



UNIVERSIDAD TECNOLÓGICA METROPOLITANA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INFORMÁTICA Y COMPUTACIÓN
ESCUELA DE INFORMÁTICA

MODELO DE APRENDIZAJE AUTOMÁTICO PARA LA PREDICCIÓN DEL COMPORTAMIENTO DE CLIENTES EN CANAL WEB

TRABAJO DE TÍTULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN MENCIÓN
INFORMÁTICA

AUTORES:

GONZÁLEZ GÁRATE, CRISTÓBAL ANDRES
TAPIA RIQUELME, MARCELO IGNACIO
OYARCE TREJO, DIEGO ESTEBAN

PROFESORA GUÍA:

CASTRO OPAZO, PAULA

SANTIAGO - CHILE

2023

Índice general

Resumen	vii
Abstract	ix
1 Presentación del proyecto	1
1.1 Descripción del trabajo de título	1
1.2 Objetivos	1
1.3 Alcances y Limitaciones	2
2 La empresa	4
2.1 Historia	4
2.2 Descripción general	6
2.3 Misión y visión	6
2.3.1 Misión	6
2.3.2 Visión	7
3 Marco teórico	8
3.1 Importancia de predecir el comportamiento del cliente en un sitio web	8
3.2 Comportamiento del cliente/afiliado en el canal web	9
3.2.1 Definición y relevancia del comportamiento del cliente para el negocio	9
3.2.2 Características del comportamiento del cliente en el canal web	11
3.2.3 Factores que afectan el comportamiento del cliente	12
3.3 Herramientas para la predicción del comportamiento del cliente en el canal web	13
3.3.1 Introducción a las herramientas de análisis de datos	13
3.3.2 Métodos, técnicas y tecnologías de análisis de datos	15

3.3.3	Modelos de predicción de comportamiento del cliente	16
3.3.4	Tabla comparativa de los modelos de predicción	36
3.3.5	Series Temporales	37
3.3.6	Redes LSTM Long Short-Term Memory	58
3.3.7	Metodología del proyecto	60
3.3.8	Metodología del sistema	62
4	Proceso ETL	65
4.1	Diseño Proceso ETL	65
4.1.1	Requisitos ETL	66
4.1.2	Identificación fuente de datos	66
4.1.3	Diseño del modelo de datos objetivo	67
4.1.4	Planificación de las transformaciones	68
4.1.5	Selección de herramientas	70
4.1.6	Construcción y prueba proceso ETL	71
4.1.7	Monitoreo proceso ETL	75
5	Exploratory Data Analysis (EDA)	77
5.1	Introducción al EDA	77
5.2	Recopilación de datos	78
5.3	Descripción de los datos	80
6	Modelos de predicción aplicados	84
6.1	Modelo de series de tiempo	84
6.1.1	Preprocesamiento y Preparación de Datos	84
6.1.2	Codificación y Transformación One-Hot	84
6.1.3	Construcción y Entrenamiento del Modelo	85
6.1.4	Conclusión del Modelo de series de tiempo ARIMA	85
6.2	Modelo de autoencoders	86
6.2.1	Preprocesamiento y Preparación de Datos	86
6.2.2	Codificación One-Hot	87
6.2.3	Construcción y Entrenamiento del Modelo	88
6.2.4	Resultados y Evaluación	92
6.2.5	Conclusión del Modelo de Autoencoder	93
6.3	Modelo de predicción secuencial	94

6.3.1	Preprocesamiento y Preparación de Datos	94
6.3.2	Ordenamiento y Etiquetado	95
6.3.3	Identificación de Sesiones	96
6.3.4	Codificación de Categorías	99
6.3.5	Construcción de Secuencias	102
6.3.6	División de Datos	105
6.3.7	Construcción del Modelo	107
6.3.8	Regularización y Compilación	110
6.3.9	Entrenamiento del Modelo	113
6.3.10	Resultados del entrenamiento del modelo	115
6.3.11	Predicción del modelo	117
6.3.12	Métricas de predicción aplicadas	118
6.3.13	Conclusión del Modelo de Predicción Secuencial	119
7	Api y modelo implementado	121
7.1	Descripción de los componentes	121
7.2	Desarrollo e Implementación de la API	122
7.3	Dockerización de la Api	125
8	Conclusión y Recomendación	129
8.0.1	Conclusión	129
8.0.2	Recomendaciones	130
	Referencias	133
	Anexos	137
	Anexo 1. Listado de métodos en la plataforma	137

Índice de figuras

1	Historia AFP Capital	5
2	Estructura de un árbol de decisión	28
3	Estructura de un random forest	32
4	Tabla comparativa modelos de predicción	37
5	Cantidad semanal de pasajeros que volaron en la clase económica de Ansett Airlines	39
6	Ventas mensuales de medicamentos antidiabéticos en Australia	41
7	Demanda electrica debido a las temperaturas en 2 ciudades en Australia	42
8	Producción trimestral de cerveza australiana	43
9	Coeficientes de autocorrelación de la producción de cerveza en Australia	45
10	ACF de la produccion trimestral de cerveza	45
11	Demanda mensual de electricidad de Australia	46
12	ACF demanda mensual de electricidad de Australia	46
13	Ruido blanco	47
14	ACF Ruido blanco	48
15	Aplicación de logartimos y diferenciación estacional a ventas de medicamentos antidiabéticos	51
16	Modelos autorregresivos con diferentes parámetros	53
17	Modelos de media móvil con diferentes parámetros	54
18	Valores nulos en el conjunto de datos.	72
19	Descripción del dataset antes del proceso ETL.	74
20	Descripción del dataset después del proceso ETL.	74
21	Distribución de métodos en el dataset.	82
22	Preparación y codificación one-hot de los datos	88
23	Arquitectura del modelo autoencoder	90
24	Entrenamiento del modelo autoencoder	92
25	Gráfico del entrenamiento del modelo autoencoder	93
26	Código indentificación de sesiones	98
27	Código codificación de categorías	101
28	Código construcción de secuencias	104

29	Código dividir el conjunto	107
30	Construccion del modelo de clasificación	110
31	Construccion del compilador del modelo	112
32	Código para entrenar el modelo	115
33	Gráficos de análisis del modelo de clasificación	117
34	Código creación API	124
35	Prueba de funcionamiento API de predicción en Postman	125
36	Código archivo Dockerfile	127
37	Código archivo docker-compose.yml	128

Índice de tablas

Tabla de métodos del canal.	140
-------------------------------------	-----

RESUMEN

El presente documento de Trabajo de Titulación tiene como objetivo mostrar la forma y el plan de trabajo que se utilizan a lo largo del proceso de desarrollo del proyecto propuesto. Además, se presentan los resultados del proceso investigativo que se ha realizado hasta la fecha, incluyendo estudios sobre el comportamiento de los clientes en canales web, modelos y algoritmos de predicción, y cómo desarrollar un proceso ETL y un Análisis Exploratorio de Datos (EDA).

El objetivo principal de este proyecto es analizar el comportamiento de los clientes de AFP Capital y sus preferencias de uso en un período de hasta 1 meses, con el fin de predecir futuras navegaciones personalizadas.

El proyecto consta de cuatro fases para su desarrollo. La primera fase abarca la planificación y el planteamiento de los antecedentes generales para la realización del proyecto. La segunda fase se centra en la investigación de la problemática en estudio, basándose en la situación actual planteada. La tercera fase abarca el modelamiento y desarrollo del proyecto, incluyendo el modelamiento de datos y el enfoque del proceso ETL y EDA. Esta fase también involucra el desarrollo del código que respaldará y ejecutará el modelo predictivo, mediante la construcción de bases de datos, APIs y la realización de pruebas para mitigar posibles errores encontrados. Por su parte el EDA contempla un análisis profundo de los datos entregados por la empresa. La cuarta y última fase concluye el desarrollo del proyecto y se enfoca en las conclusiones y recomendaciones, donde se presentarán las conclusiones obtenidas a lo largo del proceso y se elaborará un manual de usuario con las recomendaciones de uso.

Además, este proyecto se lleva a cabo bajo un marco de trabajo de desarrollo ágil, utilizando Scrum, y se están usando metodologías de análisis y minería de datos, como CRISP-DM y OSEMN. El entorno de desarrollo se basa en Python, junto con bibliotecas de análisis y minería de datos como Pandas y Numpy, y frameworks de desarrollo de APIs como Flask, Django y FastAPI.

Palabras clave: Afiliado, Administradora de Fondos de Pensiones, API (Application Programming Interfaces), EDA (Exploratory Data Analysis), Algoritmos de predicción, Algoritmos de clasificación, Modelos de predicción, ETL (Extract, Transform and Load), ARIMA (Modelo de Autorregresión integrada de media móvil), SARIMA (Modelo Estacional de Autorregresión integrada de media móvil), Redes Neuronales Artificiales (ANN), Redes LSTM (Long Short-Term Memory), Redes Neuronales Recurrentes (RNN).

ABSTRACT

The purpose of this Degree Project document is to show the form and work plan used throughout the development process of the proposed project. In addition, the results of the research process that has been carried out to date are presented, including studies on customer behavior in web channels, predictive models and algorithms, and how to develop an ETL process and an Exploratory Data Analysis (EDA).

The main objective of this project is to analyze AFP Capital customers' behavior and usage preferences over a period of up to 1 month, in order to predict future personalized browsing.

The project consists of four phases for its development. The first phase covers the planning and general background for the implementation of the project. The second phase focuses on the investigation of the problem under study, based on the current situation. The third phase covers the modeling and development of the project, including data modeling and the ETL and EDA process approach. This phase also involves the development of the code that will support and execute the predictive model, through the construction of databases, APIs and testing to mitigate possible errors found. The EDA includes an in-depth analysis of the data provided by the company. The fourth and last phase concludes the development of the project and focuses on conclusions and recommendations, where the conclusions obtained throughout the process will be presented and a user manual with recommendations for use will be prepared.

In addition, this project is carried out under an agile development framework, using Scrum, and data mining and analysis methodologies, such as CRISP-DM and OSEMN, are being used. The development environment is based on Python, together with data mining and analysis libraries such as Pandas and Numpy, and API development frameworks such as Flask, Django and FastAPI.

Keywords: Affiliate, Pension Fund Administrator, API (Application Programming Interfaces), EDA (Exploratory Data Analysis), Predictive Algorithms, Classification Algorithms, Predictive Models, ETL (Extract, Transform and Load), ARIMA (AutoRegressive Integrated Moving Average), SARIMA (Seasonal AutoRegressive Integrated Moving Average), Artificial Neural Network (ANN), LSTM Networks (Long Short-Term Memory), Recurrent Neural Network (RNN).

CAPÍTULO 1: PRESENTACIÓN DEL PROYECTO

1.1. Descripción del trabajo de título

El trabajo de titulación se basa en un proyecto empresarial que tiene como objetivo procesar los registros de navegación del sitio web para afiliados de AFP Capital. El propósito principal es detectar comportamientos de los clientes y sus preferencias en un determinado período de tiempo, con el fin de personalizar las futuras experiencias de navegación. La lectura de los registros se realizará extrayendo la información desde Kibana, una plataforma basada en Elasticsearch, que registra la información a través de diversas APIs utilizadas en el sitio web.

Los elementos fundamentales del proyecto incluyen el análisis exploratorio de datos, extracciones, transformaciones, cargas, modelos de predicción y detección de preferencias. El objetivo final es desarrollar un modelo capaz de predecir el comportamiento de los clientes en el canal web.

1.2. Objetivos

Objetivo general

Analizar el comportamiento de los clientes y sus preferencias de uso en un período de hasta 1 mes, con el fin de predecir navegaciones futuras personalizadas.

Objetivos específicos

- Realizar una investigación sobre las herramientas utilizadas para la predicción del comportamiento de usuarios en un canal web.
- Realizar un análisis y estudio de los datos proporcionados por la empresa.

- Realizar un proceso de Extracción, Transformación y Carga (ETL) con la información de navegación web de los clientes de AFP Capital, con el objetivo de analizar su comportamiento dentro del sitio web privado.
- Desarrollar un modelo capaz de predecir el comportamiento de los clientes de AFP Capital, con el propósito de ofrecer navegaciones personalizadas en el futuro.
- Establecer recomendaciones de personalización basadas en los hallazgos del modelo de predicción, para las futuras navegaciones dentro del sitio web de AFP Capital.

1.3. Alcances y Limitaciones

Alcances

El proyecto contempla los siguientes alcances:

- Se analizará el comportamiento de los clientes de AFP Capital en su nuevo sitio web privado.
- El proyecto entregará un modelo capaz de predecir el comportamiento de los clientes de AFP Capital en el sitio web, así como una API que permita obtener recomendaciones de comportamiento personalizadas para un afiliado específico, dependiendo de las secuencias de su comportamiento.

Limitaciones

El proyecto tiene las siguientes limitaciones:

- No se contará con acceso directo a las bases de datos de AFP Capital, por lo tanto, se trabajará con una muestra de datos.
- No se podrá acceder a información sensible de los clientes de AFP Capital, como los RUTs (Rol Único Tributario) u otra información personal identificable.

- El análisis se basará únicamente en datos cualitativos de la navegación web de los usuarios.
- La disponibilidad de datos se limita a un periodo de 15 días debido a las dificultades asociadas con su extracción.

CAPÍTULO 2: LA EMPRESA

AFP Capital se erige como un actor destacado en su industria, con una trayectoria consolidada y un enfoque constante en la innovación y la excelencia operativa. Este capítulo del informe proporcionará una visión detallada de su historia, una descripción general, misión, visión y su papel en el mercado actual. A lo largo de estas páginas, exploraremos los aspectos clave que definen a esta empresa y su relevancia en el panorama empresarial actual.

2.1. Historia

La historia de AFP Capital se remonta a noviembre de 1980, cuando se implementó en Chile el sistema de pensiones de capitalización individual. El 16 de enero de 1981, se constituyó la sociedad Administradora de Fondos de Pensiones Santa María, la cual más tarde se transformaría en AFP Capital S.A. Desde sus inicios, la empresa se destacó por su filosofía de servicio, enfocada en satisfacer las necesidades y expectativas de sus afiliados.

En 1995, AFP Capital estableció la filial Santa María Internacional S.A., con el propósito de expandir su alcance y ofrecer servicios a personas naturales o jurídicas del extranjero, así como invertir en AFP o sociedades relacionadas con materias previsionales en otros países. Esta iniciativa consolidó la presencia de AFP Capital en el ámbito internacional y fortaleció su posición como una administradora de fondos de pensiones líder en la región.

En el año 2000, se produjo una transacción relevante en la historia de AFP Capital. ING Group adquirió Aetna Inc., incluyendo el 96,56 % de las acciones de AFP Capital S.A. Esta adquisición tuvo como objetivo reforzar la posición de liderazgo de AFP Capital en el mercado previsional chileno y contribuir a su crecimiento y desarrollo.

Posteriormente, en 2008, AFP Capital llevó a cabo una fusión con AFP Bansander, otra reconocida administradora de fondos de pensiones en Chile. Esta fusión permitió consolidar aún más las operaciones de AFP Capital y fortalecer su presencia en el país. A fines de 2011, Grupo SURA, una empresa líder en el negocio de pensiones en Latinoamérica, adquirió las operaciones de ING en la región. Esta adquisición llevó a AFP Capital a formar parte de Grupo SURA y a beneficiarse de su amplia experiencia y recursos, consolidándose como una compañía destacada en el mercado previsional latinoamericano.

En resumen, la historia de AFP Capital está marcada por su constante evolución, consolidación y liderazgo en el mercado de administración de fondos de pensiones en Chile. A lo largo de los años, ha demostrado su compromiso con la excelencia en la prestación de servicios previsionales y su capacidad de adaptación a los cambios y desafíos del entorno económico y regulatorio.

Figura 1: Historia AFP Capital



Fuente: AFP Capital. Recuperado de <https://www.afpcapital.cl/Quienes-Somos/Paginas/Historia.aspx>

2.2. Descripción general

AFP Capital es una destacada compañía chilena dedicada al negocio de pensiones y administración de fondos de pensiones. Forma parte de SURA, una reconocida empresa que ofrece servicios financieros y previsionales en Chile y otros países de América Latina. El enfoque principal de AFP Capital es proporcionar a sus afiliados asesoría personalizada y servicios diferenciados que les permitan alcanzar una mejor pensión al momento de su jubilación. La empresa se distingue por su compromiso con la optimización de la calidad de sus servicios, la entrega de información transparente y relevante a sus afiliados, y su solidez empresarial.

Con más de tres décadas de experiencia en el mercado, AFP Capital se ha posicionado como una de las principales administradoras de fondos de pensiones en Chile. Esto se debe en gran medida a su administración seria, responsable y eficiente en el manejo de los Fondos de Pensiones, así como a su enfoque prudente y estratégico en la inversión y gestión de los recursos.

La compañía cuenta con un equipo de colaboradores altamente capacitados y comprometidos, quienes contribuyen a la excelencia en la atención al cliente y al logro de los objetivos financieros de los afiliados. Además, AFP Capital se destaca por su constante innovación y adaptación a los cambios regulatorios y a las necesidades cambiantes de los afiliados, con el fin de ofrecer soluciones efectivas y satisfactorias en el ámbito de las pensiones.

2.3. Misión y visión

2.3.1. Misión

La misión de AFP Capital es: "Acompañamos a nuestros clientes, a través de una asesoría experta y diferenciadora en soluciones de ahorro para alcanzar su número, su Pensión, creciendo sustentablemente, desarrollando a nuestros colaboradores e integrándose responsablemente a la comunidad." (*AFP Capital*, 2023)

2.3.2. Visión

La visión de AFP Capital es: "Somos Guías, acompañamos a nuestros clientes a lograr sus sueños a través del ahorro." (*AFP Capital*, 2023)

CAPÍTULO 3: MARCO TEÓRICO

En el ámbito empresarial actual, la capacidad de predecir y comprender el comportamiento del cliente en un sitio web es esencial para el éxito. Este capítulo del informe se adentrará en la importancia estratégica de estas predicciones, analizará el comportamiento de los clientes y afiliados en el canal web y presentará las herramientas clave utilizadas para anticipar y optimizar la experiencia del usuario en línea.

3.1. Importancia de predecir el comportamiento del cliente en un sitio web

La predicción del comportamiento del cliente dentro de un entorno web implica la aplicación de técnicas y modelos analíticos para anticipar, en cierta medida, las posibles necesidades, acciones, preferencias y decisiones que un cliente pueda tomar mientras interactúa en una plataforma en línea o sitio web. En los últimos años, la predicción del comportamiento de los clientes ha sido de gran importancia para las empresas, ya que les permite anticiparse a las necesidades y preferencias de sus clientes, adaptando así sus productos y servicios para brindar una mayor satisfacción al cliente (Zheng, Thompson, Lam, Yoon, y Gnanasambandam, 2013).

La lealtad de los clientes es un valor clave para las empresas, ya que un cliente leal seguirá consumiendo los productos y servicios de la empresa. Por lo tanto, mejorar la experiencia del usuario aumenta la satisfacción del cliente, lo que a su vez genera un incremento en las ganancias de la empresa.

Según (Zheng y cols., 2013), la predicción del comportamiento del cliente ayuda a las empresas a identificar oportunidades de mejora y de mercado, además de respaldar la toma de decisiones informadas sobre estrategias de publicidad y marketing. El objetivo principal de predecir el comportamiento del cliente en un entorno web es comprender y anticipar las acciones de los clientes con el fin de personalizar y mejorar

la experiencia del usuario, y así aumentar la satisfacción y fidelidad de los clientes.

Las predicciones pueden abarcar diferentes aspectos del comportamiento de un cliente dentro de un canal web. En términos generales, existen cuatro tipos de predicciones que se pueden realizar. En primer lugar, están las predicciones de compras, donde se analizan los patrones de navegación, el historial de compras, las preferencias y las características demográficas del cliente para predecir sus compras futuras. Luego, se encuentra la predicción de clics, que busca anticipar los enlaces o elementos con los cuales un cliente interactuará en un sitio web, con el objetivo de mejorar la calidad del contenido y la usabilidad del sitio. Además, se encuentra la predicción de abandono de carrito, que permite identificar a aquellos clientes que agregan productos a un carrito de compra pero no completan el proceso de compra, con el fin de tomar acciones de recuperación o retención del cliente. Por último, está la predicción de retención de clientes, que busca predecir qué clientes están más propensos a abandonar o finalizar su relación con el sitio web, para poder implementar estrategias que aumenten su fidelización y retención.

3.2. Comportamiento del cliente/afiliado en el canal web

3.2.1. Definición y relevancia del comportamiento del cliente para el negocio

Considerando los modelos de negocio establecidos por las Administradoras de Fondos de Pensiones (AFP), surge la importancia de la figura del cliente. Según la Real Academia Española, un cliente es una persona que realiza una compra o utiliza los servicios ofrecidos por un profesional o empresa (Real Academia Española, s.f). Sin embargo, en el contexto de las AFP, los clientes se denominan afiliados, ya que contribuyen o están inscritos en un plan de pensiones (Rasekhi, Fard, y Kim, 2016).

El afiliado es el centro del negocio y su importancia radica principalmente en la rentabilidad que aporta. Cada trabajador que decide afiliarse representa una ganancia, mientras que cada afiliado que decide desafiliarse genera una pérdida. Además, la

experiencia del servicio que brinda la AFP hacia el afiliado es crucial, ya que puede promover la marca si es positiva. En tercer lugar, el afiliado, al ser una fuente de ganancias para el modelo, puede contribuir al crecimiento de la empresa al darle su preferencia. Además, la experiencia del cliente y su retroalimentación son valiosas, ya que pueden proporcionar conocimientos sobre los puntos débiles y las áreas de mejora del sistema (Rodríguez, 2023).

Dentro de las diferentes funciones que tiene el cliente, en primer lugar, se encuentra el cliente como consumidor. Esta es una de las funcionalidades más tradicionales, ya que el objetivo intrínseco del cliente es consumir o contratar servicios. Como consumidor, adquiere un producto o servicio y lo utiliza para satisfacer una necesidad, lo que representa la principal fuente de ingresos para la empresa.

En segundo lugar, se encuentra el cliente como "prosumidor", es decir, alguien que consume y produce al mismo tiempo (Toffler, 1980). Además de consumir, el cliente también deja reseñas o realiza comentarios en lugares especializados, lo cual es útil para generar información que mejore la experiencia del servicio.

En tercer lugar, se considera al cliente como crítico, ya que si la experiencia del cliente es negativa, los comentarios y reseñas negativas que proporcione pueden tener un impacto constructivo o destructivo.

En cuarto lugar, el cliente es una pieza fundamental en el desarrollo de productos y servicios. Los comentarios de los clientes pueden guiar el desarrollo de servicios innovadores que se ajusten a las necesidades que ellos indican. En el caso específico de las AFP, esto se refiere a los afiliados.

En quinto lugar, el cliente se desempeña como evaluador de la experiencia. Relacionado con los puntos anteriores, la mejor manera de mejorar la experiencia del cliente es tener en cuenta sus comentarios sobre este aspecto, lo que puede marcar la diferencia frente a otras empresas competidoras en el mercado.

Por último, el cliente puede convertirse en un embajador eventual de la marca, es decir, puede promover el negocio mediante recomendaciones, comentarios y reseñas

positivas.

3.2.2. Características del comportamiento del cliente en el canal web

Para comprender la experiencia y el comportamiento del cliente en un canal web, es importante reconocer la existencia del customer journey, el cual describe las distintas etapas por las que un cliente pasa al consumir un producto o servicio. Según (Lemon y Verhoef, 2016), estas etapas incluyen la conciencia, investigación, consideración, compra, uso y evaluación. La etapa de conciencia refiere a la identificación de una necesidad o problema que debe ser resuelto, mientras que la investigación implica la búsqueda de información por parte del cliente para encontrar posibles soluciones y comparar entre diferentes opciones disponibles. Luego, en la etapa de consideración, el cliente evalúa las alternativas y elige la que mejor se adapte a sus necesidades, lo que lleva a la etapa de compra, donde se realiza la contratación o adquisición del servicio seleccionado. Posteriormente, viene la etapa de uso, en la cual el cliente experimenta y evalúa la calidad, funcionalidad y experiencia del servicio. Por último, se encuentra la etapa de evaluación, en la cual el cliente emite un feedback voluntario, tanto positivo como negativo, sobre su experiencia satisfactoria o insatisfactoria. En resumen, las opciones disponibles en el canal web buscan hacer del customer journey una experiencia eficiente y agradable.

Para acceder al canal web de AFP Capital, es necesario ser afiliado y contar con una cuenta privada personal que incluya el RUT y contraseña. Una vez ingresado al canal web privado, los afiliados tienen a su disposición diversas opciones para satisfacer sus necesidades. Estas incluyen revisión del pago o no de la cotización mensual, la obtención de certificados de cotizaciones, afiliación, antecedentes previsionales y traspaso de fondos, así como certificados tributarios. Además, se pueden obtener certificados generales, como de residencia, suscripción de ahorro previsional voluntario (APV), cuenta 2, remuneraciones imponibles, periodos no cotizados y trabajo pesado. En el caso de afiliados pensionados, también se pueden obtener certificados de asignación familiar, calidad de pensionado, pensiones pagadas, pensión en trámite, ingreso base y comprobante de pago de pensión. Además, es posible acceder a la cartola en línea. El canal web privado permite realizar el ahorro obligatorio y voluntario,

inversiones, depósitos directos, consultar planillas de pagos y ver las comisiones cobradas como afiliado. También ofrece la opción de verificar el fondo de pensiones, los tipos de fondos disponibles (A, B, C, D, E) y sus porcentajes de rentabilidad, así como realizar cambios de fondo de pensiones y acceder a educación previsional. Además, se brinda la posibilidad de realizar giros en cuentas personales, acceder a rescates financieros y tramitar la pensión.

3.2.3. Factores que afectan el comportamiento del cliente

(Lemon y Verhoef, 2016) proponen que los principales factores que influyen en el comportamiento del usuario y su experiencia son los sensoriales, afectivos, cognitivos, puntos de contacto y externos. La experiencia sensorial se refiere a los aspectos perceptibles por los sentidos del cuerpo, como la vista, el olfato y el tacto. En cuanto a la experiencia afectiva, se debe considerar la emocionalidad del cliente como resultado de la experiencia con el producto o servicio. En el aspecto cognitivo, se refiere a los pensamientos, creencias y actitudes que el cliente puede tener hacia la compañía, el producto o el servicio (Lemon y Verhoef, 2016). Los puntos de contacto hacen referencia a las diferentes formas en que el cliente y la compañía interactúan, como la publicidad, el servicio al cliente, las redes sociales o las interacciones transaccionales. Por último, el factor externo se refiere al contexto actual, las condiciones socioeconómicas y otros factores que pueden afectar la experiencia del usuario y que están fuera del control de la compañía.

Dentro de los factores que pueden influir en el comportamiento de un cliente en el canal web, se encuentran principalmente la usabilidad y el diseño. En cuanto a la usabilidad, depende de siete características que garantizan una buena experiencia para el usuario. Según Sánchez (Sánchez, 2011), la accesibilidad, legibilidad, navegabilidad, facilidad de aprendizaje, velocidad de utilización, eficiencia del usuario y tasas de error del canal web influyen en la experiencia del usuario y en el feedback que este pueda brindar sobre el uso de los servicios.

Por otro lado, el diseño del sitio web depende de cinco características para garantizar un buen contenido y estética, y lograr que el usuario encuentre lo que busca en el menor tiempo posible, es decir, eficiencia. El autor Walter Sánchez

(Sánchez, 2011) indica que el diseño debe ser entendible, novedoso, comprensible, inteligente y atractivo, lo que permite acercar los contenidos de mejor manera al usuario y lograr una navegación más intuitiva. Estos factores son de gran importancia para que el usuario pueda encontrar el contenido que busca en el menor tiempo posible y tener una experiencia positiva al interactuar con la interfaz del sitio web.

3.3. Herramientas para la predicción del comportamiento del cliente en el canal web

3.3.1. Introducción a las herramientas de análisis de datos

En el entorno empresarial actual, la capacidad de tomar decisiones informadas y basadas en datos se ha vuelto fundamental para el éxito y la competitividad de las organizaciones. El análisis de datos desempeña un papel crucial en este proceso, permitiendo a las empresas obtener información valiosa a partir de grandes volúmenes de datos y utilizarla para comprender el comportamiento del cliente de manera más profunda y precisa. Esto resulta de suma importancia, ya que la calidad de las decisiones tomadas marca la diferencia entre el éxito y el fracaso (Contreras Arteaga y Sánchez Cotrina, 2019).

Dentro de las herramientas de análisis de datos, se destacan cuatro conceptos clave que han revolucionado la forma en que se procesan y se obtiene información de los datos: Business Intelligence, Big Data, Machine Learning y Data Mining. Estas herramientas proporcionan a las empresas la capacidad de extraer conocimientos y patrones significativos de los datos, lo que a su vez les permite tomar decisiones estratégicas más acertadas y personalizar sus estrategias de marketing y atención al cliente.

El Business Intelligence (BI) se refiere a la recopilación, análisis y presentación de datos empresariales para facilitar la toma de decisiones. Mediante el uso de diversas técnicas y herramientas, el BI permite a las empresas visualizar y comprender mejor los datos de sus operaciones y clientes. Esto incluye la generación de informes, el análisis de tendencias, la monitorización de indicadores clave de rendimiento (KPI) y

la creación de tableros de control interactivos. El BI ayuda a las organizaciones a identificar oportunidades, detectar áreas de mejora y optimizar su rendimiento en función de datos históricos y en tiempo real. Sobre la inteligencia de negocios, se ha determinado que cada implementación es única para cada proceso empresarial (Garcia-Estrella y Barón Ramírez, 2021).

El Big Data se refiere a la gestión y análisis de grandes volúmenes de datos, tanto estructurados como no estructurados, que superan la capacidad de las herramientas tradicionales de almacenamiento y procesamiento. El Big Data se caracteriza por las tres V's: Volumen (gran cantidad de datos), Velocidad (alta velocidad de generación y procesamiento de datos) y Variedad (diversidad de fuentes y formatos de datos). Para aprovechar el potencial del Big Data, las empresas emplean técnicas de procesamiento distribuido y herramientas específicas para el almacenamiento, procesamiento y análisis de estos datos masivos. El análisis de Big Data permite identificar patrones, tendencias y correlaciones ocultas en los datos, lo que brinda información valiosa para entender y anticipar el comportamiento del cliente.

El Machine Learning (aprendizaje automático) es una rama de la inteligencia artificial que permite a los sistemas informáticos aprender y mejorar automáticamente a partir de la experiencia sin ser programados explícitamente. En lugar de basarse en una analítica descriptiva, el Machine Learning ofrece una analítica predictiva (Garcia-Estrella y Barón Ramírez, 2021). Mediante algoritmos y modelos, el Machine Learning permite a las empresas analizar grandes conjuntos de datos y detectar patrones complejos en el comportamiento del cliente. Esto permite realizar predicciones y recomendaciones personalizadas, así como automatizar tareas y procesos, lo que mejora la eficiencia operativa y la experiencia del cliente.

El Data Mining (minería de datos) se refiere al proceso de descubrir información valiosa, patrones y relaciones desconocidas en grandes conjuntos de datos. Utilizando técnicas estadísticas y algoritmos avanzados, el Data Mining permite identificar correlaciones y tendencias ocultas en los datos, lo que ayuda a las empresas a comprender mejor el comportamiento del cliente y tomar decisiones más acertadas. Esta herramienta es especialmente útil para la segmentación de clientes, la detección de fraudes, la recomendación de productos y la personalización de ofertas.

3.3.2. Métodos, técnicas y tecnologías de análisis de datos

En el análisis de datos para predecir el comportamiento del cliente, se utilizan una variedad de métodos, técnicas y tecnologías que permiten procesar y analizar grandes volúmenes de información con el fin de obtener información valiosa. Estas herramientas proporcionan a las empresas y organizaciones la capacidad de comprender mejor a sus clientes, identificar patrones y tendencias, y tomar decisiones estratégicas más acertadas.

Entre los métodos y modelos más utilizados se encuentran la regresión logística, que permite predecir la probabilidad de que un cliente realice una determinada acción o tome una decisión; el clustering, que agrupa a los clientes en segmentos o categorías similares con características y comportamientos comunes; los árboles de decisión, que representan un conjunto de reglas lógicas para clasificar a los clientes en diferentes grupos; el Random Forest, que combina múltiples árboles de decisión para mejorar la precisión de las predicciones; y el Gradient Boosting Machine, que utiliza múltiples modelos de aprendizaje débiles para construir un modelo más robusto y preciso.

Además de los métodos y modelos, existen diversas técnicas que se aplican en el análisis de datos para predecir el comportamiento del cliente. Entre ellas se encuentran las redes neuronales artificiales (ANN), que son modelos inspirados en el funcionamiento del cerebro humano y se utilizan para reconocer patrones y realizar predicciones complejas; y el Support Vector Machine (SVM), que es un algoritmo de aprendizaje automático utilizado para clasificar y predecir datos.

En cuanto a las tecnologías utilizadas en el análisis de datos, se destacan diversas herramientas y lenguajes de programación. Algunas de las más populares son Tableau, que permite visualizar y explorar los datos de manera interactiva; Python, con bibliotecas como Pandas, NumPy y Scikit-learn, que ofrecen una amplia gama de funciones y algoritmos para el análisis de datos; R, con paquetes como dplyr, caret y randomForest, que brindan herramientas estadísticas y de aprendizaje automático; Apache Spark, que permite procesar y analizar grandes volúmenes de datos de manera distribuida; KNIME y RapidMiner, que son plataformas de análisis de datos visuales; y QlikView y Power BI, que son herramientas de visualización de datos y creación de

tableros de control.

3.3.3. Modelos de predicción de comportamiento del cliente

En la era digital, los modelos de predicción de comportamiento del cliente son un recurso fundamental para las empresas que buscan tomar decisiones informadas y personalizar sus estrategias. Este subcapítulo del informe se adentrará en los diversos modelos utilizados para anticipar las acciones y preferencias de los clientes, destacando su relevancia en la toma de decisiones estratégicas y la mejora de la experiencia del usuario en el entorno online.

Modelos de regresión logística

La regresión logística corresponde a un algoritmo de aprendizaje automático supervisado que es empleado para resolver problemas de clasificación. Si bien, su nombre contiene “regresión”, en realidad corresponde a un método de clasificación.

Se da uso a la regresión logística cuando la variable de respuesta o variable objetivo es categórica. En lugar de predecir un valor numérico como en la regresión lineal, la regresión logística estima la probabilidad de que una observación pertenezca a una categoría específica.

Los modelos de regresión logística se basan en la función logística, también conocida como función sigmoide, que mapea cualquier valor real a un rango entre 0 y 1. La función sigmoide tiene la siguiente forma matemática (Sibila, Adriana, y Vlatko, 2020):

$$f(z) = \frac{1}{(1 + e^{-z})}$$

En la regresión logística, se ajusta un modelo lineal a los datos de entrada y se aplica la función sigmoide al resultado para obtener la probabilidad de pertenencia a una clase (Sibila y cols., 2020). La ecuación del modelo se expresa como:

$$p(y = 1|x) = \frac{1}{(1 + e^{-(b_0+b_1x_1+b_2x_2+\dots+b_nx_n)})}$$

Donde:

$p(y=1|x)$ es la probabilidad condicional de que la variable de respuesta sea igual a 1 dada la entrada x .

$b_0, b_1, b_2, \dots, b_n$ son los coeficientes del modelo que se ajustan durante el proceso de entrenamiento.

x_1, x_2, \dots, x_n son los valores de las variables de entrada.

El proceso de ajuste de la regresión logística implica encontrar los mejores valores para los coeficientes del modelo con la finalidad de maximizar la verosimilitud de los datos observados (Sibila y cols., 2020). Esto se puede hacer mediante métodos numéricos como la maximización de la función de verosimilitud o mediante algoritmos de optimización como el gradiente descendente.

Una vez entrenado el modelo, se puede utilizar para hacer predicciones clasificando nuevas observaciones según la probabilidad estimada. Por ejemplo, si la probabilidad estimada de pertenencia a una clase es superior a un umbral (generalmente 0.5), se clasificará como perteneciente a esa clase.

Para nuestro caso en particular, puede ser utilizado el modelo de regresión logística para predecir el comportamiento de usuarios en un canal web, para ello se necesitaría tener datos históricos que contengan información relevante sobre el comportamiento pasado de los usuarios y las variables predictoras asociadas. Estas variables predictoras pueden incluir características demográficas, patrones de uso del sitio web o aplicación, historial de compras, interacciones anteriores, entre otros.

Una vez que se tienen los datos y las variables predictoras, se puede entrenar un modelo de regresión logística utilizando técnicas de ajuste como la maximización

de la verosimilitud o el gradiente descendente. Una vez entrenado el modelo, puede ser utilizado para predecir el comportamiento futuro de los usuarios en función de nuevas observaciones o datos entrantes.

Es importante tener en consideración que la calidad de las predicciones dependerá de la calidad de los datos utilizados para entrenar el modelo y de la selección adecuada de las variables predictoras. Además, es fundamental realizar una validación adecuada del modelo utilizando técnicas como la validación cruzada o la separación de conjuntos de entrenamiento y prueba para evaluar su rendimiento y generalización en datos no vistos.

Ventajas de los modelos de regresión logística

- **Interpretación de resultados:** La regresión logística proporciona coeficientes que indican la dirección y la magnitud de la relación entre las variables predictoras y la variable de respuesta. Esto permite interpretar el efecto relativo de cada variable en la probabilidad de pertenecer a una clase específica.
- **Manejo de variables independientes categóricas:** La regresión logística puede manejar tanto variables independientes continuas como categóricas. Incluso puede manejar variables categóricas con más de dos categorías mediante técnicas como la codificación de variables ficticias.
- **Estimación de probabilidades:** La regresión logística estima la probabilidad de pertenencia a una clase específica en lugar de simplemente clasificar observaciones en categorías. Esto es útil cuando se necesita una medida de certeza o riesgo asociado con la clasificación.
- **Buena capacidad de generalización:** La regresión logística puede funcionar bien con conjuntos de datos pequeños o moderados, y es menos propensa al sobreajuste en comparación con otros algoritmos más complejos. Esto la hace adecuada para aplicaciones con muestras limitadas.

Desventajas de los modelos de regresión logística

- **Linealidad de la relación:** La regresión logística asume una relación lineal entre las variables predictoras y la probabilidad logarítmica de la variable de respuesta. Si existe una relación no lineal, la regresión logística puede no ajustarse adecuadamente o requerir transformaciones adicionales de las variables.
- **Sensible a valores atípicos y datos faltantes:** Los valores atípicos o datos faltantes pueden afectar negativamente el rendimiento de la regresión logística. Es necesario manejarlos adecuadamente para evitar sesgos o imprecisiones en los resultados.
- **Suposición de independencia:** La regresión logística asume que las observaciones son independientes entre sí. Si hay dependencias o correlaciones entre las observaciones, la precisión de los resultados puede verse comprometida.
- **No apto para problemas no lineales:** Si existe una relación compleja y no lineal entre las variables predictoras y la variable de respuesta, la regresión logística puede no ser el modelo más adecuado. En tales casos, se pueden requerir técnicas más avanzadas, como modelos no lineales o de aprendizaje profundo.

Modelos de recomendación

Los modelos de recomendación son algoritmos y técnicas utilizados en sistemas de recomendación para ofrecer sugerencias personalizadas a los usuarios. Estos modelos se utilizan en una amplia gama de aplicaciones, como plataformas de comercio electrónico, servicios de streaming de música y video, redes sociales y más (Elizabeth, 2023).

El objetivo de un modelo de recomendación es predecir o sugerir elementos que sean relevantes o interesantes para un usuario en particular, basándose en su historial de preferencias, comportamiento pasado o en información de usuarios similares. Estos

modelos aprovechan el poder del aprendizaje automático y la minería de datos para analizar patrones y relaciones en grandes conjuntos de datos.

Existen varios tipos de modelos de recomendación, entre los más comunes se encuentran (Vatsal, 2021):

- **Filtrado colaborativo:** Este enfoque se basa en la idea de que si a un grupo de usuarios con preferencias similares les gusta un conjunto de elementos, entonces a un usuario nuevo con características similares también le podrían gustar esos elementos. El filtrado colaborativo utiliza la información de las interacciones pasadas de los usuarios (por ejemplo, clasificaciones o historial de compras) para generar recomendaciones.
- **Filtrado basado en contenido:** Este enfoque utiliza información sobre las características y atributos de los elementos para recomendar otros elementos similares. Por ejemplo, en un servicio de streaming de música, se pueden recomendar canciones o artistas similares a los que un usuario ha escuchado anteriormente en función de género, estilo o letras.
- **Modelos híbridos:** Estos modelos combinan múltiples enfoques, como filtrado colaborativo y basado en contenido, para aprovechar sus fortalezas y proporcionar recomendaciones más precisas y personalizadas.

Los modelos de recomendación se construyen utilizando técnicas de aprendizaje automático, como regresión logística, árboles de decisión, redes neuronales o algoritmos de factorización matricial. Estos modelos se entrenan utilizando conjuntos de datos históricos que contienen información sobre las preferencias y elecciones de los usuarios, y luego se aplican en tiempo real para generar recomendaciones en función de nuevos datos.

Ventajas de los modelos de recomendación

- **Personalización:** Los modelos de recomendación ofrecen sugerencias personalizadas a los usuarios, lo que mejora la experiencia del usuario y facilita la búsqueda

de productos o contenido relevante.

- **Descubrimiento de nuevos elementos:** Los modelos de recomendación pueden ayudar a los usuarios a descubrir nuevos elementos que podrían ser de su interés, ampliando así sus opciones y experiencias.
- **Mejora de la retención y fidelidad de los usuarios:** Al proporcionar recomendaciones precisas y relevantes, los modelos de recomendación pueden aumentar la satisfacción del usuario, mejorar la retención y fomentar la fidelidad a la plataforma o servicio.
- **Eficiencia en la toma de decisiones:** Los usuarios pueden ahorrar tiempo y esfuerzo al recibir sugerencias personalizadas, lo que les ayuda a tomar decisiones más rápidas y eficientes.

Desventajas de los modelos de recomendación

- **Sesgo y burbujas de filtro:** Los modelos de recomendación pueden verse afectados por el sesgo inherente en los datos de entrenamiento y pueden crear burbujas de filtro, limitando la diversidad y la exposición a nuevas ideas o perspectivas.
- **Fracaso en captar preferencias cambiantes:** Los modelos de recomendación pueden tener dificultades para captar las preferencias cambiantes de los usuarios a medida que sus gustos y necesidades evolucionan con el tiempo.
- **Problemas de inicio en frío:** Los modelos de recomendación pueden tener dificultades para ofrecer recomendaciones precisas para nuevos usuarios o elementos que tienen una falta de información histórica.
- **Privacidad y preocupaciones éticas:** Los modelos de recomendación recopilan y utilizan datos de los usuarios, lo que puede plantear preocupaciones de privacidad y cuestiones éticas relacionadas con el manejo de la información personal.

Modelos de series temporales

Los modelos de series temporales son técnicas utilizadas para analizar y predecir datos secuenciales que están organizados en función del tiempo. En una serie temporal, los datos se registran en intervalos regulares (como horas, días, meses, etc.) y cada punto de datos está asociado con una marca de tiempo (Ajitesh, 2023).

El objetivo principal de los modelos de series temporales es comprender y capturar los patrones, tendencias y estacionalidad en los datos a lo largo del tiempo, y utilizar esta información para hacer predicciones futuras. Estos modelos son ampliamente utilizados en diversos campos, como la economía, las finanzas, la meteorología, la demanda de productos, la planificación de inventario y más.

Los modelos de series temporales se basan en la suposición de que los datos pasados pueden proporcionar información útil para predecir el futuro. Algunos de los modelos más comunes utilizados en el análisis de series temporales son (Ajitesh, 2023):

- **Media móvil (MA):** Este modelo estima el valor futuro de la serie temporal en función de un promedio de los errores pasados. Se utiliza para capturar patrones aleatorios o no sistemáticos en los datos.
- **Autoregresión (AR):** Este modelo estima el valor futuro de la serie temporal en función de valores pasados de la propia serie. Se utiliza para capturar la dependencia de la serie en sí misma a lo largo del tiempo.
- **Autoregresión de media móvil (ARMA):** Este modelo combina los enfoques AR y MA para capturar tanto la dependencia de la serie en sí misma como los patrones aleatorios.
- **Autoregresión integrada de media móvil (ARIMA):** Este modelo amplía el modelo ARMA al considerar también las diferencias entre los valores de la serie temporal. Se utiliza para capturar tendencias y estacionalidad en los datos.

Además de estos modelos clásicos, también se utilizan enfoques más avanzados, como los modelos de espacio de estados, los modelos de suavizado exponencial y los modelos de redes neuronales recurrentes (RNN), que pueden capturar relaciones más complejas y no lineales en los datos de series temporales.

Es importante destacar que el análisis de series temporales requiere un enfoque cuidadoso para la selección del modelo, la identificación de patrones y la evaluación de la precisión de las predicciones. Además, se deben tener en cuenta factores como la estacionalidad, la estacionariedad de la serie y la presencia de datos faltantes o valores atípicos para obtener resultados confiables.

Ventajas de los modelos de series temporales

- **Captura de patrones temporales:** Los modelos de series temporales pueden capturar patrones, tendencias y estacionalidad en los datos a lo largo del tiempo. Esto permite comprender mejor la dinámica de los datos y hacer predicciones más precisas.
- **Predicciones a corto plazo:** Los modelos de series temporales son adecuados para hacer predicciones a corto plazo, ya que utilizan la información histórica para predecir los valores futuros. Esto es especialmente útil en aplicaciones donde se necesita anticipar eventos próximos, como demanda de productos o pronóstico del clima.
- **Utilización de datos secuenciales:** Los modelos de series temporales aprovechan la estructura secuencial de los datos y utilizan la información de los puntos anteriores para hacer predicciones en el siguiente punto. Esto permite tener en cuenta la dependencia temporal en los datos y obtener resultados más precisos.
- **Flexibilidad en la elección del modelo:** Existen diferentes tipos de modelos de series temporales que se pueden utilizar según la naturaleza de los datos y los patrones presentes. Esto proporciona flexibilidad para seleccionar el modelo más adecuado para el problema específico.

Desventajas de los modelos de series temporales

- **Sensibilidad a datos faltantes o valores atípicos:** Los modelos de series temporales pueden verse afectados negativamente por la presencia de datos faltantes o valores atípicos. Estos pueden distorsionar los patrones y afectar la precisión de las predicciones.
- **Dificultad con tendencias no lineales:** Los modelos de series temporales asumen a menudo que las relaciones son lineales o pueden ser capturadas por modelos lineales. Si hay tendencias no lineales en los datos, los modelos lineales pueden no ajustarse adecuadamente y se pueden requerir enfoques más avanzados.
- **Necesidad de datos históricos adecuados:** Los modelos de series temporales requieren una cantidad suficiente de datos históricos para hacer predicciones precisas. En ausencia de datos suficientes, los modelos pueden tener dificultades para capturar patrones y generar resultados confiables.
- **Problemas con cambios estructurales:** Si hay cambios estructurales significativos en los datos de series temporales (por ejemplo, cambios en la estacionalidad o en los patrones), los modelos de series temporales pueden tener dificultades para adaptarse y pueden requerir ajustes manuales.

Modelos de atribución

Los modelos de atribución permiten predecir el recorrido que los clientes seguirán al momento de concretar una compra. Este recorrido puede contener las redes sociales, el uso del sitio web del vendedor, el correo electrónico, entre otros. Los modelos de atribución permiten determinar el impacto que tiene el uso de las acciones para el sistema de marketing (Baker, 2023). Este tipo de modelo permite darle mayor importancia a los canales de marketing y a los puntos de contacto que existen entre el cliente y el vendedor, que llevaron al cliente a realizar una compra.

Al asignar crédito a sus canales de marketing y puntos de contacto, se puede

aumentar la posibilidad de que los clientes logren concretar una compra, esto a través de la identificación de las áreas del recorrido del comprador que se puedan mejorar, la determinación del retorno de la inversión para cada canal o punto de contacto, el descubrimiento de las áreas más efectivas para gastar el presupuesto de marketing y la adaptación de las campañas de marketing y muestra de contenido totalmente personalizado por clientes (Baker, 2023)

Existen variados tipos de modelos de atribución, todos tienen el mismo procedimiento de asignar crédito a los canales y punto de contacto, cada uno de estos tipos de modelo le atribuyen un peso distinto a cada canal y punto de contacto (Baker, 2023). Los modelos a continuación son los más aptos para lograr la predicción del comportamiento de un cliente:

- **Modelo de atribución Multi-Touch:** Este modelo demuestra ser poderoso ya que tiene en cuenta todos los canales y puntos de contacto con los que los clientes interactúan a lo largo de su camino al concretar una compra. Deja en evidencia cuáles de los canales y punto de contacto fueron más influyentes y de cómo estas trabajaron en conjunto para influenciar al cliente.
- **Modelo de atribución Lineal:** Corresponde a un tipo de modelo de atribución Multi-Touch que le entrega el mismo peso a cada uno de los canales y puntos de contacto con los que el cliente interactúa en su camino al concretar una compra.
- **Modelo de atribución Time-Decay:** También llamado modelo de atribución de declive en el tiempo, además de considerar todos los puntos de contacto, también considera el tiempo que cada uno de estos puntos de contacto ocurrió, por lo que, los puntos de contacto o interacciones que sucedieron más cercano al momento en que se concretó la compra reciben mayor peso.

Ventajas de los modelos de atribución

- **Facilita el rastrear de mejor manera el paso a paso del cliente:** Esto gracias a la atención que se le entrega a cada canal y punto de contacto con el cual el cliente interactúa a la hora de concretar una compra.

- **Permiten mayor personalización de rastreo de los clientes:** Al saber que canales y punto de contacto tiene cada uno de los clientes, se puede llegar a entregar una experiencia personalizada a cada uno de los clientes.
- **Comprender la contribución de cada canal y punto de contacto:** Permite comprender como cada canal y punto de contacto contribuye a lograr los objetivos comerciales. Siendo de gran ayuda para identificar como asignar los recursos de manera mas efectiva y lograr optimizar las estrategias.
- **Identificar canales y puntos de contacto de alto rendimiento:** Un modelo de atribución puede revelar qué canales o puntos de contacto tienen un mayor interacción con los clientes en términos de generación de resultados. Esto permite a las empresas enfocar sus recursos en los canales más efectivos y maximizar su retorno de inversión.

Desventajas de los modelos de atribución

- **Poseen una mayor complejidad que los otros modelos:** La implementación de un modelo de atribución puede ser compleja y requerir un enfoque personalizado según las necesidades y características de cada empresa. Además, no hay un modelo de atribución único que sea universalmente aceptado, lo que puede generar falta de consenso y confusión en la industria.
- **La interpretación de los resultados puede ser subjetiva:** La interpretación de los resultados de un modelo de atribución puede estar sujeta a la interpretación y suposiciones del analista. Diferentes personas pueden llegar a conclusiones diferentes basadas en los mismos resultados, lo que puede generar cierta subjetividad en la interpretación de los datos.
- **Poseen limitaciones en la medición del seguimiento:** El modelo de atribución depende de la disponibilidad y calidad de los datos. Si los datos son limitados o imprecisos, los resultados del modelo pueden no ser confiables o representativos de la realidad.

Modelos de arboles de decisión

Los árboles de decisión son modelos de aprendizaje supervisado que se utilizan para predecir a qué clase o categoría pertenece un caso conocido mediante uno o más atributos. Estos modelos se construyen utilizando un algoritmo llamado *partición binaria recursiva*. Durante el entrenamiento, el algoritmo realiza divisiones en un subconjunto de los datos basadas en decisiones asociadas a variables conocidas, generando así dos nuevos subconjuntos. Este proceso se repite de manera recursiva hasta alcanzar un punto de terminación predefinido, lo que resulta en la creación del clasificador basado en árbol de decisión. Luego, cada nuevo dato, que posee atributos conocidos, sigue las ramificaciones del árbol siguiendo las reglas y decisiones generadas durante el proceso de entrenamiento.

En la actualidad, los árboles de decisión son unos de los modelos de aprendizaje más utilizados debido a su buen rendimiento (Arana, 2021). Estos algoritmos pueden generar modelos predictivos tanto para variables cuantitativas (regresión) como para variables cualitativas o categóricas (clasificación).

Como se mencionó anteriormente, un árbol de decisión realiza tareas de clasificación. Un clasificador es un algoritmo que nos permite asignar sistemáticamente una clase a cada uno de los casos presentados.

Figura 2: Estructura de un árbol de decisión



Fuente: Aprende IA. Recuperado de <https://aprendeia.com/arboles-de-decision-clasificacion-teoria-machine-learning/>

En la figura anterior se puede visualizar la estructura que posee un árbol de decisión, en este se aprecia como actúa el algoritmo de partición binaria mencionado al comienzo, tomando un conjunto y separándolo en subconjuntos hasta llegar a un final previamente establecido.

Para estimar la precisión de un clasificador, se calcula la tasa de error de clasificación verdadera. Esta tasa se obtiene evaluando un conjunto de valores X a los que el clasificador asigna una clase incorrecta, y se divide por el total de valores en X . Idealmente, se debería conocer la clase de todos los casos en el universo antes del entrenamiento, o en su defecto, de una muestra de tamaño similar al universo. Sin embargo, en la mayoría de los casos reales, no se dispone de todos los datos

del universo, por lo que se trabaja con una muestra y se estima la tasa de error mencionada anteriormente utilizando *estimadores internos*.

Ventajas de los árboles de decisión

- **Interpretabilidad:** Los árboles de decisión son fácilmente interpretables y comprensibles para los humanos. La estructura del árbol se puede visualizar de manera intuitiva, lo que permite comprender cómo se toman las decisiones y qué atributos son más relevantes para la clasificación.
- **Facilidad de uso:** La construcción y el uso de un árbol de decisión son relativamente sencillos en comparación con otros algoritmos de aprendizaje automático más complejos. No requieren una preparación exhaustiva de los datos ni un procesamiento previo complicado. Además, los árboles de decisión pueden manejar datos numéricos y categóricos sin requerir transformaciones adicionales, lo que simplifica el flujo de trabajo de modelado.
- **Capacidad para manejar datos faltantes y variables irrelevantes:** Los árboles de decisión tienen la capacidad de manejar datos faltantes en los atributos de forma natural. Durante la construcción del árbol, si un atributo tiene valores faltantes, el modelo puede utilizar otros atributos para tomar decisiones sin requerir imputación de datos. Además, los árboles de decisión son resistentes a variables irrelevantes, lo que significa que pueden ignorar atributos que no aportan información útil para la clasificación.
- **Flexibilidad y robustez:** Los árboles de decisión pueden manejar tanto problemas de clasificación como de regresión. Además, son capaces de capturar relaciones no lineales entre los atributos y la variable objetivo. Aunque cada árbol individual puede ser susceptible al sobreajuste, se pueden aplicar técnicas de regularización, como la poda, para mejorar la generalización y evitar el sobreajuste.
- **Eficiencia en tiempo de entrenamiento y predicción:** Los árboles de decisión tienen tiempos de entrenamiento y predicción rápidos, ya que solo

implican la evaluación de una serie de reglas de decisión. Aunque el tiempo de construcción puede ser mayor para conjuntos de datos grandes, una vez construido, el árbol puede ser utilizado eficientemente para hacer predicciones en tiempo real.

Desventajas de los árboles de decisión

- **Sensibilidad a cambios pequeños en los datos:** Los árboles de decisión son muy sensibles a cambios pequeños en los datos de entrenamiento. Una modificación mínima en los datos de entrada puede dar lugar a un árbol de decisión completamente diferente. Esto puede hacer que el modelo sea inestable y su rendimiento pueda variar significativamente.
- **Tendencia al sobreajuste:** Los árboles de decisión tienen la capacidad de adaptarse demasiado a los datos de entrenamiento. Si no se controla adecuadamente, el árbol puede memorizar el ruido o las fluctuaciones aleatorias en los datos de entrenamiento, lo que puede resultar en un mal rendimiento en datos nuevos y no vistos. La poda y otras técnicas de regularización se utilizan para mitigar este problema.
- **Limitaciones en la representación de relaciones complejas:** Aunque los árboles de decisión pueden capturar relaciones no lineales entre atributos y la variable objetivo, pueden tener dificultades para representar relaciones complejas que requieren una combinación de múltiples atributos. Las decisiones tomadas en cada nodo se basan en un solo atributo, lo que puede limitar su capacidad para modelar interacciones más sofisticadas.
- **Propensión a sesgos en los datos de entrenamiento:** Los árboles de decisión pueden verse afectados por sesgos en los datos de entrenamiento, especialmente cuando hay desequilibrios en las clases o falta representación de ciertas categorías. Esto puede resultar en una clasificación desigual o inexacta en casos minoritarios o poco representados.

Modelo Random Forest

El algoritmo random forest corresponde a un algoritmo empleado en machine learning registrado por Leo Breiman y Adele Cutler (IBM., 2023), este combina la salida de múltiples árboles de decisión para llegar a un resultado. El uso de random forest se ha hecho popular a causa de su facilidad de uso y flexibilidad, ya que puede ser empleado para problemas de clasificación y regresión.

El random forest se encuentra formado por varios árboles de decisión, los cuales son propensos a tener problemas como sesgos o sobreajuste, pero cuando se trata con una gran cantidad de árboles se logra llegar a resultados más precisos. "Mientras que los árboles de decisión consideran todas las posibles divisiones de características, los bosques aleatorios solo seleccionan un subconjunto de esas características." (IBM., 2023)

Random forest cuenta con tres hiperparámetros principales que se deben de configurar antes de iniciar el entrenamiento (IBM., 2023):

- Tamaño del nodo.
- Cantidad de árboles de decisión.
- Cantidad de características muestreadas.

El algoritmo se encuentra compuesto de un conjunto de árboles de decisión, cada árbol del conjunto se encuentra compuesto de una muestra de datos, la cual proviene de un conjunto de entrenamiento con reemplazo, llamada muestra de arranque (IBM., 2023).

A partir de la muestra de entrenamiento, se extrae un porcentaje para reservarlos como datos de prueba, los cuales se conocen como muestra fuera de la bolsa (oob). Luego, se inyecta otra instancia de aleatoriedad mediante el agrupamiento de características, lo que agrega más diversidad al conjunto de datos y reduce la correlación entre los árboles de decisión (IBM., 2023).

En un Random Forest, el proceso de predicción puede variar según el tipo de problema que se esté abordando (IBM., 2023). En el caso de tareas de regresión, se utiliza un enfoque de promediado, donde las predicciones de los árboles de decisión individuales se promedian para obtener el valor final de la predicción. Esto proporciona una estimación más precisa y estable del resultado deseado. Por otro lado, en tareas de clasificación, se utiliza un enfoque de votación mayoritaria. Cada árbol de decisión emite su propia predicción y la clase que obtiene la mayoría de votos se selecciona como la clase predicha. Esto permite tomar una decisión conjunta basada en las opiniones de múltiples árboles, lo que puede mejorar la precisión en la clasificación de las muestras.

Finalmente, la muestra extraída en un comienzo, la muestra fuera de la bolsa (oob) será utilizada para realizar una validación cruzada, finalizando la predicción.

Figura 3: Estructura de un random forest



Fuente: IBM. Recuperado de <https://www.ibm.com/mx-es/topics/random-forest>

Ventajas de random forest

- **Riesgo reducido de sobreajuste:** Los árboles de decisión corren el riesgo de sobre ajustarse, ya que tienden a ajustar todas las muestras que se encuentran dentro de los datos de entrenamiento. Sin embargo, cuando hay una gran cantidad de árboles de decisión dentro del random forest, el clasificador no será capaz de ajustarse demasiado al modelo, ya que el promedio de los árboles no correlacionados logra reducir la varianza general y el error de predicción.
- **Aporta Flexibilidad:** Debido a su capacidad para abordar con gran precisión tanto tareas de regresión como de clasificación, el método conocido como random forest es ampliamente utilizado por los científicos de datos. Además, su capacidad de agrupar características lo convierte en una herramienta eficaz para estimar valores faltantes, manteniendo la precisión incluso cuando falta parte de los datos.
- **Importancia de la característica fácil de determinar:** El random forest ofrece una forma conveniente de evaluar la importancia o contribución de las variables en un modelo. Existen varias formas de medir la importancia de las características. Por lo general, se utilizan el índice de Gini y la disminución media de impurezas (MDI) para evaluar cuánto afecta la exclusión de una variable específica a la precisión del modelo. Sin embargo, otra medida de importancia es la importancia de permutación, también conocida como precisión de disminución media (MDA). La MDA determina la disminución promedio en la precisión al permutar de forma aleatoria los valores de las características en las muestras out-of-bag (muestras que no se utilizan en el proceso de entrenamiento).

Desventajas de random forest

- **Proceso que requiere mucho tiempo:** Debido a que los algoritmos de random forest son capaces de manejar conjuntos de datos extensos, suelen ofrecer predicciones más precisas. Sin embargo, es importante tener en cuenta que el procesamiento de datos puede volverse lento, ya que se deben calcular los datos para cada árbol de decisión de forma individual.

- **Requiere más recursos:** Debido a que los random forest procesan conjuntos de datos más grandes, es cierto que se requieren más recursos para almacenar dichos datos. El aumento en el tamaño del conjunto de datos implica una mayor necesidad de memoria y capacidad de almacenamiento para garantizar un funcionamiento eficiente del algoritmo.
- **Más Complejo:** La interpretación de la predicción de un solo árbol de decisiones resulta más sencilla en comparación con la interpretación de un conjunto de árboles de decisión.

Autoencoders de Redes LSTM para Predicción de Comportamiento de Usuario

Los autoencoders de redes LSTM son una fusión de dos conceptos poderosos en el aprendizaje profundo: autoencoders y redes neuronales LSTM. Los autoencoders son una clase de redes neuronales utilizadas para la codificación de características; esencialmente, aprenden una representación comprimida de los datos de entrada, que luego pueden ser utilizados para reconstruir la entrada original. Esto es especialmente útil para reducir la dimensionalidad de los datos y descubrir relaciones latentes.

Las redes LSTM son una variante de las redes neuronales recurrentes (RNN) diseñadas para recordar información durante periodos extensos y son particularmente eficientes en el manejo de secuencias de datos con dependencias a largo plazo. Las RNN tradicionales luchan con el aprendizaje de estas dependencias debido al problema del desvanecimiento del gradiente, donde la información se pierde en cada paso a través del tiempo. Las LSTM abordan este problema con una estructura de celdas que incluye compuertas para regular el flujo de información, permitiendo que la red retenga o descarte datos a través de secuencias largas.

Al combinar ambos, los autoencoders LSTM pueden aprender a comprimir secuencias de datos temporales y, a la vez, capturar las complejidades de las secuencias temporales. Esto los hace idóneos para tareas como la predicción del comportamiento del usuario en canales web, donde es crucial comprender y actuar sobre patrones a lo largo del tiempo. Por ejemplo, pueden identificar secuencias de clics que conducen

a una compra o predecir cuándo un usuario está a punto de abandonar una sesión, permitiendo intervenir en tiempo real para mejorar la experiencia del usuario y aumentar la conversión.

En un canal web, donde cada acción del usuario es parte de una secuencia más grande de interacciones, los autoencoders LSTM pueden procesar esta secuencia como un todo cohesivo, identificando patrones en la forma en que los usuarios interactúan con el sitio. Esto va más allá de mirar eventos individuales aislados y permite a los sistemas de recomendación anticipar necesidades o intereses futuros basándose en el comportamiento previo del usuario.

Ventajas de los Autoencoders de Redes LSTM

- **Aprendizaje de secuencias temporales:** Los autoencoders LSTM son inherentemente buenos en aprender dependencias a largo plazo en datos secuenciales, lo que los hace ideales para rastrear y predecir el comportamiento del usuario en un sitio web a lo largo del tiempo.
- **Reducción de dimensionalidad:** Los autoencoders son eficaces para comprimir la información y resaltar características latentes, facilitando el manejo de grandes volúmenes de datos de usuario y su posterior análisis.
- **Capacidad de generalización:** Pueden generalizar aprendizajes a partir de datos históricos para predecir acciones futuras, lo que puede mejorar la personalización de contenidos y ofertas en un canal web.
- **Robustez frente al ruido:** Los LSTM pueden manejar el ruido en los datos de entrada, como sesiones de navegación erráticas o inusuales, identificando patrones consistentes y relevantes.

Desventajas de los Autoencoders de Redes LSTM

- **Complejidad computacional:** Los LSTM son modelos complejos que requieren una mayor capacidad de cómputo, lo que puede traducirse en un procesamiento

más lento y costos más elevados.

- **Dificultad en el ajuste de parámetros:** La configuración de estos modelos es a menudo menos intuitiva que otros métodos más simples, lo que puede llevar a un proceso de ajuste laborioso.
- **Riesgo de sobreajuste:** A pesar de su capacidad para generalizar, los LSTM pueden sobreajustarse a los datos de entrenamiento si no se implementan adecuadamente técnicas de regularización.
- **Sensibilidad a los datos de entrenamiento:** Los autoencoders LSTM aprenden de los datos disponibles, lo que significa que cualquier sesgo en estos puede llevar a recomendaciones sesgadas o irrelevantes.

3.3.4. Tabla comparativa de los modelos de predicción

La tabla comparativa proporciona información sobre los modelos planteados anteriormente, presentando de forma resumida sus Ventajas, Desventajas y Aplicaciones comunes de los modelos. Permitiendo tener una visión general de las características y consideraciones clave de cada modelo.

Figura 4: Tabla comparativa modelos de predicción

Modelo	Ventajas	Desventajas	Aplicaciones
Modelos de Regresión	Proporciona una relación cuantitativa entre variables independientes y la variable de respuesta.	Supone una relación lineal entre variables, lo que puede no ser válido en todos los casos.	Predicción de valores numéricos continuos.
Modelos de Recomendación	Personalización de sugerencias para los usuarios.	Puede requerir una gran cantidad de datos y tener problemas con datos faltantes o sesgos inherentes.	Recomendaciones de productos en comercio electrónico.
Modelos de Series Temporales	Captura patrones temporales y estacionales en los datos a lo largo del tiempo.	Sensibilidad a valores atípicos y datos faltantes, y dificultad para capturar tendencias no lineales.	Predicción de la demanda de productos. Pronóstico del clima.
Modelos de Atribución	Permite cuantificar la contribución relativa de diferentes variables a un resultado o impacto.	Puede ser difícil determinar la verdadera relación causal entre las variables.	Evaluación del retorno de inversión (ROI) de una campaña publicitaria.
Modelos de Árboles de Decisión	Proporciona una estructura de decisiones fácilmente interpretable.	Pueden ser propensos al sobreajuste si no se controla adecuadamente.	Clasificación y predicción en diversos campos, como medicina, marketing y finanzas.
Modelo Random Forest	Combina múltiples árboles de decisión para mejorar la precisión y evitar el sobreajuste.	Puede ser computacionalmente costoso y más difícil de interpretar que un solo árbol de decisión.	Clasificación y predicción en una amplia gama de aplicaciones, como análisis de datos médicos y detección de fraudes.
Autoencoders LSTM	Aprendizaje de secuencias temporales, reducción de dimensionalidad, capacidad de generalización, robustez frente al ruido.	Complejidad computacional, dificultad en el ajuste de parámetros, riesgo de sobreajuste, sensibilidad a los datos de entrenamiento.	Rastreo y predicción del comportamiento del usuario en sitios web, análisis de grandes volúmenes de datos.

Fuente: Elaboración propia.

3.3.5. Series Temporales

Luego de realizar distintas investigaciones, hemos llegado a la certeza de que serán necesarios los modelos basados en series temporales. Se llegó a esta decisión

debido que se busca predecir el comportamiento futuro de los clientes, en base a su ruta de navegación pasada. A continuación se abarcaran las distintas propiedades de series temporales y sus graficos, además, se presentaran 3 modelos ampliamente aplicados en series temporales, los cuales son, el modelo ARIMA, SARIMA y redes LSTM:

Graficos de series temporales

Uno de los primeros pasos para poder realizar cualquier tipo de análisis de datos es necesario graficar los datos, los gráficos permiten ver las distintas propiedades que los datos puedan tener, como lo son patrones, observaciones inusuales (outliers), cambios producidos por el tiempo y las relaciones que se encuentran entre las distintas variables que los datos puedan tener (Hyndman y Athanasopoulos, 2023). Por lo que, la selección de un buen grafico para representar los datos resulta ser de gran importancia para poder seleccionar el modelo predictivo más adecuado.

Dentro de esta sección revisaremos los gráficos más relevantes para nuestro estudio:

- **Trama de tiempo (Time plot):** Una de las maneras más practicas y usada para poder entender las series de tiempo es graficar los datos y una de las primeras opciones es una trama de tiempo. Esto quiere decir graficar la datos contra el tiempo de obtención de los datos.

La siguiente figura muestra la cantidad semanal de pasajeros que volaron en la clase económica de Ansett Airlines entre las dos ciudades más grandes de Australia (Hyndman y Athanasopoulos, 2023):

Figura 5: Cantidad semanal de pasajeros que volaron en la clase económica de Ansett Airlines



Fuente: Forecasting: Principles and Practice (Hyndman y Athanasopoulos, 2023). Recuperado de <https://otexts.com/fpp2/time-plots.html>

- **Patrones:** Una serie de tiempo puede presentar distintos tipos de patrones a lo largo del tiempo de la observación, y esta puede llegar a presentar este tipo de patrones (Hyndman y Athanasopoulos, 2023):
 - **Tendencia (Trend):** Una tendencia existe cuando se presenta un incremento o disminución en los datos dentro de un largo periodo de tiempo, esta tendencia no tiene por qué ser lineal, pudiendo tener tendencias positivas o negativas.
 - **Estacional (Seasonal):** Un patrón estacional existe cuando la serie de tiempo se ve afectada por factores como el tiempo del año o el día de semana. Gracias a esto se puede identificar que los patrones estacionales tienen una frecuencia conocida que por lo general no varía.

- **Cíclico (Cyclic):** Un patrón cíclico en una serie de tiempo existe cuando los datos presentan incrementos y disminuciones como lo haría un patrón estacional, pero sin una frecuencia conocida o fija, usualmente estos ciclos suelen tener una duración mayor a los de los patrones estacionales.

Muchas veces nos encontraremos con series de tiempo que contienen uno o más de estos patrones, por lo que para poder elegir un método predictivo resulta imperativo identificar los patrones presentes en los datos (Hyndman y Athanasopoulos, 2023).

- **Tramas de tiempo estacionales (Seasonal plots):** Esta grafica es similar a la trama de tiempo antes mencionada, la diferencia reside en el hecho de que los datos u observaciones están graficadas en contra de cada “estación individual” de tiempo donde se realizó la observación (Hyndman y Athanasopoulos, 2023).

En la siguiente figura se representa el valor monetario de las ventas mensuales de medicamentos antidiabéticos en Australia:

Figura 6: Ventas mensuales de medicamentos antidiabéticos en Australia



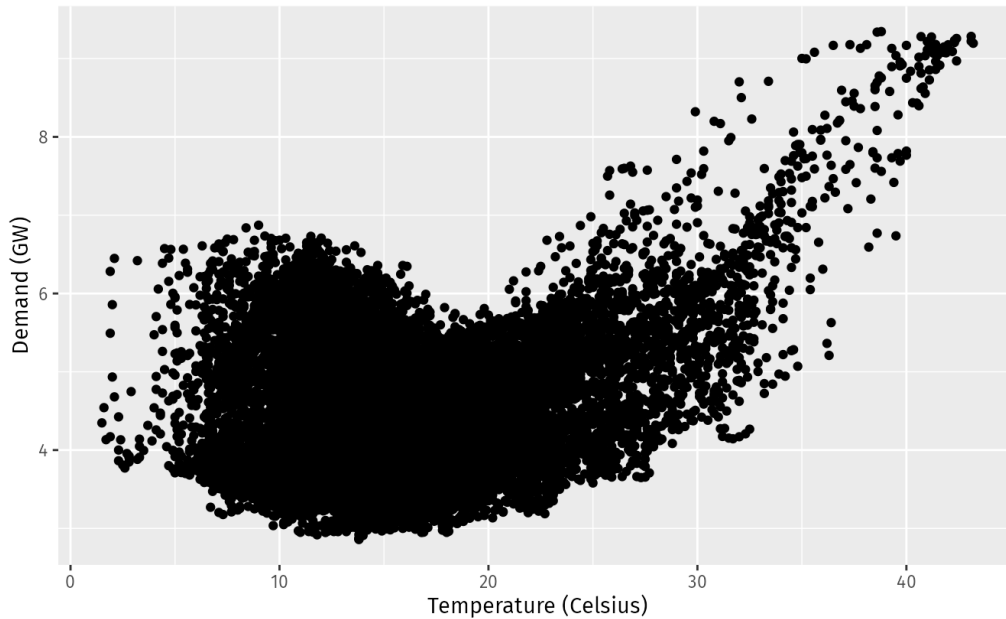
Fuente: Forecasting: Principles and Practice (Hyndman y Athanasopoulos, 2023). Recuperado de <https://otexts.com/fpp2/time-plots.html>

Este tipo de gráficos permite identificar de mejor manera los patrones que no se pudieron apreciar anteriormente en la trama de tiempo, ayuda especialmente para poder observar en que años el patrón de los datos cambia.

- **Gráficos de dispersión (Scatterplots):** Este tipo de grafico permite explorar las relaciones que existen entre series de tiempo, sus distintas variables y como esto afecta a la hora de predecir una serie de tiempo.

La siguiente figura muestra dos series de tiempo: demanda de electricidad (en GW y temperatura (Celsius)), para 2014 en Victoria, Australia. Las temperaturas son para Melbourne, la ciudad más grande de Victoria, mientras que los valores de demanda son para todo el estado (Hyndman y Athanasopoulos, 2023).

Figura 7: Demanda electrica debido a las temperaturas en 2 ciudades en Australia



Fuente: Forecasting: Principles and Practice (Hyndman y Athanasopoulos, 2023). Recuperado de <https://otexts.com/fpp2/time-plots.html>

Este grafico de dispersión es de gran ayuda para visualizar la relación que tienen las variables, y poder entender los datos de mejor manera.

- **Gráficos de desfase (Lag plots):** Este tipo de gráfico representa la observación $Y(t)$ graficada contra la observación $Y(t-k)$ para cada valor de k . Donde en el eje horizontal se muestran valores desfasados de la serie de tiempo (Hyndman y Athanasopoulos, 2023). La siguiente figura muestra diagramas de dispersión de la producción trimestral de cerveza australiana:

Figura 8: Producción trimestral de cerveza australiana



Fuente: Forecasting: Principles and Practice (Hyndman y Athanasopoulos, 2023). Recuperado de <https://otexts.com/fpp2/time-plots.html>

Los colores representan el cuatrimestre de la variable en el eje vertical. Se puede apreciar que en los desfases 4 y 8 se presenta una fuerte relación de las variables, lo que muestra un patrón estacional fuerte.

- **Correlación:** El estudio de la correlación explora la relación lineal entre dos variables, resulta de importancia calcularla, esto debido a que entrega que tan intrínsecamente relacionadas se encuentran las variables a analizar (Hyndman

y Athanasopoulos, 2023).

A continuación, se presenta la fórmula para conocer el coeficiente de correlación entre dos variables, x e y:

$$r = \frac{\sum (x_t - \bar{x})(y_t - \bar{y})}{\sqrt{\sum (x_t - \bar{x})^2} \sqrt{\sum (y_t - \bar{y})^2}}$$

El valor de r variara entre -1 y 1, dependiendo de que tan fuerte sea la correlación de las variables, mientras más cercano al -1, r representa una correlación negativa, por lo que, si r se encuentra más cercano a 1, esto quiere decir que las variables tienen una correlación positiva (Hyndman y Athanasopoulos, 2023).

- **Autocorrelación:** La autocorrelación mide la relación lineal entre valores rezagados de una serie de tiempo. Hay variados coeficientes de autocorrelación, dependiendo de cada panel del Lag plot, por ejemplo, r1 mide la relación entre las variables Y(t) y Y(t-1) y r2 mide la relación entre las variables Y(t) y Y(t-2) y así hasta considerar todos los datos (Hyndman y Athanasopoulos, 2023).

Para calcular el valor de r(k), donde T corresponde al largo de la serie de tiempo, se ocupa la siguiente fórmula:

$$r_k = \frac{\sum_{t=k+1}^T (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^T (y_t - \bar{y})^2}$$

De ejemplo, se calcularon los primeros 9 coeficientes de autocorrelación de la producción de cerveza en Australia estudiado en la sección pasada, obteniendo los siguientes valores:

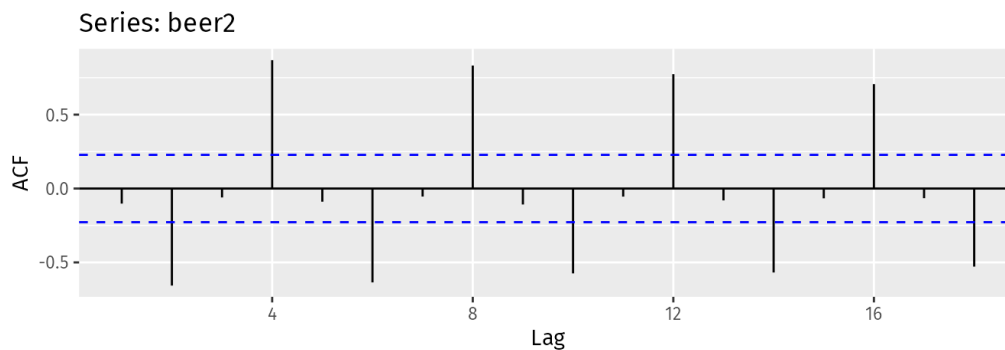
Figura 9: Coeficientes de autocorrelación de la producción de cerveza en Australia

r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9
-0.102	-0.657	-0.060	0.869	-0.089	-0.635	-0.054	0.832	-0.108

Fuente: Forecasting: Principles and Practice (Hyndman y Athanasopoulos, 2023). Recuperado de <https://otexts.com/fpp2/time-plots.html>

Estos coeficientes de autocorrelación son graficados para representar la **función de autocorrelación (ACF)**, que se muestra a continuación:

Figura 10: ACF de la producción trimestral de cerveza



Fuente: Forecasting: Principles and Practice (Hyndman y Athanasopoulos, 2023). Recuperado de <https://otexts.com/fpp2/time-plots.html>

Las líneas azules indican que tan lejos de 0 pueden estar las correlaciones para que estas sean significativas.

Cuando los datos tienen alguna tendencia, el ACF tiene valores positivos que lentamente van disminuyendo, si los datos presentan un patrón estacional, el ACF mostrara valores más grandes para los desfases estacionales, por lo general siguiendo alguna frecuencia estacional.

Cuando los datos presentan una tendencia y además un patrón estacional, se pueden apreciar ambos efectos (Hyndman y Athanasopoulos, 2023). En las

siguientes figuras se muestra la demanda mensual de electricidad de Australia, la cual presenta una tendencia y patrón estacional:

Figura 11: Demanda mensual de electricidad de Australia



Fuente: Forecasting: Principles and Practice (Hyndman y Athanasopoulos, 2023). Recuperado de <https://otexts.com/fpp2/time-plots.html>

Figura 12: ACF demanda mensual de electricidad de Australia



Fuente: Forecasting: Principles and Practice (Hyndman y Athanasopoulos, 2023). Recuperado de <https://otexts.com/fpp2/time-plots.html>

En el ACF se puede apreciar una tendencia, esto debido a que los valores van disminuyendo lentamente, mientras que la forma de ondas es debido a la estacionalidad que se presenta cada año.

- **Ruido Blanco (White Noise):** Las series de tiempo que no tienen autocorrelación son llamadas ruido blanco, para esto se espera que cada autocorrelación sea lo más cercana a 0. Debido a la variación que estos valores pueden tener, por lo que para que una serie de tiempo sea considerada ruido blanco, se espera que el 95 % de sus valores representados en el ACF estén unos límites demarcados por $\pm 2/\sqrt{T}$, donde T corresponde al largo de la serie de tiempo, estos límites se encuentran representados comúnmente por líneas azules en el ACF. Por consiguiente, si un valor o más se encuentra fuera de los límites o el más del 5 % de los valores estén fuera de los límites, la serie de tiempo en cuestión probablemente no sea ruido blanco.

Figura 13: Ruido blanco



Fuente: Forecasting: Principles and Practice (Hyndman y Athanasopoulos, 2023). Recuperado de <https://otexts.com/fpp2/time-plots.html>

Figura 14: ACF Ruido blanco



Fuente: Forecasting: Principles and Practice (Hyndman y Athanasopoulos, 2023). Recuperado de <https://otexts.com/fpp2/time-plots.html>

Teniendo en cuenta que $T = 50$, por lo tanto, los límites están calculados como $\pm 2/\sqrt{50} = \pm 0,28$. En el ACF se puede apreciar que todos los coeficientes de autocorrelación se encuentran dentro estos límites, concluyendo que los datos son ruido blanco.

Modelo ARIMA

El modelo ARIMA o modelo autorregresivo de media móvil integrado, por sobre otros modelos de series temporales, se centra en describir las autocorrelaciones que existen entre los datos (Hyndman y Athanasopoulos, 2023).

- **Estacionariedad:** Una serie de tiempo estacionaria, se presenta cuando sus propiedades no dependen del momento en el que fue registrada la observación. Por lo que, las series de tiempo que presentan tendencias o patrones estacionales son series de tiempo no estacionarias, ya que las tendencias y las distintas estaciones de tiempo pueden afectar la serie de tiempo en varias ocasiones (Hyndman y Athanasopoulos, 2023).

Las series de tiempo estacionarias por lo general son series de ruido blanco, ya que no muestran autocorrelación entre los datos y tampoco presentan patrones predecibles a lo largo del tiempo.

- **Diferenciación:** Una manera de poder cambiar una serie de tiempo no estacionaria en una estacionaria, es aplicar la diferenciación, esto se refiere a calcular la diferencia entre observaciones consecutivas (Hyndman y Athanasopoulos, 2023).

Una de las transformaciones más ocupada para estabilizar la varianza de los datos son los logaritmos. Por otro lado, la diferenciación estabiliza el promedio de la serie de tiempo debido a que es capaz de reducir o remover los distintos cambios que se puedan presentar en la serie de tiempo, tales como las tendencias y patrones estacionales.

- **Modelo Random walk:** La serie diferenciada es el cambio que se presenta entre las observaciones consecutivas de la serie original, esta se denota por:

$$y'_t = y_t - y_{t-1}$$

Ya que es imposible conseguir la diferenciada de la primera observación y'_1 , la serie diferenciada solo tendrá valores T-1. Si la serie diferenciada es un ruido blanco, la fórmula se puede escribir de la siguiente manera, donde ε_t representa el ruido blanco (Hyndman y Athanasopoulos, 2023):

$$y_t - y_{t-1} = \varepsilon_t$$

El **Modelo Random walk** se obtiene luego de despejar y_t :

$$y_t = y_{t-1} + \varepsilon_t$$

Debido a que los movimientos futuros en los datos son impredecibles, las predicciones del modelo son iguales a la última observación, por lo que este tipo de modelos es ampliamente ocupado por datos no estacionarios. Pudiendo concluir que el modelo random walk sustenta los pronósticos naïve (Hyndman y Athanasopoulos, 2023).

$$y_t - y_{t-1} = C + \varepsilon_t \leftrightarrow y_t = C + y_{t-1} + \varepsilon_t$$

El valor de C representa el promedio de los cambios entre observaciones consecutivas (Hyndman y Athanasopoulos, 2023), dependiendo del signo de C la serie se verá derivada positivamente si es positivo o negativamente en el caso contrario.

- **Diferenciación estacional:** Esta diferenciación se realiza para calcular el cambio entre una observación y la observación previa de la misma estación de tiempo (Hyndman y Athanasopoulos, 2023), obteniendo la siguiente fórmula:

$$y'_t = y_t - y_{t-m}$$

Donde m = cantidad de estaciones, también son llamadas "diferencias m -desfazadas", ya que realizamos la resta con una observación luego de m periodos (Hyndman y Athanasopoulos, 2023).

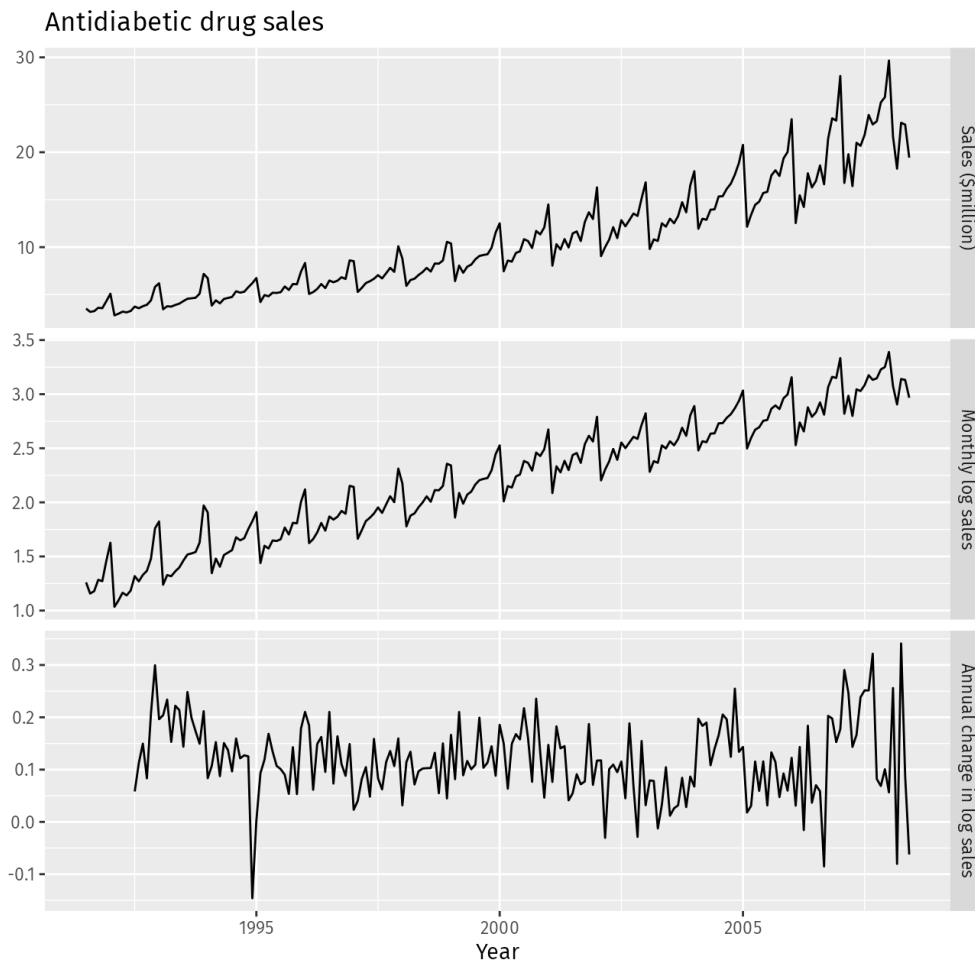
Si luego de realizar una diferenciación estacional a una serie de tiempo, esta parece haberse transformado a ruido blanco, se modelarán los datos originales de la siguiente manera:

$$y_t = y'_t + m + \varepsilon_t$$

Como este tipo de modelo entrega predicciones iguales a la última observación de la estación seleccionada, en otras palabras, este modelo entrega pronósticos naïve (Hyndman y Athanasopoulos, 2023).

En la siguiente figura se muestra cómo las diferenciaciones estacionales y la aplicación de logaritmos a una serie de tiempo, en este caso la venta de medicamentos antidiabéticos, lograr cambiar la forma de la serie a una de ruido (Hyndman y Athanasopoulos, 2023):

Figura 15: Aplicación de logartimos y diferenciación estacional a ventas de medicamentos antidiabéticos



Fuente: Forecasting: Principles and Practice (Hyndman y Athanasopoulos, 2023). Recuperado de <https://otexts.com/fpp2/time-plots.html>

Por lo general, a la diferenciación ordinaria se le llama "primera diferenciación", refiriéndose a la diferenciación en el desfase 1.

Para poder obtener una serie de tiempo que parezca ruido blanco, muchas veces se tendrá que aplicar ambas diferenciaciones, la primera diferenciación y diferenciación estacional, el orden de aplicación no afecta el resultado (Hyndman y Athanasopoulos, 2023).

Tambien existe la segunda diferenciada estacional, por lo que el modelo de la serie doblemente diferenciado se escribe de la siguiente manera:

$$y_t'' = y_t' - y_{t-1}' \leftrightarrow (y_t - y_{t-m}) - (y_{t-1} - y_{t-m-1}) \leftrightarrow y_t - y_{t-1} - y_{t-m} + y_{t-m-1}$$

Si la serie de tiempo, presenta un patrón estacional, es recomendable realizar la diferenciación estacional como primer paso para obtener una serie de tiempo estacionaria, pero si se ocupa la primera diferenciación puede que todavia hayan patrones estacionales en la serie de tiempo, teniendo que aplicar más diferenciaciones para obtener la estacionariedad (Hyndman y Athanasopoulos, 2023).

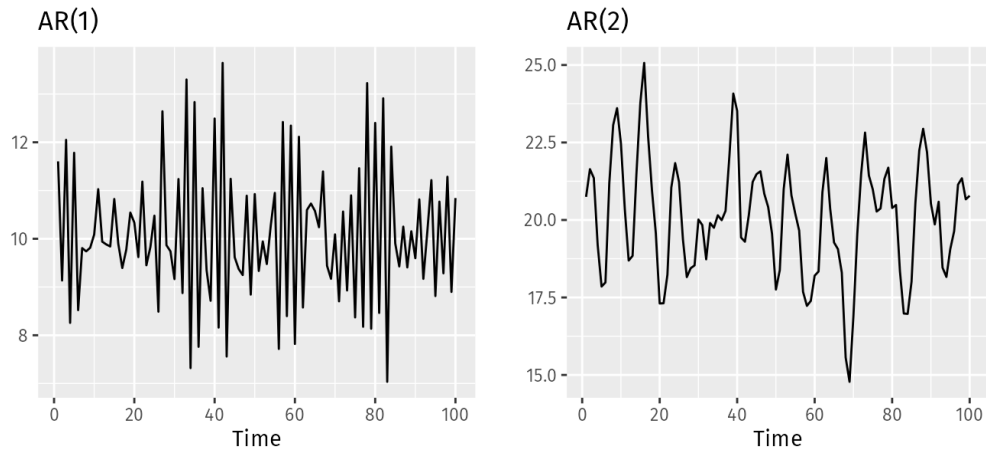
- **Modelo de autorregresión:** Como se menciona anteriormente, los modelos autorregresivos predicen la variable de interes utilizando una combinación lineal de valores pasados de la variable. El término autorregresión indica que se trata de una regresión de la variable contra sí misma.

Por lo que un modelo autorregresivo de orden p , donde ε_t corresponde a ruido blanco, se escribe de la siguiente manera:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \varepsilon_t$$

Este tipo de modelo es llamado **AR(p)**, gracias a la flexibilidad de los modelos autorregresivos, estos pueden ser aplicados en series de tiempo que presenten distintos patrones (Hyndman y Athanasopoulos, 2023). A continuación, se presenta una figura que muestran dos modelos AR, AR(1) y AR(2).

Figura 16: Modelos autorregresivos con diferentes parámetros



Fuente: Forecasting: Principles and Practice (Hyndman y Athanasopoulos, 2023). Recuperado de <https://otexts.com/fpp2/time-plots.html>

El modelo AR(1) tiene como fórmula: $y_t = 18 - 0,8y_{t-1} + \varepsilon_t$ y el modelo AR(2) tiene como fórmula: $y_t = 8 + 1,3y_{t-1} - 0,7y_{t-2} + \varepsilon_t$, en ambos casos ε_t (ruido blanco) tiene una distribución normal, con promedio igual a 0 y varianza igual a 1 (Hyndman y Athanasopoulos, 2023).

- **Modelo de media móvil:** Este tipo de modelo ocupa el error de los pronósticos como si fuera un modelo de regresión, con la diferencia en una regresión se ocupan los valores pasados.

Teniendo ε_t como ruido blanco, el modelo de media móvil de orden q o **MA(q)** se escribe de la siguiente manera (Hyndman y Athanasopoulos, 2023):

$$y_t = c + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \dots + \theta_q\varepsilon_{t-q} + \varepsilon_t$$

A continuación, se presenta una figura que muestran dos modelos MA, MA(1) y MA(2).

Figura 17: Modelos de media móvil con diferentes parámetros



Fuente: Forecasting: Principles and Practice (Hyndman y Athanasopoulos, 2023). Recuperado de <https://otexts.com/fpp2/time-plots.html>

El modelo MA(1) tiene como fórmula: $y_t = 20 + \varepsilon_t + 0,8\varepsilon_{t-1}$ y el modelo MA(2) tiene como fórmula: $y_t = \varepsilon_t - \varepsilon_{t-1} + 0,8\varepsilon_{t-2}$, en ambos casos ε_t (ruido blanco) tiene una distribución normal, con promedio igual a 0 y varianza igual a 1 (Hyndman y Athanasopoulos, 2023).

Un modelo AR(p) estacionario es capaz de ser escrito como un modelo MA(∞), esto se puede demostrar para un modelo AR(1) luego de realizar repetidas sustituciones:

$$\begin{aligned}
 y_t &= \phi_1 y_{t-1} + \varepsilon_t \\
 &= \phi_1(\phi_1 y_{t-2} + \varepsilon_{t-1}) + \varepsilon_t \\
 &= \phi_1^2 y_{t-2} + \phi_1 \varepsilon_{t-1} + \varepsilon_t \\
 &= \phi_1^3 y_{t-3} + \phi_1^2 \varepsilon_{t-2} + \phi_1 \varepsilon_{t-1} + \varepsilon_t
 \end{aligned}$$

Teniendo $-1 < \phi_1 < 1$, el valor de ϕ_1^k irá disminuyendo a medida que k siga creciendo, obteniendo un proceso MA(∞) (Hyndman y Athanasopoulos, 2023),

este tipo de modelo se escribe de la siguiente manera:

$$y_t = \varepsilon_t + \phi_1 \varepsilon_t - 1 + \phi_1^2 \varepsilon_t - 2 + \phi_1^3 \varepsilon_t - 3 + \dots$$

También se puede realizar el proceso inverso, para obtener un modelo $AR(\infty)$ desde un modelo MA, si es que se definen ciertos límites. Este tipo de modelos que se pueden transcribir de MA a $AR(\infty)$ y AR a $MA(\infty)$, son llamados modelos **invertibles** (Hyndman y Athanasopoulos, 2023).

Combinando los modelos de diferenciación, autorregresivo y de media móvil obtenemos el modelo **ARIMA** (Hyndman y Athanasopoulos, 2023), este queda modelado de la siguiente manera:

$$y'_t = c + \phi_1 y'_t - 1 + \dots + \phi_p y'_t - p + \theta_1 \varepsilon_t - 1 + \dots + \theta_q \varepsilon_t - q + \varepsilon_t$$

Teniendo en cuenta que y'_t corresponde a la serie diferenciada, a la derecha de la igualdad se encuentran las variables que permiten realizar las predicciones, entre estas se encuentran valores desfasados de y_t y errores desfasados. El modelo **ARIMA**(p, d, q) tiene 3 variables, donde cada una de estas variables está relacionada a (Hyndman y Athanasopoulos, 2023):

- p = orden de la parte autorregresiva del modelo.
- d = grado de la primera diferenciación.
- q = orden de la parte de media móvil.

De la misma manera que la estacionariedad e invertibilidad se aplican a los modelos de autorregresión y de media móvil, también se aplican para los modelos ARIMA. Este modelo presenta casos especiales, los cuales representan modelos anteriormente mencionados, los cuales son:

- White noise: ARIMA(0,0,0)
- Random walk: ARIMA(0,1,0) sin constante
- Random walk con desfase: ARIMA(0,1,0) con constante
- Autorregresión: ARIMA(p ,0,0)
- Media móvil: ARIMA(0,0, q)

Teniendo en cuenta todo lo anteriormente mencionado, se ocuparan métodos que ayudaran a medir la certeza de los modelos aplicados, entre los cuales se encuentra el método Akaike's Information Criterion (AIC).

El valor AIC se obtiene luego de aplicar el método **Akaike's Information Criterion (AIC)** al modelo de series temporales aplicado, este valor estima el error de la predicción realizada (Hyndman y Athanasopoulos, 2023). Este método se encuentra definido por:

$$AIC = T \log\left(\frac{SSE}{T}\right) + 2(k + 2)$$

Donde T corresponde al número de observaciones que se ocuparon para realizar la estimación, la variable k corresponde al número de predictores presentes en el modelo y el valor SSE corresponde a la suma de errores cuadrados, definido de la siguiente manera (Hyndman y Athanasopoulos, 2023):

$$SSE = \sum_{t=1}^T e_t^2$$

Por lo que el modelo que logre minimizar al maximo el valor AIC, se puede considerar como el mejor modelo para realizar predicciones (Hyndman y Athanasopoulos,

2023).

Modelo SARIMA

El modelo SARIMA es una extensión del ARIMA, diseñado específicamente para series temporales que presentan estacionalidad. Mientras que ARIMA es eficaz para modelar series no estacionarias sin componentes estacionales, SARIMA incorpora parámetros adicionales para capturar y predecir patrones que se repiten a intervalos regulares en el tiempo.

La estacionalidad en una serie temporal refleja un patrón regular que se repite cada S períodos. Aquí, S indica el número de períodos requeridos para que el patrón complete un ciclo y comience a repetirse (Morales Oñate, 2022).

- **Diferenciación no estacional:** Si en los datos se detecta una tendencia, es probable que sea necesario aplicar una diferenciación no estacional. A menudo, al realizar una primera diferenciación no estacional, se logra eliminar la tendencia de la serie temporal. Para este propósito, utilizamos la fórmula:

$$(1 - B)x_t = x_t - x_{t-1}$$

- **Diferenciación para tendencia:** Cuando tanto la tendencia como la estacionalidad están presentes en una serie temporal, es posible que necesitemos aplicar tanto una diferenciación no estacional como una estacional.

Por lo tanto, es esencial examinar tanto el ACF (Función de Autocorrelación) como el PACF (Función de Autocorrelación Parcial) de la expresión:

$$(1 - B^{12})(1 - B)x_t = (x_t - x_{t-1}) - (x_{t-12} - x_{t-13})$$

Es importante destacar que eliminar la tendencia no necesariamente implica que se haya eliminado toda la dependencia en la serie. Mientras que la tendencia (o

el componente de la media) puede haber sido eliminada, aún puede persistir un comportamiento periódico. Al aplicar la diferenciación, descomponemos la dependencia en eventos recientes y eventos a largo plazo (Morales Oñate, 2022).

- **Diferenciación para estacionalidad:** Cuando se tienen datos con componentes estacionales, es posible que los componentes no estacionales a corto plazo sigan influyendo en el modelo. Por ejemplo, en el caso de las ventas de componentes electrónicos, las ventas de los últimos uno o dos meses, junto con las ventas del mismo mes del año anterior, pueden ser relevantes para predecir las ventas del mes actual.

Por lo tanto, es crucial examinar el comportamiento del ACF (Función de Autocorrelación) y PACF (Función de Autocorrelación Parcial) en los primeros rezagos para determinar qué términos no estacionales pueden ser significativos en el modelo.

Siendo expresado de manera abreviada:

$$(p, d, q) \times (P, D, Q, S)$$

- P : Es el número de términos autorregresivos estacionales.
- D : Es el grado de diferenciación estacional.
- Q : Es el número de términos de promedio móvil estacional.
- S : Es el número de periodos en una temporada (por ejemplo, 12 para datos mensuales si se observa una periodicidad anual).

3.3.6. Redes LSTM Long Short-Term Memory

¿Qué es una red neuronal?

Una red neuronal es un modelo computacional inspirado en la estructura y funcionamiento del cerebro humano, utilizado en el campo de la inteligencia

artificial. Forma parte del aprendizaje profundo dentro del machine learning y está compuesta por unidades, llamadas neuronas, organizadas en capas. Estas neuronas están interconectadas y procesan información adaptándose y aprendiendo de los datos. Las redes neuronales se emplean para abordar tareas complejas, como el reconocimiento de imágenes o la traducción automática, mejorando su rendimiento a medida que se les expone a más datos (Amazon, 2023).

¿Qué son las redes Long Short-Term Memory?

La LSTM es un tipo de red neuronal recurrente (RNN) diseñada para resolver el problema del gradiente evanescente mediante la introducción de una célula de memoria que puede almacenar información durante periodos de tiempo más largos (Filzinger, 2023).

Arquitectura de una Long Short-Term Memory

- **Puerta de entrada:** La puerta de entrada en una LSTM utiliza una función de activación sigmoidea para decidir qué valores pasar a la célula de memoria. La función sigmoidea genera valores entre 0 y 1, lo que permite que la puerta de entrada *decida* en qué grado un valor debe ser almacenado en la célula de memoria. Un valor cercano a 1 indica que casi toda la información debe ser conservada, mientras que un valor cercano a 0 sugiere que la información debe ser mayormente descartada (Filzinger, 2023).
- **Puerta del Olvido:** Controla qué información de la célula de memoria del paso temporal anterior debe ser conservada y cuál debe ser descartada. Utiliza una función de activación sigmoidea para determinar esto. Un valor cercano a 1 de la sigmoidea indica que la información debe ser conservada, mientras que un valor cercano a 0 indica que debe ser olvidada. (Filzinger, 2023).
- **Puerta de Salida:** Controla qué información de la célula de memoria debe ser enviada como salida. Utiliza una función de activación sigmoidea para decidir qué partes de la memoria se deben considerar y luego aplica una función de tangente hiperbólica para escalar los valores entre -1 y 1, determinando así la salida final de la célula LSTM (Filzinger, 2023).

- **Célula de memoria:** Es el componente central de la arquitectura LSTM. Esta célula almacena información a través del tiempo, pudiendo olvidar selectivamente información no relevante y añadir nueva información a su estado interno.

En cada paso temporal, el modelo LSTM recibe un vector de entrada junto con un vector de estado oculto proveniente del paso temporal anterior. A partir de estos, la puerta de entrada decide qué nueva información se debe almacenar en la célula de memoria, mientras que la puerta de olvido decide qué información previamente almacenada debe descartarse.

Después, se genera un estado candidato utilizando la función de activación tangente hiperbólica. Este estado candidato se combina con el estado actual de la célula de memoria mediante una operación de suma elemento a elemento, actualizando así la información almacenada.

Finalmente, la puerta de salida determina cuál de esta información actualizada se emitirá como salida de la LSTM. Este output, junto con el estado actualizado de la célula de memoria, se pasa al siguiente paso temporal (Filzinger, 2023).

3.3.7. Metodología del proyecto

Para llevar a cabo el desarrollo del proyecto, se definieron cuatro fases que corresponden a la totalidad del proyecto, las cuales corresponden a:

Fase 1: Planteamiento y planificación

Para la primera fase del proyecto, se llevará a cabo una planificación de la manera en la que será abordada la problemática, para desarrollar un anteproyecto que será utilizado para evaluar y planificar las actividades correspondientes al desarrollo del proyecto. Entre ellas se encuentran:

- Planteamiento del proyecto y sus objetivos.
- Definición de alcances y limitaciones.

- Creación de un cronograma de actividades.

Fase 2: Investigación

Para la segunda fase, se realizará una investigación de herramientas y recursos necesarios para llevar a cabo un diseño de la solución para la problemática del proyecto planteado, sumado a un análisis de las bases de datos brindadas por la empresa AFP Capital. Una vez realizado lo anterior, se llevará a cabo una propuesta de diseño para la problemática, siendo entregada y analizada por la empresa, con la finalidad de pasar a desarrollo. Algunas de las actividades de esta fase corresponden a:

- Investigación del problema.
- Toma de requerimientos.
- Investigación de tecnologías de análisis de datos.

Fase 3: Modelamiento y desarrollo

Para la tercera fase, se llevará a cabo el diseño y desarrollo del sistema propuesto, además de realizar pruebas para verificar el correcto funcionamiento. Algunas de las actividades de esta fase corresponden a:

- Modelado del sistema ETL.
- Modelado de la API.
- Implementación del modelo propuesto.
- Pruebas y validaciones.
- Correcciones de errores.

Fase 4: Conclusiones y recomendaciones

Para la última fase, se dará fin al desarrollo del proyecto, elaborando un manual de usuario el cual indicaría algunas funcionalidades del sistema. Algunas de las actividades de esta fase corresponden a:

- Desarrollo de manual de usuario.
- Redacción de conclusiones y recomendaciones.
- Cierre del proyecto.

3.3.8. Metodología del sistema

CRISP-DM

La metodología CRISP-DM (Cross-Industry Standard Process for Data Mining) es un proceso estándar utilizado para realizar proyectos de minería de datos. La metodología CRISP-DM se divide en seis fases distintas que se describen a continuación (Eric, 2023):

1. **Comprensión del problema:** En esta fase se define el problema a resolver y se establecen los objetivos del proyecto. También se recopilan los datos necesarios para el proyecto.
2. **Comprensión de los datos:** En esta fase se realiza una exploración de los datos para comprender su calidad, estructura y relevancia para el problema en cuestión.
3. **Preparación de los datos:** En esta fase se limpian y procesan los datos para que puedan ser utilizados en la etapa de modelado.
4. **Modelado:** En esta fase se aplican técnicas de modelado para desarrollar un modelo predictivo. Se prueban diferentes modelos y se selecciona el que mejor se ajuste a los datos.

5. **Evaluación:** En esta fase se evalúa el modelo desarrollado en la fase anterior. Se verifica que el modelo funcione correctamente y se ajuste adecuadamente a los datos.
6. **Implementación:** En esta fase se implementa el modelo desarrollado en la fase de modelado en un entorno de producción. También se establecen planes para monitorear el rendimiento del modelo y actualizarlo según sea necesario.

Las fases de la metodología CRISP-DM son iterativas, lo que significa que es posible volver a una fase anterior si es necesario.

OSEMN

La metodología OSEMN (acrónimo de las palabras en inglés: Obtain, Scrub, Explore, Model, Interpret) es un proceso utilizado en la minería de datos y el análisis de datos para trabajar con grandes conjuntos de datos de manera efectiva (Eric, 2023).

1. **Obtener (Obtain):** En esta etapa, se recopilan los datos necesarios para el análisis. Los datos pueden provenir de diferentes fuentes, como bases de datos, archivos en línea o registros de sensores. La calidad y la cantidad de los datos obtenidos son cruciales para el éxito del análisis.
2. **Limpieza (Scrub):** Una vez que se han obtenido los datos, es necesario realizar una limpieza para eliminar datos innecesarios o incorrectos. Esta etapa puede implicar la eliminación de duplicados, la corrección de errores y la eliminación de valores atípicos. El objetivo de esta etapa es obtener datos limpios y coherentes para el análisis.
3. **Exploración (Explore):** En esta etapa, se utilizan técnicas de visualización y estadísticas para explorar los datos y obtener información sobre ellos. Se pueden identificar patrones, tendencias y relaciones entre diferentes variables. El objetivo es obtener una comprensión más profunda de los datos y de cómo se relacionan entre sí.

4. **Modelado (Model):** En esta etapa, se utilizan técnicas de modelado estadístico o de aprendizaje automático para crear modelos que puedan predecir resultados futuros o identificar patrones en los datos. El objetivo es utilizar los datos para crear un modelo que pueda utilizarse para tomar decisiones informadas.
5. **Interpretación (Interpret):** En esta etapa, se interpretan los resultados obtenidos en la etapa de modelado. Los resultados pueden ser utilizados para tomar decisiones o para generar nuevas hipótesis que puedan ser exploradas en futuros análisis.

Se propone el uso de la metodología OSEMN, ya que se enfoca en el análisis de datos y la creación de modelos predictivos. OSEMN también es una metodología más flexible que CRISP-DM, lo que puede ser útil en un proyecto de SCRUM donde se busca una mayor adaptabilidad.

Por otro lado, también se propone el uso de la metodología CRISP-DM, ya que el proyecto incluye una etapa de exploración y análisis de datos, seguida por una fase de construcción de modelos. CRISP-DM se enfoca en el proceso completo de minería de datos, desde la comprensión del problema hasta la implementación del modelo, lo que puede servir para realizar un trabajo más estructurado.

Ya que este proyecto se encuentra bajo el marco de trabajo SCRUM, ambas metodologías pueden ser utilizadas de manera complementaria, utilizando OSEMN para las fases de creación de modelos y CRISP-DM para la etapa de exploración y análisis de datos.

CAPÍTULO 4: PROCESO ETL

El proceso ETL (Extract, Transform, Load) representa el núcleo de la gestión de datos en numerosas organizaciones, desempeñando un papel esencial en la recopilación, transformación y carga de información crítica. En este capítulo del informe, exploraremos en detalle el funcionamiento de un proceso ETL, así como su diseño estratégico. Analizaremos cómo esta metodología se convierte en una piedra angular para la toma de decisiones basadas en datos y la mejora de la eficiencia operativa.

4.1. Diseño Proceso ETL

El diseño de un proceso ETL (Extracción, Transformación y Carga) implica seguir distintos pasos para asegurar que este proceso y el flujo de datos sean eficientes, precisos y cumplan con los requisitos del proyecto (Kimball y Caserta, 2004). Los pasos que se acordaron seguir son los siguientes:

- Requisitos ETL
- Identificación fuente de datos
- Diseño modelo de datos objetivo
- Planificación de las transformaciones
- Selección herramientas
- Construcción y prueba proceso ETL
- Monitoreo proceso ETL

4.1.1. Requisitos ETL

En esta etapa se definen los requisitos del proyecto, las fuentes de datos, los objetivos comerciales y del proceso ETL, las necesidades de análisis y los plazos para realizar el proceso. Estableciendo una base sólida para el diseño y buen funcionamiento del proceso ETL.

- **Fuente de datos:** La fuente de datos corresponde a un archivo .CSV o de texto plano que contienen información de la navegación web de los clientes en forma de web logs. Estos archivos los proporciona la empresa luego de realizar un proceso de recopilación, ordenamiento y anonimización de los datos.
- **Objetivos comerciales:** Analizar el comportamiento de los clientes y sus preferencias de uso en un período determinado, para poder predecir navegaciones futuras, y a partir de esto proporcionar atenciones personalizadas.
- **Objetivos proceso ETL:** Realizar las transformaciones necesarias para asegurar que el flujo de datos sea eficiente y preciso, a través de la limpieza de los datos, la normalización, la agregación, el filtrado, el enriquecimiento de datos, así como los cálculos y derivaciones necesarios.
- **Necesidades de análisis:** Llevar a cabo un análisis exploratorio de los datos proporcionados con el fin de adquirir un profundo conocimiento de los mismos, lo que permitirá desarrollar un modelo predictivo adecuado basado en los datos procesados.

4.1.2. Identificación fuente de datos

En esta etapa se determinan las fuentes de datos a ser usadas para el proyecto, incluyendo bases de datos, archivos .CSV y APIs. Esto además comprende la definición de la estructura, el formato y ubicación de cada fuente de datos dentro del proyecto.

La fuente de datos corresponde a un archivo .CSV que contiene información de la navegación web de los afiliados en forma de web logs. Los web logs registrados vienen con 4 campos, especificados a continuación:

- **rut:** Este atributo representa un identificador único por afiliado. Debido a las políticas de confidencialidad de la empresa, antes de entregar el archivo, se realiza un proceso de anonimización de este campo. En el archivo, este dato se visualiza como una cadena de caracteres que incluye letras (minúsculas y mayúsculas), números y otros caracteres.
- **fecha evento:** Representa el momento exacto en que el usuario realiza la interacción en el canal web. Esto incluye día, mes, año, hora, minuto y segundo.
- **metodo:** Este atributo representa la interacción realizada por el usuario en el canal web.
- **canal:** Corresponde al canal web donde el afiliado realiza su interacción en la plataforma.

Para almacenar la fuente de datos se utiliza una estructura de carpetas, siendo las siguientes:

- **Input:** Dentro de esta carpeta se encontrarán los archivos.
- **Intermediate:** Aquí se almacenará el archivo con la información preprocesada.
- **Output:** Dentro de esta última carpeta se almacenará la información ya procesada y lista para ser usada.

4.1.3. Diseño del modelo de datos objetivo

En esta etapa, se lleva a cabo el diseño del modelo de datos objetivo para poder trabajar con él en el desarrollo de un modelo de aprendizaje automático. Para esto se considera lo siguiente:

- **Tabla(s) necesaria(s):** En este escenario, se trabaja exclusivamente con una tabla que contiene los registros de navegación, como se ha comentado en secciones anteriores.

- **Definición de columnas o atributos:** El DataFrame contendrá nueve columnas que describirán el registro de actividad del usuario en la plataforma. Dichas columnas son: rut, año, mes, día, hora, minuto, segundo, método y canal. Cada columna proporcionará información valiosa sobre el comportamiento del usuario durante la interacción.
- **Definición de tipos de datos:** Los datos que se utilizarán serán numéricos, ya que la mayoría de los modelos de aprendizaje automático trabajan eficazmente con este tipo de datos para realizar predicciones.
- **Gestión de valores faltantes:** En este contexto, las filas que contengan valores faltantes se eliminarán del conjunto de datos. La decisión de no utilizar otras técnicas para gestionar los valores faltantes, como reemplazarlos con el promedio, se basa en la naturaleza del trabajo. El equipo concluyó que reemplazar los valores faltantes podría alterar la representación del comportamiento real de un usuario, por lo que se prefiere mantener la integridad de los registros de actividad.
- **Gestión de datos categóricos:** Dado que la mayoría de los datos son categóricos, es fundamental abordarlos de manera adecuada. Se aplicará una “codificación de etiqueta”, que asigna un valor numérico único a cada categoría en función de alguna lógica o preferencia. En este caso, se asignarán valores numéricos a las categorías en orden ascendente a medida que aparezcan en el conjunto de datos.
- **Creación del nuevo dataframe:** La creación del nuevo DataFrame se llevará a cabo en Google Colab utilizando Python, específicamente la biblioteca Pandas. Las transformaciones realizadas se profundizarán en la siguiente sección para detallar cómo se optimizan los datos para análisis y modelado subsiguientes.

4.1.4. Planificación de las transformaciones

Durante esta etapa, se planifican las transformaciones necesarias para establecer una base sólida que respalde el proyecto. Estas incluyen pasos para limpiar, filtrar,

combinar y enriquecer los datos.

La planificación de estas transformaciones es esencial para asegurar la calidad y la integridad de los datos utilizados en el proyecto. Se identifican tareas específicas alineadas con los objetivos y requisitos del proyecto y las necesidades comerciales.

Algunas transformaciones comunes incluyen (Kimball y Caserta, 2004):

- **Limpieza de datos:** Se llevan a cabo tareas de limpieza para corregir errores, eliminar valores duplicados o inconsistentes y garantizar la coherencia de los datos. En este caso, la limpieza de datos se utilizará para estandarizar el formato y eliminar valores que puedan afectar de manera negativa el aprendizaje del modelo. Inicialmente, se eliminarán todas las filas que contengan valores nulos. La columna fecha se estandarizará con un formato determinado (UTC) YYYY-MM-DDTHH:MM:SS.sssZ. Por último, se cambian los nombres de los canales mal registrados, esto como consecuencia de un error que se detectó en el registro de actividades, específicamente en el campo 'canal'.
- **Filtrado de datos:** Se aplican filtros para seleccionar y extraer los datos relevantes para el proyecto, descartando aquellos que no cumplen con ciertos criterios o condiciones específicas. Se excluyen registros sin métodos asociados o con métodos inconsistentes. Además, se excluyen los registros que contienen el campo de RUT no anonimizado, cumpliendo con la política de privacidad de la empresa. Este enfoque permite centrarse en la información más relevante y útil.

Es importante destacar que la planificación de las transformaciones considera el orden y la secuencia adecuada de ejecución, así como la documentación de cada paso y los criterios de validación y verificación para garantizar la calidad de los datos transformados. Por lo tanto, se ha diseñado un plan detallado con las transformaciones antes mencionadas que se presentará en la siguiente sección.

4.1.5. Selección de herramientas

En esta etapa, se realiza la selección de herramientas de software que se ajusten a las necesidades y requisitos del proyecto para llevar a cabo el proceso ETL de manera eficiente. Se evalúan diferentes opciones disponibles en función de su capacidad, compatibilidad y facilidad de uso, para garantizar una elección adecuada, es por esto que se seleccionaron las siguientes herramientas:

- **Colab:** También conocido como “Colaboratory”, permite programar y ejecutar Python en el navegador con las siguientes ventajas (*Google Colab*, 2023):
 - No requiere configuración
 - Acceso a GPUs sin coste adicional
 - Permite compartir contenido fácilmente

Una de las desventajas de usar Colab es que tiene una cantidad de memoria limitada, es decir, si se utiliza el total de memoria disponible no se podrá seguir ejecutando código.

- **Visual Studio Code:** Es un editor de código fuente desarrollado por Microsoft. Es conocido por su enfoque en la simplicidad, la personalización y la eficiencia. Este editor se utilizará en caso de no poder seguir utilizando Colab.
- **Python:** Es un lenguaje de programación interpretado, de alto nivel y de propósito general, conocido por su sintaxis clara y legible. Es utilizado en una amplia gama de aplicaciones, desde desarrollo web hasta ciencia de datos y aprendizaje automático. Python se destaca por su facilidad de aprendizaje y su amplia biblioteca estándar, que ofrece numerosas funcionalidades predefinidas para diversas tareas (van Rossum y Drake, 2003).
 - **Numpy:** Es una biblioteca fundamental para la computación científica en Python. Proporciona estructuras de datos eficientes y funciones para

realizar operaciones numéricas y de manipulación de arrays (van der Walt, Colbert, y Varoquaux, 2011).

- **Pandas:** Es una biblioteca poderosa para el análisis de datos basada en Numpy, que proporciona estructuras de datos flexibles y eficientes, como DataFrames, y un conjunto completo de funciones para la manipulación y transformación de datos (Mckinney, 2011).
- **Dask:** Es una biblioteca de paralelización flexible que permite escalar el procesamiento de datos en Python. Proporciona estructuras de datos paralelas y operaciones distribuidas que facilitan el procesamiento de grandes volúmenes de datos (*dask: Scale the Python tools you love*, 2023).

4.1.6. Construcción y prueba proceso ETL

En esta etapa, se lleva a cabo la implementación del diseño del proceso ETL previamente definido, utilizando las herramientas seleccionadas. Se desarrollan los flujos de extracción, transformación y carga de los datos según lo establecido en el diseño.

Una vez implementado, se procede a realizar pruebas exhaustivas para garantizar el correcto funcionamiento del proceso. Estas pruebas incluyen la verificación de la extracción de datos de las fuentes, la correcta aplicación de las transformaciones definidas y la carga exitosa de los datos en el destino final.

El objetivo de las pruebas es asegurar que el proceso ETL cumpla con los requisitos establecidos y que los resultados obtenidos sean los esperados. Esto implica validar la integridad y coherencia de los datos transformados, así como verificar el rendimiento y la escalabilidad del proceso.

En caso de encontrar inconvenientes o desviaciones durante las pruebas, se realizan los ajustes necesarios en el diseño o en la configuración de las herramientas utilizadas. Es fundamental realizar iteraciones y pruebas adicionales hasta obtener resultados consistentes y satisfactorios.

Como primer paso en la construcción del proceso ETL, se importan las bibliotecas pandas y numpy para trabajar con el dataset. Luego se le da nombre a las columnas, ya que el contenido del archivo csv solo trae los datos. Una vez nombradas las columnas se ordenan para un entendimiento más fácil del conjunto de datos, quedando de la siguiente manera respectivamente: rut, fecha, metodo, canal.

Ahora se comprueba la existencia de valores nulos en el dataframe, resultando lo siguiente:

Figura 18: Valores nulos en el conjunto de datos.

```
rut          0
fecha        0
metodo       1
canal    36852
dtype: int64
```

Fuente: Elaboración propia.

Como se puede ver en la imagen anterior solamente los campos 'metodo' y 'canal' poseen valores nulos, de hecho, el primero solo cuenta con un valor nulo. La columna 'canal' cuenta con 36,852 valores nulos. Luego de eliminar los valores nulos se pasa a eliminar las filas que contengan métodos inconsistentes o irrelevantes, que son:

- reCaptcha()
- clientes/surveys
- clientes/cuentas/validar-cuenta-corriente
- clientes/cuentas/cuentas-bancarias
- clientes/cuentas/cuentas-bancarias?rut=*****
- clientes/cuentas/cuentas-bancarias?rut=*****

- `clientes/cuentas/cuentas-bancarias?rut=*****`
- `clientes/cuentas/cuentas-bancarias?rut=*****`
- `clientes/cuentas/cuentas-bancarias?rut=*****`
- `clientes/cuentas/cuentas-bancarias?rut=*****`
- `clientes/cuentas/cuentas-bancarias?rut=*****`
- `clientes/cuentas/cuentas-bancarias?rut=*****`
- `clientes/cuentas/APV/solicitud/giros?rut=*****&account=APV`
- `clientes/cuentas/cuentas-bancarias?rut=*****`

Es importante mencionar que el método `reCaptcha()` de Google valida la autenticidad de usuarios frente a bots, pero su utilidad se limita a la seguridad en línea. No ofrece información relevante para predecir el comportamiento de los usuarios en la plataforma, siendo necesario recurrir a los otros datos para comprender mejor sus acciones y patrones de uso.

Ahora se continua con la estandarización del formato de la fecha, quedando con el formato UTC (`YYYY-MM-DDTHH:MM:SS.sssZ`). La estandarización del formato de la fecha es un paso esencial para garantizar la coherencia y la compatibilidad de los datos. En este caso, se ha optado por este formato debido a una convención global reconocida que asegura la uniformidad y comprensión en diferentes contextos y aplicaciones.

Después de ajustar el formato de la fecha, se procede a corregir la denominación incorrecta de ciertos métodos registrados. Estos métodos son: `WEB-Actualizacion-datos` y `WEB-Contacto-tramite-pensión`. Ambos valores registrados en el campo `'canal'` presentaron errores de registro. Este problema fue identificado tras el análisis del conjunto de datos, y se informó a la empresa para que corrigiera este error. En la

plataforma, únicamente existen dos canales de acceso: 'WEB' y 'PWA'. El primero corresponde a la página web de la plataforma, mientras que el segundo se refiere a la aplicación. En consecuencia, se procede a modificar el nombre de los canales incorrectamente registrados (pertenecientes al canal 'WEB', según su nombre y confirmado con la empresa), para que se ajusten al nombre correcto de canal 'WEB'.

Al finalizar el proceso de ETL se puede ver la diferencia entre el dataset inicial y el dataset final. Primero se presenta una descripción inicial del conjunto de datos, esto antes de construir y ejecutar el proceso ETL.

Figura 19: Descripción del dataset antes del proceso ETL.

	rut	fecha	metodo	canal
count	2332471	2332471	2332470	2295619
unique	35544	406475	100	4
top	MTU0NjM5Nzlx	2023-09-13T17:32:09.329	reCaptcha()	PWA
freq	339344	56	408224	1439189

Fuente: Elaboración propia.

Ahora se presenta una descripción del conjunto luego del proceso ETL.

Figura 20: Descripción del dataset después del proceso ETL.

	rut	fecha	metodo	canal
count	1885948	21149	1885948	1885948
unique	35524	21149	85	2
top	MTU0NjM5Nzlx	2023-09-13 02:47:13.827000	getAccountWithdrawal()	PWA
freq	320690	1	259733	1029538

Fuente: Elaboración propia.

Después de aplicar el proceso ETL a los datos iniciales, se observan cambios

notables en varias dimensiones del conjunto de datos. En primer lugar, el tamaño del conjunto de datos se redujo significativamente de 2,332,471 registros a 1,885,948, lo que sugiere una eficaz limpieza y eliminación de datos redundantes o inconsistentes. A nivel de valores únicos, la columna 'rut' mantuvo su estabilidad con 35,544 valores únicos, mientras que 'fecha' se estandarizó de 406,475 a 21,149 valores únicos, indicando una consolidación y uniformidad en el registro de fechas. Además, las columnas 'método' y 'canal' experimentaron reducciones significativas en la cantidad de valores únicos, pasando de 100 a 85 y de 4 a 2 respectivamente, lo que indica una simplificación y una corrección de errores en la categorización. Los valores más frecuentes ('top') variaron en todas las columnas, destacando cambios notables en 'fecha', 'método' y 'canal', lo que indica una reestructuración significativa de los datos y posiblemente una mejora en la precisión y relevancia de la información.

Una vez completadas las etapas de limpieza y transformación mencionadas previamente, estamos listos para adentrarnos en el análisis de los datos. Este análisis es fundamental para extraer la información necesaria que orientará el desarrollo de un modelo de aprendizaje automático.

4.1.7. Monitoreo proceso ETL

Se establece un sistema de monitoreo para supervisar de manera continua el rendimiento del proceso ETL, permitiendo identificar y abordar posibles problemas a tiempo y garantizar la calidad de los datos. En esta etapa, también se realiza el mantenimiento del proceso, lo cual implica actualizaciones de las transformaciones, resolución de problemas y optimización del proceso.

El sistema de monitoreo juega un papel fundamental en la detección temprana de cualquier anomalía o interrupción en el flujo de datos. Mediante la implementación de métricas y alertas, se pueden supervisar aspectos clave como el tiempo de ejecución, el uso de recursos, los volúmenes de datos y la integridad de los resultados.

Además, el mantenimiento del proceso ETL implica la capacidad de adaptación a medida que evolucionan las necesidades del proyecto. Esto puede implicar la actualización de las transformaciones para reflejar cambios en las fuentes de datos

o requerimientos del negocio, así como la solución de problemas que puedan surgir durante la ejecución del proceso.

Asimismo, se busca optimizar el proceso ETL a través de la identificación de posibles cuellos de botella o ineficiencias. Esto puede implicar ajustes en el diseño de las transformaciones, mejoras en la selección de herramientas o la optimización de los recursos utilizados.

CAPÍTULO 5: EXPLORATORY DATA ANALYSIS (EDA)

El análisis exploratorio de datos (EDA) es una etapa crucial en el proyecto de predicción del comportamiento en un entorno web. En este capítulo, adentraremos en el proceso de EDA, que nos permitirá desvelar patrones, tendencias y relaciones ocultas en los datos recopilados. A través de este análisis, obtendremos una visión profunda y enriquecedora que sentará las bases para un mejor entendimiento del comportamiento del usuario en el entorno web y facilitará la toma de decisiones estratégicas.

5.1. Introducción al EDA

El análisis exploratorio de datos (EDA, por sus siglas en inglés, Exploratory Data Analysis) es una fase fundamental en la investigación y comprensión de un conjunto de datos. Como su nombre lo indica, el EDA tiene como objetivo explorar y examinar los datos de manera detallada, utilizando resúmenes numéricos y visuales, con el fin de descubrir patrones, tendencias y características no anticipadas. Es considerado uno de los primeros pasos en el proceso de análisis, ya que proporciona una visión general de los datos antes de realizar un análisis más profundo (Ruiz, 2022).

El enfoque principal del EDA radica en el uso de herramientas y técnicas visuales y gráficas para revelar información clave sobre los datos en estudio (Parra, 2002). Estas técnicas incluyen el diagrama de tallo y hoja, el diagrama de caja y bigotes, y el diagrama de dispersión, entre otros. Al aplicar estas técnicas de análisis gráfico, podemos obtener una comprensión más profunda de la distribución y estructura de los datos, así como identificar relaciones entre las variables de interés. Además, el EDA nos brinda la capacidad de detectar posibles errores o puntos extremos, como anomalías, que podrían afectar la calidad de los resultados del análisis.

Los beneficios clave del análisis exploratorio de datos son los siguientes (Ruiz, 2022):

- **Conocer la distribución y estructura de los datos:** El EDA nos permite examinar la distribución de las variables y comprender cómo se organizan y dispersan los datos en el conjunto. Esto es fundamental para seleccionar las técnicas adecuadas de análisis estadístico y modelado.
- **Estudiar la relación entre variables:** Mediante el análisis de correlación y la visualización de patrones en los diagramas de dispersión, podemos explorar las relaciones entre las variables y comprender cómo interactúan entre sí. Esto nos brinda información valiosa para identificar posibles dependencias y tendencias en los datos.
- **Encontrar posibles errores y anomalías:** El EDA nos ayuda a identificar valores atípicos, datos faltantes u otros errores en los datos. Estas anomalías pueden tener un impacto significativo en los resultados del análisis, por lo que es importante detectarlas y tratarlas de manera adecuada.

5.2. Recopilación de datos

Los datos recopilados de los registros de navegación de los afiliados de AFP Capital constituyen una valiosa fuente de información para comprender el comportamiento y las preferencias de los usuarios en la plataforma web. Estos registros nos permiten analizar cómo interactúan los afiliados con los diferentes canales y métodos disponibles, así como realizar un seguimiento detallado de las fechas y horarios en que se llevan a cabo estas interacciones.

El dataset entregado para este proyecto cuenta con 2,332,471 registros de navegación de usuarios pero luego del ETL queda en 1,885,948, lo cual proporciona una cantidad significativa de información para su análisis. Antes de utilizar estos datos, se realizó un proceso de anonimización para proteger la privacidad de los usuarios, específicamente modificando el campo del rut para no mostrar el dato original. De esta manera, se garantiza que los registros sean tratados de forma confidencial y

segura.

Los cuatro campos que conforman el conjunto de datos son el rut, la fecha del evento, el método y el canal. El rut, que ha sido modificado, actúa como un identificador único para cada usuario y permite realizar análisis individuales sin revelar su identidad. La fecha del evento registra el momento exacto en que se llevó a cabo cada navegación, lo cual es crucial para identificar patrones y tendencias a lo largo del tiempo. El campo del método describe la interacción específica realizada por el usuario en el canal correspondiente, proporcionando información detallada sobre las acciones que realizan. Por último, el campo del canal indica el sitio o ambiente particular en el cual tuvo lugar cada interacción, lo que puede ser útil para comprender las preferencias de los usuarios en relación con los diferentes entornos disponibles.

Con respecto al preprocesamiento de datos realizado hasta la fecha, se ha seguido el proceso ETL (Extracción, Transformación y Carga) que se describe en detalle en el capítulo anterior. Este proceso implica extraer los datos de las fuentes de origen, transformarlos en un formato adecuado y cargarlos en un sistema de almacenamiento para su posterior análisis. Se utilizaron diversas herramientas para llevar a cabo estas tareas, asegurando la calidad y coherencia de los datos procesados.

Es crucial subrayar que los datos recopilados, aunque constituyen una fuente de información considerable sobre el comportamiento de los usuarios en la plataforma web, presentan ciertas limitaciones que deben ser tenidas en cuenta. La muestra de datos podría no representar de manera equitativa todos los segmentos de ingresos de los afiliados, lo que podría influir en los patrones de navegación observados. Es posible que ciertas tendencias o preferencias sean más comunes en los registros debido a una sobrerrepresentación de determinados grupos económicos en los datos disponibles. Por ende, al interpretar los resultados y al intentar generalizar las conclusiones obtenidas, es esencial considerar que los comportamientos detectados pueden no ser extrapolables a la totalidad de la base de usuarios. Se debe proceder con precaución al hacer inferencias, y sería beneficioso buscar estrategias para incorporar y analizar datos de otros segmentos para obtener una comprensión más holística y representativa del conjunto de afiliados.

5.3. Descripción de los datos

Resumen de las principales características de las variables relevantes en los registros de logs de navegación. Análisis de la distribución de los datos, incluyendo medidas de centralidad y dispersión. Usando la librería Pandas de Python, podemos obtener información importante sobre los conjuntos de datos. Estos valores nos proporcionarán un mayor entendimiento de la composición y estructura de los registros de navegación.

Podemos obtener información respecto a la cantidad de valores faltantes en el dataframe que se está utilizando, los cuales suman 36,853 en total. De estos datos faltantes, 36,853 corresponden principalmente al campo 'canal', con solo un valor faltante en el campo 'método'. Como se vio en el capítulo anterior, estos valores fueron eliminados del dataframe.

Haciendo uso de la misma librería, podemos determinar los tipos de datos que contiene cada campo:

- **rut:** Contiene datos de tipo string.
- **fecha:** Contiene datos de tipo date.
- **metodo:** Contiene datos de tipo string.
- **canal:** Contiene datos de tipo string.

Además se puede conocer la cantidad de valores únicos que posee cada una de las columnas del dataframe (luego del proceso ETL):

- **rut cliente:** Contiene 35,524 valores únicos.
- **fecha:** Contiene 21,148 valores únicos.
- **metodo:** Contiene 85 valores únicos.

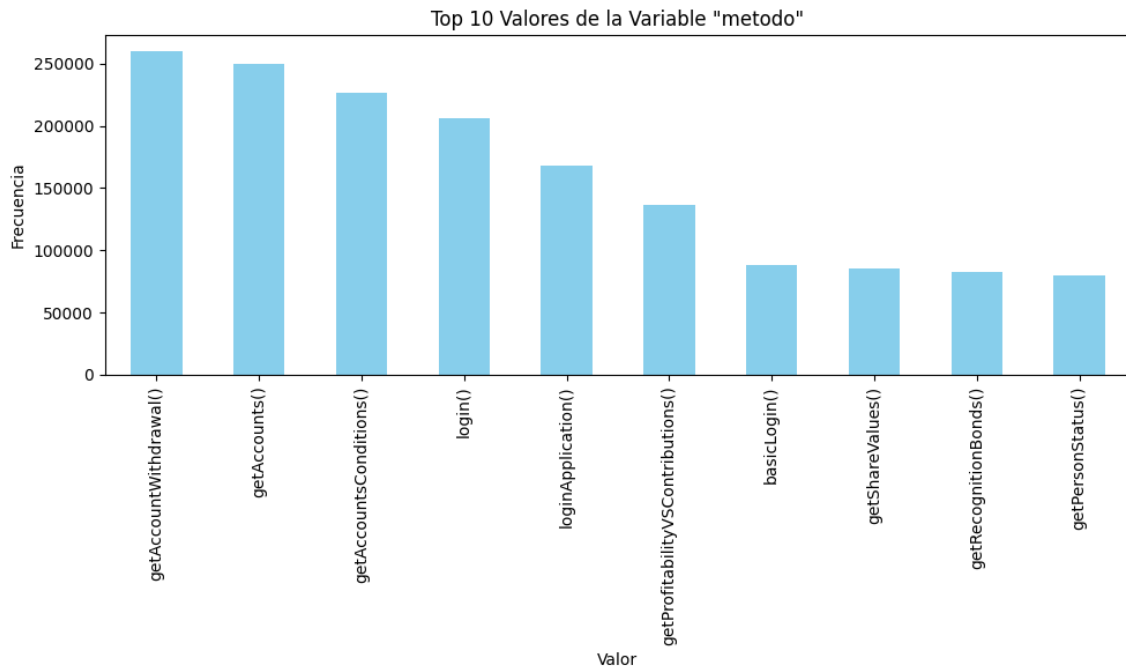
- **canal:** Contiene 2 valores únicos.

Ahora veremos la frecuencia con la que aparecen los valores más frecuentes en cada campo:

- Para rut el valor más común es 'MTU0NjM5NzIx' y tiene una frecuencia de 320,689.
- Para método el valor más común es 'getAccountWithdrawal()' y tiene una frecuencia de 259,733.
- Para el canal el más común es 'PWA' con una frecuencia de 1,029,538.
- Para el campo fecha es distinto ya que cada registro posee valores distintos, esto se debe al detalle del valor guardado ya que este llega a considerar las milésimas de segundos.

Ahora veremos un gráfico que representa la distribución de los métodos en el conjunto de datos. En esta ocasión, se presentarán únicamente los diez métodos que tienen la mayor frecuencia, dado que la cantidad de métodos registrados es demasiado grande para ser mostrada en un solo gráfico.

Figura 21: Distribución de métodos en el dataset.



Fuente: Elaboración propia.

La distribución muestra que las actividades relacionadas con cuentas (`getAccountWithdrawal()`, `getAccounts()`, y `getAccountsConditions()`) son las más frecuentes. Esto sugiere que la mayoría de los usuarios están interesados en la gestión y consulta de sus cuentas. Por otro lado, las actividades como `login()` y `loginApplication()` también tienen una alta frecuencia, indicando que el proceso de inicio de sesión es común, lo cual es esperado en una plataforma segura. De acá podemos concluir que las actividades más populares pueden ser un área clave para la mejora continua y la optimización de la experiencia del usuario. Las actividades menos frecuentes no deben ser ignoradas; se debe evaluar su importancia y cómo pueden ser promocionadas o mejoradas.

Para analizar las fechas, se decidió realizar una descripción de estas por día y hora. Año y mes quedan fuera del análisis ya que todos los datos pertenecen al mismo año y mes. Respecto a los días, en el set de datos hay un total de ocho días, y están presentes las veinticuatro horas del día. Estas tienen una frecuencia de:

Distribución de días:

- Se cuenta con un rango de 8 días consecutivos, comenzando en el día 14 y finalizando en el día 20. Cada día está asociado con un número que refleja una cantidad particular, con el día 14 teniendo la mayor cantidad de 4689, y el día 20 la menor con 138.

Distribución de horas:

- Se observa una distribución de métricas a lo largo de las 24 horas del día. Las horas con las mayores cantidades se presentan al inicio de la tarde, con la hora 14 liderando con 1382 unidades. La actividad disminuye progresivamente hacia las horas nocturnas, siendo la hora 22 la última con una cantidad significativa de 1055 unidades antes de una reducción notable durante las horas de madrugada. Las horas menos activas son las primeras horas de la mañana, con la hora 5 registrando la menor cantidad de 438 unidades.

Al analizar la distribución de los días, se observa que los días 14 y 13 presentan la mayor frecuencia en los registros, con 4689 y 4598 eventos, respectivamente. A medida que avanzan los días, se nota una disminución significativa en la cantidad de registros, llegando a solo 138 eventos el día 20.

En cuanto a la distribución de las horas, se puede observar que las horas 14, 15 y 13 tienen las frecuencias más altas, con 1382, 1263 y 1240 eventos, respectivamente. Las horas entre las 16 y las 21 también muestran una frecuencia considerablemente alta, mientras que las horas más tempranas (de 00 a 04) y las horas de la madrugada (después de las 04) presentan menor actividad, siendo la hora 05 la menos frecuente con 438 eventos.

Este análisis detallado por días y horas proporciona una visión más completa sobre la distribución temporal de los eventos registrados en el conjunto de datos, lo que puede ser útil para comprender los patrones de comportamiento de los usuarios en la plataforma.

CAPÍTULO 6: MODELOS DE PREDICCIÓN APLICADOS

6.1. Modelo de series de tiempo

Se implementó un modelo de series de tiempo ARIMA o modelo autorregresivo de media móvil integrado, un modelo centrado en describir las autocorrelaciones que existen entre los datos e intentar predecir y comprender el comportamiento del usuario.

6.1.1. Preprocesamiento y Preparación de Datos

Comenzamos con la carga de los datos desde un archivo CSV, seguida de un proceso de limpieza para asegurar la integridad de la información. En esta etapa, se filtraron registros específicos en la columna “metodo” y se descartaron aquellos casos donde la columna “canal” estaba vacía. Adicionalmente, la columna “fecha_evento” fue convertida al formato UTC para poder garantizar una estandarización completa a lo largo del conjunto de datos.

6.1.2. Codificación y Transformación One-Hot

Se transformaron las variables categóricas ‘metodo’ y ‘canal’ en representaciones numéricas que son adecuadas para el procesamiento del modelo.

- **Normalización de Fechas:** La columna ‘fecha_evento’, que contenía información de fecha y hora, se dividió y estandarizó para asegurar un formato uniforme y utilizable.
- **Eliminación de columnas innecesarias:** Las columnas originales de ‘metodo’ y ‘canal’ se eliminaron después de la codificación, manteniendo el dataset conciso y enfocado en las características relevantes.

- **Transformación Temporal Adicional:** Se llevaron a cabo codificaciones one-hot para las columnas de 'fecha', 'hora' y 'día del mes', ampliando la representatividad de los aspectos temporales de los datos.
- **Transformación y normalización de columna 'metodo':** Se llevo a cabo una transformación de la columna 'metodo' para tener valores numéricos de 0 a $n - 1$ y luego se normalizaron estos valores para obtener datos entre 0 y 1.

6.1.3. Construcción y Entrenamiento del Modelo

El dataset fue sometido al test estadístico Dicker-Fuller Aumentada (ADF) para saber si este corresponde a una serie de tiempo estacionaria o no (Mencionado en el ítem 3.3.5.), la variable que nos dira si la serie de tiempo es estacionaria o no, es P_value , si $p < 0,05$ quiere decir que la serie de tiempo es estacionaria, en el caso de que $p > 0,05$ quiere decir que la serie de tiempo no es estacionaria. Luego, debido a incidencias que se mencionaran más adelante, se decide aplicar el modelo ARIMA sobre un cliente, con el fin de probar el funcionamiento del modelo a la hora de predecir el comportamiento de un único cliente. Ocupando el modulo 'auto_arima' buscamos un modelo ARIMA(p, d, q) más optimo, el que minimice el valor AIC (Akaike's Information Criterion). El dataset es dividido en conjuntos de entrenamiento y validación, luego se implementó el modelo en el set de datos de entrenamiento y el orden (p, d, q) obtenido anteriormente con el modulo 'auto_arima'.

6.1.4. Conclusión del Modelo de series de tiempo ARIMA

Debido a la naturaleza de los datos, el modelo ARIMA no puede entregar resultados significativos, esto gracias a que la variable objetivo 'metodo' contiene eventos categóricos y el modelo ARIMA esta pensado para predecir numéricos en series de tiempo. También, para poder predecir el comportamiento de cada cliente se tendría que implementar un modelo ARIMA por cliente, haciendo el proceso a gran escala lento y poco eficaz.

6.2. Modelo de autoencoders

Como parte de nuestra estrategia de aprendizaje no supervisado, se implementó un modelo de autoencoder con el objetivo de predecir y comprender el comportamiento del usuario utilizando un conjunto de datos complejo. El autoencoder, una herramienta avanzada en el campo del aprendizaje automático, se especializa en identificar patrones y características intrínsecas dentro de grandes volúmenes de datos. Esta implementación fue clave para descifrar las complejidades y matices del comportamiento del usuario, permitiendo al modelo aprender de manera autónoma sin la necesidad de etiquetas predefinidas. La aplicación de este modelo de autoencoder facilitó el análisis detallado de los datos, contribuyendo significativamente a la comprensión de patrones subyacentes y tendencias de comportamiento de los usuarios.

6.2.1. Preprocesamiento y Preparación de Datos

El proceso de preparación de los datos para el entrenamiento del modelo comenzó con la carga del conjunto de datos desde un archivo CSV. Esta etapa inicial fue seguida por una meticulosa fase de limpieza de datos, un paso crucial para asegurar la calidad y la fiabilidad de los datos que se utilizarían en el entrenamiento del modelo. Las operaciones de preprocesamiento realizadas en los datos incluyeron las siguientes:

- **Normalización de Fechas:** En el proceso de preparación de los datos, la columna “fecha_evento”, que contenía información relevante de fecha y hora, fue objeto de una cuidadosa división y estandarización. Este procedimiento se realizó con el fin de asegurar que los datos temporales se presentaran en un formato uniforme y coherente, facilitando su utilización y análisis posterior. La estandarización de esta columna es un paso crítico, ya que permite que las marcas temporales sean procesadas de manera efectiva por el modelo, y garantiza la consistencia en la interpretación de los datos temporales a lo largo de todo el conjunto de datos. La adecuada preparación y formato de los datos temporales son esenciales para cualquier análisis que involucre aspectos temporales y secuenciales.
- **Eliminación de columnas innecesarias:** Tras el proceso de codificación, las columnas originales de “método” y “canal” en nuestro conjunto de datos

fueron eliminadas. Esta decisión se tomó para mantener el dataset conciso y enfocado exclusivamente en las características más relevantes para el análisis y el modelado. La eliminación de estas columnas originales, después de haber sido transformadas en un formato más adecuado para el procesamiento por el modelo, es una práctica estándar en el preprocesamiento de datos. Este enfoque ayuda a reducir la dimensionalidad del conjunto de datos y asegura que el modelo se centre en los atributos que aportan información valiosa y significativa, evitando así la redundancia y el ruido innecesario.

- **Transformación Temporal Adicional:** Se implementaron codificaciones one-hot en las columnas “fecha”, “hora” y “día de la semana” de nuestro conjunto de datos. Este enfoque de codificación fue elegido para mejorar la representación de los aspectos temporales en el análisis de los datos.

6.2.2. Codificación One-Hot

En relación con las variables categóricas “método” y “canal”, se aplicó la técnica de codificación One-Hot para su adecuada transformación. Esta técnica es un componente esencial en la preparación de datos categóricos para su incorporación en modelos de aprendizaje automático. Consiste en convertir estas variables en un formato numérico, optimizando así su interpretación y manejo por el modelo. Con la codificación One-Hot, cada categoría única de estas variables se traduce en una columna independiente en el conjunto de datos. En estas columnas recién creadas, la presencia de una categoría específica en una instancia se denota con un “1”, mientras que su ausencia se representa con un “0”. Este método facilita que el modelo reconozca y procese las características categóricas de manera eficiente, sin presuponer un orden jerárquico entre ellas, lo cual es vital para lograr un análisis preciso y una modelización efectiva.

Figura 22: Preparación y codificación one-hot de los datos

```
# Codificación One-Hot para las columnas metodo
metodo_onehot = pd.get_dummies(raw_data_copy['metodo'], prefix='metodo').astype(int)
raw_data_copy = pd.concat([raw_data_copy, metodo_onehot], axis=1)

# Codificación One-Hot para las columnas canal
canal_onehot = pd.get_dummies(raw_data_copy['canal'], prefix='canal').astype(int)
raw_data_copy = pd.concat([raw_data_copy, canal_onehot], axis=1)

# Separación de la fecha y hora en 2 columnas separadas
raw_data_copy['fecha'] = raw_data_copy['fecha_evento'].str.split('T').str[0]
raw_data_copy['hora'] = raw_data_copy['fecha_evento'].str.split('T').str[1]

# Normalización de la hora para que solo salga en el formato hh
raw_data_copy['hora'] = raw_data_copy['hora'].str.slice(0, 2) # Modificado para tomar solo las dos primeras cifras que representen la hora

# Convertir la columna fecha a tipo datetime
raw_data_copy['fecha'] = pd.to_datetime(raw_data_copy['fecha'])

# Añadir una nueva columna con el día de la semana
raw_data_copy['dia_semana'] = raw_data_copy['fecha'].dt.day_name()

# Drop de columnas fecha_evento, metodo y canal
raw_data_copy = raw_data_copy.drop(columns=['rut_cliente', 'fecha_evento', 'metodo', 'canal'])

# Codificación one-hot para las columnas 'fecha', 'hora', y 'dia_semana'
fecha_one_hot = pd.get_dummies(raw_data_copy['fecha'], prefix='fecha').astype(int)
hora_one_hot = pd.get_dummies(raw_data_copy['hora'], prefix='hora').astype(int)
dia_semana_one_hot = pd.get_dummies(raw_data_copy['dia_semana'], prefix='dia_semana').astype(int)

# Concatenar las codificaciones one-hot al DataFrame original
raw_data_copy = pd.concat([raw_data_copy, fecha_one_hot, hora_one_hot, dia_semana_one_hot], axis=1)

raw_data_copy = raw_data_copy.drop(['fecha', 'hora', 'dia_semana'], axis=1)
```

Fuente: Elaboración propia.

6.2.3. Construcción y Entrenamiento del Modelo

El dataset fue dividido en conjuntos de entrenamiento y validación. Luego, se construyó la arquitectura del autoencoder con capas densas y normalización por lotes, utilizando una función de activación “tanh” para las capas codificadoras y “relu” para las decodificadoras. Se implementó un mecanismo de EarlyStopping para prevenir el sobreentrenamiento y mejorar la generalización del modelo.

Un autoencoder es un tipo de red neuronal utilizada en el aprendizaje no supervisado, que tiene como objetivo aprender una representación (codificación) de un conjunto de datos, típicamente para reducción de dimensionalidad, aprendizaje de características, o descompresión de datos. La particularidad de los autoencoders reside en que están diseñados para reconstruir su entrada en la salida, pasando la información a través de una serie de capas que primero comprimen los datos (codificador) y luego los reconstruyen (decodificador).

En el caso de un autoencoder con “capas densas” (también conocidas como “capas completamente conectadas”), cada neurona en una capa está conectada a todas las neuronas en la capa siguiente. Este tipo de arquitectura es una de las más comunes en las redes neuronales y es particularmente útil para aprender patrones complejos y no lineales en los datos.

Construcción del Codificador

En la parte del codificador, los datos de entrada pasan a través de capas densas, donde se reduce gradualmente la dimensionalidad. Este proceso se logra reduciendo el número de neuronas en capas sucesivas, obligando así a la red a aprender una representación comprimida de los datos de entrada. Las funciones de activación como “tanh” (tangente hiperbólica) pueden ser usadas aquí para introducir no linealidades en el modelo, permitiendo que el codificador aprenda representaciones más complejas.

Construcción del Decodificador

La segunda parte del autoencoder es el decodificador, donde la representación comprimida se procesa a través de otras capas densas para reconstruir los datos de entrada. El objetivo es que la salida sea lo más cercana posible a la entrada original. Las funciones de activación como “relu” (Rectified Linear Unit) son comunes en esta etapa, proporcionando una manera eficiente y efectiva de realizar operaciones no lineales.

Figura 23: Arquitectura del modelo autoencoder

```
input_dim = X_train.shape[1]

# Capa de entrada
input_layer = Input(shape=(input_dim,))

# Capas codificadoras
encoded = Dense(100, activation='tanh')(input_layer) # Aumentado a 100
encoded = BatchNormalization()(encoded)
encoded = Dense(75, activation='tanh')(encoded) # Aumentado a 75
encoded = BatchNormalization()(encoded)
encoded = Dense(50, activation='relu')(encoded)
encoded = BatchNormalization()(encoded)
encoded = Dense(25, activation='relu')(encoded)
encoded = BatchNormalization()(encoded)
encoded = Dense(7, activation='relu')(encoded)

# Capas decodificadoras
decoded = Dense(7, activation='relu')(encoded)
decoded = BatchNormalization()(decoded)
decoded = Dense(25, activation='relu')(decoded)
decoded = BatchNormalization()(decoded)
decoded = Dense(50, activation='relu')(decoded)
decoded = BatchNormalization()(decoded)
decoded = Dense(75, activation='tanh')(decoded)
decoded = BatchNormalization()(decoded)
decoded = Dense(100, activation='tanh')(decoded) # Aumentado a 100
decoded = BatchNormalization()(decoded)
decoded = Dense(input_dim, activation='tanh')(decoded) # Capa de salida con misma dimensión que la entrada

# Construye el autoencoder
autoencoder = Model(input_layer, decoded)

early_stopping = EarlyStopping(monitor='val_loss', patience=5, verbose=1, restore_best_weights=True)

autoencoder.compile(optimizer='adam', loss='mean_squared_error')
```

Fuente: Elaboración propia.

El entrenamiento del autoencoder se realizó adoptando un enfoque iterativo y dinámico, donde uno de los aspectos clave fue el monitoreo de la pérdida de validación a lo largo de las iteraciones del entrenamiento, conocidas como "épocas". En el contexto del aprendizaje automático y especialmente en el entrenamiento de redes neuronales, una "época" se refiere a un ciclo completo de pase de todos los datos de entrenamiento a través del modelo.

Una época representa una iteración completa sobre el conjunto de datos de entrenamiento. Durante una época, el modelo procesa cada muestra del conjunto de entrenamiento una vez, permitiendo que el modelo aprenda de los datos. Esto implica que si tenemos un conjunto de datos de entrenamiento con, por ejemplo, 1000 muestras y el modelo se entrena durante 10 épocas, entonces cada muestra habrá sido utilizada 10 veces para ajustar los parámetros del modelo.

Las épocas son fundamentales en el proceso de entrenamiento de un modelo de aprendizaje automático:

Aprendizaje Progresivo

Con cada época, el modelo tiene la oportunidad de aprender y adaptarse a los datos, ajustando sus parámetros (como los pesos en una red neuronal) para minimizar el error o la pérdida.

Evaluación y Ajustes

Al monitorear la pérdida, especialmente la pérdida de validación, al final de cada época, se puede evaluar cómo está aprendiendo el modelo. Si la pérdida de validación deja de mejorar o comienza a aumentar, puede ser una señal de sobreajuste, indicando que el modelo está aprendiendo a memorizar los datos de entrenamiento en lugar de generalizar a partir de ellos.

Término del Entrenamiento

El número de épocas también juega un papel crucial en la determinación de cuándo detener el entrenamiento. Demasiadas épocas pueden llevar a sobreajuste, mientras que muy pocas pueden resultar en un modelo subajustado.

Figura 24: Entrenamiento del modelo autoencoder

```
history = autoencoder.fit(X_train, X_train,
                          epochs=100,
                          batch_size=32,
                          shuffle=True,
                          validation_data=(X_test, X_test),
                          callbacks=[early_stopping])
```

Train on 1865976 samples, validate on 466495 samples

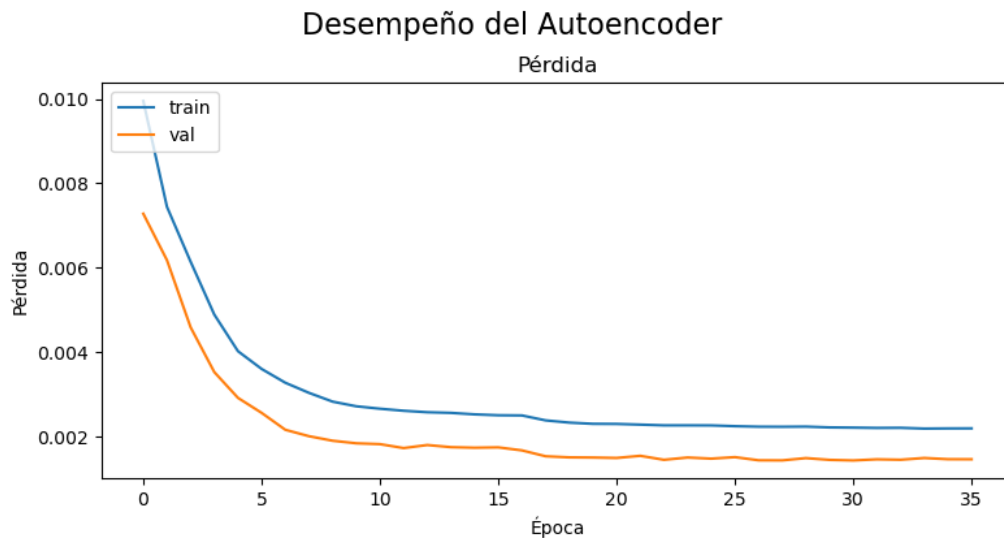
Epoch 1/100	1865976/1865976 [=====]	- 405s 217us/sample	- loss: 0.0100	- val_loss: 0.0073
Epoch 2/100	1865976/1865976 [=====]	- 423s 226us/sample	- loss: 0.0074	- val_loss: 0.0062
Epoch 3/100	1865976/1865976 [=====]	- 349s 187us/sample	- loss: 0.0061	- val_loss: 0.0046
Epoch 4/100	1865976/1865976 [=====]	- 356s 191us/sample	- loss: 0.0049	- val_loss: 0.0035
Epoch 5/100	1865976/1865976 [=====]	- 357s 191us/sample	- loss: 0.0040	- val_loss: 0.0029
Epoch 6/100	1865976/1865976 [=====]	- 355s 190us/sample	- loss: 0.0036	- val_loss: 0.0026
Epoch 7/100	1865976/1865976 [=====]	- 347s 186us/sample	- loss: 0.0033	- val_loss: 0.0022
Epoch 8/100	1865976/1865976 [=====]	- 353s 189us/sample	- loss: 0.0030	- val_loss: 0.0020
Epoch 9/100	1865976/1865976 [=====]	- 357s 191us/sample	- loss: 0.0028	- val_loss: 0.0019
Epoch 10/100	1865976/1865976 [=====]	- 348s 186us/sample	- loss: 0.0027	- val_loss: 0.0018

Fuente: Elaboración propia.

6.2.4. Resultados y Evaluación

El modelo final demostró una sólida capacidad de reconstrucción y no mostró signos de sobreajuste, como se evidencia en los gráficos de pérdida de entrenamiento. Este autoencoder está ahora listo para ser utilizado para transformar los datos en una representación de menor dimensión, lo cual facilitará la aplicación de técnicas de clustering para la segmentación de usuarios y la detección de patrones de comportamiento anómalos.

Figura 25: Gráfico del entrenamiento del modelo autoencoder



Fuente: Elaboración propia.

6.2.5. Conclusión del Modelo de Autoencoder

El autoencoder se diseñó para transformar datos de alta dimensión en una representación de menor dimensionalidad, lo cual es un paso fundamental en el pre-procesamiento para algoritmos de clustering.

Sin embargo, el propósito inicial del autoencoder no era predecir comportamientos futuros o tendencias, sino más bien identificar patrones intrínsecos y datos atípicos dentro del conjunto de datos existente. Dado que el objetivo central del proyecto es la predicción del comportamiento, se ha determinado que el autoencoder no cumple con los requisitos necesarios para avanzar hacia este nuevo objetivo.

Por tanto, hemos decidido no continuar con el desarrollo e implementación de este modelo en su forma actual. En cambio, nos enfocaremos en la búsqueda y aplicación de modelos predictivos más adecuados que estén alineados con las metas de pronosticar el comportamiento futuro y que puedan ofrecer insights más directos para la toma de decisiones y estrategias proactivas.

Este cambio de dirección subraya la importancia de una alineación clara entre las herramientas de modelado seleccionadas y los objetivos específicos del proyecto. Aunque el autoencoder es una herramienta poderosa para ciertas aplicaciones, en este caso, se ha reconocido que no es la solución óptima para las necesidades proyectadas.

Sin embargo, es importante destacar que el autoencoder es una herramienta extremadamente útil para identificar anomalías en patrones de comportamiento.

6.3. Modelo de predicción secuencial

Para el propósito de predecir y comprender con precisión el comportamiento del usuario, se desarrolló e implementó un modelo de clasificación avanzado. Este modelo se basó en el análisis de un conjunto de datos detallado y complejo, que incluía múltiples variables y patrones de interacción de los usuarios. La finalidad de este modelo es no solo prever las acciones futuras de los usuarios, sino también desentrañar las dinámicas subyacentes y las tendencias en su comportamiento. La implementación de este modelo de clasificación juega un papel crucial en la generación de insights valiosos y en la toma de decisiones informadas basadas en el análisis de datos.

6.3.1. Preprocesamiento y Preparación de Datos

El proceso de preparación de los datos comenzó con la importación de la información desde un archivo CSV, seguida de una meticulosa etapa de limpieza para asegurar la integridad y relevancia de los datos. Durante esta etapa, se realizaron filtros específicos en la columna “método” y se descartaron registros en los que la columna “canal” estaba vacía. Esta selección tenía como objetivo eliminar elementos irrelevantes o incompletos que podrían distorsionar los resultados del análisis. Adicionalmente, se llevó a cabo la conversión de las marcas temporales al formato UTC, una medida crucial para lograr una estandarización homogénea a lo largo de todo el conjunto de datos.

El preprocesamiento es un paso crítico para garantizar que los datos estén limpios, completos y adecuadamente preparados para el modelado. Esta fase incluyó

la eliminación o corrección de valores atípicos, el manejo de valores faltantes y la confirmación de la consistencia de los datos. La decisión de filtrar métodos específicos y eliminar registros con campos de canales vacíos se tomó con el fin de conservar únicamente aquellos datos que aportan valor significativo al análisis. La estandarización de las marcas temporales a UTC es fundamental para asegurar la precisión en el análisis temporal, permitiendo comparaciones efectivas de eventos a través de diferentes zonas horarias.

6.3.2. Ordenamiento y Etiquetado

El texto proporcionado es claro y detallado, describiendo eficazmente el proceso de preparación de datos para un modelo de clasificación secuencial en el aprendizaje automático. Para presentarlo en un lenguaje más formal y técnico, adecuado para un documento académico o técnico, se puede hacer una ligera modificación para mejorar la fluidez y la claridad. Aquí tienes una versión revisada:

En la fase de preparación de los datos para nuestro modelo de clasificación secuencial, se implementó un proceso de ordenamiento cronológico por usuario y fecha de evento, crucial para preservar la secuencia de acciones de cada usuario. Este ordenamiento se realizó agrupando las acciones en el conjunto de datos por la columna “rut_cliente” y luego ordenándolas dentro de cada grupo según la “fecha_evento”. Este procedimiento asegura que se mantenga la secuencia temporal de las actividades, una práctica esencial para modelos que requieren comprender y aprender de las transiciones entre diferentes acciones o estados de un usuario.

Posteriormente, procedimos a la generación de etiquetas para el aprendizaje supervisado. Excluyendo la última acción de cada secuencia o sesión, cada acción fue etiquetada con la acción subsiguiente empleando el método “.shift(-1)” en el DataFrame agrupado. Esta técnica desplaza las acciones hacia arriba, permitiendo así que cada acción se alinee con su sucesora inmediata. Las acciones siguientes se utilizaron como etiquetas para las acciones precedentes, y para las últimas acciones de las secuencias, donde no hay una acción sucesiva, se asignó una etiqueta específica como “final_del_recorrido” para indicar el fin de la sesión.

Este enfoque es vital para el entrenamiento de modelos de aprendizaje supervisado, particularmente en el contexto de modelos secuenciales como las Redes Neuronales Recurrentes (RNN) o las de Memoria a Corto y Largo Plazo (LSTM). Proporciona al modelo los “objetivos” a predecir, basándose en el contexto de las acciones previas. Con este proceso, cada acción en el conjunto de datos se asocia con una etiqueta que representa la acción futura inmediata, creando así un conjunto de datos listo para el entrenamiento de un modelo predictivo capaz de anticipar el comportamiento futuro del usuario a partir de su historial de interacciones.

6.3.3. Identificación de Sesiones

Para obtener una comprensión más profunda de las interacciones de los usuarios con el sistema, procedimos a segmentar el flujo continuo de datos en lo que denominamos sesiones discretas. En este contexto, una sesión se define como un conjunto de eventos consecutivos llevados a cabo por un usuario, los cuales están limitados por períodos de inactividad que no exceden un umbral de tiempo predeterminado. Este criterio de segmentación nos permite identificar y analizar las secuencias de acciones realizadas por los usuarios, considerando como límite de una sesión la presencia de una pausa significativa en su actividad. Esta definición de sesión es crucial para el análisis detallado del comportamiento de los usuarios, ya que facilita la identificación de patrones y tendencias dentro de las interacciones individuales con el sistema.

Cálculo de Diferencias de Tiempo

El proceso de análisis se inició con el cálculo de los intervalos de tiempo entre eventos consecutivos para cada usuario. Esta tarea se llevó a cabo aplicando el método `.diff()` a la columna de marcas temporales en nuestro conjunto de datos. Para asegurar la precisión de esta operación, la columna de marcas temporales fue previamente ordenada por usuario y por la fecha de cada evento. Como resultado de la aplicación de este método, se generó una nueva columna en el conjunto de datos. Esta columna adicional representa el tiempo transcurrido entre cada acción registrada y la acción subsiguiente, proporcionando así una medida cuantitativa del intervalo temporal entre eventos sucesivos. Esta información es crucial para identificar patrones de comportamiento y para la determinación de sesiones dentro del flujo de interacción

del usuario.

Definición de un Umbral de Tiempo

Para la identificación de nuevas sesiones en nuestro análisis, se estableció un umbral temporal específico. La definición de este umbral se basó en un estudio detallado del patrón de uso característico del sitio web. Se determinó que un período de inactividad superior a 30 minutos es indicativo del final de una sesión de usuario. Esta decisión se fundamenta en la observación de que lapsos prolongados sin actividad tienden a representar una pausa natural o conclusión en la interacción del usuario con el sitio. La implementación de este umbral temporal es esencial para segmentar el conjunto de datos de manera precisa, permitiendo así una evaluación más estructurada y coherente de las sesiones de usuario para fines de análisis y modelación predictiva.

Identificación de Nuevas Sesiones

Mediante la comparación de las diferencias temporales con un umbral preestablecido, fue posible identificar el comienzo de nuevas sesiones en el conjunto de datos. En este contexto, un valor que excede el umbral establecido en la columna de diferencias de tiempo es indicativo de que la acción correspondiente marca el inicio de una nueva sesión. Esta metodología se basa en la premisa de que intervalos de tiempo sustancialmente largos entre acciones consecutivas son representativos de una pausa en la interacción del usuario, y por lo tanto, señalan el comienzo de una nueva sesión de actividad. Esta técnica es esencial para segmentar adecuadamente el conjunto de datos en sesiones individuales, lo cual es fundamental para el análisis detallado y la modelación predictiva del comportamiento de los usuarios en secuencias temporales.

Asignación de Identificadores de Sesión

Para identificar claramente las nuevas sesiones en nuestro conjunto de datos, se llevó a cabo una transformación de los valores booleanos, obtenidos como resultado de la comparación con el umbral establecido, en un identificador numérico acumulativo. Este proceso se implementó mediante el uso del método `.cumsum()`, el cual incrementa automáticamente el identificador cada vez que se detecta el inicio de una nueva sesión. De este modo, se asigna un número único y secuencial a cada sesión para cada usuario, facilitando así la distinción y el seguimiento de las distintas sesiones en el análisis

posterior. Este enfoque asegura una organización coherente y una identificación precisa de las sesiones individuales dentro del conjunto de datos, lo cual es crucial para el análisis detallado y la modelación predictiva del comportamiento de los usuarios.

El código que se utilizó para llevar a cabo este proceso esencial de estructuración de datos se presenta a continuación. Este fragmento de código desempeña un papel crucial en la organización de los datos, asegurando que reflejen adecuadamente los períodos de interacción activa de los usuarios con el sistema. La implementación correcta de este código es fundamental para la preparación de los datos, facilitando un análisis detallado y una modelación predictiva eficaz del comportamiento futuro de los usuarios. El siguiente código detalla cómo se efectuó esta transformación y estructuración de los datos:

Figura 26: Código indentificación de sesiones

```
# Una vez que ya hemos ordenado raw_data_copy por 'rut_cliente' y 'fecha_evento'

# Definir un umbral de tiempo para una nueva sesión, por ejemplo, 30 minutos
threshold = pd.Timedelta(minutes=30)

# Calcular la diferencia de tiempo entre eventos consecutivos
raw_data_copy['tiempo_diferencia'] = raw_data_copy.groupby('rut_cliente')['fecha_evento'].diff()

# Identificar el comienzo de una nueva sesión
raw_data_copy['nueva_sesion'] = (raw_data_copy['tiempo_diferencia'] > threshold).astype(int)

# Calcular el identificador de sesión acumulativo para cada usuario
raw_data_copy['id_sesion'] = raw_data_copy.groupby('rut_cliente')['nueva_sesion'].cumsum()

# Ahora puedes eliminar la columna 'tiempo_diferencia' si ya no la necesitas
raw_data_copy.drop('tiempo_diferencia', axis=1, inplace=True)
```

Fuente: Elaboración propia.

Este método de estructuración de los datos fue fundamental para capturar con exactitud los períodos de interacción activa de los usuarios con el sistema. Al reflejar de manera precisa estas secuencias de interacción, se estableció una base sólida y fiable para el análisis detallado y la modelación predictiva del comportamiento futuro de los usuarios. Este enfoque es esencial en el aprendizaje automático, ya que proporciona el contexto y la profundidad necesarios para que el modelo pueda identificar patrones significativos y tendencias en el comportamiento de los usuarios, lo cual es crucial para realizar predicciones precisas y significativas sobre sus acciones futuras.

6.3.4. Codificación de Categorías

En el campo del aprendizaje automático, es frecuente el manejo de datos categóricos, es decir, información que no se presenta en forma numérica. Un ejemplo claro son los nombres de métodos y canales en nuestro conjunto de datos. Para facilitar el procesamiento de estos datos por parte de un modelo de clasificación, es imperativo convertir estas categorías textuales a un formato numérico que el modelo pueda interpretar y manejar de manera eficiente. Este proceso de transformación se denomina codificación de categorías. La realización de esta codificación es esencial para que el modelo pueda analizar adecuadamente los datos y generar predicciones precisas. A continuación, detallamos cómo se implementó este proceso de codificación en nuestros datos:

Mapeo de Categorías a Números:

- En el proceso de preparación de nuestros datos, se establecieron dos diccionarios distintos: uno dedicado a los **métodos** y otro a los **canales**. Esta separación asegura una organización clara y específica, permitiendo que cada tipo de información sea codificado y manejado de manera independiente. La creación de un diccionario específico para los métodos y otro para los canales facilita el mapeo preciso de estas categorías a valores numéricos, lo cual es esencial para el procesamiento posterior y el análisis en el contexto del modelo de aprendizaje automático.
- Cada diccionario en nuestro enfoque de procesamiento de datos está diseñado para asignar de manera unívoca una categoría textual a un número entero distinto. Esta correspondencia uno a uno entre las categorías textuales y los valores numéricos enteros es fundamental para la conversión eficiente de datos cualitativos en una forma cuantitativa que pueda ser procesada y analizada por el modelo de aprendizaje automático. Este mapeo asegura que cada categoría textual, representando una característica específica o una acción dentro del conjunto de datos, tenga una representación numérica única, lo cual es esencial para la integridad y la coherencia del proceso de modelado.

- Como ilustración, en nuestro proceso de mapeo, el método **login()** se asigna al valor numérico 1, mientras que el método **getAccounts()** se asocia con el número 2. Este patrón de asignación numérica se repite de manera similar para los demás métodos en el conjunto de datos. Estos mapeos transforman las funciones y operaciones del usuario, originalmente expresadas en forma textual o de código, en una serie de valores numéricos discretos. Esta conversión es esencial para permitir el procesamiento computacional y el análisis cuantitativo en las etapas de modelado y entrenamiento del sistema de aprendizaje automático.

Aplicación de la Codificación:

- En el proceso de preparación de nuestros datos, empleamos diccionarios específicos para convertir todas las instancias de métodos y canales en nuestro DataFrame a formatos numéricos. Esta transformación implicó reemplazar cada método y canal presente en el conjunto de datos por su respectivo valor numérico, tal como se define en los diccionarios de mapeo. Este enfoque asegura una representación uniforme y cuantificable de estas variables, facilitando su manipulación y análisis por parte del modelo de aprendizaje automático. La conversión a valores numéricos es esencial para la eficacia del procesamiento de datos y para la realización de operaciones matemáticas y estadísticas relevantes en las etapas subsiguientes del modelado.

Beneficios de la Codificación

- La transformación de los datos a un formato numérico es un aspecto crítico en el campo del aprendizaje automático, dado que los modelos en esta área operan de manera más eficiente con variables numéricas. Estas variables permiten la aplicación de operaciones matemáticas y técnicas estadísticas durante el proceso de entrenamiento del modelo. La representación numérica facilita el manejo y procesamiento de los datos, permitiendo que el modelo realice cálculos complejos, optimice patrones y extraiga insights significativos. Esta capacidad es fundamental para el desarrollo de modelos que pueden aprender, adaptarse y realizar predicciones precisas basadas en los datos proporcionados.

- La codificación de categorías facilita la implementación de algoritmos de incrustación (embedding), los cuales son fundamentales en el análisis avanzado de datos. Estos algoritmos de embedding tienen la capacidad de capturar y representar información sobre las categorías en un espacio de características multidimensionales, proporcionando una riqueza y una profundidad mayor que la que se obtendría mediante una simple representación numérica entera. Este enfoque permite al modelo discernir y aprender de las sutilezas y relaciones complejas inherentes a las categorías, resultando en una representación de datos más rica y matizada que es esencial para mejorar la precisión y efectividad de las predicciones del modelo.

A continuación se presenta el fragmento de código implementado para llevar a cabo la codificación de categorías. Este código es un componente crucial en el proceso de preparación de los datos, ya que transforma las categorías de métodos y canales en un formato que puede ser procesado eficazmente por nuestro modelo de clasificación secuencial. La codificación adecuada es esencial para permitir que el modelo interprete y utilice esta información categórica de manera efectiva en sus procesos de aprendizaje y predicción. El siguiente código detalla cómo se realizó esta transformación:

Figura 27: Código codificación de categorías

```
# Obtener listas únicas de métodos y canales
unique_methods = raw_data_copy['metodo'].unique()
unique_channels = raw_data_copy['canal'].unique()

# Crear mapeos de codificación
method_to_int = {method: i for i, method in enumerate(unique_methods)}
channel_to_int = {channel: i for i, channel in enumerate(unique_channels)}

# Codificar métodos y canales
raw_data_copy['metodo_codificado'] = raw_data_copy['metodo'].map(method_to_int)
raw_data_copy['canal_codificado'] = raw_data_copy['canal'].map(channel_to_int)
```

Fuente: Elaboración propia.

La codificación de categorías representa una etapa preparatoria crítica en el proceso de preparación de datos para nuestro modelo de clasificación secuencial.

Esta fase de codificación es fundamental para que el modelo pueda interpretar eficazmente los patrones presentes en los métodos y canales utilizados por los usuarios. Al convertir estas categorías en un formato estructurado y legible por la máquina, se mejora considerablemente la habilidad del modelo para analizar y comprender la secuencia de acciones del usuario. Esto, a su vez, incrementa significativamente la precisión del modelo en la realización de predicciones informadas sobre las posibles acciones futuras que el usuario podría emprender. Este paso es, por lo tanto, esencial para asegurar que el modelo no solo capture la secuencia de eventos, sino que también derive insights relevantes y significativos a partir de ellos.

6.3.5. Construcción de Secuencias

La construcción de secuencias constituye un paso fundamental en la preparación de datos para el modelado en contextos de series temporales o secuenciales, como es el caso de nuestro modelo de clasificación. Este proceso implica la organización y estructuración meticulosa de los datos para que reflejen con precisión las secuencias naturales de interacciones de los usuarios. Tal organización es clave para preservar la integridad temporal y la secuencialidad inherente a los patrones de comportamiento de los usuarios. A continuación, detallamos el enfoque y los pasos específicos empleados para llevar a cabo esta tarea esencial:

Agrupación de Eventos en Secuencias:

- Una vez completada la codificación de los métodos y canales, cada acción individual del usuario queda representada por un par de números codificados. El paso subsiguiente en el proceso de modelado de datos implica la agrupación de estas acciones codificadas en secuencias. Estas secuencias están diseñadas para representar de manera integral la trayectoria completa de la interacción del usuario durante una sesión específica. Este enfoque permite capturar la secuencia y el flujo de las actividades del usuario, proporcionando así un contexto detallado y estructurado que es esencial para el análisis posterior y el entrenamiento del modelo de aprendizaje automático.
- Cada secuencia en nuestro modelo se forma a partir de pares de métodos

y canales codificados, y se construye específicamente para cada usuario y para cada sesión que se haya identificado previamente. Para lograr esto, los datos se organizan mediante un proceso de agrupación por usuario y sesión. Posteriormente, dentro de cada grupo, se listan las acciones codificadas siguiendo un orden cronológico. Este enfoque permite que cada secuencia refleje con precisión la secuencia temporal de eventos para un usuario en una sesión determinada, asegurando así que el modelo pueda capturar y aprender de la dinámica y el flujo de las interacciones del usuario.

Preparación para el Modelo de Red Neuronal:

- Las secuencias numéricas generadas constituyen el insumo principal para la alimentación de la red neuronal. En el ámbito de las Redes Neuronales Recurrentes (RNN), incluyendo las de tipo Long Short-Term Memory (LSTM), estas secuencias son fundamentales para el modelo, ya que le permiten reconocer y aprender patrones temporales y dependencias entre eventos consecutivos. La capacidad de las LSTM y otras RNN de procesar estas secuencias numéricas les confiere la habilidad única de capturar y analizar la dinámica temporal inherente a los datos, lo cual es esencial para la predicción precisa y el análisis de series temporales.

Estructura de Datos de Secuencias:

- Desde la perspectiva de la estructura de datos, las secuencias se organizan comúnmente como listas de listas (o arrays de arrays), en las cuales cada sublista representa una secuencia completa correspondiente a un usuario. Esta forma de organización de datos es idónea para modelar secuencias, ya que permite encapsular una serie de eventos o acciones consecutivas de cada usuario de manera estructurada y coherente. En este esquema, cada sublista contiene los elementos que componen una secuencia individual, reflejando la cronología y la progresión de las acciones del usuario. Este formato es ampliamente utilizado en el procesamiento de secuencias temporales, proporcionando una representación clara y eficiente que facilita el análisis y el aprendizaje automático.

A continuación, se presenta el fragmento de código utilizado para la construcción de secuencias en nuestro modelo. Este código es esencial para transformar los datos crudos en una forma estructurada que el modelo pueda procesar eficientemente. La construcción de secuencias es un paso crítico en la preparación de datos para modelado predictivo, ya que permite que el modelo aprenda a identificar y seguir patrones de comportamiento a lo largo del tiempo. El siguiente código ilustra cómo se lleva a cabo esta transformación:

Figura 28: Código construcción de secuencias

```
# Combinar método y canal en una tupla y crear secuencias
raw_data_copy['metodo_canal'] = list(zip(raw_data_copy['metodo_codificado'], raw_data_copy['canal_codificado']))

# Crear secuencias para cada sesión
grouped = raw_data_copy.groupby(['rut_cliente', 'id_sesion'])
sequences = grouped['metodo_canal'].apply(list).tolist()
```

Fuente: Elaboración propia.

Significado para el Modelado Predictivo:

- Las secuencias desempeñan un rol crucial en el ámbito del modelado predictivo, ya que suministran el contexto integral necesario para que el modelo realice predicciones basadas en información relevante. Por ejemplo, al analizar la secuencia de acciones realizadas previamente por un usuario, es posible inferir patrones conductuales que indican cuál podría ser la próxima acción. Un caso ilustrativo sería una secuencia donde el usuario realiza varias consultas de información, lo que podría llevar al modelo a predecir que la acción subsiguiente más probable sería cerrar sesión”. Este tipo de análisis secuencial permite al modelo capturar y aprender de las tendencias y patrones en el comportamiento del usuario, facilitando así predicciones más precisas y contextualizadas.

Alineación con Objetivos de Predicción:

- Además, las secuencias generadas se encuentran alineadas con las etiquetas que se definieron en la fase de etiquetado. En este proceso, cada acción dentro de una secuencia, con la excepción de la última, se asocia con una etiqueta

correspondiente que indica la acción subsiguiente. Esta metodología es esencial para el entrenamiento del modelo, ya que facilita la tarea de predecir la próxima acción basándose en la secuencia de acciones anteriores. La correcta alineación de secuencias con sus etiquetas correspondientes es crucial, ya que proporciona al modelo el contexto necesario para aprender a anticipar las acciones futuras de manera precisa, basándose en patrones de comportamiento observados previamente.

La construcción de secuencias es un proceso que convierte un conjunto de eventos individuales en una secuencia de caminos estructurados. Estos caminos reflejan de manera integral el comportamiento y las decisiones tomadas por los usuarios. Esta transformación es crucial para el desarrollo de un modelo de aprendizaje automático que sea capaz de predecir con precisión las acciones subsiguientes en el recorrido del usuario. Al capturar las secuencias de eventos en este formato estructurado, el modelo puede aprender eficazmente los patrones y tendencias inherentes en las decisiones y acciones de los usuarios, lo que es fundamental para anticipar sus movimientos futuros de manera precisa.

6.3.6. División de Datos

La partición de los datos en conjuntos específicos para el entrenamiento y la prueba constituye un aspecto crítico en el proceso de desarrollo de modelos de aprendizaje automático. Este paso es crucial para determinar la capacidad del modelo de generalizar a datos no observados previamente. En otras palabras, es fundamental para evaluar la habilidad del modelo para realizar predicciones precisas y fiables en nuevas entradas que no formaron parte del conjunto de datos de entrenamiento. A continuación, se detalla la metodología empleada para efectuar esta división de datos:

Separación en Conjuntos de Entrenamiento y Prueba:

El conjunto de datos completo se segmenta en dos partes principales: una sección destinada al entrenamiento del modelo, denominada el conjunto de entrenamiento, y otra sección utilizada para evaluar su rendimiento, conocida como el conjunto de prueba. Una práctica comúnmente aceptada en el campo del aprendizaje automático es asignar aproximadamente el 70-80 % del total de los datos para el entrenamiento y

el 20-30 % restante para las pruebas. Esta proporción se elige para asegurar que el modelo tenga suficientes datos para aprender de manera efectiva, mientras se retiene una cantidad adecuada de datos no vistos para una evaluación fiable y representativa de su rendimiento.

Aleatorización y Estratificación:

Es imperativo que la división de los datos se realice de manera aleatoria para prevenir cualquier sesgo en la selección de estos. Este enfoque aleatorio asegura una representación equitativa y no sesgada de los datos en los conjuntos de entrenamiento y prueba. Adicionalmente, en situaciones donde el conjunto de datos es amplio y exhibe una diversidad considerable, resulta beneficioso implementar una técnica conocida como estratificación. La estratificación tiene como objetivo garantizar que la proporción de las distintas clases o categorías de respuesta se mantenga constante tanto en el conjunto de entrenamiento como en el de prueba. Este método es crucial para preservar la distribución representativa de las clases a lo largo de los conjuntos de datos, lo cual es vital para asegurar la validez y confiabilidad de la evaluación del modelo.

Importancia para la Validación del Modelo:

La división del conjunto de datos en distintas porciones para entrenamiento y prueba es una estrategia crucial para validar la eficacia del modelo en escenarios realistas. El propósito de esta separación es evaluar si el modelo es capaz de funcionar adecuadamente en condiciones que simulan el entorno real. En este contexto, el conjunto de prueba desempeña un papel vital, actuando como un sustituto de datos nuevos y no vistos en el mundo real. Al no haber sido expuesto a estos datos durante la fase de entrenamiento, el modelo se somete a una prueba rigurosa de su capacidad de generalización y robustez, lo cual es indispensable para determinar su aplicabilidad y fiabilidad práctica.

A continuación se presenta el fragmento de código implementado para la división del conjunto de datos. Este código es esencial para separar los datos en diferentes conjuntos, lo que permite una evaluación más precisa y objetiva del modelo:

Figura 29: Código dividir el conjunto

```
# Dividir Los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_padded, y_categorical, test_size=0.2, random_state=42)
```

Fuente: Elaboración propia.

Protección Contra el Sobreajuste:

En el proceso de entrenamiento de modelos de aprendizaje automático, se debe estar consciente del riesgo inherente de sobreajuste. Este fenómeno ocurre cuando el modelo se adapta excesivamente a los datos de entrenamiento, llegando a aprender y reproducir no solo las características subyacentes sino también el ruido y las particularidades idiosincrásicas de ese conjunto de datos. Para detectar y mitigar el sobreajuste, la división de los datos en conjuntos distintos para el entrenamiento y la validación/test es una práctica esencial. Esta estrategia permite evaluar el rendimiento del modelo en datos no vistos durante el entrenamiento, proporcionando así una medida más objetiva y fiable de su capacidad de generalización y su rendimiento real.

Retroalimentación para la Iteración del Modelo:

Los resultados obtenidos del conjunto de prueba son cruciales, ya que proporcionan información valiosa para el proceso iterativo de refinamiento y mejora del modelo. Un aspecto particularmente importante a observar es la comparación del rendimiento del modelo en el conjunto de prueba frente a su desempeño en el conjunto de entrenamiento. Una discrepancia notable, donde el modelo muestra un rendimiento significativamente inferior en el conjunto de prueba, sirve como un indicador claro de sobreajuste. Esta situación implica que, aunque el modelo ha aprendido eficientemente los patrones específicos de los datos de entrenamiento, no logra generalizar bien a nuevos datos, lo cual es un aspecto crítico en la evaluación de su aplicabilidad práctica.

6.3.7. Construcción del Modelo

La fase de construcción del modelo representa un momento crítico en el proceso de aprendizaje automático, en el cual se establece la arquitectura que el modelo

empleará para aprender de los datos. En nuestro proyecto, hemos seleccionado un enfoque basado en un modelo de clasificación secuencial que emplea redes neuronales. Más específicamente, se ha implementado una Red Neuronal Recurrente (RNN) con unidades de Long Short-Term Memory (LSTM). Esta elección se basa en la habilidad de las RNN con LSTM para manejar eficientemente secuencias de datos, una característica esencial para nuestro objetivo de clasificación. A continuación, presentamos una descripción detallada de la estructura y configuración de este modelo:

Selección de la Arquitectura:

La decisión de implementar una Red Neuronal Recurrente (RNN) con unidades de Long Short-Term Memory (LSTM) se fundamentó en su comprobada eficacia para el procesamiento de datos secuenciales. Las unidades LSTM son particularmente destacadas por su capacidad para aprender y retener dependencias de largo plazo presentes en los datos. Esta característica las hace extraordinariamente adecuadas para tareas que requieren una comprensión profunda del contexto secuencial, como es el caso de la predicción de la próxima acción de un usuario basándose en su historial de acciones previas. La habilidad de las LSTM para capturar estas dependencias temporales complejas y mantener información relevante a lo largo de secuencias extensas es un factor clave para el éxito en este tipo de aplicaciones de modelado predictivo.

Capa de Incrustación (Embedding):

El modelo inicia su arquitectura con una capa de incrustación. La función principal de esta capa es convertir los índices correspondientes a métodos y canales, previamente codificados, en vectores densos de características. Esta transformación es crucial, ya que dota al modelo de la capacidad para interpretar de manera efectiva las entradas categóricas, facilitando la captura y el análisis de relaciones más complejas entre dichas entradas. La capa de incrustación juega un papel fundamental en el procesamiento de datos categóricos, mejorando significativamente la habilidad del modelo para discernir y aprender patrones y correlaciones subyacentes en los datos.

Capas LSTM:

Subsiguiente a la capa de incrustación, se incorporaron una o varias capas de tipo Long Short-Term Memory (LSTM). Estas capas son especialmente adecuadas para el procesamiento de datos secuenciales, ya que están diseñadas para capturar dependencias a largo plazo y mantener un estado interno. Este estado interno actúa como un repositorio de información que refleja el contexto acumulado hasta el punto actual en la secuencia. La capacidad de las capas LSTM para recordar información a lo largo de secuencias extensas las hace particularmente valiosas en tareas donde el contexto y la secuencialidad de los datos son factores críticos para la precisión y eficacia del modelo.

Capa Densa de Salida:

La configuración de la última capa de nuestro modelo consiste en una capa densa, diseñada con una unidad de salida para cada clase potencial, correspondiente en este contexto a cada acción posible siguiente. Esta capa crucial emplea la función de activación *softmax*, que es esencial para convertir las salidas de la red en una distribución de probabilidad sobre las clases de acciones posibles. La aplicación de la función *softmax* facilita que el modelo genere predicciones probabilísticas, asignando a cada clase potencial una probabilidad que refleja la confianza del modelo en esa acción específica como la acción siguiente más probable. Esta capacidad de producir una distribución de probabilidad es fundamental para el proceso de toma de decisiones y la interpretación de resultados en tareas de clasificación multiclase.

Figura 30: Construcción del modelo de clasificación

```
# Construir el modelo
model = Sequential()

# Capa de Embedding
model.add(Embedding(input_dim=len(method_to_int) + len(channel_to_int), output_dim=50, input_length=max_sequence_length))

# Primera capa LSTM con retorno de secuencias para apilar otra capa LSTM después
model.add(LSTM(100, return_sequences=True))
model.add(Dropout(0.2)) # Agregar Dropout para regularización

# Segunda capa LSTM
model.add(LSTM(50, return_sequences=False))
model.add(Dropout(0.2)) # Agregar Dropout para regularización

# Una capa Densa adicional (opcional, para una complejidad adicional)
model.add(Dense(50, activation='relu'))
model.add(Dropout(0.2)) # Agregar Dropout para regularización

# Capa Densa final para la clasificación
model.add(Dense(len(method_to_int), activation='softmax'))
```

Fuente: Elaboración propia.

6.3.8. Regularización y Compilación

En la fase actual del desarrollo de nuestro modelo, hemos centrado nuestra atención en dos componentes fundamentales: la regularización, con el propósito de evitar el sobreajuste, y la compilación, que establece los parámetros de cómo el modelo aprende y se optimiza durante el entrenamiento. A continuación, presentamos una descripción detallada de las estrategias y metodologías adoptadas para abordar eficientemente estos elementos esenciales:

Regularización:

- **Uso de Dropout:** Con el objetivo de mitigar el riesgo de sobreajuste, se integraron capas de Dropout en la arquitectura de nuestro modelo. El Dropout, una técnica de regularización bien establecida, funciona desactivando aleatoriamente un porcentaje determinado de neuronas durante cada iteración del proceso de entrenamiento. Esta estrategia impide que el modelo se vuelva excesivamente dependiente de cualquier conjunto específico de características o caminos neuronales, promoviendo así el aprendizaje de múltiples caminos redundantes y robustos para realizar predicciones. En la estructura de nuestro modelo, se incorporaron capas de Dropout sucesivamente a las capas LSTM

y, en algunos casos, también después de las capas densas intermedias. El ratio de neuronas desactivadas se mantuvo comúnmente en un rango de 0.2 a 0.5, equilibrando efectivamente la regularización sin comprometer significativamente la capacidad de aprendizaje del modelo.

Compilación del Modelo:

- **Elección del Optimizador:** Para la compilación de nuestro modelo, se seleccionó el optimizador *Adam* debido a su amplia aceptación y eficacia demostrada en una variedad de problemas en el campo del aprendizaje automático. Una de las características más destacadas de Adam es su capacidad para ajustar la tasa de aprendizaje de manera adaptativa a lo largo del proceso de entrenamiento. Este ajuste adaptativo facilita una optimización eficiente sin la necesidad de una intensiva afinación manual, haciéndolo un optimizador adecuado para una amplia gama de escenarios de aprendizaje automático. Su versatilidad y eficacia en la gestión de las tasas de aprendizaje lo convierten en una elección preferente para la compilación de modelos en diversos contextos de investigación y desarrollo.
- **Función de Pérdida:** En nuestro enfoque, se optó por utilizar la función de pérdida *categorical_crossentropy*, considerando la naturaleza del problema de clasificación multiclase con el que estamos trabajando. Esta función de pérdida es particularmente adecuada para situaciones donde las salidas del modelo son probabilidades. *Categorical_crossentropy* es eficaz para medir el rendimiento del modelo al comparar la distribución de probabilidad que el modelo predice con la distribución de probabilidad real, que está representada por las etiquetas de las clases. Esta comparación se centra en la precisión con la que el modelo es capaz de predecir la probabilidad correcta para cada clase en los datos, lo que es crucial en la clasificación multiclase.
- **Métricas de Evaluación:** Para el monitoreo del proceso de entrenamiento, se seleccionó la métrica de *accuracy* (precisión). Esta métrica es una de las más intuitivas y comúnmente empleadas en el campo del aprendizaje automático, proporcionando una medida clara y directa del rendimiento del modelo. La

precisión indica la proporción de predicciones correctas realizadas por el modelo en comparación con el número total de predicciones. Su amplia utilización se debe a su capacidad para ofrecer una comprensión rápida y efectiva de la eficacia general del modelo en la clasificación correcta de los datos.

Figura 31: Construcción del compilador del modelo

```
# Compilar el modelo incluyendo 'accuracy' en las métricas  
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Fuente: Elaboración propia.

Balance entre Aprendizaje y Generalización

La implementación de la técnica de Dropout, en conjunto con una selección cuidadosa de la función de pérdida y el optimizador, tiene como objetivo fundamental establecer un balance óptimo en el rendimiento del modelo. Este balance se orienta hacia dos aspectos cruciales: por un lado, se busca que el modelo aprenda de manera eficiente a partir de los datos de entrenamiento, lo cual se manifiesta en la minimización de la pérdida. Por otro lado, se enfatiza la importancia de la capacidad del modelo para generalizar adecuadamente a nuevos conjuntos de datos, lo cual se refleja en una mayor precisión en el conjunto de prueba. La correcta armonización de estos elementos es esencial para garantizar que el modelo no solo se ajuste efectivamente a los datos con los que ha sido entrenado, sino que también mantenga un alto grado de precisión y relevancia al ser aplicado a datos no vistos anteriormente.

Preparación para el Entrenamiento

Tras la compilación exitosa del modelo, este se encuentra preparado para entrar en la fase de entrenamiento. La compilación efectiva es un paso crucial, ya que garantiza que el proceso de entrenamiento se desarrolle de manera eficiente. Además, asegura que el modelo esté configurado adecuadamente para aprender patrones significativos de los datos, evitando así el riesgo de memorización excesiva. Este equilibrio es esencial para el rendimiento óptimo del modelo en aplicaciones del mundo real, donde la capacidad de generalización y adaptación a datos nuevos es fundamental.

6.3.9. Entrenamiento del Modelo

La fase de entrenamiento constituye el proceso esencial mediante el cual el modelo de clasificación secuencial adquiere conocimiento a partir de los datos. En esta etapa, el modelo realiza ajustes iterativos en sus parámetros internos con el objetivo primordial de minimizar el error en sus predicciones. Este proceso de optimización es fundamental para mejorar la precisión y la eficacia del modelo en la clasificación de datos. A continuación, se presenta una descripción detallada de cómo se implementó y ejecutó el entrenamiento del modelo:

Alimentación de Datos al Modelo

El conjunto de datos de entrenamiento, que incluye tanto las secuencias de entrada (X_{train}) como las etiquetas correspondientes (y_{train}), se alimenta al modelo. El modelo aprende al ajustar sus pesos para predecir la etiqueta de cada secuencia de entrada lo más precisamente posible.

Uso de Datos de Validación

De forma simultánea al proceso de entrenamiento, se emplea un conjunto de validación, designado como (X_{test} , y_{test}), para monitorear y evaluar el rendimiento del modelo. Esta práctica es fundamental, ya que proporciona información continua sobre la capacidad del modelo para generalizar a datos no previamente vistos. La utilización de este conjunto de validación es un componente esencial para garantizar que el modelo no está incurriendo en un sobreajuste con respecto a los datos de entrenamiento. Al evaluar cómo el modelo se desempeña con un conjunto de datos independiente, se puede obtener una perspectiva más objetiva y realista sobre su eficacia y potencial aplicabilidad en escenarios del mundo real.

Configuración del Proceso de Entrenamiento

El proceso de entrenamiento del modelo se estructura en un número específico de iteraciones, denominadas épocas. Durante cada una de estas épocas, el modelo procesa de manera completa el conjunto de datos de entrenamiento. Esta metodología permite que el modelo refine iterativamente sus parámetros internos a través de la exposición repetida a todo el conjunto de entrenamiento. Este enfoque secuencial

asegura una evolución sistemática y progresiva de la capacidad del modelo para aprender y adaptarse a las características inherentes a los datos.

Monitoreo con EarlyStopping

Con el objetivo de prevenir el fenómeno de sobreajuste durante el entrenamiento del modelo, se ha implementado una técnica conocida como EarlyStopping. Este método consiste en monitorear de manera continua la pérdida observada en el conjunto de validación. El entrenamiento se interrumpe automáticamente cuando no se percibe una mejora en esta métrica de pérdida tras un número predefinido de épocas. Esta estrategia es crucial para asegurar que el modelo no se ajuste excesivamente a los datos de entrenamiento, perdiendo así su capacidad de generalización.

Adicionalmente, se ha activado la opción `restore_best_weights` dentro de la funcionalidad de EarlyStopping. Esto implica que, al finalizar el entrenamiento prematuro, el modelo restaura automáticamente los pesos correspondientes a la época en la que se logró la menor pérdida de validación. Este enfoque garantiza que el modelo retenido es el que ha demostrado el mejor rendimiento en términos de generalización, según lo medido por la pérdida en el conjunto de validación.

Evaluación del Rendimiento

Durante la fase de entrenamiento del modelo, se lleva a cabo un monitoreo constante de las métricas de precisión y pérdida, tanto en los conjuntos de entrenamiento como en los de validación. Estas métricas son fundamentales para evaluar el rendimiento del modelo, ya que proporcionan información crítica sobre su capacidad para aprender de los datos y generalizar a nuevos conjuntos de datos. El análisis continuo de la precisión y la pérdida es esencial para identificar la eficacia con la que el modelo se está ajustando a los datos. Este proceso ayuda a detectar problemas potenciales, como el sobreajuste o el subajuste, y permite realizar ajustes oportunos en los parámetros del modelo o en la estrategia de entrenamiento para optimizar su rendimiento.

Figura 32: Código para entrenar el modelo

```
# Definir el callback de EarlyStopping para monitorear tanto la pérdida de validación como la precisión
early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=1, restore_best_weights=True)

# Ajustar y entrenar el modelo con el callback de EarlyStopping
history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=100,
    batch_size=32,
    callbacks=[early_stopping],
    verbose=1
)
```

Fuente: Elaboración propia.

6.3.10. Resultados del entrenamiento del modelo

Una vez finalizado el proceso de entrenamiento del modelo, procedimos a evaluar su rendimiento. Esta evaluación se basó en el análisis detallado de las métricas de pérdida y precisión, tanto en el conjunto de entrenamiento como en el conjunto de validación. Este enfoque nos permite comprender de manera integral cómo el modelo se comporta en diferentes conjuntos de datos, destacando tanto su eficacia en el aprendizaje como su capacidad de generalización a nuevos datos. La comparación de estas métricas entre los conjuntos de entrenamiento y validación es fundamental para identificar posibles problemas de sobreajuste o subajuste, y para asegurar la robustez y fiabilidad del modelo en aplicaciones prácticas.

Análisis de la Pérdida

El análisis de los gráficos revela que tanto la pérdida de entrenamiento (Loss) como la pérdida de validación (Val_Loss) experimentan una disminución significativa durante las etapas iniciales del proceso de entrenamiento. Este comportamiento indica que el modelo está aprendiendo de manera efectiva a partir de los datos proporcionados. Con el progreso de las épocas de entrenamiento, se observa que ambas curvas de pérdida tienden a estabilizarse y a converger. Dicha convergencia es indicativa de que el modelo ha alcanzado un estado en el cual puede realizar predicciones consistentes y confiables.

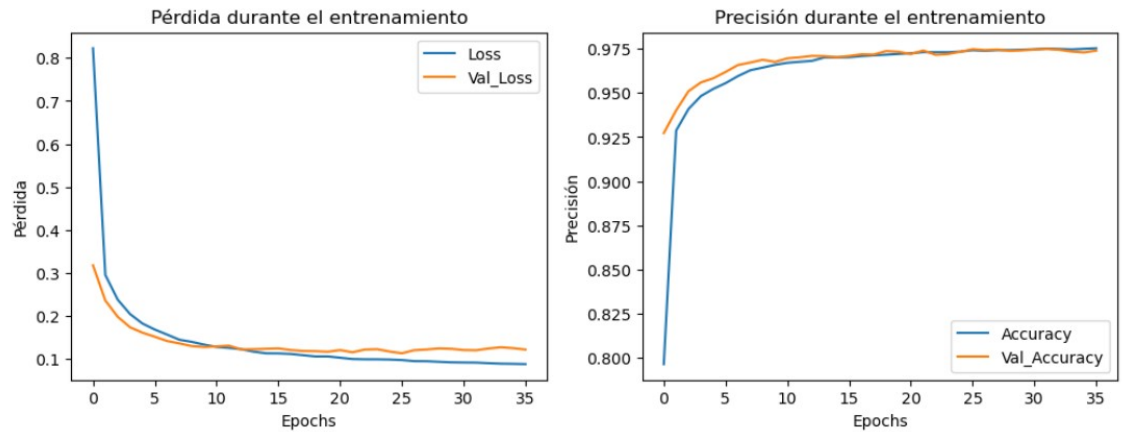
La proximidad en la trayectoria de las curvas de pérdida de entrenamiento y de validación es un indicador clave de que el modelo no está incurriendo en un sobreajuste. Esto se evidencia por el hecho de que la pérdida de validación no muestra un incremento o divergencia significativa respecto a la pérdida de entrenamiento, lo cual sería una señal clara de sobreajuste. Por tanto, estos resultados sugieren que el modelo ha logrado un equilibrio adecuado en su capacidad de generalización a partir de los datos de entrenamiento.

Análisis de la Precisión

En relación con la métrica de precisión, se observa que tanto la precisión durante el entrenamiento (Accuracy) como la precisión durante la validación (Val_Accuracy) muestran un incremento significativo en las fases iniciales, seguido de una fase de estabilización donde ambas métricas se mantienen en valores cercanos entre sí. Este patrón de comportamiento es indicativo de un alto grado de exactitud en las predicciones generadas por el modelo. Además, la proximidad entre la precisión de entrenamiento y la de validación sugiere que el modelo posee una buena capacidad de generalización frente a nuevos conjuntos de datos.

Esta tendencia en las métricas de precisión es un factor alentador, ya que implica que el modelo no solo es eficaz en el reconocimiento de patrones en los datos de entrenamiento, sino que también mantiene un rendimiento consistente cuando se expone a datos no vistos durante la fase de entrenamiento. Esta característica es esencial para la aplicabilidad práctica del modelo en entornos reales, donde la capacidad de generalizar a partir de datos nuevos y variados es crucial.

Figura 33: Gráficos de análisis del modelo de clasificación



Fuente: Elaboración propia.

6.3.11. Predicción del modelo

En la aplicación de nuestro modelo de clasificación secuencial para la generación de predicciones, se procesa una secuencia de acciones previas ejecutadas por un usuario. El modelo utiliza este historial para inferir cuál podría ser la siguiente acción del usuario. Un aspecto distintivo de este modelo es su capacidad no solo para identificar la acción subsiguiente más probable, sino también para cuantificar la confianza en dicha predicción, expresándola en términos de probabilidad.

Por ejemplo, consideremos un caso donde el historial de un usuario indica que está en proceso de recolección de información. Si el modelo anticipa que la acción siguiente será `getDetails()`, esta predicción se acompaña de una probabilidad asignada, por ejemplo, del 95 %. Tal porcentaje refleja la confianza del modelo en que `getDetails()` será la próxima acción del usuario. Esta métrica de confianza es fundamental para interpretar no solo las expectativas del modelo sobre eventos futuros, sino también el nivel de certeza asociado a estas expectativas.

En un escenario diferente, si un usuario ha estado interactuando con su cuenta y el modelo pronostica que la acción siguiente será `getSSContributionsCertificate()`, esta predicción puede presentar una probabilidad del 78.76 %. Este valor sugiere que,

aunque el modelo se inclina por `getSSContributionsCertificate` como la acción probable, existe una notable incertidumbre, posiblemente atribuible a la variabilidad en el comportamiento del usuario o a patrones ambiguos en sus acciones anteriores.

Estos ejemplos demuestran cómo el modelo va más allá de la toma de decisiones binarias, proporcionando un marco cuantitativo que resulta invaluable para la toma de decisiones basada en datos. Este enfoque permite a los sistemas automatizados o a los operadores humanos comprender con mayor profundidad y responder de manera más efectiva a las predicciones generadas por el modelo.

6.3.12. Métricas de predicción aplicadas

Explicaremos a continuación las métricas utilizadas para evaluar el rendimiento del modelo de clasificación, detallando también los resultados obtenidos y su interpretación:

Precisión (Precision)

La precisión constituye una métrica crítica en la evaluación de modelos de clasificación. Esta métrica proporciona una estimación de la fiabilidad de las predicciones positivas generadas por el modelo. En términos más concretos, la precisión representa el porcentaje de instancias clasificadas correctamente como positivas por el modelo, en relación con el total de instancias que el modelo ha identificado como positivas, particularmente en escenarios de clasificación multiclase.

Al referirse a una precisión del 57.94 % (o 0.5794 en formato decimal), se está indicando que, bajo las condiciones actuales de modelado, existe una probabilidad del 57.94 % de que una instancia clasificada por el modelo en una categoría específica sea una clasificación acertada. Esta interpretación de la precisión es crucial para comprender la eficacia del modelo en la identificación correcta de las categorías relevantes.

Recall (Sensibilidad o Tasa de Verdaderos Positivos)

El *recall* es una medida que indica qué proporción de instancias realmente positivas han sido identificadas correctamente por el modelo. Es decir, del total

de instancias que verdaderamente pertenecen a una clase, el *recall* muestra qué porcentaje ha sido reconocido de forma acertada por el modelo.

Con un *recall* de 0.5487 (o 54.87%), esto sugiere que el modelo logra identificar correctamente el 54.87% de todas las instancias positivas presentes en el conjunto de datos.

Puntuación F1

La puntuación F1 constituye una métrica integral en la evaluación del rendimiento de un modelo de clasificación. Esta métrica es particularmente relevante cuando se busca un equilibrio entre la precisión (*precision*) y el *recall*, siendo especialmente útil en contextos donde existe una distribución desigual de clases. Dicho desequilibrio se presenta cuando una clase es significativamente más frecuente que otra.

La puntuación F1 se define como la media armónica de la precisión y el *recall*. Esta media tiende a favorecer los valores menores dentro del conjunto, lo que implica que un rendimiento alto tanto en precisión como en *recall* es esencial para alcanzar una puntuación F1 elevada. La puntuación F1 varía entre 0 y 1, donde 1 representa la perfección y 0 la peor puntuación posible.

En el caso de nuestro modelo, con una puntuación F1 de 0.5366 (o 53.66%), se refleja un equilibrio moderado entre la precisión y el *recall*. Una puntuación de 53.66% sugiere un desempeño intermedio, indicando que el modelo no sobresale en ninguna de las dos métricas de manera excepcional, pero tampoco presenta deficiencias significativas. Sin embargo, se preferiría una puntuación F1 más alta, lo que indicaría una mayor armonía y efectividad en el equilibrio entre precisión y *recall*.

6.3.13. Conclusión del Modelo de Predicción Secuencial

El modelo de predicción de secuencias se ha desarrollado con el propósito de anticipar futuras acciones de usuarios basándose en patrones de comportamiento históricos. Aunque el modelo demuestra una capacidad moderada en la clasificación y predicción de estas acciones, evidenciado por una precisión del 57.94% y un *recall* del 54.87%, los resultados sugieren que existe un margen considerable para mejorar

tanto su precisión como su fiabilidad.

La puntuación F1 de 53.66 % refleja un equilibrio entre la precisión y el recall, pero también destaca la necesidad de optimizar el modelo para mejorar su rendimiento predictivo. Esto podría realizarse a través de la refinación de la arquitectura del modelo, la implementación de ajustes en la regularización o la utilización de un conjunto de datos más extenso o representativo.

A pesar de que el modelo actual constituye un punto de partida robusto para la predicción de comportamientos futuros basándose en datos históricos, es claro que para aplicaciones que requieren alta precisión o son críticas, se hace imprescindible una optimización y evaluación adicional. Este proceso de análisis y mejora continua es crucial, particularmente en contextos donde los errores en la predicción podrían tener implicaciones significativas.

Por lo tanto, aunque el modelo actual no alcanza aún los estándares requeridos para una predicción altamente precisa y confiable, representa un avance significativo hacia el entendimiento y la anticipación del comportamiento del usuario. La estrategia a futuro se centrará en la mejora y ajuste del modelo existente, así como en la exploración de nuevas técnicas y metodologías que se alineen más efectivamente con los objetivos de predicción del comportamiento.

CAPÍTULO 7: API Y MODELO IMPLEMENTADO

Se llevó a cabo la implementación de una API usando Flask, una librería de Python para crear aplicaciones web, y TensorFlow, una librería para el aprendizaje automático. La API está diseñada para realizar clasificación de secuencias, utilizando un modelo previamente entrenado y dos diccionarios JSON para el mapeo de datos.

7.1. Descripción de los componentes

Estos componentes trabajan juntos para permitir que la API reciba datos de entrada, los procese utilizando el modelo de clasificación y los mapeos, y luego devuelva una predicción con su probabilidad asociada.

Modelo de Clasificación con TensorFlow:

- **Uso de TensorFlow Keras:** El modelo está construido y entrenado utilizando TensorFlow, específicamente su API de alto nivel, Keras. TensorFlow es una plataforma de código abierto para aprendizaje automático, y Keras facilita la creación y entrenamiento de modelos de aprendizaje profundo.
- **Archivo .h5:** Este formato de archivo es específico para modelos de Keras y almacena la arquitectura del modelo, los pesos de las neuronas, y la configuración de entrenamiento. El modelo .h5 que se carga es un modelo pre entrenado y listo para realizar predicciones.

Diccionarios JSON para Mapeo de Datos:

- **Archivos `method_to_int.json` y `channel_to_int.json`:** Estos archivos contienen mapeos en formato JSON, que se utilizan para convertir los datos

categoricos de entrada (métodos y canales) en valores numéricos. Esto es necesario porque los modelos de aprendizaje automático requieren entradas numéricas.

- **Función de los Mapeos:** Cada método y canal se asigna a un número entero único. Por ejemplo, un método específico podría mapearse al número 3, y un canal al número 5. Estos mapeos son esenciales para procesar las entradas de manera que sean coherentes con el formato de datos usado para entrenar el modelo.

API de Flask

- **Flask como Marco de Trabajo:** Flask es un microframework para Python que permite desarrollar aplicaciones web de manera sencilla. En este caso, se utiliza para crear un servidor web que puede manejar solicitudes HTTP.
- **Ruta /predict:** Esta es una ruta definida en la aplicación Flask. Cuando se recibe una solicitud POST a esta ruta, la API procesa la solicitud para realizar una predicción utilizando el modelo cargado.
- **Manejo de Solicitudes y Respuestas:** Flask se encarga de interpretar las solicitudes HTTP, extraer los datos JSON enviados, y luego devolver una respuesta en formato JSON después de realizar la predicción.

7.2. Desarrollo e Implementación de la API

El proceso de creación de nuestra API inicia con la configuración del entorno de desarrollo, utilizando Python, un lenguaje de programación ampliamente reconocido, junto con diversas bibliotecas especializadas. Inicialmente, se importa Flask, una herramienta eficaz para la creación de aplicaciones web, que facilita la gestión de solicitudes a través de internet. Además, se integra TensorFlow, una biblioteca de vanguardia para el manejo de aprendizaje automático, que será fundamental en la utilización de un modelo de clasificación, es decir, un sistema entrenado para categorizar datos.

El modelo de clasificación se almacena en un archivo con formato `.h5`, que preserva íntegramente la estructura y datos del modelo. Este se carga mediante TensorFlow, preparando el modelo para realizar predicciones efectivas.

Adicionalmente, la API opera con dos archivos en formato JSON, `“method_to_int.json”` y `“channel_to_int.json”`. Este formato estructura la información de manera clara, facilitando su interpretación tanto por humanos como por sistemas automatizados. Los archivos contienen mapeos, similares a diccionarios, que permiten convertir texto en valores numéricos, adaptándose a las limitaciones del modelo que no procesa texto directamente.

Posteriormente, se define una ruta especial `“predict”` en la aplicación Flask. Las solicitudes recibidas en esta dirección son procesadas en formato JSON, utilizando los mapeos mencionados y ajustando la longitud de los datos con `“pad_sequences”` para cumplir con las especificaciones del modelo.

Los datos procesados se introducen en el modelo de clasificación para su análisis, generando predicciones sobre la categoría a la que pueden pertenecer. El modelo proporciona estimaciones con un grado de confianza asociado a cada predicción.

Finalmente, la API responde a las solicitudes, entregando en formato JSON la categoría predicha junto con la confianza del modelo en dicha predicción, asegurando una interpretación y uso sencillos y eficientes.

Figura 34: Código creación API

```
from flask import Flask, request, jsonify
import tensorflow as tf
from tensorflow.keras.preprocessing.sequence import pad_sequences
import numpy as np
import json
import os

app = Flask(__name__)

# Cargar el modelo entrenado
model = tf.keras.models.load_model('modelo_de_clasificacion_para_prediccion.h5')

# Cargar los mapeos. Asegúrate de que los archivos 'method_to_int.json' y 'channel_to_int.json' existen.
with open('method_to_int.json', 'r') as f:
    method_to_int = json.load(f)

with open('channel_to_int.json', 'r') as f:
    channel_to_int = json.load(f)

# Definir la longitud máxima de secuencia basada en el entrenamiento previo
max_sequence_length = 20 # Este valor debe coincidir con el utilizado durante el entrenamiento

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json()

    # Codificar la secuencia de entrada usando los mapeos
    try:
        secuencia_codificada = [[method_to_int[m['metodo']], channel_to_int[m['canal']]] for m in data]
        secuencia_entrada = pad_sequences([np.ravel(secuencia_codificada)], maxlen=100, padding='pre', dtype='int16')

        # Hacer la predicción
        prediccion = model.predict(secuencia_entrada)
        indice_metodo_predicho = np.argmax(prediccion, axis=1)[0]
        metodo_predicho = {value: key for key, value in method_to_int.items()}[indice_metodo_predicho]
        probabilidad_maxima = np.max(prediccion, axis=1)[0]

        # Enviar la respuesta
        respuesta = {
            'metodo_predicho': metodo_predicho,
            'probabilidad': float(probabilidad_maxima)
        }
        return jsonify(respuesta)
    except KeyError as e:
        return jsonify({'error': str(e)}), 400

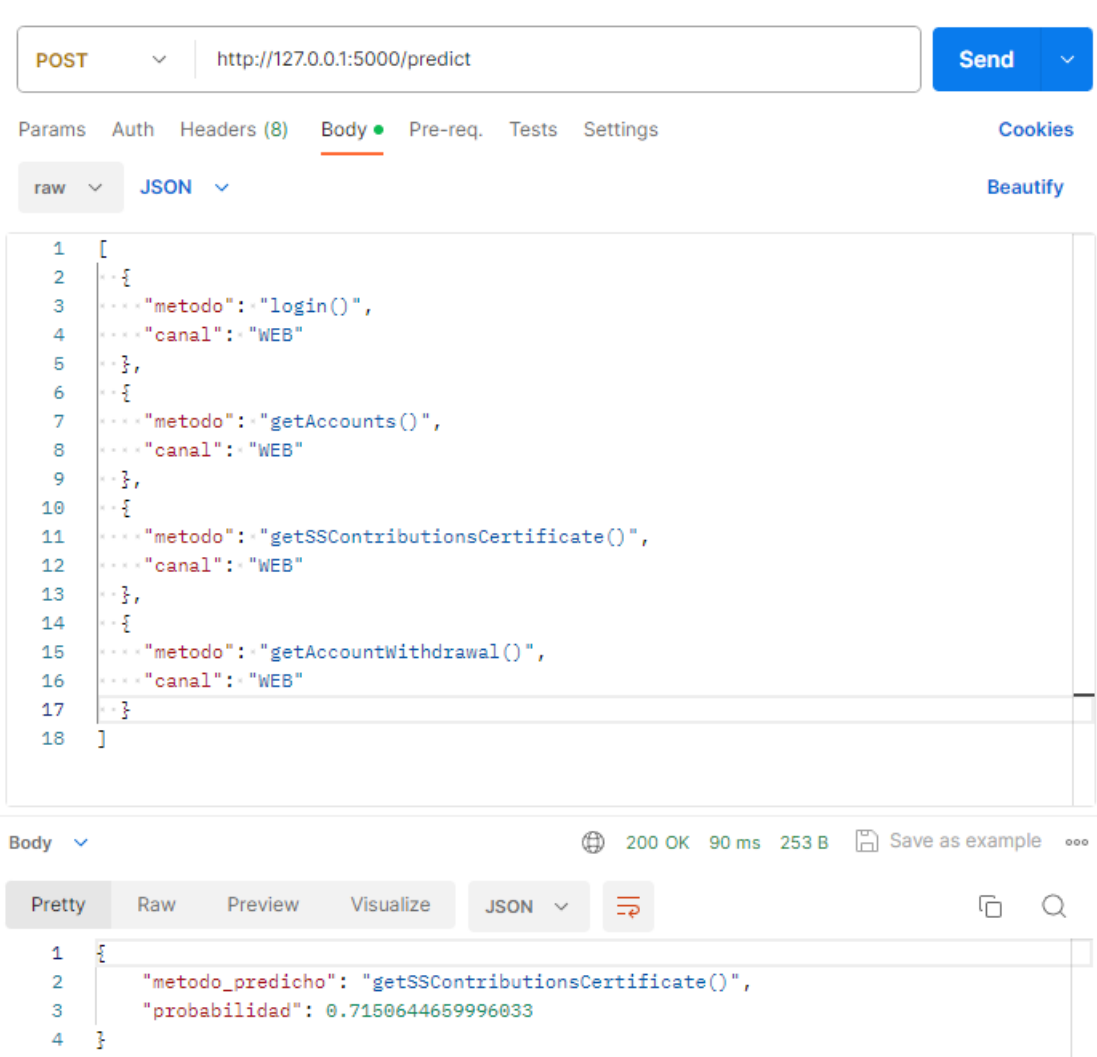
if __name__ == '__main__':
    app.run(debug=True)
```

Fuente: Elaboración propia.

Para garantizar un funcionamiento óptimo, se ejecuta la aplicación Flask en un modo de "depuración", que facilita la identificación y resolución rápida de problemas, mejorando así la estabilidad y fiabilidad del sistema. Además de contar con códigos de estatus para saber como va funcionando la API, el código 200 representa el correcto funcionamiento y cuando se presenta algún error en el ingreso de parámetros la API no

funcionara y se desplegara el código 400 que representa el incorrecto funcionamiento, junto a la descripción de la causa del error.

Figura 35: Prueba de funcionamiento API de predicción en Postman



Fuente: Elaboración propia.

7.3. Dockerización de la Api

Para tener un buen desarrollo y despliegue del proyecto se decidió por utilizar la herramienta **Docker**, esta herramienta nos permite trabajar con contenedores que

empaquetan la aplicación desarrollada, los distintos archivos de código, conexiones a bases de datos, entornos virtuales (.env), archivos de configuración, dependencias, etc (*Docker docs*, 2023). Estos contenedores de docker se almacenan en repositorios de contenedores (parecido a Github), permitiendo ser portable entre distintos desarrolladores a través de una imagen basada en linux, esta imagen contiene todos los archivos relevantes de la aplicación y es ejecutada en un contenedor de docker. Los contenedores son varias de estas imágenes juntas, por lo general, la primera capa corresponde a una distribución de linux. Los contenedores virtualizan la aplicación y utilizan el kernel del sistema operativo anfitrión, reduciendo el tamaño y tiempo de ejecución de la aplicación en gran manera.

Para empaquetar la aplicación en un contenedor de docker se siguieron los siguientes pasos:

- Primero creamos un archivo **Dockerfile**, el cual será usado para crear la imagen de la aplicación, en este archivo especificamos la imagen de la versión de Python a utilizar, en este caso la 3.9 para permitir el correcto funcionamiento de las otras dependencias, copiamos el archivo que contiene los requerimientos de la aplicación y luego se ejecuta una línea que instala estos requerimientos, para contener todos los archivos de la aplicación, se copia la ruta de la carpeta y todo su contenido. Se especifica el puerto al cual docker enlazará el contenedor creado y se escriben los comandos necesarios para ejecutar la aplicación. Todos estos comandos serán ejecutados cuando se cree un nuevo contenedor.

Figura 36: Código archivo Dockerfile

```
Dockerfile > ...
1  FROM python:3.9
2
3  COPY requirements.txt .
4
5  RUN pip install --no-cache-dir --upgrade -r requirements.txt
6
7  COPY ./app .
8
9  EXPOSE 5000
10
11 CMD [ "flask", "run", "--host=0.0.0.0", "--port=5000" ]
```

Fuente: Elaboración propia.

- Como segundo paso, se crea un archivo **.dockerignore** en el cual se especifican todas las carpetas y archivos que no deben de ser considerados en la creación de la imagen de docker.
- El ultimo paso en la dockerización de la aplicación, es la creación de un archivo **docker-compose.yml** el cual puede contener y gestionar varios servicios de docker. En este caso solo contendrá un servicio, este servicio corresponde a la aplicación que se busca contener, se especifican los nombres del contenedor e imagen a crear, el puerto a ocupar y como se creará el contenedor de docker, indicando el contexto y nombre del archivo para crear la imagen, en este caso Dockerfile.

Figura 37: Código archivo docker-compose.yml

```
🐳 docker-compose.yml
1  version: "3.7"
2
3
4  services:
5    mlapp:
6      container_name: mlapp
7      image: methodpred
8      ports:
9        - "5000:5000"
10     build:
11       context: .
12       dockerfile: Dockerfile
```

Fuente: Elaboración propia.

Con los pasos anteriormente mencionados realizados, se puede proceder a ejecutar el comando “docker-compose up --build”, este comando procedera a crear y ejecutar un contenedor basado en la imagen especificada. Con esto se asegura un despliegue de la aplicación sin problemas de rendimiento, tamaño de módulos o discrepancias en las versiones de los módulos.

CAPÍTULO 8: CONCLUSIÓN Y RECOMENDACIÓN

8.0.1. Conclusión

El desarrollo de este proyecto de tesis ha abarcado una gama integral de procesos y análisis, comenzando con un estudio bibliométrico y extendiéndose a través del proceso ETL (Extracción, Transformación y Carga), el Análisis Exploratorio de Datos (EDA) y la implementación y evaluación de modelos predictivos.

El estudio bibliométrico proporcionó una base sólida y rigurosa, permitiendo una comprensión profunda del estado actual del campo, identificando las tendencias predominantes y las áreas menos exploradas en la literatura. Este análisis ayudó a orientar el enfoque del proyecto hacia áreas de investigación que son tanto innovadoras como de relevancia práctica.

En la fase de ETL, se trabajó meticulosamente para asegurar la calidad y la integridad de los datos. Este proceso fue crucial para preparar un conjunto de datos coherente y fiable, lo que sentó las bases para un análisis más detallado y preciso. La limpieza, transformación y enriquecimiento de los datos fueron pasos fundamentales que garantizaron la confiabilidad de las conclusiones y recomendaciones posteriores.

El Análisis Exploratorio de Datos (EDA) permitió una comprensión más profunda de los datos a mano. A través de este análisis, se identificaron patrones, anomalías y relaciones clave que influyeron significativamente en la selección y el ajuste de los modelos predictivos. Esta fase fue vital para la comprensión de las dinámicas subyacentes de los datos y para formular hipótesis informadas para la modelización.

Finalmente, la fase de modelado y pruebas fue el núcleo del proyecto. La aplicación de modelos predictivos, ajustados y evaluados cuidadosamente, proporcionó

insights valiosos y respuestas a las preguntas de investigación planteadas. La evaluación de estos modelos, utilizando métricas como la precisión, el recall y la puntuación F1, reveló un rendimiento moderado, indicando áreas para futuras mejoras y ajustes. Esta fase no solo validó el enfoque metodológico empleado, sino que también abrió caminos para futuras investigaciones y desarrollos.

En conjunto, este proyecto ha demostrado la importancia de un enfoque metódico y bien estructurado en la investigación. Desde la recopilación inicial de datos hasta el análisis final y la interpretación de los resultados, cada etapa ha sido crucial para construir un entendimiento completo del tema en cuestión. Las lecciones aprendidas y los conocimientos adquiridos durante este proyecto no solo son valiosos para este campo de estudio, sino que también proporcionan un marco para futuras investigaciones y desarrollos en áreas relacionadas.

8.0.2. Recomendaciones

A través del desarrollo de este proyecto, se han identificado varias áreas clave en las que se pueden realizar mejoras y avances. Estas recomendaciones están orientadas a optimizar el modelo actual y a expandir el alcance de la investigación futura:

Optimización del Modelo Actual:

- **Aumentar el Conjunto de Datos:** Un mayor volumen de datos puede mejorar la capacidad del modelo para aprender y generalizar. Esto es especialmente relevante en modelos basados en aprendizaje profundo que requieren grandes cantidades de datos para un rendimiento óptimo.
- **Exploración de Hiperparámetros:** Ajustar los hiperparámetros del modelo, como la tasa de aprendizaje, el número de capas ocultas y la cantidad de nodos en cada capa, puede resultar en mejoras significativas en el rendimiento.
- **Incorporación de Regularización:** Implementar técnicas de regularización como Dropout o L1/L2 puede ayudar a prevenir el sobreajuste y mejorar la generalización del modelo en datos no vistos.

- **Predicción en base al Rut:** Ajustar el modelo para que sea capaz de recibir el Rut de un usuario y que realice la predicción en base a su comportamiento previo registrado en la página web.

Experimentación con Diferentes Arquitecturas de Modelos:

- **Modelos de Aprendizaje Profundo Avanzados:** Probar con arquitecturas más complejas, como Redes Neuronales Recurrentes (RNN), Transformers o modelos basados en atención, podría capturar mejor las relaciones temporales y contextuales en los datos.
- **Enfoque Híbrido:** Combinar modelos de aprendizaje automático tradicionales con técnicas de aprendizaje profundo puede proporcionar una visión más holística y mejorar la precisión de las predicciones.

Análisis de Características Adicionales:

- **Ingeniería de Características:** Explorar nuevas características derivadas de los datos existentes puede revelar patrones ocultos y mejorar la capacidad predictiva del modelo.
- **Integración de Datos Externos:** Incorporar datos externos, como tendencias de mercado o indicadores económicos, podría proporcionar contextos adicionales que mejoren la precisión predictiva.

Evaluación Continuada y Ajuste del Modelo:

- **Monitoreo de Desempeño en Tiempo Real:** Establecer un sistema de monitoreo para evaluar el desempeño del modelo en tiempo real puede ayudar a identificar rápidamente áreas de mejora y ajustar el modelo de manera proactiva.
- **Actualización de Datos y Modelos:** Regularmente actualizar el modelo con nuevos datos para mantener su relevancia y precisión frente a patrones cambiantes y tendencias emergentes.

Investigaciones Futuras:

- **Estudio de Casos Específicos:** Investigar en profundidad casos específicos donde el modelo no se desempeñó según lo esperado puede proporcionar insights valiosos para futuras mejoras.
- **Expansión a Nuevos Dominios:** Aplicar el modelo a diferentes conjuntos de datos o en distintos contextos podría ayudar a evaluar su robustez y adaptabilidad.

Estas recomendaciones están destinadas a guiar los esfuerzos futuros hacia la mejora continua del modelo y la expansión del conocimiento en este campo de estudio. La implementación exitosa de estas sugerencias requiere un enfoque iterativo y adaptable, asegurando que el modelo se mantenga relevante y efectivo frente a las cambiantes dinámicas de datos y requerimientos del mundo real.

REFERENCIAS

- Afp capital*. (2023). Sitio web. Descargado de <https://www.afpcapital.cl/Paginas/default.aspx>
- Ajitesh, K. (2023). *Different types of time-series forecasting models*. Sitio web. Descargado de <https://vitalflux.com/different-types-of-time-series-forecasting-models/>
- Amazon. (2023). *¿qué es una red neuronal?* Sitio Web. Descargado de <https://aws.amazon.com/es/what-is/neural-network/#:~:text=Una%20red%20neuronal%20es%20un,lo%20hace%20el%20cerebro%20humano.>
- Arana, C. (2021). *Modelos de aprendizaje automático mediante árboles de decisión*.
- Baker, K. (2023). *What is attribution modeling and why its so important*. Sitio web. Descargado de <https://blog.hubspot.com/marketing/attribution-modeling>
- Contreras Arteaga, A., y Sánchez Cotrina, F. (2019). *Analítica predictiva para conocer el patrón de consumo de los clientes en la empresa cienpharma s.a.c utilizando ibm spss modeler y la metodología crisp-dm*. (p.15)
- dask: Scale the python tools you love*. (2023). Sitio Web. Descargado de <https://www.dask.org/>
- Docker docs*. (2023). Sitio web. Descargado de <https://docs.docker.com/>
- Elizabeth, F. (2023). *Cómo crear un modelo de recomendación basado en machine learning*. Sitio web. Descargado de <https://aws.amazon.com/es/blogs/aws-spanish/como-crear-un-modelo-de-recomendacion-basado>

-en-machine-learning/

Eric. (2023). *Data science life cycle: Crisp-d, and osemn frameworks*. Sitio web.

Descargado de <https://datarundown.com/data-science-life-cycle/>

Filzinger, T. (2023). *Lstm long short-term memory - memoria a corto plazo de*

larga duración. Sitio Web. Descargado de [https://konfuzio.com/es/lstm/#:~:text=La%20LSTM%20es%20un%20tipo,periodos%20de%20tiempo%20m%C3%](https://konfuzio.com/es/lstm/#:~:text=La%20LSTM%20es%20un%20tipo,periodos%20de%20tiempo%20m%C3%A1s%20largos)

[A1s%20largos](https://konfuzio.com/es/lstm/#:~:text=La%20LSTM%20es%20un%20tipo,periodos%20de%20tiempo%20m%C3%A1s%20largos).

Garcia-Estrella, C., y Barón Ramírez, E. (2021). *La inteligencia de negocios y la analítica de datos en los procesos empresariales*.

Google colab. (2023). Sitio web. Descargado de [https://colab.research.google](https://colab.research.google.com/?hl=es#scrollTo=5fCEDCU_qrC0)

[.com/?hl=es#scrollTo=5fCEDCU_qrC0](https://colab.research.google.com/?hl=es#scrollTo=5fCEDCU_qrC0)

Hyndman, R. J., y Athanasopoulos, G. (2023). *Forecasting: principles and practice, 2nd edition*. Sitio Web. Descargado de 0Texts.com/fpp2

IBM. (2023). *¿qué es el random forest?* Sitio web. Descargado de <https://www.ibm.com/mx-es/topics/random-forest>

Kimball, R., y Caserta, J. (2004). *The data warehouse etl toolkit*. Wiley Publishing, Inc.

Lemon, K. N., y Verhoef, P. C. (2016). Understanding customer experience throughout the customer journey. *Journal of Marketing*.

Mckinney, W. (2011). pandas: a foundational python library for data analysis and statistics. *Python for high performand and scientific computing*, 14(9), 1-9.

Morales Oñate, V. (2022). *Series de tiempo*. Sitio Web. Descargado de https://bookdown.org/victor_morales/SeriesdeTiempo/%7D%7D

Parra, J. (2002). Análisis exploratorio y análisis confirmatorio de datos. *Espacio Abierto*, 11(1), 115-124.

- Rasekhi, S., Fard, H., y Kim, D. (2016). *Understanding the role of trust in pension system: A literature review*. The Third Wave. William Morrow and company Inc.
- Rodriguez, J. (2023). *¿qué es un cliente?* Sitio web. Descargado de <https://blog.hubspot.es/sales/que-es-un-cliente>
- Ruiz, N. (2022). *Una aproximación al análisis exploratorio de datos* (Tesis Doctoral no publicada). Universidad de Valladolid.
- Sibila, I., Adriana, L., y Vlatko, L. (2020). *Logistic regression based in-service assessment of mobile web browsing service quality acceptability*. Sitio web. Descargado de <https://jwcn-urasipjournals.springeropen.com/articles/10.1186/s13638-020-01708-2>
- Sánchez, W. (2011). La usabilidad en ingeniería de software: definición y características. *Revista de Ingeniería e Innovación de la Facultad de Ingeniería, Universidad Don Bosco*, 1(2), 7-21.
- Toffler, A. (1980). *Understanding the role of trust in pension system: A literature review*.
- van der Walt, S., Colbert, S. C., y Varoquaux, G. (2011). The numpy array: A structure for efficient numerical computation. *Computing in Science and Engineering, Institute of Electrical and Electronics Engineers*, 13(2), 22-30. doi: 10.1109/MCSE.2011.37
- van Rossum, G., y Drake, F. L. (2003). *An introduction to python*. Network Theory Limited.
- Vatsal. (2021). *Recommendation systems explained*. Sitio web. Descargado de <https://towardsdatascience.com/recommendation-systems-explained-a42fc60591ed>

Zheng, B., Thompson, K., Lam, S. S., Yoon, S. W., y Gnanasambandam, N. (2013).

Customers' behavior prediction using artificial neural network.

ANEXOS

Anexo 1. Listado de métodos en la plataforma

Tabla Anexo.1 – Tabla de métodos del canal.

Método	Descripción
login()	Login a app o web
getMensualpdf()	Obtiene cartola mensual
getAccountWithdrawal()	Obtiene cuentas para realizar giros APV y CAV
getAccounts()	Obtiene cuentas del cliente
getProfitabilityVSContributions()	Información de aportes versus rentabilidad
getProfitabilityAccountByPeriod()	Información de rentabilidad por periodo asociada a cartola mensual
getApplications()	Lista solicitudes de cambio y distribución de fondos
getSocialSecurityContribution()	Obtiene cotizaciones por producto de los clientes
getAccountsConditions()	Obtiene precondiciones para realizar cambio/distribución de fondos
getRecognitionBonds()	Obtiene bono de reconocimiento
getShareValues()	Obtiene valores cuota
obtenerSalDOS()	Obtiene saldos del cliente
obtenerBeneficiarios()	Obtiene beneficiarios registrados por el cliente
datosPersonales()	Obtiene datos personales del cliente
getBankAccount()	Obtiene lista de bancos
getDatosPersonales()	Obtiene información personal del cliente
getDatosRegiones()	Obtiene listado de regiones
getDatosPaíses()	Obtiene listado de países
getDatosComunas()	Obtiene el listado de comunas
validateSecurityKey()	Valida la clave de seguridad de previred
getCuentas()	Obtiene lista de cuentas del cliente

Continúa en la siguiente página

Tabla Anexo.1 – Continuación de la página anterior

Método	Descripción
conditionSecurityKey()	Obtiene el estado de actual de la clave de seguridad
getMensual()	Obtiene información de la cartola mensual
getCertificadoCotizaciones()	Genera el PDF del certificado de cotización según el rango de fechas
getPeriodosCuatrimestral()	Lista de periodos de consulta de la cartola cuatrimestral
getProfitability()	Consulta colección de firebase de configuraciones el documento del listado de validaciones de seguridad
getEnviarCuatrimestral()	Obtiene la cartola cuatrimestral
getCertificadoAfiliacion()	Obtiene certificado de afiliación
getCertificadoRemuneraciones()	Obtiene certificado de remuneraciones
omnichannelLogin()	Autentifica Capa Omnicanal AFP
RestructureFundsBusiness.restructureFunds()	Obtiene comprobante en PDF de cambio/distribucion de fondos
restructureFunds()	Graba cambio/distribución de fondos
getCertificadoResidencia()	Obtiene certificado de residencia
saveSurvey()	Guarda encuesta
validateTemporalPassword()	Valida clave dinámica
generateTemporalPassword()	Solicita clave dinámica
getCertificadoAntecedentes()	Obtener certificado antecedentes previsionales
getCertificadoVacaciones()	Obtener certificado vacaciones progresivas
getCertificadoTrabajosPesados()	Obtener certificado trabajos pesados
generarContacto()	Enviar formulario de contacto a ejecutivos
editarDatosPersonales()	Actualizar datos personales
sendSms()	Enviar SMS
getCertificadoComprobante()	Obtener certificado comprobante de pago pensión
getCertificadoPensionadosPagados()	Obtener certificado de pensiones pagadas
withdrawalRequest()	Grabar giro APV/CAV cliente

Continúa en la siguiente página

Tabla Anexo.1 – Continuación de la página anterior

Método	Descripción
getBankAccounts()	Obtener cuentas bancarias
validarDatosContacto()	Validar datos de contacto (Si están duplicados)
validarListaNegra()	Validar listado de cuentas email/celular no validas (email y celular)
addBankAccount()	Agregar cuenta bancaria de cliente
obtenerEstadoCuentaCorriente()	Obtener estado de cuenta corriente (Banco Santander)
validarCuentasDigitales()	Validar cuentas digitales
getTypesBanksAccounts()	Obtener tipos de cuentas bancarias
sendEmail()	Enviar email
getCertificadoNoCotizados()	Obtener certificado de periodos no cotizados
deleteBankAccount()	Eliminar cuenta bancaria de cliente
addBeneficiarios()	Agrega beneficiarios de pensión del cliente
changePassword()	Cambia la clave de seguridad
getAttentionVideo()	Obtiene el agendamiento de video atención
getCertificadoIngresoBase()	Obtiene el certificado de ingreso base para pensionados
getChangesAndDistributionsOfFunds()	Obtiene los cambios y distribuciones de fondos realizados
getDetailRequestStateChangeFunds()	Detalle estado de solicitudes de cambio y distribución de fondos
getPensionBackgroundCert()	Certificado de antecedentes previsionales para la APP
getPensionsCertificate()	Obtiene el certificado de pensiones
getRemunerationsCert()	Obtiene el certificado de remuneraciones imponibles
getSSContributionsCertificate()	Obtiene el certificado de cotizaciones
getSimulationWithdrawalCAV()	Realizar la simulación del giro de Cuenta 2
getWithdrawsAPV()	Resumen de estado de solicitudes APV
obtenerCodigoParentesco()	Obtiene los códigos de parentesco para el ingreso de beneficiarios

Continúa en la siguiente página

Tabla Anexo.1 – Continuación de la página anterior

Método	Descripción
obtenerFichaCalculo()	Obtiene el certificado de ficha de cálculo
obtenerSaldos()	Obtiene los saldos del cliente
requestAfpPassword()	Solicitud cambio de password
sendComplaintForm()	Envío de reclamo
basicLogin()	Datos básicos del cliente
crearClave()	Crea la clave de acceso
getBeneficiarios()	Obtiene los beneficiarios ingresados por el cliente
getCertificadoTributario()	Obtiene los certificados tributarios
getContactCategories()	Obtiene las categorías del formulario de contacto
getMembershipCert()	Obtiene el certificado de afiliación
getPensionTramite()	Obtiene el certificado de pensión en trámite
getPersonStatus()	Indica si es cliente nuevo y/o activo, si tiene password de 4 o 6 digitos
getRetireeQualityCertificate()	Certificado de calidad de pensionado en la app
getSimulationWithdrawalAPV()	Realizar la simulación del giro de APV
getVacationsCert()	Obtiene el certificado de vacaciones progresivas
getWithdrawsCAV()	Resumen de estado de solicitudes Cuenta 2
obtenerEstadoClave()	Obtiene el estado de la clave de seguridad
obtenerListaParentescoFormulario()	Obtiene la lista de parentescos del formulario de beneficiarios
reCaptcha()	Ejecuta el recaptcha de google
sendComplaintFile()	Adjunta documentos al formulario de reclamo desde el front
sendContactCases()	Solicitud de contacto
cambiarClaveAcceso()	Cambia la clave de 6 dígitos del cliente
existeFichaCalculo()	Busca si existe la ficha de cálculo
getRegiones()	Obtiene la lista de regiones
loginApplication()	Recupera token de sesion de apiconnect
recuperarClave()	Recupera la clave de seguridad
sendEmailPubsub()	Envío de email