UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

APLICACIÓN DE TÉCNICAS DE APRENDIZAJE AUTOMÁTICO Y ANALÍTICA PREDICTIVA PARA MEJORAR DESEMPEÑO DE REDES 4G LTE

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

JUAN JOSÉ GUTIÉRREZ TERRAZA

PROFESOR GUÍA:
JOSÉ MIGUEL RUBIO LEÓN

PROFESOR CO-GUÍA:
SEBASTIÁN ALEJANDRO RÍOS PÉREZ

COMISIÓN:
SANDRA LORENA CÉSPEDES UMAÑA

SANTIAGO DE CHILE
2021

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: **JUAN JOSÉ GUTIÉRREZ TERRAZA**
FECHA: 2021
PROF. GUÍA: JOSÉ MIGUEL RUBIO LEÓN

APLICACIÓN DE TÉCNICAS DE APRENDIZAJE AUTOMÁTICO Y ANALÍTICA PREDICTIVA PARA MEJORAR DESEMPEÑO DE REDES 4G LTE

Las redes de telecomunicaciones han generado cambios profundos en la humanidad. Desde sus inicios, con los primeros teléfonos móviles 1G, se abrió un nuevo mundo de posibilidades con la llegada de la comunicación instantánea a larga distancia. Hoy en día, esta tecnología no ha parado de evolucionar, existiendo una gran masificación de redes 4G LTE, y comenzando a desplegarse la nueva tecnología 5G en diferentes países. El 5G trae consigo una serie de desafíos y problemas a resolver para su correcta implementación, de manera de poder alcanzar las altas velocidades y bajísimas latencias prometidas. Existen problemáticas que son transversales para las redes 5G, como para las actuales redes 4G LTE, como por ejemplo, la existencia de bajas de desempeño que pueden llegar a experimentar las celdas de las redes móviles, y la falta de conocimiento del patrón de uso de las antenas a nivel geográfico, para una óptima distribución de los recursos móviles según los requerimientos reales.

El presente trabajo de título se desarrolla bajo el contexto anterior, en donde se utilizan *datasets* del funcionamiento operacional de las redes móviles 4G LTE de Santiago, para poder aplicar diferentes técnicas de aprendizaje automático o *Machine Learning* (ML) capaces de abordar las problemáticas de desempeño de las celdas, e identificar el patrón de uso de las antenas. Es por ello, que se utilizan herramientas de ML como los modelos *XGBoost*, *Random Forest*, entre otros, para poder generar clasificaciones y predicciones respecto a la cantidad de días en que, semanalmente, una celda tiene bajas de desempeño. Por otro lado, se entrenan redes LSTM, para poder generar pronósticos exactos, respecto al comportamiento futuro de una variable de interés, y finalmente, se aplica el algoritmos de modelamiento de tópicos de texto *Latent Dirichlet Allocation* (LDA), para la caracterizaciones de patrón de uso, a nivel temporal y geográfico, de las antenas de la red móvil estudiada.

Para las clasificaciones, se realiza una comparativa entre los algoritmos probados de ML respecto al puntaje ROC AUC, destacando *XGBoost* por sobre los demás, y para las predicciones y pronósticos, se aplican métricas como MAE y RMSE para la comparativa de los desempeños de los modelos. Por último, los resultados del uso de LDA en los datos de la red móvil, logran vislumbrar una diferenciación en el patrón de uso de las antenas, hallando 5 *clusters* diferentes, de los cuales se logra evidenciar comportamientos residenciales y comerciales en diferentes puntos de Santiago.

Cierre de un gran ciclo.

Comenzó gracias a la motivación y cumplimiento de metas que deseaban mis padres, y finalizó con automotivación y ambiciones propias que se fueron forjando en el camino.

Gracias por todo

Agradecimientos

Muchas gracias a todos los que han estado acompañándome en este largo proceso; a mis padres primeramente, que me han dado las herramientas y enseñanzas para poder triunfar en la vida, a mi familia: mi hermana, mis tías y abuelos, que también siempre han estado allí, apoyándome y compartiendo la hora de once casi todos los Domingos.

Mis grupos de amigos: La *Wolfpack*, que cada cierto tiempo se reúne para generar nuevas vivencias, Bachi amigos, con los cuales logramos formar un grupo muy unido, existiendo un eterno ambiente de apoyo, acompañamiento y amistad sincera, mucho cariño para la gente de la *Congre* y les *BachiInges*. Finalmente, los amigos de Eléctrica y de la FCFM en general, que descubrí en mis últimos años de carrera antes de la pandemia, los cuales no fueron muchos, pero si muy valiosos y de excelente calidad de persona. Sé que seguiremos siendo amigos eternamente.

Sin todos ellos, mi proceso universitario no hubiera sido el mismo, y por esto estoy muy agradecido de tenerlos en mi vida.

Les KM

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

NTT Data es una empresa consultora de tecnología a nivel mundial, la cual se caracteriza por proveer innovadoras soluciones relacionadas con transformación digital e inteligencia artificial, capaces de generar valor a sus clientes. Dentro de su variada clientela, en NTT Data Chile, destaca la industria de las Telecomunicaciones, en donde se trabaja con empresas como Entel, Telefónica, WOM y Claro. En este contexto, se puede identificar diversas problemáticas a resolver, pudiendo proponer soluciones capaces de optimizar la operación de estas compañías al aplicar analítica y predicciones, a partir de técnicas de *Machine Learning*.

La industria de Telecomunicaciones ha ido evolucionando constantemente en el siglo actual, dando gigantescos saltos de tecnología al avanzar a nuevas generaciones de dispositivos, como cuando aparecieron los primeros teléfonos inteligentes de pantalla táctil 3G, dejando atrás a los celulares analógicos, debido a la gran gamma de funcionalidades que traía consigo esta nueva ola de equipos. A diferencia del salto de 3G a 4G, en donde básicamente se mejora la velocidad y conectividad, la nueva red 5G, compromete cambios en tal magnitud, que se habla de una revolución tecnológica más que una evolución; no solo mejora la velocidad enormemente en comparación a la generación anterior 4G, sino que, además, se disminuye la latencia considerablemente y se propone la capacidad de generar espectros ajustables a cada tipo de uso, introduciendo categorías a los diferentes tipos de servicio según sus requerimientos de recursos como eMBB (*Mobile BroadBand*), mMTC (*massive machine type comunications*) y urLLC (*ultra-reliable low-latency comunications*).

La nueva tecnología de red 5G, implica una serie de desafíos que deberán ser resueltos para poder entregar un servicio acorde a las nuevas especificaciones de QoS (*Quality of Service*). Actualmente en Chile, ya se está comenzando a instaurar redes 5G, con despliegues de celdas 5G en zonas puntales de Santiago por parte de compañías como Entel y WOM.

Las redes tipo 4G, por su parte, se encuentran ampliamente desplegadas alrededor del mundo. En el 2019 Europa contaba con un 58 % de acceso a esta tecnología y América latina tuvo un 47 %, se espera que para el 2025, Europa cuente con 59 % de acceso, en donde la tecnología 5G tendría un 34 %, mientras que en América Latina se predice que se tenga un 67 % de acceso a 4G y un 7 % a 5G[1]. Es por lo anterior que, si bien es positivo comenzar a pensar en soluciones para los desafíos de la nueva tecnología emergente, es importante señalar que se espera que las redes 4G se mantengan y tomen mayor protagonismo en los años próximos. Sin embargo, es muy probable que las problemáticas que afecten a 4G, existan aún para las redes de la 5ta generación, y a su vez, las soluciones e investigaciones planteadas en torno a esta última, pueden ser de utilidad para las redes 4G, por lo cual es necesario indagar el estado de arte de ambas generaciones, focalizando los esfuerzos en aquellas soluciones que tengan uso de herramientas de *Machine Learning* y metodologías replicables.

Una de las grandes problemáticas que tienen las compañías de Telecomunicaciones es poder asegurar, en todo momento, la mantención de QoS percibido por los usuarios dentro de los estándares establecidos. Esta situación puede verse reflejada en diversos KPIs de la red, pero uno de los indicadores más importantes, es el *Download User Throughput*, debido a su estrecha relación con la experiencia de servicio que percibe el usuario final. Dado lo anterior, se desea entregar herramientas capaces de aportar a dar solución a dicha problemática, en el contexto del incipiente despliegue de redes 5G en Santiago.

En esta memoria se trabajará inicialmente con data de redes 4G LTE, de manera de poder realizar un análisis de la información relevante, para lograr detectar anomalías y bajas de desempeño de las celdas de la red, pudiendo clasificar las celdas según desempeño y realizar predicciones respecto a la cantidad de días en que se cuente con bajas calidades de servicio. Además, teniendo en cuenta que la locación de las celdas 5G toma un rol importante en relación a su desempeño final, se desea incluir en el estudio la componente geográfica de las antenas, de manera de poder identificar patrones en relación con el tipo de uso y sus temporalidades, al utilizar algoritmos novedosos como LDA.

De acuerdo a lo anterior, se desea obtener resultados que permitan que, por un lado, los operadores de red puedan tener acceso a predicciones de desempeño de las celdas, dado el estudio realizado del comportamiento de éstas. Por otro lado, se espera identificar diferentes tipos de uso en las antenas según su localización, permitiendo una posible futura asignación de espectro de recursos de red, acordes a los diferentes tipos de solicitudes de servicio, de manera de mantener el QoS de los servicios asignados a cada espectro.

1.2. Objetivos

1.2.1. Objetivo General

Aplicar técnicas de aprendizaje automático para poder mejorar el desempeño de celdas e identificar el patrón de uso a nivel temporal y geográfico de la operación real de una red 4G LTE.

1.2.2. Objetivos Específicos

1. Realizar una comparativa de diferentes modelos de *Machine Learning*, capaces de realizar clasificación como: *Naive Bayes*, *Random Forest*, redes MLP y *XGBoost*, para poder predecir la identificación de celdas con futuras bajas de desempeño.

2. Evaluar distintos regresores predictivos como *XGBoost Regresor* y *Random Forest Regresor* para generar predicciones respecto a la cantidad de días en situación de bajo de desempeño que pueden tener las celdas de la red 4G LTE.

3. Generar pronósticos de los valores futuros del comportamiento del KPI de la red 4G LTE *Download User Throughput*, a través de la utilización de redes LSTM.

4. Evaluar la factibilidad de utilización de LDA para análisis e identificación de patrones de uso de las antenas de la red de telecomunicaciones a nivel temporal y su relación con sus ubicaciones geográficas.

5. Proyectar metodología para casos de estudio con data 5G, de manera de poder replicar la aplicación de dichas técnicas de aprendizaje automático utilizadas en el presente documento en la nueva tecnología de redes de telecomunicaciones.

1.3. Alcances

El desarrollo de este trabajo de Título, como se mencionó anteriormente, se realizará con bases de datos del funcionamiento de redes móviles 4G LTE de la ciudad de Santiago, presentando resultados de diferentes algoritmos de ML, capaces de generar clasificaciones y pronósticos a partir de dicha data, por lo cual, la optimización de las redes 4G LTE, y las mejoras de desempeño, quedan propuestas para los operadores de red, a partir del uso las herramientas trabajadas.

También es importante mencionar, que al no contar con *datasets* de operaciones reales de redes 5G, no es posible hacer una conexión directa de la metodología usada en este trabajo con una eventual aplicación en datos de esta nueva tecnología de red, ya que probablemente ciertas componentes entre las bases de datos no sean compatibles, a nivel de escala de la data, o respecto a *features* a utilizar. Por lo tanto, se requeriría un estudio comparativo de la

estructura de la eventual base de datos 5G v/s la 4G LTE utilizada, para poder implementar la metodología planteada de manera directa. De todas formas, ciertas componentes deberían ser análogas entre ambas tecnologías, como las variables objetivos utilizadas en este trabajo.

1.4. Estructura del Trabajo

El presente trabajo contiene 5 Capítulos que se presentan brevemente a continuación:

1. El Capítulo 1: Introducción; en donde se presenta la Motivación, los Objetivos y los Alcances del trabajo de Título.
2. El Capítulo 2 corresponde al Marco Teórico y Estado del Arte. Permite ubicar al lector en el contexto actual de lo que se desea investigar, y proporciona la información necesaria para comprender el trabajo completo, presentando los conceptos esenciales de las redes móviles y herramientas de *Machine Learning*, y el estado del arte de las principales problemáticas de la tecnología 5G/4G, así como el uso de ML como solución a estas mismas.
3. El Capítulo 3 trata de la Metodología y Desarrollo, ahondando en cada paso necesario para la realización de las actividades propuestas, y detallando la forma en que se obtienen los posteriores resultados.
4. En el Capítulo 4, se presentan los Resultados, y se discute brevemente cada uno de los casos de estudios.
5. Para terminar, en el Capítulo 5 se presentan las Conclusiones y Trabajo futuro, evidenciando en qué contribuyó el trabajo realizado, si se completaron los objetivos planteados inicialmente, y proponiendo posibles pasos para dar continuidad al trabajo realizado.

Por último, se tiene una sección de Bibliografía, y un apartado de Anexos, donde se presentan más resultados de los presentados en el Capitulo 4.

Capítulo 2

Marco teórico y Estado del arte

2.1. Revisión Bibliográfica

La nueva generación 5G tiene una gran cantidad de desafíos a resolver, dadas las ambiciosas propuestas tecnológicas que implica su despliegue y su correcto funcionamiento. Esta tecnología propone proporcionar altos niveles de velocidad, bajas latencias y brindar una elevada conectividad, además de traer consigo nuevas funcionalidades de red (NF), que deben ser capaces de operar de manera automática, para alcanzar los estándares esperados de esta nueva tecnología de telecomunicaciones. En este ámbito, se hace más que necesaria la integración de herramientas capaces de procesar dinámicamente flujos de datos, contenedores de información de los equipos de usuarios (UEs) y del funcionamiento de la red misma. La incorporación de modelos de *Machine Learning* (ML) a la estructura de redes 5G, permitiría automatizar diferentes áreas como; diseño, construcción, operación, monitoreo, seguridad y detección de fallas[2].

Los modelos de ML se clasifican en 3 tipos principalmente; Aprendizaje No Supervisado: En donde se suele utilizar *clustering* para identificar características similares en la data no etiquetada, Aprendizaje Supervisado: Se enseña al modelo a clasificar y/o predecir información a partir de data etiquetada y Aprendizaje Reforzado: El modelo actúa como un agente que resuelve tareas mediante prueba y error, escogiendo la opción óptima, a partir de la entrega de recompensas cuando se elije la mejor opción. El uso de cada tipo de modelo, se basará en lo que se desea obtener y el área en donde se está trabajando, existiendo modelos y herramientas más adecuadas para abordar cierto tipo de problemáticas en el ámbito de las telecomunicaciones que otras[3].

Una de las funcionalidades que caracteriza a las redes 5G, es la capacidad de realizar *Network Slicing*, que es la construcción de particiones de red con recursos suficientes, al utilizar funciones de red de virtualización (NFV), y tecnologías de red definidas por software (SDN). Esto permite que se pueda aplicar una programación total en las redes, pudiendo crear y manejar rápidamente múltiples redes lógicas, como operaciones virtuales independientes

sobre la infraestructura física compartida: asignando recursos de red como ancho de banda, frecuencias de radio y recursos virtuales como memoria y discos duros, entre otros.

El *Network Slicing* ha sido abordado desde diferentes prismas en la literatura utilizando técnicas ML; en [4] se busca identificar la aplicación solicitada por el UE, en el ámbito de *Enhanced Mobile BroadBand* (eMMB), para realizar las asignaciones de *slices* con velocidades adecuadas, según el tipo de *request*, utilizando técnicas de clasificación como *Bayes*, *Support Vector Machine* (SVM), Redes Neuronales (NN), *Gradient Boost* (GBT) y *Random Forest* (RF), mientras que en [5], se busca clasificar los *request* en los 3 tipos de casos de uso, dados por las especificaciones de la tecnología 5G[6], que son eMMB, *Critical communication* o también llamado *Ultra-reliable low-latency communication* (urLLC) y *massive machine type comunications* (mMTC). Otros enfoques que utilizan métodos de clasificación y predicción, se centran en la utilización de la información del canal CQI (*Channel Quality Indicator*), para identificar, por un lado, si es que los UEs se encuentran en situación estática, de manera de poder reducir la cantidad de reportes enviados por el canal CQI, con el fin de poder conocer en tiempo real los cambios de QoS, cuando se logra descongestionar dicho canal[7]. Por otro lado, en [8] se identifican las celdas que generan interferencia, con el fin de aumentar la calidad y capacidad de la red, al potenciar o de-potenciar la señal de referencia (RS) o cambiar la inclinación de las antenas. En [9], se prueban diversos modelos de ML para que NWDAF (*Network Data Analytic Function*), pueda realizar predicciones de la carga de la red y estudios del comportamiento de los usuarios, implementando el ML de manera directa en la red 5G.

El *Reinforcement Learning* (RL), también ha sido explorado para dar solución a problemas de las redes 5G, en particular en el contexto de *network slicing*, este tipo de enfoque se ha utilizado en [10] y [11]. En dichas investigaciones, se busca poder generar políticas óptimas de aceptación/rechazo de *request*, por parte del proveedor de infraestructura (InP), con el objetivo de monetizar la red y así maximizar los beneficios del InP, conllevando a su vez, a la entrega de un servicio que cumpla con los requerimientos de QoS de cada *request* aceptado. En [12], se tiene como objetivo, realizar una distribución dinámica de los recursos de la red 5G RAN, maximizando la utilidad de la red. Esto se logra al asignar *request* de *slice* en un horizonte de tiempo determinado, teniendo en cuenta las restricciones de los recursos para cada instante. En [13], se propone un enfoque diferente de utilización de RL, en donde en el contexto de la movilización de los UEs a través de la red, se busca mantener los niveles de QoS sobre el estándar de calidad, al proponer rutas de viaje en que se eviten aquellas celdas con un alto nivel de congestión, basándose en la utilización de los recursos físicos (PRBs).

Otro enfoque utilizado para la mantención de niveles de QoS de la red, es la resolución de problemas de optimización, en donde se proponen modelos matemáticos a la medida que permiten maximizar las variables de interés; en [14], se busca la maximización de *throughput* del sistema, al obtener cobertura óptima y minimizar los *Radio Link Failure* (RLF). En el ámbito de *network slicing*, en [15], se desea asegurar la aislación de los *slices*, y el cumplimiento de *throughput* de dichas particiones, al generar *subslices* que permitan optimizar el *throughput* por servicio de usuario, y maximizar la cantidad de usuarios por subpartición. También en [16], se utiliza *Radio Environment Map* (REM), para disminuir el índice de *handover* en redes densas, maximizando los niveles de *throughput*. De igual manera, se han

utilizado meta-heurísticas para la resolución de los problemas de optimización; en [17], se ocupa BA (*Bat Algorithm*) para mitigar la interferencia en la comunicación D2D (*Device to Device*), reutilizando el espectro de los CU (*Cellular Users*), al designar potencias óptimas de transmisión, y en [18], se resuelve el problema LFA (*Latency-aware Flow allocation*) en base a SA (*Simulated Annealing*), de manera de designar los recursos de transmisión para flujo de datos, teniendo en cuenta QoS en relación a la latencia de los flujos.

Con la literatura anterior, se logra vislumbrar la situación actual de los estudios referentes al despliegue de la tecnología 5G, y al uso de variadas herramientas de ML y optimización para la resolución de diversas problemáticas. Sin embargo, en las redes 4G, aún existen diferentes desafíos que tienen temáticas similares a las ya mencionadas, cómo por ejemplo el aumento de cobertura y capacidad de la red, para poder mantener niveles óptimos de QoS. La problemática de planificación de construcción de BS es abordada en [19] , donde se desea mejorar el desempeño de la red y atraer más usuarios potenciales, al proponer modelos que tengan en cuenta *peaks* de *throughputs* de celda de usuario, y niveles de densidad de BS. En [20], se aborda la colocación de *Small Cells*, definidas como pico celdas, de manera de extender el rango y capacidad entre BS/eNBs, teniendo en cuenta la posibilidad de fallas en las mismas, proponiendo para ello una formulación de programación lineal entera, para la planificación del operador de redes.

Otro enfoque del problema de colocación de celdas eNBs, se evidencia en [21], donde se requiere un buen esquema de colocación al optar por desplegar celdas macro, micro y pico por parte del operador, a modo de abaratar costos de instalación, mantener una buena cobertura, y cumplir con los requerimientos de *data rate* de los UEs. Para ello, se propone un esquema de lógica difusa, que es comparado con otros esquemas de colocación de celdas. Por otro lado, en [22], se tiene en cuenta que dados los requerimientos de 5G, las redes se transformaran necesariamente en UDN (*Ultra-dense networks*), ante el gran despliegue de nuevas eNBs/gNBs, existiendo problemáticas de interferencia entre celdas, que deben ser resueltas a través de una correcta búsqueda y selección de eNBs, por lo cual, se propone un nuevo esquema basado en el rediseño de las señales de sincronización. Así también, la interferencia entre celdas, afecta directamente en el desempeño del *Uplink Throughput* percibido por el usuario, como se ve en [23], en donde mediante técnicas de ML de aprendizaje no supervisado, se logra detectar *clusters* de comportamientos similares de UEs, a partir de una selección previa de *features* importantes relacionados con *UL User Throughput*, permitiendo modificar parámetros del control de potencia que maneja las interferencias de *Uplink*, según los agrupamientos hallados. De este modo, se logra aumentar la velocidad de subida percibida por los usuarios.

La ubicación de las celdas no es algo trivial, y afecta directamente en el desempeño de la red, como se aprecia en las citas anteriores, pero el uso de la información de geolocalización de las antenas, puede permitir realizar novedosos análisis del comportamiento de los usuarios, como se propone en [24], que mediante el uso de LDA, en data generada por llamadas efectuadas durante 73 días, en diferentes antenas de una red de telecomunicaciones móviles, se logra detectar patrones de uso de determinadas zonas geográficas de Santiago. En base a este tipo de análisis, se pueden detectar posibles parámetros a modificar, que permitan una mejora en el desempeño de la red móvil, como se hizo en [23], según el tipo de uso que se

esté dando, en determinadas celdas de ciertas zonas geográficas, en los horarios que se tenga algún tipo de patrón característico. El uso del modelamiento de tópicos mediante LDA se ha utilizado en contextos más contemporáneos como la actual pandemia de COVID-19, proponiendo hacer uso de la información temporal y geográfica de las transacciones de las tarjetas de crédito, buscando con ello, evidenciar cambios en el comportamiento de las personas bajo el contexto de las medidas sanitarias de encierro y toque de queda en las diferentes comunas de Santiago[25].

Dado lo anterior, se hará uso de modelos de ML en casos concretos, específicamente en data real generada por la operación de la red 4G de Entel en Santiago, permitiendo realizar analítica predictiva que aporte al mejoramiento y aumento del desempeño en los niveles de *Download User Troughput*. Además, se realizará un análisis por zonas geográficas mediante LDA, permitiendo mejorar o abaratar costos de operación de la red. La metodología planteada se puede proyectar al futuro despliegue de 5G, gracias a la generalización otorgada por las diferentes herramientas de ML, las cuales pueden seguir siendo efectivas al cambiar la data por una 5G, pero teniendo en cuenta que, se deberá hacer un respectivo análisis de los *features* relevantes con los que se trabajará.

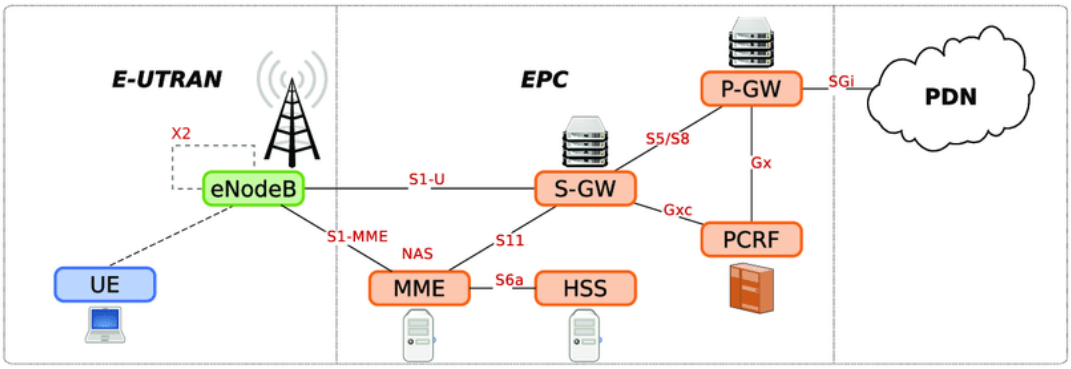
2.2. Marco Teórico

En primera instancia, es menester comprender cómo se organiza la arquitectura de redes de telecomunicaciones, de manera de poder comprender, posteriormente, la información de las base de datos trabajadas.

2.2.1. Redes Móviles

Una red móvil, hace uso de transmisores y receptores de baja potencia, para poder conectar dispositivos que permiten la comunicación entre persona a larga distancia. A medida que la tecnología ha ido evolucionando, y se le ha ido agregando más calidad al servicio móvil, en términos de velocidad, latencia y disponibilidad, el rango de los transmisores ha ido decreciendo, siendo necesario dividir las áreas de servicio en celdas, cada una implementada en su propia antena. Cada celda se ubica en una banda específica de frecuencia y es atendida por una Estación Base (BTS), o una Estación Radio Base (ERB), dependiendo del enfoque de tecnología con el que se esté explicando las topologías de red. Estas operan como transmisor, receptor y unidad de control. Las celdas adyacentes tienen asignada diferentes frecuencias, de manera de evitar interferencias, pero ya a determinada distancia, se puede utilizar bandas de igual frecuencia [26].

Más concretamente, la red se compone de equipos de usuario (UEs), también llamados móviles, que corresponde los equipos que el usuario trae consigo para lograr la comunicación con una estación base, en el momento que lo desee y en el lugar en donde exista cobertura. Estos, deben ser capaces de soportar los estándares y protocolos para operar bajo distintos sistemas de comunicación, teniendo que poder de acceder a la UTRAN(*UMTS Terrestrial Radio Access Network*), propia de la topología 3G, o a sistemas más antiguos como el GSM de 2.5G, tanto para voz, como para datos. La tecnología 4G LTE, cuenta con un eRAN(*evolved Radio Access Network*), que se compone solo de un nodo, eNodeB(eNB), que es aquel que interactúa con los UEs, y vendría a ser equivalente al Nodo B en UMTS(*Universal Mobile Telecommunications System*), que a su vez correspondería a los BTS(*Base Transceiver Station*) en GSM, pudiendo dar servicio a una o más celdas. El eNodeB cuenta con capas de *hosts* de PHYsical(PHY), *Medium Access Control*(MAC), *Radio Link Control*(RLC) y *Packet Data Control Protocol*(PDCP) [27]. En la Figura 2.1 se presenta un esquema de la topología de una red 4G:



Fuente: Tomado de "*Alocação de Recursos baseada em Clustering com Aprendizado de Características e Orientação a QoS em Redes LTE-Advanced*" por E. Santos, 2019

Figura 2.1: Topología básica de red 4G LTE

De la Figura anterior, se visualiza que para la topología de red móvil 4G LTE, la parte más cercana al UE es la red de acceso E-UTRAN(*evolved UTRAN*), compuesta por los eNB de manera principal. Posteriormente, se encuentra la capa EPC(*evolved Packet Core*), compuesta por diferentes elementos de red que hacen que se maneje el trafico de datos de manera eficiente en términos de desempeño y costos. En esta arquitectura, se separa los datos de usuario, también conocido como plano del usuario, del plano de control, o señalización [28]. Los 5 elementos del EPC son:

MME: *Movility Management Entity*, es uno de los elementos principal del plano de control, ubicado usualmente en ubicaciones seguras, como las instalaciones del operador. Cumple funciones de autenticación y seguridad, gestión de movilidad, gestión del perfil del suscriptor y servicio de conectividad.

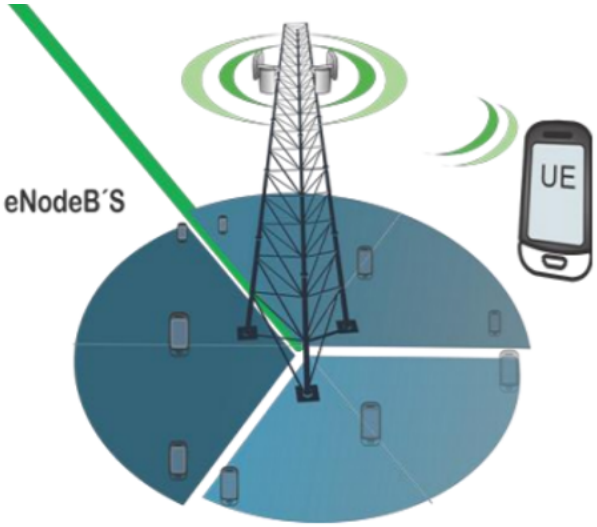
SGW: *Serving Gateway*, configuración básica de la arquitectura del sistema, realizando gestión de los túneles y conmutación en el plano del usuario.

PGW: *Packet Data Network Gateway*, o PDN-GW, es el router frontera entre el EPS y redes de paquetes externos, actuando como punto de conexión IP de los UEs, sincronizando y filtrando el tráfico según se requiera.

PCRF: *Policy and Charging Resource Function*, es el elemento de red responsable de la política y el control del charging (PCC), tomando desiciones sobre la manera en que se manejan los servicios en relación a calidad.

HSS: *Home Subscription Server*, es el repositorio de los datos permanentes de subscripción de todos los usuarios, registrando la ubicación de los UEs en el nivel de nodo de control de red visitados.

En la Figura 2.2, se visualiza de manera más didáctica un ENodeB, que es una torre tipo antena compuesta por varias celdas, las cuales pueden operar en diferentes frecuencias para atender a los diferentes UEs:



Fuente: Adoptado de "*Descripción del Proyecto y Viabilidad Técnica (Fracción I Artículo 14 Ley Asociación Privadas).*" por Secretaría de Comunicaciones y Transportes del Gobierno de México, 2019

Figura 2.2: Topología de eNBs de red móvil

Clarificando un poco el contexto de la data con la que se trabajará posteriormente, en donde se identifican celdas que operan en diferentes frecuencias de operación, y que pertenecen a un eNodeB, se procede a introducir otros conceptos que serán ocupados en este trabajo de título.

Se utilizarán diversos conceptos relacionados con *Machine Learning* y herramientas computacionales comúnmente utilizadas en ciencia de datos. Es por esto, que se hace necesario presentar definiciones formales de dichos conceptos, los cuales serán mencionados posteriormente en el Capítulo de Metodología.

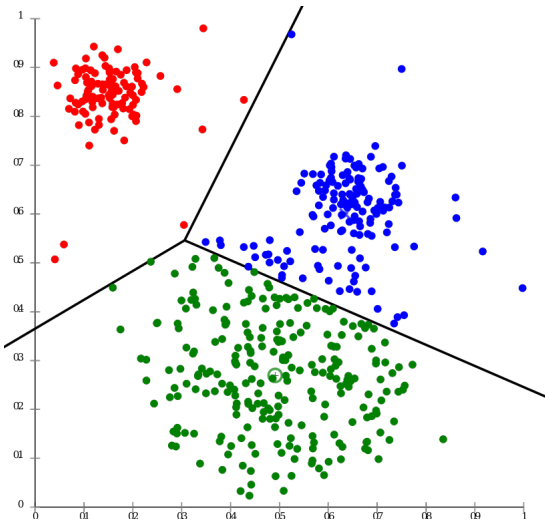
2.2.2. Clustering

Dentro de las técnicas de ML de aprendizaje No Supervisado, se puede hallar el *Clustering*, el cual permite agrupar un conjunto de datos, dados en subconjuntos de objetos con características similares, a este subconjunto se le denomina *Cluster*. Esta técnica permite poder dividir el *dataset* de manera de poder hallar patrones de interés en los grupos generados o poder representar de mejor manera la distribución de la data. Técnicas como *K-Means* y DBSCAN permiten realizar *Clustering*, las cuales son presentadas a continuación:

K-Means

El algoritmo funciona, escogiendo primeramente, una cantidad K arbitraria de *Clusters* en los datos, para luego calcular los centroides de este conjunto. Posteriormente, se verifica la distancia de cada dato, hacia los centroides calculados, reasignando los *clusters* en caso en que un centroide de otro *cluster*, tenga una distancia mínima, en comparación al centroide del *cluster* actual asignado. Con la reasignación de los *clusters*, se recalculan los centroides y el algoritmo vuelve a funcionar, hasta una determinada cantidad de pasos o hasta que no existan nuevas reasignaciones de *clusters*[29].

Existen métodos para poder hallar la cantidad K óptima de *clusters* a escoger. El método del Codo o *Elbow Method*, es una heurística que consiste en graficar la variación de la data, en función de la cantidad de *clusters*, de esta manera, se encuentra gráficamente el punto que indica que, ante una mayor cantidad de *clusters*, no se obtiene una mejora en el modelamiento de la data, evitando el sobreajuste ante el aumento de K[30]. En la Figura 2.3, se visualiza la separación de la data respecto a un centroide, escogiendo una cantidad de *clusters* igual a 3:



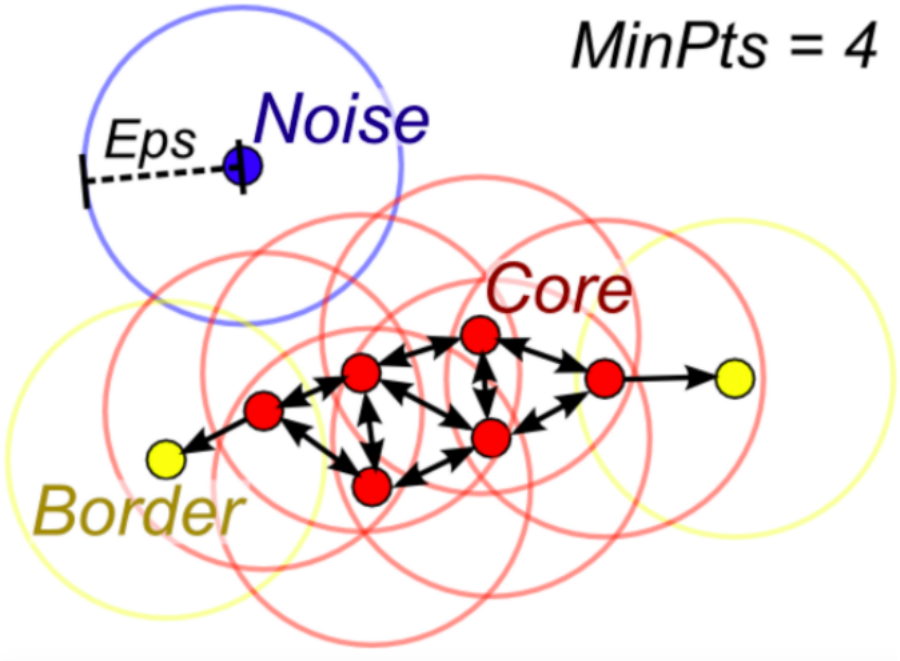
Fuente: Tomado de "*Cluster analysis with k-Means on a gaussian-distribution-based data set*" por Chire, 2011

Figura 2.3: Visualización de clustering *K-Means*

DBSCAN

El significado de las siglas de este método, corresponde a: *Density-Based Spatial Clustering of Applications with Noise*, el cual se distingue por basarse en la densidad de los puntos a analizar, agrupando aquellos que tengan mayor cantidad de puntos vecinos, diferenciando aquellos que se encuentren en regiones con menor densidad y considerando como *outlier* o puntos ruidosos aquellos que estén sumamente separados respecto al resto de los datos. Todo lo anterior, se realiza en base a la especificación del parámetro ε, que indica el radio de la vecindad de un punto, y a una cantidad mínima de datos contenidos en cada agrupación[31].

En la Figura 2.4, se aprecia el funcionamiento del algoritmo detrás de DBSCAN, en donde se ve que al fijar una cantidad mínima de 4 puntos, y un valor dado de ε, se va probando en los diferentes datos, si se logra cumplir los requerimientos para considerar dichos puntos como *clusters*, evidenciando que los puntos rojos si lo cumplen, los amarillos se considerarían otro tipo de *cluster*, y los azules pasaría a ser considerados como ruido:



Fuente: Tomado de "*DBSCAN cluster analysis (minPts=4)*" por Chire, 2011

Figura 2.4: Explicación gráfica de algoritmo de DBSCAN

2.2.3. Reducción de dimensionalidad

Cuando se cuenta con una alta cantidad de *features* en los datos a procesar, o se desea obtener una visualización exploratoria del comportamiento de la data, se suelen utilizar técnicas de reducción dimensionalidad. Algunas de ellas son presentadas a continuación:

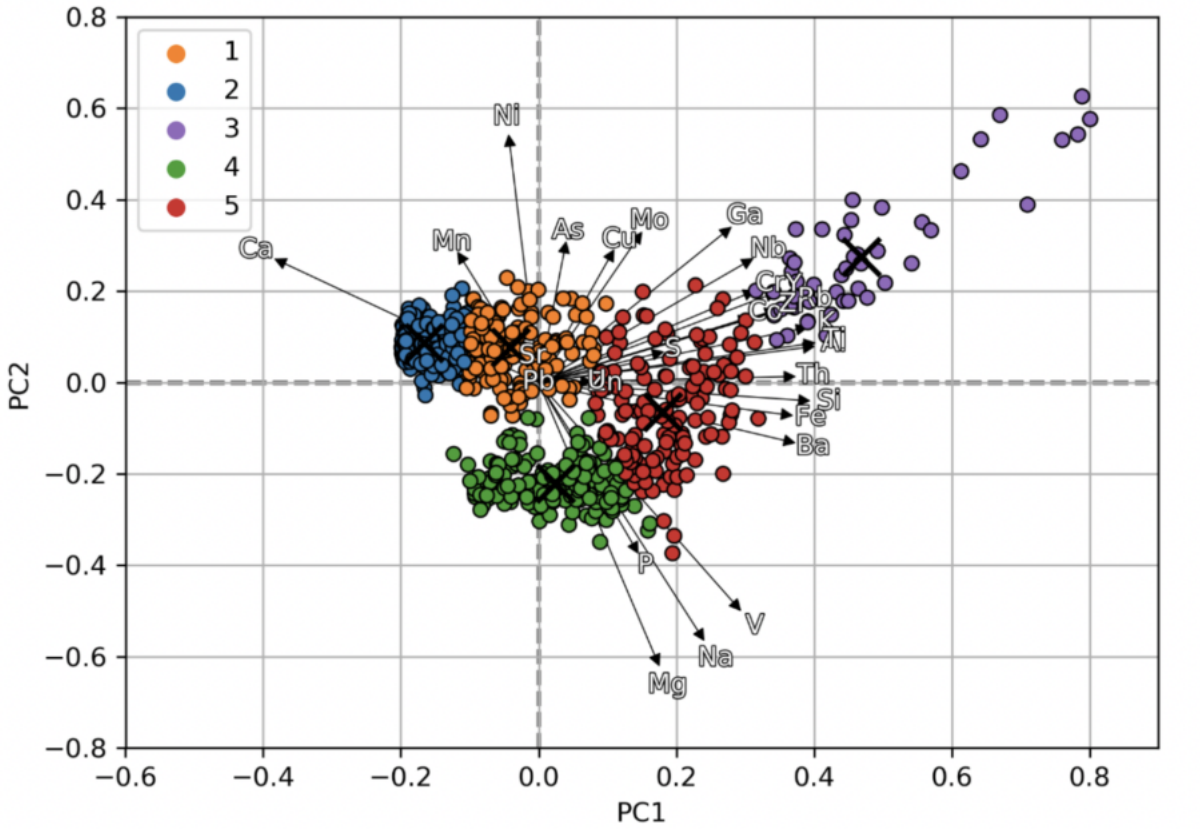
PCA

PCA, que significa *Principal Component Analysis*, es un algoritmo no supervisado, que permite reducir la cantidad de *features* que caracterizan un determinado *dataset*, puesto que existe la posibilidad no menor, que en una base de datos con alta cantidad de columnas, se tenga un alto índice de redundancia en la información contenida en ellas. Para ello, se aplican transformaciones lineales como matrices de varianza, covarianza, vectores y valores propios, que permiten mover el conjunto original de *features* a un nuevo espacio, compuesto por las componentes principales. Estas componentes, son aquellas que contengan la mayor varianza entre los datos transformados, ya que esto indicaría que contienen mayor información que permita diferenciar entre clases, por ejemplo, de aquellas nuevas columnas que tengan datos muy parecidos en todas sus instancias, por ende poca varianza[32].

Una de las formas de realizar PCA, es primeramente calcular la matriz de covarianza de los datos estandarizados. A esta matriz, se le calculan los vectores propios y sus valores propios. Posteriormente, estos vectores propios son rankeados y ordenados a partir de su magnitud de valor propio, el cual indicará la cantidad de varianza que posee cada vector. Escogiendo aquellos vectores que contengan mayor cantidad de varianza, se obtiene el *Feature Vector*, el cual se utilizará para reorientar la data original de acuerdo a los ejes de las componentes principales halladas[33]. Esto último, se realiza de la siguiente forma:

$$DataSetFinal = FeatureVector^T \cdot DataSetOriginalEstandarizado^T \tag{2.1}$$

En la Figura 2.5, se aprecia cómo un *dataset* es reorientado según sus 2 componentes principales, obtenidos tras aplicar PCA, evidenciando que se logran separar los datos, que es lo que se busca obtener al tener como objetivo la maximización de la varianza:



Fuente: Tomado de "*Making sense of it all: extracting actionable core-data from pxrf using pca and k-means cluster analysis*" por E. Pedersen, 2020

Figura 2.5: Gráfico 2D de PCA en dataset de elementos químicos

t-SNE

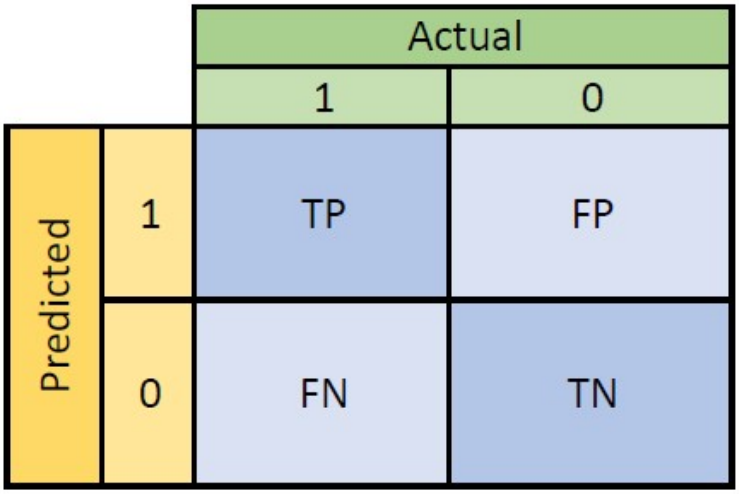
Esta técnica, *t-distributed stochastic neighbor embedding*, a diferencia de PCA, es un método no lineal de reducción de dimensionalidad, en donde se preserva la data local, más que la global. Se utiliza fijando hiperparámetros como *learning rate*, *perplexity* y número de pasos, funcionando al minimizar la divergencia de Kullback–Leibler de las distribuciones de probabilidad, de asignar los puntos del dataset de alta dimensionalidad, a una locación especifica en un mapa de 2 o 3 dimensiones, generando que los objetos similares permanezcan cercanos, y los diferentes más alejados[34].

2.2.4. Clasificación

La clasificación es una tarea en donde se desea identificar la pertenencia de un dato u observación respecto a un conjunto de categorías posibles. En *Machine Learning*, las observaciones son conocidas como instancias, las variables como *features* y las posibles categorías como clases.

Para poder medir el rendimiento de un modelo de clasificación se pueden utilizar diversas métricas, pero una de las más utilizadas es la matriz de confusión, en donde se puede detectar fácilmente los casos en que el modelo acierta a las clases correctas y cuando se equivoca en la clasificación.

En una matriz de confusión de una clasificación binaria, como la que se presenta en la Figura 2.6, se pueden identificar la siguiente información[35]:



Fuente: Elaboración propia

Figura 2.6: Matriz de Confusión de Clasificación Binaria

1. **True Positives (TP):** Cuando la clase real es 1 y la resultante también lo es.
2. **True Negatives (TN):** Cuando la clase real es 0 y la resultante también es 0.
3. **False Positives (FP):** Cuando la clase real es 0 y la pronosticada es 1
4. **False Negatives (FN):** Cuando la clase real es 1 y la pronosticada es 0

Las matrices de confusión también pueden efectuarse en clasificación multiclases, y los indicadores anteriores se ampliarían para la detección de las demás etiquetas, pero su significado vendría a ser el mismo, el escenario ideal es cuando se tiene un valor de 0 en FP y FN. Además, a partir de estas cifras, se puede calcular indicadores que dan más claridad al

desempeño del modelo[36]:

- **Accuracy (Exactitud) :** Se define como la cantidad de detecciones correctas dentro del total de predicciones. Es considerada la medida más directa de calidad de los clasificadores, pero puede ser un indicador engañoso si es que las clases cuentan con un gran desbalance. En la Ecuación 2.2 se presenta su definición formal:

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{2.2}$$

- **Recall (Sensibilidad) :** Se define como la cantidad de detecciones correctas positivas dentro del total de positivos verdaderos. En la Ecuación 2.3 se presenta su definición formal:

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

- **Precision (Precisión) :** Se define como la cantidad de detecciones correctas positivas dentro del total de elementos identificados como positivos. En la Ecuación 2.4 se presenta su definición formal:

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

- **Specificity (Especificidad) :** Se define como la cantidad de detecciones correctas identificadas como negativas fuera del total de negativos. En la Ecuación 2.5 se presenta su definición formal:

$$Specificity = \frac{TN}{TN + FP} \tag{2.5}$$

Naive Bayes

Este clasificador usa la data de entrenamiento para calcular la probabilidad observada de cada clase basándose en todos los *features*. Este algoritmo es considerado '*naive*' por las suposiciones que asume en su desarrollo, en donde por ejemplo se asume que todos los *features* son igual de importantes e independientes. Su fórmula básica se visualiza en la Ecuación 2.6, en donde se tienen n *features* F_i, y K cantidad de clases, dado por C_k:

$$P(C_k|F_1, ..., F_n) = \frac{P(F_1, ..., F_n|C_k)P(C_k)}{P(F_1, ..., F_n)} \tag{2.6}$$

Teniendo que el valor del denominador es constante, ya que está directamente relacionado con la data propiamente tal, solo el numerador depende de las etiquetas de la clase, el cual puede ser traducido a su expresión de probabilidad conjunta, y dado que se asume que cada *feature* es independiente de otro, se tiene lo siguiente:

$$P(F_1, ..., F_n, C_k) = P(C_k) \prod_{i=1}^{n} P(F_i|C_k) \tag{2.7}$$

Considerando un grupo de clases en particular L, la asignación de la de la clase corresponderá a la probabilidad máxima posterior, obteniendo así $\hat{C}$ que indicaría la clase asignada de la instancia analizada[37]:

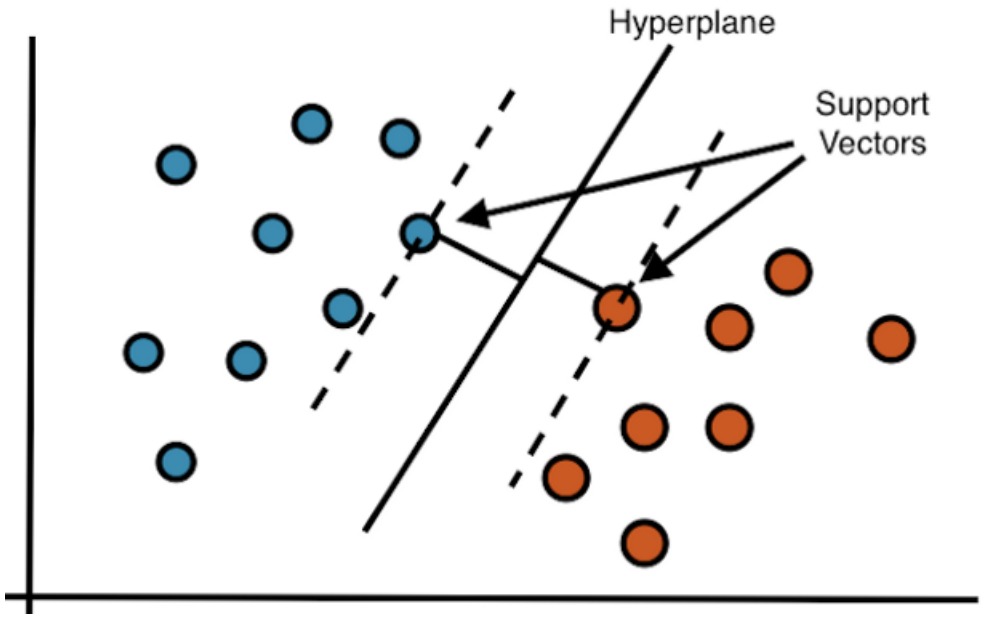
$$\hat{C} = arg \max_{L} \frac{P(C_k) \prod_{i=1}^{n} P(F_i|C_k)}{\prod_{i=1}^{n} P(F_i)} \tag{2.8}$$

Support Vector Machines

Conocido también como SVM, consiste en un conjunto de algoritmos que permiten resolver problemas de clasificación y regresión. Básicamente esta herramienta construye un hiperplano o un conjunto de estos de manera de separar las clases de un conjunto de entrenamiento, llamando a estos planes como vectores de soporte. Una buena separación entre las clases permitirá una mejor clasificación, ya que la asignación de clases dependerá directamente de en qué posición del hiperplano se ubique un punto.

Usualmente, se hace necesario realizar transformaciones de la data para poder realizar la separación espacial, por lo que se utilizan funciones *Kernel* que permiten trabajar en un espacio de características de mayor dimensionalidad. Alguno de los *Kernel* más comunes son el *Kernel* polinomial, sigmoide, RBF (*Radio Base Function*) Gaussiano[38].

En la Figura 2.7, se aprecia la aplicación de SVM lineal, ya que el hiperplano es simplemente una recta que logra separar los datos:



Fuente: Tomado de "*Explicación de la máquina de vectores de soporte (SVM)*" por V. Patel, 2020

Figura 2.7: Visualización de SVM lineal

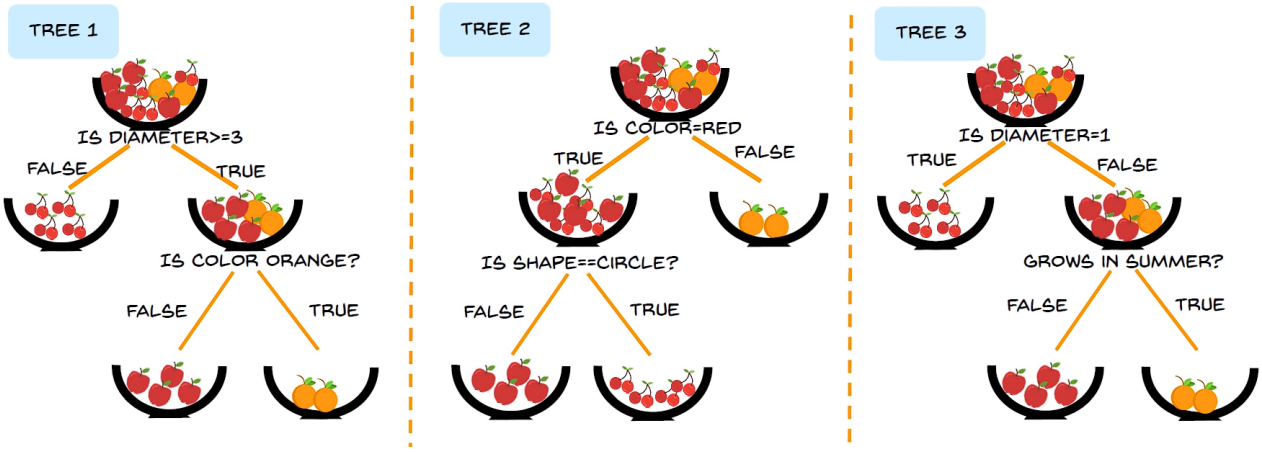
Random Forest

Los árboles de decisiones, son estructuras que permiten clasificar la data de entrada o predecir valores de salida, dado los valores de entrada, dividiendo la información en determinada cantidad de regiones, en donde se utiliza un modelo constante para predecir, como la siguiente función[39]:

$$f(x) = \sum_{j=1}^{T} w_j I(x \in R_j) \tag{2.9}$$

Esta función puede ser una consulta respecto a si el valor de entrada supera determinado umbral o no, y de esta manera el árbol va creciendo en profundidad y categorizando la data de entrada. En particular *Random Forest* es un algoritmo que hace uso del aprendizaje de los árboles de decisiones de manera robusta y práctica, usando una gran cantidad de arboles de decisiones especializados, para finalmente juntar sus diferentes salidas para hacer entrega de un *output* general. Este método es utilizado ampliamente para resolver diferentes tareas de ML, y es muy popular por entregar buenos resultados sin requerir de un proceso de entrenamiento complejo[40].

En la Figura 2.8, se visualiza un diagrama de la construcción de diferentes árboles de decisión que pueden componer un modelo de *Random Forest*:



Fuente: Adaptado de "*Random Forest K Means*" por R. Guerrero, 2020. Derechos de Simplilearn

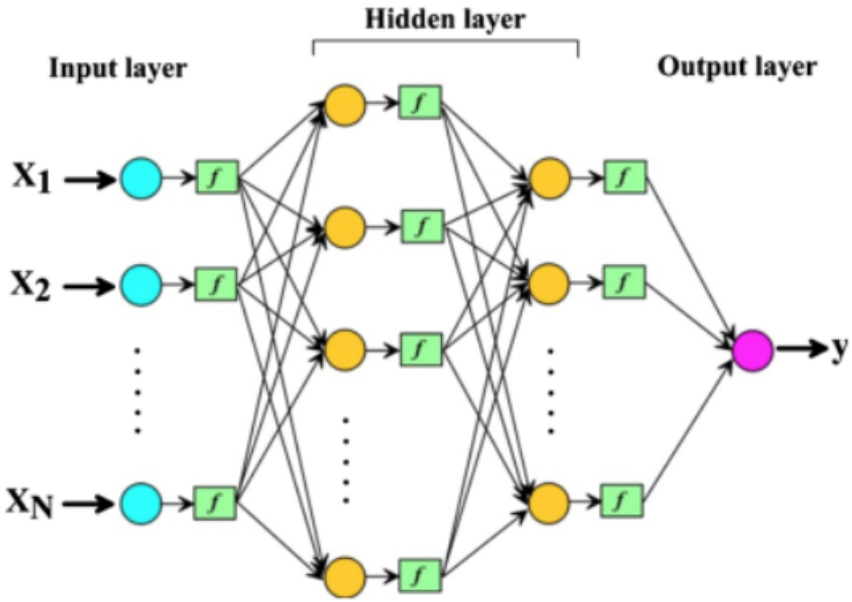
Figura 2.8: Diagrama de árboles de Random Forest

Neural Networks MLPs

Las redes neuronales MLPs o *multiplayer perceptrons*, trabajan emulando el cerebro humano, en donde se tiene una capa *input* que recibe todos los datos de entrenamiento que son procesado por varias neuronas posteriores que conforman capas ocultas en el modelo. Cada neurona de la capa oculta recibe una entrada o un conjunto de entradas, que son procesadas por una función de activación que puede ser una función sigmoide, ReLu, tangente hiperbólica, entre otras, determinando así la salida que será ponderada por un peso dado entre la conexión entre una neurona a otra, determinando así la importancia de la conexión. Finalmente se entrega un resultado en la capa *output*, el cual es comparado con la etiqueta asignada a cada dato del *input*, y se retroalimenta toda la red para modificar los pesos de las conexiones según esta diferencia, este método se conoce como *Backpropagation*[41].

De esta manera, la red neuronal se puede asimilar directamente a una maquina de aproximación de funciones, ya que la red misma debe aprender a poder obtener el $f^*(x)$ que implica obtener el valor deseado y, dada la estructura profunda que puede implicar $f^*(x)$, que vendría a equivaler $f^{(3)}(f^{(2)}(f^{(1)}(x)))$, donde $f(1)$ sería la primera capa, $f^{(2)}$ una capa intermedia oculta, y $f^{(3)}$ la capa output[42].

En la Figura 2.9, se puede apreciar una arquitectura tipo de una red MLP, en donde se logra distinguir la capa de entrada de la red, donde ingresan los datos de análisis, las capas ocultas, donde se aplican distintas funciones de activación y se almacenan los pesos de la red, y la capa de salida, en donde se puede obtener el resultado final de un proceso de clasificación:



Fuente: Tomado de "*Schematic diagram of a feed-forward MLP neural network* " por A. Barzegar, 2016.

Figura 2.9: Arquitectura de una red MLP

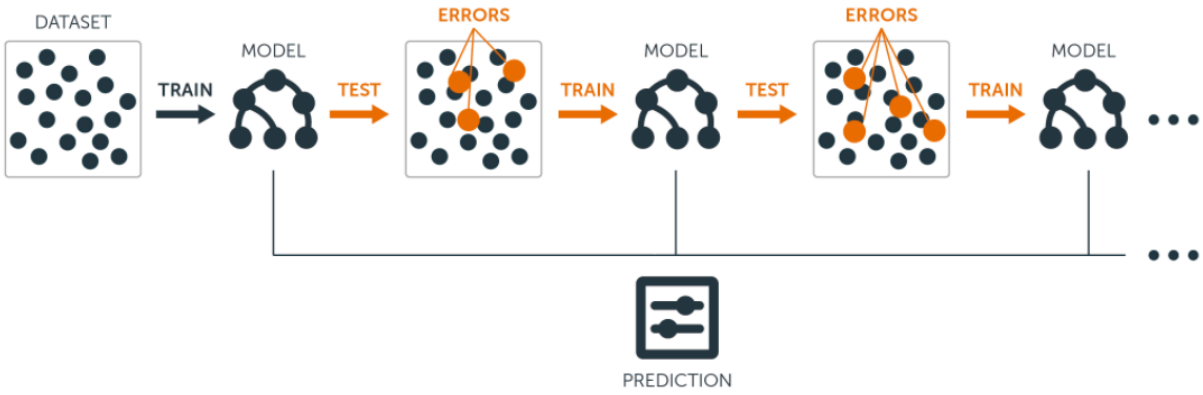
Gradient Boosting

Gradient Boosting es una herramienta de ML que se caracteriza por generar modelos predictivos de clasificación o predicción al usar en conjunto otros esquemas predictivos más simples como por ejemplo arboles de decisión.

Se pueden utilizar diversos algoritmos para los modelos de aprendizaje de árboles, CART (*Classification and Regression Trees*) es uno de ellos, y se puede representar como un árbol binario que utiliza *Greedy Splitting*, con un criterio de detención relacionado con una cantidad de instancias de entrenamiento y con capacidad de poda para mejorar el *performance*[43].

La función de costo escogida y el algoritmo de aprendizaje del árbol especificará qué tipo de *gradient boosting* se implementará, en donde se puede diferenciar algoritmos como GBM que utilizan descenso del gradiente como función de costo, mientras que *XGBoost* hace uso de un enfoque similar al de *Newton Rhapson*, entre otras diferencias como por ejemplo determinadas políticas de penalización y uso de parámetros aleatorios[44]. Gracias al uso de *gradient boosting*, se entrenan nuevos modelos de manera iterativa que permiten mejorar los puntos débiles del modelo anterior, obteniendo usualmente los mejores resultados en comparación a otras herramientas de ML.

En la Figura 2.10, se visualiza un esquema de la estructura del modelo *XGBoost*, en donde se ve que se entrenan modelos de manera iterativa, aprendiendo de las iteraciones anteriores:



Fuente: Tomado de "*Decision Trees in R*" por J. Lee, 2018.

Figura 2.10: Estructura de entrenamiento de modelo *XGBoost*

2.2.5. Pronóstico

Cuando se habla de predicción, se refiere a la salida obtenida de un algoritmo posterior a un proceso de entrenamiento con datos históricos, aplicando esta técnica en data nueva para hallar el pronóstico de alguna variable de interés, usualmente este pronóstico es de carácter continuo.

Para medir el desempeño del modelo predictivo se pueden utilizar varios indicadores de desempeño, pero para ello hay que primero definir la diferencia entre error y sesgo(*bias*), los cuales se presentan en las Ecuaciones 2.10 y 2.11 respectivamente, donde f_t es el valor del pronóstico y d_t es el valor de la data real de dicho periodo de tiempo. Mientras que el error indica la diferencia entre el pronóstico y el valor real, el sesgo es un valor promedio de este error entre los periodos que hay data histórica y pronósticos[45] :

$$e_t = f_t - d_t \tag{2.10}$$

$$bias = \frac{1}{n}\sum e_t^2 \tag{2.11}$$

- **MAPE :** *Mean Absolute Percentage Error*, se define como la suma de los errores absolutos individuales dividido en sus valores reales por periodo. Por lo anterior, si lo valores reales son muy bajos puede presentar altos errores en este indicador, implicando un sesgo[46]. En la Ecuación 2.12 se presenta su definición formal:

$$MAPE = \frac{1}{n}\sum \frac{e_t}{d_t} \tag{2.12}$$

- **MAE :** *Mean Absolute Error*, como indica su nombre, es el promedio del valor absoluto. Es un muy buen indicador, pero tiene el problema de que es de difícil interpretar sus magnitudes, es por ello que se suele dividir por el promedio de los valores reales, de manera de tener una interpretabilidad del error. En la Ecuación 2.13 se presenta su definición formal, y la Ecuación 2.14 su versión escalada a la data real:

$$MAE = \frac{1}{n}\sum e_t \tag{2.13}$$

$$MAE\,\% = \frac{\frac{1}{n}\sum e_t}{\frac{1}{n}\sum d_t} = \frac{\sum e_t}{\sum d_t} \tag{2.14}$$

- **RMSE :** *Root Mean Squared Error*, se define como la raíz cuadrada del promedio del error cuadrático, y al igual que MAE, es más interpretable si está escalado por los valores reales. En la Ecuación 2.15 se presenta su definición formal, y la Ecuación 2.16 su versión escalada a la data real:

$$RMSE = \sqrt{\frac{1}{n}\sum e_t^2} \tag{2.15}$$

$$RMSE\,\% = \frac{\sqrt{\frac{1}{n}\sum e_t^2}}{\frac{1}{n}\sum d_t} \tag{2.16}$$

- **MSE :** *Mean Squared Error*, es comúnmente utilizada por ser fácil de computar y manipular, pero al no estar escalado al error original no se puede conocer la escala con relación a los valores reales. En la Ecuación 2.17 se presenta su definición formal:

$$MSE = \frac{1}{n}\sum e_t^2 \tag{2.17}$$

ARIMA

Una serie temporal es una secuencia ordenada de observaciones, cada una de las cuales, asociada a un momento de tiempo, en donde usualmente no existe mayor independencia entre los datos, de hecho, suelen ser dependientes entre sí y esta naturaleza de dependencia es de interés. Debido a estas características especificas de los datos de series de tiempo, se han desarrollado modelos específicos que logran recoger y aprovechar esta dependencia[47].

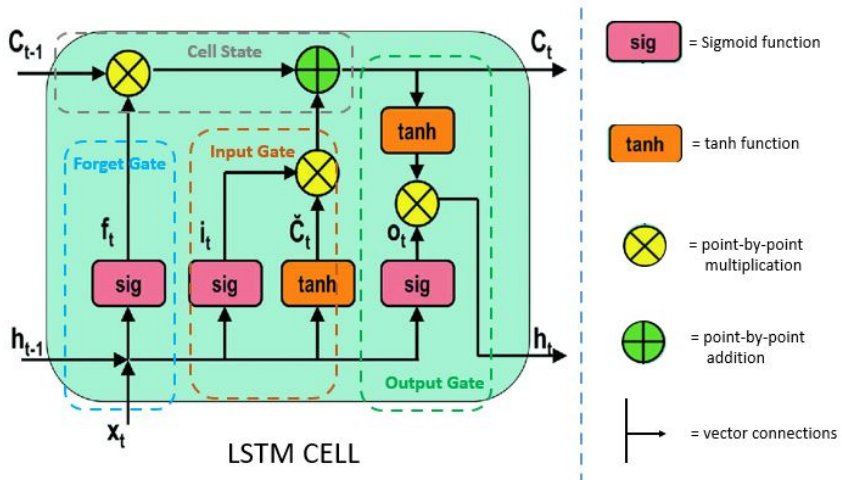
ARIMA, acrónimo que significa *AutoRegressive Integrated Moving Average*, es un método estadístico que a través de parámetros busca obtener la representación de una serie con la interrelación temporal de sus elementos. La Autoregresión da cuenta de que el modelo usa la dependencia entre las observaciones y cierta cantidad de tiempos previos. La parte Integrada refiere al uso de la diferencia de las observaciones para crear una serie de tiempo estacionario, y la parte de promedio móvil da cuenta del uso de la dependencia entre una observación y el error residual de un promedio móvil aplicado a las observaciones pasadas. Cada componente es especificado en el modelo como un parámetro, existiendo así los parámetros (p,d,q) que indicarán qué tipo de comportamiento tiene la serie de tiempo que se desea modelar con ARIMA, de esta manera se puede efectuar un posterior pronóstico una vez determinado el modelo adecuado a la data de estudio[48].

LSTM

Su primera aparición fue en 1997 por Sepp Hochreiter, en donde son propuestas como solución a los problemas que presentados por las BPTT(*Back-Propagation Through Time*) y las RTRL(*Real-Time Recurrent Learning*), cuando se deseaba recorrer tiempos pasados. En estas sistuaciones, los modelos mencionados presentaban problemas de convergencia, tendiendo a explotar, o se experimentaban perdidas de información mediante el desvanecimiento. El primer caso ocurría cuando los pesos del modelo oscilaban, y el 2do caso, cuando los espacios de tiempo eran muy grandes, los modelos tardaban tiempos prohibitivos de computación, por lo que no lograban sus cometidos [49].

Sus siglas significan *Long Short-term Memory*, y son redes neuronales recurrentes (RNN), las cuales se caracterizan por contar con conexiones *feedback*, que permiten que los gradientes calculados en pasos anteriores, se puedan mantener en el tiempo a través de los auto-bucles, en comparación a las redes tipo MLPs, en donde se presenta el problema del desvanecimiento del gradiente. Estas redes, permiten procesar no tan solo datos puntuales como fotos, sino que también secuencias de datos como audios y videos[50].

Las LSTM cuentan con 4 unidades básicas, en donde se tiene la celda, la puerta de entrada, puerta de salida y puerta del olvido o *forget gate*. La celda permite recordar valores durante tiempos arbitrarios y las 3 puertas controlan el flujo de información hacia y desde la celda. De esta manera, la información de tiempos pasados puede ser reutilizada a gusto[51]. En la Figura 2.11, se presenta un diagrama de una red LSTM:



Fuente: Tomado de "*Overview of Long Short-Term Memory Neural Networks*" por K. Smagulova y A. Pappachen, 2019.

Figura 2.11: Estructura de celda LSTM

La Figura anterior describe el funcionamiento general de una LSTM. El *forget gate* decide qué dato requiere atención, y cuál puede ser olvidado, recibiendo información del actual *input* $X(t)$ y el estado oculto del periodo anterior $h(t-1)$, que son procesados por la función sigmoide, para generar valores entre 0 y 1, dando como resultando $f(t)$, que indica si el *output* pasado es necesario, si su valor es cercano a 1. El *input gate* recibe $X(t)$ y $h(t-1)$, en donde al ser pasados por la función sigmoide, se transforma a 0, que indica que es importante, o 1 si no. Además, se pasan estas entradas por la función tangente hiperbólica *tanh*, que da como resultado valores entre -1 y 1. El *cell state* recibe la información del *input gate* y *forget gate*, y decide y almacena el nuevo valor del estado de celda $C(t)$. Para ello recibe el estado de celda anterior $C(t-1)$ y es multiplicado por $f(t)$, en donde si llega a obtener un valor igual 0, se tendrá que el estado anterior es ignorado. Posteriormente, recibe el valor resultante del *input gate*, y se actualiza el nuevo valor del estado de celda actual. Finalmente, el *output gate* determina el valor del nuevo estado oculto $h(t)$, para ello $X(t)$ y $h(t-1)$ vuelven a ser pasados por la funcion sigmoide, y dichos resultados son multiplicados punto por punto por el resultado de pasar por *tanh* el valor del estado actual de celda $C(t)$, obteniendo así el valor

final que pasará al siguiente periodo de tiempo[52].

2.2.6. RFE y GridSearch con Cross Validation

Recursive Feature Elimination es una técnica que permite escoger las columnas más relevantes de un *dataset* para lo que se desee realizar, pudiendo tener como objetivo realizar regresiones o clasificaciones. Para ello, se debe en primera instancia entrenar un predictor para un *training set* escogido y evaluar el desempeño del modelo. Posteriormente se hace un *ranking* de variables, basándose en la importancia de estas, y comienza un proceso iterativo en donde se filtran los *features* no relevantes y se vuelve a entrenar el predictor, pero ahora solo con las variables seleccionadas, observando cómo cambia el desempeño del modelo al utilizar menor cantidad de *features*.

A modo de asegurar la precisión del RFE se utiliza *Cross-Validation* (CV), en donde se puede integrar una cantidad K de evaluaciones extras al efectuar la partición *train/test* de manera diferentes en estas instancias, utilizando la media aritmética de los resultados obtenidos para conseguir un resultado final[53].

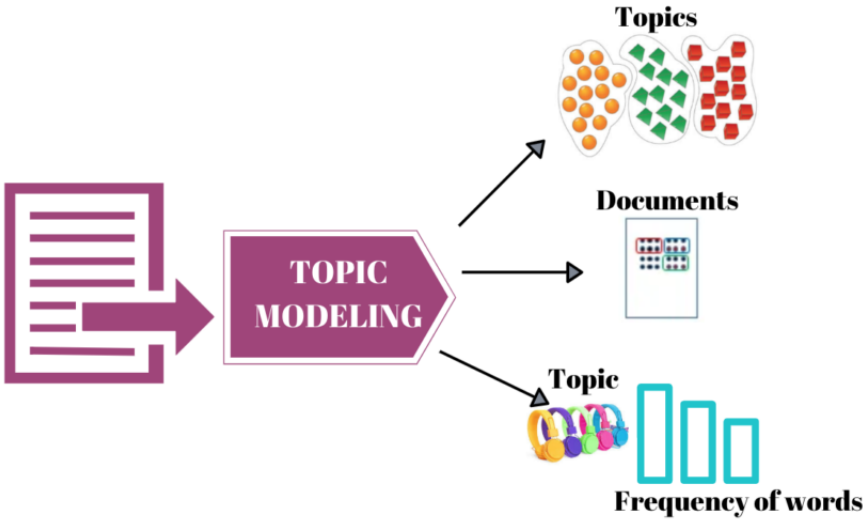
Por otro lado, *Grid Search*, es una técnica utilizada para poder hallar los mejores hiperparámetros de un modelo. Para ello se debe especificar qué hiperparámetros variar, dentro de qué rango de valores, y respecto a qué métricas evaluar el desempeño. De esta forma, el modelo es probado para todas las combinaciones posibles dentro de las opciones definidas, entregando un valor a cada combinación respecto a la métrica de evaluación escogida, pudiendo así tunear el modelo para mejorar los niveles de precisión de sus resultados[54].

La librería de `sklearn` posee una implementación de *Grid Search* con CV, permitiendo fácilmente aplicar *Cross-Validation* a la búsqueda de los mejores hiperparámetros, permitiendo encontrar el mejor tuneo para los modelos utilizados.

2.2.7. LDA

Asignación Latente de Dirichlet es un modelo generativo, o sea que genera valores aleatorios respecto a un dato observable, pudiendo especificar una distribución conjunta respecto a las etiquetas. Concretamente, esta técnica permite explicar conjuntos de observaciones, a través de nuevos conjuntos no observados, determinando relaciones similares entre ciertos datos. Usualmente, esta colección de datos está dada por un corpus de texto, con el cual, se logra desprender una determinada cantidad de tópicos tratados, en un conjunto de documentos[55].

De manera más coloquial, LDA se utiliza para realizar modelamiento de temas. Esto es un tipo de modelo estadístico de lenguaje, usado para descubrir estructuras ocultas en una colección de textos, agrupando estas estructuras en los tópicos generados. En esta técnica, los textos deben ser preprocesados para su posterior análisis, teniendo que eliminar acentos, puntuaciones y conectores, y transformar las palabras con mayúsculas a minúsculas. De esta forma el documento solo queda con *keywords*, palabras claves[56]. En la Figura 2.12, se puede apreciar un diagrama que explica las diferentes utilidades del modelamiento de tópicos:



Fuente: Tomado de "*Topic Modeling in Python : Using Latent Dirichlet Allocation (LDA)*" por P. Srinivasan, 2020.

Figura 2.12: Usos del modelamiento de tópicos

Como se visualiza en la Figura anterior, el modelamiento de tópicos, da como resultado la cantidad de tópico especificados previamente, los cuales se componen de diferentes *keywords* cada uno. Analizando las palabras que componen cada tópico, se puede designar un nombre característico a la temática abordada en cada tópico. Por otro lado, un documento puede abordar varios tópicos, pudiendo establecer porcentajes de pertenencia a determinado tema según los tipos y cantidades de palabras contenidas. Por último, se puede hacer un análisis de los tópicos generados, pudiendo hallar la frecuencia e importancia de cada *keyword* a la generación del tópico.

Capítulo 3

Metodología y Desarrollo

Con el fin de lograr los objetivos propuestos para este trabajo se propone la realización de las siguientes actividades, las cuales responden a los objetivos específicos planteados en la introducción. Se desea probar varias de las técnicas de clasificación y predicción para poder realizar una posterior comparativa a nivel de desempeños, integrando un análisis geográfico de la data.

Para la realización de este proyecto se utilizará el lenguaje de programación `Python`, ocupando librerías como `numpy` y `pandas` para el tratamiento de datos a partir de *Dataframes*. Posteriormente, el desarrollo de modelos de clasificación y predicción hará uso principalmente de librerías como `sklearn` y `keras`, y probará la implementación de LDA en `Python` para el análisis geográfico a partir de la librería `gensim`. Los gráficos y visualizaciones se confeccionarán con `matplotlib`. La versión de `Python` utilizada en el presente para la realización del proyecto es `Python` 3.9.6. Importante destacar que todas las Figuras y Tablas generadas en el siguiente Capítulo de Resultados y Discusión, son elaboradas de manera propia tras seguir la metodología planteada en el actual Capítulo.

Se utiliza `JupyterLab` como interfaz para programar, ya que permite documentar fácilmente los códigos al contar con una estructura de bloques, y además, es capaz de soportar el procesamiento de múltiples códigos simultáneamente. Estos códigos fueron procesados en un computador propio que cuenta con las siguientes especificaciones; Sistema Operativo de Windows 10 Pro, Procesador Intel(R) Core(TM) i7-4700HQ CPU @ 2.40GHz 2.39 GHz y RAM de 16 GB.

3.1. Bases de Datos

La investigación desarrollada es de carácter experimental y cuantitativa, ya que se trabajará en base a datos concretos de la red 4G de Entel, específicamente se emplearan 2 *datasets*; uno con información respecto a diversos aspectos del funcionamiento diario de las celdas de telecomunicaciones de Santiago, y otra que contiene información del mes de Julio del 2020 con una granularidad de 15 minutos entre cada dato.

La base de datos diaria, consta de 117 *features* que caracterizan el funcionamiento diario de celdas dispuestas en la Región Metropolitana del 2019, y cuenta con un total de 2.530.224 instancias. En cambio, la 2da base de datos consta de 120 *features*, difiriendo solamente en 5 columnas respecto a la base de datos diaria, y esta cuenta con 2.206.486 instancias diferentes.

La unidad mínima de los *datasets* se encuentra en el *feature* `EutrancellFDD`, el cual especifica el ID de la celda observada e indica a qué tipo de celda corresponde, pudiendo pertenecer a 3 tipos diferentes dependiendo de su banda de frecuencia de operación; existiendo la banda M que opera en los 700 [MHz], P en los 1900 [MHz] y L en los 2600 [MHz]. Además, la columna `Erbs` especifica el ID de la antena física a la cual está asignada la celda en cuestión, indicando así que una antena puede contener varios `EutrancellFDD` de distintos tipos de banda, o sea varias celdas por antena.

La base de datos diaria cuenta con 7960 celdas diferentes y 1166 antenas, las cuales se ubican en diferentes zonas de Santiago, y por otro lado, la base de datos con *timestamp* de 15 minutos tiene 761 celdas distintas, las cuales pertenecen a 106 antenas dispuestas en el sector poniente de Santiago, en las alrededores de la comuna de Maipú.

Como se mencionó anteriormente, los *features* de los *datasets* corresponden en general, a KPIs de la operación diaria o de cada 15 minutos, según corresponda, de celdas que componen la red de telecomunicaciones de Entel en Santiago. Estos KPIs se dividen en categorías como Accesibilidad, Retenibilidad, Integridad, Movilidad, Capacidad, Disponibilidad, Volumen de Tráfico, CSFB(*Circuit Switching Fall Back*), BLER(*Block Error Rate*), entre otros. En particular, para la confección de la base de datos diaria, se pueden tomar dos enfoques diferentes: Máximo por volumen o Máximo por cantidad de usuarios. La utilización de un enfoque u otro depende del área de uso de cada feature. En caso de que una variable se calcule con máximo por volumen, se identifica la hora en que se obtuvo el volumen máximo, y se escoge el dato de dicha hora para la representación de dicho día para esa variable. Para el máximo por cantidad de usuarios es análogo.

De esta forma, se puede evidenciar que el *dataset* diario no es tan representativo, ya que condensa demasiada información, acerca del funcionamiento a nivel diario de una celda, en un solo dato. Esto se tiene en cuenta para la posterior aplicacion de modelos de ML.

3.2. Preprocesamiento

Antes de ingresar la data a cualquier modelo, esta debe ser pre-procesada de manera que los resultados obtenidos sean consistentes y tengan sentido.

Es por ello, que a continuación se especifican varios de los pasos necesarios para la obtención de los *Dataframes* a procesar con las herramientas de ML:

3.2.1. Base de datos de *timestamp* diario

Como se mencionó anteriormente, esta base de datos cuenta con 117 features, de los cuales, al analizar la cantidad de datos `NaN` que existen en cada columna, se evidencia que ciertos KPIs no aportan realmente por su cantidad de información nula. Es por ello que se aplica un filtro sobre los *features* para eliminar aquellos que contengan más de un 70 % de data nula, obteniendo así un *dataframe* filtrado de 86 columnas.

Posteriormente, se agregan columnas de fechas a partir de la columna `date_id`, de manera de poder contar con la información del día, semana, día de la semana y mes para cada instancia. Además se verifica que no existan filas repetidas, eliminando aquellas entradas duplicadas a partir de las columnas: nombre de antena, nombre de celda, mes, semana y día.

Como la idea es poder trabajar con aquellas celdas que tengan información todos los días del año de manera continua, se hace una búsqueda de aquellas que cumplan con el requisito anterior, dando como resultado que 3666 celdas tienen data todos sus días, reduciéndose el *dataframe* a un total de 1.338.090 instancias, y de las cuales 969 celdas operan en 700[MHz] (Banda M), 1274 en 1900[MHz] (Banda P) y 1423 a 2600[MHz] (Banda L).

Para poder ingresar este *dataframe* a un modelo de ML, para su posterior entrenamiento, se necesita primero hacer un tratamiento de los valores `NaN` existentes. Este tratamiento de `NaNs` puede consistir en un rellenado de datos con valores 0, -1, el promedio o mediana de los datos, entre otras opciones, pero la escogida para este proyecto, fue aplicar una interpolación respecto a los 4 datos vecinos del dato faltante, para ello se hace un ordenamiento del *dataframe* según antena, tipo de celda, celda, fecha, mes y día, y luego se aplica `KNNImputer`, librería capaz de realizar interpolación a partir de los datos de K vecinos, para las filas de cada celda y para todas las columnas que contengan `NaNs`.

3.2.2. Base de datos con *timestamp* de 15 minutos

De manera análoga a lo realizado con la base de datos diaria, primero se revisa la cantidad de columnas con alto índice de `NaNs`, para posteriormente filtrar aquellas que tengan un valor mayor al 70 % del total de los datos, reduciendo así los *features* del *dataframe* a 67 columnas.

Luego, se arreglan ciertos errores de tipeo existentes en la base de datos respecto a la columna `DATETIME_ID`, de manera de poder generar nuevas columnas en relación a los *timestamps* como; minuto, hora, día, día de la semana y semana. Además, se comprueba si existen entradas duplicadas en relación a estas columnas de fechas y las celdas.

Como se tiene una granularidad de un dato cada 15 minutos, cada celda debería contar con 4·24·31=2976 datos, pero esta situación no ocurre para ninguna celda, lo cual indica que existen fechas faltantes en todos los datos. Es por ello, que se generan nuevas columnas que permitan caracterizar la falta de datos. A continuación, se presenta el significado de cada una de las columnas generadas:

DiasNoData: Cantidad de días que X celda no tiene data en su totalidad

Cant_DiasNoData_NoConsecutiva: Cantidad de agrupaciones de días consecutivos con falta de data

DiasNoData_Consecutiva: *True*, solo si tiene una agrupación de días consecutivos

DiasCosecutMAX: Largo máximo de días consecutivos con falta de datos

NoDataDia_MAX: Cantidad máxima de datos faltantes entre los días con falta de datos

NoData_SoloEnHoraUtil: *False*, si tiene falta de datos en horarios no Utiles:= [1,2,3,4,5,6,7]

horas_falt_MAX: Cantidad máxima de cantidad de horas con datos faltantes entre los días con falta de datos

horas_falt_Noconsec_MAX: Cantidad máxima de agrupaciones de horas consecutivas con falta de data entre los días con falta de datos

HorasCosecutMAX: Largo máximo de horas consecutivos con falta de datos entre los días con falta de datos

Con esta columnas generadas, se analiza el *dataframe* a través de la funcion `.describe`, la cual arroja resultados del promedio, mediana, desviación estándar, mínimos, máximos y cuartiles de cada columna del *dataframe*. Con lo anterior, se logra establecer umbrales para estos nuevos *features*, de manera de poder ejecutar un filtro que deje solo aquellas celdas que sean óptimas para una futura interpolación de datos. Los umbrales escogidos fueron:

$$
\begin{aligned}
DiasNoData &< 4 \\
DiasCosecutMAX &< 4 \\
NoDataDia\_MAX &< 20 \\
NoData\_SoloEnHoraUtil &= True \\
horas\_falt\_MAX &< 8 \\
horas\_falt\_Noconsec\_MAX &< 2 \\
HorasCosecutMAX &< 8
\end{aligned}
$$

Al aplicar el filtro, se obtiene una cantidad de 364 celdas diferentes candidatas para un proceso de rellenado de *datetimes* faltantes. Al realizar este proceso, se hace el tratamiento de data `NaN` generada por la misma operación de rellenado, aplicando `KNNImputer` con los 4 datos de sus vecinos cercanos. Esto es ejecutado posteriormente a un previo ordenamiento del *dateframe* con respecto a: tipo de celda, celda, dia, hora y minuto.

El resultado final es un *dateframe* con 1.083.264 instancias, de las cuales 78 celdas operan en 700[MHz](Banda M), 142 en 1900[MHz] (Banda P) y 144 a 2600[MHz] (Banda L).

El *dataframe* resultante de esta base de datos, se utilizará en el entrenamiento de la red LSTM que permitirá realizar pronósticos respecto a una variable de interés y en la aplicación de LDA para la identificación del patrón de uso de las antenas. Esta salvedad, se debe a que esta base datos, al tener *timestamps* de cada 15 minutos, permite capturar el detalle del comportamiento de las celdas, situación que no se puede replicar en el dataset diario.

3.3. Visualización de la data

Antes de analizar la data con modelos computacionales, es importante poder comprender de manera general cómo se comportan los KPIs de interés contenidos en los *features* disponibles en los *datasets*.

En particular, se desea analizar el `DL_User_Throughput`, graficando su comportamiento según el tipo de celda e identificando sus niveles de desempeño de manera semanal, mensual y anual. Esto permite evidenciar, a priori, si existen comportamientos estacionales o anómalos en los KPIs observados. Se utilizará la data de la base de datos diaria para esta visualización previa, ya que, de esta manera, se puede ver en forma macro el comportamiento del `DL_User_Throughput`.

Para las visualizaciones semanales, primeramente se escoge una celda objetivo, y luego se escoge un mes. Luego, se identifica el número de las semanas que corresponden a dicho mes para, posteriormente, agrupar la data de las columnas `DL_User_Throughput` y `Day_of_Week` de aquellas filas que correspondan a las semanas buscadas. De esta manera, se puede utilizar la información del día de la semana como eje x, y los valores del *feature* de interés como eje y, ploteando de forma simultánea las 4 semanas escogidas.

Para las visualizaciones mensuales, se emplea un procedimiento parecido al anterior, salvo que ahora, una vez escogida la celda objetivo, se agrupa la data de las columnas de `DL_User_Throughput` y `day` para todos los meses del año, definiendo como eje x los días, y como eje y la variable de interés, ploteando de forma simultánea los 12 meses.

Finalmente, el gráfico anual se puede generar simplemente haciendo un plot de los datos de `DL_User_Throughput` de una celda en específico.

Los gráficos semanales, mensuales y anuales de ciertas celdas se presentan en el Capítulo siguiente de Resultados.

3.4. Clasificación

Uno de los primeros objetivos del uso de herramientas de ML, en data de redes de telecomunicaciones, es poder identificar diferencias en las celdas analizadas, pudiendo así generar clasificaciones de acuerdo a determinados puntos de análisis. En particular, uno de los ámbitos de interés es poder determinar el desempeño de una celda, a partir del comportamiento de sus *features*.

Es posible clasificar las instancias según el desempeño, a partir de los valores de la columna `DL_User_Thp`, los cuales indican el *download throughput* promedio de los usuarios [Kbps]. Se considera que la celda está en funcionamiento Normal si su valor es mayor a 3,3 [Mbps], Bajo si se encuentra entre los valores 3.3 [Mbps] y 2 [Mbps], y Nulo si es menor a este último umbral. Con estas etiquetas se puede entrenar modelos que permitan clasificar las celdas según estos umbrales de desempeño. La generación de etiquetas de desempeño se realiza por celda y por día, generando una nueva columna que almacena estos *labels*, llamada `Performance`. A partir de esta nueva columna, se puede generar también una que determine si la celda tuvo alguna baja de desempeño en algún día del año o no, llamada `Perf_anual`.

Dado que la data preprocesada de la base de datos diaria aún es de tamaño considerable, y los modelos de clasificación se entrenan de mejor manera al reducir la cantidad de columnas a las estrictamente necesarias, las cuales aportan valor real al trabajo de clasificación, se utiliza RFE CV para identificar cuáles son los *features* importantes. Para ellos se aplica un estimador tipo, como un *Decision Tree Classifier* sin mayor especificación de hiperparámetros, y se usa `RFECV` con *cross-validation* de `StratifiedKFold` de 5 capas y *scoring* enfocado en *accuracy*. Posteriormente, se solicita ordenar las columnas por el *ranking* obtenido, y se filtran solo aquellas que tengan *ranking* 1, o sea aquellas más relevantes para el modelo.

De lo anterior, se logra desprender que al filtrar el *dataframe* por aquellas las columnas que aportan más información para la clasificación, se logra reducir su cantidad de*features* a 35:

[`Int_ActiveDRBsDl`, `Erbs`, `Avg_Sched_Ue_TTI_DL`, `Active_Users_DL`,
`DL_TRAFFIC_VOLUME_GB`, `Disc_Packets_AQM`, `Avg_Erl`, `Ret_ERabDrop`,
`Ret_UeCtxtRelAbnormalENB`, `uso_prb`, `DL_User_Thp`, `EutrancellFDD`,
`Ret_UeCtxtRelMME`, `BestCellEvalReport`, `UL_TRAFFIC_VOLUME_GB`,
`Avg_Sched_Ue_TTI_UL`, `DL_Sched_User_THROUGHPUT`, `DL_Cell_THROUGHPUT`,
`DL_DRB_THROUGHPUT`, `CSFB_WCDMA`, `Int_AverageDlRlcDelay`, `Ue_Ctxt_Ab_Rel_STN`,
`CB_RA_SR`, `UL_User_Thp`, `UL_Packet_Loss_Rate`, `Int_DlRadioThroughput`, `DL_Latency`,
`Int_AverageDlMacDelay`, `date_id`, `Int_MacHarqDlSuccRate`, `Avg_No_ERABs`]

Con el *dataframe* acotado a la información de las columnas especificadas anteriormente, se vuelve a procesar el *dataframe* para trabajar a nivel semanal para cada celda. Para ello, se obtienen los valores de las medianas de los *features*, y se generan las columnas `Perf_Semanal` y `Dias_PerfBaja`, los cuales indican si se tuvo desempeño Normal o Bajo en una semana y cuántos días a la semana una celda estuvo en situación de Bajo o Nulo desempeño, respectivamente.

Como resultado de lo anterior, el *dataframe* se redujo a 190.632 instancias, de las cuales 171.060 datos tienen `Perf_Semanal` Normal y 19.572 Bajo, y al analizar su comportamiento macro anual a partir de la columna `Perf_anual`, se obtiene que 112.788 instancias tienen desempeño Normal y 77.844 Bajo, correspondiendo a un 59.17 % y un 40.83 % respectivamente.

Se trabaja sólo con aquellas celdas que tengan `Perf_anual` Bajo, y particularmente, solo con aquellas que operen en la banda M. De esta manera, se entrenan los modelos con datos que funcionen en escalas parecidas, ya que el comportamiento de los *features* varía entre las diferentes bandas de frecuencia de operación. Con lo anterior, el *dataframe* se reduce a 29.744 instancias, con 572 celdas diferentes.

Como se pudo evidenciar en las cifras anteriores, existe un desbalance en la cantidad de instancias de datos con desempeño Bajo v/s aquellos con desempeño Normal. Para generar un dataset más balanceado, que permita un mejor entrenamiento de los modelos de clasificación, existe la posibilidad de realizar *OverSampling* o hacer una reducción del dataset. Se optó por no aplicar *OverSampling*, debido a que ya se había realizado previamente un proceso de interpolación para la data `NaN`, por lo cual, se decidió aplicar un filtro, que permitiera escoger aquellas celdas que cumplen con tener una cantidad menor de casos de desempeño Normal a un determinado umbral escogido. Este umbral, se determinó de manera que se cumpliera que, la cantidad de clases entre los dos tipos de desempeño, fueran similares para la semana 40 del año 2019, cumpliéndose así, que las clases se logran balancear cuando se escogen aquellas celdas que tienen menos de 49 semanas con desempeño Normal. Esto se comprueba al analizar los valores de `Perf_Semanal`, en donde se obtiene un 53.99 % con data Normal, y un 46.01 % con Bajo.

Como resultado, se reduce el *dataframe* a 20.020 instancias, con 385 celdas diferentes. Este *dataframe* es el definitivo para ingresar a las funciones que aplicarán modelos de clasificación. El set de datos de training, se generará a partir de las primeras 39 primeras semanas del año 2019, y se testeará con la semana 40, para posteriormente poder evaluar el desempeño de los modelos ya entrenados con las semanas posteriores a la semana 40. Esta partición de *train* y *test* se realiza a partir de una función programada, la cual permite también generar una nueva columna llamada `delta_semana`, la cual especifica la diferencia de la semana objetivo de *test*, con el resto de las semanas del *dataframe* de entrenamiento, siendo 0 el valor de las misma semana de testeo, 1 la semana anterior, y así sucesivamente. Esta diferenciación, se realiza con el fin de poder predecir clasificaciones de las celdas para semanas futuras.

Por lo anterior, se genera otro *dataframe* que permita desplazar la variable objetivo en una semana para cada celda, pudiendo entrenar los modelos para generar predicciones para la semana siguiente a la información de los datos.

La variable objetivo para los modelos de clasificación vendría a ser la columna `Perf_semanal`, y además, para no generar *overfitting* en los modelos, se procede a omitir ciertas columnas que podrían entorpecer el entrenamiento de los mismos por no aportar mayor información o por tener una relación directa con variable objetivo, como los siguientes *features*:

[`index`, `Perf_Semanal`, `Perf_anual`, `Dias_PerfBaja`, `Type`, `DL_User_Thp`, `week`]

Es relevante mencionar que para las columnas tipo categóricas, como el nombre de las celdas `EutrancellFDD`, se requiere realizar una codificación numérica para poder trabajar en los modelos, es por ello que se utiliza la librería `LabelEncoder` de `Sklearn` con una modificación a la librería original para poder trabajar con valores nuevos al probar los modelos ya entrenados en *dataframes* diferentes a la data utilizada para el entrenamiento, de esta manera se puede comprobar el funcionamiento de los modelos en otras bases de datos, como podría ser data de años diferentes.

Se prueban diferentes modelos de clasificación como *Bayes*, Redes MLP, *Random Forest* y *XGBoost*, y se utiliza GridSearch con *CrossValidation* en *Random Forest*, *XGBoost* y las Redes MLP, para hallar los hiperparámetros más óptimos para estos clasificadores. Luego, se grafica sus desempeños contrastando sus curvas ROC al testear las clasificaciones realizadas para la semana 40. De manera análoga, se realiza lo anterior para los modelos que permiten predecir las clasificaciones de las celdas para la semana siguiente, evidenciando a través de los gráficos y puntajes AUC ROC, qué modelo clasifica mejor.

Finalmente, con el mejor modelo escogido para la clasificación y para la predicción de clasificación, se testea sus resultados en semanas posteriores a la semana 40, visualizando la matriz de confusión y las métricas de desempeño de las clasificaciones realizadas.

3.5. Predicción

La metodología utilizada para las predicciones es similar a la que se explicó en la Sección anterior de Clasificación, de hecho, los *dataframes* utilizados son los mismos resultantes de todo el procesamiento planteado anteriormente. Esta sección, difiere de la Clasificación solamente en aspectos como: la variable objetivo a utilizar, los modelos predictivos probados y el umbral escogido del filtro de celdas que permite balancear las clases del *dataframe*.

La variable objetivo escogida, es la columna `Dias_PerfBaja`, que se tiene a nivel semanal para cada celda. Esto indica, que lo que se desea predecir, es la cantidad de días con desempeño bajo, que tendrá cada celda en una semana determinada, dada la información semanal de los diversos KPIs ingresados en la data de entrada de los modelos. Esto se probará realizar para un modelo que utiliza data de la misma semana a predecir y para un modelo capaz de predecir la cantidad de días desempeño bajo que tendrá cada celda en la siguiente semana. Por lo anterior, al igual que para las clasificaciones, se genera un *dataframe* con la variable objetivo desplazada una semana para cada celda, de manera de entrenar los modelos para predecir datos de la semana siguiente a la información de entrada.

La variable objetivo es un valor discreto de 0 a 7, ya que es la cantidad de días que una celda puede tener desempeño bajo en determinada semana. Por lo anterior, al aplicar el filtro en el *dataframe*, que permite hallar las celdas que cumplen con estar bajo un determinado umbral de cantidad máxima de semanas de desempeño Normal, se tiene que el balance de clases es diferente al realizado en la Sección de Clasificación. Ahora se busca balancear entre las diferentes posibles cantidades de días con desempeño Bajo, mientras que en el caso anterior, se contaba con una situación de clases binarias para la clasificación. Es por ello, que al ir probando diferentes umbrales, se encuentra que el valor máximo de cantidad de semanas con desempeño Normal igual a 25, es aquel que presenta mayor balance dentro de los posibles valores de la variable objetivo, para las celdas de la semana 40 del 2019. Para este filtro se tiene que la variable `Perf_Semanal`, presenta un 77.85 % datos con etiqueta Normal, y un 22.15 % con Bajo.

Como resultado de lo anterior, el *dataframe* obtenido cuenta con 8.216 instancias, con 158 celdas diferentes, data que es directamente ingresada a las funciones que permiten entrenar los modelos predictivos. Los modelos se entrenan con las primeras 39 semanas del año 2019, y se testea con la semana 40, de igual manera que las clasificaciones. Se prueba *Random Forest Regressor* y *XGBoost Regressor*, y se utiliza `GridSearchCV` para hallar la mejor combinación de hiperparámetros para los modelos, tomando como *score* métricas de desempeño como MAE y el RMSE.

Los resultados de los modelos predictivos utilizados son de carácter continuo, es por ello que se debe redondear y aplicar limites inferiores iguales a 0 y limites superiores iguales a 7 para la comparación entre la data real y la data predicha, pudiendo representar los resultados con una matriz de confusión y realizar comparativas respecto al desempeño de las predicciones

con las métricas MAE, MAE %, RMSE y RMSE %. Estas comparativas se realizan al testear los modelos ya entrenados en data de semanas posteriores a la semana 40 del año 2019.

3.6. Pronóstico

Teniendo en cuenta que, uno de los objetivos de este trabajo es, poder predecir los valores futuros de alguno de los indicadores KPIs de la red de telecomunicaciones, de manera de poder conocer con antelación el comportamiento de determinadas celdas y así poder tomar decisiones técnico-económicas eficientes, se hace uso de técnicas capaces de entregar pronósticos respecto a las variables de interés. En particular, se desea predecir el comportamiento del *feature* `DL_User_Thp`, y para ello se utilizan las redes neuronales LSTM.

Para esta labor, se utiliza el *dataframe* resultante del preprocesamiento de la base de datos con granularidad de 15 minutos, a la cual se le agregan 3 nuevas columnas: `Perf_Diaria`, `Perf_Semanal` y `Perf_Mensual`, que como su nombre indican, presentan las etiquetas de la clasificación de desempeño a nivel diario, semanal y mensual que tiene cada una de las celdas analizadas, ocurriendo así que si una semana contiene un día con desempeño Bajo, la semana completa se considera que tuvo desempeño Bajo, y de manera análoga a nivel mensual.

Al filtrar el *dataframe* respecto a los valores de la columna `Perf_Mensual`, de manera de escoger solo aquellas celdas que cuentan con desempeño Bajo, y luego, filtrando estas celdas para trabajar sólo con aquellas que cumplan con tener una cantidad de instancias de desempeño Bajo que sumen al menos un total de 7 días, o sea 672 instancias (4·24·7), se obtiene un *dataframe* que cuenta con 71.424 instancias y 24 celdas diferentes, de las cuales 14 pertenecen a la banda M, 7 a la banda P, y 3 a la banda L.

Utilizando sólo las 14 celdas de la banda M, se reduce el *dataframe* a 41.664 instancias. Estos datos corresponden al comportamiento de celdas con Bajo desempeño en el mes de Julio del 2020. Como se desea realizar pronósticos respecto al *feature* `DL_User_Thp`, con un enfoque de series de tiempo, se filtran las demás columnas y se reorganiza el *dataframe*, de manera de poder establecer como columnas determinados pasos de tiempo respecto a un instante dado. Además, las redes LSTM deben recibir datos normalizados, por lo que se hace uso de la librería `MinMaxScaler` entre 0 y 1 en el *dataframe* antes de ingresar al modelo predictivo.

Escogiendo una celda, se desea predecir 3 días al futuro, a partir de 1 semana de datos pasados, lo cual se traduce a tener un *delay* de 672 pasos de tiempo y 288 pasos futuros. Como no se desea entrenar el modelo con el conocimiento de la data del 1er y 2do día futuro, solo los datos del 3er día, se eliminan todas las columnas de instantes de tiempo futuro excepto la última, que será la columna de variable objetivo, obteniendo así una matriz de 2017 instancias y 673 columnas, ya que se está trabajando solo con la data de una celda.

El set de *training* consistirá en las primeras 2 semanas de datos y el *test* el resto de la data. La matriz de datos de entrada `X` del modelo correspondería a las primeras 672 columnas del *dataframe*, mientras que la variable objetivo `y` correspondería a la última columna, como se mencionó anteriormente. De lo anterior, se desprende que la matriz `X` *train* tiene dimensiones de (1344, 672), mientras que la matriz `X` *test* posee dimensiones de (673, 672). Por último, la red LSTM tiene un formato específico de *input* de datos, teniendo que contar con 3 dimensiones, y de la siguiente forma: [*samples*, *timesteps*, *features*]. Para este caso, la cantidad de *samples* vendría a ser de 1344 para el *training*, el *timesteps* de 672, y los *features* solo de 1, que corresponde derechamente a la variable de interés: `DL_User_Thp`.

La red LSTM utilizada proviene de la librería `Keras`, basada en `Tensorflow`. La arquitectura de la red se especifica para que cuente con una sola capa oculta, compuesta por 50 neuronas en donde, al fijar el hiperparámetro *stateful* en `True`, se utiliza la información del último estado oculto para las predicciones siguientes, otorgando mayor contexto del comportamiento de los datos a la red mientras se entrena. Se tiene en consideración que se debe tener diferentes *batches sizes* para el entrenamiento y para la predicción, usando un tamaño de *batch* igual al largo de la matriz `X` *train* para el entrenamiento de la red y para predecir se utiliza un *batch* de valor igual a 1, ya que se desea realizar predicciones a partir de una observación a la vez. Se escoge *Adam* como optimizador, con un *learning rate* de 0.001 y se varía la cantidad de épocas utilizadas en el proceso de entrenamiento, probando con 100, 500 y 1000 *epochs*, para posteriormente comparar los errores RMSE obtenidos al evaluar los modelos entrenados en otras celdas. Finalemente, se grafican las predicciones v/s la data real, de manera de visualizar el desempeño de los pronósticos resultantes de las redes LSTM.

Como se mencionó anteriormente, los modelos entrenados se prueban para la predicción del comportamiento mensual de otras celdas tipo M con *performance* Bajo, en particular, se escogen los últimos 672 datos, o sea la información de la última semana del mes de Julio del 2020 para la comparación del pronóstico resultante con la data real. Esto se realiza, debido a que si se escoge una cantidad de datos mayor al *delay* establecido de la estructura de series de tiempo del *dataframe*, los datos de *input* comienzan a contener implícitamente información de la variable objetivo.

3.7. Análisis Geolocalización de Antenas

Para esta sección, se trabaja con ambos datasets disponibles, pero a nivel más macro de lo que se había estado realizando anteriormente, en donde se realizaban pronósticos del comportamiento detallado de un KPI, o se utilizaban varios KPIs agrupados semanalmente para el entrenamiento de modelos predictivos y de clasificación, todo esto a nivel de celdas de la red de telecomunicación. En cambio, ahora se realiza un análisis a nivel de antenas, las cuales se componen por una o más celdas diferentes. En particular, se agrupa la información relativa al volumen de tráfico de bajada, entregada en la columna `DL_TRAFFIC_VOLUME_GB` de todas las celdas pertenecientes a una antena. De esta forma, se obtiene una cantidad total de volumen de tráfico por antena para cada instante de tiempo mínimo de los datasets, o sea a nivel diario y a nivel de cada 15 minutos. Todo esto se realiza con el fin de poder hacer un estudio del patrón de uso de las antenas y su relación con sus ubicaciones geográficas.

Como resultado del preprocesamiento, el dataset de *timestamp* de 15 minutos, se redujo a un *dataframe* de 50 antenas diferentes dadas por la columna `Erbs`. Por lo cual, al aplicar lo explicado en el párrafo anterior, se obtiene un *dataframe* de 148.800 instancias. Luego, se aplica un filtro sobre la columna resultante de los valores agrupados por antena del volumen de tráfico de bajada de datos, llamada `DL_VOLUME`, en donde se desea filtrar aquellas antenas que tienen más de 2 semanas de datos 0 en dicho *feature*, arrojando como resultado que se debe filtrar 1 antena del *dataframe*.

Reorganizando la información del *dataframe* para que pueda ser tratado posteriormente como un corpus de documentos que ingresan al modelo LDA, se generan columnas para cada semana de cada antena poblándolas con los valores normalizados de cada semana correspondientemente. Con esto, el *dataframe* pasa a tener dimensiones de 672 instancias y 196 columnas, cuyos datos deben ser formateados a *string* y posteriormente, llevados a una matriz de lista de listas de *Python* para poder generar así, el corpus de *tokens* de los datos que se desean analizar con el modelo de procesamiento de texto LDA.

El modelo de LDA utilizado, proviene de la libreria `gensim`, la cual permite escoger la cantidad de tópicos deseados a generar a partir del análisis del corpus de documentos ingresados al modelo. En este caso, recapitulando, cada documento equivaldría a las columnas del *dataframe* comentado anteriormente, que son las semanas de todas las antenas, y las palabras de los documentos vendrían a ser los datos normalizados por semana de cada columna del feature `DL_VOLUME` convertidos en *string*. Escogiendo 10 tópicos, se procede a etiquetar cada documento con el tópico que tenga mayor porcentaje de pertenencia, ya que puede ocurrir que un documento tenga relación con hasta 3 tópicos diferentes, puesto que esto depende de las características de su contenido, las palabras que lo conforman.

Utilizando la librería `pyLDAvis`, se puede visualizar por medio de PCA la distribución entre los diferentes tópicos pudiendo identificar *clusters* entre ellos, ya que, cuando determinados tópicos abordan temáticas parecidas, las temáticas se presentan en el gráfico PCA con

mayor cercanía entre ellas, que aquellos tópicos que tratan temas diferentes. Con lo anterior, se identifica visualmente *clusters* dentro de los 10 tópicos creados tratando de formar un máximo de 5 *clusters* que engloben todas las temáticas generadas.

Posteriormente, como cada antena tiene 4 documentos, uno por semana, se tendrá la posibilidad de que una antena esté relacionada a través de cada uno de sus documento, a diferentes tópicos, y por ende, existe la posibilidad de que esté relacionada a diferentes *clusters*. Por lo anterior, se asigna cada antena al *cluster* que se reitere más veces, el *top cluster* dentro de los 4 documentos relacionados a dicha antena.

Con los *top clusters* asignados a cada antena, se grafican las ubicaciones geográficas de cada antena en el mapa de la ciudad de Santiago, diferenciando cada una según su pertenencia a los *clusters* creados. Esto se contrasta con las gráficas del comportamiento mensual de `DL_VOLUME` de cada antena relacionada a su *cluster*, de manera de poder evidenciar, relaciones del comportamiento del patrón de uso de las antenas, con sus ubicaciones geográficas.

Todo lo anterior, se realiza de forma análoga para el dataset diario, en donde solamente se tienen las siguientes variaciones: tras el preprocesamiento del dataset diario y la agrupación de la data de volumen de tráfico de las celdas por antena, se tiene que el *dataframe* resultante contiene 532 antenas diferentes con 194.180 instancias. De las antenas anteriores, 12 son eliminadas por su elevado contenido de valores igual 0 en la columna `DL_VOLUME`. Al reestructurar el *dataframe* se generan columnas de cada mes para todas las antenas, de forma que los datos se normalizan a nivel mensual de igual manera. Esto da como resultado un *dataframe* compuesto por 31 filas y 6240 columnas, que corresponden a los 12 meses de las 520 antenas analizadas, cuyos valores son redondeados a los primeros 4 decimales y luego son formateados a *string* para, posteriormente, pasar a su forma de lista de listas, matriz que es ingresada a LDA. Se vuelve a generar 10 tópicos y a formar un máximo de 5 *clusters*. Esta vez cada antena tiene 12 documentos y ,una vez que se les asigna su *top cluster* a cada una, se procede a graficar sus ubicaciones en el mapa de Santiago, contrastando esto con el gráfico del `DL_VOLUME` de cada antena a nivel anual. Con este dataset se puede realizar un análisis más completo que el obtenido con la base de datos anterior, ya que ahora se tienen muchas más antenas para analizar y así se logra vislumbrar de mejor manera las relaciones entre los patrones de uso y las ubicaciones geográficas.

Capítulo 4

Resultados y Discusión

4.1. Visualización previa

A continuación se presentan algunos de los resultados del proceso de visualización de la data. En la Figura 4.1 se grafica el comportamiento mensual de la variable *Download User Throughput* a nivel semanal de dos celdas.

De la Figura 4.1, se ve que la imagen de la parte superior presenta el comportamiento de una celda que cuenta con mejores velocidades en los fines de semana de Junio 2019 que en otros días de la semana. Además, se evidencia una caída en la velocidad el día miércoles de la primera semana del mes. Por otro lado, la imagen inferior muestra el comportamiento de otra celda del mismo tipo que la anterior, pero que suele tener velocidades mucho más bajas, en general, que la primera celda, observando que el promedio es de aproximadamente 5 [Mbps]. Al inicio de la semana la celda tuvo velocidades similares durante el mes estudiado, y se logra apreciar que existe una baja en la velocidad los días sábados.

Figura 4.1: Comportamiento semanal de celdas

Otra visualización de interés es la del comportamiento mensual que tienen las celdas de la variable *Download User Throughput* en el año 2019. En la Figura 4.2 se presenta el comportamiento anual a nivel mensual de dos celdas.

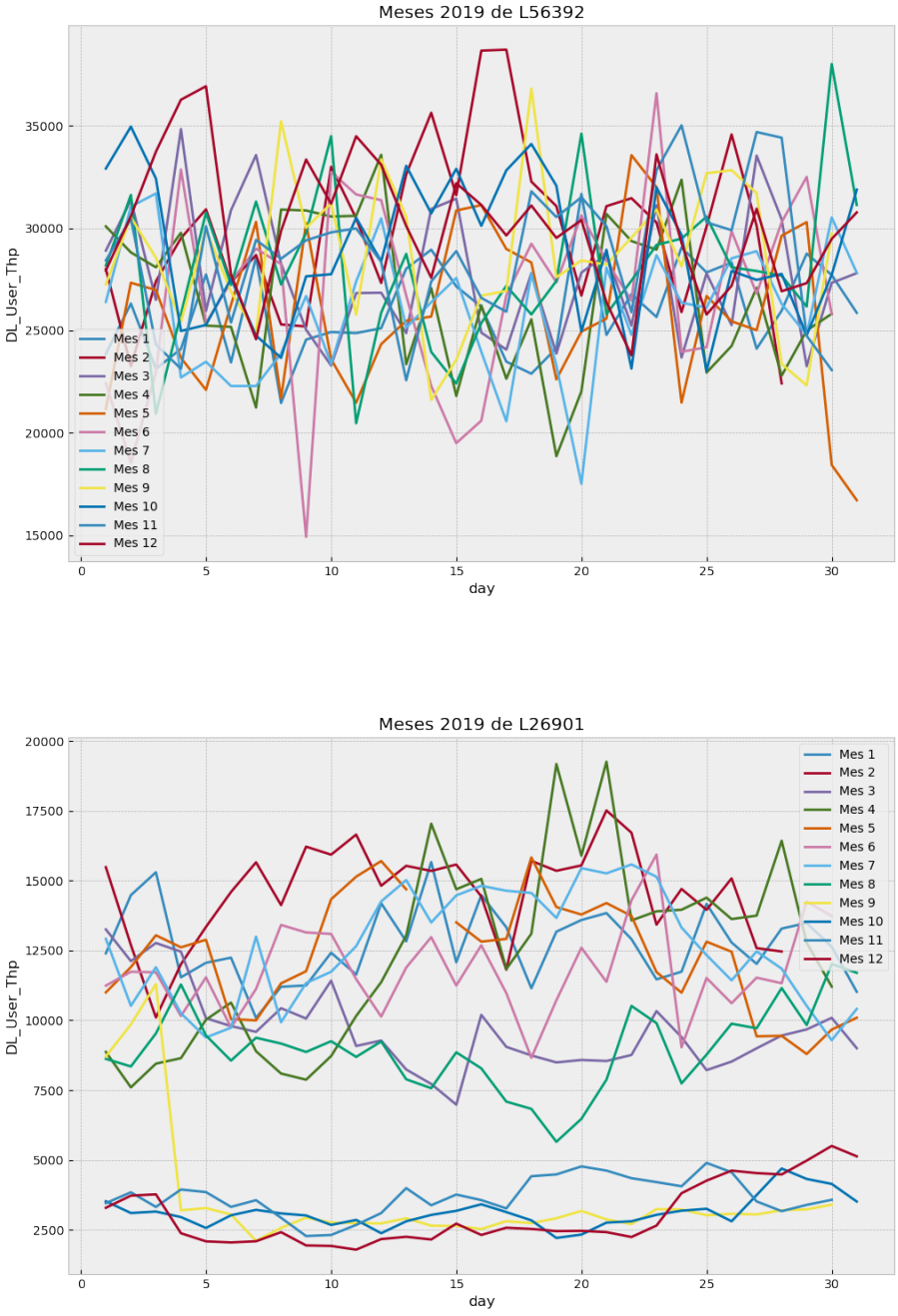


Figura 4.2: Comportamiento mensual de celdas

Se logra apreciar en la Figura 4.2 que el comportamiento a nivel mensual puede variar bastante entre los meses, pero usualmente se mantiene dentro de un rango de valores de operación establecidos como se ve en la imagen superior, en donde se advierte que su operación varía entre los 23 y 33 [Mbps] aproximadamente. En cambio, la celda presentada en la imagen inferior opera entre los 8 y 10.3 [Mbps], pudiendo evidenciar que existe una baja en su operación normal en los últimos 4 meses del año, alcanzando velocidades menores a los 2.5

[Mbps], desempeño que ya es considerado bajo.

Para finalizar, la Figura 4.3 muestra un gráfico anual de dos celdas, en donde se reitera lo planteado anteriormente y es que existe un rango de operación para cada celda. Por otro lado, en la imagen inferior, se vuelve a evidenciar la baja de velocidad en los últimos 4 meses de la celda presentada anteriormente.

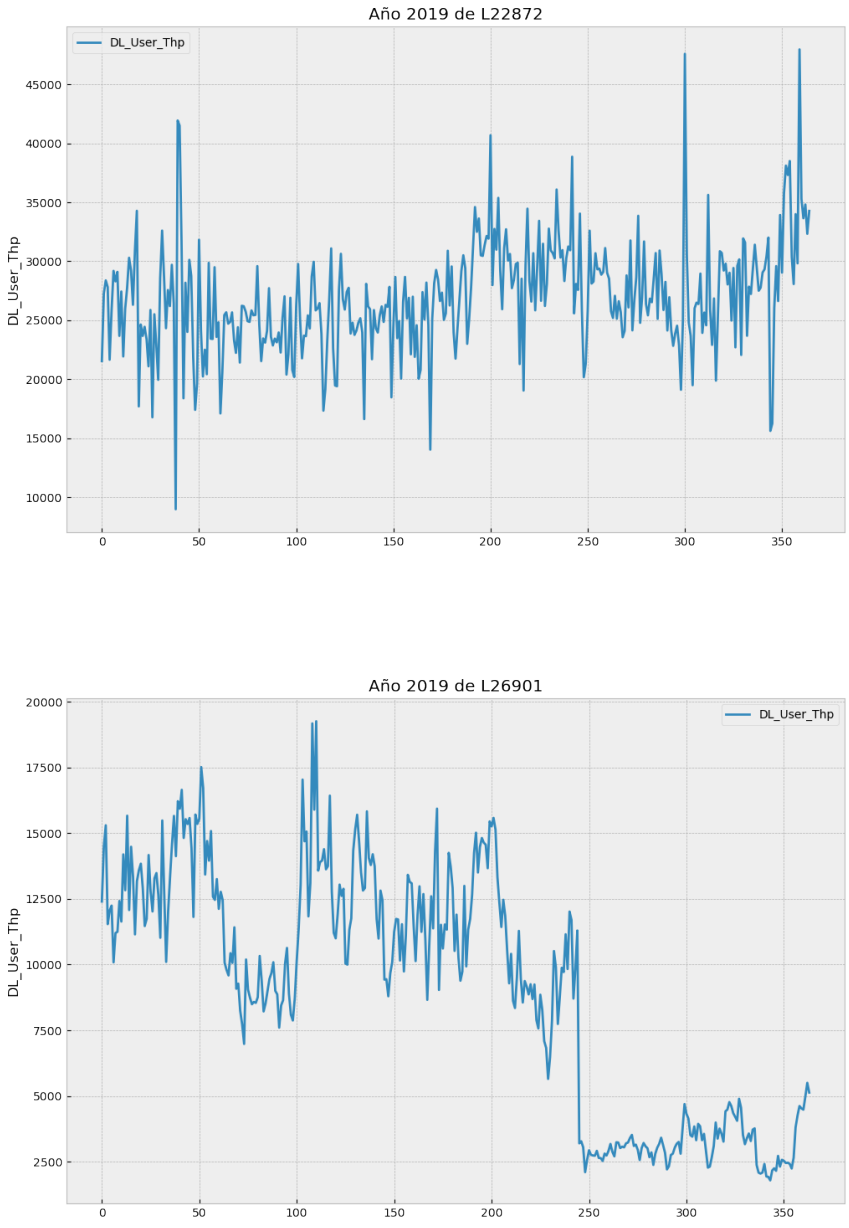


Figura 4.3: Comportamiento anual de celdas

4.2. Clasificación

Como se explicó en el Capitulo anterior, para las clasificaciones y predicciones, se utiliza la base de datos diaria agrupando la información de cada *feature* por semana, a través del valor obtenido por las mediana de cada semana de cada celda.

Habiendo utilizando RFE CV previamente en un clasificador tipo, como un *Decision Tree Classifier*, se logra filtrar los *features* para trabajar con un máximo de 33 columnas, de las cuales, al entrenar el modelo *XGBoost*, se obtienen la siguientes importancias:

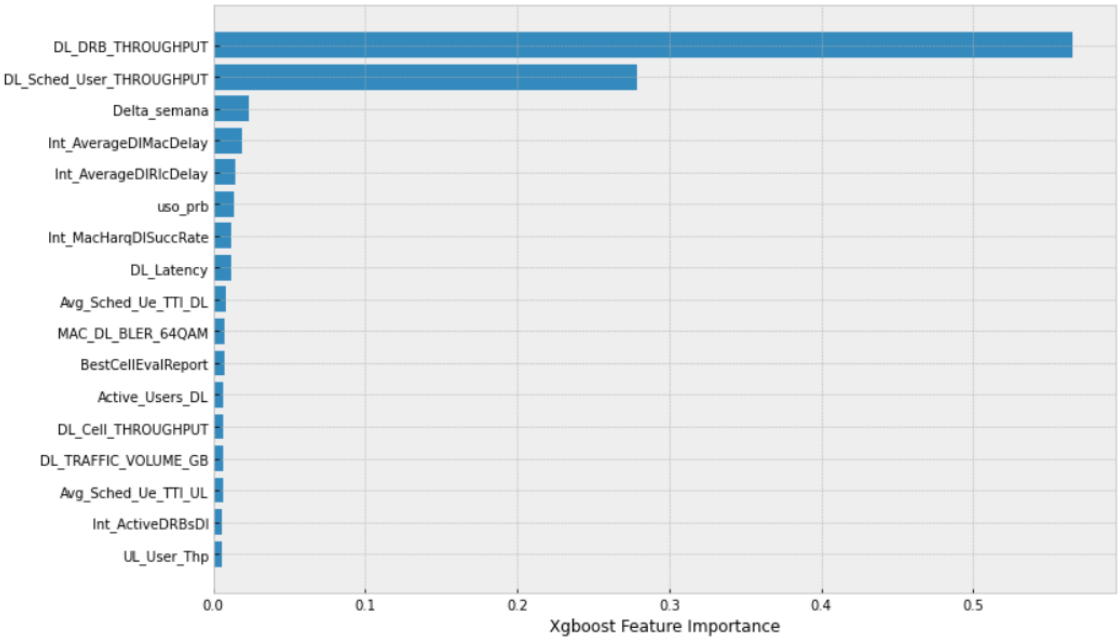


Figura 4.4: *Feature Importance* de modelo de clasificación *XGBoost*

Con estos *features*, se prueban diferentes modelos de clasificación como *Naive Bayes*, Redes MLP, *Random Forest* y *XGBoost*. Entrenando estos modelos con las primeras 40 semanas del 2019, se testean sus desempeño de clasificación con las siguientes semanas restantes en donde, en primera instancia, se evaluan sus desempeños utilizando como variable objetivo las etiquetas binarias de desempeño semanal y posteriormente se evaluan sus desempeños al entrenarlos para poder predecir la clasificación de las etiquetas de la semana siguiente. A continuación, se presentan las curvas ROC al evaluar el modelo en la data de la semana 41 del 2019, en donde se sigue la lógica recién explicada:

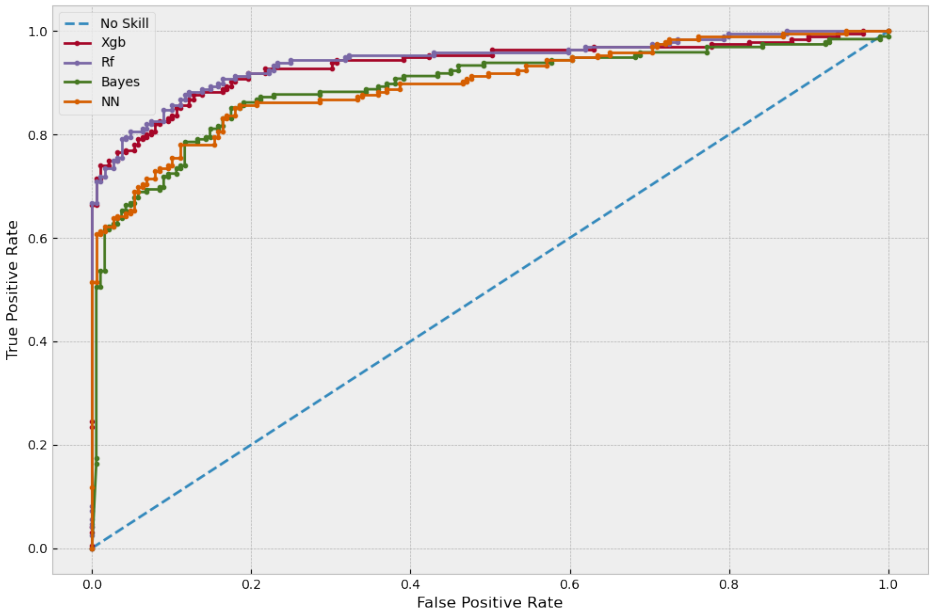


Figura 4.5: Curvas ROC de clasificación de semana 41

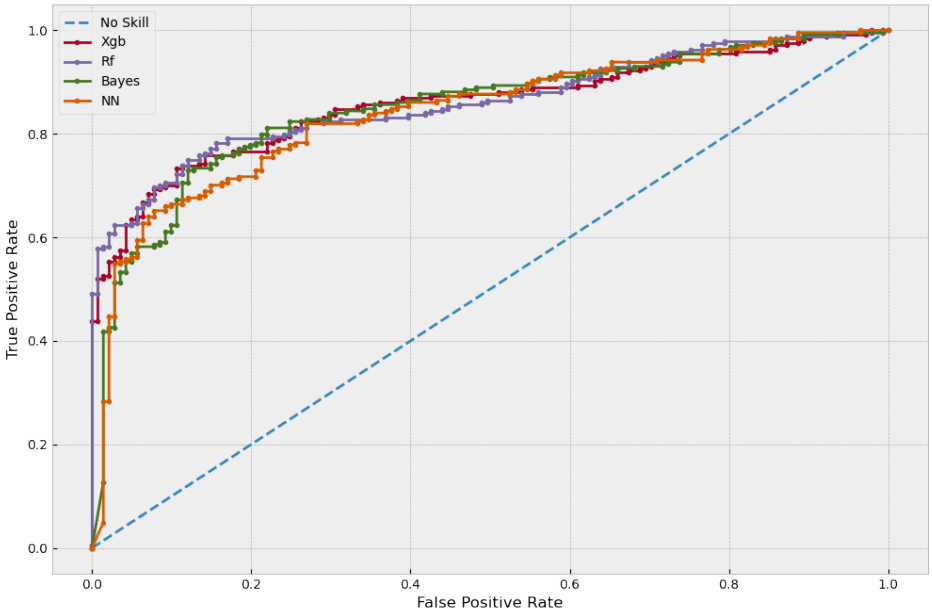


Figura 4.6: Curvas ROC de predicción de semana 42 a partir de datos de semana 41

Los valores de los puntajes ROC AUC para los gráficos anteriores se presentan en la Tabla

4.1:

	ROC AUC	
Modelos\ Test	**Semana 41 Clasificación**	**Semana 42 Predicción**
No Skill	0.5	0.5
XGBoost	0.938	0.862
Random Forest	0.943	0.861
Naive Bayes	0.893	0.850
Red MLP	0.899	0.849

Tabla 4.1: Puntaje ROC AUC utilizando data de la semana 41

Observando la Figura 4.5, se evidencia que los modelos *XGBoost* y *Random Forest* tienen mejor desempeño que las redes *MLP* y *Naive Bayes*, específicamente el área bajo la curva obtenida se visualiza en la Tabla 4.1. En la columna *Semana 41 Clasificación*, se ve que *Random Forest* supera a *XGBoost* en 0.005 puntos. Por otro lado, en la predicción de la clasificación binaria del desempeño de las celdas de la semana 42, a partir de la data de la semana 41, se observa en la Figura 4.6, que todos los modelos tienen un nivel de desempeño parecido, pero al observar los valores AUC ROC de la Tabla 4.1, se evidencia que *XGBoost* logra tener un puntaje mayor que los demás modelos, en particular mejor que *Random Forest* en 0.001 puntos.

Escogiendo el modelo *XGBoost*, se pone a prueba su desempeño para clasificar y predecir la semana posterior con data de las semanas 43 y 50. A continuación, se presentan las matrices de confusión al evaluar el modelo con data de la semana 43 del 2019:

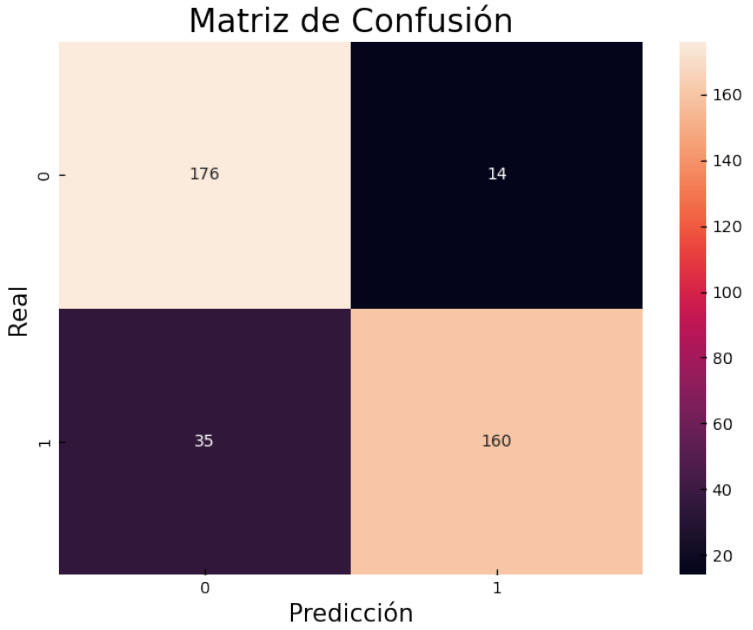


Figura 4.7: Clasificación de semana 43

A partir de la matriz de confusión anterior, se puede generar la Tabla 4.2, que indica las principales métricas del desempeño del modelo:

	precision	***recall***	***f1-score***	***support***
0	0.83	0.93	0.88	**190**
1	0.92	0.82	0.87	**195**
accuracy			**0.87**	**385**

Tabla 4.2: Métricas de desempeño de clasificación de semana 43

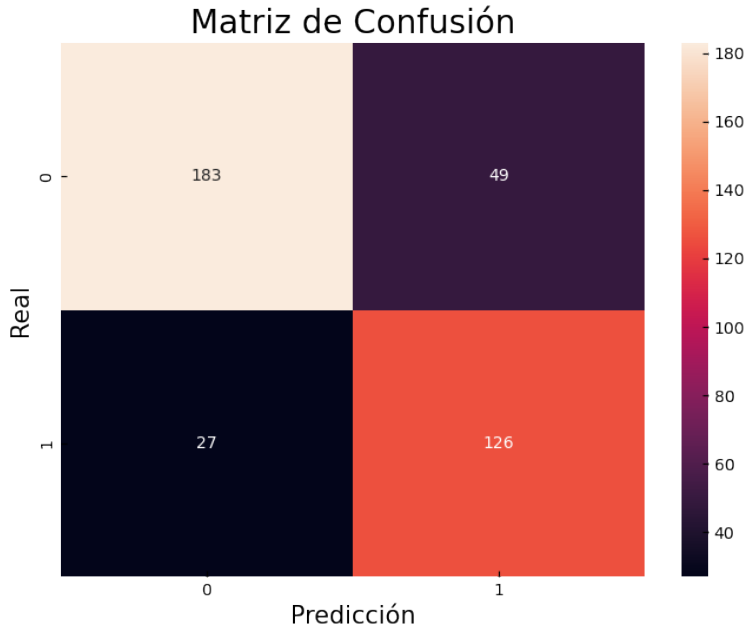


Figura 4.8: Predicción de clasificación de semana 44 a partir de datos de semana 43

A través de la información de la Figura 4.8, se puede generar la Tabla 4.3, que indica las principales métricas del desempeño del modelo:

	precision	***recall***	***f1-score***	***support***
0	0.87	0.79	0.83	**232**
1	0.72	0.82	0.77	**153**
accuracy			**0.80**	**385**

Tabla 4.3: Métricas de desempeño de predicción de clasificación de semana 44

De los resultados anteriores, se logra desprender que la exactitud del modelo para clasificar la semana 43 fue de 0.87 y para la predicción de la clasificación binaria del desempeño de la

semana 44, se obtiene un valor de 0.80, magnitud que indica un nivel de desempeño bastante elevado para el modelo *XGBoost*.

A continuación, se presentan las matrices de confusión al evaluar el modelo con data de la semana 50 del 2019:

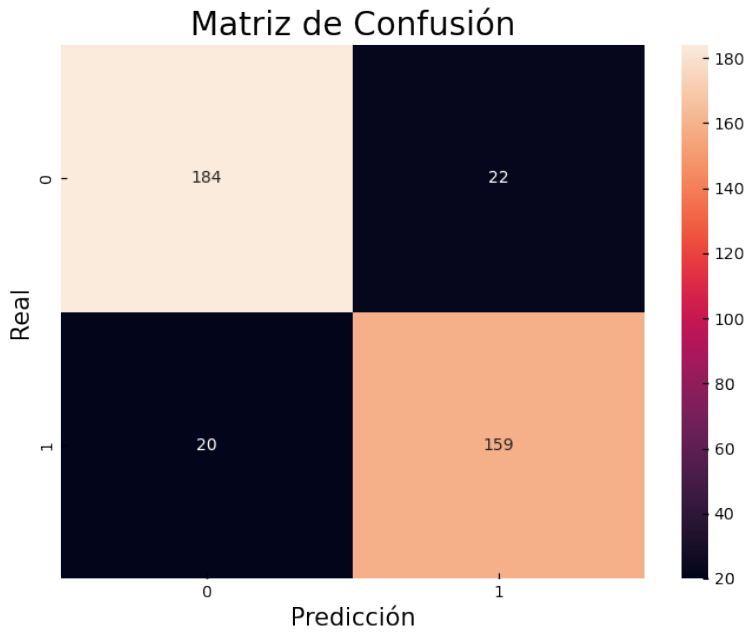


Figura 4.9: Clasificación de semana 50

A partir de la matriz de confusión anterior, se puede generar la Tabla 4.4, que indica las principales métricas del desempeño del modelo:

	precision	***recall***	***f1-score***	***support***
0	0.90	0.89	0.90	**206**
1	0.88	0.89	0.88	**179**
accuracy			**0.89**	**385**

Tabla 4.4: Métricas de desempeño de clasificación de semana 50

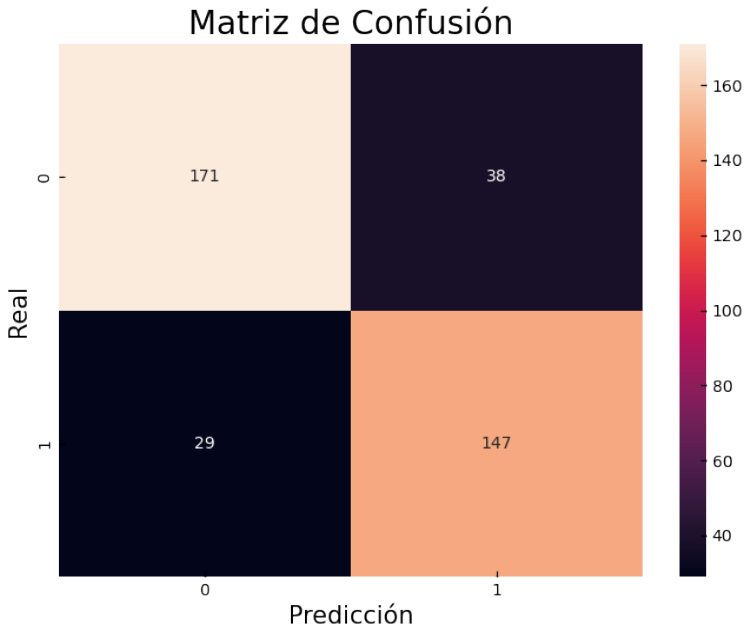


Figura 4.10: Predicción de clasificación de semana 51 a partir de datos de semana 50

A través de la información de la Figura 4.10, se puede generar la Tabla 4.5, que indica las principales métricas del desempeño del modelo:

	precision	***recall***	***f1-score***	***support***
0	0.85	0.82	0.84	**209**
1	0.79	0.84	0.81	**176**
accuracy			**0.83**	**385**

Tabla 4.5: Métricas de desempeño de predicción de clasificación de semana 51

De los resultados presentados, se evidencia que la exactitud del modelo para clasificar la semana 50 fue de 0.89 y para la predicción de la clasificación binaria del desempeño de la semana 51, se obtiene un valor de 0.83, magnitud mayor a lo obtenido en la predicción de clasificación de la semana 44 expuesta anteriormente.

En el Apéndice A, se adjuntan más resultados obtenidos al utilizar modelos *XGBoost*, testeándolos en 2 semanas diferentes a las utilizadas anteriormente.

4.3. Predicción

Utilizando *XGBoost Regressor* y *Random Forest Regressor*, se entrenan modelos predictivos con las primeras 40 semanas del año 2019, de manera de poder predecir la cantidad de días en que una celda tiene bajas de desempeño en una semana determinada. Se entrenan modelos para la realización de una predicción utilizando datos de las medianas de la misma semana, y otros para poder predecir lo que sucede con la cantidad de días con bajas de desempeño para la semana siguiente de cada celda.

A continuación, en las Figuras 4.11 y 4.12, se presentan matrices de confusión de las predicciones realizadas entrenando modelos de *XGBoost* al testear con data de la semana 41 del año 2019:

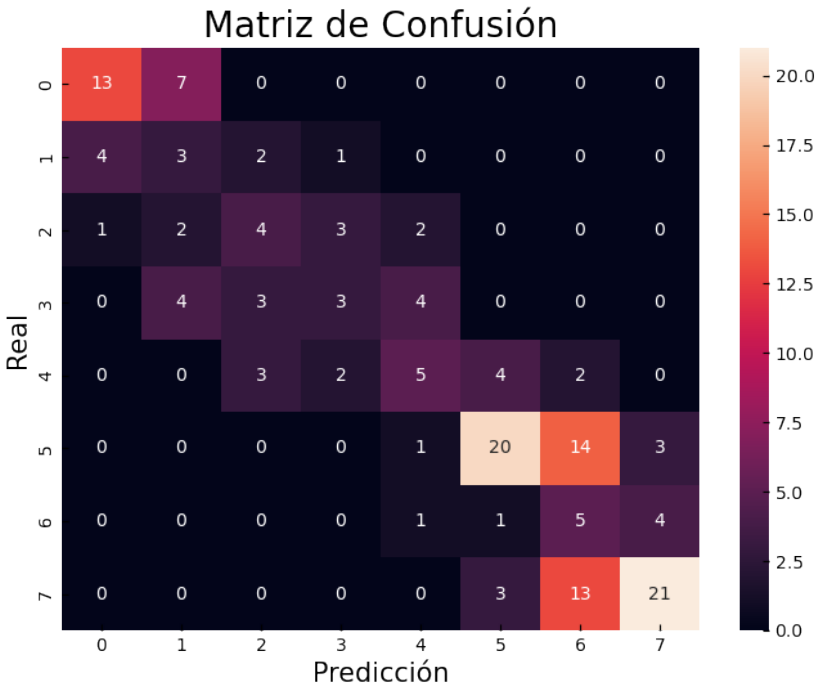


Figura 4.11: Predicción de semana 41

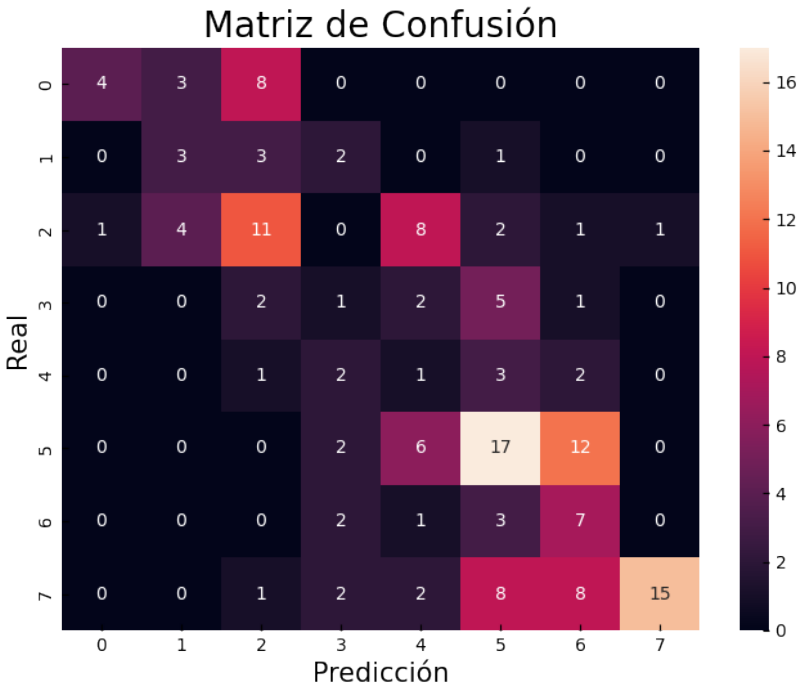


Figura 4.12: Predicción de semana 42 a partir de datos de semana 41

En las Figuras 4.13 y 4.14, se presentan matrices de confusión obtenidas con modelos *XGBoost* al testear con data de la semana 51 del año 2019:

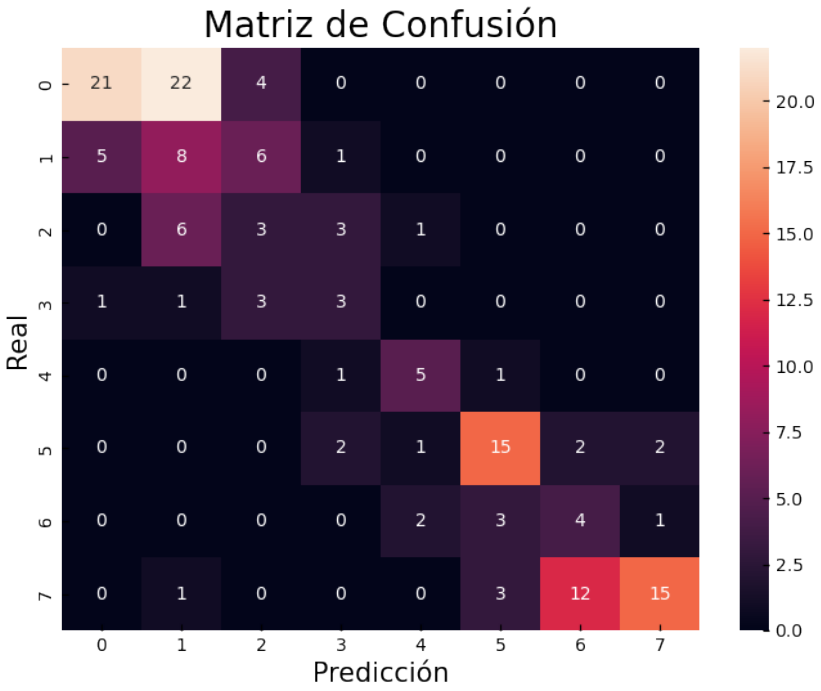


Figura 4.13: Predicción de semana 51

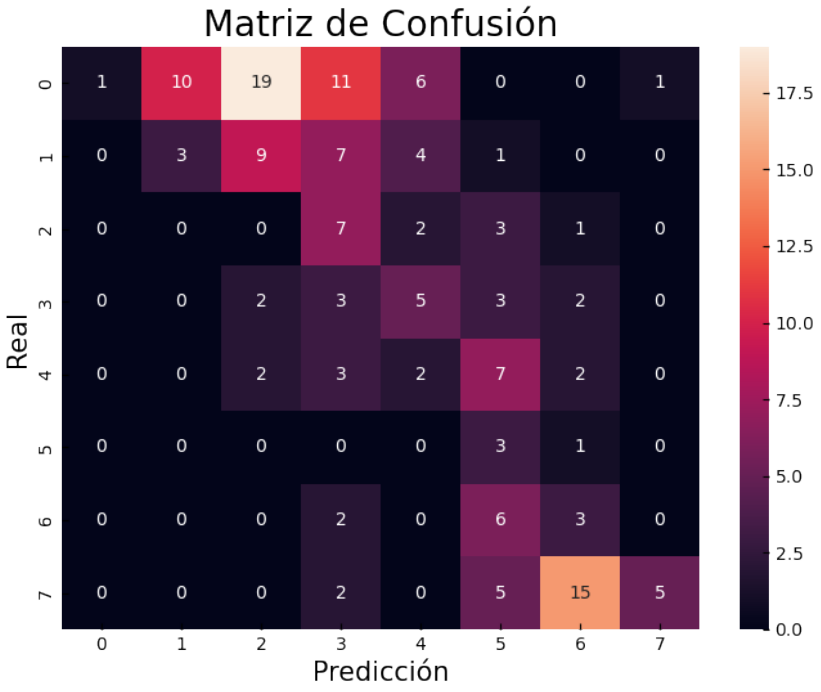


Figura 4.14: Predicción de semana 52 a partir de datos de semana 51

En la Tabla 4.6, se presentan las métricas de desempeño de los 2 modelos predictivos entrenados al testear con data de la semana 41 y 51.

		Métricas			
		MAE	***MAE %***	***RMSE***	***RMSE %***
Pred. de misma semana	***Semana 41***	0.639241	0.4154 %	0.954669	23.0287 %
	Semana 51	0.651899	0.4236 %	1.052423	34.2851 %
Pred. de semana siguiente	***Semana 42***	1.088608	0.6987 %	1.544693	37.6058 %
	Semana 52	1.677215	1.0765 %	2.000000	72.8111 %

Tabla 4.6: Métricas de *performance* de predicciones

De las Figuras anteriores, se puede evidenciar que se obtienen mejores resultados en los modelos que predicen la cantidad de días de desempeño de la misma semana, que aquellos que predicen lo que ocurrirá con las celdas de la semana posterior a los datos semanales analizados. En particular, se ve que en las Figuras 4.11 y 4.13 se va conformando la diagonal que indica la cantidad de predicciones correctas respecto a la data real. El modelo tiene un grado de error de aproximadamente ± 1 día, ya que se ven ciertas cifras en las casillas colindantes a la diagonal de la matriz de confusión, indicando que se está prediciendo erróneamente con una magnitud de 1 o 2 días de diferencia respecto a lo correcto. Por otro lado, el modelo capaz de predecir respecto a la semana posterior, presenta un mayor índice de error en las predicciones, como se puede evidenciar en las Figuras 4.12 y 4.14, las cuales analizan la situación de las celdas de las semanas 42 y 52 respectivamente. En dichas Figuras, se evidencia que la transversal que se conforma a partir de las predicciones realizadas, no ocupa el lugar de la diagonal de la matriz de confusión, sino que suele estar trasladada hacia la derecha de acuerdo a lo que se presenta respecto a las magnitudes de las casillas. Esto indica que se tiene un mayor índice de error, existiendo más cantidad de predicciones incorrectas que correctas, con un error de 2 o 3 días.

En términos de las métricas de error de los modelo, se puede apreciar en la Tabla 4.6, que al tratarse de una diferencia discreta respecto a la cantidad de días en que una celda se encuentra con *performance* bajo, las magnitudes de los errores no son muy elevadas y más aún cuando el error no suele ser mayor a 2 días de diferencia con respecto a la cantidad de días real. Comparando los valores MAE y RMSE del modelo que predice el acontecer de la misma semana de datos de estudio v/s aquel que predice la semana siguiente, se evidencia, tal como se pudo observar cualitativamente con las matrices de confusión, que se tienen magnitudes de error menores para el primer modelo en relación al 2do. Por ejemplo, el valor mínimo de error RMSE dentro de los resultados presentados, se obtiene al realizar la predicción del primer modelo para la semana 41, el cual es de 0.954, mientras que el valor máximo de error se halla en la predicción de la semana 52 del 2do modelo, obteniéndose un valor de 2. Además, las métricas porcentuales dan luz al error obtenido escalado a los valores reales, evidenciando que para el caso de menor error se tiene un porcentaje de RMSE % de 23.03 %, y para el mayor error un 72.81 %.

En el Apéndice B, se adjuntan los resultados al utilizar modelos *Random Forest*, realizando predicciones en semanas diferentes a las presentadas en este Capítulo.

4.4. Pronóstico

Utilizando la base de datos con granularidad de 15 minutos, se entrenan redes LSTM con el fin de poder realizar un pronóstico respecto al comportamiento de una variable de interés. En este caso, se utiliza el *Download User Throughput*, el cual es tratado como serie de tiempo para poder realizar el *training* de las redes.

Con el objetivo de poder predecir los valores que la variable de estudio presentará 3 días al futuro desde un instante t, a partir de la información de toda la semana anterior a este instante, se entrenan la redes LSTM con las 2 primeras semanas de Julio del 2020, y se realizan las pruebas con las semanas restantes de dicho mes.

A continuación, se presentan los resultados del pronóstico de los últimos 2 días del mes de Julio del 2020 para 2 celdas con bajas de desempeño. Primeramente, se presenta en las Figuras 4.15 y 4.16, los resultados de la red LSTM, compuesta por 20 neuronas en su capa oculta cuando es entrenada con 100 épocas, y posteriormente, en las Figuras 4.17 y 4.18, los resultados con un entrenamiento de 1000 épocas para las mismas celdas analizadas:

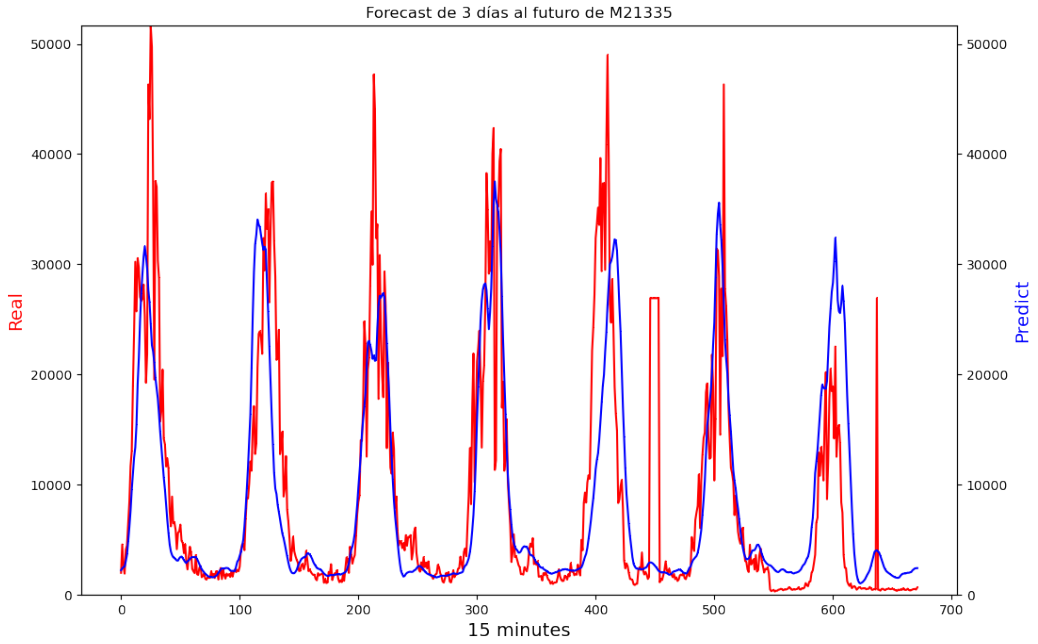


Figura 4.15: Pronóstico red LSTM entrenada 100 épocas para celda M21335

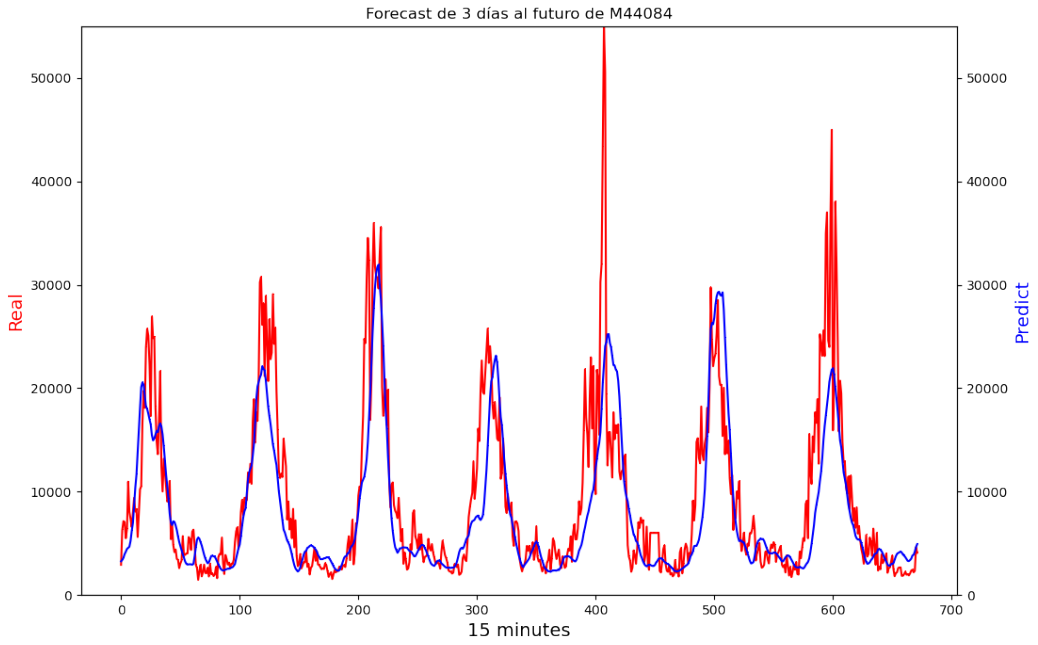


Figura 4.16: Pronóstico red LSTM entrenada 100 épocas para celda M44084

Las siguientes Figuras presentan los resultados de los pronósticos realizados con una red LSTM entrenada con 1000 épocas, como se mencionó anteriormente:

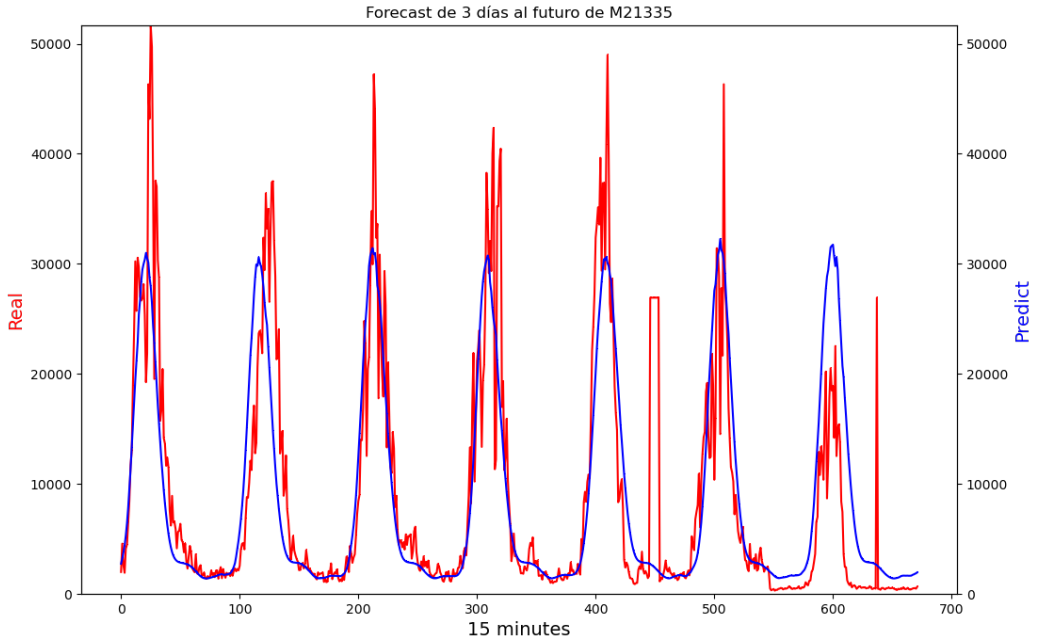


Figura 4.17: Pronóstico red LSTM entrenada 1000 épocas para celda M21335

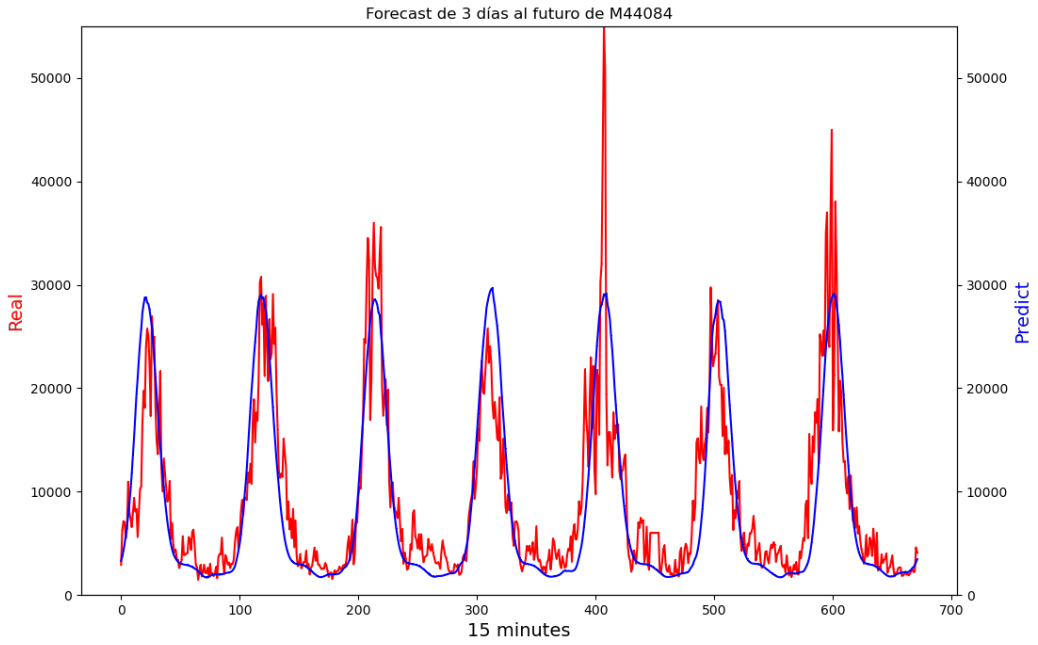


Figura 4.18: Pronóstico red LSTM entrenada 1000 épocas para celda M44084

En la Tabla 4.7 se presenta el error RMSE de los pronósticos obtenidos con las redes LSTM v/s la data real:

		Métricas		
		RMSE	***RMSE %***	***MAPE***
100 épocas	***Celda M21335***	7137.844	80.1 %	98.1 %
	Celda M44084	4952.687	53.8 %	35.3 %
1000 épocas	***Celda M21335***	6409.293	71.9 %	87.9 %
	Celda M44084	4467.647	48.6 %	32.8 %

Tabla 4.7: Métricas obtenidas de pronósticos de redes LSTM

A partir de las Figuras anteriores, en particular observando las Figuras 4.15 y 4.16, las cuales presentan los resultados de la red LSTM entrenada con 100 épocas, se logra evidenciar que los pronósticos resultantes consiguen seguir de forma cercana el comportamiento real de las celdas, existiendo errores a nivel de las magnitudes máximas de los *peaks* alcanzados en la data real. Por otra parte, al observar el desempeño de los pronósticos de la red LSTM entrenada con 1000 épocas, visualizadas en las Figuras 4.17 y 4.18, se aprecia que las curvas pronósticas se suavizan y comienzan a generalizar el comportamiento macro de la celda pero. a pesar de ello, se obtiene un menor error RMSE en comparación a la otra red LSTM, como se puede ver en los valores de la Tabla 4.7. En general, las redes LSTM entrenadas entregan con buena presición una tendencia cercana a los datos reales de los últimos días del mes, pero no se logra capturar completamente el comportamiento de los *peaks* de la data real.

Apreciando con mayor detalle las métricas de desempeño obtenidas, se aprecia que el error MAPE, castiga enormemente los casos en que los datos pronosticados no logran capturar los *peaks* del comportamiento real. Esto se aprecia, al observar que se obtiene un valor de 98.1 % de error, para el uso de la red LSTM entrenada con 100 épocas, testeada en la celda *M21335* y de un 87.9 % al testear esta celda con la red entrenada con 1000 épocas. Para la otra celda, se obtienen valores menores, como por ejemplo 32.8 % para el uso de la red LSTM entrenada con 1000 épocas, lo cual coincide con lo que se ve gráficamente en la Figura 4.18, en donde la predicción sigue con bastante cercanía el comportamiento real, ya que no existen tantos *peaks*.

En el Apéndice C, se presentan otros resultados generados al testear las redes LSTM entrenadas, en otras celdas diferentes a las utilizadas en esta Sección.

4.5. Patrón de Uso Geográfico

Como se mencionó en el Capítulo anterior, para poder observar el patrón de uso que tienen las antenas a nivel geográfico se trabaja con la base de datos de granularidad de 15 minutos y la base de datos diaria, en donde se hace una normalización de los datos a nivel semanal por antena para la primera base de datos, y a nivel mensual para la segunda.

A continuación, se presentan los resultados del análisis con LDA de la base datos de cada 15 minutos, y posteriormente con la base de datos diaria:

4.5.1. Normalización semanal

Al aplicar LDA en el corpus generado por la normalización semanal de las antenas de la base de datos con granularidad de 15 minutos de Julio del 2020 y estableciendo la generación de 10 tópicos diferentes, se pueden identificar similitudes entre los tópicos generados al visualizarlos en el gráfico PCA, lo cual se evidencia en la Figura 4.19:

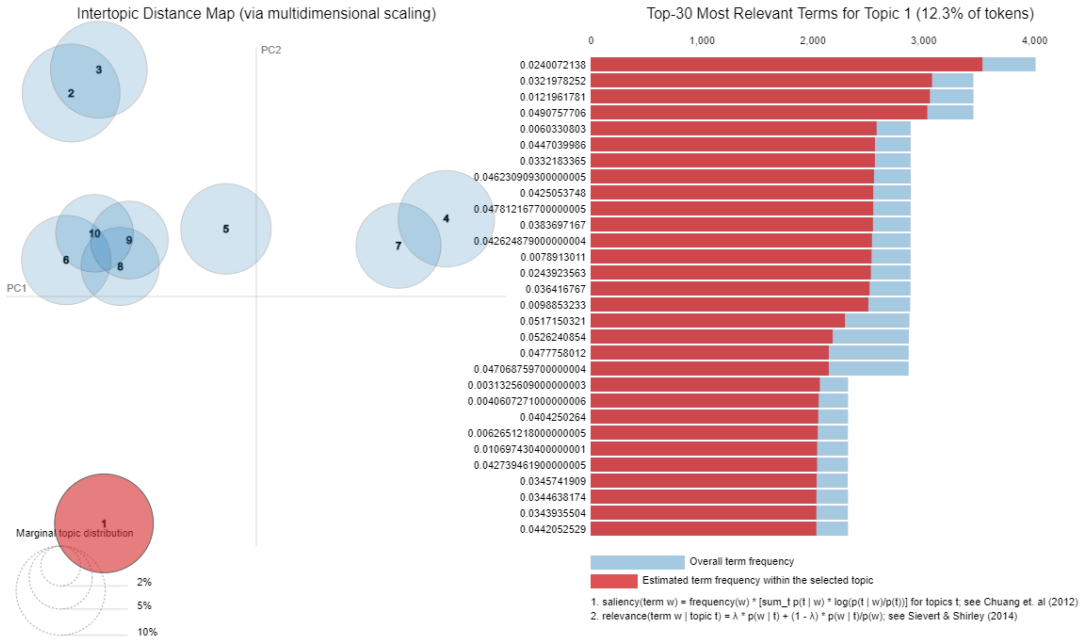


Figura 4.19: Visualización PCA de tópicos generados con normalización semanal

De la Figura anterior se puede observar que se pueden crear *clusters* dentro de los 10 tópicos generados, pudiendo fácilmente agruparlos en 5 conjuntos diferentes. Estos *clusters* se conformarían de la siguiente manera:

Cluster 1: Tópico 1

Cluster 2: Tópico 2 y 3

Cluster 3: Tópico 4 y 7

Cluster 4: Tópico 5

Cluster 5: Tópico 6, 8, 9 y 10

Relacionando los *clusters* de los tópicos generados con la antenas que componen dichas agrupaciones se puede apreciar que, efectivamente, existen diferencias en el comportamiento del volumen del tráfico de datos a nivel semanal. A continuación, en las Figuras 4.20 y 4.21, se presentan los comportamiento semanales de dos antenas diferentes que pertenecen a distintos *clusters*:

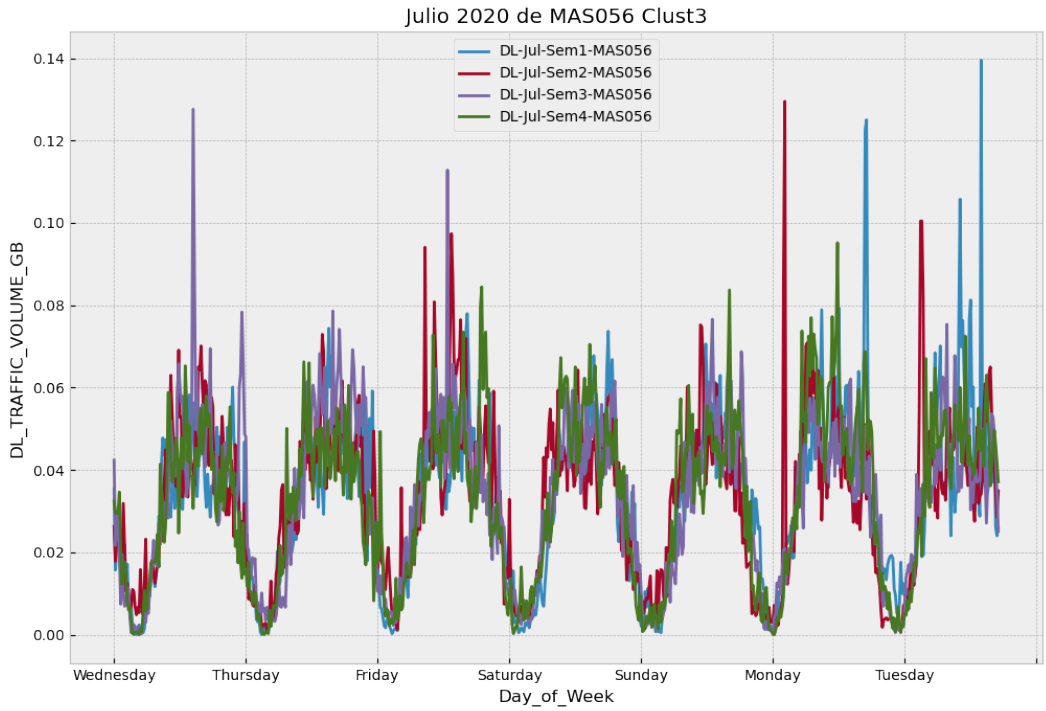


Figura 4.20: Comportamiento semanal de antena de *cluster* 3

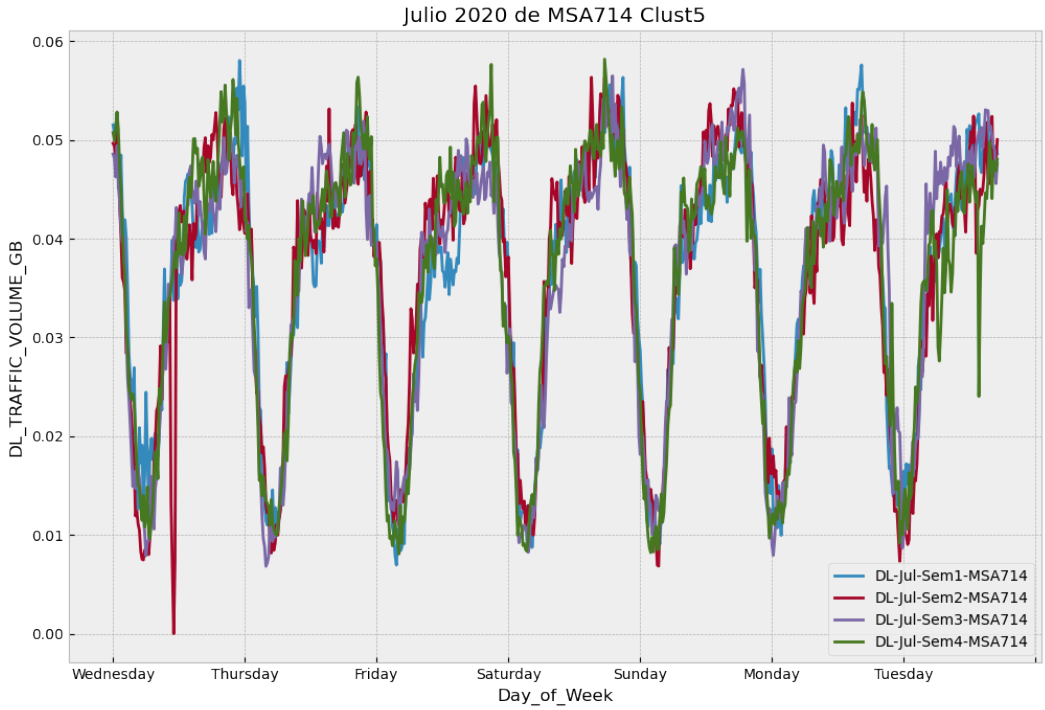


Figura 4.21: Comportamiento semanal de antena de *cluster* 5

De las Figuras anteriores, se puede observar que ambos *cluster* tienen comportamientos notoriamente diferentes. El *cluster* 3, que corresponde a la imagen superior, tiene sus *peak* máximos de uso en horarios de mediodía y el valor normalizado del volumen de tráfico de datos de 0.04, corresponde al valor mínimo que tienen los *peaks* en todos los días de la semana del mes. Por otro lado, el *cluster* 5, que corresponde a la imagen inferior, presenta unos *peaks* de uso desplazados para horarios más nocturnos, y el valor de 0.04 suele ser el inicio de las curvas *peaks* de uso.

Otro análisis que se puede desprender de la Figura 4.20, es que el *cluster* 3 alcanza valores cercanos a 0 en horarios de madrugada, en cambio la Figura 4.21, el *cluster* 5 no presenta momentos de uso nulo.

Ubicando las antenas en un mapa geográfico, diferenciándolas por su pertenencia a cada *cluster*, se genera la Figura 4.22.

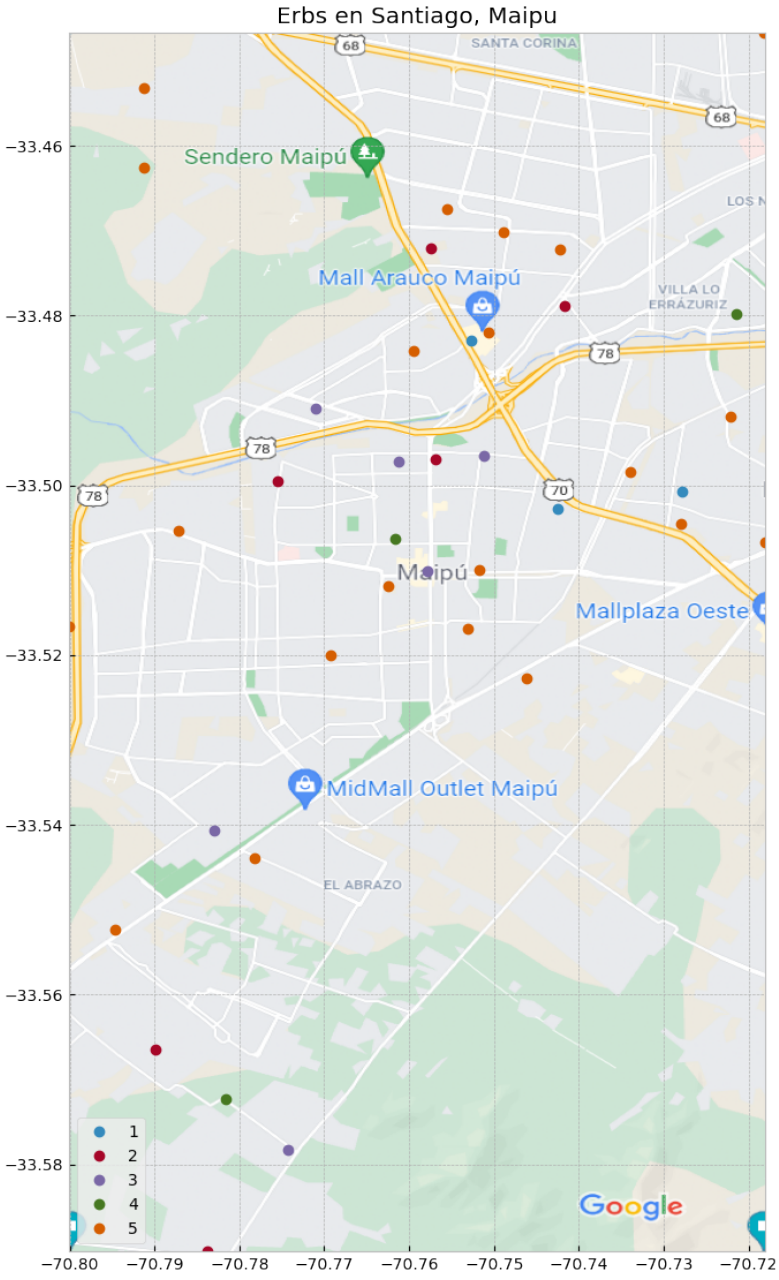


Figura 4.22: Visualización de *cluster* de antenas en mapa geográfico de Maipu

De la Figura 4.22, se puede notar que las antenas del *cluster* 5, en color naranjo, las cuales tenían un comportamiento más residencial como se observó en la Figura 4.21, corresponden efectivamente a zonas más urbanas. En cambio, aquellas antenas correspondientes al *cluster* 3, en color morado en el mapa, corresponden a lugares más públicos y comerciales, en donde existen supermercados, colegios, el metro Plaza de Maipú, tiendas y negocios aledaños.

Como la base de datos con *timestamp* de 15 minutos corresponde solamente a antenas ubicadas en los sectores cercanos a la comuna de Maipú, se procede a realizar un análisis similar al efectuado, pero con la base de datos diaria, de manera de poder observar de forma más macro el patrón de uso de las antenas, lo cual se muestra a continuación:

4.5.2. Normalización mensual

Al aplicar LDA en el corpus generado por la normalización mensual de las antenas, de la base de datos diaria del 2019, y estableciendo la generación de 10 tópicos diferentes, se pueden identificar similitudes entre los tópicos generados al visualizarlos en el gráfico PCA, lo cual se evidencia en la Figura 4.23:

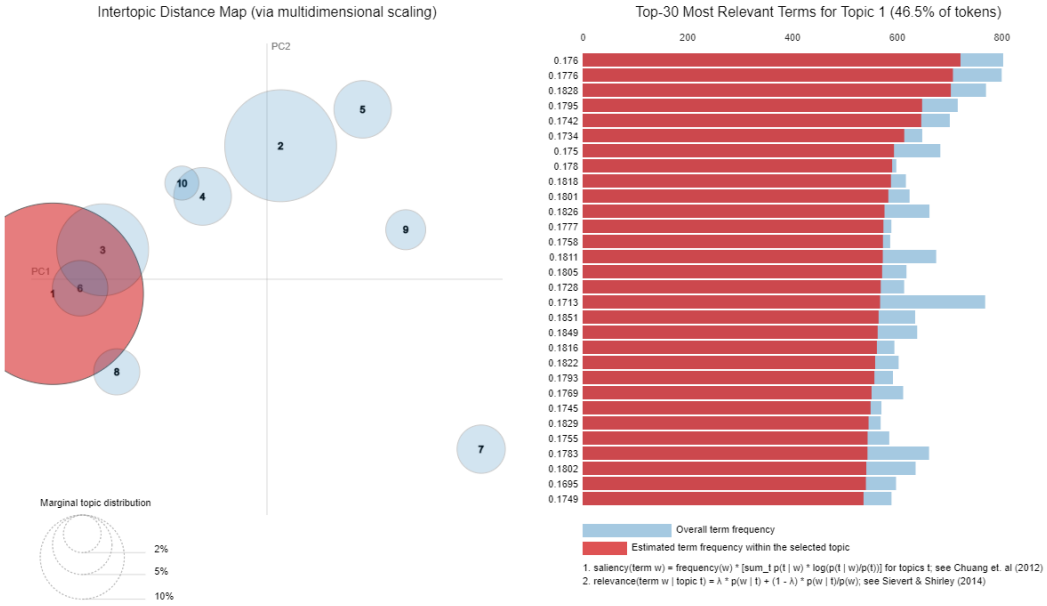


Figura 4.23: Visualización PCA de tópicos generados con normalización mensual

De la Figura anterior se puede observar que se pueden crear *clusters* dentro de los 10 tópicos generados, pudiendo fácilmente agruparlos en 6 conjuntos diferentes. Estos *clusters* se conformarían de la siguiente manera:

Cluster 1: Tópico 1, 3, 6 y 8

Cluster 2: Tópico 2

Cluster 3: Tópico 4 y 10

Cluster 4: Tópico 5

Cluster 5: Tópico 7

Cluster 6: Tópico 9

Al relacionar las antenas con el *cluster* más representativo durante los 12 meses del año, se elimina un *cluster*, y se termina trabajando sólo con 5.

Relacionando los *clusters* de los tópicos generados con la antenas que componen dichas agrupaciones, se puede apreciar que efectivamente existen diferencias en el comportamiento del volumen del tráfico de datos a nivel mensual. A continuación, en las Figuras 4.24 y 4.25, se presentan dos comparativas del comportamiento mensual de cuatro antenas diferentes que pertenecen a distintos *clusters*:

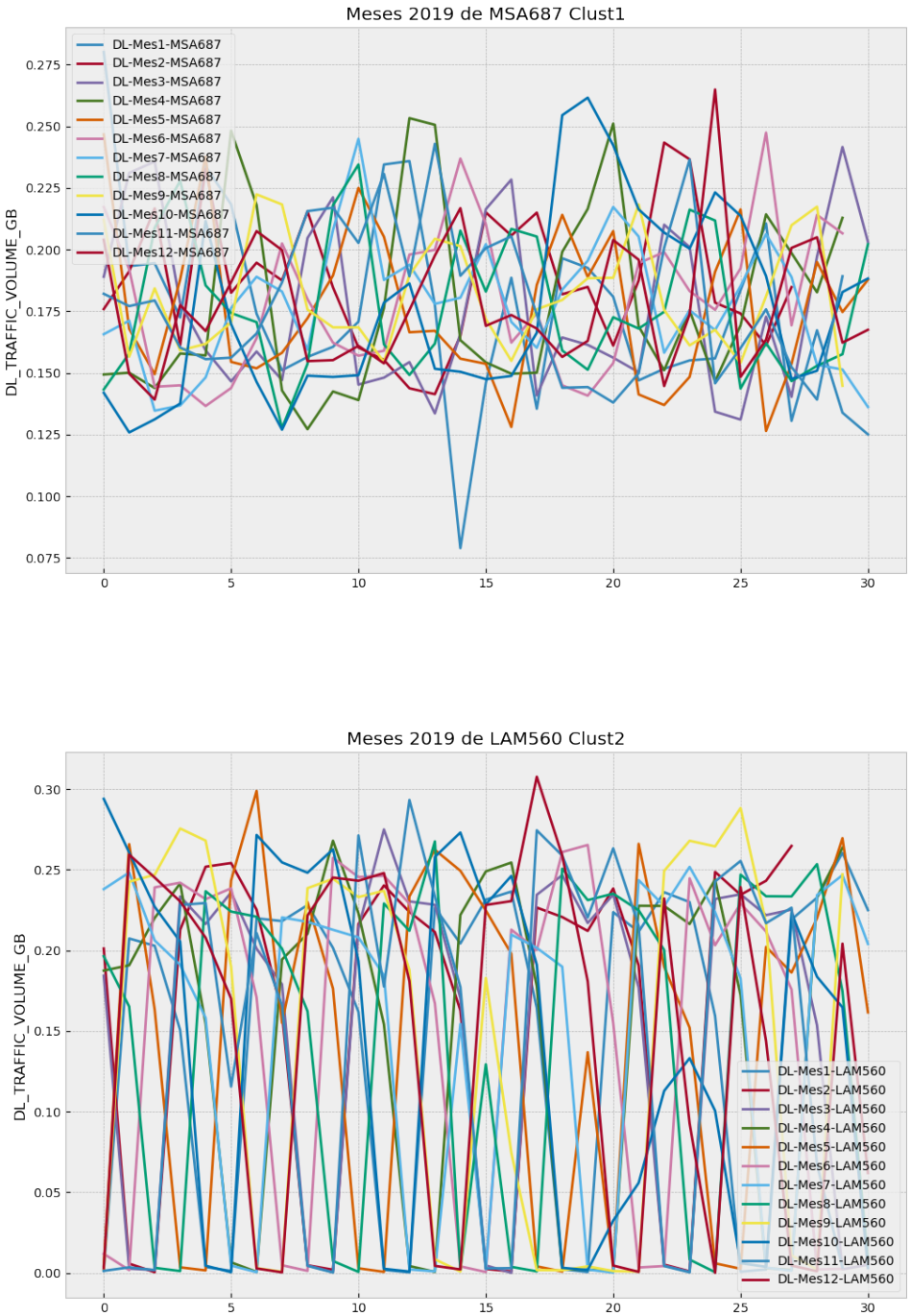


Figura 4.24: Comparativa del comportamiento mensual de antenas del *cluster* 1 y 2

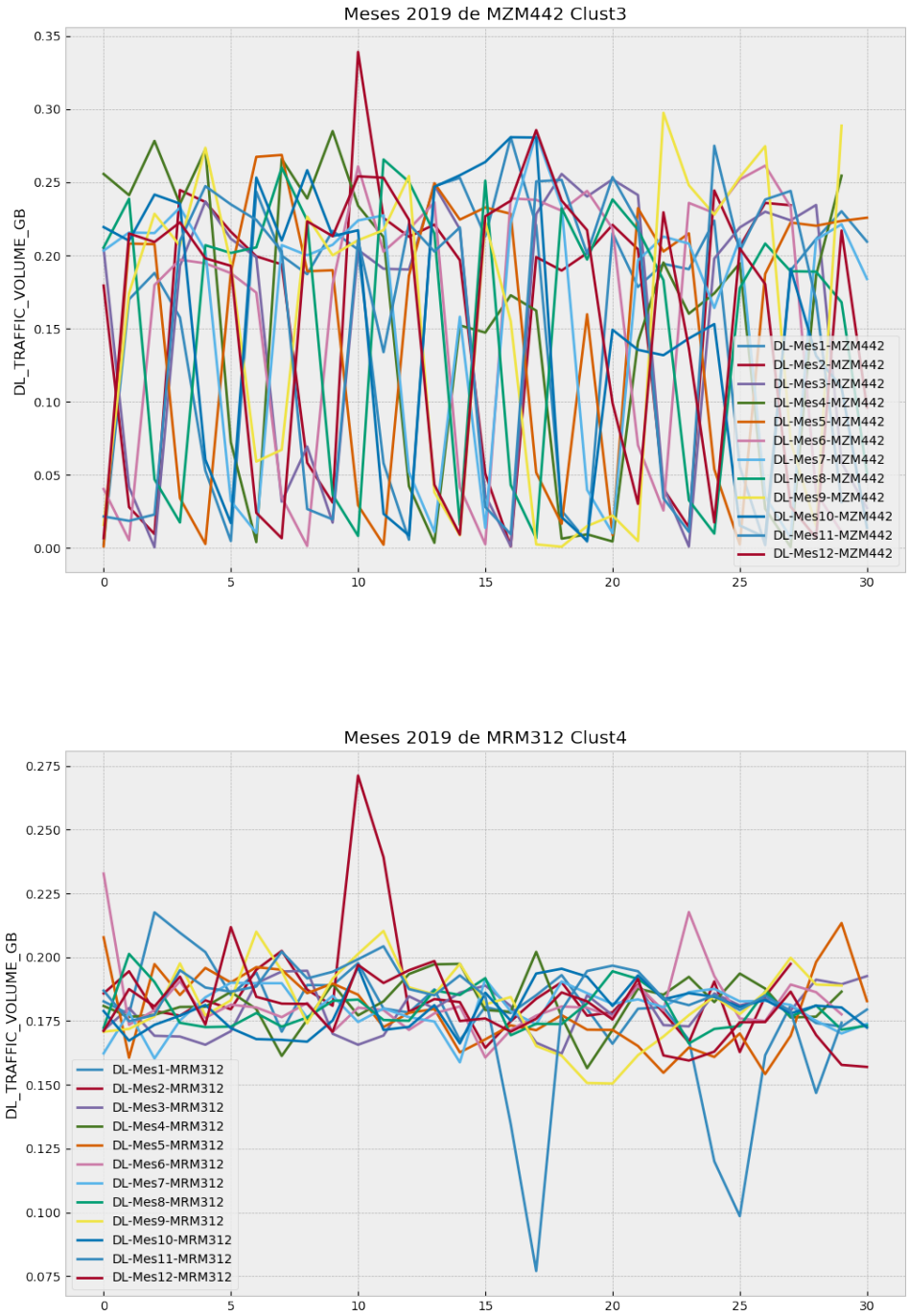


Figura 4.25: Comparativa del comportamiento mensual de antenas del *cluster* 3 y 4

De la Figura 4.24, se observa que el *cluster* 1, imagen superior, tiene operaciones entre los valores 0.15 y 0.225 aproximadamente de una manera desordenada durante todos los meses del año. Por otro lado, en el *cluster* 2, se aprecia una clara tendencia a tener al menos 4 momentos bien marcados todos los meses en donde no se hace mayor uso de la antena. El

cluster 3, presente en la parte superior de la Figura 4.25, se asemeja de gran manera al *cluster* 2, pero tiene la diferencia en que sus periodos de bajo uso son más puntuales y menos prolongados. El *cluster* 4, imagen inferior de la Figura 4.25, se parece bastante al *cluster* 1, pero presenta un uso en rangos sumamente definidos y marcados, entre los valores 0.16 y 0.2 aproximadamente, sin tener *peaks* y comportamientos tan caóticos como el primer *cluster*.

A priori, se puede interpretar que el *cluster* 1 y 4 corresponden a zonas residenciales, en donde se tiene un constante uso durante todos los días de los meses del año. En cambio el *cluster* 2 podría ser de una zona comercial o de oficinas, en donde no se suele tener mayor actividad en los fines de semana. De igual manera el *cluster* 3, tiene comportamientos de una zona más comercial, pero que cuenta con tener actividad los días sábados, teniendo solo el día domingo como instancia de uso nulo.

Ubicando las antenas en un mapa geográfico, diferenciándolas por su pertenencia a cada *cluster*, se genera la Figura 4.26.

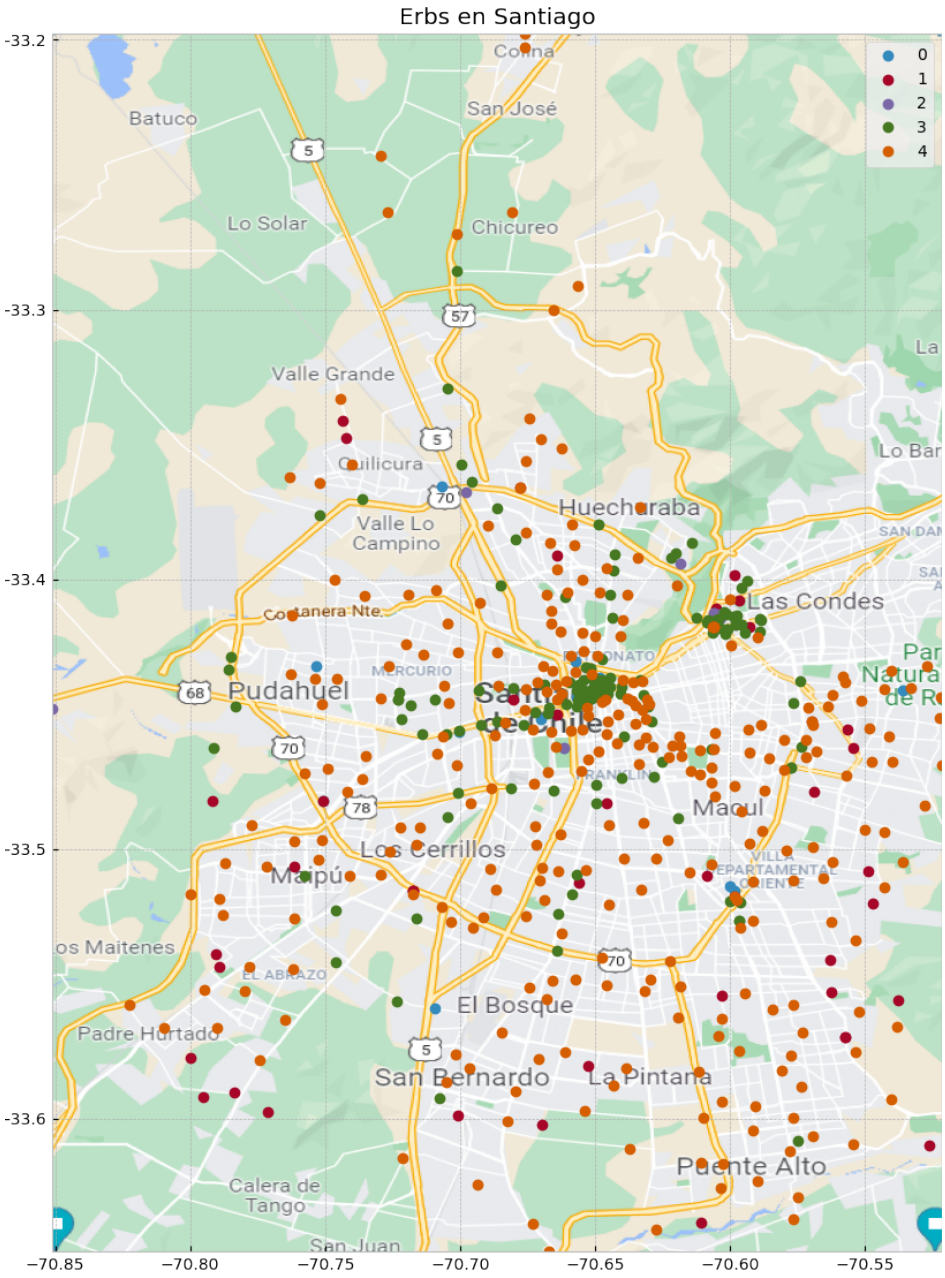


Figura 4.26: Visualización de *cluster* de antenas en mapa geográfico de Santiago

De la Figura 4.26, se evidencia que el *cluster* 4, en color naranjo, tiene mayor frecuencia que los demás *cluster*, ubicándose en diferentes zonas a lo largo de todo Santiago, pero se logra destacar que junto al *cluster* 1, que está representado con el color rojo en el mapa, se suelen ubicar en las zonas más residenciales de ciudad. Por otro lado, se puede apreciar la aglomeración de puntos verdes que corresponden al *cluster* 3, ubicadas principalmente en zonas más publicas y comerciales, como lo es Santiago Centro y zonas de oficinas en Providencia y Las Condes. El *cluster* 2, representado por el color morado, puede ser ubicado en las cercanías de Cuidad Empresarial y el Movistar Arena, coincidiendo con el comportamiento analizado dado por la Figura 4.24.

4.6. Enfoque metodológico propuesto para redes 5G

Debido a la falta de data operacional real de redes 5G, se trabajó con los datos de la operación de una red móvil 4G LTE, ya que que las problemáticas estudiadas en esta memoria son transversales para ambas tecnologías. Las bajas de desempeño en las celdas que componen las redes de telecomunicaciones son situaciones que pueden ocurrir de igual manera en las celdas 5G y la forma de abordar estos casos mediante técnicas de ML es bastante análoga a los expuesto en este trabajo, pero se requiere tener en cuenta ciertos detalles.

Una vez teniendo acceso a un *dataset* real de una red 5G que cuente con diversos KPIs operacionales, se deberá aplicar una metodología similar a la planteada en el Capítulo 3; en donde se apliquen interpolaciones para la data `NaN`, se generen agrupaciones de ciertos *features* categóricos y se eliminen las columnas que no aporten mayor información. Posteriormente, se deberán generar los *labels* correspondientes al desempeño de las celdas respecto a su *Download User Througput*, pero en esta etapa se debe tener en consideración que los umbrales de desempeño ya no serán los mismos, ya que la tecnología 5G propone unos estándares de calidad mayores a los de las redes 4G LTE. De igual forma, varios KPIs de esta nueva base de datos, serán diferentes a los de los *datasets* utilizados en esta memoria, es por ello que al entrenar los modelos de clasificación y aplicar RFE CV se podrán hallar características importantes diferentes a las encontradas en este trabajo, pudiendo evidenciar esto al obtener el porcentaje de *feature importance* de los modelos entrenados.

Teniendo en cuenta lo anterior, la metodología para la realización de clasificación de las celdas respecto a sus desempeños y las predicciones referentes a la cantidad de días con bajo desempeño se mantiene análoga a lo expuesto. De manera similar, para la generación de pronósticos del comportamiento respecto a una variable de interés se debe escoger un *feature* de análisis y se debe transformar el *dataframe* en un formato de series de tiempo para poder entrenar las redes LSTM.

Por último, para la detección del patrón de uso de las antenas de las redes móviles, se mantiene la metodología propuesta, pero estos resultados cobrarían mayor interés para las redes 5G, ya que la información referente al tipo de uso que tienen los usuarios a nivel temporal y geográfico puede ser utilizado en *Network Slicing*, funcionalidad propuesta en 5G que permite la optimización del funcionamiento de la red, realizando una óptima distribución de los recursos móviles de acuerdo a las necesidades reales de los usuarios.

Capítulo 5

Conclusiones y Trabajo futuro

5.1. Conclusiones

En el presente trabajo de título, se realizó un estudio en relación a las redes móviles 4G LTE, las cuales se encuentran ampliamente masificadas alrededor del mundo. En particular, se abordó este estudio con miras a la nueva tecnología emergente 5G, la cual posee desafíos que son transversales para ambas generaciones, como la existencia de celdas que experimentan bajas en sus niveles de desempeño, generando caídas y malas experiencias de servicio en los usuarios.

Las problemáticas abordadas fueron; la identificación de celdas con futuras bajas de desempeño, la predicción de cantidad de días en bajo desempeño de las celdas, la capacidad de generar pronósticos respecto a una variable de interés y la detección del patrón de uso de las antenas por parte de los usuarios, a nivel temporal y geográfico.

Por lo anterior, se hizo un estudio con data de la operación real de las redes móviles 4G LTE de Santiago, utilizando diversas herramientas de *Machine Learning*, como *XGBoost*, redes LSTM y LDA, para poder resolver los desafíos planteados.

Los principales resultados obtenidos fueron; La obtención de una precisión del 83 %, al predecir si una celda experimentará bajas de desempeño; Un valor de error RMSE % del 23 %, al predecir la cantidad de días de bajo rendimento de las celdas de una semana; Pronósticos de una red LSTM que logran capturar la tendencia del comportamiento de un variable de interés, con un error MAPE de 32.8 %; y la identificación de 4 patrones de uso de las antenas de Santiago, directamente relacionados con sus ubicaciones geográficas, asociadas a comportamientos residenciales y comerciales mediante LDA. Por último, se presenta una metodología capaz de replicar la aplicación de las técnicas de aprendizaje automático utilizadas en el presente documento, en un potencial *dataset* de una red 5G.

5.2. Contribución

El trabajo de título realizado contempla las siguientes contribuciones a la industria:

- Se comprobó la utilidad de algoritmos predictivos y de clasificación de ML en data de telecomunicaciones con el objetivo de predecir bajas de desempeño y cantidad de días en dicha situación.
- Se utilizó la red neuronal LSTM como modelo para la generación de pronósticos respecto a alguna variables de interés de estudio como lo fue el *Download User Throughput*.
- Se logró evidenciar diferencias en el comportamiento de las antenas a nivel de geográfico, utilizando data agrupada del volumen de tráfico de las celdas asignadas a cada antena.
- Se entrega una metodología replicable y bien documentada para el trabajo con data de redes de telecomunicaciones, pudiendo ser proyectada para su uso en la tecnología 5G. De esta manera se puede utilizar el patrón de uso de los usuarios, para optimizar los recursos móviles según los requerimientos hallados por zonas, ayudando a permitir la incorporación de *Network Slicing* en las futuras redes 5G.
- La información resultante de los modelos predictivos referentes al desempeño de las celdas, puede ser utilizada para tomar medidas correctivas de manera expedita, permitiendo reducir el tiempo en que los usuarios experimentan bajos niveles de QoS.

5.3. Trabajo Futuro

Como trabajo futuro, se proponen las siguientes propuestas para poder dar continuidad y aplicabilidad a lo ya analizado:

- Probar más configuraciones de hiperparámetros para *XGBoost Regressor* y para las redes LSTM, de manera de obtener resultados más certeros y con menor índice de error.
- Realizar conexión entre los diferentes algoritmos y modelos investigados, de manera de proporcionar resultados que permitan un análisis diverso y exhaustivo de las redes de telecomunicaciones.
- Incorporar algoritmos a los sistema de redes de telecomunicaciones, permitiendo la automatización de pronósticos y predicción de desempeño de celdas.
- Generación de sistema centralizado que permita la obtención online de datos de la red de telecomunicaciones y con ello la posibilidad de desplegar dashboards con analítica predictiva del sistema.

Bibliografía

[1] Statista Research Department, "Distribución de los accesos a redes de internet móviles en América Latina en 2019 y 2025, por generación", Statista, 2020 [Online]. Disponible en: https://es.statista.com/estadisticas/933862/internet-movil-accesos-a-redes-2g-3g-4g-y-5g-en-america-latina/ . [Último Acceso: 29 Julio 2021].

[2] V. Kafle, Y. Fukushima, P. Martinez-Julia, T. Miyazaka, "Consideration On Automation of 5G Network Slicing with Machine Learning", 2018

[3] R. Boutoba, M. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, O. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities", 2018

[4] L. Le, B. Lin, L. Tung, D. Sinh D, "SDN/NFV, Machine Learning, and Big Data Driven Network Slicing for 5G", 2018

[5] R. Gupta, R. Misra, "Machine Learning-based Slice allocation Algorithms in 5G Networks", 2019

[6] NGMN, "5G White Paper", 2020, [Online]. Disponible en: https://www.ngmn.org/wp-content/uploads/NGMN-5G-White-Paper-2.pdf . [Último Acceso: 29 Julio 2021].

[7] S. Bakri, M. Bouaziz, P. Frangoudis, A. Ksentini, "Channel stability prediction to optimize signaling overhead in 5G networks using machine learning", 2020

[8] P. Chiu, J. Reunanen, R. Luostari, H. Holma, "Big Data Analytics for 4.9G and 5G Mobile Network Optimization", 2017

[9] S. Sevgican, M. Turan, K. Gökarslan, H. Yilmaz, T. Tugcu, "Intelligent Network Data Analytics Function in 5G Cellular Networks using Machine Learning", 2019

[10] M. Raza, C. Natalino, L. Wosinska, P. Monti, "Machine Learning Methods for Slice Admission in 5G Networks", 2019

[11] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, X Costa-Perez, "A Machine Learning approach to 5G Infrastructure Market optimization", 2019

[12] Y. Shi, Y. Sagduyu, T. Erpek, "Reinforcement Learning for Dynamic Resource Optimization in 5G Radio Access Network Slicing", 2020

[13] C. Murudkar, R. Gitlin, "Optimal-Capacity, Shortest Path Routing in Self-Organizing 5G Networks using Machine Learning", 2019

[14] Y. Jo, H. Kim J. Lim, D. Hong, "Self-optimization of Coverage and System Throughput in 5G Heterogeneous Ultra-Dense Networks", 2019

[15] S. Ravindran, S. Chaudhuri, J. Bapat, D. Das, "Isolation-based Sub-Slices for Throughput Optimization in 5G Radio Access Network", 2019

[16] C. Suarez-Rodriguez, N. Haider, Y. He, E. Dutkiewicz, "Network Optimisation in 5G Networks: A Radio Environment Map Approach", 2020

[17] R. Dubey, R. Sharma, P. Mishra, S. Pandey, "BAT Optimization based Power Allotment Scheme for 5G Networks", 2020

[18] M. Klinskowsky, D. Mrozinsky, "Latency-aware flow allocation in 5G NGFI networks", 2020

[19] W. Mai, H. Liu, L. Chen, J. Li, H. Xiao, "Multi-Objective Evolutionary Algorithm For 4G Base Station Planning", 2013

[20] T. Omar, Z, Abichar, A. Kamal, J. Morris Chang, M, Alnuem, "Fault-Tolerant Small Cells Locations Planning in 4G/5G Heterogeneous Wireless Networks", 2016

[21] A. Mathew, M. Arthi, K. Vinoth Babu, "A Fuzzy Logic Based Serving Nodes Placement Scheme for Beyond 4G", 2017

[22] S. Zhang, H. Zhao, X. Hou, W. Zhao, "A New Cell Search Scheme Based On Cell-Clustering For UDN", 2015

[23] A. Gaber, M Zaki, A. Mohamed, "4G Uplink Power Control Tuning Approach Using Unsupervised Machine Learning", 2020

[24] S. Ríos, R. Muñoz, "Land Use detection with cell phone data using topic models: Case Santiago, Chile", 2016

[25] R. Muñoz-Cancino,S. Ríos, M. Goic, M. Graña."Non-Intrusive Assessment of COVID-19 Lockdown Follow-Up and Impact Using Credit Card Information: Case Study in Chile", International Journal of Environmental Research and Public Health, 2021

[26] A. Salazar, N. Hureimi, "Diseño de Red 4G/LTE para fines tarifarios", 2016

[27] P. Arzola, P. Valenzuela, "Servicios de Valor Agregado en Tecnología 4G", 2011

[28] F. Firmin, "The Evolved Packet Core", 3GPP MCC, 2021 , [Online]. Disponible en: https://www.3gpp.org/technologies/keywords-acronyms/100-the-evolved-packet-core . [Último Acceso: 18 Noviembre 2021].

[29] Miroslav Kubat, "An Introduction To Machine Learning", 2nd Edition, Springer, 2017, pp 277

[30] D. Ketchen, C. Shook, "The application of cluster analysis in Strategic Management Research: An analysis and critique", 1996

[31] E. Schubert, J. Sander, M. Ester, H. Kriegel, X. Xu, "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN", 2017.

[32] M. Lopes, "Dimensionality Reduction — Does PCA really improve classification outcome?", Towards Data Science, 2017, [Online]. Disponible en: https://towardsdatascience.com/dimensionality-reduction-does-pca-really-improve-classification-outcome-6e9ba21f0 . [Último Acceso: 18 Noviembre 2021].

[33] Z. Jaadi, "A Step-by-Step Explanation of Principal Component Analysis (PCA)", BuiltIn ,2021, [Online]. Disponible en: https://builtin.com/data-science/step-step-explanation-principal-component-analysis . [Último Acceso: 18 Noviembre 2021].

[34] S. Roweis, G. Hinton, " Stochastic neighbor embedding", Neural Information Processing Systems, 2002

[35] Ivo D. Dinov, "Data Sciencie And Predictive Analytics", Springer, 2018, pp 478

[36] Clare Liu, "More Performance Evaluations Metrics for Classification Problems You Should Know", KDNuggets, 2020, [Online]. Disponible en: https://www.kdnuggets.com/2020/04/performance-evaluation-metrics-classification.html . [Último Acceso: 29 Julio 2021].

[37] Ivo D. Dinov, "Data Sciencie And Predictive Analytics", Springer, 2018, pp 291

[38] Ivo D. Dinov, "Data Sciencie And Predictive Analytics", Springer, 2018, pp 403

[39] Jiangchun Li, "Boosting algorithm: GBM", Towards Data Science, 2017, [Online]. Disponible en: https://towardsdatascience.com/boosting-algorithm-gbm-97737c63daa3 . [Último Acceso: 29 Julio 2021].

[40] F. Chollet, "Deep Learning with Python", Manning, 2018, pp 16

[41] E. Freire, S. Silva, "Redes Neuronales", Bootcamp AI Medium, 2019, [Online]. Disponible en: https://bootcampai.medium.com/redes-neuronales-13349dd1a5bb . [Último Acceso: 29 Julio 2021].

[42] I. Goodfellow, Y. Bengio, A. Courville, "Deep Learning", 2015, pp 183

[43] Ivo D. Dinov, "Data Sciencie And Predictive Analytics", Springer, 2018, pp 311

[44] Rohith Gandhi, "Gradient Boosting and XGBoost", Medium, 2018, [Online]. Disponible en: https://medium.com/hackernoon/gradient-boosting-and-xgboost-90862daa6c77 . [Último Acceso: 29 Julio 2021].

[45] Nicolas Vandeput, "Forecast KPIs: RMSE, MAE, MAPE & Bias", Towards Data Science, 2019, [Online]. Disponible en: https://towardsdatascience.com/forecast-kpi-rmse-mae-mape-bias-cdc5703d242d . [Último Acceso: 29 Julio 2021].

[46] N. Vandeput, "Data Science for Supply Chain Forecasting", DE GRUYTER, 2020, pp 11

[47] Pilar González Casimiro, "Análisis de Series Temporales: Modelos ARIMA", Facultad de Ciencias Económicas y empresariales, Universidad del País Vasco, 2009, pp 10

[48] Jason Brownlee, "How to Create an ARIMA Model for Time Series Forecasting in Python", Machine Learning Mastery, 2020, [Online]. Disponible en: https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/ . [Último Acceso: 29 Julio 2021].

[49] S. Hochreiter, J. Schmidhuber, "LONG SHORT-TERM MEMORY", Neural Computation, pp 1735-1780, 1997

[50] I. Goodfellow, Y. Bengio, A. Courville, "Deep Learning", 2015, pp 425

[51] F. Chollet, "Deep Learning with Python", Manning, 2018, pp 202

[52] G. Singhal, "Introduction to LSTM Units in RNN", PluralSight, 2020, [Online]. Disponible en: https://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn . [Último Acceso: 29 Julio 2021].

[53] FH Joanneum, "Cross-Validation Explained", Institute for Genomics and Bioinformatics, 2005

[54] J. Brownlee, "Hyperparameter Optimization With Random Search and Grid Search", MachineLearningMastery ,2020, [Online]. Disponible en: https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/ . [Último Acceso: 22 Noviembre 2021].

[55] D. Blei, A. Ng, M. Jordan, "Latent Dirichlet Allocation", Journal of Machine Learning Research 3, 2003

[56] S. Kapadia, "Topic Modeling in Python: Latent Dirichlet Allocation (LDA)", TowardsDataScience, 2019, [Online]. Disponible en: https://towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0 . [Último Acceso: 29 Julio 2021].

Apéndice A

Clasificaciones

A continuación, se presentan más resultados de los modelos ya entrenados de *XGBoost*, al clasificar y predecir la clasificaciones de otras semanas, en particular se presentará los resultados de las semanas 44 y 48:

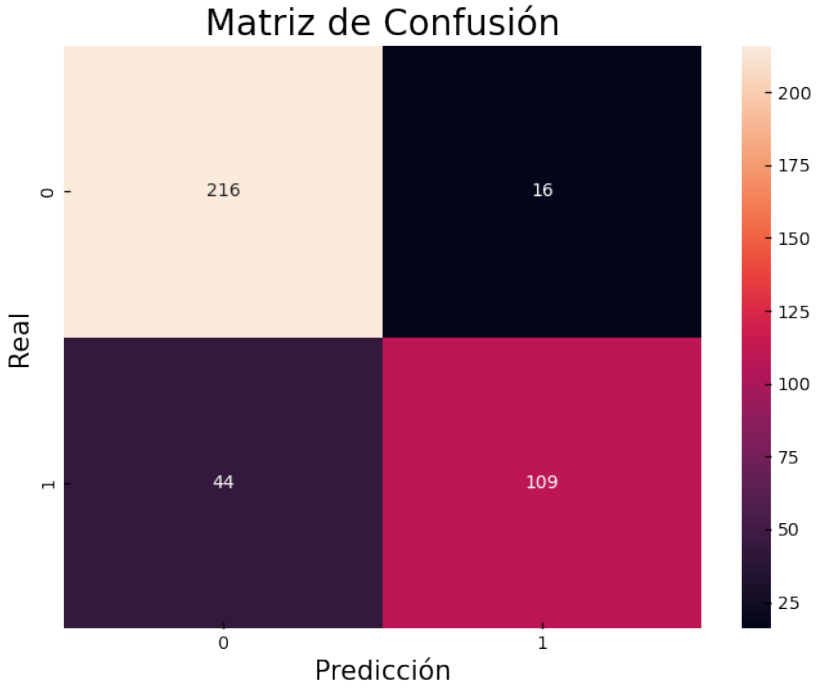


Figura A.1: Clasificación de semana 44

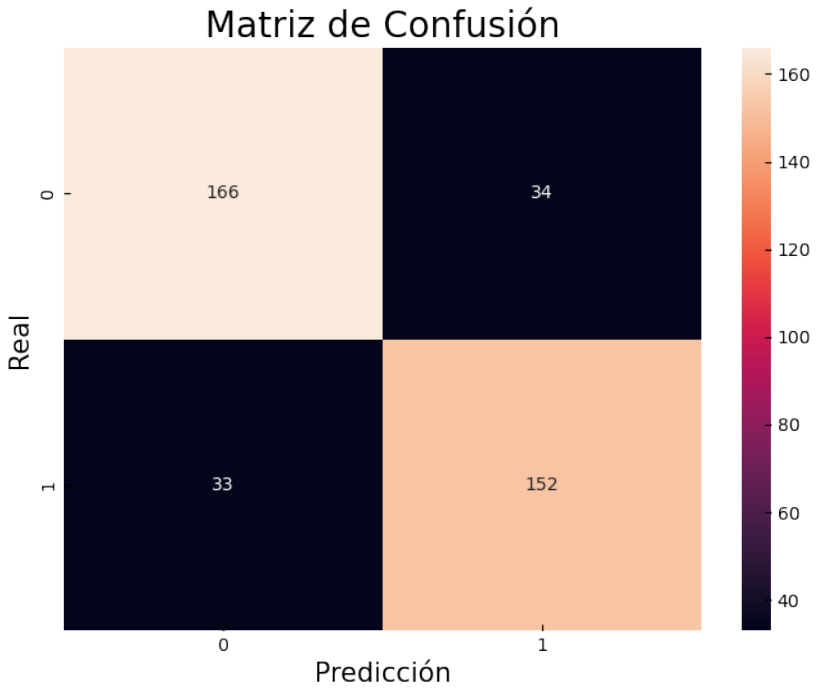


Figura A.2: Clasificación de semana 48

Las siguiente Tablas muestras la principales métricas de las matrices de confusión presentadas anteriormente:

	precision	***recall***	***f1-score***	***support***
0	0.83	0.93	0.88	**232**
1	0.87	0.71	0.78	**153**
accuracy			**0.84**	**385**

Tabla A.1: Métricas de desempeño de clasificación de semana 44

	precision	***recall***	***f1-score***	***support***
0	0.83	0.83	0.83	**200**
1	0.82	0.82	0.82	**185**
accuracy			**0.83**	**385**

Tabla A.2: Métricas de desempeño de clasificación de semana 48

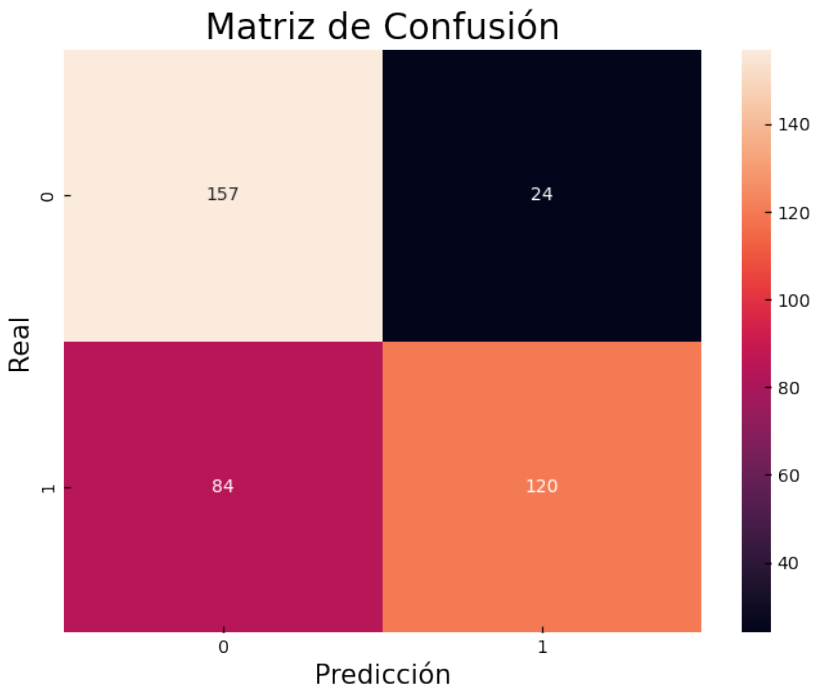


Figura A.3: Predicción de clasificación de semana 45 a partir de datos de semana 44

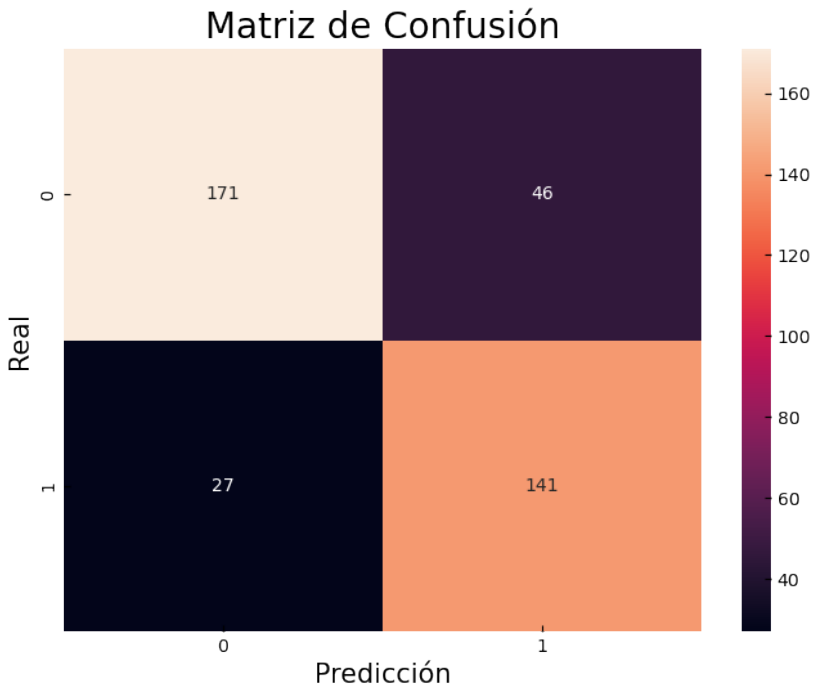


Figura A.4: Predicción de clasificación de semana 49 a partir de datos de semana 48

Las siguiente Tablas muestras la principales métricas de las matrices de confusión presentadas anteriormente:

	precision	***recall***	***f1-score***	***support***
0	0.65	0.87	0.74	**181**
1	0.83	0.59	0.69	**204**
accuracy			**0.72**	**385**

Tabla A.3: Métricas de desempeño de predicción de clasificación de semana 45

	precision	***recall***	***f1-score***	***support***
0	0.86	0.79	0.82	**217**
1	0.75	0.84	0.79	**168**
accuracy			**0.81**	**385**

Tabla A.4: Métricas de desempeño de predicción de clasificación de semana 49

Apéndice B

Predicciones

A continuación, se presentan los resultados de los modelos predictivos con *Random Forest* no presentados en el Capítulo de Resultados. Se muestran las mismas semanas de testeo utilizadas con *XGBoost*, vale decir la semana 41 y la semana 51:

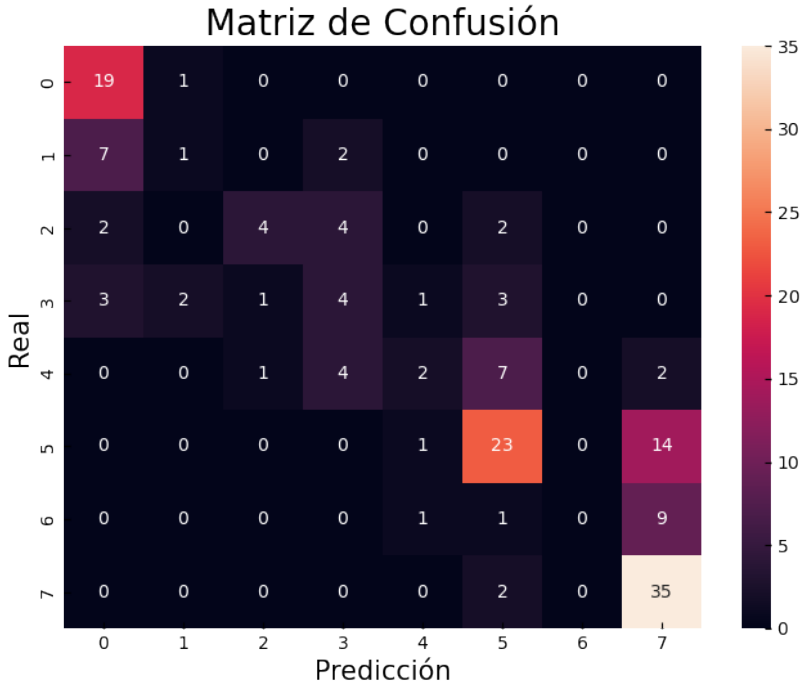


Figura B.1: Predicción de semana 41 con *Random Forest*

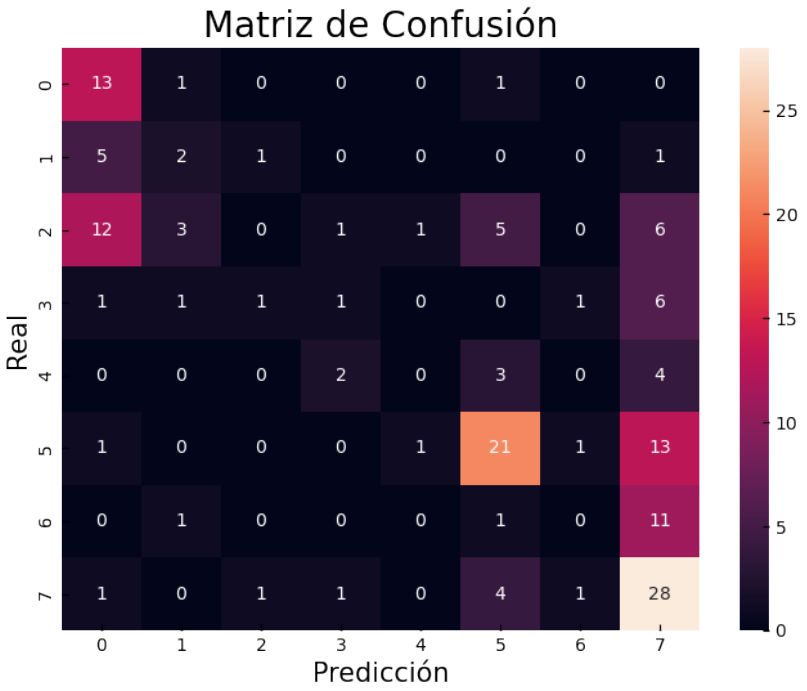


Figura B.2: Predicción de semana 42 a partir de datos de semana 41 con *Random Forest*

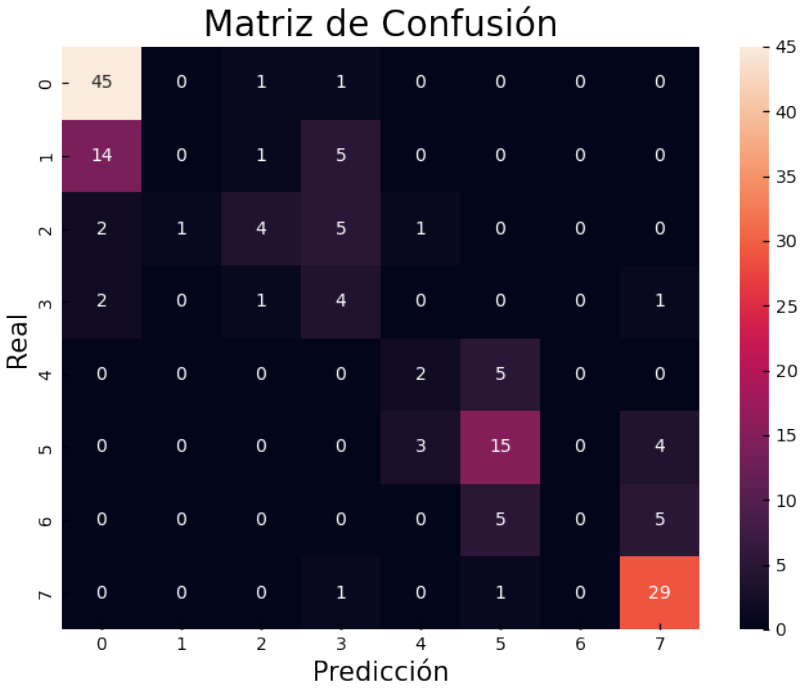


Figura B.3: Predicción de semana 51 con *Random Forest*

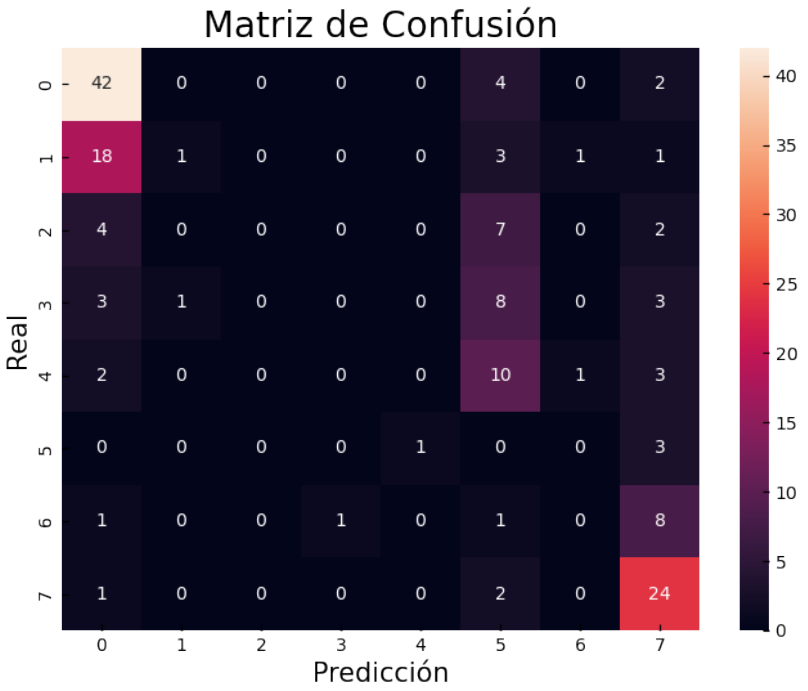


Figura B.4: Predicción de semana 52 a partir de datos de semana 51 con *Random Forest*

		Métricas			
		MAE	***MAE %***	***RMSE***	***RMSE %***
Pred. de la misma semana	***Semana 41***	0.702532	0.4565 %	1.270403	30.6448 %
	Semana 51	0.563291	0.3660 %	1.096600	35.7243 %
Pred. de semana siguiente	***Semana 42***	1.367089	0.8775 %	2.119828	51.6075 %
	Semana 52	1.392405	0.8937 %	2.224717	80.9920 %

Tabla B.1: Métricas de desempeño de predicciones con *Random Forest*

Apéndice C

Pronósticos

A continuación, se presentan los resultados de probar la red LSTM de 100 épocas de *train* en otras 2 celdas:

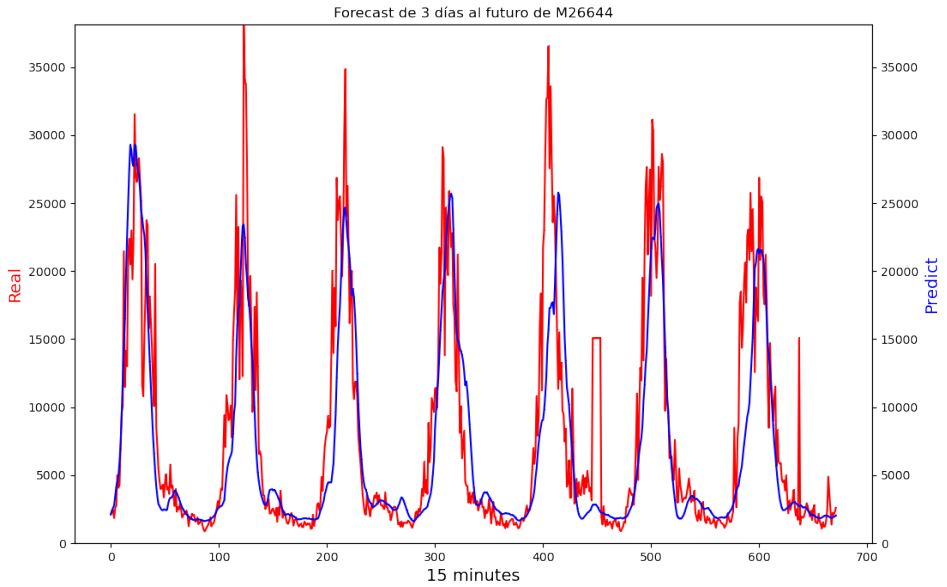


Figura C.1: Pronóstico red LSTM entrenada 100 épocas para celda M26644

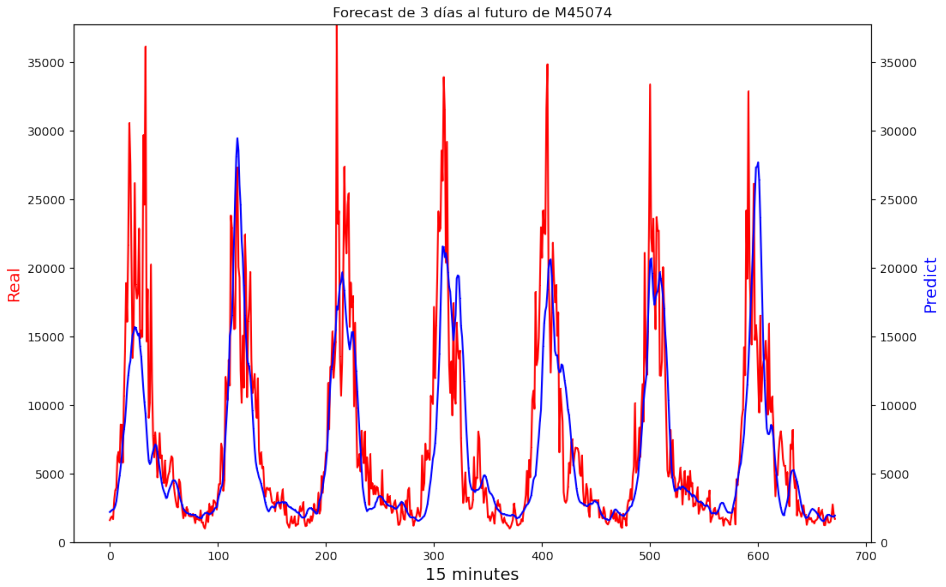


Figura C.2: Pronóstico red LSTM entrenada 100 épocas para celda M45074

Con las mismas celdas anteriores, se prueba la red LSTM de 1000 épocas de entrenamiento:

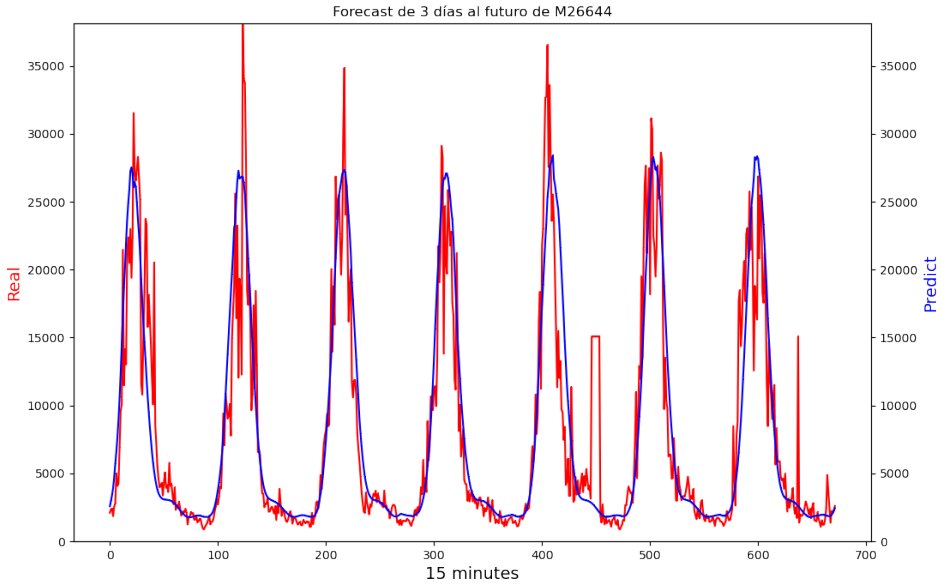


Figura C.3: Pronóstico red LSTM entrenada 100 épocas para celda M26644

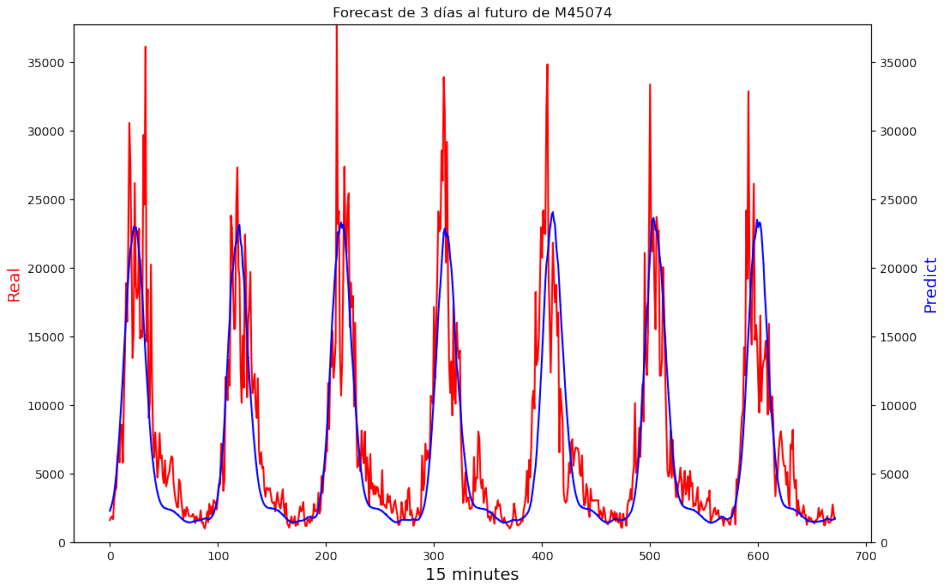


Figura C.4: Pronóstico red LSTM entrenada 100 épocas para celda M45074

		Métricas		
		RMSE	***RMSE %***	***MAPE***
100 épocas	***Celda M26644***	4597.795	58.0 %	36.8 %
	Celda M45074	4273.680	58.4 %	35.7 %
1000 épocas	***Celda M26644***	3798.544	47.9 %	31.0 %
	Celda M45074	3746.401	51.2 %	32.2 %

Tabla C.1: Métricas obtenidas de pronósticos de redes LSTM en otras celdas