MACHINE LEARNING

Fundamentos, algoritmos y aplicaciones para los negocios, industria y finanzas

JACINTO VELASCO REBOLLEDO

MACHINE LEARNING

Fundamentos, algoritmos y aplicaciones para los negocios, industria y finanzas

Madrid • Buenos Aires • México • Bogotá

Ediciones Díaz de Santos
Internet: http//www.editdiazdesantos.com
E-mail: ediciones@editdiazdesantos.com

ISBN: 978-84-9052-530-2
Depósito Legal: M-17696-2024

Fotocomposición y diseño de cubiertas: P55 Servicios Culturales

Printed in Spain Impreso en España

ÍNDICE

Prólogo ..... **XI**

MÓDULO I. Conceptos generales **1**

1.1. Conceptos generales de aprendizaje automático 3
1.2. Infraestructura y tipos de analítica de un proyecto de ciencia de datos 5
1.2.1. Fundamentos sobre infraestructura de datos 5
1.2.2. Diseño de una infraestructura de datos 8
1.2.3. Tipos de analítica 13
1.3. Tipos de aprendizaje automático 15
1.3.1. Aprendizaje supervisado 16
1.3.2. Aprendizaje no supervisado 17
1.3.3. Aprendizaje por refuerzo 17
1.4. Etapas de un proyecto de aprendizaje automático 19
1.4.1. *Feature engineering* 19
1.4.2. Etapas en un proyecto de aprendizaje automático 20
1.5. Técnicas de optimización en aprendizaje automático 25
1.6. Python y buenas prácticas de programación 31
1.6.1. Conceptos fundamentales de Python 31
1.6.2. Buenas prácticas en la programación de modelos de aprendizaje automático 41
1.7. Análisis exploratorio de datos (EDA) 42

MÓDULO II. Aprendizaje supervisado. Clasificación **47**

2.1. Técnicas de validación de modelos 49
2.1.1. Matriz de confusión para modelos de clasificación 49
2.1.2. Curva ROC 53
2.1.3. Validación cruzada o *cross validation* 57
2.1.4. Entropía e índice Gini. Curvas de aprendizaje. 59
2.1.5. Medidas de distancias 63
2.2. Teoría de la decisión: clasificación bayesiana 65
2.2.1. Teorema de Bayes 65
2.2.2. Algoritmo de Naive Bayes 67
2.3. Sistemas de clasificación básicos 71
2.3.1. Árbol de decisión simple 71
2.3.2. Algoritmo KNN 73

2.4. Sistemas de clasificación avanzados 76
2.4.1. *Random forest* 76
2.4.2. *Boosting* 80
2.5. Máquinas de soporte vectorial 87
2.6. Introducción a redes neuronales 92
2.6.1. Definición y tipos de redes neuronales 92
2.6.2. Redes neuronales multicapa 96
2.6.3. Redes neuronales convolucionales (CNN) 102
2.6.4. Redes neuronales recurrentes (RNN) 106
2.7. Procesamiento del lenguaje natural (NLP) 108
2.8. Estrategias de resolución de problemas en modelos de clasificación 115

MÓDULO III. Modelos de regresión y series temporales 119
3.1. Validación en modelos de regresión 121
3.2. Regresión lineal múltiple 128
3.3. Regresión logística 134
3.4. Modelos de árboles con regresión 139
3.5. Regresión con vector soporte 144
3.6. Series temporales con aprendizaje automático 147
3.6.1. Introducción a las series temporales y análisis de componentes 147
3.6.2. Modelos ARIMA y SARIMA 150
3.6.3. Series temporales con redes neuronales 155

MÓDULO IV. Aprendizaje no supervisado y por refuerzo 159
4.1. Análisis de componentes principales (PCA) 161
4.2. Modelos de aprendizaje no supervisado 163
4.2.1. Agrupación en clústeres: K-means 163
4.2.2. DBSCAN 167
4.2.3. Algoritmos a priori 169
4.2.4. Técnicas de validación de modelos no supervisados 173
4.3. Aprendizaje por refuerzo 177

MÓDULO V. Casos de uso 181
5.1. Caso de uso de una segmentación de prospectos 183
5.1.1. Definición del modelo 183
5.1.2. Descripción del caso de uso 184
5.1.3. Resolución técnica 184
5.1.4. Análisis de resultados 188

5.2. Caso de uso de un modelo de propensión 192
5.2.1. Definición del modelo 192
5.2.2. Descripción del caso de uso 193
5.2.3. EDA 194
5.2.4. Resolución técnica 196
5.2.5. Cálculo del tiempo de retorno 204
5.2.6. Distribución de nuevas probabilidades 208
5.2.8. Conclusiones y análisis de negocio 210
5.3. Análisis de series temporales para predecir el número de viajeros en el transporte urbano 210
5.3.1. Descripción del caso de uso 210
5.3.2 Resolución técnica usando SARIMA 211
5.3.3. Conclusiones y análisis de negocio 217
5.4. Caso de uso de un modelo de abandono (*churn*) para explicar la pérdida de clientes en el sector financiero. Optimización de parámetros 218
5.4.1. Definición del modelo 218
5.4.2. Descripción del caso de uso 219
5.4.3. EDA 220
5.4.4. Resolución técnica 223
5.4.5. Conclusiones y análisis de negocio 229
5.5. Caso de uso para el análisis de sentimiento en redes sociales mediante el teorema de Bayes. 231
5.5.1. Definición del modelo 231
5.5.2. Descripción del caso de uso 232
5.5.3. Resolución técnica 233
5.5.4. Conclusiones y análisis de negocio 235

Bibliografía 237

PRÓLOGO

En la intersección entre la innovación tecnológica y las estrategias empresariales, emerge el mundo del aprendizaje automático. En la era digital, donde los datos son valorados como el nuevo petróleo y la habilidad para tomar decisiones precisas y ágiles se considera la clave del éxito empresarial, el dominio de las herramientas y técnicas del aprendizaje automático se ha vuelto imprescindible tanto para líderes empresariales consolidados como para estudiantes universitarios.

Este campo en constante evolución no solo revoluciona la forma en que las organizaciones abordan sus operaciones y estrategias comerciales, sino que también abre un vasto espectro de oportunidades para aquellos que están dispuestos a explorarlo. Desde la personalización de experiencias para los clientes hasta la detección de patrones ocultos en grandes conjuntos de datos, el aprendizaje automático se ha convertido en un motor fundamental de la innovación y el progreso en el ámbito empresarial moderno.

En este contexto, comprender los fundamentos y aplicaciones del aprendizaje automático se convierte en una ventaja competitiva crucial. Los líderes empresariales deben estar equipados con el conocimiento necesario para aprovechar al máximo el potencial de esta tecnología, mientras que los estudiantes universitarios buscan adquirir las habilidades esenciales que les permitirán prosperar en el mercado laboral del futuro.

El autor, con más de 20 años de experiencia en la aplicación de estas metodologías para resolver casos de uso en diversas áreas empresariales, desde la financiera e industrial hasta la comercial y de marketing, ha destilado su conocimiento en este libro. La obra ha sido diseñada para ser una guía completa y accesible sobre aprendizaje automático aplicado a los negocios, representando un recurso invaluable tanto para aquellos que buscan mejorar su comprensión de las complejidades del aprendizaje automático en el contexto empresarial, como para profesores que desean impartir conocimientos actualizados y relevantes en el aula universitaria.

Dentro de las páginas de este compendio, los lectores serán guiados en un fascinante viaje que parte desde los cimientos elementales hasta las sofisticadas técnicas del aprendizaje automático, todo ello expuesto mediante un lenguaje matemático comprensible por cualquier lector con formación básica en economía, finanzas o tecnología. A lo largo de este recorrido, se explorará minuciosamente cómo aplicar algoritmos de aprendizaje supervisado y no supervisado para abordar una amplia variedad de desafíos empresariales.

Desde la anticipación de la demanda hasta la optimización de la cadena de suministro, así como el análisis de sentimientos y la personalización de ex-

periencias para los clientes, este libro se convierte en un recurso esencial para aquellos que buscan ampliar su comprensión y dominio en el ámbito del aprendizaje automático aplicado al entorno empresarial.

Este texto va más allá de meras teorías abstractas; su enfoque se concentra en proporcionar a los lectores las habilidades prácticas necesarias para afrontar los desafíos cambiantes del mundo empresarial contemporáneo. Además, está meticulosamente estructurado para ser utilizado tanto por estudiantes universitarios en busca de conocimientos fundamentales como por profesionales que aspiran a integrar el aprendizaje automático en sus actividades cotidianas.

En el libro se incluyen ejemplos en Python para ayudar a comprender mejor la teoría presentada. Estos ejemplos prácticos muestran cómo aplicar los conceptos de aprendizaje automático a situaciones empresariales concretas. No solo clarifican la teoría expuesta en el texto, sino que también proporcionan a los lectores herramientas prácticas que pueden aplicar en sus propios proyectos empresariales. Al ver cómo se implementan los conceptos de aprendizaje automático en código, los lectores pueden comprender mejor cómo adaptar y aplicar estas técnicas a sus propios casos de uso específicos dentro de sus organizaciones. Esta combinación de teoría y ejemplos prácticos en Python asegura que los lectores no solo adquieran conocimientos conceptuales sobre aprendizaje automático, sino que también desarrollen habilidades prácticas que puedan utilizar para abordar desafíos empresariales reales.

En última instancia, este libro tiene como objetivo capacitar a sus lectores para que se conviertan en líderes visionarios y tomadores de decisiones informadas, capaces de aprovechar el poder del aprendizaje automático para impulsar la innovación, la eficiencia y el crecimiento en sus organizaciones y en la sociedad en general.

Dr. Jacinto Velasco Rebolledo

Módulo IV

APRENDIZAJE NO SUPERVISADO Y POR REFUERZO

4.1. ANÁLISIS DE COMPONENTES PRINCIPALES (PCA)

El análisis de componentes principales (PCA) es una técnica estadística ampliamente utilizada en diversos campos, desde la biología hasta la ingeniería y la economía, debido a su capacidad para simplificar conjuntos de datos complejos sin perder información crucial. Su objetivo principal es reducir la dimensionalidad de los datos, lo que significa representar la información original utilizando un número menor de variables, conocidas como componentes principales. Esta reducción de dimensionalidad es esencial para manejar conjuntos de datos grandes y complejos, así como para visualizar y comprender la estructura subyacente de los datos.

El proceso de PCA implica una transformación lineal de las variables originales para encontrar un nuevo sistema de coordenadas en el que los datos estén mejor representados. Esto se logra mediante la identificación de los ejes principales de variación en los datos, que son ortogonales entre sí y ordenados según la cantidad de variabilidad que explican. Estos ejes principales, o componentes principales, son combinaciones lineales de las variables originales y están descorrelacionados entre sí, lo que significa que capturan diferentes aspectos de la variabilidad presente en los datos sin redundancia. El proceso de PCA implica los siguientes pasos:

Estandarización de datos

Se normalizan los datos para asegurar que todas las variables tengan la misma escala. Esto es importante para evitar que las variables con escalas más grandes dominen el análisis. Dada una matriz A de n filas y p columnas, la ecuación matemática sería la siguiente

$$x_{n,p}^{N} = \frac{x_{np} - \overline{x_{np}}}{\sigma}$$

Cálculo de la matriz de covarianza o matriz de correlación

Se calcula la matriz que describe las relaciones entre todas las variables del conjunto de datos.

$$S = \frac{1}{n} \cdot \sum_{i=1}^{N} \left(x_{np} - \overline{x_p}\right)^T \cdot \left(x_{np} - \overline{x_p}\right)$$

Descomposición de la matriz

Se realiza una descomposición de la matriz de covarianza o correlación para obtener los autovectores y autovalores asociados. Si denominamos a λ los autovalores, S a la matriz de covarianza y V a la matriz de autovectores, la formulación matemática sería:

$$\begin{cases} |S - \lambda \cdot I| = 0 \\ |S - \lambda \cdot I| \cdot V = 0 \end{cases}$$

Selección de componentes principales

Se ordenan los autovalores de mayor a menor y se eligen los primeros k componentes principales que explican la mayor parte de la varianza en los datos. Generalmente, se decide un umbral de varianza explicada para determinar cuántos componentes principales se mantienen.

Transformación de datos: se proyectan los datos originales sobre el espacio definido por los componentes principales seleccionados. Esto implica multiplicar la matriz de datos original por la matriz de autovectores seleccionados.

$$A_N \cdot V = PC$$

El resultado final del PCA es un conjunto de componentes principales que son combinaciones lineales de las variables originales y que capturan la mayor parte de la variabilidad presente en los datos originales. Estos componentes pueden utilizarse para visualización, análisis exploratorio de datos o como entrada para otros algoritmos de aprendizaje automático.

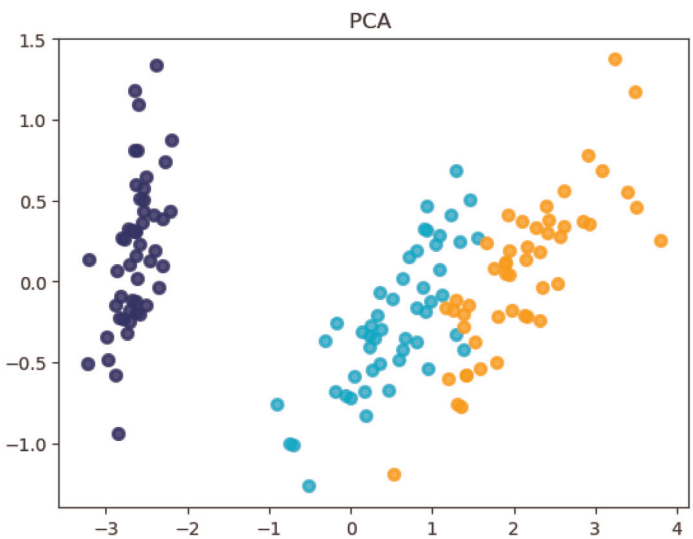

Figura 4.1. Ejemplo de clasificación PCA.

Código Python

```
PCA(n_components, whiten,svd_solver, tol, iterated_power, random_state,-
copy)
```

Parámetros:

- **Número de componentes principales (n_components)**: especifica el número de componentes principales
- **whiten**: este parámetro, si se establece en True, indica si se debe aplicar el blanqueo de datos antes de calcular los componentes principales.
- **svd_solver**: PCA utiliza una descomposición de valores singulares (SVD) para calcular los componentes principales. Algunas opciones comunes son auto, full, randomized y arpack.
- **tol**: este parámetro controla la tolerancia para el corte de la singularidad en SVD.
- **iterated_power:** este parámetro solo se aplica si svd_solver es randomized. Controla la cantidad de potencias iterativas.
- **random_state**: controla la aleatoriedad en la inicialización del estado interno del generador de números aleatorios.
- **copy**: si se establece en True, una copia de los datos se realiza antes de realizar el PCA.

4.2. MODELOS DE APRENDIZAJE NO SUPERVISADO

4.2.1. Agrupación en clústeres: K-means

El *clustering*, una técnica fundamental en el análisis no supervisado, juega un papel crucial en la exploración y comprensión de conjuntos de datos complejos al permitir descubrir estructuras y patrones inherentes que pueden pasar desapercibidos a simple vista, sin la necesidad de etiquetas predefinidas.

En el proceso de conglomerado, cada punto de datos se representa como un vector en un espacio multidimensional, donde cada dimensión corresponde a una característica o atributo del conjunto de datos. Estas características pueden abarcar variables numéricas, categóricas o incluso de texto, proporcionando así la flexibilidad necesaria para analizar una amplia gama de datos.

El objetivo principal del *clustering* es agrupar los puntos de datos de manera que los miembros de un mismo clúster sean más similares entre sí que con los puntos de otros clústeres. Esta similitud se evalúa utilizando medidas de distancia o similitud, como la distancia euclidiana, la distancia de Manhattan o la similitud del coseno, entre otras.

Los resultados del agrupamiento pueden interpretarse y visualizarse para obtener información valiosa sobre la estructura subyacente de los datos. Esta interpretación puede revelar patrones emergentes, segmentos de clientes, relaciones entre variables o incluso anomalías en los datos, lo que facilita la toma de decisiones informadas en una variedad de campos y aplicaciones.

El algoritmo K-means es una técnica de agrupación de datos ampliamente utilizada en el aprendizaje no supervisado. Su objetivo principal es dividir un conjunto de datos en un número predefinido de k grupos, con el fin de minimizar la varianza intraclúster y maximizar la varianza interclúster. Este algoritmo se basa en la asignación iterativa de puntos de datos a clústeres, llevándose a cabo iteraciones hasta alcanzar la convergencia o un criterio de parada definido.

Dado un conjunto de valores $X= \{x_1, x_2, \dots, x_n\}$ que queremos dividir en C_K grupos, de tal manera que minimicemos la suma de las distancias cuadradas de cada punto al centro de su respectivo conglomerado. Para ello, se define μ_i como el centroide C_K, cuya función a minimizar es la siguiente:

$$M = \arg\min \sum_{i=1}^{k} \frac{1}{|C_K|} \cdot \sum_{x \in C_K} \|x_i - \mu_i\|^2$$

Para realizar la minimización de la función se siguen los siguientes pasos:

1. Inicialización de centroides: seleccionamos k puntos aleatorios como centroides iniciales.
2. Asignación de puntos al clúster más cercano: para cada punto x_i, calculamos la distancia Euclidiana a cada centroide y asignamos x_i al clúster cuyo centroide está más cercano.

$$C_K^t = \left\{ x_i : \|x_i - \mu_i^t\|^2 \leq \left\|x_i - \mu_j^t\right\|^2 \forall j, 1 \leq j \leq k \right\}$$

3. Actualización de los centroides: para cada clúster, actualizamos el centroide, como el promedio de todos los puntos asignados a ese clúster.

$$\mu_k^{t+1} = \frac{1}{\left|C_k^t\right|} \cdot \sum_{x \in C_K} x_i$$

4. El algoritmo convergerá cuando los centroides no cambien o cambien por debajo de un cierto umbral predefinido.

El algoritmo K-means es una herramienta altamente valorada en diversos campos debido a su capacidad para agrupar datos de manera eficiente y sencilla. Entre sus múltiples aplicaciones, una de las más destacadas se encuentra en el ámbito del marketing, donde se emplea para llevar a cabo la segmentación de mer-

cado. En esta tarea, K-means desempeña un papel crucial al dividir a los clientes en grupos homogéneos basados en una variedad de criterios, que pueden incluir comportamientos de compra, preferencias de productos o servicios, características demográficas y otros datos relevantes. La elección del número de conglomerados, que suele situarse entre 3 y 5, aunque puede variar según la complejidad del mercado y la cantidad de datos disponibles, es fundamental para obtener segmentos significativos y útiles. La segmentación de mercado mediante K-means permite a las empresas personalizar sus estrategias de marketing y publicidad de forma más precisa y efectiva. Al comprender mejor las necesidades y preferencias específicas de cada segmento de clientes, las empresas pueden adaptar sus mensajes, ofertas y servicios para satisfacer de manera más adecuada las demandas de cada grupo. Esto no solo puede mejorar la eficacia de las campañas de marketing, sino también aumentar la retención de clientes y fortalecer la lealtad hacia la marca.

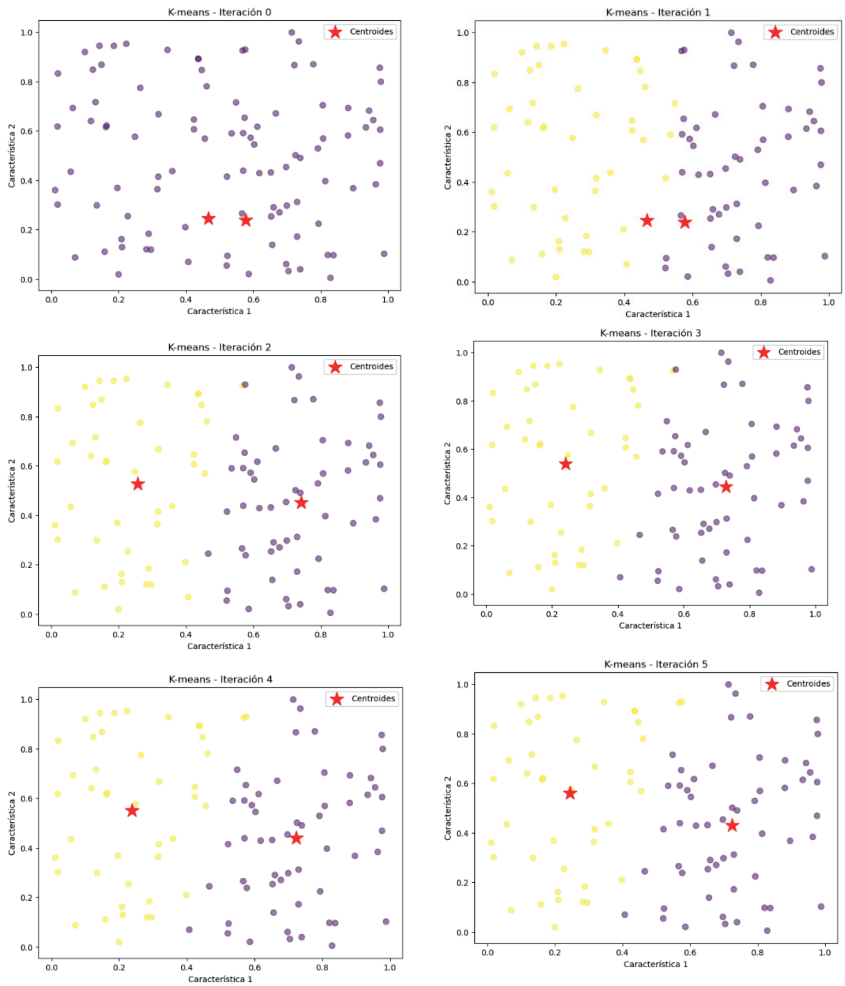

Figura 4.2. Pasos en un proceso iterativo de K-means.

Otra aplicación común de K-means es la compresión de imágenes. En este caso, el algoritmo agrupa los píxeles de una imagen en grupos representados por los colores medios, lo que reduce el número de colores necesarios para representar la imagen sin perder su calidad visual de manera significativa. Esta técnica es especialmente útil en situaciones donde se necesita optimizar el almacenamiento o la transmisión de imágenes, como en el caso de la compresión de imágenes para su uso en sitios web o en la transmisión de datos multimedia.

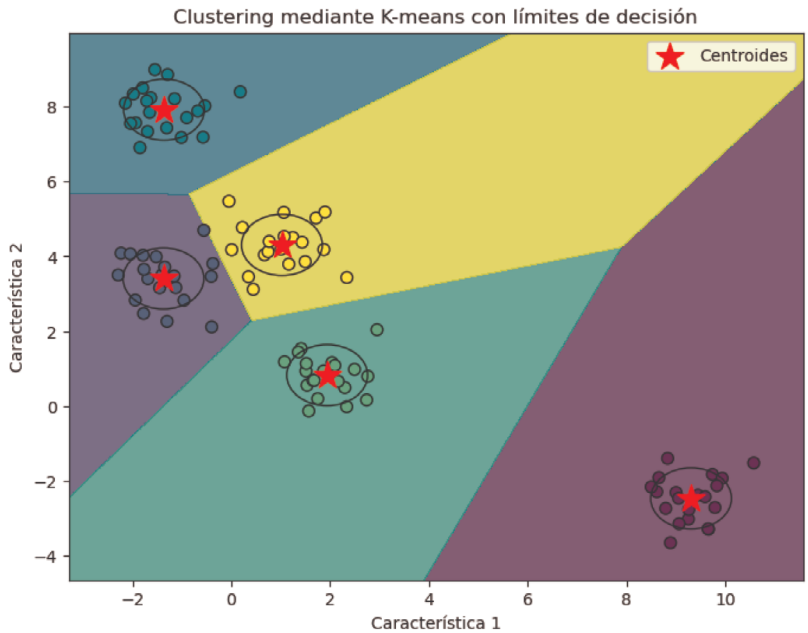

Figura 4.3. Ejemplo de clasificador K-means.

En el análisis de redes sociales, K-means se utiliza para agrupar usuarios con perfiles de comportamiento similares en diferentes segmentos. Esto facilita la identificación de comunidades dentro de una red social, así como la detección de patrones de interacción entre usuarios. Por ejemplo, en plataformas como Twitter, K-means podría emplearse para agrupar usuarios con intereses similares o comportamientos de interacción, lo que sería útil para personalizar la experiencia del usuario o para identificar influencers dentro de la plataforma.

En el ámbito del comercio electrónico, K-means se emplea para segmentar clientes en tiendas en línea según sus patrones de compra, preferencias de productos, comportamientos de navegación y otros datos relevantes. Esto ayuda a las empresas a comprender mejor a sus clientes y a ofrecer recomendaciones de productos más precisas y relevantes, así como a personalizar la experiencia de compra en línea para cada usuario.

Además, en el análisis de texto y minería de datos, K-means se utiliza para agrupar documentos de texto o palabras similares en temas o categorías rela-

cionadas. Esto facilita la organización y comprensión de grandes conjuntos de datos textuales, así como la identificación de tendencias o temas emergentes, y mejora la búsqueda y recuperación de información.

Código Python

KMeans(n_clústeres, init, n_init, max_iter, tol, algorithm)

Los parámetros son:

- **n_clústeres**: número de clústeres que se desean encontrar.
- **init**: método de inicialización de los centroides. Puede ser k-means++, random, o un array de centroides iniciales.
- **n_init**: número de veces que se ejecutará el algoritmo K-means con diferentes centroides iniciales. El resultado final será el modelo con la inercia más baja.
- **max_iter**: número máximo de iteraciones en cada ejecución del algoritmo K-means.
- **tol**: tolerancia para declarar la convergencia del algoritmo, es decir, el criterio de parada. Si la diferencia entre los centroides de dos iteraciones consecutivas es menor que tol, se considera que el algoritmo ha convergido.
- **algorithm**: algoritmo para calcular el K-means. Puede ser auto, full, elkan, entre otros.

4.2.2. DBSCAN

El algoritmo DBSCAN (*density-based spatial clustering of applications with noise*) es una técnica de agrupamiento ampliamente utilizada en el campo del aprendizaje automático y la minería de datos. A diferencia de otros algoritmos de *clustering*, como k-medias, que requieren la especificación previa del número de clústeres, DBSCAN es capaz de identificar áreas densas de puntos en el espacio de características de manera automática, sin necesidad de conocimiento previo sobre la cantidad de clústeres presentes en los datos.

DBSCAN se fundamenta en el concepto de densidad de puntos en el espacio de características. La idea principal detrás de este algoritmo es que los puntos cercanos entre sí en el espacio de características tienen más probabilidades de pertenecer al mismo clúster que puntos más alejados. Por lo tanto, en lugar de definir los clústeres mediante la minimización de una métrica de distancia como en k-medias, DBSCAN busca identificar áreas densas de puntos y considerarlos como clústeres.

Una de las ventajas más importantes de DBSCAN es su capacidad para detectar clústeres de diferentes formas y tamaños de manera automática. Esto

se debe a su capacidad para adaptarse a la densidad local de los puntos en el espacio de características. Mientras que otros algoritmos de *clustering* pueden tener dificultades para identificar clústeres no esféricos o de formas irregulares, DBSCAN puede manejar estas situaciones de manera efectiva.

Además, DBSCAN es robusto ante la presencia de ruido y *outliers* en los datos. Los puntos que no están lo suficientemente cerca de ningún otro punto se consideran ruido y no se asignan a ningún clúster. Esto significa que DBSCAN puede identificar y aislar puntos anómalos en el conjunto de datos, lo que puede ser útil en diversas aplicaciones, como la detección de anomalías o la eliminación de ruido en conjuntos de datos.

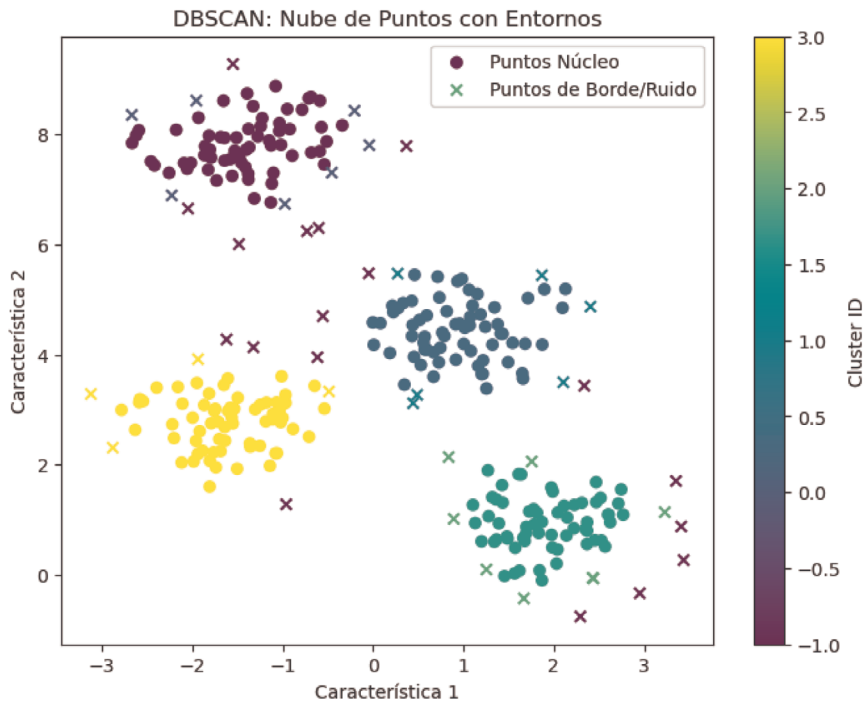

Figura 4.4. Ejemplo de clasificador DBSCAN.

Denominamos ε como el radio del vecindario alrededor de un punto x. Para cada punto p en el conjunto de datos N(p), definimos su vecindario con respecto a un punto x en un conjunto de datos X como:

$$N(x)=\{p \epsilon X: dist(p,x) \leq \varepsilon\}$$

Un punto p se considera un punto núcleo si su vecindario contiene al menos un número mínimo de puntos (MinPts), es decir, N(x) ≥MinPts. Un punto de borde es un punto que no es un punto núcleo, pero está en el vecindario de al menos un punto núcleo, matemáticamente sería N(X) ≤MinPts. Los puntos que no son ni puntos núcleo ni puntos de borde se consideran puntos de ruido

y no se asignan a ningún clúster. El proceso de clusterización para cada punto p se realiza de la siguiente manera:

- Si p es un punto núcleo, se forma un nuevo clúster con él y todos los puntos alcanzables desde él.
- Si p es un punto de borde, se asigna al mismo clúster que un punto núcleo en su vecindario.
- Si p es un punto de ruido, no se asigna a ningún clúster.

El algoritmo DBSCAN puede ser implementado de manera eficiente utilizando estructuras de datos como árboles de búsqueda espacial. Al ajustar los parámetros ε y MinPts, podemos controlar la sensibilidad del algoritmo y obtener diferentes resultados de *clustering* según las características del conjunto de datos y los requisitos del problema.

Una de las diferencias más importante con el algoritmo de k-medias es que no hay que dar un número de clústeres predefinido. Además, k-medias funciona mejor en conjuntos de datos donde los clústeres son aproximadamente esféricos y de tamaño similar, y puede ser sensible a *outliers*, ya que los puntos distantes de los centroides pueden influir en la definición de los clústeres.

Código Python

```
DBSCAN(eps, min_samples, metric, algorithm, leaf_size)
```

Parámetros:

- **eps**: la distancia máxima entre dos muestras para que una se considere como en el vecindario de la otra. Este es el parámetro más importante en DBSCAN.
- **min_samples**: el número mínimo de muestras en un vecindario para que un punto sea considerado como núcleo. Esto incluye el punto en sí
- **metric**: la métrica de distancia utilizada para calcular las distancias entre puntos. Las opciones comunes incluyen euclidean, manhattan, cosine, entre otras.
- **algorithm**: el algoritmo utilizado para calcular las distancias entre puntos. Las opciones incluyen auto, ball_tree, kd_tree y brute.
- **leaf_size**: tamaño de hoja pasado a los algoritmos de árbol. Esto puede afectar la velocidad y el uso de memoria del algoritmo.

4.2.3. Algoritmos a priori

El algoritmo Apriori es una herramienta fundamental en la minería de datos, diseñada para descubrir patrones de asociación en conjuntos de datos. Su

principal objetivo es identificar conjuntos de elementos que tienden a aparecer juntos con frecuencia en un conjunto de transacciones. Este algoritmo se basa en la premisa de que, si un conjunto de elementos es frecuente, entonces todos sus subconjuntos también deben serlo.

El nombre Apriori se deriva del concepto filosófico de "a priori", que se refiere al conocimiento previo o las suposiciones fundamentales antes de realizar un análisis. En el contexto del algoritmo Apriori, se utilizan reglas de asociación para determinar la frecuencia de los conjuntos de elementos y extraer patrones de asociación significativos a partir de ellos.

El funcionamiento del algoritmo Apriori consta de varias etapas. En primer lugar, se identifican los elementos individuales que aparecen con una frecuencia mínima en el conjunto de datos, conocido como el "umbral de soporte mínimo". Luego, se generan conjuntos de elementos más grandes a partir de estos elementos individuales, utilizando la propiedad Apriori mencionada anteriormente. Este proceso se repite iterativamente para encontrar conjuntos de elementos cada vez más grandes que cumplan con el umbral de soporte mínimo establecido.

Una vez identificados los conjuntos de elementos frecuentes, el algoritmo apriori puede emplearse para generar reglas de asociación entre los elementos. Estas reglas describen las relaciones entre los elementos y pueden aplicarse en diversas aplicaciones prácticas, como la recomendación de productos, el análisis de cestas de compra y la optimización de inventarios.

A continuación, desarrollaremos tres medidas esenciales en el algoritmo Apriori:

- **Soporte (*support*).** El soporte de un conjunto de elementos (o un conjunto de ítems) es la proporción de transacciones en el conjunto de datos que contienen ese conjunto de elementos. Matemáticamente, el soporte se calcula como la frecuencia relativa de un conjunto de elementos en todas las transacciones.

$$\text{Soporte (X)} = \frac{\text{Nº transaciones que contienen X}}{\text{Nº total de transacciones}}$$

- **Confianza (*confidence*).** La confianza mide la probabilidad de que ocurra una consecuencia (o elemento) dado que ocurre un antecedente (o conjunto de elementos). En términos de reglas de asociación, la confianza se refiere a la proporción de transacciones que contienen tanto el antecedente como la consecuencia en relación con las transacciones que contienen el antecedente.

$$\text{Confianza } (X \rightarrow Y) = \frac{\text{Soporte}(X \cup Y)}{\text{Soporte } (X)}$$

- **Lift.** Mide la importancia de una regla de asociación sobre el azar. Indica cuántas veces más probable es que ocurra la consecuencia dado el antecedente, en comparación con si fueran eventos independientes. Un valor de lift mayor que 1 indica una correlación positiva, mientras que un valor menor que 1 indica una correlación negativa.

$$\text{Lift}\ (X \rightarrow Y) = \frac{\text{Confianza}(X \rightarrow Y)}{\text{Soporte}\ (Y)}$$

El algoritmo Apriori utiliza estos conceptos para generar conjuntos de ítems frecuentes y generar reglas de asociación a partir de ellos. Comienza identificando los ítems individuales que tienen un soporte mayor que el umbral mínimo establecido. Luego, se generan conjuntos de ítems más grandes mediante la combinación de ítems frecuentes anteriores, y estos conjuntos se evalúan utilizando los criterios de soporte y confianza. Las reglas de asociación finales se seleccionan basadas en los umbrales de soporte y confianza especificados por el usuario. Veamos un ejemplo sencillo. En un restaurante se tienen las siguientes combinaciones:

Trans					
1	1	1	1	0	0
2	1	0	1	1	0
3	0	1	1	1	0
4	1	0	0	1	1
5	0	1	1	0	1

Figura 4.5. Ejemplo de caso de uso del algoritmo A priori.

Aplicando el algoritmo Apriori podemos ver qué tipo de reglas hay en la combinación de estos 5 productos y cuál es la que tiene mejor probabilidad. Esto se puede usar, perfectamente para recomendar un producto.

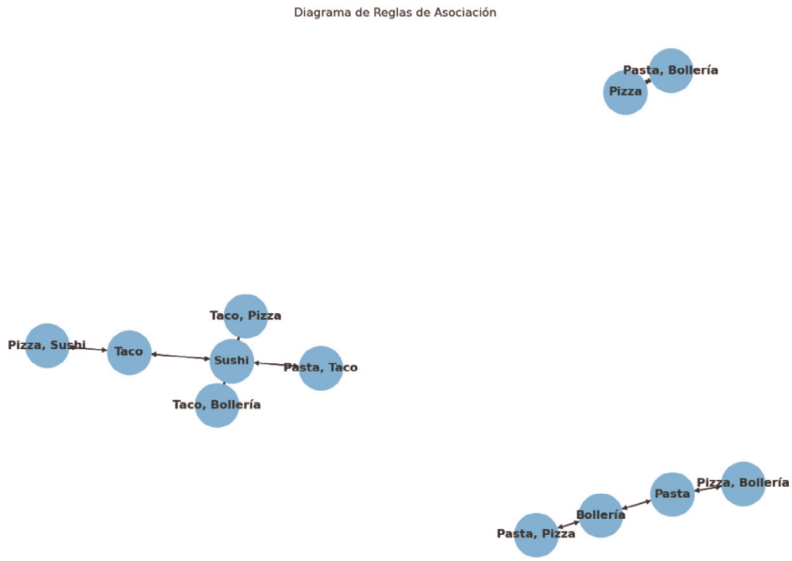

Figura 4.6. Ejemplo de reglas de asociación.

La mejor medida depende del objetivo a lograr con tu análisis. Si estás interesado en la fuerza de la relación entre el antecedente y el consecuente, la confianza podría ser la medida más relevante. Si estás buscando reglas que sean aplicables a una gran cantidad de transacciones, el soporte podría ser más importante. Por otro lado, si deseas identificar reglas que sean más que una simple correlación y tengan una relación causal o de dependencia entre el antecedente y el consecuente, entonces el *lift* podría ser la medida más adecuada. En muchos casos, es útil considerar múltiples medidas para obtener una comprensión más completa de las reglas de asociación. A continuación, se muestran los resultados del ejemplo mostrado.

Regla	Poste-rior	Soporte anterior	Soporte posterior	Soporte (S)	Confianza (C)	Lift (L)	Mejor
Taco	Sushi	0,6	0,8	0,6	1	1,25	S
Sushi	Taco	0,8	0,6	0,6	0,75	1,25	S
Taco	Sushi	0,6	0,8	0,6	1	1,25	C
Taco, Bollería	Sushi	0,2	0,8	0,2	1	1,25	C
Pasta, Pizza	Bollería	0,2	0,6	0,2	1	1,66	L
Pasta, Pizza	Bollería	0,2	0,6	0,2	1	1,66	L
Pizza, Bollería	Pasta	0,2	0,6	0,2	1	1,66	L

Como se puede apreciar la asociación taco y sushi ganan tanto en soporte como en confianza, por lo que la mejor opción es ofrecer sushi si antes se ha pedido un taco.

Código Python

```
apriori(df, min_support, use_colnames, max_len)
```

Parámetros:

- **df**: el DataFrame que contiene los datos transaccionales. Debe estar en un formato donde las filas representan transacciones y las columnas representan ítems.
- **min_support**: el umbral mínimo de soporte para considerar un conjunto de ítems como frecuente. El soporte se define como la proporción de transacciones en las que aparece un conjunto de ítems. Se expresa como un valor entre 0 y 1.
- **use_colnames**: booleano que indica si se deben usar los nombres de columna originales en el DataFrame para los conjuntos de ítems generados en lugar de los índices de columna.
- **max_len**: la longitud máxima de los conjuntos de ítems generados. Esto controla el tamaño máximo de los conjuntos de ítems que se buscarán como frecuentes.

4.2.4 Técnicas de validación de modelos no supervisados

La evaluación de los resultados del algoritmo K-means es esencial para determinar la calidad y utilidad de la agrupación de datos obtenida. Al evaluar los resultados de K-means, los analistas buscan comprender la coherencia y separación de los clústeres identificados, lo que les permite tomar decisiones informadas sobre la interpretación de los datos agrupados. Esta evaluación se realiza mediante diversas técnicas que proporcionan una visión general de la estructura y cohesión de los clústeres. Al comprender cómo los datos se agrupan y distribuyen en diferentes grupos, los analistas pueden determinar si la agrupación captura adecuadamente las relaciones y patrones subyacentes en los datos. Esta comprensión es crucial para garantizar que la agrupación resultante sea significativa y útil para su aplicación en diferentes contextos analíticos.

La inercia, también conocida como suma de los cuadrados dentro de los clústeres (WCSS), es una medida fundamental en la evaluación de los resultados del algoritmo K-means. Esta métrica cuantifica la dispersión de los puntos dentro de cada clúster, calculando la suma de las distancias al cuadrado de

cada punto al centroide de su clúster correspondiente. En esencia, la inercia representa la cohesión intraclúster: cuanto menor sea la inercia, mayor será la cohesión de los puntos dentro de cada clúster, lo que indica una agrupación más compacta y homogénea. Por otro lado, una mayor inercia sugiere una dispersión más amplia de los puntos dentro de los clústeres, lo que puede indicar una agrupación menos coherente. Matemáticamente se define como:

$$I = \sum_{i=1}^{k} \sum_{x \in C_K} \|x \quad - \mu_i\|^2$$

La silueta, también conocida como puntuación de silueta, es una métrica poderosa en la evaluación de la calidad de los clústeres generados por el algoritmo K-means. Esta medida proporciona información detallada sobre cómo se agrupan los puntos dentro de cada clúster y cómo se separan entre sí los clústeres adyacentes. La puntuación de silueta de un punto se calcula como la diferencia entre la distancia promedio hacia los otros puntos en el mismo clúster (cohesión intraclúster) y la distancia promedio hacia los puntos en el clúster más cercano diferente al que pertenece (separación interclúster), normalizada por el máximo de estas dos distancias. En esencia, una puntuación de silueta cercana a 1 indica que el punto está bien clasificado en su clúster y que está relativamente lejos de los puntos en los clústeres vecinos, lo que sugiere una agrupación coherente y bien definida. Por otro lado, una puntuación de silueta cercana a -1 indica que el punto podría haber sido mal clasificado, ya que está más cerca de los puntos en un clúster vecino que de los puntos en su propio clúster, lo que sugiere una separación inadecuada entre clústeres o una superposición entre ellos. Matemáticamente se define como:

$$S = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Donde a(i) es la distancia promedio entre i y todos los demás puntos en el mismo clúster, y b(i) es la distancia promedio entre i y todos los puntos en el clúster más cercano diferente de i.

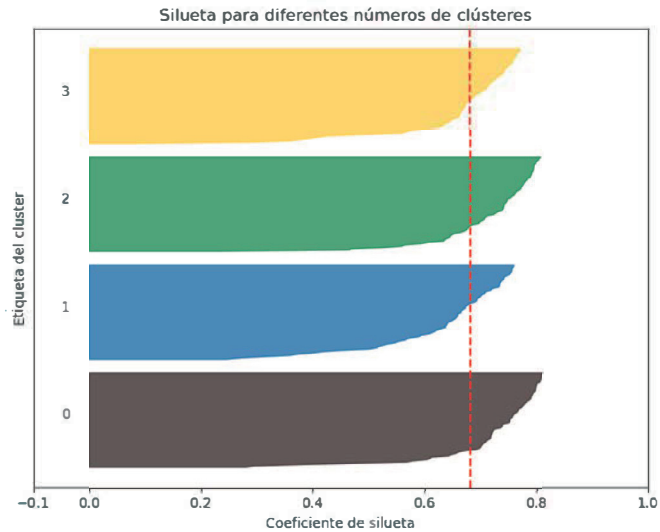

Figura 4.7. Silueta para diferentes números de clústeres.

El índice de Davies-Bouldin (*Davies-Bouldin index*) es una métrica ampliamente utilizada para evaluar la calidad de los clústeres generados por el algoritmo K-means. Este índice proporciona una medida de la "separación" entre los clústeres, donde valores más bajos indican clústeres mejor definidos y más separados entre sí. El cálculo del índice de Davies-Bouldin implica evaluar tanto la dispersión dentro de cada clúster como la distancia entre los centroides de los clústeres. Concretamente, se calcula como la media de las distancias entre los centroides de los clústeres, dividida por la distancia entre los centroides y la dispersión dentro del clúster. En esencia, un índice de Davies-Bouldin bajo sugiere que los centroides de los clústeres están bien separados entre sí y que los clústeres son cohesivos y definidos de manera clara, lo que indica una agrupación efectiva y significativa de los datos. Por otro lado, un índice de Davies-Bouldin más alto puede indicar que los clústeres están más superpuestos o que la separación entre ellos es menos clara, lo que sugiere una agrupación menos efectiva o subóptima. Matemáticamente se define como:

$$DB = \frac{1}{k} \cdot \sum_{i=1}^{N} \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)$$

Donde C_i y C_j son los centroides de los clústeres i y j, respectivamente, σ_i y σ_j son la dispersión promedio dentro de los clústeres i y j, y d es una medida de distancia entre los centroides.

La "ley del codo" es una técnica ampliamente utilizada en la determinación del número óptimo de clústeres (k) en el algoritmo K-means. Su aplicación

implica el análisis de la inercia, que es la suma de los cuadrados dentro de los clústeres, en relación con el número de clústeres utilizados en el proceso de agrupamiento. Esta técnica es fundamental para determinar un valor adecuado de k que permita una segmentación efectiva y representativa de los datos. Para aplicar la "ley del codo", se ejecuta el algoritmo K-means con diferentes valores de k, generando un conjunto de modelos con diferentes niveles de granularidad en la agrupación de datos. Luego, se calcula la inercia de cada modelo, que representa la suma de las distancias al cuadrado de cada punto a su centroide correspondiente. Estos valores de inercia se grafican en función del número de clústeres, y se busca visualmente el punto donde se produce un "codo" o cambio significativo en la pendiente de la curva. Este punto sugiere un número óptimo de clústeres donde la disminución en la inercia se vuelve significativamente más lenta al agregar clústeres adicionales. En consecuencia, la "ley del codo" proporciona una guía intuitiva para seleccionar el número apropiado de clústeres que mejor represente la estructura subyacente de los datos, lo que permite una segmentación precisa y significativa en el análisis de datos mediante el algoritmo K-means.

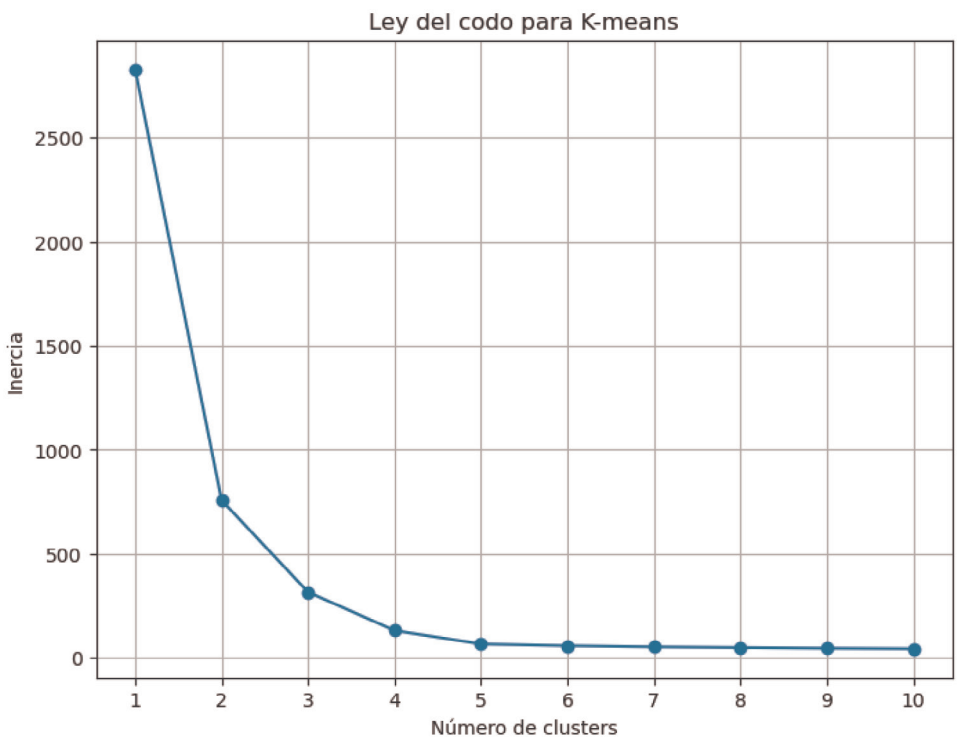

Figura 4.8. Curva obtenida mediante la ley del codo.

Código Python

Para graficar la ley del codo se puede hacer de la siguiente manera:

```
wcss = []
for i in range(1, 11):
kmeans = KMeans(n_clústeres=i, init='k-means++', max_iter=300, n_
init=10, random_state=0)
kmeans.fit(X), wcss.append(kmeans.inertia_)
```

Graficar la suma de los cuadrados dentro del clúster (WCSS) en función del número de clústeres

```
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Método del Codo')
plt.xlabel('Número de clústeres')
plt.ylabel('WCSS')
plt.show()
```

4.3. APRENDIZAJE POR REFUERZO

El aprendizaje por refuerzo es un enfoque dinámico y adaptativo en el campo del aprendizaje automático, donde un agente interactúa con un entorno complejo y desconocido con el objetivo de aprender a tomar decisiones secuenciales que maximicen una recompensa acumulativa a lo largo del tiempo. A diferencia de otros enfoques de aprendizaje, como el supervisado o el no supervisado, el aprendizaje por refuerzo se centra en la toma de decisiones autónoma y adaptativa basada en la experiencia.

En este proceso, el agente percibe el entorno a través de sus sensores, lo que le proporciona información sobre el estado actual del entorno. Luego, el agente selecciona una acción basada en su política de toma de decisiones, que puede ser determinista o estocástica, lo que determina cómo elige las acciones en función del estado actual. Después de tomar la acción, el agente recibe una señal de refuerzo del entorno, que puede ser una recompensa positiva, una penalización negativa o simplemente información sobre el éxito de la acción.

Una vez que el agente ha interactuado con el entorno y ha recibido retroalimentación, ajusta su política de toma de decisiones para mejorar su desempeño en futuras interacciones. Este proceso de aprendizaje se basa en la idea de que el agente puede aprender de sus experiencias pasadas y utilizar esta información para tomar decisiones más efectivas en el futuro.

El algoritmo Q-Learning es uno de los enfoques más utilizados en el aprendizaje por refuerzo para que un agente aprenda a tomar decisiones óptimas en un entorno desconocido. Se basa en la idea de aprender una función de valor de acción óptima, que asigna un valor a cada par estado-acción y permite al agente determinar la mejor acción a tomar en cada estado para maximizar la recompensa acumulada a largo plazo.

La ecuación de Bellman es una ecuación fundamental en el campo del aprendizaje por refuerzo. Proporciona una forma de descomponer el valor de un estado en función de las recompensas esperadas futuras y los valores de los estados alcanzables desde ese estado. Esta ecuación es crucial en algoritmos de aprendizaje por refuerzo como Q-Learning y SARSA para actualizar los valores de los estados y las acciones. La forma más común de la ecuación de Bellman para la función de valor de acción (Q) es la siguiente:

$$Q(s,a) = \left[(1-\alpha)\cdot Q(s,a) + \alpha\cdot[r + \gamma\cdot \max \quad Q(s',a')]\right]$$

Donde Q(s,a) es el valor de la función de valor de acción para el estado s y la acción a, r es la recompensa que se recibe al pasar de estado, γ el el factor de descuento, max Q(s´, a´) es el valor de la función de valor en el estado s´ o la recompensa máxima que el agente puede obtener. Esta ecuación establece que el valor de la función de valor de acción para un estado y una acción específicos es igual a la recompensa inmediata más la recompensa esperada futura, descontada por el factor de descuento γ, tomando la mejor acción posible en el próximo estado.

El proceso de Q-Learning es un algoritmo fundamental en el campo del aprendizaje por refuerzo que permite a un agente aprender a tomar decisiones óptimas en un entorno desconocido. Se puede desglosar en varios pasos clave:

1. **Inicialización de la tabla Q.** Se comienza inicializando una tabla Q, que es una estructura de datos que almacena los valores de la función de valor de acción para cada par estado-acción. Estos valores representan la utilidad esperada de elegir una acción específica en un estado dado.
2. **Selección de acciones basadas en políticas de exploración y explotación.** En cada paso, el agente selecciona una acción para tomar en el estado actual. Esto se puede hacer utilizando una política que equilibre la exploración (intentar nuevas acciones) y la explotación (elegir la mejor acción conocida). Ejemplos comunes de políticas son ε-greedy o softmax. La política ε-greedy es una estrategia comúnmente utilizada en el aprendizaje por refuerzo para equilibrar la exploración y la explotación. En esta política, el agente elige la acción óptima (aquella con el valor Q más alto) con una probabilidad 1−ε, y elige una acción aleatoria (exploración) con una probabilidad ε. Es decir, con probabilidad 1−ε, el agente sigue una estrategia puramente explicativa, mientras que con probabili-

dad ε, elige explorar acciones aleatorias. Este enfoque permite al agente explorar nuevas acciones en el entorno mientras sigue aprovechando las acciones que ha aprendido que son las mejores hasta el momento. La política softmax es otra estrategia utilizada para seleccionar acciones en el contexto del aprendizaje por refuerzo. En este enfoque, las probabilidades de selección de cada acción se calculan utilizando la función softmax, que asigna una probabilidad a cada acción proporcional a su valor Q

3. **Interacción con el entorno.** Una vez seleccionada una acción, el agente interactúa con el entorno ejecutando esa acción y observando la recompensa resultante y el siguiente estado.
4. **Actualización de la tabla Q utilizando la ecuación de Bellman.** Después de cada interacción con el entorno, el agente actualiza los valores en la tabla Q utilizando la ecuación de Bellman. Esta ecuación establece que el valor de Q para un par estado-acción debe ser igual a la recompensa inmediata más la recompensa esperada futura, ponderada por un factor de descuento y la probabilidad de transición a los siguientes estados. La idea es ajustar gradualmente los valores de Q para que se acerquen cada vez más a los valores óptimos.
5. **Repetición del proceso hasta la convergencia de la tabla Q.** El agente repite los pasos anteriores, tomando acciones, interactuando con el entorno, y actualizando la tabla Q, hasta que los valores de Q converjan o se estabilicen.

Una vez que la tabla Q ha convergido, el agente puede utilizarla para tomar decisiones óptimas en nuevos estados. Simplemente selecciona la acción con el valor Q más alto en el estado actual según lo indicado por la tabla Q.

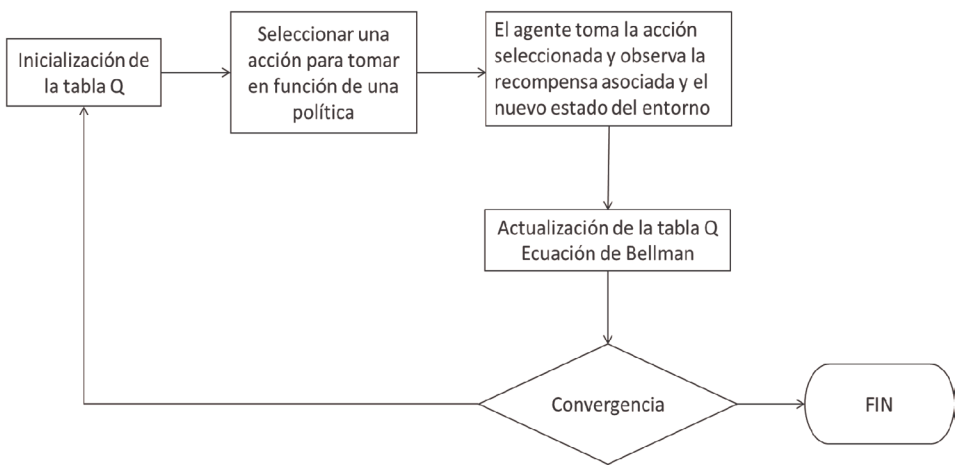

Figura 4.9. Algoritmo Q-Learning.

Para validar el rendimiento del algoritmo Q-Learning, es común realizar experimentos en un entorno simulado o en un entorno real si es posible. Se pueden utilizar métricas como la recompensa acumulada promedio por episodio o el porcentaje de episodios en los que el agente alcanza el objetivo deseado. Además, es importante ajustar los parámetros del algoritmo, como la tasa de aprendizaje (α), el factor de descuento (γ) y la probabilidad de exploración (ε), para obtener un rendimiento óptimo en el entorno específico. Mediante la validación y la optimización de estos parámetros, se puede mejorar el rendimiento y la eficacia del algoritmo Q-Learning en la tarea dada.

La tabla Q generada por el algoritmo Q-Learning representa la función de valor de acción para cada par estado-acción en el entorno del laberinto. Cada fila de la tabla corresponde a un estado del laberinto, mientras que cada columna representa una acción posible que el agente puede tomar desde ese estado. Los valores en la tabla Q indican la utilidad esperada de tomar una acción específica en un estado dado. Durante el proceso de aprendizaje, el agente actualiza gradualmente estos valores en función de las recompensas recibidas y las estimaciones previas de la función de valor de acción. Los valores más altos en cada fila representan las acciones consideradas óptimas por el agente, lo que indica las decisiones preferidas para maximizar la recompensa acumulada a largo plazo. En la fase de explotación, el agente utiliza esta información para tomar decisiones informadas y alcanzar su objetivo en el entorno del laberinto.

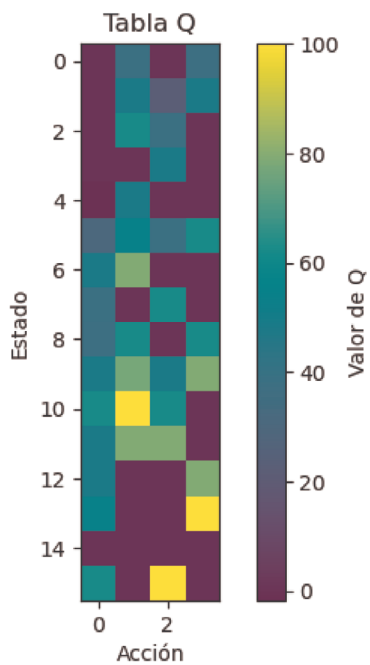

Figura 4.10. Tabla de salida del algoritmo Q-Learning.