FACULTAD DE ESTUDIOS ESTADÍSTICOS

MÁSTER EN CIENCIA DE DATOS E INTELIGENCIA DE NEGOCIOS

Curso 2024/2025

Trabajo de Fin de Máster

TITULO: *Detección de fallos por fuga de aceite en un compresor de Boil-Off Gas de una planta gas natural licuado mediante técnicas de machine learning*

Alumno: *Anyelo David Barrios Rodríguez*

Tutor: *José Fernández Menéndez – Antonio Jesús Guillen López – Antonia Sonia Liñán García*

Septiembre de 2025

UNIVERSIDAD COMPLUTENSE
MADRID

UNIVERSIDAD COMPLUTENSE MADRID

Declaración Responsable sobre Autoría y Uso Ético de Herramientas de inteligencia Artificial (IA)

Yo, Anyelo David Barrios Rodriguez con NIE: Y0731091G declaro de manera responsable que el presente:

- ☐ Trabajo de Fin de Grado (TFG)
- ■ Trabajo de Fin de Máster (TFM)
- ☐ Tesis Doctoral

Titulado **Detección de fallos por fuga de aceite en un compresor de Boil-Off Gas de una planta gas natural licuado mediante técnicas de machine learning** es el resultado de mi trabajo intelectual personal y creativo, y ha sido elaborado de acuerdo con los principios éticos y las normas de integridad vigentes en la comunidad académica y, más específicamente, en la Universidad Complutense de Madrid.

Soy, pues, autor del material aquí incluido y, cuando no ha sido así y he tomado el material de otra fuente, lo he citado o bien he declarado su procedencia de forma clara -incluidas, en su caso, herramientas de inteligencia artificial-. Las ideas y aportaciones principales incluidas en este trabajo, y que acreditan la adquisición de competencias, son mías y no proceden de otras fuentes o han sido reescritas usando material de otras fuentes.

Asimismo, aseguro que los datos y recursos utilizados son legítimos, verificables y han sido obtenidos de fuentes confiables y autorizadas. Además, he tomado medidas para garantizar la confidencialidad y privacidad de los datos utilizados, evitando cualquier tipo de sesgo o discriminación injusta en el tratamiento de la información.

En Madrid a 03 de septiembre de 2025.

Anyelo David Barrios Rodríguez.

Contenido

Índice de imágenes

Índice de tablas

1. INTRODUCCIÓN

La necesidad de avanzar hacia fuentes de energía sin emisiones de carbono cobra mayor relevancia frente a los compromisos climáticos globales y las concentraciones históricas de CO_2 registradas. En este contexto, el gas natural mantiene su importancia como un recurso de transición clave gracias a su menor costo y su huella de carbono más reducida frente a otros hidrocarburos (Al Ghafri et al., 2021).

Por consiguiente, las plantas de gasificación de gas natural licuado (GNL) aumentan cada día más su relevancia. Estas, generalmente están compuestas por tanques de almacenamiento de GNL, bombas, compresores y tuberías de gas. En la Imagen 1 se puede ver el esquema del proceso de gasificación del GNL (Jang et al., 2011).

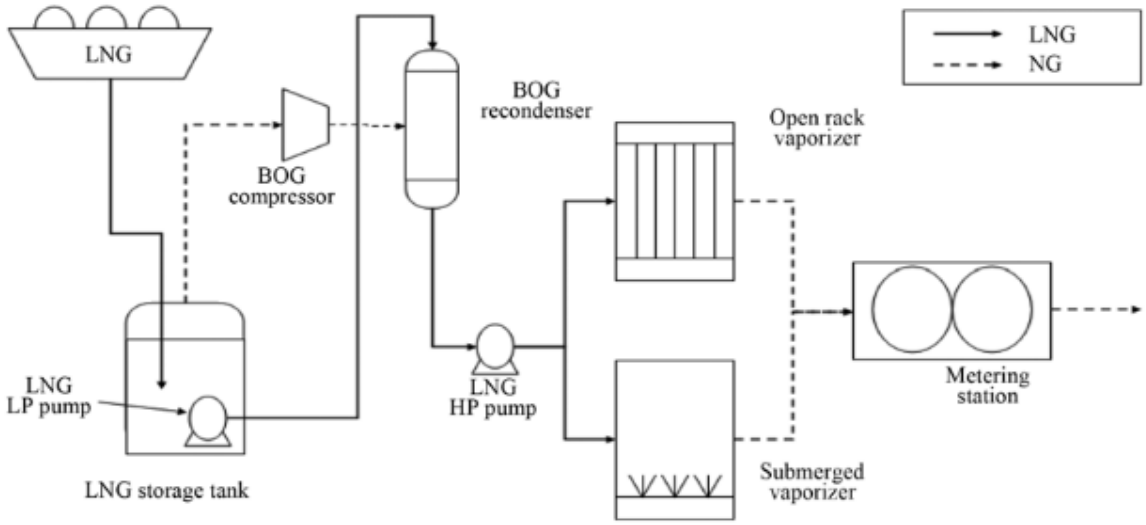


Imagen 1. Proceso típico de recepción y gasificación de GNL.
Autor: Jang et al. (2011). Dynamic simulation and optimization of the operation of boil-off gas compressors in a liquefied natural gas gasification plant

En las plantas de regasificación, el GNL es descargado desde los buques transportadores y almacenado en los tanques en estado líquido criogénico con una temperatura cercana a los -160°C y una presión apenas superior a la atmosférica. Como consecuencia del calor absorbido del ambiente, una fracción del GNL se transforma continuamente en gas de ebullición o *boil-off gas* (BOG) por sus siglas en inglés, que debe ser retirado mediante compresores para mantener la presión en niveles seguros. Si la cantidad de BOG generado supera lo previsto, se generan riesgos de seguridad por el incremento de la presión en los tanques (Jang et al., 2011), de tal manera que si la cantidad de BOG supera la capacidad de los compresores, debe ser quemado en las antorchas. Por ello, las plantas de regasificación deben controlar cuidadosamente la generación y el tratamiento del BOG y regular la presión en los tanques con el fin de mitigar los riesgos de seguridad y evitar quemarlo en las antorchas, dado que constituye un recurso muy preciado (Hidalgo-Mompeán et al., 2021).

En este escenario, la disponibilidad de los compresores de BOG se convierte en un aspecto critico a gestionar, implementando estrategias adecuadas de mantenimiento que aseguren su operación continua, minimicen los tiempos de inactividad, optimicen

los costos del ciclo de vida y garanticen tanto la eficiencia como la seguridad operacional de la planta. Desde este punto de vista, es importante integrar enfoques de Mantenimiento Centrado en Confiabilidad (RCM) o Mantenimiento Basado en Condición (CBM), metodologías que permiten anticipar fallas a partir del análisis de datos operativos (DCS) y de mantenimiento (CMMS) y priorizar intervenciones en función del riesgo reduciendo costes y mejorando la confiabilidad del sistema (Hwang et al., 2018).

Los sistemas de control distribuido (DCS) registran datos operativos de los equipos de la planta y, en muchos casos, llevan almacenando información desde mucho antes de que surgieran herramientas específicas de mantenimiento. Por su parte, los sistemas de gestión de mantenimiento computarizado (CMMS), cuando se aplican correctamente, recogen el historial de intervenciones realizadas y de fallos registrados en cada equipo. Al cruzar los datos provenientes del DCS con los del CMMS, es posible analizar el desempeño de un equipo en particular y determinar cómo ha estado funcionando cuando ha presentado un modo particular de falla como una fuga de aceite (Hidalgo-Mompeán et al., 2021).

El desarrollo de una estrategia de CBM requiere comprender y aplicar de manera secuencial las etapas de detección, diagnóstico y pronostico. Así, un algoritmo de pronóstico encargado de estimar la evolución temporal de un modo de falla o su comportamiento futuro, solo puede activarse tras la acción de un algoritmo previo de detección o diagnostico que identifique con alta certeza la falla o la transición del sistema de un estado normal a uno anómalo (Guillén et al., 2016). Por lo tanto, un paso importante en la implementación de una estrategia de CBM basado en datos de provenientes de DCS y CMMS es la construcción de un modelo capaz de detectar los modos de fallos con una buena probabilidad de acierto.

En los conjuntos de datos para CBM, uno de los principales retos es la escasez de registros de fallas frente a la abundancia de datos de operación normal. Dado que las averías en equipos, especialmente en activos críticos y de alto valor como los compresores de BOG, ocurren con poca frecuencia, surge un desbalance de clases en el que la categoría “falla” queda ampliamente superada por la de “normal”. Esta situación afecta negativamente el rendimiento de los modelos de machine learning, generando sobreajuste, métricas de desempeño poco fiables y un incremento de falsos negativos. En consecuencia, es necesario aplicar técnicas de tratamiento de datos o adaptar los algoritmos para mitigar este problema (Mahale et al., 2025).

El desbalance de clases puede abordarse desde el nivel del algoritmo o desde el nivel de los datos. En el primer caso, se ajusta el entrenamiento de los modelos penalizando con un costo mayor la clasificación errónea de la clase minoritaria. En el segundo, se busca equilibrar el dataset modificando la proporción de clases, ya sea aumentando artificialmente los casos de la clase minoritaria mediante técnicas como *Synthetic Minority Oversampling Technique* (SMOTE), o reduciendo de forma aleatoria la cantidad de registros de la clase mayoritaria.(Mahale et al., 2025).

En los años recientes, diversos trabajos han explorado la construcción de modelos de detección de fallas mediante enfoques semejantes, evidenciando la efectividad de las técnicas de machine learning para enfrentar este tipo de problemática de

mantenimiento, por ejemplo, Hidalgo-Mompeán, Goméz, Cerruela-García y Crespo se enfocaron en el desarrollo de un modelo de detección de fallos en un compresor de BOG con datos de DCS y CMMS como parte de una estrategia de CBM, afrontando como principal problema la alta dimensionalidad y utilizando algoritmos como Random Forest, KNN y Naive Bayes manejando métricas como F1-Score y AUC-ROC para evitar las interpretaciones confusas del *Accuracy*. Luego de aplicar múltiples métodos de selección encontraron de manera exitosa modelos de detección de fallos con solo cinco variables predictoras con niveles de F1-Score de 0,966 utilizando Random Forest y KNN (Hidalgo-Mompeán et al., 2021).

En otro estudio, Mahale y su equipo se centraron en un modelo de detección de fallos para la industria automotriz como parte de una estrategia de mantenimiento predictivo enfrentando como principal reto el problema del desbalance de clases. El cual fue afrontado aplicando diversos enfoques como SMOTE para balancear las clases, algoritmos sensibles al costo y algoritmos ensamblados. De esta manera, utilizando algoritmos como Random Forest, Regresión Logística, Xgboost y SVM y manejando métricas como F1-Score y AUC-ROC para evitar las interpretaciones confusas del *Accuracy,* lograron resultados exitosos después de aplicar SMOTE de F1-score de 0,9954 en Random Forest y métricas perfectas con algoritmos ensamblados (Mahale et al., 2025).

Por otra parte, el trabajo de Orrú presenta un modelo de predicción de fallos aplicado a una bomba centrifuga en la refinería SARLUX (Italia), con el objetivo de anticipar averías con una semana de antelación y favorecer la planificación del mantenimiento. Se emplearon datos históricos de sensores IoT entre 2010 y 2014, abordando el marcado desbalance de clases mediante la técnica de SMOTE, junto con normalización e imputación de valores faltantes. Los algoritmos SVM y Multilayer Perceptron alcanzaron precisiones superiores al 98% (Orrù et al., 2020).

Teniendo en cuenta todo lo anterior, en este trabajo se desarrolla un caso práctico y real de aplicación de metodologías de ciencia de datos para la gestión del mantenimiento en plantas industriales, construyendo un modelo de detección de fallos por fugas de aceite en un compresor de BOG a partir de datos industriales reales extraídos en bruto del sistema de DCS y CMMS de una planta de regasificación de GNL. En este marco, se adelanta un trabajo previo exhaustivo de depuración y limpieza de los datos incluyendo la eliminación de datos erróneos, obsoletos, anómalos o con errores de lectura, así como el tratamiento de datos atípicos y construcción de nuevas variables para lograr un dataset válido para modelización. Así mismo, se realiza un extenso análisis descriptivo para identificar las correlaciones entre las variables predictoras y su poder discriminativo para la variable objetivo.

Además, en el transcurso de este estudio, se enfrenta el problema del desbalance de clases y para ello, se contrastan modelos construidos con el desbalance original y modelos construidos con un dataset balanceado a través de la técnica SMOTE.

De esta manera, se consideran diferentes métodos de selección de variables y se investigan siete algoritmos de machine learning para seleccionar un modelo ganador que sea útil para la implementación de un programa de CBM.

1.1 Objetivos

El objetivo principal de este trabajo es desarrollar un modelo de clasificación, basado en algoritmos de machine learning, que permita detectar el modo de fallo por fuga de aceite en un compresor de BOG. El modelo se construirá a partir de los datos industriales capturados en la planta de una empresa dedicada al transporte, regasificación y almacenamiento de gas natural.

Por lo tanto, se plantean los siguientes objetivos específicos:

a) Construir un dataset de trabajo válido para la modelización mediante la fusión de los datos operativos capturados del compresor con los datos del sistema de gestión de mantenimiento.
b) Depurar y limpiar los datos a través del tratamiento de datos erróneos, anómalos, atípicos, incompletos o de cualquier índole que puedan generar ruido para la investigación.
c) Realizar un análisis descriptivo que permita entender la información incluida en los datos e identificar la relación entre las variables predictoras y la variable objetivo.
d) Crear un dataset balanceado a través de la técnica de SMOTE que permita entrenar modelos de machine learning evitando el problema del desbalance de clases.
e) Escoger un set de variables apropiado para el entrenamiento de los modelos de clasificación mediante la aplicación de distintos métodos de selección de variables.
f) Entrenar diferentes modelos de clasificación con distintos algoritmos tanto para el dataset balanceado como para el desbalanceado.
g) Evaluar y comparar los modelos de clasificación entrenados a través de las métricas de rendimiento apropiadas con el fin de seleccionar el mejor modelo.

2. METODOLOGÍA UTILIZADA

2.1 Metodología general del proyecto

Este trabajo de fin de máster se estructura siguiendo la metodología CRISP-DM (Cross Industry Standard Process for Data Mining), considerado el estándar más utilizado en proyectos de ciencia de datos. Este marco metodológico incluye seis fases que iteran entre ellas hasta concluir el proyecto y son: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue (Chapman et al., 2000).

- Comprensión del negocio *(Business Understanding)*

En esta fase se busca comprender los objetivos y necesidades teniendo en cuenta la perspectiva del negocio y poder transformarlo a un problema de ciencia de datos. En el contexto de este trabajo, al disponer de información industrial real, se reconoce la fuga aceite como un fallo importante que puede comprometer la seguridad de la operación y la eficiencia del sistema, por lo que se define como objetivo identificar un modelo de machine learning de clasificación que permita detectar este tipo de fallos y las variables que más inciden en ello.

- Comprensión de los datos *(Data Understanding)*

Lo que se busca en esta etapa es recopilar, explorar y verificar la calidad de los datos disponibles. Para el compresor objeto de análisis, los datos provienen de sensores instalados en el equipo y corresponden a variables operativas como temperaturas y presiones principalmente. También provienen datos del sistema de gestión de mantenimiento con información de las fallas presentadas en el equipo. Al trabajar con información práctica de uso industrial, esta fase ha requerido de un importante esfuerzo en la medida en que se hace necesario realizar diferentes aproximaciones para entender la naturaleza de los datos, la información de contienen, los momentos en que fueron tomados los datos, el propósito actual de su captura y demás aspectos relevantes. Gracias a esto se logra identificar que los registros de fuga de aceite son escasos en comparación con los registros de operación normal, por lo que se enfrenta un problema de desbalance de clases que será abordado en la modelización.

- Preparación de los datos *(Data Preparation)*

Consiste en limpiar, transformar y seleccionar los datos que serán usados en el modelado. En el desarrollo de este trabajo, este es uno de los puntos de mayor esfuerzo por tratarse de datos en bruto, por lo cual se adelanta un trabajo exhaustivo de depuración y limpieza de los datos incluyendo cambios en la nomenclatura y tipología de las variables, la eliminación de datos erróneos, obsoletos o con errores de lectura, así como el tratamiento de datos atípicos, construcción de nuevas variables y definición de la variable objetivo a través de la fusión de dos fuentes de datos para lograr un dataset válido para modelización.

- Modelización *(Modeling)*

En la modelización se entrenan algoritmos de machine learning para la solución del problema, se ajustan los hiperparámetros y se hace uso de técnicas de validación para la evaluar el rendimiento.

Para el desarrollo del TFM se implementaron distintos modelos de clasificación: Regresión Logística, Redes Neuronales, Bagging, Random Forest, Gradient Boosting y SVM con diferentes funciones de Kernel y debido al desbalance pronunciado de las clases, se aplicó la técnica SMOTE para crear nuevos datos sintéticos para la clase minoritaria y mejorar la capacidad de detectar las fugas.

Los conjuntos de variables utilizados en cada modelo fueron seleccionados a través de métodos de selección como Stepwise con criterio AIC/BIC, Boruta, RFE y MXM.

- Evaluación *(Evaluation)*

Durante esta etapa se interpretan los resultados obtenidos en la modelización y se evalúa la calidad de los modelos de acuerdo con las métricas de rendimiento seleccionadas. Se emplearon F1-Score, AUC, recall y precisión, priorizando la primera y se realizó comparación entre los datos originales y los datos balanceados a través de SMOTE.

- Despliegue (Deployment)

La fase final en donde se integra el modelo seleccionado a la operación. Debido a que este es un ejercicio académico, la interpretación de todos los resultados se compartirán con la empresa dueña de equipo para que tomen las decisiones sobre la integración de este modelo a sus actividades de mantenimiento.

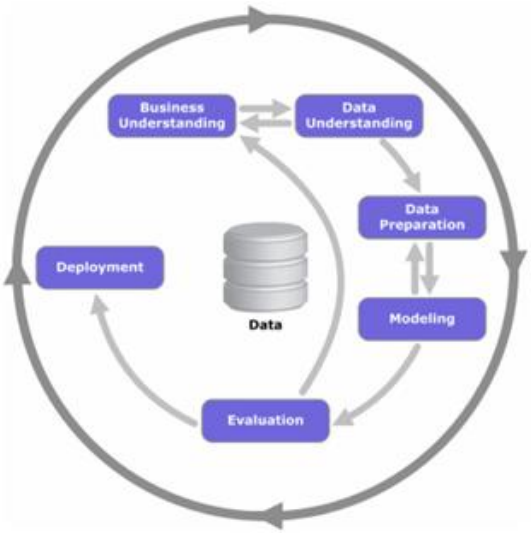


Imagen 2. Etapas de proyectos de ciencia de datos de acuerdo con metodología CRISP-DM. Autor: Chapman et al. (2000), Metodología Crisp-DM Step by step data mining guide.

2.2 Machine Learning

El aprendizaje automático (machine learning) se entiende como la disciplina que busca mejorar el desempeño de algún proceso/sistema a partir de la experiencia, y lo hace a través de métodos computacionales que construyen modelos a partir de los datos disponibles. Según Zhou, en los sistemas informáticos, la experiencia existe en formas de datos y la tarea principal del aprendizaje automático es desarrollar algoritmos de aprendizaje que construyan modelos a partir de esos datos (Zhou, 2021).

En este trabajo, el uso de machine learning permite desarrollar modelos de clasificación que sean capaces de detectar fugas de aceite en los compresores de BOG y se tomará la decisión de cual se ajusta mejor partiendo de métricas como el F1-score y el AUC.

2.2.1 Regresión logística

La Regresión Logística es un modelo estadístico, utilizado en problemas de clasificación binaria que modela la probabilidad de que ocurra un evento, en este caso la presencia de una fuga de aceite, en función de un conjunto de variables explicativas. Matemáticamente, la probabilidad condicional de que se produzca el evento Y=1 dado un conjunto de variables independientes $x_1, x_2, \dots, x_k$ se expresa como:

$$\mathrm{P}(\mathrm{Y}=1|x_1, x_2, \dots, x_m) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_m x_m)}}$$

Donde β_0 es el intercepto del modelo y $\beta_1, \dots, \beta_m$ son los coeficientes que cuantifican el efecto de cada variable sobre la probabilidad del evento (fuga).

Lo que implica que:

$$Log(\frac{p_1}{1-p_1}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_m x_m$$

El término de la izquierda corresponde al nombre de logit y es el algoritmo de la razón de las probabilidades u odds. Esta transformación linealiza la relación entre las variables independientes y la variable de respuesta, lo que permite interpretar cada coeficiente como el efecto de un cambio unitario en la variable x_m sobre el algoritmo de la odds (Kleinbaum & Klein, 2002).

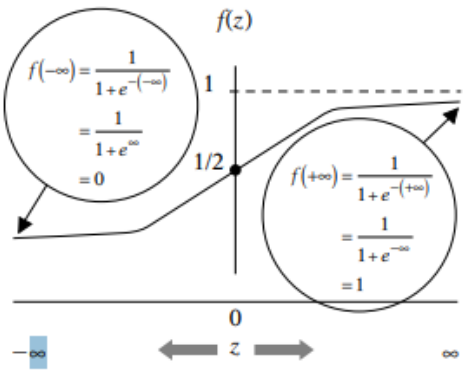


Imagen 3. Representación gráfica de la regresión logística.
Autor: Kleinbaum & Klein. (2002), Logistic Regression: A Self-Learning Text.

2.2.2 Redes neuronales

Las Redes Neuronales son un modelo que se inspira en el funcionamiento del cerebro humano. Está compuesto por nodos o “neuronas” que se organizan en capas y se conectan entre sí a través de pesos. Cada nodo recibe valores de entrada que se combinan mediante una suma ponderada y lo transforma en una función conocida como activación, generando una salida que se trasmite a las siguientes capas.

El proceso de ajuste de los pesos se llama entrenamiento y su objetivo es minimizar la función de perdida que mide la diferencia predicha por la red y la salida real observada, es por ellos que las redes son capaces de aprender patrones complejos y no lineales, por lo que supera limitaciones de otros modelos estadísticos (Aggarwal, 2023).

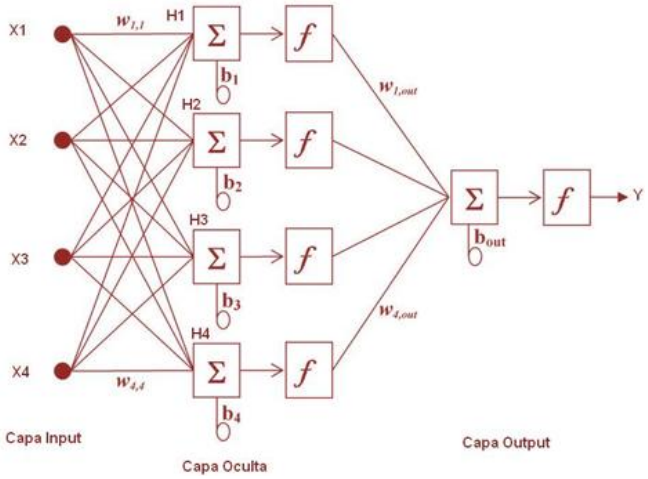


Imagen 4. Funcionamiento de la red neuronal.
Autor: Portela. (2025), Presentación en clases.

2.2.3 Bagging (*Bootstrap aggregation*)

Es una técnica de ensamblado y su objetivo es mejorar el rendimiento y la estabilidad de los modelos. Su funcionamiento se basa en generar varios subconjuntos del dataset original mediante muestreo con reemplazo (bootstrapping), entrenar cada uno de ellos y combinar los resultados de todos los modelos entrenados (aggregation).

Bagging aprovecha la diversidad de los modelos entrenados sobre los diferentes subconjuntos de datos para reducir la varianza y obtener un modelo final más robusto que cualquiera de los modelos de manera individual (Kumar & Jain, 2020).

Este modelo funciona bien cuando existen muchas variables con relación débil pero estable con la variable dependiente, cuando no hay relaciones o separaciones lineales, cuando hay interacciones ocultas o muchas variables categóricas (Portela, 2025).

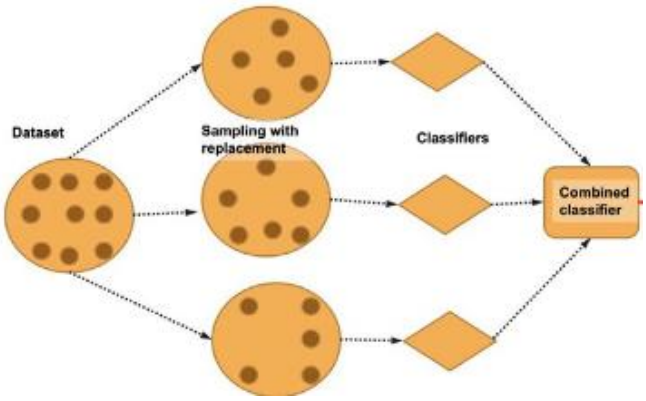


Imagen 5. Funcionamiento de bagging para problemas de clasificación.
Autor: Portela. (2025), Presentación en clases.

2.2.4 Random Forest

Este modelo de machine Learning tiene como idea principal la construcción de árboles y la combinación de sus resultados para obtener un modelo más robusto y preciso que modelos individuales. Random Forest surge como una extensión de los métodos Bagging aplicados a árboles, incorporando un paso adicional de selección aleatoria de variables en cada división del árbol

Su funcionamiento se resume en generar múltiples Bootstrap del dataset original, construir un árbol de decisión para cada muestra, considerando en cada nodo solo un subconjunto aleatorio de variables predictoras y agregar finalmente los resultados de todos los árboles mediante voto mayoritario para los problemas de clasificación o promedio para los problemas de regresión.

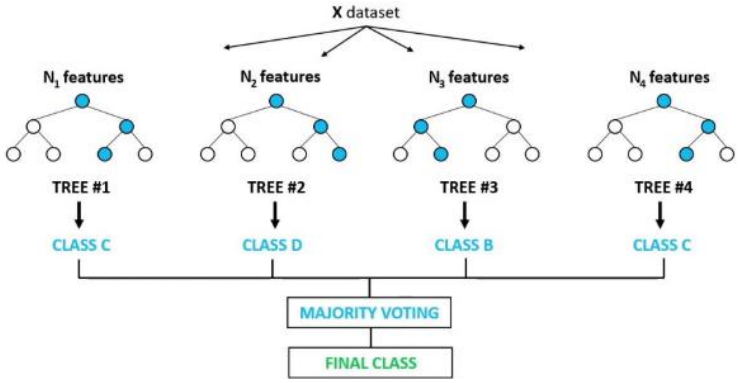


Imagen 6. Funcionamiento de random forest para problemas de clasificación.
Autor: Portela. (2025), Presentación en clases.

2.2.5 Gradient Boosting

El Gradient Boosting es una técnica de ensamblado que construye un modelo complejo por medio de la combinación de muchos modelos simples y su idea central es que cada nuevo árbol se entrene para mejorar los errores del anterior.

El modelo entrena un árbol inicial, calcula los errores cometidos y a partir de ellos entrena un nuevo árbol para mejorar la clasificación de los casos mal predichos. De esta forma, el modelo final se obtiene como la combinación secuencial de todos los árboles, donde cada uno aporta un ajuste adicional (Kumar & Jain, 2020).

El uso de Gradient Boosting da como resultado un clasificador más robusto y preciso, capaz de capturar patrones complejos en los datos y con buen desempeño en problemas de clasificación binaria como de la detección de fuga de aceite.

2.2.6 Support Vector Machines

Los SVM tienen como principio fundamental encontrar un hiperplano de separación óptimo que divida las observaciones de dos clases distintas. Este hiperplano se elige de tal forma que maximice la distancia o margen entre el propio hiperplano y las observaciones más cercanas de cada clase, llamadas vectores de soporte. Un mayor margen involucra una mejor capacidad de generalización.

De las principales fortalezas de este tipo de modelo es su capacidad para tratar problemas donde los datos no son linealmente separables, por lo que se emplea el uso del truco de Kernel, que proyecta los datos a un espacio de características de mayor dimensión en el cual es posible separarlos (Campbell & Ying, 2011).

Los kernels más utilizados son:

- Kernel lineal: Cuando las clases son aproximadamente separables en el espacio original.
- Kernel polinómico: Permite la captura de relaciones no lineales de diferente grado entre las variables.
- Kernel gaussiano (RBF): Los datos son proyectados en un espacio de dimensión infinita, lo que genera fronteras de decisión muy flexibles para problemas complejos.

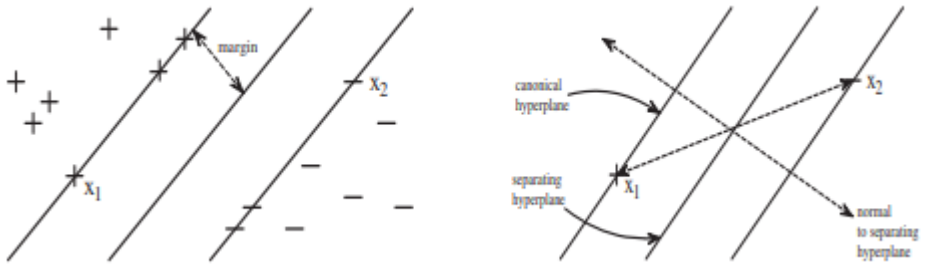


Imagen 7. Gráfico de número de variables vs. F1 en dataset.
Autor: Campbell & Ying. (2011), Learning with Support Vector Machines.

2.3 Métricas de evaluación

Para evaluar la calidad de los modelos de clasificación se emplea principalmente la exactitud (accuracy), sin embargo, cuando ocurre un desbalance de clases, el uso exclusivo de esta métrica puede ser engañoso. A continuación, las principales métricas empleadas en este tipo de algoritmos (Ye & Wang, 2023).

- Exactitud (accuracy): Corresponde a la proporción de predicciones correctas sobre el total de observaciones.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Donde:

TP = Verdaderos positivos
TN = Verdaderos negativos
FP = Falsos positivos
FN = Falsos negativos

- Precisión (Precision): Mide la proporción de predicciones positivas que realmente corresponden a la clase positiva. Es muy útil cuando el costo de los falsos positivos es elevado.

$$Precision = \frac{TP}{TP + FP}$$

- Recall (tasa de verdaderos positivos): Esta métrica corresponde a la proporción de verdaderos positivos correctamente identificados respecto al total de casos positivos reales. Esta métrica es un buen índice cuando es importante detectar la mayor cantidad de posible de positivos, incluso a costa de cometer más falsos positivos.

$$Recall = \frac{TP}{TP + FN}$$

- F1-Score: Calcula la media armónica entre precisión y recall y busca el equilibrio entre ambas métricas. Es usualmente usada para problemas con desbalance de clases ya que permite determinar la capacidad del modelo para detectar correctamente los positivos, penalizando los falsos positivos y los falsos negativos.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

2.4 SMOTE (Synthetic Minority Oversampling Technique)

El desbalance de clases resulta ser un problema y se da cuando una de las clases en un conjunto de datos está muy poco representada en comparación con la clase mayoritaria, por lo que los modelos de clasificación tienden a estar sesgados hacia la clase más frecuente.

Una de las técnicas más empleadas es SMOTE, el cual genera nuevas muestras sintéticas mediante la interpolación entre una observación minoritaria y uno de sus *k* vecinos más cercanos. Lo anterior es lo que garantiza que se incluyan datos preservando la distribución de la clase minoritaria y de esta forma mejorar la capacidad del modelo de aprender patrones (Rodrigues et al., 2019).

Para mitigar el problema de desbalance se usan dos enfoques principales:

- Oversampling. Aumenta la representación de la clase minoritaria sin duplicar datos idénticos.
- Undersampling. Reduce el número de observaciones de la clase mayoritaria, eliminando parte de ellas de manera aleatoria o mediante criterios específicos. Con esto se busca que la clase dominante no ejerza demasiada influencia en el entrenamiento del modelo.

3. FUENTE DE DATOS

Para el desarrollo del presente estudio se manejan dos fuentes de datos relacionadas con un compresor de BOG de una empresa de infraestructuras energéticas, especializada en el transporte, regasificación y almacenamiento de gas natural.

El primer dataset, llamado “Reporte de Incidencias”, proviene del sistema de gestión del mantenimiento de la planta CMMS, en el cual los operadores registran las incidencias/fallas y alarmas presentadas en el compresor.

El segundo dataset, llamado “Registro de Variables de Operación”, se genera del sistema de control distribuido del compresor DCS, el cual captura la información directamente del equipo a través de sensores que realizan seguimiento a las variables de operación más relevantes.

Es importante resaltar que estos datasets se reciben en bruto y sin ningún tipo de limpieza previa, por lo que el trabajo de depuración de los datos es exhaustivo.

A continuación, se realiza una exploración descriptiva, limpieza inicial y depuración de las dos fuentes de datos utilizadas.

3.1 Dataset “Reporte de Incidencias”

Este dataset está conformado por 88 registros de incidencias presentadas durante el periodo de estudio del 01 de enero del 2018 hasta el 24 de noviembre del 2024. De cada incidencia se registran 14 variables con las siguientes características:

Variable	Descripción
Aviso	Número de identificación del aviso generado por la máquina
Descripción	Descripción del fallo que ocasiona el aviso
Inicio deseado	Fecha de inicio deseado de la operación después del fallo
Hora inic.des.	Hora de inicio deseado de la operación después del fallo
Equipo	Número de identificación del equipo (o del subequipo) en cual se ha generado el fallo
Denominación	Nombre del equipo (o del subequipo) en cual se ha generado el fallo
Inicio avería	Fecha de inicio del fallo
Hora in.avería	Hora de inicio del fallo
Tiempo hasta el siguiente fallo (D)	Número de días entre fallos
Fin de avería	Fecha de fin del fallo
Hora fin avería	Hora de fin del fallo
Duración parada	Número de horas transcurridas entre el inicio y fin del fallo
ANOMALÍA	Descripción resumida del fallo
ANOMALÍA_AGR	Descripción resumida agrupada del fallo

Tabla 1. Descripción de las variables incluidas en el dataset reporte de incidencias

Con este dataset, es posible identificar los diferentes tipos de fallos que se presentan en el compresor, cuando se presentan, la duración de estos y cuáles son los equipos y subequipos principalmente afectados.

3.1.1 Análisis descriptivo de las Incidencias

En la siguiente tabla resumen, se puede observar cuales son los tipos de fallos que se registraron en el compresor durante el periodo de estudio, cuantas veces se presentó cada uno y la duración acumulada de las paradas ocasionadas por cada tipo de fallo:

Anomalía	Número de Incidencias Reportadas	Duración acumulada de paradas
Fuga aceite	24	3.779
Fallo en válvulas	11	6.606
Falsa alarma	8	1.447
La máquina se para	7	200
Baja la presión gas	5	225
Fusible roto	5	5.472
Caída de banda 1ª Etapa	5	1.128
Caída de banda 2ª Etapa	3	72
Filtro sucio	3	2.078
Fuga gas	2	0
No arranca el compresor	2	16
Sube la temperatura aceite	2	384
Sube la presión aceite	1	24
Baja la presión aceite	1	0
Manómetro presión roto	1	72
Carter roto	1	231
Rejilla rota	1	648
Sube la presión gas	1	24
Error en posición del cilindro	1	192
Transmisor roto	1	96
Fallo en bornas	1	120
Bloqueo	1	3
Fallo en radiador	1	2.520
Total general	**88**	**25.335**

Tabla 2. Resumen de los fallos presentados por el compresor.

Inicialmente, se identifica que el tipo de fallo que más veces se presentó fue “Fuga aceite”, pero el fallo que más horas de parada ocasionó fue “Fallo en válvulas”.

Para un mejor análisis y comparación, se logra identificar que existen fallos que están relacionados entre sí de acuerdo con la naturaleza del fallo o de acuerdo con el sistema del compresor que se ve afectado por el fallo. Por tal motivo se toma la decisión de agrupar los fallos de acuerdo con las siguientes categorías:

- **Fallo Lubricación:** fallos en donde se ve afectado el aceite de lubricación o los componentes del sistema de lubricación. Por ejemplo, los fallos reportados como ‘fuga de aceite’, ‘Baja presión de aceite’, ´’Sube la presión de aceite’, ‘Sube temperatura de aceite’, ‘Carter roto’, ‘Fallo radiador’ (del sistema de ventilación de aceite) y ‘filtro de aceite sucio’.

- **Fallo en válvulas:** todos aquellos fallos reportados en las válvulas de aspiración o impulsión del compresor.
- **Error Lectura**: fallos relacionados con errores de lectura, falsas alarmas o dispositivos de lectura rotos.
- **Caída Banda:** fallos reportados como paradas de máquina por caída de bandas, 'Caída de banda 1ª Etapa', 'Caída de banda 2ª Etapa'.
- **Maquina parada:** paradas de maquina reportadas sin identificar el motivo principal.
- **Falla presión Gas:** fallos en donde se ve afectada la presión del gas reportados como 'Baja la presión gas' y 'Sube la presión gas'.
- **Bloqueo**: Fallo reportado como 'Bloqueo'
- **Otros:** Los fallos que no se ajustaron a los criterios de las anteriores categorías.

La siguiente tabla muestra para cada categoría de fallo, el número de incidencias reportadas y la duración total de las paradas que se ocasionaron:

Anomalía	Número de Incidencias Reportadas	Duración acumulada de paradas
Fallo lubricación	32	8.703
Error lectura	16	7.399
Fallo en válvulas	13	6.606
Caída banda	10	1.248
Maquina parada	7	168
Fallo presión gas	6	249
Otros	3	960
Bloqueo	1	3
Total	88	25.335

Tabla 3 Resumen de las categorías de fallos.

Por consiguiente, se concluye que la categoría con mayor participación en la generación de incidencias y paradas son los fallos relacionados con el aceite y el sistema de lubricación del compresor.

En las siguientes gráficas de torta se puede ver la participación de los diferentes tipos de fallos en el total de la categoría 'FALLO LUBRICACIÓN'.

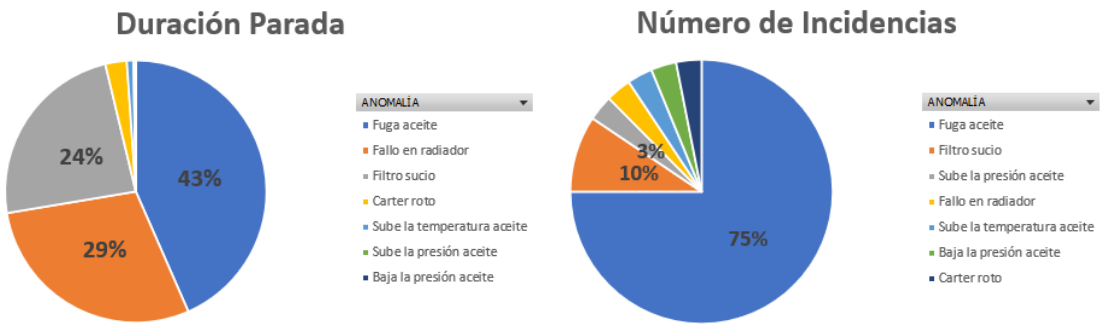


Imagen 8. Gráficos de paradas y número de incidencias.

Por lo tanto, es de notar que dentro de la categoría de 'FALLO LUBRICACIÓN', la anomalía que mayor participación tiene, tanto en el número de incidencias reportadas como en la duración de las paradas, es ´fuga de aceite'.

La participación de los subequipos en la generación de incidencias por fuga de aceite se puede ver en la siguiente gráfica de torta:

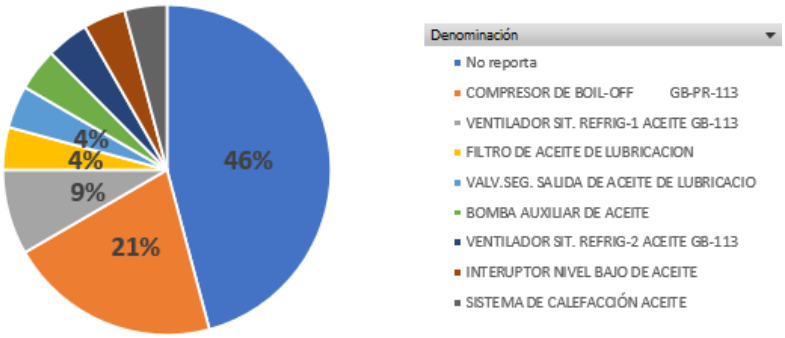


Imagen 9. Número de incidencias por sub-equipo.

De esta manera, el 46% de las incidencias no reporta el subequipo en el cual fue identificado la fuga, mientras que el 21% lo reporta en el compresor directamente, seguido del ventilador. Sin embargo, se puede afirmar que las fugas de aceite afectan a los distintos componentes del compresor.

En conclusión, se puede indicar que los fallos relacionados al sistema de lubricación del compresor son los que han afectado en mayor medida al funcionamiento del compresor.

En este sentido, es la fuga de aceite el tipo de fallo que mayor ha afectado este sistema con el 27% de las incidencias reportadas y con 15% de las horas de paradas generadas por incidencias.

3.2 Dataset "Registro de Variables de Operación"

Este dataset está conformado por 60.473 observaciones tomadas cada hora desde el 01 de enero del 2018 hasta 24 de noviembre de 2024. Cada observación contiene información obtenida del Sistema de Control Distribuido de la planta de regasificación de GNL (DCS), capturada directamente desde el compresor a través de los sensores que están instalados en el equipo. Esta información se registra en las siguientes 65 variables:

Variable	Tipo de dato	Unidad de Medida	Descripción
FECHA_HORA	Fecha	N/A	Fecha y hora de registro de las diferentes señales.
Analisis_aceite_puntual	String	N/A	Señal Obsoleta
Cambio_bandas_cilindro_1_etapa_1_puntual	String	N/A	Informa si el compresor se encuentra en un mantenimiento por cambio de banda o no
Cambio_bandas_cilindro_1_etapa_1_total	String	N/A	Señal obsoleta
Cambio_bandas_cilindro_1_etapa_2_puntual	String	N/A	Informa si el compresor se encuentra en un mantenimiento por cambio de banda o no
Cambio_bandas_cilindro_1_etapa_2_total	String	N/A	Señal obsoleta

Cambio_bandas_cilindro_2_etapa_1_puntual	String	N/A	Informa si el compresor se encuentra en un mantenimiento por cambio de banda o no
Cambio_bandas_cilindro_2_etapa_1_total	String	N/A	Señal obsoleta
Cambio_bandas_cilindro_2_etapa_2_puntual	String	N/A	Informa si el compresor se encuentra en un mantenimiento por cambio de banda o no
Cambio_bandas_cilindro_2_etapa_2_total	String	N/A	Señal obsoleta
Carga_trabajo_puntual	Float	%	% de carga a la que está trabajando el compresor. Los modos de trabajos son: 25%, 50%, 75% o 100%
Contador_grandes_mantenimientos_puntual	Int	N/A	Número de grandes mantenimientos realizados al compresor. Señal Obsoleta
Estado_puntual	String	N/A	Estado de funcionamiento del compresor: MANT. CORRECTIVO - El compresor se encuentra en mantenimiento por una acción correctiva MANT. PREVENTIVO - El compresor se encuentra en mantenimiento por una acción preventiva MARCHA - El compresor se encuentra en funcionamiento PARO DISPONIBLE - El compresor se encuentra sin funcionar, pero con posibilidad de entrar en funcionamiento en cualquier momento PARO INCIDENCIA - El compresor se encuentra parado por una incidencia detectada, que no le permite arrancar
Horas_desde_ultimo_cambio_banda_cilindro_1_etapa_1	Float	Horas	Horas de funcionamiento del compresor desde el último cambio de banda del cilindro 1 de la etapa 1
Horas_desde_ultimo_cambio_banda_cilindro_1_etapa_2	Float	Horas	Horas de funcionamiento del compresor desde el último cambio de banda del cilindro 1 de la etapa 2
Horas_desde_ultimo_cambio_banda_cilindro_2_etapa_1	Float	Horas	Horas de funcionamiento del compresor desde el último cambio de banda del cilindro 2 de la etapa 1
Horas_desde_ultimo_cambio_banda_cilindro_2_etapa_2	Float	Horas	Horas de funcionamiento del compresor desde el último cambio de banda del cilindro 2 de la etapa 2
Horas_desde_ultimo_gran_mtto	Float	Horas	Horas de funcionamiento del compresor desde su puesta en marcha
Horas_totales_operacion	Float	Horas	Horas de funcionamiento del compresor desde el último gran mantenimiento
Intensidad_puntual	Float	A	Intensidad de funcionamiento del motor del compresor
Numero_arranques_puntual	Int	N/A	Señal sin funcionar: Debería medir el número de arranques que lleva el compresor

Potencia_puntual	Float	Kw	Potencia de funcionamiento del motor del compresor
Presion_aspiracion_puntual	Float	bar	Presión de aspiración del compresor. Hay 3 presostatos diferentes para medir la presión de aspiración
Presion_aspiracion_puntual	Float	bar	Presión de aspiración del compresor. Hay 3 presostatos diferentes para medir la presión de aspiración
Presion_aspiracion_puntual	Float	bar	Presión de aspiración del compresor. Hay 3 presostatos diferentes para medir la presión de aspiración
Presion_diferencial_filtro_puntual	Float	bar	Presión diferencial del filtro del compresor.
Presion_impulsion_puntual	Float	bar	Presión de impulsión del compresor. Hay 4 presostatos diferentes para medir la presión de impulsión
Presion_impulsion_puntual	Float	bar	Presión de impulsión del compresor. Hay 4 presostatos diferentes para medir la presión de impulsión
Presion_impulsion_puntual	Float	bar	Presión de impulsión del compresor. Hay 4 presostatos diferentes para medir la presión de impulsión
Presion_impulsion_puntual	Float	bar	Presión de impulsión del compresor. Hay 4 presostatos diferentes para medir la presión de impulsión
Presion_interetapa_puntual	Float	bar	Presión interetapa (entre la etapa 1 y la etapa 2) del compresor. Hay 3 presostatos diferentes para medir la presión interetapa
Presion_interetapa_puntual	Float	bar	Presión interetapa (entre la etapa 1 y la etapa 2) del compresor. Hay 3 presostatos diferentes para medir la presión interetapa
Presion_interetapa_puntual	Float	bar	Presión interetapa (entre la etapa 1 y la etapa 2) del compresor. Hay 3 presostatos diferentes para medir la presión interetapa
Prioridad_arranque_puntual	Int	N/A	Indica la prioridad de arranque según las necesidades de operación de los compresores. Hay 4 compresores y este siempre tiene la prioridad 1
Temperatura_aspiracion_puntual	Float	ºC	Temperatura de aspiración del compresor
Temperatura_cojinete_motor_puntual	Float	ºC	Temperatura de los cojinetes del motor del compresor
Temperatura_cojinetes_compresor_zona_1_puntual	Float	ºC	Temperatura de los cojinetes del compresor en la zona 1
Temperatura_cojinetes_compresor_zona_2_puntual	Float	ºC	Temperatura de los cojinetes del compresor en la zona 2
Temperatura_cojinetes_compresor_zona_3_puntual	Float	ºC	Temperatura de los cojinetes del compresor en la zona 3

Temperatura_cojinetes_compresor_zona_4_puntual	Float	ºC	Temperatura de los cojinetes del compresor en la zona 4
Temperatura_cojinetes_compresor_zona_5_puntual	Float	ºC	Temperatura de los cojinetes del compresor en la zona 5
Temperatura_devanado_motorA_puntual	Float	ºC	Temperatura de devanado A del motor del compresor. Hay 2 termostatos diferentes para medir la temperatura de devanado A
Temperatura_devanado_motorA_puntual	Float	ºC	Temperatura de devanado A del motor del compresor. Hay 2 termostatos diferentes para medir la temperatura de devanado A
Temperatura_devanado_motorB_puntual	Float	ºC	Temperatura de devanado B del motor del compresor. Hay 2 termostatos diferentes para medir la temperatura de devanado B
Temperatura_devanado_motorB_puntual	Float	ºC	Temperatura de devanado B del motor del compresor. Hay 2 termostatos diferentes para medir la temperatura de devanado B
Temperatura_devanado_motorC_puntual	Float	ºC	Temperatura de devanado C del motor del compresor. Hay 2 termostatos diferentes para medir la temperatura de devanado C
Temperatura_devanado_motorC_puntual	Float	ºC	Temperatura de devanado C del motor del compresor. Hay 2 termostatos diferentes para medir la temperatura de devanado C
Temperatura_entre_etapas_puntual	Float	ºC	Temperatura interetapa (entre la etapa 1 y la etapa 2) del compresor. Hay 2 termostatos diferentes para medir la temperatura interetapa
Temperatura_entre_etapas_puntual	Float	ºC	Temperatura interetapa (entre la etapa 1 y la etapa 2) del compresor. Hay 2 termostatos diferentes para medir la temperatura interetapa
Temperatura_impulsion_puntual	Float	ºC	Temperatura de impulsión del compresor. Hay 2 termostatos diferentes para medir la Temperatura de impulsión
Temperatura_impulsion_puntual	Float	ºC	Temperatura de impulsión del compresor. Hay 2 termostatos diferentes para medir la Temperatura de impulsión
Vibraciones_puntual	String	N/A	Estado de las vibraciones. Siempre marca BUENO
Load Demand Signal from DCS	String	N/A	Señal obsoleta
Alimentación de BOG a Compresor GB-113	String	N/A	Señal obsoleta
XV salida compresor GB-113	String	N/A	Señal obsoleta
Estado compresor según análisis y medición	String	N/A	Señal obsoleta
Fecha análisis	String	N/A	Señal obsoleta
Fecha medición	String	N/A	Señal obsoleta
GB113. Índice de polaridad	String	N/A	Señal obsoleta

PARO GB-113	String	N/A	Señal obsoleta
COLECTOR BOG FB141 A GB113	String	N/A	Señal obsoleta
LINEA RECIRCULACION GB113	String	N/A	Señal obsoleta
GN a compresor GB-113	String	N/A	Señal obsoleta
GN a compresor GB-113	String	N/A	Señal obsoleta
entrada a relicuador línea mantenimiento en frio a compresor GB-113	String	N/A	Señal obsoleta
entrada a relicuador línea mantenimiento en frio a compresor GB-113	String	N/A	Señal obsoleta

Tabla 4 Descripción de las variables incluidas en el dataset registro de variables de operación.

3.2.1 Trabajo Previo Dataset "Registro de Variables de Operación"

Este es un dataset que no está depurado por lo que para realizar una limpieza inicial se realizan los siguientes pasos en *R* (Ver anexo 1):

1. Se eliminan las siguientes 23 variables que no aportan información por ser obsoletas:
   - Analisis_aceite_puntual
   - Cambio_bandas_cilindro_1_etapa_1_total
   - Cambio_bandas_cilindro_1_etapa_2_total
   - Cambio_bandas_cilindro_2_etapa_1_total
   - Cambio_bandas_cilindro_2_etapa_2_total
   - Contador_grandes_mantenimientos_puntual
   - Numero_arranques_puntual
   - Prioridad_arranque_puntual
   - Vibraciones_puntual
   - Estado compresor según análisis y medición
   - Fecha análisis
   - Fecha medición
   - GB113. Índice de polaridad
   - Load Demand Signal from DCS
   - Alimentación de BOG a Compresor GB-113
   - XV salida compresor GB-113
   - PARO GB-113
   - COLECTOR BOG FB141 A GB113
   - LINEA RECIRCULACION GB113
   - GN a compresor GB-113
   - GN a compresor GB-113
   - entrada a relicuador linea mantenimiento en frio a compresor GB-113
   - entrada a relicuador linea mantenimiento en frio a compresor GB-113

2. Se asignan consecutivos a las variables que se llaman igual para diferenciarlas entre sí:

- Presión_aspiracion_puntual_1 (2 y 3)
- Presión_impulsion_puntual_1 (2, 3 y 4)
- Presión_interetapa_puntual_1 (2 y 3)
- Temperatura_devanado_motorA_puntual_1 (2)
- Temperatura_devanado_motorB_puntual_1 (2)
- Temperatura_devanado_motorC_puntual_1 (2)
- Temperatura_impulsion_puntual_1 (2)
- Temperatura_interetapa_puntual_1 (2)

3. Se modifican los nombres de las variables para tener un mejor manejo de los datos y las gráficas:

Nombre Original	Nombre Reducido
FECHA_HORA	FechaHora
Cambio_bandas_cilindro_1_etapa_1_puntual	CambBand1
Cambio_bandas_cilindro_1_etapa_2_puntual	CambBand2
Cambio_bandas_cilindro_2_etapa_1_puntual	CambBand3
Cambio_bandas_cilindro_2_etapa_2_puntual	CambBand4
Carga_trabajo_puntual	CargaTrab
Estado_puntual	Estado
Horas_desde_ultimo_cambio_banda_cilindro_1_etapa_1	HrsCamb1
Horas_desde_ultimo_cambio_banda_cilindro_1_etapa_2	HrsCamb2
Horas_desde_ultimo_cambio_banda_cilindro_2_etapa_1	HrsCamb3
Horas_desde_ultimo_cambio_banda_cilindro_2_etapa_2	HrsCamb4
Horas_desde_ultimo_gran_mtto	HrsGranMt
Horas_totales_operacion	HrsTotOp
Intensidad_puntual	Intensid
Potencia_puntual	Potencia
Presion_aspiracion_puntual	PresAsp1
Presion_aspiracion_puntual_2	PresAsp2
Presion_aspiracion_puntual_3	PresAsp3
Presion_diferencial_filtro_puntual	PresDifFl
Presion_impulsion_puntual	PresImp1
Presion_impulsion_puntual_2	PresImp2
Presion_impulsion_puntual_3	PresImp3
Presion_impulsion_puntual_4	PresImp4
Presion_interetapa_puntual	PresInt1
Presion_interetapa_puntual_2	PresInt2
Presion_interetapa_puntual_3	PresInt3
Temperatura_aspiracion_puntual	TempAsp
Temperatura_cojinete_motor_puntual	TempCojM
Temperatura_cojinetes_compresor_zona_1_puntual	TempCoj1
Temperatura_cojinetes_compresor_zona_2_puntual	TempCoj2
Temperatura_cojinetes_compresor_zona_3_puntual	TempCoj3
Temperatura_cojinetes_compresor_zona_4_puntual	TempCoj4
Temperatura_cojinetes_compresor_zona_5_puntual	TempCoj5

Temperatura_devanado_motorA_puntual	TempDevA1
Temperatura_devanado_motorA_puntual_2	TempDevA2
Temperatura_devanado_motorB_puntual	TempDevB1
Temperatura_devanado_motorB_puntual_2	TempDevB2
Temperatura_devanado_motorC_puntual	TempDevC1
Temperatura_devanado_motorC_puntual_2	TempDevC2
Temperatura_entre_etapas_puntual	TempEnt1
Temperatura_entre_etapas_puntual_2	TempEnt2
Temperatura_impulsion_puntual	TempImp1
Temperatura_impulsion_puntual_2	TempImp2

Tabla 5 Codificación de variables.

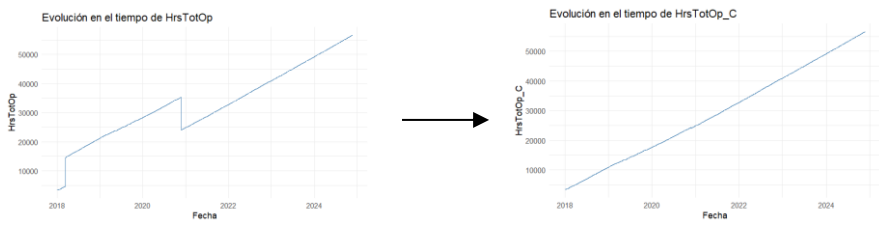
4. Se asigna a cada variable el tipo adecuado según su naturaleza, es decir se convierten a tipo factor las variables categóricas y se convierten a tipo numérico las variables cuantitativas.

5. Se identifica a lo largo de todo el dataset y en todas las variables, datos erróneos producto de errores de lectura que se observan como 'Shutdown', 'Interface down', 'Comm Fail' y 'Bad', los cuales se convierten en NAs.

6. Se crea la variable 'CargaTrab_M2' para agrupar los diferentes porcentajes de carga capturados en la variable 'CargaTrab' dentro de las 5 categorías existentes según el siguiente criterio (Ver Anexo 2):

   - '**CargaTrab_M2' = 0%,** si 'CargaTrab' no es NA y Estado es "PARO DISPONIBLE" o "MANT. PREVENTIVO" o "MANT. CORRECTIVO" o "PARO INCIDENCIA"
   - **'CargaTrab_M2' = 25%** si 'CargaTrab' está en el rango (0.1,35] y 'Estado' es igual a MARCHA.
   - **'CargaTrab_M2' = 50%** si 'CargaTrab' está en el rango (35,60] y 'Estado' es igual a MARCHA.
   - **'CargaTrab_M2' = 75%** si 'CargaTrab' está en el rango (60,85] y 'Estado' es igual a MARCHA.
   - **'CargaTrab_M2' = 100%** si 'CargaTrab' es mayor a 0,85 y 'Estado' es igual a MARCHA.
   - **'CargaTrab_M2'** = NA para las demás combinaciones posibles.

7. Se crean las variables 'HrsTotOp_C' y 'HrsGranMt_C' para corregir las variables 'HrsTotOp' y 'HrsGranMt' en las cuales se detecta una anomalía producto de un ajuste manual realizado en el contador de horas, el cual posteriormente fue corregido por los técnicos de la planta (Ver Anexo 3 y 4).



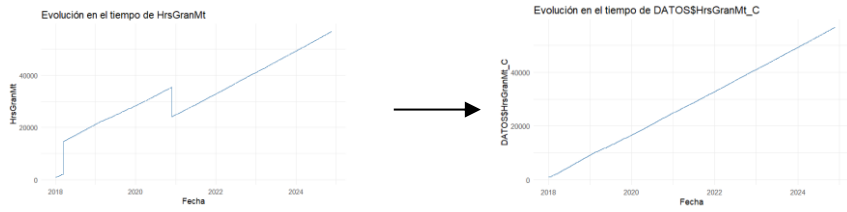


Imagen 10. Variables HrsTotOp_C' y HrsGranMt_C' antes y después de la corrección.

8. Se crea en R la variable 'HorasMismaCarga2' con el fin de identificar si está relacionada la carga de trabajo o el tiempo que está en las diferentes cargas de trabajo el compresor, con los fallos que presenta. Para hacerlo, se contabilizan cuantas horas en total ha estado en determinada carga el compresor en cada observación (Ver Anexo 5).
9. Se introduce la variable 'Fuga_Aceite' dividiendo el dataset en dos clases: 'SI' cuando el modo de falla fuga de aceite se ha reportado en el dataset "Registro de Incidencias" y 'NO' cuando no se ha reportado (Ver Anexo 6).
10. Se crean las siguientes variables, buscando descriptores más relevantes para detectar el fallo, apoyándose en el conocimiento existente de estudios previos.
    - 'RTBF' que indica el tiempo de funcionamiento entre fallos en horas.
    - 'LC' que representa los cambios de carga entre fallos. Para crear esta variable considera cada cambio de carga en los compresores, incluyendo los arranques.
    - 'Lavg': carga promedio entre fallos. Para el cálculo de esta variable solo se tuvieron en cuenta las horas de funcionamiento para que las horas de parada no distorsionaran la información.

    En estos descriptores, los datos se generaron por hora como el valor de cada uno en ese momento (Hidalgo-Mompeán et al., 2021) (Ver Anexo 7)
11. Se separan para el estudio únicamente los datos en los cuales el compresor ha trabajado al 100% porque se pretende analizar el comportamiento del compresor en su estado de operación normal (Ver Anexo 8).
12. Se observa que a menor número de horas en la misma carga de trabajo se presenta una mayor dispersión en los valores de las variables de operación a causa del periodo de estabilización que necesita el equipo, razón por la cual se decide, para evitar el ruido innecesario en la información, eliminar las observaciones con pocas horas de operación en la misma carga según el siguiente criterio (Ver Anexo 9):
    - Si el compresor ha estado trabajando a una carga de trabajo fija menos de 5 horas seguidas, se eliminan esos registros.

    A modo de ejemplo, se presenta la siguiente gráfica en donde se puede ver cómo, a menor número de horas seguidas en la misma carga de trabajo, los valores de la variable 'TempCoj5' presentan una mayor dispersión. Es más, a valores inferiores a 5 horas en la misma carga, la temperatura alcanza su mayor variación.

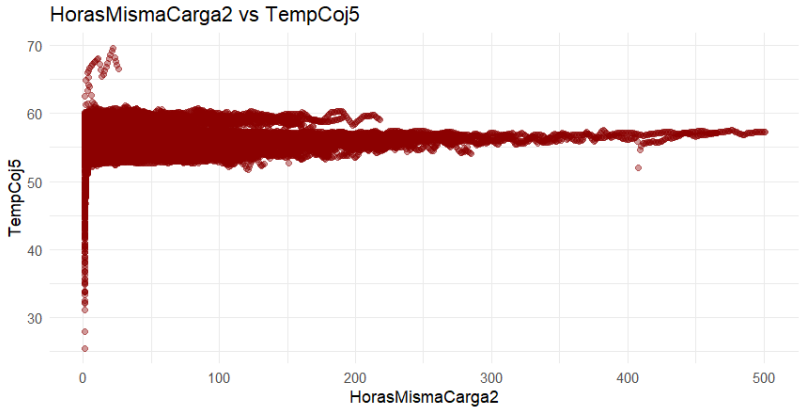


Imagen 11. Gráfica de dispersión entre temperatura del cojinete y las horas misma carga.

3.2.2 Selección de Variable Objetivo

Se selecciona la variable 'Fuga-Aceite' como variable objetivo para el desarrollo del modelo de clasificación porque representa, de acuerdo con el análisis del "Registro de Incidencias", el fallo más critico que afecta el sistema de lubricación del compresor, que a su vez es el sistema que más incidencias reporta.

Como se indica en el proceso depuración, esta variable fue agregada en el dataset "Registro de Variables de Operación" tomando la información del "Registro de Incidencias" en donde se reportan las fechas y horas en donde ha evidenciado la presencia del fallo por fuga de aceite en el compresor. De esta manera, para cada observación se indicó si el fallo fue reportado o no.

Así, 'Fuga_Aceite' se encuentra definida como una variable binaria, donde SI indica la ocurrencia de la fuga y "NO" indica la ausencia de la fuga. Con esta codificación, se permite abordar este problema como un escenario de clasificación supervisada.

3.2.3 Análisis Exploratorio Dataset “Registro de Variables de Operación”

El conjunto de variables se puede agrupar según el equipo o subequipos del cual registran datos, así como por la naturaleza del dato que están capturando. De esta forma se pueden identificar las siguientes agrupaciones para facilitar el análisis:

Compresor	Mediciones puntuales	Varios	FechaHora
			CargaTrab
			Estado
	Cambio de Bandas	Indicadores de Cambio de Bandas	CambBand1
			CambBand2
			CambBand3
			CambBand4
	Tiempo	Horas funcionamiento	HrsTotOp
			HrsCamb1
			HrsCamb2
			HrsCamb3
			HrsCamb4
			HrsGranMt

Compresor	Presión	Aspiración	PresAsp1
			PresAsp2
			PresAsp3
		Impulsión	PresImp1
			PresImp2
			PresImp3
			PresImp4
		Diferencial Filtro	PresDifFl
		Interetapas	PresInt1
			PresInt2
			PresInt3

Compresor	Temperatura	Aspiración	TempAsp
		Cojinete	TempCoj1
			TempCoj2
			TempCoj3
			TempCoj4
			TempCoj5
		Interetapas	TempEnt1
			TempEnt2
		Impulsión	TempImp1
			TempImp2

Motor del compresor	Consumos	Intensidad	Intensid
		Potencia	Potencia
	Temperatura	Cojinete	TempCojM
	Temperatura	Devanado	TempDevA1
			TempDevA2
			TempDevB1
			TempDevB2
			TempDevC1
			TempDevC2

Tabla 6 Agrupación de las variables.

Se muestra, con la ayuda de *SAS Enterprise Miner* un resumen de los principales estadísticos de las variables numéricas.

Variable	Rol	Media	Desviación estándar	No ausente	Ausente	Mínimo	Mediana	Máximo	Asimetría	Curtosis
HorasMismaCarga2	INPUT	68.50082	63.46756	44281	0	6	49	501	2.075183	6.513888
HrsCamb1	INPUT	9457.915	7959.229	44252	29	0.740332	6958.465	28667.49	0.887727	-0.38826
HrsCamb2	INPUT	9127.909	7281.98	44254	27	0.821404	7188.76	26553.08	0.801434	-0.468
HrsCamb3	INPUT	9816.551	6599.318	44246	35	0.821404	8660.661	25202.71	0.455022	-0.80442
HrsCamb4	INPUT	18769.94	13445.22	44256	25	0.821404	17045.89	43592.98	0.283791	-1.29916
HrsGranMt_C	INPUT	29597.46	16121.18	44259	22	976.9058	30447.42	56703.7	-0.08302	-1.17854
HrsTotOp_C	INPUT	30142.82	15440.05	44263	18	3339.431	30450.81	56702.29	-0.01797	-1.1925
Intensid	INPUT	102.1443	8.907157	44268	13	0	102.4645	115.6731	-9.79948	109.8689
LC	INPUT	66.46993	60.41439	44281	0	0	50	294	1.284929	1.449418
Lavg2	INPUT	97.4282	1.590915	44281	0	85.56338	97.47036	100	-0.70361	1.812434
Potencia	INPUT	811.0626	74.22679	44278	3	-0.00811	813.0381	932.4066	-8.5464	91.17546
PresAsp1	INPUT	162.1506	20.90231	44256	25	66.83633	161.4164	266.9318	0.046603	3.085385
PresAsp2	INPUT	159.8469	20.71314	44254	27	12.7676	160.051	283.3793	-0.32468	2.368567
PresAsp3	INPUT	162.0684	21.334	44259	22	62.65974	161.1522	315.9722	0.210917	3.566018
PresDifFl	INPUT	6.464965	2.310965	44251	30	2.810736	6.17253	25.65846	5.441696	33.70642
PresImp1	INPUT	7.680282	0.685872	44261	20	1.35197	7.567694	9.668807	0.326568	1.253106
PresImp2	INPUT	7.615716	0.680186	44257	24	1.341838	7.500189	9.578728	0.330971	1.2352
PresImp3	INPUT	7.63674	0.681981	44251	30	1.345983	7.526756	9.603035	0.327912	1.239747
PresImp4	INPUT	7.616777	0.680852	44259	22	1.350624	7.509084	9.595089	0.334017	1.227353
PresInt1	INPUT	2.458259	0.143833	44259	22	0.480159	2.43865	3.904133	-0.59878	10.53771
PresInt2	INPUT	2.453449	0.142624	44255	26	0.492018	2.433277	3.897075	-0.50883	9.233833
PresInt3	INPUT	2.471169	0.143391	44245	36	0.502529	2.451694	3.918424	-0.49801	9.160968
RTBF	INPUT	2029.543	1622.037	44281	0	1	1614	6146	0.754814	-0.48936
TempAsp	INPUT	-118.664	7.618679	44274	7	-145.743	-119.024	-84.5013	-0.00561	0.179563
TempCoj1	INPUT	61.3916	1.245716	44257	24	51.95677	61.29283	70.22287	0.511866	1.581432
TempCoj2	INPUT	58.30387	1.179748	44262	19	53.71028	58.26065	70.3087	0.879727	4.171577
TempCoj3	INPUT	54.16069	10.5756	44258	23	-15.0347	55.68135	68.77041	-6.29132	38.22674
TempCoj4	INPUT	55.52086	10.79339	44263	18	-15.0347	57.03981	69.88651	-6.29086	38.16459
TempCoj5	INPUT	56.16051	1.245026	44265	16	51.68428	56.11412	69.6723	0.875142	4.724706
TempCojM	INPUT	42.83805	3.702112	44260	21	32.00343	43.02615	57.17371	-0.09679	-0.68858

TempCojM	INPUT	42.83805	3.702112	44260	21	32.00343	43.02615	57.17371	-0.09679	-0.68858
TempDevA1	INPUT	68.06514	6.727872	44263	18	44.27776	68.39023	94.22092	-0.14578	-0.53217
TempDevA2	INPUT	68.05748	6.69096	44261	20	44.40957	68.38142	93.87863	-0.14954	-0.52914
TempDevB1	INPUT	66.21853	6.919904	44264	17	-237.037	66.53674	92.05784	-2.22742	85.22825
TempDevB2	INPUT	67.03018	6.75549	44264	17	43.4716	67.35501	93.03913	-0.14248	-0.54291
TempDevC1	INPUT	67.19981	6.951496	44261	20	-231.682	67.53997	93.3494	-2.13715	79.24035
TempDevC2	INPUT	66.69516	6.677314	44261	20	42.95606	67.01057	92.44773	-0.14086	-0.53465
TempEnt1	INPUT	-44.5101	10.07241	44275	6	-81.6053	-44.8607	-5.50676	-0.05442	0.147045
TempEnt2	INPUT	-43.7597	10.10139	44273	8	-82.0603	-44.1064	-4.24837	-0.04237	0.144454
TempImp1	INPUT	27.69696	12.26702	44276	5	-13.162	27.46795	70.29114	-0.04089	-0.22502
TempImp2	INPUT	28.29284	13.25832	44274	7	-13.841	28.30394	82.12127	0.061796	-0.08589

Tabla 7 Resumen de los estadísticos de las variables.

Donde se encuentra que todas las variables tienen valores ausentes generados por los errores de lectura en los sensores.

Además, se observa una serie de variables con valores máximos y otras con valores mínimos aparentemente atípicos u anómalos y en muchas de ellas se observa una marcada asimetría según el índice skew.

Por tal motivo, se decide detectar los atípicos utilizando los siguientes criterios:

- **Criterio 1:** Basado en la asimetría (skewness) de la variable.
  Si la asimetría es menor que 1 (simétrica o ligeramente asimétrica), se usan la media y la desviación estándar y un valor se considera atípico si su valor estándar (z-score) está por encima de 3 (en valor absoluto).
  Si la asimetría es mayor o igual a 1 (asimétrica), se usan la mediana y la desviación absoluta mediana (MAD) y un valor se considera atípico si está por encima de 8 veces el MAD (en valor absoluto).
- **Criterio 2:** Se define otro criterio basado en los percentiles y el rango intercuartílico (IQR).
  Los valores se consideran atípicos si están alejados por fuera del rango intercuartílico en una magnitud mayor a 1.5 veces dicho rango intercuartílico. De esta manera, se considera un dato atípico solo si cumple los dos criterios simultáneamente.

De acuerdo con estos criterios, la proporción de atípicos detectados en cada variable es la siguiente:

```
  HrsTotOp_C     HrsCamb1     HrsCamb2     HrsCamb3     HrsCamb4  HrsGranMt_C      PresAsp1      PresAsp2
0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0267609133 0.0193762562
    PresAsp3     PresImp1     PresImp2     PresImp3     PresImp4     PresDifFl     PresInt1     PresInt2
0.0274384047 0.0014678982 0.0014678982 0.0014678982 0.0014678982 0.0269189946 0.0057135114 0.0056231792
    PresInt3      TempAsp     TempCoj1     TempCoj2     TempCoj3     TempCoj4     TempCoj5     TempEnt1
0.0057586775 0.0028680472 0.0057135114 0.0105011179 0.0229895440 0.0230798762 0.0096881281 0.0027325489
    TempEnt2     TempImp1     TempImp2     Intensid     Potencia     TempCojM    TempDevA1    TempDevA2
0.0023938032 0.0010388203 0.0026422168 0.0065490843 0.0066168334 0.0001354983 0.0007000745 0.0007000745
   TempDevB1    TempDevB2    TempDevC1    TempDevC2
0.0001129152 0.0007000745 0.0001129152 0.0007226576
```

Imagen 12. Proporción de atípicos por variable.

Debido a que la proporción de atípicos en cada variable es inferior al 2,7%, se considera baja y se toma la decisión de convertirlos en NA teniendo en cuenta la cantidad de datos disponibles para el estudio (Ver Anexo 10).

Una vez convertidos los valores atípicos en NAs, se crea una nueva variable llamada ‘prop_missings’ que calcula la proporción de NAs en cada observación, de la cual se puede ver un resumen a continuación.

```
> # 2. Resumen de la proporción de valores faltantes por observación
> summary(DATOS_100_C2_F$prop_missings)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.000000 0.000000 0.000000 0.006112 0.000000 0.733333
```

Imagen 13. Creación de nueva variable propmissings y sus características.

Luego de convertir en NAs los valores atípicos, se revisa la cantidad total de NAs en cada variable, la cual se puede ver a continuación:

```
> (prop_missingsVars4 <- apply(is.na(DATOS_100_C2_SA), 2, mean))
    FechaHora    HrsTotOp_C      HrsCamb1      HrsCamb2      HrsCamb3      HrsCamb4
0.0005194101 0.0004064949 0.0006549084 0.0006097423 0.0007904067 0.0005645762
 HrsGranMt_C      PresAsp1      PresAsp2      PresAsp3      PresImp1      PresImp2
0.0004968271 0.0273254895 0.0199859985 0.0279352318 0.0019195592 0.0020098914
     PresImp3      PresImp4     PresDifFl      PresInt1      PresInt2      PresInt3
0.0021453897 0.0019647253 0.0275964861 0.0062103385 0.0062103385 0.0065716673
      TempAsp      TempCoj1      TempCoj2      TempCoj3      TempCoj4      TempCoj5
0.0030261286 0.0062555046 0.0109301958 0.0235089542 0.0234863711 0.0100494569
     TempEnt1      TempEnt2      TempImp1      TempImp2      Intensid      Potencia
0.0028680472 0.0025744676 0.0011517355 0.0028002981 0.0068426639 0.0066845826
     TempCojM     TempDevA1     TempDevA2     TempDevB1     TempDevB2     TempDevC1
0.0006097423 0.0011065694 0.0011517355 0.0004968271 0.0010839864 0.0005645762
    TempDevC2 prop_missings
0.0011743186 0.0000000000
```

Imagen 14. Proporción de NAs por variable.

En donde se puede ver que todas las proporciones de NA en las variables son inferiores a 2,8%, por lo que se toma la decisión de retirar del estudio todas estas observaciones que tienen valores faltantes para eliminar el ruido que puedan ocasionar teniendo en cuenta la gran cantidad de observaciones completas disponibles en el dataset. Además, la reducción del dataset favorece la capacidad computacional disponible (Hidalgo-Mompeán et al., 2021) (Ver Anexo 11)

3.2.4 Relación entre variables

3.2.4.1 Relación entre variables de operación

Se calcula la siguiente matriz de correlación entre todas las variables en el estudio hasta el momento.

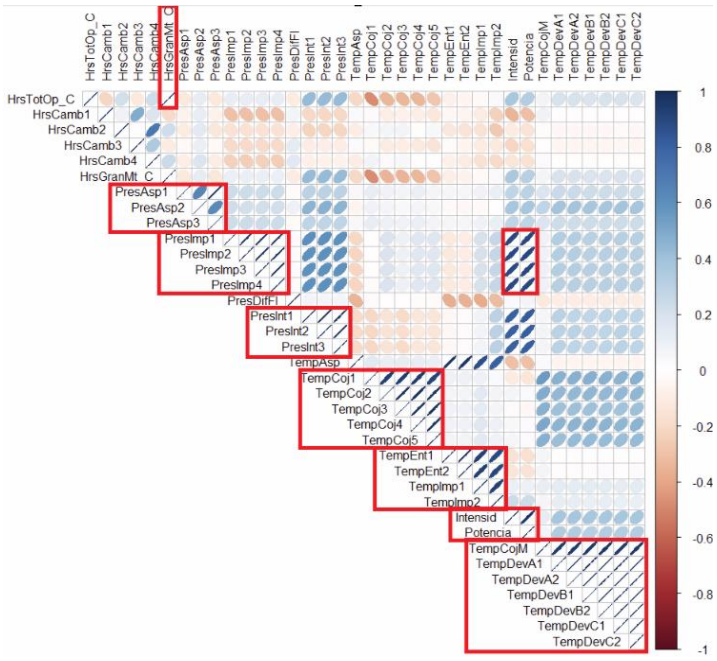


Imagen 15. Correlación entre variables de operación.

En primera instancia, se evidencia fuerte correlación entre las variables que controlan el mismo parámetro de operación del compresor enmarcadas en recuadros rojos. Por consiguiente, al existir un coeficiente de correlación de Pearson mayor a 0,9 se decide trabajar con un promedio de las variables altamente correladas para no perder la información de ninguna de ellas.

En específico, se calculan las siguientes nuevas variables (Ver Anexo 12):

- 'PresAsp': promedio entre 'PresAsp1', 'PresAsp2' y 'PresAsp3'.
- 'PresImp': promedio entre 'PresImp1', 'PresImp2', 'PresImp3' y 'PresImp4'.
- 'PresInt': promedio entre 'PresInt1', 'PresInt2' y 'PresInt3'.
- 'TempCoj': Promedio entre 'TempCoj1', 'TempCoj2', 'TempCoj3', 'TempCoj4' y 'TempCoj5'.
- 'TempEnt': promedio entre 'TempEnt1' y 'TempEnt2'.
- 'TempImp': promedio entre 'TempImp1' y 'TempImp2'.
- 'TempDev': promedio entre 'TempDevA1', 'TempDevA2', 'TempDevB1', 'TempDevB2', 'TempDevC1' y 'TempDevC2'.
- Se retira del estudio la variable 'HrsGranMt_C' por alta correlación con 'HrsTotOp_C'.
- Se retira del estudio la variable 'Potencia' por alta correlación con 'Intensid'.

Se estudia la correlación entre las variables que siguen en el estudio:

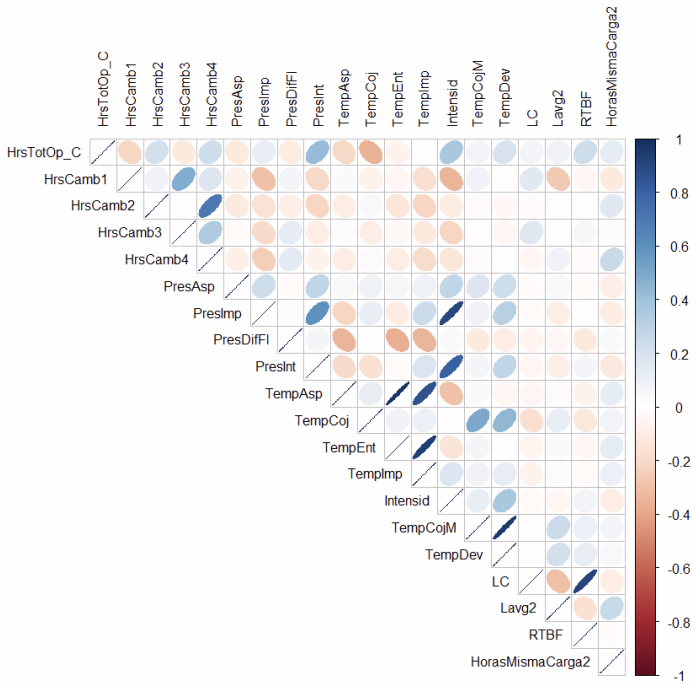


Imagen 16. Correlación que permanecen en el estudio.

En este punto, se observa una alta correlación entre las siguientes variables:

- Presión de impulsión e Intensidad.
- Presión interetapas e Intensidad.
- Temperatura de aspiración, Temperatura entreetapas y Temperatura de impulsión.
- Temperatura de cojinete del motor y Temperatura de devanado del motor.
- 'LC' y 'RBTF'.

Por lo cual se decide realizar las siguientes acciones:

- Retirar intensidad del estudio.
- Crear las variables Diferencial de temperatura ‘Dt’ y Diferencial de Presión ‘Dp’ para retirar ‘PresAsp’, ‘Temperatura de Aspiración’ y ‘Temperatura impulsión’. (Hidalgo-Mompeán et al., 2021).
- Retirar Temperatura de cojinete del motor.
- Retirar ‘RBTF’.
- Crear la variable ‘HrsCambBand’ que recoja la información de las 4 variables de cambio de banda.

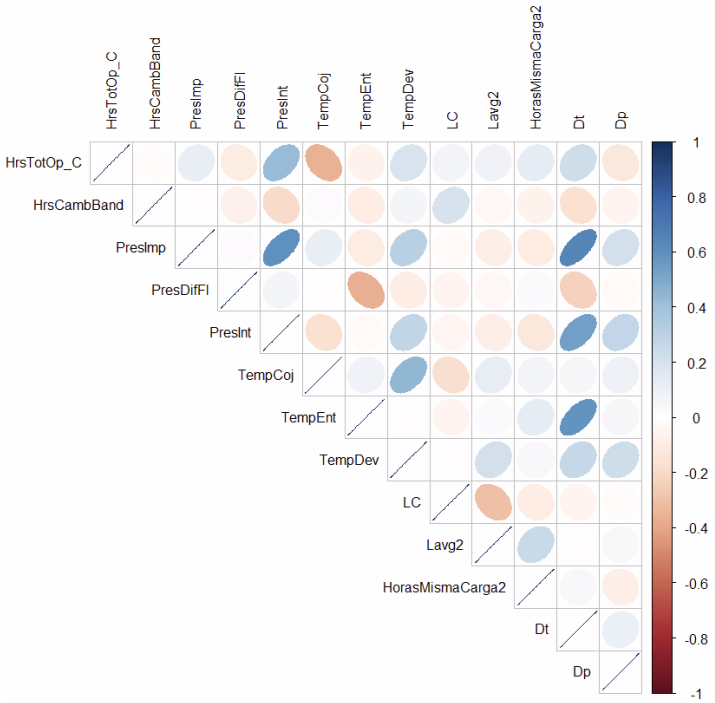


Imagen 17. Correlación entre variables finales.

En la Imagen 17, se observan las variables finales que han sido pre-seleccionadas luego de eliminar las correlaciones altas para evitar colinealidad.

3.2.4.2 Relación con la variable objetivo

3.2.4.2.1 Estadístico VdeCramer

Para estimar de manera inicial la relación entre las variables input y la variable objetivo, se calcula el estadístico V de Cramer para todas las variables. Este estadístico se utiliza para medir la independencia entre dos variables categóricas basado en el estadístico Chi-cuadrado, pero tiene la ventaja de que su valor está acotado entre 0 y 1. El cálculo se realiza de la siguiente manera:

$$v = \sqrt{\frac{x^2}{n \times \min(l-1, k-1)}}$$

Para este caso de estudio, al tener todas las variables input de tipo numerico, se hace necesario discretizar estas variables para poder calcular el estadistico en relación a la

variable objetivo ‘Fuga_Aceite’. Los resultados de este cálculo se muestran en la siguiente gráfica.

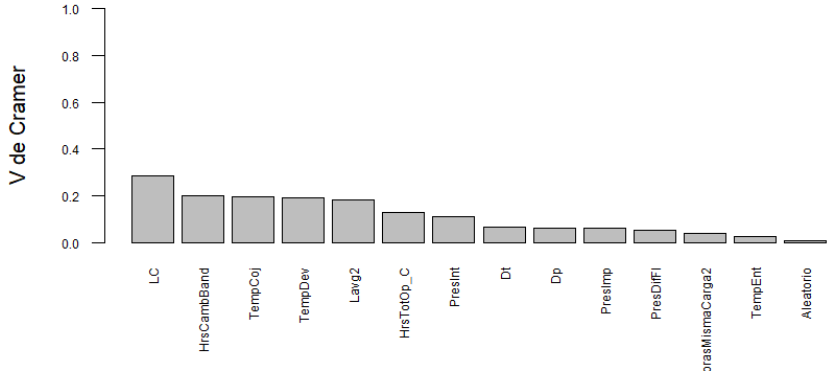


Imagen 18. Importancia de variables según V de Cramer.

Los resultados se pueden interpretar utilizando la siguiente guia (Rea & Parker, p.203):

Valor V de Cramer	Nivel de Asociación
0,0 - 0,1	Despreciable
0,1 - 0,2	Débil
0,2 - 0,4	Moderada
0,4 - 0,6	Relativamente fuerte
0,6 - 0,8	Fuerte
0,8 - 1,0	Muy fuerte

Tabla 8 Tabla de asociación según VdeCramer

Por lo cual, se observa que según el estadisto V de Cramer, la variable ‘LC’ tiene una asociación moderada, mientras que las variables ‘HrsCambBand’, ‘TempCoj’, ‘TempDev’, ’Lavg’, ‘HrsTotOp_C’, y ‘PresInt’ tienen una asociación débil. Por otro lado, las variables ‘Dt’, ‘Dp’, ‘PresImp’, ‘HorasMismaCarga2’, ‘TempEnt’ y ‘Aleatorio’ tienen una relación despreciable.

Se anota que ninguna variable tiene menos asociación (valor V de Cramer inferior) que la variable ‘Aleatorio’, la cual fue creada tomando valores aleatorios para poder identificar variables importantes por lo que se puede concluir que todas las variables guardan una relación con la variable objetivo.

En este caso, los resultados del estadistico V de Cramer deben tomarse unicamente como una aproximación inicial la relación existente entre las variables input y la variable objetivo por el sesgo que pueda generar la discretización realizada a las variables numericas.

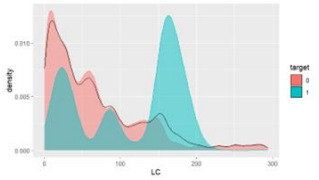
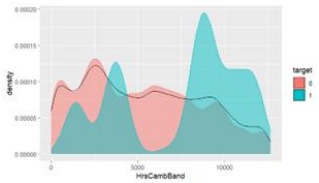
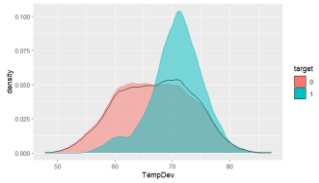
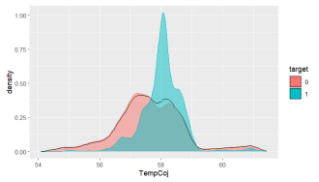
3.2.4.2.2 Análisis exploratorio frente la variable objetivo.

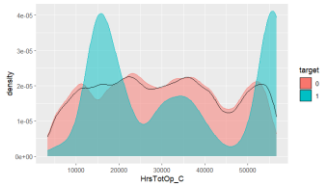
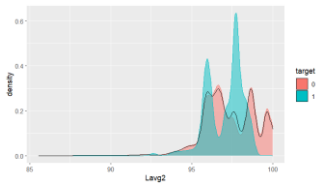
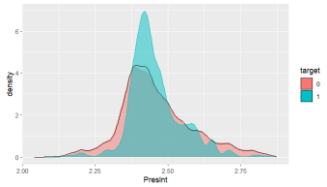
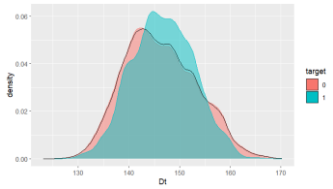
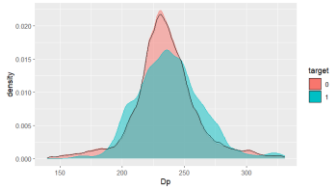
Es importante analizar el comportamiento de las variables frente a la presencia de fuga, de tal forma que se pueda confirmar si existe una asociación entre ellas como lo señala el estadístico VdeCramer.

En la siguiente tabla resumen se puede observar cuales variables, según el análisis descriptivo, tienen una capacidad discriminativa para clasificar la aparición de las fugas

de aceite, basándonos en el gráfico de densidad de frecuencias superpuestas para cada variable.

Este tipo de gráfico permite identificar diferencias en el comportamiento de la variable analizada según el valor de la variable objetivo. En la representación, la curva azul corresponde al comportamiento cuando Fuga_Aceite = 1 (SI), mientras que la curva rosada refleja el comportamiento cuando Fuga_Aceite = 0 (NO). Este tipo de visualización resulta especialmente útil en contextos con datos reales que presentan un alto nivel de ruido ya que facilita la detección de posibles diferencias en los patrones de distribución.

Variable	Capacidad discriminativa preliminar – Grafica Densidad	Interpretación
LC (cambios de carga entre fallos)	Media-Alta 	Cuando se presentan las fugas de aceite, el número de cambios de carga entre fallos promedio es marcadamente superior. Es decir, entre más cambios de carga, mayor esfuerzo en sellos, cojinetes y demás componentes aumentando el riesgo de fugas. Aunque existe solapamiento, entrega buenas señales para clasificar.
HrsCambBand (horas desde el ultimo cambio de bandas)	Media-Alta 	Aunque existe solapamiento entre las clases, se observa un comportamiento marcadamente distinto ya que cuando se presentan las fugas de aceite, el tiempo desde el ultimo cambio de bandas es generalmente mayor. Es decir, entre más tiempo haya transcurrido desde el último cambio de bandas, mayor desgaste acumulado en el compresor aumentando el riesgo de fugas. A pesar de haber solapamiento esta variable entrega buenas señales para clasificar las fugas de aceite
TempDev (temperatura de devanado del motor)	Media 	Aunque la diferencia entre las medias es relativamente pequeña, se nota una temperatura media concentrada en un valor ligeramente superior cuando se presentan las fugas, lo que indica una relación, aunque no de manera lineal simple por el solapamiento observable entre las dos clases. Una falla en el sistema de lubricación genera mayor esfuerzo por aumento de fricción lo que puede aumentar la temperatura del devanado. El alto solapamiento afirma que por sí sola no es un predictor fuerte, pero complementada por otras tal vez sí.
TempCoj (temperatura de los cojinetes del compresor)	Media 	Diferencia entre las medias es relativamente pequeña, pero se nota una temperatura media concentrada en un valor ligeramente superior cuando se presentan las fugas, lo que indica una relación, aunque no de manera lineal simple por el solapamiento observable entre las dos clases.

		Una fuga de aceite afecta la lubricación de los cojinetes lo que puede aumentar su temperatura. Tiene un alto solapamiento entre clases, lo que indica que por sí sola no es un predictor fuerte, pero complementada por otras tal vez sí.
HrsTotOp_C (horas totales de operación)	Media 	Existe solapamiento moderado entre las clases, pero se presentan diferencias en el comportamiento ya que al haber fugas de aceite el tiempo de operación tiene un pico en horas tempranas y otro en horas tardías. Es decir, existe una relación entre el desgaste acumulado indicado en las horas totales de operación y la aparición de las fugas de aceite. A pesar de haber solapamiento esta variable entrega buenas señales para clasificar las fugas de aceite
Lavg (Carga promedio entre fallos)	Media 	A pesar de que existe solapamiento entre las clases, se observa un comportamiento ligeramente distinto ya que cuando se presentan las fugas de aceite, la carga promedio se concentra en un rango de valores diferente y más reducido. Cargas sostenidas elevadas pueden incrementar esfuerzos y riesgo de fugas. Aporta señal intermedia que conviene explotar en conjunto con otras variables importantes
PresInt (Presión Interetapas)	Media-Baja 	Presenta alto solapamiento entre las clases, pero cuando hay fuga de aceite se muestra un pico ligeramente más concentrado alrededor de 2,4 bar. La aparición de una fuga de aceite y una variación de presión interetapas del compresor sería un efecto indirecto y secundario
Dt (diferencial de temperatura)	Media-Baja 	La variable muestre distribuciones muy solapadas entre las dos clases, con leves diferencias hacia mayores diferenciales de temperatura cuando hay fuga. Su poder discriminativo individual es medio-bajo, pero podría aportar información complementaria al combinarse con otras variables
Dp (Diferencial de presión)	Baja 	Presenta un alto solapamiento entre clases mostrando un bajo poder discriminativo. Aunque un mayor diferencial de presión podría reflejar esfuerzo de comprensión vinculado a fugas, las señales observadas son débiles y, en un modelo de clasificación, deberían complementarse con variables de mayor relevancia

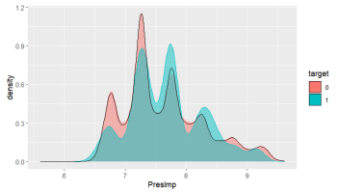
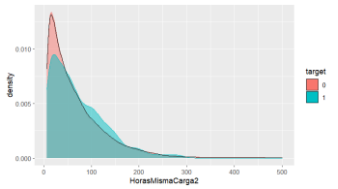
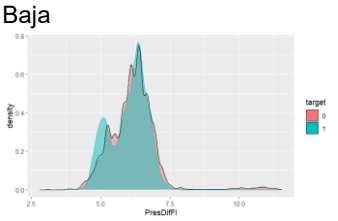
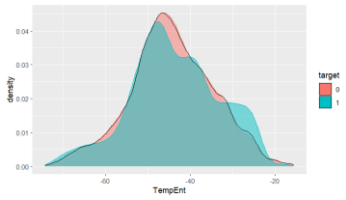
PresImp (Presión de Impulsión)	Baja 	Muestra altísima similitud entre clases, reflejando bajo poder discriminativo. Esto puede confirmar que depende principalmente de las condiciones de operación y no directamente del sistema de lubricación, salvo en casos externos de afectación global al compresor.
HorasMismaCarga (Tiempo en misma carga)	Baja 	Presenta altísimo solapamiento entre clases, definiendo su bajo poder discriminativo, aunque con ligera mayor frecuencia de fugas tras 100 horas. No parece determinante, pero puede aportar información complementaria al reflejar esfuerzo sostenido que favorece desgaste progresivo y en consecuencia incrementa el riesgo de fugas.
PresDifFl (Presión diferencial del filtro del compresor)	Baja 	Muestra un solapamiento muy alto entre clases evidenciando su bajo poder discriminativo por lo que parece no ser determinante para identificar fugas. Aun así, puede considerarse complementaria al reflejar esfuerzos adicionales en el compresor que indirectamente podrían acelerar el desgaste y aumentar el riesgo de fugas de aceite.
TempEnt (Temperatura entre etapas)	Baja 	Presenta altísimo solapamiento entre clases. Aunque se observa una ligera mayor frecuencia de fugas con temperaturas más altas, no resulta determinante según esta comparación. Se considera complementaria y con relación indirecta al riesgo de fugas de aceite en condiciones de operación exigentes.

Tabla 9 Resumen de las variables y su relación con la fuga de aceite.

4. MODELIZACIÓN

A continuación se presentan las diferentes estimaciones realizadas. En primer lugar se realiza el balanceo de las clases, con el proposito de evaluar si el desbalance tiene un efecto en los resultados de los modelos. Seguidamente se efectúa la selección de variables, posteriormente se desarrollan los algoritmos de machine learning para los cuales se investigan los parametros a tunear. Finalmente, se comparan todos los modelos estimados tanto para los datos originales como para los datos balanceados y se realiza la selección del mejor modelo.

Todos los pasos de esta sección de modelización se ejecutan utilizando tanto el dataset original, como el balanceado para contrastar los resultados.

4.1 Balanceo de clases

Para el balanceo de las clases se utiliza la técnica SMOTE a través de la cual se realiza un *oversampling* de las observaciones de la clase minoritaria y posteriormente se realiza un *undersampling* de las observaciones de la clase mayoritaria para alcanzar un balance adecuado.

Para el oversampling, se utiliza un metodo de interpolación entre un número k de vecinos más proximos, esto se realiza en R utilizando el paquete *"smotefamily"* (Ver Anexo 13).

De igual manera, se debe decidir el número de observaciones a crear de la clase minoritaria y el número de observaciones a retirar de la clase mayoritaria para alcanzar el balance objetivo, teniendo especial cuidado en no ser demasiado agresivo en la creación y/o eliminación de observaciones, de tal manera que no se cree demasiada información 'sintentica' al hacer el oversampling y que no se pierda información valiosa al realizar el undersampling.

Para este caso de estudio, se ha decido utilizar k=5 vecinos para calcular las observaciones a crear. De igual manera, se ha decidido que el balance del dataset presente una relación de 2:1, es decir que la clase mayoritaria, en este caso la negativa, tenga el doble de observaciones que la clase minoritaria.

En las siguientes graficas se puede observar el balance de clases en el dataset original y luego de aplicar SMOTE

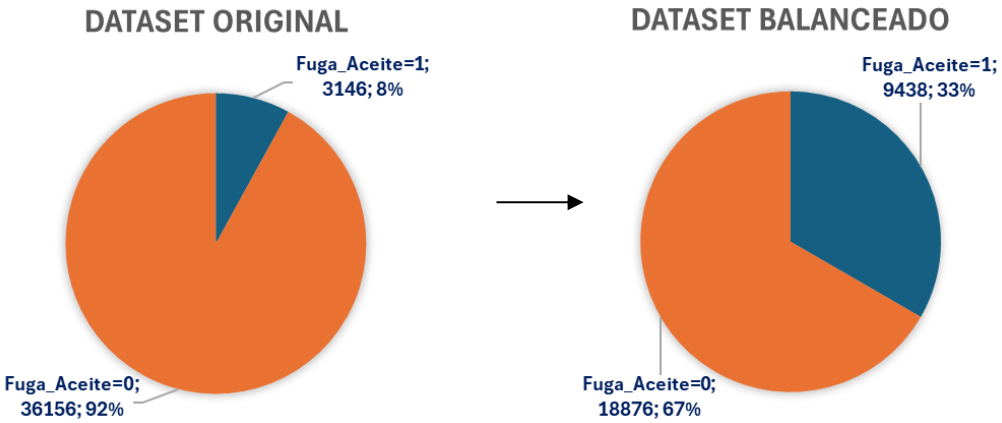


Imagen 19. Proporción de clases antes y después de aplicar SMOTE.

4.2 Selección de Variables

Para establecer qué variables se manejaran en los modelos a desarrollar, se han utilizado diferentes métodos de selección con los que se han obtenido varios sets de variables que más adelante se comparan a través de validación cruzada repetida para elegir el set que mejor se ajuste.

Los metodos que se han utilizado son (Portela, 2025):

- Stepwise AIC: Este metodo realiza, a través de una busqueda secuencial, la evaluación una por una de la significancía de todas las variables disponibles. En cada paso se evalúan todas las posibles variables a eliminar o introducir hasta que todas las incluidas aporten información significativa. Las variables se seleccionan según el menor Criterio de Información de Akaike AIC.
- Stepwise BIC: Es similar al anterior, solo que se seleccionan según el menor Criterio de información Bayesiano que penaliza más el número de parámetros.

- Boruta: Es un metodo de busqueda secuencial que hace copias de las variables inputs donde los valores fila están permutados (no en su lugar), y compara la importancia de la variable input original con la de su copia permutada. El proceso es secuencial y usa random forest
- MxM: (Max Min Parents and Children) Realiza una búsqueda heurística secuencial, utilizando inferencia condicional, quedándose con variables con relación con la objetivo condicionada a la presencia de otras. Requiere establecer el número de variables input max_k para hacer inferencia condicional. Para este caso de estudio, al tener 13 variables se recomienda utilizar un max_k=3.
- RFE: (Recursive Feature Elimination). Realiza de manera secuencial la selección de las variables más importantes tomando subconjuntos de tamaño k de las variables input. Se requiere establecer a través de un tuneo, el número k de los subconjuntos de variables a probar. Para este caso de estudio, se ha establecido k=12 a través del tuneo realizado en R de todas las combinaciones posibles optimizando la metrica F1-score:

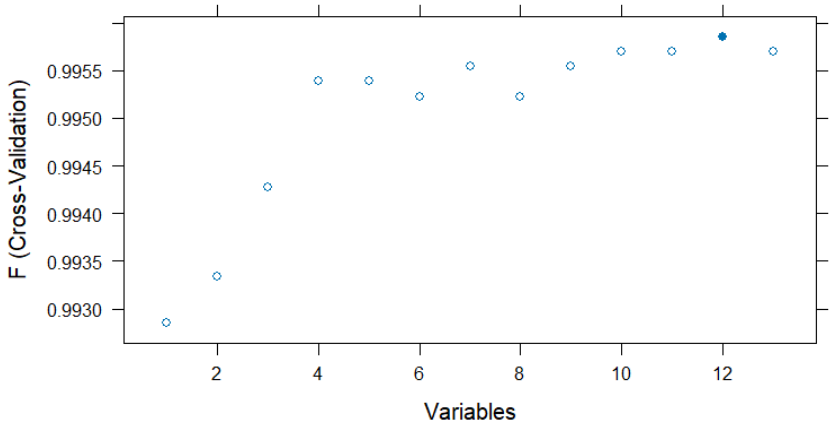


Imagen 20. Gráfico de número de variables vs. F1.

- Stepwise repetido AIC: Es el metodo Stepwise AIC pero busca estabilidad repitiendo el proceso varias veces con diferentes particiones o remuestreos reduciendo el efecto del azar. Entrega el subconjunto de variables con mayor frecuencia de selección en las repeticiones.
- Stepwise repetido BIC: Es igual que el anterior, pero utiliza el Criterio de Información Bayesiano.

Para estos dos últimos métodos, se utiliza una partición de datos del 80% para entrenamiento y 20% para prueba, variando la semilla de la partición 40 veces para obtener resultados más confiables (Ver Anexo 14).

Debido a las repeticiones entre los sets resultantes de cada metodo, se eligen 4 sets de variables a probar para el dataset original y 5 sets de variables a probar para el dataset balanceado.

Para cada uno de los sets de variables obtenidos en cada método de selección utilizado, se realiza una Regresión Logistica con validación cruzada repetida de cuatro grupos con cinco repeticiones y se comparan los resultados a través del F1-Score y el AUC. El número de variables en cada set se indica con un guión en el nombre.

Para el dataset original, se muestra en el siguiente gráfico comparativo de boxplots los resultados de F1-Score y AUC.

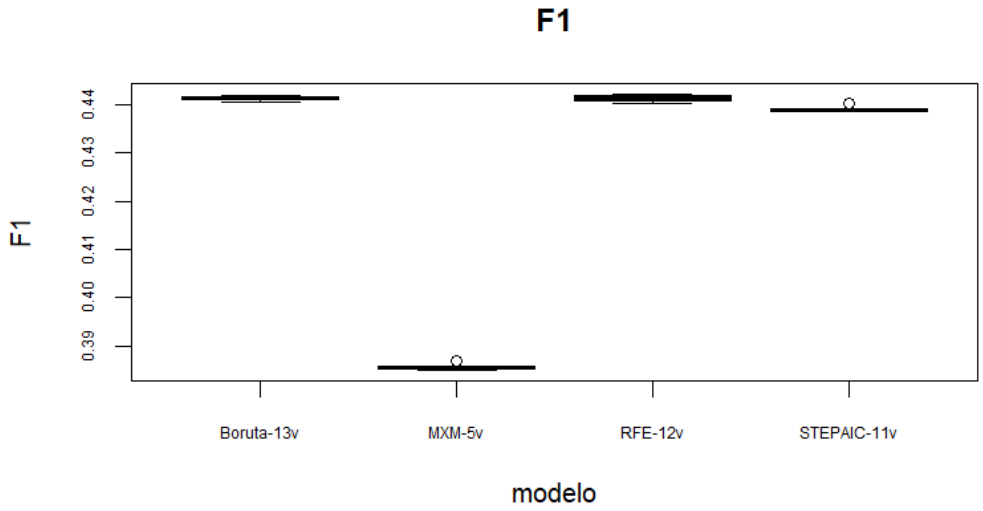


Imagen 21. F1 score para selección de variables en dataset original.

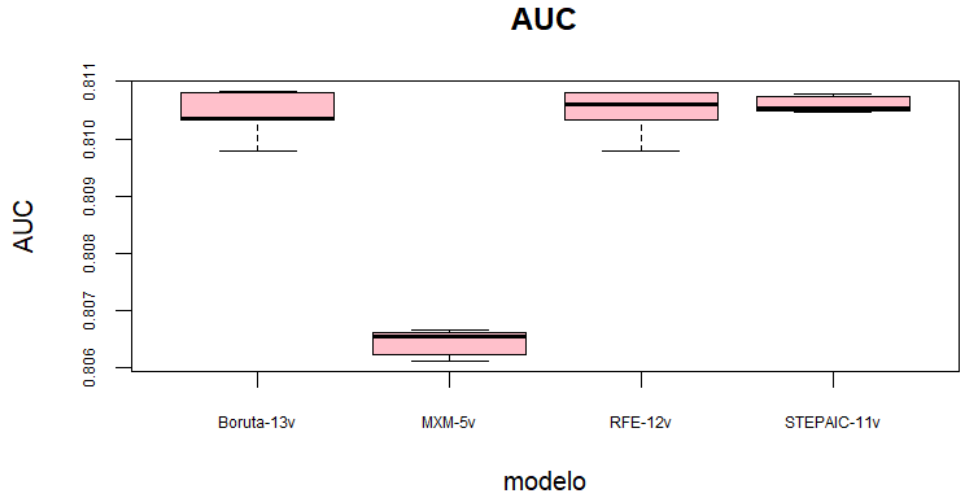


Imagen 22. AUC para selección de variables en dataset original.

Respecto al F1-Score, las diferencias en varianza para los 4 sets de variables son minimas, mientras que los sets que mejor resultado de F1-score alcanzan son los resultantes del Boruta, RFE y STEPAIC mientras que el set obtenido con MxM se queda un poco por debajo aunque con una diferencia de apenas 0,06 respecto a las demás. En cuanto al AUC se observa un comportamiento muy similar aunque en este caso la diferencia entre MxM y los demás sets es menor con apenas 0,005 respecto a las demás. Teniendo en cuenta que las diferencias en los resultados anteriormente mencionados son minimas tanto en F1-score como en AUC, se valora la diferencia entre el número de variables que compone cada set y se observa que el set obtenido con el metodo MxM contiene solo 5 variables, mientras que los demás contienen de 11 a 13 variables marcando una reducción significativa de la complejidad de los modelos a realizar, por tal motivo, teniendo en cuenta el principio de parsimonia, se decide escoger el set de variables obtenidas del metodo MxM.

De esta manera, las variables seleccionadas para el datset original son:

- 'HrsCambBand': horas desde el ultimo cambio de bandas en el compresor.

- ‘PresDifFl’: Presión diferencial en el filtro del compresor
- ‘TempCoj’: Temperatura de los cojinetes del compresor
- ‘TempDev’: Temperatura de devanado del motor.
- ‘LC’: Número de cambios de carga desde el último fallo

Por otra parte, para el dataset balanceado, se presentan en la siguiente gráfica los resultados de las métricas F1-Score y AUC obtenidos al aplicar la validación cruzada repetida con Regresión Logística.

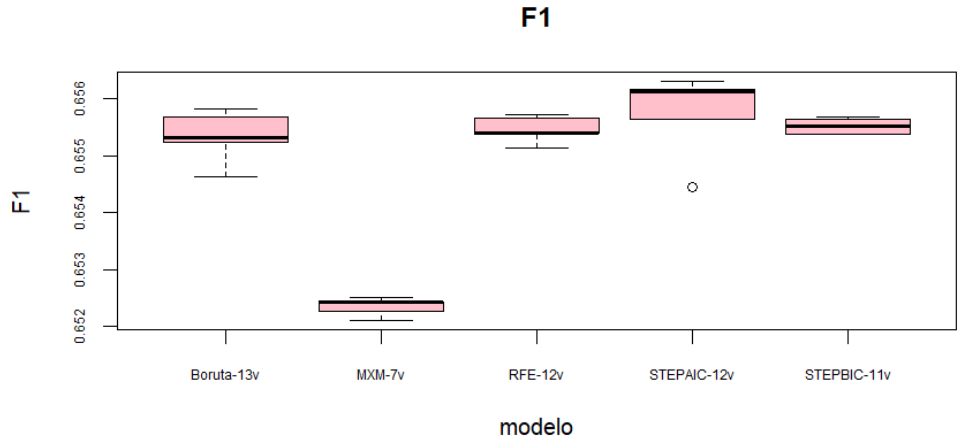


Imagen 23. F1 score para selección de variables en dataset balanceado.

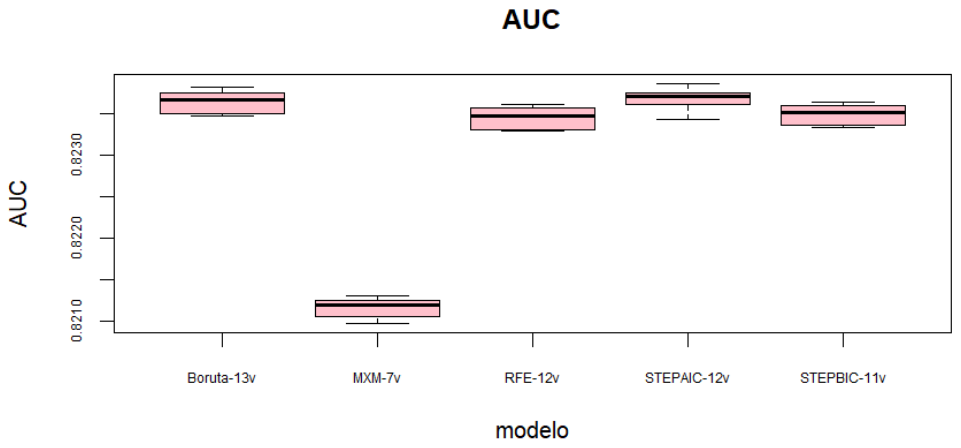


Imagen 24. AUC para selección de variables en dataset balanceado.

De manera muy similar a lo presentado con el dataset original, respecto al F1-Score, las diferencias en varianza para los 5 sets de variables son minimas, aunque se observa menor varianza en los sets de STEPBIC y MxM, mientras que el de mayor varinza es el STEPAIC y Boruta. Por otro lado, el que mejor resultado de F1-score obtiene es STEPAIC seguido del Boruta, RFE y STEPBIC mientras que el set obtenido con MxM se queda un poco por debajo aunque con una diferencia de apenas 0,003 respecto a las demás. De igual manera, en cuanto al AUC se observa un comportamiento muy similar aunque en este caso la diferencia entre MxM y los demás sets es ligeramente menor con apenas 0,0025 respecto a las demás. Al igual que para el dataset original, teniendo en cuenta que las diferencias en los resultados en F1-score como en AUC son minimas, se observa que el set obtenido con el metodo MxM contiene solo 7 variables, mientras que los demás contienen de 11 a 13 variables marcando una reducción significativa de la complejidad de los modelos a realizar, por tal motivo, teniendo en

cuenta el principio de parsimonia, se decide escoger el set de variables obtenidas del metodo MxM.

De esta manera las variables seleccionadas son:

- 'HrsCambBand': horas desde el ultimo cambio de bandas en el compresor.
- 'PresDifFl': Presión diferencial en el filtro del compresor
- 'TempCoj': Temperatura de los cojinetes del compresor
- 'TempDev'': Temperatura de devanado del motor.
- 'LC': Numero de cambios de carga desde el último fallo
- 'HorasMismaCarga': horas consecutivas de operación en el mismo nivel de carga desde el ultimo cambio de carga.
- 'Lavg': Carga promedio entre fallos

4.2.1 Resumen variables seleccionadas y excluidas

En la siguiente tabla se muestran las variables seleccionadas y excluidas para cada dataset.

Variables	Dataset Original	Dataset Balanceado
HrsTotOp_C	NO	NO
HrsCambBand	SI	SI
PresImp	NO	NO
PresInt	NO	NO
PresDifFl	SI	SI
TempCoj	SI	SI
TempEnt	NO	NO
TempDev	SI	SI
HorasMismaCarga	NO	SI
LC	SI	SI
Lavg	NO	SI
Dt	NO	NO
Dp	NO	NO

Imagen 25. Variables seleccionadas de acuerdo con el dataset usado.

Todas las variables seleccionadas para el dataset original fueron también seleccionadas para los datos balanceados, mientras que para los datos balanceados se agregaron las variables 'Lavg' y 'HorasMismaCarga'

4.3 Redes Neuronales

Una vez seleccionadas las variables que se van a utilizar, se realiza la investigación de los algoritmos de machine learning para encontrar el que mejor se ajuste al tipo de problema que se presenta en este estudio. Como se indica, el primer algoritmo que se trabaja es el de redes neuronales, el cual se tunea en R con la ayuda de la función *avnnet* de la librería *Caret*, la cual permite establecer una rejilla con los distintos parametros a probar. De esta manera se realiza con validación cruzada repetida de 4 folds y 5 repeticiones (Ver Anexo 15).

Uno de los parámetros a tunear es el número de nodos ocultos de la red, para lo cual se tiene en cuenta que en una red el número de parámetros es igual al h(k+1) + h + 1, donde h es el numero de nodos ocultos y k el número de nodos input. En este sentido,

es recomendable también establecer un número de parámetros que permita que cada parámetro tenga al menos 30 observaciones para evitar los problemas de sobreajuste.

Otros dos parámetros importantes a tunear es el *decay* o *learning rate*, el cual hace variar los pesos en la dirección de mejora de la función de pérdida y el número máximo de iteracciones, que permite establecer para cada combinación de *decay* y número de nodos ocultos, el número de veces que el algoritmo va a repetir el proceso de cálculo de los pesos.

Para este caso de estudio, tanto para los datos originales como para los balanceados se prueban valores de *decay* de 0.001, 0.01 y 0.1, mientras que para el número máximo de iteraciónes se prueban los valores 50, 100, 200, 500, 1000, 1500 y 2000.

Como el problema que se está atendiendo en este estudio es de clasificación con clases desbalanceadas, utilizar el *Accuracy* o la exactitud y la tasa de fallos para decidirse por el mejor modelo puede ser engañoso, por lo cual se utilizará la metrica F1-Score priorizando la capacidad del modelo para detectar los casos positivos y reducir los falsos positivos. De manera complementaria, se utiliza también el AUC.

El número de nodos a probar se establece para cada dataset por separado ya que este cálculo tiene en cuenta el número de observaciones disponibles.

De esta manera, para el dataset orginal, teniendo en cuenta que este dataset tiene 39302 observaciones, de las cuales 3146 pertenecen a la clase objetivo y se han seleccionado 5 variables input, manteniendo por lo menos 40 observaciones para la clase objetivo, se tendrían máximo 78 parámetros. Como N°Parámetros=h(k+1)+h+1, con k=5 (variables), se tendría h=11 nodos ocultos máximo.

De esta manera se tunea la red con los siguientes valores:

- Número de nodos: 3, 7 y 11
- Decay: 0.01, 0.1 y 0.001
- Iteraciones: 50, 100, 200, 500, 1000, 1500 y 2000

Para un total de 63 combinaciones. En la siguiente gráfica, se pueden ver los resultados del proceso de tuneo con estos parámetros:

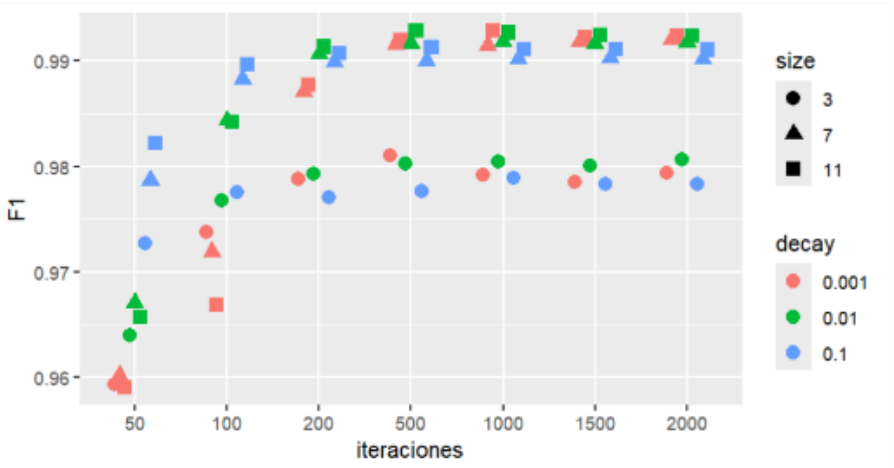


Imagen 26. F1 para la definición de los parámetros óptimos de red neuronal del dataset original.

Se puede identificar que para los diferentes números de nodos, después de 500 iteraciones el valor de F1 empieza a disminuir levemente. Por lo cual se puede seleccionar la red con 11 nodos y decay 0.01 como la mejor para este caso, reflejando un valor de F1 igual 0.9928, de AUC 0.78, Recall 0.9994 y Precisión 0.9862.

Por otra parte, para el dataset balanceado y teniendo en cuenta que este tiene 28314 observaciones, de las cuales 9438 pertenecen a la clase objetivo y se han seleccionado 7 variables input, manteniendo por lo menos 40 observaciones para la clase objetivo, se tendrían máximo 235 parámetros.

Como N°Parámetros=h(k+1)+h+1, con k=7 (variables), se tendría h=26 nodos ocultos máximo. Sin embargo, con el proposito de evitar el sobre ajuste y el consumo excesivo de recursos computacionales, se toma desición de tunear inicialmente hasta 15 nodos.

De esta manera se tunea la red con los siguientes valores:

- Número de nodos: 5, 10 y 15
- Decay: 0.01, 0.1 y 0.001
- Iteraciones: 50, 100, 200, 500, 1000, 1500 y 2000

Para un total de 63 combinaciones. En la siguiente gráfica, se pueden ver los resultados del proceso de tuneo con estos parámetros:

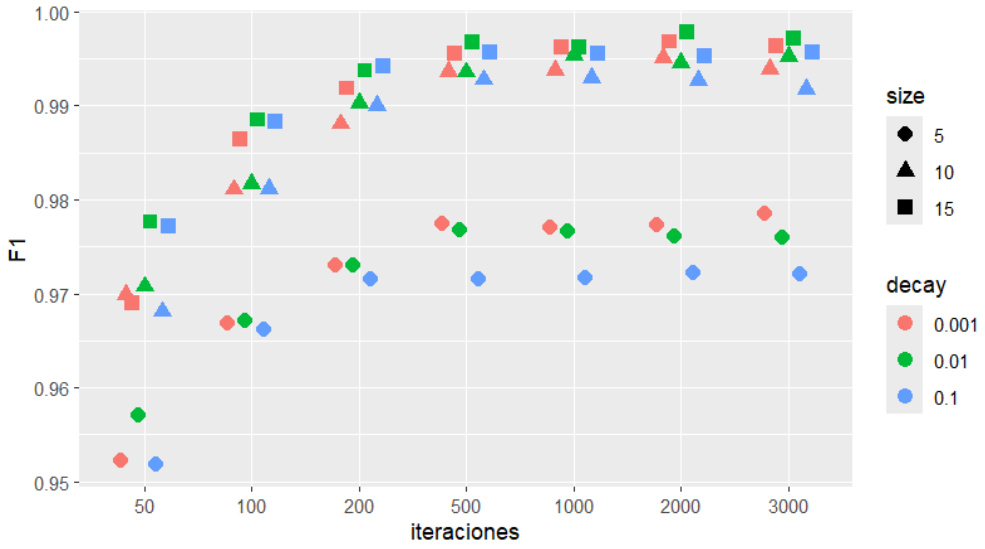


Imagen 27. F1 para la definición de los parámetros óptimos de red neuronal del dataset balanceado.

Al igual que para el dataset original, para el dataset balanceado se puede identificar que para los diferentes números de nodos, después de 500 iteraciones el valor de F1 empieza a disminuir levemente. Por lo cual, se puede seleccionar la red con 15 nodos y decay 0.01 como la mejor para este caso, reflejando un valor de F1 igual 0.9967, de AUC 0.8165, Recall 0.9986 y Precisión 0.9948.

4.4 Bagging

Para este algortimo basado en arboles de clasificación que busca mejorar las desventajas inherentes a estos últimos promediando el resultado de muchos arboles,

es importante tunear el número de arboles a combinar y el tamaño de estos arboles que se van a “remuestrear”.

En primer lugar, para tunear el número de arboles en R se utiliza la funcion *randomForest* del paquete del mismo nombre, el cual permite observar como varia el error a medida que aumenta el número de arboles. Para este caso, es importante definir los parámetros de complejidad de los árboles a construir estableciendo el número de variables a utilizar, el tamaño máximo de nodos finales y el tamaño inicial de las submuestras.

Luego de determinado el número de árboles, se procede a tunear el tamaño de las muestras a través de la validación cruzada repetida de diferentes árboles cada uno con un tamaño de muestra distinto, utilizando la función *rf* del paquete *Caret* de R (Ver Anexo 16).

Debido a que cada dataset tiene un número de observaciones y variables distintos, el tuneo de estos parámetros se realiza por separado para cada uno. Para este caso, se decide utilizar el muestreo con reemplazamiento para los dos datasets.

Para el dataset original, en primer lugar, se estudia el número de árboles a promediar en este dataset en el rango de 0 a 5000 arboles, estableciendo los siguientes parametros:

- mtry: número de variables input = 5
- nodesize: tamaño máximo de nodos finales = 10
- sampsize: tamaño de muestra de cada muestra bagging = 300
- replace = TRUE (con reemplazamiento)

En la siguiente gráfica se muestran los resultados del proceso de tuneo descrito.

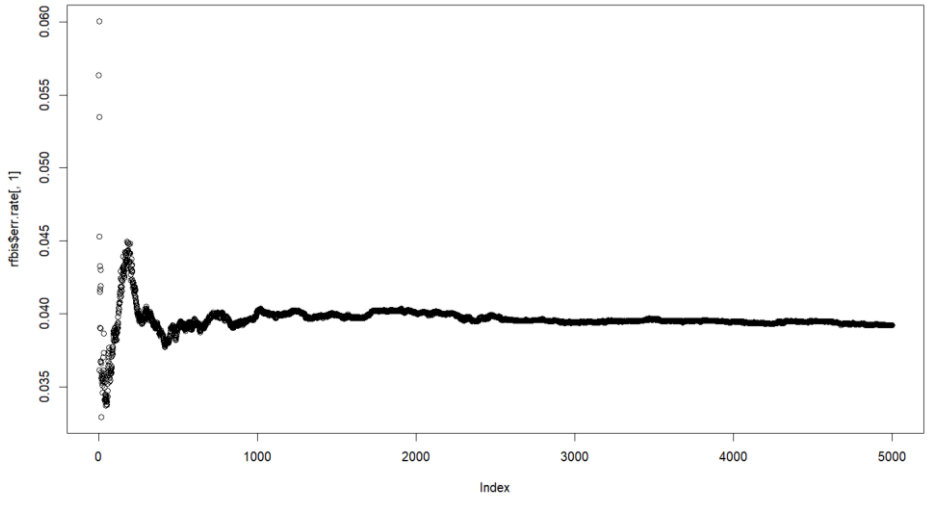


Imagen 28. Early stopping para modelo Bagging con el dataset original.

Lo que se busca es encontrar el número de arboles a partir de cual es estabiliza el error, lo cual se puede identificar a partir de 2500 árboles.

Una vez establecido el numero de arboles en 2500 se compara con validación cruzada repetida modelos de Bagging variando el tamaño de las muestras en 50, 100, 500, 1000,

5000 y 10000. Los resultados de las metricas F1 y AUC se pueden ver en las siguientes graficas.

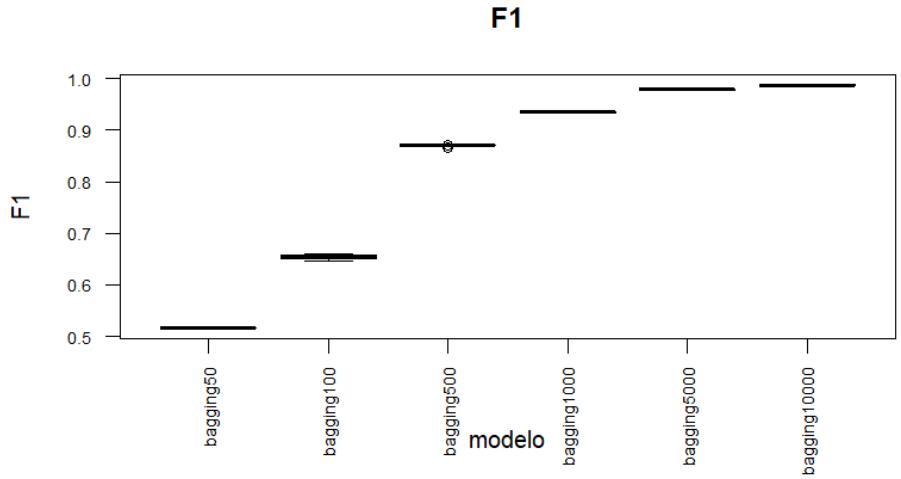


Imagen 29. F1 de los modelos Bagging probados en el dataset original.

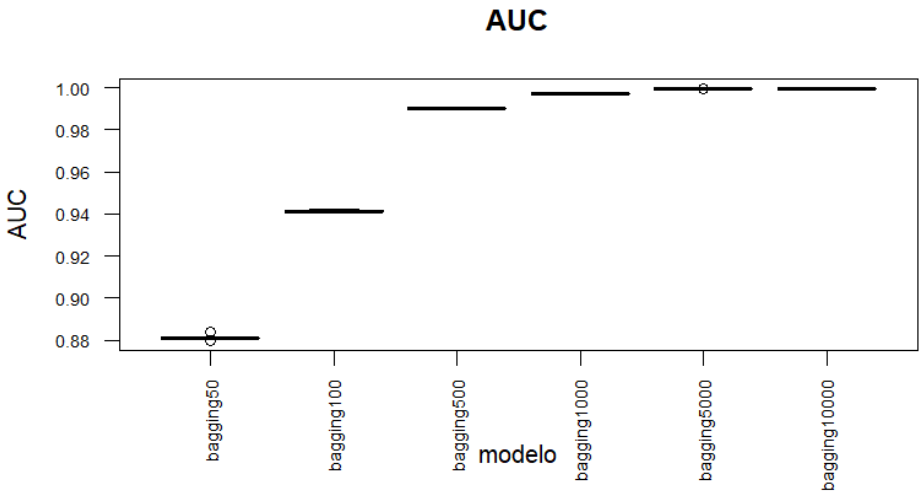


Imagen 30. AUC de los modelos Bagging probados en el dataset original.

Se identifica que con 5000 y 10000 se obtienen los mejores resultados en ambas metricas, por lo que se repite la grafica solo con estos dos tamaños para tomar una desición.

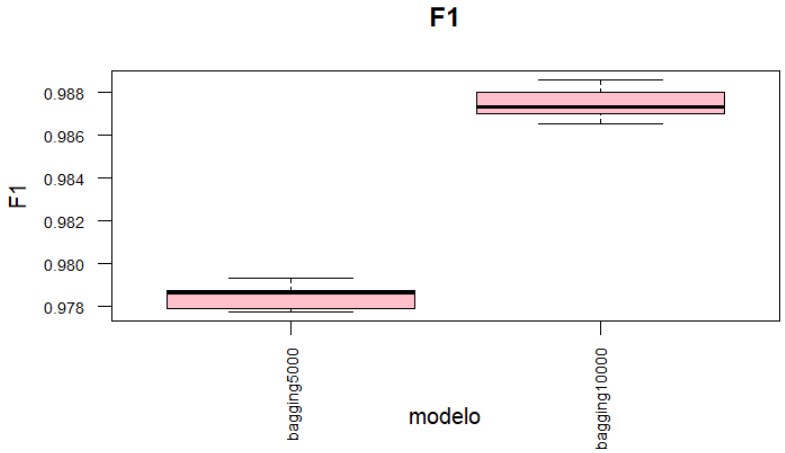


Imagen 31. F1 para los mejores modelos Bagging con el set original.

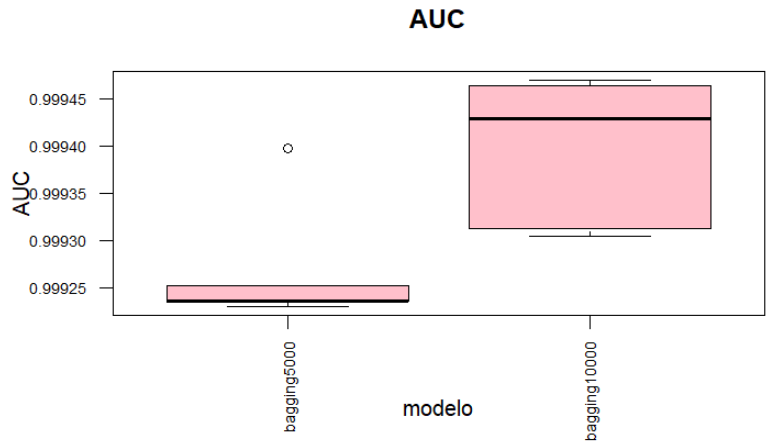


Imagen 32. AUC para los mejores modelos Bagging con el set original.

Aunque con 10000 se observan valores más altos, se puede identificar una mayor varianza en F1 y sobretodo en AUC que con 5000, lo cual puede ser síntoma de sobreajuste por lo que se decide trabajar con el tamaño de muestra en 5000. Por lo tanto, para sintetizar se establece que los mejores parámetros para trabajar este algoritmo con este dataset son:

- mtry: número de variables input = 5.
- nodesize: tamaño máximo de nodos finales = 10.
- sampsize: tamaño de muestra de cada muestra bagging = 5000.
- ntree: número de arboles a muestrear = 2500.
- replace = TRUE (con reemplazamiento).

Para el dataset balanceado, al igual que con el anterior dataset, se prueba el número de árboles a promediar en el rango de 0 a 5000 árboles, estableciendo los siguientes parametros:

- mtry: número de variables input = 7
- nodesize: tamaño máximo de nodos finales = 10
- sampsize: tamaño de muestra de cada muestra bagging = 300
- replace = TRUE (con reemplazamiento)

En la siguiente gráfica se muestran los resultados del proceso de tuneo descrito.

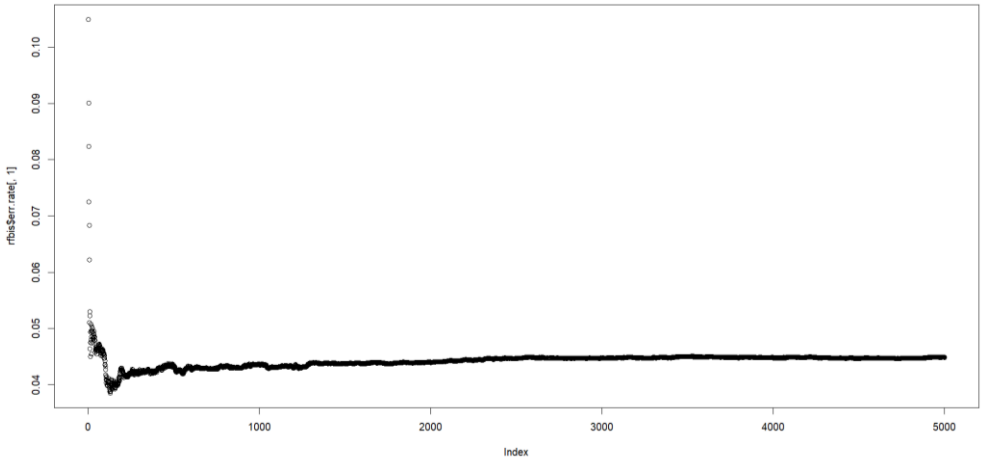


Imagen 33. Early stopping para modelo Bagging con el dataset balanceado.

En este caso, el número de arboles a partir de cual es estabiliza el error se identifica a partir de 2500 árboles.

Posteriormente, manteniendo el número de arboles en 2500, se comparan con validación cruzada repetida modelos de Bagging variando el tamaño de las muestras en 50, 100, 500, 1000, 5000 y 10000. Los resultados de las metricas F1 y AUC se pueden ver en las siguientes gráficas.

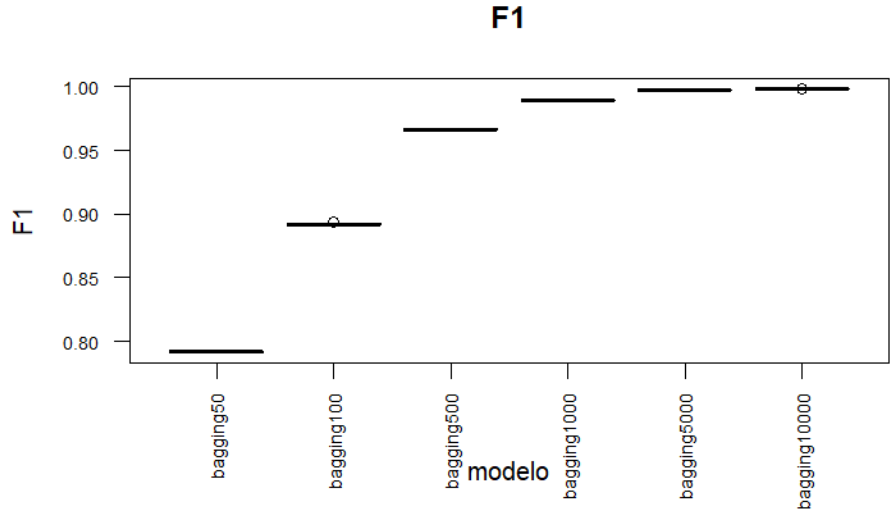


Imagen 34. F1 para los modelos Bagging con el set balanceado.

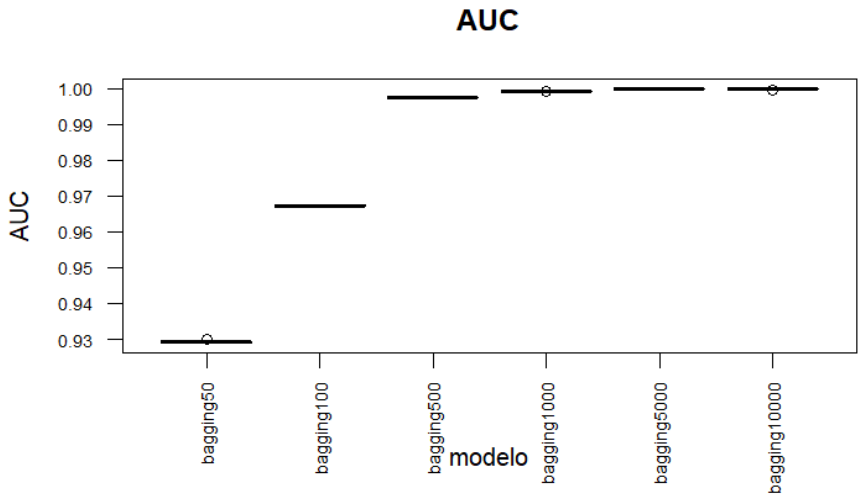


Imagen 35. AUC para los modelos Bagging con el set balanceado.

Se identifica que desde 500 hasta 10000 se obtienen los mejores resultados en ambas metricas, por lo que se repite la grafica solo con estos dos tamaños para tomar una desición.

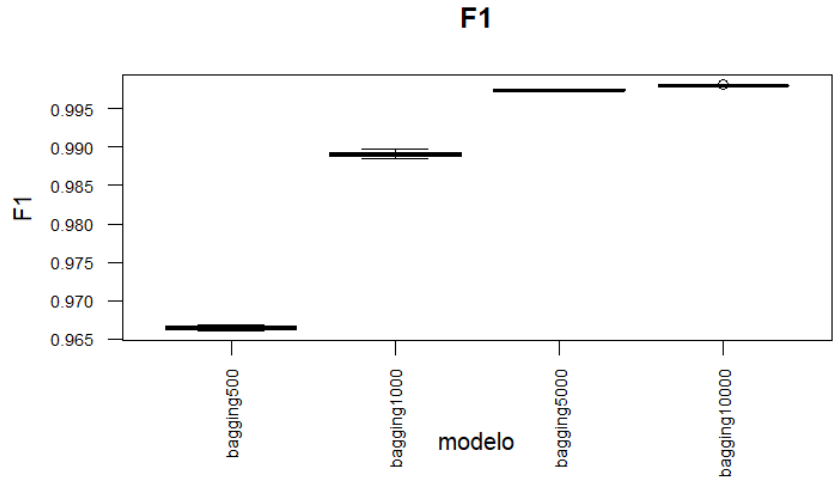


Imagen 36. F1 para los mejores modelos Bagging con el set balanceado.

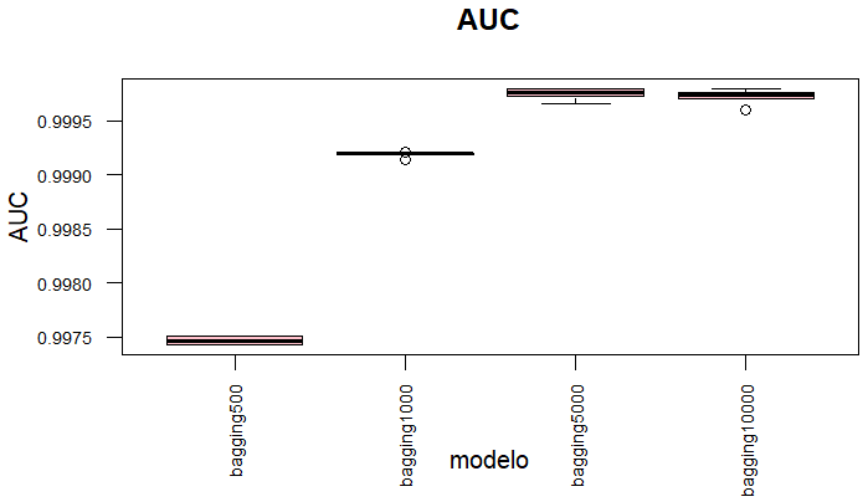


Imagen 37. AUC para los mejores modelos Bagging con el set balanceado.

A pesar de que con 5000 y 10000 se obtienen valores más altos de F1, se alcanza una ligera mayor varianza en AUC que con 1000, lo cual puede ser sintoma de sobreajuste.

Adicionalmente, al tratarse solo de milésimas de diferencia y para evitar el sobreajuste, se decide trabajar con el tamaño de muestra en 1000. Por lo tanto, para sintetizar se establece que los mejores parametros para trabajar este algoritmo con este dataset son:

- mtry: número de variables input = 7.
- nodesize: tamaño máximo de nodos finales = 10.
- sampsize: tamaño de muestra de cada muestra bagging = 1000.
- ntree: número de arboles a muestrear = 2500.
- replace = TRUE (con reemplazamiento).

4.5 Random Forest

Para este algoritmo también basado en árboles, al ser una modificación del Bagging que busca incorporar aleatoriedad en las variables utilizadas para segmentar cada nodo del árbol, es importante tunear ese número de variables.

Para este caso, se prueban para los dos datasets en estudio el rango del set de variables seleccionadas, es decir, desde 1 hasta 5 para el dataset original y desde 1 hasta 7 para el dataset balanceado. Utilizando la función *train()* y el metodo *'rf'* del paquete *Caret* de R, se permite realizar este tuneo, manteniendo los demás parámetros en los valores establecidos durante el tuneo del Bagging (Ver Anexo 17).

Por consiguiente, para el datset original, al tener este 5 variables seleccionadas, se tunea con validación cruzada el mtry en el rango de 1 a 5 con los siguientes parametros predefinidos en Bagging.

- nodesize: tamaño máximo de nodos finales = 10.
- sampsize: tamaño de muestra de cada muestra bagging = 5000.
- ntree: número de arboles a muestrear = 2500.
- replace = TRUE (con reemplazamiento).

La siguiente grafica permite ver los resultados del proceso de tuneo:

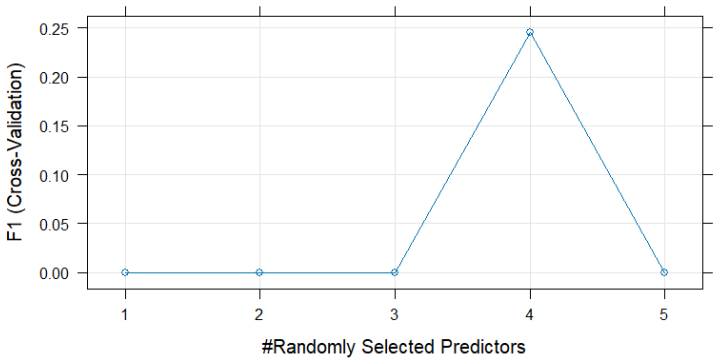


Imagen 38. Gráfico de número de variables vs. F1 en dataset original.

Se observa como con 4 variables se obtiene el mejor resultados para F1 por lo que se establece que los mejores parametros para trabajar este algoritmo con este dataset son:

- mtry: número de variables input = 4.
- nodesize: tamaño máximo de nodos finales = 10.
- sampsize: tamaño de muestra de cada muestra bagging = 5000.
- ntree: número de arboles a muestrear = 2500.
- replace = TRUE (con reemplazamiento).

Del mismo modo, para el dataset balanceado, como este tiene 7 variables seleccionadas preliminarmente, se tunea con validación cruzada el mtry en el rango de 1 a 7 con los siguientes parametros predefinidos anteriormente en Bagging.

- nodesize: tamaño máximo de nodos finales = 10.
- sampsize: tamaño de muestra de cada muestra bagging = 1000.
- ntree: número de arboles a muestrear = 2500.
- replace = TRUE (con reemplazamiento).

La siguiente gráfica permite ver los resultados del proceso de tuneo:

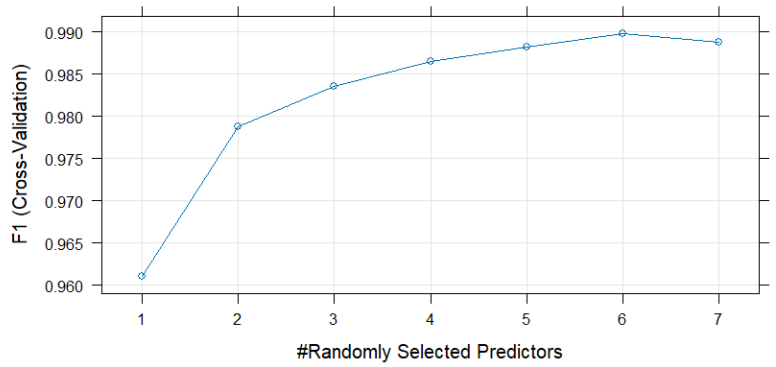


Imagen 39. Gráfico de número de variables vs. F1 en dataset balanceado.

Donde se observa que con 4 variables se obtiene el mejor resultado para F1 por lo que se establece que los mejores parámetros para trabajar este algoritmo con este dataset son:

- mtry: número de variables input = 6.
- nodesize: tamaño máximo de nodos finales = 10.
- sampsize: tamaño de muestra de cada muestra bagging = 1000.
- ntree: número de arboles a muestrear = 2500.
- replace = TRUE (con reemplazamiento).

4.6 Gradient Boosting

Al igual que Bagging y Random Forest, este algoritmo también basado en arboles requiere definir los parámetros de complejidad de los árboles como el número mínimo de observaciones en los nodos o el número de árboles a sortear, sin embargo, la principal novedad en Gradient Boosting es la inclusión de una constante de regularización que permite ir modificando ligeramente las predicciones iniciales cada vez, intentando ir minimizando los residuos en la dirección de decrecimiento. Cuanto más alta, más rápido converge, pero demasiado alta es poco fino. Por el contrario, si es muy baja requiere muchas iteraciones para converger (Portela, 2025).

Por tal motivo, es importante tunear de buena manera este parametro llamado *shrinkage* para construir un modelo de Gradient Boosting. La función *train()* y el metodo *'gbm'* del paquete *Caret* de R, permite realizar este tuneo probando distintos valores de *shrinkage,* número de observaciones en el nodo y número de arboles a sortear (Ver Anexo 18).

Para los dos datasets en estudio, se prueban inicialmente los siguientes valores en validación cruzada:

- Shrinkage: constante de regularización = 0.2, 0.1, 0.05, 0.01 y 0.001
- n.minobsinnode: número minimo de observaciones en nodo = 5, 10 y 20.
- n.trees: número de iteraciones = 100, 500, 1000, 2000, 3000

En la siguiente grafica se pueden observar los resultados del proceso de tuneado de estos parámetros en Gradient Boosting para el dataset original.

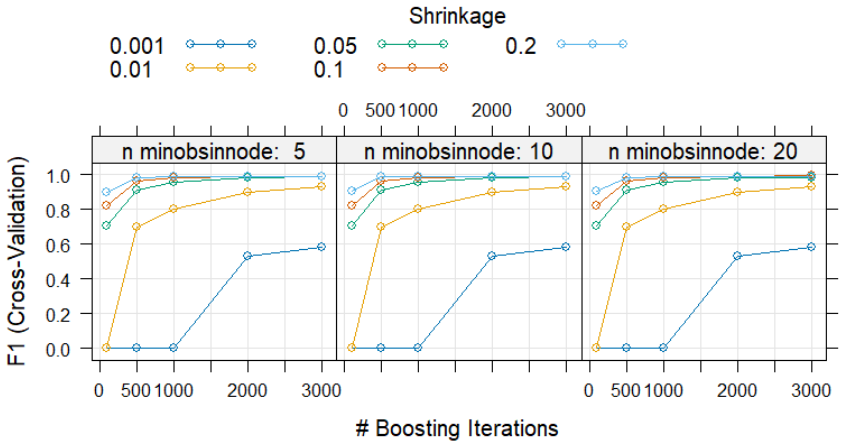


Imagen 40. Identificación de parámetros óptimos para Gradient Boosting con dataset original.

Donde se puede observar que el mejor resultado para el F1-Score para este dataset es 0.9929 con los siguientes parametros:

- Shrinkage: 0.1
- n.minobsinnode: 20.
- n.trees = 3000

Para el dataset balanceado, se pueden observar los resultados del tuneo en la siguiente grafica.

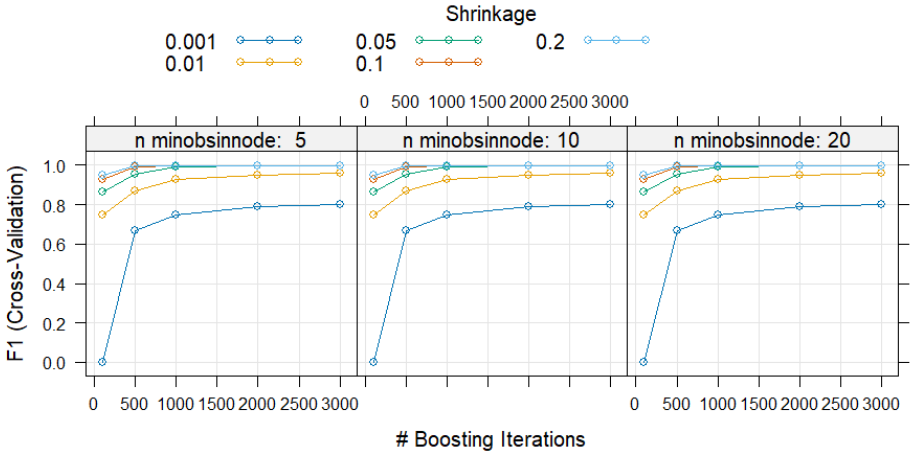


Imagen 41. Identificación de parámetros óptimos para Gradient Boosting con dataset balanceado.

Donde se puede observar que el mejor resultado para el F1-Score para este dataset es 0.9984 con los siguientes parametros:

- Shrinkage: 0.2
- n.minobsinnode: 10.
- n.trees = 2000

4.7 Support Vector Machine

A continuación, se estiman los modelos con este algoritmo utilizando tres kernels distintos en R con el paquete *Caret* (Ver Anexo 19).

4.7.1 Kernel lineal

Este modelo solo necesita determinar la constante de regularización C y se debe tener en cuenta que a menor C se permiten mayores residuos otorgando al modelo más 'permiso' para fallar y a mayor C pasa lo contrario, por lo que implica menor sesgo y mayor sobreajuste.

Para los dos datasets en el estudio, se prueban y comparan con validación cruzada, los siguientes valores de C: 0.01, 0.05, 0.1, 0.5, 1, 5 y 10.

En la siguiente gráfica se pueden observar los resultados del proceso de tuneado de C para el dataset original.

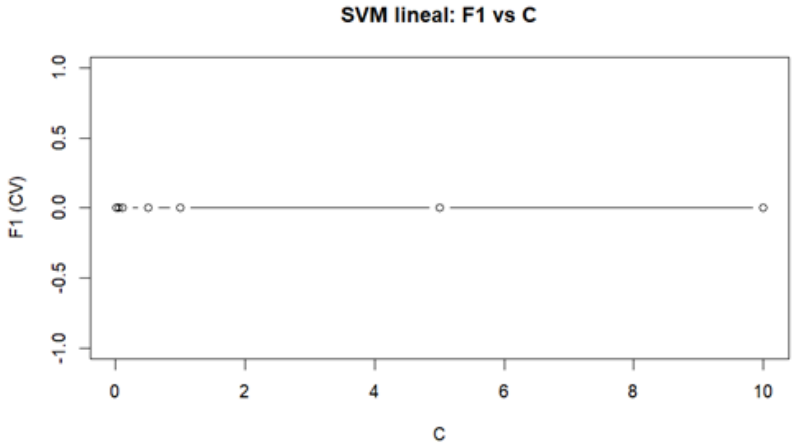


Imagen 42. Identificación de valor óptimo de C con Kernel lineal para el dataset original.

Donde se puede observar que para este caso el kernel lineal no es capaz de separar la clase positiva y por tal motivo se obtiene el valor de F1=0.

Para el dataset balanceado, se pueden observar los resultados del proceso de tuneado de C en la siguiente gráfica.

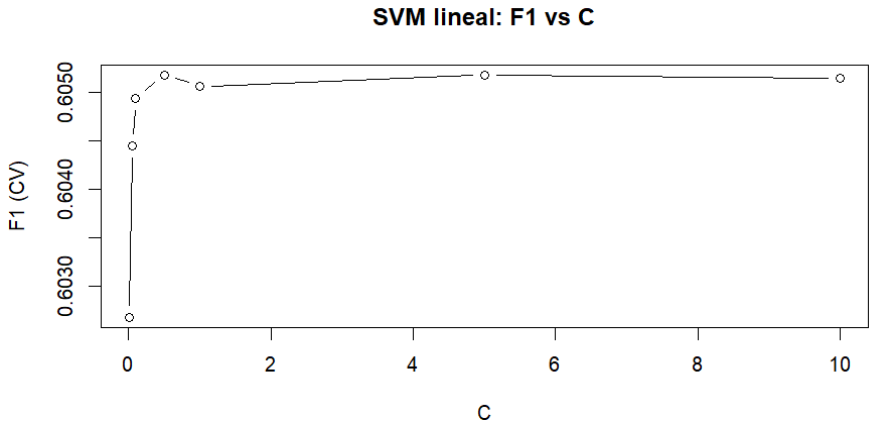


Imagen 43. Identificación de valor óptimo de C con Kernel lineal para el dataset balanceado.

Por lo tanto, para este dataset el mejor resultado para el F1-Score es 0.6051 con C igual 0.5.

4.7.2 Kernel polinomial

Este modelo, además de definir una constante de regularización C, también debe establecer el grado del polinomio y la escala. En el paquete de R a veces falla o se hace excesivamente lento a pesar de usar ayudas como el paquete *doParallel* ya que es una cuestión del algoritmo de optimización númerica utilizado para construir el SVM. Por lo tanto, se toma la desición de mantener el grado fijo en 2 y la escala en 0.1 para tunear el C.

Para los dos datasets en el estudio, se prueban y comparan con validación cruzada, los siguientes valores de C: 0.01, 0.05, 0.1, 0.5, 1, 5 y 10.

En las siguientes gráficas se pueden observar los resultados del proceso de tuneado de C para el dataset original. En este caso, se repitió el ejercicio ampliando el rango de valores a probar hasta 50.

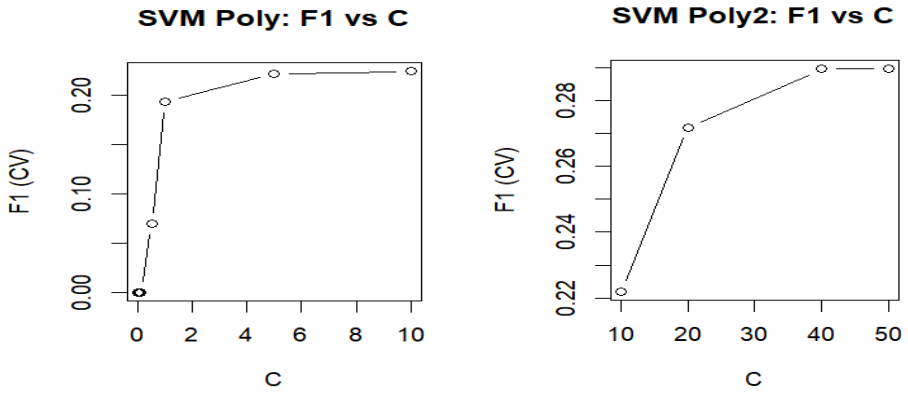


Imagen 44. Identificación de valor óptimo de hiperparámetros con Kernel polinomial para el dataset original.

Por lo tanto, para este caso el mejor resultado de F1 0.2895 se obtiene con C = 40, grado del polinomio 2 y la escala en 0.1

Por otro lado, para el dataset balanceado se pueden observar los resultados del proceso de tuneado de C en la siguiente gráfica.

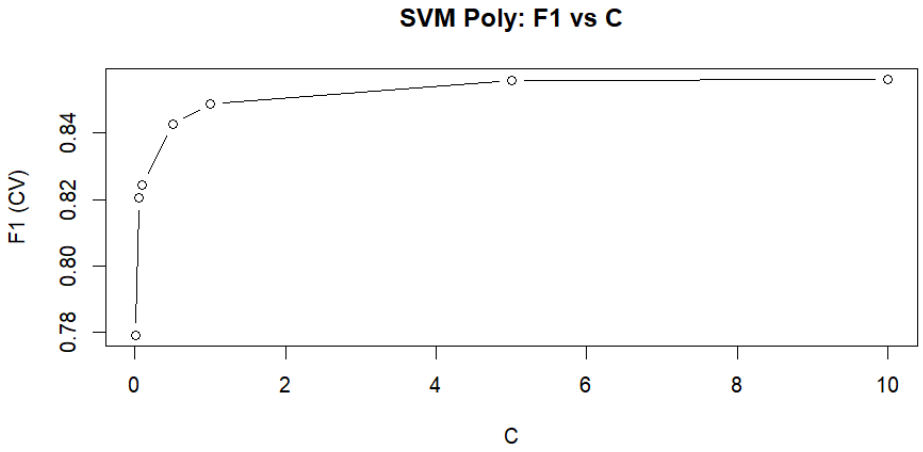


Imagen 45. Identificación de valor óptimo de hiperparámetros con Kernel lineal para el dataset balanceado.

Donde se puede observar que el mejor resultado de F1 es 0.8562 y se obtiene con C = 10, grado del polinomio 2 y la escala en 0.1.

4.7.3 Kernel radial

A diferencia de los anteriores kernels, para el kernel radial se hace necesario definir, además definir el parametro C, el parametro sigma. Por este motivo, para ambos datasets en estudio se prueban y comparan con validación cruzada los siguientes valores de C y sigma:

- C: 0.5, 1, 2, 5, 10 y 30
- Sigma: 0.0001, 0.005, 0.01, 0.05

Para un total de 24 combinaciones.

En las siguientes gráficas se pueden observar los resultados del proceso de tuneado de C y sigma para el dataset original.

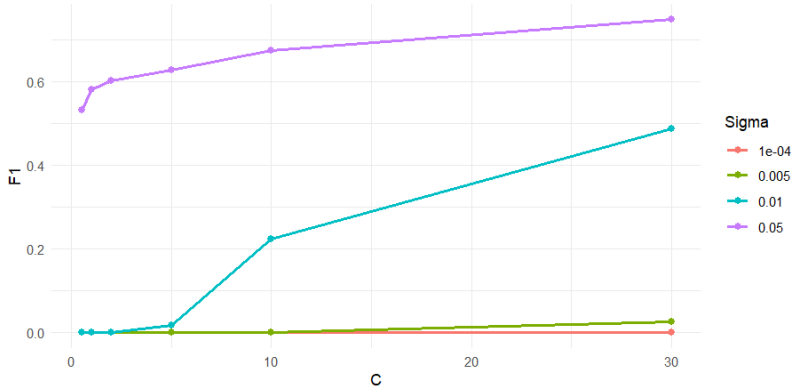


Imagen 46. Identificación de valor óptimo de hiperparámetros con Kernel radial para el dataset original.

Por lo tanto, para este caso el mejor resultado de F1 0.7491 se obtiene con C = 30 y sigma = 0.05.

Por otra parte, para el dataset balanceado, en la siguiente gráfica se pueden observar los resultados del proceso de tuneado de C y sigma.

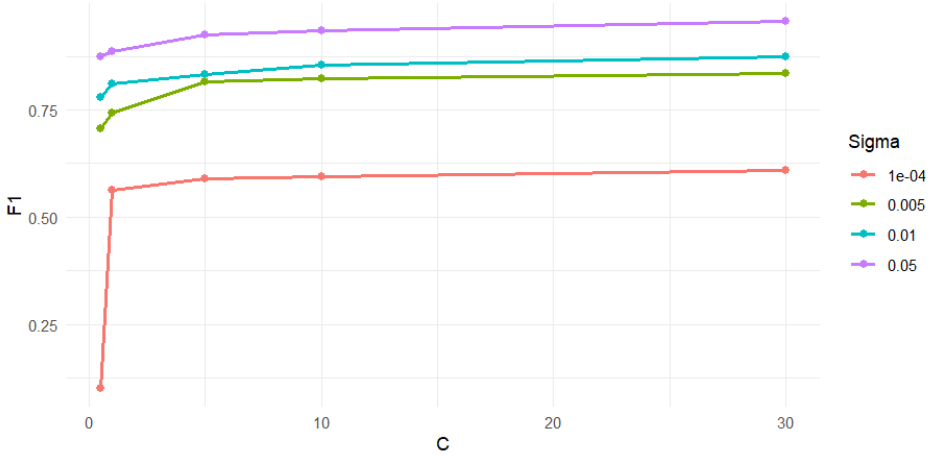


Imagen 47. Identificación de valor óptimo de hiperparámetros con Kernel radial para el dataset balanceado.

Donde se puede observar que el mejor resultado de F1 es 0.9559 y se obtiene con C = 30 y sigma = 0.05.

4.8 Evaluación de modelos

Se realiza la comparación de los siete tipos de algoritmos de machine learning y de la Regresión Logistica via validación cruzada repetida de 4 grupos y 5 repeticiones utilizando como metrica principal el F1-Score y complementando la comparación con el AUC. En la siguiente grafica se presentan los resultados incluyendo los modelos realizados con el dataset original y con el balanceado con SMOTE.

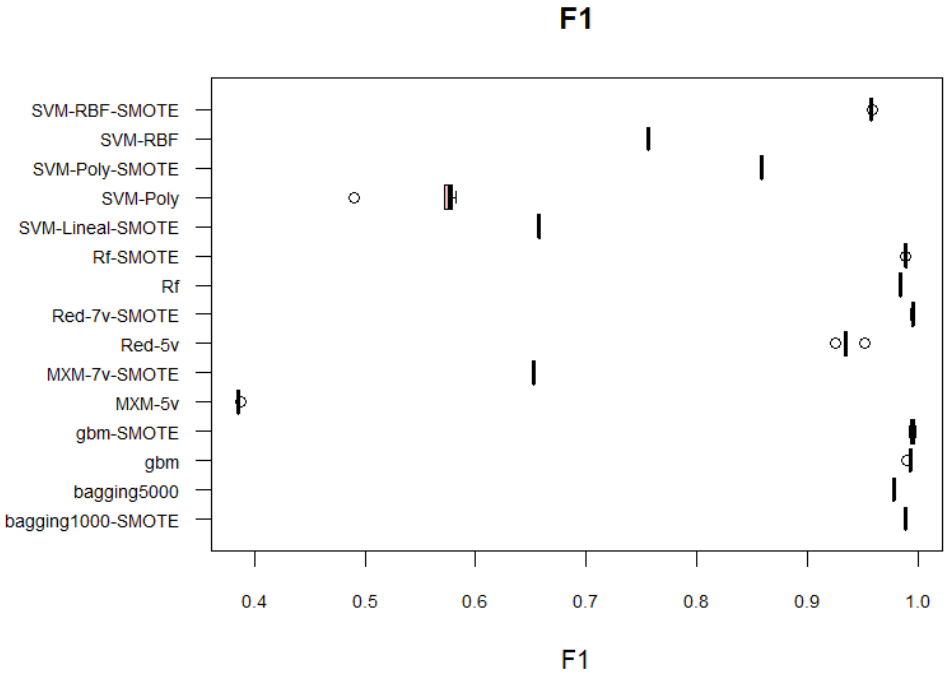


Imagen 48. F1 de los diferentes modelos evaluados.

Se observa, como es de esperarse, que el menor resultado de F1 tanto para el dataset original como para el dataset balanceado se obtiene con el modelo de Regresión Logística realizado con las variables seleccionadas bajo el método MxM, con la excepción para el SVM-Lineal-SMOTE que obtiene un resultado muy cercano a la Regresión Logistica MxM-SMOTE. Esto confirma que los datos trabajados no presentan una relación lineal.

Al comparar en primera instancia solo los modelos entrenados con el dataset original, se puede observar que los algortimos que presentan mejor resultado en F1 son los basados en árboles estando en primer lugar el Gradient Boosting (gbm), luego el Random Forest (Rf), seguido del Bagging (bagging5000), dejando de cuarto a la Red Neuronal (Red-5v) y el mejor SVM, el radial (SVM-RBF).

Del mismo modo, al comparar solo los modelos entrenados con el dataset balanceado, se puede observar que el algortimo que mejor resultado presenta en F1 es el Gradient Boosting (gbm-SMOTE), pero esta vez seguido de la Red Neuronal (Red-7v-SMOTE) y luego el Random Forest (Rf-SMOTE), seguido del Bagging (bagging1000-SMOTE), dejando de quinto el mejor SVM, el radial (SVM-RBF-SMOTE).

Realizando una comparación entre los resultados obtenidos en cada algoritmo para los dos dataset, encontramos que para todos los casos, los algoritmos entrenados con el dataset balanceado alcanzaron mejores resultados de F1 que los entrenados con el dataset original, indicando que, aunque las diferencias parezcan minimas, se pueden alcanzar mejores niveles de detección de la clase positiva en cuanto a sensiibilidad y precisión al trabajar dataset balanceados.

En la siguiente tabla se pueden ver los valores de las métricas antes y después del balanceo de clases.

Modelo	F1-Score	AUC	Recall	Precision
Antes SMOTE				
Regresión Logística	0.3857	0.8064	0.4815	0.3219
Red Neuronal	0.9365	0.9686	0.9409	0.9323
Bagging	0.9784	0.9992	0.9776	0.9794
Random Forest	0.9848	0.9988	0.9828	0.9940
Gradient Boosting	0.9928	0.9991	0.9909	0.9945
SVM-Polinomial	0.5599	0.8434	0.5467	0.5770
SVM-Radial	0.7563	0.9699	0.6325	0.9405
Después SMOTE				
Regresión Logística	0.6523	0.8211	0.7663	0.5703
Red Neuronal	0.9953	0.9993	0.9960	0.9942
Bagging	0.9890	0.9991	0.9844	0.9937
Random Forest	0.9894	0.9994	0.9863	0.9926
Gradient Boosting	0.9955	0.9967	0.9970	0.9940
SVM-Lineal	0.6576	0.8072	0.6636	0.6519
SVM-Polinomial	0.8591	0.9512	0.8785	0.8407
SVM-Radial	0.9581	0.9946	0.9485	0.9679

Tabla 10 Resumen de las métricas de evaluación de rendimiento de los modelos antes y después de aplicar SMOTE.

En donde se puede observar que en todas las métricas se obtienen mejores resultados para todos los algortimos despues de balancear las clases a excepción del AUC del Bagging y del Gradient Boosting y de la Precisión del Random Forest y el Gradient Boosting.

4.9 Selección del modelo ganador

Para seleccionar el modelo ganador, se comparan de manera más detallada los resultados de los mejores modelos de cada dataset y se presentan en la siguiente gráfica.

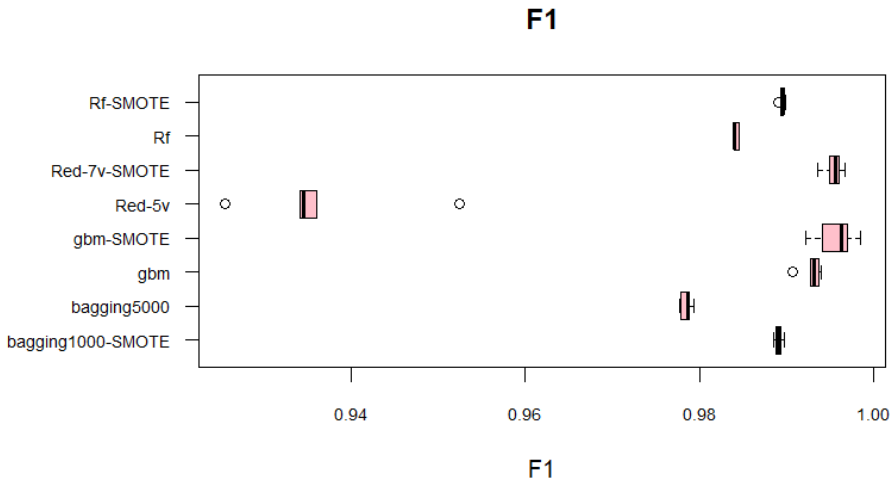


Imagen 49. Mejores modelos basados en F1.

De esta manera se recalca que el modelo de Gradient Boosting gbm-SMOTE tiene un mejor resultado de F1 con 0.9955, sin embargo, se observa una mayor varianza que los demás modelos, mientras que la Red Neuronal Red-7v-SMOTE tiene un resultado ligeramente inferior en F1 con 0.9953, pero con un menor varianza.

Una ventaja que ofrece el modelo de Gradient Boosting frente al modelo de Red Neuronal es la capacidad de interpretar la importancia de las variables predictoras para clasificar la variable objetivo. Por ejemplo, en este caso de estudio, se puede observar la influencia relativa de las variables en el modelo gbm-SMOTE en la siguiente gráfica:

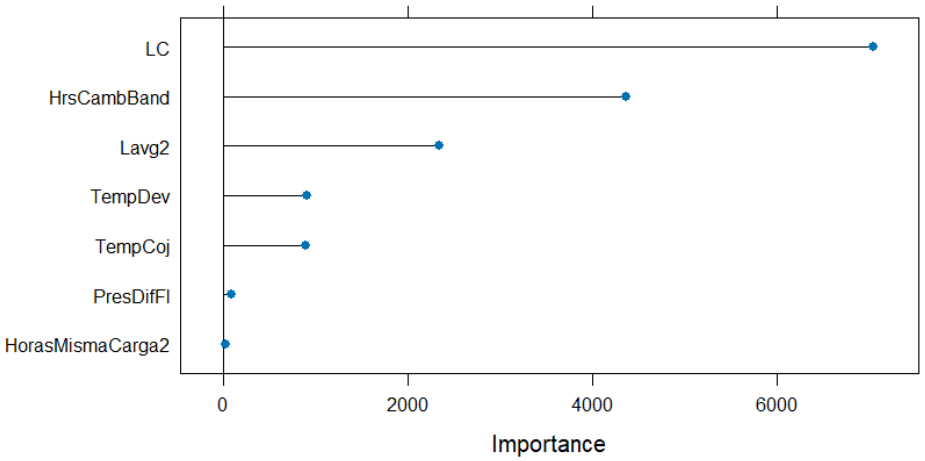


Imagen 50. Importancia de variables del modelo ganador.

Gracias a la cual se puede interpretar que la variables principales para detectar fugas de aceite son los cambios de carga entre fallos, las horas desde el último cambio de bandas y la carga promedio entre fallos, indicando que el desgaste mecánico y el esfuerzo acumulado que reflejan estas variables son muy influyentes en la aparición de

fugas de aceite. De igual manera, muestra que las variables térmicas como la temperatura de devanado del motor y la temperatura de los cojinetes del compresor juegan un rol secundario aunque importante señalando que el estrés termico, que puede generar desgaste de lubricantes y sellos, afecta complementariamente la aparición de fugas de aceite.

Gracias a esta posibilidad de interpretación y teniendo en cuenta que, a pesar de la varianza, se obtiene un mejor resultado de F1, se escoje como modelo ganador el Gradient Boosting entrenado con los datos balanceados (gbm-SMOTE).

Para validar el modelo, se dividen los datos en train y test y se ajusta el modelo con datos train y se elabora una matriz de confusión con las prediciones sobre los datos test, la cual se puede ver a continuación (Ver Anexo 20):

```
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  4713    5
       Yes    6 2354
```

Imagen 51. Matriz de confusión del modelo ganador.

Con los cuales se obtienen las siguientes metricas:

- Sensibilidad (Recall): 0.9979
- Especificidad: 0.9987
- Precisión: 0.9975
- F1-Score: 0.9954

Las cuales son consistentes con los resultados mostrados anteriormente en la tabla.

5. CONCLUSIONES

En conclusión, durante el desarrollo de este trabajo se ha generado un caso práctico de aplicación de metodologías de ciencia de datos para la gestión del mantenimiento en plantas industriales, construyendo un modelo de clasificación para la detección de fugas de aceite en un compresor de BOG de GNL a tráves de Regresión Logistica y siete técnicas de machine learning afrontando los retos de partir de datos en bruto y analizando el impacto del desbalance de las clases en el rendimiento de los modelos.

Gracias al análisis del “Registro de Incidencias” se pudo concluir que los fallos relacionados al sistema de lubricación del compresor son los que han afectado en mayor medida al funcionamiento del compresor y en este sentido, es la fuga de aceite el tipo de fallo que mayor ha afectado este sistema con el 27% de las incidencias reportadas y con 15% de las horas de paradas generadas por incidencias.

Se construyó el dataset de trabajo cruzando la información del “Registro de Incidencias” con las del “Registro de Variables de Operación” a través de la creación en este último, de la variable objetivo ‘Fuga_Aceite’ tomando del “Registro de Incidencias” la información correspondiente a las horas donde se presentaron las fugas, logrando clasificar el dataset en dos clases, con fuga de aceite y normal, permitiendose así realizar todos los análisis y estudios propuestos.

Durante la depuración de los datos se realizaron tareas como la eliminación de variables obsoletas, cambios en la nomenclatura y tipología de las variables, corrección de variables con anomalías, eliminación de datos con errores de lectura, clasificación de observaciones según el nivel de carga, clasificación de observaciones según la presencia de fuga de aceite, tratamiento y eliminación de datos atípicos, se han incluido en el estudio nuevos descriptores que luego resultaron ser muy importantes para el rendimiento de los modelos como los cambios de carga entre fallos y la carga promedio entre fallos. Durante el mismo proceso, se ha realizado un análisis de la correlación entre las variables predictoras, retirando variables altamente correladas y creando otros descriptores como el diferencial de presión, diferencial de temperatura y horas desde el ultimo cambio de bandas logrando evitar problemas de colinealidad y reducir la dimensionalidad del análisis.

Asimismo, se ha realizado un análisis descriptivo a través de gráficos exploratorios para identificar el poder discriminativo de las variables predictoras, por el cual se logró identificar preliminarmente que variables como ‘LC’, ‘HrsCambBand’, ‘TempDev’ y ‘HrsTotOp_C’ tienen una relevancia estadística media-alta que podrían aportar una señal importante para la discriminación de las fugas de aceite, mientras que variables como ‘Lavg’, ‘Dt’, ‘TempCoj’ y ‘Dp’ pueden aportar una señal más débil y complementaria para la detección de las fugas de aceite.

Buscando sortear el fuerte desbalance de clases existente en el dataset original de solo 8% para la clase positiva, se aplicó la técnica de SMOTE para balancear de mejor manera los datos, alcanzando una relación de 2:1 logrando una disminución significativa del desbalance de las clases.

Con la aplicación de los métodos de selección de variables, se logró escoger un set de variables para entrenar los modelos en cada dataset, coincidiendo en ambos datasets las variables *Número de cambios de carga desde el último fallo* (LC)*, horas desde el último cambio de bandas* (HrsCambBand)*, Presión diferencial en el filtro del compresor* (PresDifFl)*, Temperatura de los cojinetes del compresor* (TempCoj)y *Temperatura de devanado del motor* (TempDev), evidenciando su relevancia para discriminar las fugas, tal como se había concluido con el análisis exploratorio inicial. Cabe resaltar que en el set utilizado para el dataset balanceado, se seleccionó además las variables *horas misma carga* (HorasMismaCarga) y *Carga promedio entre fallos* (Lavg)*.*

De esta manera, se realizó una extensa investigación sobre diferentes tipos de algoritmos de machine learning: Red Neuronal, Random Forest, Gradient Boosting y Support Vector Machine con Kernel lineal, polinomial y radial de la cual se concluyó que la aplicación de la técnica SMOTE puede mejorar los resultados en las métricas de desempeño como F1-Score en los diferentes algoritmos para los modelos de clasificación. Asimismo, se concluyó que el mejor modelo fue el Gradient Boosting entrenado con los datos balanceados con un F1 de 0.9955 y un AUC de 0.9967 alcanzando una mejora significativa frente a la Regresión Logística, afirmando que la relación entre las variables predictoras y la variable objetivo no es de orden lineal. Estos resultados son consistentes con los estudios existentes, por ejemplo, con el de Mahale que alcanzó un F1-Score de 0,9954 después de aplicar SMOTE y con el de Hidalgo-Mompeán que alcanzó un F1-Score de 0,996 en su mejor modelo.

Estos resultados de los diferentes algoritmos permiten afirmar la idoneidad del Gradient Boosting para enfrentar este tipo de problemas ya que, a diferencia del Random Forest y el Bagging, alcanza mayores niveles de precisión al ajustar el modelo finamente con la ayuda de la constante de regularización enfocándose en disminuir los errores de clasificación, lo cual es vital al trabajar con datos desbalanceados. Asimismo, la ventaja del Gradient Boosting frente a la Red Neuronal, para este tipo de problemas, radica en la mejor interpretabilidad que ofrece ya que las Redes Neuronales, a pesar de alcanzar rendimientos muy buenos tienen muy baja interpretabilidad, lo cual es muy importante en el escenario de este estudio para poder implementar medidas de gestión de mantenimiento asociadas.

Por consiguiente, gracias a la selección del modelo Gradient Boosting como el mejor, se ha logrado identificar la relevancia de las variables para la clasificación de las fugas de aceite, encontrando que el *Número de cambios de carga desde el último fallo* (LC) es la de mayor influencia relativa, seguida de las *horas desde el último cambio de bandas* (HrsCambBand), *Carga promedio entre fallos* (Lavg), *Temperatura de devanado del motor* (TempDev) y *Temperatura de los cojinetes del compresor* (TempCoj).

A pesar de las limitaciones existentes como la poca frecuencia de fallos, datos ruidosos y necesidad de validación en la planta, un modelo de clasificación como el alcanzado se puede utilizar como uno de los primeros pasos para la implementación de una estrategia de CBM, ya que, a partir de estos resultados, sería posible desarrollar un modelo predictivo orientado a anticipar la ocurrencia del modo de fallo, permitiendo de esta forma planificar las actividades de mantenimiento.

Bajo este enfoque, gracias a los altos y consistentes niveles de precisión alcanzados se puede concluir que es viable optimizar la gestión de activos industriales como los compresores BOG, bajo un marco de mantenimiento inteligente apoyado en la aplicación de algoritmos de machine learning.

BIBLIOGRAFÍA

Aggarwal, C. C. (2023). *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing. https://doi.org/10.1007/978-3-031-29642-0

Al Ghafri, S. Z. S., Perez, F., Heum Park, K., Gallagher, L., Warr, L., Stroda, A., Siahvashi, A., Ryu, Y., Kim, S., Kim, S. G., Seo, Y., Johns, M. L., & May, E. F. (2021). Advanced boil-off gas studies for liquefied natural gas. *Applied Thermal Engineering*, *189*, 116735. https://doi.org/10.1016/j.applthermaleng.2021.116735

Campbell, C., & Ying, Y. (2011). *Learning with Support Vector Machines*. Springer International Publishing. https://doi.org/10.1007/978-3-031-01552-6

Chapman, P., Clinton, J., Kever, R., & Khabaza, T. (2000). *Metodología Crisp-DM 1—Step-by-step data mining guide*. Studocu. https://www.studocu.com/cl/document/pontificia-universidad-catolica-de-chile/business-analytics/metodologia-crisp-dm-1/8584569

Guillén, A. J., Crespo, A., Gómez, J. Fco., & Sanz, M. D. (2016). A framework for effective management of condition based maintenance programs in the context of industrial development of E-Maintenance strategies. *Computers in Industry*, *82*, 170-185. https://doi.org/10.1016/j.compind.2016.07.003

Hidalgo-Mompeán, F., Gómez Fernández, J. F., Cerruela-García, G., & Crespo Márquez, A. (2021). Dimensionality analysis in machine learning failure detection models. A case study with LNG compressors. *Computers in Industry*, *128*, 103434. https://doi.org/10.1016/j.compind.2021.103434

Hwang, H. J., Lee, J. H., Hwang, J. S., & Jun, H. B. (2018). A study of the development of a condition-based maintenance system for an LNG FPSO. *Ocean Engineering*, *164*, 604-615. https://doi.org/10.1016/j.oceaneng.2018.07.004

Jang, N., Shin, M. W., Choi, S. H., & Yoon, E. S. (2011). Dynamic simulation and optimization of the operation of boil-off gas compressors in a liquefied natural gas gasification plant. *Korean Journal of Chemical Engineering*, *28*(5), 1166-1171. https://doi.org/10.1007/s11814-010-0487-x

Kleinbaum, D. G., & Klein, M. (2002). *Logistic Regression: A Self-Learning Text* (Second Edition). Springer-Verlag New York, Inc. https://doi.org/10.1007/b97379

Kumar, A., & Jain, M. (2020). *Ensemble Learning for AI Developers: Learn Bagging, Stacking, and Boosting Methods with Use Cases*. Apress. https://doi.org/10.1007/978-1-4842-5940-5

Mahale, Y., Kolhar, S., & More, A. S. (2025). Enhancing predictive maintenance in automotive industry: Addressing class imbalance using advanced machine learning techniques. *Discover Applied Sciences*, *7*(4), 340. https://doi.org/10.1007/s42452-025-06827-3

Orrù, P. F., Zoccheddu, A., Sassu, L., Mattia, C., Cozza, R., & Arena, S. (2020). Machine Learning Approach Using MLP and SVM Algorithms for the Fault Prediction of a Centrifugal Pump in the Oil and Gas Industry. *Sustainability*, *12*(11), 4776. https://doi.org/10.3390/su12114776

Portela, J. (2025). *Apuntes Machine Learning redes neuronales*.

Rodrigues, J. M. F., Cardoso, P. J. S., Monteiro, J., Lam, R., Krzhizhanovskaya, V. V., Lees, M. H., Dongarra, J. J., & Sloot, P. M. A. (Eds.). (2019). *Computational Science – ICCS 2019: 19th International Conference, Faro, Portugal, June 12–14, 2019, Proceedings, Part III* (Vol. 11538). Springer International Publishing. https://doi.org/10.1007/978-3-030-22744-9

Ye, A., & Wang, Z. (2023). *Modern Deep Learning for Tabular Data: Novel Approaches to Common Modeling Problems*. Apress. https://doi.org/10.1007/978-1-4842-8692-0

Zhou, Z.-H. (2021). *Machine Learning*. Springer Singapore. https://doi.org/10.1007/978-981-15-1967-3

ANEXOS

Código en R

1. Código Depuración

```
# Instalar paquetes necesarios
if(!require(dplyr)){install.packages("dplyr")}
# Cargar librerías necesarias.
library(dplyr)
#Se decide retirar del conjunto de datos las variables que no tienen información relevante según criterios de negocio.
# Eliminar las variables especificadas
DATOS <- DATOS %>%
  dplyr::select(-`entrada a relicuador linea mantenimiento en frio a compresor GB-113...52`,
        -`GN a compresor...50`,
        -`GN a compresor...51`,
        -`COLECTOR BOG `,
        -`PARO `,
        -`Load Demand Signal from DCS`,
        -`Alimentación de BOG a Compresor `,
        -`LINEA RECIRCULACION `,
        -`XV salida compresor `,
        -`entrada a relicuador linea mantenimiento en frio a compresor...53`)

# Verificar que las columnas fueron eliminadas
colnames(DATOS)

#Para tener un mejor manejo de la información, se decide renombrar las variables acortando los nombres:
DATOS <- DATOS %>% dplyr::rename(
  `FechaHora` = `FECHA_HORA`,
  `CambBand1` = `Cambio_bandas_cilindro_1_etapa_1_puntual`,
  `CambBand2` = `Cambio_bandas_cilindro_1_etapa_2_puntual`,
  `CambBand3` = `Cambio_bandas_cilindro_2_etapa_1_puntual`,
  `CambBand4` = `Cambio_bandas_cilindro_2_etapa_2_puntual`,
  `CargaTrab` = `Carga_trabajo_puntual`,
  `Estado` = `Estado_puntual`,
  `HrsCamb1` = `Horas_desde_ultimo_cambio_banda_cilindro_1_etapa_1`,
  `HrsCamb2` = `Horas_desde_ultimo_cambio_banda_cilindro_1_etapa_2`,
  `HrsCamb3` = `Horas_desde_ultimo_cambio_banda_cilindro_2_etapa_1`,
  `HrsCamb4` = `Horas_desde_ultimo_cambio_banda_cilindro_2_etapa_2`,
  `HrsGranMt` = `Horas_desde_ultimo_gran_mtto`,
  `HrsTotOp` = `Horas_totales_operacion`,
  `Intensid` = `Intensidad_puntual`,
  `Potencia` = `Potencia_puntual`,
  `PresAsp1` = `Presion_aspiracion_puntual`,
  `PresAsp2` = `Presion_aspiracion_puntual_2`,
  `PresAsp3` = `Presion_aspiracion_puntual_3`,
  `PresDifFl` = `Presion_diferencial_filtro_puntual`,
  `PresImp1` = `Presion_impulsion_puntual`,
  `PresImp2` = `Presion_impulsion_puntual_2`,
  `PresImp3` = `Presion_impulsion_puntual_3`,
  `PresImp4` = `Presion_impulsion_puntual_4`,
```

```
  `PresInt1` = `Presion_interetapa_puntual`,
  `PresInt2` = `Presion_interetapa_puntual_2`,
  `PresInt3` = `Presion_interetapa_puntual_3`,
  `TempAsp` = `Temperatura_aspiracion_puntual`,
  `TempCojM` = `Temperatura_cojinete_motor_puntual`,
  `TempCoj1` = `Temperatura_cojinetes_compresor_zona_1_puntual`,
  `TempCoj2` = `Temperatura_cojinetes_compresor_zona_2_puntual`,
  `TempCoj3` = `Temperatura_cojinetes_compresor_zona_3_puntual`,
  `TempCoj4` = `Temperatura_cojinetes_compresor_zona_4_puntual`,
  `TempCoj5` = `Temperatura_cojinetes_compresor_zona_5_puntual`,
  `TempDevA1` = `Temperatura_devanado_motorA_puntual`,
  `TempDevA2` = `Temperatura_devanado_motorA_puntual_2`,
  `TempDevB1` = `Temperatura_devanado_motorB_puntual`,
  `TempDevB2` = `Temperatura_devanado_motorB_puntual_2`,
  `TempDevC1` = `Temperatura_devanado_motorC_puntual`,
  `TempDevC2` = `Temperatura_devanado_motorC_puntual_2`,
  `TempEnt1` = `Temperatura_entre_etapas_puntual`,
  `TempEnt2` = `Temperatura_entre_etapas_puntual_2`,
  `TempImp1` = `Temperatura_impulsion_puntual`,
  `TempImp2` = `Temperatura_impulsion_puntual_2`,
)

# Verificar que los nombres fueron modificados
colnames(DATOS)

# Se revisa el tipo de dato de cada variables
str(DATOS)

# Se observa que no todas las variables categóricas están como factores.
# Fuerzo a que las variables categóricas leidas como numéricas sean factores
DATOS[,c(2,3,4,5,7)] <- lapply(DATOS[,c(2,3,4,5,7)], factor)

#Se verifica que ahora sean factores
str(DATOS)

#Se realiza exploración general de los estadisticos descriptivos de las variables
summary(DATOS)

#Se detectan valores a eliminar en las variables que significan errores en la señal de
lectura
# Expresiones a convertir en NA
valores_a_na <- c("Shutdown", "Interface down", "Comm Fail", "Bad")

# Guardamos las clases originales de las columnas
tipos_originales <- sapply(DATOS, class)

# Convertimos todo a texto para reemplazar las expresiones
DATOS[] <- lapply(DATOS, function(col) {
  col_txt <- as.character(col)
  col_txt[col_txt %in% valores_a_na] <- NA
  return(col_txt)
})

# Convertimos las columnas de nuevo a su tipo original cuando sea posible
DATOS[] <- Map(function(col, tipo) {
```

```
  if (tipo[1] == "numeric") {
    suppressWarnings(as.numeric(col))
  } else if (tipo[1] == "factor") {
    factor(col)
  } else if (tipo[1] == "POSIXct") {
    as.POSIXct(col)
  } else {
    col
  }
}, DATOS, tipos_originales)

#Verificar si se eliminaron los datos equivocados
summary(DATOS)
str(DATOS)

# Vector con los valores a buscar
valores_a_buscar <- c("Shutdown", "Interface down", "Comm Fail", "Bad")

# Buscar coincidencias en todo el dataset
coincidencias <- sapply(DATOS, function(col) {
  any(grepl(paste(valores_a_buscar, collapse = "|"), col, ignore.case = FALSE))
})

# Mostrar nombres de las columnas que contienen alguna de esas expresiones
names(coincidencias[coincidencias])

# Convertimos todo el dataset a carácter para forzar la búsqueda
DATOS_txt <- as.data.frame(lapply(DATOS, as.character), stringsAsFactors = FALSE)

# Buscar coincidencias exactas
coincidencias <- sapply(DATOS_txt, function(col) {
  any(col %in% valores_a_buscar)
})

names(coincidencias[coincidencias])

#Verificar si se eliminaron los datos equivocados
summary(DATOS)
str(DATOS)

#---Estadisticos descriptivos de variables de mediciones puntuales del compresor----
summary(DATOS)
##Se crea una variable tipo factor de la carga de trabajo para facilitar el analisis de
esta variable
DATOS$CargaTrab_M <- cut(
  DATOS$CargaTrab,
  breaks = c(-Inf, 0.1, 35, 60, 85, Inf),
  labels = c("0%", "25%", "50%", "75%", "100%"),
  right = TRUE,   # Intervalos cerrados a la derecha: (a, b]
  include.lowest = TRUE
)
#Se grafica la distribución de frecuencias para CargaTrab_M
funModeling::freq(data.frame(CargaTrab_M = DATOS$CargaTrab_M))

#Se grafica la distribución de frecuencias de Estado
```

```
funModeling::freq(data.frame(Estado = DATOS$Estado))
```

2. Código Creación CargaTrab_M2

```
#Se crea la variable CargaTrab_M2 con los siguientes criterios de negocio:
#- 0% si Carga NO es NA y Estado es "PARO DISPONIBLE" o "MANT. PREVENTIVO"
o "MANT. CORRECTIVO" o "PARO INCIDENCIA"
#- 25% si CargaTrab está en el rango (0.1,35] y  Estado es igual a MARCHA
#- 50% si CargaTrab está en el rango (35,60] y  Estado es igual a MARCHA
#- 75% si CargaTrab está en el rango (60,85] y  Estado es igual a MARCHA
#- 100% si CargaTrab está en el rango >85 y  Estado es igual a MARCHA
#- NA para las demás combinaciones posibles
# Crear variable vacía con NA como base
CargaTrab_M2 <- rep(NA, nrow(DATOS))
# Condición para 0%: Estado en paro/mantenimiento Y CargaTrab no es NA
cond_0 <- !is.na(DATOS$CargaTrab) & DATOS$Estado %in% c("PARO
DISPONIBLE", "MANT. PREVENTIVO", "MANT. CORRECTIVO", "PARO
INCIDENCIA")
CargaTrab_M2[cond_0] <- "0%"
# Condición para 25%: (0.1, 35] y Estado == "MARCHA"
cond_25 <- DATOS$CargaTrab > 0.1 & DATOS$CargaTrab <= 35 & DATOS$Estado
== "MARCHA"
CargaTrab_M2[cond_25] <- "25%"
# Condición para 50%: (35, 60] y Estado == "MARCHA"
cond_50 <- DATOS$CargaTrab > 35 & DATOS$CargaTrab <= 60 & DATOS$Estado
== "MARCHA"
CargaTrab_M2[cond_50] <- "50%"
# Condición para 75%: (60, 85] y Estado == "MARCHA"
cond_75 <- DATOS$CargaTrab > 60 & DATOS$CargaTrab <= 85 & DATOS$Estado
== "MARCHA"
CargaTrab_M2[cond_75] <- "75%"
# Condición para 100%: > 85 y Estado == "MARCHA"
cond_100 <- DATOS$CargaTrab > 85 & DATOS$Estado == "MARCHA"
CargaTrab_M2[cond_100] <- "100%"
# Convertir a factor ordenado
DATOS$CargaTrab_M2 <- factor(CargaTrab_M2, levels = c("0%", "25%", "50%",
"75%", "100%"))

#Se grafica la distribución de frecuencias para CargaTrab_M2
funModeling::freq(data.frame(CargaTrab_M2 = DATOS$CargaTrab_M2))

#Se genera la distribución de frecuencias de todas las variables cuantitativas
# Generar un histograma por cada variable cuantitativa

# Seleccionar solo las variables cuantitativas
vars_cuantitativas <- names(DATOS)[sapply(DATOS, is.numeric)]

for (var in vars_cuantitativas) {
  print(
    ggplot(DATOS, aes_string(x = var)) +
      geom_histogram(bins = 100, fill = "lightblue", color = "black") +
      labs(title = paste("Distribución de", var),
           x = var,
           y = "Frecuencia") +
```

```
    theme_minimal()
  )
}

# Graficar cada variable cuantitativa en el tiempo
for (var in vars_cuantitativas) {
  print(
    ggplot(DATOS, aes_string(x = "FechaHora", y = var)) +
      geom_line(color = "steelblue") +
      labs(title = paste("Evolución en el tiempo de", var),
           x = "Fecha",
           y = var) +
      theme_minimal()
  )
}

#---Estadisticos descriptivos de variables de cambio de banda del compresor----
#Se grafica la distribución de frecuencias para CargaTrab_M
funModeling::freq(data.frame(CambBand1 = DATOS$CambBand1))
funModeling::freq(data.frame(CambBand2 = DATOS$CambBand2))
funModeling::freq(data.frame(CambBand3 = DATOS$CambBand3))
funModeling::freq(data.frame(CambBand4 = DATOS$CambBand4))

#---Estadisticos descriptivos de variables de tiempo del compresor----

DATOS$FechaHora <- DATOS_ORIGINAL$FECHA_HORA
str(DATOS$FechaHora)
# Mostrar las primeras 10 fechas con hora en formato completo
format(DATOS$FechaHora[1:10], "%Y-%m-%d %H:%M:%S")
head(DATOS$FechaHora, 10)
```

3. Código corrección HrsTotOp y creación HrsTotOp_C

```
#Correción de variable HrsTotOp
# Copiar la variable original
DATOS$HrsTotOp_C <- DATOS$HrsTotOp

# Definir fechas de inicio y fin del periodo de corrección
inicio <- as.POSIXct("2018-03-16 14:00:00")
fin    <- as.POSIXct("2020-11-24 16:00:00")

# Buscar posiciones (índices) de inicio y fin
pos_ini <- which(DATOS$FechaHora == inicio)
pos_fin <- which(DATOS$FechaHora == fin)

# Validar que ambos puntos existen
if (length(pos_ini) == 1 && length(pos_fin) == 1) {

  # Lista de estados que NO deben sumar
  estados_sin_suma <- c("PARO DISPONIBLE", "MANT. PREVENTIVO", "MANT.
CORRECTIVO", "PARO INCIDENCIA")

  # Aplicar la lógica fila por fila
  for (i in (pos_ini + 1):pos_fin) {
    estado <- DATOS$Estado[i]
```

```
    if (!is.na(estado) && estado == "MARCHA") {
      DATOS$HrsTotOp_C[i] <- DATOS$HrsTotOp_C[i - 1] + 0.95
    } else {
      DATOS$HrsTotOp_C[i] <- DATOS$HrsTotOp_C[i - 1]
    }
  }

} else {
  warning("No se encontraron las fechas de inicio o fin correctamente en el dataset.")
}

#Se observa nuevamente los descriptivos:
summary(DATOS)

#Grafica de distribución de frecuencias de HrsTotOp_C
ggplot(DATOS, aes(x = HrsTotOp_C)) +
  geom_histogram(fill = "lightblue", color = "black", bins = 60) +
  labs(
    title = "Distribución de HrsTotOp_C",
    x = "HrsTotOp_C",
    y = "Frecuencia"
  ) +
  theme_minimal()

#Grafica de evolución en el tiempo de HrsTotOp_C
ggplot(DATOS, aes(x = FechaHora, y = HrsTotOp_C)) +
  geom_line(color = "steelblue") +
  labs(
    title = "Evolución en el tiempo de HrsTotOp_C",
    x = "Fecha",
    y = "HrsTotOp_C"
  ) +
  theme_minimal()
```

4. Código corrección HrsGranMt y creación HrsGranMt_C

```
#Correción de variable HrsGranMt
# Copiar la variable original
DATOS$HrsGranMt_C <- DATOS$HrsGranMt

# Definir fechas de inicio y fin del periodo de corrección
inicio2 <- as.POSIXct("2018-03-16 21:00:00")
fin2    <- as.POSIXct("2020-11-24 19:00:00")

# Buscar posiciones (índices) de inicio y fin
pos_ini2 <- which(DATOS$FechaHora == inicio2)
pos_fin2 <- which(DATOS$FechaHora == fin2)

# Validar que ambos puntos existen
if (length(pos_ini2) == 1 && length(pos_fin2) == 1) {

  # Lista de estados que NO deben sumar
  estados_sin_suma <- c("PARO DISPONIBLE", "MANT. PREVENTIVO", "MANT.
CORRECTIVO", "PARO INCIDENCIA")
```

```
  # Aplicar la lógica fila por fila
  for (i in (pos_ini2 + 1):pos_fin2) {
    estado <- DATOS$Estado[i]

    if (!is.na(estado) && estado == "MARCHA") {
      DATOS$HrsGranMt_C[i] <- DATOS$HrsGranMt_C[i - 1] + 1.05
    } else {
      DATOS$HrsGranMt_C[i] <- DATOS$HrsGranMt_C[i - 1]
    }
  }

} else {
  warning("No se encontraron las fechas de inicio o fin correctamente en el dataset.")
}

#Se observa nuevamente los descriptivos:
summary(DATOS)

#Grafica de distribución de frecuencias de HrsTotOp_C
ggplot(DATOS, aes(x = DATOS$HrsGranMt_C)) +
  geom_histogram(fill = "lightblue", color = "black", bins = 60) +
  labs(
    title = "Distribución de DATOS$HrsGranMt_C",
    x = "DATOS$HrsGranMt_C",
    y = "Frecuencia"
  ) +
  theme_minimal()

#Grafica de evolución en el tiempo de DATOS$HrsGranMt_C
ggplot(DATOS, aes(x = FechaHora, y = DATOS$HrsGranMt_C)) +
  geom_line(color = "steelblue") +
  labs(
    title = "Evolución en el tiempo de DATOS$HrsGranMt_C",
    x = "Fecha",
    y = "DATOS$HrsGranMt_C"
  ) +
  theme_minimal()

#-----EXPLORACIÓN DE VARIABLES VS CARGA DE TRABAJO--------#

# Visualización de dos variables cuantitativas frente a la variable binaria
var_numeric <- names(Filter(is.numeric, DATOS))

CargaTrab_M21 <- DATOS$CargaTrab_M2
input_05 <- input_analisis_cuant %>%
  select(-varObjCat)

# Para cada variable cuantitativa, se generan los gráficos de boxplot y densidad frente
a la variable binaria
for (var in var_numeric) {
  #Crear el gráfico de cajas para la variable cuantitativa frente a la binaria
  print(boxplot_targetbinaria(DATOS[[var]], CargaTrab_M21, var))  # Gráfico de cajas
  print(hist_targetbinaria(DATOS[[var]], CargaTrab_M21, var))     # Gráfico de densidad
}
```

5. **Creación variable HorasMismaCarga2**

#Crear variable que cuente el numero de horas seguidas trabajando al mismo nivel de carga

```
#Creación HorasMismaCarga2
# ---- HorasMismaCarga2: ----
rle_carga <- rle(as.character(DATOS$CargaTrab_M2))

# Para cada secuencia, crea 1,2,3,... hasta la longitud de esa secuencia
HorasMismaCarga2 <- unlist(lapply(rle_carga$lengths, seq_len))

# Agregar como columna
DATOS <- DATOS %>%
  mutate(HorasMismaCarga2 = HorasMismaCarga2)

#Grafica de evolución en el tiempo de DATOS$HorasMismaCarga2
ggplot(DATOS, aes(x = FechaHora, y = DATOS$HorasMismaCarga2)) +
  geom_line(color = "steelblue") +
  labs(
    title = "Evolución en el tiempo de HorasMismaCarga2",
    x = "Fecha",
    y = "HorasMismaCarga2"
  ) +
  theme_minimal()
```

6. **Creación variable Fuga_Aceite**

```
#-----Cruzar la información de las Incidencias con los datos del compresor--------#
#Se anexa a DATOS una variable Fuga_Aceite binaria que indica la existencia o no del reporte de Fuga de Aceite.
DATOS$Fuga_Aceite <-DATOS_Incidencias_Fuga$`INCIDENCIA FUGA DE ACEITE`


# Se convierte a factor la nueva variable Fuga_Aceite NO=0, SI=1
DATOS[,c(50)] <- lapply(DATOS[,c(50)], factor)
levels(DATOS$Fuga_Aceite) <- c("0", "1")
summary(DATOS$Fuga_Aceite)
```

7. **Creación variables RBTF, LC y Lavg2**

#Se crean las variables
#RTBF: Tiempo de funcionamiento entre fallos (horas).
#Lc: Cambios de carga entre fallos. Esta variable considera cada cambio de carga en los compresores, incluyendo los arranques.
#Lavg2: Carga promedio entre fallos. Esta variable se calculó solo durante las horas de funcionamiento, por lo que las horas en espera no se consideraron ya que distorsionarían la información.

```
#Creación LC

# Crear vector de salida
LC <- integer(nrow(DATOS))
```

```
cambios <- 0
prev_carga <- NA
prev_fallo <- "0"  # Se asume que al inicio no hay fallo

for (i in seq_len(nrow(DATOS))) {
  carga_actual <- as.character(DATOS$CargaTrab_M2[i])  # Puede ser NA
  fallo_actual <- as.character(DATOS$Fuga_Aceite[i])   # "0" o "1"

  # Si la carga no es NA, se evalúa cambio
  if (!is.na(carga_actual)) {
    if (is.na(prev_carga)) {
      prev_carga <- carga_actual
    } else if (carga_actual != prev_carga) {
      cambios <- cambios + 1
      prev_carga <- carga_actual
    }
  }

  # Asignar el número de cambios actual
  LC[i] <- cambios

  # Si hubo una transición de fallo (de "1" a "0"), se reinicia el contador
  if (!is.na(fallo_actual) && fallo_actual == "0" && prev_fallo == "1") {
    cambios <- 0
  }

  # Actualizar estado anterior del fallo
  if (!is.na(fallo_actual)) {
    prev_fallo <- fallo_actual
  }
}

# Agregar la nueva variable al dataframe
DATOS$LC <- LC

#CREACIÓN Lavg2

DATOS <- DATOS %>%
  mutate(
    # Indicadores
    is_fail  = Fuga_Aceite %in% c(1, "1", TRUE),
    running  = !is.na(CargaTrab_M2_num) & CargaTrab_M2_num > 0,

    # Segmento que se reinicia SOLO cuando el fallo termina (1 -> 0)
    end_fail = dplyr::lag(is_fail, default = FALSE) & !is_fail,
    seg_end0 = cumsum(end_fail),

    # Carga contable para promedio (solo cuando corre)
    x = ifelse(running, CargaTrab_M2_num, NA_real_)
  ) %>%
  group_by(seg_end0) %>%
  mutate(
    # Acumulados solo contando cargas válidas (>0)
    cs = cumsum(replace(x, is.na(x), 0)),
    cc = cumsum(!is.na(x)),
```

```
    # Promedio acumulado dentro del segmento (incluye filas en fallo)
    Lavg2 = ifelse(cc > 0, cs / cc, NA_real_)
  ) %>%
  ungroup() %>%
  select(-is_fail, -running, -end_fail, -seg_end0, -x, -cs, -cc)

#CREACIÓN RTBF

RTBF <- numeric(nrow(DATOS))
contador <- 0
fuga <- as.character(DATOS$Fuga_Aceite)
funcionando <- !is.na(DATOS$CargaTrab_M2_num) & DATOS$CargaTrab_M2_num >
0

for (i in seq_along(fuga)) {

  # Siempre acumula cuando está funcionando
  if (funcionando[i]) {
    contador <- contador + 1
  }

  RTBF[i] <- contador

  # Reiniciar SOLO cuando pasamos de 1 a 0 (fin del fallo)
  if (i > 1 && fuga[i-1] == "1" && fuga[i] == "0") {
    contador <- 1       # Reinicia en 1
    RTBF[i] <- contador # Actualiza el valor de RTBF en la fila actual
  }
}

# Asignar al dataframe
DATOS$RTBF <- RTBF
```

8. **Separación de Dataset de acuerdo a la carga**

```
#SE SEPARAN LOS DATA SET POR MODO DE CARGA

DATOS_MARCHA <- DATOS[!is.na(DATOS$CargaTrab_M2) &
as.character(DATOS$CargaTrab_M2) != "0%", ]
DATOS_0    <- DATOS[!is.na(DATOS$CargaTrab_M2) & DATOS$CargaTrab_M2 ==
"0%", ]
DATOS_25   <- DATOS[!is.na(DATOS$CargaTrab_M2) & DATOS$CargaTrab_M2 ==
"25%", ]
DATOS_50   <- DATOS[!is.na(DATOS$CargaTrab_M2) & DATOS$CargaTrab_M2 ==
"50%", ]
DATOS_75   <- DATOS[!is.na(DATOS$CargaTrab_M2) & DATOS$CargaTrab_M2 ==
"75%", ]
DATOS_100  <- DATOS[!is.na(DATOS$CargaTrab_M2) & DATOS$CargaTrab_M2 ==
"100%", ]

#Se genera la distribución de frecuencias de todas las variables cuantitativas de
DATOS_MARCHA
# Seleccionar solo las variables cuantitativas de DATOS_MARCHA
```

```
vars_cuantitativas_M <- names(DATOS_MARCHA)[sapply(DATOS_MARCHA,
is.numeric)]

# Generar un histograma por cada variable cuantitativa de DATOS_MARCHA
for (var in vars_cuantitativas_M) {
  print(
    ggplot(DATOS_MARCHA, aes_string(x = var)) +
      geom_histogram(bins = 100, fill = "lightblue", color = "black") +
      labs(title = paste("Distribución de", var),
           x = var,
           y = "Frecuencia") +
      theme_minimal()
  )
}

# Graficar cada variable cuantitativa en el tiempo de DATOS_MARCHA
for (var in vars_cuantitativas_M) {
  print(
    ggplot(DATOS_MARCHA, aes_string(x = "FechaHora", y = var)) +
      geom_line(color = "steelblue") +
      labs(title = paste("Evolución en el tiempo de", var),
           x = "Fecha",
           y = var) +
      theme_minimal()
  )
}

#Se genera la distribución de frecuencias de todas las variables cuantitativas de
DATOS_100
# Seleccionar solo las variables cuantitativas de DATOS_100
vars_cuantitativas_100 <- names(DATOS_100)[sapply(DATOS_100, is.numeric)]

# Generar un histograma por cada variable cuantitativa de DATOS_100
for (var in vars_cuantitativas_100) {
  print(
    ggplot(DATOS_100, aes_string(x = var)) +
      geom_histogram(bins = 100, fill = "lightblue", color = "black") +
      labs(title = paste("Distribución de", var),
           x = var,
           y = "Frecuencia") +
      theme_minimal()
  )
}

# Graficar cada variable cuantitativa en el tiempo de DATOS_100
for (var in vars_cuantitativas_100) {
  print(
    ggplot(DATOS_100, aes_string(x = "FechaHora", y = var)) +
      geom_line(color = "steelblue") +
      labs(title = paste("Evolución en el tiempo de", var),
           x = "Fecha",
           y = var) +
      theme_minimal()
  )
}
```

```
#Resumen de los datasets creados

summary(DATOS_MARCHA$Fuga_Aceite)
summary(DATOS_0$Fuga_Aceite)
summary(DATOS_25$Fuga_Aceite)
summary(DATOS_50$Fuga_Aceite)
summary(DATOS_75$Fuga_Aceite)
summary(DATOS_100$Fuga_Aceite)

# Función para calcular proporción de "1"
proporcion_fuga <- function(data) {
  prop <- mean(data$Fuga_Aceite == "1", na.rm = TRUE)
  return(round(prop, 4))  # redondea a 4 decimales
}

# Calcular para cada subconjunto
prop_MARCHA <- proporcion_fuga(DATOS_MARCHA)
prop_0      <- proporcion_fuga(DATOS_0)
prop_25     <- proporcion_fuga(DATOS_25)
prop_50     <- proporcion_fuga(DATOS_50)
prop_75     <- proporcion_fuga(DATOS_75)
prop_100    <- proporcion_fuga(DATOS_100)

# Mostrar resultados
proporciones <- data.frame(
  Dataset = c("DATOS_MARCHA", "DATOS_0", "DATOS_25", "DATOS_50",
"DATOS_75", "DATOS_100"),
  Proporcion_Fuga = c(prop_MARCHA, prop_0, prop_25, prop_50, prop_75, prop_100)
)

print(proporciones)

#Graficos de dispersión vs HorasMismaCarga2
# Lista de variables a graficar contra HorasMismaCarga2
vars_y <- c("TempDevB2", "TempCojM", "Intensid", "TempCoj5", "PresInt2",
"PresImp4")

# Bucle para mostrar cada gráfico por separado
for (var in vars_y) {
  print(
    ggplot(DATOS_100, aes_string(x = "HorasMismaCarga2", y = var)) +
      geom_point(alpha = 0.4, color = "darkred") +
      labs(title = paste("HorasMismaCarga2 vs", var),
           x = "HorasMismaCarga2", y = var) +
      theme_minimal()
  )
}
```

9. **Eliminación de observaciones con menos de 5 horas**

```
#Eliminar observaciones según el criterio #2
#CRITERIO 2: Si el compresor ha estado trabajando a una carga de trabajo
# fija menos de 5 horas seguidas, ELIMINAMOS LOS REGISTROS
```

```
DATOS_100_C2_C <- DATOS_100 %>%
  filter(is.na(HorasMismaCarga2) | HorasMismaCarga2 > 5)
```

10. **Manejo de Atipicos**

```
#-------------------------------------------------------------------------------
## Atípicos ##
#-------------------------------------------------------------------------------
# Calcular la proporción de valores atípicos en cada variable numérica
# Seleccionar las columnas deseadas
DATOS_100_C2_SA <- DATOS_100_C2_C[, c(
  "FechaHora",
  "HrsTotOp_C",
  "HrsCamb1",
  "HrsCamb2",
  "HrsCamb3",
  "HrsCamb4",
  "HrsGranMt_C",
  "PresAsp1",
  "PresAsp2",
  "PresAsp3",
  "PresImp1",
  "PresImp2",
  "PresImp3",
  "PresImp4",
  "PresDifFl",
  "PresInt1",
  "PresInt2",
  "PresInt3",
  "TempAsp",
  "TempCoj1",
  "TempCoj2",
  "TempCoj3",
  "TempCoj4",
  "TempCoj5",
  "TempEnt1",
  "TempEnt2",
  "TempImp1",
  "TempImp2",
  "Intensid",
  "Potencia",
  "TempCojM",
  "TempDevA1",
  "TempDevA2",
  "TempDevB1",
  "TempDevB2",
  "TempDevC1",
  "TempDevC2",
  "Fuga_Aceite"
)]


sapply(Filter(is.numeric, DATOS_100_C2_SA), function(x) atipicosAmissing3(x)[[2]]) /
nrow(DATOS_100_C2_SA)
```

```
# Aplicar la función para tratar los atípicos en las variables seleccionadas y convertir
en NA
for (variable in vars_cuantitativas_100_C2_C) {
  # Reemplazar los valores atípicos con NA
  result <- atipicosAmissing3(DATOS_100_C2_SA[[variable]])
  DATOS_100_C2_SA[[variable]] <- result[[1]] # Variable con los atípicos
reemplazados por NA

  # Opcional: imprimir el número de atípicos detectados en cada variable
  cat(sprintf("Variable: %s - Atípicos detectados: %d\n", variable, result[[2]]))
}

#Se verifica el resultado
summary(DATOS_100_C2_SA)
```

11. **Manejo de Missings o NAs**

```
#-------------------------------------------------------------------------
## MISSINGS
#-------------------------------------------------------------------------

# 1. Calcular la proporción de valores faltantes por observación (fila)
DATOS_100_C2_SA$prop_missings <- apply(is.na(DATOS_100_C2_SA), 1, mean)

# 2. Resumen de la proporción de valores faltantes por observación
summary(DATOS_100_C2_SA$prop_missings)

#Grafica de distribución de frecuencias de prop_missings
ggplot(DATOS_100_C2_SA, aes(x = prop_missings)) +
  geom_histogram(fill = "lightblue", color = "black", bins = 60) +
  labs(
    title = "Distribución de prop_missings",
    x = "prop_missings: proporcion de NAs",
    y = "Frecuencia"
  ) +
  theme_minimal()

#2.1 Valores que toma Propmissings
cuentaDistintos(DATOS_100_C2_SA)
#son 29

# 3. Calcular la proporción de valores faltantes por variable (columna)
(prop_missingsVars4 <- apply(is.na(DATOS_100_C2_SA), 2, mean))

table(DATOS_100_C2_SA$prop_missings)

#Resumen de las variables hasta ahora:
summary(DATOS_100_C2_SA)

#Se revisa cuantas observaciones con NA tienen la Fuga=1
# 1. Número total de observaciones con prop_missings diferente de 0
n_total_con_missings <- sum(DATOS_100_C2_SA$prop_missings != 0, na.rm =
TRUE)
```

```
# 2. Número de esas observaciones con Fuga_Aceite == "1" o "SI"
n_fugas_con_missings <- sum(DATOS_100_C2_SA$prop_missings != 0 &
DATOS_100_C2_SA$Fuga_Aceite %in% c("1", "SI"), na.rm = TRUE)

# Mostrar resultados
cat("Observaciones con prop_missings != 0:", n_total_con_missings, "\n")
cat("De ellas, con Fuga_Aceite == 1 o SI:", n_fugas_con_missings, "\n")

#Observaciones con prop_missings != 0: 4979
#De ellas, con Fuga_Aceite == 1 o SI: 126

# se decide eliminar todas las observaciones con almenos 1 dato NAs, reduciendo en
un 11% el dataset
# buscando optimizar la capacidad computacional y el esfuerzo analitico y evitando la
creación de valores nuevos, teniendo suficiente información

#Agrego las variables calculadas al dataset

DATOS_100_C2_SA <- cbind(
  DATOS_100_C2_SA,
  DATOS_100_C2_C[, c(
    "LC",
    "Lavg2",
    "RTBF",
    "HorasMismaCarga2"
    )]
)

# Filtrar las filas donde prop_missings es igual a 0 y creo DATOS_100_C2_F
DATOS_100_C2_F <- DATOS_100_C2_SA %>%
  filter(prop_missings == 0)

#Resumen de las variables hasta ahora:
summary(DATOS_100_C2_F)

# Para cada variable cuantitativa, se generan los gráficos de boxplot y densidad frente
a la variable binaria fuga de caeite
Fuga_Aceite_100_C2 <- DATOS_100_C2_F$Fuga_Aceite
for (var in vars_cuantitativas_100_C2_F) {
  #Crear el gráfico de cajas para la variable cuantitativa frente a la binaria
  print(boxplot_targetbinaria(DATOS_100_C2_F[[var]], Fuga_Aceite_100_C2, var))  #
Gráfico de cajas
  print(hist_targetbinaria(DATOS_100_C2_F[[var]], Fuga_Aceite_100_C2, var))     #
Gráfico de densidad
}
```

12. **Relación entre variables**

```
###----Relación entre variables-----####

#Retiro la variable propmissings ya que todos los valores son 0 en este punto.

DATOS_100_C2_F <- dplyr::select(DATOS_100_C2_F, -prop_missings)
```

```
#Ver la matriz de correlación de las variables input para determinar su relación entre
ellas
# Matriz de correlación entre las variables numéricas
DATOS_100_C2_F_NUM <- Filter(is.numeric, DATOS_100_C2_F)
DATOS_100_C2_F_NUM2 <- DATOS_100_C2_F_NUM[, -c(37:47)]
cor_matrix_tot <- cor(DATOS_100_C2_F_NUM2, use = "complete.obs", method =
"pearson")
corrplot(cor_matrix_tot, method = "ellipse", type = "upper", tl.cex = 0.8, tl.col = "black",
number.cex = 0.8)

#Se analisan la correlación entre las variables con el mismo nombre
#Horas totales y Horas desde el ultimo gran mantenimiento
DATOS_100_C2_F_HRS <- dplyr::select(DATOS_100_C2_F, HrsTotOp_C,
HrsGranMt_C)
cor_matrix_hrs <- cor(DATOS_100_C2_F_HRS, use = "complete.obs", method =
"pearson")
corrplot(cor_matrix_hrs, method = "ellipse", type = "upper", tl.cex = 0.8, tl.col = "black",
number.cex = 0.8)

#Variables de presión de aspiración
DATOS_100_C2_F_PASP <- dplyr::select(DATOS_100_C2_F, PresAsp1, PresAsp2,
PresAsp3)
cor_matrix_pasp <- cor(DATOS_100_C2_F_PASP, use = "complete.obs", method =
"pearson")
corrplot(cor_matrix_pasp, method = "ellipse", type = "upper", tl.cex = 0.8, tl.col =
"black", number.cex = 0.8)

#Variables de presión de impulsión
DATOS_100_C2_F_PIMP <- dplyr::select(DATOS_100_C2_F, PresImp1, PresImp2,
PresImp3, PresImp4)
cor_matrix_pimp <- cor(DATOS_100_C2_F_PIMP, use = "complete.obs", method =
"pearson")
corrplot(cor_matrix_pimp, method = "ellipse", type = "upper", tl.cex = 0.8, tl.col =
"black", number.cex = 0.8)

#Variables de presión de interetapas
DATOS_100_C2_F_PINT <- dplyr::select(DATOS_100_C2_F, PresInt1, PresInt2,
PresInt3)
cor_matrix_pint <- cor(DATOS_100_C2_F_PINT, use = "complete.obs", method =
"pearson")
corrplot(cor_matrix_pint, method = "ellipse", type = "upper", tl.cex = 0.8, tl.col = "black",
number.cex = 0.8)

#Variables de Temperatura de cojinetes
DATOS_100_C2_F_TCOJ <- dplyr::select(DATOS_100_C2_F, TempCoj1, TempCoj2,
TempCoj3, TempCoj4, TempCoj5)
cor_matrix_tcoj <- cor(DATOS_100_C2_F_TCOJ, use = "complete.obs", method =
"pearson")
corrplot(cor_matrix_tcoj, method = "ellipse", type = "upper", tl.cex = 0.8, tl.col = "black",
number.cex = 0.8)

#Variables de Temperatura entreetapas
DATOS_100_C2_F_TENT <- dplyr::select(DATOS_100_C2_F, TempEnt1, TempEnt2)
```

```
cor_matrix_tent <- cor(DATOS_100_C2_F_TENT, use = "complete.obs", method =
"pearson")
corrplot(cor_matrix_tent, method = "ellipse", type = "upper", tl.cex = 0.8, tl.col = "black",
number.cex = 0.8)

#Variables de Temperatura de impulsión
DATOS_100_C2_F_TIMP <- dplyr::select(DATOS_100_C2_F, TempImp1, TempImp2)
cor_matrix_timp <- cor(DATOS_100_C2_F_TIMP, use = "complete.obs", method =
"pearson")
corrplot(cor_matrix_timp, method = "ellipse", type = "upper", tl.cex = 0.8, tl.col =
"black", number.cex = 0.8)

#Variables de Intensidad y Potencia
DATOS_100_C2_F_Int <- dplyr::select(DATOS_100_C2_F, Intensid, Potencia)
cor_matrix_int <- cor(DATOS_100_C2_F_Int, use = "complete.obs", method =
"pearson")
corrplot(cor_matrix_int, method = "ellipse", type = "upper", tl.cex = 0.8, tl.col = "black",
number.cex = 0.8)

#Variables de Temperatura de devanado del motor
DATOS_100_C2_F_TDEV <- dplyr::select(DATOS_100_C2_F, TempDevA1,
TempDevA2, TempDevB1, TempDevB2, TempDevC1, TempDevC2)
cor_matrix_tdev <- cor(DATOS_100_C2_F_TDEV, use = "complete.obs", method =
"pearson")
corrplot(cor_matrix_tdev, method = "ellipse", type = "upper", tl.cex = 0.8, tl.col =
"black", number.cex = 0.8)

#Creación de nuevas variables
DATOS_100_C2_F <- DATOS_100_C2_F %>%
  mutate(
    PresAsp = (PresAsp1 + PresAsp3 + PresAsp3) / 2,
    PresImp = (PresImp1 + PresImp2 + PresImp3 + PresImp4) / 4,
    PresInt = (PresInt1 + PresInt2 + PresInt3) / 3,
    TempCoj = (TempCoj1 + TempCoj2 + TempCoj3 + TempCoj4 + TempCoj5) / 5,
    TempEnt = (TempEnt1 + TempEnt2) / 2,
    TempImp = (TempImp1 + TempImp2) / 2,
    TempDev = (TempDevA1 + TempDevA2 + TempDevB1 + TempDevB2 +
TempDevC1 + TempDevC2) / 6
  )


#Creación de nuevo DATASET
DATOS_100_C2_F2 <- DATOS_100_C2_F %>%
  dplyr::select(
    FechaHora,
    HrsTotOp_C,
    HrsCamb1,
    HrsCamb2,
    HrsCamb3,
    HrsCamb4,
    PresAsp,
    PresImp,
    PresDifFl,
    PresInt,
    TempAsp,
```

```
    TempCoj,
    TempEnt,
    TempImp,
    Intensid,
    TempCojM,
    TempDev,
    Fuga_Aceite,
    LC,
    Lavg2,
    RTBF,
    HorasMismaCarga2,
    )

#Ver la matriz de correlación de las variables input para determinar su relación entre
ellas
# Matriz de correlación entre las variables numéricas
DATOS_100_C2_F2_NUM <- Filter(is.numeric, DATOS_100_C2_F2)
cor_matrix_tot2 <- cor(DATOS_100_C2_F2_NUM, use = "complete.obs", method =
"pearson")
corrplot(cor_matrix_tot2, method = "ellipse", type = "upper", tl.cex = 0.8, tl.col = "black",
number.cex = 0.8)

#Creación de nuevas variables
DATOS_100_C2_F2 <- DATOS_100_C2_F2 %>%
  mutate(
    Dt = (TempImp - TempAsp),
    Dp = (PresAsp - PresImp ),
    HrsCambBand = pmin(HrsCamb1, HrsCamb2, HrsCamb3, HrsCamb4, na.rm =
TRUE)
  )


#Ver la matriz de correlación de las variables input para determinar su relación entre
ellas
# Matriz de correlación entre las variables numéricas
DATOS_100_C2_F2_NUM <- Filter(is.numeric, DATOS_100_C2_F2)
cor_matrix_tot2 <- cor(DATOS_100_C2_F2_NUM, use = "complete.obs", method =
"pearson")
corrplot(cor_matrix_tot2, method = "ellipse", type = "upper", tl.cex = 0.8, tl.col = "black",
number.cex = 0.8)

##----------IMPORTANCIA DE VARIABLES SEGÚN V DE CRAMER--------##

install.packages("lsr")
library(lsr)
ls("package:lsr")

Fuga_Aceite_F <- DATOS_100_C2_F2$Fuga_Aceite
DATOS_100_C2_F2_NUM$Aleatorio <- sample(100:300, size =
nrow(DATOS_100_C2_F2_NUM), replace = TRUE)

# Generar gráficos de barras con el V de Cramer para evaluar la importancia de las
variables.
graficoVcramer(DATOS_100_C2_F2_NUM, Fuga_Aceite_F)
par(cex.axis = 0.6)
```

```
#Se decide
#- Retirar intensidad del estudio
#- Crear las variables Diferencial de temperatura Dt y Diferencial de Presión Dp y
Retiro PresAsp, Temperatura de Aspiración y Temperatura impulsión
#- Retirar Temperatura de cojinete del motor.
#- Retirar RBTF
#- Crear la variable HrsCambBand que recoja la información de las 4 variables de
cambio de banda.

DATOS_100_C2_F22 <- DATOS_100_C2_F2 %>%
  dplyr::select(
    FechaHora,
    HrsTotOp_C,
    HrsCambBand,
    PresImp,
    PresDifFl,
    PresInt,
    TempCoj,
    TempEnt,
    TempDev,
    Fuga_Aceite,
    LC,
    Lavg2,
    HorasMismaCarga2,
    Dt,
    Dp
  )

#Ver la matriz de correlación de las variables input para determinar su relación entre
ellas
# Matriz de correlación entre las variables numéricas
DATOS_100_C2_F22_NUM <- Filter(is.numeric, DATOS_100_C2_F22)
cor_matrix_tot22 <- cor(DATOS_100_C2_F22_NUM, use = "complete.obs", method =
"pearson")
corrplot(cor_matrix_tot22, method = "ellipse", type = "upper", tl.cex = 0.8, tl.col =
"black", number.cex = 0.8)

Fuga_Aceite_F <- DATOS_100_C2_F22$Fuga_Aceite
DATOS_100_C2_F22_NUM$Aleatorio <- sample(100:300, size =
nrow(DATOS_100_C2_F2_NUM), replace = TRUE)

# Generar gráficos de barras con el V de Cramer para evaluar la importancia de las
variables.
graficoVcramer(DATOS_100_C2_F22_NUM, Fuga_Aceite_F)
par(cex.axis = 0.6)

# Para cada variable cuantitativa, se generan los gráficos de boxplot y densidad frente
a la variable binaria fuga de caeite

for (var in vars_cuantitativas_100_C2_F22) {
  #Crear el gráfico de cajas para la variable cuantitativa frente a la binaria
  print(boxplot_targetbinaria(DATOS_100_C2_F22[[var]], Fuga_Aceite_F, var))  #
Gráfico de cajas
```

```
  print(hist_targetbinaria(DATOS_100_C2_F22[[var]], Fuga_Aceite_F, var))     # Gráfico
de densidad
}

#Ver estadisticos por Fuga
# Función para calcular estadísticos
estadisticos <- function(x) {
  c(
    n       = sum(!is.na(x)),            # número de datos válidos
    media   = mean(x, na.rm = TRUE),
    mediana = median(x, na.rm = TRUE),
    sd      = sd(x, na.rm = TRUE),
    min     = min(x, na.rm = TRUE),
    q1      = quantile(x, 0.25, na.rm = TRUE),
    q3      = quantile(x, 0.75, na.rm = TRUE),
    max     = max(x, na.rm = TRUE)
  )
}

# Lista para guardar resultados
lista_resultados <- list()

# Iterar sobre cada variable numérica
for (var in vars_cuantitativas_100_C2_F22) {

  # Aplicar tapply para cada grupo de Fuga_Aceite
  tabla_var <- tapply(DATOS_100_C2_F22[[var]], DATOS_100_C2_F22$Fuga_Aceite,
estadisticos) %>%
    do.call(rbind, .) %>%
    as.data.frame()

  # Agregar columnas de identificación
  tabla_var$Fuga <- rownames(tabla_var)
  tabla_var$Variable <- var

  # Reorganizar columnas
  tabla_var <- tabla_var[, c("Variable", "Fuga", names(tabla_var)[1:8])]

  # Guardar en la lista
  lista_resultados[[var]] <- tabla_var
}

# Unir todos los resultados en una sola tabla
tabla_final <- bind_rows(lista_resultados)

# Ver resultado
tabla_final
```

13. **Balanceo de clases con SMOTE – Creación dataset balanceado**

```
install.packages("smotefamily")
install.packages("dplyr")

library(smotefamily)
library(dplyr)
```

```
set.seed(123)

# ========================
# 0) Chequeos previos
# ========================
stopifnot("Fuga_Aceite" %in% names(DATOS_100_C2_F22))
# Asegura que el objetivo es factor con niveles "0","1"
DATOS_100_C2_F22 <- DATOS_100_C2_F22 %>%
  mutate(Fuga_Aceite = factor(as.character(Fuga_Aceite), levels = c("0","1")))

cat("Distribución original:\n")
print(table(DATOS_100_C2_F22$Fuga_Aceite))

#Retiro variable FechaHora
DATOS_100_C2_F22 <- DATOS_100_C2_F22 %>% select(-c(FechaHora))

# ========================
# 1) Separar X e y
# ========================
X <- DATOS_100_C2_F22 %>% select(-c(Fuga_Aceite))  # 13 predictores numéricos
+ fecha
y <- DATOS_100_C2_F22$Fuga_Aceite              # factor "0"/"1"

# ========================
# 2) SMOTE: triplicar minoritaria
#    dup_size = 2  -> 2 sintéticas por cada caso minoritario (total ~3x)
# ========================
sm_res <- SMOTE(X = X, target = y, K = 5, dup_size = 2)

# El resultado viene en sm_res$data; la última columna es la clase (0/1)
datos_smote <- sm_res$data
colnames(datos_smote)[ncol(datos_smote)] <- "Fuga_Aceite"
datos_smote$Fuga_Aceite <- factor(datos_smote$Fuga_Aceite, levels = c("0","1"))

cat("\nDistribución tras SMOTE (solo oversampling):\n")
print(table(datos_smote$Fuga_Aceite))

set.seed(123)

# ========================
# 3) SMOTE: minimizar la mayoritaria a 2:1 de la minoritaria
#
# ========================

# 1) Tamaños objetivo
n_min <- sum(datos_smote$Fuga_Aceite == "1")    # 9438
target_maj <- 2L * n_min                        # 18876

# 2) Separar por clase
df_min <- datos_smote %>% filter(Fuga_Aceite == "1")   # 9438
df_maj <- datos_smote %>% filter(Fuga_Aceite == "0")   # 36156

# 3) Undersampling de la mayoritaria a 18876
df_maj_down <- df_maj %>% slice_sample(n = target_maj)
```

```
# 4) Unir y barajar
DATOS_100_C2_F22_BAL <- bind_rows(df_maj_down, df_min) %>%
  mutate(Fuga_Aceite = factor(Fuga_Aceite, levels = c("0","1"))) %>%
  select(all_of(names(DATOS_100_C2_F22))) %>%  # mismo orden de columnas que
el original
  sample_frac(1)

# 5) Verifica la distribución final
table(DATOS_100_C2_F22_BAL$Fuga_Aceite)

#0     1
#18876 9438
```

14. **Selección de Variables** (Portela, 2025)

```
DATOS_100_C2_F7 <- DATOS_100_C2_F22_BAL

DATOS_100_C2_F7_NUM <- Filter(is.numeric, DATOS_100_C2_F7)

f<-names(DATOS_100_C2_F7_NUM)
#Estandarizo las variables numericas

listconti<-
c("HrsTotOp_C","HrsCambBand","PresImp","PresDifFl","PresInt","TempCoj","TempEnt
",
            "TempDev","LC","Lavg2","HorasMismaCarga2","Dt","Dp")
vardep <- c("Fuga_Aceite")
# c)estandarizar las variables continuas
# Calculo medias y dtipica de datos y estandarizo (solo las continuas)

means <-apply(DATOS_100_C2_F7_NUM[,listconti],2,mean)
sds<-sapply(DATOS_100_C2_F7_NUM[,listconti],sd)

# Estandarizo solo las continuas

DATOS_100_C2_F7_NUM_SC<-scale(DATOS_100_C2_F7_NUM[,listconti], center =
means, scale = sds)
numerocont<-which(colnames(DATOS_100_C2_F7_NUM)%in%listconti)
DATOS_100_C2_F7_NUM<-
cbind(DATOS_100_C2_F7_NUM_SC,DATOS_100_C2_F7_NUM[,-
numerocont,drop=FALSE ])

vardep <- c("Fuga_Aceite")
DATOS_100_C2_F7_NUM$Fuga_Aceite <- DATOS_100_C2_F7$Fuga_Aceite

DATOS_100_C2_F7_NUM[,vardep]<-as.factor(DATOS_100_C2_F7_NUM[,vardep])

library(plyr)
library(dummies)

vardep<- "Fuga_Aceite"
data<-DATOS_100_C2_F7_NUM

# Convertir la variable
```

```
data$Fuga_Aceite <- ifelse(data$Fuga_Aceite == "1", "Yes", "No")
data$Fuga_Aceite <- as.factor(data$Fuga_Aceite)

install.packages("lsr")
library(lsr)
ls("package:lsr")

# Cargar librerías
library(ggplot2)
library(reshape2)

library(parallel)
library(doParallel)
library(plyr)
library(dummies)
library(ggplot2)
library(naniar)
library(MASS)
library(caret)
library(MXM)

source("C:\\Users\\anyel\\OneDrive\\Documentos\\ANYELO\\TECNICAS MACHINE
LEARNING\\Tareas\\Tarea 2\\funcion steprepetido binaria.R")
source("C:\\Users\\anyel\\OneDrive\\Documentos\\ANYELO\\TECNICAS MACHINE
LEARNING\\Tareas\\Tarea 2\\cruzadas avnnet y log binaria.R")

dput(names(DATOS_100_C2_F7_NUM))

archivo1<-DATOS_100_C2_F7_NUM
vardep<- "Fuga_Aceite"
nombres1<-c("HrsTotOp_C", "HrsCambBand", "PresImp", "PresDifFl", "PresInt",
          "TempCoj", "TempEnt", "TempDev", "LC", "Lavg2", "HorasMismaCarga2",
          "Dt", "Dp")
y<-archivo1[,vardep]
x<-archivo1[,nombres1]

data<-DATOS_100_C2_F7_NUM


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

# CON AIC

full<-glm(Fuga_Aceite~.,data=archivo1,family = binomial(link="logit"))
null<-glm(Fuga_Aceite~1,data=archivo1,family = binomial(link="logit"))

selec1<-stepAIC(null,scope=list(upper=full),
              direction="both",family = binomial(link="logit"),trace=FALSE)

vec<-(names(selec1[[1]]))
```

```
length(vec)
# 13-1 variables
dput(vec)

#c("(Intercept)", "LC", "TempDev", "HrsCambBand", "TempCoj", "Lavg2",
#"HrsTotOp_C", "HorasMismaCarga2", "PresImp", "Dp", "PresInt",
#"Dt", "TempEnt")

# CON BIC, ponemos k=log(n) en stepAIC, en este caso n=28314 observaciones,
k=10.25

full<-glm(Fuga_Aceite~.,data=archivo1,family = binomial(link="logit"))
null<-glm(Fuga_Aceite~1,data=archivo1,family = binomial(link="logit"))

selec1<-stepAIC(null,scope=list(upper=full),
            direction="both",family = binomial(link="logit"),trace=FALSE,k=10.25)

vec<-(names(selec1[[1]]))

length(vec)
# 12 variables-1
dput(vec)

#c("(Intercept)", "LC", "TempDev", "HrsCambBand", "TempCoj", "Lavg2",
#"HrsTotOp_C", "HorasMismaCarga2", "PresImp", "Dp", "PresInt",
#"Dt")


# BORUTA

# También vale como Filter
library(Boruta)
out.boruta <- Boruta(Fuga_Aceite~., data = archivo1)

print(out.boruta)

summary(out.boruta)

sal<-data.frame(out.boruta$finalDecision)

sal2<-sal[which(sal$out.boruta.finalDecision=="Confirmed"),,drop=FALSE]
dput(row.names(sal2))

length(dput(row.names(sal2)))

# 13

#c("HrsTotOp_C", "HrsCambBand", "PresImp", "PresDifFl", "PresInt",
#"TempCoj", "TempEnt", "TempDev", "LC", "Lavg2", "HorasMismaCarga2",
#"Dt", "Dp")

# PRUEBAS RFE

library(caret)
```

```
# 1) Arreglar los niveles de la clase y poner la positiva primero
#    Mapea 1 -> "Yes", 0 -> "No"
y <- factor(ifelse(y == "1", "Yes", "No"), levels = c("Yes","No"))

# 2) Control de train: calcular F1/Precision/Recall
tc_pr <- trainControl(
  method = "cv", number = 5,
  classProbs = TRUE,
  summaryFunction = prSummary,   # devuelve "F", "Precision", "Recall"
  savePredictions = "final"
)

# 3) Control de RFE: usar caretFuncs y forzar prSummary como summary
ctrl_rfe <- rfeControl(
  functions = caretFuncs,
  method = "cv", number = 5,
  verbose = FALSE
)
ctrl_rfe$functions$summary <- prSummary   # <- clave para que RFE use
F/Recall/Precision


install.packages("MLmetrics")
library(MLmetrics)
# 4) Ejecutar RFE optimizando F1 (o cambia a "Recall"/"Precision")
set.seed(123)
rfe_F1 <- rfe(
  x, y,
  sizes = 1:13,
  rfeControl = ctrl_rfe,
  method = "rf",          # puede usar "glm", "svmRadial", etc.
  trControl = tc_pr,
  metric = "F",           # "F" = F1 en prSummary # puede usar Recall o Precision
  maximize = TRUE
)

# Ver resultados
rfe_F1
predictors(rfe_F1)
plot(rfe_F1, metric = "F")
plot(rfe_F1, metric = "AUC")
rfe_F1$optVariables

#12 VARIABLES

#c("LC","HrsTotOp_C","HrsCambBand","Lavg2","TempDev","TempCoj",
#"PresInt","PresDifFl","PresImp","Dp","HorasMismaCarga2","Dt")

# MXM

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

```
mmpc2 <- MMPC(vardep, archivo1, max_k = 3, hash = TRUE,
              test = "testIndLogistic")

mmpc2@selectedVars

a<-dput(names(archivo1[,c(mmpc2@selectedVars)]))

length(a)

a

# 7 variables

#c("HrsCambBand", "PresDifFl", "TempCoj", "TempDev", "LC", "Lavg2",
#"HorasMismaCarga2")

#Importancia de la variables:
b<-dput(names(archivo1[,c(mmpc2@selectedVarsOrder)]))

length(b)

b
#c("LC","TempDev","HrsCambBand","TempCoj","PresDifFl","HorasMismaCarga2","Lav
g2")

#STEP AIC REPETIDO

lista<-steprepetidobinaria(data=archivo1,vardep=c("Fuga_Aceite"),
                           listconti=nombres1,
                           sinicio=12345,sfinal=12385,porcen=0.8,criterio="AIC")

tabla<-lista[[1]]
dput(lista[[2]][[1]])

#12 variables
#c("LC", "TempDev", "HrsCambBand", "TempCoj", "Lavg2", "HrsTotOp_C",
#"HorasMismaCarga2", "PresImp", "Dp", "PresInt", "Dt", "TempEnt")

#El mismo resultado que STEPAIC sin repetida

#STEP BIC REPETIDO

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


lista<-steprepetidobinaria(data=archivo1,vardep=c("Fuga_Aceite"),
                           listconti=nombres1,
                           sinicio=12345,sfinal=12385,porcen=0.8,criterio="BIC")

tabla<-lista[[1]]
```

```
dput(lista[[2]][[1]])
dput(lista[[2]][[2]])

#11 Variables
#c("LC", "TempDev", "HrsCambBand", "TempCoj", "Lavg2", "HrsTotOp_C",
#"HorasMismaCarga2", "PresImp", "Dp", "PresInt", "Dt")

#El mismo resultado que STEPBAIC sin repetida

#Validación Cruzada con regresión logisitca de las variables encontradas incialmente

data<-DATOS_100_C2_F7_NUM
# Convertir la variable
data$Fuga_Aceite <- ifelse(data$Fuga_Aceite == "1", "Yes", "No")
data$Fuga_Aceite <- as.factor(data$Fuga_Aceite)

str(data)

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

medias1 <- cruzadalogistica_PR_AUC2(
  data      = data,
  vardep    = "Fuga_Aceite",
  listconti = c("LC", "TempDev", "HrsCambBand", "TempCoj", "Lavg2", "HrsTotOp_C",
                "HorasMismaCarga2", "PresImp", "Dp", "PresInt", "Dt", "TempEnt"),
  listclass = c(""),
  grupos    = 4, repe = 5,
  umbral    = "f1_rep"
)

medias1$modelo="STEPAIC-12v"

medias2<-cruzadalogistica_PR_AUC2(data=data,
                    vardep="Fuga_Aceite",listconti= c("LC", "TempDev", "HrsCambBand",
"TempCoj", "Lavg2",
                                                      "HrsTotOp_C", "HorasMismaCarga2", "PresImp",
"Dp", "PresInt",
                                                      "Dt")
                    ,
                    listclass=c(""),grupos=4,sinicio=1234,repe=5,umbral="f1_rep")

medias2$modelo="STEPBIC-11v"

medias3<-cruzadalogistica_PR_AUC2(data=data,
                    vardep="Fuga_Aceite",listconti=
                      c("HrsTotOp_C", "HrsCambBand", "PresImp", "PresDifFl", "PresInt",
                        "TempCoj", "TempEnt", "TempDev", "LC", "Lavg2",
"HorasMismaCarga2",
                        "Dt", "Dp"),
                    listclass=c(""),grupos=4,sinicio=1234,repe=5,umbral="f1_rep")
```

```
medias3$modelo="Boruta-13v"

medias4<-cruzadalogistica_PR_AUC2(data=data,
                          vardep="Fuga_Aceite",listconti=

c("LC","HrsTotOp_C","HrsCambBand","Lavg2","TempDev","TempCoj",

"PresInt","PresDifFl","PresImp","Dp","HorasMismaCarga2","Dt"),
                          listclass=c(""),grupos=4,sinicio=1234,repe=5,umbral="f1_rep")

medias4$modelo="RFE-12v"


medias5<-cruzadalogistica_PR_AUC2(data=data,
                    vardep="Fuga_Aceite",listconti=
                      c("HrsCambBand", "PresDifFl", "TempCoj", "TempDev", "LC",
"Lavg2",
                        "HorasMismaCarga2")
                    ,
                    listclass=c(""),grupos=4,sinicio=1234,repe=5,umbral="f1_rep")

medias5$modelo="MXM-7v"
#Para exportar y comparar con dataset original
medias5_2 <- medias5
medias5_2$modelo="MXM-7v-SMOTE"


#Se agrupan en un vector los resultados
union1<-rbind(medias1,medias2,medias3,medias4,medias5)

#Se grafican en un boxplot las metricas
par(cex.axis=0.6)
boxplot(data=union1,col="pink",F1~modelo,main="F1")
par(cex.axis=0.6)
boxplot(data=union1,col="pink",AUC~modelo,main="AUC")
par(cex.axis=0.6)
boxplot(data=union1,col="pink",Recall~modelo,main="Recall")
par(cex.axis=0.6)
boxplot(data=union1,col="pink",Precision~modelo,main="Precision")

#Numero de variables por metodo de selección:
#Boruta: 13; MXM: 7; RFE:12 ; STEPAIC: 12; STEPBIC: 11;
#STEPrep-BIC: 11; STEPrepAIC:
```

15. **Tuneado de Red Neuronal** (Portela, 2025)

```
#-----------------------------------------------------------------
# TUNEADO DE LA RED CON LAS 7 VARIABLES DEL MODLO DE MEJOR Balance
F1 y AUC: MXM-7v
#-----------------------------------------------------------------

c("LC","TempDev","HrsCambBand","TempCoj","PresDifFl","HorasMismaCarga2","Lavg
2")

vardep="Fuga_Aceite"
```

```
variables<-
c("LC","TempDev","HrsCambBand","TempCoj","PresDifFl","HorasMismaCarga2","Lavg
2")

data2<-data[,c(variables,vardep)]

# Calculos:
# hay 9438 obs de la categoria objetivo. y 7 variables input.
# Si consideramos 40 obs como minimo por cada obs de la clase objetivo:
# 235 parametros maximo, h(k+1)+h+1=235 ; k=7, como mucho 26 nodos, pero vamos
# a probar hasta maximo 15 observar para evitar sobreajuste

control_F1 <- trainControl(
  method = "repeatedcv", number = 4, repeats = 5,
  savePredictions = "final",
  classProbs = TRUE,
  summaryFunction = prSummary     # devuelve F, Precision, Recall
)

set.seed(1234)
nnetgrid <- expand.grid(size=c(5,10,15), decay=c(0.001,0.01,0.1), bag=FALSE)

completo <- data.frame()
listaiter <- c(50,100,200,500,1000,2000,3000)

GS_T0 <- Sys.time()
cluster <- makeCluster(detectCores() - 1); registerDoParallel(cluster)

for (iter in listaiter) {
  rednnet <- train(
    Fuga_Aceite ~ ., data = data2,
    method = "avNNet",
    linout = FALSE, maxit = iter,
    trControl = control_F1,
    tuneGrid = nnetgrid,
    metric = "F",               # <<< optimiza F1
    maximize = TRUE,
    trace = FALSE
  )
  rednnet$results$itera <- iter
  completo <- rbind(completo, rednnet$results)
}

stopCluster(cluster); registerDoSEQ()

# Ordenar por F1
completo <- completo[order(-completo$F), ]

ggplot(completo, aes(x=factor(itera), y=F,
                     color=factor(decay), pch=factor(size))) +
  geom_point(position=position_dodge(width=0.5), size=3) +
  labs(y="F1", x="iteraciones", color="decay", pch="size")

# Ordenar por AUC
completo <- completo[order(-completo$AUC), ]
```

```
ggplot(completo, aes(x=factor(itera), y=AUC,
                     color=factor(decay), pch=factor(size))) +
  geom_point(position=position_dodge(width=0.5), size=3) +
  labs(y="AUC", x="iteraciones", color="decay", pch="size")

# itera=500, size=15, decay=0.01

medias8<-cruzadaavnnetbin_PR_AUC(data=data2,
                      vardep="Fuga_Aceite",listconti=

c("LC","TempDev","HrsCambBand","TempCoj","PresDifFl","HorasMismaCarga2","Lavg
2"),
                      listclass=c(""),grupos=4,sinicio=1234,repe=5,repeticiones=5,itera=500,
                      size=c(15),decay=c(0.01))

medias8$modelo="Red-7v"
#Para exportar y comparar con dataset original
medias8_2 <- medias8
medias8_2$modelo="Red-7v-SMOTE"

#Se agrupan en un vector los resultados
union1<-rbind(medias1,medias2,medias3,medias4,medias5,medias8)

#Se grafican en un boxplot las metricas
par(cex.axis=0.6)
boxplot(data=union1,col="pink",F1~modelo,main="F1")
par(cex.axis=0.6)
boxplot(data=union1,col="pink",AUC~modelo,main="AUC")
par(cex.axis=0.6)
boxplot(data=union1,col="pink",Recall~modelo,main="Recall")
par(cex.axis=0.6)
boxplot(data=union1,col="pink",Precision~modelo,main="Precision")

#Se observa un mejor resultado de AUC Y F1 y Precision de la red neuronal,
#pero se sacrifica la sensibilidad
```

16. **Tuneado de Bagging** (Portela, 2025)

```
#------------------------------------------------------------------
# TUNEADO DE BAGGING CON LAS 7 VARIABLES DEL MODLO
#------------------------------------------------------------------

source ("C:\\Users\\anyel\\OneDrive\\Documentos\\ANYELO\\TECNICAS MACHINE
LEARNING\\Tareas\\Tarea 2\\cruzada rf binaria.R")
source ("C:\\Users\\anyel\\OneDrive\\Documentos\\ANYELO\\TECNICAS MACHINE
LEARNING\\Tareas\\Tarea 2\\cruzada arbolbin.R")


#Desición del minimo número de ntree

library(randomForest)

names(data2) <- make.names(names(data2))
names(data2)
```

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

rfbis<-randomForest(
factor(Fuga_Aceite)~LC+TempDev+HrsCambBand+TempCoj+PresDifFl+HorasMisma
Carga2+Lavg2,
                data=data2,
                mtry=7,ntree=5000,sampsize=300,nodesize=10,replace=TRUE)
plot(rfbis$err.rate[,1])


#Se observa en la grafica que a partir de ntree=2500 se estabilizan los valores
#Por lo tanto, se utiliza ntree=2500

medias_bag_1<-cruzadarfbin_PR_AUC(data=data2, vardep="Fuga_Aceite",

listconti=c("LC","TempDev","HrsCambBand","TempCoj","PresDifFl","HorasMismaCarg
a2","Lavg2"),
                        listclass=c(""),
                        grupos=4,sinicio=1234,repe=5,nodesize=10,
                        mtry=7,ntree=2500,replace=TRUE,umbral="f1_rep",sampsize=50)

medias_bag_1$modelo="bagging50"


medias_bag_2<-cruzadarfbin_PR_AUC(data=data2, vardep="Fuga_Aceite",

listconti=c("LC","TempDev","HrsCambBand","TempCoj","PresDifFl","HorasMismaCarg
a2","Lavg2"),
                         listclass=c(""),
                         grupos=4,sinicio=1234,repe=5,nodesize=10,

mtry=7,ntree=2500,replace=TRUE,umbral="f1_rep",sampsize=100)

medias_bag_2$modelo="bagging100"


medias_bag_3<-cruzadarfbin_PR_AUC(data=data2, vardep="Fuga_Aceite",

listconti=c("LC","TempDev","HrsCambBand","TempCoj","PresDifFl","HorasMismaCarg
a2","Lavg2"),
                         listclass=c(""),
                         grupos=4,sinicio=1234,repe=5,nodesize=10,

mtry=7,ntree=2500,replace=TRUE,umbral="f1_rep",sampsize=500)


medias_bag_3$modelo="bagging500"


medias_bag_4<-cruzadarfbin_PR_AUC(data=data2, vardep="Fuga_Aceite",
```

```
listconti=c("LC","TempDev","HrsCambBand","TempCoj","PresDifFl","HorasMismaCarg
a2","Lavg2"),
                        listclass=c(""),
                        grupos=4,sinicio=1234,repe=5,nodesize=10,

mtry=7,ntree=2500,replace=TRUE,umbral="f1_rep",sampsize=1000)

medias_bag_4$modelo="bagging1000"

medias_bag_5<-cruzadarfbin_PR_AUC(data=data2, vardep="Fuga_Aceite",

listconti=c("LC","TempDev","HrsCambBand","TempCoj","PresDifFl","HorasMismaCarg
a2","Lavg2"),
                        listclass=c(""),
                        grupos=4,sinicio=1234,repe=5,nodesize=10,

mtry=7,ntree=2500,replace=TRUE,umbral="f1_rep",sampsize=5000)

medias_bag_5$modelo="bagging5000"

medias_bag_6<-cruzadarfbin_PR_AUC(data=data2, vardep="Fuga_Aceite",

listconti=c("LC","TempDev","HrsCambBand","TempCoj","PresDifFl","HorasMismaCarg
a2","Lavg2"),
                        listclass=c(""),
                        grupos=4,sinicio=1234,repe=5,nodesize=10,

mtry=7,ntree=2500,replace=TRUE,umbral="f1_rep",sampsize=10000)


medias_bag_6$modelo="bagging10000"

#Grafico las medias de bagging
union_bag_1<-
rbind(medias_bag_1,medias_bag_2,medias_bag_3,medias_bag_4,medias_bag_5,
              medias_bag_6)
uni_bag<-union_bag_1
uni_bag$modelo <- with(uni_bag,
                  reorder(modelo,F1, mean))
par(cex.axis=0.7,las=2)
boxplot(data=uni_bag,F1~modelo,main="F1",col="pink")
par(cex.axis=0.7)
boxplot(data=uni_bag,col="pink",AUC~modelo,main="AUC")
par(cex.axis=0.7)
boxplot(data=uni_bag,col="pink",Recall~modelo,main="Recall")
par(cex.axis=0.7)
boxplot(data=uni_bag,col="pink",Precision~modelo,main="Precision")


#La diferencia es de milesimas por lo que se decide trabajar con sampsize=1000
#Para evitar sobre ajuste


# PROBAMOS CON SAMPSIZE=1000 CON 4 GRUPOS DE CV
```

```
medias9<-medias_bag_4
medias9$modelo="bagging1000"

#Para exportar y comparar con dataset original
medias9_2 <- medias9
medias9_2$modelo="bagging1000-SMOTE"

#Se agrupan en un vector los resultados
union1<-rbind(medias1,medias2,medias3,medias4,medias5,medias8,medias9)

#Se grafican en un boxplot las metricas
par(cex.axis=0.6)
boxplot(data=union1,col="pink",F1~modelo,main="F1")
par(cex.axis=0.6)
boxplot(data=union1,col="pink",AUC~modelo,main="AUC")
par(cex.axis=0.6)
boxplot(data=union1,col="pink",Recall~modelo,main="Recall")
par(cex.axis=0.6)
boxplot(data=union1,col="pink",Precision~modelo,main="Precision")

#Se observa un mejor resultado de AUC, F1 y Precision y Recall en la red neuronal
```

17. **Tuneado de Random Forest** (Portela, 2025)

```
#-------------------------------------------------------------
#----RANDOM FOREST-----------------------------------------------
#-------------------------------------------------------------

# TUNEADO DE MTRY CON CARET

library(caret)
library(randomForest)

# Resumen que devuelve F1
F1summary <- function(data, lev = c("No","Yes"), model = NULL){
  cm <- caret::confusionMatrix(data$pred, data$obs, positive = "Yes")
  P <- cm$byClass["Precision"]; R <- cm$byClass["Recall"]
  F1 <- ifelse(is.na(P) || is.na(R) || (P+R)==0, 0, 2*P*R/(P+R))
  c(F1 = as.numeric(F1))
}

set.seed(12345)
rfgrid <- expand.grid(mtry = c(1,2,3,4,5,6,7))

control <- trainControl(
  method = "cv",
  number = 4,
  savePredictions = "all",
  classProbs = TRUE,          # no estrictamente necesario para F1 con clases, pero útil
si luego usas probs
  summaryFunction = F1summary # <- clave
)

rf_F1 <- train(
```

```
  factor(Fuga_Aceite) ~
LC+TempDev+HrsCambBand+TempCoj+PresDifFl+HorasMismaCarga2+Lavg2,
  data = data2,
  method = "rf",
  trControl = control,
  tuneGrid = rfgrid,
  metric = "F1",            # <- optimiza F1
  ntree = 2500,
  nodesize = 10,
  replace = TRUE,
  importance = TRUE,
  sampsize = 1000
)

stopCluster(cluster); registerDoSEQ()

rf_F1$bestTune
plot(rf_F1)  # curva F1 vs mtry

# IMPORTANCIA DE VARIABLES

final<-rf_F1$finalModel

tabla<-as.data.frame(importance(final))
tabla<-tabla[order(-tabla$MeanDecreaseAccuracy),]
tabla

barplot(tabla$MeanDecreaseAccuracy,names.arg=rownames(tabla),
main="IMPORTANCIA DE VARIABLES")

# Recomienda mtry=6

# La función cruzadarfbin permite plantear random forest

medias10<-cruzadarfbin_PR_AUC(data=data2, vardep="Fuga_Aceite",

listconti=c("LC","TempDev","HrsCambBand","TempCoj","PresDifFl","HorasMismaCarg
a2","Lavg2"),
                        listclass=c(""),
                        grupos=4,sinicio=1234,repe=5,nodesize=10,

mtry=6,ntree=2500,replace=TRUE,umbral="f1_rep",sampsize=1000)

medias10$modelo="Rf"

#Se agrupan en un vector los resultados
union1<-rbind(medias1,medias2,medias3,medias4,medias5,medias8,medias9,
medias10)

#Se grafican en un boxplot las metricas
par(cex.axis=0.6)
boxplot(data=union1,col="pink",F1~modelo,main="F1")
par(cex.axis=0.6)
boxplot(data=union1,col="pink",AUC~modelo,main="AUC")
par(cex.axis=0.6)
```

```
boxplot(data=union1,col="pink",Recall~modelo,main="Recall")
par(cex.axis=0.6)
boxplot(data=union1,col="pink",Precision~modelo,main="Precision")

#Se agrupan en un vector los resultados de Red, bagging y RF
union2<-rbind(medias8,medias9, medias10)

#Se grafican en un boxplot las metricas
par(cex.axis=0.6)
boxplot(data=union2,col="pink",F1~modelo,main="F1")
par(cex.axis=0.6)
boxplot(data=union2,col="pink",AUC~modelo,main="AUC")
par(cex.axis=0.6)
boxplot(data=union2,col="pink",Recall~modelo,main="Recall")
par(cex.axis=0.6)
boxplot(data=union2,col="pink",Precision~modelo,main="Precision")

#Se observa un mejor resultado en Red de F1, Pero RF tiene mejor AUC
```

18. **Tuneado Gradient Boosting** (Portela, 2025)

```
#------------------------------------------------------------
# GRADIENT BOOSTING
#------------------------------------------------------------
# TUNEADO DE GRADIENT BOOSTING CON CARET

# Caret permite tunear estos parámetros básicos:
#
#  shrinkage (parámetro v de regularización, mide la velocidad de ajuste, a menor v,
más lento y necesita más iteraciones, pero es más fino en el ajuste)
#  n.minobsinnode: tamaño máximo de nodos finales (el principal parámetro que mide
la complejidad)
#  n.trees=el número de iteraciones (árboles)
#  interaction.depth (2 para árboles binarios)

library(caret)

set.seed(12345)

# Métrica F1 basada en clases (voto 0.5)
F1summary <- function(data, lev=c("No","Yes"), model=NULL){
  cm <- confusionMatrix(data$pred, data$obs, positive="Yes")
  P <- cm$byClass["Precision"]; R <- cm$byClass["Recall"]
  F1 <- ifelse(is.na(P)||is.na(R)||(P+R)==0, 0, 2*P*R/(P+R))
  c(F1=as.numeric(F1))
}

gbmgrid<-expand.grid(shrinkage=c(0.2,0.1,0.05,0.01,0.001),
                     n.minobsinnode=c(5,10,20),
                     n.trees=c(100,500,1000,2000,3000),
                     interaction.depth=c(2))

control<-trainControl(method = "cv",number=4,savePredictions = "all",
                      classProbs=TRUE,summaryFunction=F1summary)
```

```
gbmF1<-
train(factor(Fuga_Aceite)~LC+TempDev+HrsCambBand+TempCoj+PresDifFl+HorasMi
smaCarga2+Lavg2,
        data=data2,
        method="gbm",trControl=control,tuneGrid=gbmgrid,metric = "F1",
        distribution="bernoulli", bag.fraction=1,verbose=FALSE)

gbmF1

plot(gbmF1)

library(gbm)

# IMPORTANCIA DE VARIABLES
summary(gbmF1)
tabla<-summary(gbmF1)
par(cex=1.5,las=2)
barplot(tabla$rel.inf,names.arg=row.names(tabla))

varImp(gbmF1, scale = FALSE)
plot(varImp(gbmF1, scale = FALSE))


#Los las diferencias son de milesimas, por lo que se puede tomar con buen resultado
#shrinkage  n.minobsinnode  n.trees  F1
#0.200       10              2000   0.9984



# La función cruzadagbmbin permite plantear gradient boosting para binarias

source ("C:\\Users\\anyel\\OneDrive\\Documentos\\ANYELO\\TECNICAS MACHINE
LEARNING\\Tareas\\Tarea 2\\cruzada gbm binaria.R")

medias11<-cruzadagbmbin_PR_AUC(data=data2, vardep="Fuga_Aceite",

listconti=c("LC","TempDev","HrsCambBand","TempCoj","PresDifFl","HorasMismaCarg
a2","Lavg2"),
                listclass=c(""),
                grupos=4,sinicio=1234,repe=5,

n.minobsinnode=10,shrinkage=0.2,n.trees=2000,interaction.depth=2,umbral = "f1_rep")

medias11$modelo="gbm"

#Se agrupan en un vector los resultados
union1<-rbind(medias1,medias2,medias3,medias4,medias5,medias8,medias9,
medias10,medias11)

#Se grafican en un boxplot las metricas
par(cex.axis=0.6)
boxplot(data=union1,col="pink",F1~modelo,main="F1")
par(cex.axis=0.6)
boxplot(data=union1,col="pink",AUC~modelo,main="AUC")
```

```
par(cex.axis=0.6)
boxplot(data=union1,col="pink",Recall~modelo,main="Recall")
par(cex.axis=0.6)
boxplot(data=union1,col="pink",Precision~modelo,main="Precision")

#Se agrupan en un vector los resultados de Red, bagging, RF y gbm
union2<-rbind(medias8,medias9, medias10,medias11)

#Se grafican en un boxplot las metricas
par(cex.axis=0.6)
boxplot(data=union2,col="pink",F1~modelo,main="F1")
par(cex.axis=0.6)
boxplot(data=union2,col="pink",AUC~modelo,main="AUC")
par(cex.axis=0.6)
boxplot(data=union2,col="pink",Recall~modelo,main="Recall")
par(cex.axis=0.6)
boxplot(data=union2,col="pink",Precision~modelo,main="Precision")

#Se observa un mejor resultado en RF de AUC, F1 que en Red, bagging y gbm

# mientras que en Precision y Recall es similar, aunque mejor que bagging
```

19. **Tuneado SVM** (Portela, 2025)

```
# **************************
# TUNEADO SVM
# **************************

library(caret)

#  SVM LINEAL: SOLO PARÁMETRO C

# Resumen: devuelve F1 a partir de la matriz de confusión
F1summary <- function(data, lev = c("No","Yes"), model = NULL){
  cm <- caret::confusionMatrix(data$pred, data$obs, positive = "Yes")
  P <- cm$byClass["Precision"]; R <- cm$byClass["Recall"]
  F1 <- ifelse(is.na(P) || is.na(R) || (P+R)==0, 0, 2*P*R/(P+R))
  c(F1 = as.numeric(F1))
}

set.seed(12345)
SVMgrid <- expand.grid(C = c(0.01,0.05,0.1,0.5,1,5,10))

control <- trainControl(
  method = "cv",
  number = 4,
  savePredictions = "all",
  summaryFunction = F1summary
)

SVM_F1 <- train(
```

```
  factor(Fuga_Aceite) ~
LC+TempDev+HrsCambBand+TempCoj+PresDifFl+HorasMismaCarga2+Lavg2,
  data = data2,
  method = "svmLinear",
  trControl = control,
  tuneGrid = SVMgrid,
  metric = "F1",
  verbose = FALSE
)

SVM_F1$bestTune
plot(SVM_F1$results$C, SVM_F1$results$F1, type = "b",
     xlab = "C", ylab = "F1 (CV)", main = "SVM lineal: F1 vs C")

SVM_F1$results

#SVM_F1$bestTune
#C= 0.5

#  SVM Polinomial: PARÁMETROS C, degree, scale

# Resumen: devuelve F1 a partir de la matriz de confusión
F1summary <- function(data, lev = c("No","Yes"), model = NULL){
  cm <- caret::confusionMatrix(data$pred, data$obs, positive = "Yes")
  P <- cm$byClass["Precision"]; R <- cm$byClass["Recall"]
  F1 <- ifelse(is.na(P) || is.na(R) || (P+R)==0, 0, 2*P*R/(P+R))
  c(F1 = as.numeric(F1))
}

set.seed(12345)
SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.5,1,5,10),
                     degree=c(2),scale=0.1)

control <- trainControl(
  method = "cv",
  number = 4,
  savePredictions = "all",
  summaryFunction = F1summary
)

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


SVM_F1_Poly <- train(
  factor(Fuga_Aceite) ~
LC+TempDev+HrsCambBand+TempCoj+PresDifFl+HorasMismaCarga2+Lavg2,
  data = data2,
  method = "svmPoly",
  trControl = control,
  tuneGrid = SVMgrid,
  metric = "F1",
```

```
  verbose = FALSE
)

SVM_F1_Poly$results
SVM_F1_Poly$bestTune
plot(SVM_F1_Poly$results$C, SVM_F1_Poly$results$F1, type = "b",
     xlab = "C", ylab = "F1 (CV)", main = "SVM Poly: F1 vs C")


# C    degree   scale   F1
# 10   2        0.1     0.8562

#  SVM RBF: PARÁMETROS C, sigma

# Resumen: devuelve F1 a partir de la matriz de confusión
F1summary <- function(data, lev = c("No","Yes"), model = NULL){
  cm <- caret::confusionMatrix(data$pred, data$obs, positive = "Yes")
  P <- cm$byClass["Precision"]; R <- cm$byClass["Recall"]
  F1 <- ifelse(is.na(P) || is.na(R) || (P+R)==0, 0, 2*P*R/(P+R))
  c(F1 = as.numeric(F1))
}

set.seed(12345)
SVMgrid<-expand.grid(C=c(0.5,1,5,10,30),
                     sigma=c(0.0001,0.005,0.01,0.05))

control <- trainControl(
  method = "cv",
  number = 4,
  savePredictions = "all",
  summaryFunction = F1summary
)

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


SVM_F1_RBF <- train(
  factor(Fuga_Aceite) ~
LC+TempDev+HrsCambBand+TempCoj+PresDifFl+HorasMismaCarga2+Lavg2,
  data = data2,
  method = "svmRadial",
  trControl = control,
  tuneGrid = SVMgrid,
  metric = "F1",
  verbose = FALSE
)

SVM_F1_RBF$results
SVM_F1_RBF$bestTune
library(ggplot2)
```

```
ggplot(SVM_F1_RBF$results, aes(x = C, y = F1, color = factor(sigma), group = sigma))
+
  geom_line(size = 1) +
  geom_point(size = 2) +
    labs(x = "C", y = "F1",
       color = "Sigma",
       title = "F1 vs C para diferentes valores de Sigma") +
  theme_minimal()
# C    sigma
# 30   0.05

source ("C:\\Users\\anyel\\OneDrive\\Documentos\\ANYELO\\TECNICAS MACHINE
LEARNING\\Tareas\\Tarea 2\\cruzada SVM binaria lineal.R")
source ("C:\\Users\\anyel\\OneDrive\\Documentos\\ANYELO\\TECNICAS MACHINE
LEARNING\\Tareas\\Tarea 2\\cruzada SVM binaria polinomial.R")
source ("C:\\Users\\anyel\\OneDrive\\Documentos\\ANYELO\\TECNICAS MACHINE
LEARNING\\Tareas\\Tarea 2\\cruzada SVM binaria RBF.R")

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

medias12<-cruzadaSVMbin_PR_AUC(data=data2, vardep="Fuga_Aceite",

listconti=c("LC","TempDev","HrsCambBand","TempCoj","PresDifFl","HorasMismaCarg
a2","Lavg2"),
                          listclass=c(""),
                          grupos=4,sinicio=1234,repe=5,C=0.5,umbral="f1_rep")

medias12$modelo="SVM-Lineal"

medias13<-cruzadaSVMbinPoly_PR_AUC(data=data2, vardep="Fuga_Aceite",

listconti=c("LC","TempDev","HrsCambBand","TempCoj","PresDifFl","HorasMismaCarg
a2","Lavg2"),
                   listclass=c(""),
                   grupos=4,sinicio=1234,repe=5,C=10,degree=2,
scale=0.1,umbral="f1_rep")

medias13$modelo="SVM-Poly"

medias14<-cruzadaSVMbinRBF_PR_AUC(data=data2, vardep="Fuga_Aceite",

listconti=c("LC","TempDev","HrsCambBand","TempCoj","PresDifFl","HorasMismaCarg
a2","Lavg2"),
                     listclass=c(""),
                     grupos=4,sinicio=1234,repe=5,
                     C=30,sigma=0.05,umbral="f1_rep")

medias14$modelo="SVM-RBF"

#Se agrupan en un vector los resultados de Red, bagging, RF y gbm
```

```
union2<-rbind(medias5,medias8,medias9,
medias10,medias11,medias12,medias13,medias14)


#Se grafican en un boxplot las metricas
par(cex.axis=0.6)
boxplot(data=union2,col="pink",F1~modelo,main="F1")
par(cex.axis=0.6)
boxplot(data=union2,col="pink",AUC~modelo,main="AUC")
par(cex.axis=0.6)
boxplot(data=union2,col="pink",Recall~modelo,main="Recall")
par(cex.axis=0.6)
boxplot(data=union2,col="pink",Precision~modelo,main="Precision")

#Se observa un mejor resultado en RF de AUC, F1 que en Red, bagging, gbm y SVM

# mientras que en Precision y Recall es similar, aunque mejor que bagging


#SE ORDENAN LAS GRAFICAS
uni2<-union2
uni2$modelo <- with(uni2,
                    reorder(modelo,F1, median))
par(cex.axis=0.6)
boxplot(data=uni2,col="pink",F1~modelo,main="F1")
par(cex.axis=0.6)
boxplot(data=uni2,col="pink",AUC~modelo,main="AUC")
par(cex.axis=0.6)
boxplot(data=uni2,col="pink",Recall~modelo,main="Recall")
par(cex.axis=0.6)
boxplot(data=uni2,col="pink",Precision~modelo,main="Precision")
```

20. **Matriz de confusion Modelo Escogido**

```
#-------------------------------------
# Matriz confusion y metricas del modelo gradient bosting
#-------------------------------------

# Ajustar el modelo de gradient bosting
library(caret)

set.seed(12345)

gbmgrid_modelogbm<-expand.grid(shrinkage=c(0.2),
                    n.minobsinnode=c(10),
                    n.trees=c(2000),
                    interaction.depth=c(2))

control_modelogbm<-trainControl(method = "cv",number=4,savePredictions = "all",
                     classProbs=TRUE)

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

```
modelo_gbm<-
train(factor(Fuga_Aceite)~LC+TempDev+HrsCambBand+TempCoj+PresDifFl+HorasMi
smaCarga2+Lavg2,
        data=train,
        method="gbm",trControl=control_modelogbm,tuneGrid=gbmgrid_modelogbm,
        distribution="bernoulli", bag.fraction=1,verbose=FALSE)

modelo_gbm


# Probabilidades de la clase positiva "Yes"
prob_modgbm <- predict(modelo_gbm, newdata = test, type = "prob")[, "Yes"]

# Convertir probabilidades a clases
pred_clases <- ifelse(prob_modgbm >= 0.5, "Yes", "No")

# Convertir a factor con los mismos niveles que la variable real
pred_clases <- factor(pred_clases, levels = levels(factor(test$Fuga_Aceite)))

# Matriz de confusión
conf_mat <- confusionMatrix(pred_clases, factor(test$Fuga_Aceite), positive = "Yes")
conf_mat
```