



UniversidadeVigo

Trabajo Fin de Máster

---

# Desarrollo de un Agente para tareas-problemas de DSc específico

---

Abel Pastoriza Vilas

Máster en Técnicas Estadísticas

Curso 2025-2026



## Propuesta de Trabajo Fin de Máster

<b>Título en galego:</b> Desenvolvemento dun axente para tarefas-problemas de DSc específico
<b>Título en español:</b> Desarrollo de un agente para tareas-problemas de DSc específico
<b>English title:</b> Development of an agent for specific DSc tasks-problems
<b>Modalidad:</b> Modalidad B
<b>Autor/a:</b> Abel Pastoriza Vilas, Universidad de Vigo
<b>Director/a:</b> Beatriz Pateiro López , Universidad de Santiago de Compostela
<b>Tutor/a:</b> Ángel Rodríguez Chicote, SDG group; Miguel Arias Benítez, SDG group; Carlos Ortega Fernández, SDG group
<p><b>Breve resumen del trabajo:</b></p> <p>El TFM consiste en entrenar un agente (integrado en nuestra suite GENA mediante asistente) que le permita aprender a desarrollar una tarea concreta dentro del ámbito del <i>Machine Learning</i> seleccionada durante el desarrollo del TFM (auditoría de datos, preparación de datos...) o bien solventar un problema o caso de uso concreto (abandono de clientes, venta cruzada, mantenimiento predictivo...). El objetivo del TFM es valorar la aplicación de este tipo de técnicas de manera alternativa a los métodos clásicos 100 % desarrollados por trabajo humano y poder analizar las eficiencias generadas a través de la comparativa de calidad del resultado obtenido versus el esfuerzo dedicado.</p>
<p><b>Recomendaciones:</b> Conocimientos de programación en Python y formación académica en estadística.</p>
<p><b>Otras observaciones:</b></p>



Doña Beatriz Pateiro López , profesora titular de la Universidad de Santiago de Compostela, don Ángel Rodríguez Chicote, senior consultant de SDG group, don Miguel Arias Benítez, specialist de SDG group y don Carlos Ortega Fernández, specialist lead de SDG group, informan que el Trabajo Fin de Máster titulado

### Desarrollo de un Agente para tareas-problemas de DSc específico

fue realizado bajo su dirección por don Abel Pastoriza Vilas para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal. Además, doña Beatriz Pateiro López y don Abel Pastoriza Vilas

sí                       no

autorizan a la publicación de la memoria en el repositorio de acceso público asociado al Máster en Técnicas Estadísticas.

En Vigo, a 28 de mayo de 2026.

La directora:  
Doña Beatriz Pateiro López

El tutor:  
Don Miguel Arias Benítez

El autor:  
Don Abel Pastoriza Vilas

El tutor:  
Don Ángel Rodríguez Chicote  
*Angel Rodríguez Chicote*  
El tutor:  
Don Carlos Ortega Fernández

**Declaración responsable.** Para dar cumplimiento a la Ley 3/2022, de 24 de febrero, de convivencia universitaria, referente al plagio en el Trabajo Fin de Máster (Artículo 11, [Disposición 2978 del BOE núm. 48 de 2022](#)), **el/la autor/a declara** que el Trabajo Fin de Máster presentado es un documento original en el que se han tenido en cuenta las siguientes consideraciones relativas al uso de material de apoyo desarrollado por otros/as autores/as:

- Todas las fuentes usadas para la elaboración de este trabajo han sido citadas convenientemente (libros, artículos, apuntes de profesorado, páginas web, programas,...)
- Cualquier contenido copiado o traducido textualmente se ha puesto entre comillas, citando su procedencia.
- Se ha hecho constar explícitamente cuando un capítulo, sección, demostración,... sea una adaptación casi literal de alguna fuente existente.

Y, acepta que, si se demostrara lo contrario, se le apliquen las medidas disciplinarias que correspondan.



# Índice general

<b>Resumen</b>	<b>IX</b>
<b>Introducción</b>	<b>XI</b>
<b>1. Metodología</b>	<b>1</b>
1.1. Modelo CRISP-DM	1
1.2. Estudio de un caso real: <i>churn</i>	3
1.3. Modelos de clasificación	5
1.3.1. Descripción de modelos	5
1.3.2. Métricas de evaluación	11
<b>2. Agentes de IA</b>	<b>15</b>
2.1. Modelos de lenguaje (LLM)	15
2.2. Definición agentes de IA	16
2.3. Tipos de agentes	16
2.4. Flujo de pensamiento de un agente de IA	17
2.5. Implementación de un agente: ejemplo	17
2.6. Librerías y frameworks comunes	19
2.6.1. LangChain	20
2.6.2. LangGraph	20
2.6.3. AzureChatOpenAI	22
2.6.4. Streamlit	22
2.6.5. Logfire	22
2.7. Tipos de memoria	23
2.8. Métodos de evaluación	23
2.8.1. Evaluación empírica	23
2.8.2. Evaluación cualitativa	23
2.9. Objetivos de mejora a largo plazo	24
2.10. Marco tecnológico	24
2.10.1. Aplicaciones	25
2.10.2. Agentes	28
<b>3. Uso del sistema de agentes: <i>churn</i></b>	<b>29</b>
3.1. Análisis exploratorio	29
3.2. Análisis previos	32
3.3. Correlation funnel	38
3.4. Ajuste de un modelo de clasificación	38
3.4.1. Procesado de datos	39
3.4.2. Discusión de resultados	39

3.4.3. Modelo Gradient Boosting . . . . .	42
3.4.4. Modelo de regresión logística . . . . .	45
3.4.5. Modelo StackedEnsemble_BestOfFamily . . . . .	46
<b>4. Observaciones y conclusiones finales</b> . . . . .	<b>51</b>
4.1. Persistencia del contexto . . . . .	51
4.2. Eficiencia computacional en la generación de gráficos . . . . .	51
4.3. Precisión conceptual: histograma y gráfico de densidad . . . . .	51
4.4. Consistencia . . . . .	52
4.5. Dificultad para trabajar con varios conjuntos de datos . . . . .	52
4.6. Omisión del análisis de colinealidad . . . . .	53
4.7. Falsa sensación de corrección y falta de autonomía . . . . .	53
4.8. Conclusión . . . . .	53
<b>A. Validación análisis exploratorio</b> . . . . .	<b>55</b>

# Resumen

## Resumen en español

Este trabajo estudia la utilidad de un sistema de agentes de IA para automatizar, mediante conversación en lenguaje natural, las tareas habituales de un proyecto de ciencia de datos. Dicho sistema se basa en AI Data Science Team, una librería de Python de código abierto publicada en GitHub por el desarrollador estadounidense Matt Dancho.

Como caso de uso, se analiza en qué medida el uso de agentes de IA facilita la aplicación práctica de la denominada metodología CRISP-DM, un marco de trabajo que organiza los proyectos de ciencia de datos en una serie de fases estructuradas. La elección de esta metodología se justifica, además, por ser la empleada en SDG Group, empresa en la que se realizaron las prácticas externas y que sirvió como contexto profesional de referencia para el desarrollo del trabajo. Para ello, se emplea un conjunto de datos real de abandono de clientes (*churn*) de una compañía de telefonía. El objetivo del caso de uso es predecir qué clientes dejarán de utilizar el servicio, permitiendo así que la empresa actúe de forma preventiva antes de que se produzca la baja.

Siguiendo las fases de CRISP-DM, el trabajo aborda el entendimiento del negocio, la exploración y limpieza de datos, la preparación del conjunto de datos mediante la codificación de variables categóricas e ingeniería de características, así como el modelado predictivo.

En la fase de modelado se entrenan y comparan varios modelos predictivos con el objetivo de identificar qué clientes tienen mayor probabilidad de abandonar el servicio. Para ello, se utiliza una herramienta de aprendizaje automático automatizado que permite probar distintos algoritmos y seleccionar el más adecuado según su capacidad predictiva. Esta fase se aprovecha, además, para revisar los fundamentos teóricos de algunos de los modelos empleados, así como las métricas utilizadas para evaluar su rendimiento. El modelo finalmente elegido ofrece un buen equilibrio entre precisión e interpretabilidad. Además, el análisis de sus resultados permite identificar los factores que más influyen en el abandono: la antigüedad del cliente y los contratos de larga duración reducen el riesgo de fuga, mientras que el uso de fibra óptica, los contratos mensuales y el pago mediante cheque electrónico son los principales factores de riesgo.

El trabajo concluye valorando en qué medida el sistema de agentes de IA reduce el esfuerzo técnico y acelera la toma de decisiones en un proyecto real de analítica avanzada.

## English abstract

This paper studies the usefulness of an Artificial Intelligence agent system for automating, through natural language conversation, the routine tasks of a data science project. This system is based on AI Data Science Team, an open-source Python library published on GitHub by the American developer Matt Dancho.

As a use case, the paper analyzes how the use of Artificial Intelligence agents facilitates the practical application of the CRISP-DM methodology, a framework that organizes data science projects into a series of structured phases. The choice of this methodology is further justified by its use at SDG Group, the company where the external internship was carried out and which served as a professional reference context for the development of this work. For this purpose, a real customer churn dataset from a telecommunications company is used. The objective of the use case is to predict which customers will stop using the service, thus allowing the company to act preventively before the cancellation occurs.

Following the CRISP-DM phases, this work addresses business understanding, data exploration and cleaning, dataset preparation through categorical variable coding and feature engineering, and predictive modeling.

In the modeling phase, several predictive models are trained and compared to identify which customers are most likely to churn. An automated Machine Learning tool is used to test different algorithms and select the most suitable one based on its predictive capacity. This phase is also used to review the theoretical foundations of some of the models employed, as well as the metrics used to evaluate their performance. The model ultimately chosen offers a good balance between accuracy and interpretability. Furthermore, the analysis of its results identifies the factors that most influence churn: customer tenure and long-term contracts reduce the risk of churn, while the use of fiber optics, monthly contracts, and payment by electronic check are the main risk factors.

The work concludes by assessing the extent to which the Artificial Intelligence agent system reduces technical effort and accelerates decision-making in a real advanced analytics project.

# Introducción

La Inteligencia Artificial (IA) es una tecnología que ha experimentado un crecimiento exponencial en los últimos años, revolucionando numerosos ámbitos del conocimiento humano. Su desarrollo ha pasado por distintas etapas: desde los primeros enfoques basados en reglas y sistemas expertos, hasta los avances en aprendizaje automático y, más recientemente, en modelos de lenguaje de gran capacidad. Esta evolución se ha visto acelerada por la disponibilidad de grandes volúmenes de datos, el aumento de la capacidad computacional y la madurez de nuevas arquitecturas, situando a la IA como una tecnología transversal en contextos académicos y profesionales.

Uno de los campos donde este impacto resulta más visible es la ciencia de datos (*data science*, DSc). En este contexto, la IA no solo permite construir modelos predictivos más robustos, sino también asistir en tareas recurrentes de un proyecto analítico: exploración inicial, limpieza y transformación de datos, generación de visualizaciones, comparación de modelos y comunicación de resultados. Esta capacidad de apoyo resulta especialmente valiosa porque los proyectos de ciencia de datos combinan etapas técnicas heterogéneas, exigen iteraciones frecuentes y requieren coordinación entre perfiles con distinto nivel de especialización.

En paralelo, han adquirido relevancia los agentes de IA, entendidos como sistemas capaces de interpretar instrucciones en lenguaje natural, planificar acciones, invocar herramientas y encadenar subtareas de manera semiautónoma. Su interés en ciencia de datos radica en que pueden actuar como interfaz operativa entre el analista y los procesos técnicos, agilizando tareas repetitivas y facilitando que parte del flujo de trabajo se ejecute de forma asistida, trazable y reproducible.

Esta línea de trabajo cobra especial sentido al considerar algunas limitaciones de los flujos tradicionales de analítica. En muchos entornos, el proceso sigue siendo intensivo en trabajo manual, con dependencia de conocimiento técnico avanzado, cambios frecuentes de contexto entre herramientas y riesgo de inconsistencias entre etapas. Además, la repetición de operaciones similares en distintos proyectos puede incrementar los tiempos de desarrollo y dificultar la estandarización de buenas prácticas. Ante este escenario, resulta pertinente estudiar en qué medida los agentes de IA pueden complementar, y en ciertos casos optimizar, la ejecución de procesos basados en metodologías consolidadas como CRISP-DM.

Este trabajo toma como base AI Data Science Team [1], una librería de Python de código abierto que integra un conjunto de agentes orientados a tareas habituales del ciclo analítico, como la preparación y visualización de datos o el apoyo al modelado predictivo. La elección de esta herramienta responde a una necesidad concreta: analizar, en un caso real de negocio, hasta qué punto un sistema de agentes de IA guiado mediante conversación en lenguaje natural puede facilitar la ejecución de las etapas de un proyecto de analítica avanzada estructurado según CRISP-DM, reduciendo esfuerzo técnico y mejorando la agilidad en la toma de decisiones.

La memoria se organiza en cuatro capítulos principales. En el Capítulo 1 se presenta el marco metodológico del trabajo, centrado en la metodología CRISP-DM, el caso de estudio de abandono de clientes (*churn*) y los modelos de clasificación utilizados, junto con las métricas necesarias para evaluar su rendimiento.

El Capítulo 2 introduce los fundamentos de los agentes de IA, incluyendo los modelos de lenguaje, los tipos de agentes, sus mecanismos de memoria, las principales herramientas y entornos tecnológicos empleados, así como el marco tecnológico sobre el que se apoya el sistema analizado.

El Capítulo 3 desarrolla el caso práctico de uso del sistema de agentes sobre el conjunto de datos de *churn*.

Finalmente, el Capítulo 4 recoge las principales conclusiones del trabajo, incluyendo aspectos relacionados con la eficiencia computacional y la precisión de las respuestas generadas por el sistema.

# Capítulo 1

## Metodología

En este capítulo se describe el marco de trabajo seguido para desarrollar el caso de estudio, desde la metodología CRISP-DM hasta el tratamiento de los datos y la selección de los modelos de clasificación empleados.

### 1.1. Modelo CRISP-DM

En ámbitos profesionales, el acceso a un gran volumen de datos no garantiza el éxito de proyectos de DSc. Una mala organización o una falta de estructura puede desembocar en la pérdida de recursos y fiabilidad en las conclusiones de un proyecto.

A continuación, se describe el modelo CRISP-DM (*Cross-Industry Standard Process for Data Mining*, [2]), una metodología ampliamente utilizada en proyectos de minería de datos y ciencia de datos. Este modelo estructura el desarrollo de los proyectos de análisis de datos de forma sistemática, desde la comprensión del negocio hasta la evaluación y despliegue de los resultados. Su elección como marco metodológico se justifica, además, por ser la metodología empleada en SDG Group. Cuenta con las siguientes fases:

1. Entendimiento del negocio.
2. Entendimiento de datos.
3. Preparación de datos.
4. Modelado analítico.
5. Evaluación e informes.
6. Despliegue.

A continuación se incluye una breve descripción de cada una de ellas.

#### **Entendimiento del negocio**

Se comienza por definir con precisión los objetivos del proyecto desde un punto de vista empresarial. Esto incluye:

- Identificar objetivos y oportunidades interesantes para la empresa y sus clientes. Se define el criterio

de éxito.

- Analizar la disponibilidad de recursos para realizar el proyecto. Estudiar el riesgo y el beneficio de este.
- Especificar las metas y objetivos en términos de análisis de datos.
- Realizar un plan para el proyecto distribuyendo los recursos y herramientas necesarias para cumplir cada una de sus fases.

## Entendimiento de datos

Esta fase se centra en la recopilación y el procesamiento inicial de los datos disponibles para el desarrollo del proyecto. Se busca:

- Explorar los datos con el objetivo de comprobar si la información obtenida se ajusta a lo esperado. También visualizar e identificar relaciones entre variables.
- Verificar la calidad de los datos.
- Descartar datos innecesarios o redundantes.

## Preparación de datos

Se ordenan, limpian y preparan los datos para su modelado y análisis posterior. Se distinguen los siguientes pasos principales:

- Seleccionar los datos que se van a utilizar en el análisis, integrándolos en una única tabla. Justificar la inclusión y exclusión de los distintos conjuntos de datos en el modelo.
- Limpiar los datos. Revisar errores, estudiar atípicos, comprobar el origen de datos faltantes, etc.
- Crear nuevas variables combinando o transformando las existentes con el fin de resaltar patrones relevantes que faciliten al modelo la predicción de los resultados.
- Unir y combinar datos con distintos orígenes en la tabla final.
- Dar el formato correcto a las variables.

## Modelado analítico

Una vez preparados los datos, se procede a su modelado con el objetivo de extraer conclusiones. Los pasos a seguir son:

- Seleccionar las técnicas de modelado y *Machine Learning* que se van a aplicar junto a una métrica de rendimiento para evaluar las predicciones.
- Generar el diseño de prueba. Esto incluye dividir la muestra en los grupos de entrenamiento, test y validación.
- Construir los primeros modelos y optimización de los parámetros, iterando para conseguir el mejor modelo posible.
- Evaluar los modelos a partir de la métrica de rendimiento que se fijó en el primer paso.

- Seleccionar los mejores modelos y descartar el resto.

## Evaluación e informes

Una vez desarrollados los modelos entran en fase de evaluación con el objetivo de asegurarse de que cumplen los objetivos de negocio. Esta fase se centra en:

- Verificar la capacidad y robustez de los modelos y estudiar la interpretabilidad de sus variables y parámetros asegurándose de que cumplan los requisitos a nivel empresa.
- Seleccionar el modelo final basándonos en su rendimiento, coste computacional, complejidad e interpretabilidad.
- Documentar los pasos realizados en la confección de los modelos y destacar los resultados más relevantes.
- Encontrar relación del modelo con aspectos de negocio y estudiar el coste/beneficio del despliegue del proyecto.

## Despliegue

La fase final consiste en implementar los modelos y soluciones en un entorno operativo. Esto puede incluir:

- Planificar la implementación del modelo. Identificar problemas y generar planes para solucionarlos.
- Diseñar informes para la supervisión y mantenimiento de los modelos.
- Elaborar el informe final. Incluir problemas y herramientas de visualización para los clientes.
- Análisis final de la calidad tanto del proyecto en general como de su realización, con objetivo de mejorar a futuro.

## 1.2. Estudio de un caso real: *churn*

Se dispone de un conjunto de datos *churn* de una compañía de telefonía móvil. En el contexto empresarial, el término *churn* se emplea para describir el abandono o pérdida de clientes de un producto o servicio. A nivel de datos, es una variable binaria que toma valor 1 cuando el cliente abandona y 0 en caso contrario.

Es un problema habitual en el ámbito de negocio para empresas de suscripción o servicios periódicos, ya que retener un cliente existente suele ser más económico que conseguir uno nuevo. También ayuda a la empresa a identificar en qué áreas debe invertir más o menos para aumentar la satisfacción del cliente.

El objetivo principal es construir un modelo de predicción del *churn* a partir del resto de variables disponibles en el conjunto de datos, empleando un agente de IA. Una vez descritas las fases generales de la metodología CRISP-DM, a continuación se particulariza su aplicación al caso práctico analizado en este trabajo.

### Entendimiento de negocio

El objetivo principal es predecir si un cliente abandonará el servicio con la finalidad de retener clientela.

En esta etapa se plantean cuestiones básicas:

- Características comunes de los clientes que abandonan.

- Balance entre el coste de la retención de clientes contra el beneficio de mantenerlos.

También, podemos agrupar las variables explicativas dependiendo de sus características. Por ejemplo por cuestiones demográficas (género, edad, soltero/casado...), datos sobre el contrato (tipo de servicio, método de pago...), variables numéricas (antigüedad del cliente, cargos mensuales...) y la variable respuesta *churn*.

## Entendimiento de datos

Dentro del conjunto de datos se analizan datos faltantes, correlaciones entre variables, se identifican atípicos y se estudia la distribución de las variables, principalmente la de la respuesta, ya que en muchos modelos de clasificación es importante diferenciar si estamos ante grupos desbalanceados o no.

En esta etapa también se analiza la relevancia de las variables dentro del modelo predictivo y qué pasos se deben realizar en el conjunto de datos para la implementación del modelo.

## Preparación de datos

En esta etapa se realizan todas las acciones estudiadas en el anterior paso:

- Eliminar las observaciones con valores nulos y atípicos.
- En caso de haber dos o más variables correlacionadas tomar solamente una de ellas o crear una nueva variable que las relacione (por ejemplo un ratio  $v1/v2$ ).
- Ajustar el formato contextual de las variables.

En general, comprende cualquier cambio necesario en la estructura de los datos. Es en este paso donde se particionan las muestras de entrenamiento y test. Realizar esta partición en las observaciones es necesario para poder construir el modelo y posteriormente evaluarlo. La muestra de entrenamiento sirve para que el modelo aprenda patrones de los datos. La muestra test sirve para comparar los resultados que obtenemos del modelo con los reales evaluando así el rendimiento.

## Modelado analítico

Con los datos ya preparados, se procede al ajuste del modelo predictivo. En problemas de predicción de *churn*, suelen emplearse modelos de clasificación como árboles de decisión, regresión logística, métodos de *boosting*, *random forest* y técnicas de *bagging*.

El objetivo principal de esta etapa es el diseño de todos estos modelos ajustando sus hiperparámetros requeridos.

## Evaluación e informes

Una vez desarrollados los modelos, se procede a su análisis y comparación. En primer lugar, se define una métrica de evaluación de los modelos, como el F1-score, la curva ROC o el AUC, entre otras. También es importante estudiar la interpretabilidad del modelo y el coste asociado a su implementación.

También se estudian los umbrales de riesgo en el *churn* y se elaboran las medidas necesarias para cada caso. Estas medidas tienen en cuenta el balance entre el coste de retención y beneficio por conservar el cliente. Se agrupan y redactan en informes y manuales para su clara implementación y aplicación en el siguiente paso.

## Despliegue

Se integra el modelo en un entorno de producción. Para cada cliente de la telefonía móvil se le asigna una probabilidad de *churn*. Si esta probabilidad supera cierto umbral se tomarán las medidas necesarias y justas para su retención (definidas en el punto anterior).

Gracias a nuevas observaciones se realiza un estudio en tiempo real del rendimiento del modelo. Con esto se puede volver a entrenar y perfeccionar periódicamente en caso de ser necesario.

## 1.3. Modelos de clasificación

Como se ha mencionado previamente, en un caso de estudio de *churn* suelen emplearse distintos modelos de clasificación. A continuación, se describen brevemente las principales cuestiones metodológicas asociadas a algunos de los modelos más habituales en este contexto.

### 1.3.1. Descripción de modelos

Los modelos de clasificación son procesos de aprendizaje supervisados donde se designan etiquetas a observaciones en función de sus características. En el problema de *churn*, las etiquetas son “Yes” y “No” (en algunos casos se toman los valores 1 y 0), dependiendo de si el cliente abandona o no el servicio. Esta es la variable respuesta y a partir del resto de variables predictoras se intentará estimar o predecir su valor.

Este problema se puede abordar desde dos puntos de vista:

Se puede realizar un modelo de clasificación. Este busca predecir si el cliente abandonará el servicio en un periodo no especificado pero que se supone a corto plazo. Estos modelos se centran en buscar patrones en las variables predictoras con el objetivo de prevenir un abandono del cliente.

Puede abordarse también desde el análisis de supervivencia, especialmente cuando se dispone de datos restringidos o censurados. En este enfoque, el interés no se centra únicamente en predecir si un cliente abandonará el servicio, sino en modelizar el tiempo hasta que ocurre el abandono. En el análisis de supervivencia, las observaciones se tratan como censuradas, lo que permite aprovechar toda la información disponible sin introducir sesgos. De este modo, el análisis de supervivencia proporciona una perspectiva más rica del problema del *churn*, al incorporar explícitamente la dimensión temporal.

Este trabajo se centra en la primera de las opciones. A continuación se presentan metodologías habituales para abordar este problema:

## Regresión logística

La regresión logística se centra en la estimación de la probabilidad de que una observación pertenezca a una clase a partir de los valores de variables predictoras; para una exposición detallada de este método puede consultarse [3]. Es decir, nuestro objetivo es construir un modelo para la probabilidad

$$\pi(x) = \mathbb{P}(Y = 1|X = x)$$

donde  $Y$  es una variable respuesta binaria (toma valores 0 y 1) y  $X$  es un vector que contiene las variables predictoras del modelo. Lo más habitual es considerar un modelo lineal que a través de una función link  $g$  se relacione con la probabilidad  $\pi(x)$  restringiéndose al intervalo  $[0,1]$ . Gracias a su comportamiento suave y natural es común considerar la función logit:

$$g(x) = \log\left(\frac{x}{1-x}\right) \text{ para todo } x \in [0, 1].$$

Así, podemos definir el modelo logístico como

$$g(\pi(x)) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

siendo  $p$  el número de variables predictoras. Los coeficientes  $\beta_j$  para  $j = \{0, 1, \dots, p\}$  son habitualmente calculados por el método de máxima verosimilitud.

Dado que la variable respuesta  $Y$  sigue una distribución de Bernoulli,  $Y \sim \text{Bernoulli}(\pi(x))$ , la probabilidad de observar un valor específico  $y_i$  para una observación  $i$  viene dada por  $\pi(x_i)^{y_i} (1 - \pi(x_i))^{1-y_i}$ . Asumiendo independencia entre las  $n$  observaciones de la muestra, podemos construir la función de verosimilitud como el producto de estas probabilidades:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^n \pi(x_i)^{y_i} (1 - \pi(x_i))^{1-y_i}.$$

Para facilitar la maximización y el cálculo de las derivadas parciales respecto a los parámetros  $\boldsymbol{\beta}$ , habitualmente se trabaja con el logaritmo de esta función, conocido como Log-Likelihood:

$$l(\boldsymbol{\beta}) = \sum_{i=1}^n [y_i \log(\pi(x_i)) + (1 - y_i) \log(1 - \pi(x_i))].$$

El vector de estimadores  $\hat{\boldsymbol{\beta}}$  será aquel que maximice esta función. Al no existir una solución cerrada (a diferencia de la regresión lineal por mínimos cuadrados), estos valores se obtienen mediante algoritmos iterativos como el método de Newton-Raphson.

Una ventaja técnica del uso de la función link logit reside en la interpretación de los parámetros a través de los *Odds*. Definiendo los *Odds* como el cociente entre la probabilidad de éxito y la de fracaso:

$$\text{Odds}(x) = \frac{\pi(x)}{1 - \pi(x)} = e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p}.$$

Esta expresión nos permite interpretar el coeficiente  $\beta_j$  mediante el *Odds Ratio* (OR). Si incrementamos una variable predictora  $x_j$  en una unidad, manteniendo el resto constantes, el OR viene dado por:

$$\text{OR} = \frac{\text{Odds}(x_j + 1)}{\text{Odds}(x_j)} = e^{\beta_j}.$$

Esto implica que  $e^{\beta_j}$  representa el factor multiplicativo por el que cambia la *Odds* de que ocurra el evento  $Y = 1$  por cada unidad que aumenta  $X_j$ .

La función inversa de la función link es

$$g^{-1}(x) = \frac{1}{1 + e^{-x}},$$

importante, ya que nos permite relacionar un valor de la recta real con la probabilidad estimada objetivo del modelo logístico:

$$\pi(x) = g^{-1}(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p)}}.$$

Así, una probabilidad mayor que 0.5 en una nueva observación del modelo se asociará al grupo  $Y = 1$  y, análogamente, una probabilidad menor a 0.5 con  $Y = 0$ .

## Árboles de decisión

Los árboles de decisión son modelos simples con una interpretabilidad sencilla e intuitiva. Su capacidad predictiva no es la más elevada, pero sirve de base a otros modelos que se explicarán a continuación como el *bagging* o el *random forest*.

La idea principal de un árbol de decisión es la partición del espacio predictor en regiones simples con el objetivo de representar el modelo como un árbol binario. El proceso comienza con un nodo raíz que contiene toda la muestra de entrenamiento y, en cada iteración, los nodos se dividen en dos subconjuntos según el valor de alguna de las variables predictoras. Tras un número finito de particiones se obtienen los nodos terminales. Para más detalle consultar [4].

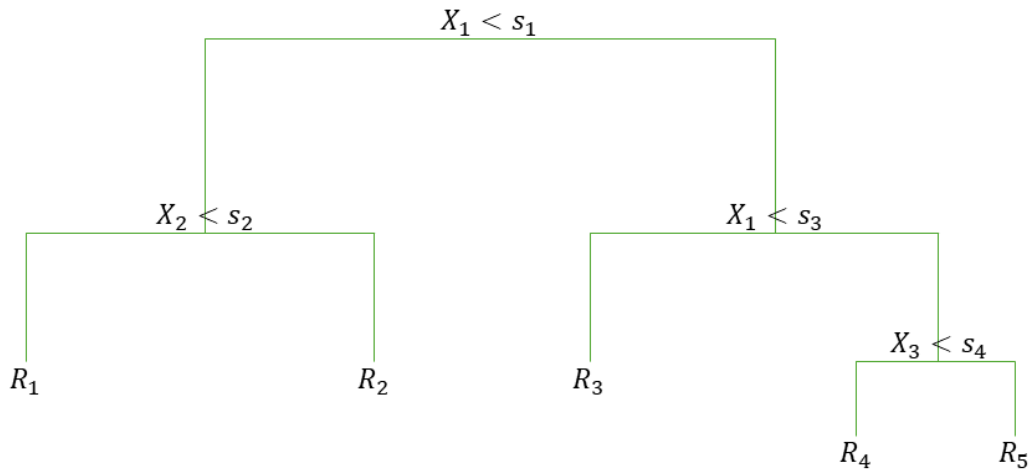


Figura 1.1: Árbol de decisión

En la Figura 1.1 se observa un ejemplo de un árbol de decisión con cinco nodos terminales.

Existen dos tipos de árboles de decisión: de regresión y de clasificación. Dado el contexto del problema *churn*, nos centramos en los árboles de clasificación, por lo que a continuación se describen algunos de sus principales aspectos técnicos.

Como ya se mencionó, el modelo consiste en la partición del espacio predictor en regiones  $R_1, R_2, \dots, R_J$ . Para cada  $R_j$  con  $j = \{1, \dots, J\}$  se calcula la moda de la variable respuesta. Este valor es el utilizado en las predicciones de nuevas observaciones que pertenezcan a  $R_j$ . En el problema de *churn*, esto significa que cada región clasifica a los clientes según predomine la clase “abandona” o “no abandona”.  $X_1, X_2$  y  $X_3$  son tres variables predictoras de nuestro conjunto de datos; como podrían ser la antigüedad, los cargos mensuales y el número de servicios contratados por el cliente. Los valores  $s_1, s_2, s_3$  y  $s_4$  son los umbrales que se fijan en la construcción del árbol de decisión y que generan las particiones. Un ejemplo de partición de la Figura 1.1 sería  $R_1 = \{X : X_1 < s_1 \cap X_2 < s_2\}$  que podría representar a todos los clientes suscritos desde hace menos de  $s_1$  meses y que pagan mensualmente una cantidad menor a  $s_2$  euros.

Una de las grandes ventajas de esta metodología es su capacidad para incorporar simultáneamente predictores numéricos y categóricos en la construcción de las regiones. Para tratar las variables categóricas, es habitual recurrir a la creación de variables *dummy* (o codificación *one-hot*), transformando una variable cualitativa con  $L$  niveles en  $L - 1$  variables binarias. Esto permite al algoritmo tratar las decisiones categóricas con la misma lógica matemática que las numéricas. Si bien esta técnica preserva la interpretabilidad del modelo, conlleva el riesgo de aumentar considerablemente la dimensionalidad del espacio predictor si las variables originales poseen una alta cardinalidad.

El concepto del modelo es sencillo. El principal problema reside en el cálculo de las particiones. Para ello se necesita definir una medida de error con el objetivo de minimizarla al construir el árbol.

La configuración óptima de un árbol de clasificación se alcanza cuando las regiones  $R_j$  resultantes presentan una máxima homogeneidad, es decir, cuando todas las observaciones de entrenamiento dentro de una misma partición pertenecen a una misma categoría de la variable *churn*. Definimos entonces  $\hat{p}_0$  y  $\hat{p}_1$  como la proporción de observaciones de ceros y unos respectivamente en la región  $R_j$  fijada de antemano.

Entonces las medidas de error habituales son las siguientes:

- Proporción de errores de clasificación:

$$1 - \max_{k \in \{0,1\}} (\hat{p}_k)$$

- Índice de Gini:

$$\sum_{k=0}^1 \hat{p}_k (1 - \hat{p}_k)$$

- Entropía:

$$-\sum_{k=0}^1 \hat{p}_k \log(\hat{p}_k)$$

Estos índices sirven como medidas de impureza. Se utilizan para decidir cómo dividir los nodos durante la construcción del árbol y representan un error global al terminar el proceso.

Obsérvese que estos conceptos están definidos para únicamente dos grupos. Esto es así ya que nuestra variable respuesta *churn* es binaria, pero podría generalizarse para  $k \in \{0, 1, 2, \dots, K\}$  de forma análoga a como se describió previamente.

## Bagging

Su nombre resulta de la unión de los términos *bootstrap* y *aggregation* y forma parte de un método de aprendizaje denominado *ensemble* centrado en combinar predicciones de múltiples modelos individuales.

El *bootstrap* es una técnica de remuestreo estadístico que consiste en generar muestras a partir de un conjunto de datos original, de modo que cada muestra bootstrap tiene el mismo tamaño que el conjunto original.

Esta técnica es utilizada en múltiples ámbitos de la estadística entre los que destacan:

- Estimación de la distribución de un estadístico sin necesidad de hacer supuestos sobre la distribución de los datos.
- Reducción de la varianza en métodos de ensemble, generando múltiples modelos a partir de diferentes muestras *bootstrap*.
- Evaluación de errores y construcción de intervalos de confianza de manera no paramétrica.

Volviendo al *bagging* [5], se basa en la construcción de múltiples muestras de entrenamiento a partir de las cuales se ajustarán una gran cantidad de modelos de clasificación, por ejemplo árboles de decisión. Una vez realizado este proceso se combinan los resultados. Esto genera una reducción en la varianza del modelo.

Desde una perspectiva teórica, si asumimos que disponemos de  $B$  modelos independientes con varianza  $\sigma^2$ , promediar sus predicciones reduciría la varianza a  $\sigma^2/B$ . Sin embargo, dado que los modelos en *bagging* no son totalmente independientes pues comparten el conjunto de datos original, la reducción no es tan drástica, pero sigue siendo significativa. Por este motivo, el *bagging* es extremadamente eficaz con los denominados aprendices inestables como árboles de decisión profundos, donde pequeñas variaciones en los datos de entrenamiento provocan grandes cambios en la estructura del modelo. Por el contrario, no suele mejorar el rendimiento de aprendices estables o con alto sesgo como la regresión lineal.

En el caso de árboles de regresión, la predicción final se obtiene como la media de las predicciones individuales:

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(x) \quad (1.1)$$

Mientras que en árboles de clasificación se emplea el voto mayoritario, asignando a cada modelo el mismo peso. Además, la proporción de votos de cada clase puede interpretarse como una estimación de su probabilidad, lo cual suaviza las fronteras de decisión respecto a un único árbol.

La principal ventaja del *bagging* es la estimación del error de predicción mediante el error *out-of-bag* (OOB), sin necesidad de emplear un conjunto de test o validación cruzada. La validación cruzada es una técnica que evalúa un modelo dividiendo los datos en múltiples particiones para entrenarlo y probarlo repetidamente, obteniendo una estimación más fiable de su rendimiento. En el proceso *bootstrap* existen observaciones que no se emplean en la construcción de los modelos (aproximadamente un 36,8% de las observaciones originales quedan fuera en cada iteración). Estas se denominan datos *out-of-bag*. Es a partir de estos como se calcula el error OOB.

Este índice también es útil para analizar el número total de muestras bootstrap que se deben realizar. En la práctica se hace un estudio de la evolución del error a medida que se incrementa el número de muestras, buscando un balance entre reducción de la varianza del modelo y su coste computacional. Cabe destacar que, a diferencia de otros modelos que mencionaremos posteriormente como el *boosting*, el *bagging* permite la paralelización del entrenamiento, ya que cada modelo es independiente de los demás, lo que supone una ventaja técnica relevante en grandes volúmenes de datos.

Aunque el *bagging* mejora notablemente la precisión predictiva, reduce la interpretabilidad respecto a un único árbol de decisión. No obstante, es posible estimar la importancia de las variables analizando la reducción del error asociada a los cortes que utilizan cada predictor a lo largo del conjunto de árboles.

## Random Forest

Los bosques aleatorios o *random forests* son una evolución del *bagging*. Al contrario que el modelo anterior, que puede aplicarse a cualquier método de aprendizaje, este está específicamente diseñado para trabajar con árboles de decisión, introduciendo una capa adicional de aleatoriedad conocida como *feature bagging*.

Este modelo se desarrolla con el objetivo de evitar un problema en el *bagging* estándar: la correlación entre los árboles generados. Este fenómeno es habitual en casos en los que una variable predictora tiene gran influencia sobre la respuesta, es decir, es un predictor muy fuerte. Como consecuencia, dicha variable aparecerá en los nodos superiores de la mayoría de los árboles de decisión, haciendo que las predicciones de los distintos árboles estén fuertemente correlacionadas. Si los árboles son muy similares entre sí, promediarlos no reduce la varianza tanto como se desearía.

Para solventar este inconveniente, se modifica el algoritmo de construcción del árbol. Supóngase que una muestra de entrenamiento consta de  $p$  variables predictoras. En un árbol estándar, en cada nodo se evalúan

las  $p$  variables para encontrar el mejor corte. En *random forest*, se define un hiperparámetro  $m < p$ . En cada nodo del árbol, se selecciona un subconjunto aleatorio de  $m$  variables y el mejor corte se busca únicamente entre esas candidatas.

La elección de  $m$  es crítica para el balance del modelo. En [6] se sugieren las siguientes reglas empíricas por defecto:

- Para problemas de clasificación:  $m \approx \sqrt{p}$ .
- Para problemas de regresión:  $m \approx p/3$ .

Al forzar a los árboles a considerar predictores menos dominantes, se consigue que las estructuras de los árboles sean heterogéneas, logrando la deseada *decorrelación*.

Como resultado se obtiene una metodología que, aunque incrementa ligeramente el sesgo de los árboles individuales pues no deja usar siempre la mejor variable disponible, reduce drásticamente la varianza del conjunto. Además, genera un proceso más eficiente en términos computacionales, pues en cada nodo se evalúa una fracción menor de las variables.

El número de árboles que forma el bosque aleatorio  $B$  puede considerarse un hiperparámetro del modelo, aunque habitualmente se aborda como un problema de convergencia del error: se añaden árboles hasta que el error OOB se estabiliza. Asimismo, el tamaño mínimo de los nodos terminales actúa como un regularizador adicional. Nodos más grandes producen árboles más simples y menos correlacionados, mientras que nodos más pequeños permiten capturar relaciones más complejas.

## Boosting

El modelo *boosting* es otro método de ensemble que, a diferencia del *bagging* en el que los modelos se podían construir paralelamente, trabaja de forma secuencial. Su filosofía se basa en el uso de aprendices débiles: modelos sencillos que tienen una capacidad predictiva ligeramente superior al azar y poco peso computacional. El objetivo es combinarlos para crear un modelo robusto capaz de reducir tanto la varianza como, principalmente, el sesgo del modelo.

Existen múltiples variantes del *boosting*. Como ya se mencionó, el problema *churn* se centra en la clasificación de observaciones por lo que nos centraremos en el *AdaBoost* (*Adaptive Boosting*), el primer algoritmo exitoso de este tipo para clasificación binaria. Detallamos su funcionamiento técnico descrito en [7]:

Primero se inicializan los pesos de las observaciones  $w_i = 1/N$ , siendo  $N$  el número total de observaciones. Se decide el número de iteraciones  $B$  y la profundidad de los árboles. Habitualmente, en AdaBoost se utilizan árboles de profundidad  $d = 1$ , es decir, una única división de los datos.

En cada iteración  $b$  de 1 a  $B$ :

1. Se ajusta un clasificador débil utilizando los pesos actuales y se calcula su error ponderado  $\epsilon_b$  del clasificador:

$$\epsilon_b = \frac{\sum_{i=1}^N w_i |\hat{y}_b(x_i) - y_i|}{\sum_{i=1}^N w_i},$$

donde  $\hat{y}_b(x_i)$  es la predicción del clasificador débil en la iteración  $b$  e  $y_i$  es el valor real.

2. Se calcula la importancia del clasificador en la votación final, denotada como  $s_b$ :

$$s_b = \frac{1}{2} \ln \left( \frac{1 - \epsilon_b}{\epsilon_b} \right)$$

Nótese que si el error es bajo ( $\epsilon_b \rightarrow 0$ ),  $s_b$  crece exponencialmente, dotando de una gran relevancia a ese árbol. Si el error es 0.5 (azar), el peso es 0. El algoritmo exige que cada clasificador débil sea estrictamente mejor que el azar ( $\epsilon_b < 0,5$ ), lo que garantiza  $s_b > 0$  en todas las iteraciones. Si un clasificador no cumple esta condición, se descarta o el algoritmo se detiene.

3. Se actualizan los pesos únicamente de las observaciones mal clasificadas para la siguiente iteración:

$$w_i \leftarrow w_i \cdot \exp(s_b)$$

Esto provoca que las observaciones mal clasificadas aumenten su peso, obligando al siguiente árbol a centrarse específicamente en los casos que el anterior clasificó incorrectamente.

Las predicciones finales de este modelo se obtienen mediante una suma ponderada de los votos de todos los clasificadores débiles:

$$\hat{y}(x) = \text{signo} \left( \sum_{b=1}^B s_b \hat{y}_b(x) \right)$$

Si esta suma es positiva se clasifica la observación en la clase positiva y análogamente en el caso negativo.

Es importante destacar un matiz crucial: dado que los árboles son dependientes entre sí, el *boosting* es más propenso al sobreajuste que el *random forest* si los datos contienen mucho ruido. Para mitigar esto, en la práctica moderna se suele introducir un parámetro de regularización que escala la contribución de cada árbol  $\nu \cdot s_b$  con  $0 < \nu < 1$ , obligando a generar más árboles pero haciendo el modelo más robusto y generalizable.

### 1.3.2. Métricas de evaluación

La evaluación del rendimiento de los modelos de clasificación constituye una etapa fundamental en cualquier proceso de aprendizaje automático, ya que permite cuantificar su capacidad de realizar predicciones correctas y comparar distintas alternativas de modelado. Para ello, se emplean diversas métricas de evaluación, cada una de las cuales captura distintos aspectos del comportamiento del modelo, como la tasa de aciertos o el equilibrio entre clases. La elección de la métrica adecuada depende del contexto del problema y de los objetivos del análisis, siendo especialmente relevante en situaciones de desbalanceo de clases.

Es habitual construir la matriz de confusión. La matriz de confusión es una tabla de contingencia donde se agrupan los resultados de las predicciones una vez comparadas con la muestra de test del modelo. Las columnas hacen referencia a las predicciones obtenidas por el modelo y las filas a las observaciones de la partición test. Esta matriz es de la forma:

Observación \ Predicción	Positivo	Negativo
Positivo	Verdaderos positivos ( $TP$ )	Falsos negativos ( $FN$ )
Negativo	Falsos positivos ( $FP$ )	Verdaderos negativos ( $TN$ )

Tabla 1.1: Matriz de confusión

Así, la suma  $TP + TN$ <sup>1</sup> es el total de observaciones clasificadas correctamente, mientras que  $FP + FN$  lo contrario. A partir de estos coeficientes se obtienen las distintas medidas de evaluación de los modelos.

La sensibilidad es la proporción de verdaderos positivos dentro de las observaciones positivas:

<sup>1</sup>TP: True Positive. TN: True Negative. FN: False Negative. FP: False Positive.

$$\text{Sensibilidad} = \frac{TP}{TP + FN}.$$

Análogamente se describe la especificidad como la proporción de verdaderos negativos dentro de las observaciones negativas:

$$\text{Especificidad} = \frac{TN}{TN + FP}.$$

La precisión es la proporción de verdaderos positivos dentro de las respuestas clasificadas como verdaderas:

$$\text{Precisión} = \frac{TP}{TP + FP}.$$

La precisión global o tasa de aciertos es la proporción de correctas clasificaciones entre el total. Se denota como ACC para diferenciar del anterior coeficiente:

$$\text{ACC} = \frac{TP + TN}{TP + FN + TN + FP}.$$

Hasta ahora no se tuvo en cuenta el caso de clases desbalanceadas, por lo que estas medidas no serán consistentes en cualquier modelo de clasificación. Con el fin de evitar este problema se describen nuevos métodos de evaluación.

Entre ellos destacan la precisión balanceada:

$$\text{BA} = \frac{\text{Sensibilidad} + \text{Especificidad}}{2},$$

o el coeficiente F1, que toma la expresión:

$$F1 = \frac{2 \cdot \text{Precisión} \cdot \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}},$$

y que si descomponemos en términos de la matriz de confusión se reescribe como:

$$F1 = \frac{2TP}{2TP + FP + FN}$$

Este índice toma valores en el intervalo  $[0, 1]$ . Cuanto más próximo a 1 indica que el modelo acierta con frecuencia. Por otro lado, si el resultado se acerca a 0 tenemos problemas de detección o de clasificación en el modelo. Por ello, el valor de F1 es un indicador sencillo de la calidad de un modelo de clasificación.

Otra medida para modelos con clases desbalanceadas es la kappa de Cohen que compara la tasa de aciertos con la obtenida en una clasificación al azar. Toma la expresión:

$$\kappa = \frac{2(TP \cdot TN - FP \cdot FN)}{(TP + FP)(FP + TN) + (TN + FN)(FN + TP)}$$

y similar al F1, un valor próximo a 1 indica una buena clasificación del modelo y un valor cercano a 0 una clasificación aleatoria.

Finalmente, como medida de la calidad de un modelo se tiene la curva ROC. La curva ROC es un gráfico donde se representa la sensibilidad frente a la proporción complementaria de la especificidad, es decir,  $1 - \text{especificidad}$ .

En la Figura 1.2 se observan dos curvas ROC con índices AUC de 0,94 y 0,86 asociados a un modelo

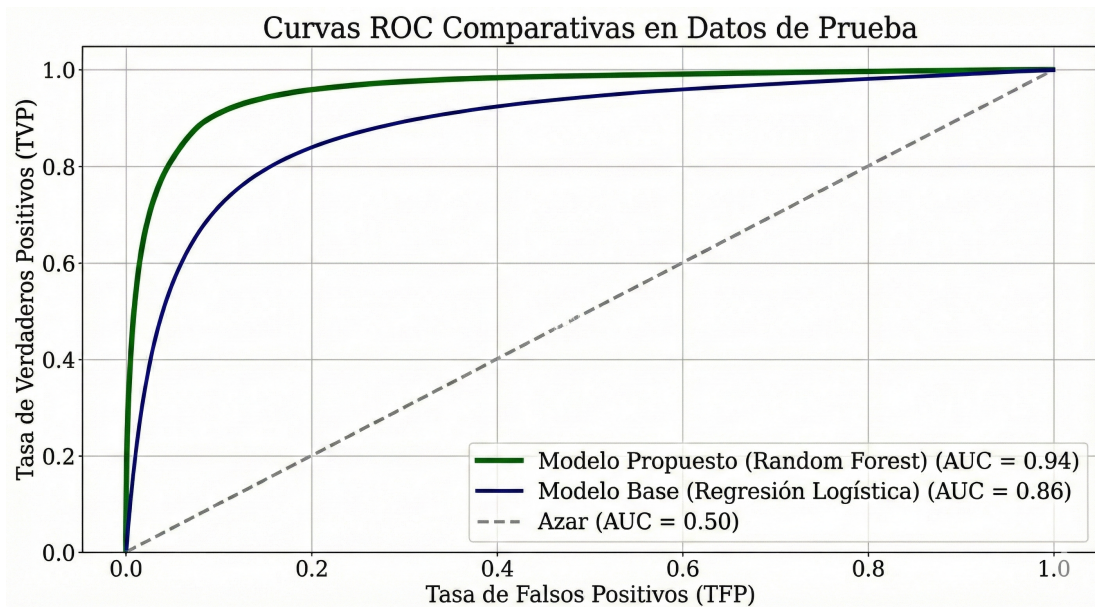


Figura 1.2: Curvas ROC

*random forest* y de Regresión Logística respectivamente para datos simulados arbitrariamente en Python.

Cuanto más se acerca esta curva a la esquina superior izquierda mejores serán las predicciones del modelo. Por otro lado, una curva totalmente diagonal representa una clasificación aleatoria, y por lo tanto, no eficiente.

Un índice que cuantifica este gráfico es el área bajo la curva ROC, denominado como coeficiente AUC, que toma valores entre 0,5 y 1, siendo este último el valor óptimo para un modelo de clasificación.

Otra métrica especialmente útil en problemas con clases desbalanceadas es el área bajo la curva precisión-recall, denominada AUCPR. Mientras que el AUC resume el compromiso entre sensibilidad y especificidad, el AUCPR se centra en la relación entre precisión y sensibilidad, por lo que resulta más exigente cuando interesa detectar correctamente la clase minoritaria. En problemas de *churn*, donde normalmente hay menos abandonos que permanencias, esta métrica ofrece una medida más representativa de la capacidad real del modelo para identificar clientes en riesgo.

También puede evaluarse la calidad de las probabilidades predichas mediante el *LogLoss*. Esta medida compara las probabilidades estimadas por el modelo con los valores reales de la variable respuesta y penaliza especialmente las predicciones erróneas hechas con una confianza muy alta. Por ello, cuanto menor es el *LogLoss*, mejor calibrado está el modelo desde el punto de vista probabilístico.

Otra medida sencilla es el *Mean Error*, que representa la proporción media de clasificaciones incorrectas. Su interpretación es directa: valores pequeños indican que el modelo comete pocos errores globales, mientras que valores altos reflejan una peor capacidad de clasificación. Sin embargo, en contextos desbalanceados esta medida debe interpretarse con cautela, ya que puede verse influida por la clase mayoritaria.

Finalmente, también se emplean el error cuadrático medio (MSE) y su raíz (RMSE). Ambas métricas resumen la distancia entre las probabilidades predichas por el modelo y los valores observados, elevando al cuadrado los errores para penalizar con más intensidad las desviaciones grandes. El RMSE tiene la ventaja adicional de expresarse en la misma escala que la variable respuesta, lo que facilita su interpretación. En ambos casos, valores menores indican un mejor ajuste del modelo.



# Capítulo 2

## Agentes de IA

El objetivo de este capítulo es mostrar una visión amplia, detallada y específica sobre agentes de IA. Se incluyen descripciones sobre su diseño y arquitectura, ejemplos de uso y consejos para su integración en producción.

### 2.1. Modelos de lenguaje (LLM)

Como punto de partida, conviene abordar los modelos de lenguaje (Large Language Models, LLMs) pues sirven como base fundamental en el desarrollo de agentes de IA.

Alrededor del año 2010 comienzan a popularizarse conceptos como *Machine Learning* o *Big Data*.

Como se describe en [8], el *Machine Learning* es una rama de la inteligencia artificial centrada en el entrenamiento automático de modelos a partir de datos. Inicialmente, se centra en modelos de predicción y clasificación, pero con el tiempo se fueron extendiendo a generadores de texto, sirviendo de origen a los llamados modelos de lenguaje. El manejo de volúmenes masivos de datos, conocido como *Big Data*, proporciona al *Machine Learning* la información necesaria para abordar problemas complejos. La combinación de estas dos técnicas resulta en un enfoque sumamente poderoso para el desarrollo de soluciones de IA.

Las redes neuronales son un subconjunto de modelos dentro del *Machine Learning* que se especializa en la búsqueda de patrones y conexiones dentro de una colección de datos. Se emplean principalmente en reconocimiento de imágenes o procesamiento de lenguaje natural.

Un LLM es un modelo de aprendizaje profundo que utiliza redes neuronales con millones de parámetros entrenados a partir de grandes volúmenes de datos. Gracias a esta escala y a su arquitectura, son capaces de comprender y redactar lenguaje natural de forma coherente, razonar sobre temas complejos y justificar respuestas de forma autónoma.

Recientemente, esta capacidad ha evolucionado hacia lo que se conoce como modelos de razonamiento. A diferencia de los LLM tradicionales, que operan bajo un esquema de respuesta rápida e intuitiva, los modelos de razonamiento dedican tiempo de cómputo adicional antes de generar una respuesta final. Este proceso, denominado como cadena de pensamiento, les permite descomponer problemas complejos en más simples, planificando pasos intermedios y rectificando errores lógicos antes de emitir una solución.

El primero en surgir fue el modelo GPT de OpenAI en el año 2018. Como consecuencia, grandes empresas

fueron desarrollando sus propios modelos, como Gemini de Google, Claude de Anthropic o Llama de Meta. Cada uno ha ayudado a expandir las capacidades y alcance de la IA generativa.

La evolución de los LLMs ha permitido pasar de modelos que solo generan texto a sistemas que pueden razonar, usar herramientas externas, interpretar instrucciones complejas y operar de forma estructurada, lo que ha dado lugar al concepto de agentes inteligentes. Estos agentes combinan las capacidades de los LLMs con mecanismos de planificación, memoria y acción, permitiendo resolver tareas que requieren algo más que la simple generación de texto.

Esta transición hacia la autonomía requiere de marcos de evaluación rigurosos, como el recientemente propuesto en [9], que mide la capacidad de estos agentes para realizar investigación de *Machine Learning* de manera autónoma y experimental.

Todos los términos mencionados serán los pilares principales sobre los que se sustenta la IA generativa y, como consecuencia, los agentes de IA.

## 2.2. Definición agentes de IA

Un agente puede definirse como un sistema de software que interactúa con un modelo de lenguaje con el fin de asistir al usuario en la resolución de una tarea específica y, como consecuencia, acelerar su flujo de trabajo.

La principal diferencia entre un LLM y un agente de IA es la integración de herramientas. Mientras que un LLM confía únicamente en sus pesos internos entrenados, un agente se define por su capacidad para externalizar procesos. Esta arquitectura basada en herramientas transforma al modelo de lenguaje en un sistema dinámico, capaz de percibir su entorno y modificarlo para cumplir un objetivo.

Están ganando relevancia en muchos ámbitos profesionales gracias a su naturalidad, ya que su contextualización se basa en un intercambio de mensajes con el cliente en lenguaje natural. Además, tienen la posibilidad de retener gran información en memoria, manteniendo el hilo de la conversación y centrándose en los problemas que el usuario realmente quiere resolver.

Si dispone de las herramientas y credenciales adecuadas, puede acceder a bases de datos, APIs y sistemas de búsqueda de información, lo que amplía su utilidad en entornos profesionales de ciencia de datos. Además, destacan también por su gran capacidad de planificar y delegar tareas entre ellos, permitiendo a cada uno especializarse en una actividad concreta.

## 2.3. Tipos de agentes

Como se mencionaba anteriormente, la funcionalidad y los objetivos de un agente se ven influenciados por las herramientas que se le implementan. Estas herramientas son funciones específicas que se incluyen en su código. De esta manera, hace uso de ellas especializándose en actividades más concretas. Puede observarse un ejemplo en la implementación de un agente básico que se describe en la Sección 2.5.

Gracias a estas herramientas, los agentes se pueden clasificar dependiendo de la función que desempeñan. Siguiendo un posible flujo de ejecución nos podríamos encontrar:

En primer lugar, se encuentran los agentes conversacionales, cuya función principal es dar soporte técnico mediante un chat de preguntas y respuestas con el usuario. Este agente se especializa en captar la naturaleza de la petición y distribuirla entre sus compañeros.

Cuando la tarea requiere la ejecución de acciones concretas, se recurre a agentes de acción. Estos son

capaces de ejecutar código y acceder a APIs. Resultan útiles para realizar trabajos automáticos y pesados.

Por otro lado, los agentes de investigación se orientan a la búsqueda y comparación de información en amplias colecciones de datos. Pueden generar resúmenes y realizar conclusiones.

Finalmente, los agentes de análisis de datos se especializan en depuración, realización de cálculos y gráficos, ajustes de modelos... Resultan muy útiles en la ciencia de datos.

## 2.4. Flujo de pensamiento de un agente de IA

La arquitectura interna de un agente inteligente presenta una estructura generalizada que se mantiene constante independientemente de su finalidad operativa. En este sentido, es posible distinguir una serie de etapas secuenciales que definen su ciclo de procesamiento.

La fase inicial comprende la recepción del mensaje del usuario y la recuperación del historial de interacciones previas. Este proceso de contextualización permite que el agente disponga de la información necesaria para garantizar la coherencia en la resolución de la tarea, integrando las directrices previas proporcionadas por el cliente dentro de su memoria de trabajo.

A partir de este flujo de información, el sistema procede a la composición de un plan de acción. La salida generada puede consistir en una respuesta en lenguaje natural o en una instrucción para la ejecución de funciones externas. Dichas capacidades se encuentran definidas en el *prompt*. Un *prompt* es el conjunto de instrucciones y directrices en lenguaje natural que orientan el comportamiento del sistema de IA para la ejecución de tareas específicas. Este proporciona al agente el marco normativo y las especificaciones técnicas sobre el uso de las herramientas disponibles.

Posteriormente, se inicia la fase de ejecución del plan, donde convergen las facultades específicas del agente mediante un proceso iterativo. Durante este ciclo, el sistema evalúa si la salida obtenida requiere un procesamiento adicional; por ejemplo, si se ha invocado una herramienta, los datos resultantes se reintroducen en el modelo para que este elabore una conclusión fundamentada. El proceso finaliza únicamente cuando el agente genera un resultado estructurado en formatos interpretables por el usuario, tales como texto plano, tablas o representaciones gráficas.

Tras la obtención de la salida definitiva, el agente procede a su estructuración y entrega. Finalmente, el sistema realiza una síntesis de las conclusiones obtenidas y actualiza su memoria, asegurando que la información relevante sea preservada para optimizar las iteraciones subsiguientes.

## 2.5. Implementación de un agente: ejemplo

Actualmente, el desarrollo de agentes impulsados por LLMs se realiza principalmente con un framework llamado LangChain. En su documentación [10] se detalla en profundidad. Esta librería está en constante evolución por lo que la metodología para implementar agentes ha variado en los últimos meses.

Interesa destacar la salida de la versión 1.0.0, que marca un punto de inflexión en la creación de agentes, cambiando su estructura prácticamente por completo. Es por esta razón que se presentan dos ejemplos de agentes de IA. El primero trabaja en la versión 0.3.12 (versión antigua) y el segundo en la 1.0.2 (versión actual).

Con el siguiente código se crea un agente capaz de responder preguntas de forma clara y simple. Por ejemplo, le preguntamos cuál es la capital de Francia:

## Ejemplo versión antigua

```

1
2     from langchain.prompts import PromptTemplate
3
4     # Clase base para agentes
5     class Agente:
6         def __init__(self, nombre):
7             self.nombre = nombre
8             self.llm = model
9
10        def ejecutar(self, input_text):
11            """
12            Función principal del agente.
13            Debe ser sobrescrita por cada tipo de agente para definir su
14            comportamiento.
15            """
16            raise NotImplementedError("La función debe implementarse en la
17            subclass")
18
19        # Ejemplo de agente que responde preguntas
20        class AgentePreguntas(Agente):
21            def __init__(self, nombre="AgentePreguntas"):
22                super().__init__(nombre)
23                self.prompt = PromptTemplate(
24                    input_variables=["pregunta"],
25                    template="Responde la siguiente pregunta: {pregunta}"
26                )
27                self.chain = self.prompt | model | parser
28
29            def ejecutar(self, pregunta):
30                """
31                Ejecuta el agente usando la cadena de LangChain
32                """
33                respuesta = self.chain.invoke(pregunta)
34                return respuesta
35
36        # Uso del agente
37        mi_agente = AgentePreguntas()
38        pregunta = "¿Cuál es la capital de Francia?"
39        respuesta = mi_agente.ejecutar(pregunta)
40        print(f"Respuesta: {respuesta}")

```

La definición del modelo LLM puede variar dependiendo del distribuidor de LLMs al cual se tenga acceso. En este trabajo se hace uso de AzureChatOpenAI, por lo que la definición es del estilo:

## Definición LLM

```

1     modelo = AzureChatOpenAI(
2         azure_endpoint= azure_endpoint,
3         openai_api_key= azure_api_key,
4         openai_api_type= "azure",
5         openai_api_version= "2024-12-01-preview",
6         deployment_name= "gpt-5-mini",
7         model_name= "gpt-5-mini",
8         request_timeout= 120,
9     )
10

```

El siguiente ejemplo está inspirado en [10], donde crearemos un agente experto en el tiempo meteorológico de cualquier ciudad del mundo:

#### Ejemplo versión actual

```

1      # Cargamos las librerías necesarias
2      from langchain.tools import tool
3      from langchain.agents import create_agent
4      from langchain_core.messages import AIMessage
5
6      # Definimos las herramientas que queremos incorporar a nuestro agente
7      @tool
8      def search(query: str) -> str:
9          """Search for information."""
10         return f"Results for: {query}"
11
12     @tool
13     def get_weather():
14         """Get weather information for a location."""
15
16     # Definición del LLM que se va a usar
17     modelo = modelo
18
19     # Definimos el agente con la función create_agent()
20     agent = create_agent(model=modelo, tools=[search, get_weather])
21
22     # Ejecutamos el agente con un prompt básico de entrada
23     result = agent.invoke({
24         "messages": [
25             {"role": "user", "content": "¿Cuál es el clima en Madrid?"}
26         ]
27     })
28
29     # Pedimos que se imprima en pantalla el contenido del último AIMessage
30     messages = result["messages"]
31     last_ai_message = [m for m in messages if isinstance(m, AIMessage)][-1]
32
33     print(last_ai_message.content)
34

```

En la versión nueva destaca la existencia de la función `create_agent()` que unifica la creación de agentes. Basta con incluir los argumentos requeridos para su producción.

El repositorio de partida [1] es anterior a la salida de esta versión. Debido a esto, este trabajo está desarrollado en la versión antigua y no se beneficia de las ventajas que ofrece la actual. Como línea de trabajo futura, podría plantearse la migración de este código a la versión actual de LangChain.

## 2.6. Librerías y frameworks comunes

En el desarrollo de agentes de IA se requiere de un ecosistema tecnológico que abarque desde los aspectos internos más técnicos hasta la interfaz de visualización. En esta sección, se detalla la selección de herramientas de Python empleadas. Desde LangChain y LangGraph que funcionan como motores y organizadores del flujo de los modelos, Streamlit para el despliegue de la interfaz interactiva, Logfire que garantiza la observabilidad y depuración del sistema, y AzureChatOpenAI que proporciona el acceso a los modelos de lenguaje.

### 2.6.1. LangChain

LangChain es un framework de Python diseñado para el desarrollo estructurado de aplicaciones basadas en LLMs. Facilita la creación de agentes, cadenas de razonamiento, flujos con herramientas externas y búsqueda de información.

Su implementación y uso básico se describe en los ejemplos de la Sección 2.5, tanto para la versión antigua (0.x) como la actual (1.x).

### 2.6.2. LangGraph

LangGraph es un framework diseñado para crear agentes y estructurar sus flujos de trabajo de forma consistente y ordenada mediante grafos de estado. Recomendable cuando aumenta el volumen de agentes en un proyecto.

Con esta librería, cada nodo del grafo representa un agente y las aristas guían los procesos del sistema, marcando un camino y un orden para las tareas que se le presentan. Además, gracias al estado del grafo, es posible conservar información de consultas anteriores y reutilizarla en iteraciones posteriores.

#### Ejemplo LangGraph

```

1     from typing import TypedDict
2     from langgraph.graph import StateGraph, END
3     from langchain_core.messages import AIMessage, HumanMessage
4     from langchain_core.prompts import ChatPromptTemplate
5
6     prompt = ChatPromptTemplate.from_messages([ ("system", "Identifica si el \\\"
7     \"siguiente texto es un saludo o no: {message}. Si es un saludo, \\\"
8     \"responde 'greeting', si no lo es, responde 'unknown'.\"), ])
9
10    chain = prompt | modelo | parser
11
12    # --- 1. Estado ---
13    class State(TypedDict):
14        message: AIMessage | None
15
16    # --- 2. Nodos (funciones) ---
17
18    # Nodo 1: Recibir mensaje
19    def receive_input(state):
20        return {"message": state["message"]}
21
22    # Nodo 2: Clasificación sencilla
23    def understand_message(state):
24        text = state["message"].content.lower()
25        classification = chain.invoke({"message": text})
26        if classification.strip() == "greeting":
27            return {"intent": "greeting"}
28        else:
29            return {"intent": "unknown"}
30
31    # Nodo 3a: Respuesta a saludo
32    def greeting_response(state):
33        return {"message": AIMessage(content="¡Hola! Encantado de ayudarte")}
34
35    # Nodo 3b: No entendí
36    def default_response(state):
37        return {"message": AIMessage(content="No entendí tu mensaje")}
38
39    # --- 3. Construcción del grafo ---

```

```

40     builder = StateGraph(State)
41
42     builder.add_node("receive", receive_input)
43     builder.add_node("classify", understand_message)
44     builder.add_node("greeting", greeting_response)
45     builder.add_node("default", default_response)
46
47     builder.set_entry_point("receive")
48
49     # Flujo principal
50     builder.add_edge("receive", "classify")
51
52     # --- Condiciones ---
53     def is_greeting(state):
54         return state.get("intent") == "greeting"
55
56     def is_unknown(state):
57         return state.get("intent") == "unknown"
58
59     builder.add_conditional_edges(
60         "classify",
61         is_greeting,
62         {
63             True: "greeting",
64             False: "default"
65         }
66     )
67
68     builder.add_edge("greeting", END)
69     builder.add_edge("default", END)
70
71     # --- 4. Compilar ---
72     graph = builder.compile()
73
74     # --- 5. Probar agente ---
75     # Caso 1: saludo
76     result1 = graph.invoke({"message": HumanMessage(content="Hola agente!")})
77     print("Caso saludo:", result1["message"].content)
78     # Caso 2: no saludo
79     result2 = graph.invoke({"message": HumanMessage(content="Qué tiempo hace?")
80 })
81     print("Caso no saludo:", result2["message"].content)
82

```

Este código representa un agente desarrollado con LangGraph que clasifica un mensaje en saludo o no saludo y actúa en consecuencia. Es un ejemplo básico en el que contamos con cuatro nodos:

1. **Receive:** Recibe y procesa el mensaje por chat.
2. **Classify:** Con la ayuda de un LLM clasifica el mensaje en saludo o desconocido.
3. **Greeting:** Entra en acción si el mensaje es declarado como saludo. Entonces el agente devuelve el saludo.
4. **Default:** El caso complementario. Como este agente solo distingue entre saludo o no saludo responde que no entiende el mensaje.

También contamos con un arco condicional que representa la clasificación saludo o no saludo.

### 2.6.3. AzureChatOpenAI

AzureChatOpenAI es un servicio de LangChain que facilita el uso de LLMs alojados en la plataforma de Microsoft Azure. Gracias a una contraseña proporcionada por el proveedor, llamada ApiKey, se conecta al modelo, permitiendo interactuar con LLMs tipo chat directamente desde Python. Su implementación se detalla en el apartado 2.5.

### 2.6.4. Streamlit

Streamlit es un framework de Python que permite crear aplicaciones web interactivas de manera rápida y sencilla, sin necesidad de conocimientos avanzados de desarrollo web. Se empleará para interactuar mediante chat con los agentes. Un ejemplo sencillo sería el siguiente:

#### Ejemplo streamlit

```

1     import streamlit as st
2
3     st.title("Mi primera app con Streamlit")
4     st.write("Hola, mundo! Este es un texto en la app.")
5
6     # Input del usuario
7     nombre = st.text_input("Escribe tu nombre:")
8     if nombre:
9         st.write(f"Hola, {nombre}!")
10

```

Esta sencilla página web contiene texto, pregunta el nombre al usuario y saluda.

### 2.6.5. Logfire

Logfire es una herramienta de observabilidad diseñada para capturar logs<sup>1</sup>. Permite registrar y monitorear fácilmente código, frameworks web, bases de datos, llamadas a LLMs y otras dependencias comunes, de modo que se pueda seguir lo que ocurre (peticiones, errores, uso de recursos...). No es una librería esencial en el uso de agentes pero tendrá un papel clave en el desarrollo de este proyecto facilitando la comprensión del repositorio base [1].

Su implementación es relativamente sencilla:

#### Implementación logfire

```

1     import logfire
2     import os
3
4     # Inicialización logfire
5     os.environ['LANGSMITH_OTEL_ENABLED'] = 'true'
6     os.environ['LANGSMITH_TRACING'] = 'true'
7
8     logfire.configure(token=token)
9

```

Para obtener el valor *token* se debe acceder a [11]. En esta página web se debe crear un nuevo proyecto. Una vez creado, en el apartado “write token” se proporcionará una cadena alfanumérica privada que debe incorporarse al código. Así tener un seguimiento preciso del flujo de trabajo del agente.

<sup>1</sup>Un log es un archivo que registra de forma secuencial y cronológica los eventos y actividades de un sistema, aplicación o servidor.

## 2.7. Tipos de memoria

Como ya se mencionó, una de las grandes ventajas de los agentes de IA es la capacidad de retener memoria de conversaciones pasadas con el cliente. Esto se consigue al almacenar los mensajes de entrada y salida de una conversación en una lista de Python. Cada vez que se interactúa con el agente se le proporciona esta lista permitiéndole aumentar el contexto de trabajo y ayudándole a entender con exactitud el problema que se intenta resolver.

En un principio, se guardaba el historial completo de la sesión. Esto no implicaba ningún problema en casos de consultas sencillas, pero para problemas más complejos, donde la conversación se podía alargar, el peso en memoria de todos los mensajes podía resultar perjudicial para el funcionamiento del agente. Es por ello que se fueron desarrollando nuevos métodos:

- **Historial de ventana:** Guarda únicamente los  $k$  últimos mensajes de la conversación, siendo  $k$  un número entero fijado de antemano.
- **Historial con resumen:** Resume lo ocurrido en cada iteración entre usuario y agente empleando un LLM.
- **Almacenamiento a largo plazo:** Mantiene el historial entre sesiones de trabajo. Esto se realiza guardando en una base de datos los mensajes entre usuario y agente. Es aconsejable que cada usuario cuente con una clave única. De esta forma, al iniciar una nueva sesión se recargan únicamente conversaciones asociadas a esa clave.

Este último puede resultar peligroso al acumular enorme información, pero combinándose con el historial con resumen resulta de gran utilidad en un entorno de trabajo compartido.

## 2.8. Métodos de evaluación

Para completar la descripción de un agente de IA se detallan a continuación los métodos más comunes de evaluar su rendimiento. Combinamos métricas técnicas junto a indicadores cualitativos.

### 2.8.1. Evaluación empírica

Un aspecto fundamental en la evaluación de agentes es su eficiencia en términos de tiempo de respuesta y consumo de recursos, especialmente el número de *tokens* utilizados en cada interacción. El objetivo principal es que el agente mantenga un coste operativo razonable en relación con la utilidad que aporta. También es importante estudiar la cobertura de uso de las herramientas a su disposición, es decir, revisar que aprovecha todos los recursos que se invierten en él.

### 2.8.2. Evaluación cualitativa

Más allá de los aspectos técnicos, también es importante la fluidez, la calidad conversacional y la coherencia de estos agentes. Es por ello que se pueden realizar encuestas de satisfacción a los usuarios u otros tipos de evaluaciones humanas.

Además de las evaluaciones humanas, se ha extendido el uso de LLMs actuando como jueces (*LLM as judge*) que analizan y califican la calidad de la salida teniendo en cuenta un criterio fijado en el *prompt*. Así, el modelo es evaluado por sí mismo o un LLM externo y en caso de no cumplir ciertas condiciones se da un aviso al desarrollador para solicitar intervención humana. Esto permite realizar evaluaciones automáticas y escalables, valorando aspectos como la coherencia, la corrección factual y la adecuación al *prompt* o estilo. No sustituyen completamente la evaluación humana, pero ofrecen una forma rápida y consistente de medir el rendimiento conversacional y compararlo entre diferentes versiones o configuraciones del agente.

La combinación de estos criterios técnico y humano es una metodología sencilla y clara para juzgar el rendimiento de un agente. Esto permite orientar de forma eficiente su evolución y mejora.

## 2.9. Objetivos de mejora a largo plazo

Si bien el desarrollo de agentes en entornos de trabajo es un avance revolucionario, sigue siendo un campo muy amplio en constante evolución. A continuación se exponen aspectos interesantes para investigar.

Una mejora importante sería la prevención del llamado *prompt injection*. Se define como un ataque malicioso destinado a manipular las instrucciones internas del sistema modificando el *prompt* que llega al LLM. Su objetivo es engañar al sistema para obtener información restringida o provocar comportamientos subóptimos o no deseados en los agentes.

Las mejoras de eficiencia son siempre bienvenidas. Un agente capaz de realizar múltiples tareas de forma inmediata sería la herramienta ideal. En este ámbito resulta necesario ser ambicioso y buscar la reducción de tiempos de ejecución sin disparar costes computacionales y energéticos.

Una línea de mejora relevante es el desarrollo de mecanismos de razonamiento verificado. En este enfoque, el agente no solo genera conclusiones a partir de las herramientas disponibles, sino que incorpora procedimientos adicionales de contraste orientados a comprobar su coherencia y veracidad antes de producir la respuesta final.

Como se puede observar, los agentes constituyen una alternativa útil en determinados entornos profesionales. Permiten automatizar tareas, generar código, ajustar modelos y, en términos generales, agilizar el flujo de trabajo.

Siguiendo pasos concretos y detallados es factible construir agentes útiles, y la investigación y la práctica seguirán afinando su fiabilidad y sus prestaciones con el paso del tiempo, hasta convertirlos en herramientas de apoyo relevantes para profesionales de ciencia de datos.

## 2.10. Marco tecnológico

Este trabajo se basa en el repositorio [1], donde se definen y estructuran distintos agentes. Cada uno tiene distintas características y objetivos y algunos se complementan entre sí. A continuación se detallan aspectos técnicos específicos del repositorio para su inicialización.

Este repositorio requiere trabajar con Python en la versión 3.11.8 para su correcto funcionamiento.

Con respecto a las librerías necesarias, se encuentran en el archivo “requirements.txt”. En un primer momento, LangChain y LangGraph presentaban problemas debido a la salida de la versión 1.0.0 y su incompatibilidad con la 0.x. Sin embargo, con la actual 1.0.5 y 1.0.3 de LangChain y LangGraph respectivamente no se generan errores. Se fijan estas versiones en el archivo y serán con las que se ejecuta el repositorio.

El resto de librerías también tienen sus versiones fijadas en el “requirements.txt”, por lo que para poder ejecutar el repositorio primero se debe instalar el archivo. Se detallan a continuación los pasos que se deben seguir.

Una vez creado un entorno virtual se ejecuta en una terminal:

### Actualización librería pip

```
1 python.exe -m pip install --upgrade pip
2
```

Así se actualiza la librería pip para instalar el resto de dependencias. Conviene tenerla en su última versión para evitar problemas de compatibilidad. Luego se ejecuta:

### Instalación archivo “requirements.txt”

```
1 pip install -r requirements.txt
2
```

Puede llevar un tiempo moderado, pues instala todas las librerías con la versión especificada en el “requirements.txt”.

Los agentes del repositorio se ejecutan como herramientas separadas dentro de aplicaciones Streamlit. Como ya mencionamos, Streamlit es un framework que permite crear una página web a partir de código.

Para poder abrir una de estas aplicaciones debemos movernos entre los directorios del repositorio hasta encontrar alguno de los archivos llamados “app.py”. Una vez delante de uno ejecutamos en la terminal:

### Ejecución de un streamlit

```
1 streamlit run app.py
2
```

Así, se iniciará la aplicación en una nueva pestaña del navegador web y podremos utilizarla libremente.

## 2.10.1. Aplicaciones

En cada aplicación trabajan uno o varios agentes con finalidades parecidas. Inicialmente, contamos con tres aplicaciones: “Panda-data-analyst-app”, “SQL-database-app” y “AI-exploratory-copilot-app”.

### Pandas Data Analyst

Esta aplicación está diseñada para el análisis de datos desde Python utilizando la librería Pandas. Es capaz de cargar datos, estudiarlos, limpiar valores, realizar cálculos a partir de ellos y generar tablas y gráficos de salida. En general, es útil para automatizar y agilizar el análisis exploratorio de datos y puede facilitar el proceso de preparación de datos en un proyecto.

La aplicación está constituida por tres agentes:

- **Data Wrangling Agent:** Realiza todo tipo de cálculos con el conjunto de datos implementado en la herramienta. Su salida siempre es una tabla con los resultados solicitados.
- **Data Visualization Agent:** Construye todo tipo de gráficos sobre el conjunto de datos implementado. Su salida es un gráfico de Python.

Ambos agentes se dividen en dos subagentes. El primero recomienda pasos que se deberían seguir para completar la consulta y el segundo programa la función que la resuelve.

- Pandas Data Analyst:** Su función es identificar el objetivo de la consulta y distribuirla entre los agentes. Su salida es un formato JSON con tres pares clave-valor. El primero, es la sentencia que debe ejecutar el agente Data Wrangling. El segundo, la sentencia que debe ejecutar el agente Data Visualization. Por último, una decisión acerca del formato de salida que debe tener la consulta (sus valores posibles son tabla y gráfico).

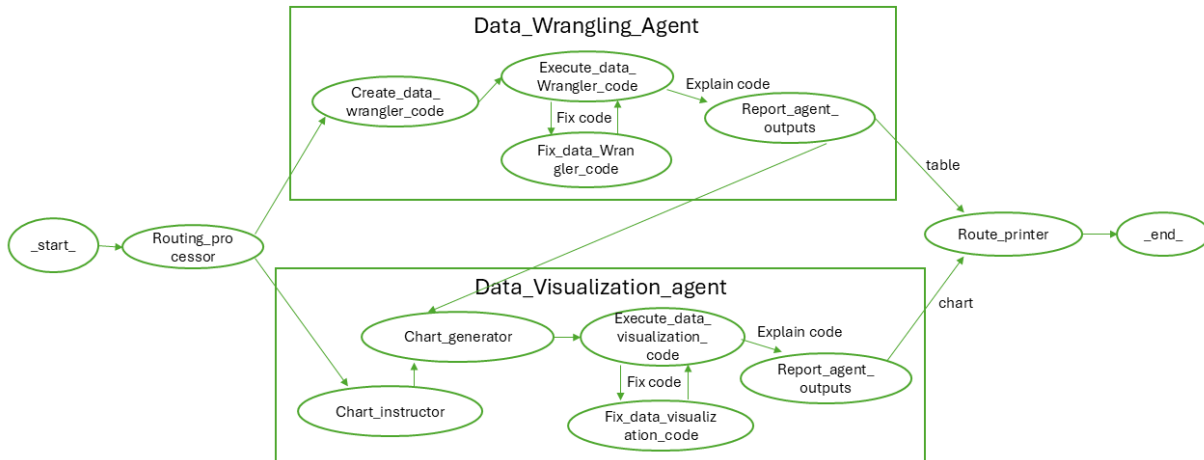


Figura 2.1: Grafo Pandas Data Analyst

En la Figura 2.1 se representa el grafo de ejecución de la aplicación Pandas Data Analyst.

Observamos que cada agente está estructurado en submódulos que realizan tareas específicas. Por ejemplo, en el Data Wrangling, se separan las tareas de crear el código, ejecutarlo, mejorarlo y mostrarlo. El resto de agentes del repositorio sigue una estructura similar centrándose cada uno en su finalidad.

## SQL database

La función de esta aplicación es facilitar el uso de código SQL a través de peticiones en un chat. Está compuesta por dos agentes:

- SQL Database:** Recomienda pasos a seguir en una consulta SQL y construye un código en este lenguaje para completar la consulta del usuario.
- Data Visualization Agent:** Se explicó en la anterior aplicación.

Para modificar la base de datos con la que opera la aplicación se debe modificar un apartado del archivo `app.py` de la carpeta `/apps/sql-database-agent-app`. A continuación se muestra cómo está implementada la base de datos “Northwind.db” que viene por defecto en el repositorio [1].

### Implementación database Northwind

```

1     DB_OPTIONS = {
2         "Northwind Database": f"sqlite:///ruta_local/data/northwind.db",
3     }
4 
```

La variable `ruta_local` debe guardar el directorio de trabajo. Para poder modificar la base de datos desde la aplicación debemos incluir una nueva opción en este apartado, por ejemplo:

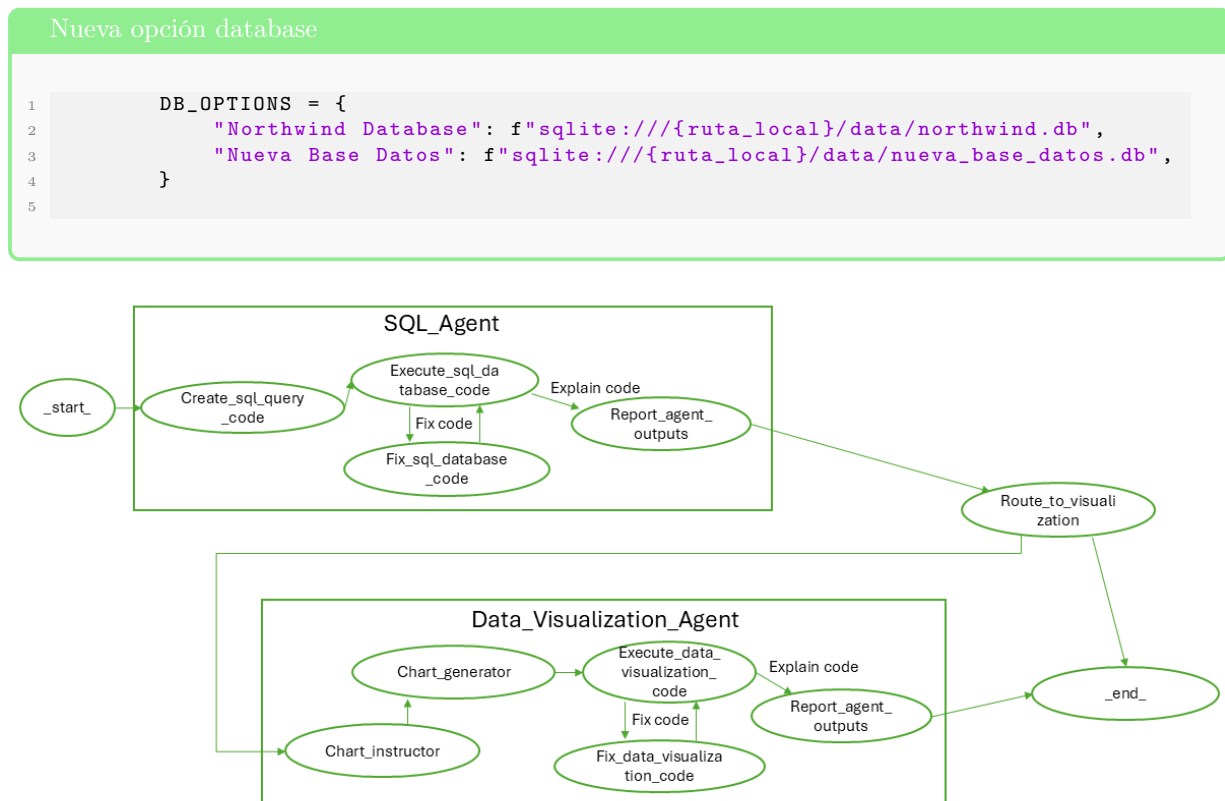


Figura 2.2: Grafo SQL database

En la Figura 2.2 se representa el grafo de ejecución de la aplicación SQL database.

### Exploratory copilot

Esta aplicación está constituida únicamente por el agente homónimo. Su principal objetivo es asistir y realizar actividades relacionadas con el análisis exploratorio (EDA) de cualquier conjunto de datos. Es un agente amplio que hace uso de múltiples herramientas listadas a continuación:

- **Explain data:** Produce un resumen narrativo extenso de un DataFrame (forma, tipos de datos, porcentaje de missings, recuentos únicos, filas del conjunto de datos y estadísticas descriptivas opcionales).
- **Describe dataset:** Calcula estadísticas descriptivas. Devuelve un texto breve explicativo y un artefacto estructurado para su uso posterior.
- **Visualization missing:** Genera visualizaciones de valores missing (gráficos tipo matriz, barra, mapa de calor...) utilizando la librería missingno.
- **Generate correlation funnel:** Ejecuta un análisis tipo funnel de correlación: binariza las características y calcula la correlación frente a un nivel objetivo.
- **Generate Sweetviz report:** Genera un informe HTML de Sweetviz EDA<sup>2</sup> para el conjunto de datos (opcionalmente se puede modificar el destino).

<sup>2</sup>Herramienta de Python. Su función principal es generar un informe visual de alto impacto y muy detallado en un único archivo HTML interactivo

- **Generate data report:** Crea una exploración data interactiva y proporciona una URL para manejarla.

Por defecto, esta aplicación no tiene permitido enviar *hyperlinks* ni ningún tipo de enlace, por lo que, para utilizar la última herramienta descrita, será necesario autorizar explícitamente dicha acción mediante el chat.

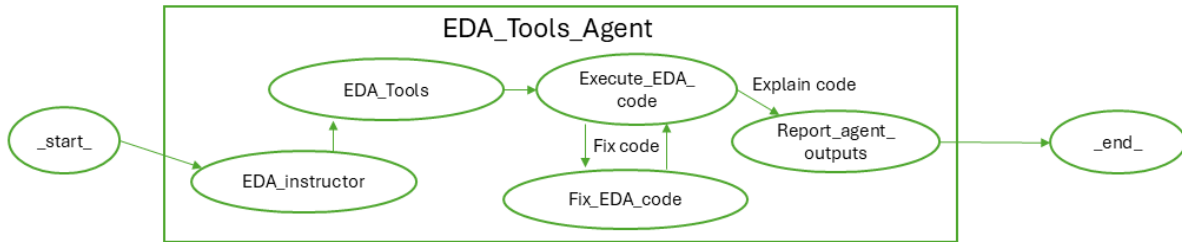


Figura 2.3: Grafo exploratory copilot

En la Figura 2.3 se representa el grafo de ejecución de la aplicación exploratory copilot.

### 2.10.2. Agentes

Aparte de las tres aplicaciones mencionadas, en el repositorio se encuentran más agentes que todavía no están implementados. A continuación se listan y se hace una breve descripción de sus funciones y utilidades.

- **Data cleaning:** Se le proporciona un conjunto de datos. A partir de este devuelve pasos recomendados para realizar su limpieza y crea una función que los lleva a cabo. Devuelve un nuevo dataframe con los datos procesados.
- **Data loader tools:** Un agente que puede interactuar con herramientas de carga de datos y buscar archivos en el sistema de archivos.
- **Feature engineering:** Puede procesar conjuntos de datos según instrucciones definidas por el usuario. Es capaz de seleccionar, transformar y crear nuevas variables a partir del conjunto de datos proporcionado. El agente genera una función de Python, la ejecuta y registra el proceso, incluyendo código y errores.
- **H2O ml:** Un agente de aprendizaje automático que utiliza AutoML de H2O, una herramienta desarrollada por H2O.ai que permite construir modelos de *Machine Learning* de forma automática, incluso sin tener conocimientos profundos de ciencia de datos. A partir de un conjunto de datos crea un código Python que construye el modelo.
- **Mlflow tools:** Agente capaz de interactuar con MLflow llamando a herramientas. MLflow es una plataforma abierta para gestionar el ciclo de vida del *Machine Learning*. Ayuda a realizar el *tracking* de los modelos desarrollados, a organizar las versiones del código, a gestionar el *packaging* de los modelos junto a sus dependencias y facilita su implementación en desarrollo.
- **SQL data analyst:** Clase multiagente que combina consultas a bases de datos SQL y capacidades de visualización de datos.

# Capítulo 3

## Uso del sistema de agentes: *churn*

Este capítulo se centra en el uso del sistema de agentes en el estudio de un problema de *churn*. Para ello se emplea el conjunto de datos *churn\_data.csv* y las funciones descritas de los distintos agentes. El objetivo es realizar un análisis completo de este fenómeno con los datos proporcionados.

### 3.1. Análisis exploratorio

El primer paso es estudiar la estructura y composición del conjunto de datos. Para ello, se realiza un análisis exploratorio con la aplicación **Exploratory-copilot** de nuestro sistema de agentes. Gracias a esta, se validará la calidad de los datos e identificaremos patrones estadísticos significativos.

El primer prompt introducido es:

```
Prompt 1
1  Háblame sobre el dataset churn. Dimensiones, variables, datos faltantes,
2  distribuciones de cada campo... lo que te parezca conveniente
3
```

A partir de este, el agente da la siguiente salida:

Perfecto, aquí tienes un resumen claro y práctico del conjunto de datos “churn”. Si quieres que profundice en algún punto (tablas de frecuencias exactas, gráficos, conversión de columnas, limpieza automática, etc.), dímelo y lo hago.

1. Dimensiones y variables:

- Filas x columnas: 7043 filas x 21 columnas
- Variables (y tipo detectado):
  - object (Categóricos/Texto): customerID, gender, Partner, Dependents, PhoneService, MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies, Contract, PaperlessBilling, PaymentMethod, TotalCharges, Churn.

- int64 (Enteros): SeniorCitizen, tenure.
- float64 (Decimales): MonthlyCharges.

## 2. Valores faltantes:

- Según el análisis inicial, todas las columnas aparecen con 0.00 % de valores nulos. Sin embargo, atención con **TotalCharges**: está en tipo object (texto), lo que sugiere que puede contener valores vacíos o espacios en blanco que no se detectan como NaN y por eso aún no aparecen como “missing”. Es algo frecuente en este conjunto de datos. Recomiendo convertir **TotalCharges** a numérico (`pd.to_numeric(..., errors='coerce')`) para revelar posibles valores no convertibles como NaN.
- Conteos de niveles / cardinalidad (resumen)
  - Identificadores: **customerID** (7043 únicos).
  - Binarios (2 valores): **gender**, **SeniorCitizen** (0/1), **Partner**, **Dependents**, **PhoneService**, **PaperlessBilling**, **Churn** (Yes/No).
  - Categorías múltiples:
    - 3 valores: **MultipleLines**, **InternetService**, **Contract**, **Servicios de internet** (Security, Backup, etc.).
    - 4 valores: **PaymentMethod**.
  - Numéricos:
    - **tenure**: 73 valores distintos (0 – 72 meses).
    - **MonthlyCharges**: 1585 valores distintos.
    - **TotalCharges**: 6531 valores distintos (actualmente texto).
- Muestra de los primeros 2 registros
  - El conjunto de datos contiene filas de ejemplo como:
    - **customerID** = 7590-VHVEG, **Female**, **SeniorCitizen** = 0, **tenure** = 1, **PhoneService** = No, **InternetService** = DSL, **MonthlyCharges** = 29,85, **TotalCharges** = 29,85, **Churn** = No
    - **customerID** = 5575-GNVDE, **Male**, **tenure** = 34, **PhoneService** = Yes, **InternetService** = DSL, **MonthlyCharges** = 56,95, **TotalCharges** = 1889,5, **Churn** = No
  - Observación: algunas columnas usan textos como “No phone service” o “No internet service” en lugar de NaN; eso es importante a la hora de agrupar o de crear variables binarias.
- Estadísticas numéricas (distribución resumida)
  - **SeniorCitizen**: Media  $\approx 0,162$  (16,2 % de clientes senior).
  - **tenure** (Antigüedad):

- Media: 32,37 meses.
- Mediana: 29 meses.
- Rango: 0 a 72 meses (alta dispersión).
- **MonthlyCharges:**
  - Media: 64,76.
  - Mediana: 70,35.
  - Min  $\approx$  18,25. Max  $\approx$  118,75.
  - *Nota:* La mediana superior a la media sugiere una ligera asimetría hacia valores altos.
- Puntos importantes a revisar / recomendaciones
  - **TotalCharges** está como object: convertir a numérico (`pd.to_numeric`) para tratarlo como variable continua e identificar valores no numéricos o vacíos.
  - Revisar las ocurrencias de textos especiales como “No phone service” / “No internet service” en variables relacionadas; a menudo conviene mapearlos a una categoría coherente o crear variables indicadoras (ej. `PhoneService == 'No'` implica `MultipleLines = 'No phone service'` → manejar esa relación).
  - Obtener la distribución exacta de la variable objetivo **Churn** (porcentaje Yes/No). En el resumen solo aparece que **Churn** tiene 2 niveles; si quieres, calculo la proporción y te doy la tabla con conteos y porcentajes.
  - Para variables categóricas conviene generar tablas de frecuencia (conteos y %), y gráficos (barras, boxplots de **MonthlyCharges** por **Churn**, histogramas de **tenure**, etc.).
  - Para datos faltantes: convertir **TotalCharges** detectará NaNs ocultos; además algunos valores inconsistentes pueden necesitar limpieza (espacios, cadenas vacías).
  - Posibles análisis siguientes:
    - Tabla de contingencia y tasa de *churn* por categoría (**Contract**, **InternetService**, **PaymentMethod**, etc.).
    - Histogramas / densidades de **tenure** y **MonthlyCharges**, y boxplots por **Churn**.
    - Transformación de variables categóricas a dummies para modelado; o usar la metodología de correlation funnel para hallar features correlacionadas con **churn**.
    - Análisis de cohortes por **tenure** y **Churn**.
  - Uso de One Hot encoding para la construcción de un nuevo conjunto de datos óptimo para el ajuste de modelos.
- ¿Qué quieres que haga ahora? Puedo:
  - Calcular y devolverte las tablas completas de frecuencias y porcentajes para todas las variables categóricas y la distribución numérica detallada (bins, deciles).

- Limpiar `TotalCharges` (convertir a numérico), mostrar cuántos se convierten a NaN y distribuirlos.
- Mostrar la proporción exacta de `Churn` (Yes/No) y tasas por variables clave (`Contract`, `InternetService`, `PaymentMethod`).
- Generar gráficos (histogramas, boxplots, heatmap de correlación) y/o un informe Sweetviz / visualizaciones de faltantes.

Dime cuál de estas acciones prefieres y lo realizo (puedo entregar tablas numéricas o gráficos).

En [A](#) se presenta un código que valida estos resultados presentados por la aplicación **Exploratoy-copilot**.

## 3.2. Análisis previos

El siguiente paso es realizar los cambios en el conjunto de datos que recomendó el agente. Introducimos entonces el apartado “Puntos importantes a revisar / recomendaciones” de la salida anterior a su compañero agente **Pandas-data-analyst**, experto en el manejo de conjuntos de datos.

Siguiendo las recomendaciones del análisis inicial, intentamos aplicar todas las correcciones y limpiezas sugeridas por el agente en una sola ejecución. Este enfoque resultó ser demasiado ambicioso para la capacidad de memoria del sistema pues el proceso colapsó debido a la longitud excesiva del contexto y los recursos computacionales requeridos. Este error nos obligó a replantear la estrategia hacia una ejecución paso a paso.

Así, el siguiente prompt que se ejecuta es:

### Prompt 2

```

1     Hemos realizado un análisis exploratorio sobre los datos ``churn_data.csv``
2     y nos han recomendado los siguientes pasos a seguir:
3
4     TotalCharges está como object: convertir a numérico (pd.to_numeric) para
5     tratarlo como variable continua e identificar valores no numéricos o vacíos.
6

```

El resultado de esta consulta fue una versión extendida del conjunto de datos `churn_data.csv`, incorporando el campo `TotalCharges` en formato numérico y una variable binaria adicional para identificar registros conflictivos. Tras inspeccionar las observaciones marcadas por esta nueva variable, confirmamos que correspondían a valores nulos.

Seguimos con la siguiente tarea recomendada del análisis inicial:

### Prompt 3

```

1     Revisar las ocurrencias de textos especiales como "No phone service" / "No
2     internet service" en variables relacionadas; a menudo conviene mapearlos a
3     una categoría coherente o crear variables indicadoras (ej.
4     PhoneService == 'No' implica MultipleLines = 'No phone service'
5     -> manejar esa relación).
6

```

Como resultado obtenemos un nuevo conjunto de datos donde se observa una estandarización semántica

de las variables categóricas. En concreto, se simplificó la cardinalidad de variables como `MultipleLines`, `OnlineSecurity` y `TechSupport`, donde etiquetas redundantes como “No phone service” o “No internet service” fueron correctamente mapeadas a la categoría base “No”. Esta reducción de niveles es beneficiosa para el modelado posterior, ya que elimina duplicidades conceptuales y evita que los algoritmos traten la ausencia de un servicio base como una categoría de comportamiento distinta.

Sin embargo, el agente introdujo un aumento de la dimensionalidad del conjunto de datos. En lugar de limitarse a la recodificación de valores, se generaron 19 nuevas columnas auxiliares de tipo indicador. Por ejemplo `MultipleLines_no_internet_service`, la cual toma valores 1 en las observaciones donde la variable `MultipleLines` tomaba el valor “No internet service” y 0 en el resto. Muchas de estas presentan únicamente valores nulos e información redundante respecto a las variables originales ya limpiadas, aportando ruido al conjunto de datos, por lo que se debe considerar el descarte de las columnas derivadas para evitar problemas de multicolinealidad y optimizar el rendimiento computacional.

Eliminamos las columnas comentadas y avanzamos en nuestro estudio del conjunto de datos:

```
Prompt 4
1   Elimina todas las columnas a partir de la variable Churn sin incluir esta.
2
3   Obtener la distribución exacta de la variable objetivo Churn
4   (porcentaje Yes/No).
5
```

Como resultado, obtenemos la distribución que se muestra en la Tabla 3.1:

Churn	Frecuencia	Proporción
No	5174	73.46 %
Yes	1869	26.55 %

Tabla 3.1: Distribución de la variable objetivo Churn

Al cuantificar la relación entre las categorías, obtenemos un ratio de  $5174/1869 \approx 2,77$  a 1 a favor de la retención del cliente. Este desequilibrio evidente en los datos constituye un factor determinante que condicionará la elección de las métricas de evaluación y las estrategias de entrenamiento posteriores.

Continuamos con el estudio de la distribución de distintas variables predictoras:

```
Prompt 5
1   Para variables categóricas conviene generar tablas de frecuencia
2   (conteos y %), y gráficos (barras, boxplots de MonthlyCharges por
3   Churn, histogramas de tenure, etc.).
4
5
```

Inicialmente, se intentó ejecutar la anterior consulta. No obstante, dada la elevada carga computacional requerida para la generación simultánea de todas las representaciones gráficas, el sistema excedió sus capacidades de procesamiento, resultando en un error de ejecución. En consecuencia, se optó por generar gráficos en lotes de dos o tres variables simultáneamente. Sin embargo, su diseño no resultaba consistente,

perdiéndose la homogeneidad estética de los resultados. Ante la imposibilidad de garantizar una estandarización visual, se optó por una estrategia más controlada y metodológicamente estable: generar los gráficos de forma individual. A continuación, se presentan los resultados obtenidos en las Figuras 3.1, 3.2 y 3.3:

**Conteo de clientes por género segmentado por Churn**

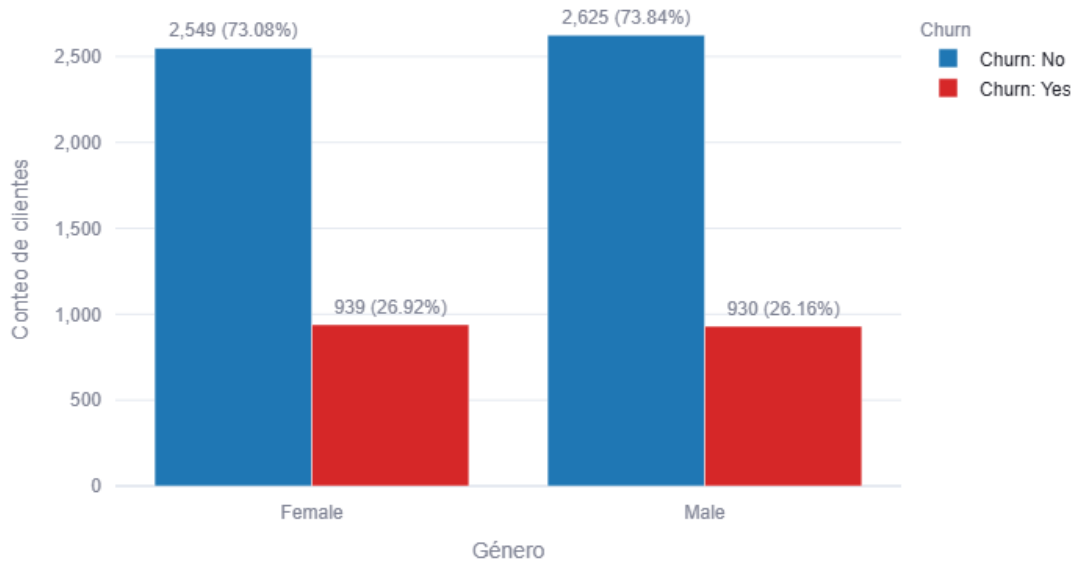


Figura 3.1: Distribución de la variable género para cada valor de Churn.

**Conteo por Partner separado por Churn**

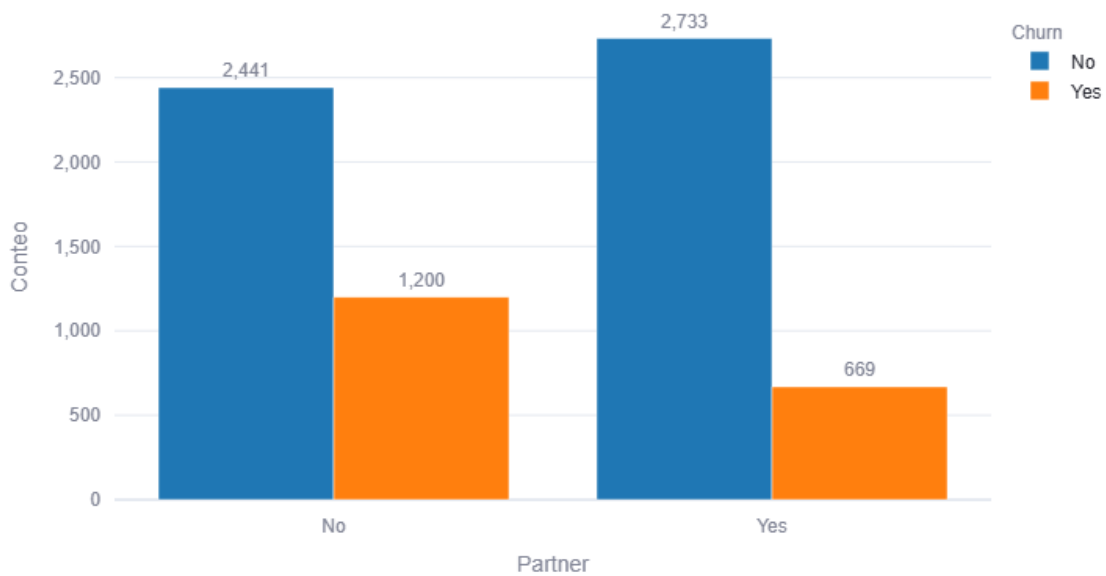


Figura 3.2: Distribución de la variable categórica Partner para cada valor de Churn.

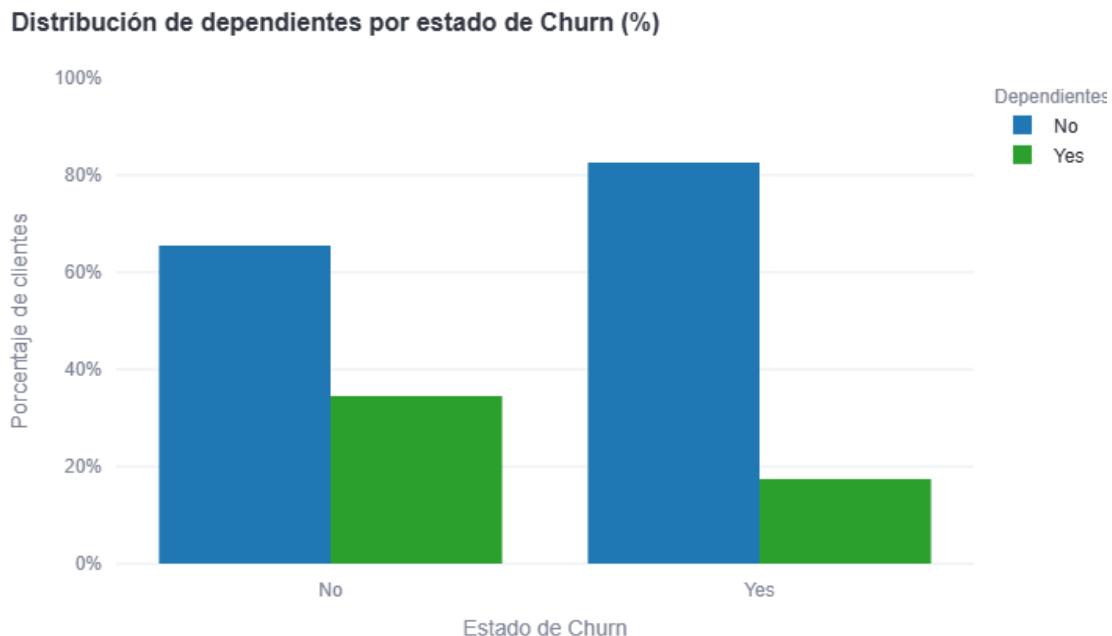


Figura 3.3: Distribución de la variable `Dependents` para cada valor de `Churn`.

La distribución del `Churn` en la Figura 3.1 es prácticamente idéntica independientemente del género de los usuarios. Estos resultados sugieren que la variable `gender` aporta escasa información discriminante y podría considerarse prescindible en fases posteriores del modelado.

En la Figura 3.2 se observa un ligero mayor abandono en usuarios solteros.

En la Figura 3.3 se tiene una distribución parecida a la variable `Partner`.

Estos gráficos se descargaron directamente desde la aplicación `Pandas-data-analyst` del repositorio [1], que incorpora esta funcionalidad en su interfaz.

Queda por estudiar las variables numéricas:

#### Prompt 6

```
1 Realiza un densityplot suave para la variable TotalCharges aplicando un
2 kernel gaussiano.
3
```

La Figura 3.4 es un gráfico de densidad de la variable `TotalCharges`. Con rectas verticales se representan tanto la media como la mediana de la variable, información que, si bien no fue solicitada, resulta relevante.

#### Prompt 7

```
1 Haz un diagrama de violín de tenure dividiendo por Churn.
2
```

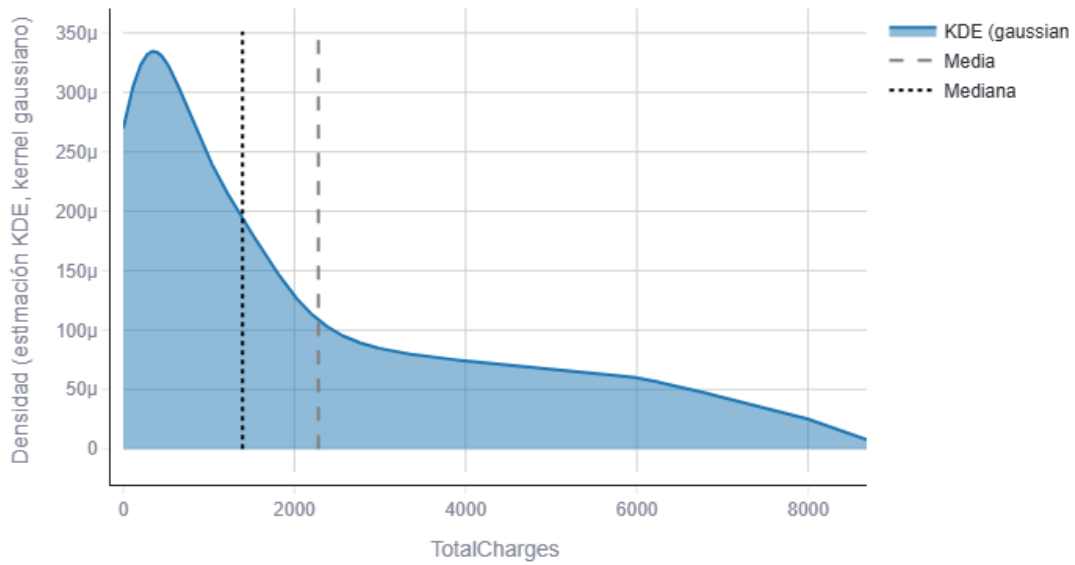
**Estimación de densidad (KDE Gaussiano) de TotalCharges**

Figura 3.4: Distribución de la variable TotalCharges

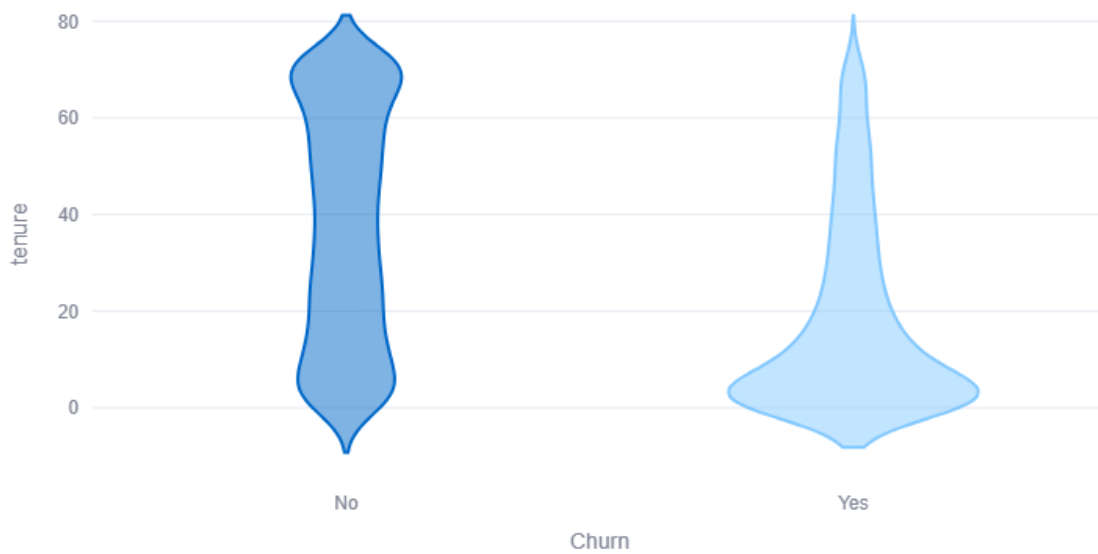
**Distribución de tenure por Churn**

Figura 3.5: Distribución de la variable tenure para cada valor de Churn

En la Figura 3.5 se observa una diferencia clara entre los clientes que no abandonan la empresa y los que sí. Los primeros presentan una distribución bastante equilibrada a lo largo del tiempo con una concentración notable cerca de los 70 meses. Por el contrario, en el grupo que sí abandona, la distribución se ensancha significativamente en la base, lo que indica que la mayor parte de las fugas de clientes ocurren en los primeros meses de suscripción. En resumen, los resultados indican que una mayor antigüedad se asocia con una menor probabilidad de abandono, mientras que los clientes nuevos son los más vulnerables a cancelar el servicio.

## Prompt 8

```
1 Realiza un diagrama de violín y caja de los cargos mensuales por abandono.  
2
```

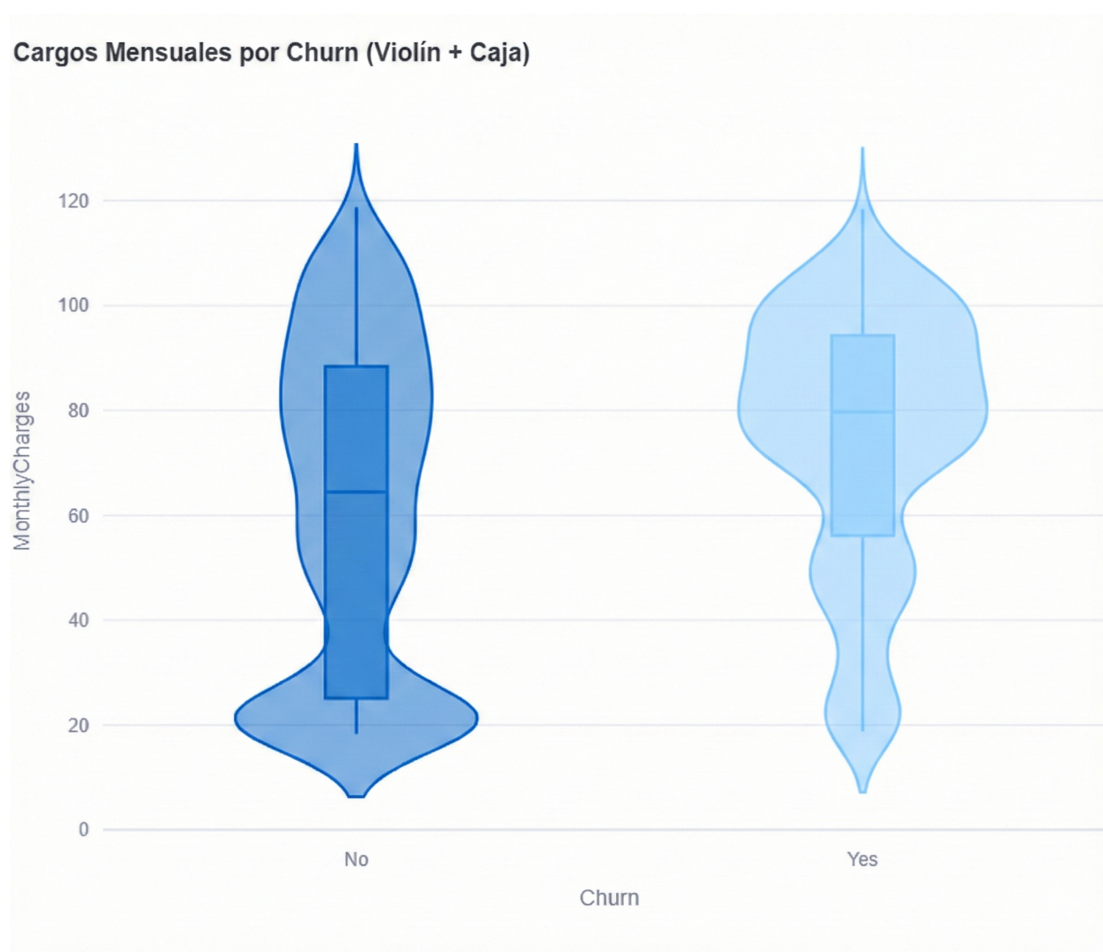


Figura 3.6: Distribución de la variable `MonthlyCharges` según su valor de `Churn`

El gráfico de la Figura 3.6 indica que existe una correlación positiva entre el coste mensual y la fuga de clientes. Mientras que el grupo de usuarios fieles presenta una fuerte concentración en tarifas bajas, los clientes que abandonan la compañía se agrupan mayoritariamente en precios más elevados. Esto sugiere que los usuarios con facturas más altas son sensibles al precio y tienen una mayor propensión a cancelar el servicio, mientras que los planes económicos favorecen la retención.

### 3.3. Correlation funnel

Para completar este estudio inicial, se aplica un análisis de tipo *Correlation Funnel*, una técnica que transforma variables categóricas y discretiza variables continuas para evaluar la asociación de sus distintos niveles o tramos con la variable objetivo, en este caso **Churn**.

#### Prompt 9

```

1 Usar la metodología de correlation funnel para hallar variables
2 correlacionadas con churn.
3

```

El agente devuelve un conjunto de datos con 58 registros compuestos por los diferentes valores de las variables categóricas del conjunto de datos original. En el caso de las variables numéricas como **TotalCharges** se han creado tramos, por ejemplo **TotalCharges** [18.8,267.07], discretizando estas variables.

A continuación, se calculó la correlación de cada tramo con la variable **Churn**. Los principales factores asociados al abandono son aquellos tramos donde la correlación con **Churn** es positiva y más fuerte.

**Contratos mensuales:** Cuando la variable **Contract** toma el valor “month-to-month” se convierte en el predictor con mayor influencia a favor de la pérdida de clientes.

**Falta de servicios de soporte:** Cuando las variables **OnlineSecurity** y **TechSupport** toman los valores “No” se identifica una tendencia al abandono. Esto sugiere que estos servicios pueden funcionar como una barrera de contención aumentando la satisfacción del cliente.

**Nuevos clientes:** La franja crítica la encontramos en los primeros 6 meses de antigüedad del usuario. En este período la tasa de abandono es del 53.7%, mientras que la del siguiente tramo desciende al 34% y sigue descendiendo al aumentar el valor de **tenure** llegando a un 6.6% en el último intervalo [60,72).

Pasamos ahora a mencionar los principales factores de retención de clientes, es decir, aquellos tramos o categorías de las variables predictoras que cuentan con una correlación negativa más fuerte con respecto al **Churn**.

Destacan los **contratos a largo plazo**. Cuando la variable **Contract** toma el valor “Two-year” la tasa de abandono desciende hasta el 2.83%. Como ya se mencionó, cuanto más longevos los clientes menor es la probabilidad de abandono. Por último, destacar los clientes **sin servicio a internet**. Los clientes que solamente consumen servicios básicos como teléfono son más fieles y estables.

A continuación se incorpora una tabla resumen de los registros mencionados en el análisis “Correlation Funnel”:

### 3.4. Ajuste de un modelo de clasificación

Este apartado se centra en el ajuste de un modelo de clasificación binaria para la predicción de *churn*. Hasta ahora, para todo el trabajo realizado con el conjunto de datos *churn\_data.csv* se utilizó el agente alojado en el repositorio [1]. Sin embargo, para este apartado tendremos que prescindir de él debido a que, en el momento del estudio, el agente no incorporaba herramientas para el ajuste de modelos.

Igualmente, previo al ajuste, hacemos uso del agente una última ocasión con objetivo de facilitar el proceso de modelado.

Variable	Categoría	Registros	Abandonos	Ratio Abandono	Correlación
Contract	Month-to-Month	3875	1655	0.4271	0.4051
OnlineSecurity	No	3498	1461	0.4177	0.3426
TechSupport	No	3473	1446	0.4163	0.3373
Tenure	[0,6)	1409	756	0.5366	0.3071
Tenure	[6,20)	1408	479	0.3402	0.0847
Tenure	[60,72)	1409	93	0.0660	-0.2258
Contract	Two-year	1695	48	0.0283	-0.3023
InternetService	No	1526	113	0.0740	-0.2279

Tabla 3.2: Correlation Funnel

### 3.4.1. Procesado de datos

A continuación, realizamos una consulta técnica orientada a ajustar un modelo de clasificación sobre nuestra muestra de entrenamiento.

Prompt 10

```

1   Crea un dataset Churn a partir del dado que esté listo para modelar:
    codificar categóricos, manejar totales/promedios
2

```

Como resultado de esta consulta obtenemos un nuevo conjunto de datos *hot\_encoding.csv* donde todas sus variables son numéricas (enteros, flotantes o booleanos). No obtenemos ningún valor nulo y se ha aplicado one-hot encoding a variables como el tipo de contrato, método de pago y servicio de internet, lo que facilita el uso de modelos de clasificación.

### 3.4.2. Discusión de resultados

Mediante la librería H2O de Python se ajustaron un gran número de modelos de clasificación a los datos proporcionados por el agente. En la siguiente tabla se resumen los resultados obtenidos:

A continuación se discuten los mejores modelos obtenidos, analizados en primer lugar según el criterio predeterminado por la herramienta: el índice AUC. Si bien esta métrica proporciona una buena medida global, no resulta la más idónea para la evaluación definitiva de nuestro problema, dado que la variable respuesta **Churn** presenta un evidente desbalanceo de clases a favor de la retención del cliente. Es por ello que el AUC tiende a mostrar una visión optimista al verse fuertemente influenciado por la facilidad del modelo para acertar el alto volumen de verdaderos negativos. Por ello, tras esta revisión inicial, el análisis profundizará empleando métricas más representativas para este contexto, como el AUCPR o Área Bajo la Curva de Precisión-Recall, que evalúa con mayor fidelidad la capacidad real de los algoritmos para detectar a la clase minoritaria, en nuestro caso, las fugas.

Modelo	AUC	LogLoss	AUCPR	Mean Error	RMSE	MSE
StackedEnsemble (BestOfFamily)	<b>0.8525</b>	0.4098	0.6661	0.2277	0.3649	0.1332
StackedEnsemble (AllModels)	0.8523	0.4098	<b>0.6683</b>	0.2282	0.3650	0.1332
GBM 1	0.8507	0.4122	0.6619	0.2246	0.3661	0.1341
GLM 1 (Regresión Logística)	0.8465	0.4171	0.6510	0.2283	0.3682	0.1356
GBM 5	0.8452	0.4186	0.6574	0.2362	0.3690	0.1361
GBM Grid 1 (modelo 1)	0.8449	0.4195	0.6568	0.2386	0.3696	0.1366
GBM 2	0.8430	0.4216	0.6525	0.2354	0.3701	0.1370
GBM 3	0.8399	0.4254	0.6455	0.2430	0.3720	0.1384
GBM 4	0.8362	0.4311	0.6367	0.2386	0.3752	0.1407
XRT 1 (Extra Rand. Trees)	0.8359	0.4571	0.6270	0.2445	0.3860	0.1490
GBM Grid 1 (modelo 2)	0.8349	0.4331	0.6344	0.2423	0.3759	0.1413
DRF 1 (Random Forest)	0.8335	0.5049	0.6328	0.2463	0.3766	0.1418

Tabla 3.3: Resultados del Leaderboard de H2O AutoML. Modelos ordenados por AUC.

El resto de métricas de la tabla se considerarán únicamente como información adicional y no se tendrán en cuenta en la selección del modelo óptimo.

En la Tabla 3.3 se posiciona como ganador por AUC el modelo `StackedEnsemble_BestOfFamily`. El “stacked ensemble” es una técnica avanzada que no constituye un algoritmo individual, sino que actúa como un consejo de expertos que agrega y pondera las predicciones de otros modelos base para emitir una decisión final más robusta. Es fundamental distinguir las dos variantes generadas por H2O que ocupan las dos primeras posiciones en la tabla: el `StackedEnsemble_AllModels` utiliza absolutamente todos los modelos entrenados durante el proceso para construir el ensamblaje, maximizando la información disponible. Por el contrario, el `StackedEnsemble_BestOfFamily` aplica una estrategia de poda, seleccionando únicamente al mejor representante de cada familia de modelos (el mejor GBM, el mejor *random forest*, etc.) para crear un modelo final más ligero y con menos redundancia.

Atendiendo a los modelos individuales, el *Gradient Boosting Machine* o GBM se consolida como el algoritmo individual más potente, con un AUC de 0.8507, a una distancia mínima de los modelos de ensemble. Resulta especialmente revelador el desempeño de la regresión logística o GLM, que con un AUC de 0.8465 compite directamente con modelos mucho más complejos. Esto sugiere que el conjunto de datos contiene relaciones lineales muy marcadas y limpias, lo que explica por qué una técnica clásica funciona casi tan bien como las avanzadas. En este experimento se excluyó el Deep Learning de la búsqueda automática dado que, tal como documentan la literatura y ejecuciones previas, los algoritmos basados en árboles superan sistemáticamente a las redes neuronales en datos tabulares estructurados. El modelo más débil del leaderboard es el DRF (*random forest*) con AUC de 0.8335, ligeramente por debajo del XRT (Extra Randomized Trees) con 0.8359.

Conviene aclarar que las denominaciones GBM 1, GBM 2, etc., no hacen referencia a modelos conceptualmente distintos diseñados manualmente, sino a diferentes instancias de la misma familia *Gradient Boosting Machine* generadas automáticamente por H2O AutoML durante el proceso de entrenamiento. La herramienta prueba varias combinaciones de hiperparámetros, como la profundidad de los árboles, el número de iteraciones, la tasa de aprendizaje o los niveles de muestreo de filas y columnas, y evalúa cada configuración mediante validación cruzada interna. Como resultado, H2O incorpora al *leaderboard* varios GBM candidatos y los identifica de forma secuencial como GBM 1, GBM 2, etc., según el orden interno con el que han sido registrados. Por tanto, estos modelos representan alternativas del mismo algoritmo ajustadas de manera automática sobre los datos del problema, y sus ligeras diferencias de rendimiento reflejan el efecto de esa búsqueda de configuraciones sobre la capacidad predictiva final.

Como mencionábamos previamente, dada la naturaleza desbalanceada de la variable objetivo *Churn*, la métrica AUC puede verse influenciada por la gran presencia de verdaderos negativos. Por ello, para una evaluación más rigurosa, se ha priorizado el análisis del AUCPR. Esta métrica aísla la capacidad del modelo para detectar correctamente la clase minoritaria, penalizando los falsos positivos. Bajo este punto de vista, se observa que de los modelos de ensemble ahora es el `AllModels` el que ofrece una capacidad ligeramente superior en la detección de abandonos (AUCPR = 0.6683) frente al ganador por AUC (AUCPR = 0.6661 del `BestOfFamily`), lo que sugiere que maximizar el número de modelos base mejora marginalmente la detección de la clase minoritaria a costa de mayor coste computacional.

Al aplicar el criterio del AUCPR a los modelos individuales se observan resultados relevantes. La regresión logística presenta un comportamiento competitivo, con un AUCPR de 0.6510, superando incluso a algunos modelos más complejos, como el segundo GBM Grid o GBM de rejilla, con un AUCPR de 0.6344. Este resultado sugiere que el conjunto de datos contiene estructura suficiente para que un modelo lineal bien especificado ofrezca un rendimiento comparable al de alternativas no lineales más costosas. En el extremo inferior se sitúan el XRT y el DRF, que, pese a ser modelos de tipo ensamblado de árboles, presentan los AUCPR más bajos entre los modelos individuales (0.6270 y 0.6328 respectivamente).

Igualmente, el mejor modelo individual es un GBM que, bien ajustado, es casi tan efectivo como combinar

un gran número de modelos, pero mucho más barato de poner en producción y más fácil de interpretar, por lo que constituye la alternativa óptima para este problema.

A continuación se incluye un análisis individual de los modelos GBM, GLM y `StackedEnsemble_BestOfFamily`. Las imágenes de las próximas tres secciones son proporcionadas por la librería H2O de Python. A partir de estas podremos extraer conclusiones específicas de cada uno de los modelos que nos servirán para obtener una conclusión del problema Churn.

### 3.4.3. Modelo Gradient Boosting

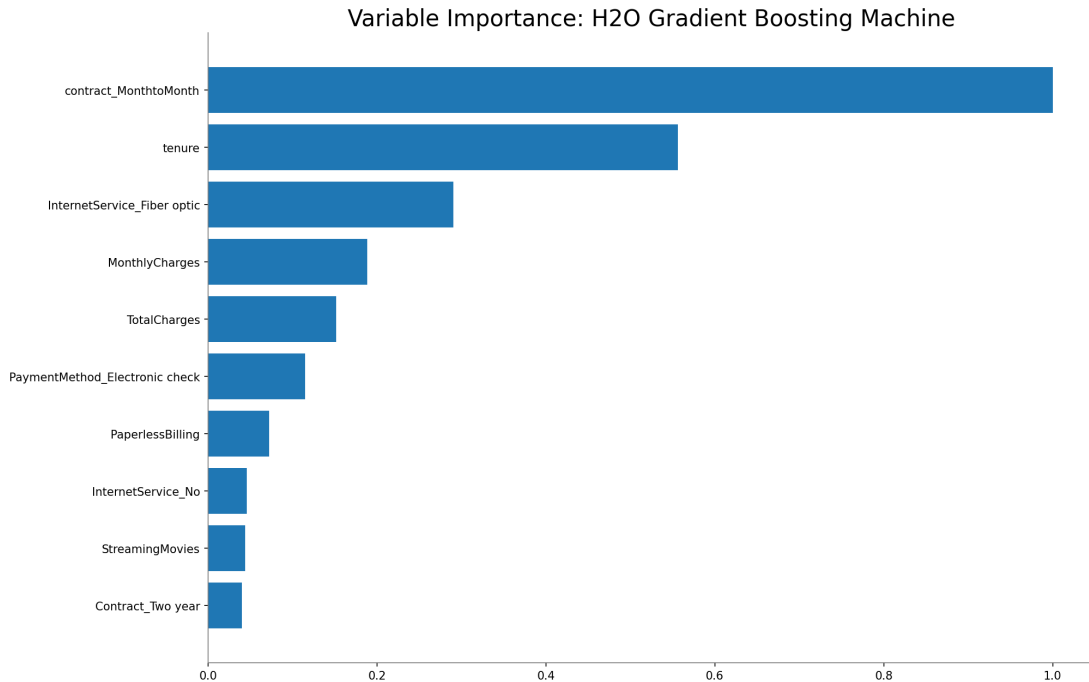


Figura 3.7: Importancia de las variables predictoras en el modelo GBM

La Figura 3.7 es el resultado del análisis de importancia de las variables predictoras en el modelo *Gradient Boosting Machine*. Revela que el tipo de contrato mensual o variable `contract_MonthtoMonth` es, con gran diferencia, el factor más determinante a la hora de predecir la fuga, seguido por la antigüedad del cliente (`tenure`), el tipo de conexión a internet (`InternetService_Fiber optic`) y los cargos mensuales (`MonthlyCharges`). Este primer gráfico nos indica qué características estructuran las decisiones del algoritmo, confirmando que la retención a lo largo del tiempo y las condiciones del servicio son los pilares del comportamiento del usuario.

Para complementar el análisis de importancia y estudiar la direccionalidad del efecto de cada variable, se aplica el método SHAP o *SHapley Additive exPlanations*. En el contexto del GBM, los valores SHAP se obtienen mediante las contribuciones marginales de cada árbol del ensemble, por lo que son consistentes con las predicciones del modelo. El resultado es el gráfico de la Figura 3.8, donde cada punto representa un cliente del conjunto de test.

El eje horizontal representa el valor SHAP, es decir, la contribución de esa variable a la predicción de la fuga para cada cliente concreto. Un valor positivo desplaza la predicción hacia el abandono, mientras que uno negativo, hacia la retención. El color codifica el valor de la propia variable: rojo indica un valor alto y azul un valor bajo. La lectura del gráfico permite extraer conclusiones directas sobre la direccionalidad de cada predictor.

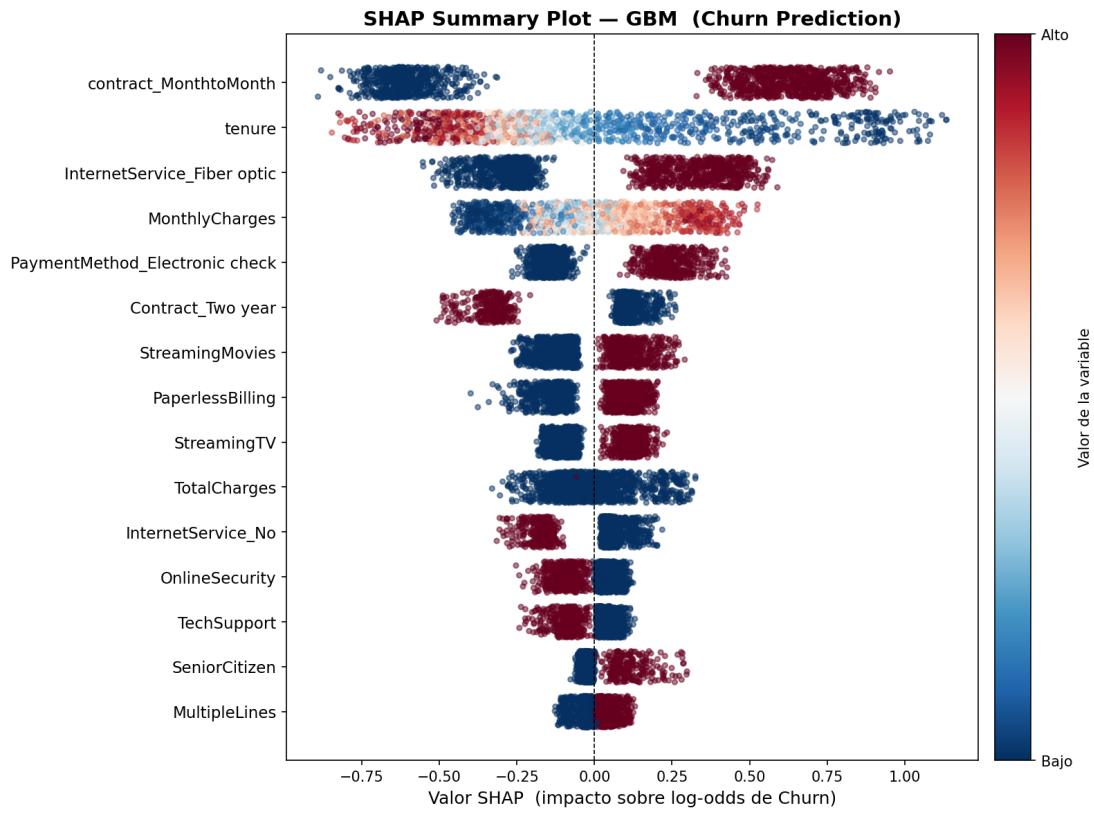


Figura 3.8: Gráfico SHAP del modelo GBM

En primer lugar, `contract_MonthtoMonth` muestra una nube roja concentrada en valores SHAP altamente positivos, lo que confirma que los clientes con contrato mes a mes tienen una probabilidad de *churn* notablemente superior. De forma simétrica, `tenure` y `Contract_Two_year` presentan nubes azules desplazadas a la izquierda, indicando que la antigüedad elevada y los compromisos de larga duración actúan como factores protectores de la retención. La variable `InternetService_Fiber_optic` exhibe puntos rojos en la región positiva del eje, confirmando que la fibra óptica se asocia a mayor riesgo de abandono, posiblemente debido a una mayor sensibilidad al precio o a expectativas de calidad más exigentes. Por último, el pago mediante cheque electrónico (`PaymentMethod_Electronic_check`) se vincula igualmente a valores SHAP positivos, coherente con perfiles de clientes menos comprometidos o con mayor propensión al cambio.

Además de la representación gráfica anterior, la Tabla 3.4 recoge los parámetros estructurales del árbol y su interpretación técnica sobre la capacidad, regularización y robustez del modelo.

Parámetro GBM	Valor	Interpretación técnica
<code>ntrees</code>	39	Número de árboles aditivos del ensemble. Controla la capacidad total del modelo; más árboles incrementan expresividad y coste computacional.
<code>max_depth</code>	15	Profundidad máxima por árbol. Permite capturar interacciones no lineales de alto orden, pero aumenta el riesgo de sobreajuste si no se regulariza.
<code>learn_rate</code>	0.1	Factor de contracción de cada árbol nuevo. Un valor moderado estabiliza el aprendizaje secuencial y reduce oscilaciones en la optimización.
<code>sample_rate</code>	0.8	Submuestreo de filas por iteración ( <i>stochastic gradient boosting</i> ). Introduce ruido controlado y mejora la generalización.
<code>col_sample_rate</code>	0.8	Submuestreo de columnas por árbol. Reduce correlación entre árboles y mitiga dependencia excesiva de variables dominantes.
<code>min_rows</code>	100	Mínimo de observaciones por hoja terminal. Impide particiones demasiado finas y actúa como regularizador estructural fuerte.

Tabla 3.4: Parámetros de árboles del GBM e interpretación técnica

A primera vista, la configuración puede sorprender: la formulación teórica del *gradient boosting* suele asociarse a conjuntos de árboles muy superficiales con un número elevado de iteraciones, donde cada árbol corrige pequeños errores residuales del anterior. Sin embargo, H2O AutoML no impone ningún supuesto teórico; explora mediante búsqueda en rejilla un amplio espacio de configuraciones y selecciona la que minimiza el error en validación cruzada interna. Si el modelo con `max_depth = 15` y únicamente 39 árboles ganó la comparación, es porque resultó la opción con mejor generalización para este conjunto de datos concreto.

La clave reside en que profundidad elevada y número reducido de árboles no son contradictorios cuando el modelo incorpora regularización suficientemente fuerte. El doble submuestreo estocástico (`sample_rate = 0.8` y `col_sample_rate = 0.8`) introduce variabilidad aleatoria en cada árbol de forma análoga a *random forest*, limitando el sobreajuste pese a la profundidad. Por su parte, `min_rows = 100` impide que cualquier hoja terminal contenga menos de 100 observaciones: aunque la profundidad máxima permitida sea 15, el árbol deja de ramificarse en cuanto un nodo no alcanza ese umbral, produciendo en la práctica estructuras

considerablemente más simples que la profundidad nominal sugiere. A ello se suma la contracción moderada (`learn_rate = 0.1`), que estabiliza el aprendizaje secuencial sin exigir centenares de iteraciones para converger. El resultado es un ensemble compacto cuyo buen poder discriminante en test confirma que la configuración seleccionada por AutoML es apropiada para el problema `Churn`.

Finalmente, se incorpora la matriz de confusión del modelo comparando las predicciones realizadas con sus valores reales en la muestra de validación:

Real / Predicción	1 (Sí Churn)	0 (No Churn)
1 (Sí Churn)	265	93
0 (No Churn)	258	784

Tabla 3.5: Matriz de Confusión GBM con umbral de decisión 0.290

La matriz de confusión revela la capacidad operativa del modelo para el negocio. En la diagonal de aciertos, el sistema ha clasificado correctamente a 784 clientes fieles o verdaderos negativos (TN) y ha detectado con éxito a 265 usuarios en riesgo real de fuga o verdaderos positivos (TP), permitiendo estudiar su situación y actuar sobre ellos. Analizando los errores, encontramos 93 falsos negativos (FN), que representan el caso más preocupante para la empresa pues son clientes que abandonan la compañía sin ser detectados. Por otro lado, los 258 falsos positivos (FP) corresponden a clientes que no pensaban irse pero fueron marcados como riesgo. Aunque esto implica un gasto en acciones de marketing innecesarias, desde una perspectiva de negocio se considera un mal menor frente a la pérdida definitiva del cliente.

El umbral de decisión del pie de tabla hace referencia a la probabilidad de pertenencia a la clase positiva, en nuestro problema el abandono del servicio. Es decir, el modelo clasificará como fuga potencial a cualquier cliente cuya probabilidad calculada de abandono sea igual o superior al 29%. Reducir este umbral respecto al estándar habitual del 50% permite aumentar la sensibilidad del modelo.

Este valor no se fija manualmente, sino que H2O lo determina de forma automática durante el entrenamiento. Al finalizar la validación cruzada interna, la librería evalúa la curva de precisión-exhaustividad (*precision-recall*) sobre las predicciones fuera de muestra y selecciona el punto de corte que maximiza el índice F1. Dado que el conjunto de datos presenta desbalance de clases en la variable `Churn`, el umbral óptimo para F1 tiende a situarse por debajo del 50%: bajarlo permite al modelo capturar más positivos reales a costa de aceptar un mayor número de falsos positivos, lo cual es precisamente el equilibrio que maximiza la media armónica entre precisión y recall.

En este caso obtenemos un índice F1 del 0.6016, lo que indica un rendimiento razonable para un modelo de clasificación con clases desbalanceadas.

#### 3.4.4. Modelo de regresión logística

La Figura 3.9 representa las magnitudes de los coeficientes estandarizados del modelo GLM y utiliza el color para indicar su signo: azul para efectos positivos sobre el `Churn` y naranja para efectos negativos. La variable con mayor peso absoluto es `tenure`, con un coeficiente negativo muy superior al resto, lo que indica que una mayor antigüedad reduce de forma clara la probabilidad de abandono. A continuación aparecen `InternetService_Fiber optic` y `contract_MonthtoMonth`, ambas con signo positivo, por lo que se comportan como factores de riesgo relevantes. Entre los efectos protectores también destacan `Contract_Two year` e `InternetService_No`, mientras que `PaperlessBilling`, `MonthlyCharges` y `PaymentMethod_Electronic check` presentan contribuciones positivas de intensidad intermedia. En conjunto, la imagen muestra que la permanencia del cliente y los contratos estables son los principales frenos al abandono, mientras que la fibra óptica, el contrato mes a mes y ciertas condiciones de facturación empujan

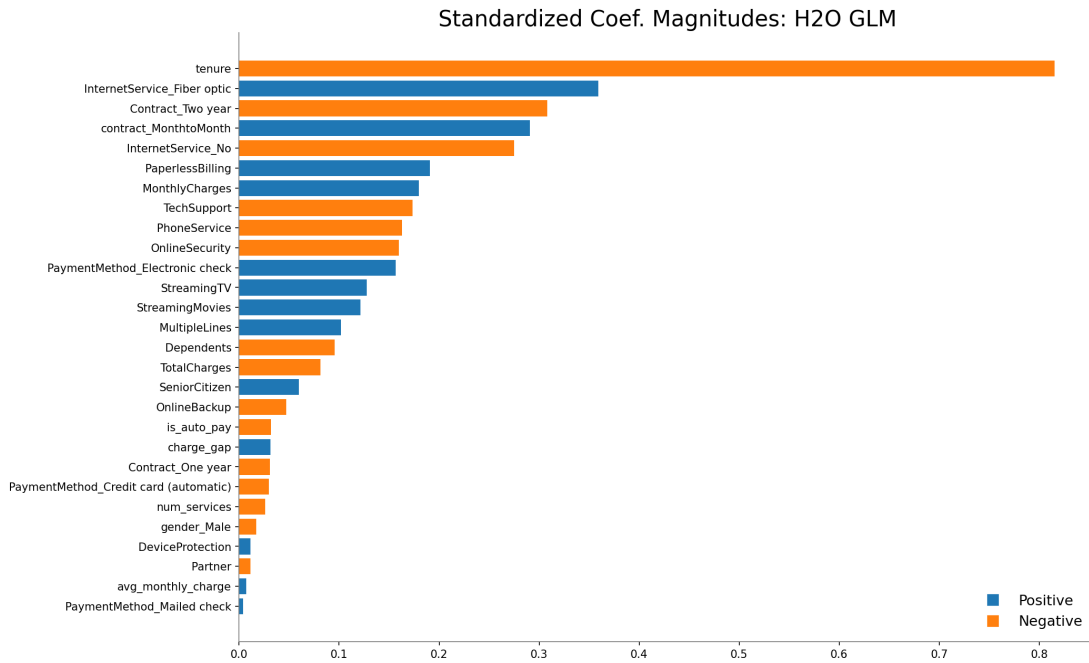


Figura 3.9: Coeficientes estandarizados del modelo de regresión logística

la predicción hacia la fuga.

Además de la importancia de las variables podemos representar los coeficientes  $\beta$  asociados a cada variable predictora, ya que cuantifican dirección e intensidad del efecto sobre el **Churn**. En este bloque se incluye también el intercepto del modelo, que representa el nivel base de la *log-odds* cuando el resto de predictores se fija en cero. La Figura 3.10 resume visualmente estos coeficientes.

Su interpretación es análoga a la realizada anteriormente en la Figura 3.9.

En la Tabla 3.6 se muestran los valores numéricos exactos de cada coeficiente  $\beta$  estimado por el GLM. Coeficientes positivos incrementan la propensión al abandono, mientras que coeficientes negativos actúan como factores protectores.

La interpretación de los resultados de esta matriz es análoga a la del modelo anterior, aunque presenta matices importantes. El principal es la reducción de los falsos negativos a 83 (frente a 93 del GBM), un detalle relevante ya que minimiza el volumen crítico de fugas silenciosas. Como contrapartida a este aumento en la capacidad de captura, se observa un incremento en los falsos positivos hasta 288 y una ligera disminución en el número total de aciertos globales. Observamos que esto puede venir ocasionado por la reducción del umbral de decisión a 0.268, creando un modelo más agresivo en la detección del **Churn** que el GBM.

En este caso obtenemos un F1 del 0.5972. Un valor razonable, aunque ligeramente menor al anterior modelo.

### 3.4.5. Modelo `StackedEnsemble_BestOfFamily`

La Figura 3.11 representa los coeficientes estandarizados del meta-modelo que combina las predicciones de los modelos base dentro del *StackedEnsemble*. Por ello, los valores del eje X deben interpretarse como intensidad del efecto de cada modelo sobre la predicción final del ensamblaje: cuanto mayor es el valor,

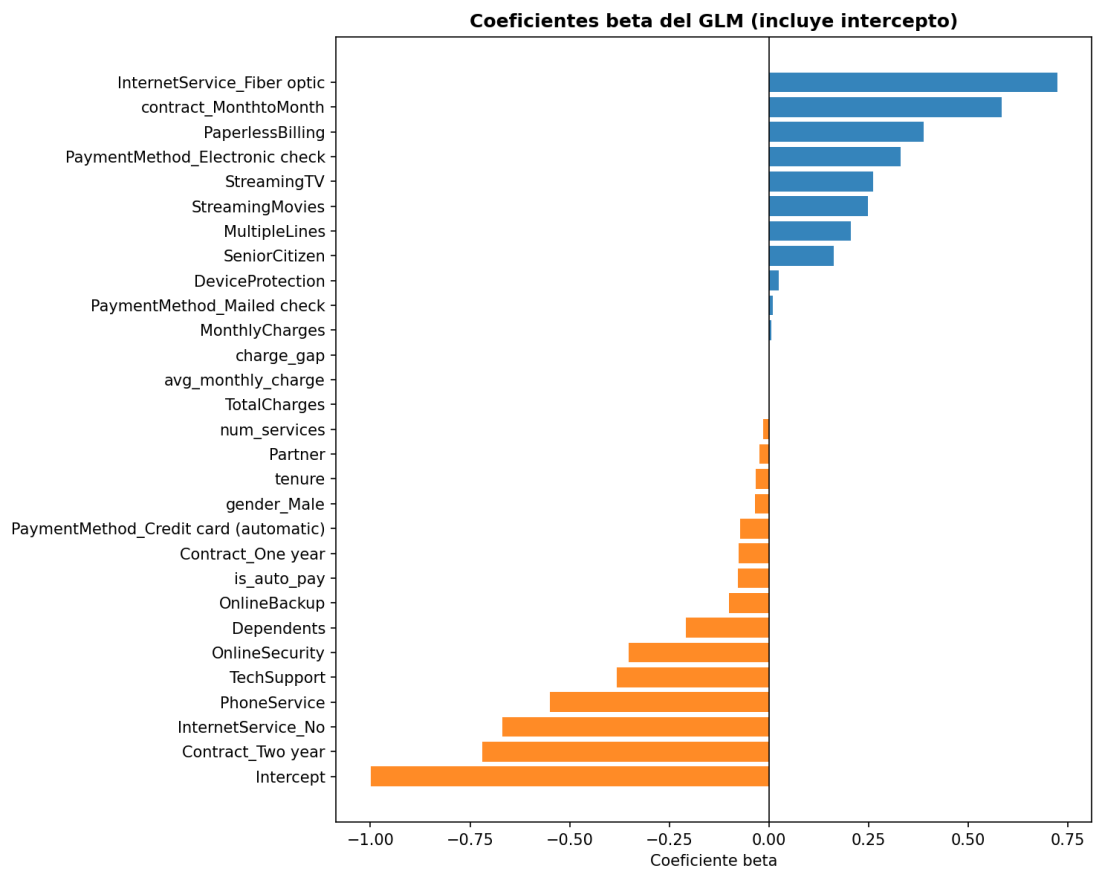


Figura 3.10: Coeficientes  $\beta$  del GLM incluyendo el intercepto

Término del modelo	$\beta$	Término del modelo	$\beta$
Intercept	-0.999691	TotalCharges	-1.54538e-06
Contract_Two year	-0.719406	avg_monthly_charge	4.38216e-07
InternetService_No	-0.669165	charge_gap	7.56937e-05
PhoneService	-0.549296	MonthlyCharges	0.005992
TechSupport	-0.382024	PaymentMethod_Mailed check	0.010006
OnlineSecurity	-0.351887	DeviceProtection	0.024278
Dependents	-0.208082	SeniorCitizen	0.162657
OnlineBackup	-0.099069	MultipleLines	0.206490
is_auto_pay	-0.077186	StreamingMovies	0.248170
Contract_One year	-0.075688	StreamingTV	0.261887
PaymentMethod_Credit card (automatic)	-0.072712	PaymentMethod_Electronic check	0.331082
gender_Male	-0.034264	PaperlessBilling	0.388386
tenure	-0.033236	contract_MonthtoMonth	0.583991
Partner	-0.022770	InternetService_Fiber optic	0.723743
num_services	-0.014173		

Tabla 3.6: Coeficientes  $\beta$  estimados por el GLM (incluye intercepto), divididos en dos bloques para mejorar legibilidad

Real / Predicción	1 (Sí Churn)	0 (No Churn)
1 (Sí Churn)	275	83
0 (No Churn)	288	754

Tabla 3.7: Matriz de Confusión GLM con umbral 0.268

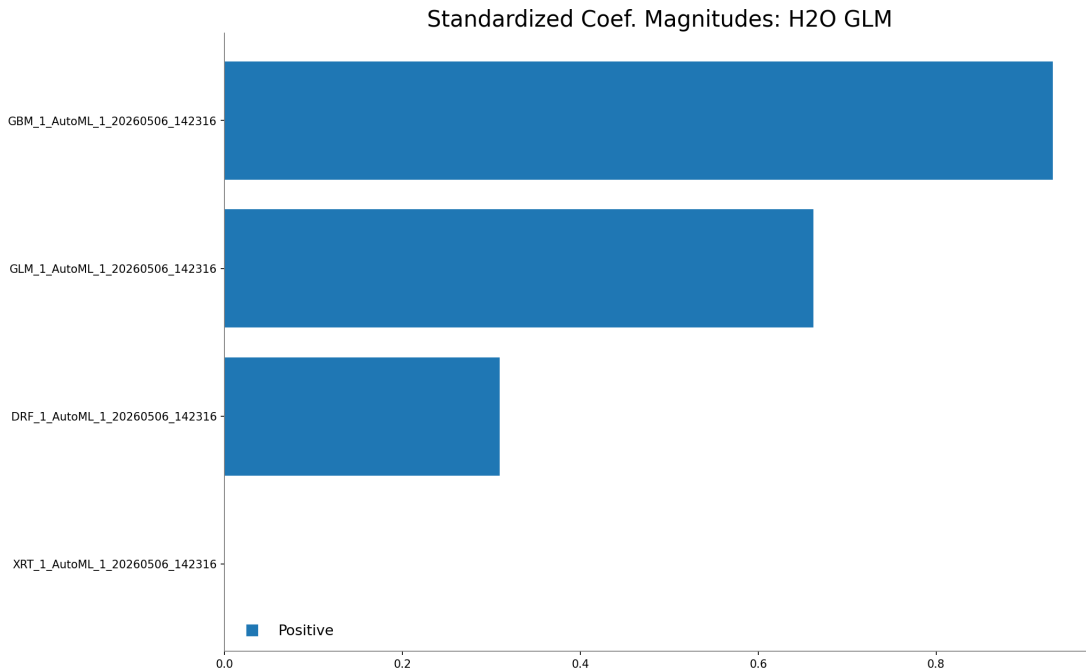


Figura 3.11: Ponderación de modelos base en el StackedEnsemble\_BestOfFamily

mayor es la contribución de ese modelo a empujar la decisión final hacia la clase positiva cuando sus propias predicciones aumentan. En este caso, el GBM aparece como el componente dominante, el GLM ocupa una segunda posición claramente relevante, el DRF aporta una señal muy pequeña y el XRT queda prácticamente anulado.

Real / Predicción	1 (Sí Churn)	0 (No Churn)
1 (Sí Churn)	240	118
0 (No Churn)	198	844

Tabla 3.8: Matriz de Confusión StackedEnsemble\_BestOfFamily con umbral 0.362

Este modelo cuenta con un umbral más alto, de 0.362. Observamos un punto de equilibrio estratégico frente a configuraciones más sensibles, como el umbral de 0.290 del GBM. Al incrementar la exigencia probabilística para clasificar a un cliente como fuga, el modelo reduce el volumen de falsos positivos a 198, lo que se traduce en un ahorro directo en costes de campañas de retención innecesarias y evita erosionar el margen de beneficio en clientes ya fidelizados. Como contrapartida esperable, la sensibilidad del sistema disminuye ligeramente, permitiendo que las fugas silenciosas o falsos negativos aumenten hasta 118. En definitiva, mientras que el umbral de 0.290 es ideal para una estrategia de retención agresiva, este umbral de 0.362 representa una alternativa más conservadora y eficiente para escenarios donde el presupuesto de marketing esté más ajustado.

El valor de F1 asociado a este modelo es de 0.6030, mínimamente superior al modelo GBM (0.6016). Como analizábamos en pasos anteriores, una mejora prácticamente nula para el coste computacional y la pérdida de interpretabilidad que conlleva combinar modelos con la técnica de ensamblaje. Es por ello que la conclusión aplicando la métrica F1 es la misma que con el AUCPR, el modelo *Gradient Boosting* es el óptimo para nuestro problema Churn.



## Capítulo 4

# Observaciones y conclusiones finales

Los resultados obtenidos en los capítulos anteriores sugieren que este repositorio posee la capacidad necesaria para agilizar y optimizar flujos de trabajo, sirviendo de apoyo sustancial en procesos de ciencia de datos. No obstante, durante la fase experimental se han identificado ciertos aspectos críticos que afectan a la eficiencia y fiabilidad de la herramienta. A continuación, se detallan las limitaciones técnicas y operativas encontradas durante la interacción con el agente.

### 4.1. Persistencia del contexto

Una de las carencias más notables es la ausencia de un seguimiento efectivo en la memoria de la conversación. El agente con el que se realizó el análisis pre-modelado no retiene información crítica sobre las decisiones tomadas en mensajes anteriores, incluso dentro de una misma sesión de trabajo.

Esto resulta contraintuitivo y subóptimo, obligando al usuario a reiterar instrucciones previas en cada nueva consulta. Un ejemplo claro de esta limitación se evidenció durante la limpieza de datos: en repetidas iteraciones fue necesario especificar que la variable `TotalCharges` debía ser tratada como numérica, lo cual ralentiza el proceso.

### 4.2. Eficiencia computacional en la generación de gráficos

Se han observado tiempos de ejecución significativamente altos en la ejecución de tareas de visualización de datos. El tiempo de respuesta para generar gráficos es desproporcionadamente alto en comparación con la complejidad de la tarea solicitada.

Esta ineficiencia sugiere una mala gestión en la construcción de las funciones de representación por parte del agente. Es probable que esté reescribiendo o recargando librerías de forma redundante en cada ejecución. En un entorno de producción o de análisis ágil, estos tiempos de espera interrumpen el flujo de pensamiento del analista, convirtiendo una tarea a priori simple en un proceso tedioso y pesado.

### 4.3. Precisión conceptual: histograma y gráfico de densidad

Desde el punto de vista técnico, se han detectado imprecisiones terminológicas y funcionales graves en la generación de visualizaciones estadísticas. El agente demuestra dificultades para diferenciar entre un histograma y un gráfico de densidad. Se tuvo que insistir en aspectos como el uso de un kernel gaussiano

o la inclusión del término “suave” para que identificase correctamente el tipo de gráfico que se buscaba.

En varias ocasiones, al solicitar explícitamente un gráfico de densidad para visualizar la distribución de una variable continua, el agente devolvió un histograma de frecuencias. Este tipo de error conceptual requiere una supervisión constante por parte del usuario experto para asegurar que la representación gráfica se corresponde con la naturaleza estadística deseada, restando autonomía a la herramienta.

La aparición de este tipo de errores parece estar relacionada, en parte, con el modelo de lenguaje utilizado en cada consulta. Los modelos más capaces tienden a resolver con mayor precisión este tipo de distinciones técnicas. Por ello, esta limitación no debe atribuirse únicamente a la arquitectura del agente, sino también a las capacidades del modelo usado. La precisión en la distinción de conceptos estadísticos sutiles depende directamente de la densidad de parámetros y del entrenamiento especializado del modelo, por tanto, el rendimiento del agente es condicionado en gran medida por la inteligencia del motor que lo impulsa, transformando lo que debería ser una instrucción directa en un problema de interpretación semántica por parte de un modelo con un conocimiento técnico limitado.

#### 4.4. Consistencia

Un aspecto crítico para la reproducibilidad científica es la consistencia de los resultados. Se ha detectado una falta de esta en las respuestas del agente: ante un mismo *prompt* introducido en momentos distintos, el agente ha generado resultados o códigos diferentes.

Esta variabilidad representa un riesgo en flujos de trabajo de ciencia de datos, donde se espera que una misma operación de transformación o análisis arroje siempre el mismo resultado. La falta de consistencia obliga a realizar validaciones adicionales para asegurar que las conclusiones no varían arbitrariamente entre ejecuciones.

#### 4.5. Dificultad para trabajar con varios conjuntos de datos

Otra deficiencia crítica detectada reside en la incapacidad del sistema para gestionar múltiples conjuntos de datos de forma simultánea. La herramienta se encuentra restringida a la carga de un único conjunto de datos por sesión, lo que impide realizar operaciones cruzadas o comparativas de manera fluida.

A esta limitación se suma una gestión deficiente de la persistencia. Igual que ocurría con los mensajes pasados, el agente no guarda en memoria los archivos generados durante el proceso de transformación. Como consecuencia, el sistema pierde la trazabilidad de su propio flujo de trabajo, siendo incapaz de identificar o recuperar versiones previas de los datos cuando se hace referencia a ellas en instrucciones posteriores. Esto obliga al usuario a reintroducir información de manera redundante, fragmentando la continuidad del análisis e interrumpiendo el flujo de trabajo de forma innecesaria.

Desde el punto de vista metodológico, esta falta de trazabilidad también debilita la reproducibilidad del estudio. Cuando se encadenan varias transformaciones sobre un mismo conjunto de datos, resulta esencial poder reconstruir con claridad qué operación produjo cada archivo intermedio, bajo qué criterio se creó una variable derivada y en qué momento se descartó o sustituyó una versión previa del conjunto de datos. Al no conservar adecuadamente esta información, el sistema dificulta la auditoría del proceso analítico y obliga a documentar externamente decisiones que, idealmente, deberían quedar integradas en el propio flujo de trabajo.

## 4.6. Omisión del análisis de colinealidad

Otra limitación observada tiene que ver con la escasa capacidad del sistema para anticipar necesidades metodológicas a partir de los archivos con los que trabaja. A lo largo del estudio con *churn\_data.csv*, el agente fue capaz de ejecutar instrucciones concretas y producir transformaciones útiles, pero no mostró iniciativa para detectar que, antes de ajustar un modelo, era recomendable estudiar la colinealidad entre las variables predictoras. Esta omisión es especialmente relevante pues el propio flujo de trabajo generó variables derivadas, codificaciones *dummy* y representaciones alternativas de una misma información, aumentando el riesgo de redundancias y dependencias lineales.

Este episodio no debe interpretarse como un fallo aislado, sino como un indicio de una carencia más general: la falta de priorización metodológica. El agente responde con solvencia razonable cuando se le indican tareas concretas, pero no demuestra un criterio sólido para ordenar qué comprobaciones deben preceder a otras dentro de un análisis estadístico completo. En un problema de modelado, revisar la colinealidad, validar la codificación de variables o estudiar el equilibrio entre clases no son pasos opcionales, sino validaciones estructurales que deberían emerger de manera natural antes de pasar al ajuste final.

Esta ausencia de jerarquización afecta especialmente a tareas encadenadas. El sistema resulta más eficaz cuando trabaja sobre instrucciones cerradas y unitarias, pero pierde solidez cuando debe mantener continuidad entre exploración, transformación, validación y modelado. En consecuencia, su comportamiento se aproxima más al de un asistente reactivo para subtareas independientes que al de un agente capaz de sostener con coherencia un flujo de trabajo completo de ciencia de datos, que es lo que se busca actualmente con este tipo de herramientas.

## 4.7. Falsa sensación de corrección y falta de autonomía

Otro problema menos evidente pero igualmente importante es la facilidad con la que una respuesta bien redactada puede transmitir una apariencia de corrección técnica que no siempre se corresponde con la solidez metodológica real de lo producido. El agente genera explicaciones fluidas y conclusiones expresadas con seguridad, pero esa calidad de discurso puede inducir a aceptar como válidas decisiones analíticas que todavía requerían comprobaciones adicionales. En consecuencia, el usuario no solo debe verificar si la salida funciona, sino también si el razonamiento estadístico es verdaderamente adecuado.

Este fenómeno, junto a otros aspectos ya mencionados en este capítulo, incrementa de manera directa el coste real de supervisión humana. Aunque el agente reduce parte del esfuerzo mecánico asociado a escribir código o resumir resultados, obliga a invertir tiempo adicional en revisar archivos generados, reformular instrucciones, contrastar interpretaciones y confirmar que los pasos ejecutados respetan una secuencia metodológica válida. Por tanto, la ganancia de productividad se ve afectada: en muchas fases del análisis, la automatización aparente se desplaza hacia una tarea continua de validación experta.

## 4.8. Conclusión

Todas las carencias que se fueron mencionando refuerzan una conclusión más general sobre la madurez del repositorio analizado. Se trata de una herramienta en desarrollo, útil como apoyo puntual, pero todavía lejos del comportamiento que ya ofrecen otras soluciones más avanzadas, capaces de inspeccionar archivos locales, relacionar automáticamente distintos artefactos y proponer modificaciones con mayor autonomía. Frente a esas alternativas, el agente estudiado sigue funcionando de forma principalmente reactiva: responde cuando se le indica qué hacer, pero le cuesta identificar por sí mismo comprobaciones estadísticas básicas o pasos metodológicos necesarios para sostener un flujo de trabajo competitivo y robusto.

Por todo ello, no resultaría aconsejable integrar este sistema, al menos en su estado actual, en un entorno de trabajo corporativo donde la trazabilidad, la reproducibilidad, la estabilidad operativa y la fiabilidad me-

todológica son requisitos básicos y no elementos deseables. Más que un componente listo para incorporarse a un flujo profesional crítico, el repositorio debe entenderse por ahora como una herramienta experimental con valor exploratorio, pero todavía insuficientemente madura para asumir responsabilidades dentro de procesos corporativos exigentes.

No obstante, pese a las limitaciones descritas, el estudio realizado también permite identificar un espacio de utilidad concreto para este tipo de herramienta. El repositorio resulta aprovechable como interfaz conversacional para explorar rápidamente alternativas de análisis, sintetizar resultados intermedios y acelerar la documentación del trabajo realizado. Su mayor aportación no reside en sustituir el criterio técnico del analista, sino en reducir la fricción operativa en fases donde conviene iterar con rapidez sobre ideas, transformaciones o consultas bien delimitadas.

Este diagnóstico encuentra respaldo en una investigación reciente del campo. MLR-Bench [9], presentado en NeurIPS 2025, es un *benchmark*<sup>1</sup> diseñado específicamente para medir y comparar las capacidades de agentes de IA en tareas abiertas de investigación en *Machine Learning*. Sus resultados son reveladores: aunque los modelos evaluados demuestran una competencia razonable en la generación de ideas y en la redacción de textos bien estructurados, esta capacidad muestra limitaciones claras en la fase de experimentación. En más del 80% de los casos analizados, los agentes de codificación generaron resultados fabricados o no validados empíricamente, priorizando la completitud formal de la salida sobre la veracidad del contenido. Adicionalmente, muchas de las ideas propuestas por los agentes resultaron ser combinaciones superficiales de métodos existentes, sin una motivación profunda ni un impacto científico genuino. Este conjunto de fallos evidencia que la capacidad generativa de los LLMs no implica, por sí misma, capacidad crítica ni rigor metodológico.

En cuanto a la evaluación de estos sistemas, MLR-Bench introduce MLR-Judge, un *framework* de evaluación automática que emplea LLMs en el rol de juez para valorar la calidad de los resultados generados. Este enfoque se alinea con el *LLM as judge* descrito previamente en la Sección 2.8.2. En su validación, los autores comparan la diferencia entre las puntuaciones de un juez LLM y revisores humanos expertos con la diferencia observada entre dos revisores humanos independientes. Los resultados muestran que ambas discrepancias son estadísticamente comparables. En términos prácticos, esto sugiere que MLR-Judge puede utilizarse como una herramienta útil para evaluar agentes de forma sistemática y escalable, aunque no deba interpretarse como un sustituto total del juicio experto.

Un aspecto especialmente relevante para este trabajo es la rapidez con la que evoluciona el campo. Los modelos evaluados en MLR-Bench, como o4-mini, Gemini-2.5-Pro-Preview y Claude-3.7-Sonnet, figuraban como los más avanzados en 2025, pero a la fecha de redacción de esta memoria ya han sido sustituidos o ampliamente superados por generaciones posteriores.

Este hecho no debilita las conclusiones del estudio, sino que más bien refuerza su alcance. Si incluso modelos de referencia mostraban limitaciones importantes en términos de fiabilidad experimental y solidez metodológica, cabe pensar que el problema responde a una dificultad más estructural que puntual.

En este sentido, aunque el progreso reciente ha sido notable y ya existen herramientas eficaces en tareas acotadas, la autonomía robusta en flujos completos de ciencia de datos aún no puede considerarse un objetivo resuelto de forma general.

---

<sup>1</sup>Un *benchmark* es un marco estandarizado de pruebas que permite evaluar y comparar, bajo criterios homogéneos, el rendimiento de distintos modelos o sistemas en una misma tarea.

## Apéndice A

# Validación análisis exploratorio

### Validación análisis exploratorio

```
1 data <- read.csv(paste0(ruta,"\\churn_data.csv"), header = TRUE, sep = ",")
2
3 dim(data)
4 summary(data)
5
6 suma <- 0
7 for(i in 1:nrow(data)){
8
9     for(j in 1:ncol(data)) if(is.na(data[i,j])) suma <- suma + 1
10
11 } # Cálculo de valores nulos en los datos
12
13 suma # Si nos fijamos en el summary anterior los 11 valores nulos pertenecen
14 # a TotalCharges.
15
16 # Es curioso que al leer los datos en R la variable TotalCharges pase a ser
17 # numérica automáticamente.
18
19 valores_unicos <- data.frame()
20 for(j in 1:ncol(data)) valores_unicos[j,1] <- length(unique(data[,j]))
21 rownames(valores_unicos) <- colnames(data)
22 valores_unicos
23 # Esta tabla muestra el número de valores distintos que toma cada variable
24
25 head(data) # Mostramos ejemplos de registros de los datos
26
27 ## Distribución de las variables numéricas:
28
29 # SeniorCitizen
30 summary(data$SeniorCitizen)
31
32 # tenure
33 summary(data$tenure)
34 plot(density(data$tenure))
35
36 # MonthlyCharges
37 summary(data$MonthlyCharges)
38 plot(density(data$MonthlyCharges))
39
```

```

40     # TotalCharges
41     summary(data$TotalCharges)
42     newTotalCharges <- data$TotalCharges
43     mediaTotalCharges <- mean(data$TotalCharges[which(!is.na(data$TotalCharges))
])
44     newTotalCharges[which(is.na(data$TotalCharges))] <- mediaTotalCharges
45     plot(density(newTotalCharges))
46
47     ## Distribución categóricas
48
49     # Debemos arreglar las categorías redundantes en variables como
MultipleLines
50
51     data$MultipleLines[which(data$MultipleLines == "No phone service")] <- "No"
52     data$OnlineSecurity[which(data$OnlineSecurity == "No internet service")] <-
"No"
53     data$OnlineBackup[which(data$OnlineBackup == "No internet service")] <- "No"
54     data$DeviceProtection[which(data$DeviceProtection == "No internet service")]
<- "No"
55     data$TechSupport[which(data$TechSupport == "No internet service")] <- "No"
56     data$StreamingTV[which(data$StreamingTV == "No internet service")] <- "No"
57     data$StreamingMovies[which(data$StreamingMovies == "No internet service")]
<- "No"
58
59     categorica <- c()
60     for (j in 1:ncol(data)) if(is.character(data[,j])) categorica[j] <- j
61     categorica <- na.omit(categorica)
62
63     tablas <- list()
64     tablas_p <- list()
65     for (k in 1:(length(categorica)-1)){
66
67         tablas[[k]] <- table(data[,categorica[k+1]])
68         tablas_p[[k]] <- round(tablas[[k]] / nrow(data), 4)
69
70     }
71     names(tablas) <- colnames(data)[categorica[-1]]
72     names(tablas_p) <- colnames(data)[categorica[-1]]
73

```

# Bibliografía

- [1] Dancho, M. (2025). *AI Data Science Team*. <https://github.com/business-science/ai-data-science-team>. Accedido el 14 de octubre de 2025.
- [2] Shearer, C. (2000). The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing*, 5(4), 13-22.
- [3] Hosmer, D. W. y Lemeshow, S. (1989). *Applied Logistic Regression* (. 1st ed.). John Wiley & Sons.
- [4] Breiman, L., Friedman, J., Olshen, R. y Stone, C. (1984). *Classification and Regression Trees* (. 1st ed.). Wadsworth & Brooks/Cole Advanced Books & Software.
- [5] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2), 123-140. <https://doi.org/10.1007/BF00058655>.
- [6] Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5-32. <https://doi.org/10.1023/A:1010933404324>.
- [7] Freund, Y. y Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1), 119-139. <https://doi.org/10.1006/jcss.1997.1504>.
- [8] Russell, S. y Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (. 4th ed.). Pearson.
- [9] Chen, H., Xiong, M., Lu, Y., Han, W., Deng, A., He, Y., Wu, J., Li, Y., Liu, Y. y Hooi, B. (2025). MLR-Bench: Evaluating AI Agents on Open-Ended Machine Learning ResearchEn. *Advances in Neural Information Processing Systems* (Vol. Vol. 38). [https://proceedings.neurips.cc/paper\\_files/paper/2025/hash/ab8dd000d6f87f40061a73f8bca7fae4-Abstract-Datasets\\_and\\_Benchmarks\\_Track.html](https://proceedings.neurips.cc/paper_files/paper/2025/hash/ab8dd000d6f87f40061a73f8bca7fae4-Abstract-Datasets_and_Benchmarks_Track.html).
- [10] LangChain (2025). *LangChain Documentation*. <https://www.langchain.com/>. Accedido el 30 de octubre de 2025.
- [11] Logfire (2025). *Logfire Documentation*. <https://logfire.pydantic.dev/>. Accedido el 16 de diciembre de 2025.