FACULTAD DE ESTUDIOS ESTADÍSTICOS

MÁSTER EN MINERÍA DE DATOS E INTELIGENCIA DE NEGOCIOS

Curso 2023/2024

Trabajo de Fin de Máster

TITULO: ANÁLISIS DE INTENCIÓN DE COMPRA EN CANALES DIGITALES MEDIANTE TÉCNICAS DE EXPLAINABLE ARTIFICIAL INTELLIGENCE (XAI)

Alumno: Carlos Iván Martínez Gaona

Tutor: Ramón Alberto Carrasco González

Julio de 2024

UNIVERSIDAD COMPLUTENSE
MADRID

Autorización de difusión

Carlos Iván Martínez Gaona

28 de junio 2024

El abajo firmante, matriculado en el Máster en Minería de Datos e Inteligencia de Negocios, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: “ANÁLISIS DE INTENCIÓN DE COMPRA EN CANALES DIGITALES MEDIANTE TÉCNICAS DE EXPLAINABLE ARTIFICIAL INTELLIGENCE (XAI)”, realizado durante el curso académico 2023-2024 bajo la dirección de Ramón Alberto Carrasco González en el Departamento de Estadística y Ciencia de los Datos, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Resumen en castellano

En el contexto del comercio electrónico en expansión, entender y predecir la intención de compra es crucial para personalizar campañas y aumentar las conversiones. Sin embargo, la opacidad de los modelos “caja negra” dificulta el análisis detallado y las decisiones informadas. Este estudio utiliza técnicas de Explainable Artificial Intelligence (XAI) para abordar estos desafíos en el marketing digital, enfocándose en identificar los factores que influyen en las intenciones de compra en línea. Los objetivos incluyen construir modelos predictivos mediante validación cruzada y una metodología basada en KDD, aplicando técnicas globales de XAI para identificar variables clave y métodos locales para explicar casos específicos. Se presenta un modelo XGBoost con un umbral de 0.4, alcanzando un ROC AUC de 0.9268 y una precisión del 0.9007 en los datos de prueba. Los resultados subrayan la importancia de métricas como PageValues y ExitRates de Google Analytics, variables temporales como el mes y la actividad del usuario en páginas de productos relacionados, como elementos críticos para entender las decisiones de los usuarios. Basado en estos hallazgos, se recomienda mejorar el contenido y la funcionalidad de las páginas web, optimizar páginas clave para retener usuarios, intensificar campañas durante períodos de alta conversión, y mejorar la interfaz del cliente para una experiencia más satisfactoria.

Palabras clave

Explainable Artificial Intelligence (XAI), Interpretabilidad, Marketing Digital, Machine Learning (ML), Modelos de Caja Negra, Extreme Gradient Boosting (XGBoost), SHapley Additive exPlanations (SHAP), Local Interpretable Model-agnostic Explanations (LIME), Partial Dependence Plots (PDP), Clasificación Binaria

Abstract

In the context of expanding e-commerce, understanding and predicting purchase intent is crucial for customizing campaigns and increasing conversions. However, the opacity of “black box” models complicates detailed analysis and informed decision-making. This study employs Explainable Artificial Intelligence (XAI) techniques to address these challenges in digital marketing, focusing on identifying factors influencing online purchase intentions. Objectives include building predictive models through cross-validation and a KDD-based methodology, applying global XAI techniques to identify key variables, and using local XAI methods to explain specific cases. An XGBoost model with a threshold of 0.4 is presented, achieving a ROC AUC of 0.9268 and a precision of 0.9007 on test data. Results underscore the importance of metrics like Google Analytics’ PageValues and ExitRates, temporal variables such as month, and user activity on related product pages as critical elements in understanding user decisions. Based on these findings, recommendations include enhancing website content and functionality, optimizing key pages for user retention, intensifying campaigns during peak conversion periods, and improving the client interface for a more satisfactory experience.

Keywords

Explainable Artificial Intelligence (XAI), Interpretability, Digital Marketing, Machine Learning (ML), Black-box Models, Extreme Gradient Boosting (XGBoost), SHapley Additive exPlanations (SHAP), Local Interpretable Model-agnostic Explanations (LIME), Partial Dependence Plots (PDP), Binary Classification

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Motivación

Los métodos de caja negra (*black box*) son modelos de aprendizaje automático complejos y sofisticados cuya lógica subyacente es difícil de entender para los expertos en el dominio de la aplicación [8]. A pesar de su capacidad para capturar patrones no lineales y relaciones entre variables que las técnicas tradicionales no pueden detectar, y de su buen desempeño demostrado en competiciones [9], la opacidad de estos modelos plantea un desafío significativo en términos de interpretación.

En el contexto del marketing, estas técnicas se utilizan ampliamente para varias tareas como el estudio del éxito de campañas [10], la retención y la gestión de relaciones con los clientes [1]. Sin embargo, la dificultad para entender estos modelos complica la interpretación de las razones detrás de las decisiones a nivel local y el impacto de una o más variables en la predicción a nivel global. Esto limita la capacidad del usuario para realizar análisis detallados y tomar decisiones informadas basadas en los resultados de los modelos [4].

Estos desafíos son especialmente relevantes en el sector de las compras en línea, donde, según estudios como [11], en 2023 las ventas minoristas globales de comercio electrónico alcanzaron un estimado de 5.8 billones de dólares estadounidenses y las proyecciones indican un crecimiento del 39 % en los próximos años, con expectativas de superar los 8 billones de dólares para 2027. En este contexto, el análisis de la predicción en tiempo real de la intención de compra y la probabilidad de abandono de un sitio web son aspectos vitales para las tiendas en línea. La capacidad de predecir estos comportamientos permite personalizar promociones y contenidos, reducir las tasas de abandono del carrito y aumentar las tasas de conversión, optimizando las estrategias de marketing y aumentando las ventas [12].

Para abordar estos desafíos, se ha desarrollado la Inteligencia Artificial Explicable (**XAI**) , que proporciona explicaciones comprensibles tanto a nivel global como local. Esto facilita el acceso a aspectos específicos del sistema para desarrolladores, diseñadores, reguladores y usuarios finales [13]. Los métodos agnósticos de modelos, que son

técnicas de XAI aplicables a cualquier modelo de IA tras su entrenamiento, ofrecen interpretaciones de decisiones sin depender de los detalles internos del modelo. Esto permite a los expertos en dominios de aplicación obtener interpretaciones claras y naturales de los resultados sin necesidad de entender la complejidad matemática, mejorando la transparencia, confianza y efectividad en el uso de los modelos de IA [4].

1.2. Objetivos

Los objetivos de esta investigación se dividen en un objetivo general y tres objetivos específicos.

1.2.1. Objetivo Principal

El objetivo principal de este trabajo es aplicar técnicas de XAI para analizar los factores determinantes que influyen en las predicciones de interacciones en línea en el contexto del marketing digital.

1.2.2. Objetivos Específicos

Para alcanzar el objetivo general, se plantean los siguientes objetivos específicos:

1. Construir y evaluar varios modelos de caja negra para predecir la intención de compra en línea, comparándolos para determinar cuál ofrece la solución más efectiva.
2. Aplicar técnicas de XAI a nivel global para identificar las variables predictoras más importantes que influyen en las predicciones de éxito.
3. Utilizar técnicas de XAI a nivel local para explicar casos de éxito, casos no exitosos y posibles fallas en los modelos.

Capítulo 2

Estado del Arte

2.1. Utilidades de XAI en Diversos Sectores

La integración generalizada de la inteligencia artificial (**IA**) en el ámbito empresarial ha provocado un cambio fundamental en los procesos operativos y decisionales de las empresas. No obstante, este avance tecnológico también ha planteado importantes desafíos, especialmente en lo que respecta a la transparencia y la comprensión de los modelos de IA empleados en entornos empresariales. En este contexto, surge la necesidad de aplicar técnicas de la Inteligencia Artificial Explicable (**XAI**) para abordar estas preocupaciones y garantizar que los sistemas de IA sean transparentes, interpretables y confiables para los usuarios humanos [2].

En este marco, resulta crucial comprender las principales motivaciones y aplicaciones de XAI en el ámbito empresarial. La necesidad de justificar decisiones, controlar sistemas, mejorar modelos y descubrir nuevos conocimientos impulsa la demanda de explicabilidad en los sistemas de IA utilizados en entornos comerciales. Al abordar estas necesidades, XAI se convierte en una herramienta poderosa para mejorar la confianza y la adopción de la IA en los negocios, al tiempo que se garantiza la responsabilidad y la ética en el uso de estas tecnologías [2].

Resaltando algunas de las aplicaciones expuestas en [2], XAI tiene una amplia gama de dominios en los que los sistemas de inteligencia artificial juegan un papel crucial. A continuación, se enumeran algunas áreas potenciales donde existe una necesidad de trabajo de investigación en modelos explicables:

Transporte: Los vehículos automatizados prometen reducir las muertes en el tráfico y proporcionar una movilidad mejorada, pero también presentan desafíos en cuanto a la explicabilidad de las decisiones de IA. Los vehículos autónomos deben tomar decisiones en fracciones de segundo basadas en la clasificación de los objetos en la escena frente a ellos. La falta de explicabilidad en estas decisiones puede ser peligrosa, como se evidenció en el incidente donde un vehículo autónomo de Uber estuvo involucrado en un accidente fatal en Arizona [2][14].

Salud: Los modelos de diagnóstico médico son responsables de la vida humana, y es crucial comprender cómo funcionan y por qué toman ciertas decisiones. Incidentes pasados, como el uso de una red neuronal artificial para predecir el riesgo de neumonía, ilustran la importancia de interpretar los modelos para evitar resultados peligrosos [15].

Legal: En el ámbito de la justicia penal, la IA puede ayudar a evaluar mejor los riesgos de reincidencia y reducir los costos asociados con el crimen y la encarcelación. Sin embargo, es crucial garantizar que los modelos de decisión criminal se comporten de manera equitativa, honesta y no discriminatoria. La opacidad en los sistemas de evaluación de riesgos puede plantear problemas éticos y legales, como se observó en el caso Loomis v. Wisconsin [16].

Finanzas: En el sector financiero, la IA ofrece beneficios como mejoras en actividades de gestión de patrimonio, acceso a asesoramiento de inversión y servicio al cliente. Sin embargo, también plantea preguntas sobre la seguridad de los datos y la equidad en la concesión de préstamos. La falta de explicabilidad en los sistemas de puntuación de crédito puede dificultar la provisión de razones claras para negar créditos, lo que subraya la necesidad de hacer que los modelos basados en IA sean más explicables y auditables [2] [17].

Militar: La IA en el ámbito militar también enfrenta desafíos en cuanto a la explicabilidad. Las decisiones autónomas en operaciones militares pueden tener consecuencias de vida o muerte, lo que plantea dilemas éticos y legales similares a los encontrados en otros dominios. La iniciativa XAI en el ámbito militar está en marcha, con una serie de proyectos de investigación destinados a mejorar la transparencia y la interpretabilidad de los sistemas de IA utilizados en este contexto [18].

Estas son solo algunas de las áreas donde XAI puede tener un impacto significativo, pero también se puede aplicar en otros dominios, como ciberseguridad, educación, entretenimiento, gobierno y reconocimiento de imágenes. En cada uno de estos campos, la necesidad de sistemas de IA explicables es evidente, lo que destaca la importancia de la investigación continua en esta área y el auge que ha tenido este tema en los últimos años [2].

2.2. Aplicaciones de XAI en Marketing

En la literatura, se encuentran varios ejemplos de aplicaciones de XAI en el sector del marketing, de los cuales podemos destacar algunos ejemplos:

En un estudio realizado por Chlebus et al. [10], se investiga el éxito de las campañas de telemarketing bancario en Portugal utilizando herramientas de Inteligencia Artificial Explicable (XAI). Se aplicaron cinco algoritmos de aprendizaje automático, que incluyeron Random Forest, AdaBoost, GBM, XGBoost y CatBoost, a un conjunto de datos de campañas de telemarketing bancario en Portugal. El estudio examinó si los algoritmos de

aumento complejos ofrecían mejores resultados y si las herramientas de XAI eran mejores indicadores del rendimiento de los modelos que las medidas comúnmente utilizadas, como el AUC. Los resultados revelaron que XGBoost y CatBoost fueron los modelos más efectivos en términos de AUC. Además, mediante el análisis de la Importancia Variable Permutada y el Perfil de Dependencia Parcial (**PDP**), se encontró que CatBoost superó ligeramente a XGBoost al detectar posibles problemas de sobreajuste en este último. Estos hallazgos sugieren que los nuevos modelos de aumento tienen un mejor rendimiento que sus contrapartes más antiguas, y que las herramientas de XAI, como la Importancia Variable Permutada y el Perfil de Dependencia Parcial, pueden ser útiles para comparar modelos y comprender mejor su funcionamiento en el contexto del marketing.

El sector del marketing abarca diversos subsectores, y el uso de XAI puede ser beneficioso en múltiples aspectos. Por ejemplo, Ohana et al. [19] describen cómo los modelos de Inteligencia Artificial Explicable (XAI) se aplican al entorno de los mercados financieros, que se caracterizan por ser sistemas multiagentes difíciles de explicar e interpretar. Utilizando el método Gradient Boosting Decision Trees (**GBDT**), el estudio predice caídas importantes en los precios del S&P 500 a partir de un conjunto de características técnicas, fundamentales y macroeconómicas. Además, se emplea el concepto de valores de Shapley para identificar las variables más importantes que predicen las crisis del mercado de valores y para proporcionar una explicación local de la probabilidad de crisis en cada fecha. Este enfoque demuestra la aplicabilidad de XAI en la comprensión y predicción de eventos importantes en los mercados financieros, lo cual puede tener un impacto significativo en las estrategias de marketing de las empresas.

En otro estudio, se presenta un enfoque integral que fusiona métodos de Machine Learning (**ML**) y de Inteligencia Artificial Explicable (XAI) para el análisis y la predicción del mercado de valores [20]. Estos métodos de ML se utilizan para analizar y prever las tendencias del mercado, mientras que la clasificación de noticias se lleva a cabo utilizando técnicas de ML para identificar y clasificar noticias relevantes relacionadas con el índice del mercado. Se emplean tres modelos distintos: Redes Neuronales Artificiales (**ANN**), ARIMA y LSTM. En el ámbito de XAI, se incorpora el framework **LIME** (Local Interpretable Model-agnostic Explanations) para ofrecer explicaciones sobre las decisiones del modelo de clasificación de noticias, asegurando una comprensión clara y transparente de las recomendaciones generadas por el sistema. Estas técnicas proporcionan a los inversores valiosas perspectivas sobre las tendencias del mercado, facilitando la toma de decisiones éticas e informadas sobre sus inversiones.

Finalmente, se destaca una aplicación de XAI encontrada en la literatura relacionada con una problemática de marketing específica: la pérdida de clientes en entornos empresariales [1]. Este estudio emplea técnicas de XAI, como el modelo **RFID**, que considera la recencia, frecuencia, importancia y duración de las interacciones cliente-empresa, para complementar los métodos tradicionales y proporcionar una visión más completa de la relación cliente-empresa. La metodología evaluó varios modelos de predicción, como XGBoost, RF, Regresión Logística y SVM, y sobre el modelo de mejor desempeño (XGBoost en este caso) se integraron técnicas de XAI, como PDP y SHapley Additive

exPlanations (**SHAP**), para permitir una comprensión más profunda de los factores que influyen en el abandono del cliente desde una perspectiva global y local. Estos hallazgos son fundamentales para diseñar estrategias de retención de clientes efectivas y mejorar la gestión de la lealtad del cliente en entornos empresariales.

Se presenta Tabla 2.1 a forma de resumen de los modelos de ML y tecnincas XAI aplicadas en los estudios mencionadso anteriormente.

Tabla 2.1: Resumen de Trabajos Relacionados

Aplicación	**Técnicas de ML Aplicadas**	**Técnicas XAI Aplicadas**
Éxito de campañas de telemarketing bancario en Portugal [10]	Random Forest, AdaBoost, GBM, XGBoost, CatBoost	Importancia Variable Permutada, Perfil de Dependencia Parcial (PDP)
Predicción de caídas en los precios del S&P 500 [19]	Gradient Boosting Decision Trees (GBDT)	Valores de Shapley
Análisis y predicción del mercado de valores [20]	Redes Neuronales Artificiales (ANN), ARIMA, LSTM	LIME (Local Interpretable Model-agnostic Explanations)
Pérdida de clientes en entornos empresariales [1]	XGBoost, Random Forest, Regresión Logística, SVM	PDP, SHapley Additive exPlanations (SHAP)

En resumen, en Tabla 2.1 se evidencia que las técnicas de XAI destacadas en los ejemplos anteriores incluyen la Importancia Variable Permutada, el Perfil de Dependencia Parcial (PDP), los valores de Shapley y SHapley Additive exPlanations (SHAP) para comprender modelos en marketing y finanzas. Además, el framework LIME puede ser usado para explicar decisiones de manera local, lo cual demostró ser de utilidad en la explicación de la clasificación de noticias realizada en el estudio mencionado.

Capítulo 3

Marco Teórico

3.1. Métodos de Machine Learning

Una tarea de Machine Learning (**ML**) involucra la combinación de características o variables con la variable objetivo. Dependiendo del tipo de variable objetivo, la tarea puede ser de clasificación, regresión, análisis de supervivencia, agrupamiento (clustering) o detección de anomalías. Según la naturaleza del problema, el ML se clasifica en aprendizaje supervisado y no supervisado. Además, se distinguen modelos de caja blanca y caja negra en función de su interpretabilidad.

Es importante destacar que existe una distinción crucial entre algoritmos y Machine Learning. Un algoritmo se define como un conjunto de reglas que una máquina sigue para lograr un objetivo específico. En otras palabras, podría entenderse como una receta que especifica las entradas, la salida y los pasos necesarios para transformar las entradas en la salida (por ejemplo, las recetas de cocina son algoritmos donde los ingredientes son las entradas, la comida cocida es la salida, y los pasos de preparación y cocción son las instrucciones del algoritmo) [4].

Por otro lado, el Machine Learning representa un conjunto de métodos que permiten a las computadoras aprender de los datos para realizar y mejorar predicciones en diversos dominios, como medicina, finanzas, investigación científica, entre otros. El Machine Learning supone un cambio de paradigma con respecto a la programación tradicional, ya que en lugar de proporcionar instrucciones explícitas a la computadora, se utiliza una “programación indirecta”, a través del suministro de datos [4].

En esta sección, se presentarán algunos métodos de Machine Learning que se emplearán en este trabajo, con una breve explicación de las diferencias entre los modelos interpretables (también conocidos como modelos de caja blanca) y los modelos de caja negra que se evaluarán en el estudio.

3.1.1. Métodos de Caja Blanca

Descripción General de Modelos de Caja Blanca

Según lo expuesto en el capítulo 5 de [4], la forma más sencilla de lograr interpretabilidad en el aprendizaje automático es mediante el uso de modelos interpretables, también conocidos como modelos de caja blanca. Aunque este trabajo se centra en algoritmos de caja negra, esta sección proporcionará una breve explicación de algunos métodos de caja blanca y sus propiedades.

El primer método de caja blanca a ser explicado es la regresión lineal, el cual establece relaciones lineales entre una variable dependiente y una o más variables independientes, representando las predicciones como sumas ponderadas. Aunque es transparente y ampliamente aceptada, presenta limitaciones como su restricción a relaciones lineales y la dificultad en la interpretación de pesos debido a la multicolinealidad [4].

Por otra parte, la regresión logística, utilizada principalmente en problemas de clasificación binaria, emplea la función logística para modelar la probabilidad de que una observación pertenezca a una de las dos categorías. Aunque proporciona probabilidades de clasificación que facilitan la interpretación, tiene limitaciones como la restricción expresiva, la dificultad de interpretación debido a la multiplicación de coeficientes y la susceptibilidad a la separación completa [4].

Otro conjunto de métodos son los árboles de decisión, los cuales son modelos de aprendizaje automático que dividen el espacio de características en regiones mediante decisiones basadas en características individuales. Son altamente interpretables (aunque puede complicarse con árboles profundos), ya que cada nodo del árbol representa una regla de decisión simple sobre una variable. En general, estos métodos tienen ventajas en la captura de interacciones de características, explicaciones contrastivas y visualización natural, además de no requerir transformaciones de características. Sin embargo, carecen de suavidad y estabilidad, y no pueden manejar relaciones lineales.

Finalmente, al combinar los árboles de decisión con la regresión lineal, obtenemos los modelos RuleFit, los cuales permiten capturar relaciones complejas y ofrecen ventajas como agregar automáticamente interacciones de características a modelos lineales, mejorando la interpretación local y proporcionando herramientas diagnósticas útiles. Sin embargo, la interpretabilidad disminuye con el aumento de características, y su rendimiento puede variar según el problema. Por otra parte, la interpretación de los pesos puede ser problemática debido a reglas superpuestas.

Comparación Entre Modelos de Caja Blanca

Todos los modelos interpretables presentados anteriormente se caracterizan por su capacidad modular de interpretación. Para facilitar las diferencias en las propiedades de algunos de los algoritmos de caja blanca presentados, se presenta de forma tabular los

datos en la Tabla 3.1 expuesto en el capítulo 5 de [4]:

Tabla 3.1: Propiedades de modelos interpretables [4]

Algoritmo	**Lineal**	**Monotonía**	**Interacción**	**Tarea**
Regresión lineal	Sí	Sí	No	Regresión
Regresión logística	No	Sí	No	Clasificación
Árboles de decisión	No	Algunos	Sí	Clasificación, Regresión
RuleFit	Sí	No	Sí	Clasificación, Regresión

De la Tabla 3.1, las propiedades mostradas se definen de la siguiente manera:

- **Lineal:** indica si la asociación entre las características y el objetivo se modela de forma lineal.
- **Monotonía:** asegura que la relación entre una característica y el resultado objetivo siempre vaya en la misma dirección en todo el rango de la característica, lo cual facilita la interpretación del modelo al hacer más comprensible la relación.
- **Interacción:** indica si el modelo incluye automáticamente interacciones entre características para predecir el resultado objetivo. Las interacciones pueden mejorar el rendimiento predictivo, pero un exceso de interacciones o interacciones demasiado complejas pueden perjudicar la interpretabilidad [4].
- **Tarea:** indica la capacidad del modelo para poder ser usado en tareas de regresión o clasificación.

Se podría argumentar que tanto la regresión logística, pero esto es solo cierto para el logaritmo de la probabilidad objetivo, en donde al aumentar una característica en un punto aumenta el logaritmo de la probabilidad objetivo por una cierta cantidad, asumiendo que todas las demás variables permanecen iguales [4].

En conclusión, los métodos de caja blanca proporcionan interpretabilidad modular, lo que facilita la comprensión de las relaciones entre características y objetivos. Algunas ventajas de ciertos modelos incluyen la claridad en la interpretación, como en las regresiones lineal y logística, y la capacidad para capturar interacciones en árboles de decisión y RuleFit. Sin embargo, estos modelos pueden tener limitaciones en la modelización de relaciones no lineales y la estabilidad, especialmente en árboles profundos y RuleFit con muchas características [4].

3.1.2. Métodos de Caja Negra

Los métodos de caja negra son enfoques opacos de toma de decisiones que utilizan modelos de aprendizaje automático complejos y sofisticados para predecir resultados, pero cuya lógica subyacente es difícil de entender para los expertos en el dominio de la aplicación [8]. Estos modelos plantean desafíos en términos de interpretabilidad y confianza, ya que los usuarios finales y los especialistas en el dominio pueden tener dificultades para comprender cómo se toman las decisiones y cómo se llega a los resultados [8].

Algunos ejemplos de estos métodos incluyen las redes neuronales artificiales (**ANNs**) y los modelos de máquinas de vectores de soporte (**SVM**). Por ejemplo, las redes neuronales, que abarcan desde las redes neuronales convolucionales (**CNNs**) hasta las redes generativas antagónicas (**GANs**), son difíciles de comprender tanto para expertos en aprendizaje automático como para especialistas en el área de aplicación, debido a las numerosas transformaciones realizadas en los datos de entrada [8]. Aunque estos modelos pueden ofrecer soluciones a problemas prácticos, su complejidad y opacidad los convierten en cajas negras difíciles de interpretar [8]. Por otro lado, al igual que los modelos basados en SVM, los ANNs se consideran enfoques de caja negra debido a su complejidad y falta de transparencia en la lógica de decisión [8].

Otro conjunto de métodos son aquellos basados en árboles, los cuales se desarrollan al aplicar técnicas de ensamblaje como Random Forest, Bagging y Boosting, con las cuales se busca mejorar la capacidad predictiva y la estabilidad de los modelos [8]. Dentro de esta categoría se encuentran métodos como Random Forest y métodos de Gradient Boosting Machine, como Gradient Boosted Regression Trees (**GBRT**), Extreme Gradient Boosting (**XGBoost**) y Categorical Boosting (**CatBoost**). Aunque la interpretación de estos métodos es más fácil que la de los modelos ANNs, es importante tener en cuenta que los conjuntos de árboles de decisión pueden ser difíciles de entender para los expertos en el dominio de aplicación debido a la complejidad de múltiples árboles inducidos [8].

En general, hay un compromiso entre la complejidad y la interpretación de los resultados de los modelos. A modo de ejemplo, la Figura 3.1 expone la relación entre la interpretabilidad y la precisión de los modelos (esperado por la complejidad de este y su capacidad de detección de patrones complejos).

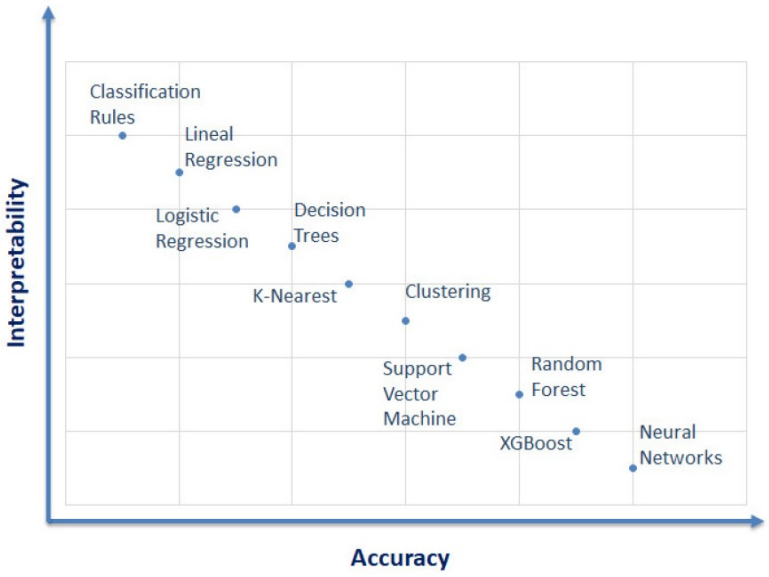


Figura 3.1: Interpretabilidad vs precisión [1].

Como se observa en la Figura 3.1, los modelos con mayor capacidad predictiva suelen ser menos interpretables, y viceversa. Este compromiso es particularmente relevante

al considerar las características de diferentes métodos de aprendizaje, tal como se muestra en la Tabla 3.2.

Tabla 3.2: Algunas características de diferentes métodos de aprendizaje [5]. Clave: △ = bueno, ◇ = aceptable y ▽ = pobre.

Característica	Redes	SVM	Árboles
Manejo natural de datos de tipo ”mixto”	▽	▽	△
Manejo de valores perdidos	▽	▽	△
Robustez a valores atípicos en el espacio de entrada	▽	▽	△
Insensibilidad a transformaciones monótonas de entrada	▽	▽	△
Escalabilidad computacional (N grande)	▽	▽	△
Capacidad para tratar con entradas irrelevantes	▽	▽	△
Capacidad para extraer combinaciones lineales de características	△	△	▽
Interpretabilidad	▽	◇	△
Poder predictivo	△	△	△

En este trabajo, nos enfocaremos en modelos basados en árboles, como Random Forest y métodos de Gradient Boosting Machine, debido a sus numerosas ventajas. Estos métodos pueden manejar datos de tipo mixto y valores perdidos, son robustos frente a valores atípicos y transformaciones monótonas, y realizan una selección interna de características, eliminando así la necesidad de un proceso adicional de selección de variables [5].

En las siguientes secciones, se describirán con más detalle los métodos basados en árboles que se utilizarán en este estudio. Cada uno de estos métodos ofrece ventajas específicas en términos de precisión, robustez y manejo de datos complejos. En particular, este trabajo empleará los siguientes métodos:

- **Random Forest:** Random Forest es un método de ensamblaje que construye múltiples árboles de decisión y los combina para mejorar la precisión y reducir el riesgo de sobreajuste. Al generar varios árboles a partir de diferentes subconjuntos de datos y características, Random Forest logra una mayor robustez y estabilidad en comparación con un solo árbol de decisión.
- **Gradient Boosting Machine (GBM):** GBM es una técnica que construye árboles de decisión secuencialmente, donde cada árbol intenta corregir los errores del árbol anterior. Este enfoque iterativo permite mejorar significativamente la capacidad predictiva del modelo, aunque puede ser más susceptible al sobreajuste si no se ajustan adecuadamente los parámetros.
- **Extreme Gradient Boosting (XGBoost):** XGBoost es una implementación optimizada de GBM que incluye técnicas adicionales para mejorar la velocidad y el ren-

dimiento del modelo. XGBoost utiliza una estrategia de regularización que ayuda a prevenir el sobreajuste y es altamente eficiente para grandes conjuntos de datos.

- **Categorical Boosting (CatBoost):** CatBoost es una variante de GBM diseñada específicamente para manejar variables categóricas de manera eficiente. Este método utiliza técnicas avanzadas de codificación y ordenamiento que mejoran tanto la precisión como la velocidad de los modelos, haciendo que CatBoost sea especialmente útil en aplicaciones con muchas variables categóricas.

3.2. Interpretabilidad

Si un modelo de aprendizaje automatizado se desempeña de buena forma, ¿sería correcto ignorar el cómo o por qué se llegó a esa conclusión? En muchos casos, y dependiendo de la naturaleza del problema, esto sería un error, ya que comprender los problemas reales con una métrica como la precisión ignoraría varias de las formas en que los problemas del mundo real funcionan [21].

Es importante aclarar que no todos los modelos requieren una explicación, ya sea por su bajo impacto en su entorno o porque el problema ya esté estudiado de forma concreta. Sin embargo, en aplicaciones donde el objetivo sea una investigación científica o la decisión del modelo pueda afectar directamente la vida de una persona, entender las decisiones detrás de cualquier decisión es de crucial importancia (algunos ejemplos son mostrados en varios capitulos de [4] a forma de ilustración). A raíz de lo anterior, la interpretabilidad es vital para este tipo de modelos y, por ende, parte fundamental de este trabajo. En esta sección, se exploran ciertas características a tener en cuenta de la interpretabilidad en el contexto de Machine Learning, rumbo a definir términos y puntos importantes para los siguientes capítulos del trabajo.

3.2.1. Importancia de la Interpretabilidad

La interpretabilidad en los modelos de aprendizaje automático juega un papel fundamental en la comprensión de cómo se llega a una decisión predictiva. Cuando un modelo ofrece predicciones, entender el proceso detrás de esas predicciones puede proporcionar valiosos conocimientos sobre el problema y los datos, así como identificar posibles fallas del modelo. Esta necesidad de interpretabilidad surge de la incompletitud en la formalización del problema, donde obtener sólo la predicción no es suficiente; el modelo también debe explicar cómo llegó a esa predicción. Además, la interpretabilidad es crucial en situaciones donde las decisiones del modelo impactan directamente en la vida de las personas, ya que proporciona una mayor transparencia y confianza en el proceso de toma de decisiones [21].

Algunos ejemplos que podemos encontrar en el capitulo 3 de [4] para ilustrar casos en donde la interpretabilidad es hiperativa son:

- **Seguridad en la conducción autónoma:** Cuando se utilizan modelos de aprendizaje automático en sistemas de conducción autónoma, es crucial entender cómo

el modelo toma decisiones para garantizar la seguridad. Por ejemplo, si un modelo de detección de peatones identifica incorrectamente objetos inanimados como peatones, necesitamos poder interpretar cómo el modelo llegó a esa conclusión para corregir el error.

- **Justicia en la concesión de préstamos:** En el contexto de la evaluación crediticia, es importante que los modelos sean interpretables para garantizar la equidad y evitar sesgos. Si un modelo rechaza una solicitud de préstamo, es necesario comprender por qué tomó esa decisión para asegurarse de que no haya discriminación indebida basada en características demográficas.

- **Detección de sesgos en modelos de aprendizaje automático:** La interpretabilidad es esencial para detectar y mitigar sesgos en los modelos de aprendizaje automático. Al comprender cómo el modelo utiliza ciertas características para tomar decisiones, podemos identificar y corregir sesgos injustos.

Sin embargo, hay escenarios donde la interpretabilidad no es necesaria o deseada. Por ejemplo, en situaciones donde el modelo no tiene un impacto significativo o cuando el problema está bien estudiado y no se buscan nuevos conocimientos. Además, en casos donde la interpretabilidad podría permitir la manipulación del sistema, como en sistemas de puntuación crediticia, donde los usuarios podrían intentar engañar al sistema para obtener beneficios personales. Por lo tanto, la necesidad de interpretabilidad depende del contexto y los objetivos específicos del problema que se está abordando, y se debe considerar cuidadosamente su aplicación en cada situación [22].

A forma de ilustracion de los casos mencionados anteriormente, podemos enumerar los siguientes dos ejemplos mencionados en el capitulo 3 de [4]:

- **Proyectos personales sin impacto significativo:** En situaciones donde los modelos de aprendizaje automático se utilizan para proyectos personales sin consecuencias importantes, como predecir destinos de vacaciones de amigos basados en datos de Facebook, la interpretabilidad puede no ser necesaria. Si el modelo no tiene un impacto significativo, la precisión de las predicciones puede ser más importante que la interpretabilidad.

- **Aplicaciones bien estudiadas con experiencia acumulada:** En casos donde las aplicaciones han sido ampliamente estudiadas y tienen una gran cantidad de experiencia acumulada, como los modelos de reconocimiento óptico de caracteres utilizados en la lectura de direcciones en sobres, la interpretabilidad puede ser menos relevante. Si los modelos han demostrado ser confiables y efectivos a lo largo del tiempo, la comprensión detallada de cómo funcionan puede no ser necesaria para su uso continuo.

3.2.2. Propiedades de la Interpretabilidad

Doshi-Velez y Kim [21] presentan tres niveles de interpretabilidad para evaluar modelos de inteligencia artificial. En primer lugar, la evaluación a nivel de aplicación implica

pruebas directas con usuarios finales, generalmente expertos, utilizando un diseño experimental. En segundo lugar, la evaluación a nivel humano simplifica el proceso anterior al realizar experimentos con personas no especializadas, lo que resulta en una metodología más económica. Finalmente, la evaluación a nivel de función se basa en la clase de modelo utilizado, proporcionando interpretaciones coherentes y mejorando la transparencia del modelo sin la necesidad de pruebas directas con usuarios finales.

Además, Robnik-Šikonja [23] discute las propiedades de los métodos de explicación para tareas de Machine Learning, que incluyen el poder expresivo para generar explicaciones en diversos formatos como reglas IF-THEN, árboles de decisión, entre otros. La translucidez varía según cómo el método de explicación depende del modelo de aprendizaje automático, desde modelos intrínsecamente interpretables hasta aquellos que manipulan entradas y observan las predicciones. Además, se evalúa la portabilidad del método y su complejidad algorítmica, considerando la computación eficiente de las explicaciones.

En cuanto a las Explicaciones Individuales, se destacan la precisión para predecir datos no vistos, la fidelidad en la aproximación a las predicciones del modelo de caja negra, la consistencia entre modelos similares, y la estabilidad frente a pequeñas variaciones en las características. La comprensibilidad es fundamental, evaluando cómo las personas entienden las explicaciones, junto con la certeza y el grado de importancia que reflejan. También se considera la novedad y la representatividad de las explicaciones en relación con la distribución de datos.

3.2.3. Métodos de Explicación

En primer lugar, la taxonomía de la interpretabilidad en ML puede ayudar a comprender mejor las propiedades fundamentales que influyen en la interpretación de los modelos. Resumiendo los puntos mencionados en el capítulo 3 de [4] podemos encontrar las siguientes clasificaciones:

- **Intrínseco vs. Post hoc:**
  - **Intrínseco:** Modelos de ML considerados interpretables debido a su estructura simple, como árboles de decisión cortos o modelos lineales dispersos.
  - **Post hoc:** Métodos de interpretación aplicados después del entrenamiento del modelo, como la importancia de características por permutación.
- **Específico del modelo vs. Agnóstico del modelo:**
  - **Específico del modelo:** Herramientas de interpretación limitadas a clases de modelos específicas, como la interpretación de pesos de regresión en un modelo lineal.
  - **Agnóstico del modelo:** Herramientas que pueden aplicarse a cualquier modelo de ML después del entrenamiento, como el análisis de pares de entrada y salida.
- **Local vs. Global:**

- **Local:** Métodos que explican una predicción individual.
- **Global:** Métodos que explican el comportamiento general del modelo.

De acuerdo a lo anterior, una forma de representar los métodos de explicacion que se buscan obtener serian:

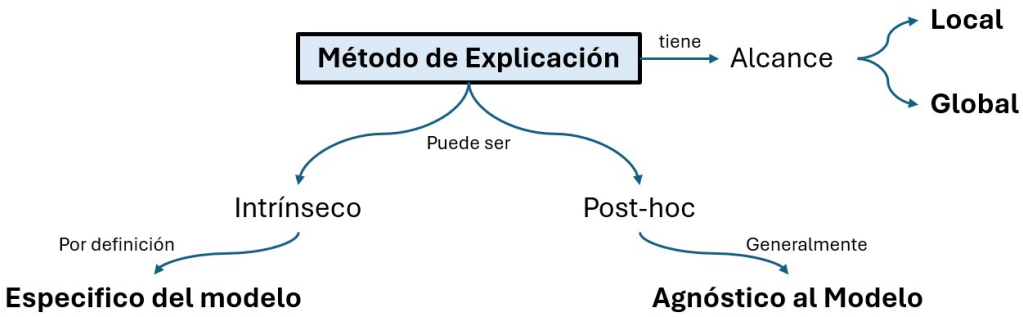


Figura 3.2: Diagrama de clasificacion de métodos de interpretacion [2].

Los resultados que podemos tener para poder brindar esta explicación serán:

- **Resultado del método de explicación:**
  - **Estadísticas resumidas de características:** Métodos que proporcionan estadísticas resumidas para cada característica, como la importancia de características o las fuerzas de interacción de características.
  - **Visualización resumida de características:** La visualización de estadísticas resumidas de características, como las curvas de dependencia parcial.
  - **Internos del modelo (por ejemplo, pesos aprendidos):** La interpretación de modelos intrínsecamente interpretables, como los pesos en modelos lineales.
  - **Punto de datos:** Métodos que devuelven puntos de datos, como explicaciones contrafactuales o la identificación de prototipos de clases predichas.

3.3. Explainable Artificial Intelligence (XAI)

La Inteligencia Artificial Explicable (**XAI**, por sus siglas en inglés) se refiere al conjunto de técnicas diseñadas para hacer transparentes y comprensibles los procesos y decisiones de los sistemas de IA para los seres humanos. En términos generales, el objetivo de la explicabilidad es facilitar el acceso a ciertos aspectos del sistema para diferentes usuarios, como desarrolladores, diseñadores, reguladores y usuarios finales [24]. Como se discutió en la sección anterior sobre métodos de explicación, los enfoques de XAI pueden proporcionar explicaciones detalladas tanto de una sola predicción de un modelo (explicabilidad local) como del modelo en su totalidad (explicabilidad global).

Es fundamental destacar que muchos expertos en dominios de aplicación no necesitan comprender la complejidad matemática detrás de los modelos de caja negra; en

cambio, requieren interpretaciones naturales de los resultados [25]. Se requieren modelos innovadores que fusionen ambos enfoques para mejorar la interpretación y la comprensión de los sistemas de IA.

En esta sección, nos centraremos en explorar varios métodos de XAI que han surgido para abordar la necesidad de explicabilidad en la inteligencia artificial. Comenzaremos examinando los métodos agnósticos del modelo, que son técnicas que pueden aplicarse a cualquier modelo de IA después de su entrenamiento, proporcionando interpretaciones locales y globales de las decisiones del modelo.

3.3.1. Métodos Agnósticos de Modelos

En el contexto de machine learning (ML), los "Métodos Agnósticos de Modelos" se refieren tecnicas que son independientes del modelo de ML específico utilizado para entrenar o hacer predicciones. Estos métodos están diseñados para proporcionar ciertas funcionalidades o análisis sin depender de los detalles internos de un modelo en particular. En el contexto de este trabajo nos enfocaremos en 2 clases de métodos:

- **Métodos Agnósticos de Modelos Locales:** Métodos de explicación para predicciones individuales.
- **Métodos Agnósticos de Modelos Globales:** Métodos que describen el comportamiento promedio de un modelo de aprendizaje automático.

3.3.2. Métodos Agnósticos de Modelos Locales

Los Métodos Agnósticos de Modelos Locales consisten en dar predicciones individuales. En esta sección, exploraremos los siguientes métodos de explicación local:

- **Local Interpretable Model-Agnostic Explanations (LIME):** Método de predicción en el que se reemplaza el módelo complejo con un módelo de sustitución localmente interpretable.
- **Valores Shapley:** Método de atribución que distribuye equitativamente la predicción a características individuales.
- **SHapley Additive exPlanations (SHAP):** Se trata de otro método de cálculo para los valores de Shapley, que además propone métodos de interpretación global basados en combinaciones de valores de Shapley a lo largo de los datos.

Tanto LIME como los valores de Shapley (utilizados en SHAP) son métodos de atribución, lo que significa que describen la predicción de una instancia individual como la suma de los efectos de las características. Otros métodos, como las explicaciones contrafactuales, se basan en ejemplos. La integración de estos métodos de explicación local puede proporcionar una comprensión más completa de cómo los modelos de inteligencia artificial toman decisiones en datos específicos [4].

Además, en esta sección nos enfocaremos únicamente en la interpretación local de los resultados de SHAP. El uso para la interpretación global de los modelos se profundiza en la sección 3.3.3.

Local Interpretable Model-Agnostic Explanations (LIME)

El primer conjunto de métodos a explorar es modelos de sustitución local, o **LIME** por sus siglas en ingles. Esta aproximacion a la interpretacion de modelos de caja negra fue introducida en [13], en donde los autores proponen una implementación concreta de modelos sustitutos locales. Estos modelos sustitutos se entrenan para aproximar las predicciones del modelo de caja negra subyacente. En lugar de entrenar un modelo sustituto global, LIME se centra en entrenar modelos sustitutos locales para explicar predicciones individuales. La idea es bastante intuitiva: primero, olvidar los datos de entrenamiento e imaginar que solo se tiene el modelo de caja negra donde se pueden introducir puntos de datos y obtener las predicciones del modelo. El objetivo es comprender por qué el modelo de aprendizaje automático realizó cierta predicción. LIME prueba qué sucede con las predicciones cuando se dan variaciones de los datos en el modelo de aprendizaje automático. LIME genera un nuevo conjunto de datos que consiste en muestras perturbadas y las predicciones correspondientes del modelo de caja negra. Sobre este nuevo conjunto de datos, LIME luego entrena un modelo interpretable, que está ponderado por la proximidad de las instancias muestreadas a la instancia de interés [4].

El modelo interpretable puede ser cualquier cosa, desde modelos interpretables como Lasso hasta árboles de decisión. El modelo aprendido debería ser una buena aproximación de las predicciones del modelo de aprendizaje automático localmente, pero no tiene que ser una buena aproximación global. Este tipo de precisión también se denomina fidelidad local (local fidelity) [4].

Matematicamente, los modelos locales subrogados usados en LIME se definen como:

$$\text{explanation}(x) = \underset{g \in G}{\text{argmin}} \left(L(f, g, \pi_x) + \Omega(g) \right) \tag{3.1}$$

Donde:

- explicación(x) representa la explicación para la instancia x.
- g es el modelo interpretable que se está considerando (por ejemplo, un modelo de regresión lineal).
- G es la familia de posibles modelos interpretables.
- $L(f, g, \pi_x)$ es la función de pérdida que mide la discrepancia entre las predicciones del modelo de caja negra f y el modelo interpretable g, ponderada por la proximidad π_x de las instancias muestreadas a la instancia de interés x.
- $\Omega(g)$ es una medida de complejidad del modelo interpretable, que se utiliza para penalizar la complejidad del modelo y evitar el sobreajuste.

En general, el proceso de entrenamiento de los modelos subrogados locales es el siguiente [4]:

1. Seleccionar la observación (instancia) de interés para la cual se desea obtener una explicación de su predicción de caja negra.
2. Perturbar el conjunto de datos y obtener las predicciones de la caja negra para estos nuevos puntos.
3. Ponderar las nuevas muestras según su proximidad a la instancia de interés.
4. Entrenar un modelo interpretable ponderado en el conjunto de datos con las variaciones.
5. Explicar la predicción interpretando el modelo local.

Limitando nuestra explicación a datos tabulares, es importante aclarar que LIME no toma muestras alrededor de la instancia de interés, sino del centro de masa de los datos de entrenamiento, lo cual es problemático, pero aumenta la probabilidad de que el resultado para algunos de los puntos de muestra difiera del punto de datos de interés y que LIME pueda aprender al menos alguna explicación. LIME utiliza un kernel de suavizado exponencial para definir el vecindario. Un kernel de suavizado es una función que toma dos instancias de datos y devuelve una medida de proximidad. El ancho del kernel determina cuán grande es el vecindario: un ancho pequeño significa que una instancia debe estar muy cerca para influir en el modelo local, mientras que un ancho más grande permite que instancias más alejadas también influyan en el modelo. La implementación en Python usa un ancho del kernel de 0.75 veces la raíz cuadrada del número de columnas del conjunto de datos, pero no existe una forma óptima de determinar el mejor kernel o ancho. Este problema se agrava en espacios de alta dimensión y al tratar características con unidades de medida diferentes [4].

Valores Shapley y SHapley Additive exPlanations (SHAP):

Los valores Shapley, conceptualizados por Shapley en [26], son una herramienta derivada de la teoría de juegos cooperativos utilizada para explicar las predicciones en modelos de aprendizaje automático. En este marco, cada valor de característica de una instancia se concibe como un “jugador” en un juego, donde la predicción del modelo es la ganancia. Los valores Shapley distribuyen equitativamente esta “ganancia” entre las características, reflejando la contribución marginal promedio de cada característica a través de todas las posibles coaliciones de características [4].

Para calcular los valores Shapley, se consideran todas las posibles combinaciones de características y se determina cómo cambia la predicción al agregar una característica específica a la combinación existente. Sin embargo, este proceso puede ser computacionalmente costoso, especialmente para conjuntos de datos grandes con muchas características [4].

Una aproximación común para estimar los valores Shapley es a través del muestreo de Monte Carlo, como se propone en [27]. Esta técnica permite calcular los valores Shapley de manera eficiente al generar múltiples instancias “híbridas” a partir de combinaciones de datos reales y de muestras aleatorias.

Con base en la teoría anterior, se introduce el método **SHAP** (SHapley Additive exPlanations) por Lundberg y Lee en [28], el cual se basa en los valores Shapley para explicar predicciones individuales atribuyendo la contribución de cada característica a la predicción del modelo. Esto se logra calculando los valores SHAP, que representan la contribución marginal promedio de cada característica a la diferencia entre la predicción del modelo para una instancia específica y la predicción promedio del modelo.

Definiendo $\phi_{i,j}$ como el valor SHAP para la variable j en el registro i de los datos:

$$\phi_{i,j} = \text{avg}\left(\frac{1}{n}\sum_{k=1}^{n} f(x_k)\right) - \frac{1}{n}\sum_{k=1}^{n} f(x_k) \tag{3.2}$$

donde:

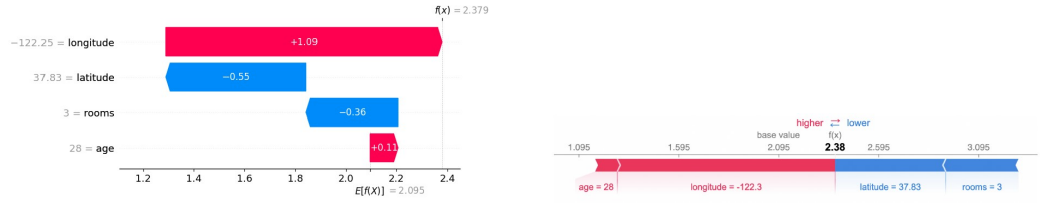
- $\phi_{i,j}$ es el valor SHAP para la característica j y la instancia de datos i.
- n es el número total de instancias de datos.
- $f(x_k)$ es la predicción del modelo para la instancia de datos k.

En general, al definir una instancia i para la predicción $f(x^{(i)})$ se puede evaluar la contribución a la predicción de característica definiendo ϕ_j para el cual se tiene [3]:

- Si $\phi_j = 0$, la característica x_j no afecta la predicción de $x^{(i)}$.
- Si $\phi_j > 0$, la característica x_j tiene un efecto positivo en $x^{(i)}$.
- Si $\phi_j < 0$, la característica x_j tiene un efecto negativo en $x^{(i)}$.

Los valores shap tienen las siguietnes propiedades o axiomas de [3]:

- **Eficiencia:** Los valores SHAP ϕ_{ij} de una instancia i se suman para dar la predicción del modelo. Esto se expresa matemáticamente como: $f(x^{(i)}) = \frac{1}{n}\sum_{k=1}^{n} f(x^{(k)}) + \sum_{j=1}^{p} \phi_{ij}$, donde n es el número total de instancias, y p es el número de características del conjunto de datos.
- **Simetría:** Si dos características j y k contribuyen igualmente a todas las combinaciones posibles de características, entonces sus contribuciones deben ser las mismas para todas las instancias i. Esto se representa como $\phi_{ij} = \phi_{ik}$.
- **Dummie:** Si una característica j no afecta el valor predicho de una instancia i, independientemente de las demás características, entonces su valor SHAP debe ser cero para todas las instancias i. Esto se expresa como $\phi_{ij} = 0$.
- **Aditividad:** Si una predicción se puede dividir en dos predicciones, entonces los valores SHAP para la predicción combinada deben ser la suma de los valores SHAP para cada predicción individual. Matemáticamente, esto se representa como $\phi_{ij} = \phi_{ij}^{(1)} + \phi_{ij}^{(2)}$, donde $\phi_{ij}^{(1)}$ y $\phi_{ij}^{(2)}$ son los valores SHAP para las predicciones individuales.



(a) Ejemplo de Gráfico Cascada de SHAP [3]

(b) Ejemplo de Gráfico de Fuerza de SHAP [3]

Figura 3.3: Ejemplo de diagramas usados para explicaciones locales de SHAP [3].

Este enfoque proporciona una interpretación intuitiva de cómo cada característica afecta la predicción del modelo y permite la comparación visual de la importancia relativa de las características a través de gráficos, como los mostrados en la Figura 3.3.

Para evitar confusiones terminológicas, en este trabajo se seguirán estas convenciones al igual que se realizo en [3]: “valores Shaple” se refiere al método original de la teoría de juegos, “SHAP” se refiere a la aplicación de los valores Shapley en la interpretación de predicciones de aprendizaje automático y “valores SHAP” para los resultados del uso de SHAP en las características.

En [3] y [4] , se mencionan dos tipos de kernels que se emplean para calcular los valores SHAP en diferentes contextos y tipos de modelos: **KernelSHAP** y **TreeSHAP**. KernelSHAP es un método agnóstico al modelo que utiliza una aproximación basada en una regresión ponderada para estimar los valores SHAP. Por otro lado, TreeSHAP es una implementación optimizada específicamente para modelos basados en árboles de decisión, como árboles de decisión individuales, bosques aleatorios y modelos de boosting de árboles. Las propiedades principales de cada uno de estos métodos se exponen en la Tabla 3.3.

Tabla 3.3: Comparación de métodos para calcular valores SHAP [3] [4]

Método	**Ventajas**	**Desventajas**
KernelSHAP	Aplicable a cualquier modelo, proporciona explicaciones precisas, consistencia teórica con valores de Shapley	Computacionalmente costoso, problemas con dependencias de características, resultados poco fiables a veces
TreeSHAP	Computación rápida y eficiente, produce resultados exactos para modelos de árboles	Puede producir atribuciones de características no intuitivas, no siempre es consistente con la teoría de Shapley

Por otra parte, es crucial destacar que, a diferencia de LIME, los valores SHAP ofrecen un único valor por característica pero no constituyen un modelo de predicción completo. Por lo tanto, no son aptos para hacer inferencias sobre cómo cambian las predicciones ante modificaciones en las entradas, como por ejemplo, “Si ganara €300 más al año, mi puntaje crediticio aumentaría en 5 puntos” [3].

Comparación entre Métodos para Explicabilidad Local

A forma de resumen, la Tabla 3.4 muestra las ventajas y desventajas de cada uno de los métodos presentados en esta sección:

Tabla 3.4: Comparación de Técnicas de XAI para Explicabilidad Local: LIME vs SHAP

Técnica	Ventajas	Desventajas
LIME	▪ Aplicable a modelos complejos. ▪ Genera explicaciones interpretables a nivel local. ▪ Funciona con datos tabulares, de texto e imágenes. ▪ Permite usar modelos interpretables diferentes al original.	▪ Problemas con la definición correcta del vecindario. ▪ Inestabilidad en las explicaciones cercanas. ▪ Puede ser manipulado para ocultar sesgos.
SHAP	▪ Proporciona una comprensión detallada de la contribución de cada característica. ▪ Conexión teórica con los valores de Shapley. ▪ Implementaciones eficientes para modelos basados en árboles (TreeSHAP). ▪ Utiliza métodos de muestreo eficientes como KernelSHAP.	▪ Posible interpretación engañosa de los valores Shapley. ▪ No es un modelo de predicción completo. ▪ Puede producir atribuciones de características no intuitivas.

3.3.3. Métodos Agnósticos de Modelos Globales

Los Métodos Agnósticos de Modelos Globales describen el comportamiento promedio de un modelo de aprendizaje automático. A menudo se expresan como valores esperados basados en la distribución de los datos. Dado que estos métodos describen el comportamiento promedio, son útiles para comprender los mecanismos generales en los datos o depurar un modelo [4].

En esta secion, se profundizaran sobre las siguientes técnicas de interpretación global, independientes del modelo:

- **Partial Dependence Plots (PDP):** un método de efecto de características que que calculan los valores de predicción esperado cuando se marginalizan todas las demás características.
- **SHapley Additive exPlanations (SHAP):** tambien usados para interpretacion local, incluye métodos de interpretación global basados en combinaciones de valores de Shapley a lo largo de los datos.
- **Modelos Subrogados Globales:** reemplazan el modelo original con un modelo más simple para su interpretación.

Partial Dependence Plots (PDP)

Partial Dependce Plot (PDP) tienen sus origenes en [29] y consisten en graficas que muestra el efecto marginal que una o dos variables tienen en el resultado predicho de un modelo de Machine Learning, mostrando si la relacion entre el objetivo y una o 2 variables es lineal, monotonico o alguna relacion mas compleja.

La función de dependencia parcial para regresión se define como:

$$\hat{f}_S(x_S) = E_{X_C}[\hat{f}(x_S, X_C)] = \int \hat{f}(x_S, X_C)\, d\mathbb{P}(X_C) \tag{3.3}$$

La función de dependencia parcial se enfoca en las características específicas de un modelo de aprendizaje automático. Las variables x_S representan estas características, mientras que X_C incluye las demás variables del modelo. Por lo general, S tiene una o dos características relevantes para el análisis. La dependencia parcial se calcula marginalizando la salida del modelo sobre la distribución de las características en C, lo que revela la relación entre las características de S y la predicción. Esta función, $\hat{f}_S$, se estima mediante promedios en los datos de entrenamiento, utilizando el método de Monte Carlo como se presenta a continuación [4]:

$$\hat{f}_S(x_S) = \frac{1}{n} \sum_{i=1}^{n} \hat{f}(x_S, x_C^{(i)}) \tag{3.4}$$

En la ecuación (3.4), $x_C^{(i)}$ son los valores reales de las características del conjunto de datos para las características en las que no estamos interesados, y n es el número de instancias en el conjunto de datos. Una suposición de la PDP es que las características en C no están correlacionadas con las características en S. Si esta suposición se viola, los promedios calculados para la trama de dependencia parcial incluirán puntos de datos que son muy improbables o incluso imposibles.

PDP, calcula la importancia de las características mediante una medida simple basada en la dependencia parcial propuesta por Greenwell en [30] . La idea general es que una dependencia parcial plana indica que la característica no es importante, y cuanto más varíe la dependencia parcial, más importante será la característica. Para características numéricas, la importancia se define como la desviación de cada valor único de la característica del promedio de la curva. Para características categóricas, la importancia se calcula como la diferencia entre el valor máximo y mínimo de la dependencia parcial dividida por cuatro. Esta medida captura solo el efecto principal de la característica y no considera posibles interacciones entre características. Es importante tener cuidado al interpretarla, ya que podría no reflejar la verdadera importancia si la característica afecta la predicción principalmente a través de interacciones con otras características. Además, esta medida considera cada valor único de manera igual, lo que puede no ser adecuado si hay valores con pocas instancias [4].

Explicaciones Globales con SHapley Additive exPlanations (SHAP)

El método SHAP (SHapley Additive exPlanations) nos permite obtener tanto interpretaciones locales (exploradas en la seccion 3.3.2) como globales de nuestro modelo. Para una interpretación global, se pueden calcular los valores SHAP para un conjunto más amplio de puntos de datos, idealmente para todo el conjunto de datos de prueba, utilizando los datos de entrenamiento como referencia. Al visualizar estos valores en todas las características y varios puntos de datos, podemos identificar patrones sobre cómo el modelo realiza predicciones, proporcionándonos así una visión general del comportamiento del modelo [3]. Usando la Figura 3.4.

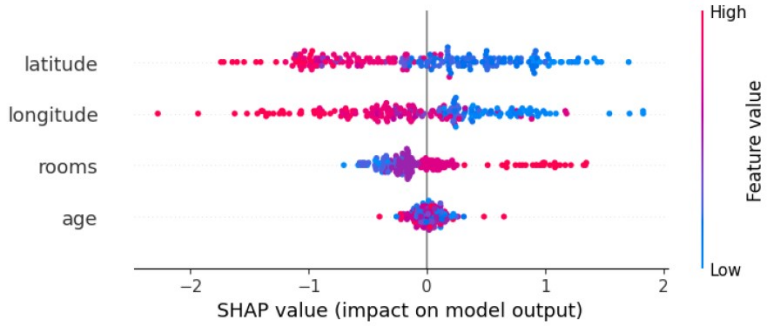


Figura 3.4: Ejemplo de gráfico beeswarm en SHAP [3].

Modelos Subrogados Globales

Un modelo subrogado global (abreviado en la sección 3.3.3 como **MS**) es un modelo interpretable que se entrena para aproximar las predicciones de un modelo de caja negra. Interpretando el modelo subrogado, se pueden sacar conclusiones sobre el modelo de caja negra, resolviendo así la interpretabilidad del aprendizaje automático usando más aprendizaje automático. En el contexto del aprendizaje automático interpretable, el modelo subyacente es un modelo de aprendizaje automático, y el modelo subrogado debe ser interpretable. El objetivo es aproximar las predicciones del modelo subyacente tan precisamente como sea posible mientras se mantiene la interpretabilidad [4].

Es importante aclarar que para el uso de estos modelos es necesario medir cuán bien el modelo subrogado replica las predicciones del modelo de caja negra. Para esto, se puede emplear la medida R-cuadrado (R^2), que indica el porcentaje de varianza capturada por el modelo subrogado.

En este contexto, es posible utilizar una versión simplificada de los datos de un modelo, obtenida a partir de los valores SHAP. Estos valores proporcionan una comprensión de la importancia de cada característica en las predicciones del modelo de caja negra. A partir de estos valores SHAP, se puede entrenar un modelo interpretable, como una regresión lineal o un árbol de decisión, que facilite la interpretación del modelo de caja negra más complejo. Este enfoque permite obtener un modelo subrogado que capture las relaciones entre las características y las predicciones del modelo de caja negra, al

tiempo que conserva la interpretabilidad del modelo resultante.

Comparación entre Métodos para Explicabilidad Global

A forma de resumen, la Tabla 3.5 muestra las ventajas y desventajas de cada uno de los métodos presentados en esta sección:

Tabla 3.5: Comparación de Métodos para Explicaciones Globales

Técnica	Ventajas	Desventajas
SHAP	▪ Detalle en la contribución de cada característica. ▪ Fundamentado en la teoría de Shapley. ▪ Implementaciones eficientes para modelos basados en árboles.	▪ Rendimiento computacional lento con KernelSHAP. ▪ Simplifica la dependencia entre características. ▪ Atribuciones no siempre intuitivas con TreeSHAP.
PDP	▪ Intuitivos y fáciles de interpretar. ▪ Capturan efectos marginales de características. ▪ Detectan relaciones lineales y no lineales.	▪ Supone independencia entre características. ▪ Oculta efectos heterogéneos entre características. ▪ Puede llevar a interpretaciones erróneas si las características están correlacionadas.
MS	▪ Aproximan predicciones de modelos complejos de manera interpretable. ▪ Evaluables mediante métricas como R-cuadrado. ▪ Flexibles en la elección de modelos interpretables.	▪ Conclusiones sobre el modelo y no sobre datos reales. ▪ Umbral incierto de R-cuadrado para confiar en la aproximación. ▪ Posible sobreajuste y inconsistencias en interpretaciones.

3.4. Metodologías para la Minería de Datos

3.4.1. Métodología KDD

En [6] se mencionan KDD, CRISP-DM y SEMMA, como métodos usados para el proceso de mineria de datos. Entre ellos, KDD tiene entre sus 9 pasos 1 que hace enfasis en la interpretabiliad de resultados, y debido a la impotancia de la interpretabilidad para el objetivo principal de este trabajo, sera el método que usaremos para este trabajo.

El método KDD, por sus siglas en ingles Knowledge Discovery Databases, es un método de iterativo y interactivao, enfocado en encontrar conocimiento en los datos, según

[6]:

1. **Desarrollo y Comprensión del Dominio de Aplicación** Este es el primer paso del proceso KDD en el cual se definen los objetivos desde el punto de vista del cliente y se desarrolla una comprensión sobre el dominio de aplicación y su conocimiento previo.

2. **Creación de un Conjunto de Datos Objetivo** Este es el segundo paso del proceso KDD, que se enfoca en crear el conjunto de datos objetivo y el subconjunto de muestras o variables de datos. Es una etapa crucial ya que el descubrimiento de conocimiento se realiza sobre estos datos.

3. **Limpieza y Preprocesamiento de Datos** Este es el tercer paso del proceso KDD, enfocado en limpiar y preprocesar los datos objetivo para obtener datos completos y consistentes sin ruido ni inconsistencias. Se desarrollan estrategias para manejar estos tipos de datos ruidosos e inconsistentes.

4. **Transformación de Datos** Este es el cuarto paso del proceso KDD, enfocado en la transformación de datos de una forma a otra para facilitar la implementación de algoritmos de minería de datos. Se aplican varios métodos de reducción y transformación de datos en los datos objetivo.

5. **Elección de la Tarea de Minería de Datos Adecuada** Este es el quinto paso del proceso KDD, en el cual se elige una tarea de minería de datos adecuada basada en los objetivos específicos definidos en el primer paso. Ejemplos de tareas de minería de datos incluyen clasificación, agrupamiento, regresión y resumen.

6. **Elección del Algoritmo de Minería de Datos Adecuado** Este es el sexto paso del proceso KDD, donde se seleccionan uno o más algoritmos de minería de datos adecuados para buscar diferentes patrones en los datos. La selección se basa en cumplir con los criterios generales de la minería de datos.

7. **Implementación del Algoritmo de Minería de Datos** Este es el séptimo paso del proceso KDD, en el cual se implementan los algoritmos seleccionados.

8. **Interpretación de los Patrones Minados** Este es el octavo paso del proceso KDD, enfocado en la interpretación y evaluación de los patrones minados. Este paso puede involucrar la visualización de los patrones extraídos.

9. **Uso del Conocimiento Descubierto** Este es el paso final del proceso KDD, en el cual el conocimiento descubierto se utiliza para diversos propósitos. El conocimiento descubierto también puede compartirse con partes interesadas o integrarse con otro sistema para acciones adicionales.

3.4.2. Comparacion de Métodos de Minería de Datos

En el anterior apartado se hizo enfasis en los pasos del modelo KDD. Pero, tambien se hizo referencia a los métodos CRISP-DM y SEMMA que pese a tener sus propias etapas y enfoques específicos, comparten muchas similitudes en sus objetivos y procedimientos

con KDD. El modelo KDD se compone de nueve etapas, mientras que CRISP-DM y SEMMA tienen seis y cinco etapas respectivamente. Cada uno de estos modelos se adapta a diferentes necesidades y contextos, pero todos buscan optimizar el proceso de descubrimiento de conocimiento a partir de los datos. Por ejemplo, el paso de KDD “Desarrollo y Comprensión del Dominio de la Aplicación” es equivalente a la fase de “Comprensión del Negocio” de CRISP-DM, y el paso de “Transformación de Datos” de KDD se corresponde con la fase de ”Preparación de Datos”de CRISP-DM y la etapa de ”Modificación”de SEMMA. La siguiente tabla presenta una comparación detallada entre estos modelos de proceso de minería de datos [6].

Tabla 3.6: Comparación de modelos de proceso de minería de datos [6]

<table>
<tr><th>KDD</th><th>CRISP-DM</th><th>SEMMA</th></tr>
<tr><td>Desarrollo y comprensión de la aplicación</td><td>Comprensión del negocio</td><td>***</td></tr>
<tr><td>Creación de un conjunto de datos objetivo</td><td rowspan="2">Comprensión de los datos</td><td>Muestra</td></tr>
<tr><td>Limpieza y preprocesamiento de datos</td><td>Exploración</td></tr>
<tr><td>Transformación de datos</td><td>Preparación de datos</td><td>Modificación</td></tr>
<tr><td>Elección de la tarea de minería de datos adecuada</td><td rowspan="3">Modelado</td><td rowspan="3">Modelo</td></tr>
<tr><td>Elección del algoritmo de minería de datos adecuado</td></tr>
<tr><td>Empleo del algoritmo de minería de datos</td></tr>
<tr><td>Interpretación de patrones minados</td><td>Evaluación</td><td>Evaluación</td></tr>
<tr><td>Uso del conocimiento descubierto</td><td>Despliegue</td><td>***</td></tr>
</table>

Capítulo 4

Implementación y Resultados del Análisis de la Intención de Compra

4.1. Metodología Propuesta

Considerando los objetivos establecidos para esta investigación, tal como se expusieron en el Capítulo 1, se propone una metodología basada en el Proceso de Descubrimiento de Conocimiento en Bases de Datos (KDD), detallado en la Sección 3.4, con algunos cambios para que la terminología usada durante esta sección sea adecuada con varios de los términos introducidos en el capítulo 3 (“Marco Teórico”). A continuación, se presenta una breve descripción de las tareas a desarrollar en cada paso:

1. **Desarrollo y Comprensión del Dominio de Aplicación** Breve explicación para dar comprensión sobre el dominio de aplicación.
2. **Creación de un Conjunto de Datos Objetivo** Se describe el proceso de obtención de datos según la información suministrada en [12] y se explica el significado de las variables disponibles.
3. **Limpieza y Preprocesamiento de Datos** Análisis exploratorio y limpieza de valores nulos y duplicados. Se realizan las primeras tareas de tratamiento de datos como la agrupación de categorías según su relación con la variable objetivo, revisión de correlaciones y distribución de los datos.
4. **Transformación de Datos** Este es el cuarto paso del proceso KDD, enfocado en la transformación de datos de una forma a otra para facilitar la implementación de los modelos de minería de datos. En este trabajo se incluirá en este paso la separación de datos, ya que varias de las transformaciones, como la estandarización y las codificaciones (encodings), necesitarán ser definidas basadas en los datos de entrenamiento (train) para aplicarse posteriormente a los datos de prueba (test).
5. **Elección de la Tarea de Minería de Datos Adecuada** Este es el quinto paso del proceso KDD, en el cual se elige una tarea de minería de datos adecuada basada en los objetivos específicos definidos en el primer paso. Ejemplos de tareas de minería de datos incluyen clasificación, agrupamiento, regresión y resumen.

6. **Elección del Método de Machine Learning Adecuado** Este es el sexto paso del proceso KDD, mencionado originalmente con el nombre “Elección del Algoritmo de Minería de Datos Adecuado”, pero para mantener la terminología mencionada en 3.1 y evitar confusión entre los métodos de ML y el término algoritmo. En este proceso, se entrenan modelos usando una validación cruzada entre 5 grupos utilizando los métodos SVM (kernels lineal, polinomial y RBF), ANN (de una sola capa), RF, GBM, XGBoost y CatBoost. Entre ellos se selecciona el mejor modelo.

7. **Implementación del Método de Machine Learning Seleccionado** Una vez elegido el modelo, se evalúa su rendimiento y se realizan las predicciones usando los datos separados para prueba.

8. **Interpretación de los Patrones Minados** Este paso está enfocado en la interpretación y evaluación de los patrones minados, por lo que en este paso se realizará el proceso de aplicación de técnicas XAI a nivel local con LIME y SHAP, y de explicación a nivel global usando PDP y SHAP. Finalmente, se usará un modelo de árbol surrogado para interpretar las relaciones más importantes entre las variables predictoras en el modelo.

9. **Uso del Conocimiento Descubierto** Este es el paso final del proceso KDD, en el cual el conocimiento descubierto se utiliza para diversos propósitos. El conocimiento descubierto también puede compartirse con partes interesadas o integrarse con otro sistema para acciones adicionales.

Una version simplificada del flujo de entrenamiento y aplicacion de tecnicas XAI se presenta en Figura 4.1:

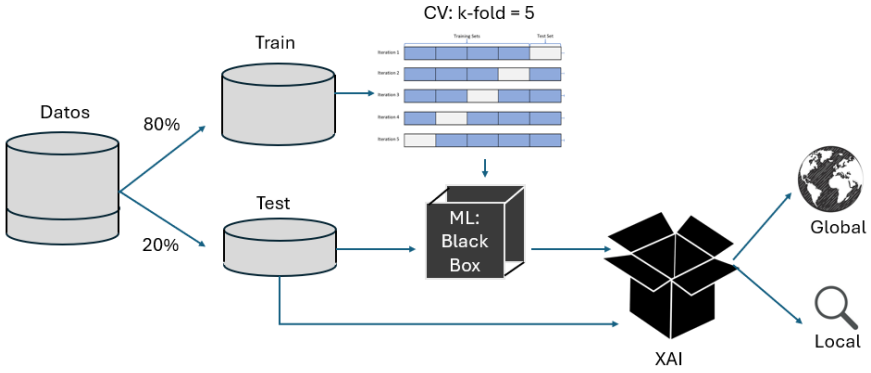


Figura 4.1: Diagrama simplificado de flujo de datos.

4.2. Análisis de la Base de Datos de Intención de Compra de Usuarios en Línea

Comprender el comportamiento de los usuarios durante las sesiones en línea que culminan en una compra es crucial para las empresas que operan en entornos digitales. Variables como el número de páginas administrativas visitadas, el tipo de tráfico y el sistema operativo utilizado por el visitante proporcionan información valiosa para entender

patrones de comportamiento, optimizar la experiencia del usuario y diseñar estrategias efectivas de marketing y ventas. Por tanto, este estudio puede ayudar a identificar áreas de mejora y a tomar decisiones informadas para aumentar las tasas de conversión y el rendimiento general del sitio web. Con base en esto, se procederá a implementar la metodología KDD para abordar esta problemática.

4.2.1. Desarrollo y Comprensión del Dominio de Aplicación

El primer paso del proceso KDD consiste en definir los objetivos desde la perspectiva del cliente y desarrollar una comprensión profunda del dominio de aplicación y su conocimiento previo. En este trabajo, se centrará en comprender el comportamiento de compra en línea de los usuarios mediante el análisis de datos recopilados durante un año en un minorista en línea, tal como se detalló en [12]. Los datos seleccionados se eligieron para evitar sesgos relacionados con campañas específicas o períodos estacionales, asegurando así la representatividad de los patrones de comportamiento observados. Se extrajeron características clave de las sesiones de los usuarios, incluyendo datos de clics y detalles de la sesión, para categorizar las visitas según su probabilidad de culminar en una compra. Este enfoque permite una comprensión profunda de los factores que influyen en la intención de compra del usuario en un entorno en línea, allanando el camino para estrategias efectivas de retención de clientes y optimización de la experiencia del usuario.

4.2.2. Creación del Conjunto de Datos Objetivo

En el segundo paso del proceso KDD, se focaliza en la creación del conjunto de datos objetivo y el subconjunto de muestras o variables de datos. Esta etapa es crucial, ya que el descubrimiento de conocimiento se lleva a cabo sobre estos datos. Como se mencionó en la sección 4.2.1, los datos utilizados para este estudio coinciden con los empleados en [12], donde se especifica que provienen de diversas fuentes y metodologías de recopilación enfocadas en comprender el comportamiento de los usuarios en entornos de comercio electrónico. A continuación, se presenta un resumen del proceso de obtención de estos datos:

1. **Seguimiento de Interacciones de Usuarios en Sitios Web de Comercio Electrónico**: Se implementaron sistemas de seguimiento en los sitios web para recopilar datos sobre las interacciones de los usuarios durante sus sesiones en línea. Esto implicó el uso de herramientas de análisis web como Google Analytics u otras soluciones personalizadas.

2. **Extracción de Características Relevantes**: A partir de los datos de seguimiento recopilados, se extrajeron características relevantes que podrían estar asociadas con la intención de compra de los usuarios. Estas características incluyeron información sobre las páginas visitadas, la duración de las sesiones, el tipo de tráfico, las acciones realizadas en el sitio y otras variables pertinentes para el análisis del comportamiento del usuario.

3. **Etiquetado de Datos**: Se etiquetaron los datos recopilados para indicar si una sesión de usuario culminó con una compra o no. Esto implicó analizar el comportamiento posterior del usuario para determinar si finalmente realizó una transacción en el sitio.

En resumen, los datos utilizados en este estudio se originan a partir de la observación y registro de las interacciones de los usuarios en sitios web de comercio electrónico. Posteriormente, se extrajeron características relevantes y se etiquetaron los datos para su posterior análisis. Este conjunto de datos está disponible en el repositorio de la **UC Irvine (UCI)** bajo el título *"Online Shoppers Purchasing Intention Dataset"* [7], y se distribuye bajo la licencia *Creative Commons Atribución 4.0 Internacional (CC BY 4.0)*, que permite compartir y adaptar los conjuntos de datos para cualquier propósito, siempre que se otorgue el crédito adecuado. La Tabla 4.1 describe las características del conjunto de datos usada en el estudio:

Tabla 4.1: Resumen de Características del Conjunto de Datos Disponible en [7]

Nombre de la Variable	**Descripción (Tipo)**
Administrative	Número de páginas administrativas visitadas por un usuario durante una sesión en el sitio web de comercio electrónico. (Numérica)
Administrative_Duration	Tiempo total (en segundos) dedicado a las páginas administrativas durante una sesión. (Numérica)
Informational	Número de páginas informativas visitadas por un usuario en una sesión. Estas páginas contienen información relevante no relacionada directamente con productos. (Numérica)
Informational_Duration	Tiempo total (en segundos) dedicado a las páginas informativas durante una sesión. (Numérica)
ProductRelated	Número de páginas relacionadas con productos visitadas por un usuario en una sesión. Estas páginas incluyen detalles de productos, comparativas, etc. (Numérica)
ProductRelated_Duration	Tiempo total (en segundos) dedicado a las páginas relacionadas con productos durante una sesión. (Numérica)
BounceRates	Porcentaje de visitantes que abandonan el sitio después de ver una página específica durante una sesión. Es una medida de la efectividad de la página para retener a los visitantes. (Numérica)
ExitRates	Porcentaje de visitas que terminan después de ver una página específica en comparación con todas las visitas a esa página. Indica la proporción de salidas de una página específica. (Numérica)
PageValues	Valor promedio de una página web que un usuario visita antes de completar una transacción de comercio electrónico. (Numérica)
SpecialDay	Indicador que muestra la proximidad de la visita a un día especial como festividades o eventos promocionales. (Numérica)
Month	Mes del año en que se realizó la sesión de usuario en el sitio web. (Categórica)
OperatingSystems	Sistema operativo utilizado por el visitante durante la sesión en el sitio web. (Categórica)

Tabla 4.1: (Continuación)

Nombre de la Variable	Descripción (Tipo)
Browser	Navegador web utilizado por el visitante durante la sesión en el sitio web. (Categórica)
Region	Región geográfica del visitante desde donde se originó la sesión en el sitio web. (Categórica)
TrafficType	Tipo de tráfico que dirige a los usuarios al sitio web, como búsqueda orgánica, referencias de otros sitios, etc. (Categórica)
VisitorType	Indica si el visitante es un usuario recurrente o nuevo en el sitio web. (Categórica)
Weekend	Indicador booleano que muestra si la visita ocurrió durante un fin de semana (sábado o domingo). (Binaria)
Revenue	Variable objetivo que indica si la sesión de usuario resultó en una transacción completada (1) o no (0). (Binaria)

El conjunto de datos resultante está compuesto por vectores de características correspondientes a 12,330 sesiones, de los cuales sólo 15.5 % (1908) de las sesiones corresponden a sesiones finalizadas en compra. Cada sesión fue asignada a un usuario diferente durante un período de un año para evitar cualquier sesgo hacia campañas específicas, días especiales, perfiles de usuario o períodos. Es importante destacar que los datos presentados en la Tabla 4.1 muestran algunas discrepancias con la descripción proporcionada en [7]. Según [12], variables como **OperatingSystems**, **Browser**, **Region** y **TrafficType**, aunque inicialmente descritas como números enteros, son en realidad variables categóricas. Por lo tanto, estas variables fueron tratadas de esta manera para el estudio en cuestión. En total, el conjunto de datos consta de 10 variables numéricas, 5 categóricas y 2 binarias, siendo una de estas últimas nuestra variable objetivo.

Además, según [7], las características **BounceRates**, **ExitRates** y **PageValues** representan métricas medidas por Google Analytics para cada página en el sitio de comercio electrónico. La **BounceRate** indica el porcentaje de visitantes que entran al sitio desde una página y luego salen sin interactuar con otras páginas. Por otro lado, la **ExitRate** representa el porcentaje de visitas que terminan en una página específica. La **PageValue** indica el valor promedio generado por una página antes de completar una transacción (un alto valor indica que una página es crucial en el proceso de conversión). Finalmente, la característica **SpecialDay** señala la proximidad de la visita a un día especial, con valores determinados por la dinámica del comercio electrónico y fechas relevantes como el Día de San Valentín.

4.2.3. Limpieza y Preprocesamiento de Datos

En esta sección se abordarán varias tareas, como el tratamiento de valores nulos, valores duplicados y la revisión de anomalías (identificación de valores atípicos o inconsistencias).

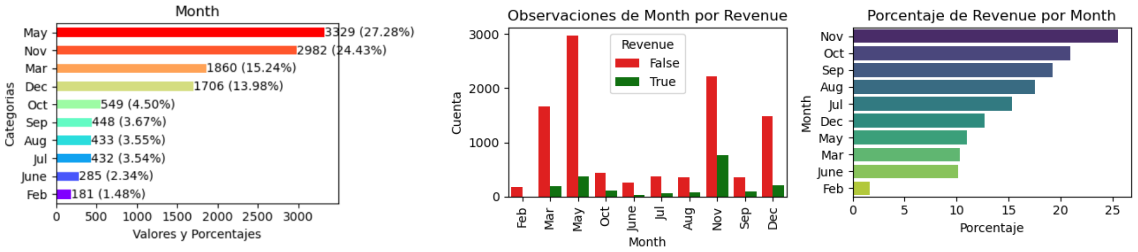
En primer lugar, se importó el conjunto de datos utilizando la librería `ucimlrepo` en **Python** para realizar el proceso de limpieza y preprocesamiento de datos, asegurando que estos fueran tratados conforme se definió en la Tabla 4.1.

Una vez importados los datos, se verificó la existencia de valores nulos u observaciones duplicadas utilizando las herramientas disponibles en la librería `Pandas` de Python. Se identificaron 125 registros duplicados, los cuales fueron eliminados.

Análisis de Variables Categóricas

Concluida la limpieza de duplicados, se realizó un análisis exploratorio para comprender mejor la distribución de las variables y evaluar la necesidad de tratamiento de los datos para las siguientes etapas. Se separaron las actividades en variables categóricas y numéricas, evaluando cada conjunto de características por separado.

Para iniciar el análisis exploratorio de las variables categóricas, se definieron algunas funciones para facilitar el análisis, destacando `univariate_categorical` y `plot_categorical_variable`, las cuales presentan gráficamente la distribución de las categorías de cada variable categórica y el porcentaje de observaciones asociadas a la clase de interés. A continuación, se presentan los resultados gráficos de algunas variables y las observaciones relevantes obtenidas.



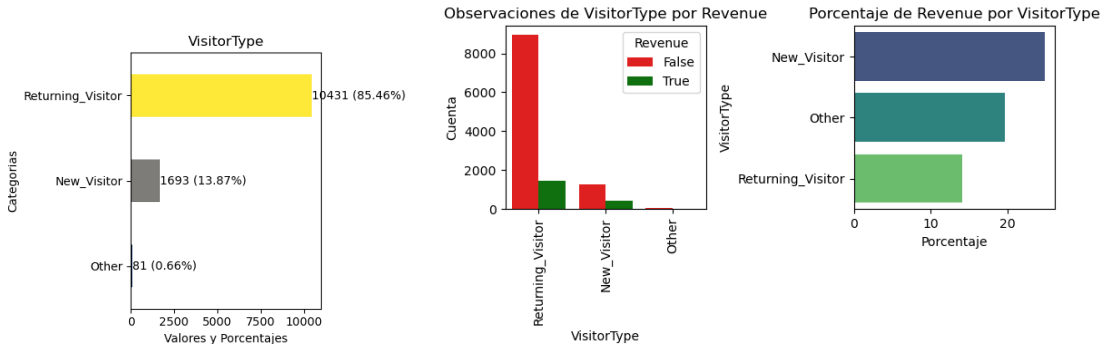
(a) Distribución de las categorías (b) Distribución de observaciones de la clase de interés

Figura 4.2: Análisis de variable categórica **Month**

De las Figura 4.2, se observa que la información de la variable **Month** indica que los meses de noviembre (Nov), octubre (Oct) y septiembre (Sep) tienen los porcentajes más altos de transacciones con **Revenue** = 1, con un 25.35 %, 20.95 % y 19.2 % respectivamente. Esto sugiere que las transacciones son más frecuentes durante estos meses. Por otra parte, los meses de febrero (Feb) y marzo (Mar) tienen los porcentajes más bajos de transacciones con Revenue = 1, con solo 1.63 % y 10.07 % respectivamente. Esto sugiere que las transacciones son menos frecuentes durante estos meses.

Por otra parte, debido a que los datos son de solo un año y existe un orden en cada una de las categorías, en la siguiente sección se sugiere el uso de codificación ordinal para esta variable.

La FiguraA.8 revela que los nuevos visitantes (**New_Visitor**) tienen el mayor porcentaje de transacciones, alcanzando un 24.91 %. Esto sugiere que los nuevos visitantes



(a) Distribución de las categorías (b) Distribución de observaciones de la clase de interés

Figura 4.3: Análisis de variable categórica **VisitorType**

tienen una probabilidad más alta de completar una transacción. En contraste, los visitantes recurrentes (**Returning_Visitor**) muestran un porcentaje menor de transacciones en comparación, aunque aún significativo, con un 13.93 %. Por último, la categoría **Other** representa menos del 1 % de las muestras, por lo que los registros asociados a esta categoría serán eliminados. Además, se transformará la variable **VisitorType** en una variable binaria llamada **VisitorType_NewVisitor**, indicando con 1 si el cliente es nuevo o 0 si es recurrente.

Al analizar la variable **Weekend** (Fin de semana), se observa que las transacciones durante estos días (**Weekend** = True) tienen un porcentaje de compras más alto, alcanzando el 17.4 %, en comparación con los días de semana que tienen un 14.89 %. Esto indica que las transacciones son más frecuentes durante los fines de semana.

Similarmente, se realizó un análisis de la variable **Region**, la cual no mostró una variación significativa entre sus categorías. Sin embargo, las variables **OperatingSystems**, **Browser** y **TrafficType** presentan numerosas categorías con similitudes entre algunas y diferencias marcadas entre otras, además de algunas categorías con baja representación.

Al igual que con las variables mencionadas anteriormente, se realizó un análisis similar para evaluar la variable **Region**, la cual no mostró una variación significativa entre sus categorías. Sin embargo, las variables **OperatingSystems**, **Browser** y **TrafficType** exhiben numerosas categorías con algunas similitudes entre ellas y otras marcadas diferencias, además de presentar algunas categorías con baja representación.

Para obtener una comprensión inicial de la importancia de estas variables en la predicción de compras, se implementó una nueva función llamada `chi_square_test` para realizar pruebas de chi-cuadrado en variables categóricas y determinar su significancia estadística con respecto a una variable objetivo. Se definió un parámetro **alpha** de 0.05 para esta prueba. Este valor asegura que las conclusiones basadas en el análisis estadístico sean conservadoras y minimicen los errores de tipo I.

Los resultados detallados se presentan en la Tabla 4.2, donde el símbolo indica una

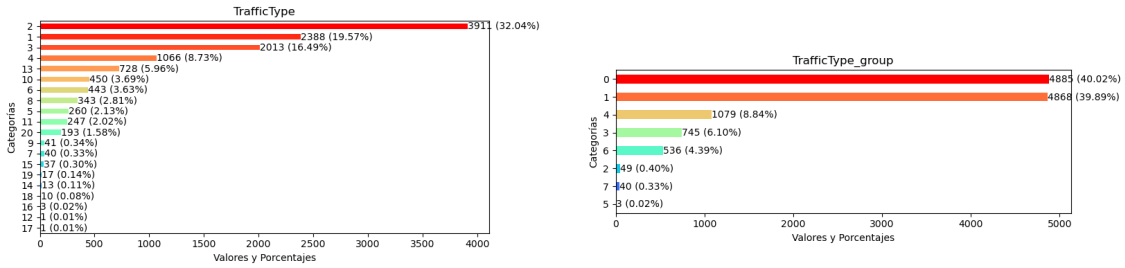
advertencia debido al escaso número de muestras en la tabla de contingencia utilizada en la prueba de chi-cuadrado, lo cual podría estar relacionado con categorías poco representadas.

Tabla 4.2: Resultados de la prueba chi-cuadrado

Variable	chi2	p-value	significance
Month	376.279817	1.566845e-75	SI
OperatingSystems	75.069931	1.387988e-13	SI
Browser	29.118987	3.782803e-03	SI
Region	9.679351	2.882562e-01	NO
TrafficType	359.224837	1.263264e-64	SI
VisitorType	130.667627	4.225571e-29	SI
Weekend	9.204803	2.413810e-03	SI

A partir de los resultados mostrados en la Tabla 4.2 se observa que todas las variables, con excepción de **Region**, muestran significancia en la prueba de chi-cuadrado, indicando que inicialmente todas las variables son relevantes para el estudio, salvo la variable **Region**. Sin embargo, las variables **OperatingSystems**, **Browser**, y **TrafficType** presentan numerosas categorías, algunas de las cuales tienen una representación limitada. Dado que estas categorías son anónimas y podrían no proporcionar suficiente información, se optó por agrupar aquellas con distribuciones similares utilizando una función denominada `categoric_similarity_clustering`. Esta función utiliza la distancia euclidiana para agrupar categorías en función de la similitud de su impacto en una variable objetivo.

A continuación se muestra un ejemplo utilizando la variable **TrafficType**, tanto antes como después de la agrupación, para ilustrar cómo cambia la distribución:



(a) Distribución de categorías de la variable **TrafficType**.

(b) Distribución de categorías de la variable **TrafficType** después de la agrupación.

Figura 4.4: Distribución de categorías de la variable **TrafficType** antes y después de la agrupación.

A continuación se presentan los grupos formados a partir de las categorías de las variables **TrafficType**, **Browser**, y **OperatingSystems**, así como sus respectivas representaciones en Tabla 4.3 donde **Rep. (%)** hace referencia a la representación del grupo en el total de muestras y **% Clase** indica el porcentaje de muestras en el grupo asociados a la clase de interes (i.e. porcentaje asociado a transacción en la sesión).

Tabla 4.3: Grupos formados a partir de **TrafficType**, **Browser**, y **OperatingSystems**.

Variable	Grupo	Lista de categorías	Rep. (%)	% Clase
TrafficType	5	1, 3, 6, 9	3.0	0.0 %
TrafficType	7	7	0.33	30.0 %
TrafficType	6	20, 8	4.39	27.05 %
TrafficType	1	10, 11, 2, 5	39.91	21.36 %
TrafficType	4	14, 4	8.84	15.48 %
TrafficType	0	1, 3, 6, 9	39.99	10.21 %
TrafficType	3	13, 19	6.10	5.91 %
TrafficType	2	12, 15, 17, 18	0.40	0.0 %
Browser	2	12, 13	0.54	28.79 %
Browser	1	10, 5	5.14	18.79 %
Browser	0	1, 11, 2, 4, 7, 8	91.99	15.55 %
Browser	4	6	1.42	11.49 %
Browser	3	3	0.86	4.76 %
Browser	5	9	0.01	0.0 %
OperatingSystems	3	8	0.61	22.67 %
OperatingSystems	1	2, 4, 5	57.50	17.67 %
OperatingSystems	0	1, 7	20.91	14.87 %
OperatingSystems	2	3, 6	20.88	10.59 %

Los resultados anteriores sugieren que algunos grupos no solo tienen una mayor cantidad de datos disponibles, sino también una propensión más alta hacia las transacciones, lo cual podría ser crucial para análisis predictivos y estratégicos futuros.

En la variable **TrafficType**, los Grupos 1 y 0 destacan por su alta representación. El Grupo 1 cuenta con 39.91 % de representación (equivalente a 4,868 muestras), representado un porcentaje significativo de compras del 21.36 %. Por otro lado, el Grupo 0 presenta una cantidad similar de muestras totales (4,885 equivalentes al 39.99 % del total de muestras) y un porcentaje de compras ligeramente inferior, del 10.21 %.

En la variable **Browser**, el Grupo 0 muestra la mayor cantidad de muestras totales con una presentación de 91.99 % (11,231 muestras), aunque el Grupo 2 sobresale con el mayor porcentaje de compras, alcanzando el 28.79 % entre sus 66 muestras totales.

En cuanto a **OperatingSystems**, el Grupo 1 se distingue por sus 7,025 muestras totales y un porcentaje de compras del 17.67 %, indicando una presencia significativa en las transacciones en comparación con otras agrupaciones.

Sin embargo, también encontramos la presencia de grupos con poca representación, como el Grupo 5 en **Browser**, que solo cuenta con 1 muestra en total y no está asociado a la clase de interés (es decir, relacionada con sesiones de usuario que hayan realizado compras). Por esta razón, se procede a revisar categorías con menos del 1 % de representación para su eliminación. Después de este análisis, se repite la prueba de chi-cuadrado, obteniendo resultados similares a los mostrados en la Tabla 4.2, pero esta vez sin ninguna advertencia que indique muestras insuficientes en la tabla de contingencia utilizada en la prueba.

Estos hallazgos subrayan la utilidad del método de agrupación basado en la similitud del impacto sobre la variable objetivo, especialmente en variables como **TrafficType**, **Browser**, y **OperatingSystems**. La identificación de diferencias significativas entre grupos en términos de propensión a las transacciones proporciona una sólida justificación para aplicar técnicas avanzadas como **Target Encoding**. Al estandarizar y simplificar la codificación de estas variables categóricas, se mejora la capacidad del modelo para capturar y utilizar de manera efectiva las relaciones observadas, optimizando así su desempeño predictivo y estratégico. Este proceso se hará en la siguiente sección 4.2.4.

Análisis de Variables Numéricas

Habiendo realizado el análisis de las variables categóricas y procedido con la primera limpieza de observaciones, el número de registros en nuestro conjunto de datos se reduce a 11,885. Ahora se procederá con el análisis de las variables numéricas. En este caso, se revisará gráficamente la distribución de las variables, la posible presencia de valores atípicos, y se realizarán pruebas dependiendo de la naturaleza de los datos para determinar la importancia de cada variable en la predicción de nuestra variable objetivo. Finalmente, se llevará a cabo un estudio de correlaciones para identificar aquellas variables que podrían presentar una alta correlación tanto entre sí como con la variable objetivo **Revenue**.

Primero, se realiza un análisis gráfico de las variables numéricas en relación con la variable objetivo. Se utiliza un histograma para evaluar la distribución de los valores de cada variable y un diagrama de cajas para visualizar cómo varía esta distribución según la actividad de compra. Los resultados de este proceso se presentan en el Anexo en la Figura A.16.

La Figura A.16 muestra que todas las variables presentan numerosos valores atípicos. Específicamente, variables como **Administrative**, **ProductRelated**, **PageValues**, **ExitRates** y **BounceRate** exhiben diferencias significativas en la distribución de sus valores entre la clase de interés y la clase mayoritaria, lo cual sugiere su potencial como variables predictoras. Se llevarán a cabo pruebas para evaluar la importancia predictiva de cada variable. Dado que los datos exhiben asimetría y valores atípicos, se evitarán pruebas paramétricas como el test de Welch o el test t de Student, que presuponen normalidad. En su lugar, se empleará la Prueba U de Mann-Whitney (Wilcoxon), una prueba no paramétrica robusta en estas condiciones. Esta prueba evalúa diferencias en las medianas de dos grupos independientes, siendo crucial para determinar la significancia de las características en la clasificación binaria. Para este propósito, se ha implementado la función `test_variable_continua_wilcoxon`. Los resultados detallados se presentan en la Tabla 4.4.

Tabla 4.4: Resultados de la Prueba U de Mann-Whitney para Variables Numéricas.

Variable	Asimetría	Outliers	p-valor	Estadístico U
Administrative	1.938692	401	4.392e-71	7046525.0
Administrative_Duration	5.600838	1111	1.769e-68	7095518.0
Informational	3.996042	2582	1.464e-33	8163532.5
Informational_Duration	7.499273	2362	2.505e-32	8225259.0
ProductRelated	4.309935	954	6.260e-96	6520358.0
ProductRelated_Duration	7.220161	925	1.383e-116	6224496.0
BounceRates	3.220826	1371	1.642e-53	11345478.5
ExitRates	2.263235	1259	5.020e-161	13026988.5
PageValues	6.089358	2680	0.000e+00	2573522.0
SpecialDay	3.275962	1223	1.020e-21	10034686.5

Según los resultados obtenidos en la Tabla 4.4, todas las variables analizadas (**Administrative**, **Administrative_Duration**, **Informational**, **Informational_Duration**, **ProductRelated**, **ProductRelated_Duration**, **BounceRates**, **ExitRates**, **PageValues** y **SpecialDay**) son estadísticamente significativas según la prueba U de Mann-Whitney, con p-valores muy bajos. La mayoría de las variables muestran alta asimetría y una cantidad notable de outliers. Como se discutió en la sección 3.1.2, modelos como los basados en árboles tienden a ser robustos ante outliers, mientras que SVM y ANN pueden enfrentar desafíos. Dado que los datos provienen de sesiones reales de usuarios, eliminar los outliers podría eliminar información valiosa sobre comportamientos atípicos pero importantes. Por lo tanto, se anticipa un mejor rendimiento inicial en modelos basados en árboles, justificando así la decisión de no tratar los outliers inicialmente.

Finalmente, se evalúan las correlaciones entre las variables numéricas, incluyendo la variable objetivo **Revenue**, para examinar si las variables predictoras tienen alguna relación con la variable objetivo y entre sí. Este estudio de correlaciones es crucial para identificar posibles multicolinealidades que puedan afectar la interpretabilidad y el rendimiento de los modelos predictivos. Además, se analiza si existe un alto nivel de correlación entre las variables predictoras, lo cual podría influir en el entrenamiento de algunos modelos.

En general, las variables muestran diversas correlaciones significativas entre sí. Por ejemplo, **ProductRelated** y **ProductRelated_Duration** tienen una alta correlación de 0.86, indicando que los usuarios que pasan más tiempo en páginas relacionadas con productos también visitan más de estas páginas. Las variables **BounceRates** y **ExitRates** presentan una correlación de 0.90, lo que sugiere que sesiones con altas tasas de rebote también tienden a terminar rápidamente. Además, **Administrative** y **Administrative_Duration** tienen una correlación de 0.60, reflejando la relación entre la cantidad de páginas administrativas visitadas y el tiempo dedicado a ellas. La Figura 4.5 muestra el resultado en mapa de calor de las correlaciones:

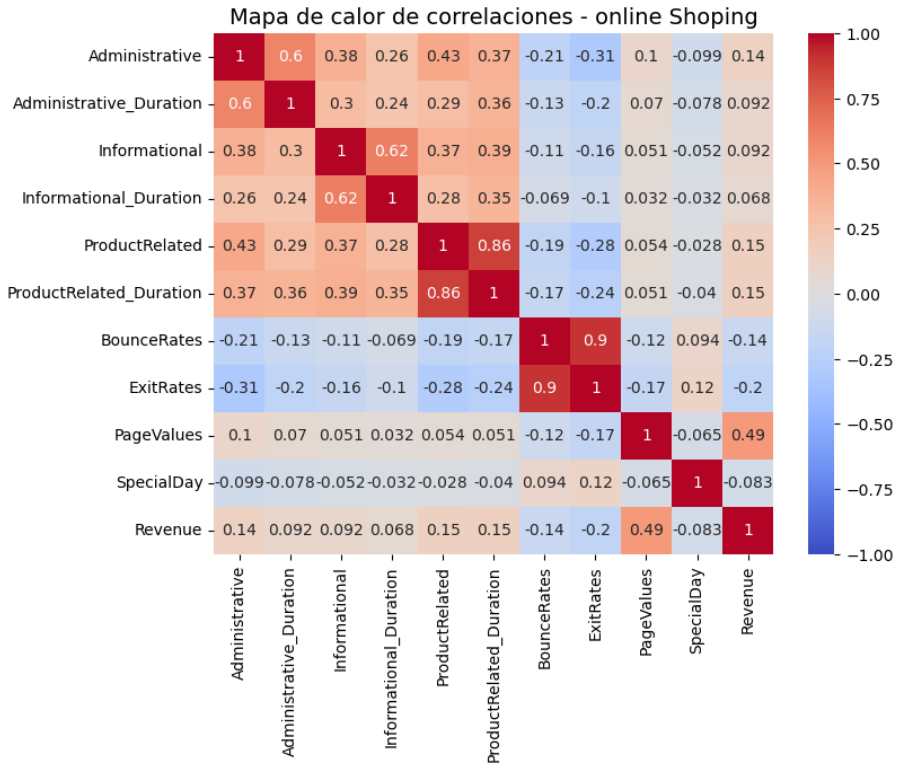


Figura 4.5: Mapa de calor de las correlaciones entre las variables numéricas y **Revenue**.

Focalizando en la variable **Revenue**, observamos que **PageValues** tiene la mayor correlación positiva con 0.49, indicando que los usuarios con un mayor valor de página tienen más probabilidades de generar ingresos. Las variables **ProductRelated** y **ProductRelated_Duration** también muestran correlaciones positivas con **Revenue** (0.15 y 0.14, respectivamente), sugiriendo que la actividad relacionada con productos está asociada con mayores ingresos. En contraste, **ExitRates** y **BounceRates** tienen correlaciones negativas con **Revenue** (-0.20 y -0.14, respectivamente), lo que implica que sesiones que terminan rápidamente o rebotan con frecuencia son menos propensas a generar ingresos. En Figura 4.6 se puede apreciar con mayor detalle el la relacion de correlacion de las variables predictoras con la activiadad de compra.

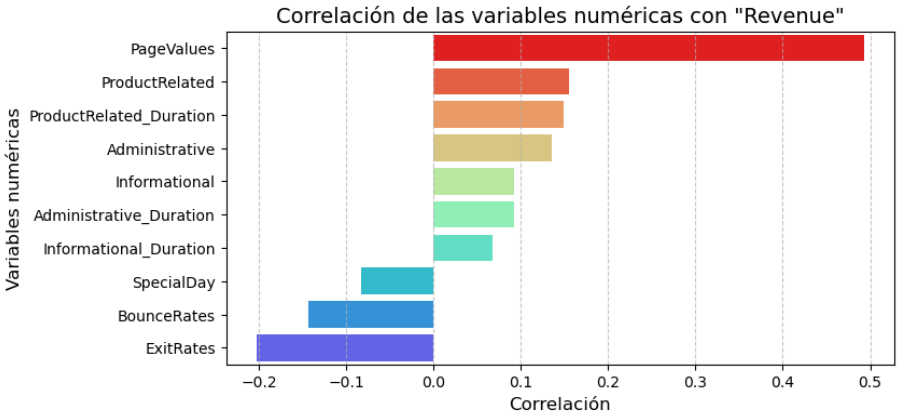


Figura 4.6: Mapa de calor de las correlaciones entre las variables numéricas y **Revenue**

Los resultados sugieren que el uso de técnicas de selección de variables puede ser útil para mejorar el desempeño de modelos como ANN, SVM, RF, GBM, XGBoost y CatBoost.

4.2.4. Transformación de Datos

En la sección de Transformación de Datos dentro del proceso KDD, se implementan estrategias cruciales para preparar los datos de manera óptima antes de aplicar métodos de minería de datos. Este paso se centra en modificar la estructura y formato de los datos para mejorar la interpretación y el rendimiento de los modelos predictivos. Entre las tareas fundamentales se incluyen el One Hot Encoding, que convierte variables categóricas en vectores binarios para capturar relaciones entre categorías sin introducir un orden numérico incorrecto. También se realiza el Ordinal Encoding, esencial para preservar la jerarquía entre categorías ordinales mediante la asignación de valores numéricos.

Otra tarea clave es el Split de Train y Test, que asegura la evaluación imparcial del modelo en datos no vistos, mitigando el riesgo de sobreajuste y garantizando su generalización. Además, el Target Encoding se emplea para codificar variables categóricas basadas en estadísticas del valor objetivo asociado, mejorando así la capacidad predictiva del modelo. El Escalamiento de Variables Numéricas ajusta las escalas para facilitar la convergencia de algoritmos, mientras que la Selección de Variables reduce la complejidad del modelo al enfocarse en aquellas más relevantes para el problema, optimizando tanto el rendimiento como la eficiencia del proceso de modelado. Estas técnicas aseguran que los datos estén adecuadamente preparados para la siguiente etapa de modelado, proporcionando un marco sólido para la generación de resultados precisos y aplicables.

Para la facilidad lectora se separará cada tarea y se presentara un breve resumen de cada una a continuacion:

One-Hot Encoding

Basado en los resultados de la prueba de chi-cuadrado final, detallados en la sección 4.2.3, se concluye que la variable categórica que no muestra una distinción clara en relación con la variable objetivo es **Region**, presentando un p-valor relativamente alto. Debido a la falta de una estrategia definida para manejar esta variable, se opta por utilizar la técnica de **One-hot encoding**. Esta técnica implica el uso de la función `pd.get_dummies`, la cual transforma la variable **Region** en 9 nuevas variables binarias correspondientes a sus categorías.

Ordinal Encoding

El ordinal encoding es idóneo para representar el mes en datos de un solo año debido a su capacidad para preservar el orden temporal y la relación secuencial entre los meses. Esto facilita la interpretación de modelos y optimiza la eficiencia computacional al mantener la estructura original de los datos. Por esta razón se aplica un diccionario en **Python** usando los valores únicos encontrados en la variable **Month** para cambiar de categorico a numerico el valor de esta variable.

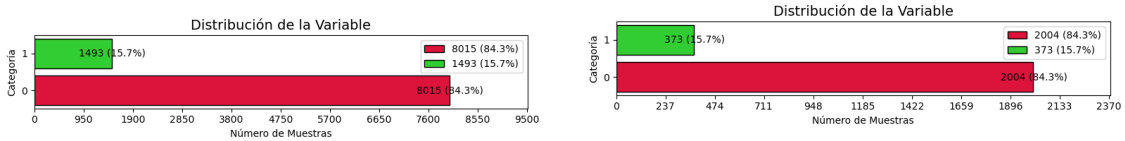
Separación de Datos en Conjuntos de Entrenamiento y Prueba

Para este trabajo, utilizaremos el enfoque de separación de datos en conjuntos de entrenamiento y prueba, seguido de una validación cruzada en los datos de entrenamiento para el desarrollo de los métodos. Este enfoque, como se menciona en el capítulo 5 de [5], combina la separación tradicional de datos con la validación cruzada tipo K-Fold. Es particularmente beneficioso para evaluar el rendimiento en datos no vistos y prevenir el sobreajuste, proporcionando una evaluación más realista y precisa del modelo final.

La principal ventaja de este enfoque es su capacidad para evitar el “data leakage” al mantener una separación clara entre los conjuntos de entrenamiento y prueba, lo cual es crucial para evaluar de manera fiable el rendimiento del modelo en situaciones del mundo real. Sin embargo, una limitación de esta metodología es que puede resultar en un menor uso de datos para entrenamiento en comparación con la validación cruzada repetida sobre todo el conjunto, lo que podría afectar la capacidad del modelo para generalizar adecuadamente. No obstante, en conjuntos de datos de tamaño moderado o grande, como el actual, este problema es menos significativo que en conjuntos de pequeño tamaño y puede ser útil en términos de eficiencia computacional [31].

Teniendo en cuenta lo anterior, en este conjunto de datos caracterizado por la baja representación de la clase de interés (aproximadamente el 15 % de las muestras), el muestreo estratificado es crucial para dividir los datos en conjuntos de entrenamiento y prueba, tal como se menciona en el capítulo 5 de [5]. Este método asegura que ambas divisiones mantengan la misma proporción de clases que el conjunto original, lo que permite un entrenamiento y evaluación más equilibrados del modelo. Sin el muestreo estratificado, existe el riesgo de obtener conjuntos de datos sesgados, lo que podría llevar a un modelo que no generaliza bien y a evaluaciones de rendimiento poco fiables.

Para este fin se emplea la función `train_test_split` del módulo `model_selection` de la librería `sklearn` y se selecciona la opción stratify sobre la variable de objetivo para realizar la sepación del conjunto de datos. Adicionalmente, para evitar obtener conjuntos distintos cada vez que se ejecuta el código se definio el parametro **random_state** en ‘42’. A continuación, se preseta la distribucion de clases en los datos de entrenamiento y prueba.



(a) Conjunto de datos de entrenamiento.

(b) Conjunto de datos de prueba.

Figura 4.7: Distribución de muestras de la variable objetivo **Revenue** en los conjuntos de entrenamiento y prueba obtenidos.

Target Encoding

La aplicación de target encoding puede ser particularmente útil para las variables categóricas que muestran una significancia estadística fuerte con la variable objetivo. En este caso, y como se ha evidenciado en la sección 4.2.3 las variables **OperatingSystems**, **Browser**, y **TrafficType** tienen un valor p (p-value) muy bajo, lo que indica una relación significativa con la variable objetivo y sugiere que el target encoding podría capturar esta relación de manera efectiva.

Para realizar este tipo de codificación se empleo la función `TargetEncoder` de la libreria `category_encoders` creando un encoder con la información del conjunto de datos de entrenamiento. Los valores resutlantes son mostrados en la Figura 4.8.

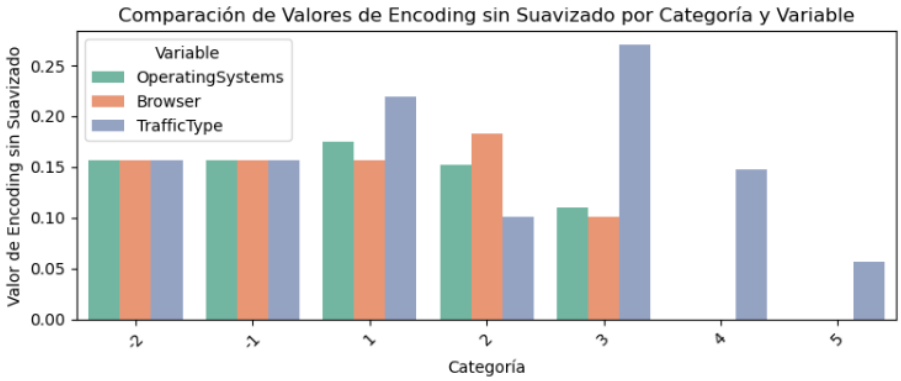


Figura 4.8: Valores resultantes de codificación por target en variables categóricas **OperatingSystems**, **Browser**, y **TrafficType**

De Figura 4.8 y apoyandonos en los valores mostrados en Tabla 4.3, se identificaron grupos importantes en las variables categóricas clave. Por ejemplo, en la variable **OperatingSystems**, se observaron diferencias significativas en la propensión de compra entre los sistemas operativos 2, 4 y 5 (valor codificado de 0.175), y los sistemas 3 y 6 (valor codificado de 0.152). En la variable **Browser**, se encontró que los navegadores 10 y 5 (valor codificado de 0.156) y los navegadores 12 y 13 (valor codificado de 0.182) mostraron impactos sustanciales en el comportamiento de compra. Además, en **TrafficType**, se evidenció una variación considerable en las categorías 10, 11, 2 y 5 (valor codificado de 0.220) y las categorías 13 y 19 (valor codificado de 0.270), indicando diferentes patrones de interacción del tráfico con el sitio web en relación con la intención de compra.

Por otro lado, es importante aclarar que en un proceso de target encoding, los valores especiales como -1 y -2 se interpretan así:

- **Valor -1:** Se asigna a categorías presentes en el conjunto de prueba pero no en el de entrenamiento, indicando que no hay suficiente información para un encoding basado en el objetivo.
- **Valor -2:** Representa categorías extremadamente raras o con poca representación

en el entrenamiento, o categorías con datos faltantes. Este valor indica que el encoding puede no ser fiable debido a la escasez de datos.

Escalamiento de Variables

El escalamiento de datos es crucial al entrenar modelos como ANN, SVM, GBM, XGBoost y CatBoost, ya que mejora la convergencia y el rendimiento del modelo al asegurar que todas las características contribuyan de manera equilibrada. En SVM y ANN, el escalamiento es especialmente importante debido a su sensibilidad a la magnitud de las características. Para conjuntos de datos con outliers y desbalanceados, el uso de la clase `StandardScaler` puede ser beneficioso, ya que normaliza las características manteniendo la estructura de la distribución.

La implementación de este proceso se realizó definiendo un objeto `scaler` usando `StandardScaler` del módulo `preprocessing` de `sklearn` en los datos de entrenamiento. Esto incluye las variables numéricas y **Month**, que fue transformada a variable numérica. Este mismo objeto será usado posteriormente en la evaluación del modelo de ML elegido.

La selección de variables es crucial en la construcción de modelos de aprendizaje automático, ya que mitiga el sobreajuste al eliminar características irrelevantes o redundantes. La decisión sobre qué variables incluir se basa en análisis de correlación y características específicas de los datos (ver sección 4.2.3). Por ejemplo, variables como **ProductRelated**, **ProductRelated_Duration** y **PageValues** muestran una correlación significativa con la variable objetivo **Revenue**, sugiriendo su importancia, mientras que **SpecialDay** puede tener un impacto limitado.

Tras aplicar transformaciones y encodings a las variables categóricas, se dispone de 25 variables: 9 del one-hot encoding de **Region** y 16 del conjunto original. Este paso asegura que todas las características relevantes estén preparadas para el análisis o modelado.

El proceso se realizará sobre el conjunto de datos de entrenamiento, enfatizando la evaluación en datos no vistos para evitar el sobreajuste. Se emplearán distintos métodos de selección de variables comparados mediante regresiones logísticas y la métrica ROC AUC.

- **SAS:** Uso del proceso `logistic` y la opción de selección de variables `stepwise`.
- **SAS-EM:** Evaluación de nodos `Variable Selection`, `Gradient Boosting` y `Partial Least Squares`.
- **R:** Selección de variables usando stepwise bajo criterios AIC y BIC, y stepwise repetido bajo criterios AIC y BIC.

Los resultados y el código de SAS, junto con el código en R, están disponibles en el Anexo A.2. El diagrama utilizado en SAS-EM se muestra en la Figura4.9.

Durante el análisis, se observó que el conjunto de variables obtenido mediante stepwise repetido bajo criterio BIC es idéntico al obtenido con stepwise BIC, por lo que se

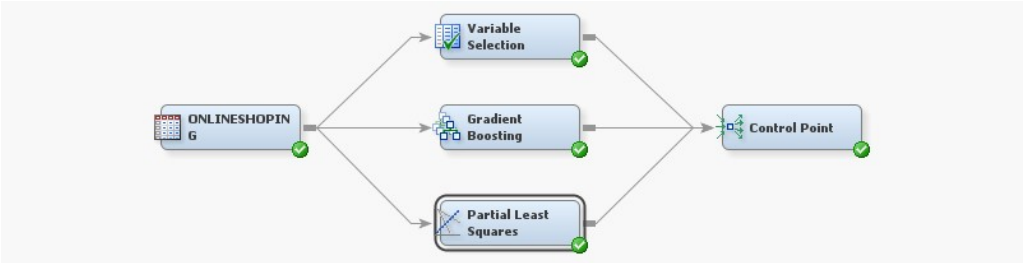


Figura 4.9: Diagrama utilizado para la selección de variables en SAS-EM.

consideran como un solo conjunto. Ambos conjuntos derivados de stepwise con criterio BIC contienen exactamente 5 variables, compartiendo 4 variables comunes: **PageValues**, **Month**, **ExitRates**, **TrafficType**. La única diferencia radica en la inclusión de **ProductRelated** en stepwise BIC y **ProductRelated_Duration** en stepwise repetido, ambas variables presentan una alta correlación de 0.86. Además, los conjuntos más extensos corresponden a las derivadas de stepwise AIC y al nodo de Gradient Boosting de SAS-EM, cada uno con 11 variables, las cuales también están presentes en las listas de BIC.

Posteriormente, se utilizó R para comparar los conjuntos de variables evaluando el rendimiento de modelos de regresión logística. Estos modelos fueron entrenados y evaluados mediante validación cruzada repetida para seleccionar el conjunto más prometedor de variables. Los resultados muestran valores de ROC AUC promedio alrededor de 0.89, demostrando una buena distinción entre las clases. Dado que todos los modelos, excepto el conjunto de variables obtenido usando el nodo `Partial Least Squares` de SAS-EM, tienen valores aproximados de 0.889, se presenta el diagrama de cajas comparativo de todos los modelos excepto este en la Figura 4.10 (el grafico con todos los modelos se encuentra en la Figura A.17 de la sección A.2).

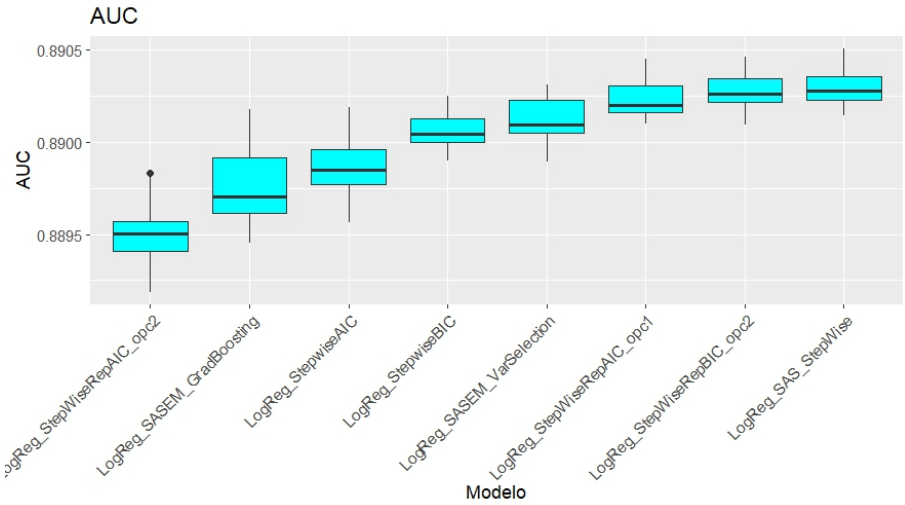


Figura 4.10: Comparación de ROC AUC de regresiones logísticas usando todos los conjuntos de variables, excluyendo el método PLS de SAS-EM.

Usando la información presentada en Figura 4.10, el conjunto de variables seleccionado fue obtenido mediante el proceso `logistic` de SAS con selección stepwise. Este

conjunto incluye **PageValues**, **Month**, **ExitRates**, **TrafficType**, **ProductRelated**, y **SpecialDay**. El modelo resultante alcanzó un ROC AUC de 0.8903 y una tasa de fallos de 0.1186. A forma de información adicional, en la Tabla A.6 se detallan los resultados de todos los modelos de regresión logística entrenados con los distintos conjuntos de selección de variables evaluados.

4.2.5. Elección de la Tarea de Mineria de Datos Adecuada

Este es el quinto paso del proceso KDD, en el cual se elige una tarea de minería de datos adecuada basada en los objetivos específicos definidos en el primer paso. En este caso, la meta es clara y consiste en predecir con precisión la actividad de compra en una sesión, utilizando técnicas de clasificación. Las variables predictoras, previamente transformadas utilizando codificadores como ordinal, one-hot y target, capturan la complejidad y la variabilidad del comportamiento del usuario en entornos de comercio electrónico. Dado el desbalance de clases y la presencia de outliers, se anticipa que modelos black box como ANN, SVM, RF, GBM, XGBoost y CatBoost superen significativamente a métodos de caja blanca como la regresión logística, ofreciendo predicciones más robustas y precisas en este contexto dinámico y variado.

4.2.6. Elección del Metodo de Machine Learning Adecuado

En el sexto paso de KDD se centra en la selección de un modelo adecuado para nuestra clasificacion. En primer lugar, se entrenaran modelos utilizando validación cruzada con 5 grupos, empleando los métodos ANN (de una sola capa), SVM (kernels lineal, polinomial y RBF), RF, GBM, XGBoost y CatBoost. Para implementar la validación cruzada, se utilizará la clase `StratifiedKFold` del módulo `model_selection` de `sklearn` con `n_splits=5`, `shuffle=True` y `random_state=42`. Además, se empleará `GridSearchCV` para una búsqueda exhaustiva de parámetros, usando `scoring='roc_auc'` para evaluar los modelos según el criterio de ROC AUC. También se han definido las funciones `plot_metric_evolution`, `plot_metric_evolution_v2` y `plot_metric_with_marker_legend` para facilitar la visualización y ajuste de hiperparámetros de los modelos.

Como primer paso, se construirá un modelo de regresión logística como referencia para la comparación con los demás métodos. Los coeficientes obtenidos se presentan en la Figura 4.11.

Una vez completado el modelo de regresión logística de referencia, se procederá a ajustar los parámetros de la red neuronal como el primer método tipo caja negra a ser explorado. Para este fin se usará la clase `MLPClassifier` y se definirá un `random_state=42` y se utilizará como algoritmo de optimización a **ADAM** (Adaptive Moment Estimation), el cual es uno de los algoritmos optimizadores más utilizados en el entrenamiento de redes neuronales debido a su capacidad para manejar grandes cantidades de datos y su eficiencia en la actualización de parámetros.

Según la Figura 4.7, el conjunto de entrenamiento consta de 9508 observaciones, de

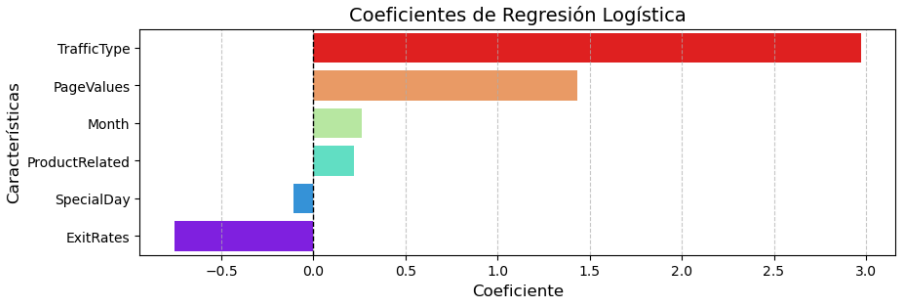


Figura 4.11: Coeficientes de la regresión logística.

las cuales 1493 pertenecen a la clase de interés. Para determinar el número adecuado de nodos a probar, se considerará el principio de tener al menos 10 observaciones por parámetro. Basado en este principio, se explorarán entre 10 y 30 observaciones por parametro, resultando en un rango de 5 a 14 nodos después de la evaluación. Se realizará una primera iteración utilizando la clase `GridSearchCV` para definir un rango del parámetro `max_iter` a ser usado (correspondiente al número máximo de iteraciones), para posteriormente realizar una iteración más detallada para definir los mejores valores de los demás parámetros (número de nodos y tasa de aprendizaje). A forma de ejemplo se presenta la ayuda visual obtenido en el proceso de ajuste de hiperparametros usando la función `plot_metric_evolution` en Figura 4.12.

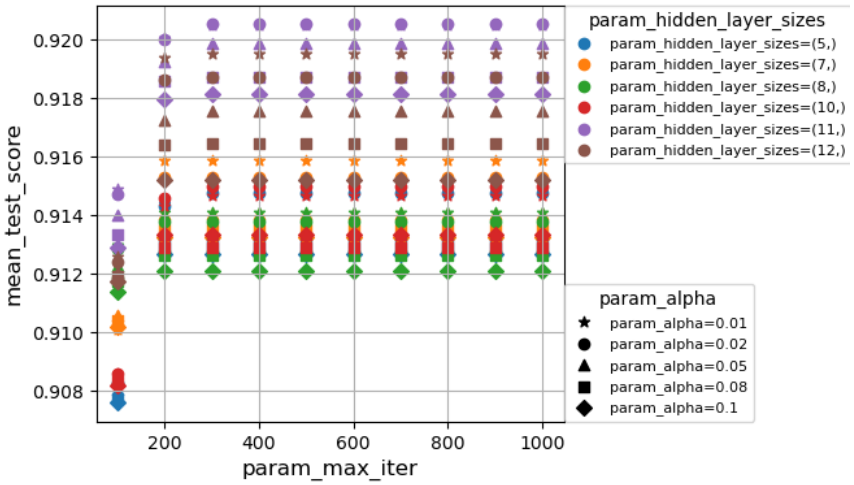


Figura 4.12: Representacion grafica de ajuste de parametros para modelo de red neuronal

Los parámetros elegidos para este modelo fueron:

- `alpha`: 0.02 (coeficiente de aprendizaje).
- `hidden_layer_sizes`: (11,) (número de nodos en cada capa, denotando en este caso el uso de 11 nodos y una sola capa).
- `max_iter`: 300 (número de iteraciones).

El siguiente conjunto de modelos a ajustar pertenece a la familia SVM. Se construyeron tres variantes: una con kernel lineal, otra con kernel polinomial y finalmente una

con kernel RBF. Para el modelo con kernel lineal, el único parámetro ajustado fue `C`, que equilibra el margen más amplio y la clasificación correcta de los puntos de entrenamiento. En el modelo con kernel polinomial, además de `C`, se ajustó el parámetro `degree`, manteniendo fijo el valor de `coef0` en 3. Para el modelo con kernel RBF, se ajustaron los parámetros `C` y `gamma`, que define el alcance de la influencia de un punto de entrenamiento. Todos los modelos se ejecutaron con `random_state=1234` (diferente al usado en los demás modelos entrenados), manteniendo el mismo k-fold definido para el resto de modelos y siguiendo un proceso similar al de la red neuronal. A modo de ejemplo, se presenta la ayuda visual del ajuste de parámetros del SVM con kernel RBF en la Figura 4.13):

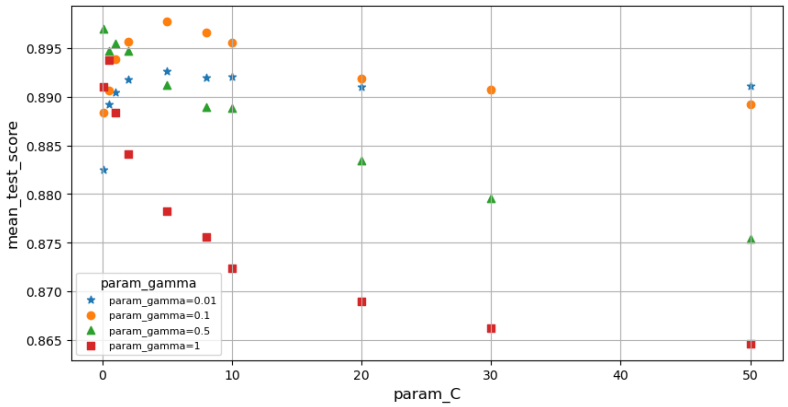


Figura 4.13: Representacion grafica de ajuste de `C` y `gamma` para SVM de Kernel RBF.

Los parámetros elegidos para estos modelos fueron:

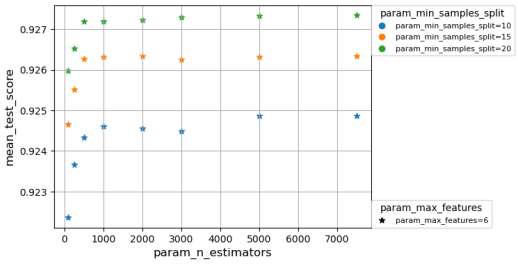
- **SVM con Kernel lineal:** `C=10`
- **SVM con Kernel polinomial:** `C=1`, `degree=3`, `coef0=3`
- **SVM con Kernel RBF:** `C=5`, `gamma=0.1`

El siguiente modelo a ser ajustado fue Random Forest. Dado que el número de observaciones de la clase de interés no es muy elevado, se decidió establecer un máximo de 30 para el parámetro `min_samples_split`, pero no se consideraron valores superiores a 50. Por esta razón, se probaron valores de 10, 15 y 20 para `min_samples_split` (equivalente en R a `nodesize`).

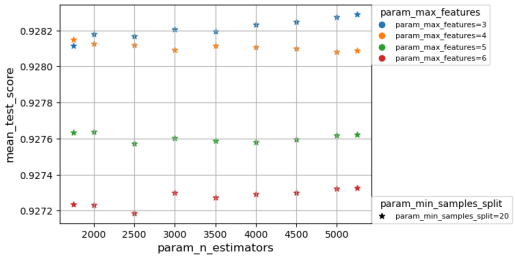
Para asegurar que cada fold de validación cruzada contenga una distribución equitativa de observaciones, se utilizó el método `cv`, dejando fuera un fold con un tamaño de aproximadamente $\frac{9508}{5} \approx 1902$ observaciones en cada ejecución de la validación cruzada. Esto implica que se utilizan $\frac{4}{5} \times 9508 \approx 7606$ observaciones para entrenar el modelo. Por lo tanto, el tamaño máximo de la muestra de entrenamiento (`sampsize`) debe ser aproximadamente 7606.

Dado lo anterior, se dejó un margen y se definió un máximo de 7500 para la primera prueba de ajuste de modelo. Una vez finalizado ese proceso se podran ajustar los demás parametros del modelo. La clase a implementar en este caso es `RandomForestClassifier` del módulo `ensemble` de `sklearn`. Los resutlados se presentan en Figuras

4.14.



(a) Representacion grafica de ajuste inicial de `n_estimators` y `min_samples_split`



(b) Representacion grafica de ajuste final de `n_estimators` y `max_features`

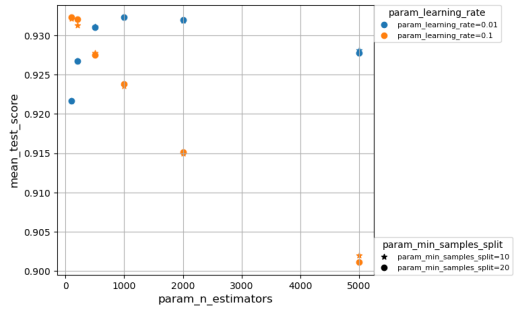
Figura 4.14: Representacion grafica de ajuste de parametros para modelo de RF

Los parametros defindos a usar para este modelo son:

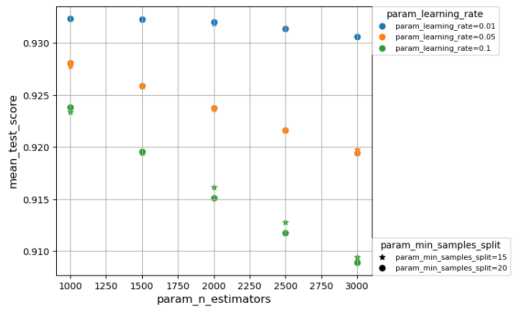
`n_estimators`=5250, `max_features`=3, `min_samples_split`=20

Indicando que se trata de un modelo RF y no bagging la mejor solución (de un conjunto de 6 variables el valor optimo de `max_features` fue 3).

En el caso de GBM (representado en este caso por el método GBTR), se empleó la clase `GradientBoostingClassifier`, en donde se hicieron pruebas para ajustar los parámetros `learning_rate`, `min_samples_split` y `n_estimators`. El proceso se dividió en un primer ajuste para determinar un rango de `n_estimators` propicio para aplicar un ajuste fino de `learning_rate` y `min_samples_split`. La Figura 4.15 muestra el proceso.



(a) Ajuste de rango de `n_estimators`.



(b) Ajuste final de todos los parámetros.

Figura 4.15: Representación gráfica del ajuste de parámetros para el modelo de GBM

Los parámetros definidos a usar para este modelo son:

`n_estimators`=1000, `min_samples_split`=20, `learning_rate`=0.01

En el caso de XGBoost, se empleó la clase `XGBClassifier` de la librería `xgboost`. Para este modelo se ejecutaron varios procesos usando `GridSearchCV` enfocándose primero en definir el rango de los parámetros `n_estimators`, `min_child_weight` y `learning_rate`, y a medida que se iba definiendo un rango para estos valores, se probaron distintos valores de los demás parámetros (`max_depth`, `gamma`, `colsample_bytree`, `subsample`). Los parámetros definidos a usar para este modelo son:

`n_estimators`=700, `min_child_weight`=15, `max_depth`=4, `learning_rate`=0.008, `gamma`=0, `colsample_bytree`=1, `subsample`=1

En el caso de CatBoost, se empleó la clase `CatBoostClassifier` de la librería `catboost`. El proceso fue muy similar al realizado para XGBoost. Primero se realizó un ajuste del rango de búsqueda para los parámetros `n_estimators` y `learning_rate`, para luego seguir con el ajuste de los demás parámetros (`l2_leaf_reg`, `depth`=6, `colsample_bylevel` y `subsample`).

Los parámetros definidos a usar para este modelo son:

`n_estimators`=1250, `l2_leaf_reg`=10, `depth`=6, `learning_rate`=0.01, `colsample_bylevel`=0.8, `subsample`=1

Después de realizar el ajuste de parametros correspondiente de todos los módelos creamos una lista con los parametros asociados a cada modelo y se emplea la clase `cross_validate` del modulo `model_selection` de `sklearn`. Para esto volveremos a definir el kfold con 5 separaciones, de forma estratificada y con un `ranodom_state=42` (igual que como se entrenaron los modelso previamente). Con estos resultados usamos algunas clases adicionales del modeulo `metrics` para evaluar metricas de interes con **ROC AUC**, **Accuracy**, **Recall** y **F1-Score**. Los resultados de **ROC AUC** la cual sera usada como metrica principal para la evalaucion de modelos se presenta en Figura 4.16, mientras Tabla 4.5 presenta el valor medio de todas las metricas asociadas a los modelos obtenidos.

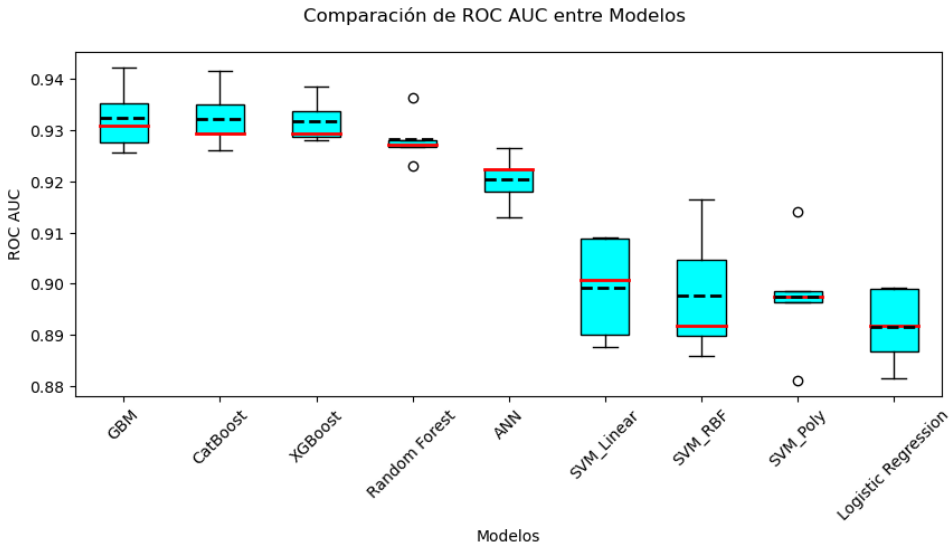


Figura 4.16: Comparación de desempeño de métrica ROC AUC de los modelos entrenados en Python.

La métrica de interés principal en esta comparación es ROC AUC. Los modelos basados en árboles presentan el mejor desempeño, con GBM y CatBoost liderando con un ROC AUC de 93.23, seguidos de XGBoost con 93.17. En contraste, el peor desempeño corresponde a la regresión logística, con un ROC AUC de 89.16. La precisión (accuracy)

Tabla 4.5: Valor medio de las métricas calculadas de los modelos entrenados

ML Model	ROC AUC Mean	ROC AUC STD	Accuracy Mean	Accuracy STD	F1-Score Mean	Recall Mean	Precision Mean
GBM	93.23	0.60	90.22	0.62	66.40	61.55	72.11
CatBoost	93.23	0.54	90.06	0.48	65.58	60.35	71.86
XGBoost	93.17	0.40	90.07	0.56	65.75	60.75	71.71
Random Forest	92.83	0.44	89.87	0.50	64.85	59.54	71.25
ANN	92.05	0.46	89.44	0.50	65.02	62.49	67.77
SVM Linear	89.92	0.91	88.43	0.27	53.00	41.60	73.22
SVM RBF	89.77	1.13	89.34	0.29	61.78	54.92	70.63
SVM Poly	89.75	1.05	89.14	0.31	59.36	50.64	71.94
Logistic Regression	89.16	0.68	88.13	0.22	49.45	37.04	74.58

y el recall de los modelos también son notables: GBM muestra una precisión del 90.22 % y un recall del 61.55 %, mientras que la regresión logística tiene una precisión del 88.13 % y un recall del 37.04 %. Estos resultados eran esperados debido a que solo el 15 % de las muestras pertenecen a la clase de interés, lo que dificulta la clasificación precisa. Se realizarán las últimas pruebas con los modelos de la familia GBM (i.e., GBM, XGBoost y CatBoost) para afinar aún más el rendimiento.

Una vez reducido el conjunto de modelos a evaluar exclusivamente aquellos de la familia GBM, procederemos a comparar el rendimiento de los modelos entrenados utilizando solo las variables seleccionadas con aquellos que utilizan todas las variables disponibles. Como se mencionó en la sección 3.1.2 del capitulo 3, los modelos GBM poseen una cierta robustez frente al ruido y utilizan la naturaleza de árboles y técnicas de ensamblaje. Sin embargo, al evaluar modelos con un número reducido de variables para mitigar el riesgo de sobreajuste, podríamos perder información importante sobre las relaciones entre características que podrían mejorar nuestra predicción.

Con base en estas consideraciones, entrenaremos tres nuevos modelos y utilizaremos métricas como ROC-AUC para evaluar su desempeño. Esta evaluación nos permitirá determinar cuál enfoque, ya sea utilizando todas las variables o un conjunto seleccionado, optimiza mejor el balance entre rendimiento predictivo y explicabilidad del modelo en el contexto de la clasificación binaria.

En primer lugar se realizo un proceso de entrenamiento de los modelos usando todas las variables disponibles con lo cual se obtuvo:

- **Gradient Boosting Machine (GBM) con todas las variables:** `learning_rate=0.01`, `min_samples_split=20`, `n_estimators=750`, `random_state=42`
- **XGBoost con todas las variables:** `colsample_bytree=0.8`, `gamma=0`, `learning_rate=0.005`, `max_depth=6`, `min_child_weight=10`, `n_estimators=750`, `subsample=0.8`, `random_state=42`
- **CatBoost con todas las variables:** `colsample_bylevel=0.8`, `depth=6`, `l2_leaf_reg=10`, `learning_rate=0.01`, `n_estimators=1000`, `subsample=0.8`, `random_sta-`

`te=42, verbose=False`

En segundo lugar, se realizo un proceso similar que el realziado para evaluar los modelos entrenados previamente cuyor resultados se presetan en Figura 4.17:

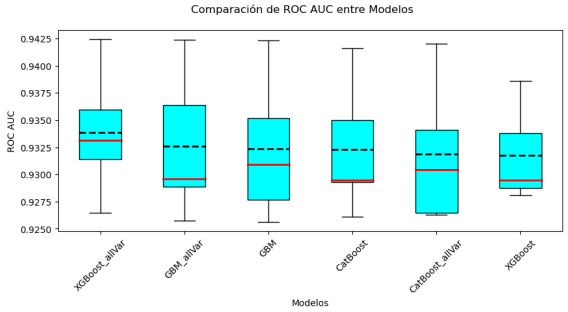


Figura 4.17: Comparación de desempeño de métrica ROC AUC de los modelos GBM comparando set de variables seleccionadas y set completo de variables disponibles

Los resultados obtenidos y presentados en la Figura 4.17 evidencian que los modelos con todas las variables y aquellos obtenidos solo con el conjunto de variables seleccionadas en el paso anterior tienen desempeños similares. Por esta razón, para las últimas dos pruebas se usarán los modelos realizados con el conjunto de variables elegidas previamente.

Por otra parte, al considerar que la clase de interés del conjunto representa solo un 15 % de las observaciones totales, realizamos algunas pruebas modificando el punto de corte (threshold en las tablas) para examinar cómo este puede mejorar las estadísticas de sensibilidad (**Recall**) y precisión (**Precision**). Los resultados se presentan en la Figura 4.18 y la Tabla 4.6.

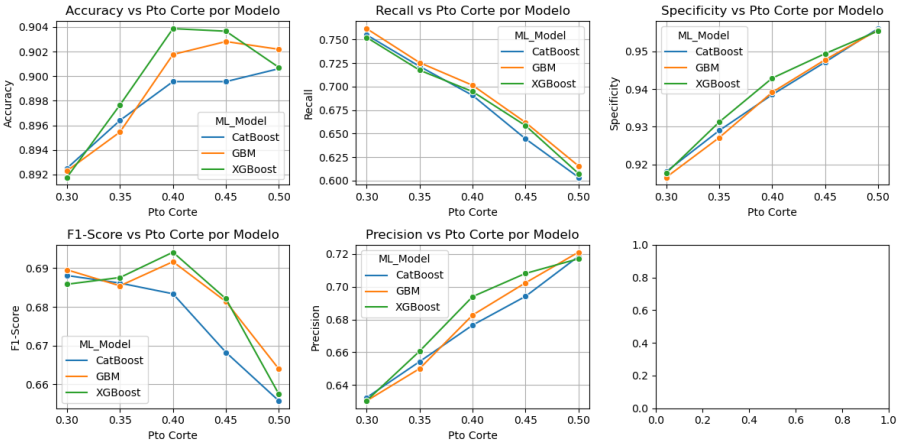


Figura 4.18: Variación de métricas en función del cambio de puntos de corte.

Los resultados presentados muestran que, al disminuir el punto de corte, la precisión (*accuracy*) y la especificidad (*specificity*) tienden a empeorar, mientras que la sensibilidad (*recall*) mejora. Los modelos CatBoost, GBM y XGBoost presentan un comportamiento similar en estas métricas. En particular, para un punto de corte de 0.40, XGBoost ofrece un

Tabla 4.6: Resultados de variación de puntos de corte

ML_Model	Pto Corte	ROC AUC	Accuracy	Recall	Specificity	F1-Score
CatBoost	0.30	0.9323	0.8925	0.7548	0.9182	0.6881
CatBoost	0.35	0.9323	0.8964	0.7213	0.9290	0.6861
CatBoost	0.40	0.9323	0.8996	0.6905	0.9385	0.6834
CatBoost	0.45	0.9323	0.8996	0.6443	0.9471	0.6683
CatBoost	0.50	0.9323	0.9006	0.6035	0.9560	0.6558
GBM	0.30	0.9323	0.8923	0.7615	0.9167	0.6896
GBM	0.35	0.9323	0.8955	0.7254	0.9271	0.6854
GBM	0.40	0.9323	0.9018	0.7013	0.9391	0.6917
GBM	0.45	0.9323	0.9028	0.6617	0.9477	0.6814
GBM	0.50	0.9323	0.9022	0.6155	0.9556	0.6640
XGBoost	0.30	0.9317	0.8918	0.7522	0.9178	0.6858
XGBoost	0.35	0.9317	0.8977	0.7173	0.9313	0.6876
XGBoost	0.40	0.9317	0.9039	0.6946	0.9429	0.6941
XGBoost	0.45	0.9317	0.9037	0.6584	0.9493	0.6821
XGBoost	0.50	0.9317	0.9007	0.6075	0.9553	0.6575

buen equilibrio entre precisión y sensibilidad, con un ROC AUC de 0.9317, una precisión de 0.9039 y una sensibilidad de 0.6946. Debido a su equilibrio entre las métricas clave, se elige utilizar el modelo XGBoost con un punto de corte de 0.40 para optimizar tanto la precisión como la sensibilidad.

Por último, cabe mencionar que existen varias técnicas para manejar problemas de clases desbalanceadas, como el *oversampling* aleatorio (*random oversampling*), el *subsampling* de la clase mayoritaria mediante NearMiss, y la penalización (*cost-sensitive learning*). Estas técnicas buscan mejorar el desempeño del modelo en la predicción de la clase minoritaria. Entre ellas, la penalización es especialmente eficiente porque ajusta el costo de los errores de predicción, asignando un mayor costo a los errores en la clase minoritaria sin alterar el conjunto de datos. Una técnica comúnmente utilizada en este contexto es la ponderación de muestras (*sample weighting*), que permite aplicar diferentes pesos a las muestras en función de su clase, facilitando así el ajuste del modelo para que tenga en cuenta el desbalance de clases. Debido a su eficacia, se realizarán pruebas en los modelos utilizando esta técnica de penalización.

El primer paso para implementar la estrategia de penalización en nuestras muestras es entrenar nuevamente los modelos utilizando el enfoque de ponderación que equivale a la penalización. En este caso, el objetivo es asegurar que las muestras positivas tengan un peso similar al de las negativas (por ejemplo, que el aproximadamente 15 % de representación de la clase positiva tenga un peso equivalente al de la clase negativa). Una forma de lograr esto es haciendo que las muestras positivas pesen 6.67 veces más que las negativas (es decir, 100/15).

Dado que los modelos de la familia GBM no disponen de un parámetro directo para la penalización en sus bibliotecas, a diferencia de la regresión logística, se emplea `fit_params={'sample_weight': np.where(y_train == 1, 6.67, 1.0)}` en el caso de GBM.

En cambio, XGBoost y CatBoost utilizan el parámetro `scale_pos_weight` para realizar este ajuste. Si se especifican tanto `scale_pos_weight` como `fit_params={'sample_weight': np.where(y_train == 1, 6.67, 1.0)}` en modelos como XGBoost o CatBoost, generalmente se considerará solo uno de estos métodos durante el entrenamiento del modelo. Ambos métodos están diseñados para cumplir la misma función: ajustar los pesos de las instancias de la clase positiva para manejar efectivamente el desbalance de clases.

Los parámetros seleccionados para los modelos con penalización fueron los siguientes:

- **Gradient Boosting Machine (GBM) con penalización:** `learning_rate=0.01`, `min_samples_split=15`, `n_estimators=1250`, `random_state=42`
- **XGBoost con penalización:** `colsample_bytree=0.8`, `gamma=0`, `learning_rate=0.005`, `max_depth=5`, `min_child_weight=20`, `n_estimators=1000`, `subsample=0.8`, `scale_pos_weight=6.67`, `random_state=42`
- **CatBoost con penalización:** `colsample_bylevel=1`, `depth=6`, `l2_leaf_reg=15`, `learning_rate=0.01`, `n_estimators=1500`, `subsample=1`, `scale_pos_weight=6.67`, `random_state=42`, `verbose=False`

Los resultados de la evaluación de métricas de los modelos con y sin penalización se presentan en la Tabla 4.7 (la comparación se realiza utilizando como referencia el punto de corte 0.5).//

Tabla 4.7: Métricas promedio de modelos sin penalización y con penalización

Modelo	ROC AUC	Accuracy	F1-Score	Recall	Precision	Penalización
GBM	0.932348	0.902188	0.663975	0.615522	0.721138	No
CatBoost	0.932300	0.900610	0.655756	0.603468	0.718590	No
CatBoost Penalización	0.931836	0.647666	0.465568	0.977228	0.305581	Sí
XGBoost	0.931747	0.900715	0.657487	0.607475	0.717124	No
GBM Penalización	0.931567	0.849285	0.642327	0.861339	0.512196	Sí
XGBoost Penalización	0.930698	0.623895	0.451238	0.984595	0.292700	Sí

Los modelos sin penalización, como GBM y CatBoost, muestran ROC AUC y precisión ligeramente superiores a 0.93 y 0.90 respectivamente, con buenos equilibrios entre F1-Score, Recall y Accuracy. Sin embargo, los modelos penalizados, aunque mejoran la sensibilidad, sacrifican significativamente el Accuracy y el equilibrio entre las métricas. Por ejemplo, CatBoost Penalización logra alta sensibilidad (0.977) pero baja precisión (0.305). En contraste, XGBoost mantiene un buen equilibrio general con ROC AUC de 0.9317, precisión de 0.9039 y sensibilidad de 0.6946, destacando como la opción preferible para un rendimiento balanceado en la clasificación.

Considerando que nuestro mejor modelo hasta el momento es XGBoost, destacamos su buen equilibrio con un ROC AUC de 0.9317, precisión de 0.9039 y sensibilidad de 0.6946 con un punto de corte de 0.4. Comparando con los modelos penalizados, el más destacado es CatBoost Penalización, que logra un ROC AUC ligeramente superior

(0.931836) pero con una precisión notablemente menor (0.647666) y una alta sensibilidad (0.977228). Sin embargo, este sacrificio en precisión indica que, dependiendo del contexto y los objetivos del modelo, XGBoost con punto de corte 0.4 sigue siendo preferible por su mejor balance entre las métricas de rendimiento.

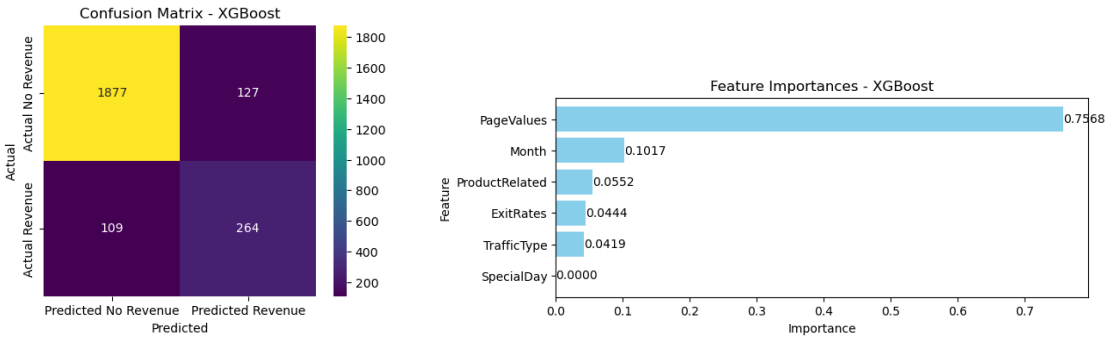
Basado en todas las consideraciones anteriores, hemos seleccionado el modelo final para nuestro problema, el cual es un modelo**XGBoost** con un **punto de corte** establecido en 0.40. La configuración específica del modelo incluye:

`n_estimators`=700, `min_child_weight`=15, `max_depth`=4, `learning_rate`=0.008, `gamma`=0, `colsample_bytree`=1, `subsample`=1

4.2.7. Implementación del Método de Machine Learning Seleccionado

Utilizando los parámetros definidos en la sección 4.2.6, configuramos y evaluamos un modelo XGBoost con datos de entrenamiento. Las métricas obtenidas fueron las siguientes: un ROC AUC de 0.9268, una precisión (accuracy) de 0.9007, un recall de 0.7078, una precisión (precision) de 0.6752 y un F1-Score de 0.6911.

Estos resultados indican una buena capacidad del modelo para discriminar y un equilibrio adecuado entre precisión y recall en las predicciones. La Figura 4.19 muestra la matriz de confusión del modelo, la cual ilustra tanto las clasificaciones correctas como los posibles errores de clasificación. Además, la figura presenta la importancia de cada característica en el modelo XGBoost, obtenida a partir del atributo `feature_importances_` del modelo entrenado. Estos análisis permiten una comprensión detallada del desempeño del modelo y de las características que más influyen en sus predicciones.



(a) Matriz de confusión. (b) Importancia de características.

Figura 4.19: Análisis de importancia de variables y desempeño del modelo XGBoost

La Figura 4.19b muestra que **PageValues** tiene un impacto significativo con un valor de 0.7568, mientras que variables como **Month**, **ProductRelated**, **ExitRates** y **BounceRates** presentan importancias más moderadas (entre 0.04 y 0.10). En contraste, **SpecialDay** tiene una importancia mínima de 0.0000. La Figura 4.19b detalla estos hallazgos.

Este análisis resalta cómo el modelo XGBoost interpreta y utiliza las características para realizar predicciones precisas, proporcionando información valiosa para la toma de

decisiones en aplicaciones prácticas.

4.2.8. Interpretación de los Patrones Minados

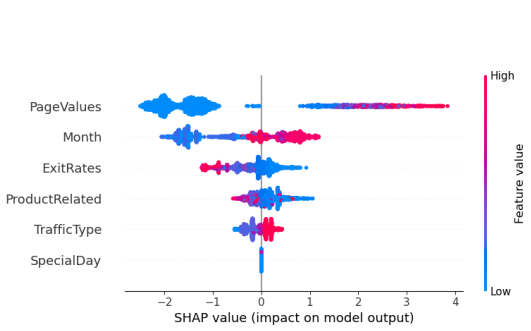
En este octavo paso del proceso de KDD, el enfoque se dirige hacia la interpretación y evaluación de los patrones descubiertos. Se utilizarán técnicas de XAI para este propósito, comenzando con análisis a nivel global mediante SHAP y PDP. Además, se investigarán las relaciones críticas entre las variables predictoras utilizando modelos de árboles surrogados basados en los valores SHAP.

Para mejorar la comprensión, también se llevará a cabo un estudio de casos acertados (True Positives y True Negatives) y un análisis de los errores en las predicciones. Este análisis se apoyará en los valores SHAP y se complementará con un estudio utilizando LIME en algunos de los casos analizados previamente con SHAP.

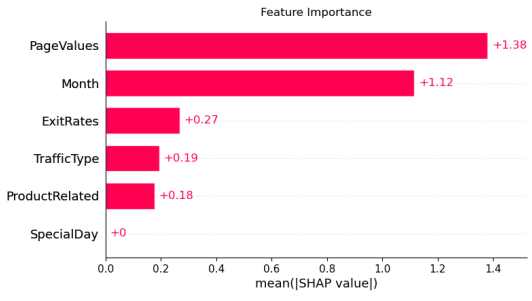
Métodos Agnósticos de Modelos Globales

En primer lugar, se emplea la librería `SHAP` en Python para calcular los valores **SHAP** utilizando `TreeExplainer`, específico para modelos de árboles como XGBoost. Posteriormente, se utiliza la función `shap.summary_plot` para visualizar la influencia de las características en las predicciones del modelo, seguida por un gráfico de barras que resume la importancia de cada característica. Estos gráficos proporcionan un buen punto de partida para entender qué atributos impactan más en las decisiones del modelo y facilitan la interpretación de su comportamiento.

La gráfica generada por `shap.summary_plot` se interpreta mediante los colores de los puntos: rojo indica un valor alto de la variable y azul un valor bajo. El eje X muestra su impacto en la predicción, donde valores positivos indican una influencia positiva y valores negativos una influencia negativa. El gráfico de barras utiliza el valor absoluto de los valores SHAP y los promedia para medir la importancia promedio absoluta de las características en la predicción. Los resultados se presentan en la Figura 4.20.



(a) Gráfico resumen de valores SHAP



(b) Importancia de las características según valores SHAP

Figura 4.20: Implementación de SHAP como método de modelo agnóstico global

Teniendo en cuenta las definiciones de las variables en la sección 4.2.2 y los resultados en la Figura 4.20, se evidencia que **PageValues** es la variable más influyente,

con un impacto positivo significativo en la probabilidad de compra, reflejado en un valor aproximado de +1.38. Esto sugiere que valores altos en **PageValues** incrementan considerablemente la probabilidad de compra. En la Figura 4.20a, altos valores SHAP de esta variable están asociados a altos valores de la misma, lo que permite concluir que un aumento en esta métrica incrementa las posibilidades de compra.

Month también tiene una influencia notable, con un valor cercano a +1.12, indicando que el mes de la sesión afecta considerablemente la probabilidad de compra, aunque su relación con la variable objetivo no es tan directa como en el caso de **PageValues**. **ExitRates** y **TrafficType** también contribuyen a las predicciones del modelo, con valores de +0.27 y +0.19, respectivamente. La relación inversa de **ExitRates** muestra que al incrementar su valor, disminuye la intención de compra, mientras que la relación de **TrafficType** se explica por el proceso de target encoding realizado para facilitar su uso en el modelo.

Las variables **ProductRelated** y **SpecialDay** muestran una influencia menor, con valores de +0.18 y 0.0 respectivamente. Aunque tienen algún impacto, su influencia en la predicción de compras es relativamente baja. En general, el análisis SHAP proporciona una visión clara de cómo cada característica contribuye al modelo y ayuda a identificar las áreas clave que influyen en la decisión de compra.

Al comparar los resultados de la Figura 4.19b con aquellos de la Figura 4.20b, se pueden apreciar algunas diferencias que se explican porque ambos resultados se obtienen de formas distintas: Figura 4.19b muestra la importancia calculada usando el modelo XGBoost basado en la contribución de cada característica durante el entrenamiento, mientras que SHAP mide el impacto marginal de cada característica en las predicciones. Ambos resultados destacan a **PageValues** y **Month** como las características más influyentes, aunque con valores distintos. Esto confirma que **PageValues** es crucial para el modelo, y **Month**, **ExitRates**, y **TrafficType** también juegan roles significativos, aunque en menor medida. La combinación de ambos enfoques proporciona una visión integral de la importancia de las características.

Asimismo, se pueden mostrar ciertas similitudes entre las tendencias mostradas en Figuras 4.20a y 4.11 (peso de los coeficientes del modelo de regresión usado para la comparación entre modelos en la sección 4.2.6). Los gráficos de valores SHAP y los coeficientes de regresión logística muestran que **PageValues** tiene una relación directa (positiva) significativa con la probabilidad de compra en ambos análisis, indicando que un aumento en **PageValues** incrementa la probabilidad de compra. **TrafficType** también presenta una relación directa en ambos gráficos, sugiriendo que ciertos tipos de tráfico aumentan significativamente la probabilidad de compra. **Month** muestra una relación directa con la probabilidad de compra según la época del año en ambos análisis. Por otro lado, **ExitRates** tiene una relación negativa en ambos gráficos, donde tasas de salida más altas disminuyen la probabilidad de compra. Estas similitudes proporcionan una validación cruzada de los hallazgos, destacando las características más influyentes en la probabilidad de compra.

La siguiente técnica XAI utilizada en este estudio es **PDP**. En este caso, se usó la función `PartialDependenceDisplay` del módulo `inspection` de la librería `sklearn` de Python. El proceso consiste en usar la clase importada mediante el comando `PartialDependenceDisplay.from_estimator` para generar gráficas PDP, mostrando cómo las características afectan las predicciones del modelo usando el conjunto de datos de prueba. Después, mediante el uso de indicadores, se seleccionan las características a evaluar y se generan tres gráficas: dos individuales y una de interacción.

Las gráficas individuales muestran cómo varía la probabilidad de la variable objetivo al cambiar cada característica, manteniendo las demás constantes. La gráfica de interacción muestra cómo cambian las predicciones cuando ambas características varían simultáneamente. Una línea ascendente sugiere un aumento en la probabilidad de la variable objetivo, mientras que una descendente indica una disminución.

Para evaluar el modelo y basados en los resultados obtenidos en la Figura 4.20, se decidió realizar gráficas de interacción entre **PageValues** y el resto de las variables, con excepción de **SpecialDay**, que mostró un impacto poco significativo en el modelo.

Las gráficas de dependencia parcial (PDP) complementan el análisis de SHAP al mostrar cómo varían las predicciones del modelo cuando se ajustan individualmente las características clave.

Primero, la gráfica de **PageValues** revela una relación positiva constante con la probabilidad de compra: un aumento en el valor de esta característica se asocia con un incremento en la probabilidad de compra, subrayando su importancia en el modelo.

Las gráficas de **Month** y **ExitRates** también muestran influencias significativas. La gráfica de **Month** evidencia variaciones en la probabilidad de compra según la época del año. Esto es esperado, ya que, como se presentó en la sección 4.2.3, meses como noviembre y octubre tienen una alta representación de observaciones asociadas a compras, mientras que meses como junio y marzo muestran una menor representación.

Por otro lado, la gráfica de **ExitRates** exhibe una relación negativa clara, donde tasas de salida más altas disminuyen la probabilidad de compra.

Las gráficas de interacción indican cómo **PageValues** interactúa con otras variables como **Month**, **ExitRates**, **TrafficType** y **ProductRelated**, proporcionando una comprensión más profunda de cómo múltiples variables afectan conjuntamente la probabilidad de compra. A pesar de que **Month** y **ExitRates** son importantes según la importancia de características obtenida del modelo XGBoost y SHAP, **PageValues** parece tener un impacto más directo y significativo. Esto sugiere que la ausencia de **PageValues** podría afectar notablemente la capacidad predictiva del modelo, ya que esta variable muestra una relación clara y fuerte con la probabilidad de compra.

Los resultados de las gráficas PDP realizadas se presentan en la Figura 4.21.

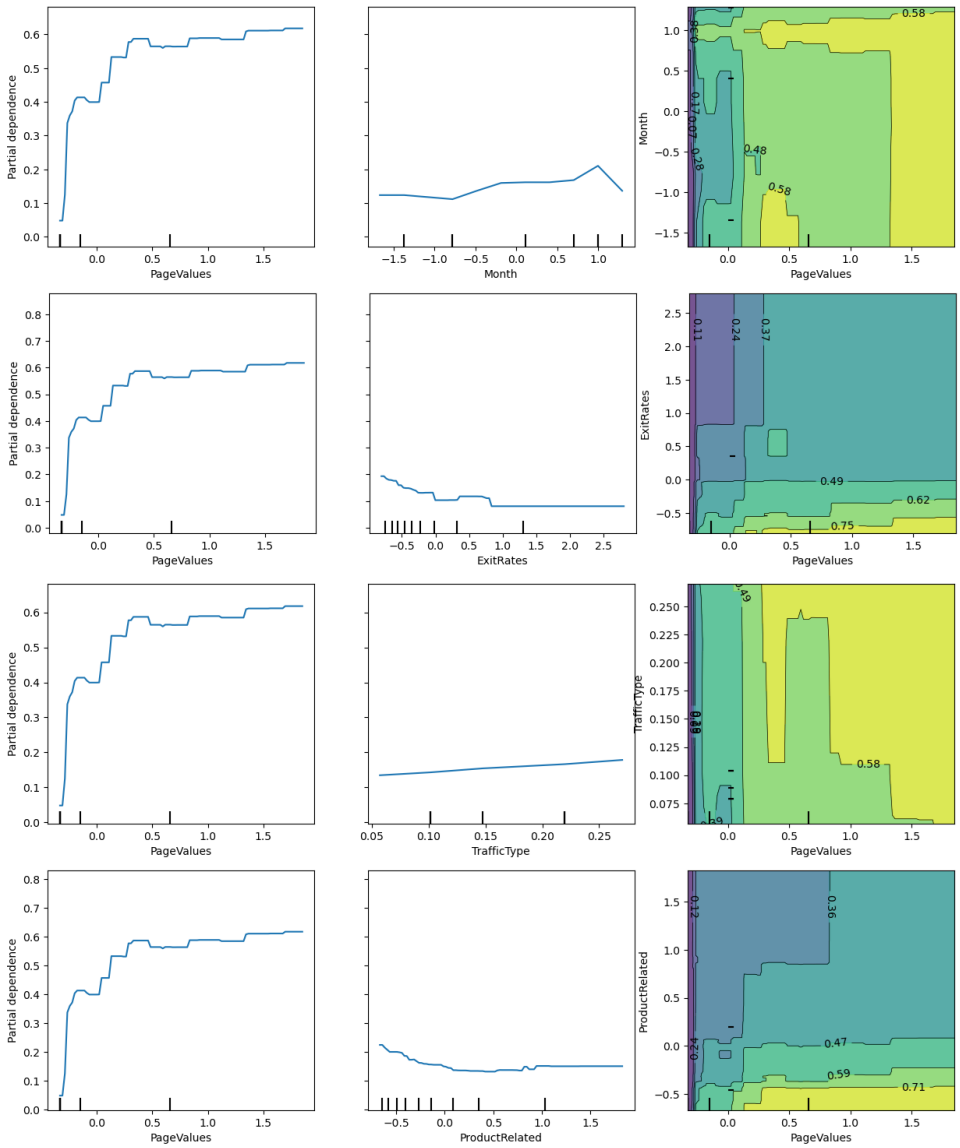


Figura 4.21: Análisis PDP y efecto de interacción de **PageValues** con otras variables

Finalmente, se emplea un arbol subrogado como modelo final para interpretar algunas de las relaciones realizadas por el modelo XGBoost contruido. Este enfoque se basa en simplificar la comprensión de las predicciones del modelo original. Después de haber entrenado el modelo XGBoost y obtenido los valores SHAP, que explican la contribución de cada característica en las predicciones, se procede a construir un árbol de decisión que imita el comportamiento del modelo XGBoost utilizando estos valores SHAP usando el modulo `DecisionTreeClassifier` de `sklearn` . Se inicializa un árbol de decisión con una profundidad máxima predefinida para evitar la sobrecomplicación y asegurar la interpretabilidad (en este caso se definio de 4). El árbol se entrena utilizando los valores SHAP como características y las predicciones del modelo XGBoost como etiquetas. Este paso asegura que el árbol subrogado aprenda a replicar las decisiones del modelo XGBoost.

Una vez entrenado, el árbol subrogado se utiliza para predecir etiquetas basándose en los valores SHAP, y su calidad se evalúa comparando sus predicciones con las del modelo XGBoost. Las métricas obtenidas son: precisión (0.985), ROC AUC (0.987), y un informe de clasificación detallado donde se resalta una exactitud general de 0.98, con un promedio macro de precisión 0.96, recall 0.99, y F1-score 0.97, y un promedio ponderado de precisión 0.99, recall 0.98, y F1-score 0.99. Estas métricas indican que el árbol subrogado replica de manera muy precisa el comportamiento del modelo XGBoost, con alta precisión y capacidad para manejar bien ambas clases.

Por ultimo, el árbol subrogado se visualiza, lo que permite una interpretación clara de cómo las características influencian las decisiones del modelo original. Esta visualización facilita la comprensión del modelo XGBoost al mostrar una estructura de decisión más sencilla y accesible, proporcionando una manera intuitiva y visual de entender el comportamiento y las decisiones del modelo XGBoost mediante la simplificación a través de un árbol de decisión. El resultado es mostrado en Figura 4.22.

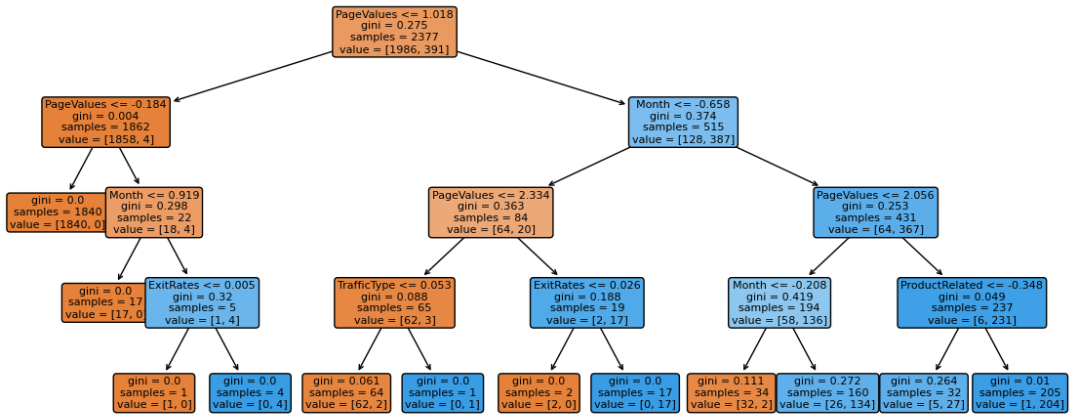


Figura 4.22: Representación grafica de modelo de arbol subrogado obtenido a partir de valores SHAP.

La Figura 4.22 muestra cómo los valores SHAP influyen en la predicción del modelo XGBoost para determinar si una sesión de usuario resultó en una transacción completada (1) o no (0). Las decisiones comienzan desde la raíz (arriba) y se dirigen hacia las hojas (abajo), con cada nodo representando una condición sobre un valor SHAP de una característica. Si la condición en el nodo se cumple, la decisión sigue la rama izquierda; de lo contrario, sigue la rama derecha. Los colores indican las clases predichas: naranja para sesiones sin transacción completada (0) y azul para sesiones con transacción completada (1).

Una relación importante es que valores SHAP de **PageValues** <= -0.184 indican que la sesión muy probablemente no resultó en una transacción completada. Si los valores SHAP de PageValues son mayores que -0.184, se requiere una evaluación adicional de los valores SHAP de **Month** y **ExitRates** para determinar el resultado. Otro punto clave es

que cuando los valores SHAP de **Month** son <= -0.658 y los de **PageValues** son >1.018, la predicción se inclina mayormente hacia una transacción completada (1). Además, si los valores SHAP de **PageValues** son >-0.184 y los de **TrafficType** son <= 0.053, la predicción es principalmente que no hubo transacción completada (0). Estas relaciones destacan las combinaciones críticas de valores SHAP que el árbol utiliza para imitar el comportamiento del modelo XGBoost en la predicción de transacciones completadas, proporcionando una interpretación clara y visual de cómo cada característica afecta las decisiones del modelo.

Métodos Agnósticos de Modelos Locales

Para ilustrar cómo se interpretan las predicciones de un modelo XGBoost a nivel local, se seleccionaron cinco casos de cada una de las siguientes clases: verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos. Utilizando los índices de estas predicciones, se generaron visualizaciones tipo “force plot” con SHAP para cada uno de los casos seleccionados. Estas gráficas muestran cómo diferentes características influyen en las predicciones del modelo.

Verdaderos Positivos: Las gráficas ”force plot”para los casos de verdaderos positivos, donde las sesiones de usuario resultaron en una transacción completada, revelan que **PageValues** es consistentemente la característica más influyente. Generalmente, altos valores de **PageValues** incrementan significativamente la probabilidad de una transacción. **Month** y **ExitRates** muestran influencias mixtas, con **Month** a veces aumentando y otras disminuyendo la probabilidad de compra. **TrafficType** y **ProductRelated** también tienen un impacto considerable, aunque varía entre los casos. Los casos evaluados se presentan en la Figura 4.23.

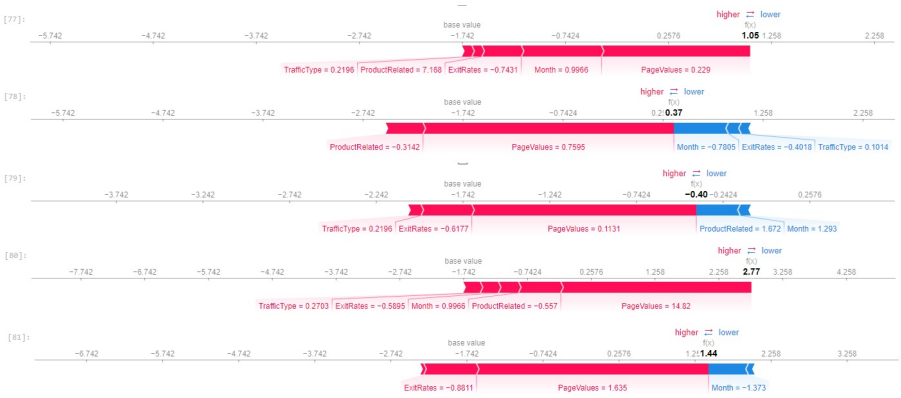


Figura 4.23: Force Plots de SHAP para ejemplos de verdaderos positivos (TP).

Evaluando la información capturada en la Figura 4.23, podemos obtener insights sobre las predicciones. Por ejemplo, en el ítem 77, **PageValues**, **Month**, **ExitRates**, **ProductRelated** y **TrafficType** impulsan la predicción hacia una alta probabilidad de compra (predicción final 1.05). En el ítem 78, aunque **PageValues** aumenta la predicción, **Month** y **ExitRates** la disminuyen, resultando en una predicción final positiva (0.37). Finalmente, en el ítem 80, **PageValues**, **Month** y **ProductRelated** contribuyen significativamente a

una alta probabilidad de compra (2.77).

Verdaderos Negativos: Para los verdaderos negativos, donde las sesiones no resultaron en una transacción completada, **PageValues** y **Month** son las características que más consistentemente reducen la probabilidad de compra. **ExitRates** también juega un papel importante, generalmente disminuyendo la probabilidad de una transacción completada. Los casos evaluados se presentan en la Figura 4.24.

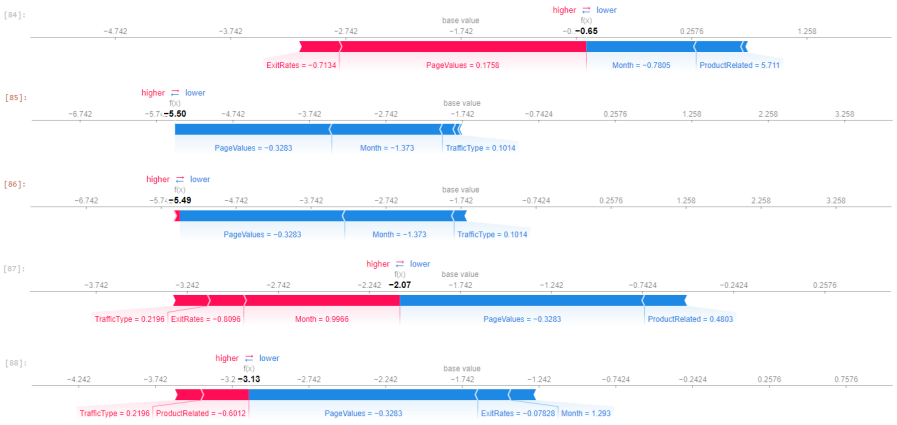


Figura 4.24: Force Plots de SHAP para ejemplos de verdaderos negativos (TN).

Evaluando la información capturada en la Figura 4.24, se pueden obtener varios insights interesantes. En el ítem 87, **TrafficType**, **ExitRates** y **Month** aumentan la predicción, pero **PageValues** y **ProductRelated** la reducen significativamente, resultando en una baja probabilidad (-2.07). En el ítem 84, **ExitRates** y **PageValues** disminuyen la probabilidad, con una influencia mixta de **Month** y **ProductRelated** (-0.65).

Falsos Positivos: En los casos de falsos positivos, donde las sesiones no resultaron en una transacción pero fueron predichas como si lo hubieran hecho, **PageValues** y **Month** son las características que más influyen en las predicciones incorrectas.

Resaltando algunos insights de los ejemplos evaluados, en el ítem 92 se observa que **PageValues** y **Month** contribuyen a una alta predicción, a pesar de la influencia negativa de **ExitRates** (1.25). Por otra parte, el ítem 93 es interesante porque, aunque el valor final de la predicción según el force plot de SHAP es -0.07, al convertirlo en una probabilidad y compararlo con el punto de corte del modelo original (0.4), se considera como una predicción positiva (0.4825). Esto muestra que el cambio de punto de corte en este caso fue desventajoso en este caso. Los casos estudiados se presentan en la Figura 4.25.

Falsos Negativos: En los casos de falsos negativos, donde las sesiones resultaron en una transacción pero fueron predichas como si no lo hubieran hecho, **PageValues** y **Month** son las características que consistentemente reducen la probabilidad de compra. A pesar de las contribuciones positivas de otras características, los valores negativos de

Figura 4.25: Force Plots de SHAP para ejemplos de falsos positivos (FP).

PageValues y los efectos mixtos de **Month** tienden a influir en la dirección opuesta. Los resultados se presentan en la Figura 4.26.

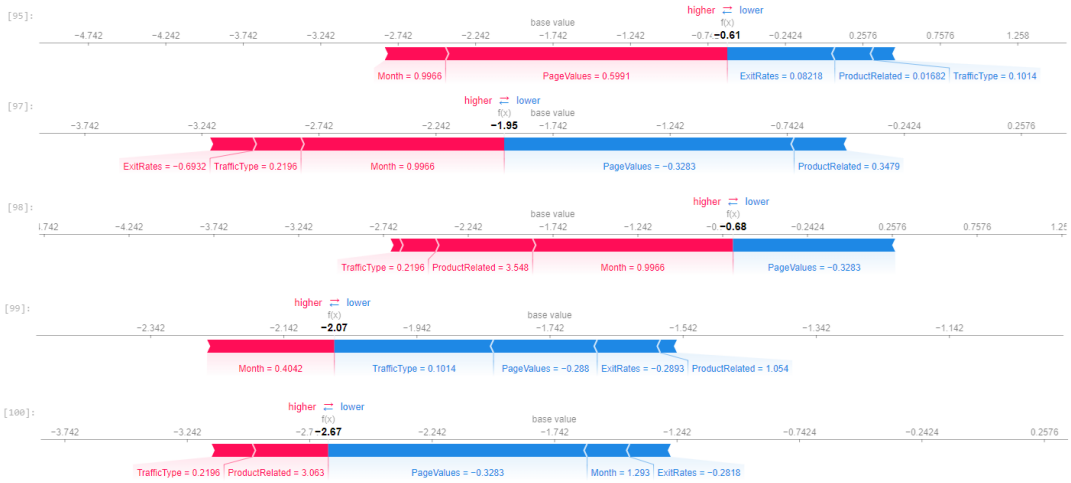


Figura 4.26: Force Plots de SHAP para ejemplos de falsos negativos (FN).

Evaluando la información capturada en la Figura 4.26, se observan varios insights. En el ítem 97, **ExitRates** (-0.6932) tiene una fuerte influencia negativa, y aunque **TrafficType** (0.2196) y **Month** (0.9966) tienen influencias positivas, **PageValues** (-0.3283) y **ProductRelated** (0.3479) contribuyen a una predicción final negativa (-1.95). El ítem 100 muestra que **TrafficType** (0.2196) y **ProductRelated** (3.063) aumentan la probabilidad de compra, pero las influencias negativas de **PageValues** (-0.3283) y **ExitRates** (-0.2818) resultan en una predicción final de -2.67.

Otra visualización útil de SHAP en Python es el gráfico de cascada (“Waterfall plot”), que no solo muestra la magnitud y dirección de la contribución de cada característica, sino también permite observar el proceso acumulativo de cómo se llega a la predicción final. Usando como ejemplo el caso del falso positivo identificado como ítem 93, se puede crear un gráfico de cascada utilizando la librería SHAP. Esto se hace empleando la

clase `shap.Explanation` para encapsular los valores SHAP, el atributo `expected_value`, los datos de la instancia, y la función `shap.plots.waterfall` para generar el gráfico. El resultado para este ejemplo se muestra en la Figura 4.27.

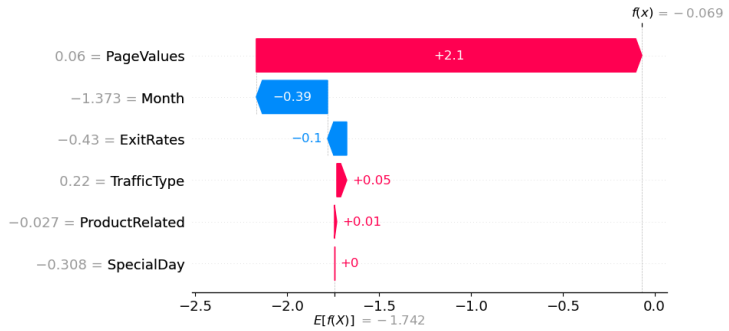


Figura 4.27: Ejemplo de waterfall plot de SHAP.

Figura 4.27 aporta información adicional a la suministrada en el forceplot mostrado en Figura 4.25, mostrando cómo cada característica contribuye secuencialmente a la predicción final. En el ítem 93, se observa que **PageValues** tiene la mayor contribución positiva (+2.1), mientras que **Month** (-0.39) y **ExitRates** (-0.1) reducen la predicción. Además, revela pequeñas contribuciones de **TrafficType** (+0.05) y **ProductRelated** (+0.01). Esta visualización clarifica el impacto acumulativo de cada característica, mejorando la comprensión de por qué la predicción final fue positiva (el valor final de -0.069 se compara con el punto de corte (threshold) del modelo para determinar la clase final. En este caso, la predicción final se traduce en una probabilidad de 0.4825, lo que supera el umbral de 0.4, resultando en una clasificación positiva).

Finalmente, se utilizará LIME para explicar el mismo caso y mostrar cómo se aplica este método para la interpretación de predicciones locales. Para realizar este análisis, se emplea la librería `LIME` (Local Interpretable Model-agnostic Explanations) para generar explicaciones locales. Primero, se crea un `LimeTabularExplainer` con los datos de entrenamiento, especificando los nombres de las características y las clases. Luego, se selecciona una instancia específica de los datos de prueba (X_test) para explicar su predicción (en este caso, se evalúa el ítem 93). La explicación se obtiene mediante el método `explain_instance`, que utiliza la probabilidad predicha por el modelo `(model_final.predict_proba)`. La instancia de prueba, sus predicciones y las explicaciones locales por característica se imprimen, y también se muestra el modelo interpretable local. Finalmente, se presenta la explicación mediante `exp.show_in_notebook`, expuesta en la Figura 4.28.

La Figura 4.28 muestra que el modelo subyacente utilizado por LIME es un `XGBClassifier` de XGBoost, correspondiente al modelo propuesto. La instancia de prueba incluye las características **PageValues** (0.059780), **ProductRelated** (-0.027322), **Traf-**

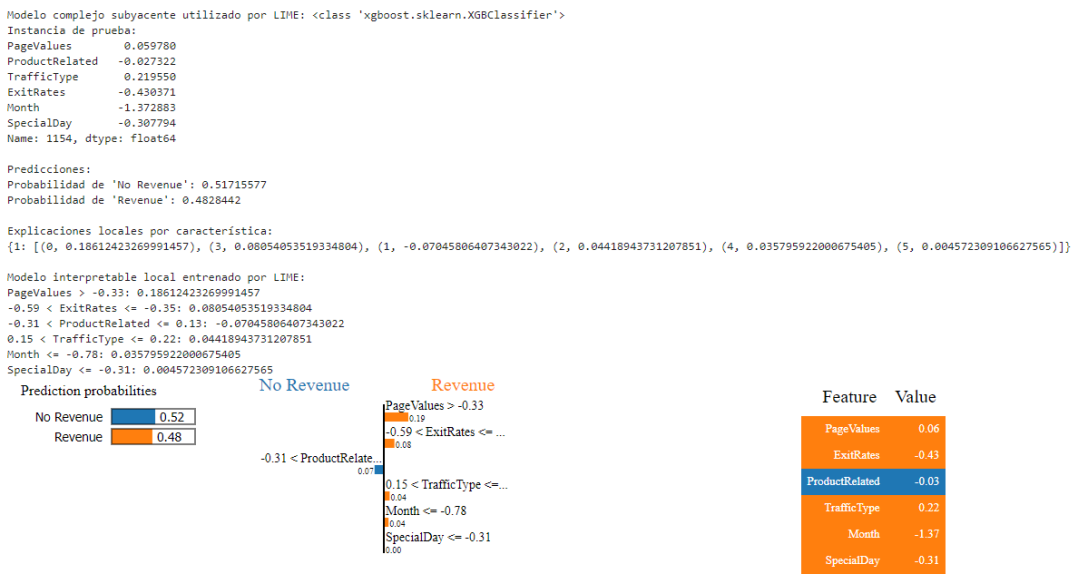


Figura 4.28: Ejemplo de explicaciones LIME de un falso positivo.

ficType (0.219550), **ExitRates** (-0.4304), **Month** (-1.372883) y **SpecialDay** (-0.307794). Las predicciones muestran una probabilidad de 0.517 para 'No Revenue' y 0.483 para 'Revenue', indicando una ligera preferencia hacia la clase 'No Revenue'. Sin embargo, si el punto de corte del modelo es 0.4 en lugar de 0.5, la predicción sería 'Revenue', ya que 0.483 supera el umbral de 0.4.

Las explicaciones locales obtenidas mediante LIME revelan cómo cada característica contribuye a la predicción. Por ejemplo, **PageValues** >-0.33 incrementa la probabilidad de 'Revenue' en +0.186, mientras que **ExitRates** <= -0.35 la disminuye en -0.081. El modelo interpretable local muestra que características como **Month** <= -0.78 y **SpecialDay** <= -0.31 también incrementan la probabilidad, con +0.036 y +0.005 respectivamente. Por otro lado, **ProductRelated** <= 0.13 y **TrafficType** <= 0.22 tienen un impacto mixto, reduciendo y aumentando la probabilidad en -0.070 y +0.044. Estos pesos reflejan la influencia de cada característica en la predicción del modelo.

Comparando con la explicación de SHAP para el ítem 93 presentado en la Figura 4.25, tanto SHAP como LIME identifican **PageValues**, **Month** y **ExitRates** como características influyentes, mostrando cómo **PageValues** contribuye positivamente mientras que **Month** y **ExitRates** reducen la predicción.

En conclusión, tanto SHAP como LIME proporcionan explicaciones detalladas y coherentes sobre la influencia de las características en las predicciones del modelo XGBoost. SHAP descompone la predicción en contribuciones individuales y convierte la predicción en una probabilidad, mientras que LIME entrena un modelo interpretable local que muestra los pesos de cada característica. Ambos enfoques resaltan la importancia de **PageValues**, **Month**, y **ExitRates** en la predicción de una transacción.

4.2.9. Uso del Conocimiento Descubierto

En este paso final del proceso KDD, se enfoca en la aplicación práctica del conocimiento descubierto para diversos propósitos, tales como la toma de decisiones estratégicas, la optimización de procesos y la mejora de productos y servicios. Los resultados obtenidos pueden ser compartidos con las partes interesadas para apoyar la toma de decisiones basada en datos, o integrarse con otros sistemas para permitir acciones adicionales, como la automatización de decisiones en tiempo real. Además, los insights obtenidos pueden servir para el desarrollo de nuevas políticas, la identificación de oportunidades de negocio y la mejora continua de los procesos existentes.

Para mejorar las recomendaciones sobre cómo aumentar el éxito de **Revenue** o compras, considerando los análisis realizados en pasos anteriores como el análisis de distribución de la clase de interés en las categorías de las variables categóricas, el análisis de correlaciones en las variables numéricas, el modelo de referencia de regresión logística creado con las variables seleccionadas, la importancia de características del modelo XGBoost y las técnicas de XAI implementadas en la sección 4.2.8, se obtienen las siguientes recomendaciones:

PageValues: Los gráficos de PDP y la importancia de características en SHAP y XGBoost muestran que PageValues es una de las características más influyentes. Un aumento en PageValues está consistentemente asociado con un aumento en la probabilidad de compra. Se recomienda mejorar el contenido y la funcionalidad de las páginas que conducen a transacciones, enfocándose en aquellas con altos valores de página.

ExitRates: Tanto los gráficos PDP como los valores SHAP indican que ExitRates tiene una influencia negativa en la probabilidad de compra. Reducir las tasas de salida optimizando las páginas clave para retener a los usuarios más tiempo puede aumentar las conversiones. Esto podría incluir mejoras en el diseño de la página, contenido atractivo y llamadas a la acción efectivas.

Month: Las visualizaciones indican que ciertos meses, como noviembre y diciembre, tienen una mayor probabilidad de compras. La gráfica de observaciones por Month muestra que noviembre y octubre son meses con altos porcentajes de Revenue. Se recomienda intensificar las campañas de marketing durante estos meses para capitalizar esta tendencia. Enfocar los recursos de marketing y promociones en los meses de alta conversión puede mejorar significativamente las tasas de éxito de Revenue.

ProductRelated: La importancia de ProductRelated en los modelos indica que un mayor número de páginas relacionadas con productos visitadas y tiempo dedicado a estas páginas está asociado con mayores probabilidades de compra. Mejorar la experiencia de navegación y la presentación de productos podría aumentar las conversiones.

TrafficType: los valores SHAP muestran que el tipo de tráfico tiene un impacto significativo. Identificar y priorizar las fuentes de tráfico que generan mayores conversio-

nes puede ser una estrategia eficaz. Esto puede incluir enfocarse en campañas de SEO (Search Engine Optimization), SEM (Search Engine Marketing) y marketing de afiliación. Los tipos de tráfico más importantes según el análisis exploratorio realizado en la sección 4.2.3 son aquellos correspondientes al grupo 1: Corresponde a las categorías 10, 11, 2 y 5, con un valor de encoding de 0.219550, y el grupo 3 conformado por los tipos de tráfico 13 y 19 con un valor de encoding de 0.270341. Esta recomendación se basa en grupos con valores de encoding altos y una representación significativa en las muestras (por lo que tipos de tráfico como el 7 son excluidos por la baja muestra en el conjunto de datos).

SpecialDay: Aunque SpecialDay no aparece como una de las características más influyentes, es notable que los días especiales aumentan la probabilidad de compra. Las estrategias de promoción deben alinearse con estos días especiales para maximizar las ventas.

Capítulo 5

Conclusiones y Líneas Futuras de Investigación

5.1. Conclusiones Principales

En este trabajo, se ha seguido una metodología basada en KDD (Knowledge Discovery in Databases) para realizar un proceso de minería de datos, desarrollar un modelo de predicción y aplicar técnicas de Interpretabilidad en Inteligencia Artificial (XAI). El objetivo fue desentrañar los factores determinantes que influyen en las predicciones de interacciones en línea en el ámbito del marketing digital. Este enfoque ha permitido cumplir con los objetivos específicos planteados, como la construcción y evaluación de diversos modelos de caja negra, la aplicación de técnicas de XAI a nivel global y local para identificar las variables predictoras más relevantes, y la explicación de casos específicos de éxito y fallo.

Se han construido y evaluado varios modelos de caja negra, incluyendo Redes Neuronales, modelos SVM, Random Forest, GBM, XGBoost y CatBoost, utilizando datos del repositorio de la UC Irvine (UCI) [7], que contiene interacciones en línea de usuarios recopiladas durante un año. Tras entrenar, ajustar y comparar los modelos, el modelo XGBoost con un punto de corte en 0.4 fue seleccionado como el más eficaz, logrando un rendimiento destacado en el conjunto de datos de prueba con un ROC AUC de 0.9268, una precisión de 0.9007, un recall de 0.7078 y un F1-Score de 0.6911.

Mediante técnicas de XAI como SHAP y PDP, se identificaron las variables más influyentes en la intención de compra en línea, destacándose PageValues, Month y ExitRates. PageValues, que representa el valor promedio de una página web visitada antes de completar una transacción, es la variable más influyente. Un aumento en PageValues está consistentemente asociado con un aumento en la probabilidad de compra. Los gráficos de SHAP también muestran que Month y ExitRates tienen influencias significativas. Ciertos meses, como noviembre y diciembre, muestran una mayor probabilidad de compras, y las tasas de salida (ExitRates), que indican el porcentaje de visitas que terminan después de ver una página específica, están asociadas negativamente con las conversiones. Reducir las tasas de salida optimizando las páginas clave para retener a los usuarios más

tiempo puede aumentar las conversiones.

Para explicar predicciones individuales, se utilizaron técnicas como LIME y SHAP a nivel local. Estas herramientas ayudaron a desentrañar las razones detrás de las compras y las no compras en sesiones específicas. Por ejemplo, los gráficos SHAP para casos específicos muestran cómo variables como TrafficType, ProductRelated y ExitRates influyen en la probabilidad de compra. Las sesiones exitosas tienden a tener valores altos de PageValues y bajos de ExitRates. En el caso de un falso positivo, LIME proporcionó explicaciones adicionales, identificando que combinaciones de valores altos en PageValues y bajos en ExitRates son determinantes clave para la probabilidad de compra.

Con base en los patrones identificados a partir del modelo propuesto y la aplicación de técnicas XAI, se realizaron recomendaciones prácticas para la toma de decisiones estratégicas, optimización de procesos y mejora de productos y servicios. Entre las recomendaciones destacan mejorar el contenido y funcionalidad de las páginas que conducen a transacciones, enfocándose en aquellas con altos valores de página; reducir las tasas de salida optimizando las páginas clave para retener a los usuarios más tiempo; intensificar las campañas de marketing en meses de alta conversión como noviembre y diciembre; mejorar la experiencia de navegación y presentación de productos para aumentar las conversiones; priorizar las fuentes de tráfico que generan mayores conversiones, como SEO, SEM y marketing de afiliación; y alinear estrategias de promoción con días especiales para maximizar las ventas.

Este estudio subraya la importancia de las técnicas de XAI para interpretar modelos de caja negra, ofreciendo insights valiosos y accionables que pueden optimizar las estrategias de marketing digital y mejorar las tasas de conversión en el comercio electrónico.

5.2. Limitaciones del Trabajo Realizado

A continuación, se discuten las limitaciones del estudio. Esta subsección permite al lector entender las posibles debilidades del estudio, como restricciones metodológicas, problemas de muestreo, limitaciones en los datos y otras circunstancias que podrían haber afectado los resultados.

El trabajo se centró en la aplicación de técnicas de Interpretabilidad en Inteligencia Artificial (XAI), sin profundizar en técnicas adicionales como SMOTE para el balanceo de clases. Aunque se mencionan estudios preliminares de NearMiss y random oversampling en el anexo, estos no se exploraron en detalle en el cuerpo principal del trabajo.

A pesar de considerar explorar diversas bases de datos como la cancelación hotelera y las campañas de telemarketing bancario, se decidió enfocarse únicamente en el conjunto de datos de UCI [7] para profundizar en un solo caso de estudio. Esto podría haber limitado la generalización de los hallazgos a otros contextos. Además, los datos utilizados abarcan solo un año, lo que puede no capturar completamente las tendencias a largo

plazo. El uso de datos adicionales y agregaciones temporales podría haber revelado patrones más complejos y significativos. En la fuente de datos utilizada, no se especifican en detalle las categorías de tipos de tráfico, sistema operativo o navegador, lo cual limita la comprensión completa del comportamiento del usuario y la contextualización de las predicciones realizadas.

No se realizó un análisis exhaustivo de outliers en las variables numéricas. El desempeño de los modelos podría haberse mejorado si se hubieran aplicado técnicas de manejo de outliers más sofisticadas. Además, la duración limitada del máster, de un año, restringió la posibilidad de realizar tareas más detalladas de adquisición y limpieza de datos. Un enfoque manual similar al del conjunto de datos de UCI con el metodo descrito en [12],podría haber requerido de un largo tiempo de captura de información.

Aunque las fuentes utilizadas, como [3] y [4], proporcionan una base sólida en técnicas XAI, esta área aún está en desarrollo y cuenta con menos investigación comparada con otras áreas de ciencia de datos.

5.3. Líneas Futuras de Investigación

Este estudio ha demostrado la eficacia de las técnicas de XAI para interpretar modelos predictivos en el marketing digital. Sin embargo, hay varias áreas prometedoras para futuros trabajos y técnicas adicionales de XAI que podrían enriquecer el análisis y la interpretación de los modelos.

Para mejorar las métricas del modelo propuesto, se pueden utilizar herramientas como AutoML y técnicas de ensamblaje como stacking y blending. También es recomendable evaluar el uso de técnicas como SMOTE para equilibrar mejor las clases y enriquecer el conjunto de datos de entrenamiento.

Desde el punto de vista de XAI, se podrían aplicar técnicas adicionales que no fueron exploradas en este estudio, pero que son explicadas en [4]:

- **Contrafactual Explanations**: Proporcionan ejemplos de cambios específicos en las entradas necesarios para obtener un resultado diferente, ayudando a los usuarios a entender cómo modificar su comportamiento para lograr el resultado deseado.
- **Scoped Rules (Anchors)**: Utilizan reglas de alta precisión que capturan las condiciones suficientes para una predicción específica, facilitando la interpretación por parte de usuarios no técnicos.

Además, SHAP puede enriquecer el análisis de datos no estructurados, como se menciona en [3]. Por ejemplo, complementar este estudio con un análisis de sentimiento en redes sociales podría ser útil, y el uso de SHAP brindaría formas de interpretar modelos de clasificación de sentimientos, identificando palabras clave que influyen en las predicciones.

Las técnicas de XAI también pueden aplicarse a problemas de aprendizaje no supervisados, ayudando a entender la formación de clusters, la representación en espacios de menor dimensión y la detección de anomalías. Estas aplicaciones pueden ser útiles en otros aspectos del análisis de marketing digital y de intención de compra en canales digitales.

En conclusión, los futuros trabajos en la aplicación de técnicas de XAI en marketing digital podrían enfocarse en la incorporación de datos adicionales, análisis temporal y estacional, personalización y segmentación dinámica, optimización de modelos y la implementación de técnicas de XAI no exploradas en este estudio. Estas acciones mejorarán la precisión y utilidad de los modelos, facilitando la interpretación y aplicación de los resultados en contextos reales, y proporcionando un marco más robusto y flexible para la toma de decisiones basadas en datos.

Bibliografía

[1] G. M. Díaz, J. J. Galán, and R. A. Carrasco, "Xai for churn prediction in b2b models: a use case in an enterprise software company," *Mathematics*, vol. 10, no. 20, p. 3896, 2022.

[2] A. Adadi and M. Berrada, "Peeking inside the black-box: a survey on explainable artificial intelligence (xai)," *IEEE access*, vol. 6, pp. 52 138–52 160, 2018.

[3] C. Molnar, *Interpreting Machine Learning Models With SHAP*. Munich, Germany: Self-published, 2023.

[4] ——, *Interpretable Machine Learning*, 2nd ed., 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book

[5] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[6] U. Shafique and H. Qaiser, "A comparative study of data mining process models (kdd, crisp-dm and semma)," *International Journal of Innovation and Scientific Research*, vol. 12, no. 1, pp. 217–222, 2014.

[7] C. Sakar and Y. Kastro, "Online Shoppers Purchasing Intention Dataset," UCI Machine Learning Repository, 2018, DOI: https://doi.org/10.24432/C5F88Q.

[8] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[9] K. Banachewicz, L. Massaron, and A. Goldbloom, *The Kaggle Book: Data analysis and machine learning for competitive data science*. Packt Publishing Ltd, 2022.

[10] M. Chlebus and Z. Osika, *Comparison of tree-based models performance in prediction of marketing campaign results using Explainable Artificial Intelligence tools*. University of Warsaw, Faculty of Economic Sciences, 2020.

[11] eMarketer, "Retail e-commerce sales worldwide from 2014 to 2027 (in billion u.s. dollars) [graph]," Statista, 2023, retrieved June 20, 2024. [Online]. Available: https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/

[12] C. O. Sakar, S. O. Polat, M. Katircioglu, and Y. Kastro, "Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and lstm recurrent neural networks," *Neural Computing and Applications*, vol. 31, no. 10, pp. 6893–6908, 2019.

[13] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[14] S. Zhai, S. Gao, L. Wang, and P. Liu, "When both human and machine drivers make mistakes: Whom to blame?" *Transportation research part A: policy and practice*, vol. 170, p. 103637, 2023.

[15] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1721–1730.

[16] J. Lightbourne, "Damned lies & criminal sentencing using evidence-based tools," *Duke L. & Tech. Rev.*, vol. 15, p. 327, 2016.

[17] M. A. Chen, Q. Wu, and B. Yang, "How valuable is fintech innovation?" *The Review of Financial Studies*, vol. 32, no. 5, pp. 2062–2106, 2019.

[18] D. Gunning, "Broad agency announcement explainable artificial intelligence (xai)," *Defense Advanced Research Projects Agency (DARPA), Tech. Rep.*, 2016.

[19] J. J. Ohana, S. Ohana, E. Benhamou, D. Saltiel, and B. Guez, "Explainable ai (xai) models applied to the multi-agent environment of financial markets," in *Explainable and Transparent AI and Multi-Agent Systems: Third International Workshop, EXTRAAMAS 2021, Virtual Event, May 3–7, 2021, Revised Selected Papers 3*. Springer, 2021, pp. 189–207.

[20] H. Bandi, S. Joshi, S. Bhagat, and D. Ambawade, "Integrated technical and sentiment analysis tool for market index movement prediction, comprehensible using xai," in *2021 International Conference on Communication information and Computing Technology (ICCICT)*, 2021, pp. 1–8, iD: 1.

[21] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[22] F. Heider and M. Simmel, "An experimental study of apparent behavior," *The American Journal of Psychology*, vol. 57, no. 2, pp. 243–259, 1944.

[23] M. Robnik-Šikonja and M. Bohanec, "Perturbation-based explanations of prediction models," *Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent*, pp. 159–175, 2018.

[24] L. Longo, M. Brcic, F. Cabitza, J. Choi, R. Confalonieri, J. Del Ser, R. Guidotti, Y. Hayashi, F. Herrera, A. Holzinger *et al.*, "Explainable artificial intelligence (xai) 2.0: A manifesto of open challenges and interdisciplinary research directions," *Information Fusion*, p. 102301, 2024.

[25] O. Loyola-Gonzalez, "Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view," *IEEE access*, vol. 7, pp. 154 096–154 113, 2019.

[26] L. S. Shapley *et al.*, “A value for n-person games,” 1953.

[27] E. Štrumbelj and I. Kononenko, “Explaining prediction models and individual predictions with feature contributions,” *Knowledge and information systems*, vol. 41, pp. 647–665, 2014.

[28] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” *Advances in neural information processing systems*, vol. 30, 2017.

[29] J. H. Friedman, “Greedy function approximation: a gradient boosting machine,” *Annals of statistics*, pp. 1189–1232, 2001.

[30] B. M. Greenwell, B. C. Boehmke, and A. J. McCarthy, “A simple and effective model-based variable importance measure,” *arXiv preprint arXiv:1805.04755*, 2018.

[31] M. Kuhn, K. Johnson *et al.*, *Applied predictive modeling*. Springer, 2013, vol. 26.

Anexo A

Resultados adicionales

A.1. Analisis Exploratorio

A.1.1. Variables Categoricas

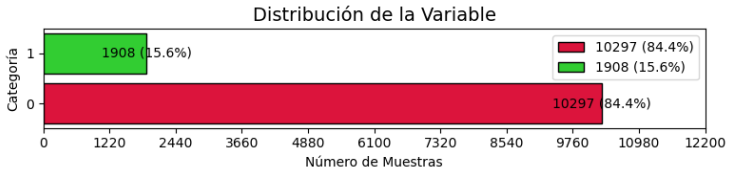
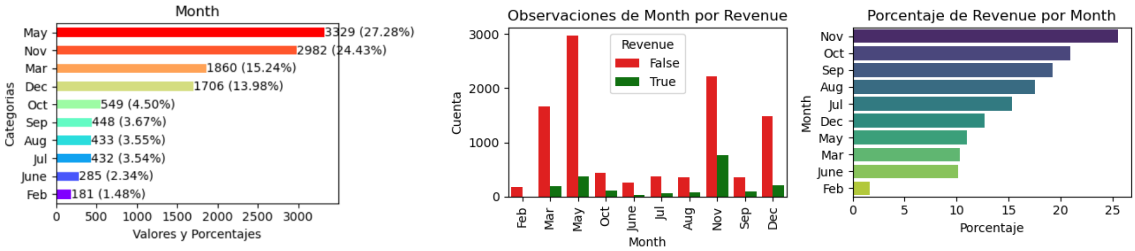


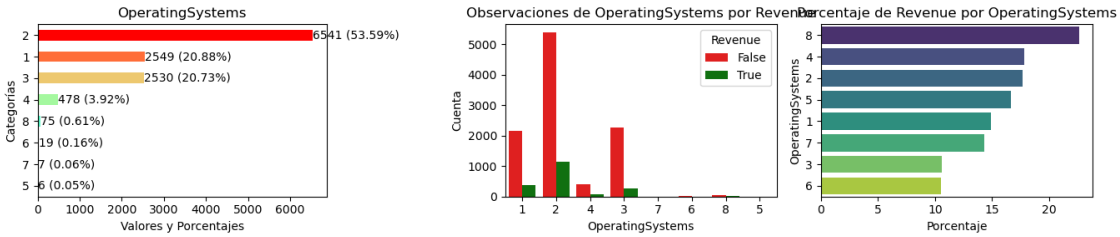
Figura A.1: Distribución inicial de la clase de interés en el conjunto de datos.



(a) Distribución de las categorías (b) Distribución de observaciones de la clase de interés

Figura A.2: Análisis de variable categórica **Month**

Month:
Nov | Total: 2982 | Revenue = 1: 760 (25.49 %)
Oct | Total: 549 | Revenue = 1: 114 (20.95 %)
Sep | Total: 448 | Revenue = 1: 86 (19.20 %)
Aug | Total: 433 | Revenue = 1: 76 (17.55 %)
Jul | Total: 432 | Revenue = 1: 66 (15.28 %)
Dec | Total: 1706 | Revenue = 1: 215 (12.66 %)
May | Total: 3329 | Revenue = 1: 365 (10.96 %)
Mar | Total: 1860 | Revenue = 1: 192 (10.32 %)
June | Total: 285 | Revenue = 1: 29 (10.18 %)
Feb | Total: 181 | Revenue = 1: 3 (1.66 %)



(a) Distribución de las categorías (b) Distribución de observaciones de la clase de interés

Figura A.3: Análisis de variable categórica **OperatingSystems**

OperatingSystems:

8 | Total: 75 | Revenue = 1: 16 (22.67 %)

4 | Total: 478 | Revenue = 1: 85 (17.78 %)

2 | Total: 6541 | Revenue = 1: 1155 (17.66 %)

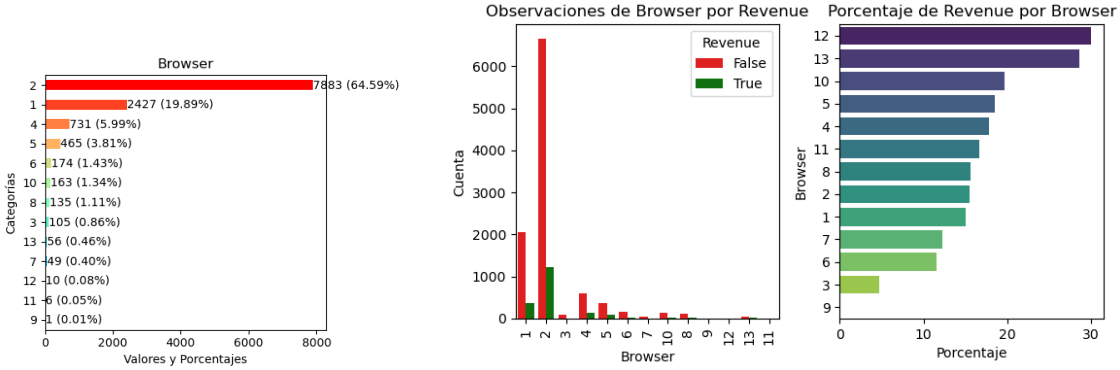
5 | Total: 6 | Revenue = 1: 0 (16.67 %)

1 | Total: 2549 | Revenue = 1: 379 (14.87 %)

7 | Total: 7 | Revenue = 1: 1 (14.29 %)

3 | Total: 2530 | Revenue = 1: 267 (10.59 %)

6 | Total: 19 | Revenue = 1: 1 (10.53 %)



(a) Distribución de las categorías (b) Distribución de observaciones de la clase de interés

Figura A.4: Análisis de variable categórica **Browser**

Browser:

12 | Total: 10 | Revenue = 1: 3 (30.0 %)

13 | Total: 56 | Revenue = 1: 16 (28.57 %)

10 | Total: 163 | Revenue = 1: 32 (19.63 %)

5 | Total: 465 | Revenue = 1: 85 (18.49 %)

4 | Total: 731 | Revenue = 1: 130 (17.78 %)

11 | Total: 6 | Revenue = 1: 0 (16.67 %)

8 | Total: 135 | Revenue = 1: 21 (15.56 %)

2 | Total: 7883 | Revenue = 1: 1222 (15.51 %)

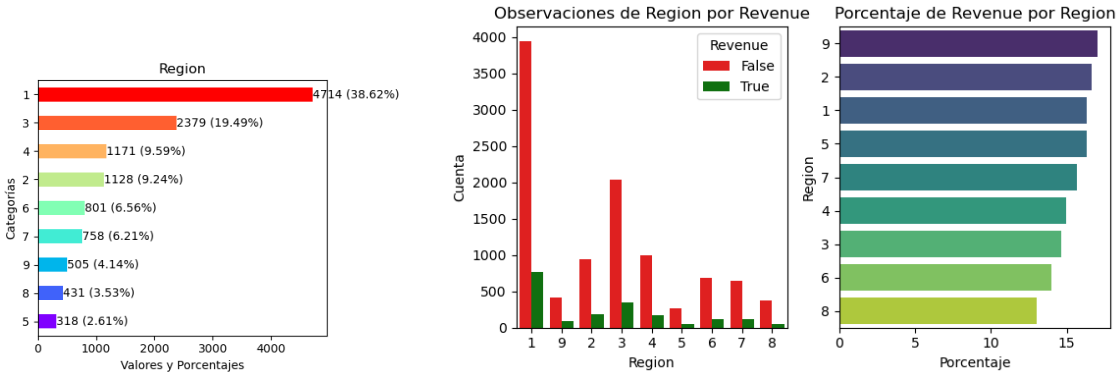
1 | Total: 2427 | Revenue = 1: 365 (15.04 %)

7 | Total: 49 | Revenue = 1: 6 (12.24 %)

6 | Total: 174 | Revenue = 1: 19 (11.49 %)

3 | Total: 105 | Revenue = 1: 5 (4.76 %)

9 | Total: 1 | Revenue = 1: 0 (0.0 %)



(a) Distribución de las categorías (b) Distribución de observaciones de la clase de interés

Figura A.5: Análisis de variable categórica **Region**

Region:

9 | Total: 505 | Revenue = 1: 86 (17.03 %)

2 | Total: 1128 | Revenue = 1: 187 (16.67 %)

1 | Total: 4714 | Revenue = 1: 771 (16.36 %)

5 | Total: 318 | Revenue = 1: 51 (16.35 %)

7 | Total: 758 | Revenue = 1: 118 (15.7 %)

4 | Total: 1171 | Revenue = 1: 175 (14.94 %)

3 | Total: 2379 | Revenue = 1: 348 (14.67 %)

6 | Total: 801 | Revenue = 1: 112 (13.98 %)

8 | Total: 431 | Revenue = 1: 56 (12.99 %)

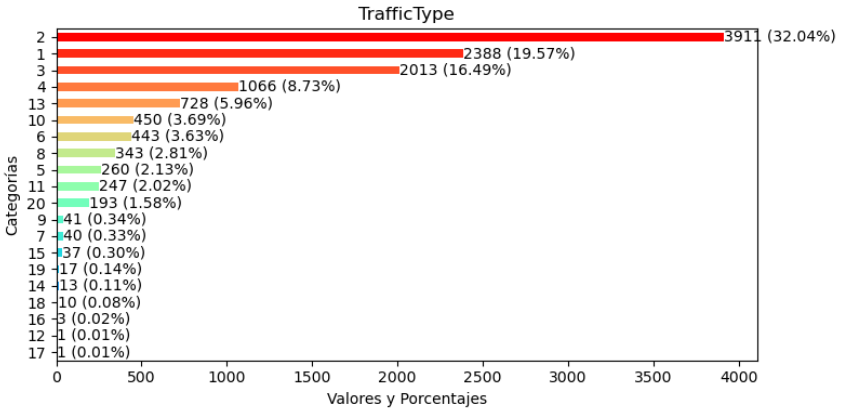


Figura A.6: Distribución de las categorías de la variable **TrafficType**

TrafficType:

16 | Total: 3 | Revenue = 1: 0 (33.33 %)

7 | Total: 40 | Revenue = 1: 12 (30.0 %)

8 | Total: 343 | Revenue = 1: 95 (27.7 %)

20 | Total: 193 | Revenue = 1: 49 (25.91 %)

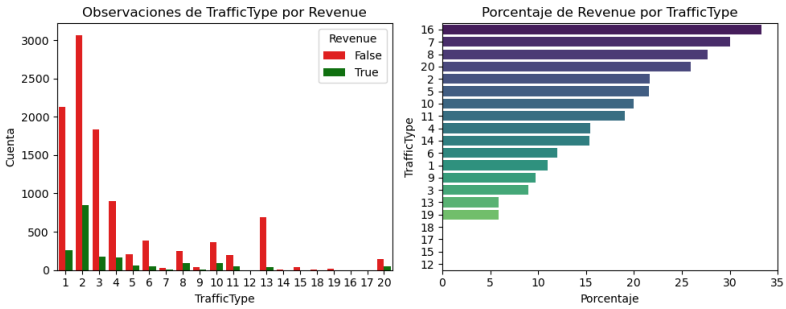


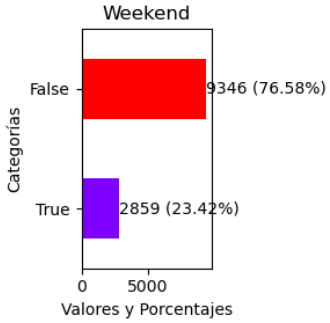
Figura A.7: Distribución de observaciones de la clase de interés en la variable **TrafficType**

TrafficType (continuación):
2 | Total: 3911 | Revenue = 1: 847 (21.66 %)
5 | Total: 260 | Revenue = 1: 56 (21.54 %)
10 | Total: 450 | Revenue = 1: 90 (20.0 %)
11 | Total: 247 | Revenue = 1: 47 (19.03 %)
4 | Total: 1066 | Revenue = 1: 165 (15.48 %)
14 | Total: 13 | Revenue = 1: 2 (15.38 %)
6 | Total: 443 | Revenue = 1: 53 (11.96 %)
1 | Total: 2388 | Revenue = 1: 262 (10.97 %)
9 | Total: 41 | Revenue = 1: 4 (9.76 %)
3 | Total: 2013 | Revenue = 1: 180 (8.94 %)
13 | Total: 728 | Revenue = 1: 43 (5.91 %)
19 | Total: 17 | Revenue = 1: 0 (5.88 %)
18 | Total: 10 | Revenue = 1: 0 (0.0 %)
17 | Total: 1 | Revenue = 1: 0 (0.0 %)
15 | Total: 37 | Revenue = 1: 0 (0.0 %)
12 | Total: 1 | Revenue = 1: 0 (0.0 %)

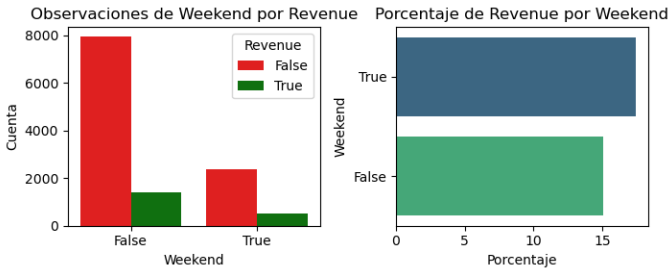
(a) Distribución de las categorías (b) Distribución de observaciones de la clase de interés

Figura A.8: Análisis de variable categórica **VisitorType**

VisitorType:
New_Visitor | Total: 1693 | Revenue = 1: 422 (24.93 %)
Other | Total: 81 | Revenue = 1: 16 (19.75 %)
Returning_Visitor | Total: 10431 | Revenue = 1: 1470 (14.09 %)



(a) Distribución de las categorías



(b) Distribución de observaciones de la clase de interés

Figura A.9: Análisis de variable categórica **Weekend**

Weekend:

True | Total: 2859 | Revenue = 1: 499 (17.45 %)

False | Total: 9346 | Revenue = 1: 1409 (15.08 %)

Agrupación de variables

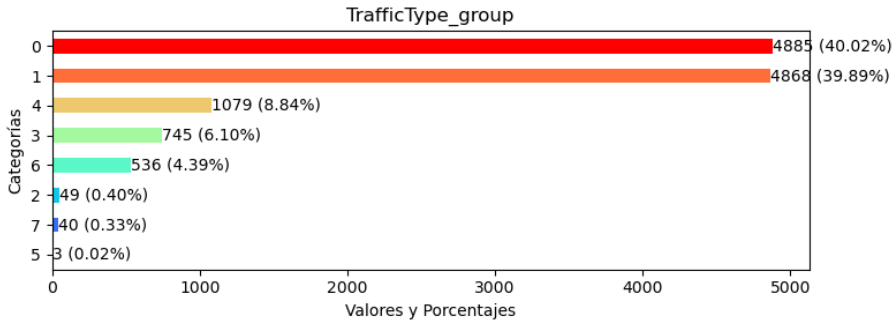


Figura A.10: Distribución de los grupos formados de la variable **TrafficType**

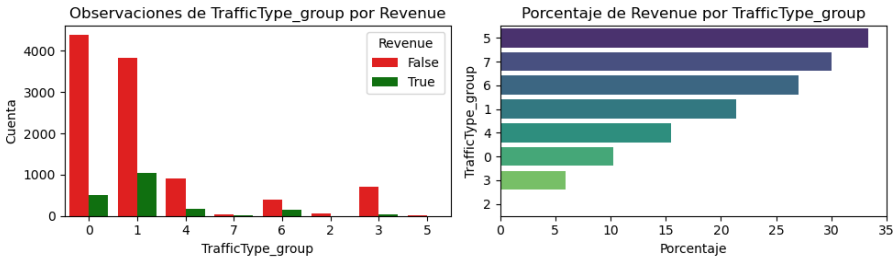


Figura A.11: Distribución de observaciones de la clase de interés en los grupos formados de la variable **TrafficType**

TrafficType_group:

5 | Total: 3 | Revenue = 1: 0 (33.33 %)

7 | Total: 40 | Revenue = 1: 12 (30.0 %)

6 | Total: 536 | Revenue = 1: 145 (27.05 %)
1 | Total: 4868 | Revenue = 1: 1040 (21.36 %)
4 | Total: 1079 | Revenue = 1: 167 (15.48 %)
0 | Total: 4885 | Revenue = 1: 499 (10.21 %)
3 | Total: 745 | Revenue = 1: 44 (5.91 %)
2 | Total: 49 | Revenue = 1: 0 (0.0 %)

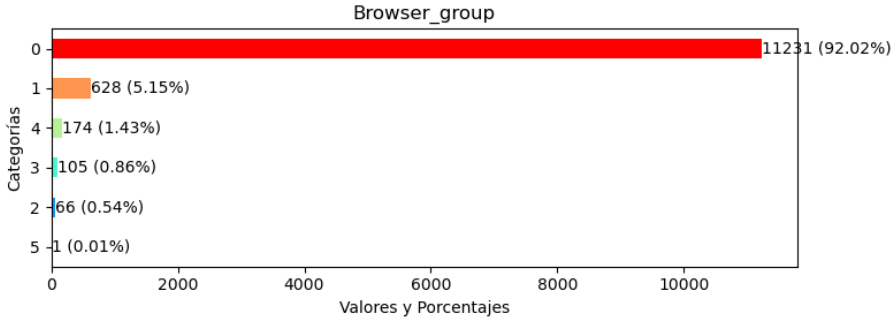


Figura A.12: Distribución de los grupos formados de la variable **Browser**

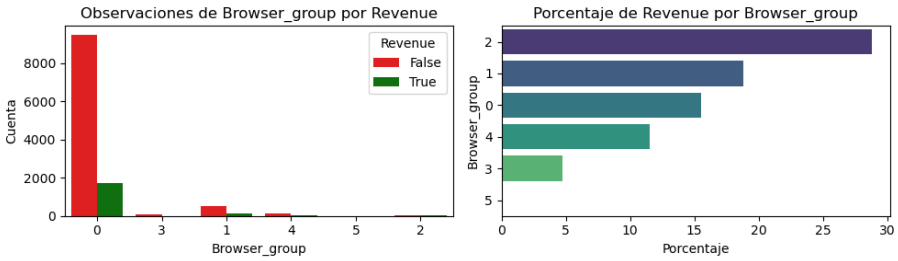


Figura A.13: Distribución de observaciones de la clase de interés en los grupos formados de la variable **Browser**

Browser_group:
2 | Total: 66 | Revenue = 1: 19 (28.79 %)
1 | Total: 628 | Revenue = 1: 118 (18.79 %)
0 | Total: 11231 | Revenue = 1: 1746 (15.55 %)
4 | Total: 174 | Revenue = 1: 19 (11.49 %)
3 | Total: 105 | Revenue = 1: 5 (4.76 %)
5 | Total: 1 | Revenue = 1: 0 (0.0 %)

OperatingSystems_group:
3 | Total: 75 | Revenue = 1: 16 (22.67 %)
1 | Total: 7025 | Revenue = 1: 1240 (17.67 %)
0 | Total: 2556 | Revenue = 1: 380 (14.87 %)
2 | Total: 2549 | Revenue = 1: 270 (10.59 %)

A continuación se presenta una tabla con los grupos creados:

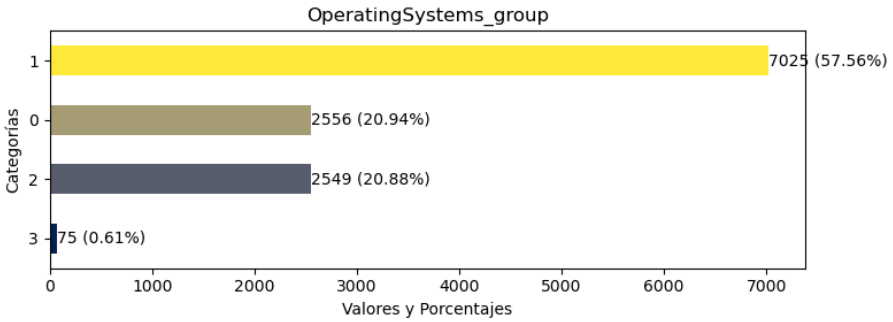


Figura A.14: Distribución de los grupos formados de la variable **OperatingSystems**

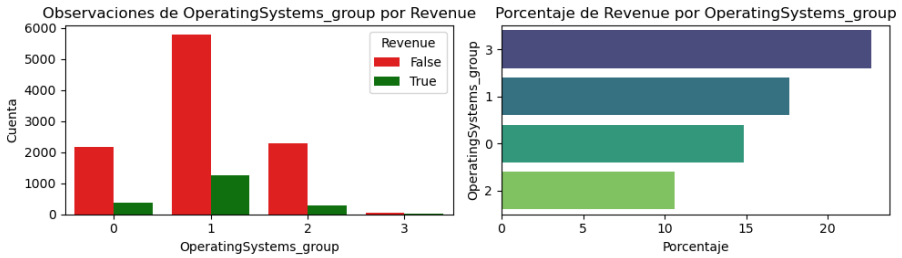


Figura A.15: Distribución de observaciones de la clase de interés en los grupos formados de la variable **OperatingSystems**

Tabla A.1: Lista de categorías por variable

Variable	Grupo	Lista de categorías
Browser	0	1, 11, 2, 4, 7, 8
	1	10, 5
	2	12, 13
	3	3
	4	6
	5	9
OperatingSystems	0	1, 7
	1	2, 4, 5
	2	3, 6
	3	8
TrafficType	0	1, 3, 6, 9
	1	10, 11, 2, 5
	2	12, 15, 17, 18
	3	13, 19
	4	14, 4
	5	16
	6	20, 8
	7	7

A.1.2. Variables Numericas

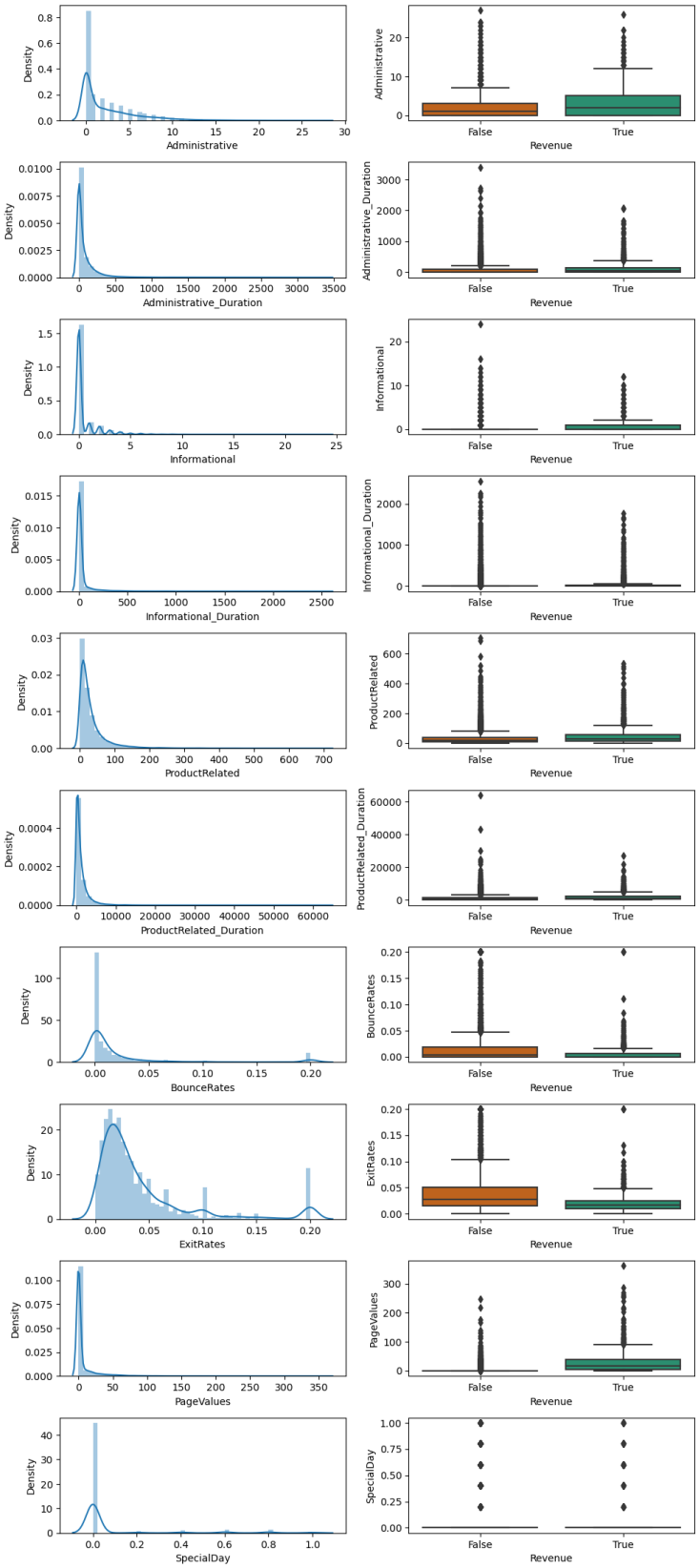


Figura A.16: Exploración gráfica de variables numéricas.

A.2. Selección de Variables

La Tabla A.2 presenta los valores obtenidos en el modelo de regresion logistica usando el procedimiento `logistic`://

Tabla A.2: Resultados de Análisis Stepwise

Step	Effect: Entered	DF	Number In	Score: Chi-Square	Pr >ChiSq
1	PageValues	1	1	2952.1827	$<$ .0001
2	ProductRelated	1	2	271.2943	$<$ .0001
3	TrafficType	1	3	146.3953	$<$ .0001
4	ExitRates	1	4	98.8235	$<$ .0001
5	Month	1	5	69.889	$<$ .0001
6	OperatingSystems	1	6	4.9	0.0269
7	ProductRelated_Durat	1	7	4.0537	0.0441
8	Weekend	1	8	4.0203	0.045

Tabla A.3: Resultados de Nodo Variable Selection de SAS-EM

Variable	R-Square	p-Value	% Variance Explained
PageValues	0.247711	$<$0.0001	9.96 %
ProductRelated	0.015666	$<$0.0001	9.75 %
TrafficType	0.011299	$<$0.0001	9.60 %
ExitRates	0.003795	$<$0.0001	9.56 %
Month	0.003151	$<$0.0001	9.52 %
BounceRates	0.000745	0.0017	9.51 %
VisitorType_NewVisitor	0.000579	0.0056	9.50 %
ProductRelated_Duration	0.000559	0.0065	9.49 %

Tabla A.4: Resultados de Nodo Gradient Boosting de SAS-EM

Variable	Number of Spliting Rules	Importance
PageValues	30	1.000000
ExitRates	23	0.219090
Month	20	0.174863
TrafficType	9	0.115901
ProductRelated	7	0.099171
BounceRates	6	0.087720
Administrative_Duration	7	0.073926
ProductRelated_Duration	6	0.073490
Administrative	2	0.040530
Region_3	1	0.020608
Region_1	1	0.018743

Tabla A.5: Resultados de Nodo Partial Least Squares de SAS-EM

Variable	Standardized Parameter Estimate	Variable Importance for Projection	Role
PageValues	0.459142484	2.640042897	Input
TrafficType	0.067721626	0.817383593	Input
BounceRates	0.06507296	0.894887994	Input
ProductRelated	0.051366508	0.83025114	Input
ProductRelated_Duration	0.044310163	0.812964697	Input
Administrative	0.009530756	0.838832653	Input
ExitRates	-0.120549036	1.070547269	Input

Comparación de ROC AUC de regresiones logísticas usando todos los conjuntos de variables disponibles.

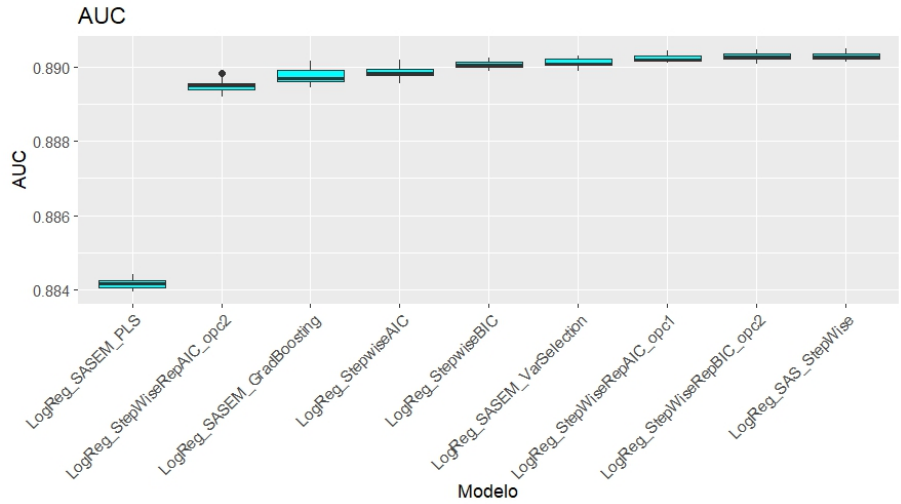


Figura A.17: Comparación de ROC AUC de regresiones logísticas usando todos los conjuntos de variables disponibles.

Resultados y reseumen de la evaluacion de conjunto de seleccion de varialbes obenidos usando modelos de regrseion logistica.

Tabla A.6: Comparación de Modelos de Regresión Logística Usando Conjuntos de Variables Obtenidos

Método	**Lista de Variables**	**ROC AUC**	**Tasa Fallos**
SAS: PROC regression con selección tipo Stepwise **Label:** LogReg_SAS_StepWise	**Número de Var:** 6 ExitRates, Month, PageValues, ProductRelated, SpecialDay, TrafficType	0.8903	0.1186

Tabla A.6: (Continuación)

Método	Lista de Variables	ROC AUC	Tasa Fallos
R: Stepwise repetido bajo criterio BIC opción 2 **Label:** LogReg_-StepWiseRepBIC_opc2	**Número de Var:** 5 ExitRates, Month, PageValues, ProductRelated_Duration, TrafficType	0.8903	0.1182
R: Stepwise repetido bajo criterio AIC opción 1 **Label:** LogReg_-StepWiseRepAIC_opc1	**Número de Var:** 7 Browser, ExitRates, Month, PageValues, ProductRelated, SpecialDay, TrafficType	0.8902	0.1185
SAS-EM: Nodo Variable Selection **Label:** LogReg_-SASEM_VarSelection	**Número de Var:** 8 BounceRates, ExitRates, Month, PageValues, ProductRelated, ProductRelated_Duration, TrafficType, VisitorType_NewVisitor	0.8901	0.1186
R: Stepwise bajo criterio BIC **Label:** LogReg_StepwiseBIC	**Número de Var:** 5 ExitRates, Month, PageValues, ProductRelated, TrafficType	0.8901	0.1185
R: Stepwise bajo criterio AIC **Label:** LogReg_StepwiseAIC	**Número de Var:** 11 Browser, ExitRates, Month, OperatingSystems, PageValues, ProductRelated, ProductRelated_Duration, Region_9, SpecialDay, TrafficType, VisitorType_NewVisitor	0.8899	0.1184
SAS-EM: Nodo Gradient Boosting **Label:** LogReg_-SASEM_GradBoosting	**Número de Var:** 11 Administrative, Administrative_Duration, BounceRates, ExitRates, Month, PageValues, ProductRelated, ProductRelated_Duration, Region_1, Region_3, TrafficType	0.8898	0.1189
R: Stepwise repetido bajo criterio AIC opción 2 **Label:** LogReg_-StepWiseRepAIC_opc2	**Número de Var:** 10 Browser, ExitRates, Month, OperatingSystems, PageValues, ProductRelated, Region_3, Region_9, SpecialDay, TrafficType	0.8895	0.1182
SAS-EM: Nodo Partial Least Squares **Label:** LogReg_SASEM_PLS	**Número de Var:** 7 Administrative, BounceRates, ExitRates, PageValues, ProductRelated, ProductRelated_Duration, TrafficType	0.8842	0.1186

Anexo B

Códigos y Funciones creadas

B.1. SAS - Selección de Variables Usando Método Stepwise en proceso logistic

```
LIBNAME Datos BASE "C:\Users\cmart\OneDrive\Documentos\2023
   _Complutense\03_TFM\zz_Code\Datos";

proc import
    DATAFILE="C:\Users\cmart\OneDrive\Documentos\2023
   _Complutense\03_TFM\zz_Code\Datos\onlineShoping_train.csv"
    out = Datos.onlineShoping dbms=csv replace;
run;

/* SELECCION DE VARIABLES */
proc logistic data=Datos.onlineShoping;
    /* Especificar la variable dependiente y las variables
   independientes */
    model Revenue = Administrative Administrative_Duration
   Informational Informational_Duration
                    ProductRelated ProductRelated_Duration
   BounceRates ExitRates PageValues
                    SpecialDay Month OperatingSystems Browser
   TrafficType VisitorType_NewVisitor
                    Weekend Region_1 Region_2 Region_3 Region_4
   Region_5 Region_6 Region_7 Region_8 Region_9
                    / selection=stepwise;
run;
```

Listing B.1: Código SAS: Selección de Variables Usando Método Stepwise en proceso logistic

B.2. R - Proceso de Selección de Variables

```
setwd("C:/Users/cmart/OneDrive/Documentos/2023_Complutense/03
    _TFM/zz_Code/Datos")

library(nnet)
library(dummies)
library(MASS)
library(reshape)
library(tibble)
library(plyr) # Operaciones de manipulación y resumen de
    DataFrames
library(caret) # Paquete para comparación y tuneado de modelos
    predictivos
library(pROC) # para calculo de AUC
library(ggplot2)
library(Boruta)
library(MXM)
library(SemiPar) # Calcular la selección de variables usando SES

# Paquetes para usar parallel computing y permitir mayor
# velocidad de procesamiento

library(parallel)
library(doParallel)

GS_T0 <- Sys.time()
cluster <- makeCluster(detectCores() - 1) # number of cores,
    convention to leave 1 core for OS
registerDoParallel(cluster) # register the parallel processing

# Cargar datos
onlineShoping <- read.csv("onlineShoping_train.csv")

# Definir lista con tipo de variables
vardep <- "Revenue" #variable objetivo

# Preparar variables X e Y
nombres <- names(onlineShoping)[names(onlineShoping) != vardep]
X <- onlineShoping[, nombres] # Variables independientes
Y <- onlineShoping[, vardep] # Variable objetivo

# FILTROS

# AIC - OK
```

```
full_model_aic <- glm(Revenue ~ ., data = onlineShoping[, c("
    Revenue", nombres)],
                      family = binomial(link = "logit"))
null_model_aic <- glm(Revenue ~ 1, data = onlineShoping[, c("
    Revenue", nombres)],
                      family = binomial(link = "logit"))

selec1_aic <- stepAIC(null_model_aic, scope = list(upper =
    full_model_aic),
                      direction = "both", family = binomial(link
    ="logit"),
                      trace=FALSE)
selected_vars_aic <- names(selec1_aic$coefficients)[-1]
selected_vars_aic



# BIC - OK
k <- log(nrow(onlineShoping)) # 9.3982297330736 -- 9

selec1_bic <- stepAIC(null_model_aic, scope = list(upper =
    full_model_aic),
                      direction = "both",family = binomial(link=
    "logit"),
                      trace = FALSE, k = 9)

selec1_bic
selected_vars_bic <- names(selec1_bic$coefficients)[-1]
selected_vars_bic


source("funcion steprepetido binaria.R")

# usar conjunto de variables obtenido en AIC en stepAIC para no
    tardar tanto

lista_steprepetido_aic<-steprepetidobinaria(data=onlineShoping,
                                            vardep=c("Revenue"),
                                            listconti= nombres,
                                            sinicio=12345,sfinal
    =12385,
                                            porcen=0.8,
                                            criterio="AIC")

tabla_aic<-lista_steprepetido_aic[[1]]
```

```
step_aic_opc1 <- dput(lista_steprepetido_aic[[2]][[1]])
step_aic_opc2 <- dput(lista_steprepetido_aic[[2]][[2]])
step_aic_opc1
step_aic_opc2

# usar conjunto de variables obtenido en BIC en stepAIC para no
   tardar tanto

lista_steprepetido_bic<-steprepetidobinaria(data=onlineShoping,
                                            vardep=c("Revenue"),
                                            listconti= nombres,
                                            sinicio=12345,sfinal
   =12385,
                                            porcen=0.8,
                                            criterio="BIC")

tabla_bic<-lista_steprepetido_bic[[1]]
step_bic_opc1 <- dput(lista_steprepetido_bic[[2]][[1]])
step_bic_opc2 <- dput(lista_steprepetido_bic[[2]][[2]])
step_bic_opc1
step_bic_opc2
```

Listing B.2: Proceso de Selección Variables Mediante Metodos Stepwise con Criterios AIC y BIC

```

# Habiendo concluido tambien el procoso previo de seleccion de
   variables en R, SAS y SAS-EM procedemos a definir los
   vectores con los conjuntos de variables candidatos

# Listas obtenidas de SAS y SAS-EM

SAS_EM_VariableSelection <- c("PageValues", "ProductRelated", "
   TrafficType",
                              "ExitRates", "Month", "BounceRates
   ",
                              "VisitorType_NewVisitor", "
   ProductRelated_Duration")

SAS_EM_GradBoosting <- c("PageValues", "ExitRates", "Month", "
   TrafficType",
                         "ProductRelated", "BounceRates", "
   Administrative_Duration",
                         "ProductRelated_Duration", "
   Administrative",
                         "Region_3", "Region_1")
```

```
SAS_EM_PLS <- c("PageValues", "TrafficType", "BounceRates", "
   ProductRelated",
               "ProductRelated_Duration", "Administrative", "
   ExitRates")

SAS_LogReg_Stepwise <- c("PageValues", "ProductRelated", "
   TrafficType",
                        "ExitRates", "Month", "SpecialDay")

# Combinar todas las listas en una sola (excluyendo
   selected_vars_bic)
all_vars <- c(SAS_EM_VariableSelection, SAS_EM_GradBoosting,
              SAS_EM_PLS, SAS_LogReg_Stepwise,
              step_bic_opc1, step_bic_opc2, step_aic_opc1,
   step_aic_opc2,
              selected_vars_aic)

# Calcular la frecuencia de cada variable
var_freq <- table(all_vars)
var_freq <- as.data.frame(var_freq)
colnames(var_freq) <- c("Variable", "Frecuencia")

# Ordenar por frecuencia descendente
var_freq <- var_freq[order(-var_freq$Frecuencia), ]
var_freq

# Pruebas bajo regresión de validación cruzada repetida y
   boxplot, para observar
# qué set de variables es más prometedor

source ("cruzadas avnnet y log binaria - Copy.R")

# Cambio de variable Objetivo a categorica con clases "Yes" y "
   No"
onlineShoping$Revenue <- ifelse(onlineShoping$Revenue == 1, "Yes
   ", "No")

# Llamar a la función cruzadalogistica
medias1 <- cruzadalogistica(data = onlineShoping,
                            vardep = vardep,
                            listconti = selected_vars_aic,
                            listclass = c(""),
                            grupos = 20,
                            sinicio = 1234,
```

```
                              repe = 20)
medias1$modelo="LogReg_StepwiseAIC"


medias2<-cruzadalogistica(data=onlineShoping,
                          vardep="Revenue",listconti=
   selected_vars_bic,
                          listclass=c(""), grupos=20,sinicio
   =1234,repe=20)

medias2$modelo="LogReg_StepwiseBIC"

medias3<-cruzadalogistica(data=onlineShoping,
                          vardep="Revenue",listconti=
   step_aic_opc1,
                          listclass=c(""), grupos=20,sinicio
   =1234,repe=20)

medias3$modelo="LogReg_StepWiseRepAIC_opc1"

medias4<-cruzadalogistica(data=onlineShoping,
                          vardep="Revenue",listconti=
   step_aic_opc2,
                          listclass=c(""), grupos=20,sinicio
   =1234,repe=20)

medias4$modelo="LogReg_StepWiseRepAIC_opc2"

medias5<-cruzadalogistica(data=onlineShoping,
                          vardep="Revenue",listconti=
   step_bic_opc2,
                          listclass=c(""), grupos=20,sinicio
   =1234,repe=20)

medias5$modelo="LogReg_StepWiseRepBIC_opc2"

medias6<-cruzadalogistica(data=onlineShoping,
                          vardep="Revenue",listconti=
   SAS_LogReg_Stepwise,
                          listclass=c(""), grupos=20,sinicio
   =1234,repe=20)

medias6$modelo="LogReg_SAS_StepWise"

medias7<-cruzadalogistica(data=onlineShoping,
```

```
                          vardep="Revenue",listconti=
   SAS_EM_VariableSelection,
                          listclass=c(""), grupos=20,sinicio
   =1234,repe=20)

medias7$modelo="LogReg_SASEM_VarSelection"

medias8<-cruzadalogistica(data=onlineShoping,
                          vardep="Revenue",listconti=
   SAS_EM_GradBoosting,
                          listclass=c(""), grupos=20,sinicio
   =1234,repe=20)

medias8$modelo="LogReg_SASEM_GradBoosting"

medias9<-cruzadalogistica(data=onlineShoping,
                          vardep="Revenue",listconti=SAS_EM_PLS,
                          listclass=c(""), grupos=20,sinicio
   =1234,repe=20)

medias9$modelo="LogReg_SASEM_PLS"

union1<-rbind(medias1,medias2,medias3,medias4,medias5,
              medias6,medias7,medias8, medias9)

# Establecer el ángulo de rotación del texto del eje x
angle <- 45  # Ángulo de rotación

u1 <- union1
u1$modelo <- with(u1, reorder(modelo,tasa, mean))

# Gráfico de cajas para TASA FALLOS
p1 <- ggplot(u1, aes(x = modelo, y = tasa)) +
  geom_boxplot(fill = "pink") +
  labs(x = "Modelo", y = "Tasa de Fallos") +
  theme(axis.text.x = element_text(angle = angle, hjust = 1)) +
  ggtitle("TASA FALLOS")
p1  # Gráfico de TASA FALLOS


u1 <- union1
u1$modelo <- with(u1, reorder(modelo,auc, mean))

# Gráfico de cajas para AUC
p2 <- ggplot(u1, aes(x = modelo, y = auc)) +
```

```
  geom_boxplot(fill = "cyan") +
  labs(x = "Modelo", y = "AUC") +
  theme(axis.text.x = element_text(angle = angle, hjust = 1)) +
  ggtitle("AUC")
p2  # Gráfico de AUC

union1<-rbind(medias1,medias2,medias3,medias4,medias5,
              medias6,medias7,medias8)

# Establecer el ángulo de rotación del texto del eje x
angle <- 45  # Ángulo de rotación

u1 <- union1
u1$modelo <- with(u1, reorder(modelo,tasa, mean))

# Gráfico de cajas para TASA FALLOS
p1 <- ggplot(u1, aes(x = modelo, y = tasa)) +
  geom_boxplot(fill = "pink") +
  labs(x = "Modelo", y = "Tasa de Fallos") +
  theme(axis.text.x = element_text(angle = angle, hjust = 1)) +
  ggtitle("TASA FALLOS")
p1  # Gráfico de TASA FALLOS


u1 <- union1
u1$modelo <- with(u1, reorder(modelo,auc, mean))

# Gráfico de cajas para AUC
p2 <- ggplot(u1, aes(x = modelo, y = auc)) +
  geom_boxplot(fill = "cyan") +
  labs(x = "Modelo", y = "AUC") +
  theme(axis.text.x = element_text(angle = angle, hjust = 1)) +
  ggtitle("AUC")
p2  # Gráfico de AUC
```

Listing B.3: Proceso de Selección de Conjunto de Variables Usando Modelos de Regresión Logistica Entrenados con los Conjuntos de Variables Candidatos

B.3. Python - Librerias Usadas y Funciones Creadas

Dado que el código completo desarrollado es extenso, en este anexo se incluyen solo ejemplos representativos de algunos fragmentos de código y funciones creadas y mencionadas en el cuerpo del documento. Si se requiere acceder al código completo, por favor, contactar a través del siguiente correo electrónico carlom36@ucm.es.

Librerias Usadas

[ ]:
```
from ucimlrepo import fetch_ucirepo

# Importar dataset desde UCI ML Repository
online_shoppers_purchasing_intention_dataset = fetch_ucirepo(id=468)

# Definir características y objetivos como dataframes de pandas
X = online_shoppers_purchasing_intention_dataset.data.features
y = online_shoppers_purchasing_intention_dataset.data.targets

# Análisis de Datos y Manipulación de Datos
import numpy as np  # Funciones numéricas eficientes
import pandas as pd  # Estructuras de datos y herramientas de análisis de␣
 ↪datos
from collections import Counter  # Conteo de elementos en colecciones

# Visualización de Datos
import matplotlib.pyplot as plt  # Visualización básica
import matplotlib.style as style  # Estilo de gráficos
import matplotlib.cm as cm  # Mapas de colores
import seaborn as sns  # Visualización estadística avanzada

from scipy.stats import chi2_contingency  # Prueba de independencia de␣
 ↪variables categóricas

# Bibliotecas de Sci-kit Learn
from sklearn.preprocessing import StandardScaler  # Escalado estándar de␣
 ↪características
from sklearn.model_selection import train_test_split, GridSearchCV  #␣
 ↪División de datos y búsqueda de hiperparámetros

# Métodos de desbalanceo de datos
from imblearn.under_sampling import NearMiss  # Submuestreo para clases␣
 ↪mayoritarias
from imblearn.over_sampling import RandomOverSampler  # Sobremuestreo de␣
 ↪clases minoritarias
from imblearn.combine import SMOTETomek  # Combinación de sobremuestreo y␣
 ↪submuestreo
from imblearn.ensemble import BalancedBaggingClassifier  # Ensamble para␣
 ↪manejo de desbalanceo

from sklearn.metrics.pairwise import euclidean_distances  # Distancias␣
 ↪euclidianas
```

```
from sklearn.preprocessing import MinMaxScaler  # Escalado Min-Max

from scipy.stats import ttest_ind, mannwhitneyu  # Prueba t y U de Mann-
Whitney para comparación de muestras

import category_encoders as ce  # Codificación de variables categóricas

import time  # Gestión del tiempo

# Bibliotecas para modelado de datos
import statsmodels.api as sm  # Modelos estadísticos y econométricos

from sklearn import svm, tree, linear_model, neighbors  # Modelos de␣
 ↪aprendizaje supervisado
from sklearn import naive_bayes, ensemble, discriminant_analysis,␣
 ↪gaussian_process
from sklearn.linear_model import LogisticRegression  # Regresión logística
from sklearn.tree import DecisionTreeClassifier  # Árboles de decisión
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis  #␣
 ↪Análisis discriminante lineal

from sklearn.neural_network import MLPClassifier  # Redes Neuronales
from sklearn.svm import SVC  # Máquinas de vectores de soporte

from sklearn.ensemble import RandomForestClassifier,␣
 ↪GradientBoostingClassifier  # Bosques aleatorios y Gradient Boosting
from xgboost import XGBClassifier  # XGBoost
from catboost import CatBoostClassifier  # CatBoost

from sklearn import model_selection  # Selección de modelos

# Módulos de sklearn para evaluación de modelos de ML
from sklearn.metrics import roc_auc_score, accuracy_score, recall_score,␣
 ↪precision_score, f1_score  # Métricas de evaluación
from sklearn.metrics import confusion_matrix, classification_report  #␣
 ↪Matriz de confusión y reporte de clasificación
from sklearn.model_selection import KFold, cross_val_score, cross_validate ␣
 ↪# Validación cruzada
from sklearn.model_selection import StratifiedKFold, RandomizedSearchCV,␣
 ↪ShuffleSplit  # Validación cruzada estratificada y búsqueda aleatoria

# Importación de funciones específicas para modelado interpretable
import shap  # SHAP (SHapley Additive exPlanations)
```

```
from sklearn.inspection import PartialDependenceDisplay, partial_dependence
 ↪# Visualización y cálculo de dependencia parcial (PDP)
import lime  # LIME (Local Interpretable Model-agnostic Explanations)
import lime.lime_tabular  # LIME para datos tabulares
from sklearn.tree import DecisionTreeRegressor, plot_tree, export_text  #
 ↪Árbol de regresión y visualización de árbol
```

Funciones

Analasis de variables categoricas

[ ]:
```
def plot_variable_distribution(df, target_variable):
    # Contar los valores de la variable objetivo
    counts = df[target_variable].value_counts()

    # Definir los colores para cada categoría (puedes ajustar estos colores
 ↪según tus necesidades)
    colors = ['crimson', 'limegreen']

    # Calcular el total de muestras
    total_samples = counts.sum()

    # Calcular los porcentajes
    percentages = counts / total_samples * 100

    # Crear el gráfico de barras horizontal
    plt.figure(figsize=(8, 2))  # Ajusta el tamaño según tus necesidades

    # Añadir las barras para cada categoría
    bars = []
    for i, (label, count) in enumerate(zip(counts.index, counts.values)):
        bar = plt.barh(label, count, color=colors[i % len(colors)],
 ↪label=f'{count} ({percentages[i]:.1f}%)', edgecolor='black')
        bars.append(bar)

    # Personalizar el gráfico
    plt.title('Distribución de la Variable', fontsize=14)
    plt.xlabel('Número de Muestras')
    plt.ylabel('Categoría')
    plt.xlim(0, total_samples)  # Establecer límites para el eje x
    plt.xticks(range(0, total_samples + 1, max(1, int(total_samples / 10))))
 ↪ # Ajustar las divisiones del eje x según el número total de muestras
    plt.legend()

    # Añadir etiquetas de valor y porcentaje en las barras
```

```
    for i, bar in enumerate(bars):
        for b in bar:
            plt.text(b.get_width() + 5, b.get_y() + b.get_height() / 2, f'{b.
↪get_width()} ({percentages[i]:.1f}%)', ha='center', va='center',␣
↪color='black')

    plt.tight_layout()
    plt.show()

# Ejemplo de uso:
# plot_variable_distribution(df_filtered, 'Revenue')
```

[ ]:
```
def univariate_categorical(df, target, column, label_rotation=False,␣
↪altura=6, ancho=16):

    if altura is None or (not isinstance(altura, (int, float))) or np.
↪isnan(altura):
        altura = 6
    if ancho is None or (not isinstance(ancho, (int, float))) or np.
↪isnan(ancho):
        ancho = 16

    plt.figure(figsize=(int(ancho), int(altura)))

    # Plot de barras para contar las ocurrencias de cada categoría en la␣
↪columna especificada
    plt.subplot(1, 2, 1)
    sns.countplot(data=df, x=column, hue=target, palette={0: 'red', 1:␣
↪'green'})
    plt.title(f'Observaciones de {column} por {target}')
    plt.xlabel(column)
    plt.ylabel('Cuenta')
    if label_rotation:
        plt.xticks(rotation=90)
    plt.legend(title=target)

    # Obtener el número de muestras en 1 de la variable 'Converted' por cada␣
↪categoría
    converted_counts = 100*df.groupby(column)[target].mean().
↪sort_values(ascending=False)

    # Imprimir el total de muestras por cada categoría y su porcentaje de␣
↪muestras en 'Converted = 1'
    print(f"{column}:")
```

```
    for cat in converted_counts.index:
        total_samples = df[column].value_counts()[cat]
        conv_samples = int(converted_counts[cat] * total_samples / 100)
        conv_percent = (converted_counts[cat]).round(2)
        print(f"{cat} | Total: {total_samples} | {target} = 1:␣
↪{conv_samples} ({conv_percent}%)")

    # Gráfico de barras ordenado por el porcentaje de registros positivos en␣
↪converted
    plt.subplot(1, 2, 2)
    sns.barplot(x=converted_counts.values, y=converted_counts.index,␣
↪palette='viridis')
    plt.title(f'Porcentaje de {target} por {column}')
    plt.xlabel('Porcentaje')
    plt.ylabel(column)

    plt.tight_layout()
    plt.show()
```

[ ]:
```
def plot_categorical_variable(df, variable, c_palette=None, altura=None,␣
↪ancho=None):
    # Obtener la cuenta de valores para la variable categórica
    variable_category_counts = df[variable].value_counts()

    if altura is None or (not isinstance(altura, (int, float))) or np.
↪isnan(altura):
        altura = 6
    if ancho is None or (not isinstance(ancho, (int, float))) or np.
↪isnan(ancho):
        ancho = 16

    plt.figure(figsize=(int(ancho), int(altura)))

    # Obtener el mapa de colores
    if c_palette is None or c_palette not in plt.colormaps():
        c_palette = 'rainbow'
    cmap = cm.get_cmap(c_palette)
    colors = cmap(np.linspace(0, 1, len(variable_category_counts)))

    # Graficar en modo horizontal
    bar_plot = variable_category_counts.sort_values().plot(kind='barh',␣
↪color=colors)

    plt.title(variable)
```

```
    plt.xlabel('Valores y Porcentajes')
    plt.ylabel('Categorías')

    # Agregar etiquetas numéricas a cada categoría
    for index, value in enumerate(variable_category_counts.sort_values()):
        plt.text(value, index, f'{value} ({value / len(df) * 100:.2f}%)',␣
 ↪ha='left', va='center')

    plt.tight_layout()
    plt.show()
```

[ ]:
```
def chi_square_test(df, target_variable, categorical_variables, alpha=0.05):
    results = []
    for cat_var in categorical_variables:
        # Crear la tabla de contingencia
        contingency_table = pd.crosstab(df[cat_var], df[target_variable])

        # Realizar la prueba de chi-cuadrado
        chi2, p, dof, expected = chi2_contingency(contingency_table)

        # Verificar tamaños de celda esperados
        if (expected < 5).any():
            warning = 'Alerta: Algunos valores esperados son menores que 5'
        else:
            warning = ''

        # Determinar la significancia
        significance = 'SI' if p < alpha else 'NO'

        # Almacenar los resultados
        results.append({
            'Variable': cat_var,
            'chi2': chi2,
            'p-value': p,
            'significance': significance,
            'warning': warning
        })

    return pd.DataFrame(results)

# Ejemplo de uso
resultados = chi_square_test(onlineShoping, 'Revenue', categorical_col)
resultados
```

[33]:

	Variable	chi2	p-value	significance	warning
0	Month	376.279817	1.566845e-75	SI	
1	OperatingSystems	75.069931	1.387988e-13	SI	Alerta: Algunos valores esperados son menores …
2	Browser	29.118987	3.782803e-03	SI	Alerta: Algunos valores esperados son menores …
3	Region	9.679351	2.882562e-01	NO	
4	TrafficType	359.224837	1.263264e-64	SI	Alerta: Algunos valores esperados son menores …
5	VisitorType	130.667627	4.225571e-29	SI	
6	Weekend	9.204803	2.413810e-03	SI	
7	Revenue	12197.419130	0.000000e+00	SI	

[ ]:
```
def categoric_similarity_clustering(df, categorical_features,␣
↪target_feature, threshold=0.1):
    """
    Función para agrupar categorías similares y asignar grupos basado en␣
↪múltiples variables categóricas.

    Parameters:
    df (DataFrame): DataFrame que contiene las variables de interés.
    categorical_features (list): Lista de nombres de columnas que contienen␣
↪variables categóricas a agrupar.
    target_feature (str): Nombre de la columna que contiene la variable␣
↪objetivo para calcular la similitud.
    threshold (float): Umbral de distancia para la agrupación (default: 0.1).

    Returns:
    df_grouped (DataFrame): DataFrame original con las columnas agrupadas␣
↪añadidas como <variable>_group.
    category_groups (list of lists): Lista de listas donde cada lista␣
↪interna contiene las categorías agrupadas por cada variable.
    category_to_group (dict): Diccionario que mapea cada categoría a su␣
↪respectivo grupo.
    """
    df_grouped = df.copy()
    category_to_group = {}
```

```
    for categorical_feature in categorical_features:
        grouped = df.groupby(categorical_feature)[target_feature].
↪agg(['mean', 'count']).reset_index()
        grouped.columns = [categorical_feature, 'Target_Mean',␣
↪'Sample_Count']

        scaler = MinMaxScaler()
        grouped['Scaled_Target_Mean'] = scaler.
↪fit_transform(grouped[['Target_Mean']])

        dist_matrix = euclidean_distances(grouped[['Scaled_Target_Mean']])

        category_groups = []
        visited = set()

        for i in range(len(grouped)):
            if i in visited:
                continue
            cluster = [grouped[categorical_feature].iloc[i]]
            visited.add(i)
            for j in range(i + 1, len(grouped)):
                if dist_matrix[i, j] <= threshold and j not in visited:
                    cluster.append(grouped[categorical_feature].iloc[j])
                    visited.add(j)
            category_groups.append(cluster)

        # Crear un diccionario para asignar un grupo a cada categoría de␣
↪esta variable
        for group_num, group in enumerate(category_groups):
            for category in group:
                category_to_group[(categorical_feature, category)] =␣
↪group_num

        # Asignar grupos al DataFrame original
        df_grouped[f'{categorical_feature}_group'] = df[categorical_feature].
↪map(
            lambda x: category_to_group.get((categorical_feature, x), -1)
        )

    return df_grouped, category_groups, category_to_group
```

Analasis de variables categoricas

[ ]:
```
import warnings

# Ignorar todas las advertencias
warnings.filterwarnings("ignore")

# Definir la paleta personalizada invirtiendo los colores de Dark2
custom_palette = sns.color_palette('Dark2', n_colors=2)
custom_palette[0], custom_palette[1] = custom_palette[1], custom_palette[0]

plt.figure(figsize=(6,30))  # Ajustar tamaño de la figura según la cantidad␣
 ↪de variables

i=1
# Iterar sobre las variables numéricas
for col in numeric_col:
    plt.subplot(16,2,i)
    sns.distplot(onlineShoping[col])
    plt.xlabel(col)  # Agregar nombre de la variable en el eje x
    plt.ylabel('Density')

    i += 1
    plt.subplot(16,2,i)
    sns.boxplot(x=onlineShoping['Revenue'], y=onlineShoping[col],␣
 ↪palette=custom_palette)
    plt.xlabel('Revenue')  # Agregar nombre del eje x para el gráfico de caja
    plt.ylabel(col)  # Agregar nombre de la variable en el eje y
    i += 1

plt.tight_layout()
plt.show()
```

Dado que los datos muestran asimetría y presencia significativa de outliers, el uso de pruebas paramétricas como el test de Welch o el test t de Student puede ser inapropiado debido a su supuesto de normalidad y sensibilidad a datos extremos. En cambio, la Prueba U de Mann-Whitney (Wilcoxon) se recomienda porque es una prueba no paramétrica que no requiere supuestos sobre la distribución de los datos y es robusta ante la presencia de asimetría y outliers. Esta prueba evalúa si las medianas de dos grupos independientes difieren, lo cual es crucial en problemas de clasificación binaria para determinar diferencias significativas en las características entre los grupos de interés.

[ ]:
```
def resumen_variable_continua(df, numeric_col, variable_binaria):
    resultados_lista = []

    for col in numeric_col:
        mediana = df[col].median()
```

```
        cuenta = df[col].count()
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        asimetria = df[col].skew()

        lim_inf = Q1 - 1.5 * IQR
        lim_sup = Q3 + 1.5 * IQR
        valores_atipicos = (df[col] < lim_inf) | (df[col] > lim_sup)
        outliers_revenue1 = valores_atipicos[df['Revenue'] == 1].sum()

        grupo_1 = df[col][variable_binaria == 0]
        grupo_2 = df[col][variable_binaria == 1]
        estadistico, p_valor = mannwhitneyu(grupo_1, grupo_2,␣
↪alternative='two-sided')

        if p_valor < 0.05:
            significancia = "Estadísticamente significativa"
        else:
            significancia = "No estadísticamente significativa"

        t_statistic, p_value = ttest_ind(df[col][df['Revenue'] == 0],␣
↪df[col][df['Revenue'] == 1])
        if p_value < 0.05:
            relacion_revenue = 'Significativa'
        else:
            relacion_revenue = 'No significativa'

        resultados = {'Variable': col,
                      'Total_Observaciones': cuenta,
                      'Mediana': mediana,
                      'Dispersión': IQR,
                      'Asimetría': asimetria,
                      'Outliers': valores_atipicos.sum(),
                      'Outliers_Revenue_1': outliers_revenue1,
                      'Outliers_Revenue_0': valores_atipicos.sum() -␣
↪outliers_revenue1,
                      'p-valor': p_valor,
                      'Estadístico U': estadistico,
                      'Significancia estadística': significancia}

        resultados_lista.append(resultados)

    df_resumen = pd.DataFrame(resultados_lista)
```

```
    return df_resumen
```

Los resultados de la prueba de hipótesis nula utilizando Mann-Whitney U indican que para todas las variables evaluadas, el valor de p es significativamente pequeño ($p < 0.05$), lo que sugiere que hay diferencias significativas entre los grupos donde Revenue = 0 y Revenue = 1 en términos de la variable específica. Esto significa que los outliers en la clase de Revenue = 0 tienen un impacto estadísticamente significativo en la distribución de estas variables en comparación con la clase de interés (Revenue = 1).

KDD: Transformación de Datos

- Feature Enginerring Se realizan cambios necesarios previo a seleccion de varialbes
  - combinacion de categorias poco representadas o eliminacion de las mismas
  - Ordinal encoding de variable Month
  - target encodign de varialbes categoricas:
    - OperatingSystems
    - Browser
    - Region
    - TrafficType

Ordinal Encoding de Variable Month

[ ]:
```
# Diccionario de meses
meses_diccionario = {
    'Jan': 1, 'Feb': 2, 'Mar': 3, 'Apr': 4, 'May': 5, 'June': 6,
    'Jul': 7, 'Aug': 8, 'Sep': 9, 'Oct': 10, 'Nov': 11, 'Dec': 12
}

# Aplica la transformación utilizando el método map()
onlineShoping['Month'] = onlineShoping['Month'].map(meses_diccionario)
```

Target Encoding

La aplicación de target encoding puede ser particularmente útil para las variables categóricas que muestran una significancia estadística fuerte con la variable objetivo. En este caso, las variables OperatingSystems, Browser, y TrafficType tienen un valor p (p-value) muy bajo, lo que indica una relación significativa con la variable objetivo y sugiere que el target encoding podría capturar esta relación de manera efectiva.

Por otro lado, la variable Region tiene un p-value relativamente alto, indicando que no hay suficiente evidencia de una relación significativa con la variable objetivo. Por lo tanto, aplicar target encoding a Region podría no ser tan beneficioso y, en su lugar, podrías considerar usar otras técnicas de codificación más simples o incluso descartar esta variable si no aporta información relevante.

instalcion de libreria category_encoders — > !pip install category_encoders # Instalar category_encoders

[ ]:
```
from sklearn.model_selection import train_test_split

# Separar características (X) y la variable objetivo (y)
X = onlineShoping.drop('Revenue', axis=1)
y = onlineShoping['Revenue']

# Partición 80-20 estratificada
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪stratify=y, random_state=1234)

# Crear los DataFrames para entrenamiento y prueba combinando X e y␣
 ↪respectivamente
onlineShoping_train = pd.concat([X_train, y_train], axis=1)
onlineShoping_test = pd.concat([X_test, y_test], axis=1)
```

En un problema de clasificación binaria con clases desbalanceadas, como cuando la clase de interés representa solo el 15% de las observaciones, es altamente recomendable utilizar el muestreo estratificado al separar los datos en conjuntos de entrenamiento y prueba. El muestreo estratificado asegura que la proporción de cada clase en el conjunto original se mantenga en ambos subconjuntos, lo cual es crucial para que el modelo tenga representaciones adecuadas de ambas clases durante el entrenamiento y para que las métricas de rendimiento sean representativas en el conjunto de prueba. Esto mitiga el riesgo de que el modelo no aprenda a clasificar correctamente la clase minoritaria y previene evaluaciones sesgadas, resultando en un mejor rendimiento del modelo y una evaluación más precisa.

[ ]:
```
# Variables categóricas y objetivo
categorical_columns = ['OperatingSystems', 'Browser', 'TrafficType']
target_column = 'Revenue'

# Codificación de objetivo sin suavizado
target_enc_no_smooth = ce.TargetEncoder(cols=categorical_columns,␣
 ↪smoothing=0.0)
target_enc_no_smooth.fit(onlineShoping_train[categorical_columns],␣
 ↪onlineShoping_train[target_column])

# Aplicar la codificación al conjunto de entrenamiento y prueba
onlineShoping_train_encoded = onlineShoping_train.copy()
onlineShoping_test_encoded = onlineShoping_test.copy()

onlineShoping_train_encoded[categorical_columns] = target_enc_no_smooth.
 ↪transform(onlineShoping_train[categorical_columns])
onlineShoping_test_encoded[categorical_columns] = target_enc_no_smooth.
 ↪transform(onlineShoping_test[categorical_columns])
```

```
# Almacenar los valores codificados para cada categoría
encoded_values = []
for col in categorical_columns:
    for cat, val_no_smooth in target_enc_no_smooth.mapping[col].items():
        encoded_values.append({
            'Variable': col,
            'Category': cat,
            'Encoded_Value_No_Smooth': val_no_smooth
        })

# Convertir la lista de valores codificados en un DataFrame
encoded_values_df = pd.DataFrame(encoded_values)
```

En un proceso de target encoding, los valores especiales como -1 y -2 se utilizan a menudo para representar categorías que no están presentes en el conjunto de datos de entrenamiento. Aquí se explica cómo se interpretan estos valores:

- Valor -1: Este valor generalmente se asigna a las categorías que aparecen en el conjunto de prueba pero no en el conjunto de entrenamiento. Esto puede ocurrir cuando se encuentran categorías nuevas o previamente no observadas en los datos de prueba. El valor -1 indica que no se tiene información suficiente para asignar un encoding basado en el objetivo, ya que esa categoría no fue vista durante el entrenamiento.

- Valor -2: Este valor se usa a veces para representar categorías que son extremadamente raras o tienen poca representación en el conjunto de datos de entrenamiento. En algunos casos, -2 también puede denotar categorías con datos faltantes o nulos. Este valor sirve para indicar que el encoding para esta categoría puede no ser fiable debido a la escasez de datos.

Estandarización

[ ]:
```
list_cont = ['Administrative', 'Administrative_Duration', 'Informational',
             'Informational_Duration', 'ProductRelated',
             'ProductRelated_Duration', 'BounceRates', 'ExitRates',
             'PageValues', 'SpecialDay', 'Month']

# Estandarizar las variables continuas y la variable objetivo
scaler = StandardScaler()

# Copiar los conjuntos de datos para aplicar el escalado
onlineShoping_train_scaled = onlineShoping_train_encoded.copy()
onlineShoping_test_scaled = onlineShoping_test_encoded.copy()

# Ajustar el scaler con los datos de entrenamiento y luego transformar los␣
 ↪datos de entrenamiento
```

```
onlineShoping_train_scaled[list_cont] = scaler.
 ↪fit_transform(onlineShoping_train_encoded[list_cont])

# Utilizar el scaler ajustado para transformar los datos de prueba
onlineShoping_test_scaled[list_cont] = scaler.
 ↪transform(onlineShoping_test_encoded[list_cont])
```

Entrenamiento de Modelos BlackBox

- Red Neuronal
  - suponiendo 10 observaciones por parámetro: $h(6+1)+h+1=\frac{1493}{10}\times 0.8$ (4 de 5 grupos definidos en CV) $\rightarrow h=14.8$
  - suponiendo 30 observaciones por parámetro: $h(6+1)+h+1=\frac{1493}{30}\times 0.8$ (4 de 5 grupos definidos en CV) $\rightarrow h=4.93$

Se define funcion para ver de forma grafica la evoluciond el promedio de la metrica elegida para determinar el mejor modelo usando grid_search de los resultados obtenidos al entrnear los modelso con los paremtros del grid y la CV con kfold = 5

[ ]:
```
def plot_metric_evolution(results, x_param, metric, color_param,␣
 ↪marker_param, figsize=(12, 8), legend_padding=0.01):
    # Crear un DataFrame a partir de los resultados
    df = pd.DataFrame(results)

    # Agrupar por los parámetros y calcular el promedio de la métrica
    grouped_df = df.groupby([x_param, color_param, marker_param])[metric].
 ↪mean().reset_index()

    # Obtener los valores de los parámetros y la métrica
    x_values = grouped_df[x_param]
    metric_values = grouped_df[metric]
    color_values = grouped_df[color_param]
    marker_values = grouped_df[marker_param]

    # Obtener valores únicos para el parámetro de color y de marcador
    unique_colors = np.unique(color_values)
    unique_markers = np.unique(marker_values)

    # Definir marcadores para cada valor único en el parámetro de marcador
    markers = ['*', 'o', '^', 's', 'D', 'P', 'X', 'v', '<', '>']  # Puedes␣
 ↪expandir esta lista si es necesario

    # Crear el gráfico
    fig, ax = plt.subplots(figsize=figsize)
```

```
    for i, color_val in enumerate(unique_colors):
        for j, marker_val in enumerate(unique_markers):
            mask = (color_values == color_val) & (marker_values ==␣
↪marker_val)
            ax.scatter(x_values[mask], metric_values[mask],
                       color=f'C{i}', marker=markers[j % len(markers)],␣
↪label=f'{color_param}={color_val}, {marker_param}={marker_val}')

    # Añadir la primera leyenda (para color_param)
    handles, labels = ax.get_legend_handles_labels()

    unique_color_labels = {f'{color_param}={color_val}' for color_val in␣
↪unique_colors}
    unique_marker_labels = {f'{marker_param}={marker_val}' for marker_val in␣
↪unique_markers}

    color_legend_handles = [ax.scatter([], [], color=f'C{i}',␣
↪label=f'{color_param}={color_val}') for i, color_val in␣
↪enumerate(unique_colors)]
    marker_legend_handles = [ax.scatter([], [], color='black',␣
↪marker=markers[j % len(markers)], label=f'{marker_param}={marker_val}')␣
↪for j, marker_val in enumerate(unique_markers)]

    # Calcular el ancho del margen según la longitud de los textos de la␣
↪leyenda
    max_color_label_length = max([len(f'{color_param}={color_val}') for␣
↪color_val in unique_colors])
    max_marker_label_length = max([len(f'{marker_param}={marker_val}') for␣
↪marker_val in unique_markers])
    max_label_length = max(max_color_label_length, max_marker_label_length)

    # Ajustar el margen derecho dinámicamente con límites para evitar errores
    right_margin = min(0.8, legend_padding + 0.02 * max_label_length)  #␣
↪Limitar el margen derecho entre 0.1 y 0.8

    # Ajustar los márgenes de la figura
    plt.subplots_adjust(left=0., right=1-right_margin, top=0.9, bottom=0.1) ␣
↪# Ajustar left a 0.1 para minimizar el espacio a la izquierda

    # Añadir la leyenda
    handles, labels = ax.get_legend_handles_labels()
    plt.legend(handles, labels, loc='center left', bbox_to_anchor=(1, 0.5),␣
↪title="Legend", fontsize=8)
```

```
    # Ajustar posición de la leyenda de color_param a la derecha
    legend1 = ax.legend(handles=color_legend_handles, loc='upper left',␣
 ↪title=f'{color_param}', bbox_to_anchor=(1, 1), borderaxespad=0.,␣
 ↪fontsize=8)
    ax.add_artist(legend1)

    # Ajustar posición de la leyenda de marker_param a la derecha y abajo
    legend2 = ax.legend(handles=marker_legend_handles, loc='lower left',␣
 ↪title=f'{marker_param}', bbox_to_anchor=(1, 0), borderaxespad=0.,␣
 ↪fontsize=8)

    # Ajustes adicionales del gráfico
    plt.xlabel(x_param, fontsize=12)
    plt.ylabel(metric, fontsize=12)
    plt.title(f'Evolución de {metric} con respecto a {x_param},\n coloreado␣
 ↪por {color_param} y marcado por {marker_param}', fontsize=16)
    plt.grid(True)

    plt.show()
```

[ ]:
```
# Definir los valores para los parámetros que se van a ajustar
param_grid = {
    'hidden_layer_sizes': [(5,),(7,),(8,),(10,), (10,), (11,), (12,)],
    'alpha': [0.01, 0.02, 0.05, 0.08, 0.1],
    'max_iter': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]
}
```

[ ]:
```
# Crear el modelo MLPClassifier
model = MLPClassifier(solver='adam', random_state=42)

# Definir la validación cruzada estratificada con k-fold = 5
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Configurar GridSearchCV para encontrar los mejores parámetros
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=kfold,␣
 ↪scoring='roc_auc', n_jobs=-1, verbose=2)

# Obtener el tiempo inicial
start_time = time.time()
```

```
# Realizar la búsqueda de los mejores parámetros
grid_search.fit(X_train, y_train)

# Obtener el tiempo final y calcular el tiempo transcurrido
end_time = time.time()
elapsed_time = end_time - start_time

# Obtener el mejor modelo y los mejores parámetros
best_model = grid_search.best_estimator_
best_params = grid_search.best_params_
print("Mejores parámetros encontrados:")
print(best_params)
```

```
Fitting 5 folds for each of 350 candidates, totalling 1750 fits
Mejores parámetros encontrados:
{'alpha': 0.02, 'hidden_layer_sizes': (11,), 'max_iter': 300}
```

[ ]:
```
# Ejemplo de uso
plot_metric_evolution(results_ann_2, 'param_max_iter', 'mean_test_score',␣
↪'param_hidden_layer_sizes', 'param_alpha', figsize=(12, 4),␣
↪legend_padding=0.1)
```

[ ]:
```
def plot_metric_evolution_v2(results, x_param, metric, color_param=None,␣
↪marker_param=None, figsize=(12, 8), legend_padding=0.01):
    # Crear un DataFrame a partir de los resultados
    df = pd.DataFrame(results)

    if color_param is None and marker_param is None:
        # Si no se especifican color_param y marker_param, graficar sin␣
↪agrupar
        plt.figure(figsize=figsize)
        plt.scatter(df[x_param], df[metric])
        plt.xlabel(x_param, fontsize=12)
        plt.ylabel(metric, fontsize=12)
        plt.title(f'{metric} vs {x_param}', fontsize=16)
        plt.grid(True)
        plt.show()
        return

    # Agrupar por los parámetros y calcular el promedio de la métrica si se␣
↪especifican color_param y marker_param
    if color_param and marker_param:
        grouped_df = df.groupby([x_param, color_param,␣
↪marker_param])[metric].mean().reset_index()
    elif color_param:
```

```
        grouped_df = df.groupby([x_param, color_param])[metric].mean().
↪reset_index()
    elif marker_param:
        grouped_df = df.groupby([x_param, marker_param])[metric].mean().
↪reset_index()

    # Obtener los valores de los parámetros y la métrica
    x_values = grouped_df[x_param]
    metric_values = grouped_df[metric]

    if color_param:
        color_values = grouped_df[color_param]
        unique_colors = np.unique(color_values)
        color_legend_handles = [plt.scatter([], [], color=f'C{i}',␣
↪label=f'{color_param}={color_val}') for i, color_val in␣
↪enumerate(unique_colors)]
        legend1 = plt.legend(handles=color_legend_handles, loc='upper left',␣
↪title=f'{color_param}', bbox_to_anchor=(1, 1), borderaxespad=0.,␣
↪fontsize=8)
        plt.gca().add_artist(legend1)

    if marker_param:
        marker_values = grouped_df[marker_param]
        markers = ['*', 'o', '^', 's', 'D', 'P', 'X', 'v', '<', '>']  #␣
↪Puedes expandir esta lista si es necesario
        unique_markers = np.unique(marker_values)
        marker_legend_handles = [plt.scatter([], [], color='black',␣
↪marker=markers[j % len(markers)], label=f'{marker_param}={marker_val}')␣
↪for j, marker_val in enumerate(unique_markers)]
        plt.legend(handles=marker_legend_handles, loc='lower left',␣
↪title=f'{marker_param}', bbox_to_anchor=(1, 0), borderaxespad=0.,␣
↪fontsize=8)

    # Crear el gráfico
    plt.figure(figsize=figsize)

    for i, row in grouped_df.iterrows():
        if color_param and marker_param:
            plt.scatter(row[x_param], row[metric], color=f'C{i %␣
↪len(unique_colors)}', marker=markers[i % len(markers)],␣
↪label=f'{color_param}={row[color_param]},␣
↪{marker_param}={row[marker_param]}')
        elif color_param:
```

```
            plt.scatter(row[x_param], row[metric], color=f'C{i %␣
↪len(unique_colors)}', label=f'{color_param}={row[color_param]}')
        elif marker_param:
            plt.scatter(row[x_param], row[metric], marker=markers[i %␣
↪len(markers)], label=f'{marker_param}={row[marker_param]}')

    # Ajustes adicionales del gráfico
    plt.xlabel(x_param, fontsize=12)
    plt.ylabel(metric, fontsize=12)
    plt.title(f'Evolución de {metric} con respecto a {x_param},\n coloreado␣
↪por {color_param} y marcado por {marker_param}', fontsize=16)
    plt.grid(True)

    plt.show()
```

- SVM Polinomial
  - Rango típico: Enteros positivos (por ejemplo, 2, 3, 4).
  - Consideraciones: Un grado más alto puede llevar a un modelo más complejo y propenso a sobreajustar.
- SVM RBF
  - Rango típico: Valores positivos, generalmente en escala logarítmica (por ejemplo, 0.001, 0.01, 0.1, 1, 10).
  - Consideraciones: Valores más altos de gamma pueden llevar a un sobreajuste, mientras que valores bajos pueden llevar a un subajuste.

[ ]:
```
def plot_metric_with_marker_legend(results, x_param, metric,␣
↪marker_param=None, figsize=(12, 8)):
    # Crear un DataFrame a partir de los resultados
    df = pd.DataFrame(results)

    # Verificar si marker_param se especificó y obtener los valores únicos
    if marker_param is not None:
        marker_values = df[marker_param].unique()
    else:
        marker_values = [None]  # Usar None si no se especificó marker_param␣
↪para poder iterar al menos una vez

    # Definir marcadores para cada valor único en el parámetro de marcador
    markers = ['*', 'o', '^', 's', 'D', 'P', 'X', 'v', '<', '>']  # Puedes␣
↪expandir esta lista si es necesario

    # Crear el gráfico
    plt.figure(figsize=figsize)
```

```
    for i, marker_val in enumerate(marker_values):
        if marker_param is not None:
            mask = df[marker_param] == marker_val
            plt.scatter(df[x_param][mask], df[metric][mask],␣
↪marker=markers[i % len(markers)], label=f'{marker_param}={marker_val}')
        else:
            plt.scatter(df[x_param], df[metric], marker=markers[i %␣
↪len(markers)], label=f'{marker_param}')

    # Añadir leyenda para marker_param si está presente
    if marker_param is not None:
        plt.legend(title=marker_param, fontsize=8)

    # Ajustes adicionales del gráfico
    plt.xlabel(x_param, fontsize=12)
    plt.ylabel(metric, fontsize=12)
    plt.title(f'{metric} vs {x_param} con leyenda de {marker_param}',␣
↪fontsize=16)
    plt.grid(True)

    plt.show()
```

- RF
  - Inicio con Modelo Bagging y viendo numero de arboles (n_estimators cuanto afecta)
  - Como el numero de observaciones de la clase de interes no es tan elevado un valor de 30 seria un maximo pero no se podria usar 50, por esta razon se decide probar 10,15,20 en el nuemro de min_sample_split (equivalente en R a nodesize)

Para asegurar que cada fold de validación cruzada contenga una distribución equitativa de observaciones, cuando se utiliza el método “cv”, se deja fuera un fold con un tamaño de aproximadamente 9508/5 = 1902 observaciones en cada ejecución de la validación cruzada. Esto significa que se utilizan 4/5 × 9508 = 7606 observaciones para entrenar el modelo. Por lo tanto, el tamaño máximo de la muestra de entrenamiento (sampsize) debe ser aproximadamente 7605.

Por lo anterior se dejara un margen y se definira un maximo de 7500 para la primera prueba de ajuste de modelo.

[ ]:
```
# Definir el grid de parámetros para RandomForestClassifier
param_grid_RF = {
    'min_samples_split': [20],      # Equivalente a nodesize en R
    'max_features': [3, 4, 5, 6],                # Equivalente a mtry en R
    'n_estimators': [1750,2000, 2500, 3000,3500,4000,4500,5000,5250],  #␣
↪Equivalente a ntree en R
    'bootstrap': [True],                    # Equivalente a sampsize en R
```

```
    'random_state': [42]
}
```

- GBM

[ ]:
```
# Definir el grid de parámetros para GradientBoostingClassifier
param_grid_GBM = {
    'learning_rate': [0.1, 0.05, 0.01],
    'min_samples_split': [15,20],
    'n_estimators': [1000,1500,2000,2500,3000]
}
```

[ ]:
```
# Crear el modelo GradientBoostingClassifier
model = GradientBoostingClassifier(random_state=42)

# Definir la validación cruzada estratificada con k-fold = 5
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Configurar GridSearchCV para encontrar los mejores parámetros
grid_search = GridSearchCV(estimator=model, param_grid=param_grid_GBM,␣
 ↪cv=kfold, scoring='roc_auc', n_jobs=-1, verbose=2)

# Obtener el tiempo inicial
start_time = time.time()

# Realizar la búsqueda de los mejores parámetros
grid_search.fit(X_train, y_train)

# Obtener el tiempo final y calcular el tiempo transcurrido
end_time = time.time()
elapsed_time = end_time - start_time

# Obtener el mejor modelo y los mejores parámetros
best_model = grid_search.best_estimator_
best_params = grid_search.best_params_
print("Mejores parámetros encontrados:")
print(best_params)
```

```
Fitting 5 folds for each of 30 candidates, totalling 150 fits
Mejores parámetros encontrados:
{'learning_rate': 0.01, 'min_samples_split': 20, 'n_estimators': 1000}
```

- XGBoost

[ ]:
```
# Definir los valores para los parámetros que se van a ajustar
param_grid_XGB = {
```

```
    'min_child_weight': [15, 20],
    'learning_rate': [0.01, 0.05],
    'n_estimators': [300, 400, 500, 700, 750, 800, 1000, 1250, 1500, 1750],
    'max_depth': [4, 5, 6],
    'gamma': [0],
    'colsample_bytree': [0.8, 1],
    'subsample': [0.8, 1]
}
```

- CatBoost

[ ]:
```
param_grid_CatBoost = {
    'learning_rate': [0.01, 0.05],
    'n_estimators': [500, 750, 1000, 1250, 1500, 1750, 2000],
    'depth': [6],
    'l2_leaf_reg': [10, 15, 20],  # Regularización L2
    'subsample': [0.8, 1],
    'colsample_bylevel': [0.8, 1]
}
```

[ ]:
```
# Crear el modelo CatBoostClassifier
model = CatBoostClassifier(objective='Logloss', eval_metric='AUC',␣
 ↪random_seed=42, verbose=False)
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
grid_search = GridSearchCV(estimator=model, param_grid=param_grid_CatBoost,␣
 ↪cv=kfold, scoring='roc_auc',  n_jobs=-1, verbose=2)

start_time = time.time()
grid_search.fit(X_train, y_train)
end_time = time.time()
elapsed_time = end_time - start_time
elapsed_time

best_model = grid_search.best_estimator_
best_params = grid_search.best_params_
print("Mejores parámetros encontrados:")
print(best_params)
```

```
Mejores parámetros encontrados:
{'colsample_bylevel': 0.8, 'depth': 6, 'l2_leaf_reg': 10, 'learning_rate': 0.
 ↪01,
'n_estimators': 1250, 'subsample': 1}
```

Comparacion de Modelos

[ ]:
```
# Modelos y sus hiperparámetros
models = []

models.append(('Logistic Regression', LogisticRegression(solver='liblinear',␣
 ↪random_state=42)))
models.append(('ANN', MLPClassifier(alpha=0.02, hidden_layer_sizes=(11,),␣
 ↪max_iter=300, solver='adam', random_state=42)))
models.append(('SVM_Linear', SVC(kernel='linear', C=10, random_state=1234)))
models.append(('SVM_Poly', SVC(kernel='poly', C=1, degree=3, coef0=3,␣
 ↪random_state=1234)))
models.append(('SVM_RBF', SVC(kernel='rbf', C=5, gamma=0.1,␣
 ↪random_state=1234)))
models.append(('Random Forest', RandomForestClassifier(n_estimators=5250,␣
 ↪max_features=3, min_samples_split=20, random_state=42, n_jobs=-1,␣
 ↪bootstrap=True)))
models.append(('GBM', GradientBoostingClassifier(learning_rate=0.01,␣
 ↪min_samples_split=20, n_estimators=1000, random_state=42)))
models.append(('XGBoost', XGBClassifier(colsample_bytree=1, gamma=0,␣
 ↪learning_rate=0.008, max_depth=4, min_child_weight=15, n_estimators=700,␣
 ↪subsample=1, random_state=42)))
models.append(('CatBoost', CatBoostClassifier(colsample_bylevel=0.8,␣
 ↪depth=6, l2_leaf_reg=10, learning_rate=0.01, n_estimators=1250,␣
 ↪subsample=1, random_state=42)))
```

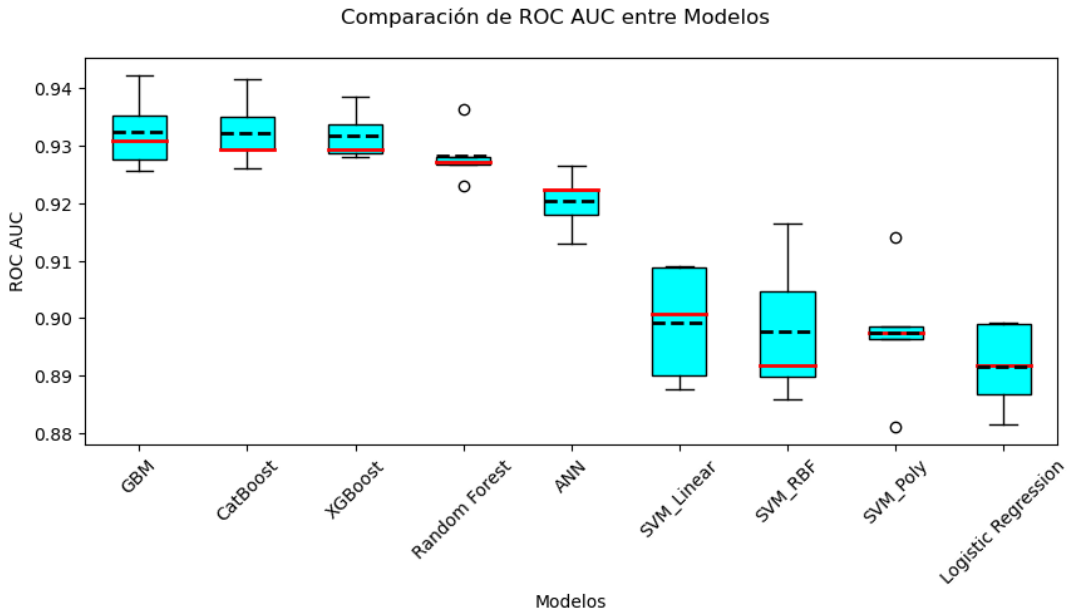
[ ]:
```
# Evaluación del modelo
acc_results = []
auc_results = []
f1_results = []
recall_results = []
precision_results = []
names = []
# Configuración de la tabla de resultados
col = ['ML_Model', 'ROC AUC Mean', 'ROC AUC STD', 'Accuracy Mean', 'Accuracy␣
 ↪STD', 'F1-Score Mean', 'Recall Mean', 'Precision Mean']
df_results = pd.DataFrame(columns=col)
i = 0

# Evaluar cada modelo utilizando validación cruzada en todo el conjunto de␣
 ↪datos
for name, model in models:
    kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42) # 5-
fold cross-validation
    cv_results = cross_validate(model, X_train, y_train, cv=kfold,␣
 ↪scoring=['roc_auc', 'accuracy', 'f1', 'recall', 'precision'], n_jobs=-1)
```

```
    roc_auc_mean = cv_results['test_roc_auc'].mean()
    roc_auc_std = cv_results['test_roc_auc'].std()
    acc_mean = cv_results['test_accuracy'].mean()
    acc_std = cv_results['test_accuracy'].std()
    f1_mean = cv_results['test_f1'].mean()
    recall_mean = cv_results['test_recall'].mean()
    precision_mean = cv_results['test_precision'].mean()

    acc_results.append(cv_results['test_accuracy'])
    auc_results.append(cv_results['test_roc_auc'])
    f1_results.append(cv_results['test_f1'])
    recall_results.append(cv_results['test_recall'])
    precision_results.append(cv_results['test_precision'])
    names.append(name)

    df_results.loc[i] = [name,
                         round(roc_auc_mean * 100, 2),
                         round(roc_auc_std * 100, 2),
                         round(acc_mean * 100, 2),
                         round(acc_std * 100, 2),
                         round(f1_mean * 100, 2),
                         round(recall_mean * 100, 2),
                         round(precision_mean * 100, 2)
                        ]

    i += 1
```

[ ]:
```
def plot_metrics_comparison(metric_results, model_names, metric_name,␣
↪colors=None):
    # Calcula el promedio de la métrica para cada modelo
    mean_metric = [np.mean(metric) for metric in metric_results]

    # Ordena los modelos por el promedio de la métrica de mayor a menor
    sorted_indices = np.argsort(mean_metric)[::-1]
    sorted_metric_results = [metric_results[i] for i in sorted_indices]
    sorted_model_names = [model_names[i] for i in sorted_indices]

    # Si no se especifican colores, usar colores por defecto
    if colors is None:
        colors = ['cyan', 'lightgreen', 'gold']  # Colores por defecto para␣
↪las cajas
```

```
    # Crea el gráfico de cajas (boxplot) ordenado por el promedio de la␣
↪métrica
    fig = plt.figure(figsize=(10, 4))
    fig.suptitle(f'Comparación de {metric_name} entre Modelos')

    ax = fig.add_subplot(111)
    bp = ax.boxplot(sorted_metric_results, patch_artist=True,␣
↪showmeans=True, meanline=True)

    # Colorea el interior de las cajas con los colores especificados
    for i, box in enumerate(bp['boxes']):
        box.set(facecolor=colors[i % len(colors)])

    # Configura el color de las líneas de la mediana y media
    for median in bp['medians']:
        median.set(color='red', linewidth=2)

    for mean in bp['means']:
        mean.set(color='black', linewidth=2)

    # Configura etiquetas del eje x
    ax.set_xticklabels(sorted_model_names)

    # Configura etiquetas y título de los ejes
    ax.set_xlabel('Modelos')
    ax.set_ylabel(metric_name)

    plt.xticks(rotation=45)  # Rotar las etiquetas para mejor visualización␣
↪si es necesario
    plt.show()
```

[ ]:
```
# Ejemplo de uso con especificación de colores
colors_auc = ['cyan']  # Colores personalizados para ROC AUC
plot_metrics_comparison(auc_results, names, 'ROC AUC', colors=colors_auc)
```



Comparacion de Resultados

[ ]:
```
# Lista de modelos a evaluar con sus respectivos parámetros
models = []
models.append(('GBM', GradientBoostingClassifier(learning_rate=0.01,␣
 ↪min_samples_split=20, n_estimators=1000, random_state=42)))
models.append(('XGBoost', XGBClassifier(colsample_bytree=1, gamma=0,␣
 ↪learning_rate=0.008, max_depth=4, min_child_weight=15, n_estimators=700,␣
 ↪subsample=1, random_state=42)))
models.append(('CatBoost', CatBoostClassifier(colsample_bylevel=0.8,␣
 ↪depth=6, l2_leaf_reg=10, learning_rate=0.01, n_estimators=1250,␣
 ↪subsample=1, random_state=42, verbose=False)))
models.append(('GBM_allVar', GradientBoostingClassifier(learning_rate=0.01,␣
 ↪min_samples_split=20, n_estimators=750, random_state=42)))
models.append(('XGBoost_allVar', XGBClassifier(colsample_bytree=0.8,␣
 ↪gamma=0, learning_rate=0.005, max_depth=6, min_child_weight=10,␣
 ↪n_estimators=750, subsample=0.8, random_state=42)))
models.append(('CatBoost_allVar', CatBoostClassifier(colsample_bylevel=0.8,␣
 ↪depth=6, l2_leaf_reg=10, learning_rate=0.01, n_estimators=1000,␣
 ↪subsample=0.8, random_state=42, verbose=False)))

# Crear objeto de validación cruzada (5-fold)
kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
```

[ ]:
```
df_results = df_results.sort_values(by=['ROC AUC Mean'], ascending=False)
df_results
```

[19]:

	ML_Model	ROC AUC Mean	ROC AUC STD	Accuracy Mean	Accuracy STD	\
4	XGBoost_allVar	93.39	0.53	90.01	0.64	
3	GBM_allVar	93.26	0.60	90.15	0.45	
0	GBM	93.23	0.60	90.22	0.62	
2	CatBoost	93.23	0.54	90.06	0.48	
5	CatBoost_allVar	93.19	0.59	89.77	0.52	
1	XGBoost	93.17	0.40	90.07	0.56	

	F1-Score Mean	Recall Mean	Precision Mean
4	64.47	57.74	73.00
3	65.45	59.48	72.78
0	66.40	61.55	72.11
2	65.58	60.35	71.86
5	64.11	58.20	71.36
1	65.75	60.75	71.71

Sin la modificacion de un putno de corte el el modelo GBM con todas las varialbes parece ser la mejor opcion al ser el segundo mejor resutlado en tanto en ROC-AUC como en accuracy y tener buenos resutlado sen Recall

Evaluacion con distintos putnos de corte

[ ]:
```
# Lista de modelos a evaluar con sus respectivos parámetros
models = []
models.append(('GBM', GradientBoostingClassifier(learning_rate=0.01,␣
 ↪min_samples_split=20, n_estimators=1000, random_state=42)))
models.append(('XGBoost', XGBClassifier(colsample_bytree=1, gamma=0,␣
 ↪learning_rate=0.008, max_depth=4, min_child_weight=15, n_estimators=700,␣
 ↪subsample=1, random_state=42)))
models.append(('CatBoost', CatBoostClassifier(colsample_bylevel=0.8,␣
 ↪depth=6, l2_leaf_reg=10, learning_rate=0.01, n_estimators=1250,␣
 ↪subsample=1, random_state=42, verbose=False)))

# Crear objeto de validación cruzada (5-fold)
kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Lista de puntos de corte a evaluar
puntos_de_corte = [0.3, 0.35, 0.4, 0.45, 0.5]
```

[ ]:
```
# Configuración de la tabla de resultados
col = ['ML_Model', 'Threshold', 'ROC AUC', 'Accuracy', 'Recall',␣
 ↪'Precision', 'Specificity', 'F1-Score', 'TP', 'TN', 'FP', 'FN']
df_results = pd.DataFrame(columns=col)
```

```
# Evaluar cada modelo utilizando validación cruzada en todo el conjunto de␣
 ↪datos
for name, model in models:
    if '_allVar' in name:
        X_train_model = X_train_all  # Utilizar X_train_all para modelos con␣
 ↪_allVar en el nombre
    else:
        X_train_model = X_train  # Utilizar X_train por defecto

    for train_index, test_index in kf.split(X_train_model, y_train):
        X_train_fold, X_test_fold = X_train_model.iloc[train_index],␣
 ↪X_train_model.iloc[test_index]
        y_train_fold, y_test_fold = y_train.iloc[train_index], y_train.
 ↪iloc[test_index]

        model.fit(X_train_fold, y_train_fold)
        y_proba = model.predict_proba(X_test_fold)[:, 1]

        for threshold in puntos_de_corte:
            y_pred = (y_proba >= threshold).astype(int)
            roc_auc = roc_auc_score(y_test_fold, y_proba)
            accuracy = accuracy_score(y_test_fold, y_pred)
            recall = recall_score(y_test_fold, y_pred)
            precision = precision_score(y_test_fold, y_pred)
            tn, fp, fn, tp = confusion_matrix(y_test_fold, y_pred).ravel()
            specificity = tn / (tn + fp)
            f1 = f1_score(y_test_fold, y_pred)

            # Crear un DataFrame temporal para concatenar
            temp_df = pd.DataFrame([{
                'ML_Model': name,
                'Threshold': threshold,
                'ROC AUC': roc_auc,
                'Accuracy': accuracy,
                'Recall': recall,
                'Precision': precision,
                'Specificity': specificity,
                'F1-Score': f1,
                'TP': tp,
                'TN': tn,
                'FP': fp,
                'FN': fn
            }])
            df_results = pd.concat([df_results, temp_df], ignore_index=True)
```

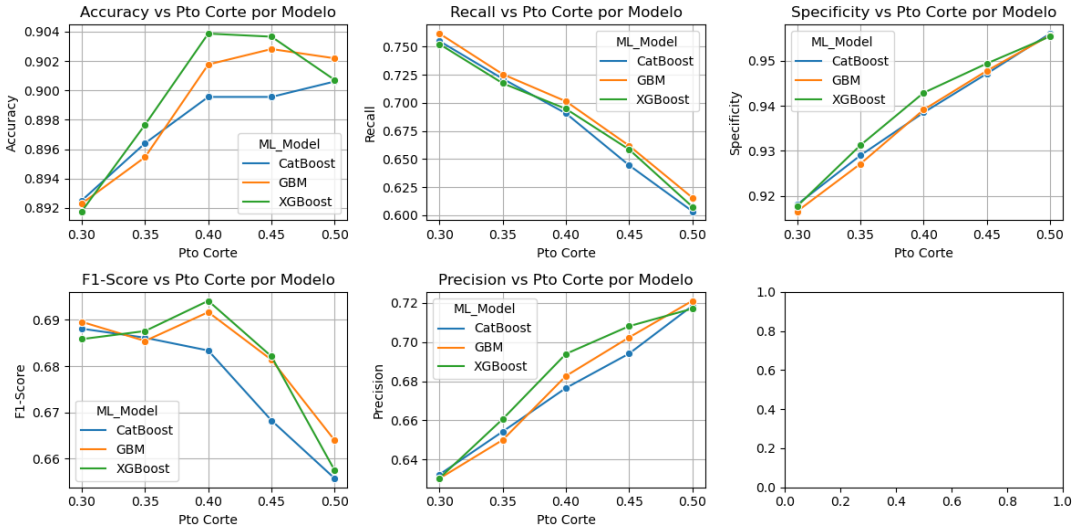
[ ]:
```
# Agrupar por ['ML_Model', 'Threshold'] y calcular los promedios
df_ptoCorte_Summary = df_results.groupby(['ML_Model', 'Threshold']).mean().
 ↪reset_index()

# Renombrar la columna 'Threshold' a 'Pto Corte'
df_ptoCorte_Summary.rename(columns={'Threshold': 'Pto Corte'}, inplace=True)
df_ptoCorte_Summary
```

[46]:

	ML_Model	Pto Corte	ROC AUC	Accuracy	Recall	Precision	Specificity
0	CatBoost	0.30	0.932300	0.892511	0.754841	0.632275	0.918153
1	CatBoost	0.35	0.932300	0.896403	0.721349	0.654269	0.929008
2	CatBoost	0.40	0.932300	0.899558	0.690541	0.676548	0.938490
3	CatBoost	0.45	0.932300	0.899558	0.644333	0.694143	0.947099
4	CatBoost	0.50	0.932300	0.900610	0.603468	0.718590	0.955958
5	GBM	0.30	0.932348	0.892301	0.761541	0.630273	0.916656
6	GBM	0.35	0.932348	0.895457	0.725378	0.649980	0.927137
7	GBM	0.40	0.932348	0.901767	0.701261	0.682697	0.939114
8	GBM	0.45	0.932348	0.902819	0.661747	0.702429	0.947723
9	GBM	0.50	0.932348	0.902188	0.615522	0.721138	0.955583
10	XGBoost	0.30	0.931747	0.891775	0.752172	0.630405	0.917779
11	XGBoost	0.35	0.931747	0.897665	0.717340	0.660651	0.931254
12	XGBoost	0.40	0.931747	0.903871	0.694568	0.693993	0.942857
13	XGBoost	0.45	0.931747	0.903660	0.658389	0.708175	0.949345
14	XGBoost	0.50	0.931747	0.900715	0.607475	0.717124	0.955334

	F1-Score	TP	TN	FP	FN
0	0.688090	225.4	1471.8	131.2	73.2
1	0.686140	215.4	1489.2	113.8	83.2
2	0.683379	206.2	1504.4	98.6	92.4
3	0.668276	192.4	1518.2	84.8	106.2
4	0.655756	180.2	1532.4	70.6	118.4
5	0.689560	227.4	1469.4	133.6	71.2
6	0.685439	216.6	1486.2	116.8	82.0
7	0.691659	209.4	1505.4	97.6	89.2
8	0.681370	197.6	1519.2	83.8	101.0
9	0.663975	183.8	1531.8	71.2	114.8
10	0.685840	224.6	1471.2	131.8	74.0
11	0.687578	214.2	1492.8	110.2	84.4
12	0.694108	207.4	1511.4	91.6	91.2
13	0.682100	196.6	1521.8	81.2	102.0
14	0.657487	181.4	1531.4	71.6	117.2

[ ]:
```
# Gráficos de líneas para comparar las métricas por modelos y puntos de corte
metrics = ['Accuracy', 'Recall', 'Specificity', 'F1-Score', 'Precision']
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(12, 6))

for i, metric in enumerate(metrics):
    row = i // 3
    col = i % 3
    sns.lineplot(ax=axes[row, col], data=df_ptoCorte_Summary, x='Pto Corte',␣
 ↪y=metric, hue='ML_Model', marker='o')
    axes[row, col].set_title(f'{metric} vs Pto Corte por Modelo')
    axes[row, col].set_xlabel('Pto Corte')
    axes[row, col].set_ylabel(metric)
    axes[row, col].legend(title='ML_Model')
    axes[row, col].grid(True)

plt.tight_layout()
plt.show()
```

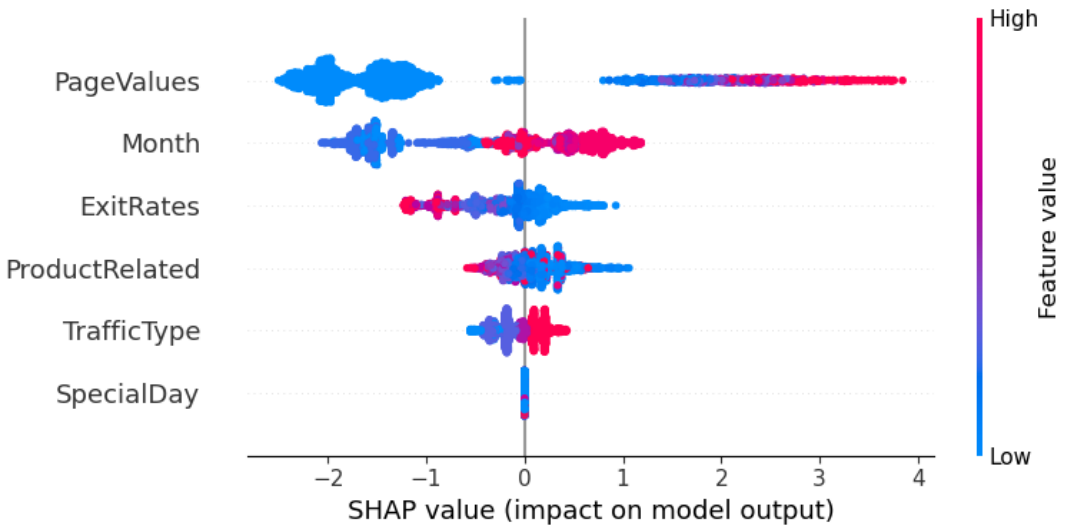


SHAP

SHAP es una técnica que utiliza la teoría de juegos para asignar una contribución a cada característica en función de su valor esperado condicionalmente. shap.summary_plot proporciona una visión general de la importancia relativa de las características para todas las predicciones del modelo.

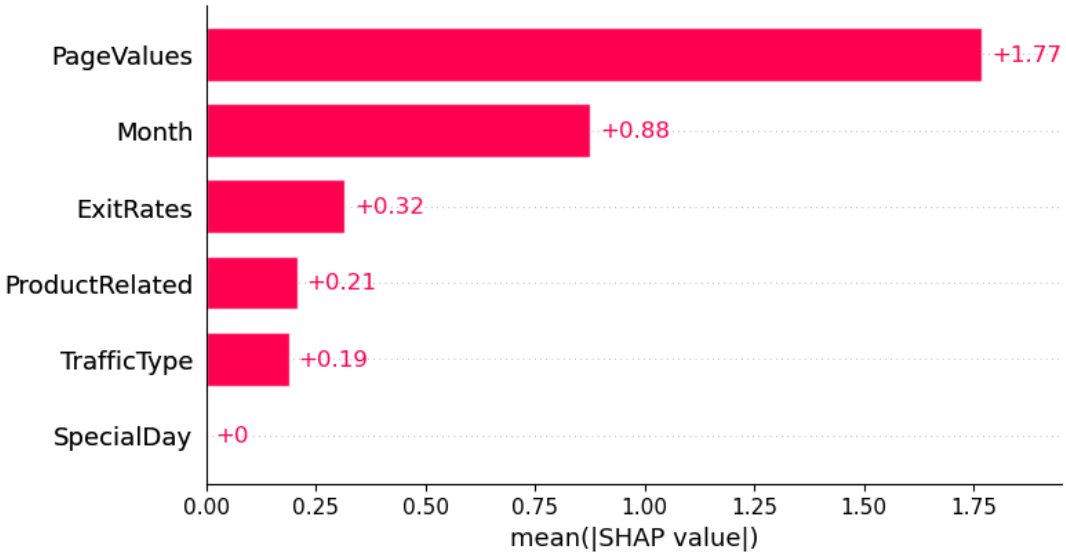
[ ]:
```
# Paso 1: Crear el explainer con TreeExplainer
explainer = shap.TreeExplainer(model_final)

# Paso 2: Calcular los valores SHAP para los datos de prueba (X_test)
```

```
shap_values = explainer.shap_values(X_test)

# Paso 3: Mostrar la gráfica de resumen de importancia de características
shap.summary_plot(shap_values, X_test, max_display=15)
plt.show()
```



Los valores SHAP proporcionan una explicación individualizada del impacto de cada característica en las predicciones del modelo, considerando la interacción con otras características. Es normal que las clasificaciones y magnitudes de importancia varíen entre el Modelo que se desea explicar y SHAP, ya que utilizan criterios diferentes para evaluar la importancia y la contribución de las características. Lo imporante es que mientras que las importancias de características de un modelo de Black Box indican su relevancia según el criterio de construcción del modelo, los valores SHAP ofrecen una perspectiva más detallada y específica sobre cómo cada característica influye en las predicciones del modelo para cada instancia de datos.

[ ]:
```
# Opcional: Mostrar un gráfico de barras de importancia de características
shap_values = explainer(X_test)
plt.title('Importancia de Características')
shap.plots.bar(shap_values, max_display=15)
plt.show()
```



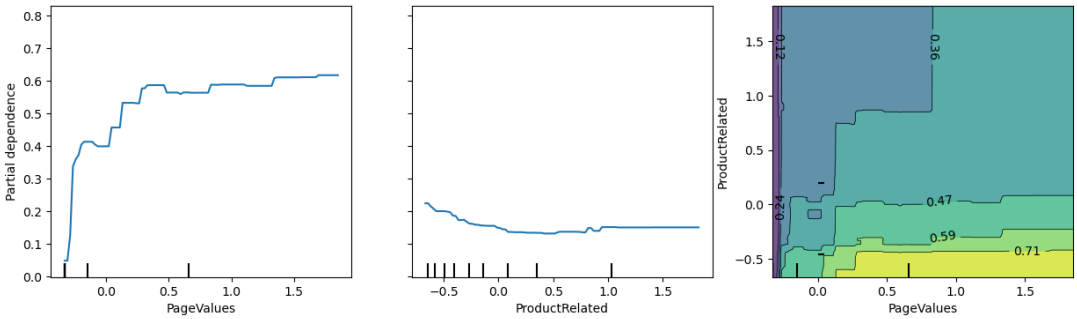
PDP

Los Partial Dependence Plots (PDPs) muestran cómo cambia la predicción del modelo en promedio cuando se varía una característica específica mientras se mantiene todo lo demás constante. Compararemos dos instancias diferentes para observar cómo afectan ciertas características a las predicciones del modelo

[ ]:
```
i=0
for col in X_test.columns:
    print(f'{col}: {i}')
    i=i+1
```

```
PageValues: 0
ProductRelated: 1
TrafficType: 2
ExitRates: 3
Month: 4
SpecialDay: 5
```

[ ]:
```
features = [0, 1, (0, 1)]
#PartialDependenceDisplay.from_estimator(model, X_train, features)

# Crear la visualización con un tamaño de figura personalizado
fig, ax = plt.subplots(figsize=(15, 4))
PartialDependenceDisplay.from_estimator(model_final, X_test, features, ax=ax)

plt.show()
```



Métodos Agnósticos de Modelos Locales

Diferencias Clave:

- Modelo Simple vs. Valor Esperado: LIME ajusta un modelo simple localmente para aproximar el modelo complejo, mientras que SHAP utiliza el valor esperado condicionalmente para determinar la contribución de cada característica.
- Interpretación Local vs. Global: LIME se centra en la interpretación local, explicando cómo cambia la predicción para una instancia particular. En cambio, SHAP proporciona una interpretación global, mostrando la importancia relativa de las características en promedio para todas las predicciones del modelo.

Selección del Método:

- LIME es útil cuando se necesita una interpretación local rápida y comprensible en un vecindario específico de la instancia de interés, especialmente cuando el modelo subyacente es complejo y no lineal.
- SHAP es preferible para obtener una comprensión global de cómo cada característica contribuye a las predicciones del modelo, proporcionando una visión más completa de las relaciones entre las características y las predicciones.

SHAP - local

SHAP (SHapley Additive exPlanations):

- Enfoque: SHAP aplica la teoría de juegos para asignar una contribución a cada característica en función de su valor esperado condicionalmente.
- Proceso: Evalúa todas las posibles combinaciones de características para calcular cómo contribuye cada característica al modelo en promedio.
- Explicación: Los valores SHAP representan la contribución media de cada característica a la predicción del modelo en una escala global.

[ ]:
```
# Crear el explainer utilizando TreeExplainer
explainer = shap.TreeExplainer(model_final)

shap_values = explainer.shap_values(X_test)
```

[ ]:
```
# Obtener tres muestras aleatorias de cada grupo
tp_samples = true_positives.sample(n=5, random_state=42)
tn_samples = true_negatives.sample(n=5, random_state=42)
fp_samples = false_positives.sample(n=5, random_state=42)
fn_samples = false_negatives.sample(n=5, random_state=42)
```

[ ]:
```
i=3
shap.initjs()
shap.force_plot(explainer.expected_value, shap_values[fp_samples['Index'].
 ↪iloc[i]], X_test.iloc[fp_samples['Index'].iloc[i]])
```

```
<IPython.core.display.HTML object>
```

[]:
```
<shap.plots._force.AdditiveForceVisualizer at 0x14502a532d0>
```

LIME

LIME (Local Interpretable Model-agnostic Explanations):

- Enfoque: LIME construye un modelo interpretable local (generalmente lineal) en el vecindario de la instancia de interés.
- Proceso: Perturba el conjunto de datos alrededor de la instancia seleccionada y observa cómo cambian las predicciones del modelo complejo subyacente.
- Explicación: Los pesos obtenidos son coeficientes que indican la contribución relativa de cada característica en el cambio de la predicción localmente.

LIME necesita entender la distribución de los datos para generar perturbaciones realistas y obtener una explicación local precisa. Para lo cual usa los siguientes pasos:

- Definir el explainer de LIME con X_train:
- Seleccionar una instancia de X_test para explicar:
- Generar y mostrar la explicación:

[ ]:
```
# Crear un explainer de LIME
# Crear un explainer de LIME usando los datos de entrenamiento
explainer = lime.lime_tabular.LimeTabularExplainer(
    X_train.values,
    feature_names=X_train.columns,
    class_names=['No Revenue', 'Revenue'],
    discretize_continuous=True  # Optional: discretize continuous features
)
```

LIME utiliza un modelo interpretable más sencillo, típicamente lineal, para aproximar el comportamiento del modelo complejo subyacente en un vecindario local alrededor de la instancia de interés. Los pesos obtenidos representan coeficientes relativos que explican cómo las predicciones podrían cambiar si las características experimentaran pequeñas variaciones alrededor de sus valores actuales.

[ ]:
```
instance_index = tp_samples['Index'].iloc[0]  # Índice de la instancia a␣
↪explicar

# Obtener una explicación local para la instancia seleccionada
exp = explainer.explain_instance(X_test.iloc[instance_index].values,␣
↪model_final.predict_proba)

# Imprimir el tipo de modelo complejo subyacente utilizado por LIME
print("Modelo complejo subyacente utilizado por LIME:", type(model_final))

# Mostrar la instancia y las predicciones
print("Instancia de prueba:")
print(X_test.iloc[instance_index])
print("\nPredicciones:")
print("Probabilidad de 'No Revenue':", exp.predict_proba[0])
print("Probabilidad de 'Revenue':", exp.predict_proba[1])

# Acceder a las explicaciones locales por característica
print("\nExplicaciones locales por característica:")
print(exp.local_exp)

# Imprimir el modelo interpretable local entrenado por LIME
print("\nModelo interpretable local entrenado por LIME:")
for feature, weight in exp.as_list():
    print(f"{feature}: {weight}")

# Acceder a las etiquetas principales si existen
if exp.top_labels is not None:
    print("\nEtiquetas principales:")
    print(exp.top_labels[0])  # La etiqueta más probable

# Mostrar la explicación en el notebook
exp.show_in_notebook(show_table=True)
```

```
Modelo complejo subyacente utilizado por LIME: <class
'xgboost.sklearn.XGBClassifier'>
Instancia de prueba:
PageValues         0.229026
ProductRelated     7.167920
TrafficType        0.219550
ExitRates         -0.743076
Month              0.996580
SpecialDay        -0.307794
Name: 566, dtype: float64
```

```
Predicciones:
Probabilidad de 'No Revenue': 0.25863034
Probabilidad de 'Revenue': 0.74136966

Explicaciones locales por característica:
{1: [(1, -0.23641843519636713), (0, 0.1833170008959621), (3,
0.1801446398695117), (2, 0.04174547710425903), (5, 0.0043659530898985275), (4,
0.0019321389585620166)]}

Modelo interpretable local entrenado por LIME:
ProductRelated > 0.19: -0.23641843519636713
PageValues > -0.33: 0.1833170008959621
ExitRates <= -0.60: 0.1801446398695117
0.15 < TrafficType <= 0.22: 0.04174547710425903
SpecialDay <= -0.31: 0.0043659530898985275
0.11 < Month <= 1.00: 0.0019321389585620166

<IPython.core.display.HTML object>
```

0.0.1 Arbol Surrogado de vlaores SHAP

[ ]:
```
from sklearn.tree import DecisionTreeClassifier

# Definir el máximo de niveles del árbol
max_depth = 4  # Aquí puedes ajustar el valor según lo que consideres␣
 ↪adecuado

# Inicializar un árbol de decisión clasificador con máximo de niveles␣
 ↪definido
surrogate_tree = DecisionTreeClassifier(max_depth=max_depth)

# Extraer los valores SHAP desde el objeto Explanation
shap_values_extracted = shap_values.values  # Extraer solo los valores SHAP␣
 ↪(shape: [n_samples, n_features])

# Ajustar el árbol subrogado utilizando solo los valores SHAP como␣
 ↪características de entrada
surrogate_tree.fit(shap_values_extracted, y_pred)
```

[ ]:
```
# Predecir las etiquetas utilizando el árbol surrogado
y_predTree = surrogate_tree.predict(shap_values_extracted)

# Evaluar el rendimiento del árbol surrogado
accuracy = accuracy_score(y_pred, y_predTree)
auc_roc = roc_auc_score(y_pred, y_predTree)
```

```
report = classification_report(y_pred, y_predTree)

print("Accuracy:", accuracy)
print("ROC AUC:", auc_roc)
print("Classification Report:\n", report)
```

```
Accuracy: 0.9848548590660496
ROC AUC: 0.9868285157225901
Classification Report:
                precision    recall  f1-score   support

           0       1.00      0.98      0.99      1986
           1       0.92      0.99      0.96       391

    accuracy                           0.98      2377
   macro avg       0.96      0.99      0.97      2377
weighted avg       0.99      0.98      0.99      2377
```

[ ]:
```
# Obtener los nombres de las características utilizadas en el modelo de árbol
feature_names = X_test.columns.tolist() + ['Prediction']

#  Visualizar el árbol subrogado
plt.figure(figsize=(14, 6))
plot_tree(surrogate_tree, feature_names=feature_names, filled=True,␣
 ↪rounded=True, fontsize=8)
plt.savefig('arbol_subrogado.png', dpi=300)  # Guardar la figura en formato␣
 ↪PNG con 300 dpi de resolución
plt.show()
```