Trabajo Fin de Máster

Estudio de la aplicación de métodos de análisis de supervivencia respecto a métodos tradicionales al modelado del tiempo hasta un evento

Matías Cabaleiro Lago

Máster en Técnicas Estadísticas

Curso 2024-2025

Propuesta de Trabajo Fin de Máster

Título en galego: Estudo da aplicación de métodos de análise de supervivencia fronte a métodos tradicionais ao modelado do tempo ata un evento
Título en español: Estudio de la aplicación de métodos de análisis de supervivencia respecto a métodos tradicionales al modelado del tiempo hasta un evento
English title: Study of the application of survival analysis methods versus traditional methods to time-to-event modeling
Modalidad: Modalidad A
Autor/a: Matías Cabaleiro Lago, Universidade de Vigo
Director/a: Rosa María Crujeiras Casais, Universidade de Santiago de Compostela
Tutor/a: Rubén Martínez Covelo, SDG Group; David Novoa Paradela, SDG Group
Breve resumen del trabajo: Este Trabajo Fin de Máster tiene como objetivo comparar el rendimiento de modelos de regresión en un contexto de aprendizaje con datos completos con otras propuestas que integran herramientas del análisis de supervivencia, en contextos donde hai presencia de censura en las observaciones. Esta comparativa se realiza mediante estudios empíricos con distintos conjuntos de datos.

Doña Rosa María Crujeiras Casais, catedrática de universidad de la Universidade de Santiago de Compostela, don Rubén Martínez Covelo, Specialist Lead de SDG Group, y don David Novoa Paradela, Senior Consultant de SDG Group, informan que el Trabajo Fin de Máster titulado

Estudio de la aplicación de métodos de análisis de supervivencia respecto a métodos tradicionales al modelado del tiempo hasta un evento

fue realizado bajo su dirección por don Matías Cabaleiro Lago 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 Rosa María Crujeiras Casais y don Matías Cabaleiro Lago

√ sí □ no

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

En Redondela, a 30 de mayo de 2025.

La directora:

Doña Rosa María Crujeiras Casais

Firmado por CRUJEIRAS CASAIS ROSA MARIA - ***3178** el día 21/05/2025 con un certificado

El tutor:

Don David Novoa Paradela

NOVOA PARADELA DAVID - 44663354E

Firmado digitalmente por NOVOA PARADELA DAVID - 44663354E
Fecha: 2025.05.22 10:22:33 +02'00'

El autor:

Don Matías Cabaleiro Lago

El tutor:

Don Rubén Martínez Covelo

Con DNI 53195196Z a 21/05/2025

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.

Agradecimientos

Quiero agradecer primero de todo a mi directora Rosa María Crujeiras Casais y a mis tutores Rubén Martínez Covelo y David Novoa Paradela por ser mis referencias y guías a lo largo de este curso y por el empeño y cariño que pusieron conmigo para llevar a cabo este trabajo.

También quiero agradecer a mi familia, en especial a mi madre y a mi hermana, por el apoyo que me proporcionaron durante todos estos años en mi vida y hacer que, gracias a ellas, sea capaz de sacar en adelante cualquier situación que se me plantee en mi vida.

Por último, quiero agradecer a mis amigos de Redondela y de Santiago por considerarme parte de sus vidas y hacer que pueda ser yo mismo con la naturalidad que me permite llevar con ánimo mi día a día, lo cuál me permitió realizar este trabajo con la mayor energía y fuerza posible.

Índice general

Resumen

Resumen en español

Dentro del ámbito de actuación de las soluciones de la Ciencia de Datos en la industria, existe un área que se encarga de resolver los problemas que modelizan el tiempo hasta la ocurrencia de un evento. Este tipo de problemas se han resuelto tradicionalmente con soluciones de regresión/clasificación modelando la ocurrencia (o no) en un determinado horizonte, o simplemente estimando el tiempo hasta ocurrencia. Una nueva tendencia plantea aprovechar un ámbito de solución tradicional como el análisis de supervivencia para modelar este tipo de situaciones. El objetivo de este TFM es el de comparar en la práctica estos dos planteamientos, y posteriormente realizar un estudio empírico con una serie de conjuntos de datos para evaluar el rendimiento real de ambos enfoques.

English abstract

Within the scope of Data Science solutions in the industry, there is an area dedicated to solving problems that model the time until the occurrence of an event. Traditionally, such problems have been solved using regression/classification solutions by either modeling the occurrence or non-occurrence within a specific horizon or simply estimating the time to occurrence. A new trend proposes leveraging a traditional solution domain, such as survival analysis, to model these types of situations. The objective of this MSc Thesis is to practically compare these two approaches and subsequently conduct a simulation study with a series of datasets to evaluate the actual performance of both methods.

Capítulo 1

Motivación

Este Trabajo Fin de Máster está enmarcado en el estudio de técnicas de aprendizaje automático (en inglés, Machine Learning, ML) que modelan el tiempo que transcurre hasta que sucede un evento definido. Este tipo de proyectos son comunes en SDG Group, una consultora tecnológica internacional especializada en datos y analítica avanzada cuyos servicios se enfocan en ayudar a empresas a aprovechar el potencial de los datos para realizar la mejor toma de decisiones, mejorar la competitividad en sus respectivas industrias u optimizar procesos. En uno de sus proyectos, SDG decidió hacer uso de herramientas de análisis de supervivencia combinando algoritmos de Machine Learning (en concreto modelos Random Forest y Gradient Boosting) para abordar un problema planteado por parte de un cliente y optimizar los resultados obtenidos.

El Machine Learning, o aprendizaje automático, es la rama de la Inteligencia Artificial encargada de desarrollar algoritmos que permiten a los ordenadores aprender de manera automática a partir de conjuntos de datos de ejemplo. A partir de estos grandes conjuntos de datos, los modelos son entrenados para resolver tareas de toda clase, permitiéndonos así resolver problemas del mundo real de gran complejidad. Ejemplos de dichas tareas en las que el Machine Learning resulta de utilidad son los problemas de clasificación o los de regresión. Los problemas de clasificación consisten en determinar cuál es la clase o categoría de un ejemplo dado. Los problemas de regresión consisten en predecir un valor de una variable a estudiar a través de una o varias variables explicativas.

El análisis de supervivencia consiste en el estudio del tiempo hasta evento (*time-to-event*) para un conjunto de individuos. Por tiempo hasta evento se interpreta la duración sobre un individuo desde un instante inicial hasta un instante final que deben ser definidos de manera previa y precisa. En el contexto del Big Data, al monitorizar una gran cantidad de individuos,

el objetivo principal que nos encontramos en este análisis es realizar la mejor estimación posible del tiempo hasta evento de un evento particular para nuestro conjunto de individuos. Dicho tiempo nos lo podemos encontrar definido en otros trabajos en los que se involucre el análisis de supervivencia como *time of ocurrence* o *survival time*, como se puede observar en [16]. A diferencia de un modelo de aprendizaje automático tradicional, en el análisis de supervivencia se presentan una serie de problemas en la observación de los tiempos. Uno de los problemas en la observación de los tiempos más habituales es la *censura*, que aparece cuando desconocemos el evento inicial y/o el evento final, por lo que el tiempo de vida solo se observa de manera parcial. A pesar de ello, las técnicas de análisis de supervivencia pueden hacer uso de esta información incompleta para mejorar la construcción de los modelos y hacer el estudio temporal. La problemática más habitual es el problema de la *censura por la derecha*, que consiste en no observar el evento final totalmente porque hay un evento previo (conocido como *censura*) que lo impide. Este concepto se puede consultar con más detalle en [9]. También es habitual la presencia de los datos truncados en estos estudios. Existen dos tipos de truncamiento: el *truncamiento por la izquierda*, que consiste en que solo se incluyen los individuos que alcanzaron el tiempo en el que se inicia el estudio, y el *truncamiento por la derecha*, que consiste en incluir en nuestro estudio los individuos que experimentaron el evento antes de un tiempo fijado [9].

En el proyecto del cliente llevado a cabo por SDG que sirvió de motivación para la creación de este trabajo, se planteó el uso de técnicas de Machine Learning junto a herramientas de análisis de supervivencia para el estudio del mercado inmobiliario con el objetivo de realizar una predicción del tiempo esperado hasta la venta de un inmueble (que sería nuestro *time-to-event*) a partir de las características específicas de dicho inmueble. Dado que el conjunto de datos presentaba *censura por la derecha*, resultaba de gran interés hacer uso de herramientas de análisis de supervivencia para poder trabajar con datos temporales que contuviesen dicha censura, y que aprovechando dicha información permita un ajuste más preciso con tiempos condicionados a evoluciones temporales. Los resultados obtenidos fueron favorables, disponiendo de modelos capaces de predecir el tiempo de venta de un inmueble con una precisión elevada. Sin embargo, se observó lo que semejaba un comportamiento temporal estacional por el cual en determinados meses del año las ventas parecían acelerarse, pero que los modelos de predicción aparentemente no detectaban. Hay que tener en cuenta dos aspectos de la metodología empleada. El primero es la incertidumbre generada por el uso de la mediana de las curvas de supervivencia para realizar la predicción temporal, ya que no se conoce con precisión cuál es el comportamiento generado por el uso de esta medida cuando se emplea el valor mencionado sobre las curvas de supervivencia, por lo que es importante estudiar su repercusión así como plantear la utilización de medidas alternativas. Por otra parte, para la

predicción de los *time-to-events* se realizó una agregación de datos los cuales se evaluaron en una unidad temporal distinta a la que se empleó para entrenar el modelo. Para el entrenamiento del modelo se empleó como unidad temporal *días* y para la agregación de datos se empleó *meses*. Dicha agregación puede afectar a la predicción temporal a la hora de ser realizada para nuestro *time-to-event*, por lo que es necesario atajar este escenario empleando las técnicas adecuadas.

El objetivo de este trabajo es estudiar los beneficios y las problemáticas derivadas del uso de técnicas de machine learning tradicional y de sus implementaciones basadas en el análisis de supervivencia. Para ello se replicará el escenario real del proyecto del cliente con el objetivo de analizar varias líneas de trabajo abiertas, como son la influencia del uso de la media como medida en las curvas de supervivencia a la hora de modelar el tiempo hasta evento, o las dificultades presentadas por los modelos de Machine Learning a la hora de detectar patrones temporalmente estacionarios. Además de esto, se llevará a cabo un estudio completo que nos permita evaluar las ventajas y desventajas de los diferentes tipos de aproximaciones a la hora de resolver problemas de tiempo hasta evento de una manera más generalizada, sirviendo así de guía para posibles proyecto futuros. Para lograr esto se probarán diferentes algoritmos y conjuntos de datos de referencia, combinando diferentes volumetrías (en cuanto a número de variables e instancias), aparición de censura, truncamiento, o agregaciones temporales.

En el Capítulo 2 se expondrán en detalle los fundamentos teóricos que sustentan el presente Trabajo Fin de Máster, con una definición rigurosa de los conceptos previamente introducidos en el presente capítulo. A continuación, en el Capítulo 3, se llevarán a cabo diversas pruebas empíricas utilizando distintos conjuntos de datos, con el propósito de aplicar y validar los conceptos teóricos desarrollados en el capítulo anterior. Para realizar las mencionadas pruebas se hicieron uso de las librerías `scikit-learn`, `scikit-survival`, `matplotlib`, `pandas` y `numpy` del programa informático `Python` (versión 3.10) para la creación de las funciones y gráficas necesarias y el uso de los algoritmos de aprendizaje automático definidos en el Capítulo 2. Además, se hizo uso de las IA generativas `ChatGPT` y `Claude` para la corrección de código en las pruebas y para realizar una mejor estructura en la redacción del presente trabajo. Por último, en el Capítulo 4 se realizará una discusión de los resultados obtenidos tras realizar un análisis de las pruebas que aparecen en el Capítulo 3.

Capítulo 2

Metodología

En este capítulo se presentan los fundamentos teóricos que sustentan el trabajo desarrollado a lo largo de este documento. Con el objetivo de dotar al lector de una base sólida, se abordarán los principales conceptos, métodos y marcos de referencia que serán empleados en el análisis, desarrollo y evaluación de los modelos planteados.

La exposición se organiza de manera progresiva, comenzando por los conceptos básicos y generales del área de estudio, para posteriormente introducir las herramientas específicas, modelos matemáticos y de Machine Learning empleados y las métricas que nos servirán de referencia para realizar una comparativa precisa para nuestros modelos. Esta estructura busca facilitar la comprensión y contextualización del resto del contenido del documento, permitiendo que el lector pueda interpretar adecuadamente los resultados y argumentos que se presentarán en los capítulos posteriores.

Asimismo, se incluyen definiciones clave, y también referencias a trabajos previos relevantes para poder realizar el estudio planteado en este trabajo. Este capítulo, por tanto, no solo tiene una función introductoria, sino también justificativa, al mostrar la pertinencia y la solidez teórica del enfoque adoptado.

2.1. Análisis de supervivencia

El análisis de supervivencia, como bien definimos en el capítulo 1, se ocupa principalmente del estudio de los tiempos de fallo de un conjunto de individuos u observaciones. El tiempo de fallo se comprende como el tiempo que transcurre entre un instante inicial hasta un instante final (que se considera fallo o evento). Estos tiempos se definen previamente con precisión.

En los problemas en los que se emplea el análisis de supervivencia se emplea una variable respuesta Y que representa el tiempo desde el instante inicial hasta el fallo o la ocurrencia definida de interés. En otros documentos relacionados, esta variable se puede denotar como *time to event*, *lifetime*, *survival time*,.... La característica más importante que consideramos del análisis de supervivencia es que nos permite realizar un estudio de dicha variable temporal Y cuando está afectada por problemas en la observación de los tiempos, de forma que nos permite realizar estimaciones de los tiempos de fallo sin conocer la totalidad de la información de las observaciones. Otra característica a destacar de la variable Y es que es una variable aleatoria no negativa (bien puede ser discreta o continua).

Las problemáticas más frecuentes que nos podemos encontrar en las observaciones de los tiempos de fallo son el concepto de censura y el concepto de truncamiento. Los conceptos desarrollados en este apartado se basan en [9].

Definición 2.1. La censura consiste en las observaciones que presencian el evento fuera de la ventana de observación que disponemos.

Dentro de la censura, podemos distinguir tres tipos de censura:

- **Censura por la derecha**. Es el caso más habitual de censura. Se produce cuando solo se sabe que el evento final excede el valor observado. Un ejemplo de este caso de censura sería en un estudio de mercado de una vivienda que la venta del inmueble mediante una página web suceda después del final del estudio, o que el inmueble que siga a la venta pero que deje de estar disponible en la página web.

- **Censura por la izquierda**. Este caso sucede con menos frecuencia. En este caso, el evento de interés para el individuo ya sucedió antes del tiempo observado. Un ejemplo para esta censura podría ser el estudio de enfermedades pre-sintomáticas (como por ejemplo el cáncer). Esta censura es más díficil de manejar que la censura por la derecha.

- **Censura por intervalo**. En este caso, solo se sabe que el fallo ocurrió dentro de un intervalo específico de tiempo. Se suele presenciar en estudios donde el seguimiento de un individuo es periódico.

Para este trabajo, los casos que vamos a presenciar solo contemplan la situación de la censura por la derecha. Cabe destacar que dentro de la **censura por la derecha** existen tres tipos de clasificación:

- **Censura de Tipo I**. En este tipo de censura, los tiempos de censura están preespecificados.

- **Censura de Tipo II**. En este tipo de censura, se siguen los objetos experimentales o los individuos a estudiar hasta que un porcentaje o un número de individuos previamente especificado experimente el fallo o evento de interés.

- **Censura aleatoria**. Este tipo de censura está ocasionada principalmente por abandonos de los individuos en el estudio (*dropout*), por pérdidas de seguimiento por otras causas (*loss to follow-up*) o por la terminación del estudio.

Independientemente del tipo de censura, la muestra observada se denotará como $\{(Z_i, \delta_i)\}_{i=1}^n$, donde $Z_i = \text{mín}(Y_i, C_i)$ y $\delta_i = 1\{Y_i \leq C_i\}$, siendo Y_i el tiempo de fallo del individuo i, C_i el tiempo de censura del individuo i y δ_i una variable indicadora, donde el valor 0 nos indica que el tiempo de censura es menor que el tiempo de fallo y el valor 1 nos indica que el fallo sucedió antes del tiempo de censura.

Definición 2.2. El truncamiento consiste cuando los fallos de los individuos recaen sobre una ventana de observación (Y_L, Y_R).

Se debe tener cuidado de no confundir el concepto de truncamiento con la censura. En el caso de que el tiempo de fallo de un individuo suceda fuera de dicho intervalo de observación no será incluido en el estudio y por tanto no aportará información. Este problema de observación difiere de la censura debido a que la censura nos aporta información parcial del individuo. Dado que solo disponemos de información que se encuentra en la ventana de observación, la inferencia para los datos truncados se restringe a la estimación condicional.

Tenemos dos tipos de truncamiento distintos posibles:

- **Truncamiento por la izquierda**: En este caso, tenemos que $Y_R = \infty$, por lo que solo tenemos las observaciones de los individuos que verifiquen que $Y_L < Y_i, i \in \{1, \ldots, n\}$.

- **Truncamiento por la izquierda**: En este caso, tenemos que $Y_L = 0$, por lo que solo tenemos las observaciones de los individuos que verifiquen que $Y_i < Y_R, i \in \{1, \ldots, n\}$.

Una vez visto los conceptos de censura y truncamiento definamos las funciones de interés para el análisis estadístico más comunes empleadas en los estudios que emplean análisis de supervivencia. Supongamos que nuestra variable de interés Y que mide el tiempo de fallo es una variable aleatoria continua no negativa. Al realizar esta suposición tenemos que posee una distribución con una función de densidad f y una función de distribución acumulada F. Con el conocimiento de dichas funciones podemos definir dos funciones que son de suma importancia en el análisis de supervivencia:

- Función de supervivencia, $S(t) = \mathbb{P}(Y > t) = 1 - F(t)$. Esta función representa la probabilidad de que el fallo ocurra en t o antes del tiempo t.

- Función de riesgo:
$$\lambda(t) = \lim_{\Delta t \to 0} \frac{\mathbb{P}\left(t \leq Y < t + \Delta t | Y \geq t\right)}{\Delta t}.$$
Esta función representa la probabilidad de que un individuo experimente el fallo en el instante inmediato $[t, t + \Delta t)$ sabiendo que sobrevive en el tiempo t.

Si a partir de la función de riesgo $\lambda(t)$ definimos la función de riesgo acumulada $H(t) = \int_0^t \lambda(u)d(u)$, podemos establecer una relación entre la función de supervivencia y la de riesgo, siendo estas relaciones:

$$\lambda(t) = -\frac{d}{dt}S(t) \cdot \frac{1}{S(t)} = -\frac{d}{dt}[\ln S(t)]$$
$$S(t) = \exp\left(-\int_0^t \lambda(u)du\right) = \exp\left(-H(t)\right)$$

debemos tener en cuenta que no conocemos la distribución teórica de nuestra variable aleatoria continua no negativa Y, por lo que es necesario emplear estimadores para obtener dichas funciones. Para estimar la función de supervivencia, tenemos que para una muestra de tiempos de fallo donde no hay censura $\{Y_i\}_{i=1}^n$ el estimador no paramétrico de máxima verosimilitud de la función de supervivencia es

$$\hat{S}(t) = \frac{1}{n}\sum_{i=1}^{n} I\{Y_i > t\}. \tag{2.1}$$

En el caso de que exista presencia de datos censurados, Kaplan y Meier propusieron un estimador de S que es una generalización del estimador empírico 2.1. Se define de la siguiente forma mediante el método del producto.

Sean $t_1 < \ldots < t_n$ los tiempos observados de fallo de nuestros sujetos. Sean entonces d_i el número de fallos hasta el tiempo t_i y n_i el número de sujetos que están en riesgo para el tiempo t_i, es decir, el número de individuos que aún no presenciaron el fallo hasta dicho tiempo. Entonces, el estimador de Kaplan-Meier de $S(t)$ viene dado por

$$\hat{S}(t) = \prod_{t_i \leq t}\left(1 - \frac{d_i}{n_i}\right). \tag{2.2}$$

Este estimador, como se indica en [9], está basado en la suposición de que la censura es no informativa, lo que significa que el conocimiento del tiempo de censura de un sujeto no nos proporciona información a mayores sobre su supervivencia en un tiempo futuro en el caso de haber continuado en el estudio.

Además, el estimador (2.2) está bien definido para todos los tiempos menores o iguales al más grande observado t_n. En el caso de que dicho tiempo t_n corresponde a un tiempo de fallo, la curva de supervivencia estimada es cero una vez superado dicho tiempo, pero si el tiempo t_n corresponde a un tiempo censurado desconocemos la curva de supervivencia a partir de dicho punto debido a que no conocemos cuando el último superviviente experimentaría el fallo.

2.2. Técnicas de aprendizaje

Las técnicas de aprendizaje (conocido en inglés como Machine Learning), como bien se mencionó en el capítulo 1, es una rama de la ciencia computacional cuyo propósito es aprender de manera automática a partir de conjuntos de datos, el cual denotaremos como $\mathcal{D}$, y aprovechar dicha información para establecer patrones para datos futuros o realizar predicciones. Los conceptos presentados en este apartado se pueden consultar en [10] y [16] .

El Machine Learning se divide principalmente en dos tipos: el aprendizaje supervisado y el aprendizaje no supervisado. El aprendizaje supervisado establece una aplicación entre el los datos de entrada $\mathbf{x}$ (que se conocen también como *inputs*) con los datos de salida y (que se conocen también como *output*), dado un conjunto de pares etiquetados $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, que dicho conjunto $\mathcal{D}$ se conoce como conjunto de entrenamiento.

Por otra parte, el aprendizaje no supervisado consiste en establecer patrones de interés sobre el conjunto de datos, que en este caso se proporciona como $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{N}$. Este tipo de problemas no son tan comunes como los que se plantean bajo el aprendizaje supervisado, debido a que no se definen los patrones que buscamos y no disponemos de métricas de evaluación para evaluar el error cometido (a diferencia del aprendizaje supervisado, donde podemos comparar la predicción realizada de y para un valor $\mathbf{x}$ dado a un valor observado).

Para este trabajo nos vamos a centrar exclusivamente en los algoritmos que se encuentran en la categoría de aprendizaje supervisado. Este tipo de aprendizaje de ML nos encontramos dos tipos de problemas a resolver, que son los problemas de clasificación y los de regresión.

2.2.1. Clasificación

El objetivo de los problemas de clasificación es aprender una aplicación de los datos de entrada $\mathbf{x}$ hacia los datos de salida y, donde $y \in \{1, \ldots, C\}$, con C siendo el número de clases en la que clasificamos los datos. Si $C = 2$ estos problemas se conocen como *clasificación binaria*, donde generalmente asumimos que $y \in \{0, 1\}$, y si $C > 2$ se conocen como *clasificación multiclase*.

Una manera de formalizar este tipo de problemas es bajo una función de aproximación. Asumimos $y = f(\mathbf{x})$ para una función desconocida f, y el objetivo del aprendizaje es estimar la función f dado un conjunto de entrenamiento etiquetado para realizar predicciones utilizando $\hat{y} = \hat{f}(\mathbf{x})$. Una vez obtenida la estimación $\hat{f}$ procederemos entonces a la generalización, que consiste en realizar predicciones sobre datos de entrada que no se encuentran en el conjunto de entrenamiento.

2.2.2. Regresión

Cuando queremos realizar un análisis y/o una predicción de una variable objetivo (o respuesta) mediante la relación que tiene dicha variable junto a una o varias variables explicativas. Este análisis es posible de realizar cuando empleamos modelos de regresión. El objetivo de realizar un modelo de regresión es encontrar una función que explique cómo cambia la variable objetivo en función de los valores que pueden adoptar las variables explicativas y realizar predicciones sobre dicha variable objetivo. En este trabajo nos centraremos en este tipo de problemas con aprendizaje supervisado.

El modelo más simple de regresión que se puede emplear (y el cual emplearemos en este trabajo) es un modelo de regresión lineal. En este contexto, tendremos una variable objetivo Y y un grupo de variables explicativas $X_1, X_2, \ldots, X_{p-1}$. Por tanto, para obtener un modelo de regresión lineal, basta con considerar una combinación lineal de variables explicativas

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_{p-1} X_{p-1} + \varepsilon$$

siendo $\beta_0, \ldots, \beta_{p-1}$ los parámetros que nos permiten realizar la combinación lineal de las variables explicativas y ε el error.

Si queremos realizar un modelo de regresión múltiple podemos representarlo bajo una matriz de diseño de la siguiente forma

$$\begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1,p-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{n,p-1} \end{pmatrix} \cdot \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{p-1} \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

donde Y_i representa la variable respuesta para el individuo i-ésimo, $x_{i,1}, \cdots, x_{i,p-1}$ las variables explicativas del mismo, $x_1, \cdots, x_n$ son las filas de de la matriz de diseño y ε_i el error del mismo, con $i \in \{1, \cdots, n\}$. Sustituyendo cada vector o matriz por un símbolo, llegaríamos a la expresión abreviada

$$Y = X\beta + \varepsilon$$

donde Y es el vector de respuestas, X es la matriz de diseño que contiene $n \times p$ números de forma que cada fila representa cada individuo y cada columna a una característica en

específico, β representa el vector de parámetros y ε es un vector que contiene los errores y verifica $\varepsilon \in N_n(0, \sigma^2 I_n)$.

Estimación de parámetros

Para un modelo de regresión lineal debemos abordar el problema de estimar los parámetros del modelo: β y σ^2.

Para la estimación de β, realizamos una estimación de mínimos cuadrados sobre la Ecuación $Y = X\beta + \varepsilon$, de forma que

$$\hat{\beta} = (X'X)^{-1}X'Y.$$

Destacamos que para que el estimador esté bien definido es necesario que la matriz $X'X$ sea no singular.

Por otra parte, para estimar σ^2 haremos uso del cálculo $\hat{\varepsilon}_i = Y_i - \hat{Y}_i = Y_i - x_i\beta$, $i \in \{1, \cdots, n\}$. Mediante dichos residuos, la estimación de la varianza del error sería

$$\hat{\sigma}^2 = \frac{1}{n-p}\sum_{i=1}^{n}\hat{\varepsilon}_i^2 = \frac{1}{n-p}\sum_{i=1}^{n}\left(Y_i - x_i\beta\right)^2.$$

Regresión penalizada

La regresión penalizada constituye una extensión de los métodos clásicos de regresión, diseñada para mejorar la capacidad de generalización del modelo y mitigar los efectos adversos que pueden surgir con la multicolinealidad o con el sobreajuste, especialmente en contextos de alta dimensionalidad. Esta penalización actúa como un mecanismo de regularización que restringe la complejidad del modelo y favorece soluciones más estables. Se muestra con más detalle en [6]

Los modelos penalizados que nos podemos encontrar son la regresión Ridge, la regresión Lasso y la regresión Elastic Net. La regresión Rigde, o también conocida como regresión L_2, consiste en realizar una minimización del vector de parámetros β bajo un conjunto $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, con $x_i = (x_{i1}, \ldots, x_{ip-1})$, bajo un criterio de la forma

$$\min_{\beta} \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=0}^{p-1}\beta_j^2,$$

tal que se busca la reducción de los problemas generados por la multicolinealidad de las variables sin eliminar todos los predictores asociados al vector de parámetros β.

Por otra parte, la regresión Lasso, o también conocida como regresión L_1 consiste en realizar una minimización del parámetro β bajo un conjunto $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, con $x_i = (x_{i1}, \ldots, x_{ip-1})$, bajo un criterio de la forma

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=0}^{p-1} |\beta_j|,$$

tal que se busca la reducción de los problemas generados por la multicolinealidad de las variables (como pueden ser la inestabilidad de los coeficientes estimados, la redundancia de la información proporcionada por las variables o la dificultad de identificar el efecto individual de cada variable sobre el modelo), llegando a eliminar los predictores asociados con variables irrelevantes o débiles en el vector de parámetros β.
Es posible realizar una combinación de los modelos de regresión Ridge y Lasso, resultando en la regresión Elastic Net. Este tipo de regresión consiste en realizar una minimización del parámetro β bajo un conjunto $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, con $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip-1})$, bajo un criterio de la forma

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda_1 \sum_{j=0}^{p-1} \beta_j^2 + \lambda_2 \sum_{j=0}^{p-1} |\beta_j|.$$

Regresión en análisis de supervivencia

En un contexto de regresión con datos censurados, además del tiempo observado y el indicador de censura, para cada individuo, $i = 1, \cdots, n$ vamos a tener un vector de $p-1$ covariables de forma que la muestra observada es $\{(Z_i, \delta_i, x_i)\}_{i=1}^n$, donde $x_i = (x_{i1}, \cdots, x_{ip-1})$. Las covariables x_{ij} $1 \leq j \leq p-1$ pueden ser fijas o dependientes del tiempo.

Para este contexto, un modelo muy utilizado es el modelo de regresión de Cox de riesgos proporcionales. Sea $\lambda(t|x)$ la función de riesgo de un individuo con vector de covariables $x_i = x$. El modelo de Cox tiene la forma

$$\lambda(t|x) = \lambda_0(t) e^{x\beta^\top}$$

donde $\lambda_0(t)$ es una función de riesgo de referencia que se deja sin especificar y corresponde al riesgo cuando todas las covariables son 0, y $\beta = (\beta_1, \cdots, \beta_{p-1})$ es un vector de parámetros.

Por definición, el modelo de regresión de Cox es un modelo semiparamétrico. Esto es debido a que no asume una forma paramétrica para $\lambda_0(t)$. Por tanto, para estimar los parámetros β se realiza maximizando la función de verosimilitud parcial. Para obtener dicha función suponemos que no hay empates entre tiempos de ocurrencia $t_1 < t_2 < \ldots < t_n$. La función de

verosimilitud parcial se define entonces como

$$L(\beta) = \prod_{i=1}^{n} \frac{\lambda_0(t_i) e^{x_i \beta^\top}}{\sum_{j \in R(t_i)} \lambda_0(t_i) e^{x_j \beta^\top}} = \prod_{i=1}^{n} \frac{e^{x_i \beta^\top}}{\sum_{j \in R(t_i)} e^{x_j \beta^\top}}$$

siendo $R(t_i)$ el conjunto de individuos a riesgo en el tiempo t_i y x_i el vector de covariables del individuo que experimenta el evento en t_i. Esta función se denomina verosimilitud parcial porque pierde factores correspondientes a las observaciones censuradas.

Otra forma de escribir la verosimilitud parcial a partir de la muestra observada $\{(Z_i, \delta_i, x_i)\}_{i=1}^{n}$ es

$$L(\beta) = \prod_{i=1}^{n} \left[\frac{e^{x_i \beta^\top}}{\sum_{j=1}^{n} I\{Z_j \geq Z_i\} e^{x_j \beta^\top}} \right]^{\delta_i}. \tag{2.3}$$

En el caso de que haya empates, la Ecuación 2.3 toma la forma

$$L(\beta) = \prod_{i=1}^{n} \left[\frac{e^{S_i \beta^\top}}{\sum_{j=1}^{n} I\{Z_j \geq Z_i\} e^{X_j \beta^\top}} \right]^{\delta_i}$$

donde $S_i = \sum x_j$ sobre todos los individuos que experimentan el evento en t_i.

Una vez visto los dos tipos de problemas que nos encontramos en el aprendizaje supervisado, mencionando que nos centraremos en los problemas de regresión para este trabajo, definiremos dos algoritmos más del Machine Learning que se emplearán en este documento.

2.2.3. Random Forest

Un bosque aleatorio consiste en un ejemplo de método *ensemble* (método combinado), esto es, un método predictivo dentro del Machine Learning que se basa en combinar las predicciones de un gran número de modelos promediándolas en un predictor resultante llamado *ensemble*. En el Random Forest se utilizan árboles de decisión para los predictores iniciales y realiza una aleatorización dentro del método combinado. Dicha aleatorización se realiza en dos pasos. El primer paso utiliza una muestra bootstrap aleatoriamente de los datos para construir un árbol. Posteriormente, durante la fase de construcción del árbol, en cada nodo del árbol se selecciona de manera aleatoria un subconjunto de variables como candidatas para realizar el split (este proceso se conoce también como *random feature selection*).

El objetivo de esta aleatorización es eliminar la posible correlación que exista entre árboles construidos debido a la reducción de la varianza ocasionada por el *bagging* (bootstrap aggregation). Además, los Random Forest se construyen habitalmente con mucha profundidad, es decir, se realizan muchas divisiones desde la raíz del árbol hasta el nodo terminal. Finalmente,

combinando dicha profundidad de árboles (que es una técnica para reducir sesgo) junto a la aleatorización realizada (técnica para reducir la varianza) obtenemos mediante Random Forest una aproximación precisa de las funciones a estimar manteniendo un error de generalización bajo.

Para estudiar un problema basado en un modelo de regresión tomaremos como referencia [3] y consideremos $\{\varphi_1, \ldots, \varphi_B\}$ una colección de predictores donde cada predictor $\varphi_b : \mathcal{X} \to \mathbb{R}, b \in \{1, \ldots, B\}$, siendo $\mathcal{X}$ el dominio de nuestro vector de variables $X = (X_1, \ldots, X_{p-1})$, está entrenado con el mismo conjunto de datos $\mathcal{L} = \{(x_1, Y_1), \ldots, (x_n, Y_n)\}$. El objetivo es estimar la función de regresión $f(x) = \mathbb{E}(Y|x)$. Asumimos que cada predictor se entrena de manera separada a cualquier otro. Debido a que los predictores se entrenan bajo el mismo conjunto de datos, tenemos que son dependiendentes, pero a su vez son idénticamente distribuidos marginalmente.

Definimos entonces el estimador combinado como

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} \varphi_b(x).$$

Sea (x, Y) un conjunto de datos de evaluación independiente con la misma distribución que el conjunto de datos de entrenamiento. Asumiendo un modelo de regresión $Y = f(x) + \varepsilon$ tal que $x \perp \varepsilon$ y $\mathrm{Var}(\varepsilon) = \sigma^2$, el error generalizado para el estimador $\hat{f}$ es

$$\mathrm{Err}(\hat{f}) = \sigma^2 + \mathbb{E}_{\mathbf{X}} \left\{ \mathrm{Sesgo}\{\varphi|x\}^2 + \mathrm{Var}\{\hat{f}|x\} \right\}.$$

Esta generalización del error basada en el sesgo condicionado y varianza condicionada para $\hat{f}$ nos permite establecer el siguiente resultado que se detalla en [3].

Teorema 2.3. *Si $\{\varphi_1, \ldots, \varphi_B\}$ son predictores idénticamente distribuidos de $\mathcal{L}$, entonces*

$$\mathrm{Err}(\hat{f}) = \sigma^2 + \mathbb{E}_{\mathbf{X}} \left\{ \mathrm{Sesgo}\{\varphi|\mathbf{X}\}^2 + \frac{1}{B}\mathrm{Var}\{\varphi|\mathbf{X}\} + \left(1 - \frac{1}{B}\right) \overline{\mathrm{Cov}}(\mathbf{X}) \right\}.$$

tal que $\overline{\mathrm{Cov}}(\mathbf{X}) = \mathrm{Cov}(\varphi_b, \varphi_{b'}|\mathbf{X})$.

La idea que nos proporciona el Teorema 2.3 es que para una colección de predictores lo suficientemente grande podemos esperar un valor próximo para la generalización del error al límite

$$\lim_{B \to \infty} \mathrm{Err}(\hat{f}) = \sigma^2 + \mathbb{E}_x \left\{ \mathrm{Sesgo}\{\varphi|x\}^2 + \overline{\mathrm{Cov}}(x) \right\}.$$

El error generalizado ideal sería σ^2, por lo que para obtener dicho valor necesitamos que nuestros predictores sean insesgados y que $\overline{\mathrm{Cov}}(x) = 0$. El problema que nos podemos encontrar es que a medida que reducimos el sesgo los predictores se vuelven más complejos, lo cuál nos

lleva a obtener un aumento en la covarianza debido a las técnicas de corrección del sesgo.

Por tanto, el Random Forest es ideal. En este algoritmo los predictores son árboles aleatorios (reducen la correlación) cuya profundidad suele ser alta (reducen el sesgo), entonces tenemos un algoritmo que busca un equilibrio a la hora de evaluar ambas condiciones para el error.

Para realizar el estudio con un Random Forest aplicando herramientas de análisis de supervivencia tan solo debemos tener en cuenta que estaremos empleando en los predictores para la construcción de árboles curvas de supervivencia para las variables a estudiar.

2.2.4. Gradient Boosting

La metodología boosting es una metodología de aprendizaje predictivo en la que se combinan muchos modelos obtenidos mediante un método que no necesariamente debe presentar una alta capacidad predictiva para así ser impulsados y dar lugar a un mejor predictor. Esta metodología se puede consultar con detalle en [7] y [2]. En este contexto, son muy útiles los árboles de decisión que no poseen una gran profundidad, ya que tienen una capacidad de generación muy rápida y son fáciles de implementar. Consideremos entonces que nuestros árboles se construyen de la forma que generan la función $g(\mathbf{x};\theta)$. Entonces, la idea que nos propone este método es, a partir de M árboles, construir un modelo

$$f(\mathbf{x}) = \sum_{m=1}^{M} f_m(\mathbf{x}) = \sum_{m=1}^{M} \omega_m g(\mathbf{x};\theta_m) \tag{2.4}$$

donde θ_m es el vector de parámetros que conforma del árbol m-ésimo y $\omega_m \in \mathbb{R}$ es el vector de los pesos que conforman el árbol m-ésimo.

A diferencia de los modelos de Random Forest, en los que la combinación de árboles se realizaba mediante un promedio una vez realizados los ajustes de manera independiente, en los modelos de Gradient Boosting la construcción de los árboles se realiza de manera secuencial. Esta interpretación secuencial (mostrada con mayor detalle en [2] se puede sacar de la Ecuación (2.4), donde consideramos $f_0(\mathbf{x}) = \beta_0 g(\mathbf{x};\theta_0)$ como la función de inicio, y $\{f_m\}_{m=1}^{M} = \{\beta_m g(\mathbf{x};\theta_m)\}_{m=1}^{M}$ como las funciones de incremento o *boosts*. Estos *boosts* se obtienen mediante un método de optimización, que es el método del descenso del gradiente.

El método del descenso del gradiente se define de la siguiente forma en [7]. Para un conjunto de entrenamiento $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$, se define como función de inicio $f_0(\mathbf{x}) = \arg\min_\alpha \sum_{i=1}^{N} \mathcal{L}(y_i, \alpha)$. Para obtener los modelos de manera secuencial, tenemos que

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \beta_m g(\mathbf{x};\theta_m)$$

de forma que los modelos secuenciales se esperan que minimicen

$$(\beta_m, g(\mathbf{x}; \theta_m)) = \arg\min_{\beta,\theta} \sum_{i=1}^{N} \mathcal{L}\left(y_i, f_{m-1}(\mathbf{x}_i) + \beta g(\mathbf{x_i}; \theta)\right).$$

Para cada función $g(\cdot, \theta_m)$ se puede observar el efecto de realizar un paso con la optimización del método del descenso de gradiente para f. Se tiene que la función $g(\cdot, \theta_m)$ se entrena con un nuevo conjunto de entrenamiento $\{\mathbf{x}_i, r_{mi}\}_{i=1}^{N}$, donde

$$r_{mi} = \left[\frac{\partial \mathcal{L}(y_i, f(\mathbf{x}))}{\partial f(\mathbf{x})}\right]_{f(\mathbf{x})=f_{m-1}(\mathbf{x})}$$

de forma que el valor de β_m se obtiene secuencialmente bajo la resolución de un problema de búsqueda de optimización lineal.

Para realizar el ajuste bajo el uso de herramientas de análisis de supervivencia, la construcción es análoga a la expresada bajo la Ecuación (2.4). En este caso, lo único que debemos tener en cuenta es como definimos la función de pérdida $\mathcal{L}(y, f(\mathbf{x}))$, que en este caso emplearemos la función de pérdida de verosimilitud parcial del modelo de riesgos proporcionales de Cox (como bien se ilustra en la librería `scikit-survival` [12]), definida en la Ecuación (2.5) en el caso del Gradient Boosting donde el objetivo es maximizar la función de log-verosimilitud parcial.

$$\arg\min_{f} \sum_{i=1}^{N} \delta_i \left[f(\mathbf{x}_i) - \log\left(\sum_{j \in \mathcal{R}_i} \exp(f(\mathbf{x}_j)) \right) \right]. \tag{2.5}$$

2.3. Métricas de evaluación

Cuando empleamos algoritmos de Machine Learning para realizar un estudio de clasificación o de predicción en un conjunto de datos, es necesario comparar el funcionamiento entre los modelos que empleemos. Esto es debido principalmente a dos razones. La primera es descartar modelos cuyo funcionamiento sea peor en comparación con otros, y la segunda es poder optimizar aquellos modelos cuyo funcionamiento se espera positivo. Cuando trabajamos con modelos de Machine Learning supervisados, el procedimiento general para evaluar los modelos es dividir el conjunto de datos en un conjunto de entrenamiento y en un conjunto de prueba que es complementario al conjunto de entrenamiento. Los datos que se hallan en el conjunto de entrenamiento se emplean para ajustar el modelo y los datos que pertenecen en el conjunto de prueba sirven para medir el rendimiento a través de diversas métricas.

Con el propósito de analizar y comparar el rendimiento de los algoritmos de aprendizaje automático previamente definidos, a continuación se presentan las principales métricas de evaluación que emplearemos a lo largo de este trabajo. Estas métricas permiten cuantificar la

calidad de las predicciones generadas por los modelos, facilitando una evaluación objetiva y sistemática de su desempeño.

2.3.1. Índice de concordancia

Para explicar esta métrica emplearemos como referencia [16]. En el análisis de supervivencia una forma de evaluar un modelo es considerar los riesgos de un evento para diferentes observaciones en vez de los tiempos de supervivencia para cada individuo. Para realizar dicha evaluación, emplearemos la probabilidad de concordancia o índice de concordancia (C-index).

Los tiempos de supervivencia para dos individuos se pueden ordenar en dos casos. El primero de ellos es que ambos individuos presencien el evento, mientras que el otro caso es que el tiempo de supervivencia de un individuo que presencia el fallo sea menor que el tiempo censurado del otro individuo. Exponiendo estos casos aclaramos que para esta métrica no tendremos en cuenta la evaluación de un individuo cuyo tiempo presencie censura con otro individuo que, o bien está también censurado, o bien presencia el evento con un tiempo mayor que el de censura del individuo original.
Consideremos entonces los tiempos de fallo y los valores predichos de dos individuos $(y_1, \hat{y}_1)$ y $(y_2, \hat{y}_2)$, donde y_i representa el tiempo de fallo del individuo i e $\hat{y}_i$ representa el valor predicho del individuo i. La probabilidad de concordancia entre ambos individuos se define entonces como

$$c = \mathbb{P}\left(\hat{y}_1 > \hat{y}_2 | y_1 \geq y_2\right). \tag{2.6}$$

Dado que en la práctica no podemos aplicar de forma directa la definición de la Ecuación (2.6), como bien se explica en [16], existen dos formas de calcular el C-index.

La primera forma de calcular el C-index es considerando los ratios de riesgos (como los que podemos obtener mediante los modelos basados en Cox), de forma que la estimación sería

$$\hat{c} = \frac{1}{|P|} \sum_{i:\delta_i=1} \sum_{j:y_i<y_j} I\left[x_i \hat{\beta}^\top > x_j \hat{\beta}^\top\right],$$

donde $i, j \in \{1, \ldots, N\}$, P denota al conjunto de los pares que se pueden formar, $I[\cdot]$ es la función indicadora y $\hat{\beta}$ es el vector de la estimación de parámetros de un modelo basado en la regresión de Cox (Ecuación (2.3)).

La otra forma de calcular el C-index sería mediante métodos de supervivencia cuya idea es directamente estimar el tiempo de supervivencia, de forma que el C-index se calcularía

mediante

$$\hat{c} = \frac{1}{|P|} \sum_{i:\delta_i=1} \sum_{j:y_i<y_j} I\left[S(\hat{y}_j|X_j) > S(\hat{y}_i|X_i)\right],$$

donde $S(\cdot)$ es la función de supervivencia estimada.

2.3.2. Error absoluto medio

El error absoluto medio (MAE, del inglés *Mean Absolute Error*) mide la diferencia absoluta esperada entre las predicciones obtenidas mediante el conjunto de prueba aplicando el modelo ajustado y los valores reales del conjunto de prueba. Formalmente, para un conjunto de prueba $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ podemos expresar la aproximación empírica del MAE (que, abusando de la notación, denotaremos del mismo modo) como

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |\hat{y}_i - y_i| \tag{2.7}$$

siendo $\hat{y}_i = \hat{f}(x_i), \forall i \in \{1, \ldots, n\}$.

El problema de utilizar esta métrica surge cuando queremos realizar dicha predicción al emplear análisis de supervivencia. Para poder afrontar los problemas que nos plantea esta situación nos basaremos en una idea desarrollada por Qi et al (2023)[11]. Por lo general, un conjunto de datos de supervivencia contienen n *time-to-event* tuplas, $\mathcal{D} = \{(x_i, t_i, \delta_i)\}_i^n$, donde $x_i \in \mathbb{R}^p - 1$ representa las características p-dimensionales de la observación i-ésima, $t_i \in \mathbb{R}_+$ representa el tiempo de evento o de censura, y $\delta_i \in \{0, 1\}$ es un indicador de la censura en el que $\delta_i = 0$ significa que la observación i-ésima presenta censura por la derecha y $\delta_i = 1$ significa que la observación i-ésima presencia el evento indicado.

En el caso de querer hacer una predicción temporal los modelos que ajustamos no van asociados a valores específicos, sino que van asociados a curvas de supervivencia. En nuestro caso, si $Y = T$ es el *time-to-event*, tenemos que las curvas de supervivencia se denotan mediante $S(t|\mathbf{x}_i) = P(T > t|X = \mathbf{x}_i)$. Sin embargo, podemos pensar en utilizar o bien la media de tiempo de supervivencia o bien la mediana de tiempo de supervivencia, que son respectivamente:

$$t_{i,mean} = \mathbb{E}_t\left[S(t|\mathbf{x}_i)\right] = \int_0^\infty S(t|\mathbf{x}_i)\, dt,$$
$$t_{i,median} = S^{-1}\left(\tau = 0.5|\mathbf{x}_i\right),$$

donde $\tau \in [0, 1]$ representa el cuantil correspondiente al nivel de probabilidad. En el caso de no alcanzar el cuantil τ fijado mediante las curvas de supervivencia proporcionadas, será

necesario realizar una extrapolación sobre la curva para obtener la estimación del cuantil.

Una vez expuesto estas dos medidas temporales podemos observar que un problema de predicción de supervivencia es similar a uno de regresión para un individuo con variables $x \in X$. Dada a esta similitud, queremos realizar una evaluación en los modelos de predicción de supervivencia haciendo uso de métricas de evaluación para modelos de regresión, como el MAE expuesto en la Ecuación (2.7). En el caso de predicción de supervivencia, el valor $y_i = t_i$ sería el tiempo de supervivencia del individuo i-ésimo y el valor $\hat{y}_i = \hat{t}_i$ sería la predicción del tiempo de supervivencia del individuo i-ésimo mediante el modelo ajustado. En nuestro caso de supervivencia, consideraríamos o bien $\hat{t}_{i,mean}$ o bien $\hat{t}_{i,median}$.
Para calcular la predicción temporal en las curvas de supervivencia en este trabajo utilizaremos el tiempo de supervivencia mediano t_{median}. Esto es debido a dos motivos: el primero es que el uso de t_{median} con el MAE es una métrica de evaluación adecuada, como se demuestra en el Teorema C.1 de [11], y el segundo es que la extrapolación solo es necesaria para curvas que no alcancen un valor de supervivencia de 0.5, mientras que para el tiempo de supervivencia medio t_{mean} es necesario para todas las curvas que no alcanzan un valor de supervivencia de 0. Una vez indicado que instancia temporal vamos a emplear para nuestra predicción, observemos que opciones tenemos para calcular el MAE.
La primera opción que se nos puede ocurrir y la más simple es utilizar los datos que no presentan censura, y a partir de ahí utilizar la Ecuación (2.7) considerando $y_i = t_{i,median}$ e $\hat{y}_i = \hat{t}_{i,median}$ de forma que calcularíamos dicho error medio absoluto solo para sujetos que experimentan el evento. Esta idea puede ser útil para cuando apenas existen eventos censurados, pero en caso de presenciar una cantidad elevada de censura nuestra métrica estaría sujeta a sesgo, ya que estaríamos ignorando una parte considerable de datos que nos pueden proporcionar información útil.
Una forma de incorporar datos que presentan censura es hacer uso de la *hinge loss*, una métrica que considera los datos censurados cuyo tiempo de predicción es inferior al tiempo que presenta con la censura. Para un dato censurado, el MAE-Hinge score es

$$\mathcal{R}_{MAE-hinge}\left(\hat{t}_i, t_i, \delta_i = 0\right) = \text{máx}\{t_i - \hat{t}_i,\ 0\}.$$

Esta métrica es optimista a la hora de obtener el valor verdadero del MAE debido a dos razones: la primera es asignar el valor 0 si la censura sucede antes del tiempo de supervivencia predicho y la segunda es que asigna una pérdida de $t_i - \hat{t}_i$ si la censura sucede antes que la predicción. En ambos casos los valores son menores o iguales al verdadero error de predicción, por lo que en un dataset con un ratio extremadamente alto de censura un modelo puede alcanzar un valor bajo MAE-Hinge por sobreestimar el tiempo de evento para todos los individuos y asignar ceros a una parte de los individuos que presentan censura, resultando en

un score total optimista.

2.3.3. Error cuadrático medio

El error cuadrático medio (MAE, del inglés *Mean Squared Error*) mide la diferencia cuadrática esperada entre las predicciones obtenidas mediante el conjunto de prueba aplicando el modelo ajustado y los valores reales del conjunto de prueba. Formalmente, para un conjunto de prueba $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ podemos expresar la aproximación empírica del MSE (que, abusando de la notación, denotaremos del mismo modo) como

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y_i} - y_i)^2 \tag{2.8}$$

siendo $\hat{y_i} = \hat{f}(x_i), \forall i \in \{1, \ldots, n\}$.

Como bien sucedía para el MAE y se discutió en la Sección 2.3.2, las medidas temporales se aplicaría de la misma manera para el MSE, haciendo uso del tiempo de supervivencia mediano t_{median} y teniendo como opciones utilizar solo los datos que presencian el evento o hacer uso de la *hinge loss*, que en este caso se definiría el MSE-Hinge score para un dato censurado como

$$\mathcal{R}_{MSE-hinge}\left(\hat{t}_i, t_i, \delta_i = 0\right) = (\text{máx}\{t_i - \hat{t}_i,\ 0\})^2.$$

2.3.4. Ratios

En el ámbito de la evaluación de modelos de aprendizaje automático, resulta especialmente relevante disponer de herramientas que permitan comparar de forma normalizada y relativa el rendimiento de diferentes algoritmos. En este sentido, se propone el uso de ratios basados en una misma métrica evaluada en distintos modelos, tomando como referencia el mejor valor alcanzado por dicha métrica entre todos los modelos considerados. Esta estrategia consiste en calcular, para cada modelo, el cociente entre su valor obtenido en una métrica específica y el valor óptimo (máximo o mínimo, según el caso) alcanzado por esa métrica en el conjunto de modelos comparados. Estas aproximaciones, al normalizar los valores de rendimiento con respecto al óptimo observado de forma que los tenemos en una escala (0,1], permite una interpretación más directa y comparable entre métricas y modelos heterogéneos, considerando que un valor próximo a 1 nos indica un rendimiento próximo sobre el mejor modelo, mientras que, cuanto menor sea el valor del ratio, observaremos una caída del rendimiento sobre el mejor modelo. Además, estos ratios favorecen una visualización compacta del comportamiento relativo de los modelos, y pueden ser utilizados como criterio adicional para la selección del modelo más adecuado, especialmente en contextos donde se valoran tanto la eficacia como la

eficiencia comparativa.

Para el caso en el que consideramos el C-index como métrica sobre una colección de modelos $M = m_1, \ldots, m_n$, el ratio se define mediante

$$Ratio_{C-index} = \frac{c_i}{c}$$

siendo c_i el C-index del modelo $m_i \in M$, y siendo $c = \text{máx}\{c_1, \ldots, c_n\}$.

No obstante, para el caso en el que consideramos una métrica de error medio (bien sea el MAE o el MSE) sobre la colección de modelos M, el ratio se define mediante

$$Ratio_{ME} = \frac{l}{l_i}$$

siendo l_i el error medio (bien sea MAE o MSE) del modelo $m_i \in M$, y siendo $l = \text{mín}\{c_1, \ldots, c_n\}$.

Capítulo 3

Ejemplos empíricos

En este capítulo se tiene como objetivo poner en práctica los fundamentos teóricos expuestos en el Capítulo 2 mediante la realización de una serie de pruebas empíricas. Para ello, se emplearán distintos conjuntos de datos que permitirán evaluar la aplicabilidad y efectividad de los conceptos vistos en el Capítulo 2.

3.1. Ajuste de la experimentación

Para el estudio en nuestro trabajo, emplearemos 3 conjuntos de datos, los cuales se mencionan en la Tabla 3.1 con el número de observaciones que contienen, la cantidad de variables explicativas y el porcentaje de observaciones censuradas que hay en cada conjunto de datos. Los datos los obtuvimos de [4] y [5].

Dataset	Nº de observaciones	% de censura	Nº de variables explicativas
HDFail	52422	94,49 %	7
FLChain	7874	72,45 %	11
Taxis	1048575	0 %	17

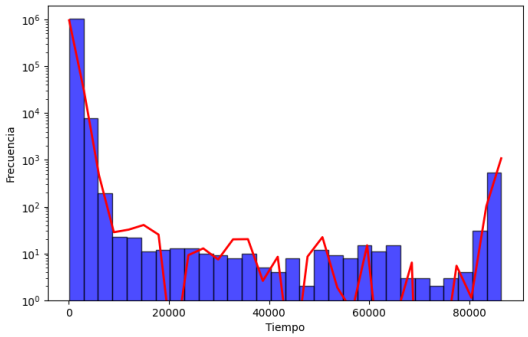
Tabla 3.1: Tabla descriptiva de los conjuntos de datos.

El conjunto de datos HDFail consiste en un estudio de mantenimiento de discos duros, en el que se considera como evento el fallo o rotura del disco duro. El conjunto de datos FLChain consiste en un estudio médico en el que se mide la cantidad de cadenas ligeras kappa y lambda libres en la sangre, que son útiles para el diagnóstico de enfermedades asociadas en la médula osea, y el evento en dicho estudio se considera la muerte del individuo. Por último, el conjunto de datos Taxis consiste en el estudio de los trayectos realizados por los taxis en Nueva York en el mes de enero de 2015, de forma que el evento a estudiar es la finalización del

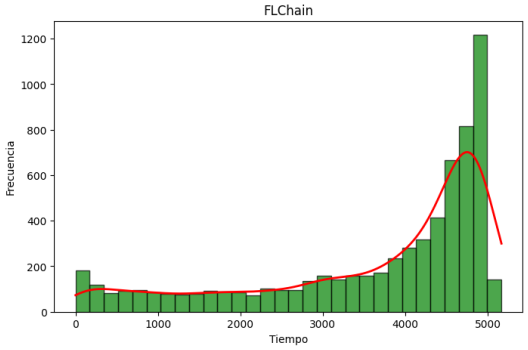
trayecto. Realizamos un análisis descriptivo de nuestros conjuntos de datos para comprender sobre que valores se manejan nuestra variable a predecir de los conjuntos de datos a estudiar, y podemos observar dicho análisis tanto en la Tabla 3.2 como en los histogramas de la Figura 3.1.

	Media	Mediana	Desviación típica	Mínimo	Máximo
HDFail (días)	685.77	530.125	531.70	0.166	11295.125
FLChain (días)	3214.51	3749.5	1592.38	0	5139
Taxis (segundos)	788.90	600	2150.96	-14040	86400

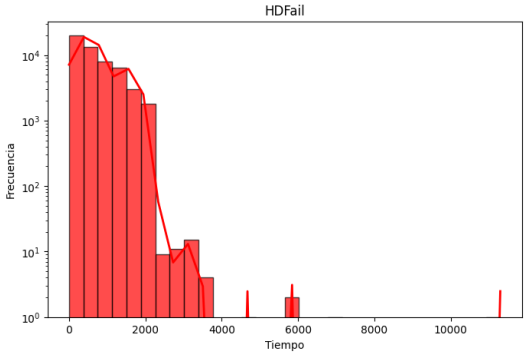
Tabla 3.2: Análisis descriptivos de los conjuntos de datos de la Tabla 3.1.



(a) Taxis



(b) FLChain



(c) HDFail

Figura 3.1: Distribución de los tiempos hasta evento para los conjuntos de datos a estudiar.

Para los conjuntos de datos mencionados realizamos una partición en el conjunto de datos del 80 % de los datos para el entrenamiento de los modelos y del 20 % restante de los datos para la evaluación de los modelos (*train-test* split), y emplearemos los siguientes algoritmos

para el caso de las técnicas de machine learning tradicional, los cuales fueron definidos en el capítulo anterior:

- Regresión.
- Random Forest.
- Gradient Boosting.

Para el caso en el que estudiaremos las técnicas de Machine Learning con herramientas de supervivencia emplearemos los algoritmos derivados de los modelos tradicionales, los cuales también se definen en el capítulo anterior, que son:

- Regresión de Cox.
- Random Forest Survival.
- Gradient Boosting Survival.

Con el fin de optimizar el rendimiento de los modelos estudiados, empleamos una estrategia de búsqueda sistemática de hiperparámetros mediante el uso de la técnica de GridSearch. Esta técnica consiste en definir una malla de búsqueda compuesta por diferentes combinaciones posibles de valores para los hiperparámetros relevantes de cada modelo, y luego evaluar exhaustivamente el desempeño del modelo para cada combinación. La evaluación se realiza utilizando una métrica de evaluación previamente seleccionada, que puede variar en función del objetivo del modelo. Esta búsqueda se complementa con validación cruzada, lo que permite estimar de forma más robusta el rendimiento generalizado de cada configuración. En nuestro caso, tomamos un 20 % del conjunto de entrenamiento para realizar dicha validación cruzada. Al finalizar el proceso, el GridSearch selecciona la combinación de hiperparámetros que obtiene el mejor desempeño promedio en las particiones del conjunto de entrenamiento, permitiendo así construir un modelo más ajustado y con mejor capacidad predictiva en nuevos datos. Esta metodología garantiza una búsqueda ordenada y exhaustiva que, aunque computacionalmente pueda resultar costosa, es altamente efectiva para encontrar configuraciones óptimas en contextos donde el ajuste fino de los hiperparámetros tiene un impacto significativo sobre el desempeño final del modelo. Para nuestro estudio, la malla definida para cada uno de los modelos es la que se muestra en la tabla 3.3.

Para los modelos Random Forest (RF), con la librería `scikit-learn` de Python [13], empleamos los parámetros `n_estimators` como la colección de parámetros, `max_depth` como la profundidad máxima de cada árbol, `min_samples_leaf` como el número mínimo de muestras necesarias para cada hoja de un nodo y `min_samples_split` como el número mínimo de

muestras requeridas para dividir un nodo. Para la construcción de los modelos que emplean herramientas de supervivencia utilizaremos la librería `scikit-survival` de `Python` [15], con los mismos parámetros definidos para los modelos Random Forest.

Para los modelos Gradient Boosting (GB), con la librería `scikit-learn` de Python [14], empleamos los parámetros `n_estimators` como la colección de parámetros, `max_depth` como la profundidad máxima de cada árbol, `subsample` como la fracción de muestras utilizadas para el ajuste de predictores base iniciales y `learning_rate` como la ponderación de cada árbol en la construcción del modelo. Para la construcción de los modelos que emplean herramientas de supervivencia utilizaremos la librería `scikit-survival` de `Python` [12] con los mismos parámetros empleados para el método Gradient Boosting.

Modelo	**Hiperparámetro**	**Valores considerados**
Random Forest (RF)	`n_estimators`	50, 100, 200
	`max_depth`	5, 10, 15
	`min_samples_split`	$n\times$ 0.01, $n\times$ 0.05
	`min_samples_leaf`	$n\times$ 0.01, $n\times$ 0.05
Random Forest Survival	`n_estimators`	50, 100, 200
	`max_depth`	5, 10, 15
	`min_samples_split`	$n\times$ 0.01, $n\times$ 0.05
	`min_samples_leaf`	$n\times$ 0.01, $n\times$ 0.05
Gradient Boosting	`learning_rate`	0.01, 0.1, 0.2
	`n_estimators`	50, 100, 200
	`subsample`	0.1
	`max_depth`	5, 10, 15
Gradient Boosting Survival	`learning_rate`	0.01, 0.1, 0.2
	`n_estimators`	50, 100, 200
	`subsample`	0.1
	`max_depth`	5, 10, 15
Regresión ElasticNet	`alpha`	0.01, 0.1, 1, 10, 100
	`l1_ratio`	0.1, 0.5, 0.7, 0.9, 1.0
Cox (Regresión de Cox)	`penalizer`	0.0001, 0.000464, 0.00215, 0.01, 0.0464, 0.215, 1, 4.64, 21.54, 100

Tabla 3.3: Valores considerados para la búsqueda de hiperparámetros en cada modelo, con n el número de observaciones que hay en el conjunto de entrenamiento.

Una vez mencionados los modelos que se van a emplear indicaremos los pasos en los que se van a sustentar el estudio. En primer lugar, se evaluará el impacto que puede tener la aplicación de técnicas de muestreo sobre el conjunto de datos de entrenamiento, lo cual podría alterar la distribución original de los datos y, por tanto, modificar la capacidad de los diversos modelos para realizar predicciones.

Además, se explorará el efecto que tiene la elección de diferentes funciones de evaluación (o funciones de score) en los procesos de búsqueda y ajuste de los hiperparámetros óptimos para cada modelo, dado que esta selección puede orientar de manera distinta el proceso de optimización y dar lugar a modelos con comportamientos distintos.

Por otra parte, se estudiará cómo la presencia de datos censurados afecta el rendimiento de aquellos modelos que incorporan técnicas de análisis de supervivencia, ya que la censura introduce desafíos adicionales en la estimación precisa del tiempo hasta la ocurrencia de un evento. En este sentido, se pondrá posteriormente especial énfasis en los métodos utilizados para realizar la extrapolación necesaria que permita estimar los tiempos de supervivencia sobre el conjunto de evaluación.

Finalmente, también se considerará el impacto de la presencia de valores atípicos o anomalías en los datos, tanto en la fase de ajuste del modelo como en la evaluación, debido a que estas observaciones pueden inducir sesgos o deteriorar el rendimiento predictivo si no se abordan adecuadamente. En conjunto, el análisis planteado permitirá una comprensión de los elementos que afectan la robustez y la precisión de los modelos considerados en este trabajo.

3.2. Impacto del muestreo

La evaluación del rendimiento de los algoritmos expuestos en la sección 3.1 requiere una estrategia de muestreo adecuada que permita obtener resultados representativos y comparables. En este contexto, es fundamental analizar cómo distintos niveles de muestreo impactan en la estimación del desempeño de los modelos, valorando el nivel de ajuste bajo las métricas de evaluación definidas en el capítulo anterior y observando los beneficios e inconvenientes que nos aporta dicho muestreo, sobre todo en el costo computacional. Este apartado tiene como objetivo contrastar diferentes niveles de muestreo para determinar cuales son más beneficiosos para nuestra comparativa.

En este caso, las comparativas a nivel muestral solo las aplicamos en el conjunto de datos Taxis, ya que los conjuntos de datos FLChain y HDFail presentan un número de instancias

muy bajo como para realizar este tipo de muestreos. Realizamos la comparativa empleando el conjunto de datos al completo, un muestreo aleatorio del 10 % de los datos y un muestreo aleatorio del 5 % de los datos.

	RF	**RF (10 %)**	**RF (5 %)**
C-index (ratio)	0.856 (0.999)	0.857 (1)	0.853 (0.995)
MAE (ratio)	260.32 (0.988)	258.99 (0.993)	257.36 (1)
MSE (ratio)	$4.395 \cdot 10^6$ (0.998)	$4.393 \cdot 10^6$ (1)	$4.397 \cdot 10^6$(0.999)
Tiempo total (seg)	$1.590 \cdot 10^4$	$3.856 \cdot 10^3$	919.08
Tiempo mejor modelo (seg)	88.58	88.09	6.03

Tabla 3.4: Comparativa de RF con distintos niveles de muestreo en el conjunto de datos Taxis.

Como podemos observar en la tabla 3.4 las métricas no sufren una gran variación a la hora de realizar ambos niveles de muestreo, por lo que tenemos un ajuste similar de la población que estamos estudiando. Por otra parte, podemos observar una gran mejoría en el tiempo de ajuste tanto del mejor modelo como del tiempo total en la búsqueda de los hiperparámetros del mejor modelo, por lo que el planteamiento de aplicar el muestreo para nuestro conjunto de datos es idóneo. Además, este comportamiento que observamos en la tabla 3.4 es análogo para los algoritmos de Gradient Boosting y Regresión Lineal, por lo que aplicar un muestreo aleatorio en nuestro conjunto de datos es beneficioso para nuestro estudio, debido a que nos proporciona un gran ahorro computacional a la hora de ajustar los modelos.

Este estudio de muestreo solo lo realizamos sobre los modelos tradicionales de Machine Learning y no lo realizamos en los modelos que aplican herramientas de supervivencia. Esto fue ocasionado por el coste computacional que empleaban dichos modelos. Como podemos observar en la tabla 3.5, los modelos que precisan herramientas de supervivencia emplean de mucho más tiempo de búsqueda de los hiperparámetros óptimos bajo un GridSearch y del ajuste de los mejores modelos, hasta tal punto que en el modelo de Gradient Boosting Survival para un muestreo del 10 % no obtuvimos un ajuste con más de dos días de entrenamiento.

	RF	**RF survival**	**GB**
GridSearch (Muestra 10 %)	$3.856 \cdot 10^3$	56543.61	726.28
Mejor modelo (Muestra 10 %)	88.094	$1.282 \cdot 10^3$	7.567
GridSearch (Muestra 5 %)	919.08	$1.211 \cdot 10^4$	246.90
Mejor modelo (Muestra 5 %)	6.028	266.875	3.176
	GB survival	**Regresión Lineal**	**Regresión de Cox**
GridSearch (Muestra 10 %)	+	146.92	-
Mejor modelo (Muestra 10 %)	$3.371 \cdot 10^4$	3.124	-
GridSearch (Muestra 5 %)	+	804.97	33.45
Mejor modelo (Muestra 5 %)	$1.741 \cdot 10^3$	57.065	1.73

Tabla 3.5: Tiempos de entrenamiento por modelos para el conjunto de datos Taxis (en segundos).

Debido al gran coste computacional que existe en los modelos de supervivencia y al observar en los modelos tradicionales que la aplicación de los niveles de muestreo del 10 % y 5 % no afectaban de manera negativa en el ajuste de los modelos, aplicaremos un muestreo del 5 % del conjunto de datos para el ajuste y comparativa de los modelos para las siguientes pruebas. Destacar que en la tabla 3.5 no hay tiempos para la regresión de Cox (representados con -) dado que con ese conjunto de datos el algoritmo obtenía un problema de convergencia y para el algoritmo de Gradient Boosting Survival el coste computacional era tan elevado que no terminamos de obtener resultados (representados con +).

3.3. Score de selección

A la hora de realizar la búsqueda de hiperparámetros bajo un GridSearch, tenemos dos funciones score como opción de validación cruzada para la selección del mejor modelo, que es el uso del MSE y el uso del C-index. Para realizar la comparativa de los modelos ajustados con diferentes funciones score empleamos una comparativa entre los conjuntos de datos de Taxis con un muestreo del 5 % de los datos en el conjunto de entrenamiento, las observaciones no censuradas del conjunto de datos FLChain y las observaciones no censuradas del conjunto de datos HDFail. Destacar que para estos casos en los conjuntos de datos FLChain y HDFail, el modelo de regresión de Cox no se pudo ajustar por problemas de convergencia.

Como podemos observar en las Tablas 3.6, 3.7 y 3.8, la elección de la función score no influye en el ajuste de los modelos tradicionales, pero si observamos una diferencia entre los modelos que emplean herramientas de supervivencia.

Métrica	**RF**	**RF survival**	**GB**	**GB survival**	**Regresión**	**Cox**
C-index (Score C-index)	0.975	0.897	0.982	1	0.937	0.934
C-index (Score MSE)	0.973	0.868	0.976	1	0.935	0.932
MAE (Score C-index)	1	0.483	0.731	0.599	0.865	0.599
MAE (Score MSE)	1	0.537	0.846	0.599	0.865	0.599
MSE (Score C-index)	1	0.945	0.975	0.944	0.991	0.944
MSE (Score MSE)	1	0.950	0.977	0.944	0.991	0.944

Tabla 3.6: Ratios de las métricas de evaluación para el conjunto de datos Taxis.

Métrica	**RF**	**RF survival**	**GB**	**GB survival**	**Regresión**
C-index (Score C-index)	0.987	0.976	0.979	0.970	1
C-index (Score MSE)	0.991	0.966	0.991	0.980	1
MAE (Score C-index)	0.974	0.971	0.912	0.915	1
MAE (Score MSE)	0.987	0.952	0.927	1	0.952
MSE (Score C-index)	0.958	0.950	0.875	0.874	1
MSE (Score MSE)	0.970	0.926	0.978	0.978	1

Tabla 3.7: Ratios de las métricas de evaluación para las observaciones no censuradas del conjunto de datos FLChain.

Métrica	**RF**	**RF survival**	**GB**	**GB survival**	**Regresión**
C-index (Score C-index)	0.964	0.933	0.991	0.989	1
C-index (Score MSE)	0.968	0.920	0.989	0.977	1
MAE (Score C-index)	0.987	0.625	0.981	0.852	1
MAE (Score MSE)	0.988	0.620	0.983	1	0.927
MSE (Score C-index)	0.993	1	0.272	0.215	0.139
MSE (Score MSE)	0.989	1	0.242	0.157	0.146

Tabla 3.8: Ratios de las métricas de evaluación para las observaciones no censuradas del conjunto de datos HDFail.

Con el objetivo de analizar visualmente las diferencias entre las predicciones generadas por los modelos y los valores reales observados en el conjunto de evaluación, se llevó a cabo una representación gráfica basada en nubes de puntos. En estas representaciones, cada punto refleja la diferencia entre el valor predicho y el valor real correspondiente a una observación específica. Los valores positivos en el eje vertical indican que la predicción del modelo ha sobreestimado

el valor real, mientras que los valores negativos señalan una estimación inferior al valor real. Este tipo de visualización permite una interpretación clara y directa de la distribución de errores del modelo a lo largo del eje temporal. Acompañando a estas nubes de puntos, se incluyó en el propio gráfico un histograma que resume la distribución de los valores reales en función del tiempo, lo cual facilita la identificación de posibles patrones o concentraciones de eventos. Dichas visualizaciones se presentan en las Figuras 3.3, 3.4 y 3.5, correspondientes a los diferentes conjuntos de datos analizados.

Adicionalmente, en el caso particular del conjunto de datos Taxis, se elaboró una comparación gráfica específica entre las diferencias de predicción obtenidas al ajustar el modelo utilizando las dos funciones score empleadas durante el proceso de búsqueda de hiperparámetros: el error cuadrático medio (MSE) y el índice de concordancia (C-index). En esta comparación, los valores positivos de la nube de puntos indican que la predicción realizada con los hiperparámetros seleccionados mediante el C-index supera a aquella obtenida utilizando el MSE como función de evaluación. Por el contrario, los valores negativos señalan una mejor predicción por parte del modelo ajustado con base en el MSE. Esta representación, complementada también con un histograma de distribución temporal de los valores reales, se encuentra ilustrada en la Figura 3.2, manteniendo la coherencia visual y analítica con las Figuras previamente mencionadas.

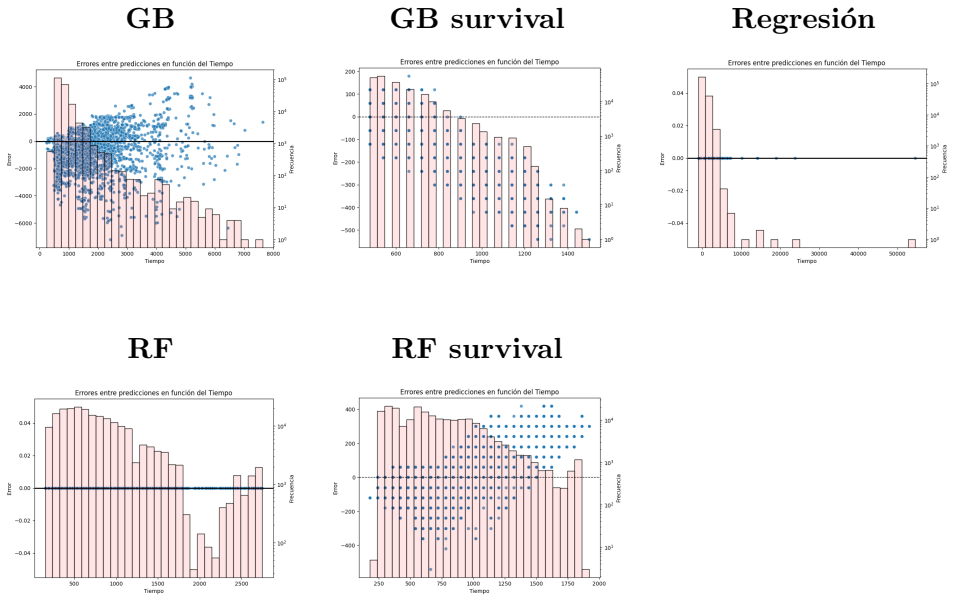


Figura 3.2: Diferencias entre las predicciones de los modelos obtenidos con función de score MSE y C-index

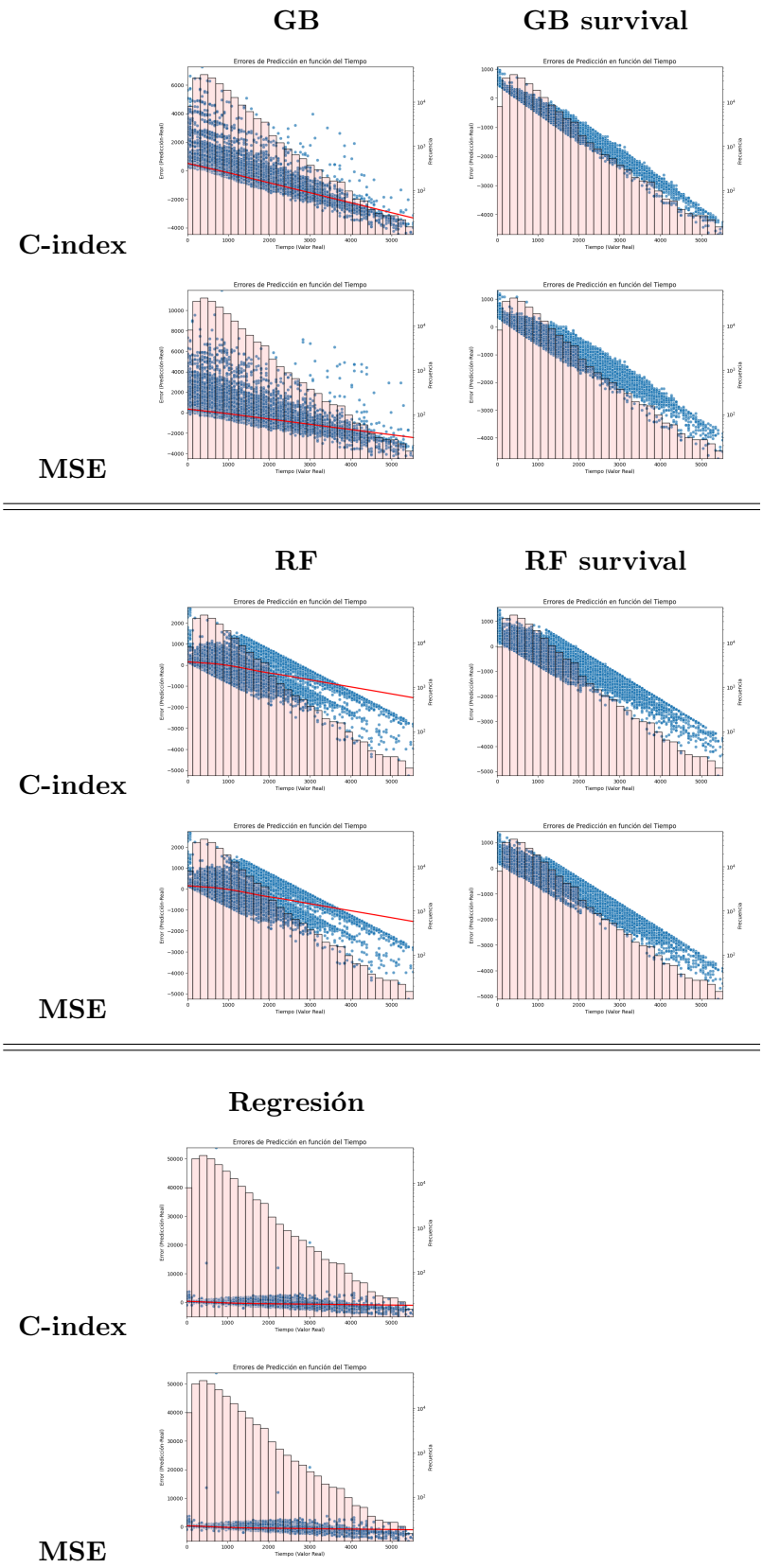


Figura 3.3: Diferencias entre valores reales y predicciones de los modelos en el conjunto de datos Taxis

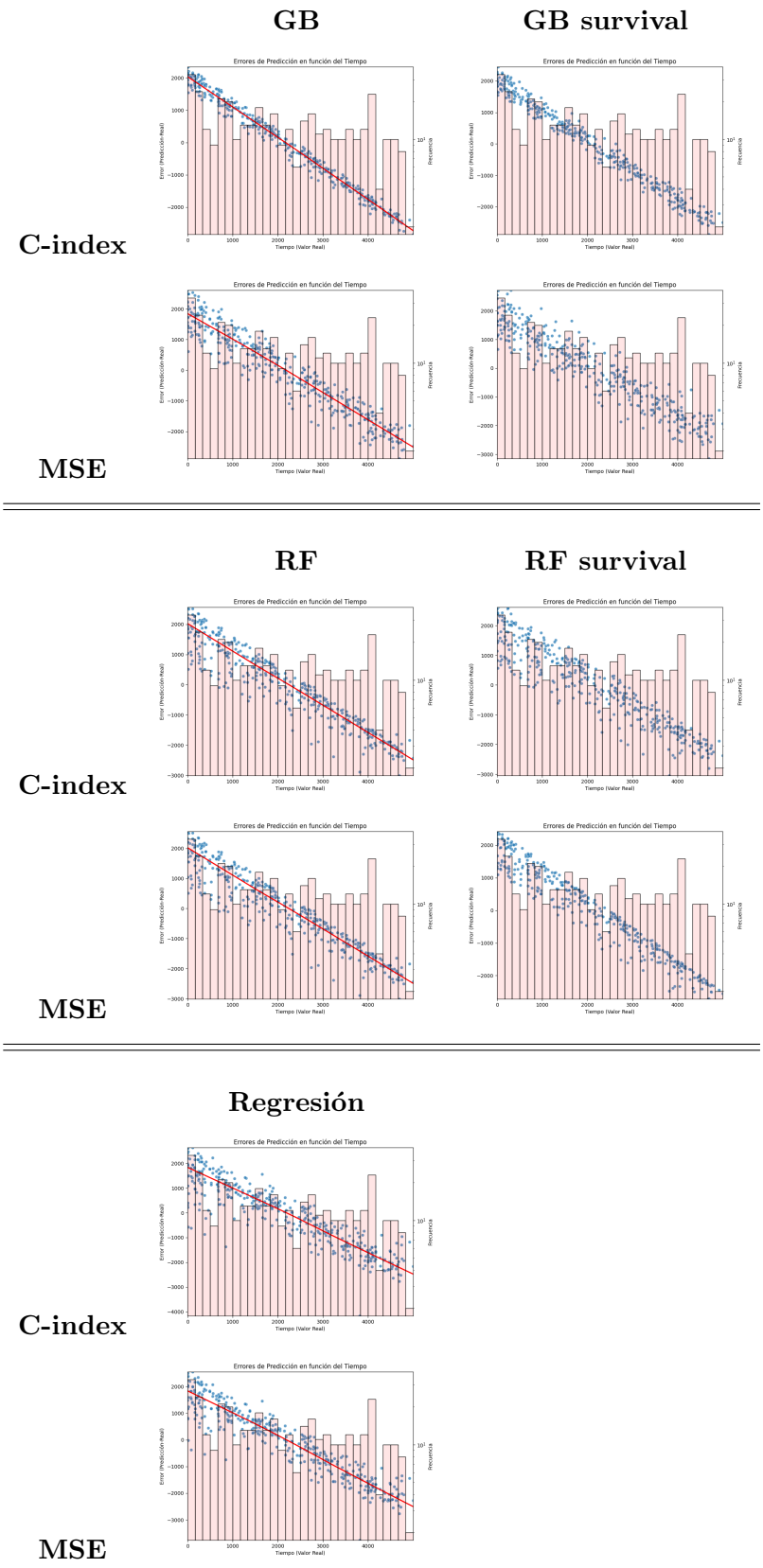


Figura 3.4: Diferencias entre valores reales y predicciones de los modelos en el conjunto de datos FLChain

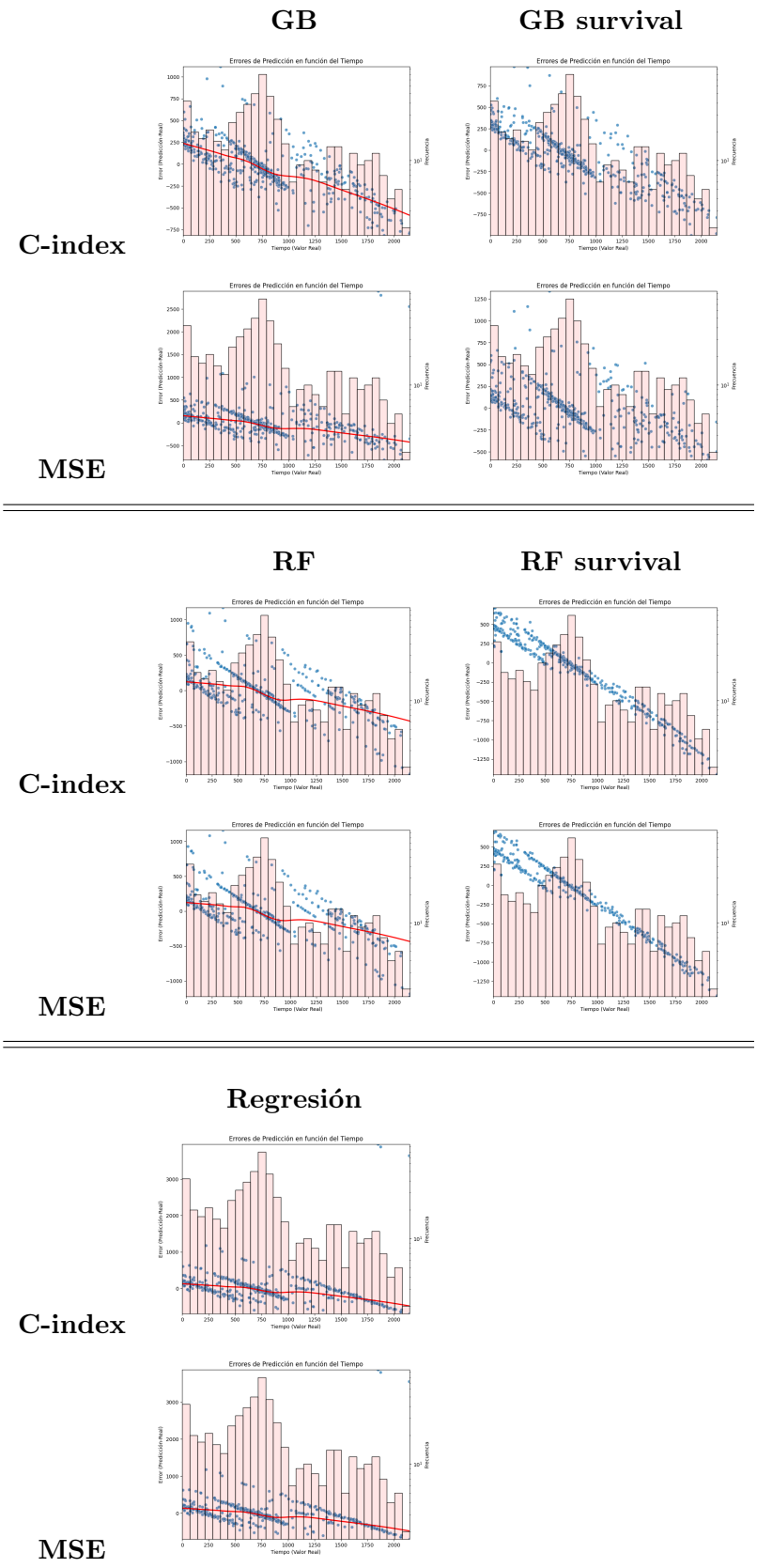


Figura 3.5: Diferencias entre valores reales y predicciones de los modelos en el conjunto de datos HDFail

Tal como se puede apreciar en las Figuras 3.3, 3.4 y 3.5, no se evidencia una diferencia notoria o sistemática en los errores de predicción cometidos por los modelos cuando se emplea como función de puntuación el **C-index** en comparación con el **MSE** durante el proceso de ajuste de hiperparámetros. En efecto, al observar las representaciones gráficas correspondientes, se advierte que los modelos tienden a cometer errores similares en instancias temporales equivalentes, independientemente de la función de score utilizada. Esta similitud sugiere que, en términos generales, el tipo de métrica empleada para guiar el ajuste del modelo no tiene un impacto visualmente evidente en la calidad de las predicciones a lo largo del tiempo, al menos desde la perspectiva gráfica de las diferencias entre valores reales y estimados. Por tanto, no se observa una clara superioridad de una función de puntuación sobre la otra en el comportamiento predictivo global de los modelos considerados.

No obstante, al centrar el análisis en el conjunto de datos Taxis, y en particular en la Figura 3.2, se puede advertir un matiz adicional que podría resultar relevante bajo ciertos enfoques analíticos. Aunque, de forma general, las diferencias entre las predicciones obtenidas con cada función de score no revelan un patrón que indique una ventaja clara a lo largo de toda la ventana temporal del estudio, sí pueden observarse ciertas áreas específicas del dominio temporal en las que un modelo ajustado con una función de score determinada parece ofrecer mejores resultados que el otro. Por ejemplo, en el caso del modelo Gradient Boosting Survival, se observa que para valores temporales más elevados, las predicciones generadas mediante el ajuste guiado por el MSE tienden a ser más precisas que aquellas obtenidas utilizando el C-index. De manera opuesta, cuando se analiza el comportamiento del modelo Random Forest Survival, se evidencia que en ese mismo intervalo temporal, las predicciones realizadas tras un ajuste basado en el C-index superan en precisión a las obtenidas a través del MSE. Estos hallazgos, aunque puntuales, abren la posibilidad de que, dependiendo del contexto o del interés específico en ciertos rangos temporales, pueda ser beneficioso seleccionar una función de score particular que favorezca el rendimiento del modelo en la zona de interés. Por tanto, aunque el análisis global no sugiere una diferencia determinante, el análisis localizado podría ofrecer información valiosa para aplicaciones prácticas que se centren en ventanas temporales específicas.

Por tanto, tras los resultados obtenidos y comentarios realizados a lo largo de este apartado, para el resto del estudio en nuestros conjuntos de datos, como no existe un ajuste claramente mejor dependiendo de la función de score que empleemos, trabajaremos a partir de ahora con los modelos obtenidos con función de score C-index.

3.4. Simulación con censura

Una vez visto en el apartado anterior que emplearemos la función de score C-index para el ajuste de nuestros modelos, estudiaremos como es el impacto de la censura en los modelos que emplean herramientas de supervivencia. Para ello, definiremos 3 niveles de censura a la hora de realizar el estudio con los algoritmos de supervivencia, que son los niveles del 0 %, 20 % y 50 %. Destacar que el conjunto de entrenamiento para los algoritmos con el nivel del 0 % de censura coincide con el conjunto de entrenamiento para los algoritmos que no emplean herramientas de supervivencia y necesitan de observaciones que presencien el evento.

Para el caso del conjunto de datos de Taxis tenemos que no existen observaciones censuradas, pero para observar el efecto de la censura en los modelos que emplean herramientas de supervivencia podemos simular la censura de manera artificial. Para esta situación, se plantearon dos casos de censura artificial:

- **Caso 1**: Censura aleatoria uniforme. Para este caso, seleccionamos el porcentaje de observaciones que nos resulte de interés, de manera que en el intervalo de tiempo que se genera [0,t_i], siendo t_i el tiempo de evento para la observación i-ésima, seleccionamos un valor aleatorio bajo la distribución $Unif[0, t_i]$. En este caso, los valores de las diversas métricas para cada uno de los algoritmos se reflejan en la tabla 3.9.

- **Caso 2**: Censura aleatoria bajo una distribución Weibull. Bajo el análisis de supervivencia se puede buscar una parametrización del conjunto de datos mediante una función de supervivencia, de forma que posteriormente podemos fijar los tiempos obtenidos bajo la parametrización como los valores de censura de nuestro conjunto de datos. En este caso, la distribución que mejor se asemeja es la distribución Weibull, que se define de la siguiente forma

$$F(t; \lambda, k) = 1 - e^{-(t/\lambda)^k}, \quad t \geq 0, \tag{3.1}$$

  y la función de supervivencia correspondiente es:

$$S(t; \lambda, k) = e^{-(t/\lambda)^k} \tag{3.2}$$

  Para generar un conjunto de datos con el % de censura de interés debemos tener en cuenta el parámetro de forma $(1/\lambda)$. Si realizamos Monte Carlo para el conjunto de datos de Taxis de 500 simulaciones, observamos que el parámetro de forma es 2.75 para obtener un 20 % de censura y es 0.75 para obtener un 50 % de censura. En este caso, los valores de las diversas métricas para cada uno de los algoritmos se reflejan en la tabla 3.10.

Métrica	RF	RF survival	GB	GB survival	Regresión
C-index (0 %)	0.853	0.785	0.859	0.875	0.820
C-index (20 %)	*	0.775	*	0.870	*
C-index (50 %)	*	0.764	*	0.868	*
MAE (0 %)	257.36	532.68	351.77	429.09	297.47
MAE (20 %)	*	256.53	*	432.69	*
MAE (50 %)	*	294.49	*	442.54	*
MSE (0 %)	$4.397 \cdot 10^6$	$4.650 \cdot 10^6$	$4.507 \cdot 10^6$	$4.657 \cdot 10^6$	$4.435 \cdot 10^6$
MSE (20 %)	*	$4.432 \cdot 10^6$	*	$4.638 \cdot 10^6$	*
MSE (50 %)	*	$4.440 \cdot 10^6$	*	$4.626 \cdot 10^6$	*

Tabla 3.9: Comparación de métricas entre modelos con distintos niveles de censura artificial aleatoria uniforme para el dataset Taxis.

Métrica	RF	RF survival	GB	GB survival	Regresión
C-index (0 %)	0.8536	0.7852	0.8596	0.8751	0.8205
C-index (20 %)	*	0.8179	*	0.8750	*
C-index (50 %)	*	0.8213	*	0.8727	*
MAE (0 %)	257.36	532.68	351.77	429.09	297.47
MAE (20 %)	*	257.46	*	360.09	*
MAE (50 %)	*	293.05	*	352.13	*
MSE (0 %)	$4.397 \cdot 10^6$	$4.650 \cdot 10^6$	$4.507 \cdot 10^6$	$4.657 \cdot 10^6$	$4.435 \cdot 10^6$
MSE (20 %)	*	$4.442 \cdot 10^6$	*	$4.569 \cdot 10^6$	*
MSE (50 %)	*	$4.488 \cdot 10^6$	*	$4.576 \cdot 10^6$	*

Tabla 3.10: Comparación de métricas entre modelos con distintos niveles de censura artificial aleatoria Weibull para el dataset Taxis.

La etiqueta * en las Tablas nos indica que no se realizó entrenamiento con los conjuntos de datos que contienen observaciones censuradas, ya que los modelos tradicionales no contemplan dichas observaciones para los entrenamientos. Al emplear censura los modelos de regresión de Cox no fueron posibles de ajustar por problemas de convergencia en este conjunto de datos, por lo que no los tenemos en cuenta para la comparativa en las Tablas 3.9 y 3.10.

Al observar los valores obtenidos en las métricas de evaluación presentadas en las Tablas 3.9 y 3.10 la introducción de censura en los datos tiene un impacto significativo en el rendimiento de los modelos de supervivencia. Este impacto se manifiesta de forma directa en la capacidad

predictiva de los modelos diseñados para manejar observaciones censuradas. Sin embargo, a pesar de su supuesta adecuación teórica a este tipo de datos, los modelos que incorporan mecanismos explícitos para el tratamiento de censura no logran superar en desempeño a los modelos tradicionales que operan únicamente con observaciones completas. Específicamente, en el caso del conjunto de datos Taxis, y bajo el conjunto de métricas de evaluación que hemos establecido, se observa de manera consistente que los modelos convencionales, que no consideran censura, presentan un rendimiento superior. Esto sugiere que, para este conjunto de datos en particular, el beneficio teórico de utilizar modelos de supervivencia capaces de manejar censura no se traduce en una mejora práctica en términos de predicción.

En lo que respecta a los conjuntos de datos FLChain y HDFail, el enfoque adoptado para evaluar el impacto de la censura difiere del utilizado con el conjunto Taxis. En estos casos, se realizó una selección inicial que incluye únicamente aquellas observaciones que presentan un evento observado (es decir, sin censura), y posteriormente se incorporaron observaciones censuradas de forma aleatoria hasta alcanzar el porcentaje de censura deseado en cada experimento. Esta metodología permite controlar de manera precisa el grado de censura presente en los datos y facilita una comparación más clara del efecto que tiene este factor sobre el rendimiento de los modelos. A partir de esta estrategia experimental, y como se refleja en las métricas recopiladas en las Tablas 3.11 y 3.12, es posible analizar con mayor profundidad el comportamiento de los modelos bajo diferentes niveles de censura. Bajo este escenario de estudio de la censura obtenemos distintos resultados dependiendo del conjunto de datos. Para el conjunto de datos FLChain podemos observar una notoria mejoría en las métricas de evaluación a medida que aumentamos el nivel de censura, obteniendo mejores valores en las métricas. Esto no sucede para el conjunto de datos HDFail, donde las métricas incluso empeoran agregando observaciones censuradas a los modelos que emplean técnicas de análisis de supervivencia.

Métrica	RF	RF survival	GB	GB survival	Regresión	Cox
C-index (0 %)	0.643	0.636	0.637	0.631	0.651	-
C-index (20 %)	*	0.732	*	0.727	*	0.728
C-index (50 %)	*	0.873	*	0.877	*	0.874
MAE (0 %)	$1.130 \cdot 10^3$	$1.135 \cdot 10^3$	$1.208 \cdot 10^3$	$1.201 \cdot 10^3$	$1.101 \cdot 10^3$	-
MAE (20 %)	*	$1.117 \cdot 10^3$	*	$1.126 \cdot 10^3$	*	$1.123 \cdot 10^3$
MAE (50 %)	*	$1.089 \cdot 10^3$	*	$1.122 \cdot 10^3$	*	$1.057 \cdot 10^3$
MSE (0 %)	$1.755 \cdot 10^6$	$1.769 \cdot 10^6$	$1.914 \cdot 10^6$	$1.899 \cdot 10^6$	$1.683 \cdot 10^6$	-
MSE (20 %)	*	$1.716 \cdot 10^6$	*	$1.701 \cdot 10^6$	*	$1.738 \cdot 10^6$
MSE (50 %)	*	$1.608 \cdot 10^6$	*	$1.682 \cdot 10^6$	*	$1.593 \cdot 10^6$

Tabla 3.11: Comparación de métricas entre modelos con distintos niveles de censura para el conjunto de datos FLChain.

Métrica	RF	RF survival	GB	GB survival	Regresión
C-index (0 %)	0.792	0.766	0.813	0.812	0.821
C-index (20 %)	*	0.680	*	0.753	*
C-index (50 %)	*	0.722	*	0.783	*
MAE (0 %)	191.73	302.83	190.90	219.55	192.01
MAE (20 %)	*	309.82	*	349.46	*
MAE (50 %)	*	372.96	*	276.18	*
MSE (0 %)	$7.794 \cdot 10^4$	$1.831 \cdot 10^5$	$6.726 \cdot 10^4$	$8.543 \cdot 10^4$	$7.180 \cdot 10^4$
MSE (20 %)	*	$1.791 \cdot 10^5$	*	$2.120 \cdot 10^5$	*
MSE (50 %)	*	$3.177 \cdot 10^5$	*	$1.638 \cdot 10^5$	*

Tabla 3.12: Comparación de métricas entre modelos con distintos niveles de censura para el conjunto de datos HDFail.

Con los resultados que se muestran en la Tabla 3.11 podemos observar que para el conjunto de datos FLChain el uso de la censura sí que mejora el rendimiento de los ajustes, obteniendo mejores resultados en los modelos que emplean herramientas de supervivencia que los modelos tradicionales que no emplean observaciones censuradas. Destacar que en este conjunto de datos el modelo de regresión de Cox solo se pudo ajustar sin problemas de convergencia para los conjuntos de datos que presenciaban censura.

Con los resultados que se muestran en la Tabla 3.12 podemos observar que para el conjunto de datos HDFail la censura no tiene el mismo efecto que en el conjunto de datos FLChain. En este conjunto de datos podemos observar que no se obtiene un modelo de regresión de Cox

ajustado para ninguno de los 3 casos, y para los modelos restantes que emplean herramientas de supervivencia observamos que las métricas de evaluación muestran peores valores que los modelos tradicionales que no emplean observaciones censuradas.

A partir de los resultados obtenidos en los distintos niveles de censura y conjuntos de datos analizados, se puede concluir que el impacto de la inclusión de observaciones censuradas en los modelos de predicción no responde a una tendencia generalizable o uniforme. Por el contrario, el efecto de la censura sobre el rendimiento de los modelos depende de manera significativa tanto del nivel específico de censura aplicado como de las características particulares del conjunto de datos considerado. Esta dependencia contextual sugiere que la incorporación de censura en el modelado requiere un análisis caso por caso, ya que su influencia puede variar sustancialmente en función de la estructura y distribución de los datos, así como de la naturaleza del evento estudiado.

3.5. Tipo de extrapolación

Como explicamos en el capítulo anterior, al trabajar con modelos de supervivencia, cuando no podíamos obtener de manera directa para la observación i-ésima el valor temporal t_i debido a que la curva de supervivencia no paramétrica generada no alcanza la probabilidad de supervivencia 0.5 o inferior (esto es $S(t) > 0.5 \ \forall t \in D$ siendo D el conjunto de tiempos que se observan en la curva de supervivencia no paramétrica) teníamos dos opciones de realizar la estimación del cuantil 0.5. Estas opciones eran la extrapolación lineal y la extrapolación exponencial (definidas en el capítulo previo). Estas extrapolaciones las vamos a observar únicamente en los modelos GB survival y RF survival, debido a que el ajuste en la regresión de Cox, como bien observamos en apartados previos, suele ser problemática para los conjuntos de datos con los cuales estamos trabajando de forma que no se alcanza un ajuste por problemas de convergencia. Realizando el estudio para los niveles de censura vistos en el apartado previo para los conjuntos de datos correspondientes de la tabla 3.1, podemos observar los valores de las métricas para las diferentes extrapolaciones en las Tablas 3.13, 3.14 y 3.15

Métrica	RF survival		GB survival	
	Lineal	**Exponencial**	**Lineal**	**Exponencial**
MAE (0 %)	532.68	256.37	429.09	355.60
MAE (20 %)	256.53	256.53	432.69	349.30
MAE (50 %)	294.49	294.49	442.54	342.34
MSE (0 %)	$4.650 \cdot 10^6$	$4.440 \cdot 10^6$	$4.657 \cdot 10^6$	$4.538 \cdot 10^6$
MSE (20 %)	$4.432 \cdot 10^6$	$4.432 \cdot 10^6$	$4.638 \cdot 10^6$	$4.546 \cdot 10^6$
MSE (50 %)	$4.440 \cdot 10^6$	$4.440 \cdot 10^6$	$4.626 \cdot 10^6$	$4.481 \cdot 10^6$

Tabla 3.13: Comparación de métricas entre modelos con distintas extrapolaciones para el conjunto de datos Taxis.

Métrica	RF survival		GB survival	
	Lineal	**Exponencial**	**Lineal**	**Exponencial**
MAE (0 %)	$1.135 \cdot 10^3$	$1.135 \cdot 10^3$	$1.201 \cdot 10^3$	$1.201 \cdot 10^3$
MAE (20 %)	$1.117 \cdot 10^3$	$1.117 \cdot 10^3$	$1.126 \cdot 10^3$	$1.126 \cdot 10^3$
MAE (50 %)	$1.089 \cdot 10^3$	$1.089 \cdot 10^3$	$1.122 \cdot 10^3$	$1.122 \cdot 10^3$
MSE (0 %)	$1.769 \cdot 10^6$	$1.769 \cdot 10^6$	$1.899 \cdot 10^6$	$1.899 \cdot 10^6$
MSE (20 %)	$1.716 \cdot 10^6$	$1.716 \cdot 10^6$	$1.701 \cdot 10^6$	$1.701 \cdot 10^6$
MSE (50 %)	$1.608 \cdot 10^6$	$1.608 \cdot 10^6$	$1.682 \cdot 10^6$	$1.682 \cdot 10^6$

Tabla 3.14: Comparación de métricas entre modelos con distintas extrapolaciones para el conjunto de datos FLChain.

Métrica	RF survival		GB survival	
	Lineal	**Exponencial**	**Lineal**	**Exponencial**
MAE (0 %)	302.83	302.83	219.55	219.55
MAE (20 %)	309.82	309.82	349.46	349.46
MAE (50 %)	372.96	372.96	276.18	276.18
MSE (0 %)	$1.831 \cdot 10^5$	$1.831 \cdot 10^5$	$8.543 \cdot 10^4$	$8.543 \cdot 10^4$
MSE (20 %)	$1.791 \cdot 10^5$	$1.791 \cdot 10^5$	$2.120 \cdot 10^5$	$2.120 \cdot 10^5$
MSE (50 %)	$3.177 \cdot 10^5$	$3.177 \cdot 10^5$	$1.638 \cdot 10^5$	$1.638 \cdot 10^5$

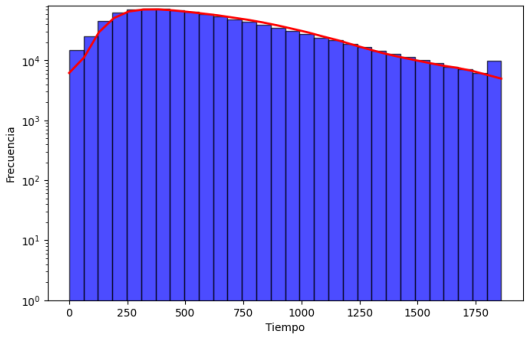
Tabla 3.15: Comparación de métricas entre modelos con distintas extrapolaciones para el conjunto de datos HDFail.

Destacamos que en la tabla 3.13 hicimos uso de los valores que obtuvimos mediante la censura artificial uniforme. Como podemos observar en la tabla 3.13, el uso de la extrapolación exponencial nos permite obtener unas métricas de error medio con valores más bajos, por lo que son unas métricas más precisas para nuestro conjunto de evaluación para el conjunto de datos Taxis. Por otra parte, tanto para el conjunto de datos FLChain como HDFail, si observamos los resultados mostrados en las Tablas 3.14 y 3.15 las métricas de error medio son iguales independientemente del tipo de extrapolación que empleemos. Esto sucede debido a que al realizar el ajuste de los modelos que emplean herramientas de supervivencia las curvas estimadas para nuestros modelos siempre alcanzan el cuantil 0.5 para la supervivencia de los individuos, por lo que no se emplea nunca el recurso de la extrapolación para extraer un tiempo estimado de supervivencia para los individuos que conforman nuestro conjunto de evaluación.

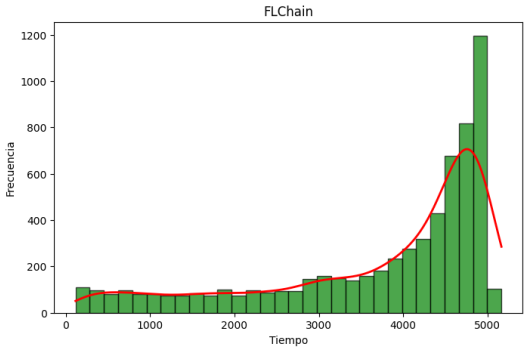
3.6. Impacto de los outliers

Con el objetivo de mejorar la calidad de los datos y, por ende, la robustez de los modelos predictivos empleados, en esta sección se analiza el efecto del filtrado de valores atípicos sobre el rendimiento de distintos enfoques de modelado. Para ello, se aplicó un proceso de detección y eliminación de outliers utilizando el rango intercuartílico (IQR) sobre cada uno de los conjuntos de datos considerados definidos en la tabla 3.1, de manera que consideraremos valores atípicos todos los valores que no se encuentren en el intervalo $[Q_1 - 1.5\text{IQR},\ Q_3 + 1.5\text{IQR}]$, siendo Q_1 el primer cuartil, Q_3 el tercer cuartil e $\text{IQR} = Q_3 - Q_1$. Este método, al centrarse en la dispersión central de los datos, permite reducir el impacto de observaciones extremas que podrían sesgar las métricas de evaluación o influir negativamente en el entrenamiento de los modelos que estamos estudiando.

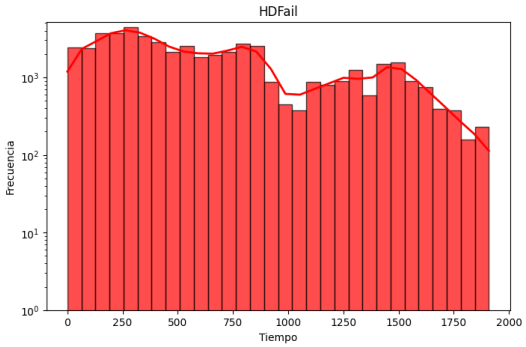
Si realizamos el filtrado de los datos haciendo uso del rango intercuartílico bajo los conjuntos de datos Taxis, FLChain y HDFail que se muestran en la Tabla 3.1 en el tiempo hasta evento de cada conjunto de datos podemos observar como varían el número de observaciones en la tabla y las distribuciones de dichas observaciones podemos observarlas en los histogramas de la Figura 3.6.



(a) Taxis



(b) FLChain



(c) HDFail

Figura 3.6: Distribución de los tiempos hasta evento para los conjuntos de datos a estudiar con filtrado de outliers.

Una vez realizado ese filtrado de datos, pasaremos a estudiar las métricas de evaluación empleadas en los casos anteriores con los conjuntos de datos filtrados y observar si existe una variación en comparación con los resultados previos. Para el conjunto de datos Taxis, recordemos que trabajamos con la muestra del 5 % de los datos y que este conjunto de datos se caracteriza por no tener censura. En la tabla 3.16 podemos observar una comparativa entre las métricas con la muestra original y con la muestra de los datos filtrados bajo IQR.

Para los conjuntos de datos HDFail y FLChain recordemos que existía presencia de censura, por lo que observaremos las métricas para los 3 niveles de censura de 0 %, 20 % y 50 % que empleamos en la sección . En las Tablas 3.17 y 3.18 podemos observar una comparativa entre las métricas con la muestras originales y con la muestras de los datos filtrados bajo IQR

	RF	**RF survival**	**GB**	**GB survival**	**Regresión**	**Cox**
C-index	0.853	0.785	0.859	0.875	0.820	0.818
C-index filtrado	0.856	0.838	0.872	0.870	0.830	0.821
MAE	257.36	532.68	351.77	429.09	297.47	429.09
MAE filtrado	154.52	373.01	224.59	316.79	205.15	256.50
MSE	$4.397 \cdot 10^6$	$4.650 \cdot 10^6$	$4.507 \cdot 10^6$	$4.657 \cdot 10^6$	$4.435 \cdot 10^6$	$4.657 \cdot 10^6$
MSE filtrado	$4.409 \cdot 10^4$	$1.890 \cdot 10^5$	$8.101 \cdot 10^4$	$1.564 \cdot 10^5$	$7.298 \cdot 10^4$	$5.273 \cdot 10^5$

Tabla 3.16: Comparación de modelos para el conjunto de datos Taxis entre filtrado IQR y sin filtrado

Métrica		**RF**	**RF survival**	**GB**	**GB survival**	**Regresión**
C-index (0 %)	Sin filtrar	0.792	0.766	0.813	0.812	0.821
	Filtrado	0.784	0.767	0.806	0.786	0.799
C-index (20 %)	Sin filtrar	0.796	0.680	0.801	0.753	0.813
	Filtrado	0.775	0.664	0.798	0.762	0.799
C-index (50 %)	Sin filtrar	0.776	0.722	0.793	0.783	0.808
	Filtrado	0.783	0.697	0.791	0.763	0.803
MAE (0 %)	Sin filtrar	191.73	302.83	190.90	219.55	189.24
	Filtrado	184.70	277.64	175.50	192.01	177.67
MAE (20 %)	Sin filtrar	210.82	309.82	208.57	349.46	198.32
	Filtrado	207.34	292.40	205.65	239.34	196.96
MAE (50 %)	Sin filtrar	246.95	372.96	233.81	276.18	230.20
	Filtrado	221.94	346.98	211.92	263.6391	212.79
MSE (0 %)	Sin filtrar	$7.794 \cdot 10^4$	$1.831 \cdot 10^5$	$6.726 \cdot 10^4$	$8.543 \cdot 10^4$	$1.314 \cdot 10^5$
	Filtrado	$7.011 \cdot 10^4$	$1.476 \cdot 10^5$	$6.038 \cdot 10^4$	$7.180 \cdot 10^4$	$6.041 \cdot 10^4$
MSE (20 %)	Sin filtrar	$8.435 \cdot 10^4$	$1.791 \cdot 10^5$	$7.966 \cdot 10^4$	$2.120 \cdot 10^5$	$7.426 \cdot 10^4$
	Filtrado	$7.773 \cdot 10^4$	$1.554 \cdot 10^5$	$7.286 \cdot 10^4$	$9.899 \cdot 10^4$	$6.732 \cdot 10^4$
MSE (50 %)	Sin filtrar	$1.260 \cdot 10^5$	$3.177 \cdot 10^5$	$1.176 \cdot 10^5$	$1.638 \cdot 10^5$	$1.001 \cdot 10^5$
	Filtrado	$8.455 \cdot 10^4$	$2.362 \cdot 10^5$	$7.733 \cdot 10^4$	$1.293 \cdot 10^5$	$7.300 \cdot 10^4$

Tabla 3.17: Comparación de modelos para el conjunto de datos HDFail entre filtrado IQR y sin filtrado para diferentes niveles de censura.

Métrica		RF	RF survival	GB	GB survival	Regresión	Cox
C-index (0 %)	Sin filtrar	0.643	0.636	0.637	0.631	0.651	-
	Filtrado	0.615	0.606	0.609	0.595	0.594	-
C-index (20 %)	Sin filtrar	0.666	0.732	0.653	0.727	0.648	0.728
	Filtrado	0.644	0.745	0.636	0.747	0.650	0.748
C-index (50 %)	Sin filtrar	0.664	0.873	0.655	0.877	0.713	0.874
	Filtrado	0.660	0.857	0.645	0.860	0.692	0.860
MAE (0 %)	Sin filtrar	$1.130 \cdot 10^3$	$1.135 \cdot 10^3$	$1.208 \cdot 10^3$	$1.201 \cdot 10^3$	$1.101 \cdot 10^3$	-
	Filtrado	$1.057 \cdot 10^3$	$1.057 \cdot 10^3$	$1.069 \cdot 10^3$	$1.082 \cdot 10^3$	$1.069 \cdot 10^3$	-
MAE (20 %)	Sin filtrar	$1.266 \cdot 10^3$	$1.117 \cdot 10^3$	$1.287 \cdot 10^3$	$1.126 \cdot 10^3$	$1.391 \cdot 10^3$	$1.123 \cdot 10^3$
	Filtrado	$1.264 \cdot 10^3$	$1.079 \cdot 10^3$	$1.274 \cdot 10^3$	$1.108 \cdot 10^3$	$1.294 \cdot 10^3$	$1.071 \cdot 10^3$
MAE (50 %)	Sin filtrar	$1.566 \cdot 10^3$	$1.089 \cdot 10^3$	$1.568 \cdot 10^3$	$1.122 \cdot 10^3$	$1.501 \cdot 10^3$	$1.057 \cdot 10^3$
	Filtrado	$1.473 \cdot 10^3$	$1.089 \cdot 10^3$	$1.566 \cdot 10^3$	$1.426 \cdot 10^3$	$1.509 \cdot 10^3$	$1.053 \cdot 10^3$
MSE (0 %)	Sin filtrar	$1.755 \cdot 10^6$	$1.769 \cdot 10^6$	$1.914 \cdot 10^6$	$1.899 \cdot 10^6$	$1.683 \cdot 10^6$	-
	Filtrado	$1.511 \cdot 10^6$	$1.532 \cdot 10^6$	$1.532 \cdot 10^6$	$1.603 \cdot 10^6$	$1.576 \cdot 10^6$	-
MSE (20 %)	Sin filtrar	$2.147 \cdot 10^6$	$1.716 \cdot 10^6$	$2.221 \cdot 10^6$	$1.701 \cdot 10^6$	$2.537 \cdot 10^6$	$1.738 \cdot 10^6$
	Filtrado	$2.129 \cdot 10^6$	$1.609 \cdot 10^6$	$2.160 \cdot 10^6$	$1.645 \cdot 10^6$	$2.219 \cdot 10^6$	$1.636 \cdot 10^6$
MSE (50 %)	Sin filtrar	$3.019 \cdot 10^6$	$1.608 \cdot 10^6$	$3.046 \cdot 10^6$	$1.682 \cdot 10^6$	$2.761 \cdot 10^6$	$1.593 \cdot 10^6$
	Filtrado	$2.688 \cdot 10^6$	$1.618 \cdot 10^6$	$3.056 \cdot 10^6$	$2.880 \cdot 10^6$	$2.819 \cdot 10^6$	$1.574 \cdot 10^6$

Tabla 3.18: Comparación de modelos para el conjunto de datos FLChain entre filtrado IQR y sin filtrado para diferentes niveles de censura.

Tal como se observa en las Tablas 3.16, 3.17 y 3.18, la aplicación de un proceso de filtrado para la detección y eliminación de observaciones anómalas como es el criterio del rango intercuartílico (IQR) tiene un efecto claramente beneficioso sobre el rendimiento de los modelos evaluados en términos de error medio. En los tres conjuntos de datos analizados, el filtrado de outliers conduce a una reducción sistemática en los valores de error medio para la totalidad de los modelos considerados, lo cual sugiere que estas observaciones extremas actúan como fuentes de perturbación que comprometen la precisión de las predicciones. Esta disminución de los errores medios respalda la eficacia de una etapa de preprocesamiento orientada a la limpieza de datos como componente crucial en la mejora del rendimiento predictivo.

Sin embargo, y a pesar de esta mejora general en las métricas cuantitativas, es importante destacar que el filtrado de outliers no conlleva un cambio significativo en la jerarquía de desempeño relativa entre los modelos. Es decir, aunque todos los modelos se ven beneficiados en términos absolutos, la comparación entre ellos permanece inalterada. Esto se evidencia, por ejemplo, en la Tabla 3.16, correspondiente al conjunto de datos Taxis, donde los modelos tradicionales continúan mostrando un rendimiento superior respecto a aquellos orientados al análisis de supervivencia. En particular, el modelo Random Forest sigue destacando como el de menor error medio, consolidando su posición como el más preciso dentro de ese conjunto.

Además, en lo que respecta al C-index, observamos una variación en la superioridad relativa de los modelos: donde previamente el modelo Gradient Boosting Survival era el mejor clasificado, observamos que su contraparte tradicional, el modelo Gradient Boosting, logra obtener el mejor valor tras el filtrado de outliers.

Estos resultados permiten inferir que, si bien la limpieza de datos mediante la eliminación de valores atípicos mejora de manera global el ajuste y precisión de todos los modelos, dicha mejora no modifica las conclusiones respecto a cuál modelo es preferible en cada caso. Esto refuerza la robustez de la comparación entre modelos y pone de manifiesto la relevancia de considerar tanto la calidad de los datos como la estabilidad de las métricas al momento de seleccionar un modelo para tareas de predicción en contextos con observaciones censuradas o no censuradas.

Capítulo 4

Conclusiones

Para finalizar este Trabajo Fin de Máster en este capítulo presentan las conclusiones que se pueden extraer a raíz de los Capítulos 2 y 3. En esta sección se resumen los hallazgos más relevantes, se reflexiona sobre las implicaciones de los resultados obtenidos y se discuten las posibles limitaciones del enfoque adoptado. Asimismo, se sugieren líneas futuras de investigación que podrían complementar o extender los resultados alcanzados en el presente trabajo.

Primero de todo, fuimos capaces de definir de manera precisa el análisis de supervivencia y Machine Learning, de forma que pudimos establecer una relación entre ambos conceptos bajo modelos de Machine Learning que precisan de herramientas de análisis de supervivencia. A su vez, también definimos métricas de evaluación que nos permiten realizar comparativas entre modelos de Machine Learning que precisan herramientas de supervivencia y modelos de Machine Learning tradicionales, es decir, modelos que no precisan de dichas herramientas de supervivencia.

Una vez definidos los conceptos necesarios en el Capítulo 2, procedimos a realizar pruebas con diversos conjuntos de datos en el Capítulo 3 que nos permitieron abarcar diversos puntos interesantes en el estudio. En primer lugar, pudimos observar que los modelos que precisan herramientas de supervivencia tienen un mayor coste computacional que aquellos modelos que no precisan de dichas herramientas. Siendo conocedores de esta característica con el coste computacional, en el caso de trabajar con conjuntos de datos que contengan una gran cantidad de observaciones en el escenario del Machine Learning (considérense +100000 observaciones) es de utilidad aplicar un muestreo aleatorio sobre el conjunto de datos ya que nos permite reducir dicho coste computacional.

Otra prueba de interés que realizamos en este trabajo fue realizar un ajuste de hiperparáme-

tros dependiendo de varias funciones de puntuación (score) para los diversos modelos que empleamos, pudiendo realizar un análisis para poder seleccionar que función sería idónea para cada modelo y/o para el intervalo temporal de interés.

El estudio del impacto de la censura fue otro punto interesante realizado en el presente trabajo. Pudimos observar en el conjunto de datos con más observaciones, que era el conjunto Taxis, que los modelos que precisan de herramientas de supervivencia no proporcionaban ni un mejor ajuste ni una mejor predicción sobre nuevas observaciones en comparación con los modelos tradicionales. En los conjuntos de datos FLChain y HDFail no podemos concluir la misma idea que para el conjunto de datos Taxis, ya que al trabajar con un conjunto de entrenamiento con una cantidad reducida de observaciones se da la situación de que en ciertos escenarios los modelos que emplean herramientas de supervivencia tienen un mejor rendimiento que los modelos tradicionales.

La extrapolación resultó ser una herramienta de gran utilidad para resolver el problema de la estimación del tiempo en las situaciones donde las curvas de supervivencia no alcanzan el cuantil 0.5. Además, fuimos capaces de plantear dos tipos de extrapolaciones, la cuál solo fue necesaria de emplear sobre el conjunto de datos Taxis y en la que pudimos observar que para ese conjunto de datos tenía mejores valores para las métricas de evaluación la extrapolación Weibull frente a la extrapolación lineal.

Por último, también empleamos una técnica de filtrado de anomalías, que fue haciendo uso del rango intercuartílico, y nos permitió obtener mejores ajustes y predicciones para todos los modelos planteados y los conjuntos de datos empleados, pero no generó un cambio en la preferencia de los modelos, es decir, los que se consideraban los mejores modelos antes del filtrado seguían considerándose los mejores modelos una vez realizado el filtrado de datos.

Apéndice A

Código

```
import pandas as pd
import numpy as np
import time
from sklearn.model_selection import train_test_split
from sksurv.ensemble import RandomSurvivalForest,
    ComponentwiseGradientBoostingSurvivalAnalysis
from sklearn.model_selection import KFold, GridSearchCV
from sksurv.metrics import concordance_index_censored
from sklearn.ensemble import RandomForestRegressor,
    GradientBoostingRegressor
from datetime import datetime
from scipy.optimize import curve_fit
import pickle
from sklearn.linear_model import ElasticNet
from lifelines import CoxPHFitter
from sklearn.metrics import mean_absolute_error, mean_squared_error
from itertools import combinations
from lifelines.utils import concordance_index
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KernelDensity
from scipy.optimize import curve_fit
```

Listing A.1: Librerías empleadas

```
import pandas as pd
table = pd.read_csv("./datasets/yellow_tripdata_2015-01.csv", sep=";")
table = pd.DataFrame(table)
```

```
table['tpep_pickup_datetime'] = pd.to_datetime(table['
    tpep_pickup_datetime'], format = '%d/%m/%Y %H:%M')
table['tpep_dropoff_datetime'] = pd.to_datetime(table['
    tpep_dropoff_datetime'], format = '%d/%m/%Y %H:%M')
table['time'] = table['tpep_dropoff_datetime'] - table['
    tpep_pickup_datetime']
table['time_seconds'] = table['time'].dt.total_seconds()
table['week_day'] = table['tpep_pickup_datetime'].dt.day_name()
table['hour'] = table['tpep_pickup_datetime'].dt.hour
table['event'] = 1
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
datos = table.drop(columns=['fare_amount','extra','mta_tax','
    tip_amount','tolls_amount','improvement_surcharge','total_amount'])
columns_to_check = ['dropoff_latitude', 'dropoff_longitude', '
    pickup_latitude', 'pickup_longitude']
mask = datos[columns_to_check].apply(lambda x: x.str.contains('E|e')).
    any(axis=1)
datos = datos[~mask]
X = datos.drop(columns = ['tpep_pickup_datetime', '
    tpep_dropoff_datetime','store_and_fwd_flag','time','event','
    time_seconds'])
X = pd.get_dummies(X,columns=['week_day'])
data = datos.drop(columns = ['tpep_pickup_datetime', '
    tpep_dropoff_datetime','store_and_fwd_flag','time'])
data = pd.get_dummies(data,columns=['week_day'])
datos['event'] = datos['event'].astype(bool)
surv_data = np.array([(e, t) for e, t in zip(datos['event'], datos['
    time_seconds'])],
dtype=[('event', '?'), ('time', '<f8')])
y = datos['time_seconds']
def transformar_coordenada(coord):
coord_sin_puntos = coord.replace('.', '')
coord_final = coord_sin_puntos[:3] + '.' + coord_sin_puntos[3:]
return float(coord_final)
X['dropoff_latitude'] = X['dropoff_latitude'].apply(
    transformar_coordenada)
X['dropoff_longitude'] = X['dropoff_longitude'].apply(
    transformar_coordenada)
X['pickup_latitude'] = X['pickup_latitude'].apply(
```

```
    transformar_coordenada)
X['pickup_longitude'] = X['pickup_longitude'].apply(
    transformar_coordenada)
scaler = StandardScaler()
X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
data['dropoff_latitude'] = data['dropoff_latitude'].apply(
    transformar_coordenada)
data['dropoff_longitude'] = data['dropoff_longitude'].apply(
    transformar_coordenada)
data['pickup_latitude'] = data['pickup_latitude'].apply(
    transformar_coordenada)
data['pickup_longitude'] = data['pickup_longitude'].apply(
    transformar_coordenada)
X_train, X_test, y_train, y_test, y_surv_train, y_surv_test =
    train_test_split(X, y, surv_data, test_size=0.2, random_state=42)
data_train, data_test = train_test_split(data, test_size=0.2,
    random_state=42)
```

Listing A.2: Preprocesado del conjunto de datos Taxis.

```
import pandas as pd
from SurvSet.data import SurvLoader
loader = SurvLoader()
table, ref = loader.load_dataset(ds_name='flchain').values()
print(f"Dimensión del Tablon: {table.shape}")
table = pd.DataFrame(table)
table = pd.get_dummies(table,columns=['fac_chapter','num_flc_grp'])
table['fac_sex'] = table['fac_sex'].map({'M':1,'F':0})
table['fac_mgus'] = table['fac_mgus'].map({'Y':1,'N':0})
print(table.head())
filas_con_nan = table[table.isna().any(axis=1)]
table = table.dropna()
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
#Si queremos usar todo el dataset, no utilizar las siguientes lineas
    hasta definir X
#Caso de que queramos el 50% de censura en el Dataset
#sample_size = (table['event']==1).sum()
#sampled_rows = table.query("event == 0").sample(n=sample_size,
    random_state=3)
```

```
#table = pd.concat([table.query("event == 1"), sampled_rows]).
    reset_index(drop=True)
#Caso en el que queramos el 20% de censura en el Dataset
#sample_size = int(0.25 * (table['event']==1).sum() )
#sampled_rows = table.query("event == 0").sample(n=sample_size,
    random_state=3)
#table = pd.concat([table.query("event == 1"), sampled_rows]).
    reset_index(drop=True)
#Caso en el que no empleemos datos censurados
#table = table[table['event'] == 1]
X = table.drop(columns = ['pid', 'event','time'])
X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
surv_data = np.array([(e, t) for e, t in zip(table['event'], table['
    time'])],
dtype=[('event', '?'), ('time', '<f8')])
y = table['time']

X_train, X_test, y_train, y_test, y_surv_train, y_surv_test =
    train_test_split(X, y, surv_data, test_size=0.2, random_state=42)
datos = table.drop(columns= ['pid'])
data_train, data_test = train_test_split(datos, test_size=0.2,
    random_state=42)
```

Listing A.3: Preprocesado del conjunto de datos FLChain.

```
import pandas as pd
from SurvSet.data import SurvLoader
loader = SurvLoader()
table, ref = loader.load_dataset(ds_name='hdfail').values()
print(f"Dimensión del Tablon: {table.shape}")
table = pd.DataFrame(table)
print(table.head())
table = pd.get_dummies(table,columns=['fac_brand','fac_type'])
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
#Si queremos usar todo el dataset, no utilizar las siguientes lineas
    hasta definir X
#Caso de que queramos el 50% de censura en el Dataset
#sample_size = (table['event']==1).sum()
#sampled_rows = table.query("event == 0").sample(n=sample_size,
```

```
    random_state=3)
#table = pd.concat([table.query("event == 1"), sampled_rows]).
    reset_index(drop=True)
#Caso en el que queramos el 20% de censura en el Dataset
#sample_size = int(0.25 * (table['event']==1).sum() )
#sampled_rows = table.query("event == 0").sample(n=sample_size,
    random_state=3)
#table = pd.concat([table.query("event == 1"), sampled_rows]).
    reset_index(drop=True)
#Caso en el que no empleemos datos censurados
#table = table[table['event'] == 1]
X = table.drop(columns = ['pid', 'event','time'])
X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
surv_data = np.array([(e, t) for e, t in zip(table['event'], table['
    time'])],
dtype=[('event', '?'), ('time', '<f8')])
y = table['time']
X_train, X_test, y_train, y_test, y_surv_train, y_surv_test =
    train_test_split(X, y, surv_data, test_size=0.2, random_state=42)
datos = table.drop(columns= ['pid'])
data_train, data_test = train_test_split(datos, test_size=0.2,
    random_state=42)
```

Listing A.4: Preprocesado del conjunto de datos HDFail.

```
import matplotlib.pyplot as plt
from scipy.stats import gaussian_kde
# Crear el histograma
plt.figure(figsize=(8, 5))
counts, bins, _ = plt.hist(y, bins=30, color='red', alpha=0.7,
    edgecolor='black')
plt.yscale("log")
# Añadir etiquetas y título
plt.xlabel("Tiempo")
plt.ylabel("Frecuencia")
plt.ylim(10**0, None)
kde = gaussian_kde(y)
x_vals = np.linspace(min(y), max(y), 30)
kde_vals = kde(x_vals) * len(y) * (bins[1] - bins[0])
plt.plot(x_vals, kde_vals, color='red', linewidth=2, label='Densidad
    kernel')
# Mostrar la gráfica
```

```
plt.show()
```

Listing A.5: Generación de histogramas para los tiempos hasta evento.

```
def ajuste_random_forest_survival(X, y, score):
#Definimos el split y la malla de interés
#Ajustar porcentual
param_grid = {
        'n_estimators': [50, 100, 200],
        'max_depth': [5, 10, 15],
        'min_samples_split': [int(X.shape[0] * 0.01), int(X.shape[0] *
            0.05)],
        'min_samples_leaf': [int(X.shape[0] * 0.01), int(X.shape[0] *
           0.05)]
}

#Inicializacion del modelo y cv
rsf_model = RandomSurvivalForest(random_state=42)
cv=KFold(n_splits=3, shuffle=True, random_state=42)

#Aplicar el gridsearch
start_time = time.time()
grid_search = GridSearchCV(estimator=rsf_model,param_grid=param_grid,
    cv=cv, scoring = score,verbose=3)

grid_search.fit(X,y)
end_time = time.time()

#Guardar en un archivo .pkl
fecha_actual = datetime.now().strftime("%Y-%m-%d_%H-%M")
nombre_pkl = f"gridsearch_RFS_model_{fecha_actual}.pkl"
with open(nombre_pkl, "wb") as file:
pickle.dump(grid_search, file)

#Extraer resultados de ajuste y escribirlos en un archivo .txt
cv_results = grid_search.cv_results_
nombre_archivo = f"gridsearch_RFS_results_{fecha_actual}.txt"
with open(nombre_archivo, "w") as file:
file.write(f"Tiempo total de ejecucion: {end_time - start_time:.2f}
    segundos\n\n")
file.write("Resultados por combinacion de parametros:\n")
for i in range(len(cv_results["params"])):
```

```
file.write(f"Parametros: {cv_results['params'][i]}\n")
file.write(f"  Media del puntaje: {cv_results['mean_test_score'][i]:.4
    f}\n")
file.write(f"  Desviacion estandar: {cv_results['std_test_score'][i
    ]:.4f}\n")
file.write(f"  Tiempo de ajuste (s): {cv_results['mean_fit_time'][i
    ]:.4f}\n")
file.write("\n")
file.write(f"Mejores parametros: {grid_search.best_params_}\n")
file.write(f"Mejor puntaje (C-index): {grid_search.best_score_:.4f}\n"
    )

return nombre_pkl, nombre_archivo
```

Listing A.6: GridSearch Random Forest Survival

```
def ajuste_gradient_boosting_survival(X, y, score):
# Definir el modelo
param_grid = {
        "learning_rate": [0.01, 0.1],
        "n_estimators": [50, 100, 200],
        "subsample" : [0.1],
        "max_depth" : [5, 10, 15]
}
#Inicializacion del modelo y cv
model = GradientBoostingSurvivalAnalysis()
cv=KFold(n_splits=3, shuffle=True, random_state=42)

#Aplicar el gridsearch
grid_search = GridSearchCV(estimator=model,param_grid=param_grid,cv=cv
    ,scoring = score,verbose=3)
start_time = time.time()
grid_search.fit(X,y)
end_time = time.time()

fecha_actual = datetime.now().strftime("%Y-%m-%d_%H-%M")
nombre_pkl = f"gridsearch_GBS_model_{fecha_actual}.pkl"
with open(nombre_pkl, "wb") as file:
pickle.dump(grid_search, file)

#Extraer resultados de ajuste y escribirlos en un archivo .txt
cv_results = grid_search.cv_results_
```

```
nombre_archivo = f"gridsearch_GBS_results_{fecha_actual}.txt"
with open(nombre_archivo, "w") as file:
file.write(f"Tiempo total de ejecucion: {end_time - start_time:.2f}
    segundos\n\n")
file.write("Resultados por combinacion de parametros:\n")
for i in range(len(cv_results["params"])):
file.write(f"Parametros: {cv_results['params'][i]}\n")
file.write(f"  Media del puntaje: {cv_results['mean_test_score'][i]:.4
    f}\n")
file.write(f"  Desviacion estandar: {cv_results['std_test_score'][i
    ]:.4f}\n")
file.write(f"  Tiempo de ajuste (s): {cv_results['mean_fit_time'][i
    ]:.4f}\n")
file.write("\n")
file.write(f"Mejores parametros: {grid_search.best_params_}\n")
file.write(f"Mejor puntaje (C-index): {grid_search.best_score_:.4f}\n"
    )

return nombre_pkl, nombre_archivo
```

Listing A.7: GridSearch Gradient Boosting Survival

```
ef regressor_traditional(X,y,alpha,l1_ratio):
model = ElasticNet(alpha=alpha,l1_ratio=l1_ratio,random_state=42)
start_time = time.time()
model.fit(X,y)
end_time = time.time()

fecha_actual = datetime.now().strftime("%Y-%m-%d_%H-%M")
nombre_pkl = f"regressor_model_{fecha_actual}.pkl"
with open(nombre_pkl, "wb") as file:
pickle.dump(model, file)

#Extraer resultados de ajuste y escribirlos en un archivo .txt
nombre_archivo = f"regressor_results_{fecha_actual}.txt"
with open(nombre_archivo, "w") as file:
file.write(f"Parámetros: Alpha: {alpha}, L1_ratio:{l1_ratio}\n\n")
file.write(f"Tiempo total de ejecucion: {end_time - start_time:.2f}
    segundos\n\n")

return nombre_pkl, nombre_archivo
```

Listing A.8: GridSearch Regresión

```
def test_evaluation_c_index(X_pred,y_test,nombre_archivo,modelo_pkl):
with open(modelo_pkl, "rb") as file:
loaded = pickle.load(file)
try:
best_model = loaded.best_estimator_
except AttributeError:
best_model = loaded
y_pred_risk = best_model.predict(X_pred)
c_index = concordance_index_censored(y_test['event'], y_test['time'],
    y_pred_risk)[0]
with open(nombre_archivo, "a") as file:
file.write("\n")
file.write(f"C-Index test: {c_index:.4f}\n")
return c_index

def test_evaluation_traditional_c_index(X_pred,y_test,nombre_archivo,
    modelo_pkl):
with open(modelo_pkl, "rb") as file:
loaded = pickle.load(file)
try:
best_model = loaded.best_estimator_
except AttributeError:
best_model = loaded
y_pred = best_model.predict(X_pred)
c_index = concordance_index(y_test, y_pred)
with open(nombre_archivo, "a") as file:
file.write("\n")
file.write(f"C-Index test: {c_index:.4f}\n")
return c_index

def mse_survival(y_test, X_pred, modelo_pkl, nombre_archivo):
# Filtrar los casos donde el evento es 1 (evento ocurrió)
event_occured = y_test['event'] == 1  # Filtrar donde el evento es 1
# Obtener los tiempos reales y las predicciones correspondientes
y_real = y_test['time'][event_occured]  # Solo tiempos reales donde el
    evento ocurrió
with open(modelo_pkl, "rb") as file:
loaded = pickle.load(file)
try:
best_model = loaded.best_estimator_
except AttributeError:
```

```
best_model = loaded
median_times = median_times_lineal(best_model, X_pred)
y_pred = np.array(median_times)[event_occured]  # Predicciones de tiempo donde el evento ocurrió
# Verificar si hay eventos ocurridos para evitar errores
if len(y_real) == 0:
raise ValueError("No hay eventos (evento = 1) en los datos para calcular el MSE.")
# Calcular el MSE: Promedio del error cuadrático
mse = np.mean((y_real - y_pred) ** 2)
with open(nombre_archivo, "a") as file:
file.write("\n")
file.write(f"MSE test: {mse:.4f}\n")
return mse

def mae_survival(y_test, X_pred, modelo_pkl, nombre_archivo):
# Filtrar los casos donde el evento es 1 (evento ocurrió)
event_occured = y_test['event'] == 1  # Filtrar donde el evento es 1
# Obtener los tiempos reales y las predicciones correspondientes
y_real = y_test['time'][event_occured]  # Solo tiempos reales donde el evento ocurrió
X_test = X_pred[event_occured]
with open(modelo_pkl, "rb") as file:
loaded = pickle.load(file)
try:
best_model = loaded.best_estimator_
except AttributeError:
best_model = loaded

median_times = median_times_lineal(best_model, X_test)
y_pred = np.array(median_times)# Predicciones de tiempo donde el evento ocurrió
# Verificar si hay eventos ocurridos para evitar errores
if len(y_real) == 0:
raise ValueError("No hay eventos (evento = 1) en los datos para calcular el MAE.")
# Calcular el MAE: Promedio del error absoluto
errores_abs = np.abs(y_real - y_pred)
mae = np.mean(errores_abs)
with open(nombre_archivo, "a") as file:
file.write("\n")
```

```
75 file.write(f"MAE test: {mae:.4f}\n")
76 return mae
```

Listing A.9: Métricas de evaluación

Bibliografía

[1] Andersen, P. K., Klein, J. P., & Rosthøj, S. (2003). Generalised linear models for correlated pseudo-observations, with applications to multi-state models. *Biometrika*, 90(1): 15-27.

[2] Bentéjac, C., Csörg, A., & Martínez-Muñoz, G. (2021). *A comparative analysis of gradient boosting algorithms.* Artificial Intelligence Review, 54, 1937-1967.

[3] Breiman, L. (2001). *Random forests.* Machine learning, 45, 5-32.

[4] Drysdale, E. (2022). *SurvSet: An open-source time-to-event dataset repository.* arXiv. https://arxiv.org/abs/2203.03094

[5] Elemento. (2016) *NYC Yellow Taxi Trip Data [Conjunto de datos].* Kaggle. https://www.kaggle.com/datasets/elemento/nyc-yellow-taxi-trip-data

[6] Ergisi, S., Erdogan, B. D., & Yavuz, Y. (2024). Performance Comparison of Least Squares, Ridge, Lasso and Principal Component Regression for Addressing Multicollinearity in Regression Analysis. *Statistik Artrma Dergisi*, 14(2), 59-72.

[7] Friedman, J. H. (2001). *Greedy function approximation: a gradient boosting machine.* Annals of statistics, 1189-1232.medicine, 14(1), 73-82.

[8] Haider, H., Hoehn, B., Davis, S., and Greiner, R. *Effective ways to build and evaluate individual survival distributions.* Journal of Machine Learning Research, 21(85): 1?63, 2020.

[9] Klein, J. P., & Moeschberger, M. L. (2006). *Survival analysis: techniques for censored and truncated data.* Springer Science & Business Media.

[10] Murphy, K. P. (2012). *Machine Learning: a Probabilistic Perspective.* MIT Press.

[11] Qi, S. A., Kumar, N., Farrokh, M., Sun, W., Kuan, L. H., Ranganath, R., ... & Greiner, R. (2023). *An effective meaningful way to evaluate survival models.* arXiv preprint arXiv:2306.01196.

[12] Scikit-survival developers. (2023). *Boosting user guide [Jupyter Notebook]*. GitHub. Recuperado de https://github.com/sebp/scikit-survival/blob/v0.23.1/doc/user_guide/boosting.ipynb

[13] Scikit-learn developers. (2023). *RandomForestRegressor*. GitHub. Recuperado de https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[14] Scikit-learn developers. (2023). *GradientBoostingRegressor*. GitHub. Recuperado de https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html

[15] Scikit-survival developers. (2023). *sksurv.ensemble.RandomSurvivalForest*. GitHub. Recuperado de https://scikit-survival.readthedocs.io/en/latest/api/generated/sksurv.ensemble.RandomSurviva

[16] Wang, P., Li, Y., & Reddy, C. K. (2019). Machine learning for survival analysis: A survey. *ACM Computing Surveys* (CSUR), **51(6)**, 1–36.