unir
LA UNIVERSIDAD
EN INTERNET

La Universidad en Internet

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

MAESTRÍA EN ANÁLISIS Y VISUALIZACIÓN DE DATOS MASIVOS

Proyecto Final de Maestría

Imputación de Datos mediante Métodos Estadísticos, Aprendizaje Automático y Aprendizaje Profundo

UNIR México

Presenta

Diego Alberto Morales Ibáñez

5296049

Directora: Patricia Rayón Villela

Guatemala, febrero de 2023

Índice general

Índice de figuras

Índice de cuadros

CAPÍTULO 1

Resumen

Se propone una metodología para la comparación de imputación mediante métodos estadísticos convencionales, métodos de aprendizaje automático y métodos de aprendizaje profundo a partir de una medición de desempeño y tiempo de ejecución para variables numéricas (regressión) y categóricas (clasificación) con distintos porcentajes de valores nulos. Finalmente, se exponen los resultados obtenidos y se concluye sobre los métodos que ofrecen los mejores resultados de forma eficiente.

CAPÍTULO 2

Abstract

A methodology is proposed for the comparison of imputation using conventional statistical methods, machine learning methods and deep learning methods based on a measurement of performance and execution time for numerical (regression) and categorical (classification) variables with different percentages of null values. Finally, the results obtained are presented and we conclude on the methods that offer the best results in an efficient way.

CAPÍTULO 3

Introducción

El objetivo del presente trabajo consiste en realizar un estudio comparativo entre distintas técnicas de imputación para tareas de clasificación y regresión sobre un conjunto de datos utilizando métodos estadísticos, de aprendizaje automático y aprendizaje para tareas de clasificación y regresión. La propuesta surge de la necesidad de completar fuentes de información que poseen datos faltantes, las cuales pueden demorar tiempo en ser corregidas desde el origen, involucran las capacidades de los equipos a cargo, son incompatibles con modelos, o incluso, para casos en los que es imposible recuperar la información de fechas anteriores.

CAPÍTULO 4

Planteamiento del Problema

Actualmente, las empresas se encuentran en un proceso de transformación digital liderado fuertemente por la calidad y cantidad de la información. Sin embargo, en muchas ocasiones, las fuentes de datos se encuentran incompletas o con valores erróneos debido a procesos manuales durante la captura de información o inconvenientes dentro del flujo de datos. La corrección de las bases de datos desde el origen puede implicar largos tiempos de espera debido a la complejidad en la recuperación de la información y al reprocesamiento de enormes cantidades de datos. Dependiendo del manejo de los datos, incluso podría ser imposible recuperar la información de fechas anteriores. La falta de información puede ser un problema grave para aquellos proyectos que requieran de completitud de los datos para cumplir con requisitos regulatorios o que sean utilizados para modelamiento ya que son incompatibles con valores nulos o pueden ser entrenados sobre datos erróneos proporcionando resultados no esperados.

CAPÍTULO 5

Justificación

Debido a la importancia de la disponibilización inmediata de los datos y la completitud de la información, se realizará un estudio comparativo entre distintas técnicas de imputación utilizando métodos estadísticos clásicos, algoritmos de aprendizaje automático y aprendizaje profundo con el objetivo de proponer una metodología para la imputación de datos que permita completar de forma eficiente y precisa la información faltante para ser utilizada en reportería e insumos para modelos. Se busca determinar la metodología que permita realizar las tareas de clasificación y regresión sobre conjuntos de datos de distintos tamaños y naturaleza tomando en consideración el error asociado a una función de costo y tiempo de ejecución para completar la información, el cual se encuentra relacionado con el costo computación del algoritmo.

CAPÍTULO 6

Objetivos

6.1. General

Realizar un estudio comparativo para la imputación de datos mediante métodos estadísticos, de aprendizaje automático, aprendizaje profundo para tareas de clasificación y regresión.

6.2. Específicos

1. Evaluar el desempeño de distintos algoritmos estadísticos, de aprendizaje automático y aprendizaje profundo sobre conjuntos de datos con información faltante para la imputación mediante regresión y clasificación.

2. Medir el tiempo de ejecución de distintos algoritmos estadísticos, de aprendizaje automático y aprendizaje profundo sobre conjuntos de datos con información faltante para la imputación de datos.

3. Determinar el método que ofrezca los mejores resultados para tareas de clasificación y regresión a partir de métricas de desempeño y tiempo de ejecución.

CAPÍTULO 7

Antecedentes

En el año 2009, se publica el artículo “Multiple imputation for missing data in epimediological and clinical research: potential and pitfalls” (Sterne et al., 2009) en el cual se revisan las razones por las que la falta de datos puede provocar sesgos y pérdida de información en la investigación epidemiológica y clínica. Se analizan las circunstancias en las que la imputación múltiple puede ayudar a reducir el sesgo o aumentar la precisión, y se describen los posibles obstáculos en su aplicación. Se menciona al final del análisis el entusiasmo por el uso potencial de imputación múltiple y otros métodos para mejorar la validación de resultados de investigaciones médicas y reducir el desperdicio de recursos causados por fatos faltantes. El costo de los análisis de imputación múltiples es pequeño comparados con el costo de recolectar datos. Ya no se considera excusable realizar análisis con datos faltantes simplemente ignorándolos, en su lugar, se deberían implementar métodos de imputación para disponibilizar casos completos que eviten resultados engañosos o ineficaces.

El estudio “Missing data imputation using statistical and machine learning methods in a real breast cancer problem” (Jerez et al., 2010), tenía como objetivo evaluar el rendimiento de varios métodos de imputación estadísticos y de aprendizaje automático que se utilizaron para predecir la recurrencia en pacientes en un amplio conjunto de datos reales de cáncer de mama. Se aplicaron métodos de imputación basados en técnicas estadísticas como imputación media, hot-deck e imputación múltiple, y técnicas de aprendizaje automático como perceptrón multicapa (MLP), mapas de auto organización (SOM) y k-nearest neighbour (KNN), a los datos recogidos a través del proyecto ’El Álamo-I’, y luego se compararon los resultados con los obtenidos con el méto-

do de imputación de supresión por listas (LD). La base de datos incluye información demográfica, terapéutica y de recurrencia-supervivencia de 3679 mujeres con cáncer de mama invasivo operable diagnosticadas en 32 hospitales diferentes pertenecientes al Grupo Español de Investigación en Cáncer de Mama (GEICAM). Se midieron las precisiones de las predicciones sobre la recaída temprana del cáncer mediante redes neuronales artificiales (RNA), en las que se estimaron diferentes RNA utilizando los conjuntos de datos con valores perdidos imputados. Se concluyó que los métodos basados en técnicas de aprendizaje automático fueron los más adecuados para la imputación de valores perdidos y condujeron a una mejora significativa de la precisión del pronóstico en comparación con los métodos de imputación basados en procedimientos estadísticos.

Duan et al., 2014 publican "A Deep Learning Based Approach for Traffic Data Imputation" en el cual mencionan que los datos de tráfico son un componente fundamental para las aplicaciones e investigaciones en sistemas de transporte. Sin embargo, los datos de tráfico reales recogidos de detectores de bucle u otros canales a menudo incluyen datos faltantes que afectan a las aplicaciones e investigaciones relativas. En su trabajo proponen un enfoque basado en el aprendizaje profundo para imputar los datos de tráfico que faltan. El enfoque propuesto trata los datos de tráfico, incluidos los datos observados y los datos que faltan, como un elemento de datos completo y restaura los datos completos con la red estructural profunda. El enfoque de aprendizaje profundo puede descubrir las correlaciones contenidas en la estructura de datos mediante un preentrenamiento por capas y mejorar la precisión de la imputación mediante un ajuste fino posterior. Se analizan los patrones de imputación que pueden realizarse con el enfoque propuesto y se lleva a cabo una serie de experimentos. Los resultados muestran que el enfoque propuesto puede mantener un error estable bajo diferentes tasas de ausencia de datos de tráfico. El aprendizaje profundo es prometedor en el campo de la imputación de datos de tráfico.

El mismo año, se realiza la publicación de "Single imputation with multilayer perceptrón and multiple imputation combining multilayer perceptrón and k-nearest neighbours for monotone patterns" por Silva-Ramírez et al., 2015. En este artículo describen que el proceso de descubrimiento de conocimientos se apoya en la información de archivos de datos recopilada a partir de conjuntos de datos recogidos, que a menudo contienen errores en forma de valores omitidos. La imputación de datos es la actividad destinada a estimar los valores de los elementos de datos que faltan. Este estudio se centra en el desarrollo de modelos automatizados de imputación de datos, basados en redes neuronales artificiales para patrones monótonos de valores perdidos. En su trabajo proponen un enfoque de imputación simple basado en un perceptrón multicapa cuyo entrenamiento se realiza con diferentes reglas de aprendizaje, y un enfoque de imputación múltiple basado en la combinación de perceptrón multicapa y k-nearest neighbours. Dieciocho bases de datos reales y simuladas se sometieron a un experimento de perturbación con generación aleatoria de patrones

monótonos de datos perdidos. Se realizó una prueba empírica con estos conjuntos de datos, incluyendo ambos enfoques (imputaciones simples y múltiples), y también se consideraron tres procedimientos clásicos de imputación simple: imputación media/modo, regresión y hot-deck. Por lo tanto, los experimentos incluyeron cinco métodos de imputación. El diseño de experimentos consideró un conjunto de datos fijo, T, con patrones monótonos de faltantes. Cada conjunto correcto, T, fue separado en un conjunto de entrenamiento ($70\,\%$) y un conjunto de prueba ($30\,\%$), obteniendo los archivos T_1 y T_2. Un $30\,\%$ de las variables, $p - f$, fueron elegidas aleatoriamente para obtener X_M y una versión perturbada T_{m2} a partir de T_2 colocando como faltantes todos los valores de las variables de X_m. Los valores no incluidos en X_m fueron representados por X_0. De esta manera, el modelo de imputación para predecir X_m a partir de X_0 se podría entregar sobre T_1. Las filas de T_{m2} se introducen al modelo entrenado para producir T^{*2}, los cuales se comparan finalmente con los registros reales T_2. Se repitió este método de división 50 veces para evitar que los resultados dependieran solamente de un único proceso de imputación individual. Los resultados, considerando diferentes medidas de rendimiento, demostraron que, en comparación con las herramientas tradicionales, ambas propuestas mejoran el nivel de automatización y la calidad de los datos ofreciendo un rendimiento satisfactorio. Recomiendan que los trabajos futuros podrían incluir otros modelos RNA u otras herramientas de aprendizaje automático, distintos mecanismos de datos omitidos y nuevos criterios de evaluación. Otro tema de investigación futuro sería el tratamiento de los valores perdidos en conjuntos de datos que varían en el tiempo, por ejemplo, encuestas de panel o series temporales econométricas.

Luego, en el año 2019, aparece el trabajo “Comparison of Performance of Data Imputation Methods for Numeric Dataset” (Jadhav et al., 2019) en el cual comparten que la falta de datos es un problema común al que se enfrentan los investigadores y los científicos de datos. Por lo tanto, es necesario tratarlos adecuadamente para obtener resultados mejores y más precisos del análisis de datos. El objetivo del trabajo de investigación fue proporcionar una mejor comprensión del mecanismo de ausencia de datos, los métodos de imputación de datos y evaluar el rendimiento de los métodos de imputación de datos ampliamente utilizados para conjuntos de datos numéricos. Ayudará a los profesionales y científicos de datos a seleccionar el método apropiado de imputación de datos para conjuntos de datos numéricos al realizar tareas de minería de datos. En este trabajo se comparan de forma exhaustiva siete métodos de imputación de datos: imputación de la media, imputación de la mediana, imputación kNN, ajuste predictivo de la media, regresión lineal bayesiana (norm), regresión lineal, no bayesiana (norm.nob) y muestra aleatoria. Se utilizan cinco conjuntos de datos numéricos diferentes obtenidos del repositorio de aprendizaje automático de la UCI para analizar y comparar el rendimiento de los métodos de imputación de datos. El rendimiento de los métodos de imputación de datos se evalúa mediante el método del error cuadrático

medio normalizado (RMSE). En la metodología de investigación comparte que se inyectaron varios porcentajes de valores faltantes (10 %, 20 %, 30 %, 40 % y 50 %), los cuales luego fueron imputados utilizando los métodos propuestos. Los resultados del análisis muestran que el método de imputación kNN supera a los demás métodos. También se ha comprobado que el rendimiento del método de imputación de datos es independiente del conjunto de datos y del porcentaje de valores perdidos en el conjunto de datos.

En el año 2021 se realiza la publicación de "Investigating the impacto f missing data imputation techniques on battery energy management system" (Pazhoohesh et al., 2021) en el cual se dedican al estudio de imputación de datos a partir de sistemas de almacenamiento de energía. La falta de datos es un problema real en cualquier sistema de operaciones, y parece ser más común en los sistemas de potencias debido a problemas de mal funcionamiento de los sensores y/o de la red. Las técnicas de imputación de datos ausentes han evolucionado en la investigación de los sistemas eléctricos utilizando datos de contadores inteligentes, pero se ha investigado poco sobre la mejor manera de gestionar los datos ausentes en los sistemas de gestión de almacenamiento. Este artículo se basa en un año de datos de carga y descarga recogidos de una batería real de iones de litio desplegada en la red de distribución de Leighton Buzzard, Reino Unido. Utilizando el software de código abierto R Studio, se aplicaron ocho técnicas de imputación seleccionadas para identificar la técnica más adecuada para reemplazar diversas cantidades y patrones de datos perdidos. Los resultados del estudio abren vías de discusión y debate para identificar una técnica de imputación adecuada en el contexto de la gestión del almacenamiento. El estudio también es pionero en la comprensión de la importancia de la descomposición a la hora de evaluar la técnica de imputación adecuada.

Finalmente, en el año 2022, aparece el artículo "A Review of Missing Data Handling Techinques for Marhine Learning" (Oluwaseye et al., 2022) se menciona que los datos del mundo real suelen contener valores omitidos, lo que afecta negativamente al rendimiento de la mayoría de los algoritmos de aprendizaje automático cuando se emplean en estos conjuntos de datos. Dado que la precisión y la eficacia de los modelos de aprendizaje automático dependen de la calidad de los datos utilizados, es necesario que los analistas de datos y los investigadores que trabajan con datos busquen algunas técnicas pertinentes que puedan utilizarse para gestionar estos valores perdidos ineludibles. Este artículo revisa algunas de las prácticas más avanzadas obtenidas en la literatura para tratar los problemas de datos perdidos en el aprendizaje automático. Enumera algunas métricas de evaluación utilizadas para medir el rendimiento de estas técnicas. Este estudio intenta poner estas técnicas y métricas de evaluación en términos claros, seguidos de algunas ecuaciones matemáticas. Además, se ofrecen algunas recomendaciones para tener en cuenta a la hora de tratar las técnicas de tratamiento de datos ausentes.

CAPÍTULO 8

Marco Teórico

8.1. Mecanismos de datos faltantes

8.1.1. Faltantes completamente aleatorios (MCAR)

Si la falta de datos no depende de los valores de los datos, faltantes u observadores, entonces, se dice que los datos faltantes son completamente aleatorios. Si la probabilidad de que falte es la misma para todos los casos, se dice que los datos faltan completamente al azar (MCAR). Esto implica efectivamente que las causas de los datos que faltan no están relacionadas con los datos. En consecuencia, podemos ignorar muchas de las complejidades que surgen porque faltan datos, aparte de la evidente pérdida de información.

8.1.2. Faltantes aleatorios (MAR)

Se dice que son los datos son faltantes aleatorios si el faltante depende únicamente de las componentes observadas. Si la probabilidad de que falte es la misma sólo dentro de los grupos definidos por los datos observados, entonces los datos faltan al azar (MAR). MAR es una clase mucho más amplia que MCAR. MAR es más general y realista que MCAR. Los métodos modernos de datos ausentes suelen partir de la hipótesis MAR.

8.1.3. Faltantes no aleatorios (MNAR)

El mecanismo de faltantes se conoce como Faltantes no aleatorios cuando si los valores de los datos dependen de los componentes faltantes. Si no se cumplen ni el MCAR ni el MAR, se habla de falta no aleatoria (MNAR). En la literatura también se puede encontrar el término NMAR (not missing at random) para el mismo concepto. MNAR significa que la probabilidad de faltar varía por razones que desconocemos (van Buuren et al., 2021).

8.2. Métodos Estadísticos de Imputación

8.2.1. Imputación individual

Se refiere a la imputación de un valor plausible por cada valor que falta de una variable concreta en el conjunto de datos y, a continuación, realizar el análisis como si todos los datos se hubieran observado originalmente.

8.2.2. Imputación múltiple

En los métodos de imputación única se asume que el valor de imputación única es el correcto y se exagera la precisión. Sin embargo, nunca puede haber certeza absoluta sobre la validez de los valores imputados. Por lo tanto, la incertidumbre en torno a estos valores imputados debe incorporarse en los métodos de datos. Se desarrolló un método para promediar el resultado a través de múltiples conjuntos de datos imputados. Así, en la imputación múltiple, en lugar de sustituir un único valor por cada observación que falta, sustituye múltiples valores plausibles para reflejar la incertidumbre sobre los valores correctos a imputar. Así, el método de imputación múltiple genera "mçonjuntos de datos completos diferentes con valores observados e imputados. Todo método de imputación múltiple sigue tres pasos: (1) Imputación: De forma similar a la imputación simple, se imputan los valores que faltan; sin embargo, los valores imputados se generan "m"veces en lugar de una sola vez. Por tanto, podría haber "mçonjuntos de datos completos diferentes tras la imputación. (2) Análisis de cada conjunto de datos: Tras la imputación y la generación de "mçonjuntos de datos diferentes, se analiza cada uno de los "mçonjuntos de datos. (3) Agrupación: Por último, se consolidan los resultados obtenidos de cada conjunto de datos analizado.

8.2.3. Imputación por media o mediana

Una solución rápida para los datos que faltan es sustituirlos por la media o mediana. La imputación de la media o mediana es una solución rápida y sencilla para los datos que faltan. Sin embargo, subestimará la varianza, perturbará las relaciones entre variables, sesgará casi cualquier estimación que no sea la media y sesgará la estimación de la media cuando los datos no sean MCAR. La imputación de la media o mediana quizá sólo debería utilizarse como solución rápida cuando faltan unos pocos valores, y debería evitarse en general.

8.2.4. Imputación de valor con mayor frecuencia

La sustitución de los valores perdidos por la categoría más frecuente es el equivalente de la imputación media/mediana/moda. Consiste en sustituir todas las ocurrencias de valores perdidos dentro de una variable por la etiqueta o categoría más frecuente de la variable.

8.2.5. Imputación de cubierta caliente (Hot deck)

Que imputa valores individuales extraídos de unidades de respuesta "similares". La imputación en caliente es común en la práctica de las encuestas y puede implicar esquemas muy elaborados para seleccionar unidades similares para la imputación.

8.2.6. Imputación de cubierta fría (Cold deck)

Sustituye el valor que falta de una variable por un valor constante procedente de una fuente externa, como un valor de una encuesta anterior. Como en el caso de la sustitución, la práctica actual suele tratar los datos resultantes como una muestra completa, es decir, ignora las consecuencias de la imputación. Una teoría satisfactoria para el análisis de los datos obtenidos por imputación en frío es obvia o inexistente.

8.2.7. Imputación por regresión

La imputación por regresión incorpora el conocimiento de otras variables con la idea de producir imputaciones más inteligentes. El primer paso consiste en construir un modelo a partir de los datos observados. A continuación, se calculan las predicciones para los casos incompletos según el modelo ajustado, y sirven como sustitutos de los datos que faltan. La imputación de regresión,

así como sus modernas encarnaciones en el aprendizaje automático, es probablemente el más peligroso de todos los métodos descritos aquí. Se nos puede hacer creer que vamos a hacer un buen trabajo preservando las relaciones entre las variables. En realidad, sin embargo, la imputación por regresión refuerza artificialmente las relaciones en los datos. Las correlaciones se sesgan al alza. La variabilidad se subestima. Las imputaciones son demasiado buenas para ser ciertas. La imputación por regresión es una receta para falsos positivos y relaciones falsas.

8.2.8. Imputación por regresión estocástica

La imputación por regresión estocástica es un refinamiento de la imputación por regresión que intenta abordar el sesgo de correlación añadiendo ruido a las predicciones. El siguiente código imputa Ozono a partir de Solar.R mediante imputación por regresión estocástica. No obstante, la imputación por regresión estocástica representa un gran avance conceptual. A algunos analistas les puede parecer contraintuitivo .estropear"la mejor predicción añadiendo ruido aleatorio, pero esto es precisamente lo que la hace adecuada para la imputación. Una imputación de regresión estocástica bien ejecutada conserva no sólo los pesos de regresión, sino también la correlación entre variables. La idea principal de extraer de los residuos es muy poderosa, y constituye la base de técnicas de imputación más avanzadas (Little y Rubin, 2019).

8.3. Algoritmos de Regresión

8.3.1. Regresión lineal simple

Es un método muy sencillo para predecir una respuesta cuantitativa Y a partir de una única variable predictiva X. Supone que existe aproximadamente una relación lineal entre X e Y.

8.3.2. Regresión lineal múltiple

La regresión lineal simple es un método útil para predecir una respuesta a partir de una única variable de predicción. Sin embargo, en la práctica a menudo tenemos más de un predictor.

En lugar de ajustar un modelo de regresión lineal simple separado para cada predictor, un enfoque mejor es ampliar el modelo de regresión lineal simple para que pueda acomodar directamente múltiples predictores. Podemos hacerlo dando a cada predictor un coeficiente de pendiente separado en un único modelo. En general, supongamos que tenemos p predictores distintos.

8.3.3. Regresión de Vectores de Soporte

Una máquina de vectores soporte (SVM) es un modelo de aprendizaje automático potente y versátil, capaz de realizar clasificación lineal o no lineal, regresión e incluso detección de novedades. Las SVM brillan con conjuntos de datos no lineales de tamaño pequeño o medio (es decir, de cientos a miles de instancias), especialmente para tareas de clasificación. Sin embargo, no se adaptan muy bien a conjuntos de datos muy grandes, como se verá más adelante.

8.3.4. Regresión con Árboles de Decisión

Los árboles de decisión pueden aplicarse tanto a problemas de regresión como de clasificación. Un árbol de clasificación es muy similar a un árbol de regresión, salvo que se utiliza para predecir una respuesta cualitativa en lugar de una cuantitativa. En un árbol de regresión, la respuesta prevista para una observación viene dada por la respuesta media de las observaciones de entrenamiento que pertenecen al mismo nodo terminal. En cambio, en un árbol de clasificación, predecimos que cada observación pertenece a la clase de observaciones de entrenamiento más frecuente en la región a la que pertenece. Al interpretar los resultados de un árbol de clasificación, a menudo nos interesa no sólo la predicción de clase correspondiente a una región de nodo terminal concreta, sino también las proporciones de clase entre las observaciones de entrenamiento que caen en esa región.

8.3.5. Regresión con Bosques Aleatorios

Los bosques aleatorios suponen una mejora con respecto a los árboles ensacados gracias a un pequeño ajuste que decorrelaciona los árboles. Al igual que en el ensacado, se construye una serie de árboles de decisión sobre muestras de entrenamiento. Pero al construir estos árboles de decisión, cada vez que se considera una división en un árbol, se elige una muestra aleatoria de m predictores como candidatos a la división del conjunto completo de p predictores. La división sólo puede utilizar uno de esos m predictores. En cada división se toma una nueva muestra de m predictores (James et al., 2021).

8.4. Algoritmos de Aprendizaje Profundo

8.4.1. Redes Neuronales Artificiales

Una red neuronal toma un vector de entrada de p variables $X = (X1, X2, ..., Xp)$ y construye una función no lineal $f(X)$ para predecir la respuesta Y. Las redes neuronales modernas suelen tener más de una capa oculta y, a menudo, muchas unidades por capa. En teoría, una sola capa oculta con un gran número de unidades tiene capacidad para aproximarse a la mayoría de las funciones. Sin embargo, la tarea de aprendizaje de descubrir una buena solución es mucho más fácil con múltiples capas de tamaño modesto.

8.5. Mediciones de desempeño

8.5.1. Error Cuadrático Medio

El error cuadrático medio ($RMSE$) es la desviación estándar de los residuos (errores de predicción). Los residuos son una medida de la distancia a la que se encuentran los puntos de datos de la línea de regresión. Es una medida de la dispersión de estos residuos. En otras palabras, indica lo concentrados que están los datos en torno a la recta de mejor ajuste.

8.5.2. Precisión

La precisión de un clasificador se refiere a la cantidad de predicciones acertadas sobre un conjunto de datos. Está dada por la siguiente ecuación $precision = TP/(TP + FP)$. Donde TP es la cantidad de verdaderos positivos y FP es la cantidad de falsos positivos (Géron, 2022).

CAPÍTULO 9

Metodología

9.1. Regresión

9.1.1. Repositorio

El desarrollo metodológico descrito a continuación se encuentra disponible en el siguiente cuaderno de Jupyter Notebook desde Google Colab.

UNIR - Proyecto Final de Maestría - Regresión

9.1.2. Conjunto de datos

El conjunto de datos utilizado para la metodología fue el California Housing dataset disponible en el repositorio de StatLib. La variable objetivo es el valor medio de la vivienda en los distritos de California, expresado en cientos de miles de dólares ($100.000\$$).

Este conjunto de datos se obtuvo a partir del censo de EE. UU. de 1990, utilizando una fila por grupo de bloques censales. Un grupo de bloques es la unidad geográfica más pequeña para la que la Oficina del Censo de EE. UU. publica datos de muestra (un grupo de bloques suele tener una población de 600 a 3000 personas).

Un hogar es un grupo de personas que residen en una vivienda. Dado que el número medio de habitaciones y dormitorios en este conjunto de datos se proporciona por hogar, estas colum-

nas pueden tomar valores excesivamente grandes para grupos de manzanas con pocos hogares y muchas casas vacías, como los complejos vacacionales.

El conjunto de datos está conformado por 20,640 registros con 8 atributos numéricos:

- MedInc renta media del grupo de bloques
- HouseAge edad media de la vivienda en el grupo de bloques
- AveRooms número medio de habitaciones por hogar
- AveBedrms número medio de dormitorios por hogar
- Population población del grupo de bloques
- AveOccup número medio de miembros del hogar
- Latitud grupo de bloques latitude
- Longitude grupo de bloques longitude
- Pronóstico de ventas de datos en casa: Boston 80 variables aprox.

Se importó directamente como un DataFrame de Pandas mediante la siguiente línea.

```
from sklearn.datasets import fetch_california_housing
data = fetch_california_housing(as_frame=True).frame
```

Se ejecutó el siguiente código para obtener una descripción estadística de los datos (Cuadro 9.1.)

```
data.describe()
```

9.1.3. Preparación de los datos

El conjunto de datos se divide en una matriz de variables independientes, X, y un vector de salida con la variable dependiente, y. En este caso, se realizará la imputación sobre la columna MedHouseVal.

```
X = data[['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', '
    AveOccup', 'Latitude', 'Longitude']]
y = data['MedHouseVal']
```

Luego, utilizando el *train_test_split* de *scikit-learn*, se divide el conjunto de datos ($100\,\%$) en un conjunto de entrenamiento (*X_train*, *y_train*) y un conjunto de datos de prueba (*X_test*, *y_test*). Se define un estado aleatorio fijo de 1 para la reproducibilidad de los resultados. El tamaño del conjunto de prueba se utilizará como parámetro para definir el porcentaje de valores que se imputarán sobre la variable objetivo. Por lo que el objetivo de imputación de valores se realizará sobre *y_test*. Se definió un iterador con los porcentajes de nulidad para 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40 y 0.45 mediante el siguiente código.

```
per_null_values = map(lambda x: x/100, range(5, 50, 5))
```

La iteración sobre sobre los distintos porcentajes de nulidad se realizó de la siguiente manera.

```
for per_null in per_null_values:
    times = []
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=
   per_null, random_state=random_seed)
```

9.1.4. Visualización de valores nulos

En cada iteración, se recopilaron los índices de los elementos pertenecientes a *y_test* para identificarlos dentro del conjunto de datos original. Utilizando el módulo *missingno* del usuario *ResidentMario* disponible en el repositorio de GitHub se generó la Figura 9.1 en la cual se pueden visualizar los nulos introducidos dentro de la variable objetivo para cada porcentaje de nulidad.

```
missing_data = data.copy()
missing_data.loc[y_test.index, var_name] = np.nan
df_null[f'{str(per_null)}'] = missing_data[var_name]
```

9.1.5. Imputación mediante métodos estadísticos

En la primera sección, se realiza la imputación de los valores *y_test* utilizando métodos estadísticos mediante el uso de las funciones de *Numpy* y *Pandas*. La imputación a partir del promedio se realizó aplicando la función mean sobre el conjunto de entrenamiento, *y_train*, para asignar los valores del conjunto de prueba, *y_pred_mean*, que corresponde a la predicción del conjunto de prueba real, *y_test*. Obteniendo un valor promedio de 2.06924 para los 6192 registros.

```
mean = y_train.mean()
y_pred_mean = pd.Series(data=, index=y_test.index, name=var_name)
```

La imputación a partir de la mediana se realizó de forma similar, pero utilizando la función median, obteniendo *y_pred_median*, con un valor de 1.793 para todos los registros.

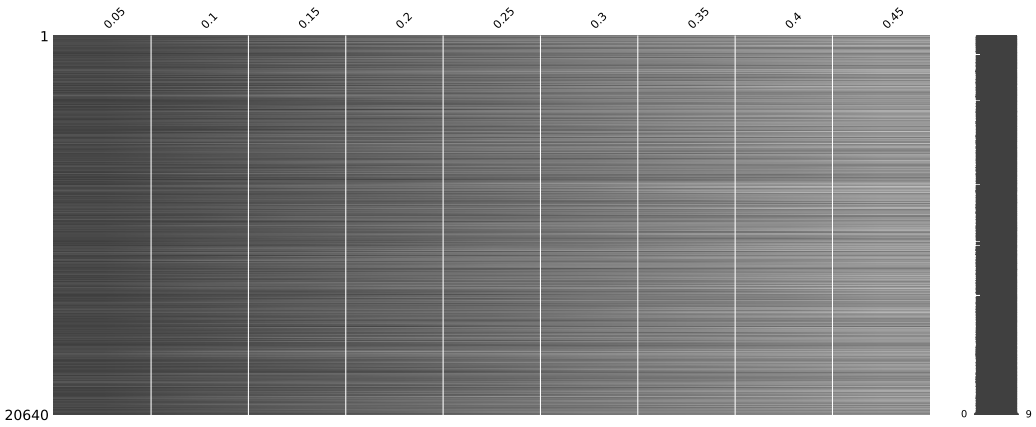

Figura 9.1: Nulos introducidos en MedHouseVal del California Housing

```
median = y_train.median()
y_pred_median = pd.Series(data=median, index=y_test.index, name=var_name)
```

Por último, se realizó la imputación mediante el valor con mayor frecuencia con *value_counts*.

```
max_freq = y_train.value_counts().idxmax()
y_pred_max_freq = pd.Series(data=max_freq, index=y_test.index, name=
    var_name)
```

9.1.6. Imputación mediante métodos de aprendizaje automático

En la segunda sección, se realiza la imputación de datos mediante métodos de aprendizaje automático (*Machine Learning*) utilizando la librería de *scikit-learn*. A diferencia de los métodos estadísticos que solamente utilizan los valores de la columna sobre la cual se imputarán los valores, en los métodos a continuación se utilizan los demás atributos del conjunto de datos para completar los datos faltantes. Se inicia con una regresión lineal múltiple utilizando el modelo **LinearRegression**.

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train, y_train)
y_pred_linreg = lm.predict(X_test)
y_pred_linreg = pd.Series(data=y_pred_linreg, index=y_test.index, name=
    var_name)
```

A continuación, se realiza imputación mediante una máquina de vector de soporte con el modelo **SVR**.

```
from sklearn.svm import SVR
svr = SVR()
svr.fit(X_train, y_train)
y_pred_supvec = svr.predict(X_test)
y_pred_supvec = pd.Series(data=y_pred_supvec, index=y_test.index, name=
    var_name)
```

Luego, se emplea un árbol de decisión de regresión con el modelo **DecisionTreeRegressor**.

```
from sklearn import tree
dtr = tree.DecisionTreeRegressor(random_state=random_seed)
dtr.fit(X_train, y_train)
y_pred_trereg = dtr.predict(X_test)
y_pred_trereg = pd.Series(data=y_pred_trereg, index=y_test.index, name=
    var_name)
```

Por último, se implementó un árbol aleatorio de regresión con el modelo **RandomForestRegressor**.

```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(random_state=random_seed)
rfr.fit(X_train, y_train)
y_pred_ranfor = rfr.predict(X_test)
y_pred_ranfor = pd.Series(data=y_pred_ranfor, index=y_test.index, name=
    var_name)
```

9.1.7. Imputación mediante métodos de aprendizaje profundo

En la tercera sección, se realiza imputación utilizando una Red Neuronal Artificial utilizando el modelo **MLPRegressor** con los parámetros por defecto de 100 neuronas con una capa escondida, función de activación ReLU, solucionador Adam, tasa de aprendizaje constante de 0.001 y 200 iteraciones.

```
from sklearn.neural_network import MLPRegressor
ann = MLPRegressor(random_state=random_seed)
ann.fit(X_train, y_train)
y_pred_artneu = ann.predict(X_test)
y_pred_artneu = pd.Series(data=y_pred_artneu, index=y_test.index, name=
    var_name)
```

9.1.8. Comparación de resultados

Finalmente, se realiza una comparativa en los resultados de las imputaciones entre los distintos métodos a partir de los valores reales, *y_test*, y las predicciones, *y_pred*. La medición del desempeño se realizó utilizando el error cuadrático medio (RMSE) y la duración midió el tiempo en segundos de la imputación (sin considerar el entrenamiento).

```
from sklearn.metrics import mean_squared_error
methods = ['Media', 'Mediana', 'Maxima Frecuencia',
           'Regresion lineal multiple', 'Maquina de Vector de Soporte', '
    Arbol de Decision',
           'Bosque Aleatorio', 'Red Neuronal Artificial']
y_pred = [y_pred_mean, y_pred_median, y_pred_max_freq,
           y_pred_linreg, y_pred_supvec, y_pred_dectre,
           y_pred_ranfor, y_pred_artneu]

for index, method in enumerate(methods):
    results.append({
        'Porcentaje de Nulos' : per_null,
        'Metodo': method,
        'RMSE': round(mean_squared_error(y_test, y_pred[index]), 2),
        'Duracion' : times[index]})

results = pd.DataFrame(results)
```

Se empleó el siguiente código para crear el Cuadro 9.2 y la Figura 9.2 en la cual se pueden visualizar los resultados del RMSE obtenidos por cada método de imputación para distintos porcentajes de nulidad.

```
regression_results = results.pivot(index='Porcentaje de Nulos', columns='
    Metodo', values='RMSE')
regression_results.plot(grid=True, legend=True, figsize=(12,7))
```

Luego, se empleó el siguiente código para crear el Cuadro 9.11 y la Figura 9.9 en la cual se pueden visualizar los resultados de los tiempos de imputación para cada método a distintos porcentajes de nulidad.

```
regression_times = results.pivot(index='Porcentaje de Nulos', columns='
    Metodo', values='Duracion')
regression_times.plot(grid=True, legend=True, figsize=(12,7))
```

Finalmente, se creó el siguiente código para agrupar los resultados promedios obtenidos por cada método de forma ascendente (Cuadro 9.4).

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
count	20640.00	20640.00	20640.00	20640.00	20640.00	20640.00	20640.00	20640.00	20640.00
mean	3.87	28.64	5.43	1.10	1425.48	3.07	35.63	-119.57	2.07
std	1.90	12.59	2.47	0.47	1132.46	10.39	2.14	2.00	1.15
min	0.50	1.00	0.85	0.33	3.00	0.69	32.54	-124.35	0.15
25 %	2.56	18.00	4.44	1.01	787.00	2.43	33.93	-121.80	1.20
50 %	3.53	29.00	5.23	1.05	1166.00	2.82	34.26	-118.49	1.80
75 %	4.74	37.00	6.05	1.10	1725.00	3.28	37.71	-118.01	2.65
max	15.00	52.00	141.91	34.07	35682.00	1243.33	41.95	-114.31	5.00

Cuadro 9.1: Estadística descriptiva del California Housing Data Set

Porcentaje de Nulos	Arbol de Decision	Bosque Aleatorio	Maquina de Vector de Soporte	Maxima Frecuencia	Media	Mediana	Red Neuronal Artificial	Regresion lineal multiple
0.05	0.51	0.27	1.30	10.18	1.30	1.34	0.56	0.55
0.10	0.50	0.25	1.35	10.12	1.35	1.40	1.12	0.54
0.15	0.50	0.25	1.35	10.08	1.34	1.39	0.62	0.54
0.20	0.50	0.25	1.32	10.11	1.31	1.36	0.59	0.53
0.25	0.53	0.26	1.33	10.07	1.32	1.37	0.62	0.54
0.30	0.49	0.26	1.34	10.02	1.32	1.38	0.66	0.53
0.35	0.52	0.26	1.34	9.96	1.31	1.37	0.53	0.52
0.40	0.53	0.27	1.36	9.95	1.33	1.40	0.67	0.52
0.45	0.51	0.26	1.37	9.98	1.33	1.40	0.65	0.52

Cuadro 9.2: Error para métodos de imputación sobre MedHouseVal

Porcentaje de Nulos	Arbol de Decision	Bosque Aleatorio	Maquina de Vector de Soporte	Maxima Frecuencia	Media	Mediana	Red Neuronal Artificial	Regresion lineal multiple
0.05	0.002676	0.076061	0.896299	0.000184	0.000745	0.000421	0.003703	0.005672
0.10	0.005860	0.116073	3.041378	0.000171	0.000209	0.000178	0.007054	0.004034
0.15	0.003337	0.147687	3.289611	0.000128	0.000179	0.000164	0.004973	0.008051
0.20	0.004001	0.178817	4.695273	0.000120	0.000172	0.000160	0.006646	0.003296
0.25	0.004092	0.249394	5.172662	0.000145	0.000194	0.000161	0.006826	0.003315
0.30	0.004326	0.221766	4.127768	0.000131	0.000177	0.000181	0.011904	0.003485
0.35	0.004604	0.267163	6.116815	0.000138	0.000319	0.000170	0.008276	0.004052
0.40	0.004772	0.304799	4.631769	0.000145	0.000202	0.000164	0.009102	0.002579
0.45	0.005584	0.289498	4.779438	0.000123	0.000192	0.000160	0.009458	0.006891

Cuadro 9.3: Duración para métodos de imputación sobre MedHouseVal

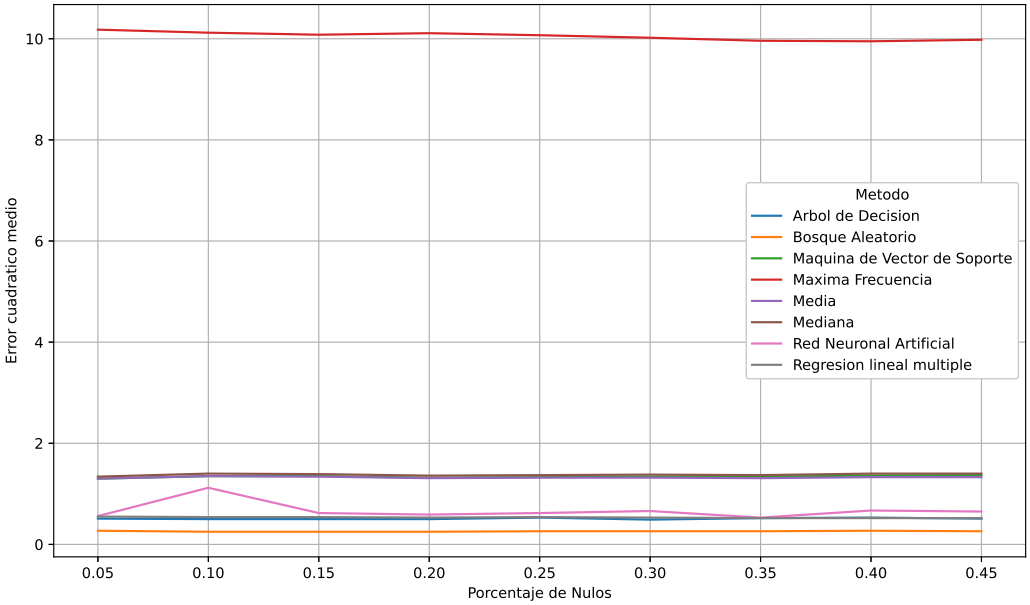

Figura 9.2: Error para métodos de imputación sobre MedHouseVal

```
results.groupby(by='Metodo').mean().sort_values(by='RMSE', ascending=True)
    [['RMSE', 'Duracion']]
```

9.2. Clasificación

9.2.1. Repositorio

El desarrollo metodológico descrito a continuación se encuentra disponible en el siguiente cuaderno de Jupyter Notebook desde Google Colab.

UNIR - Proyecto Final de Maestría - Clasificación

9.2.2. Conjunto de datos

El conjunto de datos utilizado para la metodología fue el Iris dataset disponible en el repositorio de UCI. La variable objetivo es el tipo de planta Iris. Es una de las bases de datos más conocidas sobre reconocimiento de patrones. El conjunto de datos contiene 3 clases de 50 instancias cada

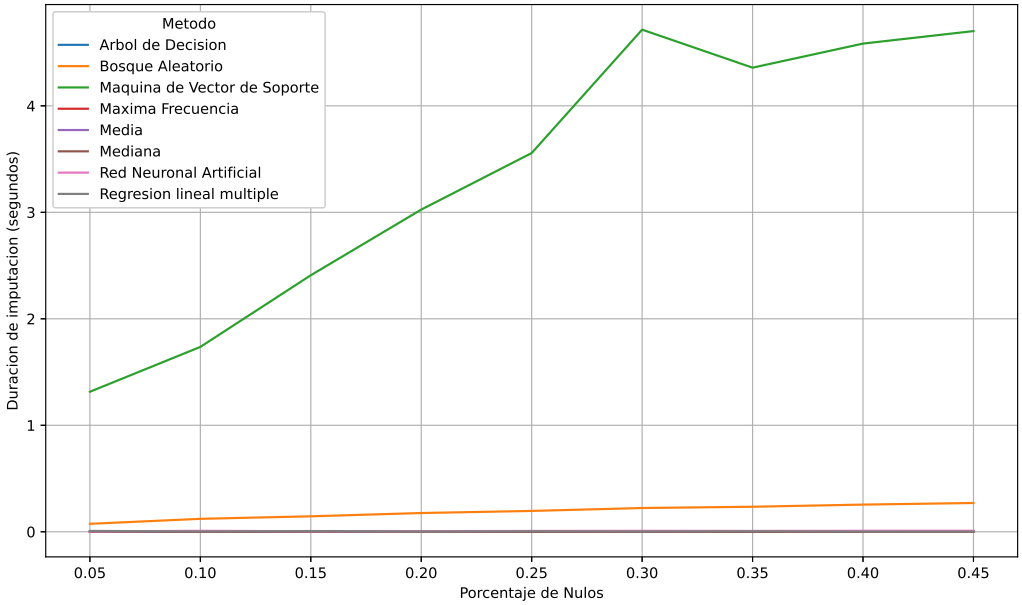

Figura 9.3: Duración para métodos de imputación sobre MedHouseVal

una, donde cada clase se refiere a un tipo de planta de iris. Una clase es linealmente separable de las otras 2; estas últimas no son linealmente separables entre sí. El conjunto de datos está conformado por 150 registros con 3 atributos numéricos y 1 categórico:

- Largo del sépalo en centímetros.
- Ancho del sépalo en centímetros.
- Largo del pétalo en centímetros.
- Ancho del pétalo en cenímetros.
- Clase de planta: Iris Setosa, Iris Versicolour, Iris Virginica.

Se importó directamente como un DataFrame de Pandas mediante la siguiente línea.

```
from sklearn.datasets import load_iris
data = load_iris(as_frame=True).frame
```

Se ejecutó el siguiente código para obtener una descripción estadística de los datos (Cuatro 9.5).

```
data.describe()
```

Metodo	RMSE	Duracion
Bosque Aleatorio	0.258889	0.205695
Arbol de Decision	0.510000	0.004361
Regresion lineal multiple	0.532222	0.004597
Red Neuronal Artificial	0.668889	0.007549
Media	1.323333	0.000265
Maquina de Vector de Soporte	1.340000	4.083446
Mediana	1.378889	0.000196
Maxima Frecuencia	10.052222	0.000143

Cuadro 9.4: Resultados promedio para métodos de imputación sobre MedHouseVal

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
count	150.00	150.00	150.00	150.00	150.00
mean	5.84	3.06	3.76	1.20	1.00
std	0.83	0.44	1.77	0.76	0.82
min	4.30	2.00	1.00	0.10	0.00
25 %	5.10	2.80	1.60	0.30	0.00
50 %	5.80	3.00	4.35	1.30	1.00
75 %	6.40	3.30	5.10	1.80	2.00
max	7.90	4.40	6.90	2.50	2.00

Cuadro 9.5: Estadística descriptiva del Iris Data Set

9.2.3. Preprocesamiento de datos

Los atributos del conjunto de datos son numéricos, por lo que no fue necesario realizar ninguna codificación de datos categóricos. Sin embargo, en caso de ser necesario, se sugiere implementar LabelEncoder para campos ordinales o binarios y OneHotEncoder para nominales.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data[:, -1] = le.fit_transform(data[:, -1])
 from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])],
    remainder='passthrough')
data = np.array(ct.fit_transform(data))
```

9.2.4. Preparación de los datos

El conjunto de datos se divide en una matriz de variables independientes, X, y un vector de salida con la variable dependiente, y. En este caso, se realizará la imputación sobre la columna target.

```
X = data[['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', '
    petal width (cm)']]
y = data['target']
```

Al igual que en la sección de Regresión, se dividió el conjunto de datos utilizando el *train_test_split* de *scikit-learn*. Se definió el mismo iterador con los porcentajes de nulidad para 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40 y 0.45.

9.2.5. Visualización de valores nulos

Se recopilaron los índices de los elementos pertenecientes a *y_test* para identificarlos dentro del conjunto de datos original. Se generó la Figura 9.4 en la cual se pueden visualizar los nulos introducidos dentro de la variable objetivo para cada porcentaje de nulidad.

9.2.6. Imputación mediante métodos estadísticos

En la primera sección, se realiza la imputación de los valores *y_test* utilizando métodos estadísticos mediante el uso de las funciones de *Numpy* y *Pandas*. La imputación a partir del promedio se realizó aplicando la función mean sobre el conjunto de entrenamiento, *y_train*, para asignar los

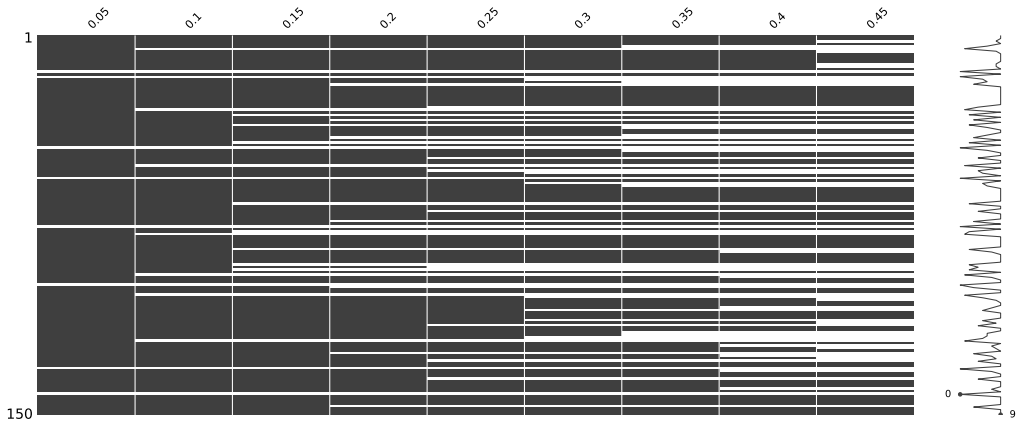

Figura 9.4: Nulos introducidos en target del Iris dataset

valores del conjunto de prueba, *y_pred_mean*, que corresponde a la predicción del conjunto de prueba real, *y_test*. Obteniendo un valor promedio de 1 para los 45 registros.

```
mean = y_train.mean()
y_pred_mean = pd.Series(data=mean, index=y_test.index, name=var_name,
    dtype=int)
```

La imputación a partir de la mediana se realizó de forma similar, pero utilizando la función median, obteniendo *y_pred_median*, con un valor de 1 para todos los registros.

```
median = y_train.median()
y_pred_median = pd.Series(data=median, index=y_test.index, name=var_name,
    dtype=int)
```

Por último, se realizó la imputación mediante el valor con mayor frecuencia con *value_counts*.

```
max_freq = y_train.value_counts().idxmax()
y_pred_max_freq = pd.Series(data=max_freq, index=y_test.index, name=
    var_name)
```

9.2.7. Imputación mediante métodos de aprendizaje automático

En la segunda sección, se realiza la imputación de datos mediante métodos de aprendizaje automático utilizando la librería de *scikit-learn*. A diferencia de los métodos estadísticos que solamente utilizan los valores de la columna sobre la cual se imputarán los valores, en los métodos a continuación se utilizan los demás atributos del conjunto de datos para completar los datos faltantes. Se inicia con una regresión logística utilizando el modelo **LogisticRegression**.

```
from sklearn.linear_model import LogisticRegression
lm = LogisticRegression()
lm.fit(X_train, y_train)
y_pred_logreg = lm.predict(X_test)
y_pred_logreg = pd.Series(data=y_pred_logreg, index=y_test.index, name=
    var_name)
```

A continuación, se realiza imputación mediante una máquina de vector de soporte con el modelo **SVC**.

```
from sklearn.svm import SVC
svc = SVC()
svc.fit(X_train, y_train)
y_pred_supvec = svc.predict(X_test)
y_pred_supvec = pd.Series(data=y_pred_supvec, index=y_test.index, name=
    var_name)
```

Luego, se emplea un árbol de decisión de clasificación con el modelo **DecisionTreeClassifier**.

```
from sklearn import tree
dtc = tree.DecisionTreeRegressor(random_state=random_seed)
dtc.fit(X_train, y_train)
y_pred_dectre = dtc.predict(X_test)
y_pred_dectre = pd.Series(data=y_pred_dectre, index=y_test.index, name=
    var_name)
```

Por último, se implementó un árbol aleatorio de clasificación con el modelo *RandomForestClassifier*.

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(random_state=random_seed)
rfc.fit(X_train, y_train)
y_pred_ranfor = rfc.predict(X_test)
y_pred_ranfor = pd.Series(data=y_pred_ranfor, index=y_test.index, name=
    var_name)
```

9.2.8. Imputación mediante métodos de aprendizaje profundo

En la tercera sección, se realiza imputación utilizando una Red Neuronal Artificial utilizando el modelo **MLPClassifier** con los parámetros por defecto de 100 neuronas con una capa escondida, función de activación ReLU, solucionador Adam, tasa de aprendizaje constante de 0.001 y 200 iteraciones.

```
from sklearn.neural_network import MLPClassifier
ann = MLPClassifier(random_state=random_seed)
ann.fit(X_train, y_train)
y_pred_artneu = ann.predict(X_test)
y_pred_artneu = pd.Series(data=y_pred_artneu, index=y_test.index, name=
    var_name)
```

9.2.9. Comparación de resultados

Finalmente, se realiza una comparativa en los resultados de las imputaciones entre los distintos métodos a partir de los valores reales, *y_test*, y las predicciones, *y_pred*. La medición del desempeño se realizó utilizando la precisión (*accuracy*).

```
methods = ['Media', 'Mediana', 'Maxima Frecuencia',
           'Regresion logistica', 'Maquina de Vector de Soporte', 'Arbol
    de Decision',
           'Bosque Aleatorio', 'Red Neuronal Artificial']
y_pred = [y_pred_mean, y_pred_median, y_pred_max_freq,
          y_pred_logreg, y_pred_supvec, y_pred_dectre,
          y_pred_ranfor, y_pred_artneu]

for index, method in enumerate(methods):
    results.append({
        'Porcentaje de Nulos' : per_null,
        'Metodo': method,
        'Precision': round(accuracy_score(y_test, y_pred[index])*100, 2),
        'Duracion' : times[index]})

results = pd.DataFrame(results)
```

Se empleó el siguiente código para crear el Cuadro 9.6 y la Figura 9.5 en la cual se pueden visualizar los resultados de precisión obtenidos por cada método de imputación para distintos porcentajes de nulidad.

```
classification_results = results.pivot(index='Porcentaje de Nulos',
    columns='Metodo', values='Precision')
classification_results.plot(grid=True, legend=True, figsize=(12,7))
```

Luego, se empleó el siguiente código para crear el Cuadro 9.7 y la Figura 9.6 en la cual se pueden visualizar los resultados de los tiempos de imputación para cada método a distintos porcentajes de nulidad.

Porcentaje de Nulos	Arbol de Decision	Bosque Aleatorio	Maquina de Vector de Soporte	Maxima Frecuencia	Media	Mediana	Red Neuronal Artificial	Regresion logistica
0.05	100.00	100.00	100.00	25.00	37.50	37.50	100.00	100.00
0.10	100.00	100.00	100.00	26.67	40.00	40.00	100.00	100.00
0.15	95.65	95.65	95.65	17.39	47.83	47.83	82.61	95.65
0.20	96.67	96.67	96.67	20.00	43.33	43.33	86.67	96.67
0.25	97.37	97.37	97.37	23.68	42.11	42.11	89.47	97.37
0.30	95.56	95.56	97.78	28.89	40.00	40.00	91.11	97.78
0.35	96.23	96.23	98.11	28.30	35.85	35.85	94.34	98.11
0.40	96.67	96.67	98.33	31.67	31.67	35.00	95.00	96.67
0.45	97.06	97.06	98.53	30.88	33.82	30.88	97.06	97.06

Cuadro 9.6: Precisión para métodos de imputación sobre target

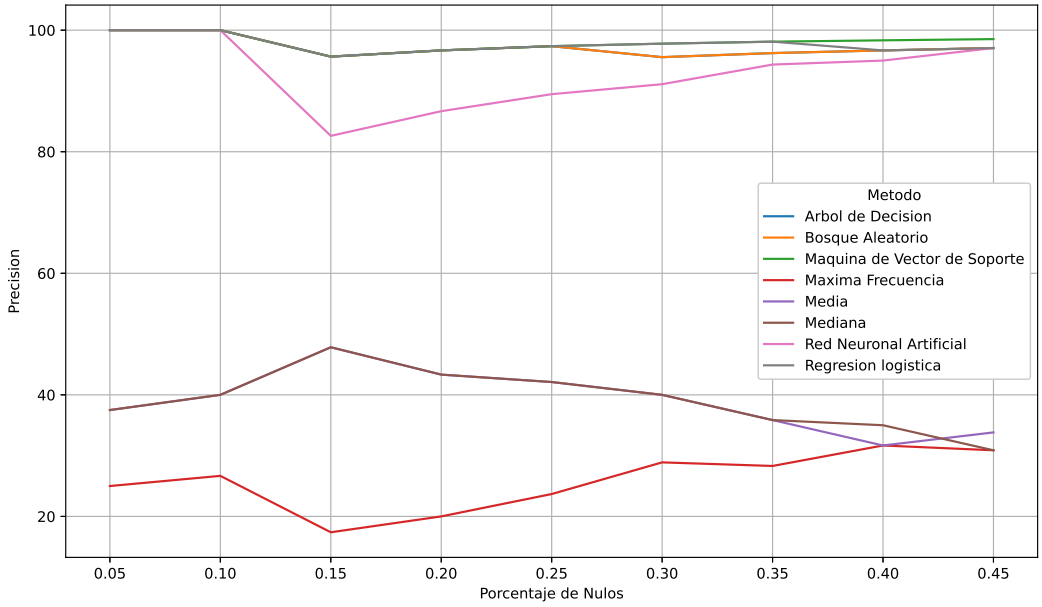

Figura 9.5: Precisión para métodos de imputación sobre target

```
classification_times = results.pivot(index='Porcentaje de Nulos', columns=
    'Metodo', values='Duracion')
classification_times.plot(grid=True, legend=True, figsize=(12,7))
```

Finalmente, se creó el siguiente código para agrupar los resultados promedios obtenidos por cada método de forma ascendente (Cuadro 9.8).

```
results.groupby(by='Metodo').mean().sort_values(by='Precision', ascending=
    False)[['Precision', 'Duracion']]
```

Porcentaje de Nulos	Arbol de Decision	Bosque Aleatorio	Maquina de Vector de Soporte	Maxima Frecuencia	Media	Mediana	Red Neuronal Artificial	Regresion logistica
0.05	0.001142	0.018218	0.001379	0.000077	0.000161	0.000148	0.003829	0.001823
0.10	0.001053	0.011215	0.001296	0.000118	0.000152	0.000157	0.001914	0.001591
0.15	0.001072	0.011154	0.001362	0.000073	0.000112	0.000078	0.002234	0.001796
0.20	0.001076	0.011035	0.001318	0.000068	0.000107	0.000083	0.002785	0.001614
0.25	0.001093	0.013253	0.001394	0.000079	0.000180	0.000621	0.007279	0.001668
0.30	0.001110	0.016671	0.001473	0.000070	0.000133	0.000089	0.002714	0.001739
0.35	0.001127	0.013248	0.002061	0.000083	0.000106	0.000075	0.002228	0.002170
0.40	0.001071	0.012715	0.001384	0.000068	0.000103	0.000106	0.001817	0.001650
0.45	0.001351	0.012508	0.001538	0.000122	0.000175	0.000118	0.001863	0.001977

Cuadro 9.7: Duración para métodos de imputación sobre target

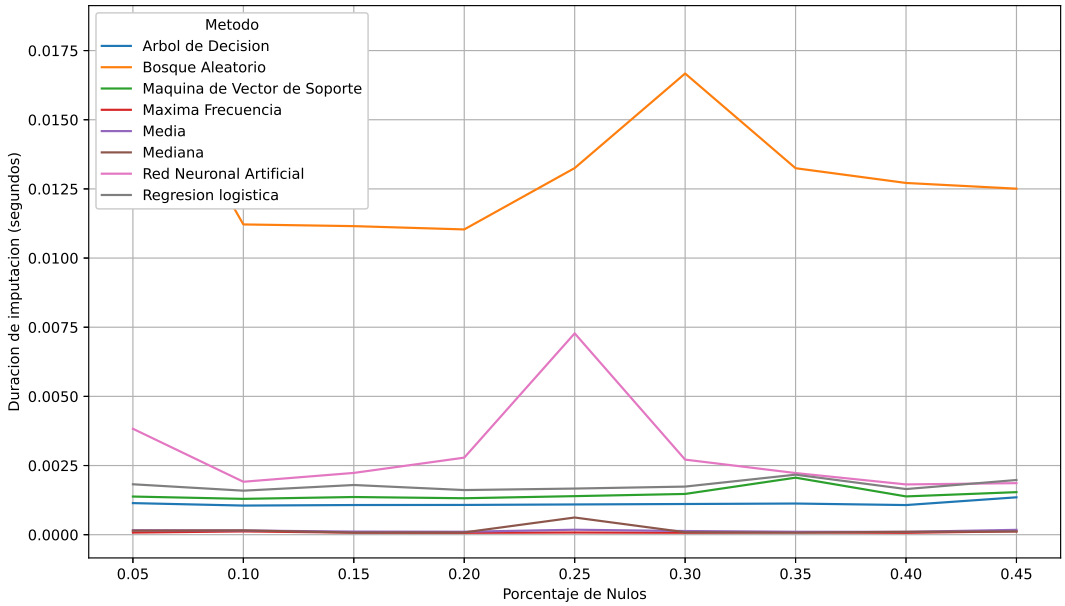

Figura 9.6: Duración para métodos de imputación sobre target

9.3. Mixto

9.3.1. Repositorio

El desarrollo metodológico descrito a continuación se encuentra disponible en el siguiente cuaderno de Jupyter Notebook desde Google Colab.

UNIR - Proyecto Final de Maestría - Mixto

Metodo	Precision	Duracion
Regresion logistica	81.467778	0.002448
Red Neuronal Artificial	80.047778	0.003697
Bosque Aleatorio	79.481111	0.022638
Arbol de Decision	77.215556	0.001577
Maquina de Vector de Soporte	66.314444	0.017495
Maxima Frecuencia	60.874444	0.000146
Media	60.874444	0.000194
Mediana	60.874444	0.000129

Cuadro 9.8: Resultados promedio para métodos de imputación sobre target

9.3.2. Conjunto de datos

El conjunto de datos utilizado para la metodología fue el Titanic dataset disponible en el repositorio de OpenML. Las variables objetivos fueron: edad, para los métodos de regresión; y sobreviviente, para los métodos de clasificación.

Este conjunto de datos describe el estado de supervivencia de los pasajeros del Titanic. La principal fuente de datos sobre los pasajeros del Titanic es la Enciclopedia Titanica con los aportes y revisión de Thomas Cason de la Universidad de Virginia.

Las variables de nuestro conjunto de datos extraídos son pclass, survived, name, age, embarked, home.dest, room, ticket, boat, y sex. pclass se refiere a la clase de pasajero (1ª, 2ª, 3ª), y es una aproximación a la clase socioeconómica. La edad se expresa en años, y algunos niños tenían valores fraccionarios.

El conjunto de datos está conformado por 1309 registros, se utilizaron los siguientes atributos numéricos y categóricos:

- pclass: Clase del pasajero (1a, 2a, 3a).
- sex: Género (masculino o femenino).
- age: Edad en años (incluyendo fracciones para niños).
- sibsp: Cantidad de hermanos o cónyuges abordo.
- parch: Cantidad de padres o hijos abordo.
- fare: precio del boleto.
- survived: si el pasajero sobrevivió o no.

	pclass	age	sibsp	parch	fare
count	1309.00	1046.00	1309.00	1309.00	1308.00
mean	2.29	29.88	0.50	0.39	33.30
std	0.84	14.41	1.04	0.87	51.76
min	1.00	0.17	0.00	0.00	0.00
25 %	2.00	21.00	0.00	0.00	7.90
50 %	3.00	28.00	0.00	0.00	14.45
75 %	3.00	39.00	1.00	0.00	31.28
max	3.00	80.00	8.00	9.00	512.33

Cuadro 9.9: Estadística descriptiva del Titanic Data Set

Se importó directamente como un DataFrame de Pandas mediante la siguiente línea.

```
from sklearn.datasets import fetch_openml
data = fetch_openml("titanic", version=1,as_frame=True, target_column=None
    ).data
```

Se ejecutó el siguiente código para obtener una descripción estadística de los datos (Cuadro 9.9.)

```
data.describe()
```

9.3.3. Preprocesamiento de datos

Se identificaron algunos valores nulos en las columnas *age* y *fare*, por lo que se implementó el siguiente código para imputar los datos faltantes en ambas columnas utilizando la media con el objetivo de tener la completitud de la información, antes de realizar la metodología de comparación ya que de lo contrario, no se podrían utilizar estas columnas durante la implementación de los métodos de aprendizaje automático.

```
for column in data:
    if data[column].isnull().sum() > 0:
        data[column].fillna(data[column].mean(), inplace=True)
```

Se identificaron distintos tipos de datos utilizando el siguiente método.

```
data.dtypes
```

Debido a que los atributos de género y supervivencia son categóricos fue necesario realizar codificación de los datos para transformarlos a valores numéricos. En el caso del atributo de género que poseía los valores de "malez "female", se utilizó el *OneHotEncoder* para crear una columna "malez "femaleçon la codificación correspondiente.

```
from sklearn.preprocessing import OneHotEncoder

oe = OneHotEncoder()
data = data.join(pd.DataFrame(columns= ['female', 'male'], data=oe.
    fit_transform(data[['sex']]).toarray()))
data.drop(columns='sex', inplace=True)
data.head()
```

Para el atributo *survived* que poseía los valores ['0', '1'], simplemente se aplicó el *LabelEncoder* para convertir los valores a [0, 1].

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
data['survived'] = le.fit_transform(data['survived'])
```

9.3.4. Regresión

En la comparativa de regresión, se divide el conjunto de datos en una matriz de variables independientes, X, y un vector de salida con la variable dependiente, y. En este caso, se realizará la imputación sobre la columna *age*.

```
X = data[['pclass', 'fare', 'sibsp', 'parch', 'female', 'male', 'survived'
    ]]
y = data['age']
```

En este análisis comparativo, se ejecutarán todos los métodos empleados en el Capítulo de Regresión para distintos porcentajes de nulidad de los datos y midiendo los tiempos de ejecución para cada uno. En esta sección, en lugar de explicar nuevamente la implementación independiente de cada método de imputación, se profundizará más en la metodología utilizada para realizar las comparativas.

Metodología de imputación para distintos porcentajes de nulidad

1. Se define un iterador con el rango de valores desde 0.05 hasta 0.45 en incrementos de 0.05 y se inicializan las variables de la columna de imputación, la semilla, una lista de resultados y un DataFrame para recopilar los valores nulos.

2. Por cada valor de porcentaje de nulidad, se realiza lo siguiente:

3. Se divide el conjunto de datos en un conjunto de entrenamiento (*X_train*, *y_train*) y un conjunto de datos de prueba (*X_test*, *y_test*) sobre el cual se realizará la imputación de valores.

4. Se retiran los valores del conjunto de datos de prueba del conjunto de datos original para generar la visualización de nulidad.
5. Por cada método de imputación, se realiza lo siguiente:
6. Se entrena el algoritmo sobre el conjunto de datos de entrenamiento.
7. Se inicia el temporizador.
8. Se realiza la imputación sobre el conjunto de datos de prueba.
9. Se detiene el temporizador.
10. Se agregan los resultados de medición de desempeño y tiempo de imputación a la lista de resultados y se genera un DataFrame.

```
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from time import time
import missingno as msno
import matplotlib.pyplot as plt

per_null_values = map(lambda x: x/100, range(5, 50, 5))
var_name = 'age'
random_seed = 1
results = []
df_null = pd.DataFrame()

for per_null in per_null_values:
    times = []
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=
   per_null, random_state=random_seed)
    #recopilacion de columnas con valores nulos
    missing_data = data.copy()
    missing_data.loc[y_test.index, var_name] = np.nan
    df_null[f'{str(per_null)}'] = missing_data[var_name]

```

```
    #imputacion mediante metodos estadisticos
    mean = y_train.mean()
    start_time = time()
    y_pred_mean = pd.Series(data=mean, index=y_test.index, name=var_name)
    end_time = time()
    duration = end_time - start_time
    times.append(duration)

    median = y_train.median()
    start_time = time()
    y_pred_median = pd.Series(data=median, index=y_test.index, name=
var_name)
    end_time = time()
    duration = end_time - start_time
    times.append(duration)

    max_freq = y_train.value_counts().idxmax()
    start_time = time()
    y_pred_max_freq = pd.Series(data=max_freq, index=y_test.index, name=
var_name)
    end_time = time()
    duration = end_time - start_time
    times.append(duration)

    #imputacion mediante metodos de aprendizaje automatico
    lm = LinearRegression()
    lm.fit(X_train, y_train)
    start_time = time()
    y_pred_linreg = lm.predict(X_test)
    y_pred_linreg = pd.Series(data=y_pred_linreg, index=y_test.index, name
=var_name)
    end_time = time()
    duration = end_time - start_time
    times.append(duration)

    svr = SVR()
    svr.fit(X_train, y_train)
    start_time = time()
    y_pred_supvec = svr.predict(X_test)
    y_pred_supvec = pd.Series(data=y_pred_supvec, index=y_test.index, name
=var_name)
    end_time = time()
```

```
    duration = end_time - start_time
    times.append(duration)

    dtr = tree.DecisionTreeRegressor(random_state=random_seed)
    dtr.fit(X_train, y_train)
    start_time = time()
    y_pred_dectre = dtr.predict(X_test)
    y_pred_dectre = pd.Series(data=y_pred_dectre, index=y_test.index, name
=var_name)
    end_time = time()
    duration = end_time - start_time
    times.append(duration)

    rfr = RandomForestRegressor(random_state=random_seed)
    rfr.fit(X_train, y_train)
    start_time = time()
    y_pred_ranfor = rfr.predict(X_test)
    y_pred_ranfor = pd.Series(data=y_pred_ranfor, index=y_test.index, name
=var_name)
    end_time = time()
    duration = end_time - start_time
    times.append(duration)

    #imputacion mediante metodos de aprendizaje profundo
    ann = MLPRegressor(random_state=random_seed)
    ann.fit(X_train, y_train)
    start_time = time()
    y_pred_artneu = ann.predict(X_test)
    y_pred_artneu = pd.Series(data=y_pred_artneu, index=y_test.index, name
=var_name)
    end_time = time()
    duration = end_time - start_time
    times.append(duration)

    methods = ['Media', 'Mediana', 'Maxima Frecuencia',
              'Regresion lineal multiple', 'Maquina de Vector de Soporte',
    'Arbol de Decision',
              'Bosque Aleatorio', 'Red Neuronal Artificial']
    y_pred = [y_pred_mean, y_pred_median, y_pred_max_freq,
              y_pred_linreg, y_pred_supvec, y_pred_dectre,
              y_pred_ranfor, y_pred_artneu]

```

```
    for index, method in enumerate(methods):
        results.append({
            'Porcentaje de Nulos' : per_null,
            'Metodo': method,
            'RMSE': round(mean_squared_error(y_test, y_pred[index]), 2),
            'Duracion' : times[index]})

results = pd.DataFrame(results)
```

Se generó la Figura 9.7 en la cual se pueden visualizar los nulos introducidos dentro de la variable objetivo para cada porcentaje de nulidad.

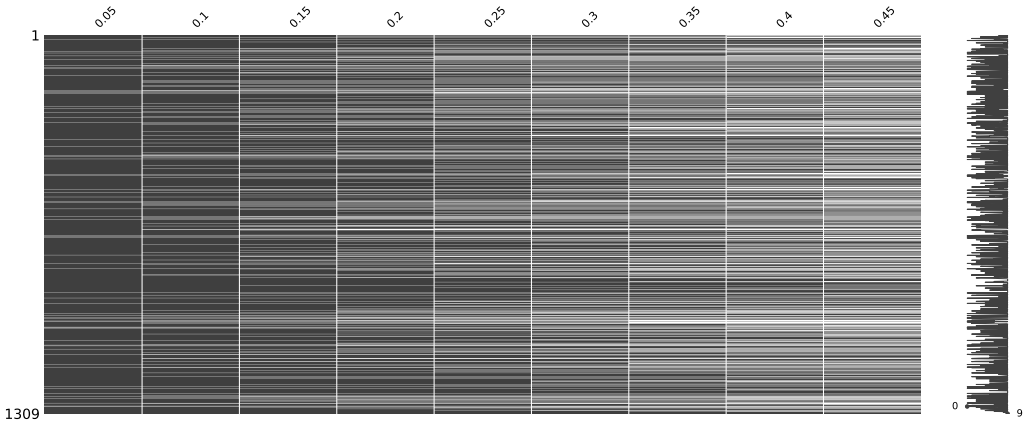

Figura 9.7: Nulos introducidos en age del Titanic Dataset

Finalmente, se empleó el siguiente código para crear tablas pivotadas que permiten comparar de mejor manera los resultados obtenidos para el error medio (Cuadro 9.10) y la duración del algoritmo (Cuadro 9.11). Además, se crearon las figuras para visualizar el error (Figura 9.8) y el tiempo de ejecución (Figura 9.9) de cada método para distintos porcentajes de nulidad.

```
regression_results = results.pivot(index='Porcentaje de Nulos', columns='
    Metodo', values='RMSE')
regression_results.plot(grid=True, legend=True, figsize=(12,7))
regression_times = results.pivot(index='Porcentaje de Nulos', columns='
    Metodo', values='Duracion')
regression_times.plot(grid=True, legend=True, figsize=(12,7))
```

Un resumen de los resultados promedio para el error y duración se encuentra en el Cuadro 9.12.

Porcentaje de Nulos	Arbol de Decision	Bosque Aleatorio	Maquina de Vector de Soporte	Maxima Frecuencia	Media	Mediana	Red Neuronal Artificial	Regresion lineal multiple
0.05	130.45	121.39	170.88	174.90	175.07	174.90	150.52	139.10
0.10	141.54	112.97	154.67	159.52	159.60	159.52	143.63	125.79
0.15	183.27	131.64	171.24	175.48	175.48	175.48	163.56	129.53
0.20	177.33	133.80	168.15	172.81	172.82	172.81	146.09	129.51
0.25	194.60	131.42	169.40	172.75	172.86	172.75	166.09	129.28
0.30	180.73	121.11	163.97	166.17	166.20	166.17	152.66	125.22
0.35	180.58	123.39	158.55	160.20	160.27	160.20	159.36	124.35
0.40	182.14	137.04	163.49	165.29	165.30	165.29	170.29	131.61
0.45	190.68	139.92	160.97	162.86	162.90	162.86	166.69	132.82

Cuadro 9.10: Error para métodos de imputación sobre age

Porcentaje de Nulos	Arbol de Decision	Bosque Aleatorio	Maquina de Vector de Soporte	Maxima Frecuencia	Media	Mediana	Red Neuronal Artificial	Regresion lineal multiple
0.05	0.002217	0.016297	0.005690	0.000076	0.000363	0.000103	0.003727	0.001531
0.10	0.001761	0.013636	0.009728	0.000113	0.000163	0.000119	0.015439	0.002095
0.15	0.002762	0.029754	0.024430	0.000115	0.000168	0.005362	0.008409	0.008018
0.20	0.002771	0.027079	0.037796	0.000096	0.000142	0.000112	0.006459	0.009187
0.25	0.002789	0.027909	0.068680	0.000364	0.000166	0.000129	0.005599	0.002712
0.30	0.002774	0.029014	0.034116	0.000244	0.000164	0.000319	0.014025	0.002661
0.35	0.002965	0.077009	0.038486	0.000100	0.000206	0.000135	0.009192	0.004487
0.40	0.002724	0.120457	0.038611	0.000093	0.000157	0.000122	0.007217	0.017840
0.45	0.010104	0.034708	0.052550	0.000122	0.000160	0.000143	0.006691	0.002626

Cuadro 9.11: Duración para métodos de imputación sobre age

Metodo	RMSE	Duracion
Bosque Aleatorio	128.075556	0.041763
Regresion lineal multiple	129.690000	0.005684
Red Neuronal Artificial	157.654444	0.008528
Maquina de Vector de Soporte	164.591111	0.034454
Maxima Frecuencia	167.775556	0.000147
Mediana	167.775556	0.000727
Media	167.833333	0.000188
Arbol de Decision	173.480000	0.003430

Cuadro 9.12: Resultados promedio para métodos de imputación sobre age

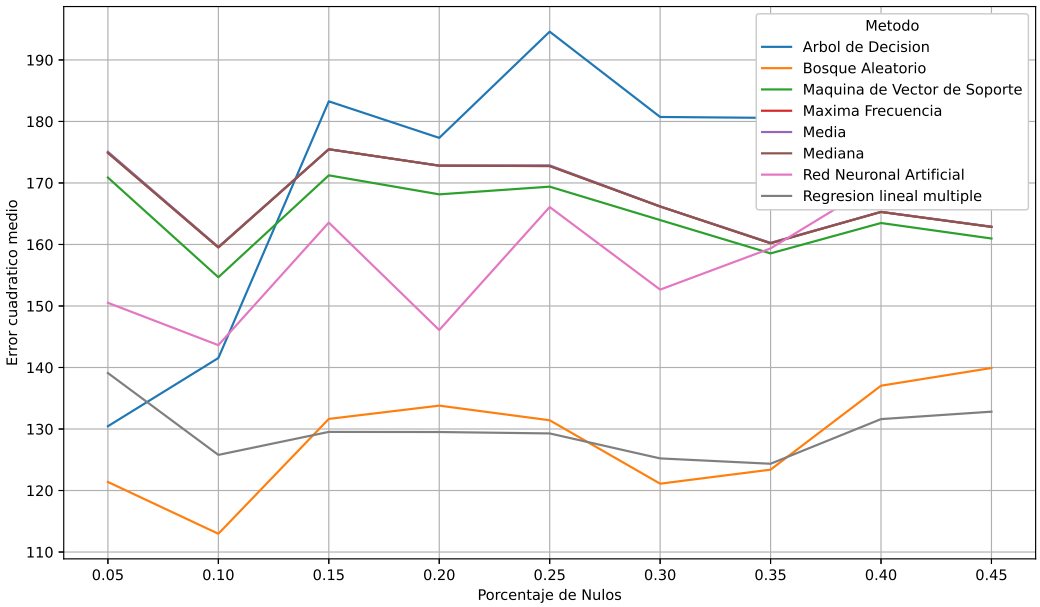

Figura 9.8: Error para métodos de imputación sobre age

9.3.5. Clasificación

En la comparativa de clasificación, se divide el conjunto de datos en una matriz de variables independientes, X, y un vector de salida con la variable dependiente, y. En este caso, se realizará la imputación sobre la columna *survived*.

```
X = data[['pclass', 'age', 'sibsp', 'parch', 'female', 'male', 'fare']]
y = data['survived']
```

En este análisis comparativo, se ejecutarán todos los métodos empleados en el Capítulo de Clasificación para distintos porcentajes de nulidad de los datos y midiendo los tiempos de ejecución para cada uno. Se utiliza la misma metodología explicada en la sección anterior pero para métodos de clasificación.

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

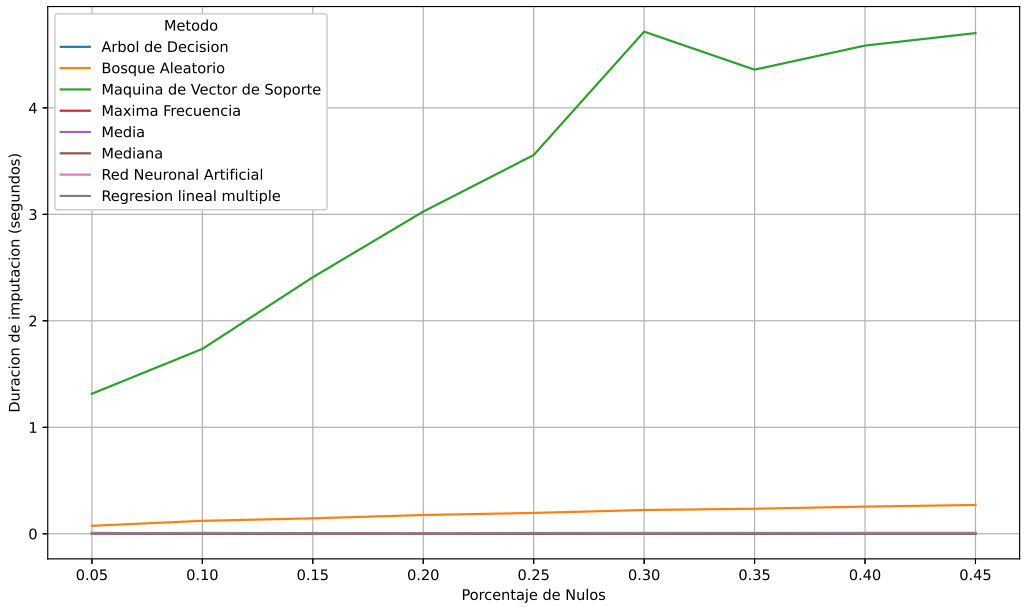

Figura 9.9: Duración para métodos de imputación sobre age

```
from time import time
import missingno as msno
import matplotlib.pyplot as plt

per_null_values = map(lambda x: x/100, range(5, 50, 5))
var_name = 'survived'
random_seed = 1
results = []
df_null = pd.DataFrame()

for per_null in per_null_values:
    times = []
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=
   per_null, random_state=random_seed)
    #recopilacion de columnas con valores nulos
    missing_data = data.copy()
    missing_data.loc[y_test.index, var_name] = np.nan
    df_null[f'{str(per_null)}'] = missing_data[var_name]

    #imputacion mediante metodos estadisticos
```

```
    mean = y_train.mean()
    start_time = time()
    y_pred_mean = pd.Series(data=mean, index=y_test.index, name=var_name,
dtype=int)
    end_time = time()
    duration = end_time - start_time
    times.append(duration)

    median = y_train.median()
    start_time = time()
    y_pred_median = pd.Series(data=median, index=y_test.index, name=
var_name, dtype=int)
    end_time = time()
    duration = end_time - start_time
    times.append(duration)

    max_freq = y_train.value_counts().idxmax()
    start_time = time()
    y_pred_max_freq = pd.Series(data=max_freq, index=y_test.index, name=
var_name)
    end_time = time()
    duration = end_time - start_time
    times.append(duration)

    #imputacion mediante metodos de aprendizaje automatico
    lm = LogisticRegression()
    lm.fit(X_train, y_train)
    start_time = time()
    y_pred_logreg = lm.predict(X_test)
    y_pred_logreg = pd.Series(data=y_pred_logreg, index=y_test.index, name
=var_name)
    end_time = time()
    duration = end_time - start_time
    times.append(duration)

    svc = SVC()
    svc.fit(X_train, y_train)
    start_time = time()
    y_pred_supvec = svc.predict(X_test)
    y_pred_supvec = pd.Series(data=y_pred_supvec, index=y_test.index, name
=var_name)
    end_time = time()
```

```
    duration = end_time - start_time
    times.append(duration)

    dtc = tree.DecisionTreeClassifier(random_state=random_seed)
    dtc.fit(X_train, y_train)
    start_time = time()
    y_pred_dectre = dtc.predict(X_test)
    y_pred_dectre = pd.Series(data=y_pred_dectre, index=y_test.index, name
=var_name)
    end_time = time()
    duration = end_time - start_time
    times.append(duration)


    rfc = RandomForestClassifier(random_state=random_seed)
    rfc.fit(X_train, y_train)
    start_time = time()
    y_pred_ranfor = rfc.predict(X_test)
    y_pred_ranfor = pd.Series(data=y_pred_ranfor, index=y_test.index, name
=var_name)
    end_time = time()
    duration = end_time - start_time
    times.append(duration)

    #imputacion mediante metodos de aprendizaje profundo
    ann = MLPClassifier(random_state=random_seed)
    ann.fit(X_train, y_train)
    start_time = time()
    y_pred_artneu = ann.predict(X_test)
    y_pred_artneu = pd.Series(data=y_pred_artneu, index=y_test.index, name
=var_name)
    end_time = time()
    duration = end_time - start_time
    times.append(duration)

    methods = ['Media', 'Mediana', 'Maxima Frecuencia',
               'Regresion logistica', 'Maquina de Vector de Soporte', '
Arbol de Decision',
               'Bosque Aleatorio', 'Red Neuronal Artificial']
    y_pred = [y_pred_mean, y_pred_median, y_pred_max_freq,
              y_pred_logreg, y_pred_supvec, y_pred_dectre,
              y_pred_ranfor, y_pred_artneu]
```

```
102
103     for index, method in enumerate(methods):
104         results.append({
105             'Porcentaje de Nulos' : per_null,
106             'Metodo': method,
107             'Precision': round(accuracy_score(y_test, y_pred[index])*100,
    2),
108             'Duracion' : times[index]})
109
110 results = pd.DataFrame(results)
```

Se generó la Figura 9.10 en la cual se pueden visualizar los nulos introducidos dentro de la variable objetivo para cada porcentaje de nulidad.

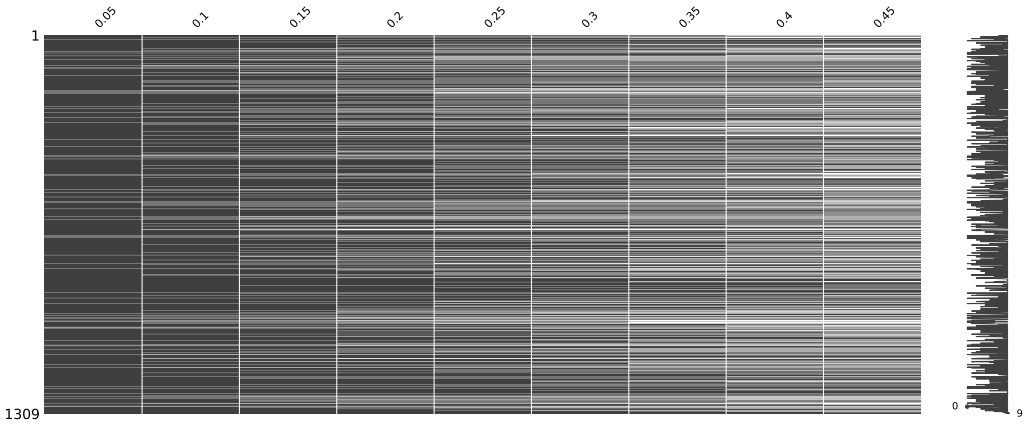

Figura 9.10: Nulos introducidos en survived del Titanic Dataset

Finalmente, se empleó el siguiente código para crear tablas pivotadas que permiten comparar de mejor manera los resultados obtenidos para la precisión (Cuadro 9.13) y la duración del algoritmo (Cuadro 9.14). Además, se crearon las figuras para visualizar el error (Figura 9.5) y el tiempo de ejecución (Figura 9.12) de cada método para distintos porcentajes de nulidad.

```
1 classification_results = results.pivot(index='Porcentaje de Nulos',
    columns='Metodo', values='Precision')
2 classification_results.plot(grid=True, legend=True, figsize=(12,7))
3 classification_times = results.pivot(index='Porcentaje de Nulos', columns=
    'Metodo', values='Duracion')
4 classification_times.plot(grid=True, legend=True, figsize=(12,7))
```

Porcentaje de Nulos	Arbol de Decision	Bosque Aleatorio	Maquina de Vector de Soporte	Maxima Frecuencia	Media	Mediana	Red Neuronal Artificial	Regresion logistica
0.05	81.82	84.85	74.24	66.67	66.67	66.67	80.30	83.33
0.10	80.15	83.21	69.47	61.07	61.07	61.07	80.92	83.21
0.15	77.16	79.19	65.99	59.90	59.90	59.90	83.25	83.25
0.20	77.10	78.24	64.12	59.54	59.54	59.54	80.53	82.44
0.25	76.52	77.13	63.11	59.45	59.45	59.45	79.27	81.10
0.30	76.84	78.88	64.38	60.05	60.05	60.05	79.64	80.15
0.35	76.03	78.00	64.92	60.35	60.35	60.35	79.30	80.17
0.40	74.24	77.86	64.50	60.50	60.50	60.50	78.24	79.39
0.45	75.08	77.97	66.10	60.34	60.34	60.34	78.98	80.17

Cuadro 9.13: Precisión para métodos de imputación sobre survived

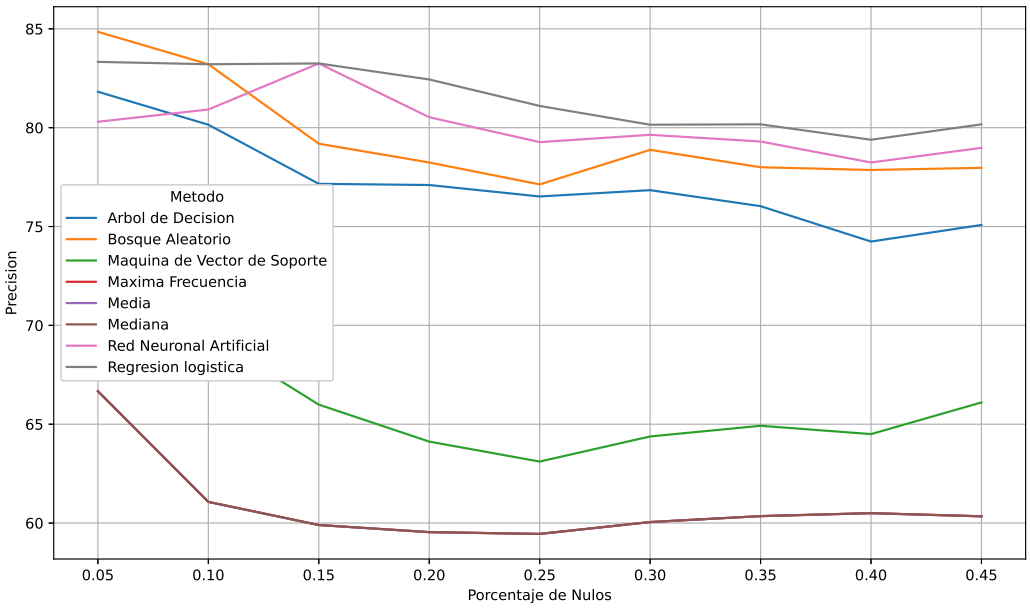

Figura 9.11: Precisión para métodos de imputación sobre survived

Un resumen de los resultados promedio para la precisión y duración se encuentra en el Cuadro 9.15.

Porcentaje de Nulos	Arbol de Decision	Bosque Aleatorio	Maquina de Vector de Soporte	Maxima Frecuencia	Media	Mediana	Red Neuronal Artificial	Regresion logistica
0.05	0.001528	0.013255	0.004560	0.000351	0.000427	0.000096	0.002418	0.001564
0.10	0.001817	0.022794	0.007607	0.000183	0.000146	0.000159	0.003203	0.002808
0.15	0.001366	0.016757	0.008991	0.000111	0.000144	0.000123	0.002551	0.002093
0.20	0.001557	0.017911	0.011388	0.000122	0.000170	0.000135	0.002615	0.002141
0.25	0.001669	0.020736	0.013835	0.000110	0.000190	0.000134	0.004188	0.002993
0.30	0.001351	0.020363	0.019758	0.000104	0.000196	0.000137	0.003131	0.002110
0.35	0.001431	0.021024	0.026620	0.000099	0.000150	0.000136	0.002703	0.002092
0.40	0.001516	0.024595	0.031796	0.000111	0.000145	0.000126	0.003663	0.003513
0.45	0.001960	0.046310	0.032901	0.000123	0.000182	0.000118	0.008800	0.002717

Cuadro 9.14: Duración para métodos de imputación sobre survived

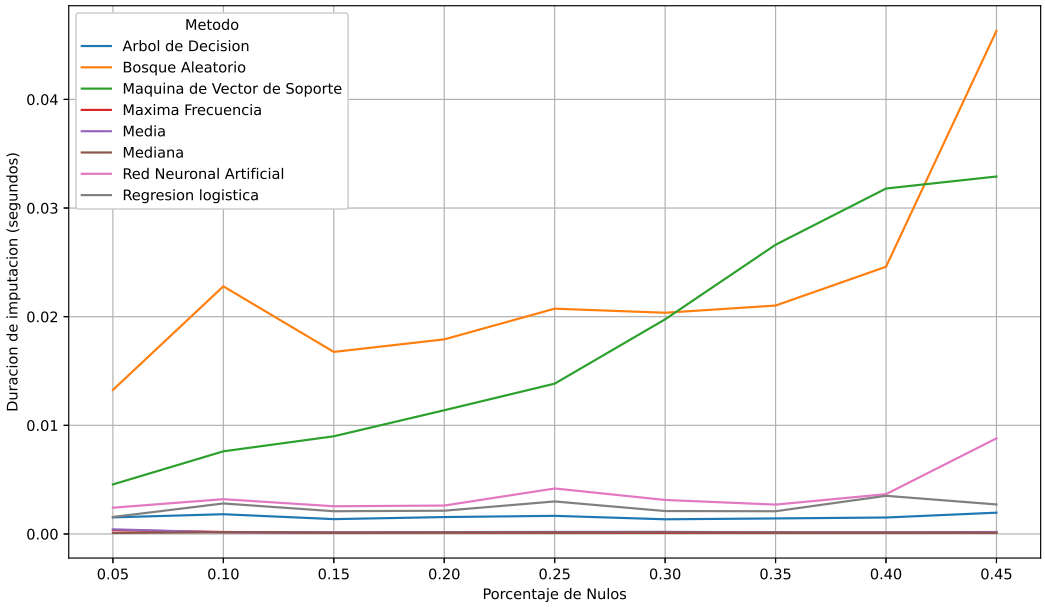

Figura 9.12: Duración para métodos de imputación sobre survived

Metodo	Precision	Duracion
Regresion logistica	81.467778	0.002448
Red Neuronal Artificial	80.047778	0.003697
Bosque Aleatorio	79.481111	0.022638
Arbol de Decision	77.215556	0.001577
Maquina de Vector de Soporte	66.314444	0.017495
Maxima Frecuencia	60.874444	0.000146
Media	60.874444	0.000194
Mediana	60.874444	0.000129

Cuadro 9.15: Resultados promedio para métodos de imputación sobre survived

CAPÍTULO 10

Discusión de Resultados

En el estudio de regresión, los resultados obtenidos a partir de la función de costo del error cuadrático medio demuestran la superioridad de los algoritmos de aprendizaje automático frente a los métodos de aprendizaje profundo y métodos estadísticos convencionales. El menor error promedio fue obtenido por el bosque aleatorio con un 0.258889, seguido por el árbol de decisión para regresión con 0.510000. Sería de evaluar el tamaño del conjunto de datos para considerar si resulta conveniente la reducción de este margen de error a cambio de un incremento significativo del tiempo de ejecución. Luego, la regresión lineal múltiple obtuvo un error cercano de 0.532222. La red neuronal artificial obtuvo un error ligeramente superior de 0.668889. Los métodos estadísticos obtuvieron los errores más elevados con 1.323333, 1.378889 y 10.052222 para la media, mediana y máxima frecuencia, respectivamente. La máquina de vector de soporte también obtuvo un error de 1.340000 ubicándose entre la media y mediana.

Los algoritmos fueron implementados a partir de los parámetros por defecto de la librería de *scikit-learn*, por lo que se sugeriría evaluar su desempeño modificando las configuraciones para obtener los mejores resultados posibles de cada algoritmo de regresión. Los resultados obtenidos se realizaron a partir de una semilla de aletoriedad definida, por lo que también sería conveniente replicar la metodología para distintas distribuciones y evaluar si el desempeño de los algoritmos se mantiene. El conjunto de datos utilizado contempla únicamente valores numéricos por lo que se sugeriría complementar el estudio con algoritmos de clasificación y la implementación de codificación de variables categóricas para obtener una metodología que abarque la totalidad de los

tipos de datos existentes en el mundo real.

Por esta razón, se decidió replicar el estudio sobre un conjunto de datos con una variable objetivo de clasificación. En este caso, la métrica utilizada fue la precisión y, nuevamente, los métodos de aprendizaje automático y aprendizaje profundo obtuvieron los mejores resultados frente a los métodos de imputación clásicos. Los métodos de aprendizaje de máquina obtuvieron las precisiones más elevadas con un 81.467778 % par la regresión logística, seguida de la red neuronal artificial con 80.047778 % y el bosque aleatorio con 79.481111 %. Los métodos de imputación estadística obtuvieron la misma precisión de 60.874444 %. En este caso, la regresión logística demostró ser el método más eficiente alcanzando la mayor precisión y un tiempo de ejecución menor que el de la red neuronal y el bosque aleatorio.

Finalmente, se realizó un estudio comparativo mixto en el cual se evalúo el desempeño de los métodos de regresión y clasificación sobre un mismo conjunto de datos compuesto por variables numéricas y categóricas nominales así como ordinales.

Los resultados obtenidos para la tarea de regresión demuestran un menor error para el método de bosque aleatorio con un valor de 128.075556 seguido por la regresión lineal múltiple con 129.69 y, finalmente, la red neuronal artificial con un error de 157.654444. Los errores obtenidos por el bosque aleatorio y la regresión lineal múltiple son bastante similares, sin embargo, el tiempo de imputación del bosque aleatorio es 7 veces más grande que el de la regresión lineal múltiple.

Los resultados para clasificación muestran las mayores precisiones para la regresión logística con un 81.467778 %, la red neuronal artificial con 80.047778 % y el bosque aleatorio con 79.481111 %. Los tiempos de ejecución para estos tres algoritmos demuestran un menor tiempo para la regresión logística con 0.002448 segundos, seguido por la red neuronal artificial con 0.003697 segundos y el bosque aleatorio con 0.022638.

CAPÍTULO 11

Conclusiones

1. La imputación mediante métodos de aprendizaje automático y aprendizaje profundo obtuvieron el menor error cuadrático medio y la mayor precisión comparado con los métodos estadísticos convencionales.

2. Los métodos de bosque aleatorio y regresión lineal múltiple obtuvieron los menores errores para la imputación de datos numéricos. El tiempo de ejecución fue menor para la regresión lineal múltiple.

3. Los métodos de regresión logística, red neuronal artificial y bosque aleatorio obtuvieron las mayores precisiones para la imputación de datos categóricos. El tiempo de ejecución de la regresión logística y la red neuronal fueron inferiores al del bosque aleatorio.

CAPÍTULO 12

Recomendaciones

1. Evaluar la metodología propuesta sobre conjuntos de datos diversos con distintos volúmenes de datos para validar los resultados obtenidos y determinar los modelos de regresión y clasificación que ofrecen los mejores resultados en el menor tiempo de ejecución.
2. Modificar los parámetros de configuración de los métodos de aprendizaje supervisado para disminuir el error (regresión) y aumentar la precisión (clasificación).
3. Replicar la metodología sobre otros métodos de imputación que no fueron consideradores en el presente estudio comparativo.
4. Incorporar un análisis de la distribución de los datos de las columnas a imputar para evaluar el desempeño de los algoritmos bajo múltiples escenarios estadísticos y determinar la consistencia de los resultados.

CAPÍTULO 13

Bibliografía

Duan, Y., Lv, Y., Kang, W., & Zhao, Y. (2014). A deep learning based approach for traffic data imputation. *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. https://doi.org/10.1109/itsc.2014.6957805

Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Van Duuren Media.

Jadhav, A., Pramod, D., & Ramanathan, K. (2019). Comparison of Performance of Data Imputation Methods for Numeric Dataset. *Applied Artificial Intelligence*, *33*(10), 913-933. https://doi.org/10.1080/08839514.2019.1637138

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning: With Applications in R* (2nd 2021 ed.). Springer.

Jerez, J. M., Molina, I., García-Laencina, P. J., Alba, E., Ribelles, N., Martín, M., & Franco, L. (2010). Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artificial Intelligence in Medicine*, *50*(2), 105-115. https://doi.org/10.1016/j.artmed.2010.05.002

Little, R., & Rubin, D. (2019). *Statistical Analysis with Missing Data*. Wiley.

Oluwaseye, J. L., Wesley, D., & Sena, P. B. (2022). A Review of Missing Data Handling Techniques for Machine Learning. *International Journal of Innovative Technology and Interdisciplinary Sciences*, *5*(3), 971-1005. https://doi.org/10.15157/IJITIS.2022.5.3.971-1005

Pazhoohesh, M., Allahham, A., Das, R., & Walker, S. (2021). Investigating the impact of missing data imputation techniques on battery energy management system. *IET Smart Grid*, *4*(2), 162-175. https://doi.org/10.1049/stg2.12011

Silva-Ramírez, E.-L., Pino-Mejías, R., & López-Coello, M. (2015). Single imputation with multilayer perceptron and multiple imputation combining multilayer perceptron and k-nearest neighbours for monotone patterns. *Applied Soft Computing*, *29*, 65-74. https://doi.org/10.1016/j.asoc.2014.09.052

Sterne, J. A. C., White, I. R., Carlin, J. B., Spratt, M., Royston, P., Kenward, M. G., Wood, A. M., & Carpenter, J. R. (2009). Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. *BMJ*, *338*(jun29 1), b2393-b2393. https://doi.org/10.1136/bmj.b2393

van Buuren, S., Taylor, Group, F., & van Buuren, S. (2021). *Flexible Imputation of Missing Data, Second Edition*. Taylor; Francis.