Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

MÁSTER UNIVERSITARIO EN

INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

DISEÑO DE UN SISTEMA PARA LA MEJORA DE LA QOS DE UNA RED WI-FI MEDIANTE APRENDIZAJE AUTOMÁTICO

Estudiante *Aguado Marín, Alejandro*
Directora *Ibarrola Armendariz, Eva*
Departamento
Curso académico *2021-2022*

Bilbao, 18 de septiembre de 2022

Índice

Resumen trilingüe

Resumen

Las telecomunicaciones son cada vez más de banda ancha. En enero de este año, se alcanzaron 4950 millones de usuarios de internet en todo el mundo, alrededor del 62,5 % de la población mundial. Asimismo, el consumo de Internet diario fue de casi 7 horas el año pasado gracias a la banda ancha. Este tipo de red proporciona tasas de bits de acceso en sentido descendente y ascendente que soportan todos los tipos de servicios disponibles originando un Internet de buena calidad. No obstante, la aparición de estos nuevos servicios hace que la medición de la calidad del servicio QoS (*Quality of Service*) sea más ardua porque la complejidad de la red aumenta y sobre todo en las redes inalámbricas debido a que sufren mayor número de interferencias.

La información obtenida del tráfico de la red es demasiado extensa para que un ser humano sea capaz de obtener las relaciones entre los diferentes factores y establecer cuáles de ellos son los que más ayudan a estimar la QoS. La solución pasa por el uso del aprendizaje automático, conocido como *machine learning,* que es una rama de la inteligencia artificial formada por diversos algoritmos que están diseñados para establecer patrones entre diversos conjuntos de datos, *datasets*. Así pues, el objetivo de este trabajo consiste en el diseño de un sistema para la mejora de la QoS en redes Wi-Fi.

Palabras clave: aprendizaje automático, QoS, cluster, grado de influencia, recursos

Laburpena

Telekomunikazioak gero eta gehiago dira banda zabalekoak. Aurtengo urtarrilean, 4950 milioi interneteko erabiltzaile izan ziren mundu osoan, munduko biztanleriaren % 62,5 inguru. Era berean, Interneten eguneroko kontsumoa ia 7 ordukoa izan zen iaz banda zabalari esker. Sare mota honek sarbide-biten tasak eskaintzen ditu, beheranzko eta goranzko noranzkoan, eskuragarri dauden zerbitzu mota guztiei eusten dietenak, kalitate oneko Internet sortuz. Hala ere, zerbitzu berri horien agerpenaren ondorioz, QoS (*Quality of Service*) zerbitzuaren kalitatearen neurketa nekezagoa da, sarearen konplexutasuna handitu egiten baita, eta batez ere hari gabeko sareetan, interferentzia gehiago izaten baitituzte.

Sareko trafikotik lortutako informazioa zabalegia da gizaki bat gai izan dadin faktoreen arteko harremanak lortzeko eta faktore horietako zeinek laguntzen duen gehien QoS kalkulatzen. Soluzioa ikasketa automatikoaren erabilera da, machine learning bezala ezagutzen dena, adimen artifizialaren adar bat da, hainbat algoritmok osatzen dutena, datu multzoen artean (*datasets*) patroiak ezartzeko diseinatuta daudenak. Horrela, bada, lan honen helburua Wi-Fi sareetan QoS hobetzeko sistema bat diseinatzea da.

Gako-hitzak: ikaskuntza automatikoa, QoS, klusterra, eragin-maila, baliabideak

Abstract

Telecommunications are increasingly broadband. In January this year, the number of Internet users reached 4.49 billion worldwide, about 62.5% of the world's population. Also, daily Internet consumption was almost 7 hours last year thanks to broadband. This type of network provides downstream and upstream access bit rates that support all types of available services resulting in a good quality Internet. However, the emergence of these new services makes the measurement of QoS (Quality of Service) more difficult because the complexity of the network increases, especially in wireless networks due to the higher number of interferences.

The information obtained from network traffic is too extensive for a human being to be able to obtain the relationships between the different factors and establish which of them are the most helpful in estimating QoS. The solution involves the use of machine learning, known as machine learning, which is a branch of artificial intelligence formed by various algorithms that are designed to establish patterns between different datasets. Thus, the objective of this work is the design of a system for the improvement of QoS in Wi-Fi networks.

Keywords: machine learning, QoS, cluster, degree of influence, resources.

Listado de acrónimos y siglas

ANN: Artificial Neural Networks
AP: Access Point
API: Application Programming interface
BSS: Basic Service Set
CNN: Convolution Neural Networks
CNRI: Communications Network Research Institute
CSV: Comma-Separated Values
DBN: Deep Belief Network
DBSCAN: Density-based Spatial Clustering of Applications with Noise
EDCA: Enhanced Distributed Channel Access
HCA: Hierarchical Cluster Analysis
HCCA: HFC Controlled Channel Access
HFC: Hybrid Coordination Function
HTML: HyperText Markup Language
HTTP: Hypertext Transfer Protocol
IEEE: Institute of Electrical and Electronics Engineers
IoT: Internet of things
IP: Internet Protocol
ITU-T: International Telecommunication Union - Telecommunication Standardization Sector
JSON: JavaScript Object Notation
K-NN: K-Nearest Neihgbors
KPI: Key Perfomance Indicator
LAN: Local Area Network
LDA: Linear Discriminant Analysis
MAC: Media Access Control
PCA: Principal Component Analysis
QoE: Quality of Experience
QoS: Quality of Service
RBM: Restricted Boltzmann Machines
REST: Representational State Transfer
RNN: Recurrent Neural Networks
SNMP: Simple Network Management Protocol
SQL: Structured Query Language
SSID: Service Set Identifier
SVM: Support Vector Machines
SVR: Support Vector Regression
TUD: Technological University Dublin
UPV/EHU: Universidad del País Vasco/Euskal Herriko Unibertsitatea
WECA: Wireless Ethernet Compatibility Alliance
WLAN: Wireless Local Area Network
WWW: World Wide Web
XML: eXtensible Markup Language

Listado de figuras

Listado de tablas

1 Introducción

Hasta principios de los años 60, las telecomunicaciones estaban basadas en la conmutación de circuitos. Su característica principal radica en que los recursos necesarios a lo largo de la ruta, buffers y velocidad de transmisión del enlace, están reservados durante la sesión de la comunicación entre los hosts. En la siguiente figura se observa un ejemplo de este tipo de red, donde la conexión extremo a extremo hace uso del segundo circuito en el primer enlace y el cuarto circuito en el segundo. Dado que cada enlace tiene cuatro circuitos, dicha conexión obtiene un cuarto de la capacidad total de transmisión del enlace mientras dure la conexión.

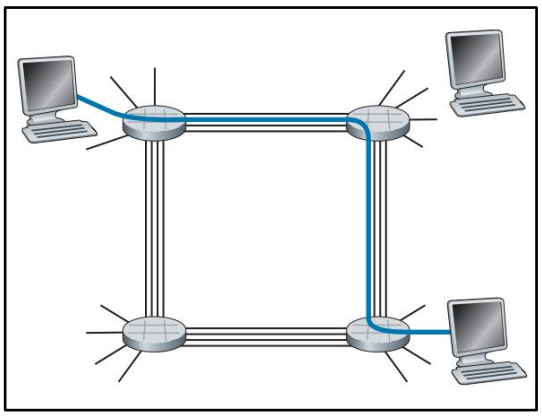

Figura 1: Una red simple de conmutación de circuitos, compuesta por cuatro conmutadores y cuatro enlaces [1]

La aparición de los primeros ordenadores en esta década hizo que se considerara que el uso de este tipo de conmutación no era adecuado dado que el tráfico de este tipo de dispositivos es a ráfagas, es decir, compuesto por periodos de actividad e inactividad. Así pues, se comenzó el desarrollo de la conmutación de paquetes donde se asigna el uso del enlace bajo demanda mediante uso de colas. La sencillez, la eficiencia y menor coste de esta tecnología originó Internet.

Tras la creación de la aplicación WWW (*World Wide Web*) en los 90, surgieron nuevos tipos servicios en Internet para todos los usuarios como la navegación web. En consecuencia, se comenzó a desarrollar el término de calidad de servicio QoS (*Quality of Service*) que mide el nivel del servicio.

Por último, ya en este siglo, el acceso a banda ancha en las viviendas, el abaratamiento de los equipos y el uso de redes inalámbricas hizo que Internet despegara por completo. Asimismo, aparecieron nuevos servicios basados en multimedia por lo que la QoS adquiere más importancia.

2 Contexto

En el año 1997, sólo el 1% de la población mundial estaba abonada a la telefonía móvil y el 11% a la fija. En enero de este año, se alcanzaron 4950 millones de usuarios de internet en todo el mundo, alrededor del 62,5 % de la población mundial. Asimismo, el consumo de Internet diario fue de casi 7 horas el año pasado gracias a la banda ancha.

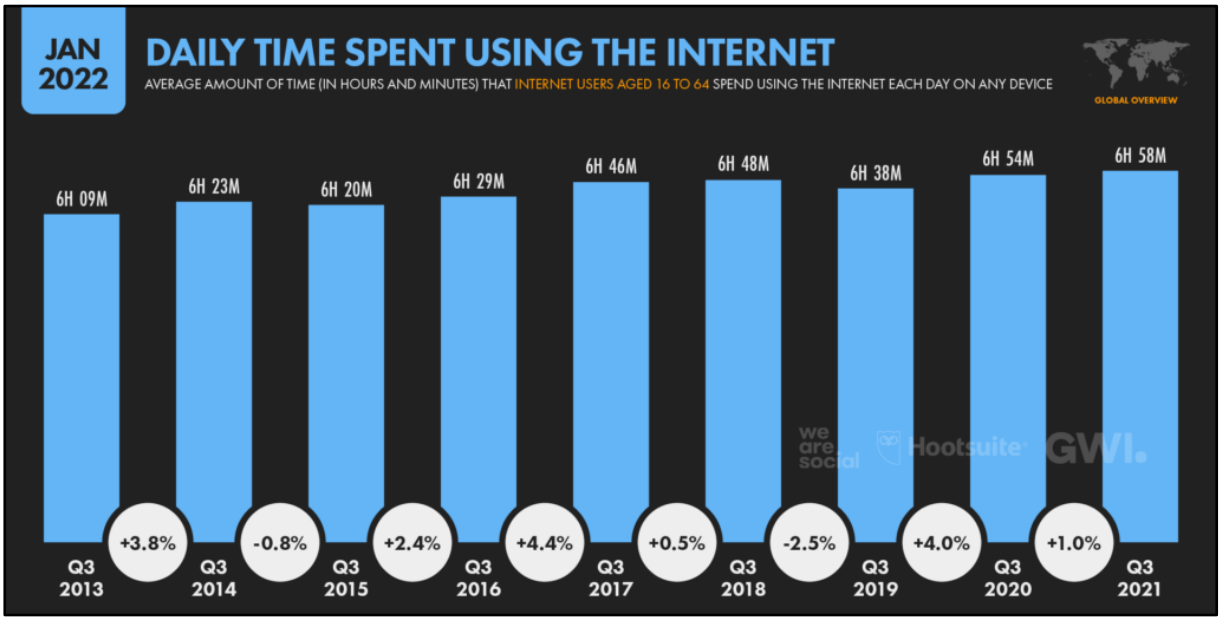

Figura 2: Evolución del consumo diario Internet en el intervalo 2013-2021 [2]

El uso de este tipo de red proporciona tasas de bits de acceso en sentido descendente y ascendente que soportan todos los tipos de servicios disponibles originando un Internet de buena calidad. Hace dos décadas, se ofrecía principalmente en cientos de kbps, hoy proporciona decenas de Mbps lo que permite, por ejemplo, ofrecer vídeo de alta definición.

No obstante, la aparición de estos nuevos servicios hace que la medición de la QoS sea cada vez más compleja. La calidad puede verse afectada por muchos factores a nivel de dispositivo, infraestructura de red, servicio y aplicaciones.

Además, la información obtenida es demasiado extensa para que un ser humano sea capaz de obtener las relaciones entre los diferentes factores y establecer cuáles de ellos son los que más ayudan a estimar la QoS. La solución pasa por el uso del aprendizaje automático, comúnmente conocido como *machine learning,* que es una rama de la inteligencia artificial formada por diversos algoritmos que están diseñados para establecer patrones entre diversos conjuntos de datos, *datasets*. Dichos patrones se consiguen gracias a que la máquina realiza un proceso de aprendizaje, es decir, va aprendiendo hasta llegar a la conclusión de que los patrones obtenidos son la mejor solución para el problema propuesto. En este contexto se enmarca este trabajo, que pretende mejorar la QoS haciendo uso de este tipo de herramienta.

3 Objetivos y alcance

El objetivo principal de este proyecto consiste en el diseño de un sistema para la mejora de la QoS de redes Wi-Fi mediante aprendizaje automático. Dicho sistema se basa en un *dashboard* para que el usuario pueda manipular la consulta de los datos fácilmente y que los resultados obtenidos se representen con unas imágenes claras y legibles.

Existen dos razones por la que se ha optado por este tipo de redes. La primera razón consiste en que la red Wi-Fi es la red inalámbrica LAN (*Local Area Network*) más desplegada y usada en el mundo y, como cualquier red inalámbrica, es más sensible a la QoS que las redes cableadas por los siguientes motivos:

- **Intensidad decreciente de la señal**: la radiación electromagnética se atenúa a medida que incrementa la distancia entre el emisor y receptor ya sea atravesando la materia o incluso en el espacio vacío (pérdida de propagación, *path loss*).
- **Interferencias de otros orígenes**: los dispositivos que transmiten en la misma banda de frecuencia interfieren entre sí. Asimismo, el ruido electromagnético presente en el entorno (motor cercano, microondas, etc.) también puede provocar interferencias.
- **Propagación multicamino** (*multipath*): fenómeno que origina que la señal recibida en el receptor sea menos limpia. Ocurre cuando partes de la onda electromagnética se reflejan en los objetos y en el suelo, tomando caminos de diferentes longitudes entre el emisor y receptor.

Por otra parte, la segunda razón se basa en que se ha conseguido obtener información perteneciente a 2 redes Wi-Fi. Una de ellas es la de la Universidad del País Vasco/Euskal Herriko Unibersitatea (UPV/EHU) cuyo tráfico de red se extrae mediante una API (*Application Programming interface*) propietaria de Cisco [3]. La otra es la red del Instituto de Tecnología de Dublín *TUD (Technological University Dublin*) [4] cuya información se extrae mediante sondas Wi-Fi de la empresa OptiWi-fi [5].

El proyecto divide en dos módulos, el primero consiste en obtener las estadísticas del tráfico, es decir, porcentaje de muestras en función del día de la semana, hora, etc.

Por último, en el segundo módulo se va a hacer uso de algoritmos de aprendizaje automático. En primer lugar, se obtienen agrupaciones de las muestras, también llamada *clustering*. Ya con los clusters definidos se comprueba la proporcionalidad de los escenarios. En otras palabras, si las muestras agrupadas en un cluster mayormente pertenecen a un escenario único. Por último, mediante la reducción de la dimensión del conjunto de datos se obtiene el grado de influencia de cada parámetro a la hora de establecer los clusters.

4 Beneficios

En este apartado se establecen los diferentes beneficios que supondría una implementación correcta del proyecto. Dichos beneficios son los siguientes:

4.1 Económicos

Al obtener las estadísticas del tráfico se puede establecer que escenario necesita más recursos, es decir, mantenimiento, luz, monitorización, etc. Por otra parte, al establecer que parámetros son los que más influyen en una buena QoS permite a empresas aumentar sus ganancias debido a las recomendaciones de los clientes satisfechos.

4.2 Sociales

Una buena QoS permite al usuario utilizar el servicio con garantías de funcionamiento aumentando su satisfacción personal ya que no pierde el tiempo en esperar a que el servicio vuelva a funcionar correctamente.

4.3 Tecnológicos

El saber qué parámetros son los más influyen en la QoS origina que los trabajadores del sector de las telecomunicaciones tengan un mejor conocimiento de éstos y puedan resolver fallos de servicio de una forma más rápida. Asimismo, se crea una base para las nuevas investigaciones partiendo de las conclusiones obtenidas. Esto puede originar la creación de nuevos métodos de predicción o nuevos estándares relacionados con las redes Wi-Fi.

5 Estado del arte

5.1 Aprendizaje automático

El aprendizaje automático, *machine learning*, es la ciencia que consiste en programar a los ordenadores para que aprendan de los datos. Una explicación más técnica fue la manifestada en 1997 por el profesor especializado en ciencia computacional Tom Mitchell, que definió *machine learning* como "*Un programa de un ordenador aprende de la experiencia E con respecto a alguna tarea T y alguna medida del rendimiento P, si su rendimiento en T, medido por P, mejora con la experiencia E"*. Un ejemplo para plasmar esta definición es la creación de un filtro spam mediante un algoritmo de aprendizaje automático. El filtro aprende automáticamente qué palabras y frases son buenos predictores de spam (Tarea T) al detectar inusualmente frecuentes patrones de palabras en los ejemplos de spam (Medida P) en comparación con los ejemplos de correos deseados (Experiencia E). En caso de que se hubiera escrito el filtro mediante programación tradicional hubiera sido más tedioso ya que el desarrollador tendría que establecer de forma manual todas las combinaciones posibles para que un correo se declare como spam y además el código se tendría que actualizar cada vez que se descubran nuevos patrones.

Un término que va ligado con machine learning es la minería de datos, *big data*. Consiste en obtener una mejor compresión del problema a evaluar cuando los algoritmos revelan correlaciones insospechadas o nuevas tendencias entre grandes conjuntos de datos a analizar.

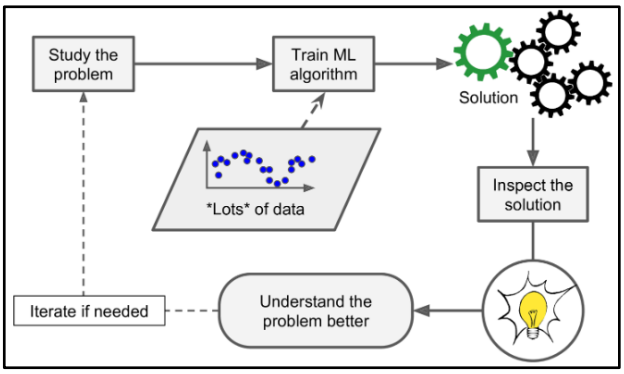

Figura 3: Minería de datos [6]

Así pues, las características principales se pueden declarar en los siguientes puntos:

- Código más legible y óptimo.
- Solucionar problemas que no son correctamente resueltos con métodos tradicionales.
- Adaptación automática a entornos fluctuantes.
- Obtener información sobre problemas complejos y grandes cantidades de datos.

Existen un gran conjunto de diferentes sistemas de aprendizaje automático, pero se categorizan en función del criterio a utilizar permitiendo la combinación entre sistemas con distintos criterios.

5.1.1 Aprendizaje en función de la supervisión humana

5.1.1.1 Aprendizaje supervisado

En este conjunto de sistemas los datos de entrenamiento están etiquetado (*labels*), es decir, se conoce la salida para la entrada. En función del valor de las salidas se distinguen dos formas:

- **Clasificación:** la salida es una variable discreta (categoría). Un ejemplo de este tipo sería el filtro de spam ya que la salida puede ser spam o correo válido. Algunos de los algoritmos de este tipo son los siguientes: clasificación con máquinas de soporte vectorial SVM (*Support Vector Machines*), con árboles de decisión, con bosques aleatorios y los k vecinos más cercanos k-NN (*k-Nearest Neighbors*).

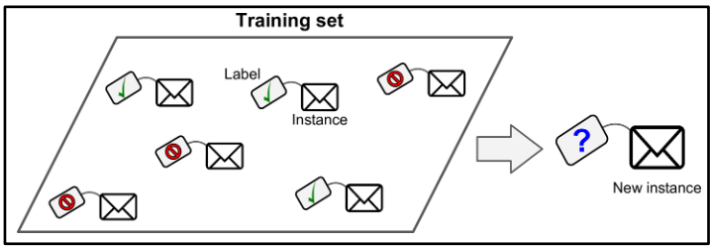

Figura 4: Clasificación en algoritmos supervisados [6]

- **Regresión:** estudia las relaciones entre una variable dependiente y una o más variables independientes llamadas predictores. La salida es un valor continuo, Ej. el cálculo del precio de un coche. Algunos de los algoritmos de este tipo son los siguientes: regresión lineal, polinómica, de soporte vectorial SVR (*Support Vector Regression*), con árboles de decisión y con bosques aleatorios.

Figura 5: Regresión en algoritmos supervisados [6]

5.1.1.2 Aprendizaje no supervisado

A diferencia del tipo anterior, en éste los datos de entrenamiento no se encuentran etiquetados por lo que el algoritmo aprende sin un conocimiento previo. Los algoritmos de este tipo de aprendizaje se dividen en los siguientes grupos:

- **Agrupación:** también conocido como *clustering*, su objetivo es crear una segmentación del conjunto de los datos en grupos más o menos homogéneos llamados clusters. Sin embargo, no existe una definición universal de lo que es un cluster. Esto es debido a que se obtener mediante la búsqueda de datos centrados en torno a un punto concreto, llamado centroide, a través de regiones continuas de datos densamente empaquetadas, haciendo uso de una jerarquía obteniendo cluster de clusters, etc. Algunos ejemplos de este tipo de grupo son:
  - K-Means
  - Agrupamiento espacial basado en densidad de aplicaciones con ruido DBSCAN (*Density-based spatial clustering of applications with noise*).
  - Análisis de agrupamiento jerárquico HCA (Hierarchical Cluster Analysis).

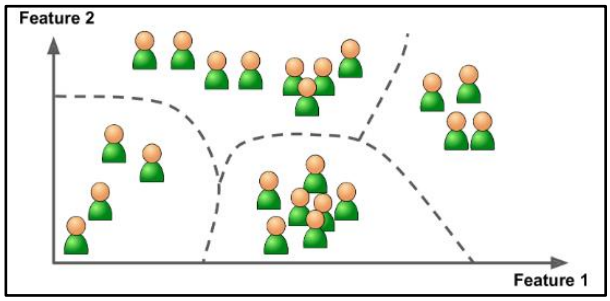

Figura 6: Clustering [6]

- **Detección de anomalías y novedades:** en los algoritmos de detección de anomalías (Ej. bosque de aislamiento) la mayoría de los datos son normales durante el entrenamiento, de modo que va aprendiendo a reconocerlos y cuando detecta un nuevo dato decide si es una anomalía o no. Po otra parte, la característica de los algoritmos de detección de la novedad (Ej. SVM de una clase) consiste en detectar nuevos datos que parezcan diferentes de todos los del conjunto de entrenamiento muy "limpio".

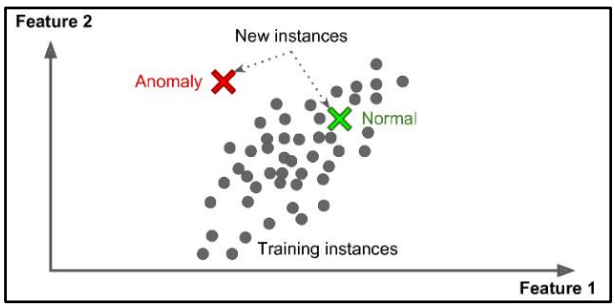

Figura 7: Detección de anomalías [6]

- **Visualización y reducción de la dimensión:** si cada dato perteneciente al conjunto de entrenamiento tiene un número amplio de características se origina una dificultad en la visualización de éste con el correspondiente coste computacional. Para ello, se hace uso de este tipo de algoritmos que se encargan de reducir la dimensión, seleccionando las características más importantes con la correspondiente eliminación de las que no aportan gran información. Asimismo, esta reducción disminuye la carga computacional. Algunos ejemplos son el análisis de componentes principales PCA (*Principal Component Analysis*) (ACP) y el análisis discriminante lineal LDA (*Linear Discriminant Analysis*).

- **Reglas de asociación:** el objetivo de estos algoritmos es profundizar en grandes cantidades de datos y descubrir relaciones interesantes entre características. Algunos de ellos son apriori y eclat.

5.1.1.3 Aprendizaje semisupervisado

Cuanto más grande el conjunto de datos a analizar, el etiquetado de los datos será más costoso y largo pudiendo originar que los datos se queden sin etiquetar o parcialmente etiquetados. Los algoritmos de aprendizaje semisupervisado son los idóneos para lidiar con esta situación. En la siguiente figura se observa un ejemplo donde los datos no etiquetados (círculos) ayudan a clasificar un nuevo dato (la cruz) en la clase de los triángulos, aunque esté más cerca de los cuadrados.

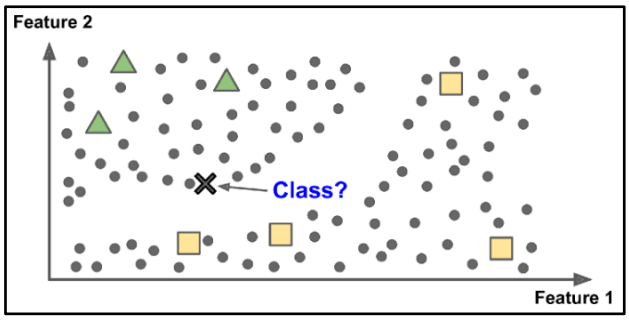

Figura 8: Aprendizaje semisupervisado [6]

La mayoría de los algoritmos de aprendizaje semisupervisado son combinaciones de no supervisados y supervisados. Ej. las redes de creencias profundas DBNs (*Deep Belief Networks*) se basan en componentes no supervisados denominados máquinas de Boltzmann restringidas RBM (*Restricted Boltzmann Machines*) apiladas unas sobre otras. Las RBM se entrenan secuencialmente de forma no supervisada y, a continuación, el sistema se ajusta mediante técnicas de aprendizaje supervisado.

5.1.1.4 Aprendizaje reforzado

En este tipo de aprendizaje, el sistema, llamado agente, observa el entorno, selecciona y realiza acciones, y obtiene recompensas a cambio ya sean positivas o negativas (penalizaciones). A continuación, debe aprender por sí mismo cuál es la mejor estrategia, llamada política, para obtener la mayor recompensa a lo largo del tiempo. Muchos robots aplican algoritmos de este aprendizaje para aprender a caminar.

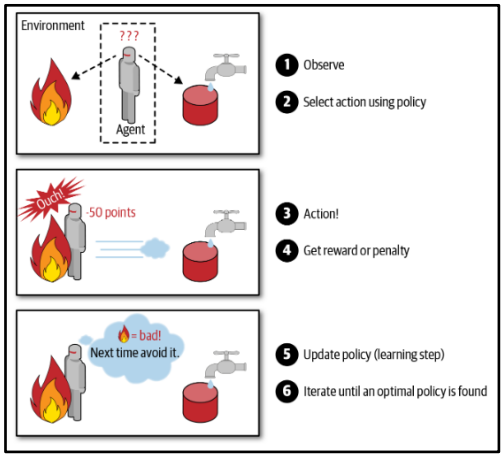

Figura 9: Aprendizaje reforzado [6]

5.1.2 Aprendizaje en función de la incrementación de los datos

5.1.2.1 Aprendizaje por lotes

La característica principal de este aprendizaje consiste en que el sistema es incapaz de aprender de forma incremental por lo que debe ser entrenado utilizando todos los datos disponibles. Sin embargo, esto suele requerir mucho tiempo y recursos informáticos, por lo que suele hacerse fuera de línea. Primero se entrena el sistema, y luego se lanza a la producción y se ejecuta sin aprender más; sólo aplica lo que ha aprendido. Esto se llama aprendizaje offline. Si se quiere que conozca nuevos datos, hay que entrenar una nueva versión del sistema desde cero con el nuevo conjunto de datos completo y luego detener el sistema antiguo y sustituirlo por el nuevo.

No obstante, todo el proceso de entrenamiento, evaluación y puesta en marcha de un sistema de aprendizaje automático puede automatizarse con bastante facilidad. Para ello, basta con actualizar los datos y entrenar una nueva versión del sistema desde cero tantas veces como sea necesario.

5.1.2.2 Aprendizaje en línea

El sistema se entrena de forma incremental, alimentándolo de forma secuencial, ya sea individualmente o en pequeños grupos denominados minilotes. Cada paso del aprendizaje es rápido y barato, por lo que el sistema puede aprender sobre la marcha mientras llegan los nuevos datos.

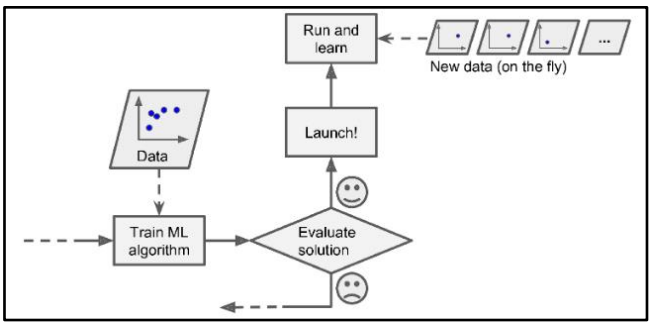

Figura 10: Aprendizaje en línea [6]

Este aprendizaje es ideal para los sistemas que reciben datos como un flujo continuo (Ej. precios de las acciones) y necesitan adaptarse a los cambios de forma rápida o autónoma. También es una buena opción si se tienen recursos informáticos limitados ya que una vez que este tipo de sistema ha aprendido sobre nuevas instancias de datos, ya no las necesita, por lo que puede descartarlas.

5.1.3 Aprendizaje en función del conocimiento de los datos conocidos

5.1.3.1 Aprendizaje basado en instancias

El sistema aprende los ejemplos de memoria y luego generaliza a nuevos casos utilizando una medida de similitud para compararlos con los ejemplos o subconjuntos aprendidos. En la siguiente figura, el nuevo dato será un triángulo porque la mayoría de los cercanos pertenece a esa clase.

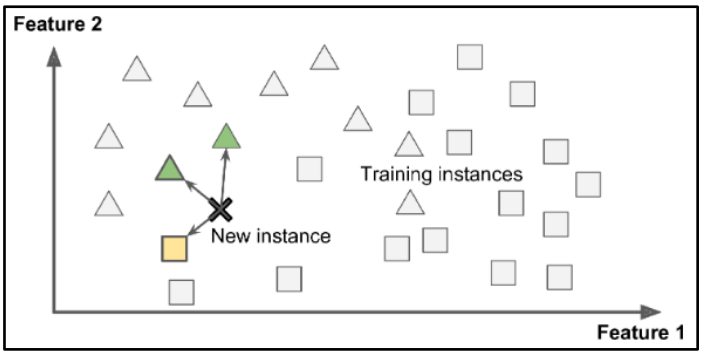

Figura 11: Aprendizaje basado en instancias [6]

5.1.3.2 Aprendizaje basado en modelos

El modelo está parametrizado con un cierto número de parámetros que no cambian según el tamaño de los datos de entrenamiento. El proceso consiste en lo siguiente:

1. Seleccionar los datos
2. Seleccionar el modelo
3. Entrenarlo mediante herramientas para minimizar la función de coste de las características.
4. Generalizar el modelo para las predicciones resultantes.

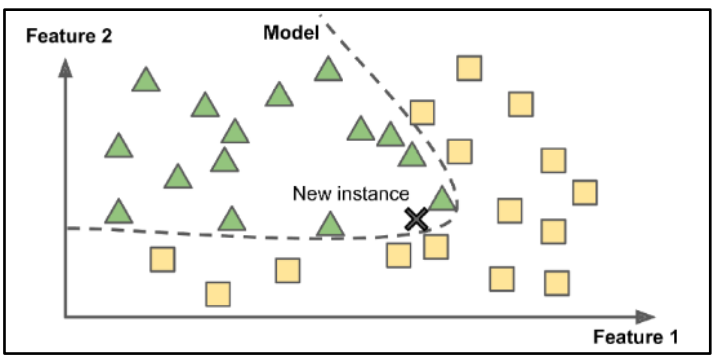

Figura 12: Aprendizaje basado en modelos [6]

5.1.4 Aprendizaje profundo

Es un subcampo específico del aprendizaje automático llamado también *deep learning*. Su estructura jerárquica es en capas y está basada en el funcionamiento de las redes neuronales humanas:

- **Capa de entrada**: compuesta por las neuronas que reciben los datos de entrada.
- **Capa oculta**: se encarga de hacer el procesamiento. A mayor capas mayor nivel de procesamiento.
- **Capa de salida**: establece decisiones en función de los resultados generados en la capa oculta. Un ejemplo es el decidir si una imagen es un perro o no.

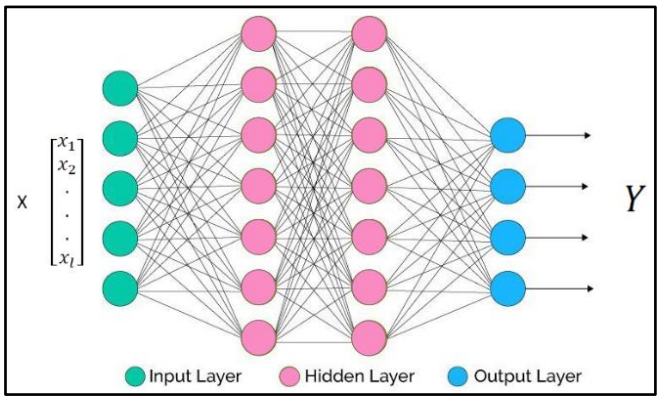

Figura 13: Red neuronal [7]

La diferencia principal con el aprendizaje automático es que la propia red neuronal se encarga de extraer las características, situación que no ocurre en el primero porque tiene que ser el usuario el que se encargue de indicarlas. Sin embargo, las operaciones de la red neuronal requieren una mayor capacidad computacional que los algoritmos de aprendizaje automático. Algunas de las redes neuronales más utilizadas son:

- **Redes neuronales artificiales ANN** (*Artificial Neural Networks*): redes neuronales simples. Se utilizan por ejemplo para problemas de clasificación o regresión.
- **Redes neuronales convolucionales CNN** (*Convolution Neural Networks*): diseñadas para datos matriciales en el que se hace uso de la convolución como herramienta de procesamiento. El uso general de estas redes está relacionado con el reconocimiento de imágenes.

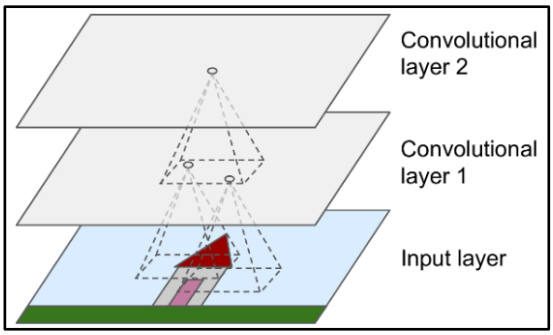

Figura 14: CNN [6]

- **Redes neuronales recurrentes RNN** (*Recurrent Neural Networks*): diseñadas para hacer uso de datos secuenciales, es decir, el nuevo dato tiene relación con el anterior. Algunos de los ejemplos de estas redes son la traducción simultánea o la predicción de acciones de bolsa.

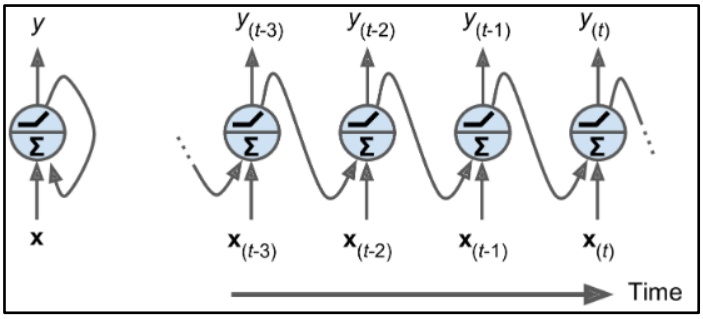

Figura 15: RNN [6]

5.2 Calidad de servicio

La ITU-T (International Telecommunication Union - Telecommunication Standardization Sector) Rec. E.800 define la calidad de servicio como la totalidad de las características de un servicio de telecomunicaciones de un servicio telecomunicaciones que influyen en su capacidad para satisfacer necesidades declaradas e implícitas del usuario.

Antiguamente, la QoS se abordaba desde la perspectiva de que el usuario final era una persona con capacidad para oír y ver y ser tolerante a cierta degradación de los servicios (Ej. una relación de pérdida de paquetes baja es aceptable para la voz mientras que el retardo de extremo a extremo debe ser inferior a 400 ms). No obstante, con la llegada de nuevos tipos de comunicaciones, los nuevos servicios pueden ser tratados de forma diferente según sean utilizados por máquinas o por humanos en uno o ambos extremos de una determinada sesión de comunicación o conexión.

La percepción del usuario final de un servicio de telecomunicaciones ya no solo no se limita a las características técnicas del servicio (rendimiento de la red y terminal) y los aspectos no técnicos (punto de venta, la atención al cliente, etc.) sino también a la calidad expectativa personal. Algunos de los factores que influyen en ella son las tendencias sociales, la publicidad, las tarifas y los costes.

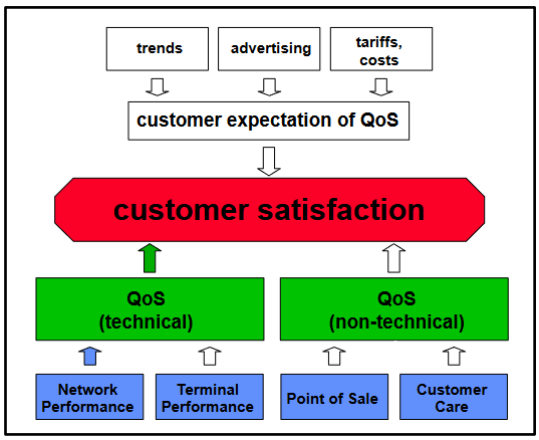

Figura 16: Punto de vista técnico y no técnico de la QoS y satisfacción del cliente [8]

Debido a esta situación surgió el término de calidad de experiencia QoE (*Quality of Experience*). Según la ITU-T Rec. P.10/G.100 se define como *el grado de agrado o molestia del usuario de una aplicación o servicio*. En la misma recomendación se definen estos términos:

1. **Factores que influyen en la QoE**: las expectativas del usuario con respecto a la aplicación o el servicio y su cumplimiento, sus antecedentes culturales y estado emocional, las cuestiones socioeconómicas y otros factores que se irán ampliando con nuevas investigaciones.

2. **Evaluación de la QoE**: proceso de medición o estimación de la QoE para un conjunto de usuarios de una aplicación o un servicio con un procedimiento específico, y teniendo en cuenta los factores influyentes. El resultado del proceso puede ser un valor escalar, una representación multidimensional de los resultados y/o descriptores verbales.

En la siguiente figura se observa de una forma detallada la relación entre el rendimiento total de la red, la QoS y la QoE. En primer lugar, el rendimiento de la red se aplica a la planificación, desarrollo, operaciones y mantenimiento del proveedor cuyos elementos son la red de acceso de transporte IP (*Internet Protocol*), la red central y el resto del camino extremo a extremo. La QoS aúna el rendimiento de la red junto al rendimiento del terminal del usuario cuyo conjunto es conocido como indicadores clave de rendimiento (KPI, *Key Perfomance Indicator*). Por último, la QoE tiene el alcance más amplio, ya que se ve afectada por la QoS, así como por las expectativas del usuario y el contexto.

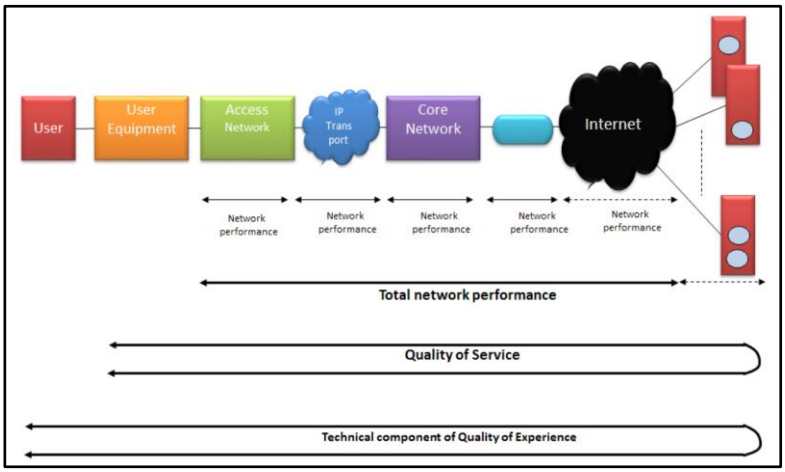

Figura 17: Relación entre el rendimiento de la red, QoS y QoE [8]

Por último, para los servicios de datos los KPI son los siguientes:

- **Ancho de banda**: número máximo de bits que puede transportar una vía de transmisión.
- **Retardo de propagación**: tiempo que requiere un paquete, en función de la longitud combinada de todas las rutas de transmisión y la velocidad de la luz a través de la ruta de transmisión.
- **Retardo de cola**: tiempo que un paquete espera antes de ser transmitido. Tanto el retardo medio como la variabilidad del retardo (*jitter*) son importantes debido que ambos establecen un intervalo de confianza para el tiempo en el que se puede esperar que un paquete llegue a su destino.
- **Pérdida de paquetes**: probabilidad de que un paquete no llegue nunca a su destino. Lo más frecuente es que los paquetes se pierden porque el número de paquetes en espera de transmisión es mayor que la capacidad de los *buffers*.

5.3 Redes Wi-Fi

En el año 1985 la Comisión Federal de Comunicaciones de Estados Unidos liberó las bandas del espectro radioeléctrico de 900 MHz, 2.4 GHz y 5.8 GHz para su uso sin licencia por cualquiera. Las empresas tecnológicas empezaron a construir redes y dispositivos inalámbricos para aprovechar el nuevo espectro radioeléctrico disponible, pero al no tener un estándar inalámbrico común los dispositivos de distintos fabricantes rara vez eran compatibles. En consecuencia, en 1997 el Instituto de Ingenieros Eléctricos y Electrónicos IEEE (*Institute of Electrical and Electronics Engineers*) aprobó el estándar IEEE 802.11. Dos años después, un grupo de grandes empresas formó la WECA (Wireless Ethernet Compatibility Alliance), una organización mundial sin ánimo de lucro creada para promover el nuevo estándar inalámbrico denominándolo Wi-Fi. En 2002 pasó a llamarse Wi-Fi Alliance.

El componente fundamental de la arquitectura 802.11 es el conjunto de servicios básicos BSS (*Basic Service Set*). Un BSS contiene una o más estaciones inalámbricas (*hosts*) y una estación base central, conocida como punto de acceso AP (*Access Point*). Cada host tiene que asociarse con un AP antes de poder enviar o recibir datos de la capa de red. Cuando un administrador de red instala un AP, le tiene que asignar un identificador de conjunto de servicios SSID (*Service Set Identifier*) y un número de canal para que se puede realizar tal asociación.

En la siguiente figura se observa un ejemplo de esta arquitectura donde para cada uno de los BSS hay un AP que se interconectan a un dispositivo de interconexión (switch, router, etc.) que a su vez conduce a Internet. En una red doméstica, el AP y router suelen estar integrados en uno solo. Por otra parte, se puede dar el caso de que la red se tenga que formar sin un control central, es decir, una red *ad hoc*. En este caso, la red se forma sobre la marcha por dispositivos móviles que se encuentran cerca unos de otros, que necesitan comunicarse y que no encuentran una infraestructura de red preexistente en su ubicación.

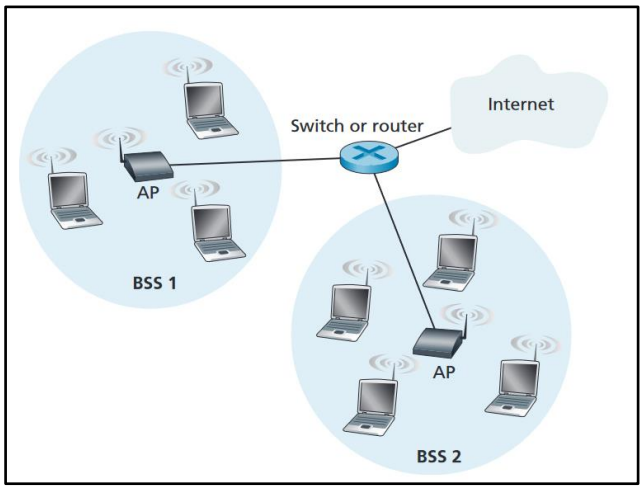

Figura 18: Arquitectura de una red LAN IEEE 802.11 [1]

El estándar 802.11 requiere que el AP envíe periódicamente tramas de baliza (*beacon frame*), cada una de las cuales incluye el SSID y la dirección MAC (*Media Access Control*) del AP. El host inalámbrico es consciente del envío de las tramas y escanea los canales para poder realizar la petición de conexión. A este proceso se le conoce como escaneo pasivo. El host está completamente anónimo porque no hay transmisión de datos al exterior.

Sin embargo, un host también puede realizar un escaneo activo, emitiendo una trama de sondeo que será recibida por todos los APs dentro de su rango. Los APs responden a la trama de solicitud de sondeo con una trama de respuesta de sondeo y el host puede entonces elegir el AP con el que con el que asociarse de entre los AP que respondan. Según el número de canales utilizados el consumo de energía del host puede ser alto.

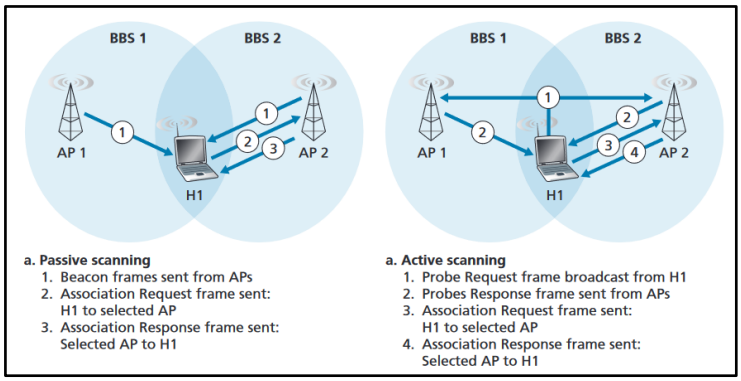

Figura 19: Escaneo activo y pasivo de los APs [1]

Debido a que el estándar se desarrolló a finales de los 90 no se tuvo en cuenta su uso para contenido multimedia por lo que no tenía soporte para la QoS. Sin embargo, en el año 2005 la versión del estándar 802.11e hizo que se pudiera implementar. La novedad principal radica en la adicción del esquema de control HFC (*Hybrid Coordination Function*). Se definen dos métodos de acceso al canal, priorizando aquellos datos que sean más sensibles:

- **EDCA** (*Enhanced Distributed Channel Access*): el acceso se basa en la variación de los temporizadores presentes en los controles estándar de las redes Wi-Fi.
- **HCCA** (*HFC Controlled Channel Access*): dota al AP de la capacidad de dirigir que tráfico debe de ir a los hosts. Ofrece mejores resultados, pero requiere una alta carga de procesamiento.

Por causa de esta gran carga, EDCA es el método más extendido donde todos los sistemas certificados con esta versión del estándar deben implementarlo obligatoriamente. El tráfico se clasifica en cuatro categorías, ordenadas de mayor a menor importancia:

- **Voz** (AC_VO): tráfico de voz.
- **Video** (AC_VI): tráfico de video.
- ***Best Effort*** (AC_BE): tráfico que se debe transmitir lo antes posible tras atender a los tráficos de las anteriores categorías. Ej. control remoto de un equipo.
- ***Background*** (AC_BK): tráfico que no necesita ningún tratamiento especial. Ej. email.

5.4 Cisco Prime Infrastructure API

API propietaria de Cisco que permite a los usuarios acceder y consumir datos capturados por sus equipos de forma sencilla, segura y escalable. Se basa en patrones REST (*Representational State Transfer*) con JSON (*JavaScript Object Notation*) como formato de datos, y soporta filtrado, paginación, ordenación y codificación de transferencia en trozos. Sus características principales son las siguientes:

- **Monitorización de alarmas y eventos:**
  - **Alarma**: es una representación del fallo o cambio de estado que se ha producido en el sistema gestionado; puede estar relacionada con el recurso o con el servicio y el cliente. Se asocian a un grupo de eventos recibidos de los recursos gestionados, normalmente con la misma fuente y categoría, que indican que ese fallo o evento se ha producido.
  - **Evento**: representa un registro normalizado de un suceso notificado por la red (o cualquier sistema capaz de notificar dicho suceso); podría ser un syslog, una SNMP (*Simple Network Management Protocol*) trap o cualquier otro tipo. El término normalizado denota un formato único abstraído sobre todos los tipos de eventos notificados, independientemente de la fuente y la estructura.
  - **Syslog**: es un método estándar para el registro de mensajes tales como eventos del sistema en una red IP. Tiene tres tipos: emergencia (0), alerta (1) y crítico (2).
- **Recogida de inventario de dispositivos:**
  - **Dispositivo**: proporciona diversa información, como el nombre, el tipo, la dirección IP, el tipo de software, la versión, y también la accesibilidad y el estado de gestión.
  - **Controlador WLAN** (Wireless Local Area Network).
  - **Puntos de acceso**: proporciona atributos del dispositivo como el tipo, la versión, el controlador de asociación, el número de clientes asociados, etc.
  - **Interfaces de radio** de los puntos de acceso.
- **Supervisión de clientes y su uso:**
  - **Clientes**: dirección MAC, dirección IP, nombre de usuario, estado, etc
  - **Sesiones de los clientes**: atributos relacionados con el dispositivo y la sesión, incluyendo la seguridad, el dispositivo conectado, el tiempo de la sesión, el tráfico, etc.

El conjunto de los datos se encuentra dividido en 59 entidades cuyas descripciones se encuentran detallas en el Anexo 1.

6 Análisis de alternativas

6.1 Lenguaje de programación

Los tres lenguajes más utilizados en el aprendizaje automático son los siguientes:

- **Matlab**: lenguaje utilizado principalmente por ingenieros y analistas de datos para cálculos numéricos. La ventaja principal consiste en que la visualización de los datos es amigable al usuario, esto es debido a que el usuario tiene un control excelso de cómo quiere que visualicen los datos incluyendo entornos multidimensionales. Además, el usuario no tiene por qué tener un conocimiento profundo en la estadística o en las matemáticas para comprender dicha visualización.
- **R**: está diseñado para ejecutar análisis estadísticos y gráficos de salida. Se centra en el lado analítico en lugar de la legibilidad de los datos. Dicha afirmación repercute en que este lenguaje es más recomendable para aquellos que estén más preocupados en la producción de modelos de datos en lugar de la codificación.
- **Python**: es un lenguaje de programación disponible al que pueden acceder y utilizar fácilmente los programadores más experimentados, pero también los estudiantes novatos. Debido a que es un lenguaje adaptable y bien desarrollado se puede usar para proyectos mayores y pequeños.

Aunque los 3 lenguajes tienen un gran número de herramientas disponibles se selecciona Python por ser el lenguaje más popular en Stack Overflow por lo que la solución a posibles fallos en el código es más rápida y fácil de obtener y porque ya se tiene conocimientos de éste por parte del autor.

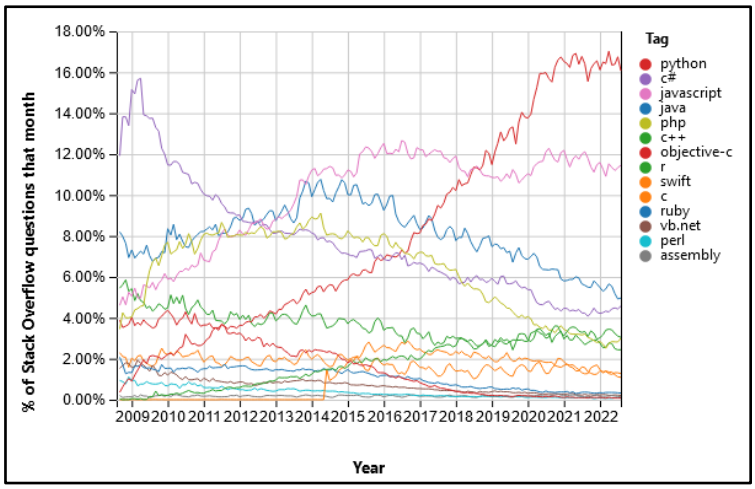

Figura 20: Porcentaje de preguntas anuales de cada lenguaje en Stack Overflow en el intervalo 2009-2022 [9]

6.2 Formato de almacenamiento de los datos

Los dos formatos más usados en el aprendizaje automático son CSV (*Comma-Separated Values*) y SQL (*Structured Query Language*). El primero de ellos es un archivo de texto que utiliza un delimitador (coma, tabulador, semicoma, etc.) para separar los valores. Un archivo CSV almacena datos tabulares (números y texto) en texto plano. Cada línea del archivo es un registro de datos y cada registro consta de uno o varios campos, separados por el delimitador. Destaca por ser muy legible para conjunto de datos pequeños y no requiere realizar configuraciones adicionales, es decir, basta con rellenar los datos en el fichero.

Por otra parte, se tiene el lenguaje SQL. Es el lenguaje estándar para los sistemas de gestión de bases de datos relacionales. Las sentencias SQL se utilizan para realizar tareas como la actualización de datos en una base de datos o la recuperación de datos de una base de datos. Algunos sistemas de gestión de bases de datos relacionales comunes que utilizan SQL son: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc. La complejidad de este lenguaje reside en la creación de una base de datos y la correcta inserción de estos ya que es sensible al tratamiento de las claves primarias y foráneas.

Sin embargo, debido a que el conjunto de datos de este proyecto es muy extenso y es necesario establecer relaciones entre distintos conjuntos de datos se ha optado por SQL. El rendimiento de la ejecución de consultas es mejor en esta ya que a la hora de programar un script basta con reservar en memoria un puntero para realizarlas mientras que en CSV es necesario volcar todo el fichero. Asimismo, debido a que se va a hacer uso de datos comprometidos el grado de seguridad que aporta SQL es mayor que en CSV ya que los propios sistemas de gestión aportan una gran cantidad de herramientas de seguridad.

6.3 Algoritmos de agrupación

- **K-Means**: este algoritmo de clustering se basa en los siguientes pasos:

  1. Elegir el número *k* de clusters.
  2. Seleccionar al azar *k* puntos, llamados centroides.
  3. Asignar cada punto al centroide más cercano, formando los *k* clusters. Repetir este paso hasta que no haya nuevas asignaciones en el contenido de los clusters.

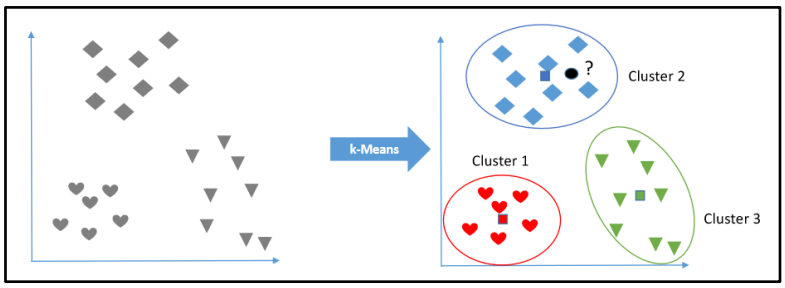

Figura 21: k-Means clustering [7]

- **Clustering jerárquico:** el proceso de este algoritmo es el siguiente:

  1. Cada muestra es un cluster.
  2. Elegir los dos clusters más cercanos, basándose en la distancia euclídea, y juntarlos en un único cluster. Repetir este paso hasta que solo se obtenga uno.

La representación más común de este algoritmo es un dendograma que representa las agrupaciones realizadas. Al finalizar el proceso se traza una línea horizontal en la mayor distancia euclídea y se elige el número de cluster en función de las mayores agrupaciones posibles que haya debajo de ella.

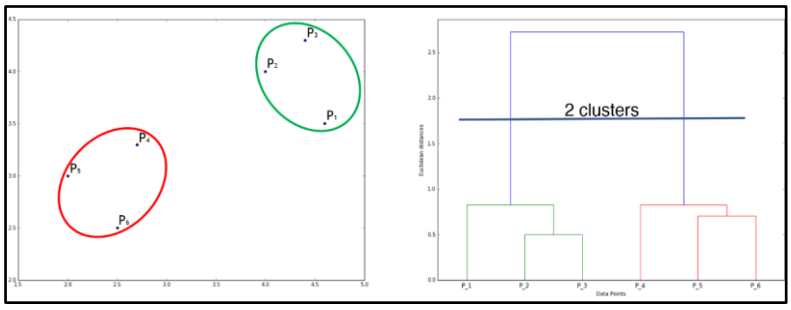

Figura 22: Uso del dendograma en el clustering jerárquico [10]

- **DBSCAN**: en este caso, el proceso del algoritmo es el siguiente:

1. Se agrupan las muestran en función de una pequeña distancia llamada vecindad ε o k-distancia.
2. Si la vecindad de dicha muestra tiene un número mínimo de puntos requeridos (*minPts*) se inicia un cluster sobre ella y todas las muestras de dicha vecindad se añaden al cluster. Sin embargo, en el caso de que cumpliera el *minPts* establecido dicha muestra es considerada ruido. Se repite hasta que todas las muestras sean etiquetadas como ruido o cluster.

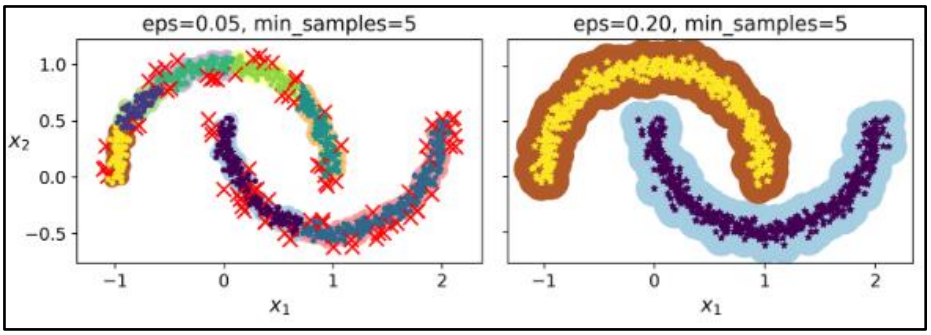

Figura 23: DBSCAN [6]

El algoritmo escogido ha sido el k-Means debido a que es el mejor rendimiento aporta con grandes conjuntos de datos, es el más sencillo de implementar y aun siendo sensible a la naturaleza de los datos, es más tolerante en comparación con los otros dos.

6.4 Algoritmos de visualización y reducción de dimensión

- **PCA**: este algoritmo se encarga de reducir la dimensión de un conjunto de datos no etiquetados. El proceso el siguiente:

  1. Escalar las variables de la matriz de características *X* formada por *m* variables independientes.
  2. Calcular la matriz de covarianzas de las *m* variables independientes de *X*.
  3. Calcular los valores *y* vectores propios de la matriz de covarianzas.
  4. Elegir un porcentaje *P* de varianza explicada y elegir los *p<=m* valores propios más grandes. Las componentes principales *C* son los *p* vectores propios asociados a estos *p* valores más grandes. El espacio *m*-dimensional del conjunto original se proyecta a un nuevo subespacio *p*-dimensional, reduciéndose la dimensionalidad.

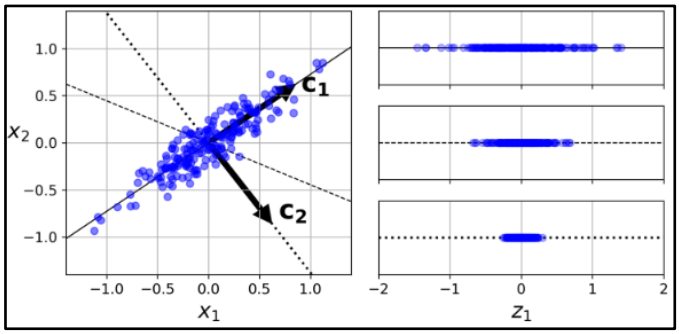

Figura 24: PCA [6]

- **LDA**: a diferencia de los vectores en PCA en este algoritmo se hace uso de clases *C* como si fuera un algoritmo de clasificación. El proceso es el siguiente:

  1. Escalar las variables de la matriz de características *X* formada por *m* variables independientes.
  2. Calcular *C* vectores *m*-dimensionales de modo que cada uno contenga las medias de las características para cada clase.
  3. Calcular la matriz de productos cruzados en la media para cada clase.
  4. Calcular la matriz de covarianza global entre clases *B*.
  5. Calcular los *p* vectores propios de la matriz y elegir los más grandes como el número de dimensiones reducidas.
  6. Los *p* vectores propios asociados a los *p* valores propios más grandes son los discriminantes lineales. El espacio *m*-dimensional del conjunto de datos original se proyecta al nuevo subespacio *p*-dimensional de las características.

Debido a que la ejecución de esta familia de algoritmos se va a hacer después de realizar el clustering se opta por LDA ya que los clusters obtenidos serán las clases C del algoritmo.

7 Análisis de riesgos

7.1 Obtención errónea de los datos

El proyecto no se puede realizar en caso de que las sondas hayan capturados datos corruptos o que el código que hace las consultas a la API no procese bien el JSON de respuesta.

- Probabilidad: baja.
- Impacto: muy alto.
- Plan de contingencia: realizar pruebas de testeo en el código de la API y comprobar detalladamente la salida del tráfico capturado por las sondas.

7.2 Imprecisión de los algoritmos

El núcleo de este proyecto reside en hacer uso de este tipo de algoritmos por lo que una mala optimización originaría una imprecisión con los resultados.

- Probabilidad: media.
- Impacto: alto.
- Plan de contingencia: técnicas de optimización.

7.3 Mala planificación

Como cualquier proyecto es posible que ocurran retrasos por la variabilidad en el tiempo y por la existencia de los propios riesgos.

- Probabilidad: baja.
- Impacto: medio.
- Plan de contingencia: planificar las tareas con un cierto margen.

7.4 Pérdida de la información

Cualquier pérdida de información ya sea de documentación o de software origina que se puedan perder semanas de trabajo.

- Probabilidad: muy baja.
- Impacto: alto.
- Plan de contingencia: utilización de la nube.

7.5 Matriz probabilidad-impacto

Con el fin de representar claramente las probabilidades y el impacto de cada uno de los riesgos se ha realizado la siguiente matriz:

Impacto / Probabilidad	Muy bajo	Bajo	Medio	Alto	Muy alto
Muy baja				Pérdida de la información	
Baja			Mala planificación		Obtención errónea de los datos
Media				Imprecisión de los algoritmos	
Alta					
Muy alta					

Tabla 1: Matriz probilidad-impacto

8 Descripción de la metodología

8.1 Descripción de los datos

Como se indicó en el apartado 3 se va a hacer uso de dos conjuntos de datos diferentes donde la forma de obtención difiere.

8.1.1 Sondas OptiWi-fi

El primer método consiste en el uso de sondas Wi-Fi de la empresa OptiWi-fi en TUD. Este tipo de empresa se encarga de ofrecer una solución de alto nivel relacionada con la información de tráfico de una red Wi-Fi. Para ello, sus sondas capturan el contenido de los paquetes de interconexión entre los hosts y los APs. El formato de almacenamiento se basa en un modelo entidad-relación de una base de datos SQL formada por 31 entidades:

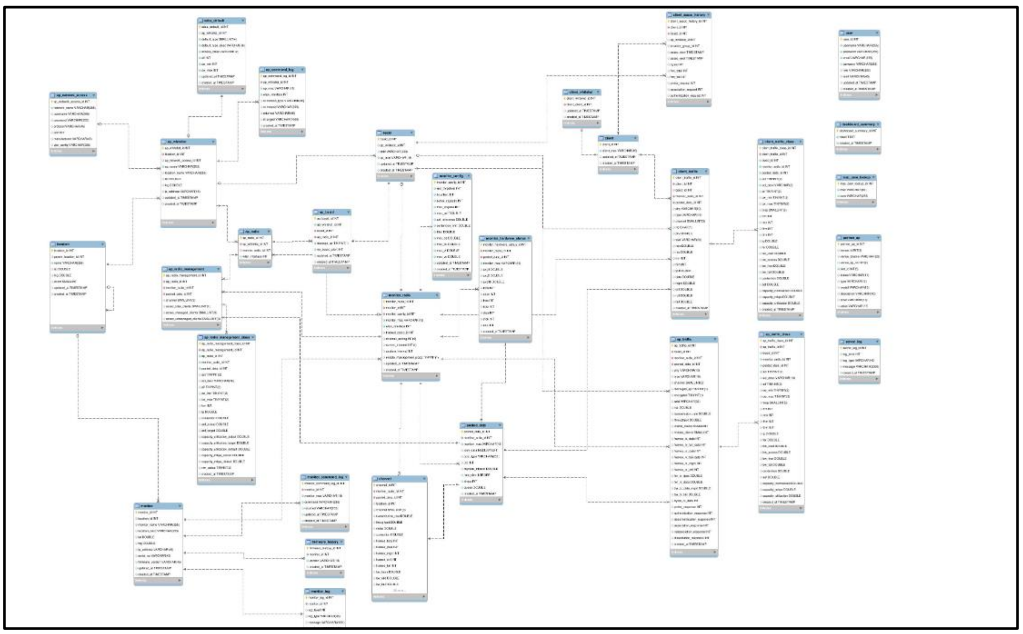

Figura 25: Modelo entidad-relación de OptiWi-fi

Sin embargo, los datos proporcionados, tráfico capturado desde el 2 de mayo de 2018 al 30 de mayo de 2018, no son las 31 entidades si no una serie de ellas almacenadas cada una en un fichero CSV.

- **channel**: contiene la información del tráfico para cada canal de la sonda.
- **client_traffic**: tráfico capturado de cada cliente.
- **client**: contiene la dirección MAC de cada cliente.
- **ap_traffic**: tráfico capturado de cada AP.
- **ap_traffic_class**: informa sobre las categorías de tráfico y clave foránea de ap_traffic.
- **bssid**: relaciona las entidades client_traffic y ap_traffic. Además, indica la dirección MAC de cada AP perteneciente a una SSID.

Por último, como se observó en el análisis de alternativas, el formato CSV no es el óptimo como almacenamiento por lo que se crea una base de datos, una para cada uno de los dos escenarios, replicando casi el modelo de la de por defecto. A aparte de las entidades anteriormente mostradas se ha creado la entidad monitor_radio debido a que se disponía información sobre la localización de las sondas. Dicha entidad se encarga de establecer las relaciones entre las entidades y establecer el tipo de escenario en función de la localización. Las distintas localizaciones son las siguientes:

- ***Church Lane Building***: 8 sondas. Este edificio está formado por clases, despachos de profesores y de secretarías de departamento.
    - Planta baja: departamentos y laboratorios.
    - Primera planta: aulas y laboratorios.
- **Bibliotecas**: 2 sondas.
- **CNRI** (*Communications Network Research Institute*): 4 sondas. Este edificio está formado por grupos de investigación con sus correspondientes laboratorios.

Así pues, los distintos escenarios son:

- Bibliotecas
- Clases y laboratorios
- Secretarías y laboratorios
- Sólo laboratorios

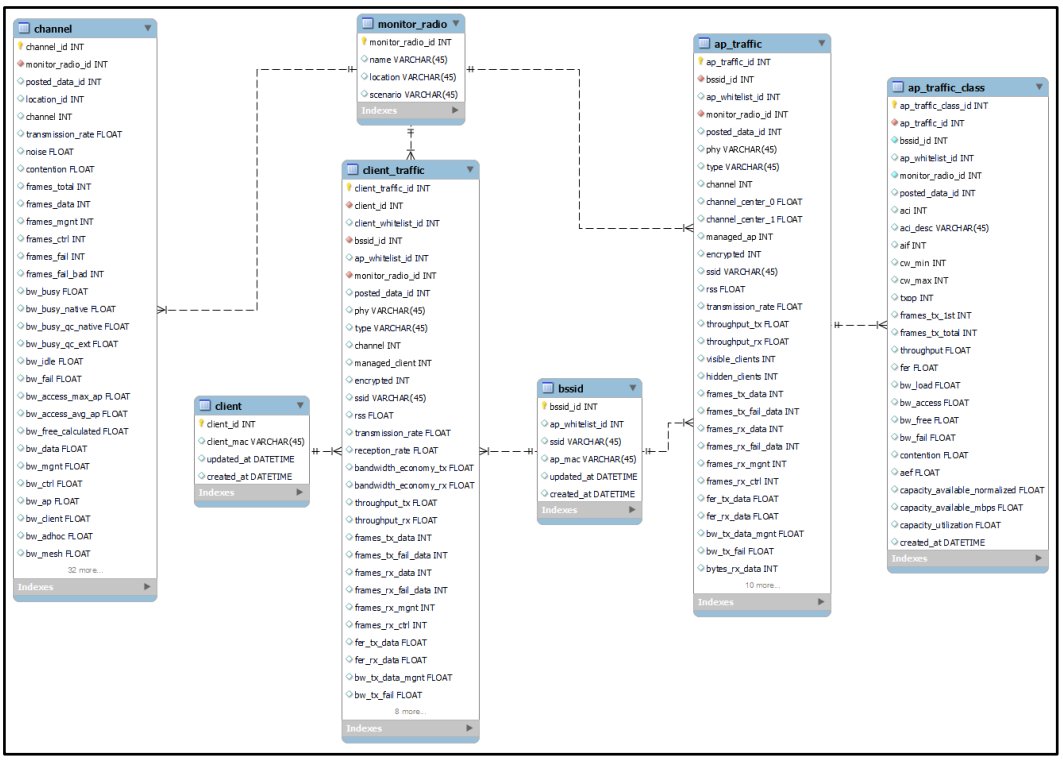

Figura 26: Base de datos TUD (Anexo 2)

8.1.2 Cisco Prime Infrastructure API

A diferencia de la herramienta anterior, haciendo uso de esta API se pueden obtener datos continuos de tráfico. Para ello, basta con realizar una petición como se muestra en la siguiente figura:

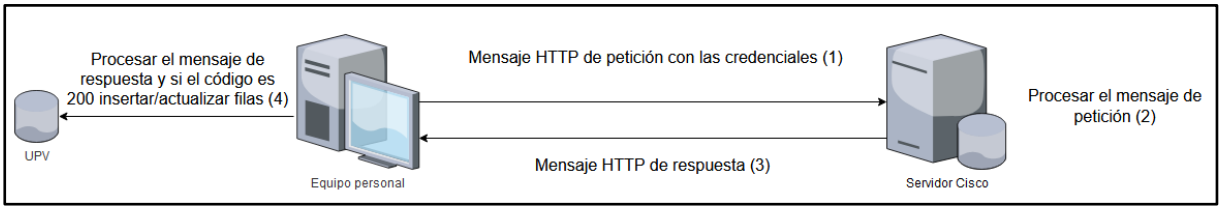

Figura 27: Proceso de obtención de datos de la API de Cisco

En primer lugar, el equipo del usuario envía una petición HTTP (*Hypertext Transfer Protocol*) a la dirección del dominio del servidor de Cisco. En dicho mensaje se indican las credenciales de acceso, el formato de envío de datos y el recurso a obtener, es decir, la entidad. Como formato de datos se indica JSON porque en comparación con XML (*eXtensible Markup Language*) consume menos ancho de banda de la red. Además, se añade la instrucción *?.full=true* a la dirección del dominio para que se reciba todo el conjunto de información de cada entidad.

Cuando el servidor recibe la petición, la procesa y genera una respuesta HTTP con unos de los siguientes códigos:

- 200: las credenciales son correctas.
- 400: credenciales incorrectas.
- 404: el recurso no está disponible.

Cuando el equipo recibe el mensaje de respuesta comprueba el código y si es 200 procesa el JSON con los datos. Por último, tras procesar la información se establece la conexión con la base de datos y se insertan o actualizan las filas con dicha información. En el Anexo 3 se encuentra el código realizado.

8.2 Características seleccionadas

Ya con los datos descritos el siguiente paso radica en que características de tráfico de cada base de datos se van a hacer uso como datos de entrada del algoritmo. Para ello, se va a hacer uso del estudio '*A Survey on Machine Learning-Based Performance Improvement of Wireless Networks: PHY, MAC and Network Layer*' [7] que engloba una serie de estudios relacionados con la implementación del aprendizaje automático en redes inalámbricas. En dicho estudio se identifican dos categorías u objetivos donde el aprendizaje automático dota a las redes inalámbricas la capacidad de aprender e inferir de los datos y extraer patrones:

- **Mejoras de rendimiento:** se basan en los KPI y los conocimientos del entorno (Ej. medio radioeléctrico) adquiridos de los dispositivos. Los resultados obtenidos permiten modificar los parámetros de funcionamiento en las capas físicas, enlace y red pudiéndose obtener una mejor QoS.
- **Procesamiento de la información:** consiste en los datos generados por los dispositivos inalámbricos en la capa de aplicación (Ej. monitorización Ambiental en IoT (*Internet of things),* localización, etc.)

Debido a que el propósito del proyecto está relacionado con la QoS, se busca información referente al primer punto del documento. En él, se establecen tres tipos de categorías para los datos del tráfico de red: radio espectro, la trama MAC y la red. Esta diferenciación permite que las características de cada conjunto de datos tengan una varianza parecida, por lo que los resultados no van a estar alterados en comparación de que se hubieran introducido datos sin ningún tipo de diferenciación.

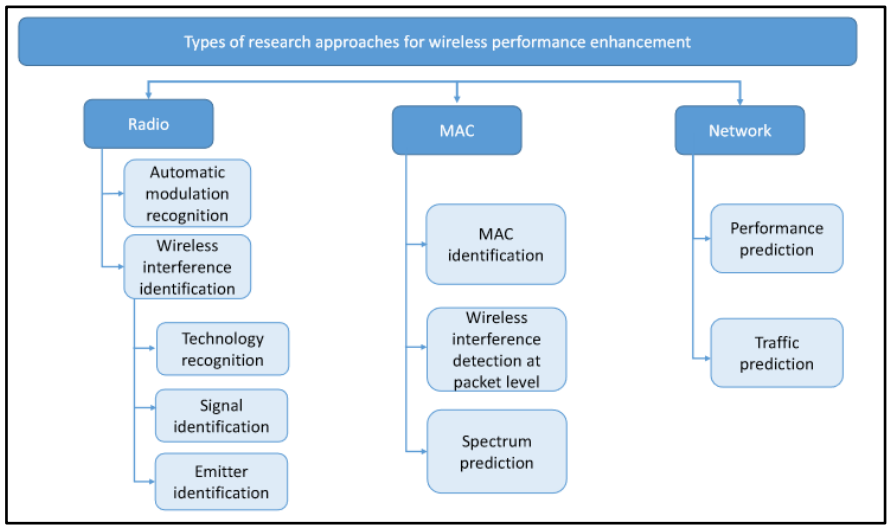

Figura 28: Tipos de categorías de datos a la hora de hacer un análisis de aprendizaje automático en redes inalámbricas [7]

8.2.1 TUD

Para esta base de datos se escoge la entidad *channel* ya que se basa en el tráfico por canal en cada sonda, por lo que es la que más información aporta. Debido a la disponibilidad de uso de la entidad *monitor_radio* se va a realizar un análisis basado en la diferenciación del escenario, es decir, el conjunto de datos se va a pasar por el algoritmo K-Means clustering. Este algoritmo establecerá una serie de clusters y el objetivo es comprobar si se establecen 4 clusters, el mismo número de los escenarios mencionados en el subapartado anterior. Asimismo, se calculan los valores medios de las características para cada cluster y se hace uso del algoritmo ADL para obtener el grado de influencia de cada característica en el cluster originando cuales de ellas hay que priorizar para mejorar la QoS. Por otra parte, se va a utilizar el DBSAN para plasmar la cantidad exacta de ruido que puede haber en el conjunto de datos.

Existen un gran conjunto de características pero con el fin de no seleccionar todas se ha hecho un filtrado basándose en que algunas de ellas tienen muchos registros nulos o que el sumatorio de alguna de ellas queda definida en otra. Así pues, las características para cada categoría a utilizar son las siguientes:

- **Radio espectro**: ancho de banda ocupado, inactivo, perdido y libre.
- **Trama MAC**: tramas de información, gestión, control y corruptas.
- **Red**: número de clientes y APs en el canal y número de clientes asociados y ocultos en la red BSS.

8.2.2 UPV/EHU

En este caso se va a hacer uso de la entidad *client_stat* (Anexo 4), que es en la que se almacena el tráfico de los clientes. Sin embargo, las credenciales proporcionadas para hacer consultas a la API no tienen permisos de escritura por motivos de seguridad. Por defecto, el número de últimos registros en las entidades relacionadas con los APs es 100 por lo que al no poderse modificar no se puede relacionar el campo *collectionTime* (fecha de capturas) con el de los APs para obtener la localización del cliente. En consecuencia, la diferencia con el análisis de TUD es que no se puede establecer una relación entre el número de clusters con los escenarios. Al disponer de la MAC se podrán focalizar los recursos en los clientes que hayan sido agrupados con peor QoS.

Al analizar las características disponibles se observa que los controladores de la API de Cisco no tienen tanta capacidad de captura de la información en comparación con las sondas Wi-Fi de OptiWi-fi. Por ello, se van a agrupar todas en un mismo conjunto. Dicho conjunto está formado por la tasa de datos de lectura, el número de reintentos de datos durante la sesión, la intensidad de la señal recibida en el AP por parte del cliente y la relación de señal a ruido. Las demás características tienen la mayoría valores nulos o las que se van a hacer uso ya la representan.

8.3 Resultados obtenidos

Para visualizar los resultados se ha optado por la creación de un *dashboard* dinámico con el fin de obtener una buena experiencia de usuario. Se ha optado por el uso de la herramienta de visualización Dash [11] debido a su detallada documentación, pero se podría haber optado por otros ya que la diferencia reside en la sintaxis del código embebido HTML (*HyperText Markup Language*) y JavaScript que ejecutará el navegador.

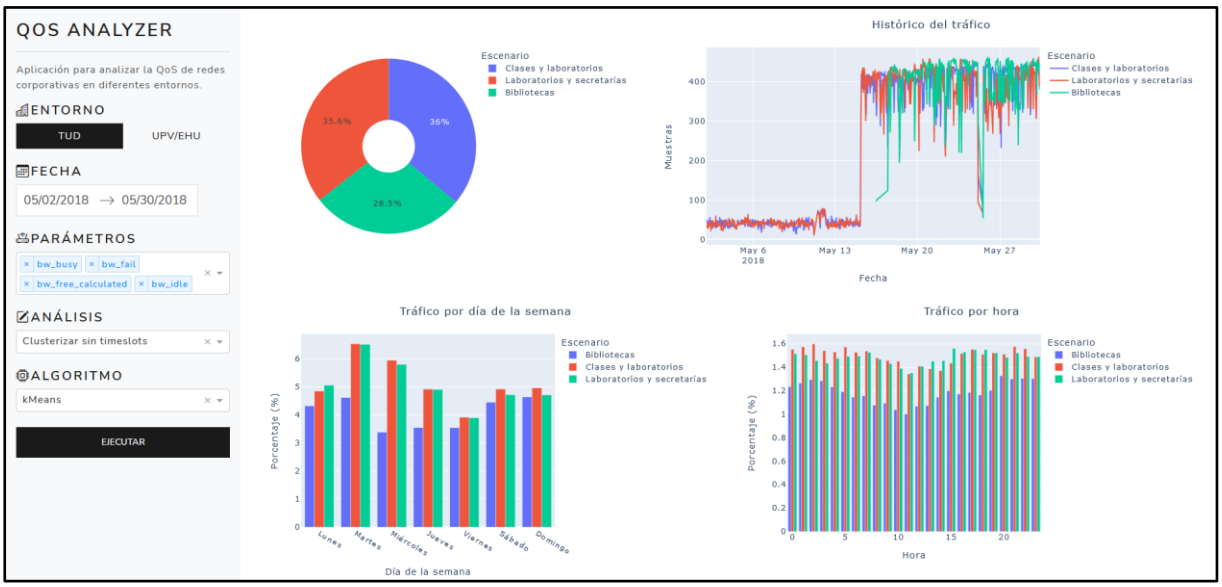

Figura 29: Vista del dashboard

El primer componente es una barra lateral, *sidebar*, donde el usuario elige la información que quiere visualizar:

- **Entorno:** se elige la base de datos a consultar.
- **Fecha:** automáticamente muestra el rango de fechas para el entorno seleccionado. Por otra parte, para que no haya errores la selección de fechas fuera de rango están deshabilitadas.
- **Parámetros:** desplegable en el que se obtienen de forma automática las características del entorno que se quieren usar. Soporta selección múltiple y eliminación de forma individual.
- **Análisis:** desplegable en el que se muestra el análisis a realizar en función del entorno seleccionado.
- **Algoritmo:** desplegable con el conjunto de algoritmos disponibles a usar.
- **Ejecutar**: botón que se encarga de ejecutar el análisis.

El segundo componente son las gráficas. Por defecto, cuando se selecciona el entorno automáticamente se muestran las referentes al histórico, por día de la semana y por hora de las muestras. Al modificar el rango de fechas automáticamente se actualizan las gráficas. Cuando el usuario pulsa el botón de ejecución se muestra un *spinner* para indicarle que el algoritmo se está ejecutando y si da error se muestra un mensaje de alerta (Ej. cuando solo se obtiene un cluster).

Por último, antes de explicar los resultados es importante establecer el proceso de tratamiento de los datos:

Figura 30: Proceso de tratamiento de los datos

En primer lugar, se eliminan las filas del conjunto de datos que contengan valores nulos. Después se estandarizan, esto quiere decir que todos los datos se encuentran escalados entre el [-1,1] por lo que características que tenían unidades de medida mayores quedan igualadas con las demás. La estandarización genera una campana de Gauss donde muchos valores estarán cercanos a 0 y los valores que difieran mucho de la media estarán situados en la parte derecha o izquierda del 0 con una distancia superior a la mayoría de ellos. A la matriz formada se la conoce como matriz de características X que es la que se introduce en los algoritmos.

El siguiente paso consiste en el cálculo de los valores óptimos de los algoritmos. En el caso de K-Means es necesario establecer el número de clusters a analizar. La técnica más utilizada para establecerlos consiste en el análisis de siluetas. Un coeficiente de silueta de una instancia es igual a *(b-a) / max(a,b)* donde *a* es la distancia media a las otras instancias en el mismo grupo y *b* es la distancia media a las instancias del siguiente cluster más cercano. Dicho coeficiente puede variar entre -1 y 1 donde un coeficiente cercano a 1 significa que la instancia está bien definida dentro de su propio cluster y lejos de otros clusters, mientras que un coeficiente cercano a 0 significa que está cerca de un límite de cluster, y finalmente un coeficiente cercano a -1 significa que la instancia puede haber sido asignada a un incorrecto cluster.

Por último, para DBSCAN se debe optimizar la vecindad (k-distancia). Para ello, se va a hacer uso del algoritmo K-NN que predice a que categoría pertenece un nuevo dato basándose a que categorías pertenecen los datos que están a su alrededor. Para establecer dicha pertenencia se deben escoger un número de *k* vecinos y la asignación se basa en la categoría que tenga la menor distancia euclídea con el nuevo punto. Para este algoritmo el valor *k* vecinos es el doble del mínimo de puntos requeridos *minPts*.

8.3.1 TUD

A la hora de seleccionar este entorno en el dashboard se muestran gráficas referentes al tráfico. En primer lugar, se observa que el escenario perteneciente a las clases y laboratorios es el que más muestras tiene (156647 (36 %)) mientras que el de las bibliotecas es el que menos (123738 (28.5%)). Asimismo, se muestra que no hay ninguna muestra del escenario laboratorios, es decir, ninguna sonda localizada en el CNRI captó información. Seguramente fuera debido a que no estaban todavía operativas en tales fechas. En cuanto al histórico se visualiza que a partir del 15 de mayo hubo un aumento del tráfico y que las primeras muestras de las bibliotecas se capturaron el día siguiente a las 11:00.

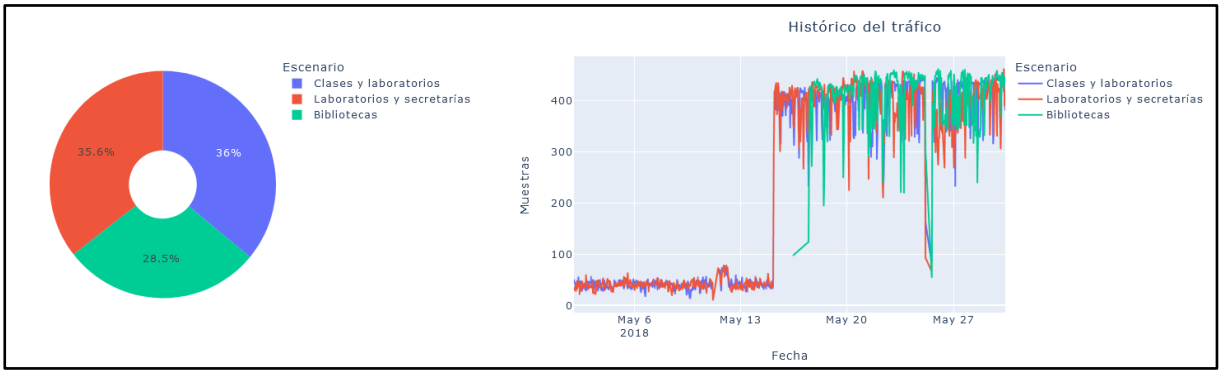

Figura 31: Porcentaje de escenarios e histórico del tráfico en TUD

A la hora de hacer un análisis del tráfico en función de la hora y día de la semana se concluye que el martes es el día donde mayor tráfico hay mientras que el viernes es donde menos mientras que en el tráfico por hora la distribución es casi uniforme.

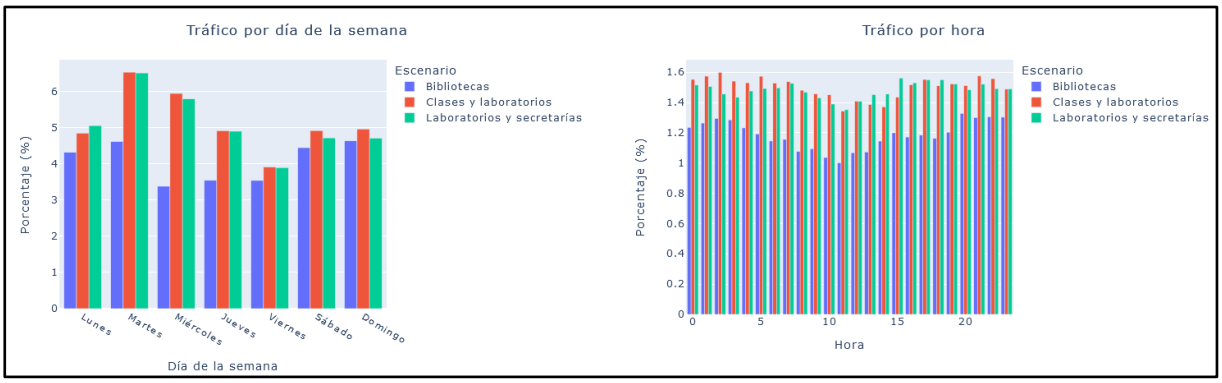

Figura 32: Porcentaje de muestras en función de la hora y día de la semana en TUD

El siguiente paso consiste en ejecutar el algoritmo k-Means con el fin de poder establecer agrupaciones y obtener que características son las más influyentes. El conjunto de muestra de la base de datos entera es muy grande (434891 muestras) por lo que se va a escoger el intervalo más corto debido a que en caso contrario el algoritmo tardaría bastantes horas en ejecutarse. En tales circunstancias se ha seleccionado el rango del 29 de mayo al 30 de mayo cuya proporción de escenarios es casi igual:

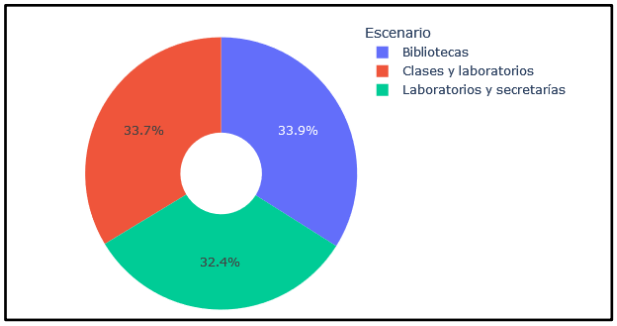

Figura 33: Proporción de la muestras en función de los escenarios entre el 29-30 de mayo en TUD

8.3.1.1 Radio espectro

El número de cluster óptimo es 2 ya que para dicho valor el coeficiente de siluetas es el máximo. El 77.6 % de las muestras pertenecen al cluster 2 por lo que no hay diferencia del tipo de tráfico en función del escenario. Se puede observar que en las clases y laboratorios es donde menor diferencia de proporción hay entre los clusters.

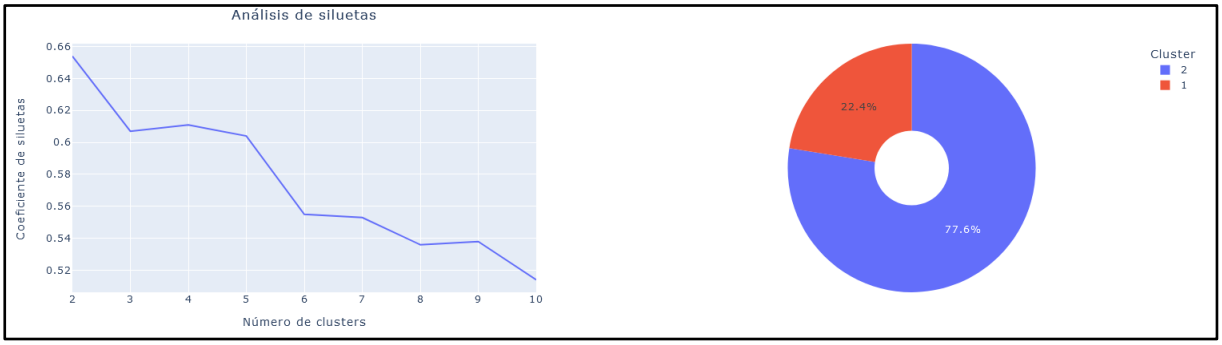

Figura 34: Análisis de siluetas y porcentaje de clusters para la categoría del radio espectro en TUD

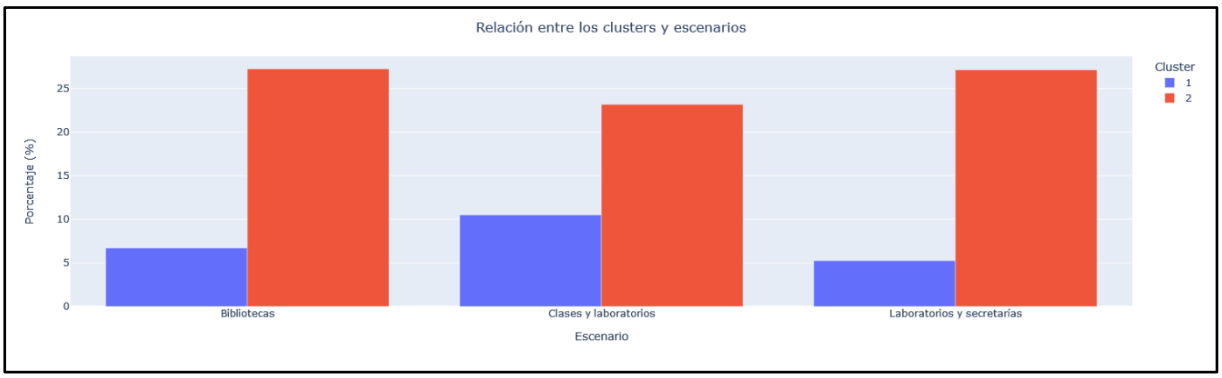

Figura 35: Relación entre los clusters y escenarios para la categoría del radio espectro en TUD

La ejecución del LDA establece que con un solo discriminante lineal se puede representar toda la varianza de los datos. A la hora de analizar los coeficientes se concluye que la característica más influyente es el ancho de banda inactivo *bw_idle* y mientras que el ancho de banda libre *bw_free_calculated* es la de menos.

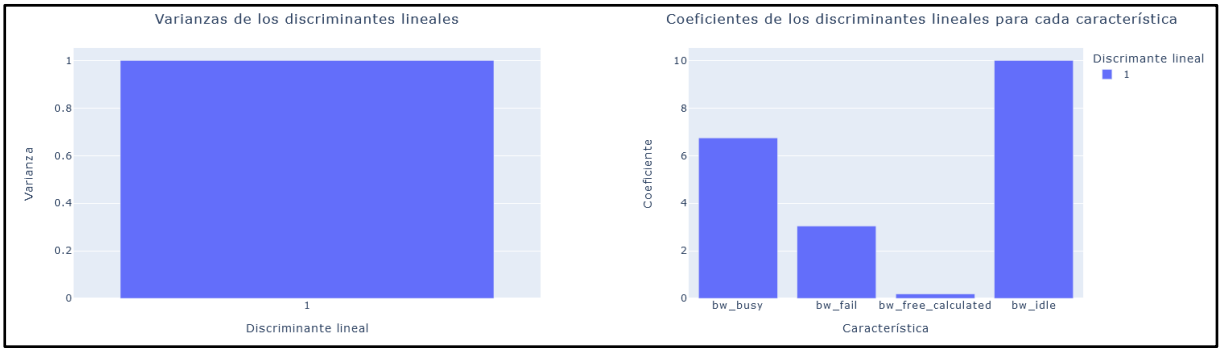

Figura 36: Resultados del LDA en la categoría del radio espectro en TUD

En la relación entre los clusters y las características y las medias de cada una por cluster se visualiza una diferencia bastante favorable al cluster 2 en el ancho de banda perdido *bw_fail* y ocupado *bw_busy* por lo que los equipos conectados a los canales pertenecientes al cluster 1 esos días experimentaron una peor QoS.

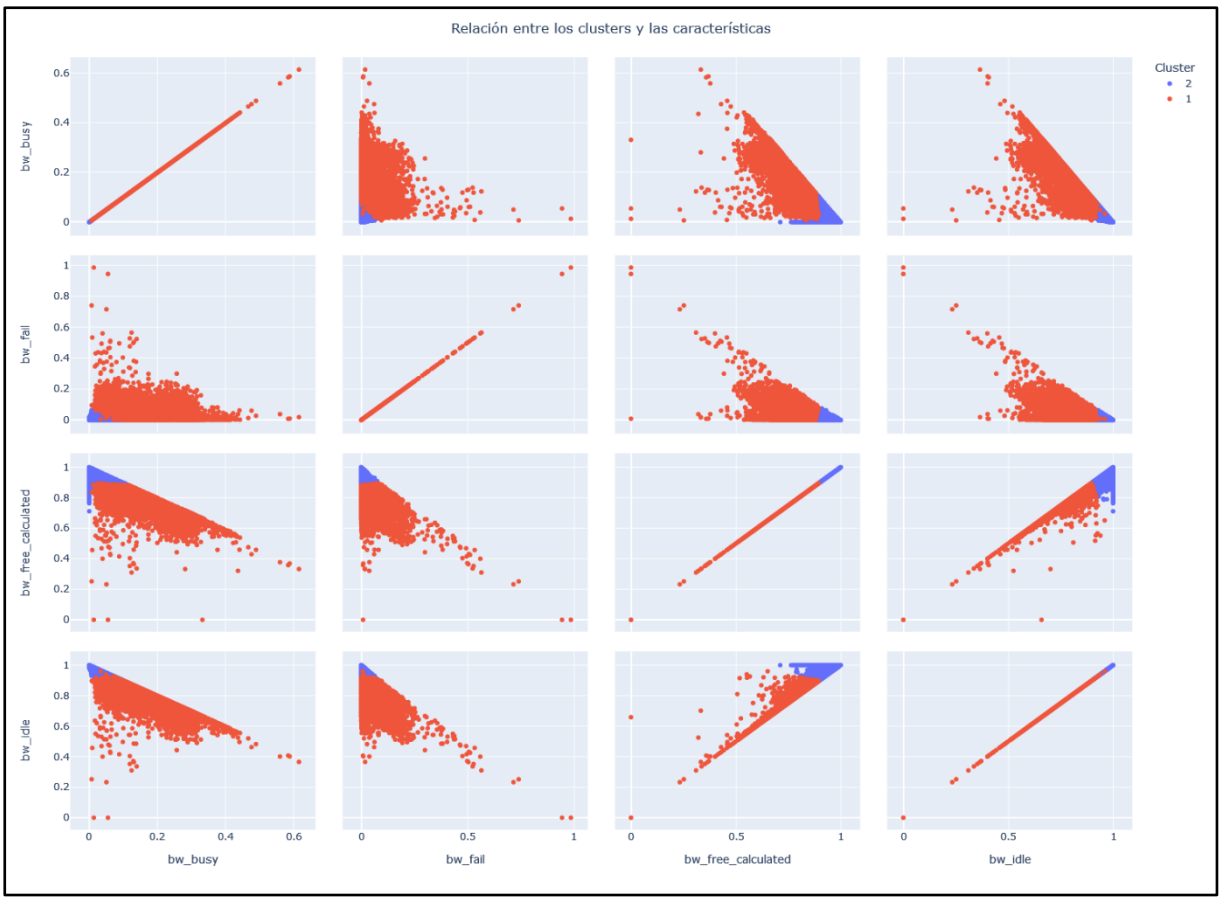

Figura 37: Relación entre los clusters y las características en la categoría del radio espectro en TUD

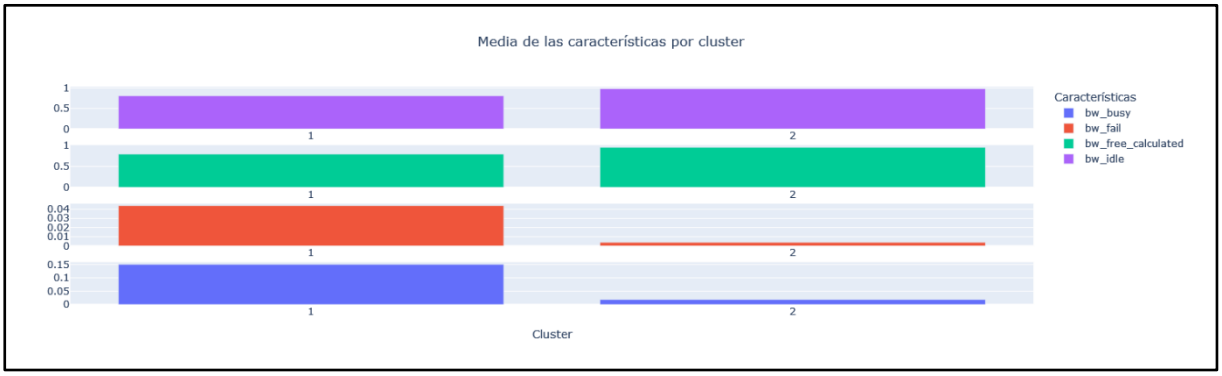

Figura 38: Medias por cluster en la categoría del radio espectro en TUD

La ejecución del DBSCAN devuelve que el 15.7 % de las muestras son ruido:

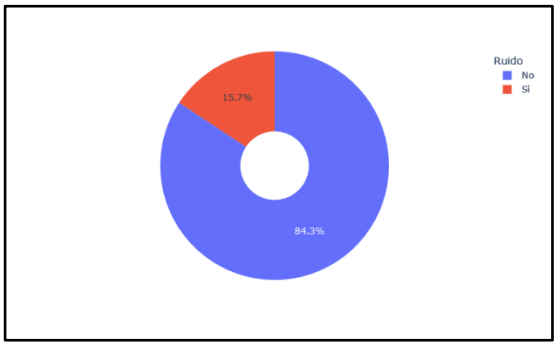

Figura 39: Porcentaje de ruido en la categoría del radio espectro en TUD

8.3.1.2 Trama MAC

El número de cluster óptimo es 2 donde el 97.9% es el cluster 1 por lo que no hay diferencia del tipo de tráfico en función del escenario.

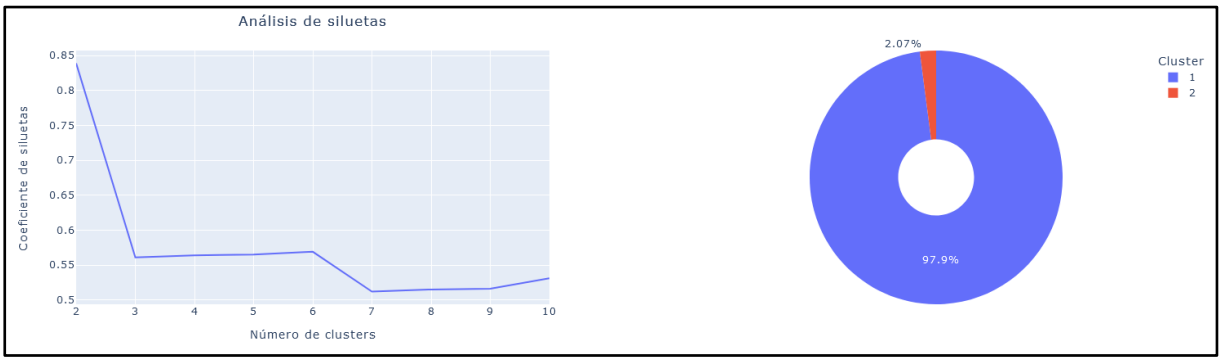

Figura 40: Análisis de siluetas y porcentaje de clusters para la categoría de la trama MAC en TUD

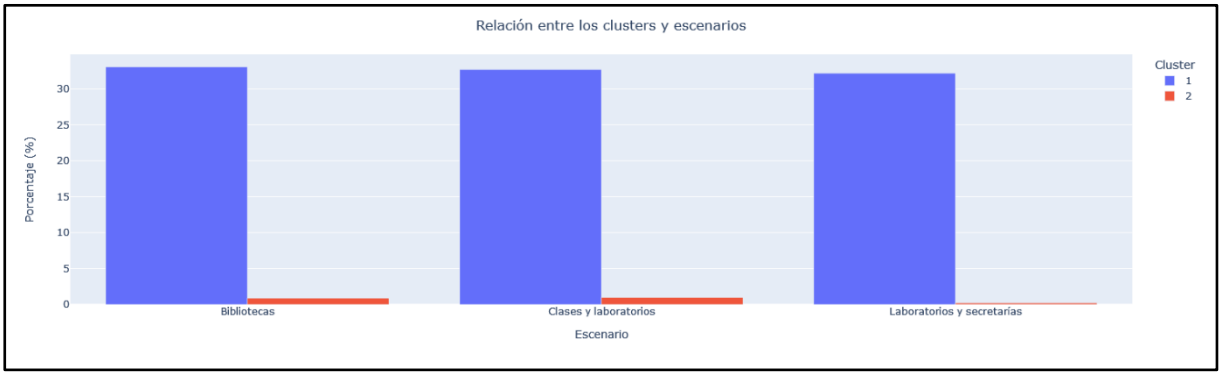

Figura 41: Relación entre los clusters y escenarios para la categoría de la trama MAC en TUD

La ejecución del LDA establece que la característica más influyente es el número de tramas erróneas *frames_fail* mientras la menos influyente es el número de tramas de gestión *frames_mgnt*.

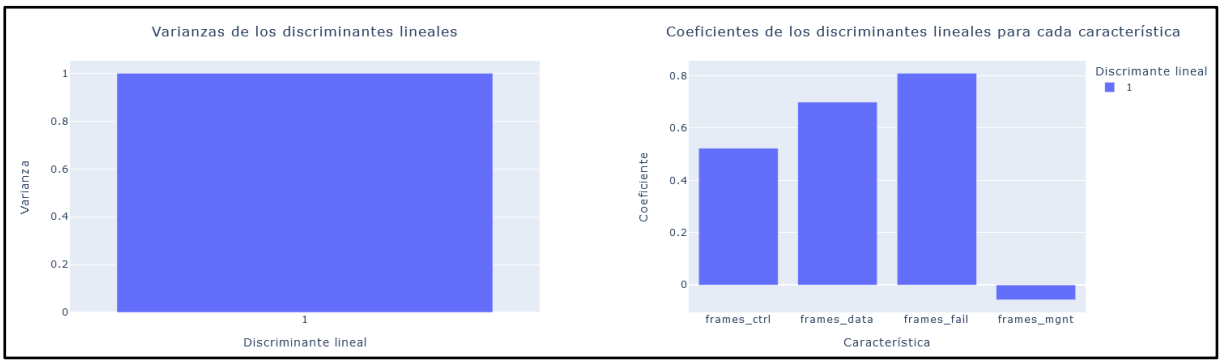

Figura 42: Resultados del LDA en la categoría de la trama MAC en TUD

Para esta categoría, en la relación entre los clusters y las características y las medias de cada una por cluster se concluye que los equipos pertenecientes al cluster 2 utilizaron una gran cantidad de ancho de banda por lo que la focalización de los recursos tendría que haber sido en ellos para que experimentaran la mejor QoS posible.

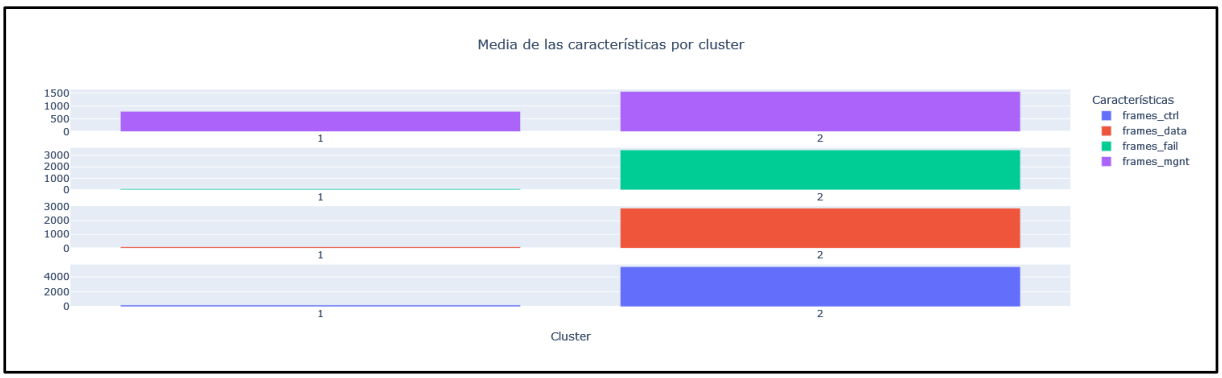

Figura 43: Medias por cluster en la categoría de la trama MAC en TUD

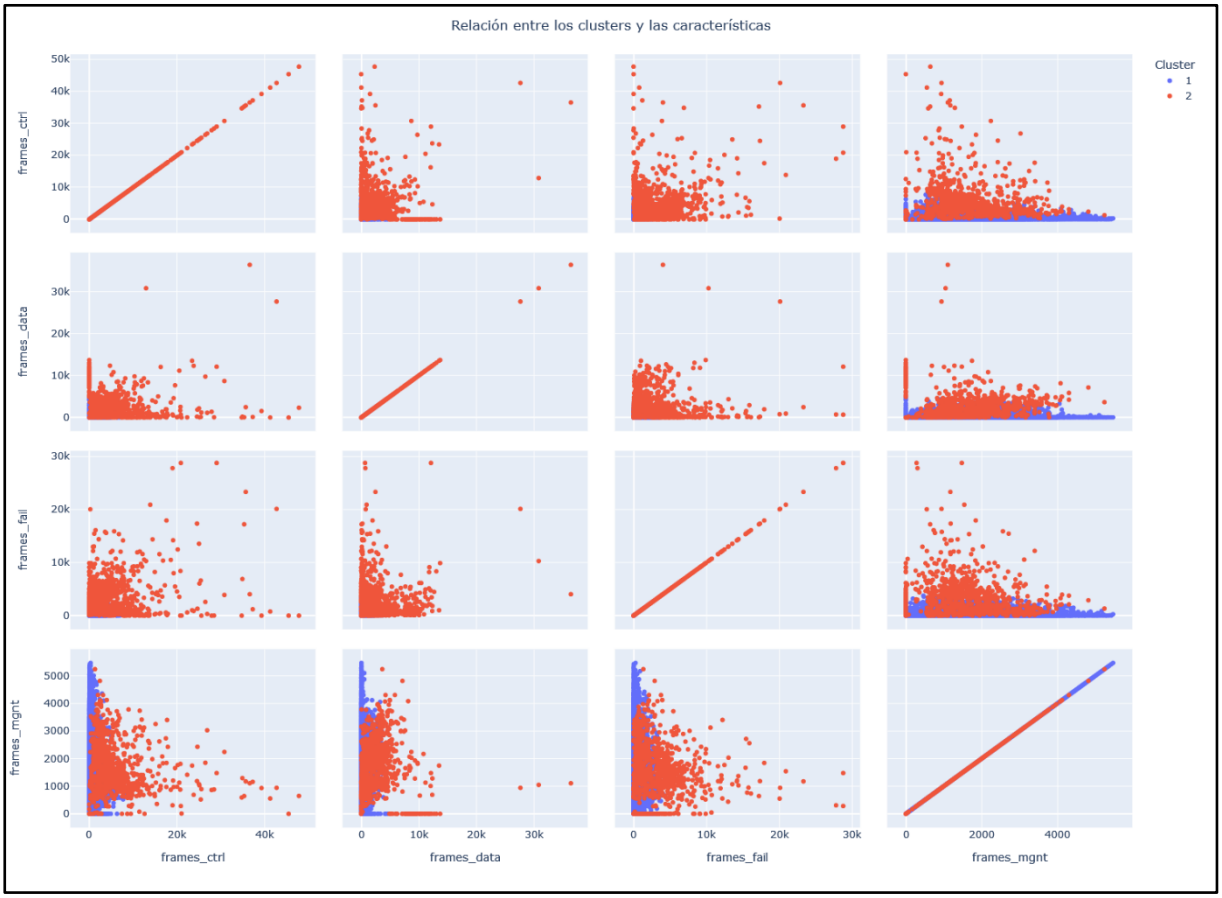

Figura 44: Relación entre los clusters y las características en la categoría de la trama MAC en TUD

A la hora de ejecutar DBSCAN se observa que el 10.1 % de las muestras son ruido:

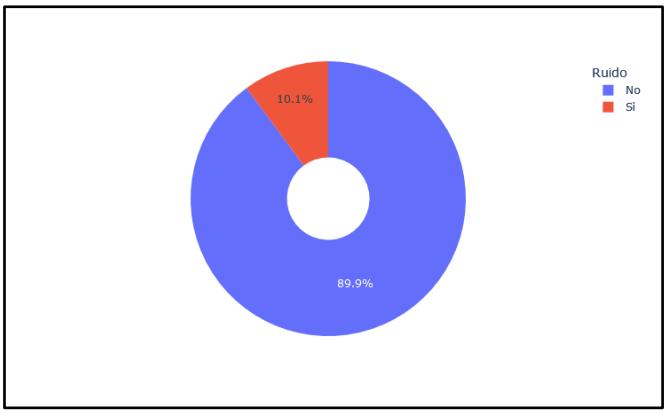

Figura 45: Porcentaje de ruido en la categoría de la trama MAC en TUD

8.3.1.3 Red

Para esta última categoría el número de clusters óptimo es 2 donde el 80.3% es el cluster 1 por lo que no hay diferencia del tipo de tráfico en función del escenario. El escenario correspondiente a laboratorios y secretarías es donde la proporción entre los clusters es más lejana.

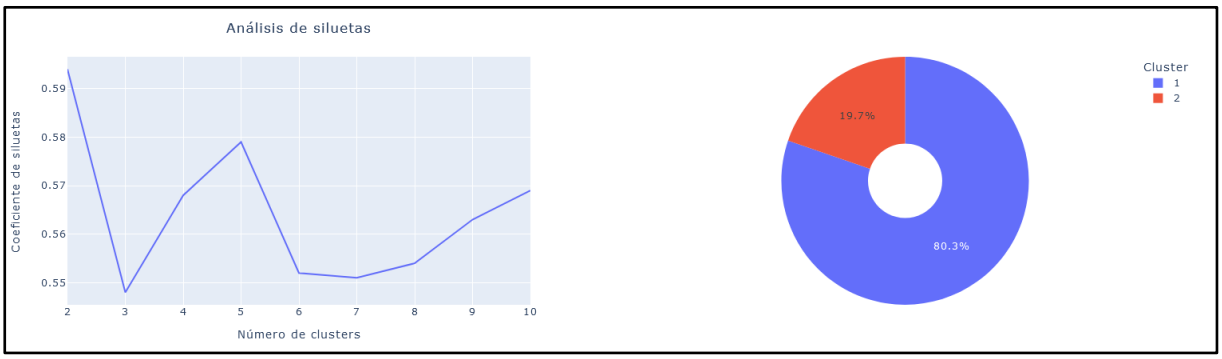

Figura 46: Análisis de siluetas y porcentaje de clusters para la categoría de la red en TUD

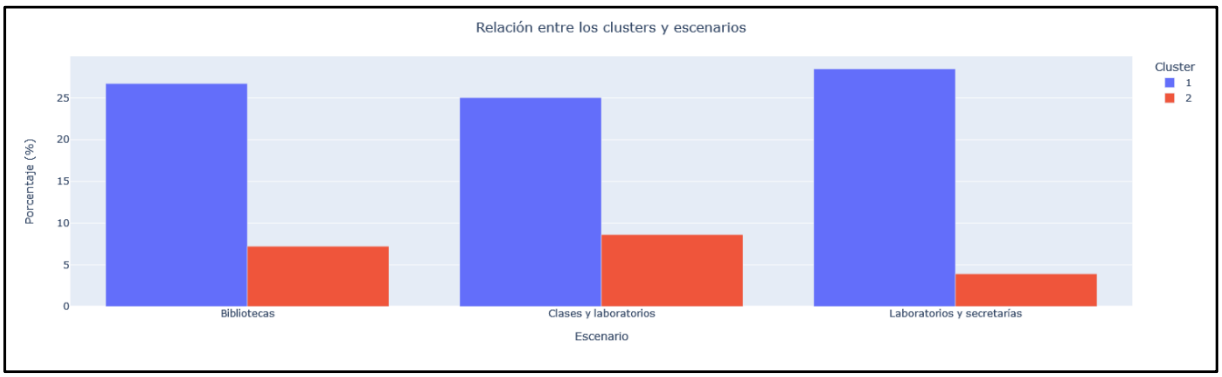

Figura 47: Relación entre los clusters y escenarios para la categoría de la red en TUD

El LDA informa que la categoría más influyente es el número de clientes en canal *clients_channel* mientras que la menor es el número de APs en el canal *aps_channel*.

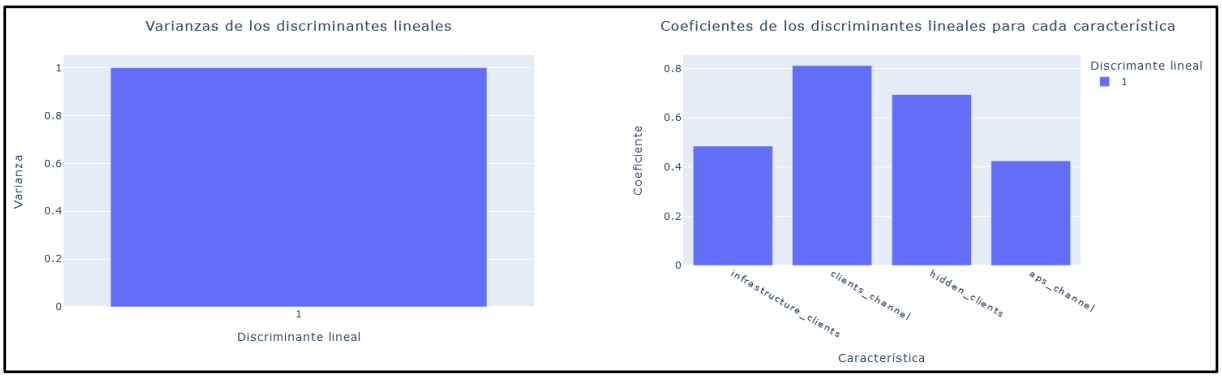

Figura 48: Resultados del LDA en la categoría de la red en TUD

Por otra parte, en la relación entre los clusters y las características y las medias de cada una por cluster se concluye que los equipos pertenecientes al cluster 2 se encontraban en un canal bastante saturado por lo que experimentaron una peor QoS.

Figura 49: Relación entre los clusters y las características en la categoría de la red en TUD

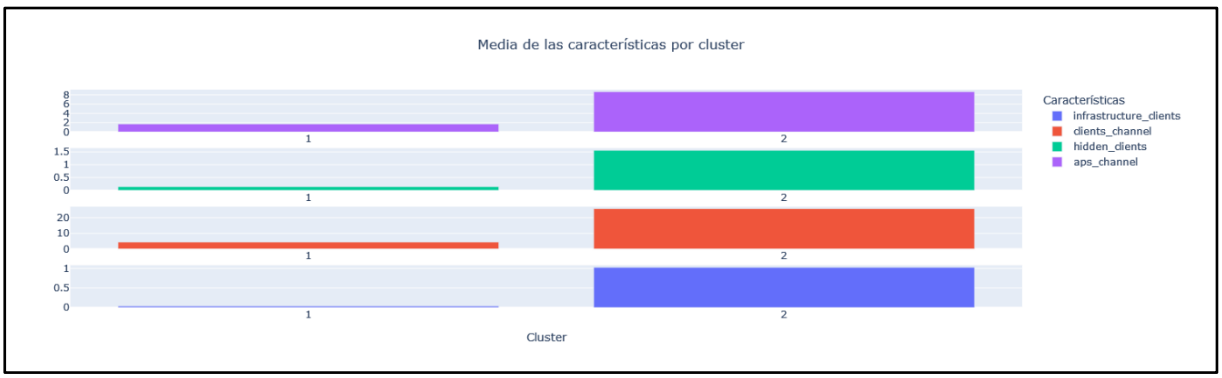

Figura 50: Medias por cluster en la categoría de la red en TUD

Por último, con DBSCAN solo se ha obtenido un cluster por lo que se deduce que estas características o la propia categoría no son adecuadas para este algoritmo.

8.3.2 UPV/EHU

En las gráficas referentes al tráfico se observa que para las 248430 muestras capturadas, al mediodía es donde hay mayor tráfico mientras el jueves es el día de mayor carga. En el histórico se visualiza que durante algunos días no se pudo capturar el tráfico debido a fallos ajenos a la conexión con la API. Por otra parte, se observa que a partir de las vacaciones de Julio lógicamente el tráfico se reduce. Ya que el conjunto de datos es menor que TUD se establece un rango de fechas mayor, la semana del 13 al 19 de junio.

Figura 51: Tráfico histórico en la UPV/EHU

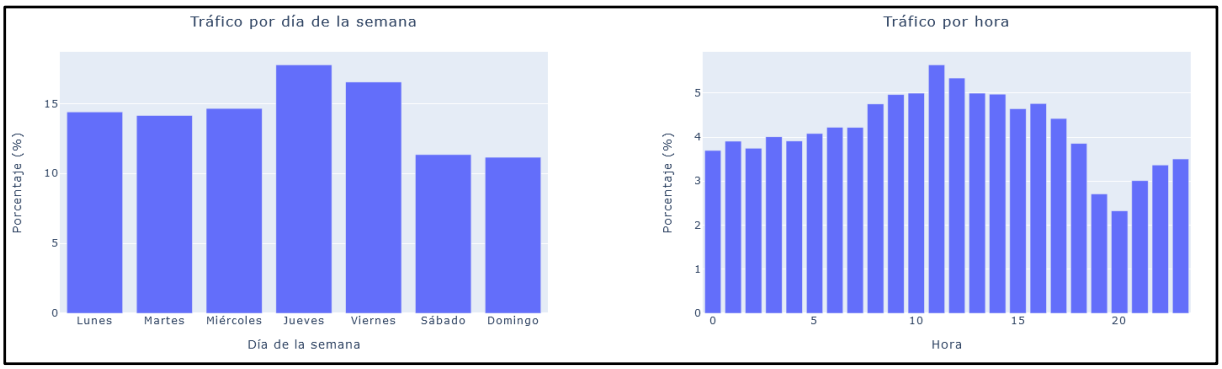

Figura 52: Tráfico por día de la semana y hora en la UPV/EHU

El número de clusters obtenido es 3 pero uno de ellos solo agrupa al 1.05 % de la muestras :

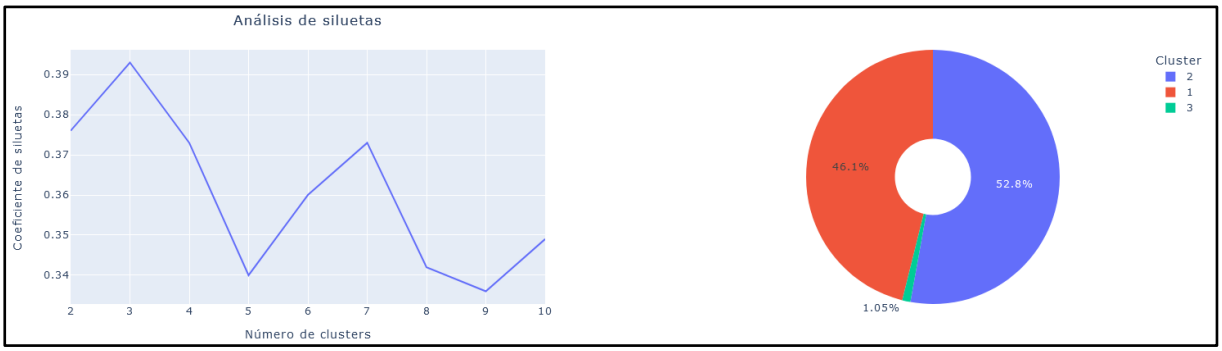

Figura 53: Análisis de siluetas y porcentaje de clusters para la UPV/EHU

Tras la ejecución del LDA se observa que a diferencia de TUD se han obtenido dos discriminantes lineales. La categoría más influyente es el número de reintentos de datos *dataRetries*. En caso de que se hubiera mantenido la misma gráfica de coeficientes y la varianza del primer discriminante fuera igual o superior a 0.7 la elección hubiera sido la intensidad de la señal del cliente recibida *rssi*.

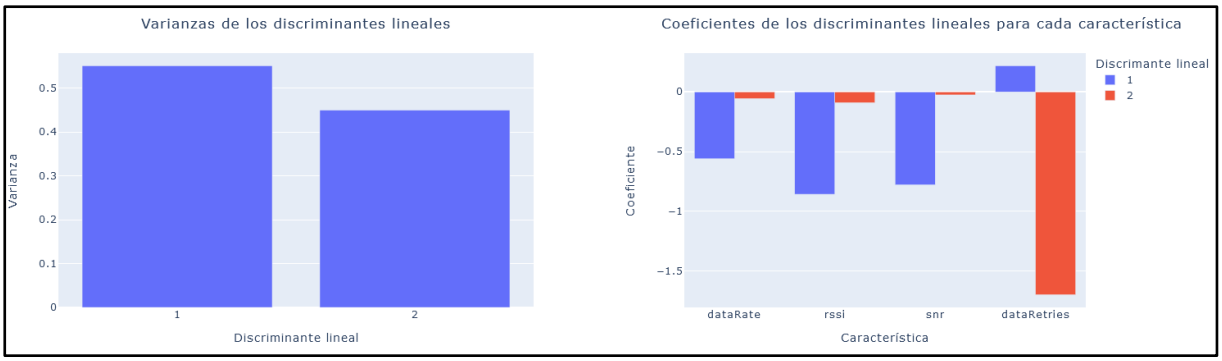

Figura 54: Resultados del LDA en la UPV/EHU

En la relación entre los clusters y las características y las medias de cada una por cluster se concluye que los equipos pertenecientes al cluster 3 son con diferencia los que peor QoS experimentaron debido al número de reintentos de datos realizados.

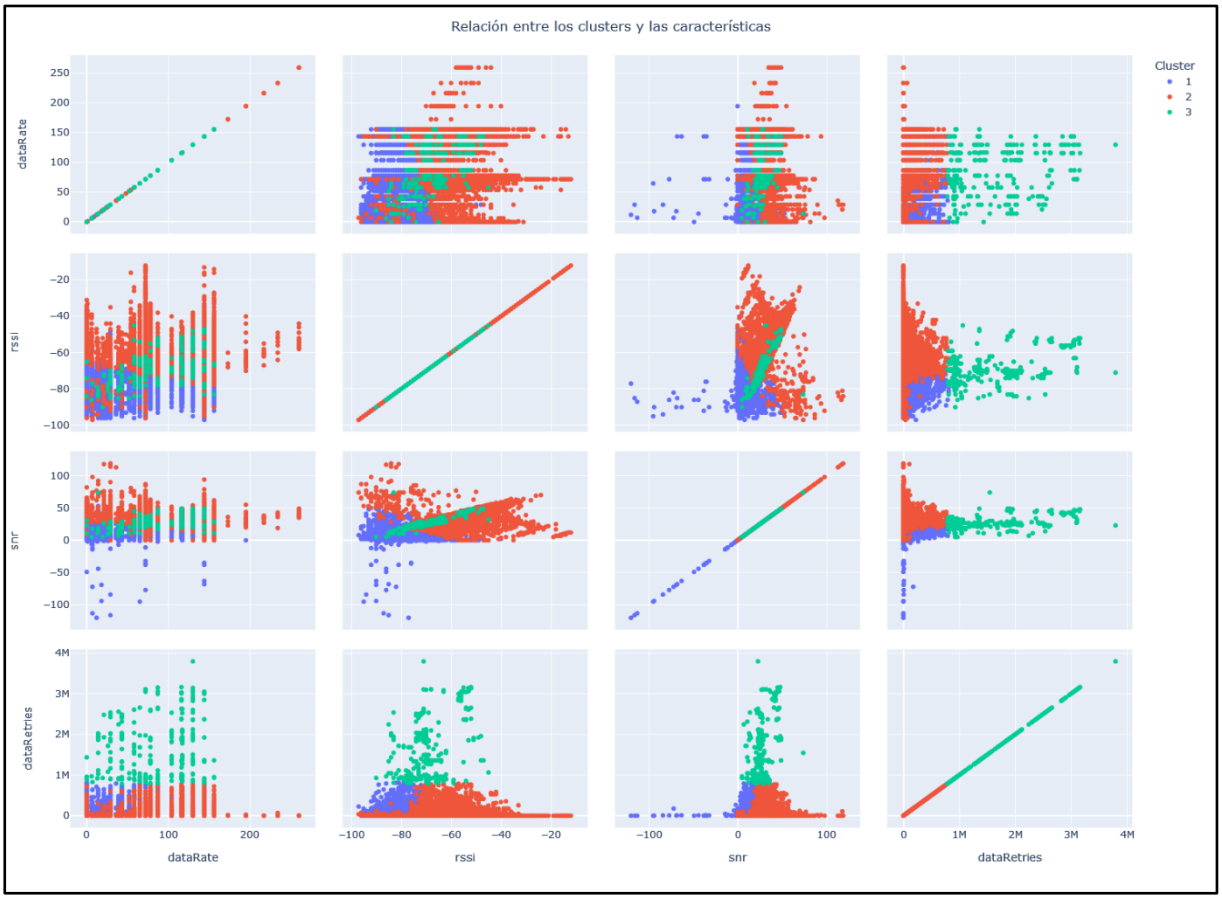

Figura 55: Relación entre los clusters y las características en la UPV/EHU

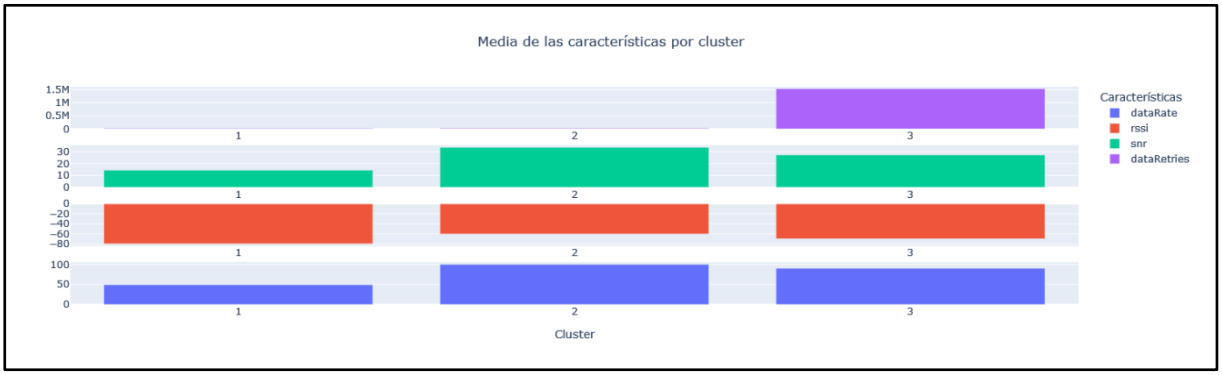

Figura 56: Medias por cluster en la UPV/EHU

Por último, el porcentaje de ruidos en este caso es del 14.6%:

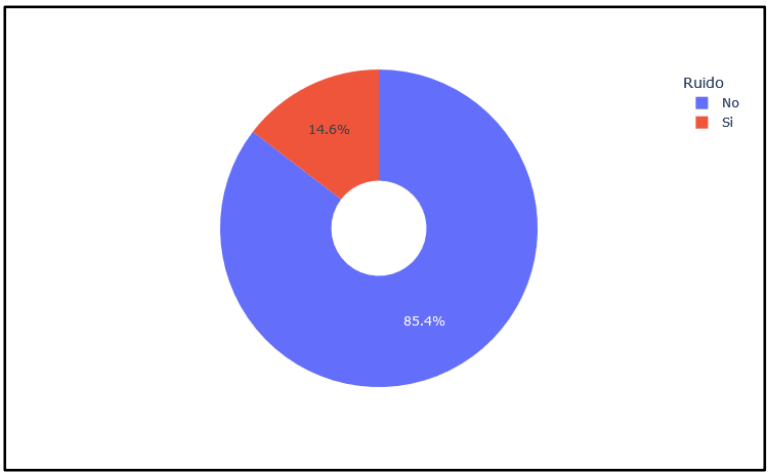

Figura 57: Porcentaje muestras de ruido en la UPV/EHU

9 Planificación

9.1 Equipo de trabajo

Nombre y apellidos	Responsabilidad	Tarea
Alejandro Aguado Marín	Ingeniero junior	Realización del proyecto
Eva Ibarrola Armendariz	Ingeniera senior	Dirección y seguimiento del proyecto

Tabla 2: Equipo de trabajo

9.2 Fases del proyecto

1. Proposición del proyecto y seguimiento: establecer el objetivo y alcance haciendo el seguimiento para comprobar el correcto cumplimiento del mismo.
2. Formación: algoritmos de aprendizaje automático, redes Wi-Fi, QoS y herramienta de visualización de resultados.
3. Análisis de alternativas: analizar y escoger la elección más óptima para el lenguaje de programación, algoritmos, formato de almacenamiento y herramienta de visualización de los resultados.
4. Desarrollo del proyecto: tratamiento de los datos y programación del código.
5. Análisis de resultados y conclusiones: en base a los resultados obtenidos del estudio establecer las conclusiones pertenecientes.

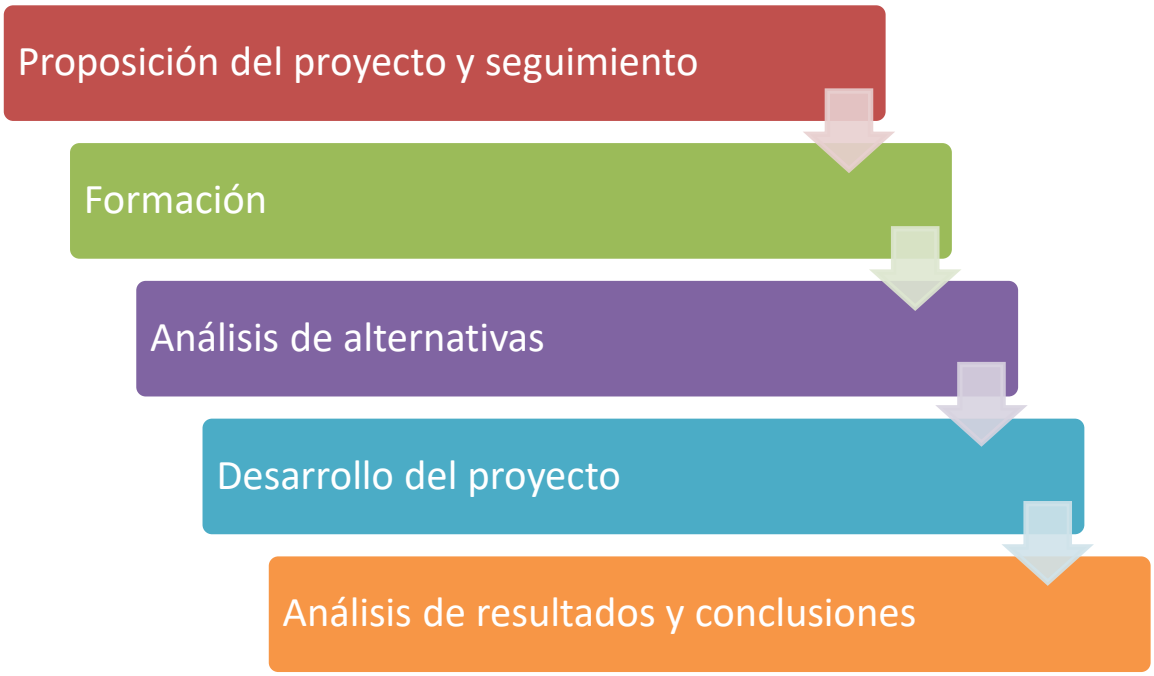

Figura 58: Fases del proyecto

9.3 Descripción de las tareas

9.3.1 Paquete de trabajo 1: Proposición y seguimiento del proyecto

Duración	188 días
Comienzo	15/03/2022
Fin	18/09/2022

Tabla 3: Paquete de trabajo 1: Proposición y seguimiento del proyecto

- Tarea 1: Proposición del proyecto

Comienzo	15/03/2022
Fin	15/03/2022
Descripción	Proposición del proyecto al alumno
Recursos técnicos	Ninguno
Recursos humanos	Ingeniera senior (1 hora)

Tabla 4: Tarea correspondiente a la proposición del proyecto

- Tarea 2: Seguimiento del proyecto

Comienzo	15/03/2022
Fin	18/09/2022
Descripción	Reuniones periódicas
Recursos técnicos	Ordenador (20 horas)
Recursos humanos	Ingeniera senior (20 horas)

Tabla 5: Tarea correspondiente al seguimiento del proyecto

- Tarea 3: Memoria

Comienzo	15/03/2022
Fin	18/09/2022
Descripción	Redactar documentación
Recursos técnicos	Ordenador (40 horas)
Recursos humanos	Ingeniero junior (40 horas)

Tabla 6: Tarea correspondiente a la memoria

9.3.2 Paquete de trabajo 2: Formación

Duración	67 días
Comienzo	20/03/2022
Fin	25/05/2022

Tabla 7: Paquete de trabajo 2: Formación

- Tarea 1: Aprendizaje automático

Comienzo	20/03/2022
Fin	10/04/2022
Descripción	Formación para obtener el conocimiento
Recursos técnicos	Ordenador y libro (70 horas)
Recursos humanos	Ingeniero junior (70 horas)

Tabla 8: Tarea correspondiente a la formación en aprendizaje automático

- Tarea 2: QoS

Comienzo	11/04/2022
Fin	18/04/2022
Descripción	Formación para obtener el conocimiento
Recursos técnicos	Ordenador (18 horas)
Recursos humanos	Ingeniero junior (18 horas)

Tabla 9: Tarea correspondiente a la formación en QoS

- Tarea 3: Redes Wi-Fi

Comienzo	19/04/2022
Fin	24/04/2022
Descripción	Formación para obtener conocimiento
Recursos técnicos	Ordenador y libro (15 horas)
Recursos humanos	Ingeniero junior (15 horas)

Tabla 10: Tarea correspondiente a la formación en redes Wi-Fi

- Tarea 4: Herramienta de visualización de los resultados

Comienzo	25/04/2022
Fin	15/05/2022
Descripción	Formación para obtener conocimiento
Recursos técnicos	Ordenador (50 horas)
Recursos humanos	Ingeniero junior (50 horas)

Tabla 11: Tarea correspondiente a la formación en la herramienta de visualización

- Tarea 4: API Cisco

Comienzo	16/05/2022
Fin	25/05/2022
Descripción	Formación para obtener conocimiento
Recursos técnicos	Ordenador (20 horas)
Recursos humanos	Ingeniero junior (20 horas)

Tabla 12: Tarea correspondiente a la formación en la API de Cisco

9.3.3 Paquete de trabajo 3: Análisis de alternativas

Duración	**16 días**
Comienzo	26/05/2022
Fin	10/06/2022

Tabla 13: Paquete de trabajo 3: Análisis de alternativas

- Tarea 1: Selección

Comienzo	25/06/2022
Fin	06/06/2022
Descripción	Seleccionar las alternativas
Recursos técnicos	Ordenador (15 horas)
Recursos humanos	Ingeniero junior (15 horas)

Tabla 14: Tarea correspondiente a la selección de alternativas

- Tarea 2: Descripción de la elección

Comienzo	07/06/2022
Fin	10/06/2022
Descripción	Describir la elección
Recursos técnicos	Ordenador (6 horas)
Recursos humanos	Ingeniero junior (6 horas)

Tabla 15: Tarea correspondiente a la descripción de la elección

9.3.4 Paquete de trabajo 4: Desarrollo del proyecto

Duración	**56 días**
Comienzo	15/07/2022
Fin	09/09/2022

Tabla 16: Paquete de trabajo 4: Desarrollo del proyecto

- Tarea 1: Toma de contacto con los datos

Comienzo	15/07/2022
Fin	22/07/2022
Descripción	Analizar los datos
Recursos técnicos	Ordenador (20 horas)
Recursos humanos	Ingeniero junior (20 horas)

Tabla 17: Tarea correspondiente con la toma de contacto con los datos

- Tarea 2: Programación del código

Comienzo	23/07/2022
Fin	09/09/2022
Descripción	Programar el código
Recursos técnicos	Ordenador (65 horas)
Recursos humanos	Ingeniero junior (65 horas)

Tabla 18: Tarea correspondiente con la programación del código

9.3.5 Paquete de trabajo 5: Análisis de los resultados y conclusiones

Duración	**7 días**
Comienzo	10/09/2022
Fin	17/09/2022

Tabla 19: Paquete de trabajo 5: Análisis de los resultados y conclusiones

- Tarea 1: Análisis de los resultados

Comienzo	10/09/2022
Fin	14/09/2022
Descripción	Análisis de resultados
Recursos técnicos	Ordenador (15 horas)
Recursos humanos	Ingeniero junior (15 horas)

Tabla 20: Tarea correspondiente con el análisis de los resultados

- Tarea 2: Definir las conclusiones

Comienzo	15/09/2022
Fin	17/09/2022
Descripción	Establecer conclusiones
Recursos técnicos	Ordenador (5 horas)
Recursos humanos	Ingeniero junior (5 horas)

Tabla 21: Tarea correspondiente a la definición de las conclusiones

9.4 Hitos

Hito	**Descripción**	**Fecha**
H1	Proyecto definido	15/03/2022
H2	Formación realizada	25/05/2022
H3	Solución establecida	10/06/2022
H4	Proyecto desarrollado	09/09/2022
H5	Conclusiones realizadas	17/09/2022
H6	Memoria entregada	18/09/2022

Tabla 22: Hitos

9.5 Diagrama de Gantt

	Código	Nombre de tarea	Duración	Comienzo	Fin
1	**P.T.1**	**Proposición del proyecto y seguimiento**	**188 días**	**mar 15/03/22**	**dom 18/09/22**
2	T.1.1	Proposición del proyecto	1 día	mar 15/03/22	mar 15/03/22
3	T.1.2	Seguimiento del proyecto	188 días	mar 15/03/22	dom 18/09/22
4	T.1.3	Memoria	188 días	mar 15/03/22	dom 18/09/22
5	H.1	Proyecto definido	1 día	mar 15/03/22	mar 15/03/22
6	H.6	Memoria entregada	1 día	dom 18/09/22	dom 18/09/22
7	**P.T.2**	**Formación**	**67 días**	**dom 20/03/22**	**mié 25/05/22**
8	T.2.1	Aprendizaje automático	22 días	dom 20/03/22	dom 10/04/22
9	T.2.2	QoS	8 días	lun 11/04/22	lun 18/04/22
10	T.2.3	Redes Wi-FI	6 días	mar 19/04/22	dom 24/04/22
11	T.2.4	Herramienta de visualización de los resultados	21 días	lun 25/04/22	dom 15/05/22
12	T.2.5	Cisco API	10 días	lun 16/05/22	mié 25/05/22
13	H.3	Formación realizada	1 día	mié 25/05/22	mié 25/05/22
14	**H.2**	**Análisis de alternativas**	**16 días**	**jue 26/05/22**	**vie 10/06/22**
15	T.3.1	Selección de las alternativas	12 días	jue 26/05/22	lun 06/06/22
16	T.3.2	Descripción de la elección	4 días	mar 07/06/22	vie 10/06/22
17	H.3	Solución establecida	1 día	vie 10/06/22	vie 10/06/22
18	**P.T.4**	**Desarrollo del proyecto**	**57 días**	**vie 15/07/22**	**vie 09/09/22**
19	T.4.1	Toma de contacto con los datos	8 días	vie 15/07/22	vie 22/07/22
20	T.4.2	Programación del código	49 días	sáb 23/07/22	vie 09/09/22
21	H.4	Proyecto desarrollado	1 día	vie 09/09/22	vie 09/09/22
22	**P.T.5**	**Análisis de los resultados y conclusiones**	**8 días**	**sáb 10/09/22**	**sáb 17/09/22**
23	T.5.1	Análisis de los resultados	5 días	sáb 10/09/22	mié 14/09/22
24	T.5.2	Definir las conclusiones	3 días	jue 15/09/22	sáb 17/09/22
25	H.5	Conclusiones realizadas	1 día	sáb 17/09/22	sáb 17/09/22

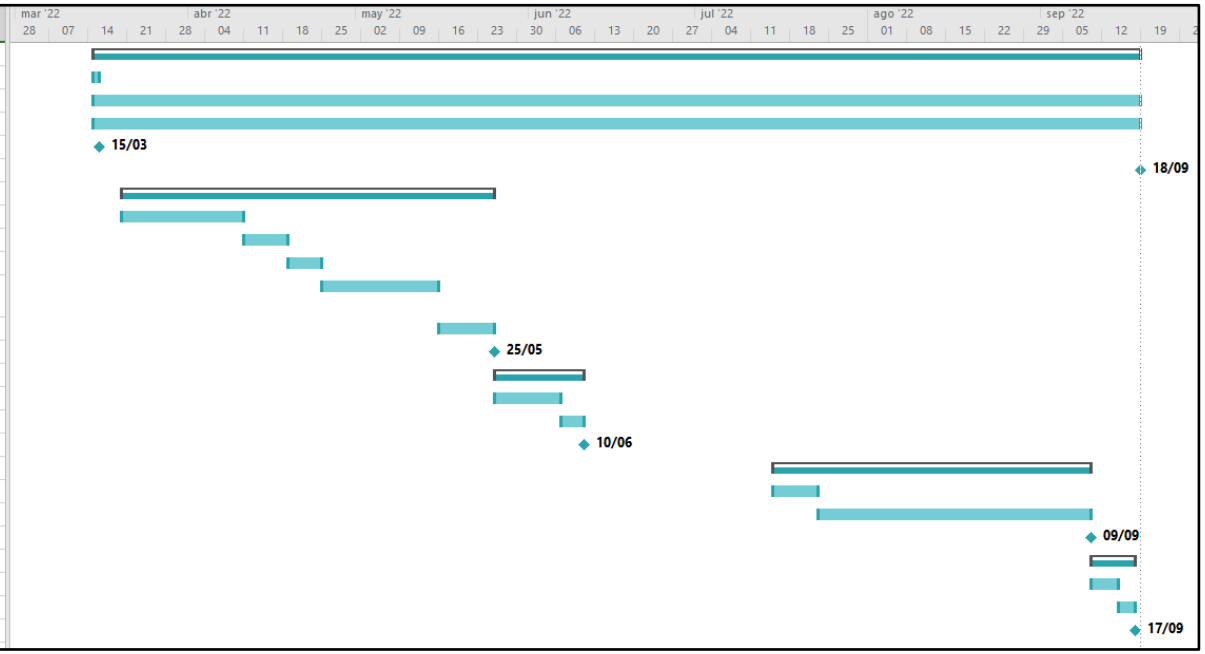

Figura 59: Diagrama de Gantt

10 Presupuestos y costes

10.1 Amortizaciones

Concepto	Coste inicial (€)	Vida útil (horas)	€/horas	Horas	Total (€)
Portátil Ing.Junior	600	2000	0,3	297	89,1
Ordenador Ing.Senior	1000	2500	0,4	21	8,4
Libro Redes	43,41	10000	0,0043	12	0,051
Libro ML	33,69	10000	0,0033	30	0,099
Licencia M.Office	15	10000	0,0015	40	0,06
SUBTOTAL					97,71

Tabla 23: Amortizaciones

10.2 Gastos

Concepto	Coste (€)
Material de oficina	60
Electricidad	35
SUBTOTAL	95

Tabla 24: Gastos

10.3 Horas internas

Nombre	Cargo	Categoría	Coste (€/h)	Horas	Total (€)
Alejandro Aguado Marín	Proyectista	Ingeniero Junior	15	339	5085
Eva Ibarrola Armendariz	Directora de proyecto	Ingeniera Senior	50	21	1050
SUBTOTAL					6135

Tabla 25: Horas internas

10.4 Presupuesto

Concepto	Coste (€)
Amortizaciones	97,71
Gastos	95
Horas internas	6135
Subcontrataciones	0
TOTAL	**6237,71**

Tabla 26: Presupuesto

11 Conclusiones

En este proyecto se ha conseguido desarrollar un sistema para la mejora de la QoS mediante aprendizaje automático. La solución ha consistido en implementar un dashboard interactivo para que el usuario pueda realizar las consultas de una forma clara y concisa. El usuario puede cambiar de entorno en tiempo real, filtrar el rango de fecha de capturar, seleccionar y eliminar las características, elegir el algoritmo, etc. En cuanto a los gráficos resultantes, se puede descargar, hacer zoom, mostrar solo el contenido de una de las categorías de la leyenda, etc. Asimismo, este dashboard se ha programado con la idea de que fuera modular, esto indica que el usuario podría implementar su algoritmo en un script a parte y luego importarlo al código del dashboard con la creación de una pequeña función en éste.

Por otra parte, se ha desarrollado un script que automatiza la inserción o actualización de registros en una base de datos que hace réplica del tráfico en la de UPV/EHU, que se basa en la API de Cisco. Dicho script está preparado para que se ejecute en bucle debido a que captura posibles fallos en la recepción de datos de la API haciendo uso de excepciones. Además, hace uso de temporizadores para no superar el límite máximo de consultas establecidas en dicha API.

En cuanto a los resultados se ha obtenido lo siguiente:

- **TUD:** el escenario perteneciente a las clases y laboratorios es el que más muestras tiene (156647 (36 %)) mientras que el de las bibliotecas es el que menos (123738 (28.5%)). El martes es el día donde mayor tráfico hay, mientras que el viernes es donde menos mientras que en el tráfico por hora la distribución es casi uniforme. En cuanto a los clusters obtenidos se han obtenido 2 para cada una de las categorías con una proporcionalidad menor al 25 % pero con diferencia en la trama MAC es donde más difiere, exactamente solo un 2.1 % para uno de los pares. En referente a los resultados se obtiene que el número de clientes conectados, el ancho de banda inactivo y el número de tramas erróneas son características muy influyentes en la QoS.

- **UPV/EHU:** la mayor de carga de tráfico se produce al mediodía y como día de la semana es el jueves el de mayor carga. Por otra parte, en este entorno se han obtenido 3 clusters para agrupar a los clientes. Dos de ellos tienen un alto porcentaje sobre la muestra pero hay uno que tiene un porcentaje inferior al 2 % . Este último con diferencia es donde los clientes tuvieron una peor QoS debido al número de reintentos de datos realizados.

Como indicaciones futuras se engloban las siguientes:

- Tener los accesos de escritura en la API de Cisco para poder replicar de una manera 100 % exacta.
- Continuar con diferentes análisis con diferentes algoritmos (Ej. uso de algoritmos de regresión para predecir la relación señal a ruido sabiendo el número de paquetes perdidos y el ancho de banda ocupado).
- Hacer uso de redes neuronales si se dispone de un equipo lo suficientemente potente.
- Explorar otras herramientas de captura diferentes a la API de Cisco y sondas OptiWi-fi.

Referencias

[1] Kurose, J. F., & Ross, K. W. (2022). Computer Networking: A Top-Down Approach (Octava ed.). Harlow, Reino Unido: Pearson.

[2] Social, W. A. (s.f.). Digital Report 2022: El informe sobre las tendencias digitales, redes sociales y mobile. Obtenido de https://wearesocial.com/es/blog/2022/01/digital-report-2022-el-informe-sobre-las-tendencias-digitales-redes-sociales-y-mobile/

[3] Cisco Prime Infrastructure API. (s.f.). Obtenido de https://developer.cisco.com/site/prime-infrastructure/documents/api-reference/rest-api-v3-0/

[4] TUD. (s.f.). Obtenido de https://www.tudublin.ie/

[5] OptiWi-fi. (s.f.). Obtenido de http://www.optiwifi.com/

[6] Géron, A. (2019). Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow (Segunda ed.). O'Reilly.

[7] Kulin, M., Kazaz, T., De Poorter, E., & Moerman, I. (2021). A Survey on Machine Learning-Based Performance Improvement of Wireless Networks: PHY, MAC and Network Layer. Obtenido de https://www.mdpi.com/2079-9292/10/3/318

[8] ITU, I. T. (2017). Quality of Service: Regulation manual. Obtenido de https://www.itu.int/pub/D-PREF-BB.QOS_REG01-2017

[9] Stack Overlow Trends. (2022). Obtenido de https://insights.stackoverflow.com/trends?tags=java%2Cc%2Cc%2B%2B%2Cpython%2Cc%23%2Cvb.net%2Cjavascript%2Cassembly%2Cphp%2Cperl%2Cruby%2Cvb%2Cswift%2Cr%2Cobjective-c

[10] Comilla J.G., de Ponteves H., Eremenko K.,SuperDataScience Team. (2021). Machine Learning de A a la Z:R y Phyton para Data Science. Obtenido de https://www.udemy.com/course/machinelearning-es/

[11] Dash. (s.f.). Obtenido de https://dash.plotly.com/

[12] Britannica, E. (s.f.). Wi-Fi: Networking techonology. Obtenido de https://www.britannica.com/technology/Wi-Fi

[13] Colliau T., Grace R.,Hughes Z.,Ceyhun O. (2017). Valparaiso University: MatLab vs. Python vs. R. Obtenido de https://scholar.valpo.edu/cgi/viewcontent.cgi?article=1049&context=cba_fac_pub

Anexo 1: Entidades de la API de Cisco

Entidad	Descripción
AccessPointDetails	Información detallada de un punto de acceso inalámbrico. Incluye atributos básicos, inventario, CDP, cliente, etc.
AccessPoints	Proporciona atributos del dispositivo como el tipo, la versión, el controlador de asociación, el número de cliente asociado, etc.
Alarms	Representación del fallo o cambio de estado que se ha producido en el sistema gestionado.
ApOnboardingProfile	Lista de perfiles de incorporación de AP
ApiHealthRecords	Rendimiento de la API y la información de diagnóstico.
ApiResponseTimeSummary	Información sobre los tiempos de respuesta de cada servicio.
Applications	Lista de aplicaciones predefinidas y definidas por el usuario.
AutoApRadioDetails	Representa una interfaz de radio de un punto de acceso autónomo.
BulkSanitizedConfigArchives	Ficheros de configuración limpios.
BulkUnsanitizedConfigArchives	Ficheros de configuración corruptos.
CliTemplate	Plantillas de configuración CLI.
ClientCounts	Número de clientes contados durante el último ciclo de sondeo. La combinación de tipo, clave y subtipo representa el número de clientes asociados a la entidad.
ClientDetails	Vista detallada de un cliente. Proporciona atributos del dispositivo del cliente, información de seguridad, dispositivo conectado, tráfico e información de la sesión. Toda la información se recoge en la sesión actual o en la última sesión.
ClientSessions	Vista detallada de las sesiones de los clientes. Proporciona atributos relacionados con el dispositivo y la sesión, incluyendo la seguridad, el dispositivo conectado, el tiempo de la sesión, el tráfico, etc.
ClientStats	Últimos datos estadísticos de los clientes encuestados. Los datos representados aquí son los contadores recuperados de los controladores.
ClientTraffics	Información de tráfico de los clientes recogida durante el último ciclo de sondeo.
Clients	Vista del cliente con información sobre los puntos finales. Como las direcciones MAC e IP, el nombre de usuario y el estado.
ConfigArchives	Archivo de configuración para el dispositivo.
ConfigVersions	Información sobre la versión de los archivos de configuración.
Devices	Representa la vista del dispositivo con información sobre los elementos de red gestionados. Proporciona información del dispositivo, como el nombre del dispositivo, el tipo de dispositivo, la dirección ip, el tipo de software, la versión, y también proporciona la accesibilidad y el estado de gestión.
Events	Representa un registro normalizado de un suceso notificado por la red, podría ser un syslog, una trampa SNMP, etc.
GroupSpecification	Proporciona una vista de la información específica del grupo de

	dispositivos.
GuestUsers	Devuelve una lista de usuarios invitados.
HistoricalClientCounts	Representa el recuento de clientes recogido en las últimas 24 horas con un intervalo de 5 minutos (por defecto).
HistoricalClientStats	Representa las estadísticas de los clientes recogidas en las últimas 24 horas con un intervalo de 15 minutos (por defecto).
HistoricalClientTraffics	Representa la información de tráfico del cliente recogida en las últimas 24 horas con un intervalo de 15 minutos (por defecto).
HistoricalRFCounters	Representa los contadores 802.11 recogidos para las interfaces de radio de los puntos de acceso inalámbricos ligeros en las últimas 24 horas con un intervalo de 15 minutos (por defecto).
HistoricalRFLoadStats	Representa las estadísticas de carga de las interfaces de radio de los puntos de acceso inalámbricos ligeros recogidas en las últimas 24 horas con un intervalo de 15 minutos (por defecto).
HistoricalRFStats	Representa las estadísticas de las interfaces de radio de los puntos de acceso inalámbricos ligeros recogidas en las últimas 24 horas con un intervalo de 15 minutos (por defecto).
HistoricalWLCCPUUtilizations	Representa la información histórica de utilización de la CPU recopilada de los controladores WLAN en las últimas 24 horas.
HistoricalWLCMemUtilizations	Representa la información histórica de utilización de la memoria recopilada de los controladores WLAN en las últimas 24 horas.
InventoryDetails	Proporciona una vista agregada de toda la información de inventario disponible para el dispositivo.
JobSummary	Proporciona una vista consolidada del último estado de todos los trabajos programados.
MacFilterTemplates	Devuelve una lista de plantillas de filtros MAC.
MerakiAccessPoints	Representa la información básica sobre los AP de Meraki.
MerakiDevices	Representa la información básica sobre los dispositivos Meraki.
RFCounters	Representa los contadores 802.11 más recientes de las interfaces de radio de los puntos de acceso inalámbricos ligeros recogidos de los controladores.
RFLoadStats	Representa la información más reciente de las estadísticas de carga de las interfaces de radio de los puntos de acceso inalámbricos ligeros recopilada de los controladores WLAN.
RFStats	Representa la información estadística más reciente de las interfaces de radio de los puntos de acceso inalámbricos ligeros recopilada de los controladores WLAN.
RadioDetails	Representa la información detallada de las interfaces de radio en el punto de acceso ligero.
Radios	Recupera las interfaces de radio de los puntos de acceso ligeros y autónomos.
RogueApAlarms	Representa las alarmas de los puntos de acceso no autorizados. Incluye los atributos básicos de las alarmas y detalles adicionales para las alarmas de Rogue AP.
RogueApDetectionHistory	Representa los datos del historial de detección de Rogue AP.
ServiceDomains	El dominio de servicio o mapa de sitio inalámbrico es un objeto que representa sitios, edificios, pisos y áreas exteriores.

Syslogs	Representa la vista del syslog. Los syslogs son un método estándar para el registro de mensajes tales como eventos del sistema en una red IP.
ThirdpartyAccessPoints	Representa un punto de acceso de terceros. Proporciona atributos del dispositivo como el tipo, la versión, el controlador asociado, el número de clientes asociados, etc.
UserDefinedFieldDefinition	Vista de las definiciones de campo definidas por el usuario del sistema
VDAssociatedAccessPoints	Punto de acceso (autónomo, unificado o de terceros) asociado a un dominio virtual.
VDAssociatedDevices	Dispositivo de red asociado a un dominio virtual.
VDAssociatedDynamicGroups	Representa un grupo de ubicación o definido por el usuario asignado a un dominio virtual actual. Los dispositivos de red asignados a grupos dinámicos se asociarán al dominio virtual automáticamente.
VDAssociatedGroups	Representa un grupo de ubicación o definido por el usuario asignado explícitamente a un dominio virtual actual.
VDAssociatedSiteMaps	Mapa del Sitio asociado a un dominio virtual.
VDAssociatedVirtualElements	Elemento virtual asociado a un dominio virtual.
WLCCPUUtilizations	Última utilización de la CPU recogida de un controlador WLAN.
WLCMemoryUtilizations	Última utilización de la memoria recogida de un controlador WLAN.
WlanControllerDetails	Información detallada sobre un controlador WLAN.
WlanControllers	Información resumida sobre un controlador WLAN.
WlanProfiles	Información sobre las WLAN presentes en los controladores gestionados por este sistema.
WlanTemplates	Información sobre las plantillas WLAN almacenadas en este sistema.

Tabla 27: Descripción de las entidades de la API de Cisco

Anexo 2: Entidades de TUD

ap_traffic

Columna	Descripción
ap_traffic_id	Clave primaria.
bssid_id	Clave foránea de la entidad bssid.
ap_whitelist_id	Clave foránea de la entidad ap_whitelist.
monitor_radio_id	Clave foránea de la entidad monitor_radio.
posted_data_id	Clave foránea de la entidad posted_data.
phy	Tipo de PHY utilizado.
type	Tipo de nodo.
channel	Canal donde está operando la red BSS.
managed_ap	Indica si el nodo está siendo gestionado por un gestor de recursos.
encrypted	Indica si la red BSS está encriptada.
Sssid	SSID de la red.
rss	Intensidad de la señal recibida del nodo.
transmission_rate	Velocidad media de transmisión en Mbps que se utiliza en el canal.
throughput_tx	Rendimiento medio transmitido en el nodo en Mbps.
throughput_rx	Rendimiento medio recibido en el nodo en Mbps.
visible_clients	Número de clientes visibles asociados al AP.
hidden_clients	Número de clientes ocultos asociados al AP.
frames_tx_data	Número de tramas DATA transmitidas con éxito por el nodo.
frames_tx_fail_data	Número total de tramas (incluye fallidas) DATA transmitidas.
frames_rx_data	Número de tramas DATA recibidas con éxito por el nodo.
frames_rx_fail_data	Número total de tramas (incluye fallidas) DATA recibidas.
frames_rx_mgnt	Número de tramas MGNT recibidas con éxito.
frames_rx_ctrl	Número de tramas CTRL recibidas con éxito.
fer_tx_data	Tasa media de error (FER) para las tramas DATA transmitidas.
fer_rx_data	Tasa media de error (FER) para las tramas DATA recibidas.
bw_tx_data_mgnt	Ancho de banda de carga normalizado (BWload) para las tramas DATA y MGNT transmitidas con éxito por el nodo.
bw_tx_fail	Ancho de banda de carga normalizado (BWload) para las tramas corruptas fallidas transmitidas por el nodo.
bytes_rx_data	Número de bytes recibidos (sólo tramas DATA).
capture_quality_mngt	Calidad de la captura evaluada a partir del incremento medio en el contador utilizado para la nueva transmisión de todos los no QMF.
capture_quality_data	Calidad de la captura evaluada a partir del incremento medio en el contador utilizado para la nueva transmisión de todas las tramas de datos QoS dirigidas, agregadas a través de todas las AC y los clientes asociados.
probe_response	Número de tramas de respuesta a la sonda transmitidas.
authentication_response	Número de tramas de autenticación transmitidas por el nodo.
deauthentication_response	Número de tramas de desautenticación transmitidas por el nodo.

association_response	Número de tramas asociadas transmitidas por el nodo.
reassociation_response	Número de tramas de reasociación transmitidas por el nodo.
dissociation_response	Número de tramas de disociación transmitidas por el nodo.
bandwidth_economy	Indica la eficiencia con la que un nodo está utilizando sus oportunidades de transmisión para transportar su carga de MSDU. Las unidades son Megabytes/unitBW y sólo se aplica a las tramas DATA unicast.
created_at	Fecha de la captura.

Tabla 28: Entidad ap_traffic de TUD

ap_traffic_class

Columna	Descripción
ap_traffic_class_id	Clave primaria.
ap_traffic_id	Clave foránea de la entidad ap_traffic.
bssid_id	Clave foránea de la entidad bssid.
ap_whitelist_id	Clave foránea de la entidad ap_whitelist.
monitor_radio_id	Clave foránea de la entidad monitor_radio.
posted_data_id	Clave foránea de la entidad posted_data.
aci	Categoría del tráfico (BE-0, BK-1 , VI-2, VO-3).
aci_desc	Categoría del tráfico (BE, BK , VI, VO).
aif	El ajuste AIFSN en el nodo AC.
cw_min	El ajuste ECWmin del nodo AC.
cw_max	El ajuste ECWmax del nodo AC.
txop	El ajuste TXOP del nodo AC.
frames_tx_1st	Número de tramas transmitidas por el nodo AC como primer intento.
frames_tx_total	Número total de tramas transmitidas por el nodo AC.
throughput	Rendimiento medio transmitido en el nodo en Mbps.
fer	Tasa media de error (FER) experimentada.
bw_load	Ancho de banda de carga normalizado del nodo AC
bw_access	Ancho de banda de acceso normalizado del nodo AC.
bw_free	Ancho de banda de libre normalizado del nodo AC.
bw_fail	Ancho de banda normalizado consumido por las tramas corruptas transmitidas por el nodo AC.
contention	Contención media de acceso experimentada por el nodo AC.
aef	Factor de eficiencia de acceso experimentado por el nodo AC.
capacity_available_normalized	Capacidad disponible normalizada experimentada por el nodo AC.
capacity_available_mbps	Capacidad disponible en Mbps experimentada por el nodo AC.
capacity_utilization	Utilización de la capacidad experimentada por el nodo AC.
created_at	Fecha de la captura.

Tabla 29: Entidad ap_traffic_class de TUD

channel

Columna	Descripción
channel_id	Clave primaria.
monitor_radio_id	Clave foránea de la entidad monitor_radio.
posted_data_id	Clave foránea de la entidad posted_data.
location_id	Clave foránea de la entidad location.
channel	Número de canal en el que funciona el adaptador WLAN.
transmision_rate	Velocidad media de transmisión (Mbps) que se utiliza en el canal.
noise	Nivel medio de ruido en el canal.
contention	Contención media para el acceso que se experimenta en el canal.
frames_total	Número de tramas capturados en el intervalo de medición.
frames_data	Número de tramas de DATOS capturadas en el canal.
frames_mgnt	Número de tramas MGNT capturadas en el canal.
frames_ctrl	Número de tramas CTRL capturadas en el canal.
frames_fail	Número de tramas corruptas capturadas en el canal.
frames_fail_bad	Número de tramas corruptas en los que los datos han sido predeterminados por el controlador
bw_busy	Ancho de banda ocupado normalizado del canal. Todas las tramas, incluidas las corruptas.
bw_busy_native	Ancho de banda ocupado de los dispositivos que operan de forma nativa en el canal.
bw_busy_qc_native	Ancho de banda consumido por las estaciones en el canal estimado a partir del bw busy ponderado por la captura de calidad para cada estación
bw_busy_qc_ext	Ancho de banda consumido por los AP que no están en el canal estimado a partir de bw busy ponderado por la captura de calidad para cada estación
bw_idle	Ancho de banda inactivo normalizado del canal.
bw_fail	Ancho de banda normalizado perdido por tramas corruptas en el canal.
bw_access_max_ap	El máximo ancho de banda de acceso consumido por cualquier AP detectado desde el canal.
bw_access_avg_ap	Acceso medio al ancho de banda para los STA nativos en un canal
bw_free_calculate	Ancho de banda libre.
bw_data	Ancho de banda normalizado consumido por las tramas de datos.
bw_mgnt	Ancho de banda normalizado consumido por las tramas de gestión.
bw_ctrl	Ancho de banda normalizado consumido por las tramas de control.
bw_ap	Ancho de banda normalizado consumido por las transmisiones AP.
bw_client	Ancho de banda normalizado consumido por las transmisiones de los clientes.
bw_adhoc	Ancho de banda normalizado consumido por las transmisiones IBSS.
bw_mesh	Ancho de banda normalizado consumido por las transmisiones

	MBSS.
bw_unassociated_clients	Ancho de banda normalizado consumido por transmisiones de clientes no resueltas.
total_devices	Dispositivos totales.
infrastructure_aps	Número de redes BSS encontradas.
infrastructure_clients	Número de clientes asociados a una red BSS.
unassociated_clients	Número de nodos de clientes no resueltos (clientes no asociados a un AP) encontrados.
adhoc_networks	Número de redes IBSS (ad hoc) encontradas.
adhoc_devices	Número de nodos IBSS encontrados.
mesh_devices	Número de redes MBSS (malla) encontradas.
mesh_networks	Número de nodos MBSS encontrados.
mesh_devices	Número de nodos ocultos descubiertos.
hidden_devices	Número de nodos AP ocultos descubiertos.
hidden_aps	Número total de nodos ocultos descubiertos
hidden_clients	Número de nodos ocultos del cliente descubiertos.
hidden_adhoc_devices	Número de nodos ocultos IBSS descubiertos.
hidden_mesh_devices	Número de nodos ocultos MBSS descubiertos.
clients_channel	Número total de clientes en ese canal específico.
aps_channel	Número total de AP en ese canal específico.
aps_rss_combined	La suma del nivel de potencia recibida de todos los AP detectados desde ese canal.
aps_rss_factor	Factor para representar la interferencia de los AP cocanal o cercanos al canal.
survey_dump_noise	Nivel de ruido en dBm según el volcado de la encuesta.
survey_dump_active_time	Tiempo activo (tiempo de medición) en ms según el volcado de la encuesta.
survey_dump_busy_time	Tiempo de recepción ocupado (basado en la energía) en ms según el volcado de la encuesta
survey_dump_receive_time	Tiempo de recepción de 802.11 en ms según el volcado de la encuesta.
survey_dump_transmit_time	Tiempo de transmisión de 802.11 en ms según el volcado de la encuesta.
rss_40Integer	Número de AP activos en los que -40 dBm <= RSS < 50 dBm.
rss_50Integer	Número de AP activos en los que -50 dBm <= RSS < 60 dBm.
rss_60Integer	Número de AP activos en los que -60 dBm <= RSS < 70 dBm.
rss_70Integer	Número de AP activos en los que -70 dBm <= RSS < 80 dBm.
rss_80Integer	Número de AP activos en los que -80 dBm <= RSS < 90 dBm.
rss_90Integer	Número de AP activos en los que -90 dBm <= RSS < 100 dBm.
mean_interframe	Valor medio del intervalo entre tramas (en milisegundos).
var_interframe	Varianza del intervalo entre tramas (en milisegundos).
created_at	Fecha de la captura.

Tabla 30: Entidad channel de TUD

client_traffic

Columna	Descripción
client_traffic_id	Clave primaria.
client_id	Clave foránea de la entidad client.
client_whitelist_id	Clave foránea de la entidad client_whitelist.
bssid_id	Clave foránea de la entidad bssid.
ap_whitelist_id	Clave foránea de la entidad ap_whitelist.
monitor_radio_id	Clave foránea de la entidad monitor_radio.
posted_data_id	Clave foránea de la entidad posted_data.
phy	Tipo de PHY utilizado.
type	Tipo de nodo.
channel	Canal donde está operando la red BSS.
managed_client	Indica si el client está siendo gestionado por un gestor de recursos.
encrypted	Indica si la red BSS está encriptada.
sssid	SSID de la red.
rss	Intensidad de la señal recibida del nodo.
transmission_rate	Velocidad media de transmisión en Mbps que se utiliza en el canal.
reception_rate	Velocidad media de recepción en Mbps que se utiliza en el canal.
bandwidth_economy_tx	Indica la eficiencia con la que un nodo está utilizando sus oportunidades de transmisión.
bandwidth_economy_rx	Indica la eficiencia con la que un nodo está utilizando sus oportunidades de recepción.
throughput_tx	Rendimiento medio transmitido en el nodo en Mbps.
throughput_rx	Rendimiento medio recibido en el nodo en Mbps.
frames_tx_data	Número de tramas DATA transmitidas con éxito por el nodo.
frames_tx_fail_data	Número total de tramas (incluye fallidas) DATA transmitidas.
frames_rx_data	Número de tramas DATA recibidas con éxito por el nodo.
frames_rx_fail_data	Número total de tramas (incluye fallidas) DATA recibidas.
frames_rx_mgnt	Número de tramas MGNT recibidas con éxito.
frames_rx_ctrl	Número de tramas CTRL recibidas con éxito.
fer_tx_data	Tasa media de error (FER) para las tramas DATA transmitidas.
fer_rx_data	Tasa media de error (FER) para las tramas DATA recibidas.
bw_tx_data_mgnt	Ancho de banda de carga normalizado (BWload) para las tramas DATA y MGNT transmitidas con éxito por el nodo.
bw_tx_fail	Ancho de banda de carga normalizado (BWload) para las tramas corruptas fallidas transmitidas por el nodo.
bytes_rx_data	Número de bytes recibidos (sólo tramas DATA).
probe_request	Número de tramas de petición a la sonda transmitidas.
authentication_response	Número de tramas de autenticación transmitidas por el nodo.
deauthentication_response	Número de tramas de desautenticación transmitidas por el nodo.
association_response	Número de tramas asociadas transmitidas por el nodo.
reassociation_response	Número de tramas de reasociación transmitidas por el nodo.
dissociation_response	Número de tramas de disociación transmitidas por el nodo.
created_at	Fecha de la captura.

Tabla 31: Entidad client_traffic de TUD

client

Columna	Descripción
client_id	Clave primaria.
client_mac	MAC del cliente.
updated_at	Fecha de última captura.
created_at	Fecha de captura inicial.

Tabla 32: Entidad client de TUD

bssid

Columna	Descripción
bssid_id	Clave primaria.
ap_whitelist_id	Clave foránea de la entidad client.
ssid	Clave foránea de la entidad client_whitelist.
ap_mac	MAC del AP.
updated_at	Fecha de última captura.
created_at	Fecha de captura inicial.

Tabla 33: Entidad bssid de TUD

monitor_radio

Columna	Descripción
monitor_radio_id	Clave primaria.
name	Nombre.
location	Localización física.
scenario	Tipo de escenario.

Tabla 34: Entidad monitor_radio de TUD

Anexo 3: Código conexión con la API

```
from asyncio.windows_events import NULL
import dateutil.parser
import datetime
import json
import mysql.connector
import requests
import time
import urllib3

urllib3.disable_warnings()

db = mysql.connector.connect(
   host='localhost', user='root', password='********', database='upv')
if db.is_connected():
   cursor = db.cursor()
   cursor.execute(
      "SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA=' upv '")
   tables = list(map(",".join, cursor.fetchall()))
   api_tables = list(map(lambda table: table.title().replace('_', '') if table.endswith(('cli_template',
'definition', 'history',
                'onboarding_profile', 'specification', 'summary')) else table.title().replace('_', '') + 's',
tables))
   root_url = 'https:// ********/webacs/api/v4/data/'
   for table, api_table in zip(tables, api_tables):
      if 'Rf' or 'VdA' or 'WlcM' or 'WlcCpu' in api_table:
         if 'Rf' in api_table:
            api_table = api_table.replace('Rf', 'RF')
         elif 'VdA' in api_table:
            api_table = api_table.replace('VdA', 'VDA')
         elif 'WlcCpu' in api_table:
            api_table = api_table.replace('WlcCpu', 'WLCCPU')
         elif 'WlcM' in api_table:
            api_table = api_table.replace('WlcM', 'WLCM')
      url = root_url + api_table + '.json' + '?.full=true'
      response = requests.request('GET', url, auth=(
         '********', '********'), verify=False)
      time.sleep(2)
      if response.status_code == 200:
         json_response = json.loads(response.text)
         try:
            entities = json_response['queryResponse']['entity']
            data = []
            cursor.execute(
```

```
            f"SELECT COLUMN_NAME,DATA_TYPE FROM INFORMATION_SCHEMA.COLUMNS WHERE
TABLE_NAME='{table}' AND TABLE_SCHEMA='upv'")
            table_columns = cursor.fetchall()
            id_column = table_columns[0][0]
            table_columns = table_columns[1:]
            for entity in entities:
               values = []
               no_null_table_columns = []
               dtotype = entity['@dtoType']
               if table.startswith('rf'):
                  dtotype = dtotype.replace('rF', 'rf')
               elif table.startswith('wlc'):
                  if table == 'wlc_memory_utilization':
                     dtotype = dtotype.replace('wLC', 'wlc')
                  else:
                     dtotype = dtotype.replace('wLCCPU', 'wlccpu')
               values.append(entity[dtotype]['@id'])
               for column in table_columns:
                  if column[1] in ('bigint', 'bit', 'double', 'float', 'int'):
                     try:
                        if column[0] == 'hotspot2Enable':
                           values.append(
                              entity[dtotype][column[0]]['hotSpot2Enabled'])
                        else:
                           values.append(entity[dtotype][column[0]])
                           if column[0] == 'channelNumber' and table != 'rogue_ap_detection_history':
                              values[-1] = int(values[-1][1:])
                        no_null_table_columns.append(column[0])
                     except:
                        values.append(NULL)
                  elif column[1] == 'datetime':
                     try:
                        date = dateutil.parser.isoparse(
                           entity[dtotype][column[0]])
                        if api_table in ('ApOnboardingProfile', 'GuestUsers', 'HistoricalRFCounters',
'HistoricalRFLoadStats',
                                         'HistoricalRFStats', 'RFCounters', 'RFLoadStats', 'RFStats', 'Syslogs',
'WlanControllerDetails'):
                           date = date - datetime.timedelta(hours=4)
                        values.append(date.strftime(
                           '%Y-%m-%d %H:%M:%S'))
                        no_null_table_columns.append(column[0])
                     except:
                        values.append('0000-00-00 00:00:00')
                  elif column[1] == 'json':
                     if column[0] == 'cpus':
                        column = ('CPUs',) + column[1:]
                     try:
```

```
                values.append(json.dumps(
                    entity[dtotype][column[0]]))
                no_null_table_columns.append(column[0])
            except:
                values.append('{}')
        elif column[1] == 'text':
            try:
                values.append(
                    entity[dtotype][column[0]]['address'])
                no_null_table_columns.append(column[0])
            except:
                try:
                    values.append(
                        entity[dtotype][column[0]]['octets'])
                    no_null_table_columns.append(column[0])
                except:
                    try:
                        values.append(
                            entity[dtotype][column[0]]['value'])
                        no_null_table_columns.append(column[0])
                    except:
                        try:
                            values.append(
                                entity[dtotype][column[0]])
                            no_null_table_columns.append(
                                column[0])
                        except:
                            values.append('')
    try:
        id = values[0]
        cursor.execute(
            f"SELECT * FROM {table} WHERE {id_column} = '{id}'")
        records = cursor.fetchone()[1:]
        values = values[1:]
        update_query = f"UPDATE {table} SET"
        check_update = False
        for i in range(len(records)):
            value = values[i]
            record = records[i]
            table_column = table_columns[i][0]
            no_null_table_column = no_null_table_columns[i]
            if type(record) == "<class 'datetime.datetime'>":
                record = record.strftime('%Y-%m-%d %H:%M:%S')
            if value != record and table_column == no_null_table_column:
                update_query = f"{update_query} {table_column} = '{values[i]}',"
                check_update = True
        if check_update == True:
            update_query = update_query[:-1]
```

```
                update_query = f"{update_query} WHERE {id_column} = '{id}'"
                try:
                    cursor.execute(update_query)
                    db.commit()
                except:
                    print(
                        f'No se pudo actualizar la query {update_query}')
            except:
                data.append(tuple(values))
        data = str(data)[1:-1]
        insert_queries = f'INSERT INTO {table} VALUES {data}'
        try:
            cursor.execute(insert_queries)
            db.commit()
        except:
            print('No se insertó ninguna query')
    except:
        print(
            'ERROR, la repuesta no tiene el formato JSON correcto para realizar las consultas')
elif response.status_code == 401:
    print('ERROR, credenciales de la API incorrectas')
elif response.status_code == 404:
    print(
        f'ERROR, no ha sido posible consultar información sobre la entidad {table}')
else:
    print('ERROR, no se ha podido conectar con la DB')
```

Anexo 4: Entidad client_stat

Columna	Desripción
bytesReceived	Número de bytes recibidos durante la sesión
bytesSent	Número de bytes enviados durante la sesión
collectionTime	La hora de finalización de la recogida de este registro, medida en milisegundos.
dataRate	Tasa de datos de lectura, medida en Mbps.
dataRetries	Número de reintentos de datos durante la sesión.
macAddress	MAC del cliente.
packetsReceived	Número de paquetes recibidos durante la sesión.
packetsSent	Número de paquetes enviados durante la sesión.
raPacketsDropped	Número de paquetes IPv6 RA perdidos durante la sesión.
rssi	El indicador de intensidad de la señal recibida detectado por el punto de acceso al que está asociado el cliente, medido en dBm.
rtsRetries	Número de reintentos de RTS durante la sesión.
rxBytesDropped	Número de bytes de recepción perdidos durante la sesión
rxPacketsDropped	Número de paquetes de recepción perdidos durante la sesión
snr	Relación señal/ruido de la sesión del cliente detectado por el punto de acceso al que está asociado el cliente.
txBytesDropped	Número de bytes de transmisión perdidos durante la sesión.
txPacketsDropped	Número de paquetes de transmisión perdidos durante la sesión.

Tabla 35: Entidad client_stat de la API de Cisco

Anexo 5: Código de la clase K-Means

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
class Kmeans:
    def __init__(self, x):
        self.X = x
        self.silhouette_score_list = []
        self.max_silhouette_score = 0
        self.n_clusters = 0
        self.clusters_list = []

    def silhouette_method(self):
        silhouette_avg = []
        for i in range(2, 11):
            kmeans = KMeans(
                n_clusters=i, init="k-means++", max_iter=300, n_init=10, random_state=0
            )
            kmeans.fit(self.X)
            try:
                silhouette_avg.append(silhouette_score(self.X, kmeans.labels_))
            except:
                pass
        if silhouette_avg:
            self.silhouette_score_list = [round(s, 3) for s in silhouette_avg]
            self.max_silhouette_score = max(self.silhouette_score_list)
            self.n_clusters = (
                self.silhouette_score_list.index(self.max_silhouette_score) + 2
            )

    def fit(self):
        self.silhouette_method()
        if self.n_clusters > 1:
            kmeans = KMeans(
                n_clusters=self.n_clusters,
                init="k-means++",
                max_iter=300,
                n_init=10,
                random_state=0,
            )
            self.clusters_list = kmeans.fit_predict(self.X) + 1
```

Anexo 6: Código de la clase DBSCAN

```
import numpy as np
from kneed import KneeLocator
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics import silhouette_score


class Dbscan:

    def __init__(self, x, c):
        self.X = x
        self.min_pts = 2 * c
        self.distances = 0
        self.eps = 0
        self.noise_prob = 0
        self.noise_points = []
        self.n_clusters = 0
        self.silhouette_score = 0
        self.clusters_list = []

    def hyperparameters(self):
        nbrs = NearestNeighbors(n_neighbors=self.min_pts).fit(self.X)
        distances, indices = nbrs.kneighbors(self.X)
        distances = np.sort(distances, axis=0)
        self.distances = distances[:, 1].tolist()
        y = list(set(self.distances))
        y.sort()
        x = np.arange(len(y))
        sensitivity = [1, 3, 5, 10, 100, 250, 500]
        eps_list = []
        for s in sensitivity:
            kl = KneeLocator(
                x,
                y,
                S=s,
                curve="convex",
                direction="increasing",
                interp_method="polynomial",
            )
            if kl.knee is None:
                break
            eps_list.append(y[kl.knee])
        eps_list = list(set(eps_list))
        eps_list.sort()
```

```
    eps_list = [round(eps, 3) for eps in eps_list]
    self.eps = max(eps_list)

  def fit(self):
    self.hyperparameters()
    if self.eps > 0:
      dbscan = DBSCAN(eps=self.eps, min_samples=self.min_pts)
      dbscan.fit(self.X)
      self.noise_prob = round(
        (sum(1 for i in dbscan.labels_ if i < 0) /
         len(dbscan.labels_)) * 100, 2
      )
      if self.noise_prob != 100.00 and len(list(set(dbscan.labels_))) > 2:
        self.noise_points = [
          i for i in range(len(dbscan.labels_)) if dbscan.labels_[i] == -1
        ]
        self.X = np.delete(self.X, self.noise_points, axis=0).reshape(
          -1, np.shape(self.X)[1]
        )
        no_noise_labels = np.delete(
          dbscan.labels_, self.noise_points, axis=0
        ).tolist()
        self.n_clusters = len(list(set(no_noise_labels)))
        self.silhouette_score = silhouette_score(
          self.X, no_noise_labels)
        self.clusters_list = [x+1 for x in no_noise_labels]
```

Anexo 7: Código del dashboard

```
from dash import Dash, Input, Output, dcc, html, exceptions, State
import dash_bootstrap_components as dbc
from datetime import date, timedelta, datetime
import pandas as pd
from sqlalchemy import create_engine
import plotly.express as px
from sklearn.preprocessing import StandardScaler
from classes.dbscan import Dbscan
from classes.kmeans import Kmeans
from plotly.subplots import make_subplots
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
import numpy as np


def cleanDataframe(df):
    df = df.replace("NULL", float("NaN"))
    df = df.dropna()
    return df


def convertDateColumn(df, date_column):
    df["hour"] = df[date_column].dt.hour
    df["day_of_week"] = df[date_column].dt.dayofweek
    df["day_of_week"] = df["day_of_week"].replace(
        [0, 1, 2, 3, 4, 5, 6],
        [
            "Lunes",
            "Martes",
            "Miércoles",
            "Jueves",
            "Viernes",
            "Sábado",
            "Domingo",
        ],
    )
    return df


def getDataframes():
    cnx = create_engine(
        "mysql+pymysql://root:********@localhost:3306/tud").connect()
    df_tud = pd.read_sql(
        "SELECT
bw_busy,bw_fail,bw_free_calculated,bw_idle,frames_ctrl,frames_data,frames_fail,frames_mgnt,infr
```

```
astructure_clients,hidden_clients,clients_channel,aps_channel,created_at,scenario FROM Channel
INNER JOIN Monitor_radio ON Channel.monitor_radio_id=Monitor_radio.monitor_radio_id",
      cnx,
   )
   df_tud = cleanDataframe(df_tud)
   df_tud = convertDateColumn(df_tud, "created_at")
   cnx.close()
   cnx = create_engine(
      "mysql+pymysql://root:**********@localhost:3306/upv").connect()
   df_upv = pd.read_sql(
      "SELECT dataRate,dataRetries,rssi,snr,collectionTime FROM client_stat",
      cnx,
   )
   df_upv = cleanDataframe(df_upv)
   df_upv["collectionTime"] = pd.to_datetime(
      df_upv['collectionTime'], unit='ms')
   df_upv = convertDateColumn(df_upv, "collectionTime")
   cnx.close()
   return df_tud, df_upv


def show_kmeans_silhouette_graph(silhouette_score_list):
   silhouette_graph = dcc.Graph(
      id="silhouette_line",
      figure=px.line(
         pd.DataFrame(
            dict(
               x=list(
                  range(
                     2,
                     len(silhouette_score_list)
                     + 2,
                  )
               ),
               y=silhouette_score_list,
            )
         ),
         x="x",
         y="y"
      ).update_layout(
         title_text="Análisis de siluetas",
         title_x=0.5,
         xaxis_title="Número de clusters",
         yaxis_title="Coeficiente de siluetas",
         showlegend=False,
      )
   )
   return silhouette_graph
```

```
def show_dbscan_eps_noise_graphs(distances, eps, min_pts, noise_points, samples):
    y = list(set(distances))
    y.sort()
    x = np.arange(len(y))
    eps_graph = dcc.Graph(
        id="eps_line",
        figure=px.line(
            x=x, y=y
        ).update_layout(
            title_text=f"Muestras ordenadas por distancia al k-{min_pts} vecino más cercano",
            title_x=0.5,
            xaxis_title="Muestras",
            yaxis_title="K-distancia (eps)",
            showlegend=False,
        ).add_hline(y=eps, line_width=3, line_dash="dash", line_color="red")
    )
    noise_pie_graph = dcc.Graph(
        id="noise_pie",
        figure=px.pie(
            values=[noise_points, samples-noise_points],
            names=['Si', 'No'],
            hole=0.3,
        ).update_layout(
            legend_title_text="Ruido",
        ),
    )
    return eps_graph, noise_pie_graph


def show_tud_cluster_graphs(X, df, features):
    clusters_pie_graph = dcc.Graph(
        id="clusters_pie",
        figure=px.pie(
            df.groupby(["cluster"])
            .cluster.count()
            .reset_index(name="samples"),
            values="samples",
            names="cluster",
            hole=0.3,
        ).update_layout(
            legend_title_text="Cluster",
        ),
    )
    df_graph = df.groupby(["cluster", "scenario"]
                          ).scenario.count().reset_index(name="samples")
    df_graph["percent"] = (df_graph["samples"] /
```

```
                df_graph["samples"].sum()) * 100
    relationship_clusters_scenarios_graph = dcc.Graph(
        id="relationship_clusters_scenarios_bar",
        figure=px.bar(
            df_graph,
            x="scenario",
            y="percent",
            color="cluster",
            hover_data=['samples'],
            barmode="group",
        ).update_layout(
            title_text="Relación entre los clusters y escenarios",
            xaxis_title="Escenario",
            yaxis_title="Porcentaje (%)",
            legend_title_text="Cluster",
            title_x=0.5
        ),
    )
    relationship_clusters_features_graph = dcc.Graph(
        id="relationship_clusters_features_scatter_matrix",
        figure=px.scatter_matrix(
            df,
            dimensions=features,
            color="cluster"
        ).update_layout(
            title_text="Relación entre los clusters y las características",
            legend_title_text="Cluster",
            title_x=0.5,
            height=1100
        )
    )
    df_graph = df.loc[:, features]
    df_graph['cluster'] = df['cluster']
    df_graph = df_graph.groupby(
        "cluster").mean().reset_index()
    means_figure = make_subplots(
        rows=len(features), cols=1, start_cell="bottom-left")
    counter = 0
    for column in df_graph.columns:
        if column != 'cluster':
            means_figure.add_bar(x=df_graph.loc[:, 'cluster'],
                                 y=df_graph.loc[:, column], name=column, row=1+counter, col=1)
            counter += 1
    means_graph = dcc.Graph(id="clusters_features_means_bar",
figure=means_figure.update_layout(
        title_text="Media de las características por cluster", title_x=0.5, xaxis_title="Cluster",
        legend_title_text="Características"))
    lda = LinearDiscriminantAnalysis()
```

```
    lda.fit_transform(X, df['cluster'])
    indexLD = []
    for i in range(1, len(lda.explained_variance_ratio_) + 1):
        indexLD.append('%d' % i)
    lda_variances_graph = dcc.Graph(id="lda_variances_bar",  figure=px.bar(
        x=list(map(str, list(range(1, len(lda.explained_variance_ratio_) + 1)))),
        y=lda.explained_variance_ratio_,
        barmode="group").update_layout(
        title_text="Varianzas de los discriminantes lineales",
        xaxis_title="Discriminante lineal",
        yaxis_title="Varianza",
        title_x=0.5
    ))
    df_graph = pd.DataFrame(lda.scalings_, columns=list(
        map(str, list(range(1, len(lda.explained_variance_ratio_) + 1)))), index=features)
    lda_coefficients_graph = dcc.Graph(id="lda_coefficients_bar",  figure=px.bar(
        df_graph,
        x=list(df_graph.index),
        y=df_graph.columns,
        barmode="group").update_layout(
        title_text="Coeficientes de los discriminantes lineales para cada característica",
        xaxis_title="Característica",
        yaxis_title="Coeficiente",
        legend_title_text="Discrimante lineal",
        title_x=0.5
    ))
    lda_coefficients_graphs = dbc.Row(
        [dbc.Col(lda_variances_graph, md=6), dbc.Col(lda_coefficients_graph, md=6)])
    return clusters_pie_graph, relationship_clusters_scenarios_graph,
relationship_clusters_features_graph, means_graph, lda_coefficients_graphs


def show_upv_cluster_graphs(X, df, features):
    clusters_pie_graph = dcc.Graph(
        id="clusters_pie",
        figure=px.pie(
            df.groupby(["cluster"])
            .cluster.count()
            .reset_index(name="samples"),
            values="samples",
            names="cluster",
            hole=0.3,
        ).update_layout(
            legend_title_text="Cluster",
        ),
    )
    relationship_clusters_features_graph = dcc.Graph(
        id="relationship_clusters_features_scatter_matrix",
```

```
    figure=px.scatter_matrix(
        df,
        dimensions=features,
        color="cluster"
    ).update_layout(
        title_text="Relación entre los clusters y las características",
        legend_title_text="Cluster",
        title_x=0.5,
        height=1100
    )
  )
  df_graph = df.loc[:, features]
  df_graph['cluster'] = df['cluster']
  df_graph = df_graph.groupby(
    "cluster").mean().reset_index()
  means_figure = make_subplots(
    rows=len(features), cols=1, start_cell="bottom-left")
  counter = 0
  for column in df_graph.columns:
    if column != 'cluster':
      means_figure.add_bar(x=df_graph.loc[:, 'cluster'],
                  y=df_graph.loc[:, column], name=column, row=1+counter, col=1)
      counter += 1
  means_graph = dcc.Graph(id="clusters_features_means_bar",
figure=means_figure.update_layout(
    title_text="Media de las características por cluster", title_x=0.5, xaxis_title="Cluster",
    legend_title_text="Características"))
  lda = LinearDiscriminantAnalysis()
  lda.fit_transform(X, df['cluster'])
  indexLD = []
  for i in range(1, len(lda.explained_variance_ratio_) + 1):
    indexLD.append('%d' % i)
  lda_variances_graph = dcc.Graph(id="lda_variances_bar",  figure=px.bar(
    x=list(map(str, list(range(1, len(lda.explained_variance_ratio_) + 1)))),
    y=lda.explained_variance_ratio_,
    barmode="group").update_layout(
    title_text="Varianzas de los discriminantes lineales",
    xaxis_title="Discriminante lineal",
    yaxis_title="Varianza",
    title_x=0.5
  ))
  df_graph = pd.DataFrame(lda.scalings_, columns=list(
    map(str, list(range(1, len(lda.explained_variance_ratio_) + 1)))), index=features)
  lda_coefficients_graph = dcc.Graph(id="lda_coefficients_bar",  figure=px.bar(
    df_graph,
    x=list(df_graph.index),
    y=df_graph.columns,
    barmode="group").update_layout(
```

```
        title_text="Coeficientes de los discriminantes lineales para cada característica",
        xaxis_title="Característica",
        yaxis_title="Coeficiente",
        legend_title_text="Discrimante lineal",
        title_x=0.5
    ))
    lda_coefficients_graphs = dbc.Row(
        [dbc.Col(lda_variances_graph, md=6), dbc.Col(lda_coefficients_graph, md=6)])
    return clusters_pie_graph, relationship_clusters_features_graph, means_graph,
lda_coefficients_graphs


df_tud, df_upv = getDataframes()
app = Dash(
    external_stylesheets=[dbc.themes.LUX, dbc.icons.BOOTSTRAP],
    suppress_callback_exceptions=True,
)

SIDEBAR_STYLE = {
    "position": "fixed",
    "top": 0,
    "left": 0,
    "bottom": 0,
    "width": "23rem",
    "padding": "2rem 1rem",
    "background-color": "#f8f9fa",
    "overflow-y": "scroll",
}

CONTENT_STYLE = {
    "margin-left": "23rem",
    "margin-right": "1rem",
    "padding": "1rem 1rem",
}

sidebar = html.Div(
    [
        html.H2("QoS Analyzer"),
        html.Hr(),
        html.P(
            "Aplicación para analizar la QoS de redes corporativas en diferentes entornos."
        ),
        html.H4("Entorno", className="bi bi-building"),
        dbc.Nav(
            [
                dbc.NavLink("TUD", href="/tud", active="exact"),
                dbc.NavLink("UPV/EHU", href="/upv", active="exact"),
            ],
```

```
        justified=True,
        pills=True,
      ),
      html.Div(id="sidebar_content"),
   ],
   style=SIDEBAR_STYLE,
)


content = html.Div(id="page_content", style=CONTENT_STYLE)

app.layout = html.Div(
   [
      dcc.Location(id="url"),
      dcc.Store(id="memory_data_upv"),
      dcc.Store(id="memory_data_tud"),
      sidebar,
      content,
   ]
)


@ app.callback(
   Output("sidebar_content", "children"),
   Output("page_content", "children"),
   Input("url", "pathname"),
)
def render_content(pathname):
   if pathname in ("/tud", "/upv"):
      date_label = html.H4("Fecha", className="bi bi-calendar3")
      features_label = html.H4("Parámetros", className="bi bi-router")
      analysis_label = html.H4("Análisis", className="bi bi-pencil-square")
      algorithm_label = html.H4("Algoritmo", className="bi bi-cpu")
      if pathname == "/upv":
         calendar = dcc.DatePickerRange(
            id="calendar_upv",
            min_date_allowed=df_upv["collectionTime"].min().strftime(
               "%Y-%m-%d"),
            max_date_allowed=df_upv["collectionTime"].max().strftime(
               "%Y-%m-%d"),
            start_date=df_upv["collectionTime"].min().strftime("%Y-%m-%d"),
            end_date=df_upv["collectionTime"].max().strftime("%Y-%m-%d"),
         )
         features_dropdown = dcc.Dropdown(
            id="features_upv",
            options=df_upv.drop(
               ['hour', 'day_of_week', 'collectionTime'], axis=1).columns,
            multi=True,
```

```
        )
        analysis_dropdown = dcc.Dropdown(
            id="analysis_upv",
            options=[
                "Clusterizar sin timeslots",
                "Clusterizar con timeslots",
            ],
        )
        algorithm_dropdown = dcc.Dropdown(
            id="algorithm_upv",
            options=["DBSCAN", "kMeans"],
        )
        historic_traffic_graph = dcc.Graph(id="historic_line_upv")
        days_hours_traffic_graphs = dbc.Row(
            [
                dbc.Col(
                    dcc.Graph(
                        id="days_bar_upv",
                    ),
                    md=6
                ),
                dbc.Col(
                    dcc.Graph(
                        id="hours_bar_upv",
                    ),
                    md=6
                )
            ]
        )
        execute_ml_button = html.Div(
            [
                dbc.Button(
                    "Ejecutar",
                    id="execute_ml_upv",
                )
            ],
            className="d-grid gap-2",
        )
        spinner = dbc.Spinner(html.Div(id="loading-output_upv"))
        return [
            html.Br(), date_label, calendar, html.Br(), html.Br(), features_label, features_dropdown,
html.Br(
            ), analysis_label, analysis_dropdown, html.Br(), algorithm_label, algorithm_dropdown,
html.Br(),
            execute_ml_button, html.Br(), html.Br(), spinner
        ], [
            historic_traffic_graph,
            days_hours_traffic_graphs,
```

```
        html.Div(id="ml_results_upv")
    ]
elif pathname == "/tud":
    calendar = dcc.DatePickerRange(
        id="calendar_tud",
        min_date_allowed=df_tud["created_at"].min().strftime(
            "%Y-%m-%d"),
        max_date_allowed=df_tud["created_at"].max().strftime(
            "%Y-%m-%d"),
        start_date=df_tud["created_at"].min().strftime("%Y-%m-%d"),
        end_date=df_tud["created_at"].max().strftime("%Y-%m-%d"),
    )
    analysis_dropdown = dcc.Dropdown(
        id="analysis_tud",
        options=[
            "Clusterizar sin timeslots",
            "Clusterizar con timeslots",
        ],
    )
    features_dropdown = dcc.Dropdown(
        id="features_tud",
        options=df_tud.drop(
            ['scenario', 'hour', 'day_of_week', 'created_at'], axis=1).columns,
        multi=True
    )
    algorithm_dropdown = dcc.Dropdown(
        id="algorithm_tud",
        options=["DBSCAN", "kMeans"],
    )
    scenarios_traffic_graphs = dbc.Row(
        [
            dbc.Col(
                dcc.Graph(
                    id="scenarios_pie_tud",
                ),
                md=5
            ),
            dbc.Col(
                dcc.Graph(
                    id="historic_line_tud",
                ),
                md=7
            )
        ]
    )
    days_hours_traffic_graphs = dbc.Row(
        [
            dbc.Col(
```

```
                dcc.Graph(
                    id="days_bar_tud",
                ),
                md=6
            ),
            dbc.Col(
                dcc.Graph(
                    id="hours_bar_tud",
                ),
                md=6
            )
        ]
    )
    execute_ml_button = html.Div(
        [
            dbc.Button(
                "Ejecutar",
                id="execute_ml_tud",
            )
        ],
        className="d-grid gap-2",
    )
    spinner = dbc.Spinner(html.Div(id="loading-output_tud"))
    return [
        html.Br(), date_label, calendar, html.Br(), html.Br(), features_label, features_dropdown,
html.Br(
        ), analysis_label, analysis_dropdown, html.Br(), algorithm_label, algorithm_dropdown,
html.Br(),
        execute_ml_button, html.Br(), html.Br(), spinner
    ],  [
        scenarios_traffic_graphs,
        days_hours_traffic_graphs,
        html.Div(id="ml_results_tud")
    ]
else:
    raise exceptions.PreventUpdate


@ app.callback(
    Output("memory_data_tud", "data"),
    Output("scenarios_pie_tud", "figure"),
    Output("historic_line_tud", "figure"),
    Output("days_bar_tud", "figure"),
    Output("hours_bar_tud", "figure"),
    State("url", "pathname"),
    Input("calendar_tud", "start_date"),
    Input("calendar_tud", "end_date"),
)
```

```
def show_tud_traffic(pathname, start_date, end_date):
  if pathname == "/tud":
    df = df_tud.loc[
      (df_tud["created_at"] >= start_date)
      & (
        df_tud["created_at"]
        <= (datetime.strptime(end_date, "%Y-%m-%d") + timedelta(days=1))
      )
    ]
    df["scenario"] = df["scenario"].replace(
      ["Classrooms&Labs", "Labs&Offices", "Libraries", "Laboratories"],
      ["Clases y laboratorios", "Laboratorios y secretarías",
        "Bibliotecas", "Laboratorios"],
    )
    scenario_figure = px.pie(
      df.groupby(["scenario"]).scenario.count(
      ).reset_index(name="samples"),
      values="samples",
      names="scenario",
      hole=0.3,
    ).update_layout(legend_title_text="Escenario")
    historic_traffic_figure = px.line(
      df.groupby(
        [pd.Grouper(key="created_at", freq="H"), "scenario"])
      .scenario.count()
      .reset_index(name="samples"),
      x="created_at",
      y="samples",
      color="scenario",
    ).update_layout(
      title_text="Histórico del tráfico",
      xaxis_title="Fecha",
      yaxis_title="Muestras",
      legend_title_text="Escenario",
      title_x=0.5,
    )
    df_graph = df.groupby(["day_of_week", "scenario"]
                ).scenario.count().reset_index(name="samples")
    df_graph["percent"] = (df_graph["samples"] /
                 df_graph["samples"].sum()) * 100
    days_traffic = px.bar(
      df_graph,
      x="day_of_week",
      y="percent",
      hover_data=['samples'],
      color="scenario",
      barmode="group",
    ).update_layout(
```

```
        title_text="Tráfico por día de la semana",
        xaxis_title="Día de la semana",
        yaxis_title="Porcentaje (%)",
        legend_title_text="Escenario",
        title_x=0.5,
    ).update_xaxes(
        categoryarray=[
            "Lunes",
            "Martes",
            "Miércoles",
            "Jueves",
            "Viernes",
            "Sábado",
            "Domingo",
        ],
        categoryorder="array",
    )
    df_graph = df.groupby(["hour", "scenario"]
                          ).scenario.count().reset_index(name="samples")
    df_graph["percent"] = (df_graph["samples"] /
                           df_graph["samples"].sum()) * 100
    hours_traffic = px.bar(
        df_graph,
        x="hour",
        y="percent",
        color="scenario",
        hover_data=['samples'],
        barmode="group",
    ).update_layout(
        title_text="Tráfico por hora",
        xaxis_title="Hora",
        yaxis_title="Porcentaje (%)",
        legend_title_text="Escenario",
        title_x=0.5,
    )
    return (
        df.drop(columns=["created_at", "day_of_week"]).to_json(
            date_format="iso", orient="split"
        ),
        scenario_figure,
        historic_traffic_figure,
        days_traffic,
        hours_traffic
    )
else:
    raise exceptions.PreventUpdate
```

```
@ app.callback(
   Output("memory_data_upv", "data"),
   Output("historic_line_upv", "figure"),
   Output("days_bar_upv", "figure"),
   Output("hours_bar_upv", "figure"),
   State("url", "pathname"),
   Input("calendar_upv", "start_date"),
   Input("calendar_upv", "end_date"),
)
def show_upv_traffic(pathname, start_date, end_date):
   if pathname == "/upv":
      df = df_upv.loc[
         (df_upv["collectionTime"] >= start_date)
         & (
            df_upv["collectionTime"]
            <= (datetime.strptime(end_date, "%Y-%m-%d") + timedelta(days=1))
         )
      ]
      df_graph = df.groupby(
         [pd.Grouper(key="collectionTime", freq="H")]).count().reset_index()
      df_graph["samples"] = df_graph["rssi"]
      historic_traffic_figure = px.line(
         df_graph,
         x="collectionTime",
         y="samples"
      ).update_layout(
         title_text="Histórico del tráfico",
         xaxis_title="Fecha",
         yaxis_title="Muestras",
         title_x=0.5,
      )
      df_graph = df.groupby(["day_of_week"]).count().reset_index()
      df_graph["samples"] = df_graph["rssi"]
      df_graph["percent"] = (df_graph["samples"] /
                             df_graph["samples"].sum()) * 100
      days_traffic = px.bar(
         df_graph,
         x="day_of_week",
         y="percent",
         barmode="group",
      ).update_layout(
         title_text="Tráfico por día de la semana",
         xaxis_title="Día de la semana",
         yaxis_title="Porcentaje (%)",
         title_x=0.5,
      ).update_xaxes(
         categoryarray=[
            "Lunes",
```

```
            "Martes",
            "Miércoles",
            "Jueves",
            "Viernes",
            "Sábado",
            "Domingo",
          ],
          categoryorder="array",
      )
      df_graph = df.groupby(["hour"]
                      ).count().reset_index()
      df_graph["samples"] = df_graph["rssi"]
      df_graph["percent"] = (df_graph["samples"] /
                       df_graph["samples"].sum()) * 100
      hours_traffic = px.bar(
          df_graph,
          x="hour",
          y="percent",
          barmode="group",
      ).update_layout(
          title_text="Tráfico por hora",
          xaxis_title="Hora",
          yaxis_title="Porcentaje (%)",
          title_x=0.5,
      )
      return (
          df.drop(columns=["collectionTime", "day_of_week"]).to_json(
              date_format="iso", orient="split"
          ),
          historic_traffic_figure,
          days_traffic,
          hours_traffic
      )
    else:
      raise exceptions.PreventUpdate


@ app.callback(
    Output("loading-output_tud", "children"),
    Output("ml_results_tud", "children"),
    State("algorithm_tud", "value"),
    State("analysis_tud", "value"),
    State("features_tud", "value"),
    State("memory_data_tud", "data"),
    State("url", "pathname"),
    Input("execute_ml_tud", "n_clicks"),
)
def show_tud_ml_results(algorithm, analysis, features, data, pathname, n_clicks):
```

```
  if pathname == "/tud":
    if data is None:
      raise exceptions.PreventUpdate
    else:
      if [algorithm, analysis, features] is None:
        raise exceptions.PreventUpdate
      else:
        df = pd.read_json(data, orient="split").reset_index(drop=True)
        sc_X = StandardScaler()
        if analysis == "Clusterizar sin timeslots":
          X = sc_X.fit_transform(df[features])
          if algorithm == 'DBSCAN':
            dbscan = Dbscan(X, len(features))
            dbscan.fit()
            if dbscan.n_clusters > 1:
              df = df.drop(dbscan.noise_points, axis=0)
              df["cluster"] = list(
                map(str, dbscan.clusters_list))
              eps_graph, noise_pie_graph = show_dbscan_eps_noise_graphs(
                dbscan.distances, dbscan.eps, dbscan.min_pts, len(dbscan.noise_points), len(df))
              clusters_pie_graph, relationship_clusters_scenarios_graph,
relationship_clusters_features_graph, means_graph, lda_coefficients_graphs =
show_tud_cluster_graphs(
                dbscan.X, df, features)
              return None, [eps_graph, dbc.Row([dbc.Col(noise_pie_graph, md=6),
dbc.Col(clusters_pie_graph, md=6)]), relationship_clusters_scenarios_graph,
relationship_clusters_features_graph, means_graph, lda_coefficients_graphs]
            else:
              return None, dbc.Alert("El algoritmo solo obtiene un cluster", color="danger")
          elif algorithm == 'kMeans':
            kmeans = Kmeans(X)
            kmeans.fit()
            if kmeans.n_clusters > 1:
              df["cluster"] = list(
                map(str, kmeans.clusters_list))
              silhouette_graph = show_kmeans_silhouette_graph(
                kmeans.silhouette_score_list)
              clusters_pie_graph, relationship_clusters_scenarios_graph,
relationship_clusters_features_graph, means_graph, lda_coefficients_graphs =
show_tud_cluster_graphs(
                X, df, features)
              return None, [dbc.Row([dbc.Col(silhouette_graph, md=6),
dbc.Col(clusters_pie_graph, md=6)]), relationship_clusters_scenarios_graph,
relationship_clusters_features_graph, means_graph, lda_coefficients_graphs]
            else:
              return None, dbc.Alert("El algoritmo solo obtiene un cluster", color="danger")
        elif analysis == "Clusterizar con timeslots":
          df_ts1 = df[(df.hour > 8) & (df.hour <= 14)]
```

```
            df_ts2 = df[(df.hour > 14) & (df.hour <= 17)]
            df_ts3 = pd.concat([df[(df.hour > 17) & (df.hour <= 24)], df[(
              df.hour >= 0) & (df.hour <= 8)]], axis=0)
            df_ts = [df_ts1, df_ts2, df_ts3]
            X_ts1 = sc_X.fit_transform(df_ts1[features])
            X_ts2 = sc_X.fit_transform(df_ts2[features])
            X_ts3 = sc_X.fit_transform(df_ts3[features])
            X_ts = [X_ts1, X_ts2, X_ts3]
            labels = ['8:00-14:00',
                    '14:00-17:00', '17:00-08:00']
            content = []
            if algorithm == 'DBSCAN':
              for i in range(0, len(df_ts)):
                X = X_ts[i]
                dbscan = Dbscan(X, len(features))
                dbscan.fit()
                if dbscan.n_clusters > 1:
                  df = df_ts[i]
                  df = df.drop(dbscan.noise_points, axis=0)
                  df["cluster"] = list(
                    map(str, dbscan.clusters_list))
                  eps_graph, noise_pie_graph = show_dbscan_eps_noise_graphs(
                    dbscan.distances, dbscan.eps, dbscan.min_pts, len(dbscan.noise_points), len(df))
                  content.append(
                    html.H4(children=labels[i]))
                  content.append(eps_graph, noise_pie_graph)
                  content.append(show_tud_cluster_graphs(
                    dbscan.X, df, features))
                else:
                  content.append(
                    dbc.Alert(f"En el rango {labels[i]} solo obtiene un cluster", color="danger"))
              return None, content
            elif algorithm == 'kMeans':
              for i in range(0, len(df_ts)):
                X = X_ts[i]
                kmeans = Kmeans(X)
                kmeans.fit()
                if kmeans.n_clusters > 1:
                  df = df_ts[i]
                  df["cluster"] = list(
                    map(str, kmeans.clusters_list))
                  silhouette_graph = show_kmeans_silhouette_graph(
                    kmeans.silhouette_score_list)
                  clusters_pie_graph, relationship_clusters_scenarios_graph,
relationship_clusters_features_graph, means_graph, lda_coefficients_graphs =
show_tud_cluster_graphs(X,

df, features)
```

```
                    content.append(html.H4(children=labels[i]))
                    content.append(dbc.Row([dbc.Col(silhouette_graph, md=6), dbc.Col(
                        clusters_pie_graph, md=6)]))
                    content.append(
                        relationship_clusters_scenarios_graph)
                    content.append(
                        relationship_clusters_features_graph)
                    content.append(means_graph)
                    content.append(lda_coefficients_graphs)
                else:
                    content.append(dbc.Alert(
                        f"En el rango {labels[i]} solo obtiene un cluster", color="danger"))
            return None, content
    else:
        raise exceptions.PreventUpdate


@ app.callback(
    Output("loading-output_upv", "children"),
    Output("ml_results_upv", "children"),
    State("algorithm_upv", "value"),
    State("analysis_upv", "value"),
    State("features_upv", "value"),
    State("memory_data_upv", "data"),
    State("url", "pathname"),
    Input("execute_ml_upv", "n_clicks"),
)
def show_upv_ml_results(algorithm, analysis, features, data, pathname, n_clicks):
    if pathname == "/upv":
        if data is None:
            raise exceptions.PreventUpdate
        else:
            if [algorithm, analysis, features] is None:
                raise exceptions.PreventUpdate
            else:
                df = pd.read_json(data, orient="split").reset_index(drop=True)
                sc_X = StandardScaler()
                if analysis == "Clusterizar sin timeslots":
                    X = sc_X.fit_transform(df[features])
                    if algorithm == 'DBSCAN':
                        dbscan = Dbscan(X, len(features))
                        dbscan.fit()
                        if dbscan.n_clusters > 1:
                            df = df.drop(dbscan.noise_points, axis=0)
                            df["cluster"] = list(
                                map(str, dbscan.clusters_list))
                            eps_graph, noise_pie_graph = show_dbscan_eps_noise_graphs(
                                dbscan.distances, dbscan.eps, dbscan.min_pts, len(dbscan.noise_points), len(df))
```

```
                    clusters_pie_graph,  relationship_clusters_features_graph, means_graph,
lda_coefficients_graphs = show_upv_cluster_graphs(
                        dbscan.X, df, features)
                    return None, [eps_graph, dbc.Row([dbc.Col(noise_pie_graph, md=6),
dbc.Col(clusters_pie_graph, md=6)]), relationship_clusters_features_graph, means_graph,
lda_coefficients_graphs]
                else:
                    return None, dbc.Alert("El algoritmo solo obtiene un cluster", color="danger")
            elif algorithm == 'kMeans':
                kmeans = Kmeans(X)
                kmeans.fit()
                if kmeans.n_clusters > 1:
                    df["cluster"] = list(
                        map(str, kmeans.clusters_list))
                    silhouette_graph = show_kmeans_silhouette_graph(
                        kmeans.silhouette_score_list)
                    clusters_pie_graph, relationship_clusters_features_graph, means_graph,
lda_coefficients_graphs = show_upv_cluster_graphs(
                        X, df, features)
                    return None, [dbc.Row([dbc.Col(silhouette_graph, md=6),
dbc.Col(clusters_pie_graph, md=6)]), relationship_clusters_features_graph, means_graph,
lda_coefficients_graphs]
                else:
                    return None, dbc.Alert("El algoritmo solo obtiene un cluster", color="danger")
        elif analysis == "Clusterizar con timeslots":
            df_ts1 = df[(df.hour > 8) & (df.hour <= 14)]
            df_ts2 = df[(df.hour > 14) & (df.hour <= 17)]
            df_ts3 = pd.concat([df[(df.hour > 17) & (df.hour <= 24)], df[(
                df.hour >= 0) & (df.hour <= 8)]], axis=0)
            df_ts = [df_ts1, df_ts2, df_ts3]
            X_ts1 = sc_X.fit_transform(df_ts1[features])
            X_ts2 = sc_X.fit_transform(df_ts2[features])
            X_ts3 = sc_X.fit_transform(df_ts3[features])
            X_ts = [X_ts1, X_ts2, X_ts3]
            labels = ['8:00-14:00',
                      '14:00-17:00', '17:00-08:00']
            content = []
            if algorithm == 'DBSCAN':
                for i in range(0, len(df_ts)):
                    X = X_ts[i]
                    dbscan = Dbscan(X, len(features))
                    dbscan.fit()
                    if dbscan.n_clusters > 1:
                        df = df_ts[i]
                        df = df.drop(dbscan.noise_points, axis=0)
                        df["cluster"] = list(
                            map(str, dbscan.clusters_list))
                        eps_graph, noise_pie_graph = show_dbscan_eps_noise_graphs(
```

```
                    dbscan.distances, dbscan.eps, dbscan.min_pts, len(dbscan.noise_points), len(df))
                clusters_pie_graph, relationship_clusters_features_graph, means_graph,
lda_coefficients_graphs = show_upv_cluster_graphs(dbscan.X,
                                                                                df, features)
                content.append(
                  html.H4(children=labels[i]))
                content.append(eps_graph)
                content.append(
                  dbc.Row([dbc.Col(noise_pie_graph, md=6), dbc.Col(clusters_pie_graph, md=6)]))
                content.append(
                  relationship_clusters_features_graph)
                content.append(means_graph)
                content.append(lda_coefficients_graphs)
              else:
                content.append(
                  dbc.Alert(f"En el rango {labels[i]} solo obtiene un cluster", color="danger"))
            return None, content
          elif algorithm == 'kMeans':
            for i in range(0, len(df_ts)):
              X = X_ts[i]
              kmeans = Kmeans(X)
              kmeans.fit()
              if kmeans.n_clusters > 1:
                df = df_ts[i]
                df["cluster"] = list(
                  map(str, kmeans.clusters_list))
                silhouette_graph = show_kmeans_silhouette_graph(
                  kmeans.silhouette_score_list)
                clusters_pie_graph, relationship_clusters_features_graph, means_graph,
lda_coefficients_graphs = show_upv_cluster_graphs(X,
                                                                                df, features)
                content.append(html.H4(children=labels[i]))
                content.append(dbc.Row([dbc.Col(silhouette_graph, md=6), dbc.Col(
                  clusters_pie_graph, md=6)]))
                content.append(
                  relationship_clusters_features_graph)
                content.append(means_graph)
                content.append(lda_coefficients_graphs)
              else:
                content.append(dbc.Alert(
                  f"En el rango {labels[i]} solo obtiene un cluster", color="danger"))
            return None, content
  else:
    raise exceptions.PreventUpdate


if __name__ == "__main__":
  app.run_server()
```