UNIVERSIDAD DE SALAMANCA

FACULTAD DE CIENCIAS

DEPARTAMENTO DE ESTADÍSTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

TRABAJO DE FIN DE GRADO

USO DEL CLUSTERING ESPACIAL BASADO EN DENSIDAD DE APLICACIONES CON RUIDO EN LA MINERÍA DE DATOS. CREACIÓN DE UNA APLICACIÓN WEB CON R PARA CLASIFICAR O PREDECIR DATOS REALES

Tutor: Miguel Rodríguez Rosa

Autora: Isabel Sánchez Gutiérrez

Grado en Estadística

Curso académico 2023-24

UNIVERSIDAD DE SALAMANCA

FACULTAD DE CIENCIAS

DEPARTAMENTO DE ESTADÍSTICA

TRABAJO DE FIN DE GRADO

USO DEL CLUSTERING ESPACIAL BASADO EN DENSIDAD DE APLICACIONES CON RUIDO EN LA MINERÍA DE DATOS. CREACIÓN DE UNA APLICACIÓN WEB CON R PARA CLASIFICAR O PREDECIR DATOS REALES

Firmado por:

Autora: Isabel Sánchez Gutiérrez

Tutor: Miguel Rodríguez Rosa

Grado en Estadística

Curso académico 2023-24

Certificado del tutor TFG Grado en Estadística

D. Miguel Rodríguez Rosa, profesor del Departamento de Estadística de la Universidad de Salamanca,

HACE CONSTAR:

Que el trabajo titulado "*Uso del clustering espacial basado en densidad de aplicaciones con ruido en la minería de datos. Creación de una aplicación web con R para clasificar o predecir datos reales*", que se presenta, ha sido realizado por D.ª Isabel Sánchez Gutiérrez, con DNI 70926717F y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado en Estadística en esta Universidad.

Salamanca, a 2 de julio de 2024.

RODRIGUEZ ROSA MIGUEL - 70821913Z

Firmado digitalmente por RODRIGUEZ ROSA MIGUEL - 70821913Z
Fecha: 2024.07.02 17:02:04 +02'00'

Fdo.: Miguel Rodríguez Rosa

Índice general

Resumen

Formalmente, el clustering espacial basado en densidad de aplicaciones con ruido (DBSCAN) es un algoritmo de clustering no paramétrico que agrupa puntos de datos en función de su densidad en un espacio métrico. Su objetivo principal es encontrar áreas de alta densidad en el espacio de datos y asignar puntos a clusters en función de la conectividad de dichas áreas densas. Es especialmente útil para identificar clusters de diferentes formas y tamaños en conjuntos de datos donde la densidad varía significativamente. Además es capaz de detectar outliers, y no requiere que se especifique el número de clusters de antemano.

Este trabajo consiste en buscar un conjunto de datos y crear una aplicación web con R para llevar a cabo el procedimiento del DBSCAN, y usarlo para clasificar los datos o hacer predicciones, documentando cualquier pre-procesamiento requerido para hacer los datos adecuados para el ajuste del modelo, limpiando la base de datos, y dividiéndola en datos de entrenamiento y testeo.

Ejemplos de situaciones reales en las que usar esta técnica pueden ser: detectar patrones de uso del transporte público para analizar cómo los usuarios se agrupan en función de sus patrones de viajes, detección de clusters de lugares turísticos y descubrir áreas de alta concentración de visitantes en una ciudad en función de sus preferencias, agrupar datos de actividades deportivas y descubrir patrones de comportamiento en función de su actividad (carreras, caminatas, ciclismo) y rendimiento...

Summary

Formally, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a non-parametric clustering algorithm that groups data points based on their density in a metric space. Its main objective is to find high-density areas in the data space and assign points to clusters based on the connectivity of these dense areas. It is particularly useful for identifying clusters of different shapes and sizes in datasets where the density varies significantly. Additionally, it can detect outliers and does not require specifying the number of clusters beforehand.

The task involves selecting a dataset and creating a web app using R to perform the DBSCAN procedure, and using it to classify the data or make predictions. This includes documenting any preprocessing required to make the data suitable for model fitting, cleaning the database, and splitting it into training and testing data.

Examples of real-life situations where this technique can be used include: detecting public transport usage patterns to analyze how users group based on their travel patterns, detecting clusters of tourist spots and discovering areas of high visitor concentration in a city based on their preferences, grouping sports activity data, and discovering behavioral patterns based on their activity (running, walking, cycling) and performance...

Introducción

La disponibilidad de grandes cantidades de información y el uso extendido de herramientas informáticas han revolucionado el análisis de datos, enfocándolo hacia distintos métodos específicos agrupados bajo el nombre de Minería de Datos o Data Mining.

Según detallan Fayyad, Piatetsky-Shapiro, Smyth y Uthurusamy (1996), "la Minería de Datos puede definirse inicialmente como un proceso de descubrimiento de nuevas y significativas relaciones, patrones y tendencias al examinar grandes cantidades de datos".

La Minería de Datos es una disciplina que se centra en explorar, analizar y extraer información significativa de grandes bases de datos mediante el uso de métodos y algoritmos especializados. Combina técnicas de estadística, aprendizaje automático e inteligencia artificial para descubrir patrones ocultos, relaciones y tendencias dentro de los datos que no serían evidentes a simple vista.

La idea de descifrar información valiosa o útil de un conjunto de datos no es un concepto reciente; durante siglos, las personas han buscado métodos para extraer datos significativos de grupos de información y emplear esos hallazgos para alcanzar un objetivo útil o un beneficio.

Tradicionalmente, la exploración de los datos se llevaba a cabo principalmente a través de métodos estadísticos, implementando técnicas como la regresión y correlación. A medida que nos acercamos a finales de los años ochenta, se produce una evolución en este campo con la incorporación de métodos más avanzados como la lógica difusa, el razonamiento heurístico y las redes neuronales. Este cambio marca una expansión en las técnicas utilizadas para explorar y buscar la información, permitiendo un enfoque más amplio y sofisticado en la comprensión y extracción de patrones en los datos. En la actualidad, el aprovechamiento de todas estas técnicas ha permitido generar conocimiento a partir de conjuntos de datos.

EL PROCESO DE EXTRACCIÓN DEL CONOCIMIENTO

La Minería de Datos es una etapa del Proceso de Extracción de Conocimiento a partir de datos. El término KDD (iniciales de *Knowledge Discovery in Databases*), surge en 1989 y comprende la totalidad del procedimiento para hallar conocimientos valiosos dentro de grandes volúmenes de datos. Este concepto marca un cambio significativo en la forma en la que se percibe y utiliza la información contenida en bases de datos, y subraya la relevancia de extraer información significativa más allá de la simple acumulación de datos. En el primer estado del arte sobre el área, Fayyad, Piatetsky-Shapiro y Smyth (1996), se dice:

"La mayoría de los trabajos previos en KDD, se centraban en [...] la etapa de Minería de Datos. Sin embargo, los otros pasos son de considerable importancia para el éxito de las aplicaciones de KDD en la práctica".

Lo que resalta la importancia de incluir otros pasos en el proceso de extracción del conocimiento, tales como el preprocesamiento de los datos antes su análisis y la interpretación y formalización del conoci-

miento obtenido después de este proceso.

KDD se ha definido como “el proceso no trivial de identificación en los datos de patrones válidos, nuevos, potencialmente útiles, y finalmente comprensibles” (Fayyad, Piatetsky-Shapiro & Smyth, 1996).

Este proceso incluye varias etapas, como la preparación de datos (selección, limpieza y transformación), exploración y auditoría de los mismos, Minería de Datos propiamente dicha (desarrollo de modelos y análisis de datos), evaluación, difusión y aplicación de los modelos (output). Además, el proceso de extracción de conocimiento integra diversas técnicas (árboles de decisión, regresión lineal, redes neuronales artificiales, técnicas bayesianas, máquinas de soporte vectorial, etc.) provenientes de diferentes campos como el Aprendizaje Automático, inteligencia artificial o la Estadística. También se enfrenta a una variedad de problemas, como la clasificación, estimación, regresión o agrupamiento, entre otros.

El Aprendizaje Automático (*Machine Learning*) es una de las áreas más importantes en el campo de la Minería de Datos. Murphy (2012), lo define como “un conjunto de métodos que pueden detectar automáticamente patrones en los datos y posteriormente hacer uso de ellos para predecir datos futuros o para realizar otros tipos de toma de decisiones en condiciones de incertidumbre”.

En el Aprendizaje Supervisado, los modelos predicen el valor de un atributo de un conjunto de datos basándose en otros atributos conocidos. Utilizando datos etiquetados donde se conoce la salida deseada para cada entrada, el modelo aprende a establecer una relación entre estas salidas etiquetadas y los diferentes atributos presentes. Esta relación permite al modelo realizar predicciones precisas sobre datos nuevos cuyas etiquetas son desconocidas, utilizando el conocimiento adquirido durante el entrenamiento con ejemplos previamente etiquetados (Velimirovic, 2024).

Por otro lado, en el Aprendizaje no Supervisado el modelo se entrena con datos sin etiquetar, en los que las entradas no tienen etiquetas de salida correspondientes. El objetivo es permitir que el modelo aprenda patrones y estructuras a partir de datos sin etiquetar sin que el administrador proporcione orientación explícita.

Con estos algoritmos de Aprendizaje no Supervisado, o descubrimiento del conocimiento, se revelan patrones y tendencias en los datos actuales. El descubrimiento de esa información sirve para llevar a cabo acciones y obtener un beneficio de ellas.

Dado que el modelo solo recibe datos de entrada, los algoritmos de Aprendizaje no Supervisado intentan descubrir patrones, correlaciones y estructuras de datos ocultos por sí solos. Este enfoque proporciona información sobre las propiedades intrínsecas de los datos que no son visibles para un analista humano. Los modelos entrenados con Aprendizaje no Supervisado destacan en las siguientes tareas:

- Reducción de dimensionalidad: Disminuyen la cantidad de variables manteniendo la mayor cantidad posible de información original.
- Asociación: Descubren relaciones entre variables en los datos.
- Detección de anomalías: Identifican elementos y eventos inusuales dentro de un conjunto datos.
- Agrupación (clustering): Clasifican datos similares en grupos basados en sus similitudes o diferencias. Destacan el clustering *k-means*, jerárquico o DBSCAN.

TÉCNICAS DE MINERÍA DE DATOS

La clasificación inicial de las técnicas de Minería de Datos (Pérez-López & Santín-González, 2007) distingue entre técnicas predictivas, descriptivas y auxiliares. Las técnicas predictivas diferencian entre

variables dependientes e independientes, mientras que en las técnicas descriptivas se tratan todas las variables por igual. Finalmente, las técnicas auxiliares complementan a las otras dos, ayudando a mejorar la interpretación y aplicación de los modelos predictivos y descriptivos.

Las técnicas predictivas definen el modelo basándose en un conocimiento teórico previo. El modelo obtenido para los nuevos datos debe ser verificado y contrastado tras el proceso de Minería de Datos para asegurar su validez. Este proceso incluye la identificación del modelo más adecuado a partir de los datos, la estimación de sus parámetros, la evaluación de su validez y precisión, y la utilización del modelo validado para realizar predicciones sobre variables futuras. En ocasiones, el modelo resultante es una combinación de conocimientos obtenidos antes y después de la Minería de Datos, y debe ser igualmente validado. Dentro de estas técnicas se incluyen la regresión, análisis de series temporales, análisis de la varianza y covarianza, análisis discriminante, árboles de decisión, redes neuronales, algoritmos genéticos y técnicas bayesianas.

En las técnicas descriptivas no se asigna ningún papel predeterminado a las variables y no requieren un modelo preestablecido. En su lugar, los modelos se generan automáticamente a partir del reconocimiento de patrones en los datos. En este grupo están incluidos los métodos de clustering y segmentación, las técnicas de asociación y dependencia, las técnicas de análisis exploratorio de datos y las técnicas de reducción de la dimensión y de escalamiento multidimensional.

Las técnicas auxiliares, son herramientas complementarias más superficiales y restringidas. Se fundamentan en métodos estadísticos descriptivos, consultas e informes, y están orientadas principalmente hacia la validación de los modelos predictivos y descriptivos.

En la Figura 1, se muestra una clasificación de las técnicas de Minería de Datos:

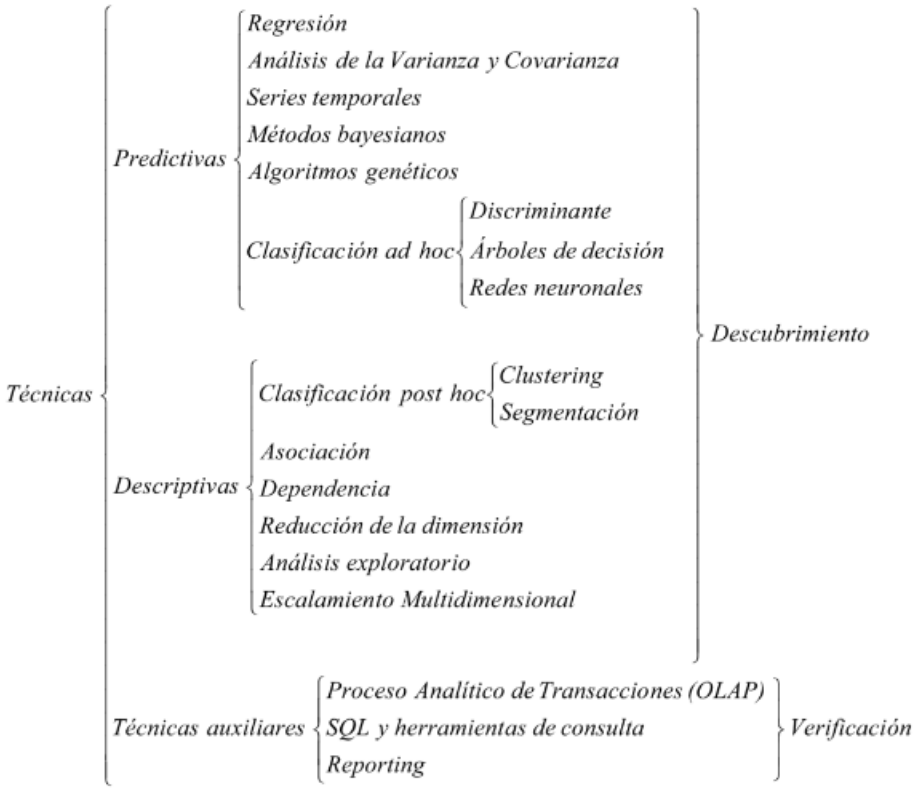

Figura 1: Clasificación de las técnicas de Minería de Datos (Fuente: Pérez-López y Santín-González (2007)).

Se observa que las técnicas de clasificación pueden integrarse tanto en los métodos predictivos como descriptivos. “La clasificación implica aprender una función que clasifica un elemento de datos en una de varias clases predefinidas” (Weiss & Kulikowski, 1991). Las técnicas predictivas de clasificación, comúnmente conocidas como *ad hoc*, clasifican observaciones dentro de grupos previamente establecidos, mientras que las técnicas descriptivas, denominadas *post hoc*, llevan a cabo la clasificación sin haber especificado previamente los grupos.

“El clustering es una tarea descriptiva común donde se busca identificar un conjunto finito de categorías o grupos para describir los datos” (Jain & Dubes, 1988; Titterington et al., 1985). Las categorías pueden ser mutuamente exclusivas y exhaustivas o consistir en una representación más rica, como categorías jerárquicas o superpuestas. Ejemplos de aplicaciones de agrupamiento en un contexto de descubrimiento de conocimiento incluyen: descubrir subpoblaciones homogéneas para consumidores en bases de datos de marketing, e identificar subcategorías de espectros de mediciones del cielo infrarrojo (Cheeseman & Stutz, 1996).

ALGORITMOS CLUSTERING

La agrupación de datos es un proceso de partición o detección de valores atípicos para encontrar un patrón, puntos u objetos. Los objetos dentro de un cluster son más similares entre sí que los objetos fuera del cluster. Webster (2008) define el análisis de conglomerados como “una técnica de clasificación estadística para descubrir si los individuos de una población se dividen en diferentes grupos mediante comparaciones cuantitativas de múltiples características”.

El conjunto de clusters obtenido del análisis puede denominarse como una agrupación. En este contexto, distintas técnicas de clustering pueden dar lugar diversas agrupaciones para el mismo grupo de datos. La partición no la realiza una persona, si no que la realiza el algoritmo. Por lo tanto, estos métodos son útiles debido a que son capaces de descubrir grupos que a priori se desconocen en los datos.

El análisis de clusters ha sido ampliamente utilizado en muchos campos como inteligencia de negocios, reconocimiento de patrones en imágenes, búsqueda en la web, biología y seguridad (Han et al., 2012).

En inteligencia de negocios, el clustering se emplea para agrupar un gran número de clientes en segmentos, donde los clientes dentro de cada uno de ellos comparten características similares significativas. Esto hace posible el desarrollo de estrategias empresariales más efectivas para mejorar la gestión de las relaciones con los clientes.

En cuanto al reconocimiento de imágenes, el clustering puede ser utilizado para identificar clusters o “subclases” en sistemas de reconocimiento de caracteres manuscritos. Por ejemplo, si se dispone de un conjunto de dígitos manuscritos, donde cada uno de ellos está etiquetado como 1, 2, 3, y así sucesivamente. Es muy posible que la manera en la que cada persona escriba el mismo número varíe. Tomando como ejemplo el número 2, algunas personas pueden anotarlo con un pequeño círculo en la parte inferior izquierda, mientras que otras no. Al aplicar un clustering, es posible identificar las subclases para el “2”, donde una de cada ellas muestra una forma diferente en la que se puede anotar el número. Utilizando diferentes modelos basados en estas subclases, se puede aumentar la precisión del reconocimiento general.

El clustering también se ha utilizado en numerosas aplicaciones relacionadas con la búsqueda en la web. Por ejemplo, una búsqueda por palabra clave puede devolver a menudo un número muy grande de resultados (incluyendo páginas irrelevantes) debido al número extremadamente grande de páginas web existentes. El clustering puede emplearse para clasificar los resultados de búsqueda en grupos, presentándolos de manera concisa. Por otro lado, también se han desarrollado métodos de clustering para clasificar documentos según temas, lo cual es una práctica muy habitual en la recuperación de información.

El análisis de clusters en Minería de Datos puede funcionar como una herramienta para entender la distribución de datos y observar las características de cada cluster. También, permite enfocarse en clusters específicos para un análisis más profundo de éstos. Alternativamente, puede servir como un paso previo para otros algoritmos, facilitando la caracterización, selección de subconjuntos de atributos y clasificación. Esta técnica ayuda a obtener una visión clara de cómo se agrupan los datos, lo cual puede resultar muy útil para estudios o aplicaciones posteriores.

Dado que un cluster es un conjunto de objetos (datos) que son similares entre sí y diferentes a los objetos en otros grupos, un cluster de objetos puede considerarse como una clase implícitamente. En este contexto, el clustering a menudo se considera un tipo de clasificación automática, es decir, que la diferencia clave con el resto de métodos de clasificación es que el clustering puede identificar los grupos de manera automática, y esta capacidad es la que hace al análisis de clusters contar con una ventaja distintiva.

El clustering también se conoce como segmentación de datos en algunos contextos, ya que divide grandes conjuntos de datos en grupos de acuerdo a su similitud. Puede ser utilizado para la detección de valores atípicos, donde estos (valores que están "muy alejados" de cualquier grupo) pueden ser de mayor interés que los casos habituales. La detección de valores atípicos se aplica en diversas áreas, como la identificación de fraudes con tarjetas de crédito y la vigilancia de actividades delictivas en el comercio electrónico. Por ejemplo, transacciones inusuales con tarjetas de crédito, tales como compras muy costosas y poco frecuentes, pueden indicar posibles fraudes. Estas aplicaciones permiten supervisar y detectar movimientos sospechosos, ayudando a prevenir acciones fraudulentas en plataformas de comercio en línea.

Actualmente, el clustering de datos está en pleno auge. Diversas áreas de investigación, como la Minería de Datos, la Estadística, el Aprendizaje Automático, las tecnologías de bases de datos espaciales, la recuperación de información, la búsqueda web y el marketing, entre otras, contribuyen a su desarrollo (Han et al., 2012). Recientemente, el análisis de clusters se ha convertido en un tema destacado en la investigación de Minería de Datos, impulsado por la inmensa cantidad de datos almacenados en bases de datos.

Dentro de las técnicas de Aprendizaje Automático, la clasificación se conoce como Aprendizaje Supervisado porque se proporcionan etiquetas de clase. Esto implica que el algoritmo de aprendizaje recibe datos sobre la clase correspondiente a cada tupla de entrenamiento. Por otro lado, el clustering se considera Aprendizaje no Supervisado, ya que no dispone de esta información sobre las etiquetas de clase. Debido a este motivo, el clustering considera como una forma de aprendizaje por observación, en lugar de un aprendizaje basado en ejemplos.

En la Minería de Datos, se ha puesto un gran énfasis en desarrollar métodos que permitan un análisis de clusters eficiente y preciso en grandes bases de datos. Las investigaciones actuales se enfocan en mejorar y adaptar estos algoritmos para que puedan manejar y procesar de manera efectiva cantidades crecientes de datos sin perder rendimiento o precisión. En otras palabras, se está trabajando en hacer que estos algoritmos sean capaces de funcionar bien incluso cuando el tamaño de los datos aumenta significativamente.

Los algoritmos de clustering crean una partición de un conjunto de datos D con n elementos en k agrupaciones, siendo k un parámetro que se proporciona como entrada para estos algoritmos, es decir, es necesario poseer algo de conocimiento previo o dominio del tema, lo cual lamentablemente no se cumple en muchas ocasiones. Los algoritmos de partición generalmente empiezan con una partición preliminar de D y a continuación aplican una estrategia de control iterativa para optimizar una función objetivo. En los algoritmos *k-means*, cada cluster se representa mediante su centro de gravedad. En contraste, en los algoritmos *k-medoids*, cada grupo se representa por uno de los objetos del cluster que está más cercano a su centro (Ester et al., 1996).

Por consiguiente, los algoritmos de partición utilizan un proceso en dos etapas. Primero, se eligen k representantes que minimizan la función objetivo. En segundo lugar, cada elemento se asigna al grupo cuyo representante sea el "más similar" al objeto en cuestión (ver Figura 2). Este segundo paso implica que la partición se asemeja a un diagrama de Voronoi, con cada grupo contenido en una de las celdas de Voronoi. De este modo, todos los grupos formados por un algoritmo de partición tienen una forma convexa, lo que resulta bastante restrictivo (Ester et al., 1996).

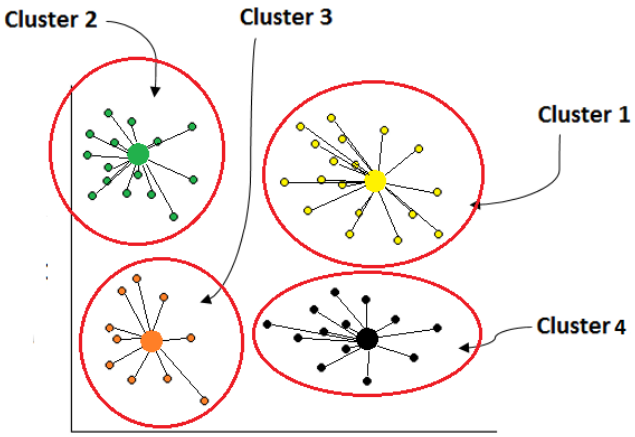

Figura 2: Clustering *k-means*.

Ng y Han (1994) exploran algoritmos de partición usando el KDD en bases de datos espaciales, como el algoritmo llamdo CLARANS (Agrupación de aplicaciones grandes basadas en RANdomized Search), que es un método mejorado de *k-medoids*. En relación a otros algoritmos *k-medoids*, CLARANS es más efectivo y rápido. Una evaluación experimental realizada por estos autores indica que CLARANS se ejecuta eficientemente en bases de datos de miles de objetos. Ng y Han (1994) también analizan métodos para determinar el número "natural" k_{nat} de conglomerados en una base de datos. Proponen ejecutar CLARANS una vez por cada k de 2 a n, a continuación para cada uno de los agrupamientos descubiertos se calcula el coeficiente de Silhouette (Kaufman & Rousseeuw, 1990) y, finalmente, se elige como agrupamiento "natural" aquel que tiene el máximo coeficiente de Silhouette. Lamentablemente, el tiempo de ejecución de este enfoque es inaccesible para n grande, porque implica $O(n)$ llamadas de CLARANS (Ester et al., 1996).

CLARANS asume que todos los puntos a agrupar pueden almacenarse en la memoria principal simultáneamente, lo cual no es factible para conjuntos de datos grandes. Además, en conjuntos de datos con muchas observaciones, el tiempo de ejecución de CLARANS es muy elevado. Por lo tanto, Ester et al. (1995) presentan varias técnicas con diferentes enfoques que abordan ambos problemas centrando el proceso de agrupamiento en las partes relevantes de la base de datos. Por un lado, el enfoque es lo bastantemente menor como para almacenarse en la memoria y, además, el tiempo de procesamiento de CLARANS en la parte donde se está enfocando es notablemente inferior que su tiempo de ejecución en el conjunto entero de datos (Ester et al., 1996).

Otro tipo de algoritmos de clustering son los que crean una descomposición jerárquica de D (Figura 3). Ésta está representada por un dendrograma, un árbol que iterativamente divide D en subconjuntos más pequeños hasta que cada subconjunto consta de un solo objeto. De este modo, cada nodo del árbol representa un grupo de D. El dendrograma puede construirse de abajo hacia arriba (enfoque aglomerativo) o de arriba hacia abajo (enfoque divisivo), fusionando o dividiendo los clusters en cada etapa. A diferencia de los algoritmos de partición, los algoritmos jerárquicos no requieren que k sea proporcionado como parámetro de entrada. Sin embargo, se debe definir una condición de terminación que indique cuándo el proceso de fusión o división debe ser terminado (Ester et al., 1996).

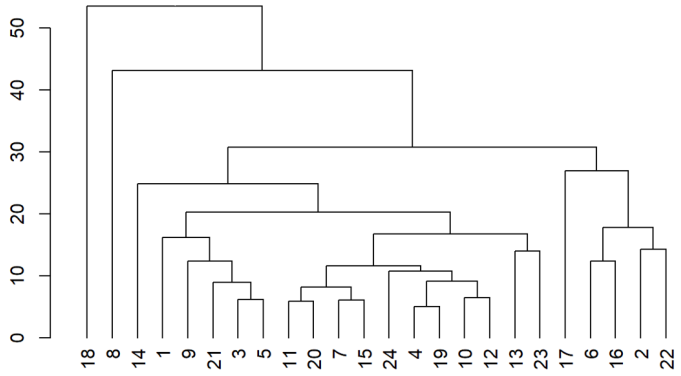

Figura 3: Dendograma de un clustering jerárquico.

DISEÑO DE LÍNEAS DE AUTOBUSES

Las decisiones en la planificación de líneas de buses tienen un impacto significativo en la sociedad, ya que afectan a la movilidad de la población y su calidad de vida, así como al medio ambiente. Un sistema de transporte público eficiente hace posible que personas de todos los niveles socioeconómicos puedan acceder a servicios esenciales tales como la educación, salud y empleo, promoviendo así la inclusión social y la equidad. Además, al reducir la dependencia de vehículos privados, se disminuye la congestión del tráfico y la contaminación, mejorando la calidad del aire y la salud pública.

La creación de líneas de autobuses es un proceso que involucra múltiples factores técnicos, sociales y económicos. No solo requiere un estudio detallado de la demanda de transporte público y las características geográficas de una zona, sino también de las necesidades y comportamientos de los usuarios.

Se comienza con la recopilación de datos sobre movilidad urbana y patrones de tráfico (Arredondo, 2011). Se identifica la necesidad de servicio en un área específica, que puede ser impulsada por el desarrollo urbano, cambios en la demografía o peticiones de la comunidad. A continuación, se realiza un estudio detallado para determinar la viabilidad de la ruta propuesta, considerando factores como la demanda potencial de pasajeros, costes operativos y beneficios ambientales. Utilizando técnicas de análisis y modelado, se identifican patrones significativos en los datos, como áreas de alta demanda o importantes nodos de actividad. Una vez determinadas las ubicaciones óptimas para las paradas de autobús y las conexiones entre diferentes trayectos, se planifica la logística de la ruta, incluyendo horarios y frecuencias. Tras el análisis del coste económico y la comprobación de existencia de crédito, se procederá a implementar la nueva línea.

En este Trabajo de Fin de Grado, se estudiará cómo el algoritmo DBSCAN puede ser aplicado para mejorar la planificación y operación de líneas de autobuses urbanos, estudiando las áreas de alta demanda de transporte y principales nodos de actividad a través de un conjunto de datos.

La ciudad de Aracaju, Brasil, cuenta con un extenso sistema de transporte público, compuesto por numerosas líneas de autobús que conectan diversos barrios y áreas de interés. En este contexto, se ha implementado el algoritmo DBSCAN con el propósito de analizar y optimizar la distribución de las rutas y la densidad de vehículos. El objetivo es identificar patrones y clusters que permitan una mejor planificación del transporte, reduciendo la congestión y mejorando la accesibilidad y la frecuencia del servicio público. A través de esta investigación, se propondrán soluciones basadas en los resultados obtenidos que contribuyan a un sistema de transporte más eficiente y sostenible para la ciudad de Aracaju.

Se ha seleccionado el algoritmo DBSCAN (*Density-Based Spatial Clustering of Applications with Noise)* debido a su capacidad para manejar datos que presentan formas irregulares o que no están claramente separados por límites geométricos definidos. A diferencia de otros algoritmos de agrupación que presuponen una forma específica para cada grupo, DBSCAN identifica los clusters basándose en la densidad de los puntos. Esta característica lo hace idóneo para detectar patrones de movilidad y áreas de alta demanda en entornos urbanos, donde las distribuciones de datos pueden ser desconocidas.

IMPORTANCIA DEL ESTUDIO

Este estudio es fundamental para entender y mejorar la movilidad urbana en la ciudad de Aracaju. Al analizar la distribución de coches y autobuses utilizando el algoritmo DBSCAN, se identificarán patrones de uso y áreas de alta densidad vehicular. Estos hallazgos ayudarán a planificar un sistema de transporte público más eficiente y sostenible. La identificación de clusters específicos de coches y autobuses permitirá a los planificadores urbanos centrar sus esfuerzos en áreas críticas, donde la congestión y la demanda de transporte sean más elevadas.

Además, este estudio proporciona una base sólida para la implementación de mejoras en el transporte público. La fusión de líneas de autobús redundantes, la creación de nuevas rutas y la optimización de la frecuencia y cobertura del servicio pueden contribuir a disminuir la congestión vehicular y promover el uso del transporte público. Al mejorar la accesibilidad y la eficiencia de las líneas de autobús, se puede reducir la dependencia del coche privado, disminuyendo así la contaminación y mejorando la calidad del aire en la ciudad.

Otro aspecto relevante de este estudio es su contribución a la sostenibilidad ambiental. Al abordar los problemas de movilidad y proponer soluciones basadas en los datos, se promueve un uso más racional y eficiente de los recursos urbanos. Esto no solo mejora la experiencia de los usuarios del transporte público, sino que también contribuye a la creación de una ciudad respetuosa con el medio ambiente.

En resumen, la importancia de este estudio radica en su capacidad para ofrecer soluciones prácticas y basadas en datos que mejoren la movilidad urbana, reduzcan la congestión del tráfico y promuevan la sostenibilidad en Aracaju. Estas mejoras no solo beneficiarán a los residentes y visitantes de la ciudad, sino que también contribuirán al desarrollo de un entorno urbano más eficiente y sostenible.

Ahora, una vez detallados los conceptos clave de este trabajo, se presenta su estructura:

Tras una primera sección dedicada a los objetivos del estudio, el primer capítulo tratará sobre el desarrollo estadístico del algoritmo DBSCAN. Además, en este capítulo tambien se incluyen otros algoritmos de clustering basados en densidad, aunque no se abordarán en profundidad.

En el segundo capítulo se describirá la base de datos seleccionada para el estudio y su preprocesamiento.

El siguiente capítulo se centrará en la implementación del algoritmo DBSCAN a los datos y en el desarrollo de una aplicación en RShiny.

En los dos capítulos finales se expondrán los resultados obtenidos tras implementar el algortimo al y las conclusiones obtenidas.

Finalmente, se presentará la bibliografía utilizada para la realizar este trabajo, la cual incluye diversos libros, artículos y páginas web.

Objetivos

- Analizar los patrones de movimiento y zonas de alta incidencia en un conjunto de datos de trayectorias de vehículos utilizando el algoritmo de clustering DBSCAN.

- Ajustar y validar los parámetros del algoritmo DBSCAN para mejorar su eficacia en la detección de patrones de tráfico densos, evaluando su precisión y eficiencia para la planificación urbana de transporte.

- Analizar los clusters formados para descubrir características uniformes en los patrones de movimiento de los vehículos y evaluar cómo las zonas de alta incidencia podrían beneficiarse de la implementación de nuevas rutas de autobuses urbanos.

- Utilizar los resultados del análisis de clustering para proponer y diseñar nuevas líneas de autobuses, mejorando así la eficiencia y cobertura del servicio de transporte público en la región estudiada.

- Valorar la eficacia del algoritmo DBSCAN en el análisis de datos geoespaciales para la planificación urbana de transporte.

- Crear una aplicación web interactiva utilizando R que permita al usuario ejecutar el algoritmo sobre cualquier base de datos y facilite la visualización de los resultados.

Capítulo 1

Desarrollo Estadístico

Como se mencionó en la introducción, la agrupación en clusters se describe típicamente como el proceso de encontrar estructura en los datos agrupando objetos similares, donde los grupos resultantes se llaman clusters. La premisa de muchos algoritmos de agrupación en clusters es que los objetos asignados al mismo cluster deben ser más similares entre sí que a los objetos en otros clusters.

La similitud a menudo se captura a través de alguna noción de distancia, derivada del hecho de que los objetos se asumen como puntos de datos incrustados en un espacio de datos en el que se puede definir una medida de distancia (Hahsler et al., 2019). A continuación, se presentan algunas de las medidas de distancia más comunes utilizadas en algoritmos de clustering:

- Distancia Euclidiana.

  La distancia Euclidiana es la medida más común y se calcula como la longitud del segmento de línea recta entre dos puntos en un espacio Euclidiano. Se utiliza la fórmula:

  $$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

  Esta distancia es adecuada para datos en un espacio donde la noción de distancia directa entre puntos es relevante.

- Distancia de Manhattan.

  La distancia de Manhattan calcula la distancia entre dos puntos sumando las diferencias absolutas de sus coordenadas. La fórmula es:

  $$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

  Es útil en espacios donde el movimiento se restringe a ejes perpendiculares, como en una cuadrícula de calles de una ciudad.

- Distancia de Minkowski.

  La distancia de Minkowski es una generalización de la distancia Euclidiana y la distancia de Manhattan. Se define por la fórmula:

  $$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

  Cuando $p = 2$, es la distancia Euclidiana; cuando $p = 1$, es la distancia de Manhattan.

- Distancia de Chebyshev.

  La distancia de Chebyshev mide la distancia máxima entre las coordenadas correspondientes de dos puntos. Se calcula con:

  $$d(x, y) = \max_i |x_i - y_i|$$

  Es útil en contextos donde se debe minimizar la distancia máxima a lo largo de cualquier dimensión.

- Distancia Coseno.

  La distancia coseno mide la similitud entre dos vectores calculando el coseno del ángulo entre ellos. La fórmula es:

  $$d(x, y) = 1 - \frac{x \cdot y}{||x|| \cdot ||y||}$$

  Esta medida es adecuada para datos de alta dimensión y textos, donde el ángulo entre vectores es más relevante que su magnitud.

- Distancia de Mahalanobis. . La distancia de Mahalanobis tiene en cuenta las correlaciones entre las variables y la dispersión de los datos. Se define como:

  $$d(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)}$$

  donde S es la matriz de covarianzas. Es útil para datos que presentan diferentes escalas y correlaciones entre variables.

- Distancia de Hamming.

  La distancia de Hamming se utiliza principalmente para variables categóricas y mide el número de posiciones en las que dos cadenas de igual longitud son diferentes. La fórmula es:

  $$d(x, y) = \sum_{i=1}^{n} [x_i \neq y_i]$$

  Es útil en la comparación de secuencias binarias o cadenas de texto.

Los métodos de agrupación típicos se centran en resolver el problema del *k-means*, que consiste en el desconocimiento de la distribución de los datos (MacQueen, 1967). Esto se consigue resolver mediante la búsqueda de diferentes agrupaciones para cada valor del parámetro desconocido, y eligiendo aquel para el cual es más probable que hayan surgido los datos observados (Hahsler et al., 2019).

Muchos de estos enfoques asumen clusters con formas convexas, hiperesféricas (Jain et al., 1999). En contraste, los enfoques de agrupación basados en densidad se han vuelto cada vez más populares debido a su capacidad para capturar clusters de formas arbitrarias, incluidas las formas no convexas. Los enfoques basados en densidad postulan que los clusters son simplemente zonas densas contiguas en el conjunto de datos (es decir, regiones de alta densidad de puntos), separadas por espacios de baja densidad (Sander, 2011).

La agrupación basada en densidad puede manejar ruido, donde los puntos que residen en áreas de muy baja densidad no se asignan a un cluster, sino que se tratan como observaciones ruidosas o atípicas. Estas propiedades proporcionan ventajas para muchas aplicaciones. Por ejemplo, los datos geoespaciales pueden estar llenos de puntos de datos ruidosos debido a errores de estimación en los sensores habilitados para GPS (Chen et al., 2014) y pueden tener formas de clusters no convexas causadas por la topología del espacio físico en el que se capturaron los datos.

La agrupación basada en densidad también ha mostrado ventajas para caracterizar datos de alta dimensión (Kailing et al., 2004), donde las particiones son difíciles de descubrir y donde es más probable que

se violen las restricciones de forma física asumidas por los métodos basados en modelos (Hahsler et al., 2019). En la siguiente sección, se profundiza en este tipo de clustering basado en densidad.

1.1. Clustering basado en densidad

Conceptualmente, la idea detrás de la agrupación basada en la densidad es simple: dado un conjunto de puntos de datos, definir una estructura que refleje con precisión la densidad subyacente (Sander, 2011). Así, en este enfoque, los grupos de datos se consideran como regiones de puntos de datos con alta densidad, donde ésta se define como el número de puntos de datos dentro de un radio dado (Li & Ye, 2002).

Una distinción importante entre los métodos de agrupación basados en densidad y el resto de técnicas de análisis de conglomerados, como los modelos de mezcla (gaussianos), es que estos últimos representan un enfoque paramétrico en el que se supone que los datos observados han sido producidos por una mezcla de distribuciones paramétricas (que a menudo se supone que son gaussianas). Aunque son útiles en muchas aplicaciones, las técnicas paramétricas asumen de forma natural que los conglomerados tendrán alguna forma convexa (hiperesférica o hiperelíptica). Otros métodos, como el clustering *k-means* (en el que el parámetro k indica el número de clusters preestablecidos a encontrar), comparten esta idea común al asumir que se pueden encontrar buenos clusters minimizando alguna medida de varianza intracluster, lo que también conduce a formas convexas de los conglomerados (Hahsler et al., 2019).

Por el contrario, los métodos de agrupamiento basados en la densidad no asumen distribuciones paramétricas ni utilizan la varianza, por lo que son capaces de encontrar conglomerados de forma arbitraria, manejar cantidades variables de ruido y no requieren conocimientos previos sobre cómo establecer el número de conglomerados. Estos enfoques son particularmente eficaces en el contexto de datos espaciales, donde las agrupaciones pueden tener formas elongadas o lineales debido a restricciones geográficas. En los métodos basados en densidad no se hacen suposiciones sobre el número de agrupaciones o su distribución, sino que permiten la detección de áreas densas conectadas separadas por áreas dispersas tratadas como ruido, que no se incorporan a los grupos formados (Aggarwal & Reddy, 2014).

A continuación, se presentan algunos de los principales algoritmos de agrupamiento basados en densidad y se analizará cómo cada uno de ellos estima la densidad, define la conectividad y qué estructuras de datos soportan su implementación eficiente.

DBSCAN es un algoritmo de agrupamiento que agrupa objetos basándose en parámetros de entrada como Eps (el radio máximo de un vecindario) y $MinPts$ (el número mínimo de puntos necesarios en la vecindad de un objeto central). Estos ajustes de parámetros suelen estar establecidos empíricamente (Han et al., 2012). En la sección 1.2. se analiza detalladamente este algoritmo.

A diferencia de otros algoritmos de clustering basados en densidad como DBSCAN, **DENCLUE** utiliza una aproximación basada en la función de densidad de los datos, que modela matemáticamente la influencia de un punto en su vecindario. Una función de influencia debe ser simétrica, continua y diferenciable, y ejemplos típicos de funciones de influencia son las funciones de onda cuadrada o las funciones gaussianas (Aggarwal & Reddy, 2014).

La función global de densidad se define como la suma de los núcleos de todos los puntos de datos. Dados N objetos de datos descritos por un conjunto de vectores de características d-dimensionales, la función de densidad global se define como:

$$f^D(x) = \sum_{i=1}^{n} e^{-\frac{d(x,x_i)^2}{2\sigma^2}}$$

DENCLUE se basa en las siguientes definiciones (Yu & Jian, 2005):

- Atractor de Densidad. Un punto $x^* \in \mathbb{R}^d$ se llama un atractor de densidad de una función de densidad f^D, si y solo si x^* es un máximo local de f^D. Un punto $x \in \mathbb{R}^d$ es atraído por densidad a un atractor de densidad x^*, si y solo si un procedimiento de búsqueda de la máxima pendiente iniciado en x converge a x^*.
- Cluster Definido por el Centro. Un cluster definido por el centro para un atractor de densidad $x^* \left(f^D(x^*) \geq \xi\right)$ es un subconjunto $C \subseteq D$ con $x \in C$ siendo atraído por densidad por x^*. Los puntos $x \in D$ se llaman outliers o ruido si son atraídos por densidad por un máximo local x_0^* con $f^D\left(x_0^*\right) < \xi$.

Así, DENCLUE modela la densidad de los datos utilizando funciones de núcleo y forma clusters alrededor de atractores de densidad, puntos de alta densidad local. La densidad en algún punto se estima por la suma de las influencias de todos los puntos de datos. Se dice que un punto es atraído por densidad a un llamado atractor de densidad, si están conectados a través de un camino de puntos de alta densidad (Aggarwal & Reddy, 2014).

El algoritmo depende de dos parámetros principales. El parámetro σ determina el ancho de la función de núcleo para la estimación de densidad mientras que ξ establece el umbral de densidad para distinguir entre puntos que pertenecen a clusters y ruido.

Según Yu y Jian (2005), las principales ventajas que presenta el algoritmo DENCLUE son que cuenta con una base matemática firme, lo que proporciona un fundamento teórico sólido, y presenta excelentes propiedades de agrupamiento incluso en conjuntos de datos con grandes cantidades de ruido. Además, proporciona una representación matemática concisa de grupos con formas arbitrarias en conjuntos de datos de alta dimensionalidad.

Sin embargo, en muchas bases de datos de la vida real, la estructura intrínseca de los conglomerados no puede caracterizarse mediante parámetros de densidad global, y es posible que se necesiten densidades locales muy diferentes para revelar conglomerados en diferentes regiones del espacio de datos (Aggarwal & Reddy, 2014). La mayoría de los algoritmos son sensibles a estos valores: configuraciones ligeramente diferentes puede dar lugar a agrupaciones muy distintas de los datos. Además, los conjuntos de datos del mundo real a menudo tienen distribuciones sesgadas, por lo que un único conjunto de parámetros de densidad global puede no representar bien la estructura intrínseca de los grupos (Han et al., 2012).

La idea básica del algoritmo **OPTICS** para abordar este desafío no es crear una agrupación explícita, sino producir un nuevo ordenamiento de los puntos de la base de datos con respecto a su estructura de agrupación basada en la densidad. Este orden se visualiza gráficamente para respaldar el análisis interactivo de la estructura del cluster. (Aggarwal & Reddy, 2014)

Para un valor $MinPts$ constante, los clusters basados en densidad con respecto a una densidad más alta (es decir, un valor más bajo para Eps) están completamente contenidos en grupos con respecto a una densidad más baja (es decir, un valor más alto para Eps). Sin embargo, los puntos a los que se puede alcanzar en densidad con respecto al valor de Eps más bajo siempre tendrían que procesarse primero para garantizar que los grupos con respecto a los de mayor densidad se terminan primero (Aggarwal & Reddy, 2014).

OPTICS funciona inicialmente como una versión extendida del algoritmo DBSCAN que considera un número infinito de parámetros de distancia Eps_i, todos menores que una distancia generadora Eps. La única diferencia es que no asigna miembros de cluster, sino que almacena el orden en el que se procesan

los puntos (el orden de agrupación) y los dos datos siguientes que serían utilizados por un algoritmo DBSCAN extendido para asignar miembros de cluster (Aggarwal & Reddy, 2014).

Los conglomerados basados en densidad son monótonos con respecto al umbral de vecindad. Es decir, en DBSCAN, para un valor $MinPts$ fijo y dos umbrales de vecindad, $Eps_1 < Eps_2$, el grupo C con respecto a Eps_1 y $MinPts$ deben ser un subconjunto de un grupo C' con respecto a Eps_2 y $MinPts$. Esto significa que si dos objetos están en un grupo basado en densidad, también deben estar en un grupo con un requisito de menor densidad (Han et al., 2012). En consecuencia, el algoritmo DBSCAN podría extenderse para agrupar simultáneamente la base de datos para varios valores de Eps.

El orden en el que se crean los grupos es equivalente a la agrupación basada en densidad obtenida a partir de una amplia gama de configuraciones de parámetros. Por lo tanto, OPTICS no requiere que el usuario proporcione un umbral de densidad específico (Han et al., 2012).

Para construir los diferentes clusters simultáneamente, los objetos se procesan en un orden específico. Este orden selecciona un objeto que sea alcanzable en densidad con respecto al valor de Eps más bajo para que los grupos con mayor densidad (Eps más bajo) se terminen primero (Han et al., 2012).

OPTICS se basa en dos conceptos clave (Hahsler et al., 2019):

- Distancia al núcleo: El valor mínimo de Eps que convierte a un objeto en un objeto central. Se define como

$$\text{core-dist}(p) = \begin{cases} \text{INDEFINIDO} & \text{si } \left|N_{Eps}(p)\right| < MinPts, \\ \textit{MinPts-dist(p)} & \text{de otro modo.} \end{cases}$$

  donde *MinPts-dist(p)* es la distancia de p a su $(MinPts - 1)$-ésimo vecino más cercano, es decir, el radio mínimo que tendría un vecindario de tamaño *MinPts* centrado en p e incluyendo a p.

- Distancia alcanzable: El valor mínimo de radio que hace que un objeto sea alcanzable en densidad desde un objeto central. Se define como:

$$\text{reachability-dist}(p, q) = \begin{cases} \text{INDEFINIDO} & \text{si } \left|N_{Eps}(p)\right| < MinPts, \\ \text{máx}(\text{core-dist}(p), d(p, q)) & \text{de otro modo.} \end{cases}$$

CLIQUE (Agrupación en QUEst) fusiona métodos basados en densidad y basados en cuadrícula. Sin embargo, los resultados de este método son bastante imprecisos (Yu & Jian, 2005).

En conjuntos de datos de alta dimensión, los clusters tienden a residir en subespacios de dimensiones inferiores en lugar de en el espacio de dimensión completa, lo que ha motivado la tarea de agrupación subespacial. El algoritmo CLIQUE, discretiza el espacio de datos a través de una cuadrícula y estima la densidad contando el número de puntos en una celda de la cuadrícula. Se considera que dos celdas de la cuadrícula están conectadas si comparten algún borde o cara. Un grupo subespacial es entonces un conjunto de células denso vecinas en un subespacio arbitrario (Aggarwal & Reddy, 2014).

CLIQUE asume que los nodos conocen la identidad del cluster al que pertenecen. Las formas y tamaños de los grupos no son importantes para CLIQUE, lo que significa que no depende de ninguna propiedad específica del cluster. Por simplicidad, se asume que la membresía del cluster se basa en un rectángulo-cuadrícula uniforme y cada nodo sabe la identidad de la celda a la que pertenece. El cómputo de tal cuadrícula es sencillo incluso cuando es aproximado (Forster & Murphy, 2009).

Además, se asume que los miembros del cluster son de múltiples saltos, están lejos unos de otros y, si bien conocen a sus vecinos más inmediatos, no tienen información sobre los demás nodos en el mismo cluster.

Según Aggarwal y Reddy (2014), el algoritmo CLIQUE descompone la tarea de agrupación subespacial en tres subtareas:

1) Identificación de subespacios que contienen clusters.
2) Identificación de conglomerados.
3) Generación de descripciones mínimas.

Su solución algorítmica se basa en la siguiente propiedad de monotonicidad: si un conjunto de puntos S es un grupo en un espacio k-dimensional, entonces S también es parte de un grupo en cualquier proyección $(k-1)$-dimensional de este espacio. Debido a esta propiedad, las unidades densas (y, por lo tanto, los subespacios con grupos) se pueden descubrir de manera nivelada, comenzando con unidades unidimensionales y extendiendo las unidades densas en una dimensión en cada nivel. La determinación de la densidad de todas las unidades candidatas en un nivel determinado requiere un escaneo de la base de datos. El algoritmo finaliza cuando no se generan más candidatos en el nivel actual.

Si existe una unidad densa en k dimensiones, entonces todas sus proyecciones en cualquiera de los subconjuntos $O(2k)$ en k dimensiones también son densas. Por tanto, el tiempo de ejecución del algoritmo CLIQUE es exponencial en la dimensión más alta de cualquier unidad densa.

En CLIQUE, como en todos los enfoques basados en cuadrícula, la calidad de los resultados depende crucialmente de la elección adecuada del número y ancho de las particiones y celdas de la cuadrícula.

Por último, se analiza el algoritmo de clustering basado en la densidad más popular, llamado DBSCAN y en el que se centrará el estudio de la creación de líneas de buses.

1.2. DBSCAN: Agrupamiento espacial basado en densidad de aplicaciones con ruido

El algoritmo DBSCAN (Ester et al., 1996) es probablemente el algoritmo de agrupación basado en la densidad más utilizado en la comunidad científica actual. La idea central detrás de DBSCAN es la noción de que los puntos se asignan a un mismo cluster si son alcanzables en densidad entre sí.

Se trata un algoritmo de clasificación no supervisada, puesto que no requiere etiquetas de clase para los datos de entrada. Trabaja directamente con datos no etiquetados, descubriendo estructuras y patrones basados en la densidad de los puntos en el espacio, sin necesidad de especificar el número de clusters a priori.

Dado que el algoritmo depende de dos parámetros: el radio y el número mínimo de puntos, es posible obtener diferentes resultados de agrupamiento utilizando distintos valores para estos parámetros, lo que permite comparar y evaluar los diferentes resultados obtenidos y determinar cuál es el más adecuado. (Ye, 2013)

El clustering comienza con un conjunto de datos D que contiene un conjunto de puntos $p \in D$. Los algoritmos basados en la densidad necesitan obtener una estimación de la densidad sobre el espacio de datos. DBSCAN estima la densidad alrededor de de cada punto utilizando el concepto de "vecindad".

Al observar los conjuntos de puntos mostrados en la Figura 1.1, se pueden identificar claramente y de forma unívoca los conjuntos de puntos y los puntos de ruido que están incluidos en ningún grupo.

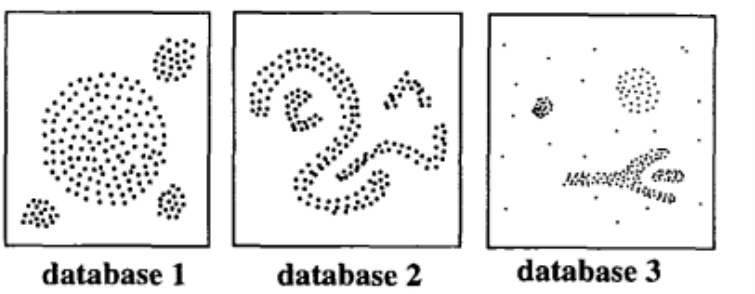

Figura 1.1: Bases de datos de muestra.

El principal motivo por el cual es posible identificar los grupos es que, dentro de cada grupo, la concentración de puntos es significativamente mayor en comparación con el exterior. Asimismo, la densidad de puntos en las zonas de ruido es inferior que la densidad dentro de los distintos grupos.

A continuación, se intentará formalizar la idea de "clusters" y "ruido" en un conjunto de datos D de puntos de un espacio k-dimensional S. Nótese que tanto esta noción de grupos como el algoritmo DBSCAN se aplican tanto al espacio euclidiano 2D o 3D, como a algún espacio de características de alta dimensión. La idea fundamental es que para que un punto pertenezca a un grupo, su vecindad dentro de un radio específico debe incluir al menos un número mínimo de puntos. En otras palabras, la densidad en esa vecindad debe ser superior a un cierto umbral. La forma de una vecindad está determinada por la elección de una función de distancia para dos puntos p y q, representada como $dist\,(p, q)$. Por ejemplo, empleando la distancia de Manhattan en un espacio bidimensional, la vecindad toma una forma rectangular. Este enfoque es válido para cualquier función de distancia, permitiendo seleccionar la más adecuada para cada aplicación específica. Con el propósito de una visualización adecuada, todos los ejemplos que se definirán a continuación se considerarán en un espacio 2D utilizando la distancia euclidiana.

Definición 1: Vecindad Eps de un punto. La vecindad Eps de un punto p, denotada por $N_{Eps}(p)$, se define como $N_{Eps}(p) = \{q \in D \mid dist(p, q) \leq Eps\}$.

Obsérvese que, junto con $p \in D$, esta definición implica que el punto p siempre forma parte de su propia vecindad, es decir, $p \in N_{Eps}(p)$ siempre se cumple. Siguiendo esta definición, el tamaño del vecindario $\left|N_{Eps}(p)\right|$ puede verse como una simple estimación no normalizada de la densidad del núcleo alrededor de p utilizando un núcleo uniforme con un ancho de banda de Eps.

Un enfoque naive podría requerir que cada punto en un grupo tenga por lo menos un número mínimo ($MinPts$) de puntos dentro de un radio Eps alrededor de ese punto. No obstante, este enfoque es insuficiente, porque existen dos tipos de puntos en un grupo: los que están dentro de él (puntos centrales) y los puntos en la periferia del grupo (puntos de borde).

- Un punto p es central (*core point*) si $N_{Eps}(p)$ tiene una densidad alta, es decir, $\left|N_{Eps}\,(p)\right| \geq MinPts$.
- Un punto p es de borde (*border point*) si no es un punto central, pero se encuentra en la vecindad de un punto central $q \in D$, es decir, $p \in N_{Eps}(q)$.
- En caso contrario, un punto p es de ruido.

Normalmente, la vecindad Eps de un punto de borde contiene considerablemente un menor número de puntos que la vecindad Eps de un punto central. Por lo tanto, sería necesario fijar un número mínimo de puntos en un valor considerablemente reducido para abarcar todos los puntos que pertenecen al mismo

grupo. Sin embargo, este valor no sería representativo del grupo, especialmente si hay ruido presente. Por lo tanto, se establece que para que un punto p pertenezca a un grupo C, si existe un punto q en C de modo que p esté dentro del vecindario Eps de q y que el vecindario $N_{Eps}(q)$ contenga como mínimo $MinPts$ puntos. A continuación, se detalla esto.

Definición 2: Densidad alcanzable directa (*Directly density-reachable*) Un punto p es directamente denso alcanzable desde un punto q con respecto a Eps y $MinPts$ si

1) $p \in N_{Eps}(q)$.
2) $\left|N_{Eps}(q)\right| \geq MinPts$. (Condición de punto central para q).

Entre pares de puntos centrales, la densidad alcanzable directa es simétrica. Sin embargo, en general, no es simétrica cuando involucra a un punto central y un punto de borde. La Figura 1.2 muestra el caso asimétrico.

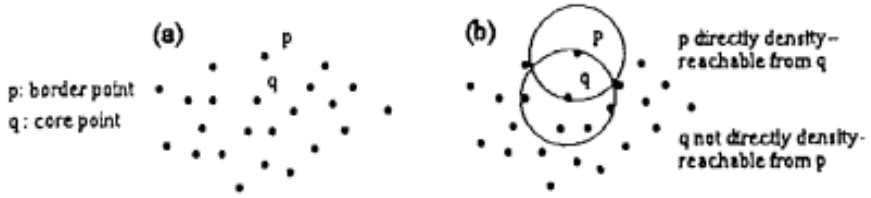

Figura 1.2: Puntos centrales y puntos de borde.

Definición 3: Densidad alcanzable (*density-reachable*). Un punto p es denso alcanzable desde un punto q con respecto a Eps y $MinPts$ si existe un conjunto de puntos $p_1, \ldots, p_n$ con $p_1 = q, p_n = p$ tales que cada punto p_{i+1} es directamente denso alcanzable desde p_i.

La densidad alcanzable es una extensión de la densidad alcanzable directa. Esta relación no es simétrica, pero sí transitiva. La Figura 1.3 muestra las relaciones entre algunos puntos de muestra y, específicamente, un caso asimétrico. Aunque la densidad alcanzable no es simétrica en general, es evidente que sí lo es para los puntos centrales.

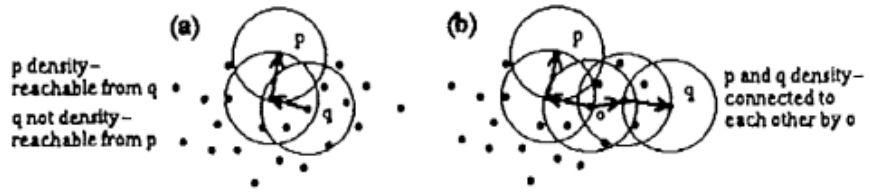

Figura 1.3: Densidad alcanzable y densidad conectada.

Es posible que dos puntos de borde que pertenezcan al mismo grupo C no sean denso alcanzables entre sí porque la condición de punto central podría no cumplirse para ambos. Sin embargo, debe haber un punto central en C desde el cual ambos puntos de borde de C sean denso alcanzables. Por lo tanto, se introduce el concepto de densidad conectada, que abarca esta relación entre los puntos de borde.

Definición 4: Densidad conectada (*density-connected*). Un punto p es denso conectado a un punto q con respecto a Eps y $MinPts$ si existe un punto o tal que tanto p como q son denso alcanzables desde o con respecto a Eps y $MinPts$.

La densidad conectada es una relación simétrica. En el caso de puntos que son denso alcanzables, la

relación de densidad conectada también es reflexiva (ver Figura 1.3).

Ahora, es posible definir la noción de un cluster basado en la densidad. De manera intuitiva, se considerará que un cluster es una agrupación de puntos denso conectados que es maximal en términos de la densidad alcanzable. El ruido se definirá en relación con un conjunto dado de clusters. Se trata de la colección de puntos en D que no están incluidos en ninguno de los clusters definidos.

Definición 5: Cluster. Sea D una base de datos (puntos). Un cluster C con respecto a Eps y $MinPts$ es un subconjunto no vacío de D que satisface las siguientes condiciones:

1) $\forall p, q$: si $p \in C$ y q es denso alcanzable desde p con respecto a Eps y $MinPts$, entonces q está en C. (Maximalidad).
2) $\forall p, q \in C$: p está denso conectado a q con respecto a Eps y $MinPts$. (Conectividad).

Definición 6: Ruido. Sean $C_1, \ldots, C_k$ los clusters de la base de datos D con respecto a los parámetros Eps_i y $MinPts_i$, $i = 1, \ldots, k$. Entonces el ruido se define como la colección de puntos en la base de datos D que no se asignan a ningún cluster C_i, es decir, $ruido = \{p \in D \mid p \notin C_i \ \forall i\}$.

Es importante señalar que un cluster C, definido en función de Eps y $MinPts$, contiene por lo menos $MinPts$ puntos por el siguiente motivo: como C incluye al menos un punto p, este punto debe estar denso conectado consigo mismo a través de algún punto o (que podría ser el mismo p). En consecuencia, al menos o debe cumplir con la premisa de punto central y, como resultado, el vecindario Eps de o incluye por lo menos $MinPts$ puntos.

Los siguientes lemas son fundamentales para asegurar que el algoritmo funciona correctamente. Básicamente, afirman lo siguiente: dados los parámetros Eps y $MinPts$, es posible identificar un cluster a través de un proceso de dos fases. Primero, se selecciona un punto al azar que cumpla con la premisa de ser un punto central (es decir, tiene al menos $MinPts$ puntos dentro de su vecindario de radio Eps). Este punto actúa como una semilla. A continuación, se identifican todos los puntos que son denso alcanzables desde esta semilla, formando de este modo el cluster completo que incluye a la semilla.

Lema 1: Sea p un punto en D y $\left|N_{Eps}(p)\right| \geq MinPts$. Entonces, el conjunto de los puntos de D denso alcanzables desde p con respecto a Eps y $MinPts$ es un cluster con respecto a Eps y $MinPts$.

No es evidente que un cluster C con respecto a Eps y $MinPts$ esté determinado de manera única por cualquiera de sus puntos centrales. No obstante, cada punto en C es denso alcanzable desde cada uno de los puntos centrales de C, por lo que un cluster C contiene concretamente los puntos que son denso alcanzables desde un punto central dentro de C.

Lema 2: Sea C un cluster con respecto a Eps y $MinPts$ y sea p cualquier punto arbitrario en C con $\left|N_{Eps}(p)\right| \geq MinPts$. En consecuencia, C se define como el conjunto de los puntos denso alcanzables desde p con respecto a Eps y $MinPts$.

En la siguiente sección, se describe cómo funciona el algoritmo DBSCAN, diseñado para identificar clusters y ruido en una base de datos espacial según las definiciones 5 y 6. Idealmente, se necesitaría conocer los parámetros adecuados Eps y $MinPts$ para cada cluster, así como al menos un punto perteneciente a cada cluster. Con esta información, sería posible obtener cada uno de los puntos que son denso alcanzables desde el punto dado utilizando únicamente los parámetros apropiados. Sin embargo, no es sencillo obtener esta información de antemano para cada cluster del conjunto de datos.

Aun así, existe una heurística simple y efectiva (descrita en la sección 1.2.2) para determinar los pará-

metros Eps y $MinPts$ del cluster "más fino", que se corresponde con el cluster menos denso de todos los que conforman conjunto de datos. De este modo, DBSCAN utiliza valores comunes para estos dos parámetros, aplicando el mismo Eps y $MinPts$ a todos los clusters. Los parámetros de densidad del cluster "más fino" son buenos aspirantes para estos valores generales, determinando la densidad mínima que no se considera ruido.

1.2.1. El algoritmo

Para encontrar un cluster, DBSCAN comienza con un punto arbitrario p e intenta encontrar otro punto central en su vecindario respecto a Eps y $MinPts$. Si se encuentra, la búsqueda se amplía para incluir también todos los puntos de su vecindario. Si se trata de un punto de borde, entonces el cluster está completo (ver Lema 2), y DBSCAN busca entre los puntos restantes del conjunto de datos si existe otro punto central para empezar un nuevo cluster. Después de procesar todos los puntos, los puntos que no se asignaron a ningún cluster se establecen como puntos de ruido.

Dado que se utilizan valores generales para Eps y $MinPts$, DBSCAN puede fusionar dos clusters según la Definición 5 en un solo cluster, si dos clusters de diferente densidad están "cerca" (a una distancia menor o igual que Eps el uno del otro). Sea la distancia entre dos conjuntos de puntos S_1 y S_2 definida como $dist\,(S_1, S_2) = min\,\{dist(p, q) \mid p \in S_1, q \in S_2\}$. Entonces, dos conjuntos de puntos que tienen al menos la densidad del cluster más fino, se separarán solo si la distancia entre los dos conjuntos es mayor que Eps.

En consecuencia, puede ser necesario una llamada recursiva de DBSCAN para los clusters detectados, con un valor más alto para $MinPts$. Este reajuste en $MinPts$ aumenta el umbral de densidad necesario para que un punto sea considerado un núcleo, haciendo que únicamente las áreas del conjunto de datos más densamente pobladas cumplan con el nuevo criterio para formar clusters. En consecuencia, las áreas menos densas no alcanzarán este umbral y no formarán un nuevo cluster.

Esto permite una mejor separación de clusters con diferentes densidades, y se realiza de manera eficiente y precisa, manteniendo la simplicidad del algoritmo original. Además, la agrupación recursiva de los puntos que forman un cluster solo es necesaria en determinadas circunstancias que pueden detectarse de forma simple.

1.2.2. Determinando los parámetros de Eps y MinPts

En esta sección, se va a presentar un método sencillo pero eficiente para establecer los parámetros Eps y $MinPts$ del "cluster más fino" en el conjunto de datos. Este método se fundamenta en la siguiente idea: sea d la distancia desde un punto p hasta su k-ésimo vecino más cercano, entonces la vecindad con radio d alrededor de p contendrá exactamente $k + 1$ puntos para casi todos los puntos p. La vecindad con radio d alrededor de p contiene más de $k + 1$ puntos solo si la mayoría de ellos están exactamente a la misma distancia d de p, lo cual es bastante improbable. Además, cambiar el valor de k para un punto dentro de un cluster no provoca grandes variaciones en d. Esto solo sucedería si los k-ésimos vecinos más cercanos de p para $k = 1, 2, 3, \ldots$ estuvieran alineados en una línea recta, lo cual generalmente no ocurre para un punto dentro de un cluster.

Para un valor específico de k, se determina una función $k - dist$ que asigna a cada punto de la base de datos D su distancia a su k-ésimo vecino más cercano. Al ordenar de forma descendente todos los puntos que conforman la base de datos según sus valores de *k-dist*, el gráfico resultante proporciona información sobre la distribución de densidad en la base de datos. Este gráfico se denomina el gráfico de *k-dist* ordenado. Al elegirse un punto p cualquiera, se fija el parámetro Eps en *k-dist(p)* y $MinPts$ en k, y se designarán como puntos centrales todos aquellos con un valor de *k-dist* igual o menor que *k-dist(p)*. Si fuese posible identificar un punto umbral con el máximo valor de $k - dist$ dentro del "cluster más fino"

de *D*, se obtendrían los valores de parámetro necesarios. Este punto umbral es el que corresponde con el primer "valle" del gráfico de *k-dist* ordenado (ver Figura 1.4). Todos los puntos con un valor de *k-dist* mayor (a la izquierda del umbral) se consideran ruido, mientras que los puntos con un valor de *k-dist* menor o igual (a la derecha del umbral) se asignan a algún cluster.

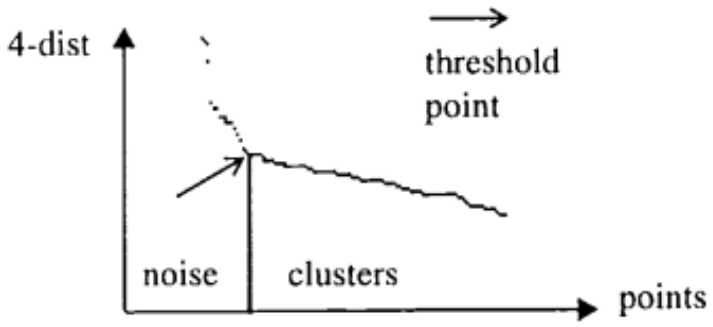

Figura 1.4: Gráfico 4-dist ordenado para la base de datos 3 de muestra.

Generalmente, detectar automáticamente el primer "valle" en el gráfico es muy complicado, pero resulta bastante fácil para un usuario identificarlo visualmente. Por esta razón, Ester et al. (1996) sugieren un método interactivo para establecerlo. Aunque DBSCAN requiere dos parámetros, Eps y $MinPts$, Ester et al. (1996) encontraron en sus experimentos que los gráficos de *k-dist* para $k \geq 4$ no varían significativamente del gráfico de *4-dist* y requieren más recursos computacionales. Por lo tanto, fijaron el parámetro $MinPts$ en 4 para todas las bases de datos bidimensionales, eliminando así la necesidad de ajustarlo. El método iterativo propuesto para determinar el parámetro Eps de DBSCAN es el siguiente:

- Se calcula y presenta el gráfico de *4-dist* para el conjunto de datos.
- Si el usuario puede hacer una estimación del porcentaje de ruido, este se ingresa y el sistema sugiere un punto umbral basado en esta estimación.
- El usuario tiene la opción de aceptar el umbral sugerido o seleccionar otro. El valor de *4-dist* del punto elegido se empleará como valor del parámetro Eps en el algoritmo.

1.2.3. Evaluación del rendimiento

En esta sección, se presenta una evaluación del rendimiento de DBSCAN. Ester et al. (1996) compararon este algoritmo con el rendimiento de CLARANS, que fue el primer y único algoritmo de agrupamiento diseñado para el propósito de KDD en aquel momento.

Para comparar DBSCAN con CLARANS en términos de efectividad (precisión), se utilizarán las tres bases de datos de muestras sintéticas mostradas en la Figura 1.1. Dado que DBSCAN y CLARANS son algoritmos de agrupamiento de diferentes tipos, no tienen una medida cuantitativa común de la precisión de la clasificación. Por lo tanto, se evalúa la precisión de ambos algoritmos mediante inspección visual. En la base de datos 1 de muestra, hay cuatro clusters en forma de bola de tamaños significativamente diferentes. La base de datos 2 de muestra contiene cuatro clusters de forma no convexa. En la base de datos 3 de muestra, hay cuatro clusters de forma y tamaño diferentes con ruido adicional. Para darle a CLARANS alguna ventaja, se estableció el parámetro *k* en 4 para estas bases de datos de muestra. Los agrupamientos descubiertos por CLARANS se muestran en la Figura 1.5.

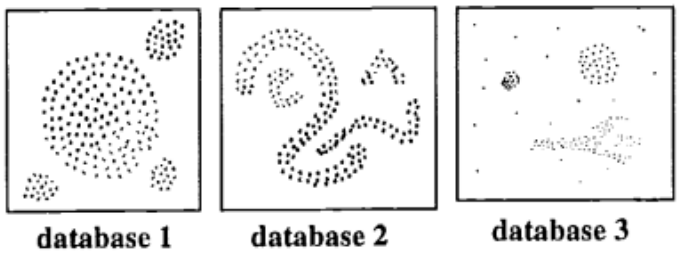

Figura 1.5: Clusters descubiertos por CLARANS.

Para DBSCAN, se estableció el porcentaje de ruido en 0 % para las bases de datos 1 y 2 de muestra, y en 10 % para la base de datos de muestra 3. Los agrupamientos clusters descubiertos por DBSCAN se muestran en la Figura 1.6. DBSCAN descubre todos los clusters (según la Definición 5) y detecta los puntos de ruido (según la Definición 6) de todas las bases de datos de muestra. Sin embargo, CLARANS divide los clusters si son relativamente grandes o si están cerca de algún otro cluster. Además, CLARANS no tiene una noción explícita de ruido. De este modo, todos los puntos se asignan a su *medoid* más cercano.

Figura 1.6: Clusters descubiertos por DBSCAN.

1.2.4. Complejidad del algoritmo

La complejidad computacional del algoritmo es $O(n\log(n))$ (Aggarwal & Reddy, 2014). Utilizando métodos de acceso espacial, se ha encontrado que DBSCAN es eficiente incluso para bases de datos espaciales muy grandes.

Las implementaciones estándar de DBSCAN se basan en índices espaciales, que proporcionan información sobre la eficiencia en la búsqueda de la región vecindario Eps de un punto dado. En el peor de los casos, DBSCAN realiza la búsqueda de la región alrededor de cada punto de la base de datos y esto conduce a una complejidad de tiempo de ejecución de $O(n\log(n))$ para DBSCAN, donde n denota el número de puntos de la base de datos. Desafortunadamente, los índices espaciales degeneran para datos de alta dimensión, es decir, el rendimiento en la búsqueda de la región degenera de $O(\log(n))$ a $O(n)$, y la complejidad de tiempo de ejecución de DBSCAN se convierte en $O\left(n^2\right)$ para tales datos. Por otro lado, si la estructura de los datos permitiese que la búsqueda de la región vecindario Eps estuviera basada en una cuadrícula, la complejidad de la búsqueda disminuiría a $O(1)$, y la complejidad de tiempo de ejecución de DBSCAN disminuiría a $O(n)$, (nótese que una complejidad de tiempo de ejecución de $O(n\log(n))$ se considera manejable para conjuntos de datos grandes).

Una versión incremental de DBSCAN puede mejorar aún más su eficiencia en bases de datos dinámicas con inserciones y eliminaciones. Ester et al. (1998) muestran que un clustering basado en densidad se puede actualizar incrementalmente sin tener que volver a ejecutar el algoritmo DBSCAN en la base de datos actualizada. Examinan qué parte de un clustering existente se ve afectada por una actualización de la base de datos y presentan algoritmos para actualizaciones incrementales de un clustering después de inserciones y eliminaciones. Debido a la naturaleza local de los clusters basados en densidad, la porción

de objetos de la base de datos afectados tiende a ser pequeña, lo que hace que el algoritmo incremental sea muy eficiente.

La idea básica de los clusters basados en densidad se puede generalizar de varias maneras (Sander et al., 1998). En primer lugar, cualquier noción de vecindario se puede emplear en lugar de un vecindario Eps basado en distancia siempre que la definición del vecindario se base en una proposición $NProp(p,q)$ que sea simétrica y reflexiva. El vecindario N de p se define entonces como el conjunto de todos los puntos q que satisfacen $NProp(p,q)$. En segundo lugar, en lugar de simplemente contar los elementos en un vecindario, también podemos usar una proposición más general, $MinWeight(N)$, para determinar si el vecindario N es denso, esto es, si $MinWeight$ es monótono en N, es decir, si $MinWeight$ se cumple para todos los superconjuntos de conjuntos que satisfacen N. Finalmente, no solo objetos de tipo punto sino también objetos extendidos espacialmente como polígonos pueden ser agrupados. Al agrupar polígonos, existen proposiciones más naturales que la del vecindario Eps y la de la restricción de cardinalidad $MinPts$.

1.3. Validación del clustering

Dividir el conjunto de datos en subconjuntos de entrenamiento, validación y prueba es una práctica esencial en el desarrollo y evaluación de modelos de aprendizaje automático. Esta división ayuda a garantizar que el modelo sea robusto, preciso y extrapolable a nuevos datos.

Inicialmente, el modelo se entrena con un subconjunto de datos de entrenamiento para ajustar sus parámetros. En la práctica, el conjunto de datos de entrenamiento suele estar compuesto por pares de un vector de entrada y el correspondiente vector de salida, donde la respuesta esperada se conoce como objetivo o etiqueta. El modelo se ejecuta con el conjunto de datos de entrenamiento y genera un resultado que se compara con el objetivo para cada vector de entrada. Basándose en el resultado de esta comparación y el algoritmo de aprendizaje específico, se ajustan los parámetros del modelo.

En la fase de validación, se ajustan los hiperparámetros del modelo para prevenir el sobreajuste. Durante esta fase el modelo se aplica al conjunto de validación y permite identificar si el modelo está sobreajustando los datos de entrenamiento, es decir, si está memorizando los datos de entrenamiento en lugar de generalizar bien a nuevos datos.

Finalmente, el conjunto de prueba se utiliza para evaluar la precisión final del modelo. Una vez que el modelo ha sido entrenado y los hiperparámetros han sido ajustados utilizando los conjuntos de entrenamiento y validación, se evalúa el rendimiento final del modelo utilizando el conjunto de prueba. La métrica obtenida, como la precisión o el error cuadrático medio, en el conjunto de prueba es una estimación de cómo se comportará el modelo en la práctica real.

Para determinar la efectividad de los resultados de un algoritmo de clustering, se puede utilizar una métrica basada en el porcentaje de puntos correctamente clasificados, excluyendo los puntos clasificados como ruido. Este enfoque es especialmente útil cuando se dispone de etiquetas verdaderas para los datos, permitiendo así una comparación directa entre las etiquetas predichas y las reales.

Considérese un conjunto de datos compuesto por n puntos, donde cada punto i tiene una etiqueta verdadera y_i y una etiqueta predicha $\hat{y}_i$. Los puntos clasificados como ruido se excluyen de la evaluación para evitar distorsionar los resultados. Sea C el conjunto de todas las categorías, es decir, los clusters verdaderos. Para cada categoría c en C, se define N_c como el número total de puntos en la categoría c y $N_{c,\text{correcto}}$ como el número de puntos correctamente clasificados en dicha categoría.

La precisión para cada categoría c se calcula como la fracción de puntos correctamente clasificados respecto al total de puntos en esa categoría, expresada por la fórmula:

$$\text{Precisión}(c) = \frac{N_{c,\text{correcto}}}{N_c}$$

Para obtener el número total de puntos correctamente clasificados en todo el conjunto de datos, se suman los puntos correctamente clasificados en todas las categorías:

$$N_{\text{total,correcto}} = \sum_{c \in C} N_{c,\text{correcto}}$$

El número total de puntos considerados en la evaluación, excluyendo el ruido, se calcula sumando los puntos en todas las categorías:

$$N_{\text{total}} = \sum_{c \in C} N_c$$

Finalmente, la precisión global del clustering se determina como la fracción de puntos correctamente clasificados respecto al total de puntos, excluyendo el ruido. Se expresa mediante la fórmula:

$$\text{Precisión global} = \frac{N_{\text{total,correcto}}}{N_{\text{total}}}$$

Esta métrica proporciona una evaluación clara de la calidad del clustering y refleja la precisión de la clasificación de los puntos excluyendo aquellos clasificados como ruido. Un valor alto de precisión global indica una alta calidad en los clusters formados por el modelo, demostrando que las predicciones coinciden estrechamente con las etiquetas verdaderas de los datos.

Capítulo 2

Datos Reales

La base de datos de *GPS Trajectories* ha sido extraído del *UCI Machine Learning Repository* (Cruz et al., 2016). En él se recopila información de trayectorias de vehículos obtenidas a través de una aplicación de Android llamada *Go!Track*. Incluye datos de 163 trayectorias distintas con 15 características cada una, tales como la velocidad media durante el trayecto, distancia recorrida y evaluaciones sobre las condiciones del tráfico, ocupación de autobuses y el clima.

El conjunto de datos consta de dos archivos principales:

1) Go_track_tracks.csv: proporciona atributos generales sobre cada trayectoria.
    - *Id*: clave única para identificar cada trayectoria.
    - *Id_android*: representa el dispositivo utilizado para capturar la instancia.
    - *Speed*: representa la velocidad promedio (Km/h).
    - *Time*: representa la duración del trayecto (min).
    - *Distance*: representa la distancia total recorrida (Km).
    - *Rating*: es un parámetro de evaluación. Califica la percepción de los voluntarios sobre el tráfico durante el viaje. (3: bueno, 2: normal, 1: malo).
    - *Rating_bus*: es otro parámetro de evaluación. (3: el autobús está lleno, 2: el autobús no está lleno, 1: viajan pocas personas en el autobús).
    - *Rating_weather*: es otro parámetro de evaluación de las condiciones meteorológicas durante el trayecto. (2: soleado, 1: lluvioso).
    - *Car_or_bus*: indica el medio de transporte en el que se ha realizado el trayecto. (1: coche, 2: autobús).
    - *Linha*: información sobre el autobús que realiza el recorrido.

2) Go_track_trackspoints.csv: registra las coordenadas GPS específicas para cada punto a lo largo de las trayectorias.
    - *Id*: clave única para identificar cada punto.
    - *Latitude*: latitud desde donde se toma el punto.
    - *Longitude*: longitud desde donde se toma el punto.
    - *Track_id*: identifica la trayectoria a la que pertenece el punto.
    - *Time*: fecha y hora en que se recopiló el punto (GMT-3).

2.1. Preprocesamiento de los datos

Para llevar a cabo el análisis, se han fusionado los dos conjuntos de datos en un solo archivo tomando como referencia la variable *Id* del archivo Go_track_tracks.csv y *Track_Id* del archivo Go_track_trackspoints.csv. Esta columna sirve como identificador único para cada trayectoria. De este modo, se ha unido la información general de cada recorrido con las coordenadas exactas de los puntos que lo componen. Como resultado, se ha generado una nueva base de datos que contiene 18.107 puntos geográficos pertenecientes a 163 trayectorias distintas y 17 atributos definidos sobre cada fila.

A continuación, se ha llevado a cabo una recategorización y selección de las variables incluidas para el análisis con el fin de mejorar las conclusiones obtenidas. En primer lugar, se han eliminado las siguientes variables:

- *Id_Android*: se eliminó esta variable ya que el dispositivo utilizado no aporta información relevante para el análisis de las trayectorias.
- *Rating_bus* y *Rating_weather*: se han eliminado estas dos variables debido a la alta incidencia de valores no evaluados, lo que indica una baja tasa de respuesta.
- *Time*: se ha eliminado esta variable dado que los aspectos temporales específicos de cada punto no son de interés para el objetivo del estudio.

Para simplificar el conjunto de datos, se conservaron únicamente dos identificadores principales:

- *Id_punto*: este identificador ahora representa cada punto geoespacial único dentro de la base de datos.
- *Track_Id_trayectoria*: este identificador distingue cada trayectoria individual.

Así, el nuevo conjunto de datos contiene las siguientes variables:

- *Id_punto*: identificador que representa cada punto geoespacial único.
- *Latitude*: latitud desde donde se toma el punto.
- *Longitude*: longitud desde donde se toma el punto.
- T*rack_Id_trayectoria*: identificador que distingue cada trayectoria individual.
- *Speed*: representa la velocidad promedio (Km/h).
- *Time*: representa la duración del trayecto (min).
- *Distance*: representa la distancia total recorrida (Km)
- *Car_or_bus*: indica el medio de transporte en el que se ha realizado el trayecto. (1: coche, 2: autobús).
- *Linha*: información sobre el autobús que realiza el recorrido.
- *Rating*: califica la percepción de los voluntarios sobre el tráfico durante el viaje. (3: bueno, 2: normal, 1: malo).

El primer paso para crear los conjuntos de datos a los que se les aplicará el algoritmo ha sido convertir todos los archivos de texto a formato .xlsx. Posteriormente, se ha verificado que no hubiera datos faltantes ni trayectorias duplicadas.

Para clasificar las diferentes trayectorias en los subconjuntos de entrenamiento, testeo y predicción, se han separado los registros de coches y buses basándose en la variable *car_or_bus* (1 = coche, 2 = bus).

La variable *linha* puede tomar los valores “carro”, o la línea de bus correspondiente a las observaciones que en la variable *car_or_bus* tienen valor 2, es decir, autobuses. Cuenta con 11.307 valores faltantes,

que no se han eliminado del conjunto de datos debido a que se van a predecir posteriormente. Esto se justifica porque la técnica utilizada es no supervisada, lo que permite trabajar con datos incompletos y hacer predicciones sin necesidad de etiquetar previamente los datos faltantes.

Para los coches (*car_or_bus* = 1), se extrajeron aquellos etiquetados como “carro” en *linha* para el conjunto de entrenamiento. El resto se dividió en dos mitades a partes iguales para los conjuntos de testeo y predicción.

Para los buses (*car_or_bus* = 2), los registros con líneas de bus explícitas formaron el conjunto de entrenamiento, mientras que aquellos sin información (*linha* = NA) se dividieron también a partes iguales en testeo y predicción.

Capítulo 3

Software

El paquete *fdm2id* (Fouille de Données en Master 2 Informatique Décisionnelle) en R (Blansché, 2023) contiene funciones para simplificar el uso de métodos de Minería de Datos (clasificación, regresión, clustering, etc.), para estudiantes y principiantes en programación con R. Se utilizan varios paquetes de R y se construyen *wrappers* alrededor de las funciones principales, para estandarizar el uso de métodos de minería de datos (entrada/salida). Esto conlleva una cierta pérdida de flexibilidad, pero también un aumento de la simplicidad.

La función DBSCAN(d, minpts, epsilonDist, ...), incluida en este paquete, necesita como argumentos: d, el conjunto de datos (puede ser una matriz o data.frame); minpts, el número mínimo de puntos alcanzables; y epsilonDist, la distancia de alcance.

La librería *fpc* (Flexible Procedures for Clustering) (Hennig, 2024) ofrece herramientas para realizar y evaluar análisis de clustering y algunas funciones adicionales para el análisis de datos.

La librería *flexclust* (Flexible Cluster Algorithms) (Leisch et al., 2024) proporciona una infraestructura para realizar clustering de una manera flexible, incluyendo diversas técnicas y herramientas para la evaluación de clusters. Predice las clasificaciones de nuevos datos basados en un modelo de clustering previamente ajustado y visualiza los resultados.

Previamente a realizar el algoritmo DBSCAN, se ha dividido el conjunto de datos en entrenamiento, testeo y predicción. Los índices *x* e *y* indican cuántas observaciones conformarán cada conjunto.

```
x <- X
y <- Y
entrenamiento <- datos[1:x, ]
testeo <- datos[(x + 1):(x+y) ,]
prediccion <- datos[(x + y + 1):nrow(datos), ]
```

A continuación, se generan los conjunto de datos para cada fase excluyendo la variable respuesta que, en caso de existir, corresponde con la última columna del conjunto de datos. Los conjuntos resultantes son los que se emplearán para la aplicación del algoritmo.

```
variables_entrenamiento<-entrenamiento[,-ncol(datos)]
variables_testeo<-testeo[,-ncol(datos)]
variables_prediccion<-prediccion[,-ncol(datos)]
```

3.1. Código

3.1.1. Código DBSCAN en R

Antes de implementar DBSCAN, es fundamental seleccionar adecuadamente el parámetro Eps. Para ello, primero se calcula la matriz de distancias de las variables de entrenamiento. La búsqueda de Eps se realiza mediante un gráfico de distancia k-vecinos (*k-dist*) ordenado, utilizando en R la función kNN-distplot contenida en el paquete *dbscan*. Se establece $k = 4$, dado que se ha determinado anteriormente que para conjuntos de datos bidimensionales, $MinPts = 4$.

```
library(dbscan)
distancias<-as.matrix(dist(variables_entrenamiento))
kNNdistplot(distancias, k = 4)
```

La elección del parámetro Eps se identifica en el primer "valle" de la gráfica kNN. Una vez definido este parámetro, se almacena en la variable $epsilonDist$ y se procede a la ejecución del algoritmo.

```
epsilonDist = Eps
```

Se emplea la función DBSCAN del paquete *fdm2id*. Este comando devolverá el modelo ajustado, identificando los clusters para los datos de entrenamiento.

```
library(fdm2id)
library(fpc)
library(flexclust)
dbscan<-DBSCAN(variables_entrenamiento, minpts = 4, epsilonDist = epsilonDist)
```

Los clusters creados por el modelo se almacenan en la categoría *cluster* dentro del objeto *dbscan*. El data.frame *tabla_cluster* almacena la información de cuántas observaciones conforman cada cada uno de ellos, por lo que la dimensión de éste mismo se corresponde con el número de clusters definidos por el modelo más uno, el correspondiente a los puntos ruidosos.

```
tabla_cluster<-table(dbscan$cluster)
as.data.frame(tabla_cluster)
names(tabla_cluster)<-c(çluster","frecuencia")
dim(tabla_cluster)[1]
```

A efectos de hacer más visual la salida de los resultados del clustering, se ha construido un gráfico Biplot de los cluster.

```
svd<-svd(variables_entrenamiento)
A<-svd$u[,1:2]%*%diag(svd$d[1:2])
plot(A,col=factor(dbscan$cluster),xlab=,ylab=))
```

El modelo se aplica a la base de datos excluyendo la variable respuesta, ya que se trata de una técnica no supervisada. En el caso de que el usuario disponga de una base de datos con variable respuesta, al finalizar la fase de entrenamiento, se comprueba si los cluster definidos por el modelo coinciden con las

categorías reales de la variable respuesta.

Esta comprobación se realiza con el siguiente fragmento de código, donde el índice *i* corresponde al número de cluster creado por el modelo y *ncol(datos)* a la última columna del dataset, es decir, la variable respuesta. El bucle *for* devueve la matriz de confusión de cada cluster, permitiendo analizar la composición de cada uno de ellos.

```
for(i in 1:(dim(tabla_cluster)[1]-1)){
    tab<-data.frame(table(entrenamiento[dbscan$cluster==i, ncol(datos)]))
    if(nrow(tab)==2)
      {cat(paste0(capture.output(print(tab,row.names=c(paste(c̈luster ",i),))),
          collapse="\n"),"\n")}
    else
      {cat(paste0(capture.output(print(tab, row.names=paste(çluster ",i)))
          collapse="\n"),"\n")}
    }
```

Continuando con el supuesto de que la base de datos del usuario contiene una variable respuesta, esta debe ser binaria y estar codificada númericamente en 1 y 2 para aplicar las fases de testeo y predicción. En el vector *clusters_1* se almacenan los clusters en los que predomina la etiqueta 1 de la variable respuesta a la vista de las matrices de confusión.

```
clusters_1<-c(a, b, c)
```

En el siguiente paso, se aplica el modelo ajustado para predecir las respuestas de las observaciones en el conjunto de datos de testeo. Utilizando la función *predict*, se compara para cada observación si lo que predice el modelo creado en la fase entrenamiento coincide con la clase real de la variable respuesta.

```
predicho<-predict(dbscan, variables_testeo)
```

Finalmente, mediante la bondad de ajuste se evalúa la precisión y la capacidad predictiva del modelo para clasificar correctamente los puntos de testeo en función de los clusters obtenidos durante el entrenamiento. Se calcula dividiendo el porcentaje de puntos correctamente clasificados en cada categoría de la variable de respuesta entre el total de puntos de testeo, excluyendo los puntos clasificados como ruido. Dado que el vector *clusters_1* contiene los cluster clasificados como 1, este cálculo de la bondad del ajuste se realiza con el siguiente código:

```
bondad_ajuste<-(sum(testeo[predicho%in%clusters_1,ncol(datos)]==1)
                +sum(testeo[!(predicho%in%clusters_1),ncol(datos)]==2))/
                (dim(testeo)[1]-sum(predicho==0))
```

Por último, en la fase de predicción se toma un conjunto de nuevas observaciones, y se predice a qué cluster pertenece cada una utilizando el modelo previamente entrenado.

```
predicho_prediccion<-predict(dbscan, variables_prediccion)
prediccion_final<-cbind(variables_prediccion, predicho_prediccion)
```

En caso de que el usuario disponga de una base de datos en la que desconoce la variable respuesta, el código también puede aplicarse para predecir para cada observación cuál sería la clase de esa variable respuesta según el modelo creado en la fase de entrenamiento y que ya se ha demostrado que tiene buena bondad de ajuste en la fase de testeo.

3.1.2. Interfaz de la aplicación web

Con Shiny (Chang, 2024), es posible crear una interfaz de usuario interactiva (Jia et al., 2022). Permite a los usuarios ajustar parámetros y ver resultados en tiempo real sin necesidad de conocimientos avanzados de programación web.

El desarrollo de una aplicación RShiny incluye tanto el diseño de una interfaz gráfica de usuario como el procesamiento de las solicitudes de usuario recibidas de dicho lado. En esta segunda estructura, se procesan las entradas del usuario, se realizan los cálculos necesarios y se generan las salidas que se mostrarán en la interfaz gráfica.

Las aplicaciones en Shiny están contenidas en un único script llamado App.R. El script app.R se encuentra en un directorio y la aplicación se puede ejecutar con runApp. Una App.R tiene tres componentes: un objeto de interfaz de usuario (ui.R), una función servidor (server.R) y una llamada a la *shinyAppfunction*. Esta última funcion toma dos argumentos principales: *ui* y *server*, y los combina para lanzar la aplicación (posit Team, 2024).

La apariencia y el diseño de una aplicación RShiny se configuran en ui.R. Además, se pueden definir varios widgets de entrada, que se utilizan para recibir entradas del usuario, incluida la carga de archivos, pegado de texto, configuración de parámetros, etc.

Las entradas del usuario recopiladas por ui.R se transmiten al lado del servidor y se procesan por server.R. Los resultados de los cálculos de server.R se muestran luego como figuras, tablas, etc., en las posiciones especificadas por ui.R en la interfaz de usuario. La Figura 3.1 muestra cómo funciona una aplicación en RShiny.

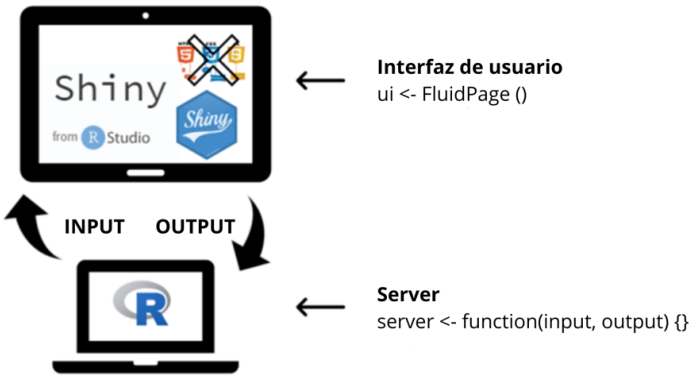

Figura 3.1: Funcionamiento de una aplicación RShiny.

El primer paso en el desarrollo de una aplicación RShiny es diseñar la interfaz de usuario con ui.R (Figura 3.2). La función `fluidPage()` (Figura 3.2 se utiliza para crear un diseño de página única que se ajusta automáticamente a las dimensiones de la ventana del navegador del usuario, mientras que `fluidRow()` (Figura 3.3) permite crear filas dentro de esa página donde se pueden organizar elementos en columnas. La interfaz de la aplicación se puede diseñar colocando elementos de la interfaz de usuario dentro del diseño de la página. Por lo general, una sola página se dividiría en varios paneles colocando las funciones `Panel()` dentro de las funciones `Page()`.

Los estilos CSS son un lenguaje de diseño utilizado para definir y mejorar la presentación visual de documentos web escritos en HTML o XML. CSS se encarga de cómo se verá y se presentará el contenido de una página, separando la estructura del documento de su apariencia.

En CSS, se utilizan selectores para identificar los elementos HTML a los que se aplicarán los estilos. Los selectores pueden ser etiquetas HTML, clases o identificadores, y permiten aplicar propiedades específicas a esos elementos.

Los selectores de clase precedidos por un punto (Figura 3.2), permiten aplicar estilos específicos a los elementos con la clase correspondiente. `.title` se utiliza para dar formato y animación al título, mientras que `.custom-img` posiciona y dimensiona la imagen en la interfaz.

Algunos ejemplos de funciones para incluir estilos CSS personalizados en la app son:

- `tags$head()`: Esta función se utiliza para incluir elementos dentro de la sección <head>del documento HTML.
- `tags$style()`: Esta función permite incluir estilos CSS directamente en el documento HTML.
- `HTML()`: Esta función se utiliza para incluir código HTML como una cadena de texto sin escaparlo. Permite insertar HTML o CSS crudo.

```
ui <- fluidPage(
  tags$head(
    tags$style(HTML("
      @keyframes fadeIn {
        0% { opacity: 0; }
        100% { opacity: 1; }
      }
      .title {
        animation: fadeIn 2s;
        color: white;
        background-color: grey;
        padding: 10px;
        text-align: center;
      }
      body {
        background-color: grey;
        color: white;
      }
      .custom-img {
        position: absolute;
        top: 140px;
        left: 70%;
        transform: translateX(-50%);
        width: 30%;
      }
    "))
  ),
```

Figura 3.2: Estilos CCS.

El objetivo del código de la Figura 3.2 es personalizar la apariencia de la aplicación mediante estilos CSS específicos. Utilizando el selector de clase `.title` se incluye una animación de desvanecimiento para el título, y con `.custom-img` se posiciona y dimensiona una imagen de dentro de la interfaz gráfica de la aplicación. Se establece un esquema de colores claro y oscuro (texto blanco sobre fondo gris) para toda la interfaz, incluyendo el título.

Un panel es una interfaz de usuario gráfica que proporciona enfoques interactivos para rastrear, analizar y visualizar métricas y puntos de datos clave relevantes para un objetivo particular, que se han convertido en la norma para interactuar con los datos.

Las funciones `titlePanel()` y `mainPanel()` (Figura 3.3), se utilizan para combinar un conjunto de elementos de la interfaz de usuario en un panel unificado. El panel de título se utiliza para definir la región del título de la aplicación, mientras que el panel principal es el área principal que se utiliza para

colocar las salidas renderizadas por server.R en forma de imágenes, tablas, textos, etc.

En cualquier panel, podemos agregar elementos de la interfaz de usuario colocando las funciones correspondientes dentro de las funciones `Panel()` en ui.R. El paquete *shiny* proporciona muchos widgets prediseñados como funciones R. Un widget es un elemento de la interfaz de usuario con el que los usuarios pueden interactuar, proporcionando una forma para que los usuarios envíen entradas a la aplicación RShiny.

Por ejemplo, la función `fileInput()`(Figura 3.3) se utiliza para crear un asistente de control de carga de archivos y la función `actionButton()` (Figura 3.3) para crear un botón de acción. Para la función R de cada widget, se requieren tres argumentos obligatorios para definir el identificador, la etiqueta y el valor inicial del widget. El identificador del widget sirve para acceder al widget en el lado del servidor, mientras que la etiqueta se usa para mostrar un texto al lado del widget en la interfaz de usuario.

```
fluidRow(column(9,
                titlePanel(strong(span(class = "title", "BIENVENIDO A LA APLICACIÓN DE CLUSTERING DBSCAN"))),
                mainPanel(
                  strong("La variable respuesta de la base de datos seleccionada debe ser binaria,
                         codificada numéricamente como 1 y 2, y ubicada en la última columna."),
                  br(),
                  br(),
                  fileInput("datos", "Cargue su base de datos en formato .xlsx", accept = ".xlsx"),
                  numericInput("x", "Indique el número de filas de entrenamiento:", value = 0),
                  numericInput("y", "Indique el número de filas de testeo:", value = 0),
                  actionButton("submit", "Enviar"),
                  br(),
                  br()
                )
)
),
```

Figura 3.3: Interfaz de usuario.

La función `numericInput()` (Figura 3.3) se utiliza para crear un campo de entrada que permite al usuario introducir valores numéricos, `textInput()` (Figura 3.3) sirve para crear un campo de entrada de texto donde el usuario puede escribir texto libremente. Para ambas opciones es necesario definir un valor inicial.

`plotOutput()` (Figura 3.3) se emplea para reservar un espacio en la interfaz de usuario donde se mostrarán gráficos generados por el servidor, permitiendo definir un identificador único para asociar el gráfico correspondiente. Por último, `verbatimTextOutput()` (Figura 3.3) crea un área de salida para mostrar texto generado por el servidor en su formato original.

Los elementos de la interfaz de usuario también se pueden agregar usando las funciones de etiquetas HTML del paquete *shiny*. Por ejemplo, la función `strong()` se utiliza para enfatizar un texto en negrita y `br()` para insertar un salto de línea. La función `span()` permite aplicar estilos personalizados a un segmento específico de texto, haciendo más visual el contenido dentro de la página web. En la Figura 3.3, `span()` se utiliza que la animación `fadeIn` solo se aplique al título.

El server.R crea un objeto similar a una lista, llamado output, que contiene todo el código necesario para actualizar los objetos R en su aplicación. Cada objeto definido en ui.R (como gráficos, tablas o textos) debe tener su propia entrada en la lista (posit Team, 2024).

Las entradas del usuario recopiladas por los elementos de la ui.R se almacenan en la variable de entrada en el entorno R, a la que se puede acceder en server.R con los identificadores de los objetos definidos en ui.R (Figuras 3.2 y 3.3).

Por ejemplo, `input$id01` se utiliza para acceder al valor más reciente de un elemento de la interfaz de usuario con el identificador *id01*. Con base en las entradas del usuario y otras variables definidas en

server.R, se pueden realizar varias operaciones, incluida la lectura de archivos cargados por el usuario, manipulación de datos, creación de gráficos, construcción de modelos estadísticos, etc.

El resultado de estas operaciones se puede mostrar como tablas, gráficos, textos, imágenes, etc., en la interfaz de usuario mediante el uso de funciones de renderizado en server.R, incluidas `renderTable(), renderPlot(), renderText() y renderImage()`. Las posiciones de los elementos de la interfaz de usuario presentadas por las funciones de representación están definidas por las funciones `Output()` en ui.R.

Por ejemplo, la posición de una tabla representada en la interfaz de usuario está determinada por la posición de la función `tableOutput()` en ui.R. El resultado de una función de representación debe asignarse a un elemento de la variable de salida en server.R, cuyo nombre luego se utiliza como identificador de la función `Output()` correspondiente. De esta manera, los elementos de la interfaz de usuario presentados por una función de representación en server.R están vinculados a una función `Output()` en ui.R, que crea marcadores de posición para los elementos de la interfaz de usuario. Cada función `Output()` en ui.R debe tener una función de representación correspondiente en server.R.

En la Tabla 3.1 se muestra la correspondencia entre las funciones de renderizado utilizadas en el archivo server.R y las funciones de interfaz de usuario utilizadas en el archivo ui.R

Tabla 3.1: Tabla de funciones RShiny.

Función Render	Función ui	Objeto
renderDataTable()	dataTableOutput()	Tabla de datos
renderImage()	imageOutput()	Imagen
renderPlot()	plotOutput()	Gráfico
renderPrint()	verbatimTextOutput()	Texto
renderTable()	tableOutput()	Tabla de datos
renderText()	textOutput()	Texto
renderUI()	uiOutput()	Texto

Dentro de `shinyServer()`, se utiliza la función `output$<nombre>` para definir las salidas que se mostrarán en la interfaz de usuario. Por ejemplo, `output$plot <- renderPlot()` se usa para crear un gráfico, donde el código dentro de `renderPlot()` especifica cómo se genera el gráfico basándose en los inputs del usuario. De manera similar, `renderTable(), renderText()` y otras funciones de renderizado permiten generar diferentes tipos de salidas.

El archivo server.R también incluye los cálculos y transformaciones de datos necesarios para generar las salidas. Estos cálculos pueden involucrar la manipulación de datos, la aplicación de modelos estadísticos, o cualquier otra operación necesaria para preparar los datos antes de presentarlos al usuario. En la Figura 3.4 se presenta un esquema detallado de la aplicación DBSCAN, mostrando cómo se integran los componentes de ui.R y server.R para crear la interfaz de la aplicación.

Los paneles de texto y el título indicados en ui.R con la función `mainPanel()` están resaltados en color granate. Los elementos de entrada (input), como el campo para cargar la base de datos y los campos numéricos para indicar el número de filas de entrenamiento y de testeo, se muestran en azul. La base de datos cargada se guarda como `datos`. El valor del campo numérico correspondiente a las filas de entrenamiento se almacenará como `x` en server.R, mientras que las filas de testeo se guardarán como `y`. Todos estos elementos de entrada están ubicados en una columna de color azul marino, siguiendo la estructura definida en el archivo ui.R.

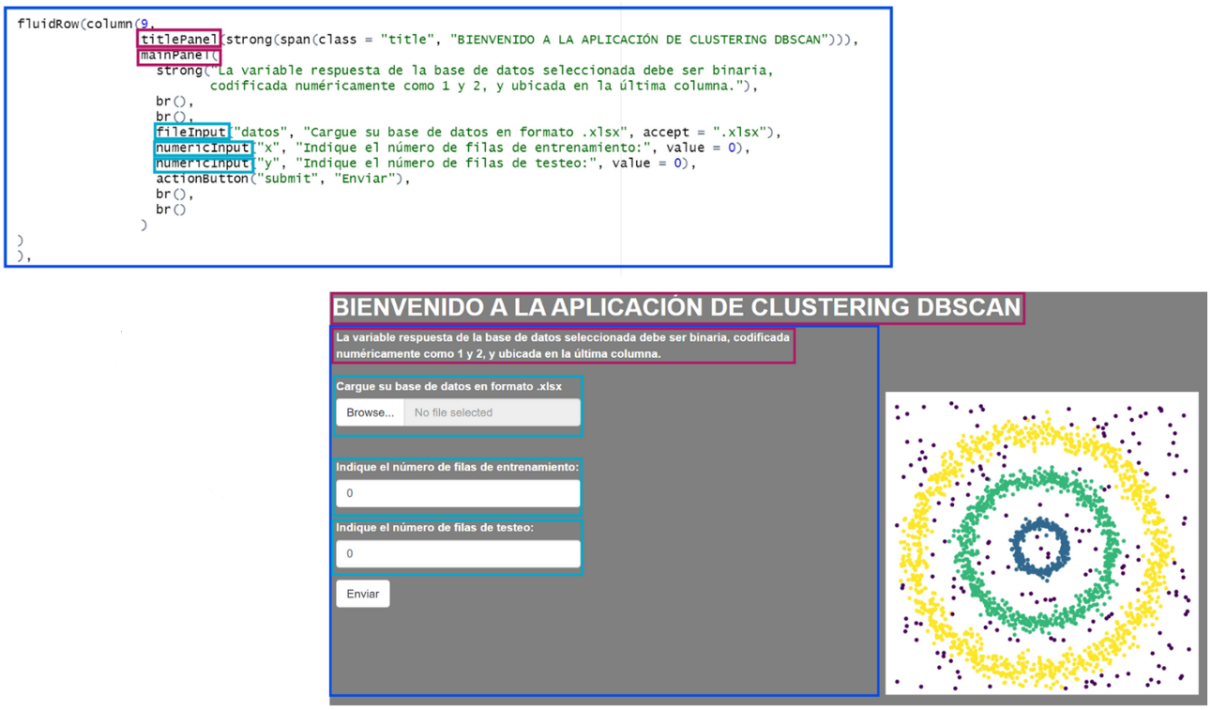

Figura 3.4: Esquema aplicación DBSCAN.

En la Figura 3.5, los elementos de entrada están resaltados en color azul, los elementos de salida en gris y las funciones relacionadas se muestran en el mismo color.

En la parte superior, ui.R, define la interfaz de usuario. Con la función `mainPanel()` se muestran las instrucciones sobre cómo interpretar el gráfico que se mostrará a continuación. `plotOutput("knn")` reserva un espacio en la interfaz para el gráfico KNN-Dist y, por último, el campo de entrada numérica `numericInput("epsilonDist")` permite al usuario ingresar un valor cuya etiqueta será `epsilonDist` en server.R.

En server.R, la función `renderPlot()` se encarga de generar el gráfico KNN-Dist, que será mostrado en el espacio reservado por `plotOutput` en ui.R. Estas dos funciones están diseñadas para trabajar juntas (Tabla 3.1) y que el gráfico generado en el servidor se visualice correctamente.

Por último, la imagen de la interfaz de la aplicación ejemplifica cómo estas dos funciones trabajan juntas. El título y las instrucciones se presentan como se indica en ui.R con la función `mainPanel`. El gráfico KNN-Dist se visualiza en el espacio designado, y el campo de entrada numérica permite al usuario introducir el valor de Epsilon.

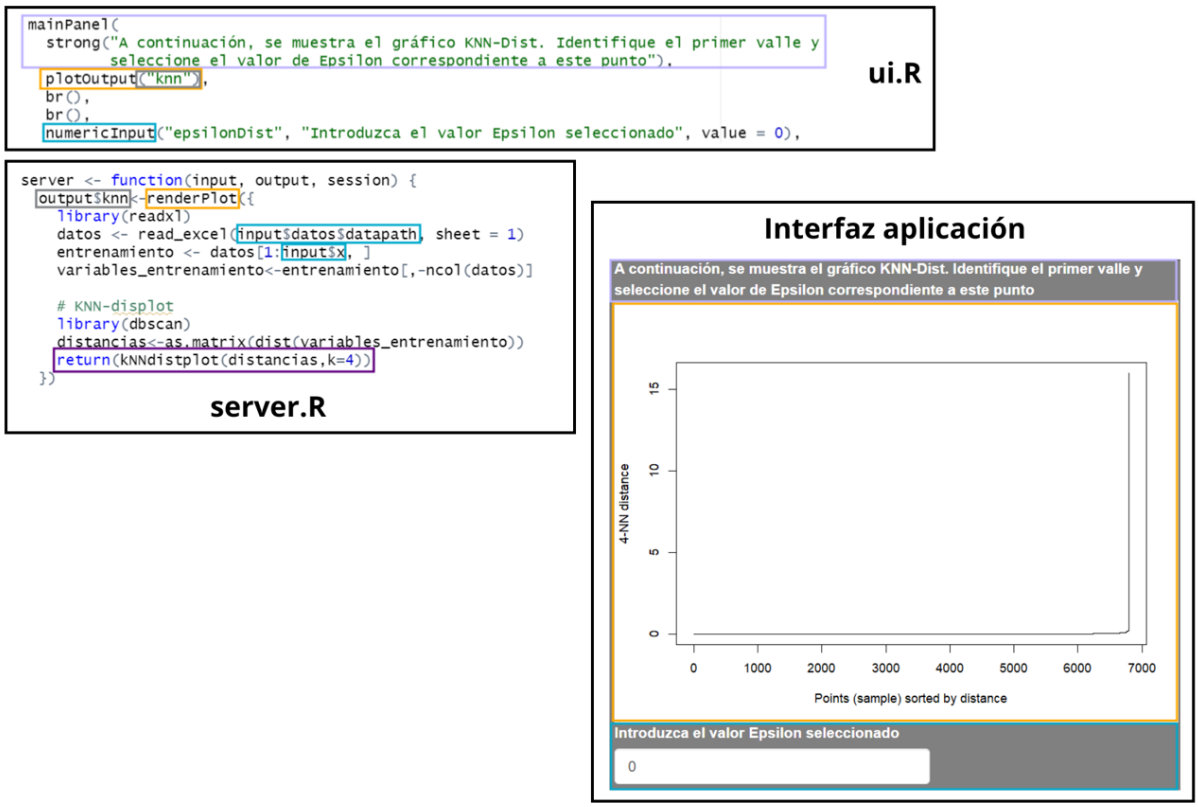

Figura 3.5: Esquema aplicación DBSCAN.

La Figura 3.6 muestra otro ejemplo de funciones que se complementan en la aplicación DBSCAN. En la parte superior, se define la interfaz de usuario. Con la función `mainPanel()` se informa de lo que se está mostrando en pantalla, mientras que `verbatinTextOutput("confusion")` reserva un espacio en la interfaz para la matriz de confusión de cada cluster.

En server.R, la función `renderPrint()` se encarga de devolver la matriz de confusión, que será mostrada en el espacio reservado por `verbatinTextOutput("confusion")` en ui.R. Estas dos funciones permiten que la matriz de confusión generada en el servidor como texto con un bucle *for*, se visualice correctamente en la interfaz del usuario.

Por último, la imagen de la interfaz de la aplicación muestra el resultado de la implementación de estas dos funciones. La matriz de confusión se visualiza en el espacio designado, permitiendo al usuario interpretar los resultados del clustering de manera efectiva. En la siguiente fase, él mismo deberá introducir información basada en los resultados mostrados en esta matriz de confusión.

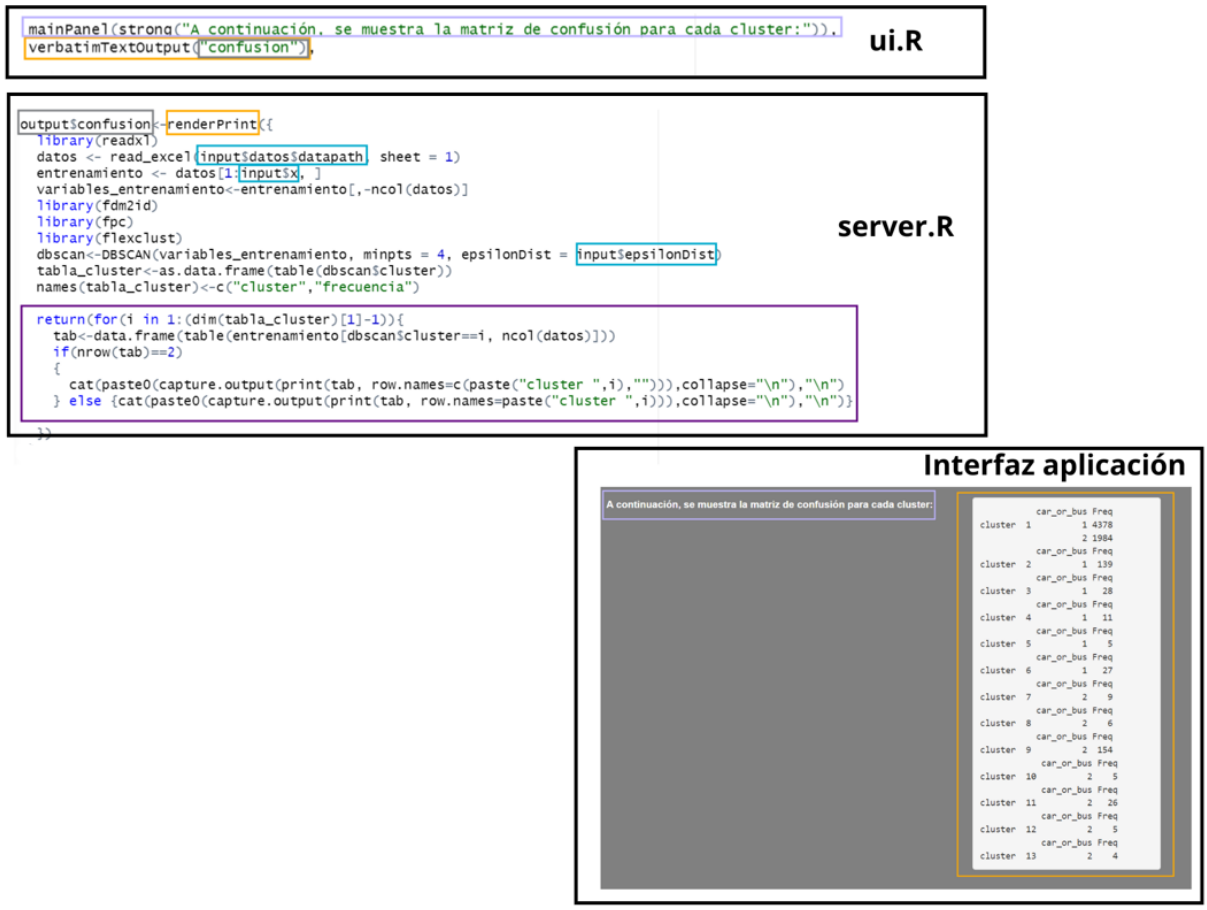

Figura 3.6: Esquema aplicación DBSCAN.

La Figura 3.7 muestra cómo se complementan las funciones `renderPrint()` y `verbatinTextOutput()` empleando la información ingresada por el usuario a través de `inputNumeric()`.

En ui.R, haciendo uso de la función `mainPanel()` se dan instrucciones al usuario para que introduzca los números de los clusters en los que predomina la categoría 1 de la variable respuesta. La función `textInput()` permite al usuario ingresar estos números, y `verbatimTextOutput()` reserva un espacio para mostrar el valor de la bondad de ajuste del modelo.

En server.R, la función `renderPrint()` calcula la bondad de ajuste del modelo y muestra el resultado en el espacio reservado por `verbatinTextOutput()` en ui.R. Los números de los clusters en los que predomina la categoría 1 de la variable respuesta, ingresados por el usuario y almacenados en la variable `input$clusters_1`, participan directamente en este cálculo. Con estos inputs, se realiza el cálculo de la bondad de ajuste, que luego se mostrará en la interfaz del usuario.

En la parte inferior de la Figura 3.7, la imagen de la interfaz de la aplicación ilustra cómo se implementan estas funciones. El valor de la bondad de ajuste se visualiza en el espacio designado, permitiendo al usuario evaluar la precisión del modelo.

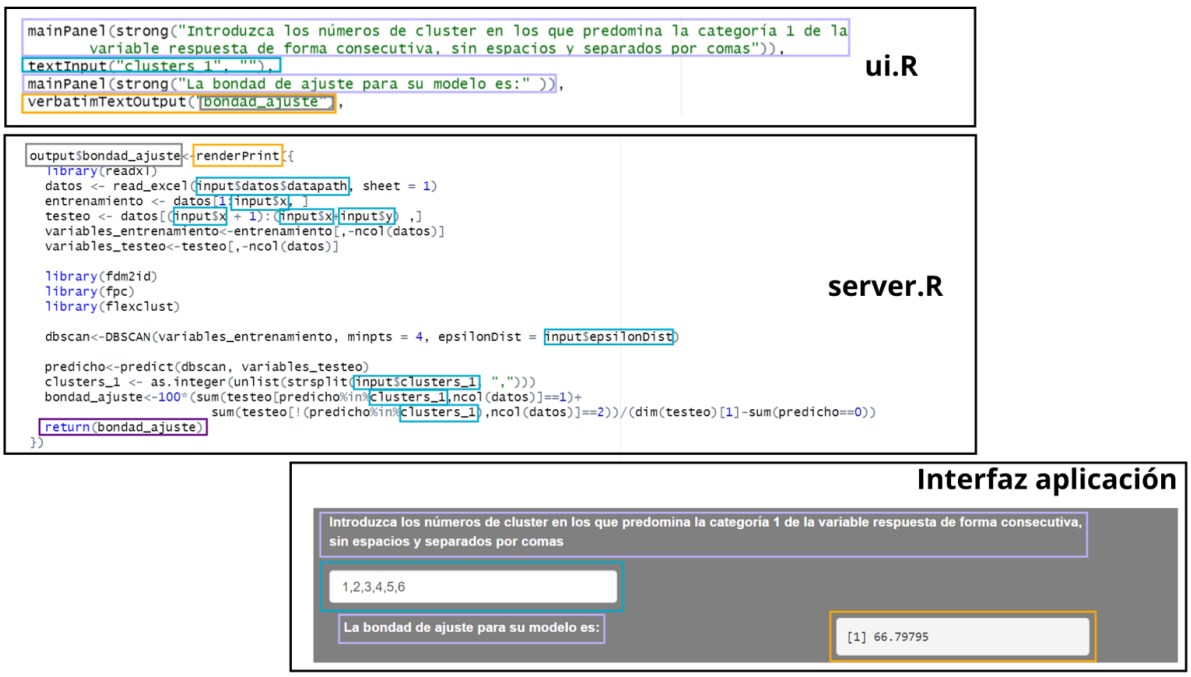

Figura 3.7: Esquema aplicación DBSCAN.

Por último, la función `shinyApp()` en una aplicación Shiny de R se encarga de combinar la interfaz de usuario y la lógica del servidor en una única aplicación y lanzarla. Integra y despliega la aplicación, permitiendo que sea accesible y funcional a través de un navegador web.

Capítulo 4

Resultados

Se ha implementado el algoritmo DBSCAN al conjunto de datos con parámetros establecidos Eps = 0,002 y $MinPoints$ = 4. El algoritmo ha dividido el conjunto de datos en trece clusters distintos y uno adicional que representa el ruido. La bondad de ajuste del modelo ha sido del 66.79 %.

Una vez identificados los grupos definidos por el modelo, se ha analizado su composición para determinar si los puntos que conforman cada uno de ellos pertenece mayoritariamente a coches o autobuses, teniendo en cuenta la variable respuesta.

Posteriormente, se ha aplicado el modelo ajustado para predecir las respuestas de las observaciones en el conjunto de datos de testeo. Para cada punto de testeo, se ha calculado a qué cluster de los de entrenamiento pertenece, basándose en la proximidad a los clusters previamente identificados.

Finalmente, se ha calculado la bondad de ajuste del modelo para evaluar la precisión y la capacidad predictiva del modelo para clasificar correctamente los puntos de testeo en función de los clusters obtenidos durante el entrenamiento.

Los resultados obtenidos (Figura 4.4) incluyen seis cluster compuestos mayoritariamente por coches (Tabla 4.1) y siete por autobuses (Tabla 4.2), cada uno asociado a líneas específicas de transporte público. A continuación, se describen detalladamente los resultados y se proponen posibles soluciones o mejoras basadas en los grupos identificados.

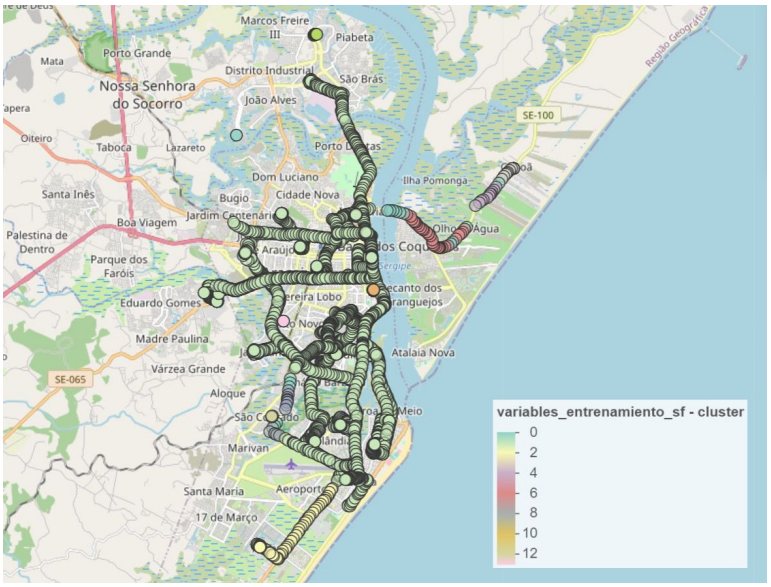

Figura 4.1: Distribución de los clusters definidos por el modelo.

Cluster	Coches	Autobuses
1	4378	1984
2	139	0
3	28	0
4	11	0
5	5	0
6	27	0

Tabla 4.1: Clusters de coches.

Cluster	Coches	Autobuses	Línea
7	0	9	080
8	0	6	080
9	0	154	001 + 020
10	0	5	061
11	0	26	707
12	0	5	001
13	0	4	721

Tabla 4.2: Clusters de autobuses.

El primer cluster (resaltado en color verde en la Figura 4.1) destaca por su alta densidad, con 4.378 coches y 1.984 autobuses. La presencia de una gran cantidad de vehículos privados y de transporte público a su vez, sugiere una zona con alta demanda de movilidad. Una posible solución sería la creación de nuevas líneas de autobús para descomprimir el alto flujo vehicular, y promover así el uso del transporte público.

Las líneas presentes en este cluster son: 001 - A Franco Bugio, 007 - Fernando Collor Atalaia, 008 - Santa Tereza B Industria, 020 - Piabeta Dia, 031 - Eduardo Gomes Des, Mayna, 034 - Term Rod L Batista, 040 - Marcos Freire II Dia, 051 - Atalaia Centro, 060 - Padre Pedro Campus, 080 - Bugio Atalaia, 100 1 - Circular Shoppings, 409 - Riomar Dia, 702 - Augusto Franco Beira Mar y 715 - Tijuquinha Des Maynard.

La presencia de tantas líneas de autobús en un solo cluster sugiere que existen rutas redundantes que podrían ser reestructuradas para mejorar la eficiencia del transporte. Por este motivo, una de las recomendaciones principales es la fusión de líneas que compartan rutas similares o tengan destinos cercanos. Por ejemplo, líneas que conectan áreas de alta movilidad como Franco Bugio (Figura 4.2) y Piabeta Dia (Figura 4.3) podrían ser combinadas.

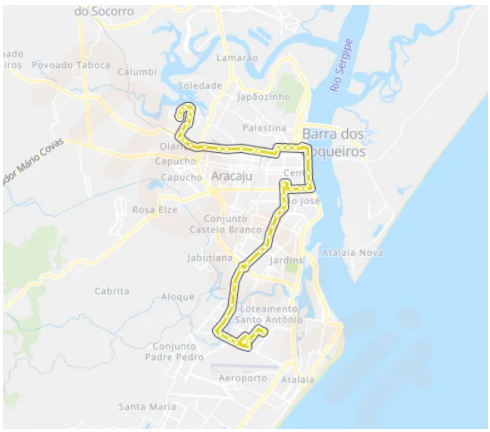

Figura 4.2: Línea 001 - Franco Bugio.

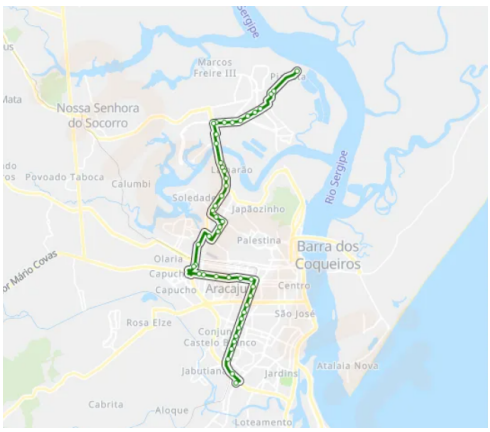

Figura 4.3: Línea 020 - Piabeta Dia.

Un punto de interés en Aracaju es Atalaia, conectada por las líneas 007, 051 y 080. Esta zona es reconocida principalmente por sus playas, y combina áreas residenciales con atractivos turísticos, lo que la convierte en un destino popular tanto para locales como para visitantes. Esta popularidad entre residentes y turistas explica la presencia de múltiples líneas de autobús en el cluster 1 que tienen a Atalaia como destino.

Dentro de este primer cluster, compuesto por catorce líneas de autobús, destacan tres líneas que comunican con Atalaia: las líneas 007, 051 y 080, las cuales por sí solas ya suponen 145, 560 y 102 observaciones respectivamente, lo que representa el 40.7 % del total de observaciones en el cluster.

Debido a su considerable peso dentro del cluster y la alta demanda, se recomienda revisar estas líneas para optimizar su eficiencia y mejorar la cobertura del servicio. Las rutas que comunican con Atalaia podrían ser fusionadas para optimizar recursos y ofrecer un servicio más frecuente y directo, ayudando a reducir el número de autobuses circulando simultáneamente en una misma zona. Además, debido a la elevada cantidad de viajeros y considerando que las líneas actuales no tienen su foco principal en Atalaia, sino que simplemente incluyen esta área en su recorrido, sería beneficioso estudiar la implementación de una nueva línea que comunique directamente con Atalaia. Esta nueva línea podría mejorar significativamente la conectividad y satisfacer mejor la demanda de los usuarios en esta zona tan concurrida.

Asimismo, las líneas que conectan con otras zonas de alta demanda, como la 040 y 060, podrían ser fusionadas. Esta unión no solo reduciría la superposición de rutas, sino que también permitiría aumentar la frecuencia en las áreas más transitadas.

Por otro lado, la presencia de un gran número de coches en el primer cluster indica una alta dependencia del transporte privado en esta área. Al analizar las trayectorias correspondientes a estos grupos de coches, se observa que la mayoría de estos desplazamientos tienen como destino principal el centro de la ciudad (Figura 4.1). Esta tendencia sugiere que el centro urbano es un lugar de gran actividad, atrayendo tanto a residentes como a trabajadores y visitantes. La gran cantidad de vehículos en esta zona provoca congestión de tráfico, lo que dificulta la gestión del transporte urbano y afecta negativamente al medio ambiente.

La preferencia por el coche privado para acceder al centro puede estar influenciada por la percepción de que el transporte público es ineficiente o insuficiente, ya sea por una cobertura limitada o una frecuencia baja. Por lo tanto, es fundamental abordar este problema mediante la implementación de estrategias que promuevan el uso del transporte público. Sería conveniente estudiar la creación de nuevas líneas de autobús en respuesta a la creciente demanda en esta zona, o aumentar la frecuencia de las líneas ya existentes. Mejorar la conectividad de las líneas de autobús que llegan al centro de la ciudad podría incentivar a los habitantes a optar por el uso de transporte público.

La Figura 4.4 muestra únicamente la distribución de puntos del primer cluster. Al observar el origen de las trayectorias en el mapa, se puede notar que muchas de ellas comienzan o terminan en las áreas de Eduardo Gomes, el Distrito Industrial, el aeropuerto y Jardim Centenário.

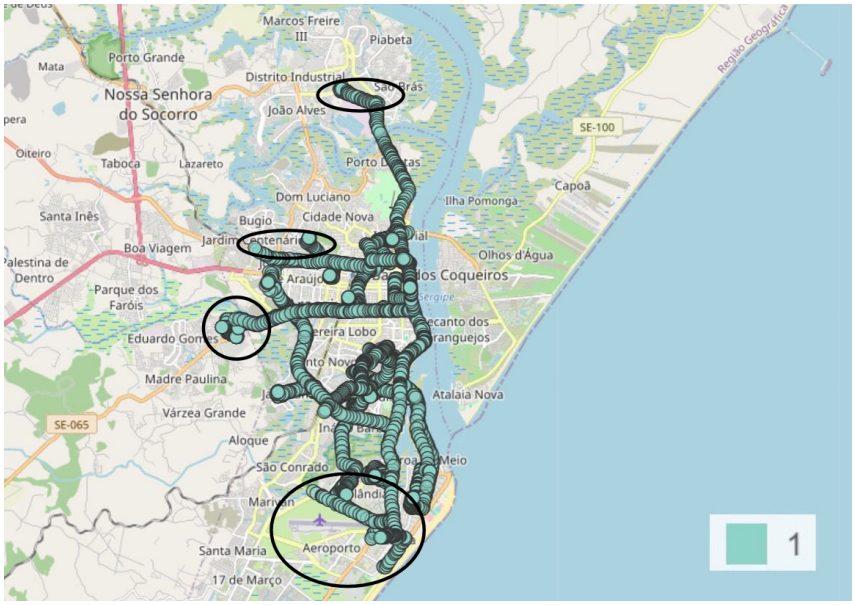

Figura 4.4: Distribución del cluster 1.

Las líneas de autobuses existententes que cubren las áreas mencionadas son:

- Eduardo Gomes: Líneas 031 y 1001.
- Distrito Industrial: Línea 005.
- Aeropuerto: Líneas 008, 100 y 007.
- Jardín Centenário: Línea 061.

Todas estas zonas se encuentran cubiertas por alguna línea de autobús. En Eduardo Gomes, podría estudiarse una mayor frecuencia de las líneas 031 y 1001 para motivar a los usuarios a optar por el transporte público en lugar de sus vehículos privados. En el Distrito Industrial, aumentar la frecuencia de la línea 005 podría reducir significativamente el tráfico de coches particulares, especialmente en horas de entrada y salida del trabajo, ya que se trata de una zona industrial. El Aeropuerto de Aracaju, atendido por las líneas 008, 100 y 007, podría necesitar un incremento en la frecuencia de la línea 061 para controlar mejor la llegada de pasajeros y disminuir el uso de taxis y coches particulares. Por último, en Jardín Centenário, podría revisarse la frecuencia de la línea 061 para proporcionar una alternativa al uso del coche particular para los residentes.

El segundo y tercer cluster cuentan con 139 y 28 coches respectivamente y ninguna presencia de autobús, lo que indica una falta de cobertura de transporte público en estas zonas. Se recomienda estudiar la viabilidad de extender una línea de autobús existente o crear una nueva línea para atender estas áreas y ofrecer así una alternativa al uso del transporte privado.

El cuarto, quinto y sexto cluster, presentan una baja densidad de coches y carecen de servicio de autobuses. Dado que la demanda de transporte en estas áreas podría no ser suficiente para justificar una nueva línea fija, se podrían considerar opciones de transporte más flexibles, como microbuses o servicios bajo demanda.

En cuanto a los clusters de autobuses (Tabla 4.2), algunos están compuestos por pocas trayectorias, lo que sugiere una posible insuficiencia en la cobertura del servicio. El séptimo y octavo cluster, compuestos únicamente por la línea 080, junto con el décimo cluster, asociado a la línea 061, podrían no estar cubriendo adecuadamente las necesidades de transporte en sus zonas. Por lo tanto, se recomienda estudiar la fusión de estas líneas con otras para mejorar la cobertura del servicio o ajustar la frecuencia tras un análisis detallado de la demanda.

De manera similar, el duodécimo cluster, con tan solo 5 autobuses asociados a la línea 001, podría beneficiarse de una redistribución de la ruta, dado que esta línea también aparece en el noveno con una mayor cantidad de autobuses. Finalmente, el cluster trece, con 4 autobuses asociados a la línea 721, requiere también un análisis detallado de la demanda para decidir si es necesario incrementar la frecuencia, fusionar esta línea con otra, o suprimirla, ya que no aparece en ningún otro grupo.

Para estos clusters con baja densidad de coches y sin servicio de autobuses, se podrían implementar servicios de transporte bajo demanda para cubrir las necesidades de movilidad de los residentes sin necesidad de establecer una línea de autobús fija.

En cambio, el noveno cluster agrupa las líneas 001 y 020. La alta densidad de autobuses sugiere que estas líneas podrían estar superpuestas y se podría mejorar la eficiencia fusionándolas en una sola línea con mayor frecuencia o redistribuyendo sus rutas.

Por último, el cluster once está compuesto de autobuses pertenecientes a la línea 707. De nuevo, un análisis de la demanda podría revelar si es necesario aumentar la frecuencia de los autobuses o si la línea está correctamente diseñada para la demanda actual.

En términos generales, las líneas de autobuses con baja cantidad de pasajeros, como la 061 y la 721, podrían requerir un incremento en la frecuencia de los servicios para satisfacer mejor la demanda. La fusión de líneas como las 080 en los clusters siete y ocho, así como la 001 en el nueve y doce, podría aumentar la eficiencia del servicio. Además, la creación de nuevas líneas de autobús en grupos con alta densidad de coches, como el primero, podría disminuir la congestión de tráfico y promover el uso del transporte público.

En conclusión, el análisis mediante el algoritmo DBSCAN ha permitido identificar patrones de movilidad y áreas de oportunidad tanto en el uso de coches como de autobuses. Las recomendaciones propuestas buscan optimizar el sistema de transporte, mejorando la eficiencia del servicio de autobuses y promoviendo alternativas sostenibles al uso del coche. La implementación de estas mejoras podría beneficiar tanto a los usuarios del transporte público como a la comunidad en general.

Capítulo 5

Conclusiones

El análisis mediante el algoritmo DBSCAN ha permitido identificar patrones de movilidad y áreas de oportunidad tanto en el uso de coches como de autobuses. Este algoritmo ha creado un modelo predictivo que se ajusta correctamente a datos espaciales, lo que resulta fundamental para el estudio de la distribución y el comportamiento del transporte en áreas urbanas. La capacidad de DBSCAN para manejar datos espaciales y detectar clusters de densidad variable ha sido esencial para comprender la dinámica del transporte y proponer soluciones adecuadas.

Además, se ha desarrollado una aplicación interactiva utilizando R.Shiny, que permite a los usuarios aplicar el algoritmo DBSCAN a sus propios conjuntos de datos. Esta herramienta no solo ha facilitado la implementación del algoritmo, sino que también permitirá a los usuarios visualizar y analizar los resultados de manera interactiva. La aplicación ofrece una plataforma para explorar diferentes parámetros del algoritmo y observar cómo afectan a la formación de clusters y la identificación de ruido en los datos. Esta herramienta es especialmente valiosa para planificadores urbanos y gestores de transporte que buscan mejorar la eficiencia y la cobertura del servicio de transporte público.

Las recomendaciones propuestas buscan optimizar el sistema de transporte, mejorando la eficiencia del servicio de autobuses y promoviendo alternativas sostenibles al uso del coche. La identificación de clusters con alta densidad de coches han sugerido la necesidad de nuevas líneas de autobús o el aumento de la frecuencia de las existentes para disminuir la dependencia del transporte privado. Asimismo, la reestructuración de rutas redundantes y la implementación de servicios de transporte más flexibles en áreas con baja demanda pueden mejorar significativamente la cobertura y eficiencia del sistema. La implementación de estas mejoras podría beneficiar tanto a los usuarios del transporte público como a la comunidad en general, promoviendo un entorno urbano más sostenible y accesible.

Bibliografía

Aggarwal, C., & Reddy, C. (2014). *DATA CLUSTERING Algorithms and Applications*. Chapman & Hall/CRC Data Mining; Knowledge Discovery Series.

Arredondo, M. (2011). *Entérate: ¿Cómo nace una línea de autobús?* [Accessed: 15-06-2023]. https://elpasajero.metro.net/2011/05/05/enterate-como-nace-una-linea-de-autobus/

Blansché, A. (2023). *fdm2id: Data Mining and R Programming for Beginners* [Accessed: 28-06-2024]. https://cran.r-project.org/web/packages/fdm2id/index.html

Chang, W. (2024). *shiny: Web Application Framework for R* [Accessed: 28-06-2024]. https://cran.r-project.org/web/packages/shiny/index.html

Cheeseman, P. C., & Stutz, J. (1996). Bayesian classification (AutoClass): theory and results. *Advances in knowledge discovery and data mining*, *180*, 153-180.

Chen, W., Ji, M., & Wang, J. (2014). T-DBSCAN: A Spatiotemporal Density Clustering for GPS Trajectory Segmentation. *International Journal of Online Engineering*, *10*(6), 19-24. https://doi.org/10.3991/ijoe.v10i6.3881

Cruz, M., Macedo, H., & Guimares, R. B. A. (2016). *GPS Trajectories* [Accessed: 28-06-2024]. https://doi.org/10.24432/C54S5Z

Ester, M., Kriegel, H., Sander, J., Wimmer, M., & Xu, X. (1998). Incremental clustering for mining in a data warehousing environment. *VLDB*, 323-333.

Ester, M., Kriegel, H., & Xu, X. (1995). A Database Interface for Clustering in Large Spatial Databases. *1st International Conference on Knowledge Discovery and Data Mining*, 94-99.

Ester, M., Kriegel, H., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD*, *96*(34), 226-231.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, *17*(3), 37. https://doi.org/10.1609/aimag.v17i3.1230

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (1996). *Advances in Knowledge Discovery and Data Mining*. American Association for Artificial Intelligence.

Forster, A., & Murphy, A. (2009). CLIQUE: Role-Free Clustering with Q-Learning for Wireless Sensor Networks. *2009 29th IEEE International Conference on Distributed Computing Systems*, 441-449. https://doi.org/10.1109/ICDCS.2009.43

Hahsler, M., Piekenbrock, M., & Doran, D. (2019). dbscan: Fast Density-Based Clustering with R. *Journal of Statistical Software*, *91*(1), 1-30. https://doi.org/10.18637/jss.v091.i01

Han, J., Kamber, M., & Pei, J. (2012). *Data mining concepts and techniques, third edition*. University of Illinois at Urbana-Champaign Micheline Kamber Jian Pei Simon Fraser University.

Hennig, C. (2024). *fpc: Flexible Procedures for Clustering* [Accessed: 28-06-2024]. https://cran.r-project.org/web/packages/fpc/index.html

Jain, A., & Dubes, R. (1988). *Algorithms for Clustering Data*. Prentice-Hall.

Jain, A., Murty, M., & Flynn, P. (1999). Data Clustering: A Review. *ACM Computing Surveys*, *31*(3), 264-323. https://doi.org/10.1145/331499.331504

Jia, L., Yao, W., Jiang, Y., Li, Y., Wang, Z., Li, H., Huang, F., Li, J., Chen, T., & Zhang, H. (2022). *Briefings in Bioinformatics*, *23*(1). https://doi.org/10.1093/bib/bbab415

Kailing, K., Kriegel, H., & Kröger, P. (2004). Density-Connected Subspace Clustering for High-Dimensional Data. *4th SIAM International Conference on Data Mining*, 246-257.

Kaufman, L., & Rousseeuw, P. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons.

Leisch, F., Dimitriadou, E., & Grün, B. (2024). *flexclust: Flexible Cluster Algorithms* [Accessed: 28-06-2024]. https://cran.r-project.org/web/packages/flexclust/index.html

Li, X., & Ye, N. (2002). Grid- and Dummy-Cluster-Based Learning of Normal and Intrusive Clusters for Computer Intrusion Detection. *Quality and Reliability Engineering International*, *18*(3), 231-242.

MacQueen, J. (1967). Some Methods for Classification and Analysis of Multivariate Observations. *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, *1*, 281-297.

Murphy, K. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Ng, R., & Han, K. (1994). Efficient and Effective Clustering Methods for Spatial Data Mining. *20th International Conference on Very Large Data Bases*, 144-155.

Pérez-López, C., & Santín-González, D. (2007). *Minería de datos. Técnicas y herramientas*. Ediciones Paraninfo, SA.

posit Team. (2024). *Shiny Basics* [Accessed: 28-06-2024]. https://shiny.posit.co/r/getstarted/shiny-basics/lesson4/

Sander, J. (2011). *Encyclopedia of Machine Learning*. Springer-Verlag (cap enciclopedia).

Sander, J., Ester, M., Kriegel, H., & Xu, X. (1998). Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, *2*(2), 169-194.

Titterington, D. M., Smith, A., & Makov, U. (1985). *Statistical Analysis of Finite Mixture Distributions*. Wiley.

Velimirovic, A. (2024). *Supervised vs. Unsupervised Machine Learning Explained* [Accessed: 28-06-2024]. https://phoenixnap.com/blog/supervised-vs-unsupervised-learning

Webster, M. (2008). *Cluster Analysis* [Accessed: 28-06-2024]. https://www.merriam-webster.com/dictionary/cluster%20analysis

Weiss, S., & Kulikowski, C. (1991). *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufmann.

Ye, N. (2013). *Data mining: theories, algorithms, and examples*. CRC press.

Yu, X., & Jian, Y. (2005). A new clustering algorithm based on KNN and DENCLUE. *2005 International Conference on Machine Learning and Cybernetics*, *4*, 2033-2038. https://doi.org/10.1109/ICMLC.2005.1527279