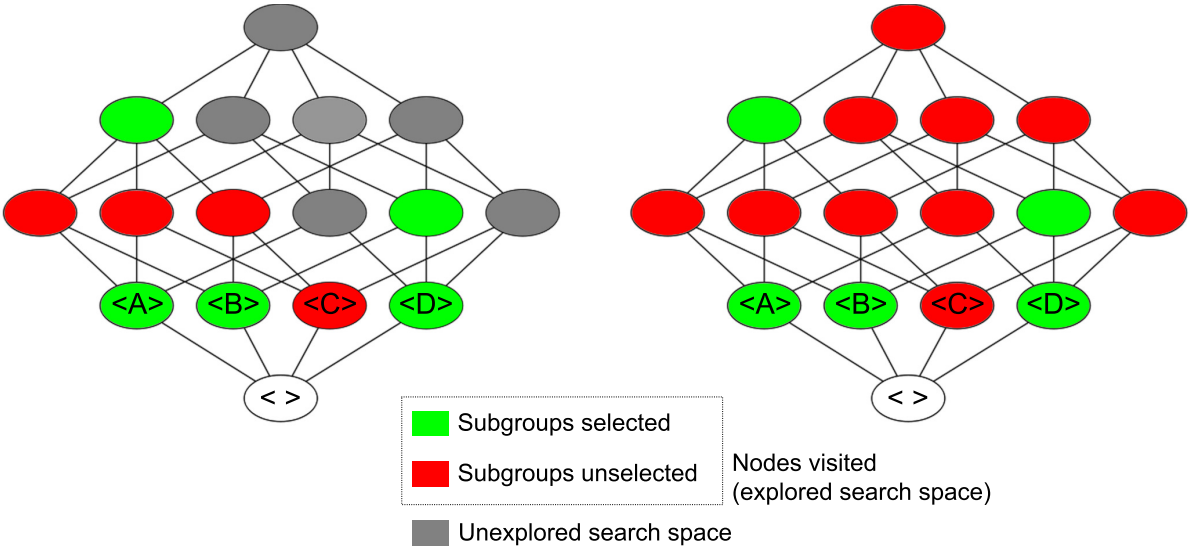


Figure 3.3: Examples of search spaces with (left-hand side) and without (right-hand side) optimistic estimate.

the entire search space, the latter always do. This difference is illustrated in Figure 3.3.

According to the above, it is to be expected that algorithms lacking a pruning method based on optimistic estimates will have exponential runtime and memory usage as the input dataset size increases. Nonetheless, the adoption of optimistic estimates or other pruning techniques could potentially result in lower magnitudes or mitigate the steepness of the exponential trend. This aspect will be examined in detail in the following analysis.

In order to assess the scalability of the VLSD algorithm, the 'mushroom' dataset is employed to measure the runtime (see Figure 3.4) and the maximum memory usage (see Figure 3.5) as the number of attributes (i.e. lattice depth) increases. The evaluation begins with 2 attributes and progressively adds more attributes until there are 22 (all available attributes). Note that all instances are always used.

The evaluation of the scalability of the VLSD algorithm in terms of runtime (Figure 3.4) reveals significant distinctions between datasets with fewer than 20 attributes and those with more than 20 attributes. While the former spend less than 1 hour, the latter require significantly more time. Moreover, employing higher threshold values (with which the search space is not fully explored) leads to a notable decrease in runtime. These outcomes are a consequence of the exponential nature of the data space, as the algorithm explores a data structure that grows exponentially with respect to the number of attributes from the dataset. Furthermore, despite this exponential behaviour, this

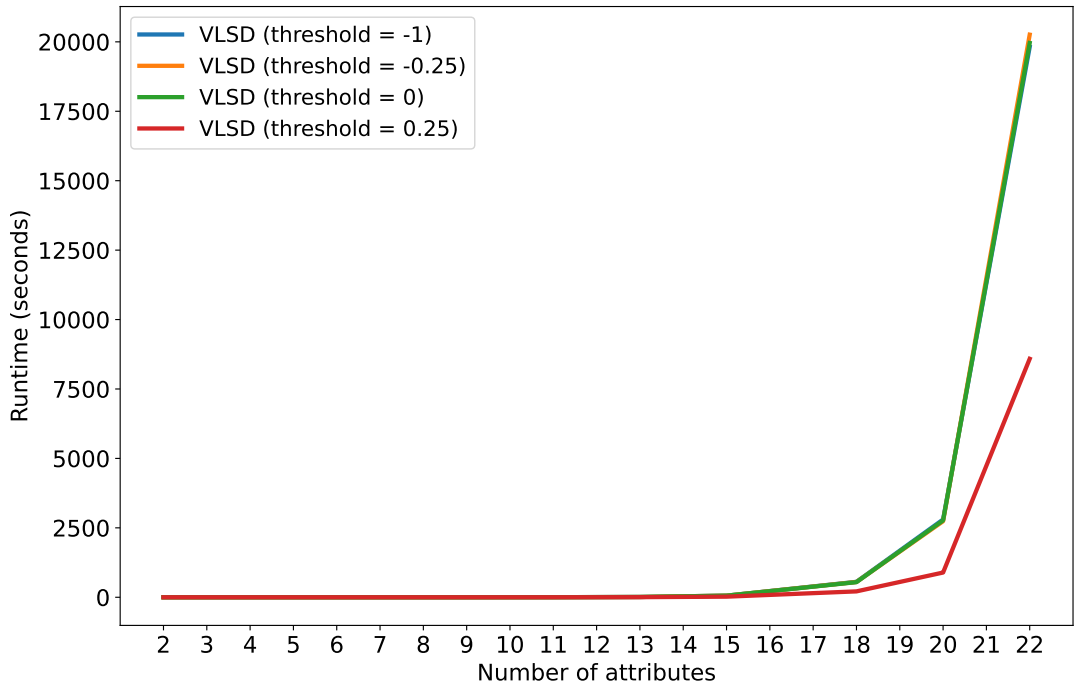


Figure 3.4: VLSD algorithm: runtime of mushroom dataset varying the number of attributes.

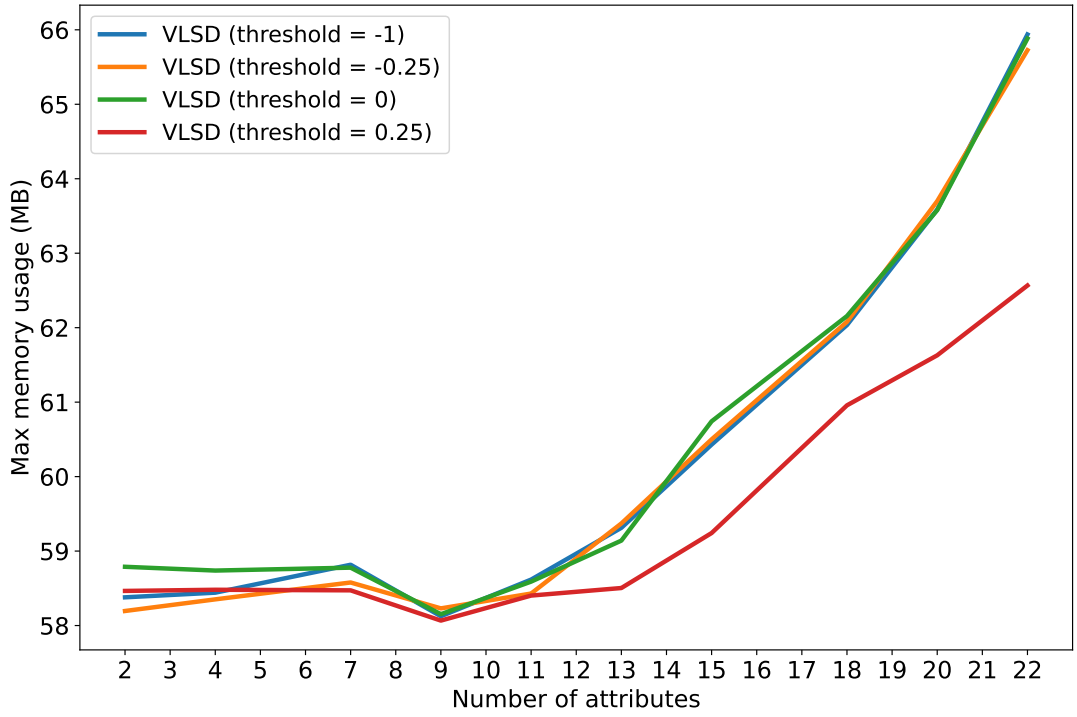


Figure 3.5: VLSD algorithm: max memory usage of mushroom dataset varying the number of attributes.

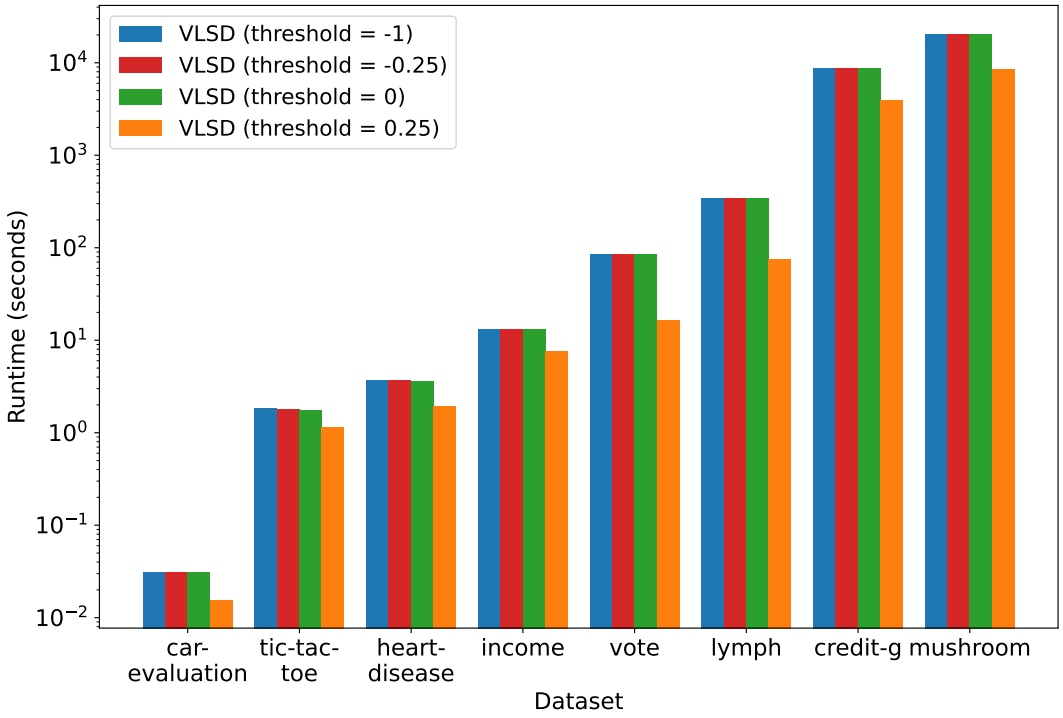


Figure 3.6: VLSD algorithm: runtime for each dataset (logarithmic scale).

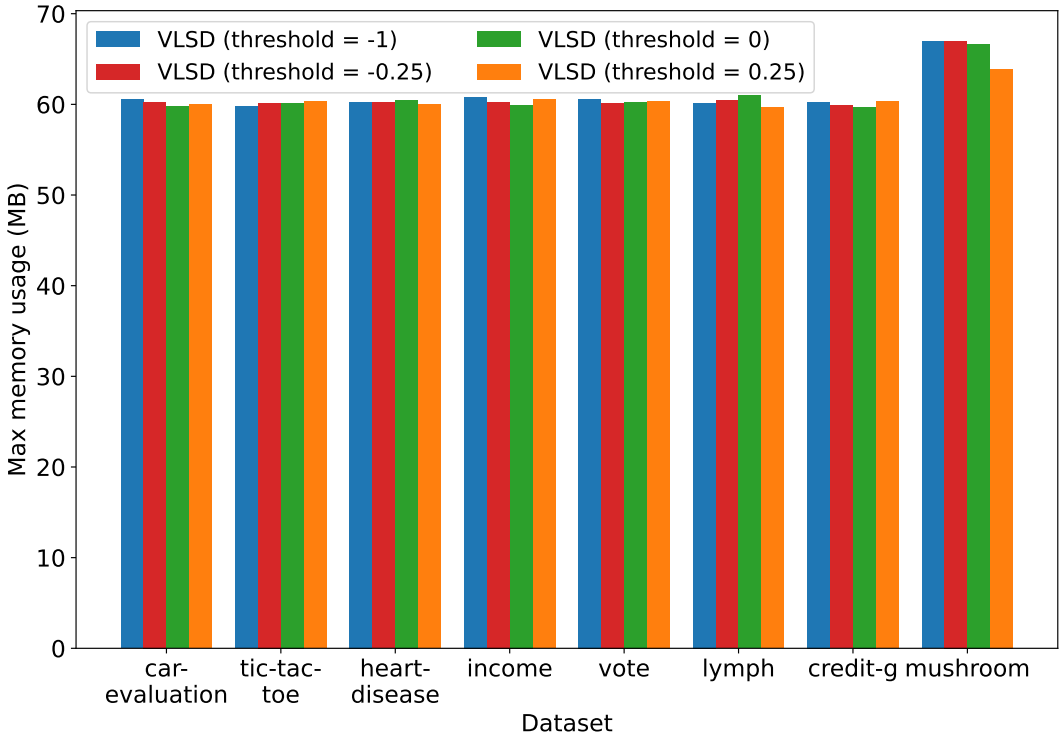


Figure 3.7: VLSD algorithm: max memory usage for each dataset.

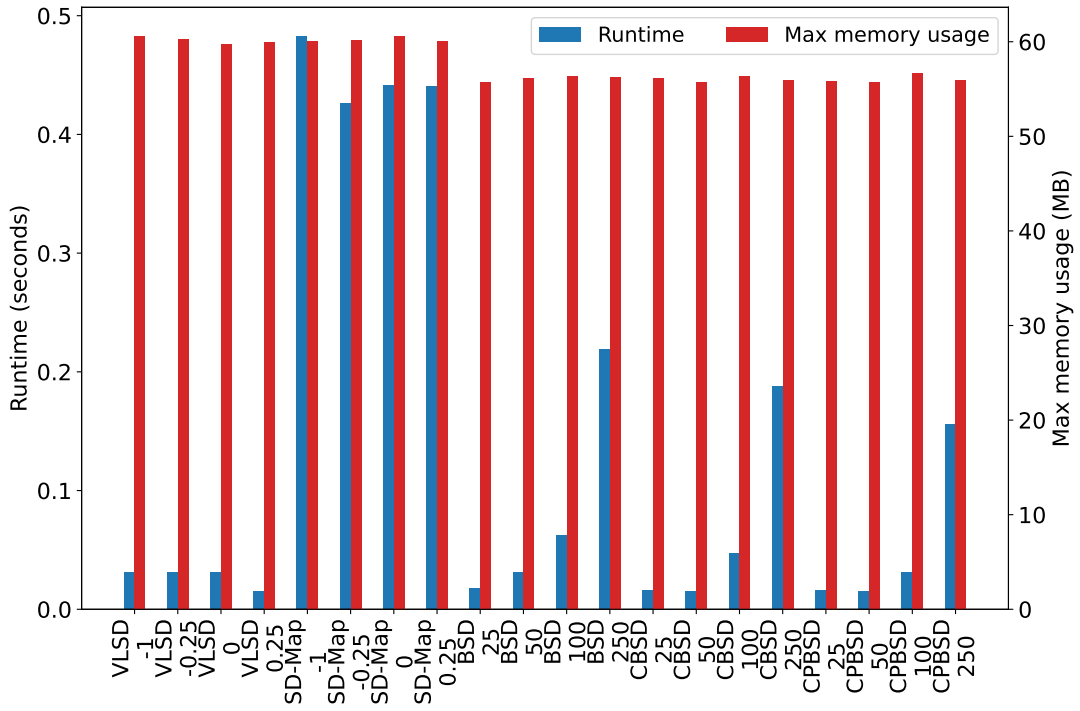


Figure 3.8: Runtime and max memory usage of all algorithms for 'car-evaluation' dataset.

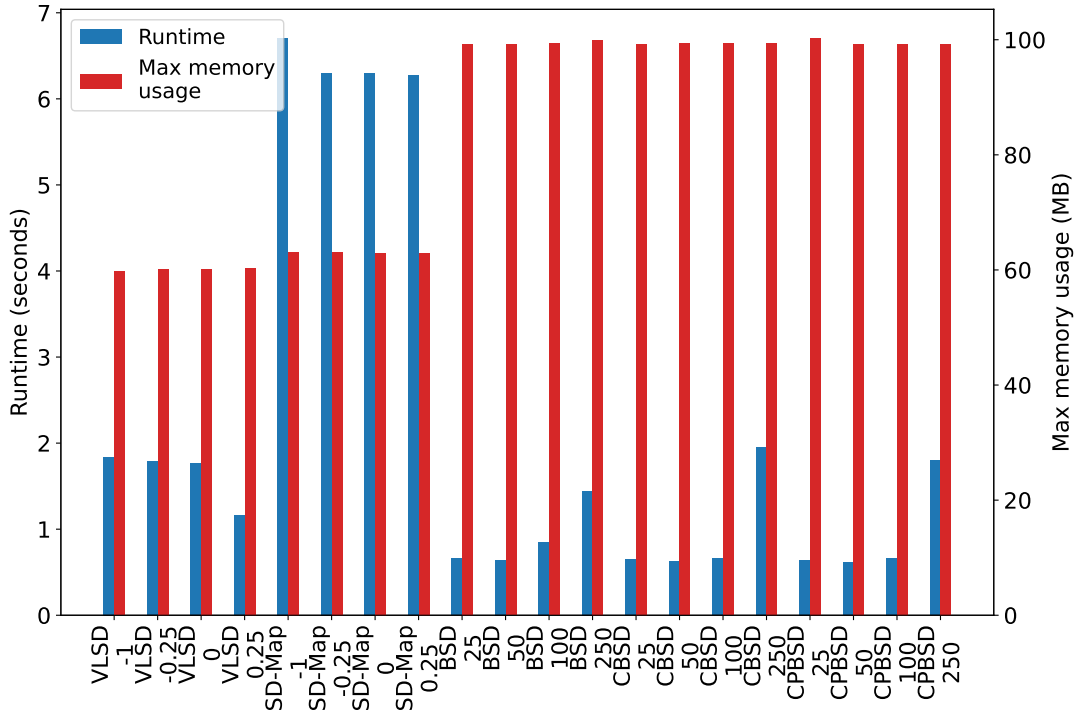


Figure 3.9: Runtime and max memory usage of all algorithms for 'tic-tac-toe' dataset.

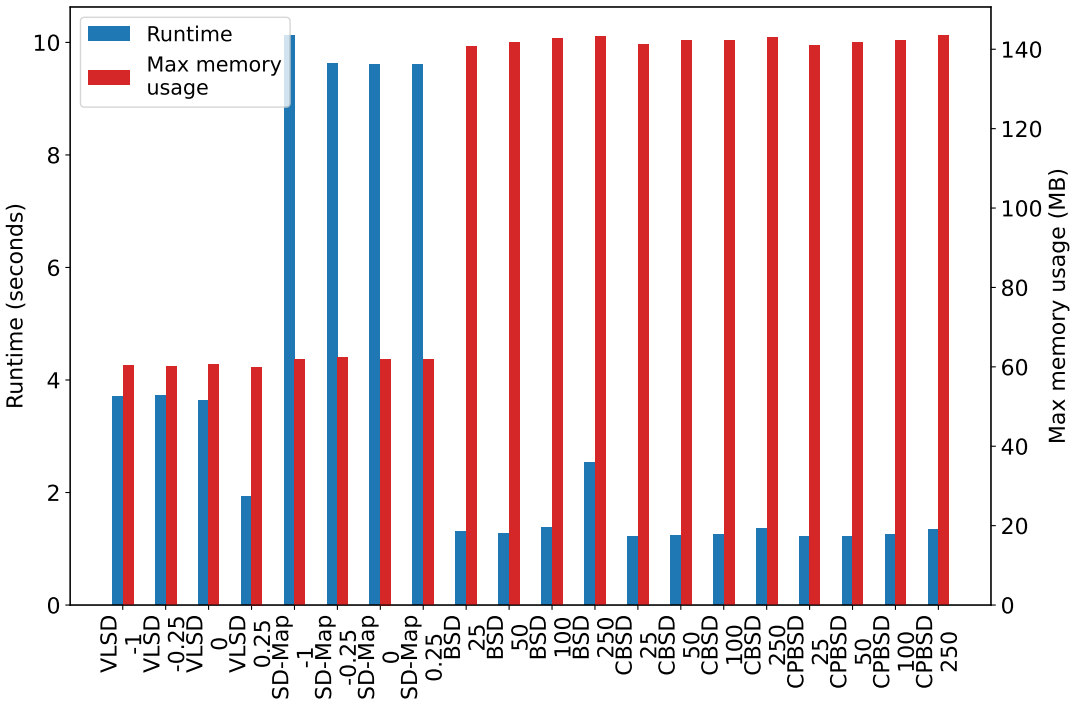


Figure 3.10: Runtime and max memory usage of all algorithms for 'heart-disease' dataset.

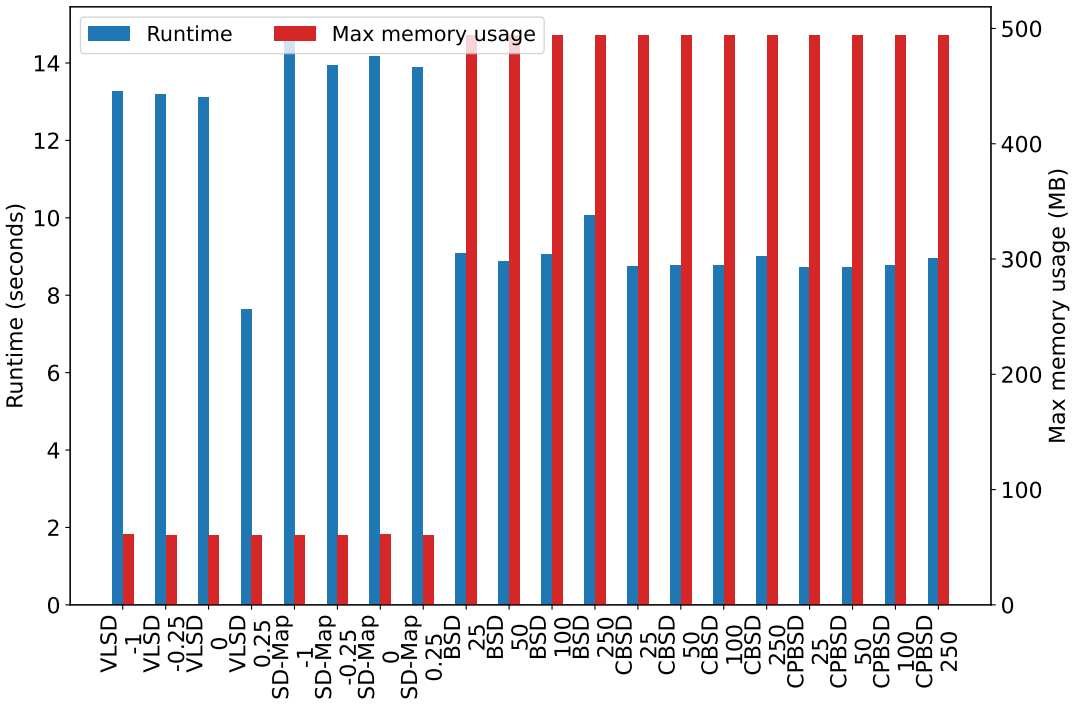


Figure 3.11: Runtime and max memory usage of all algorithms for 'income' dataset.

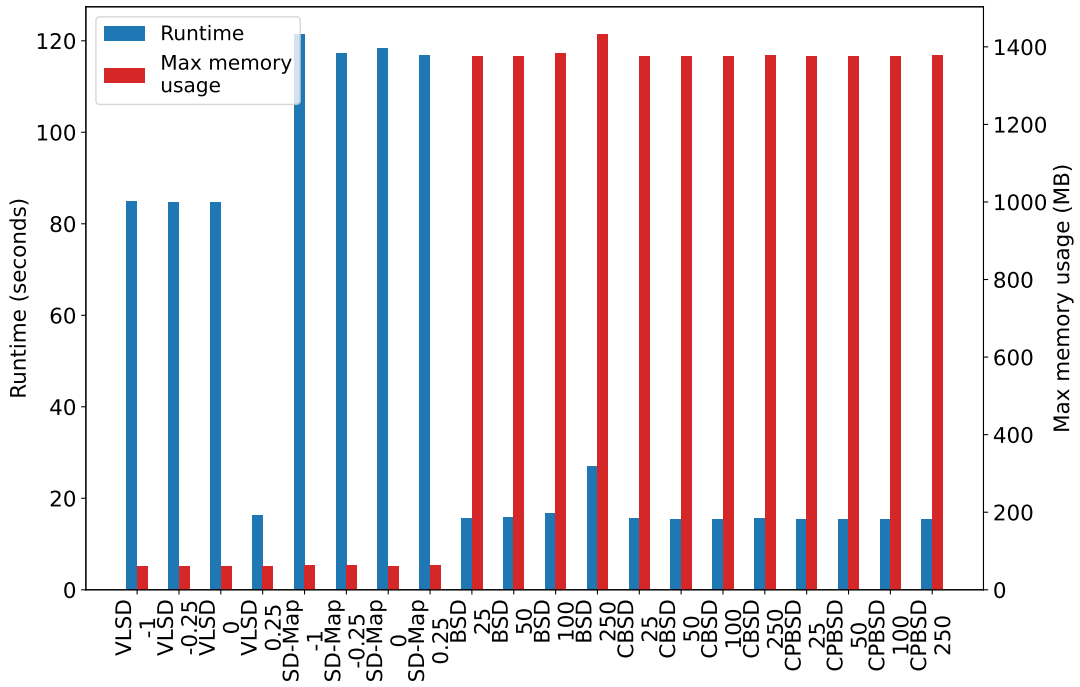


Figure 3.12: Runtime and max memory usage of all algorithms for 'vote' dataset.

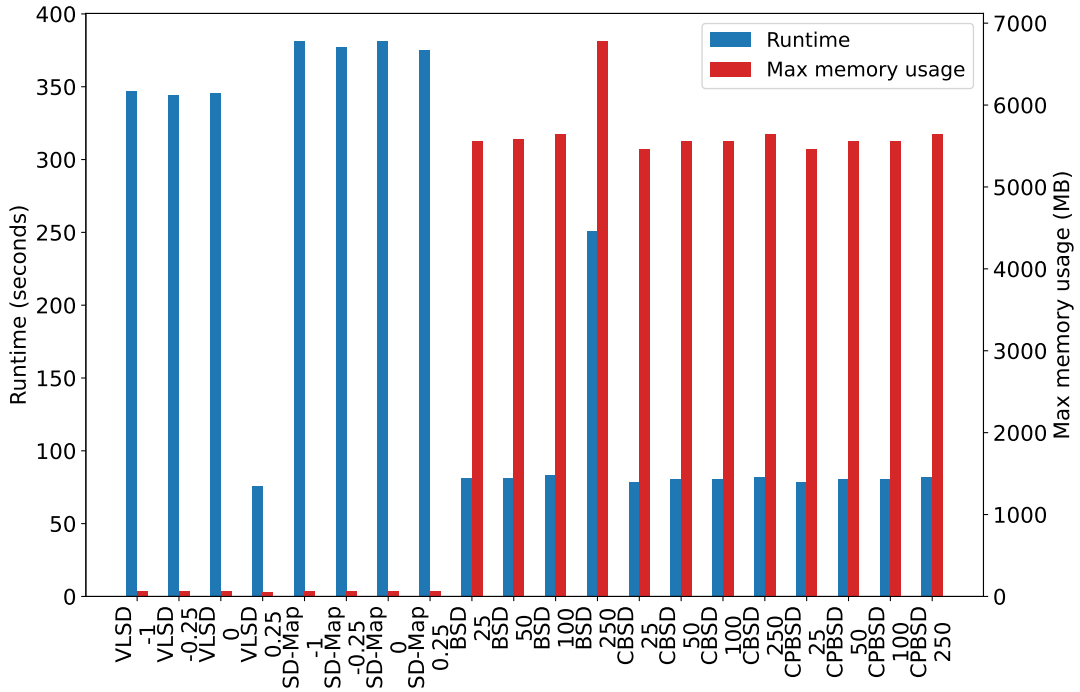


Figure 3.13: Runtime and max memory usage of all algorithms for 'lymph' dataset.

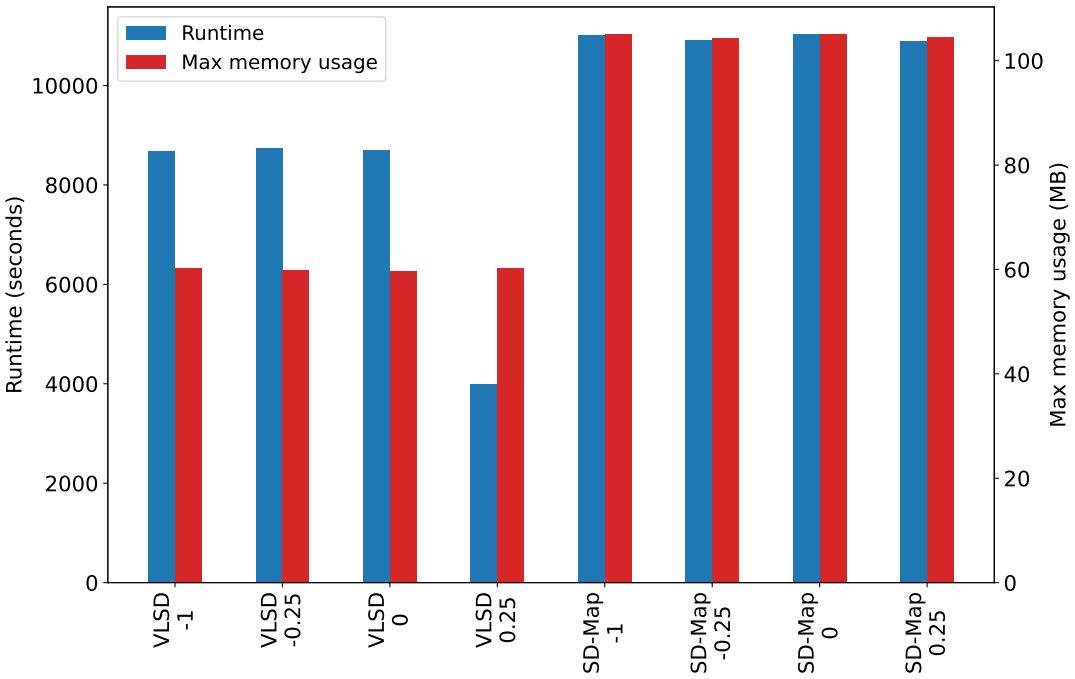


Figure 3.14: Runtime and max memory usage of all algorithms for 'credit-g' dataset.

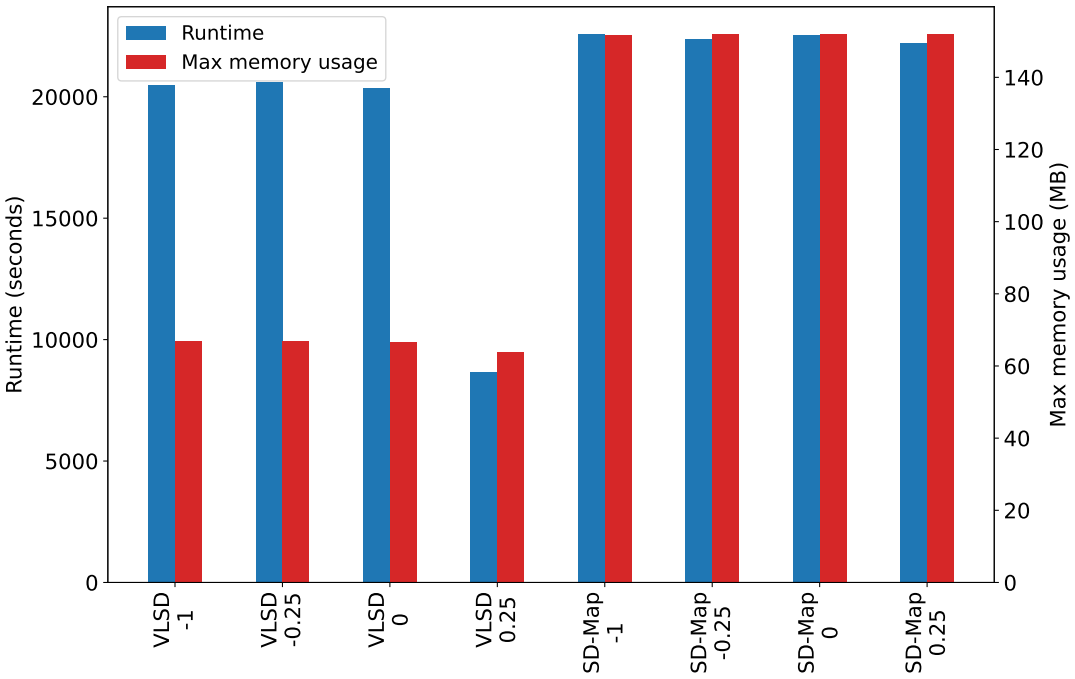


Figure 3.15: Runtime and max memory usage of all algorithms for 'mushroom' dataset.

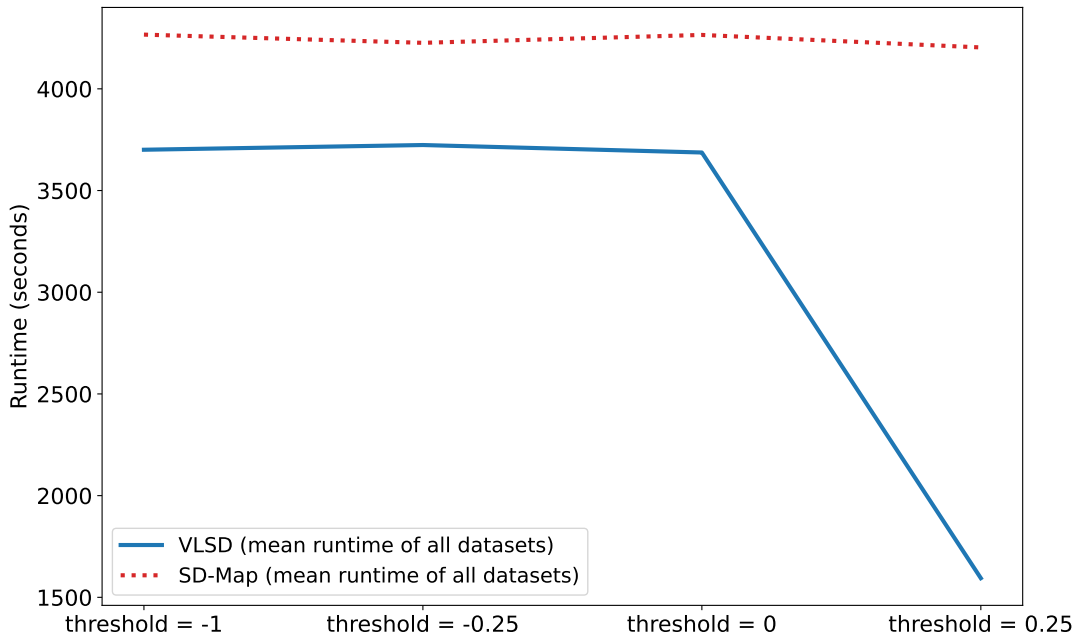


Figure 3.16: Mean runtime of all datasets for each quality threshold.

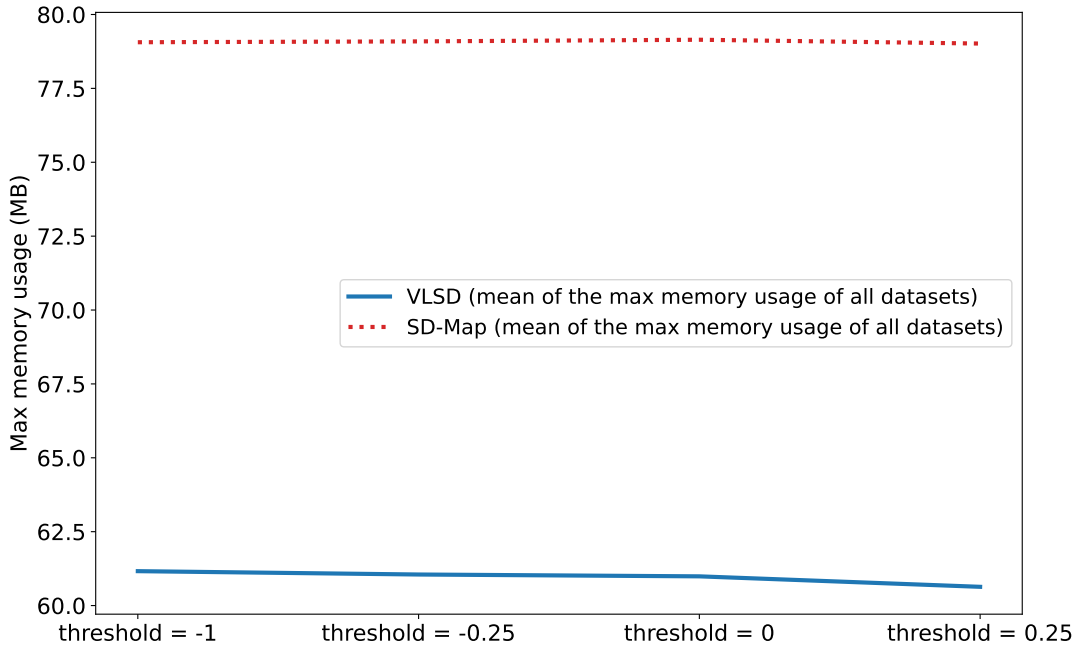


Figure 3.17: Mean of the max memory usage of all datasets for each quality threshold.

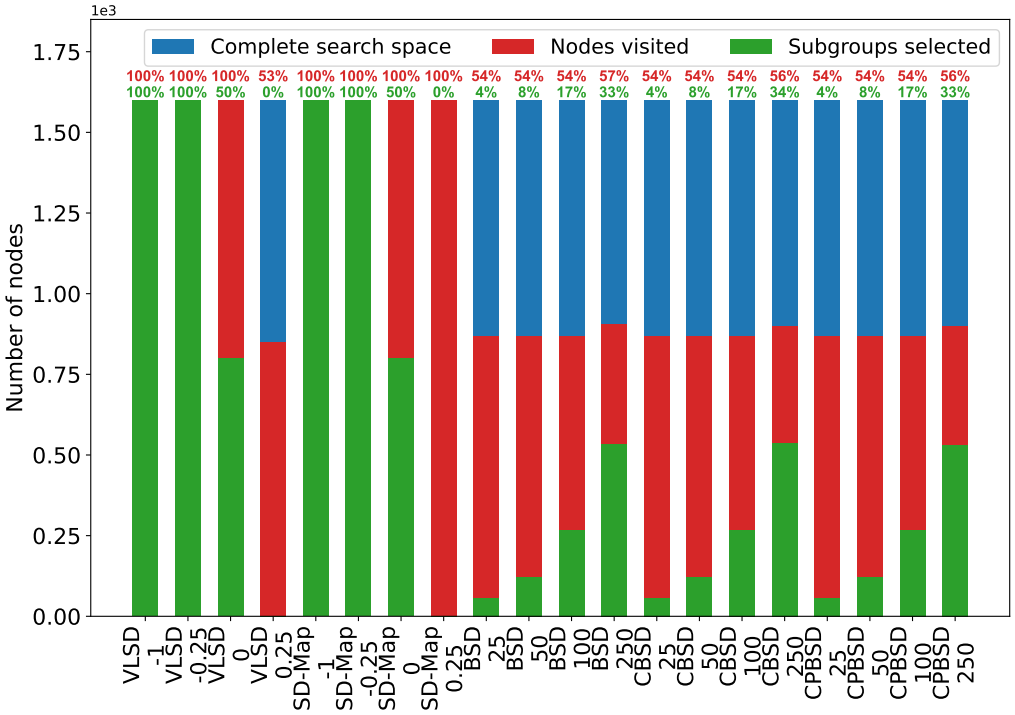


Figure 3.18: Search space nodes of all algorithms for 'car-evaluation' dataset.

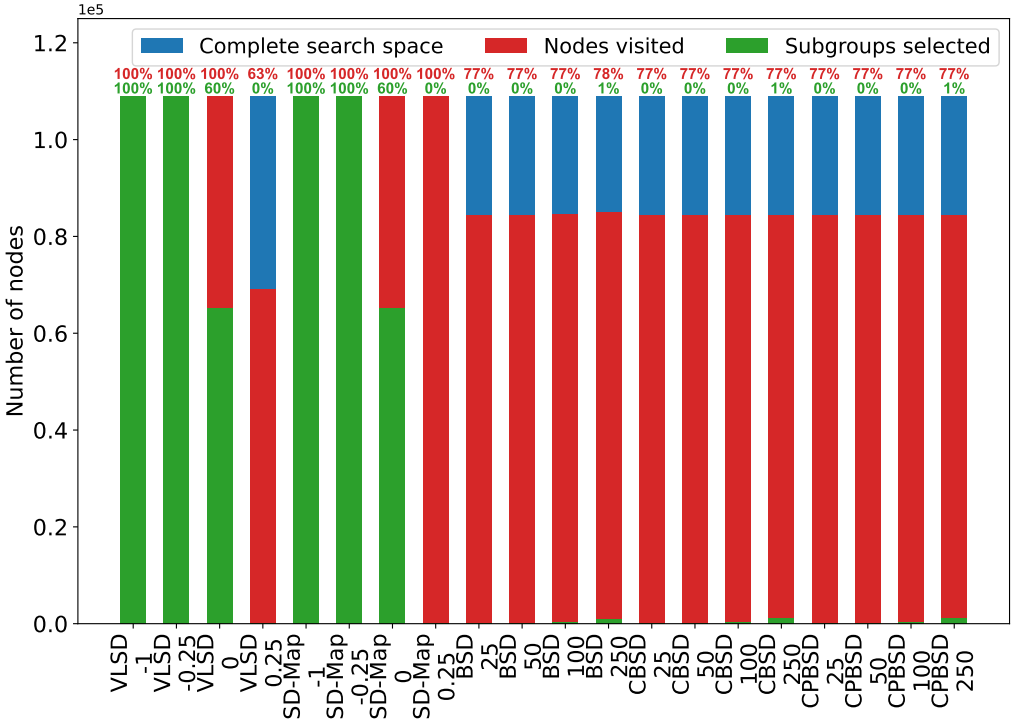


Figure 3.19: Search space nodes of all algorithms for 'tic-tac-toe' dataset.

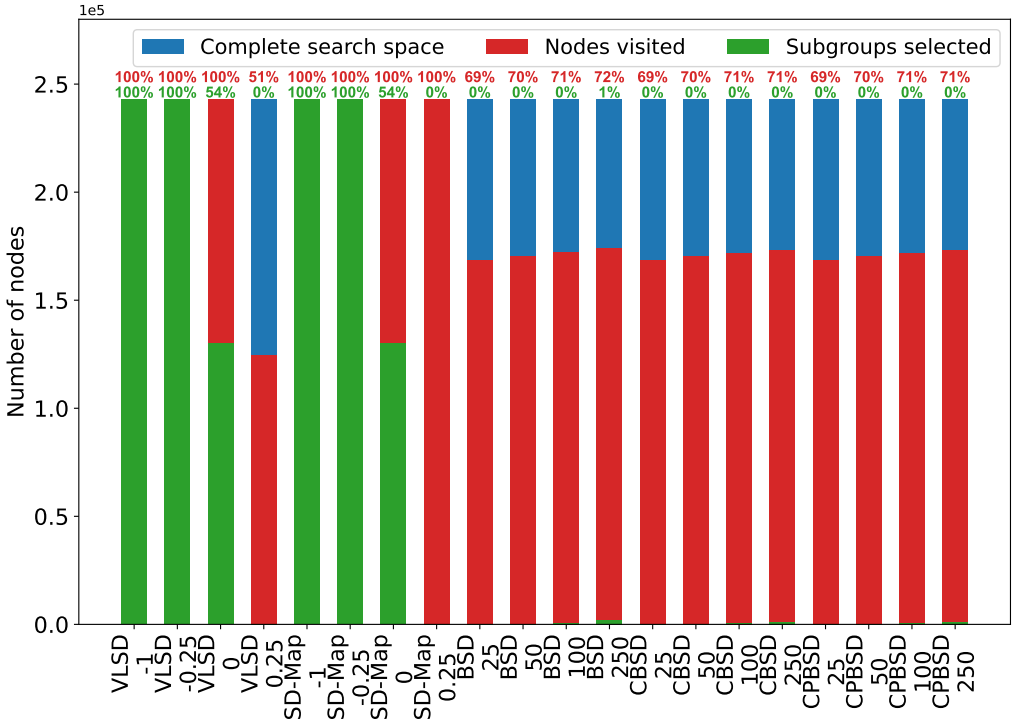


Figure 3.20: Search space nodes of all algorithms for 'heart-disease' dataset.

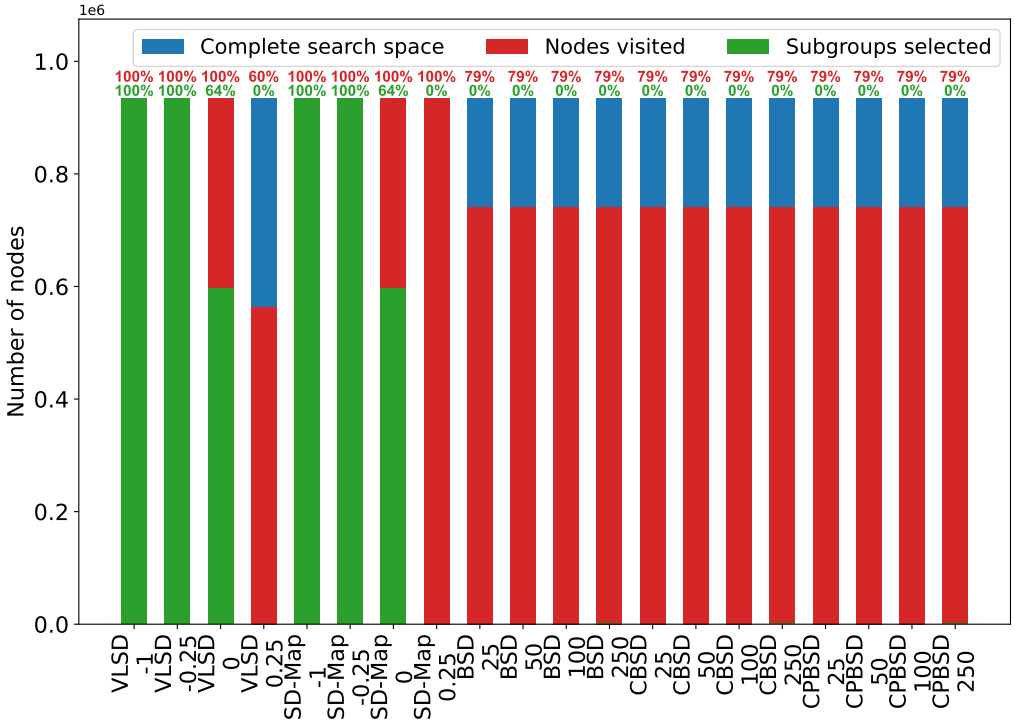


Figure 3.21: Search space nodes of all algorithms for 'income' dataset.

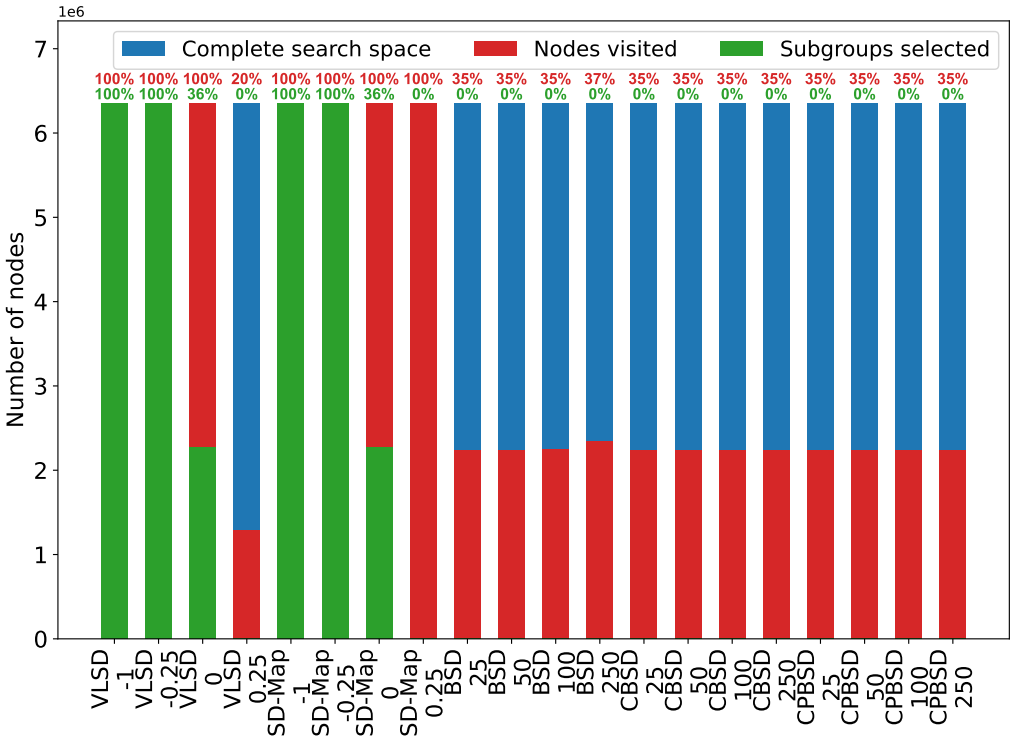


Figure 3.22: Search space nodes of all algorithms for 'vote' dataset.

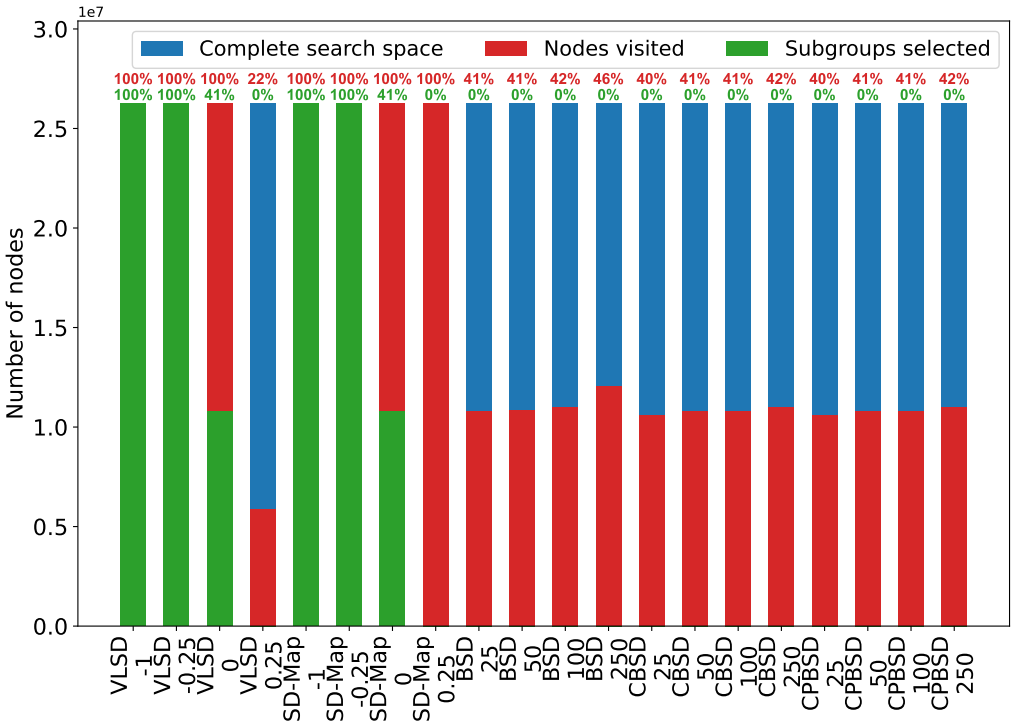


Figure 3.23: Search space nodes of all algorithms for 'lymph' dataset.

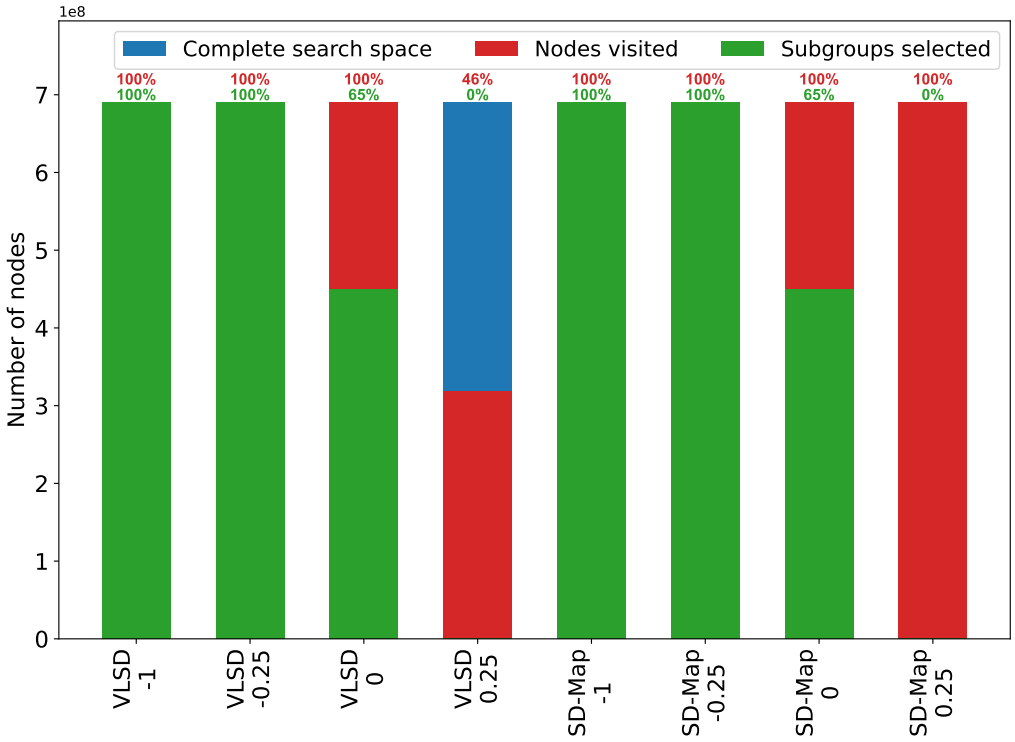


Figure 3.24: Search space nodes of all algorithms for 'credit-g' dataset.

Figure 3.25: Search space nodes of all algorithms for 'mushroom' dataset.

figure also demonstrates that incorporating pruning based on an optimistic estimate mitigates the steepness of the exponential trend. It is noteworthy that the curves corresponding to threshold values of -1, -0.25, and 0 have a similar pattern, while the curve for the 0.25 threshold value has a less pronounced trend.

The evaluation of the scalability of the VLSD algorithm in terms of max memory usage (Figure 3.5) demonstrates that the increase in memory usage relative to the number of attributes is insignificant. Furthermore, higher threshold values lead to a decrease in maximum memory usage, indicating that the algorithm does not need to explore the entire search space. These results are owing to the way in which the algorithm was designed and the utilisation of the equivalence class exploration strategy, both of which enhance its efficiency in terms of maximum memory usage. Unlike the SD-Map algorithm, VLSD does not store the entire search space simultaneously in memory, as it eliminates the regions that have already been explored. This advantage will be explored in greater depth below.

Figures 3.6 and 3.7 provide further support for the observations made earlier regarding the VLSD algorithm. These figures display the runtime and maximum memory usage for each dataset and threshold value, thus reinforcing the results discussed earlier: when the number of attributes increases, the runtime increases, but the max memory usage does not increase significantly.

With regard to the runtime of the SD-Map and VLSD algorithms, Figures 3.8, 3.9, 3.10, 3.11, 3.12, 3.13, 3.14 and 3.15 demonstrate that: (1) the executions of the VLSD algorithm are significantly different (with higher thresholds resulting in shorter execution times); (2) there are no significant differences among the executions of the SD-Map algorithm, which does not employ pruning based on optimistic estimates using different threshold values, as the complete search space is always explored, and (3) both algorithms have an exponential trend, although the overall runtime of VLSD is generally lower than that of SD-Map. These statements are also corroborated by Figure 3.16.

Considering the runtime of the VLSD, BSD, CBSD and CPBSD algorithms, Figures 3.8, 3.9, 3.10, 3.11, 3.12, 3.13, 3.14 and 3.15 indicate that increasing the top-k parameter in the BSD algorithm leads to significant differences, while these increases do not have a significant impact in the CBSD and CPBSD algorithms. The BSD algorithm explores a larger search space when compared to that of the CBSD and CPBSD algorithms, which involve additional pruning techniques for closed and closed-on-the-positives subgroups. When the value of the top-k parameter is increased, the search space consequently expands more moderately in the CBSD and CPBSD algorithms. As a result, the runtime

of the BSD algorithm undergoes a more significant increase when compared to that of the CBSD and CPBSD algorithms. It will also be observed that: (1) when the VLSD algorithm uses lower threshold values, thus exploring more search space, its runtime is significantly higher than that of the BSD, CBSD, and CPBSD algorithms, and (2) when the VLSD algorithm uses higher threshold values, thus exploring less search space, there are no significant differences in their runtimes.

With regard to the max memory usage of the SD-Map and VLSD algorithms, Figures 3.8, 3.9, 3.10, 3.11, 3.12, 3.13, 3.14 and 3.15 demonstrate that: (1) there are no significant differences among the executions of the VLSD algorithm owing to its efficient design and the utilisation of the equivalence class exploration strategy (although not all cases explore the complete search space, memory usage is generally reduced); (2) there are no significant differences among the executions of the SD-Map algorithm when using different threshold values since the entire search space is always stored in the FPTree data structure, and (3) there are significant differences between the two algorithms, as clearly illustrated in Figures 3.14 and 3.15. The information depicted in Figure 3.17 demonstrates that the average maximum memory usage across all datasets for each quality threshold value is consistently over 20% higher when employing the SD-Map algorithm.

Figures 3.8, 3.9, 3.10, 3.11, 3.12, 3.13, 3.14 and 3.15 demonstrate consistent behaviour in terms of max memory usage for the VLSD, BSD, CBSD, and CPBSD algorithms and for the same reasons as in the previous case. These algorithms generally consume significantly more memory when compared to the VLSD and SD-Map algorithms, and it is for this reason that executing them with the last two datasets was not feasible owing to their high memory requirements.

When focusing on the search space nodes of the VLSD algorithm (Figures 3.18, 3.19, 3.20, 3.21, 3.22, 3.23, 3.24 and 3.25), although this algorithm uses pruning based on an optimistic estimate and may not explore certain regions in the search space with lower quality, it ensures the discovery of the best subgroups since it is exhaustive.

With regard to the search space nodes of the VLSD and SD-Map algorithms, Figures 3.18, 3.19, 3.20, 3.21, 3.22, 3.23, 3.24 and 3.25 demonstrate that both algorithms always obtain the same number of subgroups across different datasets and threshold values. This demonstrates the accurate design and implementation of VLSD in contrast to SD-Map, which is an exhaustive algorithm without an optimistic estimation. The aforementioned figures also support the use of a pruning mechanism based on an optimistic estimate since, while VLSD does not always explore the entire search space, SD-Map always does

so. It should be noted that VLSD examines fewer nodes as the threshold value increases.

Figures 3.18, 3.19, 3.20, 3.21, 3.22, 3.23, 3.24 and 3.25 illustrate that the BSD, CBSD, and CPBSD algorithms have the following characteristics when compared to the VLSD and SD-Map algorithms: (1) they do not explore the entire search space owing to the implementation of a pruning technique based on optimistic estimates (in the same way as occurs with VLSD), and (2) they select significantly fewer subgroups because they implement additional pruning mechanisms based on relevant subgroups, closed subgroups and closed-on-the-positives subgroups.

One detail that has a slight impact on the experiments is that the bitsets utilised in the VLSD algorithm differ from those employed in the BSD, CBSD, and CPBSD algorithms. While our algorithm incorporates all dataset instances in both bitsets, the others utilise bitsets of varying sizes. Our approach produces the flexibility required in order to compute any quality measure with the fewest number of operations possible at a low cost as regards memory.

In summary, the comparison between the VLSD and SD-Map algorithms reveals that the fact that the VLSD algorithm employs an optimistic estimate-based pruning technique has a noticeable impact. This pruning strategy enables the VLSD algorithm to achieve reduced time and memory requirements while exploring fewer nodes, without compromising exhaustiveness or subgroup generation. Moreover, when comparing the VLSD algorithm with the BSD, CBSD, and CPBSD algorithms, it becomes evident that the latter three algorithms significantly lag behind in terms of maximum memory usage. However, overall, the BSD, CBSD, and CPBSD algorithms have shorter execution times and select fewer nodes owing to the pruning techniques based on optimistic estimates, relevant subgroups, closed subgroups, and closed-on-the-positives subgroups.

3.6 Conclusions

The purpose of this study was to develop and apply an advanced SD algorithm that surpasses existing methods in terms of efficiency. Our proposed solution is the VLSD algorithm, which incorporates a new data structure denominated as a vertical list that is based on the work developed by Zaki et al. (1997). This algorithm operates by employing an equivalence class exploration strategy and utilises a pruning technique based on optimistic estimation.

It is necessary to highlight that although each of the elements used in this proposal has been discussed individually in existing literature, this is, to the best of our knowledge,

the first time that they have been combined, executed and verified together.

Certain existing SD algorithms, such as SD-Map or BSD, have incorporated traditional data structures such as FPTree or Bitsets. In contrast, our algorithm employs a vertical list data structure that simultaneously represents a subgroup and the dataset instances it encompasses. It additionally provides a straightforward and effective method for the computation of subgroup refinements and quality measures. The VLSD algorithm can be easily parallelised thanks to the utilisation of the equivalence class exploration strategy and the aforementioned data structure.

We conducted experiments using widely recognised and commonly used datasets from literature. Our analysis focused on several metrics: runtime, max memory usage, subgroups selected and nodes visited. The results consistently demonstrated that our approach outperforms the alternative algorithms in terms of efficiency.

There are several potential directions for future research into this algorithm. Firstly, adjustments could be made in order to eliminate the requirement of extracting all explored subgroups, such as focusing on extracting only the top-k subgroups, and lastly, additional pruning strategies could be incorporated so as to further enhance the efficiency of the VLSD algorithm, such as considering closed subgroups or closed-on-the-positives subgroups.

CHAPTER 4

Diverse top-k Subgroup List Discovery

This chapter shows a formal definition of the new problem of mining diverse top-k subgroup lists and applies it to patient phenotyping. This new approach is based on the Subgroup Discovery (SD) technique, the subgroup list model and the Minimum Description Length (MDL) principle. The results generated by this technique are easy for clinicians to read since they are multiple subgroup lists, each formed of a collection of subgroups.

4.1 Motivation

As explained in Chapter 1, the objective of this PhD thesis is to propose different Machine Learning (ML) techniques and methods with which to find sets of patients with interesting characteristics. These patient phenotypes might refer to different individuals from the population, thus forming a single explanation of the input dataset. However, obtaining a single interpretation of the population studied might, in some cases, be useless to the eyes of a clinical expert, thus providing a limited view of the medical phenomenon studied. Moreover, there is the possibility of valuable characteristics and descriptions being ruled out. For example, a phenotype automatically generated by a particular ML algorithm might not make sense to clinicians and consequently be discarded, signifying that the discovery of multiple phenotypes would be useful in such cases. Moreover, when generating multiple phenotypes, both the characteristics used and the patients covered must be diverse in order to provide experts with different explanations/interpretations for the same medical dataset. It is, therefore, definitively necessary to propose a new

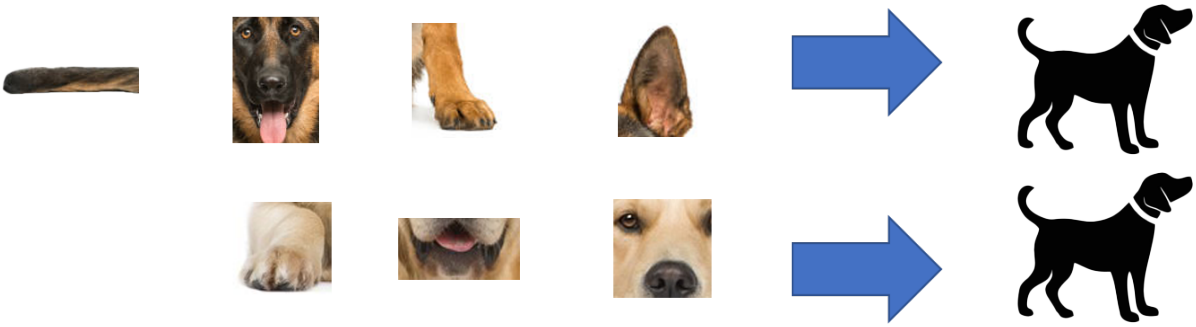


Figure 4.1: Example of two phenotypes in the form of two subgroup lists that explain the target “dog”.

approach with which to mine diverse top-k patient phenotypes.

In this context, the SD technique (explained in Chapter 3) can be used as a building block of this new approach. However, one disadvantage of SD is that only sets containing a few subgroups can be easily interpreted by experts. It is, therefore, possible to use the subgroup list model to solve this problem. An example of two phenotypes in the form of two subgroup lists is shown in Figure 4.1. These phenotypes represent different characteristics with which to describe the dog population, and their descriptions are diverse since they contain different observable features such as colour or anatomical parts.

It is essential to emphasize that another great difficulty when creating phenotypes in the form of subgroup lists is the large number of candidates mined by the SD algorithm owing to the pattern explosion problem. One feasible solution by which to solve this problem is that of using the MDL principle to guide the generation of the subgroup lists.

In summary, the three elements enumerated are used together in this chapter to create a new approach with which to generate readable patient phenotypes in the form of multiple subgroup lists in the context of the antibiotic resistance problem, thus contributing to proving our hypothesis.

The main contributions of this chapter are the following:

- We formally define the new problem of mining diverse top-k patient phenotypes in the form of multiple subgroup lists.

- We propose two new algorithms that tackle this problem by using SD, the subgroup list model and the MDL principle. These are called GMSL (Generation of Multiple Subgroup Lists) and DSLM (Diverse Subgroup Lists Miner).

The remainder of this chapter is structured as follows: Section 4.2 provides a background to the new elements introduced in this approach, i.e. the subgroup list model and the MDL principle, while Section 4.3 describes the new problem of mining diverse top-k patient phenotypes in the form of multiple subgroup lists. Sections 4.4 and 4.5 show and explain both of the algorithms proposed in this chapter (the GMSL and DSLM algorithms, respectively), along with the experiments carried out with them, the results obtained after these experiments had been completed and the discussions of those results. Finally, Section 4.6 provides the conclusions reached after carrying out this research.

4.2 Background

After providing a background to the SD technique in Chapter 3, this section provides the state of the art of the new elements introduced in this chapter, which complement the SD technique with which to mine diverse top-k patient phenotypes. These are the subgroup list model and the MDL principle.

With regard to the subgroup list model, it was initially proposed by Proença, Grünwald, Bäck, and Leeuwen (2021) and was further expanded and detailed by Proença, Grünwald, Bäck, and van Leeuwen (2022). A subgroup list consists of a collection of ordered subgroups followed by a default rule (also called a default subgroup). Its purpose is to iteratively divide input data into various subsets and to provide a distinct description for each subset, with the exception of the last one, which represents the default subgroup. In this model, these individual subgroups cover instances that are statistically different and interesting when compared to the overall dataset distribution, while the default subgroup represents the dataset average and covers instances that align well with the dataset distribution. Each instance in the input dataset can consequently be covered only by either an individual subgroup or the default subgroup, but not both. For example, if a subgroup list contains 7 subgroups, the input dataset is partitioned into 8 subsets: the first 7 correspond to the individual subgroups, and the last one represents the default subgroup. Figure 4.2 provides an example of the subgroup list model.

The main difference between the subgroup list model and the traditional approach that generates a set of independent subgroups is that, while the subgroups in a set are unordered and independent of each other, the subgroups contained in a subgroup list are ordered and dependent on each other, since the inclusion of one subgroup in a subgroup list is conditioned by the previous subgroups already added.

A subgroup list can be interpreted as a decision list, since it is a collection of the form

$subgroup_1$:	IF	$description_1$	THEN	$distribution_1\,(target)$
$subgroup_2$:	ELSE IF	$description_2$	THEN	$distribution_2\,(target)$
		$\vdots$		
$subgroup_w$:	ELSE IF	$description_w$	THEN	$distribution_w\,(target)$
$dataset$:	ELSE			$distribution(target)$

Figure 4.2: Example of subgroup list with w subgroups.

"else-if", meaning that a subgroup from a subgroup list is attained only if the previous subgroups are not true. Another existing model is the subgroup set (Lakkaraju, Bach, & Leskovec, 2016), which is an unordered collection of subgroups of the form "if", and can be interpreted as a decision set. The key advantage of a subgroup list with respect to a subgroup set is that the former is easier for clinicians to interpret since it is ordered and prioritises the explanations. We believe that the subgroup list model is a more convenient means to represent phenotypes and that the generation of multiple subgroup lists might be a suitable approach with which to mine and represent diverse phenotypes.

The MDL principle (Grünwald, 2007) is an inductive inference method based on the idea that the best explanation of the data is that which best compresses the data. In the context of SD and subgroup lists, the utilisation of the MDL principle is equivalent to performing a Bayesian statistical test and multiple hypothesis testing correction for every subgroup (Proença et al., 2021, 2022), which provides a statistically solid foundation for the approach.

The MDL encoding for the optimal subgroup list generated from a specific dataset was introduced by Proença et al. (2021, 2022). In this encoding scheme, when considering a dataset d, a subgroup list model M, and a candidate subgroup s, the best subgroup to be included in a single subgroup list is determined by maximising the compression gain. The compression gain is defined as follows:

$$\Delta_\beta L(d, M \oplus s) = \frac{L(d,M) - L(d, M \oplus s)}{(n_s)^\beta} + \frac{L(M) - L(M \oplus s)}{(n_s)^\beta} \tag{4.1}$$

In this formula, the $\oplus$ operator represents the action of adding s to the end of M (before the default subgroup), and n_s denotes the number of instances covered by the description of s. Although more detailed information and explanations regarding Equation 4.1, $\Delta_\beta L$ and the β parameter can be found in Proença et al. (2021, 2022), the underlying idea can be summarised as follows: (1) maximising $\Delta_\beta L$ corresponds to

finding a candidate subgroup that optimises the Bayesian proportions test between the subgroup distribution and the dataset distribution, while penalising larger descriptions; (2) $\Delta_\beta L > 0$ indicates stronger statistical evidence in favour of including the candidate subgroup in the list rather than excluding it; and (3) β values close to zero give priority to candidate subgroups that cover a greater number of instances, while β values close to one prioritise candidate subgroups that cover fewer instances.

The problem of finding an optimal subgroup list poses a challenge owing to its NP-hard nature. In order to address this, Proença et al. (2021, 2022) introduced a greedy approach known as SDD++. This method involves adding subgroups to the subgroup list one at a time, following the last subgroup and preceding the default subgroup. In essence, the algorithm begins with an empty subgroup list and gradually incorporates subgroups until no further compression can be gained, as determined by Equation 4.1. Lastly, the algorithm outputs the subgroup list mined.

Focusing on the clinical field, Giurcaneanu, Mircean, Fuller, and Tabus (2006); Glaab, Bacardit, Garibaldi, and Krasnogor (2012); Glueck et al. (2018) show the application of the MDL principle to this area and for the patient phenotyping process.

Finally, we refer the reader to Grünwald (2007); Proença et al. (2021, 2022) for more detailed explanations of the MDL principle, its utilisation along with the SD technique, and the subgroup list model.

4.3 Definition of problem

In this section, we formalise the problem of mining of diverse top-k patient phenotypes. Since the necessary formal definitions concerning the SD technique have already been shown in Chapter 3, we shall continue extending and providing more details related to the elements introduced in this chapter.

In Chapter 3, a selector e was informally defined as a binary relation between an attribute from a dataset and a value in the domain of that attribute. In the context of the approach presented in this chapter, descriptions contained in a subgroup list are, therefore, related to a specific target value represented by a selector e.

After detailing this concept, the following definitions can be provided:

Definition 4.1 (Positive instance i_{pos})**.** Given a dataset d, a pattern p and a target e, then an instance i from d is positive (denoted as i_{pos}) if it is covered by both p and e, i.e. if $i \sqsubset p$ and $i \sqsubset e$.

We consequently say that a pair (p,e) is positive if there is an instance i from d covered by both p and e (i.e. i is positive).

Definition 4.2 (Negative instance i_{neg})**.** Given a dataset d, a pattern p and a target e, then an instance i from d is negative (denoted as i_{neg}) if it is covered by p, but not by e, i.e. if $i \sqsubset p$ and $i \sqsupset\!\!\!\!\!+ e$.

The concept of phenotype used in the approach proposed in this chapter is defined as follows:

Definition 4.3 (Phenotype l)**.** Given a dataset d and a target e, then a phenotype l is a list of patterns $< p_1,\ldots,p_z >$ such that $\forall p_z \in l,(p_z,e)$ is positive, and that $\forall p_z \in l$, p_z covers dataset instances that are statistically different with respect to the dataset distribution.

Different measurements are used for each pattern contained in a phenotype: (1) the number of positive and negative instances covered by the pattern individually (without considering its position in the subgroup list); (2) the number of positive and negative instances with which the pattern contributes to the phenotype (considering its position in the subgroup list), and (3) the number of positive and negative instances covered by the phenotype to that point (the cumulative sum of the previous measurement).

Furthermore, in our proposed scenario, multiple phenotypes must include positive pattern-target pairs, since our goal is to explain a specific target value from a dataset (e.g., exitus = yes). It is consequently useless for the descriptions generated not to be related to this target value.

When mining multiple phenotypes represented by multiple subgroup lists, two desirable properties are “top-k” and “diversity”. Firstly, the exhaustive generation of all possible phenotypes is not feasible owing to the limited computational capacity, signifying that only the top-k phenotypes can be mined. Secondly, diversity plays a notable role in this context, as clinicians are provided with multiple explanations of the same dataset. These multiple explanations must, therefore, be non-redundant and different. There are two ways in which to achieve diversity: (1) by considering coverage and (2) by considering descriptions. Diversity in terms of coverage is relevant when mining a single phenotype l_x, necessitating the minimisation of dataset instances already covered by previous descriptions within l_x. Given two patterns, $p_a \in l_x$ and $p_b \in l_x$, the instances simultaneously covered by both patterns should, therefore, be kept to a minimum. Diversity in terms of descriptions pertains to employing different selectors and patterns in the different

phenotypes mined in order to ensure multiple explanations of the same target value. For any two phenotypes l_x and l_y, it therefore follows that $\forall p_a, if\ p_a \in l_x, then\ p_a \notin l_y$.

With respect to the diversity in terms of coverage, an overlap factor can be used in order to minimise the number of dataset instances simultaneously covered by a phenotype l_x (i.e. patterns $< p_1, \ldots, p_z >$ from l_x) and a candidate pattern p_{cand}. This is defined as follows:

Definition 4.4 (Overlap factor of)**.** Given a phenotype l_x, which contains a list of patterns $< p_1, \ldots, p_z >$, and a candidate pattern p_{cand}, then the overlap factor of is a real number in the domain $[0,1]$ such that values close to 0 mean a low overlap between l_x and p_{cand} and values close to 1 mean a high overlap between l_x and p_{cand}.

An example of an overlap factor is shown in Equation 4.2. This function computes the proportion between the number of instances simultaneously covered by each pattern from l_x and the candidate pattern p_{cand}, and the number of instances covered by each pattern from l_x.

$$of(l_x, p_{cand}) = \frac{|\{\forall i_x \in d, i_x \sqsubset p_1 \wedge i_x \sqsubset p_{cand}\}| + \ldots + |\{\forall i_x \in d, i_x \sqsubset p_z \wedge i_x \sqsubset p_{cand}\}|}{|\{\forall i_x \in d, i_x \sqsubset p_1\}| + \ldots + |\{\forall i_x \in d, i_x \sqsubset p_z\}|} \tag{4.2}$$

Note that this overlap factor has the following two properties:

$$of = 0 \iff \{\forall i_x \in d, i_x \sqsubset p_1 \wedge \ldots \wedge i_x \sqsubset p_z\} \neq \{\forall i_x \in d, i_x \sqsubset p_{cand}\} \tag{4.3}$$

$$of = 1 \iff \{\forall i_x \in d, i_x \sqsubset p_1 \wedge \ldots \wedge i_x \sqsubset p_z\} = \{\forall i_x \in d, i_x \sqsubset p_{cand}\} \tag{4.4}$$

The overlap factor is computed using the instances covered by the patterns individually considered (without considering its position in the subgroup list). It can also be used separately for the positive instances and for the negative instances, having a positive overlap factor and a negative overlap factor, respectively.

Finally, the problem of mining diverse top-k phenotypes is defined as follows:

Definition 4.5 (Mining diverse top-k phenotypes problem)**.** Given a dataset d, a target e and the k maximum number of phenotypes to be discovered, the problem of mining diverse top-k phenotypes consists of generating a set of phenotypes $\{l_1, \ldots, l_k\}$ such that they contain only positive pairs pattern-target and are diverse.

GMSL and DSLM algorithms, which were defined in Sections 4.4 and 4.5, tackle this problem by using SD, the subgroup list model, and the MDL principle. First, the GMSL algorithm is applied to a well-known dataset from literature in order to generate diverse top-k subgroup lists, after which, the DSLM algorithm is applied to real data from MIMIC-III in order to mine diverse top-k patient phenotypes. Both algorithms consider both types of diversity, and the latter introduces the generation of diverse top-k subgroup lists with only positive pairs pattern-target and the utilisation of an overlap factor.

4.4 GMSL algorithm

In this chapter, we propose the Generation of Multiple Subgroup Lists algorithm (GMSL), whose objective is to obtain diverse top-k subgroup lists by combining SD, the subgroup list model and the MDL principle.

4.4.1 Formal aspects

The details of this proposal can be found in Algorithm 6, which requires the following parameters: a dataset d, a collection of candidate subgroups $\mathcal{C}$, the maximum number of subgroup lists that will be generated, and the normalisation parameter β, which is required by the compression gain $\Delta_\beta L$ (see Equation 4.1). It is also necessary to note that the candidate subgroups from $\mathcal{C}$ can be generated using any algorithm and can be filtered before running the GMSL algorithm.

This algorithm begins by initialising a list denominated as $\mathcal{L}$ with empty subgroup lists, which has a size of max_sl (line 1). It then proceeds to iterate through $\mathcal{L}$ (loop of line 2), and continuous iterations through $\mathcal{C}$ are performed for each subgroup list in order to search for the best candidate subgroup to be added (lines 6 - 12). This is done by calculating the compression gain of each candidate subgroup using the $\Delta_\beta L$ function (line 7). The candidate with the highest compression gain is selected (lines 8 - 11) and included in the current subgroup list (lines 13 - 17), and this process is repeated until there are no candidate subgroups with a positive compression gain. Finally, the algorithm returns the collection $\mathcal{L}$ that contains max_sl subgroup lists. It is essential to note that the use of the MDL principle to compute the compression gain for each candidate subgroup ensures that all the subgroups added to a subgroup list are statistically robust.

Algorithm 6 GMSL algorithm.

Input: d { dataset } ; $\mathcal{C}$ { candidate subgroups } ; max_sl { maximum number of subgroup lists to generate ($\mathbb{N}$) } ; β { normalisation parameter $\in [0,1]$ }
Output: $\mathcal{L}$: collection of subgroup lists.

```
1:  L := create a collection with max_sl empty subgroup lists.
2:  for each sl ∈ L do
3:      repeat
4:          best_candidate := NULL
5:          bc_comp_gain := 0
6:          for each current_candidate ∈ C do
7:              cc_comp_gain := Δ_β L(d, sl ⊕ current_candidate)
8:              if cc_comp_gain > bc_comp_gain then
9:                  best_candidate := current_candidate
10:                 bc_comp_gain := cc_comp_gain
11:             end if
12:         end for
13:         if best_candidate ≠ NULL then
14:             sl := sl ⊕ best_candidate
15:             C.delete(best_candidate)
16:             C.deleteRefinements(best_candidate)
17:         end if
18:     until best_candidate = NULL
19: end for
20: return L
```

The algorithm is able to produce diverse subgroup lists thanks to two factors. Firstly, the utilisation of the subgroup list model ensures diversity in terms of coverage. According to the formal definition, this model ensures that each instance in the input dataset is exclusively covered by either an individual subgroup or the default subgroup. However, if the subgroups are individually considered without the subgroup list model, there may be overlaps among them. This will be solved by our second proposal, i.e. the DSLM algorithm, which introduces an overlap factor. Secondly, the algorithm guarantees diversity in terms of descriptions. Each time that a candidate subgroup from $\mathcal{C}$ is added to a subgroup list, that specific subgroup and its refinements are removed (lines 15 and 16). As a result, each candidate subgroup appears at most once and the occurrence of duplicate selectors across different patterns is also minimised.

The implementation of the GMSL algorithm is available at the `subgroups` python library[1], which will be introduced in Chapter 5.

[1] `https://github.com/antoniolopezmc/subgroups`

4.4.2 Experiments and Discussion

We verified the ability of the GMSL algorithm to produce diverse top-k subgroup lists by employing the well-known car-evaluation dataset from the UCI repository, focusing on "class = acc" as a target, which indicates the cars that are considered acceptable for purchase. We then converted the attributes into binary form by applying the One Hot Encoding technique to the dataset. After this preprocessing had been carried out, the dataset consisted of 1,728 instances and 18 attributes. We subsequently employed the VLSD algorithm with the WRAcc quality measure, establishing a threshold value of 0. We then filtered the subgroups in order to only use those whose description had a maximum of 2 selectors for the sake of understandability and the legibility of the results. As a result, we obtained 302 subgroups. These candidate subgroups ($\mathcal{C}$) served as the main input for the GMSL algorithm in order to generate diverse top-k subgroup lists.

The top-3 subgroup lists obtained are presented in Table 4.1. Each subgroup list includes the following information: (1) the individual subgroups and the default subgroup; (2) the number of positive and negative instances covered by the subgroups individually; (3) the contribution of the subgroups to the subgroup list (that is, the number of positive and negative instances covered by the subgroups according to their position in the subgroup list), and (4) the subgroup list coverage (that is, the cumulative sum of positive and negative instances covered by the subgroups up to that time).

The three diverse subgroup lists are composed of different descriptions and patterns that contribute to the generation of diverse explanations from the same data.

Several observations can be extracted from Table 4.1. Firstly, both the first and second subgroup lists contain the original attribute `buying` (represented as `buying_vhigh` and `buying_low` after executing One Hot Encoding), while the third subgroup list does not utilise this attribute. Secondly, various attributes derived from the original `doors` attribute are employed by all the subgroup lists. Furthermore, the first and second subgroup lists consist of 2 subgroups with a single selector in their descriptions, while the third subgroup list comprises 3 subgroups with a single selector in their descriptions.

According to the subgroup list coverage values in the last subgroup (before the default subgroup), it is apparent that the first and third subgroup lists cover a greater number of positive examples when compared to the second subgroup list. Similarly, the first subgroup list is more general, consisting of fewer subgroups, while the second subgroup list is more specific, containing a higher number of subgroups.

It is worth noting that while subgroups serve as local models, subgroup lists function as global models, encompassing the entire dataset, but they focus on modelling solely a

	Subgroup description (Pattern)	**Subgroup coverage Pos-Neg**	**Contribution Pos-Neg**	**Subgroup list coverage Pos-Neg**
s1	doors_2='no', lug_boot_low='no'	384-384	384-384	384-384
s2	buying_vhigh='no'	312-984	0-720	384-1104
s3	buying_low='no'	292-1004	0-240	384-1344
	Default subgroup	-	0-0	384-1344

	Subgroup description (Pattern)	**Subgroup coverage Pos-Neg**	**Contribution Pos-Neg**	**Subgroup list coverage Pos-Neg**
s1	doors_2='no', lug_boot_high='yes'	204-180	204-180	204-180
s2	doors_2='no', lug_boot_med='no'	204-564	0-384	204-564
s3	doors_2='no', persons_small='no'	279-489	145-111	349-675
s4	persons_small='no'	279-873	0-384	349-1059
s5	lug_boot_high='yes'	204-372	0-64	349-1123
s6	buying_vhigh='yes', lug_boot_low='no'	72-216	0-48	349-1171
	Default subgroup	-	35-173	384-1344

	Subgroup description (Pattern)	**Subgroup coverage Pos-Neg**	**Contribution Pos-Neg**	**Subgroup list coverage Pos-Neg**
s1	doors_4='yes', lug_boot_low='no'	198-186	198-186	198-186
s2	doors_more='no', lug_boot_low='no'	198-570	0-384	198-570
s3	lug_boot_low='no'	384-768	186-198	384-768
s4	maint_2='no'	303-993	0-432	384-1200
s5	doors_2='no'	384-768	0-96	384-1296
	Default subgroup	-	0-48	384-1344

Table 4.1: Diverse top-3 subgroup lists generated from car-evaluation dataset (i.e. three different explanations of this dataset) with "class = acc" as target.

single target value or class. Moreover, each subgroup list contains a different number of subgroups: three in the first case, six in the second case, and five in the third case.

In our approach, the candidate subgroups are generated a-priori and then used as input for the GMSL algorithm. While this approach may increase memory consumption, it also has the advantage of flexibility by allowing the collection to be prefiltered. For example, certain negative subgroups such as “doors_2 = no” or “doors_more = no” were generated from the car-evaluation dataset. However, these subgroups may not be logically meaningful to the user and can be removed before executing the GMSL algorithm.

Please recall that, according to the definition provided by Proença et al. (2022), subgroups within a subgroup list do not overlap. Nevertheless, when examining each of these subgroups separately, without considering the subgroup list model, it is possible for them to cover the same instances of the database.

In summary, our study demonstrates the suitability of our proposal as regards addressing the initial problem by effectively identifying diverse top-k subgroup lists.

4.5 DSLM algorithm

In this chapter, we propose the Diverse Subgroup Lists Miner (DSLM), whose objective is the same as that of GMSL, but two new improvements have been made: (1) this algorithm adds a subgroup to the subgroup list only if it contributes with at least one positive instance from the dataset, and (2) it enhances the diversity by incorporating an overlap factor and improving the refinements removal process.

4.5.1 Formal aspects

Given the problem formalised in Section 4.3, we propose the DSLM (shown in Algorithm 7), which takes the following parameters as input: a dataset d, a collection $\mathcal{C}$ of candidate subgroups, the number k of phenotypes to be generated, the maximum number of subgroups for each phenotype (sl_max_size), the maximum positive overlap permitted ($max\_positive\_overlap$), the maximum negative overlap permitted ($max\_negative\_overlap$), and the normalisation parameter β, which is required by the compression gain $\Delta_\beta L$ (see Equation 4.1). As before, the subgroups within $\mathcal{C}$ can be created using any SD algorithm and subsequently filtered before running this algorithm.

The DSLM algorithm initially creates an empty collection $\mathcal{L}$ (line 1), after which it iterates k times (line 2). For each iteration, it initialises an empty subgroup list, adds it

Algorithm 7 DSLM algorithm.

Input: d { dataset }, $\mathcal{C}$ { candidate subgroups }, k { $\mathbb{N}$ }, sl_max_size { $\mathbb{N}$ }, $max\_positive\_overlap$ { $\mathbb{R}$ }, $max\_negative\_overlap$ { $\mathbb{R}$ } ; β { normalisation parameter $\in [0,1]$ }

Output: $\mathcal{L}$ { set of diverse top-k phenotypes }

$\mathcal{L} := \{\}$
for $i := 1 \ldots k$ **do**
  sl := create empty phenotype (subgroup list) ; $\mathcal{L}.add(sl)$
  $positive\_overlap\_counter$:= list of size $|d.instances|$ initialised with 0s
  $negative\_overlap\_counter$:= list of size $|d.instances|$ initialised with 0s
  repeat
    $best\_candidate := NULL$; $best\_score := 0$
    for each $c \in \mathcal{C}$ **do**
      $pof := compute\_overlap\_factor(positive\_overlap\_counter, c)$
      $nof := compute\_overlap\_factor(negative\_overlap\_counter, c)$
      $c\_score := \Delta_\beta L(d, sl \oplus c) * (1 - pof) * (1 - nof)$
      if ($c\_score > best\_score$) and ($is\_positive(c)$) and
       ($pof \leq max\_positive\_overlap$) and
       ($nof \leq max\_negative\_overlap$) **then**
        $best\_candidate := c$; $best\_score := c\_score$
      end if
    end for
    if $best\_candidate \neq NULL$ **then**
      for each $Instance\ i \in d$ **do**
        if ($i \sqsubset best\_candidate.description$) and
         ($i \sqsubset best\_candidate.target$) **then**
          $positive\_overlap\_counter[i]++$
        end if
        if ($i \sqsubset best\_candidate.description$) and
         ($i \not\sqsubset best\_candidate.target$) **then**
          $negative\_overlap\_counter[i]++$
        end if
      end for
      $sl := sl \oplus best\_candidate$
      $\mathcal{C}.delete(best\_candidate)$
      $\mathcal{C}.deleteRefinementsOfThis(best\_candidate)$
      $\mathcal{C}.deleteOfWhichThisIsRefinement(best\_candidate)$
    end if
  until ($best\_candidate = NULL$) or ($|sl| \geq sl\_max\_size$)
end for
return $\mathcal{L}$

to $\mathcal{L}$ (line 3), and creates two lists (overlap counters) in which each element represents the number of times that a positive/negative dataset instance has been covered by the subgroups already contained in sl. The algorithm then iterates over $\mathcal{C}$ (lines 6 and 8) in order to fill the current subgroup list sl with the best candidate from each iteration. Each candidate subgroup from $\mathcal{C}$ is selected depending on the conditions of line 12, i.e. only if: (1) the subgroup has a higher score than the best one found until that moment, which is determined by the MDL principle and the overlap factor (line 11); (2) the subgroup is positive, meaning that it is formed of a positive pair pattern-target, and (3) the overlap (positive and negative) of the subgroup with the subgroups already present in the subgroup list is less than or equal to the maximum permitted. In this case, the overlap factor is considered separately for the positive and negative instances from d and is computed by employing the $compute\_overlap\_factor$ function (lines 8 and 9), which uses Equation 4.2. When the best candidate is selected: (1) positive/negative overlap counters are updated (lines 17-24); (2) it is inserted into the subgroup list and deleted from $\mathcal{C}$ (lines 25 and 26); (3) its refinements are removed from $\mathcal{C}$ (line 27), and (4) all subgroups from $\mathcal{C}$ of which it is a refinement are removed (line 28). Lastly, the collection $\mathcal{L}$ with top-k subgroup lists is returned.

Both the calculation of overlap and the operations performed in lines 27 and 28 contribute to diversity. What is more, they contribute to reducing the number of candidates that will be explored, which would not be chosen anyway because of their overlap.

The implementation of the DSLM algorithm is available at the `subgroups` python library[2], which will be introduced in Chapter 5.

4.5.2 Experiments and Discussion

The aim of the experiments was to verify the validity of our proposal in the context of phenotyping antimicrobial resistances. We utilised real clinical data obtained from the MIMIC-III public database (Johnson et al., 2016), focusing on patients who were infected by an Enteroccous Sp. bacterium resistant to Vancomycin. The dataset consisted of 9,240 instances and 12 attributes, with 2,126 positive instances and 7,114 negative instances. We subsequently extracted the subgroups using the VLSD algorithm by employing the WRAcc quality measure with a threshold value of 0. We then filtered the subgroups in order to use only those whose description has 3 selectors at maximum for the sake of understandability and the legibility of the results. This led to the attainment of 473

[2] `https://github.com/antoniolopezmc/subgroups`

	Subgroup description (Pattern)	Subgroup coverage Pos-Neg	Contribution Pos-Neg	Phenotype coverage Pos-Neg
s1	culture_type = 'SWAB', icu_when_culture = 'SICU'	466-168	466-168	466-168
s2	culture_type = 'URINE', previous_vancomycin = 'yes'	145-85	145-85	611-253
s3	days_admitted_before_ICU = 'OneDayOrMore', discharge_location = 'DEAD/EXPIRED', patient_age = 'ADULT'	167-100	147-96	758-349
s4	culture_type = 'BLOOD_CULTURE', previous_vancomycin = 'yes'	112-111	98-110	856-459
s5	admission_location = 'PHYS_REFERRAL/NORMAL_DELI', readmission = 'yes', service_when_culture = 'SURG'	124-86	81-80	937-539
s6	culture_type = 'SWAB', previous_vancomycin = 'yes'	114-97	76-89	1013-628
	Default subgroup	-	1113-6486	2126-7114

	Subgroup description (Pattern)	Subgroup coverage Pos-Neg	Contribution Pos-Neg	Phenotype coverage Pos-Neg
s1	culture_type = 'SWAB', service_when_culture = 'SURG'	380-241	380-241	380-241
s2	days_admitted_before_ICU = 'OneDayOrMore', previous_vancomycin = 'yes'	166-136	150-126	530-367
s3	service_when_culture = 'OMED'	115-95	85-92	615-459
s4	days_admitted_before_ICU = 'ZeroDays', previous_vancomycin = 'yes', service_when_culture = 'SURG'	110-77	76-64	691-523
s5	culture_type = 'SWAB', service_when_culture = 'MED'	205-321	193-320	884-843
s6	days_admitted_before_ICU = 'OneDayOrMore', discharge_location = 'DISTINCT_PART_HOSP', service_when_culture = 'SURG'	125-121	76-81	960-924
	Default subgroup	-	1166-6190	2126-7114

Table 4.2: Diverse top-2 phenotypes from our dataset.

subgroups.

The DSLM algorithm was run using these subgroups and with $k = 2$, $sl\_max\_size = 6$, $max\_positive\_overlap = 0.06$, $max\_negative\_overlap = 0.06$ and $\beta = 0$, obtaining the subgroup lists depicted in Table 4.2. The first column represents subgroup descriptions, the second column shows the number of positive and negative instances covered by the subgroups, the third column represents the contribution of the subgroups to the subgroup list (that is, the number of positive and negative instances covered by the subgroups according to their positions in the subgroup list), and the fourth column shows the phenotype coverage that is the cumulative sum of positive and negative instances covered by the subgroups up to that time.

With regard to the diversity of the phenotypes, we consider that there are two desirable situations. On the one hand, the set of phenotypes is good if they cover as many instances of the population as possible. This also implies that if two phenotypes have a

low overlap, then they are focused on different parts of the dataset and, therefore, could have common selectors in their subgroups and still be useful for the clinician. On the other hand, the set of phenotypes is also good if they use different selectors that provide different views of the population even if the instances overlap. If two phenotypes have a high overlap, this means that they explain the same subpopulation of the dataset, and should consequently have different selectors in their subgroup descriptions for them to still be useful for the clinician.

The experiments show that both subgroup lists cover almost 50% of the 2126 positive instances from the dataset, with 1013 instances in the first phenotype and 960 in the second case. Moreover, the percentage of overlap between the two subgroup lists was 26.8%. Despite being specific to a subset of the population, it will be noted that they are still general and cover a good number of negative instances. This is thanks to the use of the MDL principle, which provides further benefits as regards the predictive capacity of the phenotypes evaluated later. It will also be noted that most of the negative instances are covered by the default subgroup in both cases, meaning that they are not covered by the subgroups from the subgroup list. Note that the default subgroup is not part of the phenotype, but is present in order to score the model with the MDL encoding.

Note in Table 4.2 ("Subgroup coverage" column) that the subgroups selected cover more positive than negative instances since filtering was carried out in order to attain only those subgroups with a minimum WRAcc quality value of 0. Each subgroup added to the phenotype contributes to the coverage with a proportionally high number of positive instances and the MDL score of the resulting phenotype is higher, in other words, from an information theory perspective, the amount of information regarding the phenotype has increased. For example, in the first phenotype in Table 4.2, subgroup s3 covers 167 patients with antibiotic resistance, of which 147 were not covered by the phenotype when using only subgroups s1 and s2. Up to that point (with subgroups s1, s2 and s3), the phenotype covered a total of 758 patients with antibiotic resistance and 349 without it.

With regard to the diversity of the descriptions, the two phenotypes shown in Table 4.2 contain different patterns. More precisely, the first phenotype contains 11 unique selectors, while the second phenotype includes 8 unique selectors, and both phenotypes have 4 common selectors. In general, and reading the subgroups of each phenotype in order, the first phenotype describes patients admitted to an ICU ward, adults, and different types of culture (swab, urine or blood), along with treatment by vancomycin in previous admissions and those that could die. The population of the second phenotype can be described as a population from emergencies or medical wards (SURG, OMED or MED)

Algorithm	Accuracy model 1	Accuracy model 2	p-value
Random Forest	0.8279	0.8391	0.0075
Gradient Boosting Classifier	0.8174	0.8268	0.0088
Logistic Regression	0.7741	0.8073	1.936408e-07

Table 4.3: Evaluation of predictive accuracy.

that might or might not become complicated and moved to ICU, and the cultures were mainly swabs. Moreover, vancomycin had been administered in previous admissions.

Finally, in order to assess the phenotypes from an objective perspective, we demonstrate their predictive capacity by incorporating them as dummy variables in a classification algorithm that forecasts the target attribute associated with the phenotypes. We fit two classification models by: (1) employing the original dataset, and (2) augmenting the original dataset with two attributes indicating whether or not an instance is covered by each phenotype. The results presented in Table 4.3 consistently indicate that the second classification model achieved statistically higher accuracy when compared to the first model, thereby confirming the predictive capacity of both phenotypes. Implementations of Random Forest, Gradient Boosting Classifier and Logistic Regression from the *scikit-learn* python library were used for this evaluation, the dataset was split in a stratified manner (70% for training, 30% for testing), and McNemar's test (implemented in the *pingouin* python library) was used in order to measure statistical significance.

In summary, the experiments carried out in this section show a specific clinical problem to which our proposal is well-suited as regards addressing and solving the initially defined problem.

4.6 Conclusions

In this chapter, we proposed a new approach for phenotyping that is focused on the task of mining diverse top-k subgroup lists. Our goal was to provide clinicians with a limited number of diverse (as regards both coverage and descriptions) descriptions concerning a particular target value of interest.

In order to tackle this problem, we defined the GMSL (Generation of Multiple Subgroup Lists) and DSLM (Diverse Subgroup Lists Miner) algorithms, which take a set of pre-computed candidate subgroups as input and return a collection of diverse top-k subgroup lists, by using the SD paradigm, the subgroup list model and the MDL principle. On the one hand, the GMSL algorithm was applied to a well-known dataset

from literature so as to generate diverse top-k subgroup lists. On the other, we carried out experiments concerning phenotyping antimicrobial resistances in the MIMIC-III database with the DSLM algorithm.

With regard to the outcomes achieved using the DSLM algorithm, both subgroup lists effectively represent both patient phenotypes, in addition to having valuable characteristics. These phenotypes are statistically robust, readable by clinicians, have diversity, and involve only a small number of selectors. In terms of coverage, there is a 26.8% overlap between the phenotypes obtained. With regard to the descriptions, the selectors used are diverse and non-redundant, providing various explanations for the medical condition selected from the dataset. Moreover, the results are straightforward and can be easily understood by clinicians, and the predictive capacity of the phenotypes obtained has been confirmed by the fact that they significantly improve the accuracy of all classification algorithms used in order to predict the same phenotype outcome. This improvement was achieved by incorporating the phenotypes as new independent variables in the dataset.

Finally, there are several potential avenues for future research with which to enhance and expand upon the suggested algorithms. One possibility is to develop methods that can generate subgroup lists without relying on a preloaded set of candidate subgroups. Another interesting direction would be to explore the problem in a multiclass setting.

CHAPTER 5

Subgroup Discovery with 'subgroups'

In this chapter, we present and describe the 'subgroups' library, a public, accessible and open-source Python library created in order to work with the Subgroup Discovery (SD) technique. This library implements the necessary components related to the SD technique and contains a collection of SD algorithms (all those developed in this research, along with others that already existed in literature). This chapter explains and details its general structure and its different parts. Note that this library has been used to carry out all the experiments described in this thesis that are related to the SD technique.

5.1 Motivation

Despite the utility of the SD technique and the great variety of SD algorithms that appear in literature (see Chapter 3), few implementations are available for scientific research. Present-day data scientists and Machine Learning (ML) researchers often depend on highly reliable libraries in order to test and compare state-of-the-art algorithms such as the classical ML tool Weka, scikit-learn, or Keras or PyTorch in the Deep Learning area. This is not the case in the SD field, in which no community supports the few libraries that are available. This is a major problem, particularly as regards AI-based medical research. Some examples of existing SD tools are Vikamine, Keel or Cortana.

The aforementioned disadvantages signify that it is necessary to use different libraries and tools when working with different SD algorithms, and that there is no single reference library that implements and brings together a large number of SD algorithms, not even the most popular ones.

It is, therefore, definitively necessary to have a complete library available that implements the most popular algorithms in a faithful form with respect to the original definition without adding modifications, along with a complete documentation of use with different examples. This is precisely the main objective of the 'subgroups' Python library.

Our proposal is, therefore, the 'subgroups' library, which is a public, accessible and open-source Python library that faithfully implements the most popular SD algorithms with regard to their original definition. It is available on GitHub[1] and PyPI for researchers, developers and, in general, all those who wish to work with the SD technique.

5.2 Design and implementation

This section describes the general design and structure of the 'subgroups' library, along with the implementation details and other decisions made when developing it. We refer the reader to Chapter 3 for a complete formal definition and other details of the SD technique, and to Appendix A for a user guide to the installation and use of the 'subgroups' library.

With respect to general design of the 'subgroups' library, the following components can be highlighted:

1. `core`: this contains the basic elements required in order to work with the SD technique, such as `pattern` or `subgroup`.

2. `quality_measures`: this implements a wide variety of quality measures used by different SD algorithms, such as `WRAcc` or `Binomial Test`, along with some optimistic estimates of these quality measures.

3. `data_structures`: this includes the data structures used by the SD algorithms implemented, such as `FPTree` or `Vertical list`, among others.

4. `algorithms`: this implements the SD algorithms, such as `SDMap` or `VLSD`, among others.

This library also contains many built-in datasets from literature, the corresponding tests for all the functionalities implemented, and a file containing some new software exceptions used by the library. The general structure of the 'subgroups' library is shown in Figure 5.1.

[1] https://github.com/antoniolopezmc/subgroups

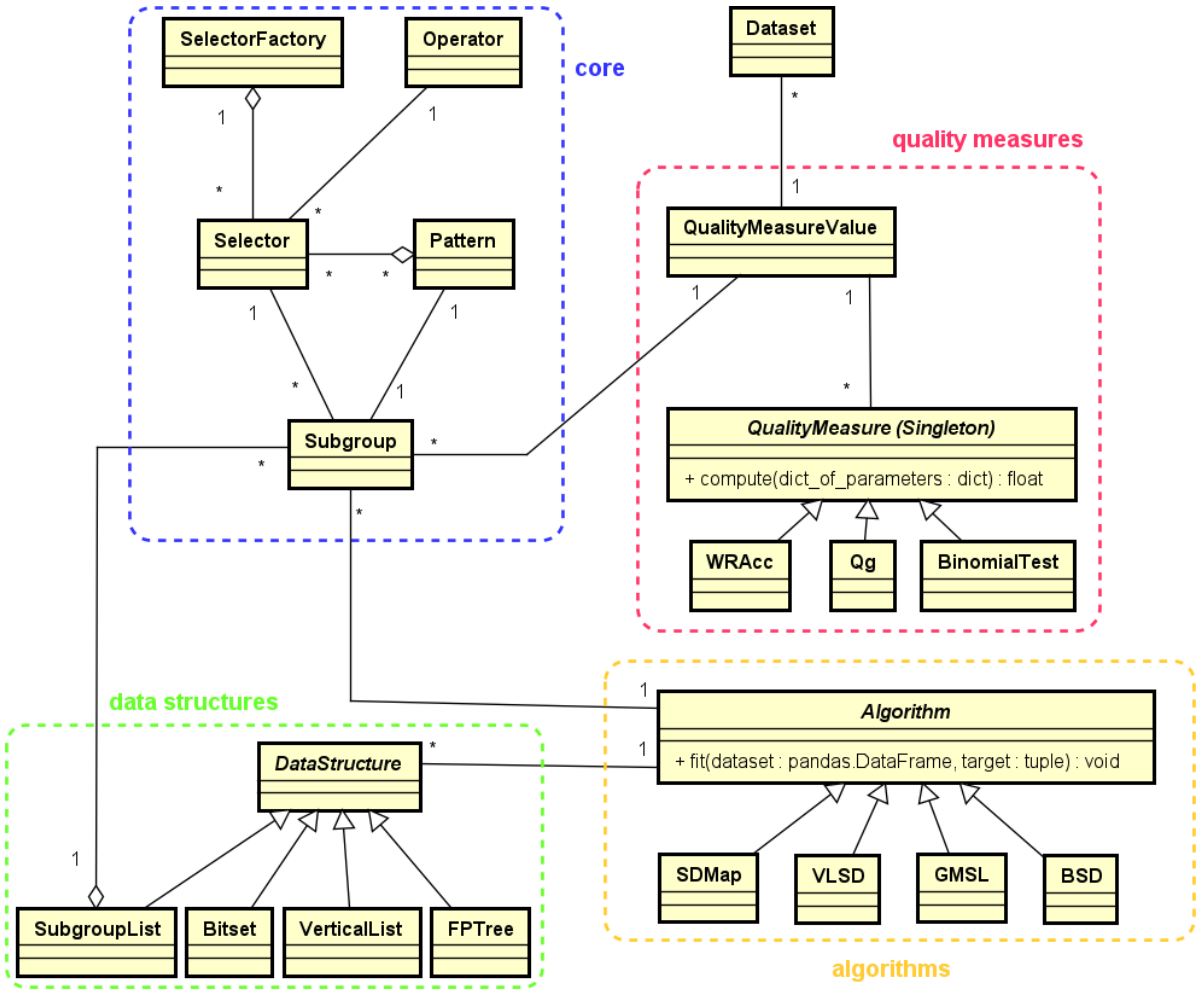


Figure 5.1: General structure of the 'subgroups' Python library.

The different classes belonging to the 'subgroups' library are detailed as follows.

The `core` component contains the following:

The `Selector` class, which includes all the properties and functionalities corresponding to a selector, which is an immutable structure formed of an attribute name, an operator and a value. In this implementation, an attribute name must be of the string type and a value must be of either a string or a numeric type. Moreover, since a selector is immutable, we use the Flyweight design pattern, meaning that there is only one instance for each different selector. This makes it possible to increase the performance of the library in terms of memory consumption.

The `Operator` class, which implements all properties and functionalities corresponding to an operator. In this case, an operator is of an enumerated type, which can take the following values: (1) equal, (2) not equal, (3) less, (4) greater, (5) less or equal, and (6)

greater or equal.

The `Pattern` class, which contains all the properties and functionalities corresponding to a pattern. In this library, a pattern is formed of a list of selectors, which is sorted in order to be able to efficiently compare two patterns.

The `Subgroup` class, which includes all properties and functionalities corresponding to a subgroup. In this case, it is composed of a pattern, called a description, and a selector, called a target.

The `quality_measures` component contains the following:

The `QualityMeasure` class, which is an abstract class that incorporates all general properties corresponding to a quality measure, meaning that all specific quality measures must inherit from it. The most remarkable method from this class is the `compute` method, which computes the value of the specific quality measure using a set of parameters corresponding to the results obtained by the functions described in Chapter 3. The parameter received by this method is a Python dictionary containing the aforementioned parameters and, if required by the quality measure, other parameters with which to compute it. The Singleton design pattern is used for each quality measure.

With regard to the data structures, although Figure 5.1 depicts a `DataStructure` abstract class for the sake of understandability, it does not actually exist in practice, but all specific data structures are implemented independently since they do not have common properties.

The `Algorithm` class is an abstract class that includes the common functionality corresponding to all algorithms, meaning that all specific algorithms must inherit from it. More precisely, the most important method in this class is the `fit` method, which is, as occurs in *scikit-learn* library, the entry point at which to execute the specific algorithm. Moreover, as is the case in *scikit-learn*, the initial properties and configurations of the algorithm are established in the constructor method from the class.

With respect to the implementation details, the 'subgroups' library has been implemented in Python 3 using the object-oriented programming paradigm and following the style of *scikit-learn*, which is a reference in the ML field and is one of the libraries that is best known and most frequently used by the community. Moreover, this library uses different data structures and functionalities provided by *pandas*, which is also a reference in the ML community.

With regard to the quality measures, the following are currently implemented in the library: Binomial Test, Coverage, Piatetsky Shapiro, Positive Predictive Value (PPV), Negative Predictive Value (NPV), Qg, Sensitivity, Support, Weighted Relative Accuracy

(WRAcc), Absolute WRAcc, Specificity, Incremental Response Rate (IRR), F1 Score, and Youden. The library also implements different optimistic estimates for Binominal Test, Piatetsky Shapiro and WRAcc quality measures.

With regard to the algorithms, the following are currently implemented in the library: BSD, CBSD, CPBSD, QFinder, SDMap, SDMap* and VLSD, which generate a subgroup set (i.e. individual subgroups), and GMSL and DSLM, which generate subgroup lists.

A key aspect for developers and researchers is to be able to easily compare, evaluate and test SD algorithms. It is for this reason that this library also implements and offers other important features that are described as follows.

This library offers different metrics after executing the algorithms, such as the selected subgroups, the unselected subgroups or the visited nodes, thus allowing users to evaluate the performance of specific executions and to compare different executions with each other.

It also provides the possibility of not printing the results either on the screen or in a file when it is not necessary, signifying that the execution runtimes are not incremented and no bias is added to the experiments.

Another feature is that the library is also easily extensible, since users can add new quality measures, data structures and algorithms. This is one of the most notable advantages of this library, and is what allows it to constantly improve and grow. This process is detailed in Appendix B.

The 'subgroups' library also offers a large set of tests that can be executed in order to verify that all components, algorithms and features are correctly implemented.

It is necessary to state that the algorithms implemented in this library have been manually compared with executions on toy datasets and, specifically, the results obtained by the exhaustive algorithms have also been compared with those of other exhaustive algorithms from this library and other existing tools. This makes it possible to verify that both the algorithms and the results obtained by them are correct.

In addition, the directory structure of the 'subgroups' library is shown in Figure 5.2.

5.3 Conclusions

In this chapter, we have presented 'subgroups', a Python library with which to work with the SD technique. This library is public, accessible and open-source and implements the necessary components related to the SD technique, along with a collection of SD

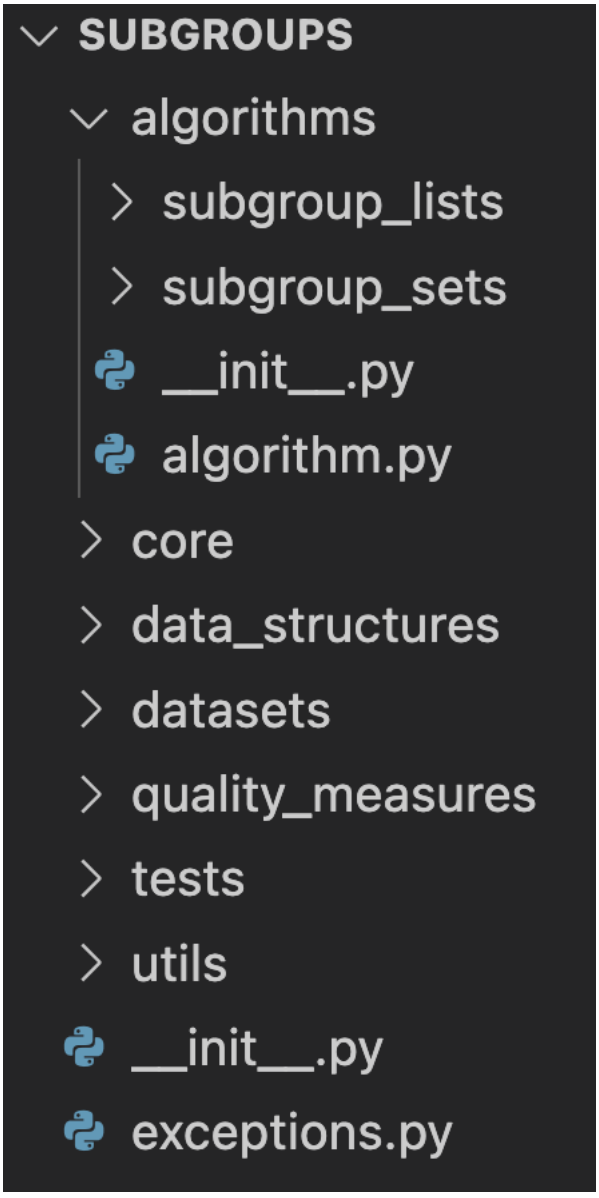


Figure 5.2: Directory structure of the ‘subgroups’ Python library.

algorithms. This chapter also describes the library presented, its general structure and the different parts of which it is formed.

Moreover, as explained in Section 5.1, the development of this Python library was motivated by one of the most notable disadvantages of the SD technique in practice, which is the lack of SD algorithm implementations and libraries.

The fundamental objective of this library is, therefore, to allow users to access a wide variety of SD algorithms, focusing on the most popular ones, in a faithful form with respect to their original definition. Another relevant aspect of this proposal is that it provides detailed documentation of use with different examples.

Finally, it is necessary to state that, apart from the SD algorithms already implemented, our library is still growing and new SD algorithms are being implemented in order to make the library more complete with, for example, CN2-SD or Apriori-SD, among others.

CHAPTER 6

Conclusions

This PhD thesis addresses the antibiotic resistance problem from a perspective based on Machine Learning (ML) with the general aim of developing and implementing several ML techniques and methods with which to automatically obtain patient phenotypes.

The hypothesis of this PhD thesis was that clustering and subgroup discovery (SD), which are two ML techniques, are effective as regards supporting the patient phenotyping process in the clinical context of antibiotic resistance. We hypothesized that refined and adapted versions of these techniques could generate helpful and readable phenotypes for clinicians. Several objectives were consequently described in Chapter 1 (Section 1.3) in order to prove this hypothesis.

After carrying out the research shown herein, the main conclusions of this PhD thesis in relation to the objectives proposed are:

- Objective 1:
  - Clustering and SD can be used as a basis on which to design new ML techniques for phenotyping whose results are useful and are easy for clinicians to understand.
  - The new ML techniques created in this work from either clustering or SD can be successfully applied to the antibiotic resistance problem.
  - Both local and global models can serve for patient phenotyping.
- Objective 2:

  - The Trace-based clustering technique generates patient phenotypes and its results are easy for clinicians to understand.
  - The generation of different partitions from the same dataset in order to evaluate their overlap reduces dependency on the randomness of the traditional clustering techniques.
  - The stability concept allows the identification of interesting sets of patients and is generic, transversal and not dependent on the specific ML technique used.
  - A representation based on a heat-map helps to easily visualise the overlap between a large number of pairs of clusters.
  - Statistical metrics such as the mean make it possible to rank and filter the results obtained by the Trace-based clustering technique.

- Objective 3:
  - The new 5-step methodology provides a straightforward guide with which to identify and rank patient phenotypes, and allows clinical experts to be involved in the discovery process.
  - The Hopking statistic is an interesting alternative to consider as regards determining the best values of the k hyperparameter when the elbow method is not conclusive.
  - Ranking cluster techniques supported by visual methods allow users to easily select and analyse a reduced number of clusters.
  - A classification-based evaluation can be an alternative when the patients' personal records cannot be examined by a clinical expert.
  - The high accuracy obtained in the classification-based evaluation provides objective evidence of the validity of our methodology.
  - The definition of generic concepts such as clustering function or matching function allows the versatility of our methodology to be enhanced.

- Objective 4:
  - The VLSD algorithm can be used either to directly extract patient phenotypes or as part of other phenotyping techniques.
  - The results obtained by the VLSD algorithm are easily understood by users.

- The VLSD algorithm performs better than the other state-of-the-art SD algorithms considered in terms of runtime, max memory usage and nodes visited owing to the combined use of the equivalence class exploration strategy, a pruning based on an optimistic estimate and a pruning based on the matrix $\mathcal{M}$.
- The utilisation of pruning techniques such as those based on an optimistic estimate improves the general performance of an SD algorithm.
- The utilisation of higher threshold values when implementing a pruning based on an optimistic estimate allows the algorithm to visit fewer nodes with respect to other algorithms that do not implement this pruning.
- The pruning based on the matrix $\mathcal{M}$ allows the algorithm to visit fewer nodes with respect to other algorithms that do not implement this pruning.
- The data structure proposed makes it possible to efficiently generate subgroup refinements and compute all quality measures.

- Objective 5:
  - The new problem of mining diverse top-k subgroup lists provides a new approach for patient phenotyping.
  - The GMSL and DSLM algorithms can generate diverse top-k phenotypes in the form of subgroup lists, and their results are useful and are easy for experts to understand.
  - The use of an overlap factor ensures higher diversity in terms of coverage.
  - The use of a refinement deletion mechanism enhances diversity in terms of descriptions.
  - The predictive capacity is a useful metric with which to evaluate phenotypes from an objective perspective.
  - The combination of the Minimum Description Length (MDL) principle with existing ML algorithms is a promising approach by which to solve the new problem defined and to provide a solid foundation for the models.
  - The combination of the MDL principle and the SD technique is a well-founded approach from a theoretical and practical point of view and solves the problem described in this research.

- Objective 6:

  - The 'subgroups' library is easy for data scientists to understand since it follows an interface similar to that of *scikit-learn*.
  - The 'subgroups' library has been designed to be extensible, since extensibility is a key property that allows users to easily contribute to a library.
  - The 'subgroups' library can be easily accessed, since it is available on GitHub and PyPI.
  - The 'subgroups' library already implements several quality measures and SD algorithms, which currently allows its use by an end-user for tasks such as phenotyping.
  - The 'subgroups' library has tests and performance metrics with which to validate and compare new algorithm implementations, which allows it to be used by ML researchers.

- Objective 7:
  - The MIMIC-III database is an excellent data source that provides rich data concerning the antibiotic resistance problem, helps researchers in this field, and ensures research reproducibility.
  - Public clinical databases such as MIMIC-III can be used in order to either reproduce existing research or carry out new research when no other data sources are available.

Having presented the conclusions, we can state that all of the initial objectives proposed have been accomplished.

Additionally, this thesis and all the research carried out could be used as a starting point for the following future work:

- After verifying the suitability of the public MIMIC-III database as regards extracting clinical data related to the antibiotic resistance problem, the new version (MIMIC-IV) could be explored in order to attain different data source alternatives and reproduce the work carried out in this research. MIMIC-III and MIMIC-IV could also be used to reproduce other research from literature.
- Apart from the real clinical data from MIMIC-III or MIMIC-IV, a simulator with which to generate synthetic clinical data could be an interesting new approach to explore, since it could be used in the absence of real clinical data or to reproduce both our research and other works from literature.

- The Trace-based clustering technique proposed in Chapter 2 was designed to be applied by using partitional clustering algorithms such as K-Means. However, our technique could be adapted in order to use other types of clustering methods such as hierarchical clustering.
- The Trace-based clustering technique proposed in Chapter 2 was used as a key part of the methodology proposed in Chapter 2. The utilisation of this technique as a part of other patient phenotyping methodologies from literature could, therefore, be explored.
- The methodology for patient phenotyping proposed in Chapter 2 was based mainly on the utilisation of the Trace-based clustering technique. However, other ML techniques with which to obtain readable phenotypes, such as tree-based techniques, could be used by either replacing Trace-based clustering or in conjunction with it.
- The methodology for patient phenotyping proposed in Chapter 2 included a visualisation method based on a heat-map. However, the use of other visualisation methods in this methodology could be explored.
- The methodology for patient phenotyping proposed in Chapter 2 defined some general concepts such as clustering function or matching function. The use of different specific clustering functions or matching functions could, therefore, be explored in order to enhance its versatility.
- The methodology for patient phenotyping proposed in Chapter 2 could be applied to other phenotype problems by, for example, focusing on other target attributes or using other data sources.
- The methodology for patient phenotyping proposed in Chapter 2 could be packaged along with a graphic interface for ease of use.
- The new and efficient SD algorithm proposed in Chapter 3 could be further improved by, for example, using pruning based on Close and Close-on-the-positive subgroups.
- The two algorithms proposed in Chapter 4 could be further improved by, for example, not using a collection of candidate subgroups loaded a-priori.
- New algorithms with which to successfully tackle the new problem of mining diverse top-k patient phenotypes could be designed.

- The 'subgroups' Python library could be improved by adding more heuristic SD algorithms such as CN2-SD or Apriori-SD, by implementing evolutionary SD algorithms such as SDIGA, or by adding the SDD++ algorithm in order to generate a single subgroup list.

Most of the contents included in this PhD thesis were published in the following peer-reviewed conferences and journals:

- Chapter 2:
  - CONFERENCE: **Antonio Lopez-Martinez-Carrasco**, Jose M. Juarez, Manuel Campos, Antonio Morales Nicolás, Francisco Palacios, and Lucía López-Rodríguez. "Interpretable Patient Subgrouping Using Trace-Based Clustering". In: *17th International Conference on Artificial Intelligence in Medicine*, 2019, pp. 269–274.
  - JOURNAL: **Antonio Lopez-Martinez-Carrasco**, Jose M. Juarez, Manuel Campos, and Bernardo Canovas-Segura. "A methodology based on Trace-based clustering for patient phenotyping". In: *Knowledge-Based Systems*, 232 (2021), p. 107469.
- Chapter 3:
  - CONFERENCE: **Antonio Lopez-Martinez-Carrasco**, Jose M. Juarez, Manuel Campos, and Bernardo Canovas-Segura. "Phenotypes for Resistant Bacteria Infections Using an Efficient Subgroup Discovery Algorithm". In: *19th International Conference on Artificial Intelligence in Medicine*, 2021, pp. 246-251.
  - JOURNAL: **Antonio Lopez-Martinez-Carrasco**, Jose M. Juarez, Manuel Campos, and Bernardo Canovas-Segura. "VLSD - An efficient Subgroup Discovery algorithm based on Equivalence Classes and Optimistic Estimate". In: *Algorithms*, 16 (2023), p. 274.
- Chapter 4:
  - CONFERENCE: **Antonio Lopez-Martinez-Carrasco**, Hugo Manuel Proença, Jose M. Juarez, Matthijs van Leeuwen, and Manuel Campos. "Discovering Diverse Top-K Characteristic Lists". In: *Advances in Intelligent Data Analysis XXI - 21st International Symposium on Intelligent Data Analysis*, 2023, pp. 262-273.

  - CONFERENCE: **Antonio Lopez-Martinez-Carrasco**, Hugo Manuel Proença, Jose M. Juarez, Matthijs van Leeuwen, and Manuel Campos. “Novel approach for phenotyping based on diverse top-k subgroup lists”. In: *21st International Conference on Artificial Intelligence in Medicine*, 2023, pp. 45-50.

Finally, other publications related to this work are enumerated as follows:

- CONFERENCE: Bernardo Canovas-Segura, Antonio Morales Nicolás, **Antonio Lopez-Martinez-Carrasco**, Manuel Campos, Jose M. Juarez, Lucía López-Rodríguez, and Francisco Palacios. “Exploring Antimicrobial Resistance Prediction Using Post-hoc Interpretable Methods”. In: *Artificial Intelligence in Medicine: Knowledge Representation and Transparent and Explainable Systems*, 2019, pp. 93-107.
- CONFERENCE: Bernardo Canovas-Segura, Antonio Morales Nicolás, **Antonio Lopez-Martinez-Carrasco**, Manuel Campos, Jose M. Juarez, Lucía López-Rodríguez, and Francisco Palacios. “Improving Interpretable Prediction Models for Antimicrobial Resistance”. In: *IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS)*, 2019, pp. 543-546.

Bibliography

Alpaydin, E. (2004). *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Atzmueller, M. (2015). Subgroup Discovery - Advanced Review. *WIREs: Data Mining and Knowledge Discovery*, *5*(1), 35-49.

Atzmueller, M., & Puppe, F. (2006). SD-Map – A Fast Algorithm for Exhaustive Subgroup Discovery. In *Knowledge Discovery in Databases: PKDD 2006* (p. 6-17).

Atzmüller, M., Puppe, F., & Buscher, H.-P. (2005, 01). Exploiting Background Knowledge for Knowledge-Intensive Subgroup Discovery. In *Ijcai international joint conference on artificial intelligence* (p. 647-652).

Banerjee, A., & Dave, R. N. (2004). Validating clusters using the Hopkins statistic. In *Procs. of the ieee international conference on fuzzy systems* (Vol. 1, pp. 149–153).

Boongoen, T., & Iam-On, N. (2018). Cluster ensembles: A survey of approaches with recent extensions and applications. *Computer Science Review*, *28*, 1 - 25.

Chen, X., Garcelon, N., Neuraz, A., Billot, K., Lelarge, M., Bonald, T., Garcia, H., Martin, Y., Benoit, V., Vincent, M., Faour, H., Douillet, M., Lyonnet, S., Saunier, S., & Burgun, A. (2019, 10). Phenotypic similarity for rare disease: Ciliopathy diagnoses and subtyping. *Journal of Biomedical Informatics*, *100*, 103308.

Dice, L. R. (1945). Measures of the Amount of Ecologic Association Between Species. *Ecology*, *26*(3), 297-302.

Doron, S., & Davidson, L. E. (2011, Nov 01). Antimicrobial stewardship. *Mayo Clinic Proceedings*, *86*(11), 1113-1123.

Elbattah, M., & Molloy, O. (2017, 02). Clustering-Aided Approach for Predicting Patient Outcomes with Application to Elderly Healthcare in Ireland. In *Procs. of the AAAI-17 Joint Workshop on Health Intelligence* (p. 533-541).

Fayyad, U. M., & Irani, K. B. (1993). Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *IJCAI*.

Fournier-Viger, P., Gomariz, A., Campos, M., & Thomas, R. (2014). Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information. In *Advances in Knowledge*

Discovery and Data Mining - 18th Pacific-Asia Conference, PAKDD 2014, Tainan, Taiwan, May 13-16, 2014. Proceedings, Part I (Vol. 8443, pp. 40–52).

Friedman, J., & Fisher, N. (1999, 04). Bump hunting in high-dimensional data. *Statistics and Computing*, *9*.

Fröhlich, H., Balling, R., Beerenwinkel, N., Kohlbacher, O., Kumar, S., Lengauer, T., Maathuis, M., Moreau, Y., Murphy, S., Przytycka, T., Rebhan, M., Röst, H., Schuppert, A., Schwab, M., Spang, R., Stekhoven, D., Sun, J., Weber, A., Ziemek, D., & Zupan, B. (2018, 12). From hype to reality: Data science enabling personalized medicine. *BMC Medicine*, *16*, 150.

Gamberger, D., & Lavrac, N. (2002, Dec). Expert-Guided Subgroup Discovery: Methodology and Application. *Journal of Artificial Intelligence Research*, *17*, 501–527.

Garriga, G., Kralj Novak, P., & Lavrac, N. (2006, 09). Closed Sets for Labeled Data. In *The journal of machine learning research* (Vol. 9, p. 163-174).

Giurcaneanu, C. D., Mircean, C., Fuller, G. N., & Tabus, I. (2006). Finding functional structures in ggioma gene-expressions using gene shaving clustering and mdl principle. In *Computational and statistical approaches to genomics* (pp. 89–118). Springer US.

Glaab, E., Bacardit, J., Garibaldi, J. M., & Krasnogor, N. (2012, 07). Using rule-based machine learning for candidate disease gene prioritization and sample classification of cancer gene expression data. *PLOS ONE*, *7*(7), 1-18.

Glueck, M., Naeini, M. P., Doshi-Velez, F., Chevalier, F., Khan, A., Wigdor, D., & Brudno, M. (2018). Phenolines: Phenotype comparison visualizations for disease subtyping via topic models. *IEEE Transactions on Visualization and Computer Graphics*, *24*(1), 371-381.

Grosskreutz, H., Rüping, S., & Wrobel, S. (2008). Tight Optimistic Estimates for Fast Subgroup Discovery. In *Proc. of Machine Learning and Knowledge Discovery in Databases (ECML PKDD)* (p. 440-456).

Grünwald, P. D. (2007). *The Minimum Description Length Principle* (Vol. 1). The MIT Press.

Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001, Dec 01). On Clustering Validation Techniques. *Journal of Intelligent Information Systems*, *17*(2), 107–145.

Han, J., Pei, J., & Yin, Y. (2000, May). Mining Frequent Patterns without Candidate Generation. *SIGMOD Rec.*, *29*(2), 1–12.

Herrera, F., Carmona, C. J., González, P., & Del Jesus, M. J. (2011, Dec). An overview on subgroup discovery: Foundations and applications. *Knowledge and Information*

Systems, *29*, 495-525.

Hielscher, T., Niemann, U., Preim, B., Völzke, H., Ittermann, T., & Spiliopoulou, M. (2018). A framework for expert-driven subpopulation discovery and evaluation using subspace clustering for epidemiological data. *Expert Systems with Applications*, *113*, 147–160.

Hoeper, M. M., Pausch, C., Grünig, E., Klose, H., Staehler, G., Huscher, D., Pittrow, D., Olsson, K. M., Vizza, C. D., Gall, H., Benjamin, N., Distler, O., Opitz, C., Gibbs, J. S. R., Delcroix, M., Ghofrani, H. A., Rosenkranz, S., Ewert, R., Kaemmerer, H., Lange, T. J., Kabitz, H.-J., Skowasch, D., Skride, A., Jureviciene, E., Paleviciute, E., Miliauskas, S., Claussen, M., Behr, J., Milger, K., Halank, M., Wilkens, H., Wirtz, H., Pfeuffer-Jovic, E., Harbaum, L., Scholtz, W., Dumitrescu, D., Bruch, L., Coghlan, G., Neurohr, C., Tsangaris, I., Gorenflo, M., Scelsi, L., Vonk-Noordegraaf, A., Ulrich, S., & Held, M. (2020). Idiopathic pulmonary arterial hypertension phenotypes determined by cluster analysis from the compera registry. *The Journal of Heart and Lung Transplantation*, *39*(12), 1435-1444.

Johnson, A., Pollard, T., Shen, L., Lehman, L.-w., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L., & Mark, R. (2016, 05). MIMIC-III, a freely accessible critical care database. *Scientific Data*, *3*, 160035.

Jorge, A. M., Pereira, F., & Azevedo, P. J. (2006). Visual Interactive Subgroup Discovery with Numerical Properties of Interest. In *Discovery science* (p. 301-305).

Kavšek, B., Lavrac, N., & Jovanoski, V. (2008, 06). APRIORI-SD: Adapting association rule learning to subgroup discovery. In *International symposium on intelligent data analysis* (Vol. 20, p. 230-241).

Kim, B., Lee, H., & Kang, P. (2018). Integrating cluster validity indices based on data envelopment analysis. *Applied Soft Computing*, *64*, 94 - 108.

Klösgen, W., & May, M. (2002). Census Data Mining - An Application. In *Proceedings of the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2002)* (p. 733–739).

Klösgen, W. (1996). Explora: A Multipattern and Multistrategy Discovery Assistant. In *Advances in knowledge discovery and data mining* (p. 249-271).

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (p. 1675–1684).

Lavrac, N., & Gamberger, D. (2004, 01). Relevancy in Constraint-Based Subgroup Discovery. In *European workshop on inductive databases and constraint based*

mining (p. 243-266).

Lavrac, N., Kavsek, B., Flach, P. A., & Todorovski, L. (2004). Subgroup Discovery with CN2-SD. *J. Mach. Learn. Res.*, *5*, 153-188.

Lavrac, N., Železný, F., & Flach, P. (2003, 03). RSD: Relational Subgroup Discovery through First-Order Feature Construction. In *Lecture notes in artificial intelligence (subseries of lecture notes in computer science)* (Vol. 2583, p. 149-165).

Le, T., Nguyen, T.-L., Huynh, B., Nguyen, H., Hong, T.-P., & Snasel, V. (2021, 12). Mining colossal patterns with length constraints. *Applied Intelligence*, *51*, 1-12.

Lei, Y., Bezdek, J. C., Romano, S., Vinh, N. X., Chan, J., & Bailey, J. (2017). Ground truth bias in external cluster validity indices. *Pattern Recognition*, *65*, 58 - 70.

Lemmerich, F., Atzmüller, M., & Puppe, F. (2015). Fast exhaustive subgroup discovery with numerical target concepts. *Data Mining and Knowledge Discovery*, *30*, 711-762.

Lemmerich, F., Rohlfs, M., & Atzmüller, M. (2010, 01). Fast Discovery of Relevant Subgroup Patterns. In *Proceedings of the 23rd international florida artificial intelligence research society conference, flairs-23.*

Liao, M., Li, Y., Kianifard, F., Obi, E., & Arcona, S. (2016, 12). Cluster analysis and its application to healthcare claims data: a study of end-stage renal disease patients who initiated hemodialysis. *BMC Nephrology*, *17*, 25.

Mitchell, T. M. (1997). *Machine learning* (Vol. 1) (No. 9). McGraw-Hill New York.

Mueller, M., Rosales, R., Steck, H., Krishnan, S., Rao, B., & Kramer, S. (2009). Subgroup Discovery for Test Selection: A Novel Approach and Its Application to Breast Cancer Diagnosis. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII* (p. 119–130).

Mühlbacher, T., & Piringer, H. (2013). A Partition-Based Framework for Building and Validating Regression Models. *IEEE Transactions on Visualization and Computer Graphics*, *19*(12), 1962-1971.

Mühlbacher, T., Piringer, H., et al. (2014). Opening the Black Box: Strategies for Increased User Involvement in Existing Algorithm Implementations. *IEEE Transactions on Visualization and Computer Graphics*, *20*(12), 1643-1652.

Nannings, B., Abu-Hanna, A., & Jonge, E. (2008, 05). Applying PRIM (Patient Rule Induction Method) and logistic regression for selecting high-risk subgroups in very elderly ICU patients. *International Journal of Medical Informatics*, *77*, 272-9.

Nilsson, N. J. (1998). *Artificial intelligence: A new synthesis*. Morgan Kaufmann Publishers Inc.

Nouioua, M., Fournier Viger, P., Wu, C.-W., Lin, C.-W., & Gan, W. (2021, 10). FHUQI-Miner: Fast high utility quantitative itemset mining. *Applied Intelligence*, *51*, 1-25.

Proença, H. M., Grünwald, P., Bäck, T., & Leeuwen, M. v. (2021). Discovering Outstanding Subgroup Lists for Numeric Targets Using MDL. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2020)* (pp. 19–35).

Proença, H. M., Grünwald, P., Bäck, T., & van Leeuwen, M. (2022, Sep 01). Robust subgroup discovery. *Data Mining and Knowledge Discovery*, *36*(5), 1885-1970.

Qu, J.-F., Fournier-Viger, P., Liu, M., Hang, B., & Wang, F. (2020). Mining high utility itemsets using extended chain structure and utility machine. *Knowledge-Based Systems*, *208*, 106457.

Russell, S., & Norvig, P. (2020). *Artificial intelligence: A modern approach (4th edition)*. Pearson.

Salmanpour, M. R., Shamsaei, M., Saberi, A., Hajianfar, G., Soltanian-Zadeh, H., & Rahmim, A. (2021). Robust identification of parkinson's disease subtypes using radiomics and hybrid machine learning. *Computers in Biology and Medicine*, *129*, 104142.

Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O. P., Tiwari, A., Er, M. J., Ding, W., & Lin, C.-T. (2017). A review of clustering techniques and developments. *Neurocomputing*, *267*, 664 - 681.

Segura, B. C., Morales, A., Juarez, J. M., Campos, M., & Palacios, F. (2020). Waspss: A clinical decision support system for antimicrobial stewardship. In *Recent advances in digital system diagnosis and management of healthcare* (chap. 8). IntechOpen.

Silitonga, P. (2018, 03). Clustering of Patient Disease Data by Using K-Means Clustering. *International Journal of Computer Science and Information Security*, *15*, 219-221.

Stiglic, G., & Kokol, P. (2012, 07). Discovering Subgroups Using Descriptive Models of Adverse Outcomes in Medical Care. *Methods of Information in Medicine*, *51*, 348–52.

Theodoridis, S., & Koutroumbas, K. (2008). *Patter Recognition* (4th ed.). Academic Press.

Uddin, M., Wang, Y., & Woodbury-Smith, M. (2019, Nov 21). Artificial intelligence for precision medicine in neurodevelopmental disorders. *npj Digital Medicine*, *2*(1), 112.

Umek, L., Zupan, B., Toplak, M., Morin, A., Chauchat, J.-H., Makovec, G., & Smrke, D. (2009). Subgroup Discovery in Data Sets with Multi–dimensional Responses: A Method and a Case Study in Traumatology. In *Procs. of the artificial intelligence*

in medicine conference, aime 2009, lecture notes in computer science, vol 5651 (pp. 265–274). Springer Berlin Heidelberg.

Vega-Pons, S., & Ruiz-Shulcloper, J. (2011, 05). A Survey of Clustering Ensemble Algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, *25*, 337-372.

Ventura, S., & Luna, J. M. (2018). *Supervised Descriptive Pattern Mining*. Springer.

Wang, Y., Zhao, Y., Therneau, T. M., Atkinson, E. J., Tafti, A. P., Zhang, N., Amin, S., Limper, A. H., Khosla, S., & Liu, H. (2020). Unsupervised machine learning for the discovery of latent disease clusters and patient subgroups using electronic health records. *Journal of Biomedical Informatics*, *102*, 103364.

WHO Regional Office for Europe, & European Centre for Disease Prevention and Control. (2022). *Antimicrobial resistance surveillance in Europe.*

Wojczynski, M. K., & Tiwari, H. K. (2008). Definition of Phenotype. In *Genetic dissection of complex traits* (Vol. 60, p. 75-105). Academic Press.

Wrobel, S. (1997). An algorithm for multi-relational discovery of subgroups. In *Principles of data mining and knowledge discovery* (p. 78-87).

Zaki, M. J., Parthasarathy, S., Ogihara, M., & Li, W. (1997). Parallel Algorithms for Discovery of Association Rules. *Data Mining and Knowledge Discovery*, *1*(4), 343-373.

APPENDIX A

Installation and use of the 'subgroups' library

This appendix provides a user guide in order to take the first steps with the 'subgroups' Python library and use the algorithms that it includes.

A.1 First steps

The first steps in order to work with the 'subgroups' library are to download and install it. On the one hand, the source code is currently hosted on GitHub at [1] and can be downloaded and manually installed. On the other, the library is also packaged and hosted on PyPI and can be automatically installed as follows:

```
pip install subgroups
```

After the installation, the next step is to test whether the library is successfully installed and works properly. For that, a collection of tests can be executed in the Python interpreter as follows:

```
import subgroups.tests as st
st.run_all_tests()
```

At this point, the library is correctly installed and ready to be used.

[1] https://github.com/antoniolopezmc/subgroups

A.2 SDMap algorithm

An example of the utilization of the SDMap algorithm is shown as follows:

```
# Import the needed libraries.
import pandas as pd
from subgroups.quality_measures import WRAcc
from subgroups.algorithms import SDMap
# Dataset and target.
dataset = pd.DataFrame({'att1': ['v3', 'v2', 'v1'], 'att2': ['v1', 'v2',
    'v3'], 'att3': ['v2', 'v1', 'v1'], 'class': ['no', 'yes', 'no']})
target = ('class', 'yes')
# Create the SDMap object.
alg = SDMap(quality_measure = WRAcc(), minimum_quality_measure_value = -1,
    minimum_n = 0, write_results_in_file = True, file_path = "./results.txt")
# Run the algorithm.
alg.fit(dataset, target)
# Show the results.
print("Selected subgroups: " + str(alg.selected_subgroups))
print("Unselected subgroups: " + str(alg.unselected_subgroups))
print("Visited nodes: " + str(alg.selected_subgroups+alg.unselected_subgroups))
```

After executing the previous code, the output is the following one:

```
Selected subgroups: 20
Unselected subgroups: 0
Visited nodes: 20
```

Moreover, a line from the output file is the following one:

```
Description: [att3 = 'v1'], Target: class = 'yes' ; Quality Measure WRAcc =
    0.11111111111111112 ; tp = 1 ; fp = 1 ; TP = 1 ; FP = 2
```

The output file contains all subgroups generated by the SDMap algorithm along with other metrics. More precisely, each line contains a subgroup (description and target), the quality measure value, and the tp, fp, TP and FP parameter values.

A.3 VLSD algorithm

An example of the utilization of the VLSD algorithm is shown as follows:

```
# Import the needed libraries.
import pandas as pd
from subgroups.quality_measures import WRAcc
from subgroups.quality_measures import WRAccOptimisticEstimate1
from subgroups.algorithms import VLSD
# Dataset and target.
dataset = pd.DataFrame({'att1': ['v3', 'v2', 'v1'], 'att2': ['v1', 'v2',
    'v3'], 'att3': ['v2', 'v1', 'v1'], 'class': ['no', 'yes', 'no']})
target = ('class', 'yes')
# Create the VLSD object.
alg = VLSD(quality_measure = WRAcc(), q_minimum_threshold = -1,
    optimistic_estimate = WRAccOptimisticEstimate1(), oe_minimum_threshold =
    -1, sort_criterion_in_s1 = VLSD.SORT_CRITERION_NO_ORDER,
    sort_criterion_in_other_sizes = VLSD.SORT_CRITERION_NO_ORDER,
    vertical_lists_implementation = VLSD.VERTICAL_LISTS_WITH_BITSETS,
    write_results_in_file = True, file_path = "./results.txt")
# Run the algorithm.
alg.fit(dataset, target)
# Show the results.
print("Selected subgroups: " + str(alg.selected_subgroups))
print("Unselected subgroups: " + str(alg.unselected_subgroups))
print("Visited nodes: " + str(alg.selected_subgroups+alg.unselected_subgroups))
```

After executing the previous code, the output is the following one:

```
Selected subgroups: 20
Unselected subgroups: 0
Visited nodes: 20
```

Moreover, a line from the output file is the following one:

```
Description: [att3 = 'v1'], Target: class = 'yes' ; Sequence of instances tp =
    bitarray('010') ; Sequence of instances fp = bitarray('001') ; Quality
    Measure WRAcc = 0.11111111111111112 ; Optimistic Estimate
```

```
WRAccOptimisticEstimate1 = 0.33333333333333337 ; tp = 1 ; fp = 1 ; TP = 1
; FP = 2
```

The output file contains all subgroups generated by the VLSD algorithm along with other elements. More precisely, each line contains a subgroup (description and target), the positive bitarray of this subgroup (i.e., for each instance from the dataset, whether it contains the subgroup description and is positive), the negative bitarray of this subgroup (i.e., for each instance from the dataset, whether it contains the subgroup description and is negative), the quality measure value, the optimistic estimate value, and the tp, fp, TP and FP parameter values.

A.4 GMSL algorithm

An example of the utilization of the GMSL algorithm is shown as follows:

```
# Import the needed libraries.
import pandas as pd
from subgroups.algorithms import GMSL
# Dataset and target.
dataset = pd.read_csv("heart-disease.csv")
target = ('HeartDisease', 'yes')
# Create the GMSL object.
alg = GMSL(input_file_path = "subgroups.txt", max_sl = 5, beta = 0.0,
    output_file_path = "subgroup_lists.txt")
# Run the algorithm.
alg.fit(dataset, target)
```

An example of an output file generated after executing this algorithm is shown as follows:

```
Dataset information:
  - Number of instances: 918.
  - Number of positive instances: 508.
  - Number of negative instances: 410.
  - Total number of attributes (including the target): 12.


```

```
Reading input file.
Read subgroups: 7145.
Input file read.


## Subgroup list (5 subgroups) ##
s1: Description: [ChestPainType = 'ASY', ST_Slope = 'Flat', Sex = 'M'],
    Target: HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 259
  - negative instances covered: 15
  - total instances covered: 274
  Considering it individually:
  - positive instances covered: 259
  - negative instances covered: 15
  - total instances covered: 274
s2: Description: [Cholesterol = '<124.5', FastingBS>120mg/dl = 'yes'], Target:
    HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 47
  - negative instances covered: 1
  - total instances covered: 48
  Considering it individually:
  - positive instances covered: 90
  - negative instances covered: 1
  - total instances covered: 91
s3: Description: [RestingECG = 'Normal', Sex = 'M'], Target: HeartDisease =
    'yes'
  Considering its position in the list:
  - positive instances covered: 75
  - negative instances covered: 169
  - total instances covered: 244
  Considering it individually:
  - positive instances covered: 256
  - negative instances covered: 178
  - total instances covered: 434
s4: Description: [Age = '>=54.5', ExerciseAngina = 'Y', Oldpeak = '>=0.85'],
```

```
    Target: HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 47
  - negative instances covered: 5
  - total instances covered: 52
  Considering it individually:
  - positive instances covered: 159
  - negative instances covered: 13
  - total instances covered: 172
s5: Description: [RestingECG = 'ST'], Target: HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 22
  - negative instances covered: 53
  - total instances covered: 75
  Considering it individually:
  - positive instances covered: 117
  - negative instances covered: 61
  - total instances covered: 178
default rule:
  positive instances covered: 58
  negative instances covered: 167
  total instances covered: 225

## Subgroup list (2 subgroups) ##
s1: Description: [ST_Slope = 'Flat', Sex = 'M'], Target: HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 342
  - negative instances covered: 43
  - total instances covered: 385
  Considering it individually:
  - positive instances covered: 342
  - negative instances covered: 43
  - total instances covered: 385
s2: Description: [Sex = 'M'], Target: HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 116
  - negative instances covered: 224
```

```
78   - total instances covered: 340
79   Considering it individually:
80   - positive instances covered: 458
81   - negative instances covered: 267
82   - total instances covered: 725
83 default rule:
84   positive instances covered: 50
85   negative instances covered: 143
86   total instances covered: 193
87
88 ## Subgroup list (3 subgroups) ##
89 s1: Description: [ChestPainType = 'ASY', ST_Slope = 'Flat'], Target:
       HeartDisease = 'yes'
90   Considering its position in the list:
91   - positive instances covered: 289
92   - negative instances covered: 29
93   - total instances covered: 318
94   Considering it individually:
95   - positive instances covered: 289
96   - negative instances covered: 29
97   - total instances covered: 318
98 s2: Description: [ExerciseAngina = 'Y', MaxHR = '<132.5'], Target:
       HeartDisease = 'yes'
99   Considering its position in the list:
100  - positive instances covered: 78
101  - negative instances covered: 13
102  - total instances covered: 91
103  Considering it individually:
104  - positive instances covered: 225
105  - negative instances covered: 23
106  - total instances covered: 248
107 s3: Description: [Cholesterol = '>=124.5', FastingBS>120mg/dl = 'no', MaxHR =
       '<132.5'], Target: HeartDisease = 'yes'
108  Considering its position in the list:
109  - positive instances covered: 6
110  - negative instances covered: 53
111  - total instances covered: 59
```

```
  Considering it individually:
  - positive instances covered: 151
  - negative instances covered: 77
  - total instances covered: 228
default rule:
  positive instances covered: 135
  negative instances covered: 315
  total instances covered: 450

## Subgroup list (3 subgroups) ##
s1: Description: [ChestPainType = 'ASY', ExerciseAngina = 'Y'], Target:
    HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 268
  - negative instances covered: 29
  - total instances covered: 297
  Considering it individually:
  - positive instances covered: 268
  - negative instances covered: 29
  - total instances covered: 297
s2: Description: [MaxHR = '<132.5', ST_Slope = 'Flat'], Target: HeartDisease =
    'yes'
  Considering its position in the list:
  - positive instances covered: 106
  - negative instances covered: 17
  - total instances covered: 123
  Considering it individually:
  - positive instances covered: 253
  - negative instances covered: 27
  - total instances covered: 280
s3: Description: [Cholesterol = '>=124.5', MaxHR = '<132.5'], Target:
    HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 4
  - negative instances covered: 58
  - total instances covered: 62
  Considering it individually:
```

```
  - positive instances covered: 199
  - negative instances covered: 87
  - total instances covered: 286
default rule:
  positive instances covered: 130
  negative instances covered: 306
  total instances covered: 436

## Subgroup list (2 subgroups) ##
s1: Description: [ChestPainType = 'ASY', Oldpeak = '>=0.85'], Target:
    HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 268
  - negative instances covered: 31
  - total instances covered: 299
  Considering it individually:
  - positive instances covered: 268
  - negative instances covered: 31
  - total instances covered: 299
s2: Description: [Cholesterol = '<124.5', ST_Slope = 'Flat'], Target:
    HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 65
  - negative instances covered: 4
  - total instances covered: 69
  Considering it individually:
  - positive instances covered: 106
  - negative instances covered: 4
  - total instances covered: 110
default rule:
  positive instances covered: 175
  negative instances covered: 375
  total instances covered: 550
```

The output file contains some basic information about the input dataset and the read subgroups from the input file. Furthermore, it also contains the subgroup lists generated.

A.5 DSLM algorithm

An example of the utilization of the DSLM algorithm is shown as follows:

```
# Import the needed libraries.
import pandas as pd
from subgroups.algorithms import DSLM
# Dataset and target.
dataset = pd.read_csv("heart-disease.csv")
target = ('HeartDisease', 'yes')
# Create the DSLM object.
alg = DSLM(input_file_path = "subgroups.txt", max_sl = 5, sl_max_size = 5,
    beta = 0.0, maximum_positive_overlap = 0.25, maximum_negative_overlap =
    0.25, output_file_path = "subgroup_lists.txt")
# Run the algorithm.
alg.fit(dataset, target)
```

An example of an output file generated after executing this algorithm is shown as follows:

```
Dataset information:
  - Number of instances: 918.
  - Number of positive instances: 508.
  - Number of negative instances: 410.
  - Total number of attributes (including the target): 12.


Reading input file.
Read subgroups: 7145.
Input file read.


## Subgroup list (4 subgroups) ##
s1: Description: [ChestPainType = 'ASY', ST_Slope = 'Flat', Sex = 'M'],
    Target: HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 259
  - negative instances covered: 15
```

```
  - total instances covered: 274
  Considering it individually:
  - positive instances covered: 259
  - negative instances covered: 15
  - total instances covered: 274
s2: Description: [Cholesterol = '<124.5', FastingBS>120mg/dl = 'yes'], Target:
    HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 47
  - negative instances covered: 1
  - total instances covered: 48
  Considering it individually:
  - positive instances covered: 90
  - negative instances covered: 1
  - total instances covered: 91
s3: Description: [Age = '>=54.5', Cholesterol = '>=124.5', ExerciseAngina =
    'Y', Oldpeak = '>=0.85'], Target: HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 61
  - negative instances covered: 7
  - total instances covered: 68
  Considering it individually:
  - positive instances covered: 123
  - negative instances covered: 10
  - total instances covered: 133
s4: Description: [Age = '<54.5', MaxHR = '<132.5'], Target: HeartDisease =
    'yes'
  Considering its position in the list:
  - positive instances covered: 18
  - negative instances covered: 48
  - total instances covered: 66
  Considering it individually:
  - positive instances covered: 108
  - negative instances covered: 51
  - total instances covered: 159
default rule:
  positive instances covered: 123
```

```
  negative instances covered: 339
  total instances covered: 462

## Subgroup list (3 subgroups) ##
s1: Description: [MaxHR = '<132.5', ST_Slope = 'Flat', Sex = 'M'], Target:
    HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 238
  - negative instances covered: 14
  - total instances covered: 252
  Considering it individually:
  - positive instances covered: 238
  - negative instances covered: 14
  - total instances covered: 252
s2: Description: [ChestPainType = 'ASY', MaxHR = '>=132.5,<159.5', Oldpeak =
    '>=0.85'], Target: HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 73
  - negative instances covered: 10
  - total instances covered: 83
  Considering it individually:
  - positive instances covered: 73
  - negative instances covered: 10
  - total instances covered: 83
s3: Description: [ChestPainType = 'ASY', Cholesterol = '<124.5'], Target:
    HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 46
  - negative instances covered: 6
  - total instances covered: 52
  Considering it individually:
  - positive instances covered: 122
  - negative instances covered: 9
  - total instances covered: 131
default rule:
  positive instances covered: 151
  negative instances covered: 380
```

```
  total instances covered: 531

## Subgroup list (2 subgroups) ##
s1: Description: [ChestPainType = 'ASY', ExerciseAngina = 'Y', Sex = 'M'],
    Target: HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 244
  - negative instances covered: 20
  - total instances covered: 264
  Considering it individually:
  - positive instances covered: 244
  - negative instances covered: 20
  - total instances covered: 264
s2: Description: [Cholesterol = '<124.5', ST_Slope = 'Flat'], Target:
    HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 61
  - negative instances covered: 4
  - total instances covered: 65
  Considering it individually:
  - positive instances covered: 106
  - negative instances covered: 4
  - total instances covered: 110
default rule:
  positive instances covered: 203
  negative instances covered: 386
  total instances covered: 589

## Subgroup list (2 subgroups) ##
s1: Description: [ChestPainType = 'ASY', Oldpeak = '>=0.85', Sex = 'M'],
    Target: HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 238
  - negative instances covered: 20
  - total instances covered: 258
  Considering it individually:
  - positive instances covered: 238
```

```
  - negative instances covered: 20
  - total instances covered: 258
s2: Description: [Oldpeak = '<0.85', ST_Slope = 'Flat', Sex = 'M'], Target:
    HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 117
  - negative instances covered: 15
  - total instances covered: 132
  Considering it individually:
  - positive instances covered: 117
  - negative instances covered: 15
  - total instances covered: 132
default rule:
  positive instances covered: 153
  negative instances covered: 375
  total instances covered: 528

## Subgroup list (2 subgroups) ##
s1: Description: [ChestPainType = 'ASY', ExerciseAngina = 'Y', Oldpeak =
    '>=0.85'], Target: HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 215
  - negative instances covered: 15
  - total instances covered: 230
  Considering it individually:
  - positive instances covered: 215
  - negative instances covered: 15
  - total instances covered: 230
s2: Description: [FastingBS>120mg/dl = 'yes', ST_Slope = 'Flat'], Target:
    HeartDisease = 'yes'
  Considering its position in the list:
  - positive instances covered: 81
  - negative instances covered: 7
  - total instances covered: 88
  Considering it individually:
  - positive instances covered: 121
  - negative instances covered: 7
```

```
154   - total instances covered: 128
155 default rule:
156   positive instances covered: 212
157   negative instances covered: 388
158   total instances covered: 600
```

The output file contains some basic information about the input dataset and the read subgroups from the input file. Furthermore, it also contains the subgroup lists generated.

APPENDIX B

Extending the 'subgroups' library

This appendix explains how to extend the 'subgroups' library by adding new quality measures, data structures and algorithms.

After adding new functionality to the library, it is required to implement its corresponding tests in the `tests` folder (see Figure 5.2) in order to verify that this functionality is well-implemented and works properly.

B.1 Adding a new quality measure

This section describes how to add a new quality measure to the library by using the WRAcc quality measure as an example.

The first step is to create a python file in the `quality_measures` folder (see Figure 5.2) whose name is the name of the specific quality measure to implement, `wracc.py` in this case. Note that the file name is always in lowercase.

Then, the file content is the following:

```
1 """This file contains the implementation of the Weighted Relative Accuracy
    (WRAcc) quality measure.
2 """
3
4 from subgroups.quality_measures.quality_measure import QualityMeasure
5 from subgroups.exceptions import SubgroupParameterNotFoundError
6
7 # Python annotations.
```

```
8  from typing import Union
9
10 class WRAcc(QualityMeasure):
11     """This class defines the Weighted Relative Accuracy (WRAcc) quality
           measure.
12     """
13
14     _singleton = None
15     __slots__ = ()
16
17     def __new__(cls) -> 'WRAcc':
18         if WRAcc._singleton is None:
19             WRAcc._singleton = object().__new__(cls)
20         return WRAcc._singleton
21
22     def compute(self, dict_of_parameters : dict[str, Union[int, float]]) ->
           float:
23         """Method to compute the WRAcc quality measure (you can also call to
               the instance for this purpose).
24
25         :param dict_of_parameters: python dictionary which contains all the
               necessary parameters used to compute this quality measure.
26         :return: the computed value for the WRAcc quality measure.
27         """
28         if type(dict_of_parameters) is not dict:
29             raise TypeError("The type of the parameter 'dict_of_parameters'
                   must be 'dict'.")
30         if (QualityMeasure.TRUE_POSITIVES not in dict_of_parameters):
31             raise SubgroupParameterNotFoundError("The subgroup parameter 'tp'
                   is not in 'dict_of_parameters'.")
32         if (QualityMeasure.FALSE_POSITIVES not in dict_of_parameters):
33             raise SubgroupParameterNotFoundError("The subgroup parameter 'fp'
                   is not in 'dict_of_parameters'.")
34         if (QualityMeasure.TRUE_POPULATION not in dict_of_parameters):
35             raise SubgroupParameterNotFoundError("The subgroup parameter 'TP'
                   is not in 'dict_of_parameters'.")
36         if (QualityMeasure.FALSE_POPULATION not in dict_of_parameters):
```

```
37            raise SubgroupParameterNotFoundError("The subgroup parameter 'FP'
                  is not in 'dict_of_parameters'.")
38        tp = dict_of_parameters[QualityMeasure.TRUE_POSITIVES]
39        fp = dict_of_parameters[QualityMeasure.FALSE_POSITIVES]
40        TP = dict_of_parameters[QualityMeasure.TRUE_POPULATION]
41        FP = dict_of_parameters[QualityMeasure.FALSE_POPULATION]
42        return ( (tp+fp) / (TP+FP) ) * ( ( tp / (tp+fp) ) - ( TP / (TP+FP) ) )
43
44    def get_name(self) -> str:
45        """Method to get the quality measure name (equal to the class name).
46        """
47        return "WRAcc"
48
49    def optimistic_estimate_of(self) -> dict[str, QualityMeasure]:
50        """Method to get a python dictionary with the quality measures of
              which this one is an optimistic estimate.
51
52        :return: a python dictionary in which the keys are the quality measure
              names and the values are the instances of those quality measures.
53        """
54        return dict()
55
56    def __call__(self, dict_of_parameters : dict[str, Union[int, float]]) ->
          float:
57        """Compute the WRAcc quality measure.
58
59        :param dict_of_parameters: python dictionary which contains all the
              needed parameters with which to compute this quality measure.
60        :return: the computed value for the WRAcc quality measure.
61        """
62        return self.compute(dict_of_parameters)
```

This file contains only one class, which inherits from the `QualityMeasure` abstract class and whose name is the name of the specific quality measure to implement, `WRAcc` in this case. Since this class is a singleton, it contains a class attribute called `_singleton` and the `__new__` method as indicated in the previous code. At the same time, this class also overwrite the `compute`, `get_name`, `optimistic_estimate_of` and `__call__`

methods.

The `compute` method computes the quality measure value by using a set of parameters corresponding to the results obtained by the functions described in Chapter 3. This method receives as a parameter a Python dictionary containing the mentioned parameters and, if it is required by the quality measure, other parameters to compute it.

The `get_name` method returns the name of the quality measure as a string format.

The `optimistic_estimate_of` retrieves all quality measures of which this quality measure is an optimistic estimate. More precisely, this method returns a Python dictionary in which the key is the name of the optimistic estimate quality measure as a string format and the value is the quality measure object of such optimistic estimate. In this case, the WRAcc quality measure is not an optimistic estimate, meaning that this method returns an empty dictionary.

The `__call__` method is a Python magic method that, in this case, calls to the `compute` method.

After that, the last step is to add the following line in the `quality_measures/__init__.py` file:

```
1 from subgroups.quality_measures.wracc import WRAcc
```

B.2 Adding a new data structure

This section explains how to add a new data structure to the library.

The first step is to create a python file in the `data_structures` folder (see Figure 5.2) whose name is the name of the specific data structure to implement. Remember that the file name is always in lowercase.

As explained in Chapter 5, it does not exist an abstract class from which all specific data structures inherit since they do not have common properties. For this reason, the only implementation restriction is to have only one class in each file.

After that, using as an example the subgroup list data structure, the last step is to add the following line in the `data_structures/__init__.py` file:

```
1 from subgroups.data_structures.subgroup_list import SubgroupList
```

B.3 Adding a new algorithm

This section details how to add a new algorithm to the library by using the VLSD algorithm as an example.

The first step is to create a python file either in the `algorithms/subgroup_sets` folder or in the `algorithms/subgroup_lists` folder (see Figure 5.2) depending on the algorithm type to implement. The file name is the name of the specific algorithm to implement, `vlsd.py` in this case. Note that the file name is always in lowercase.

This file contains only one class, which inherits from the `Algorithm` abstract class and whose name is the name of the specific algorithm to implement, `VLSD` in this case. At the same time, this class overwrites the `fit` method, whose definition is as follows:

```
1 def fit(self, pandas_dataframe : DataFrame, target : tuple[str, str]) -> None:
```

The `fit` method takes as parameters the dataset in the form of a pandas DataFrame and a tuple indicating the target attribute name and the target value. The other algorithm parameters are passed and initialized in the constructor (`__init__` method).

After that, the last step is to add the following line in the `algorithms/__init__.py` file:

```
1 from subgroups.algorithms.subgroup_sets.vlsd import VLSD
```