UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
SEDE VIÑA DEL MAR – JOSÉ MIGUEL CARRERA

PROPUESTA DE PREDICCIÓN DE FALLOS MEDIANTE MACHINE LEARNING A COMPONENTE CRÍTICO DE CAMIÓN MINERO.

Trabajo de titulación para optar al título de ingeniero en mantenimiento industrial.

Alumno:
Sr. Sergio Enrique Brito Pangue.

Profesor Guía:
Mg. Ing. Sr. Andrés Eduardo Aránguiz Garrido

2024

RESUMEN

KEYWORDS: MACHINE LEARNING, WEIBULL, ARIMA, RANDOM FOREST, TBF, PREDICCION.

La introducción de herramientas de Machine Learning en la gestión del mantenimiento se impulsa principalmente por la necesidad de aumentar la eficiencia y reducir costos. Además, gracias al avance tecnológico y la recopilación masiva de datos, el Machine Learning puede prever fallos futuros, generando aportes significativos al mantenimiento predictivo.

El objetivo principal es desarrollar un método eficiente para prever fallos en un componente crucial de camiones mineros, combinando el modelo ARIMA con técnicas de Machine Learning para obtener predicciones más sólidas y proactivas.

En este estudio, se emplea una metodología que implica un análisis inicial de los datos históricos de una flota de camiones de extracción. En una primera fase, se identifica el sistema crítico de un equipo, seguido de la estimación de futuros fallos mediante el modelo de Weibull. Además, se utiliza el modelo ARIMA y por último, se incorpora el modelo Random Forest, un algoritmo de Machine Learning que combina múltiples árboles de decisión para lograr predicciones más precisas.

Los resultados obtenidos a través de las metodologías utilizadas tienen como objetivo mejorar la capacidad predictiva del modelo de Weibull. En la fase inicial, dicho modelo muestra un Error Absoluto Medio (MAE) de 755,89 [hrs], lo que indica que, en promedio, las predicciones realizadas por este modelo alejan considerablemente de los valores reales. Posteriormente, al incorporar el modelo auto ARIMA, se logra un aumento del 55% en la precisión, obteniendo un MAE de 338,93 [hrs]. A pesar de este avance, persiste un error de predicción significativo. Finalmente, al emplear el algoritmo Random Forest de Machine Learning, se experimenta un incremento considerable en la precisión de los resultados. El MAE alcanza un valor de 12,21 [hrs], lo que representa un notable aumento del 98% en comparación con el modelo de Weibull. Este enfoque demuestra ser altamente efectivo para mejorar la capacidad predictiva y reducir la discrepancia entre las predicciones y los valores reales.

La implementación de estas herramientas no solo mejora la eficacia operativa en mantenimiento, sino que también facilita la toma de decisiones. Al utilizar tecnologías como el Machine Learning y modelos estadísticos como ARIMA, las organizaciones obtienen análisis más precisos sobre el estado de sus equipos y predicciones de fallos. En resumen, la utilización de estas herramientas no solo significa eficiencia operativa, sino también una optimización significativa de los recursos empresariales

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICOS

SIGLAS Y SIMBOLOGÍAS

SIGLA

ARMA : Autoregressive Moving Average
ARIMA : Autoregressive Integrated Moving Average
IA : Inteligencia Artificial
AI : Artificial Intelligence
MAE : Mean Absolute Error
MAPE : Mean Absolute Percentage Error
ML : Machine Learning
MSE : Mean Standard Error
MTBF : Mean Time Between Failures
MTTR : Mean Time to Repair
RF : Random Forest
RMSE : Root Mean Square Error

SIMBOLOGÍA

hrs : Horas
Ton : Toneladas
$T\ fun$: Tiempo de funcionamiento en horas
$f(t)$: Probabilidad de falla instantánea
$R(t)$: Confiabilidad
$F(t)$: Probabilidad acumulada de falla
α : Alfa
β : Beta
γ : Gamma
$\lambda(t)$: Tasa de fallas
$MTBF$: Tiempo medio entre fallas
Γ( ) : Función Gamma
t : Tiempo “t” de una serie de tiempo
p : Orden de coeficientes para parte autorregresiva
d : Orden de diferenciaciones para series no estacionarias
q : Orden de coeficientes para media móvil

INTRODUCCIÓN

La presente investigación hace referencia a la implementación de una metodología para la estimación de fallos futuros en componente crítico de la flota de transporte, compuesta por camiones 930E-4, debido a que los camiones de extracción son equipos cruciales para mantener la producción continua del cobre. refinado. Por lo tanto, es necesario efectuar intervenciones en los tiempos adecuados para mantener la operatividad de estos equipos. Esta investigación se realizó con la intención de disminuir los fallos de estos camiones, ya que debido a esto podemos ocasionar el no cumplimiento de las metas propuestas por la organización. Por lo tanto, la estimación de las intervenciones preventivas en el momento justo es clave, ya que podemos disminuir de forma significativa los costos en intervenciones no contempladas.

Es por ello por lo que las metodologías que se utilizarán para el desarrollo a lo largo del trabajo en primera instancia será la distribución de Weibull, es el modelo clásico de predicción utilizado por la industria. El principal inconveniente en este modelo es que entrega valores de predicción a intervalos fijos y constantes, asumiendo que las condiciones de operación no varían respecto al tiempo. Debido a esto, no consideramos que estos tengan un comportamiento inconstante o mutable. Por tales motivos, se propone la nueva implementación de herramientas y técnicas de estimación y predicción a través de Machine Learning y modelos de regresión, estos proveen de pronósticos más acertados en base al comportamiento de datos históricos aplicados a un equipo o sistema en particular.

La distribución de los diversos temas en la estructura del trabajo, en el capítulo 1, se definieron las metodologías con las que se desarrolló el presente trabajo. Además, se estudió el histórico de fallos de la flota de transporte para seleccionar el componente crítico. En el capítulo 2, se desarrollaron las metodologías de predicción de fallos futuros, las cuales se realizaron mediante programación en Python. Y para finalizar, en el capítulo 3, se compararon y se prefirió el modelo con mejores métricas de rendimiento obtenidas en los distintos modelos

OBJETIVOS

OBJETIVO GENERAL

Elaborar un enfoque metodológico para mejorar la predicción de eventos de fallos en un componente crítico de un camión minero mediante la implementación del modelo ARIMA (autorregresivo integrado media móvil) y la aplicación de técnicas de Machine Learning.

OBJETIVOS ESPECIFICOS

- Definir las metodologías predictivas para el estudio del historial de fallos de la flota de transporte de camiones mineros, con el propósito de establecer una base sólida para el análisis y la predicción de fallos futuros.

- Diseñar modelos predictivos utilizando la distribución Weibull, modelo ARIMA y la técnica de Random Forest para estimar el error de predicción en relación con el tiempo entre fallos (TBF).

- Comparar las metodologías de predicción propuestas utilizando métricas de rendimiento, con el fin de seleccionar el modelo que presente el menor error de estimación en comparación con los valores predichos y los tiempos entre fallos reales, para obtener el mejor modelo de predicción.

CAPITULO 1: ANTECEDENTES GENERALES Y METODOLOGIA

1.1. CASO DE ESTUDIO

Este caso de estudio ofrece un registro histórico de fallos en una flota de transporte, específicamente camiones Komatsu 930E-4. Dada su importancia crucial, estos datos son esenciales para realizar estimaciones, analizar comportamientos y proyectar escenarios en el proceso productivo. En los últimos años, el campo del mantenimiento industrial ha incorporado herramientas de Machine Learning (ML) que han demostrado obtener resultados precisos al anticipar fallas tempranas.

1.1.1 Descripción del equipo

Para comenzar, la función principal de los camiones mineros es transportar material desde una zona de carga a un punto de acopio. Son de gran tamaño; un camión minero puede llegar a cargar 300 toneladas y su mayor potencia se consigue mediante motores diésel y propulsión eléctrica. Presentan una buena estabilidad debido a las suspensiones delanteras y traseras dispuestas en el camión, lo que genera una gran comodidad para el operador que debe estar 12 horas en la operación de la mina. A continuación, se mostrarán las especificaciones técnicas del equipo.

Características	930 E-4
Carga Util (dependiendo de la Tolva)	300 Ton cortas
M. Diesel	QSK 60
Potencia	2700 HP
Alternador	GTA 41
Rueda	GDY 106
Grupo control	17K535 C PSC v. 22.02 b TCI v. 22.02 a
Grupo de retardo	20 Parillas RP1 Y RP2

Tabla 1-1. Especificaciones técnicas 930 E-4.
Fuente: Curso de familiarización de equipos Komatsu CaEx.

También contamos con el módulo de potencia del camión eléctrico, el cual tiene como función principal actuar como una fuente de energía para alimentar los sistemas eléctricos, hidráulicos y mecánicos. Los componentes principales del módulo de potencia son el motor diésel, alternador, gabinete de control y motores eléctricos.

Para continuar, también se encuentra el sistema de propulsión del camión eléctrico, el cual tiene como función controlar la tracción del equipo. Cuando hablamos de tracción,

nos referimos a que el operador tiene absoluto control del desplazamiento del camión tanto hacia adelante como en reversa. Los componentes principales se presentan a continuación:

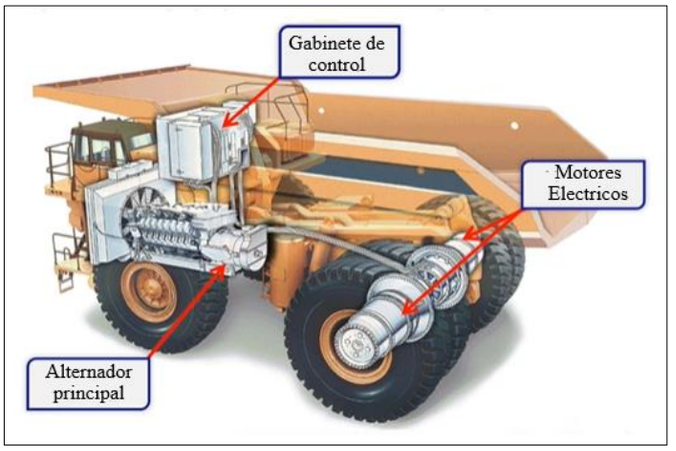

Figura 1-1. Componentes principales de sistema de propulsión.
Fuente: Curso de familiarización de equipos Komatsu CaEx.

Estos componentes se relacionan de manera simple: el motor diésel le entrega movimiento al alternador principal, este a su vez genera energía eléctrica alterna constante, la cual viaja a nuestro gabinete de control. Este último transforma la energía, dejándola disponible para que pueda viajar a los motores de tracción, quienes a su vez transforman dicha energía en movimiento, logrando que el equipo comience a desplazarse.

Para finalizar, tenemos el sistema de retardo del camión eléctrico, cuya función es disminuir la velocidad del equipo mediante un fenómeno electromagnético provocado en los motores de tracción. A continuación, se presentarán los componentes principales y la secuencia de cómo trabaja el sistema de retardo.

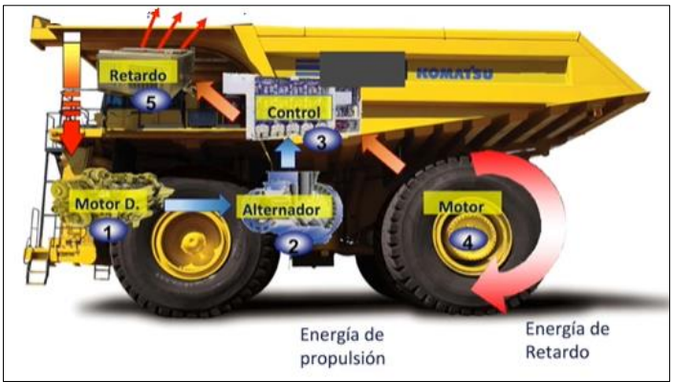

Figura 1-2. Secuencia de trabajó de sistema de retardo.
Fuente: Curso de familiarización de equipos Komatsu CaEx.

La etapa de retardo se puede resumir como un proceso donde los motores se transforman en generadores de energía eléctrica. Esta energía viaja al gabinete de control, que a su vez la transforma y la dirige al banco de parrillas para ser disipada en forma de calor. Esto se realiza con el objetivo de que los motores de tracción generen una resistencia al movimiento o desplazamiento que posee el equipo en ese momento, logrando un efecto de disminución de velocidad que puede llegar hasta los 5 km/hora aproximadamente. Posterior a esa velocidad, se aplicarán los frenos hidráulicos para lograr detener completamente el camión.

1.1.2 Análisis de histórico de datos

Para lograr realizar un análisis más completo a una flota de camiones, se puede utilizar una metodología para la jerarquización de los equipos, esto con el fin de priorizar el componente más crítico de la flota, esto se logra con un gráfico de dispersión, donde de manera visual mediante dos variables, se logra identificar qué componentes aporta mayor indisponibilidad al prestar atención en el tiempo medio de reparación y una menor confiabilidad en cuanto a la cantidad total de detenciones imprevistas por componentes.

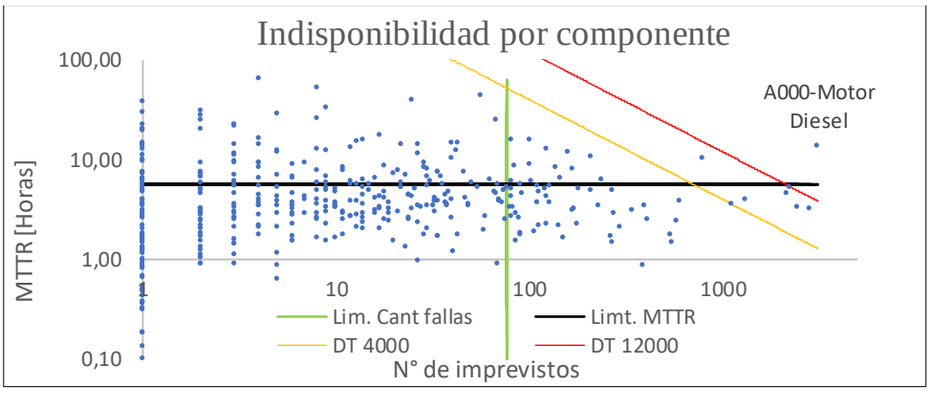

Gráfico 1-1. Gráfico de dispersión de componentes.
Fuente: Elaboración propia través de Excel.

Al visualizar el gráfico 1-1, se demuestra que el subsistema motor diésel es el que posee mayor criticidad en comparación con los demás subsistemas. Es por ello por lo que se seleccionará este subsistema para desarrollar un análisis en el que se pretende localizar cuál es el camión que tiene mayor criticidad. Para ello, se utilizará la información disponible del equipo CA-60 seleccionado anteriormente, con el fin de lograr priorizar el subsistema crítico de este camión.

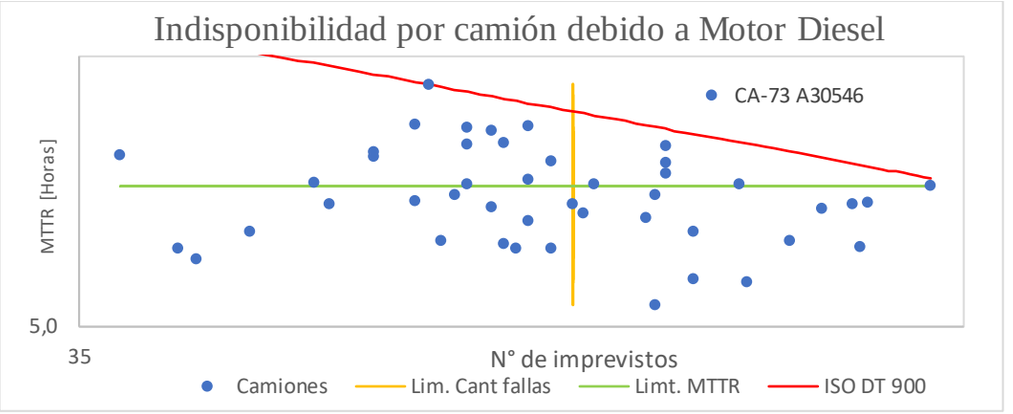

Gráfico 1-2. Gráfico de dispersión de camiones.

Fuente: Elaboración propia través de Excel

Al visualizar el gráfico 1-2, se demuestra que el camión que posee mayor criticidad es el CA-73, modelo 930E-4. Es por ello por lo que se calcularán los Tiempos Entre Fallas (TBF) de este camión para poder realizar el desarrollo a lo largo del trabajo.

1.1.3 Optimizar la predicción de fallas en equipos críticos integrando Machine Learning

Para abordar con mayor precisión la predicción de fallas en el equipo crítico del camión 930E, el cual fue seleccionado a partir de datos de fallas y contexto operacional, es esencial superar las limitaciones de técnicas tradicionales como el modelo de Weibull. Estas técnicas, aunque valiosas, pueden no proporcionar predicciones lo suficientemente certeras para prevenir fallas críticas. En este sentido, la integración de técnicas predictivas avanzadas basadas en machine learning emerge como una necesidad significativa. Al combinar el contexto operacional del camión 930E con datos detallados de fallas, podemos identificar y focalizar en el equipo crítico. La incorporación de algoritmos de machine learning permite analizar patrones complejos y generar modelos predictivos más precisos, lo que resulta en una mayor capacidad de anticipación y en la reducción de riesgos operacionales significativos. Este enfoque integral no solo optimiza la eficiencia en la gestión del mantenimiento, sino que también fortalece la fiabilidad y seguridad de los equipos críticos en entornos operacionales exigentes como el de los camiones mineros.

1.2. MACHINE LEARNING

Para entender el Machine Learning (ML), lo primero que debemos considerar es que pertenece a una rama de la Inteligencia Artificial (IA) Es decir, Inteligencia Artificial es un conjunto de aprendizaje automático y Machine Learning es uno de ellos. Debido a

esto el ML se puede considerar como un aprendizaje automático de la IA, como se muestra a continuación.

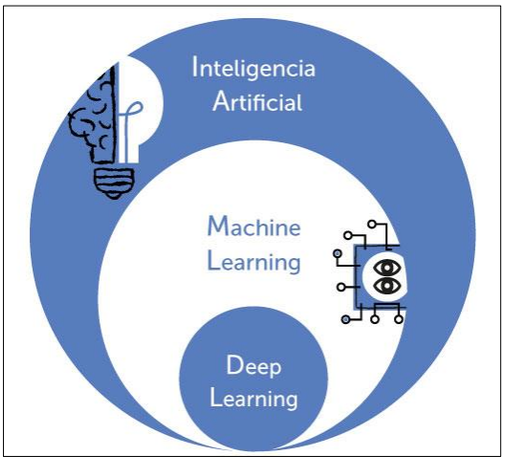

Figura 1-3. La inteligencia artificial y sus conjuntos.
Fuente: Talento Corporativo, Diferencias entre Machine Learning, Inteligencia Artificial y Robótica.

Este conjunto perteneciente a la Inteligencia Artificial está detrás de infinidad de aplicaciones cotidianas hoy en día como, por ejemplo, las recomendaciones de películas en Netflix, los asistentes de voz, como Siri y Alexa, la capacidad de los automóviles para ser autónomos, etc.

1.2.1. Definición de machine learning

El machine learning o Aprendizaje Automático es una rama de la Inteligencia Artificial (IA) que estudia cómo otorgar a las computadoras la capacidad de aprender por sí mismas, basándose en algoritmos capaces de identificar patrones en grandes bases de datos y aprender de ellos. Gracias a este aprendizaje, las máquinas son capaces de crear predicciones lo más precisas posibles y aportar soluciones en un campo determinado.

Iniciando desde un gran número de ensayos y errores, se puede fabricar un modelo de predicción lo más adecuado posible al mínimo error y generalizar un comportamiento ya contemplado. Como ejemplo, se pueden predecir los valores de las acciones en el futuro si se analiza y entrena un modelo con una cantidad suficiente de datos del pasado.

1.2.2. Tipos de machine learning

I. Aprendizaje Supervisado: Este tipo de Aprendizaje supone que partimos de una serie de datos en donde se conoce el valor del atributo objetivo (predicción), para esto debemos analizar los datos para generar algoritmos que aprenden iterativamente de los datos en base a ellos, la máquina se va entrenando y

detectando patrones con los que será capaz de predecir el atributo objetivo para un nuevo conjunto de datos.

II. Aprendizaje No Supervisado: Este tipo de Aprendizaje trabaja con los algoritmos que basan su proceso de entrenamiento en un conjunto de datos no etiquetados, lo cual este quiere decir que no se tiene ningún resultado con antelación del valor objetivo o de clase Este método de aprendizaje no supervisado está dedicado a las tareas de agrupamiento, también llamadas clustering o segmentación, donde su objetivo es encontrar grupos similares en el conjunto de datos.

III. Aprendizaje Reforzado (Reinforcement Learning): Este tipo de aprendizaje intenta conseguir que una IA aprenda a decidir mediante su propia experiencia. Es decir, que, ante una situación determinada, este es capaz de entrenarse hasta dar con una buena solución mediante un proceso interactivo de prueba y error a base de aprender de forma inteligente cuando empieza a tomar buenas decisiones.

1.2.3 Machine learning en mantenimiento industrial

El aprendizaje automático tiene diversas aplicaciones en el campo del mantenimiento industrial. Estas aplicaciones se centran en mejorar la eficiencia, la confiabilidad y la planificación del mantenimiento, con lo que podemos prevenir fallas, disminuir costos y reducir los tiempos de inactividad no planificados. A continuación, se darán a conocer algunas formas en las que el aprendizaje automático se puede aplicar en el mantenimiento industrial:

I. Mantenimiento predictivo: El aprendizaje automático se utiliza para predecir fallas y determinar el momento óptimo para realizar mantenimiento en los equipos. Se recopilan y analizan datos de registros históricos y otros factores relevantes para identificar patrones y tendencias que indican la aparición de fallos o la degradación del rendimiento. Esto permite programar el mantenimiento antes de que ocurran problemas graves, evitando costosos tiempos de inactividad.

II. Análisis de condición: El aprendizaje automático ayuda a analizar los datos en tiempo real o históricos de sensores y sistemas de monitoreo para identificar patrones anómalos o condiciones de funcionamiento inusuales. Esto permite detectar anomalías en el rendimiento de los equipos y tomar medidas preventivas antes de que se produzcan fallas.

III. Gestión de inventario y repuestos: El aprendizaje automático puede analizar los datos de inventario, el uso de repuestos y la demanda histórica para predecir las necesidades futuras de repuestos y optimizar el inventario. Esto ayuda a evitar el exceso de inventario y los costos asociados, al tiempo que garantiza la disponibilidad de los repuestos necesarios en el momento adecuado.

estas son solo algunas de las aplicaciones del aprendizaje automático en el mantenimiento industrial.

1.3. DISTRIBUCIÓN WEIBULL

Esta función es una distribución de probabilidad continua y triparamétrica, es decir, está completamente definida por tres parámetros y es la más empleada en el campo de la confiabilidad. Recibe su nombre de Waloddi Weibull, quien la describió detalladamente en 1951. Además, este modelo es el más utilizado estadísticamente para tratar con datos históricos. También es la más empleada por los ingenieros de confiabilidad debido a que puede utilizarse con tamaños de muestras muy pequeños.

En mantenimiento, es una poderosa herramienta para los ingenieros de confiabilidad industrial, ya que se utiliza para modelar la vida útil de los activos. También se utiliza para estimar la confiabilidad de un sistema o componente, es decir, la probabilidad de que este funcione correctamente sin fallas durante un período específico. Para finalizar, sirve para planificar actividades de mantenimiento para mejorar la disponibilidad y el rendimiento de los equipos.

1.3.1. Parámetros de Weibull

Los parámetros que considera la distribución de Weibull son tres fundamentales para caracterizar la forma y las características de la distribución Weibull, así como su utilidad en el análisis de datos y en aplicaciones prácticas. Estos permiten ajustar la distribución a los datos de falla observados y modelar la vida útil de los componentes y sistemas de manera efectiva. Estos parámetros son esenciales para el análisis de confiabilidad.

El factor de localización a hace referencia a la ubicación desde donde será el origen de los datos para realizar el cálculo de la distribución, este parámetro se designa con la

legra griega “Gamma” (γ), para efectos de cálculos dentro del análisis de este trabajo, este factor tendrá el valor de cero.

El factor de escala a hace referencia a la vida de un activo donde se evalúa la distribución en un tiempo determinado, este factor se representa con la letra griega “Alpha” (α).

El factor de forma a hace referencia a la manera en la cual se encuentran distribuidos los datos, este se representa con la letra griega “Beta” (β).

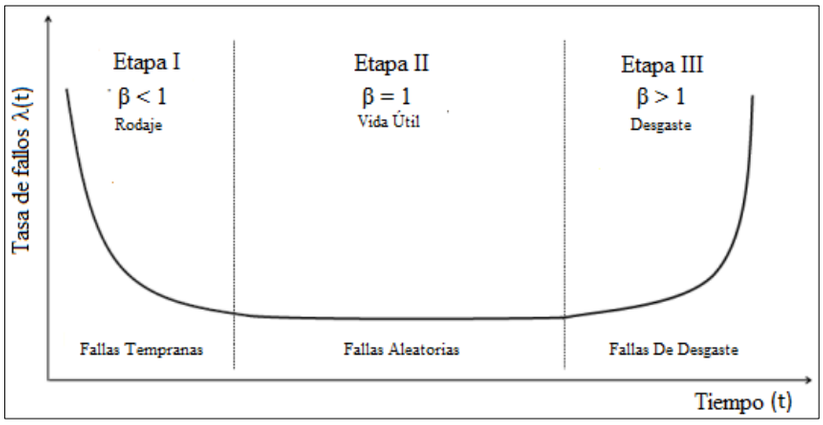

Figura 1-4. Representación curva típica de falla “curva de la bañera”.
Fuente: Elaboración propia a través de Paint.

A continuación, se aclarará cada etapa de la Figura 1-4. En donde se puede representar el estado en el cual se encuentra un activo:

I. Etapa I: En esta etapa el factor de forma es menor a 1 ($\beta < 1$), esto indica fallas tempranas donde la tasa de fallas es decreciente y está asociado directamente a un equipo nuevo que se encuentra en rodaje, también se asocia a personal con falta de capacitación, problemas de diseño o configuración incorrecta.

II. Etapa II: En esta etapa el factor de forma es igual a 1 ($\beta = 1$), acá el activo se encuentra en la etapa de operación donde las fallas son aleatorias y la tasa de fallos es constante, esto indican que el equipo o sistema se encuentra en su etapa de vida operativa.

III. Etapa III: En esta etapa el factor de forma es mayor a 1 ($\beta > 1$), esto indica que la tasa de fallas aumenta en el tiempo lo que indica que el equipo tiende a tener más fallas por desgaste debido al proceso de envejecimiento.

A esta representación del factor de forma y la tasa de fallas se le conoce como la curva de la bañera, y normalmente corresponde a la vida de operación de un equipo. Esto nos ayuda a entender de manera gráfica el comportamiento que tiene desde el momento en que comienza su operación hasta el momento en el cual es necesario restituirlo.

1.3.2. Confiabilidad

Para los análisis de confiabilidad, la función de densidad o probabilidad de falla instantánea, "f(t)", corresponde a la probabilidad de que un componente o sistema falle dentro de un instante predefinido a partir del inicio de servicio. Se define con la ecuación (1).

$$f(t) = \frac{\beta}{\alpha} * \left(\frac{t-\gamma}{\alpha}\right)^{\beta} * e^{-\left(\frac{t-\gamma}{\alpha}\right)^{\beta}} \quad (1)$$

La función de probabilidad acumulada de fallas "F(t)" corresponde a la probabilidad de que un componente o sistema falle en un instante establecido, es decir, no sobreviva en funcionamiento correcto hasta ese instante (t). Se define con la ecuación (2).

$$F(t) = 1 - e^{-\left(\frac{t-\gamma}{\alpha}\right)^{\beta}} \quad (2)$$

La confiabilidad "R(t)" corresponde a la probabilidad de que un componente o sistema se encuentre en funcionamiento durante un tiempo (t) bajo condiciones ambientales dadas. Además, está relacionada directamente con la función de probabilidad acumulada de fallas "F(t)", y se define con la ecuación (3).

$$R(t) = e^{-\left(\frac{t-\gamma}{\alpha}\right)^{\beta}} = 1 - F(t) \quad (3)$$

Uno de los indicadores utilizados en mantenimiento corresponde a la esperanza de vida media o esperanza del tiempo medio entre fallas (MTBF). Según una distribución de Weibull, se define con la ecuación (4).

$$MTBF = \gamma + \alpha * \Gamma\left(1 + \frac{1}{\beta}\right) \quad (4)$$

La tasa de fallas "λ" corresponde a la probabilidad de tener fallas en un componente o sistema, a condición de que éste haya sobrevivido hasta el instante anterior. Según una distribución de Weibull, se define con la ecuación (5).

$$\lambda = \frac{\beta}{\alpha} \cdot \left(\frac{t-\gamma}{\alpha}\right)^{(\beta-1)} = \frac{f(ti)}{R(ti)} \tag{5}$$

1.3.3. Procedimiento para obtener parámetros Weibull ajustado

La aplicación del modelo de uso de vida Weibull ajustado representa menores errores en relación con la aplicación tradicional de la metodología.

Para la obtención de los parámetros Weibull, además de obtener información estadística y gráfica, se utiliza el software Python. Este software, solo diseñando un algoritmo simple, es capaz de entregar la información deseada. Para el cálculo de MTBF, el proceso se realizará utilizando los 8 primeros datos del TBF, donde se calculará el valor de Alpha y de Beta mediante la fórmula (4) anteriormente mencionada. Una vez obtenido el valor, se utilizan los 9 primeros datos del TBF con el fin de determinar cuándo sucederá el décimo MTBF. Siguiendo este ciclo repetitivo, se logra proyectar cada evento de falla hasta llegar al último dato. En la figura 1-5 se muestra un diagrama de flujo de la obtención de los parámetros.

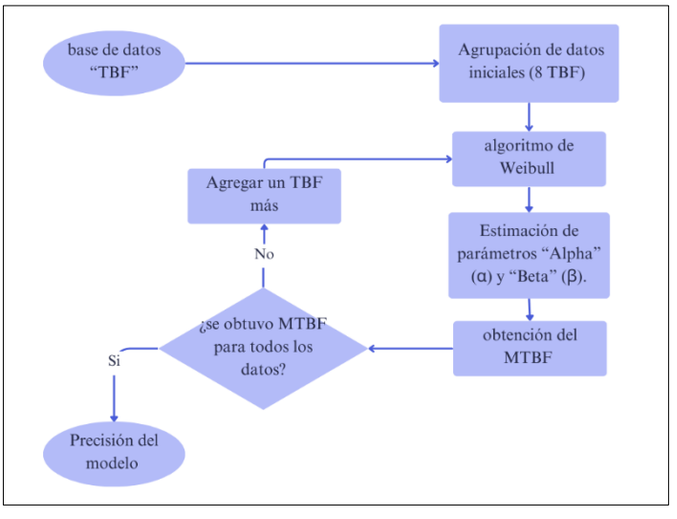

Figura 1-5. Diagrama de flujo Weibull ajustado.
Fuente: Elaboración propia a través de canva.

1.4. MODELO ARIMA

El presente modelo ARIMA (Auto Regressive Integrated Moving Average) significa (Autorregresivo (AR) Integrado (I) Media móvil (MA)), este modelo se utiliza para

estudiar y predecir series temporales, con el motivo de encontrar patrones para la predicción de valores futuros. Por medio del aprendizaje de los valores pasados, es decir, sus propios retrasos y los errores de Pronóstico retrasados, de esta manera que se pueda usar la ecuación para pronosticar valores futuros. Este método fue descrito inicialmente por los estadísticos George Edward Box y Gwilym Meirion Jenkins en 1970, además este modelo es ampliamente utilizado en diversas áreas, como economía, finanzas, pronóstico del clima y análisis de datos en general.

El modelo consta de tres términos (p, d, q) que, a su vez, señalan un valor que influye en los resultados del modelo, en donde:

- p: es el coeficiente del término AR
- q: es el coeficiente del término MA
- d: es el grado de diferenciación para que la serie de tiempo sea estacionaria

La letra "p" es el orden del término "Auto regresivo" (AR) se refiere al número de rezagos de Y que se utilizan como predictores. Un modelo auto regresivo puro (solo AR) es aquel en el que Y_t depende solo de sus propios retrasos. Es decir Y_t es una función de los rezagos de Y_t. Este modelo puede ser expresado por la siguiente ecuación:

$$Y_t = \alpha + \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \varepsilon_t$$

Donde:

α: es una constante.

$\phi_1 \ldots, \phi_p$: Parámetro que define el modelo (valor a estimar).

ε_t: es el termino de error.

La letra "q" es el orden del término "Media móvil" (MA)se refiere al número de errores de pronóstico retrasado que se debe incluir en el modelo Un modelo puro de media móvil (solo MA) es aquel en el que Y_t depende solo de los errores de pronósticos retrasados. Este modelo puede ser expresado por la siguiente ecuación:

$$Y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_p \varepsilon_{t-p} \quad (6)$$

Donde:

ε_t: es el término de error.

$\theta_1 \ldots, \theta_p$: Parámetro que define el modelo (valor a estimar).

La letra “d” es el grado de diferenciación “Integrado” (I) el cual indica la cantidad de diferenciaciones que se deben realizar para que la serie o conjunto de datos pueda lograr la estacionariedad., un modelo ARMA (p, q) es equivalente a un moldeo ARIMA de orden (p,0, q), Este último modelo puede ser expresado por la siguiente ecuación:

$$Y_t = \alpha + \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_p \varepsilon_{t-p} \quad (7)$$

Un modelo ARIMA es aquel en el que la serie de datos se diferencia al menos una vez para hacerla estacionaria, y se combinan los términos "AR" y "MA". Por lo cual se debe conocer si la serie de datos es estacionaria. Si lo es, entonces d = 0, por lo cual sería un modelo ARMA. En caso contrario, donde la serie de datos no es estacionaria, se debe buscar la cantidad de diferenciaciones necesarias para que esta lo sea. Dependiendo de la cantidad de veces que se realice esta diferenciación, será el valor que obtendrá

1.4.1. Dickey Fuller

Una manera de corroborar si la serie de datos es estacionaria, es con la prueba Aumentada de Dickey Fuller, es una prueba de raíz única que detecta estadísticamente la presencia de conducta tendencial estocástica (es decir, no estacionaria) en las series temporales de las variables mediante un contraste de hipótesis.

La hipótesis nula (H0) de esta prueba se refiere la presencia de tendencia estocástica en las observaciones, la hipótesis alternativa rechaza la hipótesis nula (H1) y establecemos no tendencia estocástica en las observaciones.

- H0: Hipótesis nula, la serie no es estacionaria y tiene tendencia estocástica en las observaciones.

- H1: Hipótesis nula es rechazada, la serie es estacionaria y no tiene tendencia estocástica en las observaciones.

Para interpretar de mejor manera los resultados que se obtienen de la prueba, se utiliza el valor de “p”, teniendo dos intervalos:

- Si “p” > 0.05: No se rechaza la hipótesis nula (H0), los datos no son estacionarios.

- Si “p” <= 0.05: Se rechaza la hipótesis nula (H0), los datos son estacionarios.

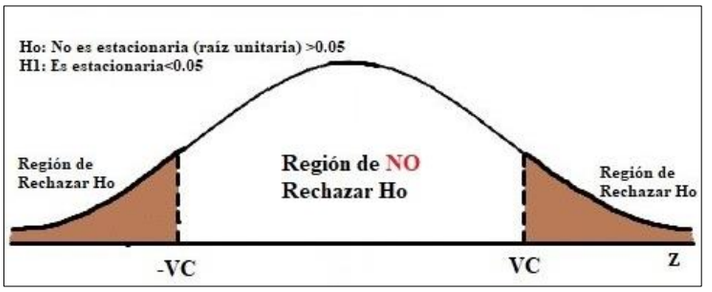

Figura 1-6. Prueba Dickey Fuller para raíz unitaria.
Fuente: Naren Castellón. Estacionariedad en la Serie de Tiempo

1.4.2. Ljung-Box

La prueba estadística de Ljung-Box consiste en verificar si dentro de una serie existe autocorrelación entre los residuos (es decir saber si existe ruido blanco), para corroborar esto se utiliza la siguiente hipótesis.

- H0: Hipótesis nula, los residuos se distribuyen de forma independiente (Existe ruido blanco).

- H1: Se rechaza la hipótesis nula, y los residuos no se distribuyen de forma independiente (No existe ruido blanco).

Para aceptar o rechazar la hipótesis plantada se utiliza el p-valor de Ljung-Box, en donde:

- Si “p” > 0.05: No se rechaza la hipótesis nula (H0), los residuos del modelo se distribuyen de forma independientes (Existe ruido blanco).

- Si “p” <= 0.05: Se rechaza la hipótesis nula (HA), los residuos del modelo se distribuyen de forma dependientes (No existe ruido blanco).

1.4.3. Aplicación del modelo ARIMA

Para lograr el desarrollo deseado del modelo, se utiliza el software Python, el cual, solo diseñando un algoritmo, es capaz de seguir un proceso coherente en el trabajo. Por lo cual, lo primero que se realiza es una prueba de raíz única que detecta estadísticamente la

presencia de conducta tendencial estocástica, con el fin de estar al tanto si nuestra serie es estacionaria. Además, se pueden utilizar gráficas de función de autocorrelación y función de autocorrelación parcial para determinar "p" y "q". Por otra parte, una vez obtenida la serie estacionaria, se debe estimar un modelo acorde con la estructura de este y, finalmente, el modelo obtenido se debe validar a través de las correlaciones de los valores residuales para comprobar si nuestro modelo se adapta de buena forma con el fin de obtener buenas predicciones en el modelo. A continuación, se expresa de manera más sencilla el funcionamiento del modelo ARIMA.

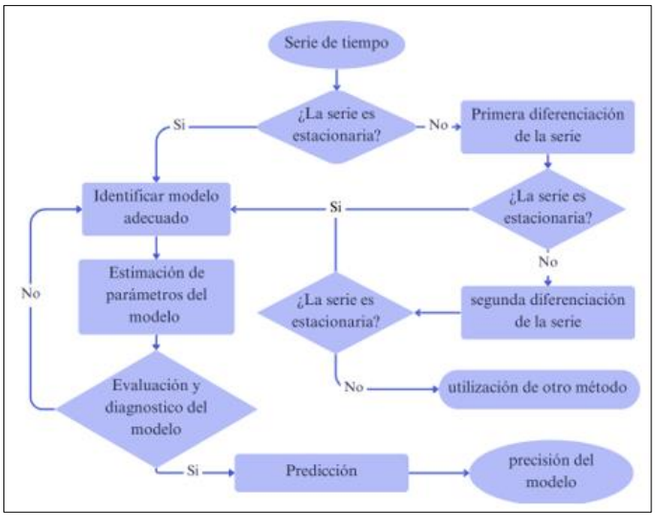

Figura 1-7. Diagrama de flujo modelo ARIMA.
Fuente: Elaboración propia a través de canva.

1.5. RANDOM FOREST (RF)

Random Forest (RF) es un algoritmo de machine learning de aprendizaje supervisado de uso común registrado por Leo Breiman y Adele Cutler. Está formado por un conjunto de árboles de decisión individuales. Por lo tanto, primero debemos comprender lo que es un árbol de decisión.

Un árbol de decisión básicamente realiza una serie de preguntas verdaderas o falsas que conducen a una respuesta determinada. De esta manera, se logra dividir los datos de entrada en función de sus características y así predecir la variable de salida. Estos tienen una estructura de árbol jerárquica que consta de tres tipos diferentes de nodos: nodos de probabilidad, nodos de decisión y nodos terminales.

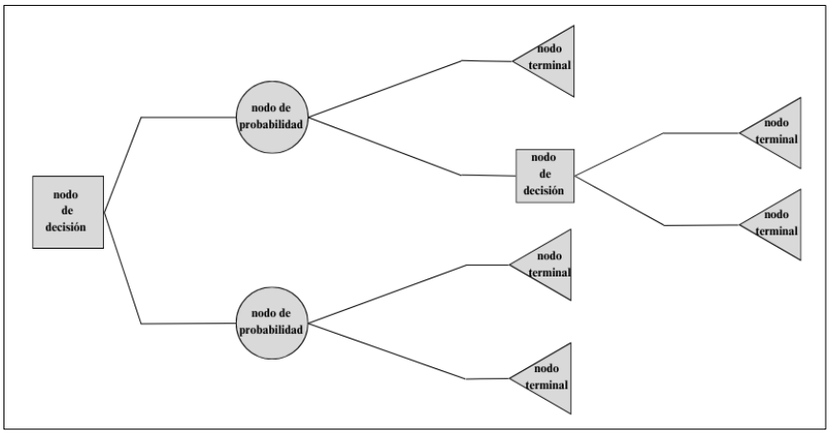

Figura 1-8. interpretación de un árbol de decisión.
Fuente: Elaboración propia a través de canva.

Los árboles de decisión también se pueden representar con símbolos de diagramas de flujo, lo cual a algunas personas les parece más fácil de leer y comprender.

Figura	Nombre	Significado
	Nodos de decisión	Indica una decisión que se tomará
	Nodos de probabilidad	Muestra múltiples resultados inciertos
	Ramificaciones alternativas	Cada ramificación indica un posible resultado o acción
	Nodos terminales	Indica un resultado definitivo

Tabla 1-2. Símbolos de los árboles de decisión.
Fuente: Elaboración propia a través de Excel.

Como mencionamos anteriormente un modelo RF está formado por un conjunto de árboles de decisión individuales combinados con bagging. Al usar bagging, lo que en realidad está pasando, es que cada árbol se entrena con unos datos ligeramente distintos. En cada árbol individualmente. Ningún árbol ve todos los datos de entrenamiento. Esto hace que cada árbol se entrene con distintas muestras de datos para un mismo problema, las observaciones se van distribuyendo por bifurcaciones (nodos) generando la estructura del árbol hasta alcanzar un nodo terminal.

Bagging es una técnica usada para reducir la varianza de las predicciones a través de la combinación de los resultados de varios clasificadores, cada uno de ellos modelados con diferentes subconjuntos tomados de la misma data set como se muestra en la figura a continuación.

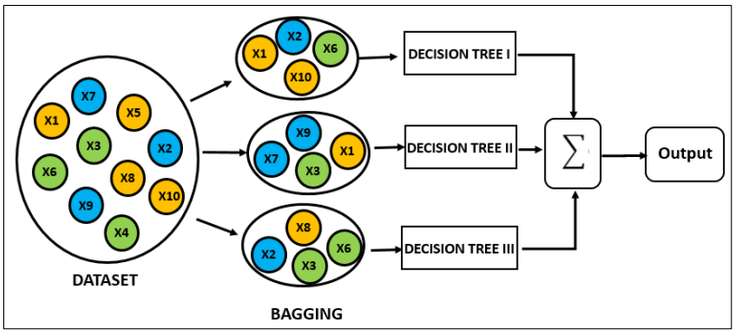

Figura 1-9. Implementación de bagging.
Fuente: Johanna Orellana Alvear. Árboles de decisión y Random Forest.

Las muestras generadas después del proceso de Bagging son el inicio para cada árbol. Por lo tanto, la correlación entre los resultados que se obtienen disminuye, aumentando la precisión del modelo al tener diferencias en sus datos de entrada. Esto se debe a que los resultados obtenidos por cada árbol se clasificarán y solo se seleccionará el que se encuentre con mayor frecuencia como resultado final.

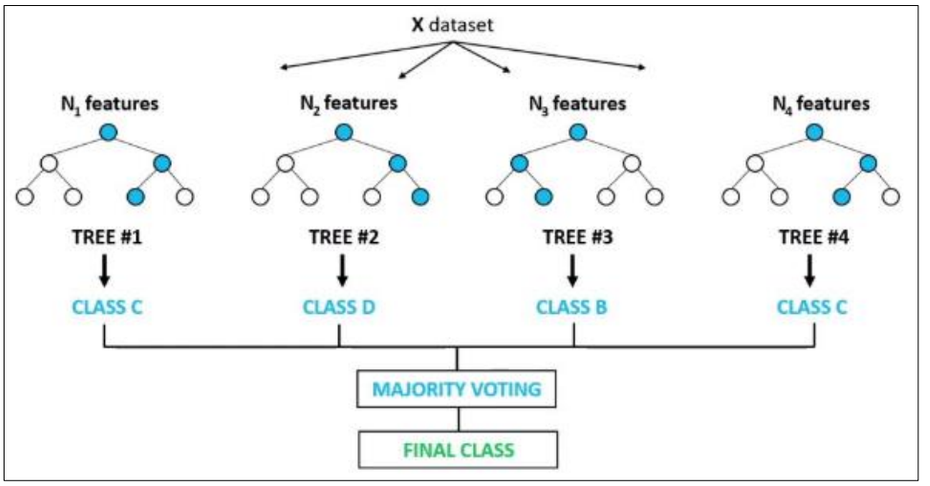

Figura 1-10. Funcionamiento Random Forest.
Fuente: Fernando Cardellino. Tutorial para un clasificador basado en bosques aleatorios.

Al visualizar la figura 1-10, se demuestra el funcionamiento de este algoritmo, el cual comienza a crear árboles de decisión para cada muestra seleccionada. Posteriormente, obtendrá un resultado de predicción de cada árbol creado. Finalmente, se puede observar cómo el modelo predice, donde en cada árbol se destaca el camino que tomó hasta finalmente llegar a un nodo terminal.

1.5.1. Aplicación del algoritmo de Random Forest

Para lograr el desarrollo deseado del modelo, se utiliza el software Python, el cual, solo diseñando un algoritmo, es capaz de realizar el funcionamiento. Este consiste

inicialmente en dividir los datos desde la base de datos, de tal manera que el 20% de los datos disponibles se utilicen para prueba o testeo, y el 80% restante se utilice para la etapa de entrenamiento. Luego, se define el algoritmo Random Forest que se utilizará para entrenar el modelo, de tal forma de obtener predicciones en base a los datos de testeo. A continuación, se expresa de manera más sencilla el funcionamiento del modelo Random Forest.

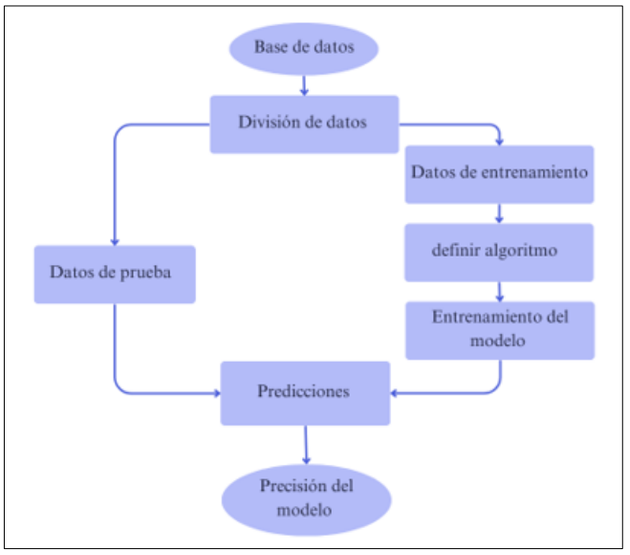

Figura 1-11. Diagrama de flujo modelo Random Forest.
Fuente: Elaboración propia a través de canva.

CAPITULO 2: DESARROLLO DE METODOLOGÍAS PROPUESTAS

2.1. HISTORICO DE FALLOS DEL COMPONENTE CRITICO

En la información del histórico de datos a analizar para desarrollar los distintos modelos, estos se encuentran ordenados de forma cronológica, donde el inicio es a partir del 8 de enero de 2016 y finaliza el 24 de mayo del 2022. Por lo cual, aproximadamente contamos con 6 años recompilando datos. Además, contamos con el tiempo de buen funcionamiento (TBF), también conocido como tiempo entre falla. Esta base de datos cuenta con una cantidad de 81 eventos de fallas correspondientes al componente Motor Diesel.

A través del lenguaje de programación Python, es posible graficar los valores de TBF del componente crítico del camión minero. Dentro de todos los eventos de fallas presentes, el mayor tiempo de buen funcionamiento es de 7897 horas. Este evento muestra un pico en el Gráfico 2-1. Además, se puede observar una gran cantidad de picos a lo largo del gráfico. Esto podría demostrar que las intervenciones de mantenimiento preventivo causan un aumento en los intervalos entre falla. Por último, para el modelo de distribución de Weibull, estos picos generan una gran dificultad para entregar resultados más cercanos a los reales. En primera instancia, esto reduce la precisión de dicho modelo.

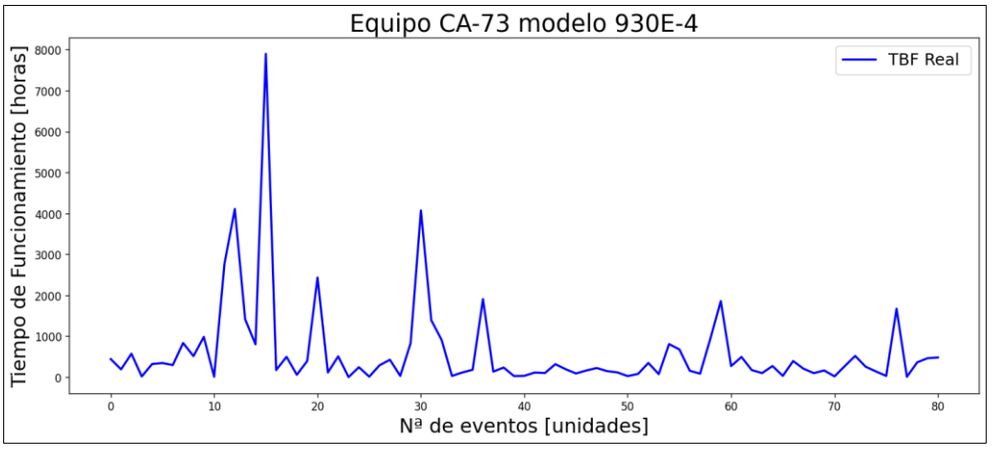

Gráfico 2-1. Gráfico de tiempo de funcionamiento real.
Fuente: Elaboración propia en Python.

2.2. DESAROLLO DE DISTRIBUCIÓN WEIBULL

Mediante programación en Python y basándonos en la subsección 1.3.3 (Procedimiento para obtener parámetros Weibull ajustado), procedemos a ejecutar los cálculos de la distribución de Weibull ajustado. Además, en el Gráfico 2-2, se pueden

apreciar gráficamente los valores obtenidos mediante el modelo y los tiempos reales hasta la falla.

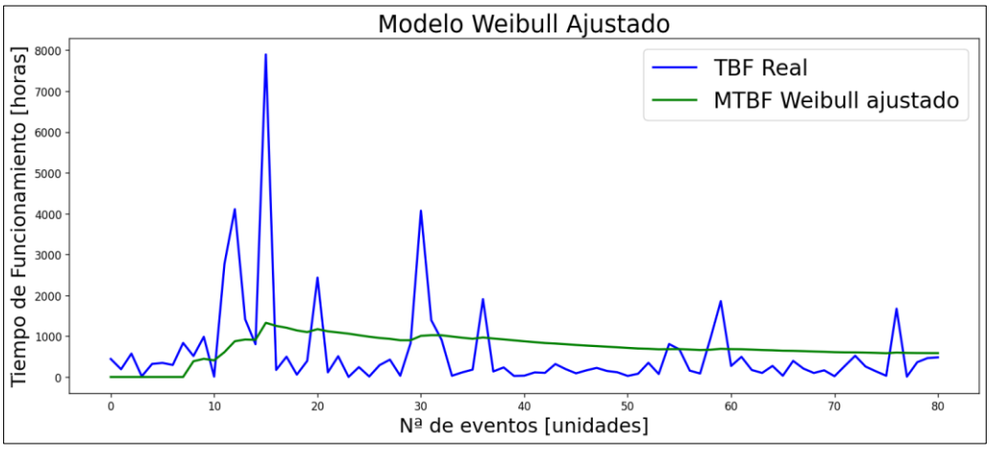

Gráfico 2-2. Gráfica de tiempo de funcionamiento real y predicción de modelo Weibull. Fuente: Elaboración propia en Python.

En consecuencia, en el gráfico se puede apreciar que los primeros 8 valores del MTBF de Weibull ajustado equivalen a 0. Esto se debe a que con los primeros 8 tiempos de funcionamiento se obtuvo el noveno MTBF. Además, en los valores obtenidos se presenta una gran discrepancia entre el valor del TBF real y el comportamiento de los resultados de la estimación de Weibull.

2.2.1. Resultados del modelo Weibull ajustado

Los resultados del modelo Weibull obtenidos al considerar la totalidad de los datos en solo una iteración, nos entrega como resultado los parámetros Weibull ajustado, en donde el valor de Alpha (α) es de 464,3 [hrs], el valor de Beta (β) obtenido es de 0,7, basándonos en la subsección 1.3.1. (Parámetros de Weibull) y considerando que el factor de forma es menor a 1 ($\beta < 1$), esto indica que se encuentra en la Etapa "I", donde aparecen fallas tempranas, la tasa de fallas es decreciente y está asociado directamente a un equipo nuevo que se encuentra en rodaje o en una etapa inicial de su vida útil.

Como consecuencia el valor del MTBF esperado para la totalidad de los datos es de 584,8 [hrs], lo que es equivalente a 22 días de operación continua sin presentar fallos. A pesar de todo el valor máximo real de funcionamiento presente en el histórico de datos es de 7897,3 [hrs], lo que representa un equivalente de 329 días sin presentar fallos o averías, por otro lado, el valor máximo de predicción es de 1326,8 [hrs], lo que representa un equivalente de 55 días sin presentar fallos, por otra parte el valor mínimo real de funcionamiento corresponde a 1 [hrs], lo que no alcanza a equivaler a 1 día de operación,

en cambio el valor mínimo de predicción es de 407,2 [hrs], lo que representa un equivalente de 16 días sin presentar fallos o averías.

Para finalizar con lo propuesto la utilización de la distribución de Weibull ajustado para la predicción de fallas presenta grandes errores en los cálculos de predicción como se puede apreciar en el Gráfico 2-2, lo que conlleva a errores al momento de estimar el tiempo oportuno de intervención del componente crítico, por lo tanto, se requiere la aplicación de otros modelos con la finalidad de obtener mejores resultados de predicción.

2.3. DESAROLLO DE MODELO ARIMA

2.3.1 Prueba Dickey Fuller

Mediante programación en Python y basándonos en la subsección 1.4.1.(Dickey Fuller), se logra aplicar la prueba de Dickey Fuller, obteniendo como resultado que el valor de p es de $1{,}806 * 10^{-13}$, claramente este valor es muy cercano a 0 por lo tanto, al utilizar la regla anteriormente descrita se rechaza la hipótesis nula, declarando que la serie temporal es estacionaria.

Por lo tanto, esta metodología se discierne el parámetro d, el cual corresponde a la cantidad de diferenciaciones que se tuvieron que realizar para que la serie tuviese un comportamiento estacionario. Para este caso, es d=0.

2.3.2 Diseño ARIMA

Se debe establecer si existe la necesidad de que el modelo requiera un término AR. Una forma de hacerlo es verificando el grafico de autocorrelación parcial, donde el orden del término AR corresponde a la cantidad de rezagos que cruzan el límite de significancia en la gráfica. A continuación, en el Gráfico 2-3. se puede apreciar el grafico de autocorrelación parcial.

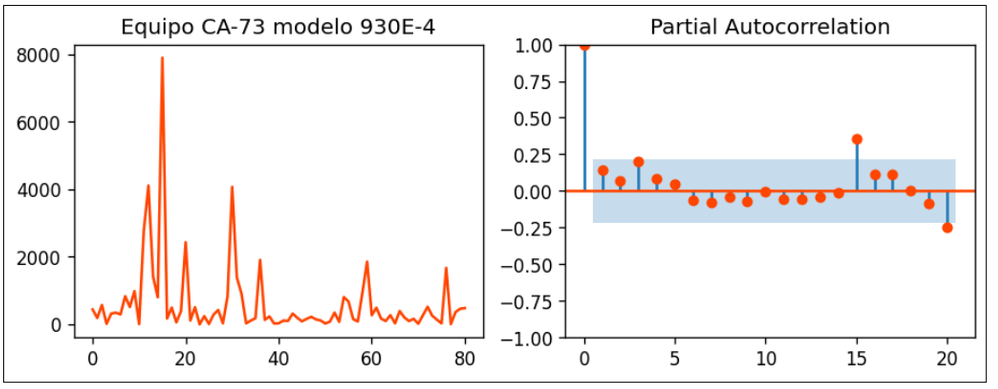

Gráfico 2-3. Gráfico de autocorrelación parcial.

Fuente: Elaboración propia en Python.

Como consecuencia del Gráfico 2-3 Podemos observar que se presenta un solo rezago significativo el cual está muy por encima del límite de significancia (zona azul), también se puede observar que el ultimo rezago también cruza levemente el límite por lo cual no se considerara por lo cual el parámetro p (AR) se fijara como 1.

El segundo aspecto que se debe establecer si existe la necesidad de que el modelo requiera un término MA. Una forma de hacerlo es verificando el grafico de autocorrelación, donde el orden del término MA corresponde a la cantidad de rezagos que cruzan el límite de significancia en la gráfica. A continuación, en el Gráfico 2-3 se puede apreciar el grafico de autocorrelación.

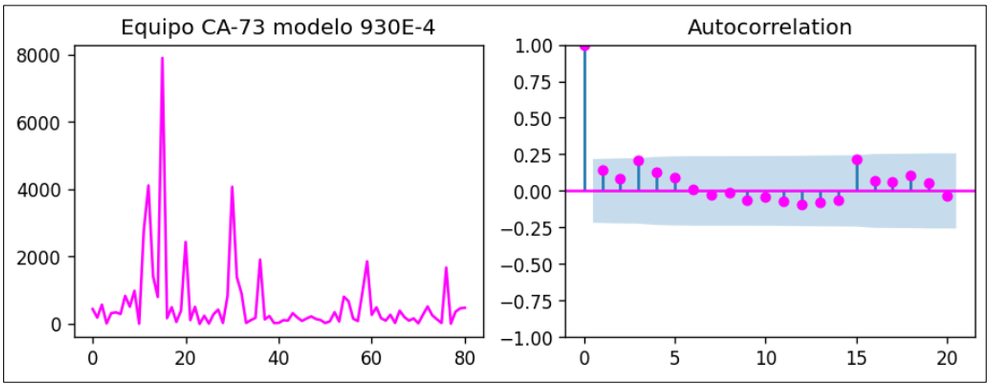

Gráfico 2-4. Gráfico de autocorrelación.

Fuente: Elaboración propia en Python.

Como consecuencia del Gráfico 2-4 podemos observar que no se presenta ningún rezago el cual este por encima del límite de significancia (zona azul) por lo cual el parámetro q (MA) se fijara como 0.

En resumen, para el modelo ARIMA con el cálculo de los parámetros de forma manual se obtuvo un modelo ARIMA (1,0,0). En el Gráfico 2-5, se puede observar los resultados obtenidos, donde se aprecian los valores predichos por el modelo ARIMA.

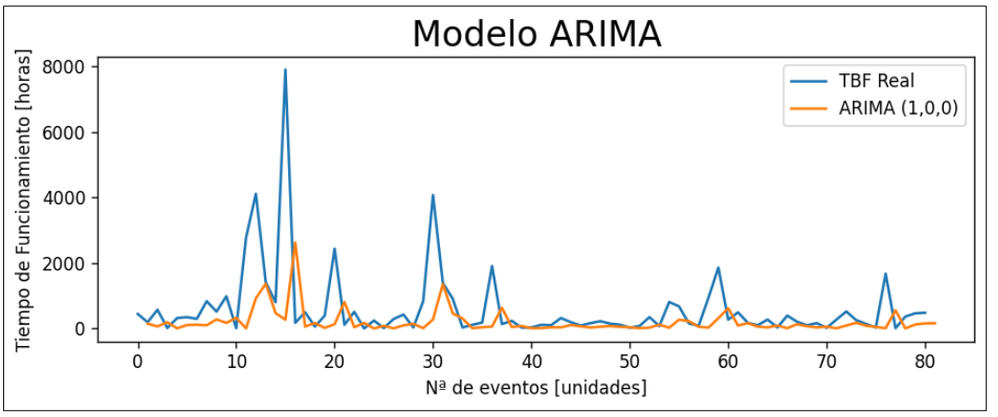

Gráfico 2-5. Gráfica de tiempo de funcionamiento real y predicción de modelo ARIMA (1,0,0).
Fuente: Elaboración propia en Python.

Este resultado puede mejorar con la utilización del modelo auto ARIMA, el cual se desarrollará a continuación con la finalidad de obtener mejores predicciones en base a los TBF reales del equipo.

2.3.3. AUTO ARIMA

Mediante programación en Python se ejecutará la función de auto ARIMA presente en la librería de este. El proceso de auto ARIMA buscará sus propios valores óptimos de (p, d, q) en base a la cantidad de datos que contenga nuestro TBF real.

Una vez ejecutado el proceso de auto ARIMA se obtuvo un nuevo modelo en el cual nos entregó como resultado un modelo ARIMA (1,1,0). Si bien no era necesario diferenciar la serie la función auto ARIMA nos entrega que realizando una diferenciación en la serie de datos obtendremos mejores predicciones. En el Gráfico 2-6 se observan los resultados obtenidos.

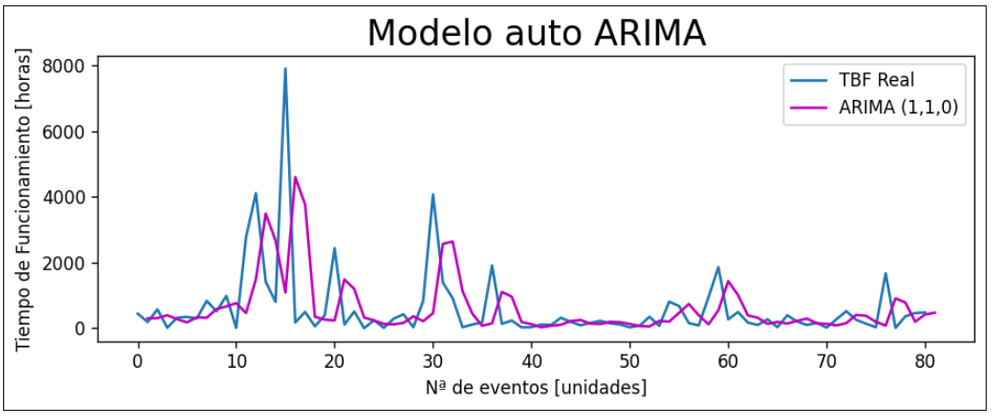

Gráfico 2-6. Gráfica de tiempo de funcionamiento real y predicción de modelo auto ARIMA.
Fuente: Elaboración propia en Python.

En relación con el Gráfico 2-6 las predicciones mejoran utilizando este modelo, en comparación a las estimaciones realizadas con el modelo de Weibull ajustado, ARIMA, en el Gráfico 2-7 se puede observar una gráfica con todos los modelos realizados hasta el momento.

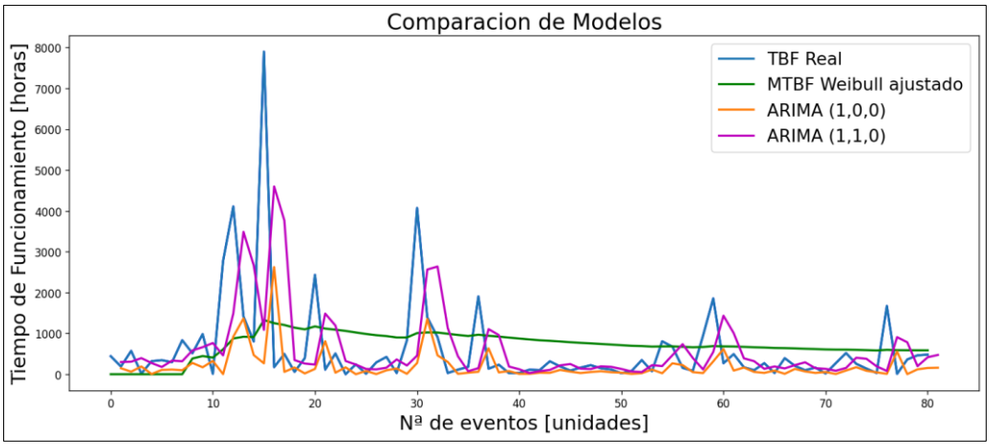

Gráfico 2-7. Gráfica de comparación de modelos.
Fuente: Elaboración propia en Python.

Como consecuencia, el modelo auto ARIMA obtiene los mejores resultados en comparación con el modelo de Weibull y ARIMA, debido a que el auto ARIMA se ajusta más al comportamiento que tiene el histórico de fallos. El valor de "p" al ejecutar la prueba de Ljung-Box es de 3,11, siendo este mayor a 0,05. Por lo tanto, no se rechaza la hipótesis nula, lo que indica que nuestro modelo se ajusta bien y existe ruido blanco. A pesar de todo, se presentan errores de predicción. Para mejorar los resultados de predicción, se utilizará Random Forest, el cual logra predecir el comportamiento con mayor exactitud.

2.4. DESARROLLO DE RANDOM FOREST

2.4.1 Datos de entrenamiento y prueba

Mediante programación en Python y basándonos en la subsección 1.5.1. (Aplicación del algoritmo de Random Forest), se procede a realizar el modelo Random forest en donde lo primero que se logra ejecutar fue la división de los datos de tal manera que 20% de los datos disponibles se utilizaron de prueba o testeo y el 80% restantes se utilizaron para la etapa de entrenamiento. A continuación, en el Gráfico 2-8. se puede apreciar el grafico de la división de datos.

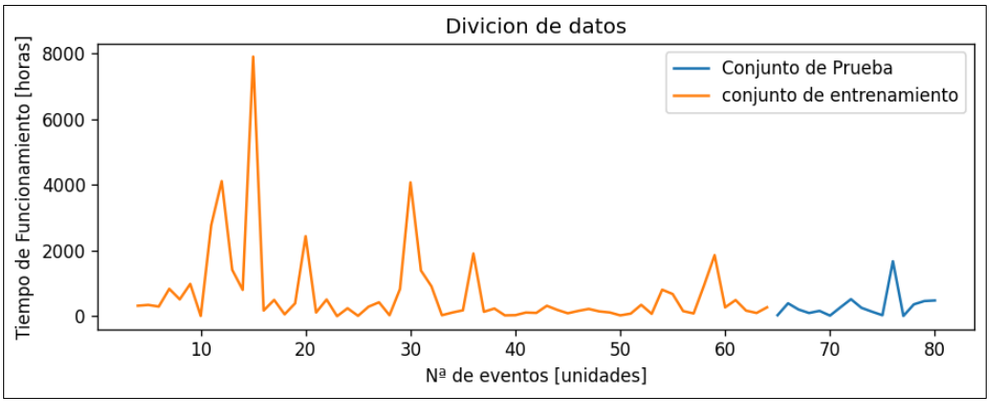

Gráfico 2-8. Gráfica de división de datos de entrenamiento y prueba.

Fuente: Elaboración propia en Python.

Por ende, como se muestra en el Gráfico 2-8, para la etapa de entrenamiento se utilizarán los primeros 64 valores de la serie de datos, mientras que para la etapa de prueba se utilizarán los últimos 16 valores. El objetivo consiste en definir los parámetros del modelo Random Forest que mejor se ajuste a los datos de entrenamiento, para luego realizar una predicción sobre datos nuevos con la mayor precisión y exactitud posible.

2.4.2 Random Forest con auto ARIMA

Para lograr el desarrollo deseado del modelo se utilizarán los datos predichos obtenidos mediante el modelo de auto ARIMA y todos los datos de TBF reales, siendo una única variable predictora, se utilizarán 50 árboles para que trabajen en conjunto para tener mayor probabilidad de predicciones y de esta manera se logren resultados con mayor exactitud. A continuación, en el Gráfico 2-9. se puede apreciar el grafico con las predicciones del modelo Random Forest con auto ARIMA.

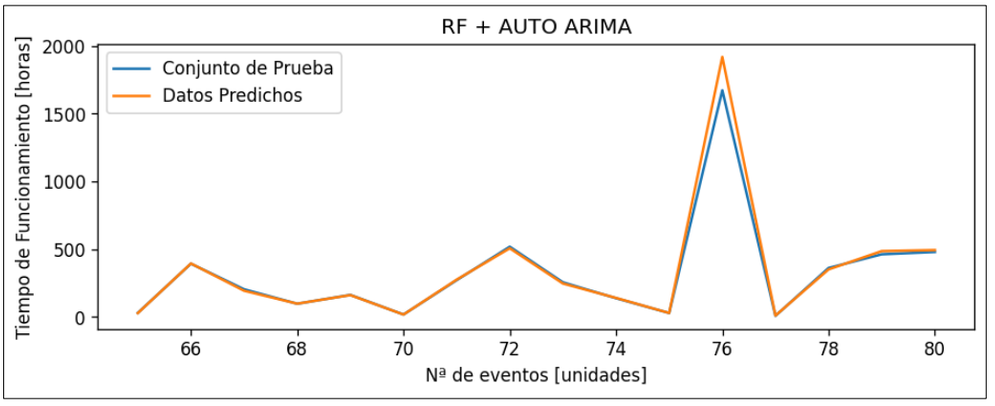

Gráfico 2-9. Gráfica de predicciones con Random Forest con auto ARIMA y datos de prueba.

Fuente: Elaboración propia en Python.

En el Gráfico 2-9, se logra apreciar como el modelo se ajusta a los datos de prueba (color azul), logrando una predicción (color amarillo) en donde se muestra que se obtuvieron unos valores de predicción bastante certeros que solo se presenta un mayor error en el pick del evento numero 76 aproximadamente. Pese a las predicciones obtenidas ajustando el modelo Random forest se pueden mejorar aún más la exactitud de estas.

2.4.3. Ajuste al modelo Random Forest

Si bien los resultados obtenidos con el modelo anterior son prometedores, aún existe un rango de error es por ello, que finalmente se propone un último modelo, el cual contempla como datos de entrada solo valores reales del histórico de fallos, utilizando los valores de tiempos fuera de servicio y los valores de TBF reales. Además, se utilizarán 50 árboles igual que en el modelo anterior para obtener resultados con mayor exactitud. A continuación, en el Gráfico 2-10. se puede apreciar el grafico con las predicciones del modelo Random Forest ajustado.

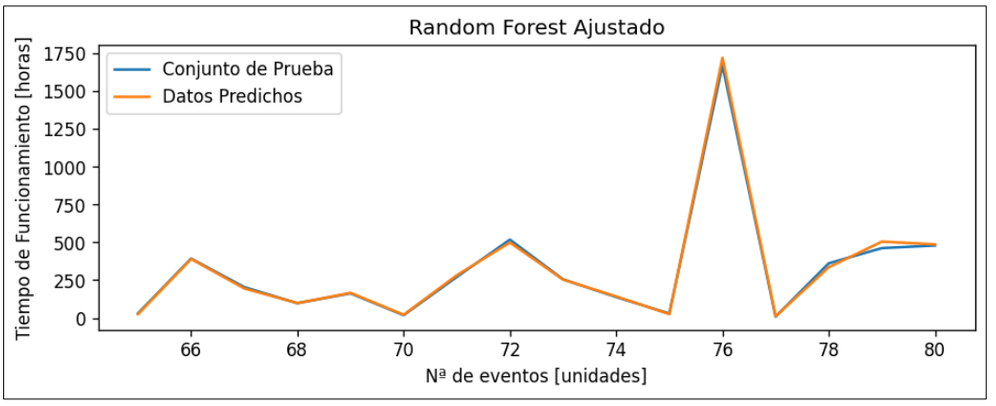

Gráfico 2-10. Gráfica de predicciones con Random Forest ajustado y datos de prueba.
Fuente: Elaboración propia en Python.

Al ejecutar el código nuevamente se puede apreciar en el Gráfico 2-10, como el modelo se ajusta a los datos de prueba (color azul), logrando una predicción (color amarillo) en donde se muestra que se obtuvieron unos valores de predicción bastante certeros con un error en la predicción bastante bajo además es el que obtiene las mejores métricas de rendimiento en cuanto a todos los modelos propuesto.

CAPITULO 3: EVALUACIÓN DE LOS MODELOS DE PREDICCIÓN PROPUESTOS

3.1. MÉTRICAS DE RENDIMIENTO

Cuando se formula un modelo de predicción, el objetivo primordial es lograr la mayor precisión posible mediante ajustes iterativos de sus parámetros. No obstante, es crucial reconocer la imposibilidad de crear un modelo completamente preciso, ya que la total ausencia de errores es inalcanzable. En consecuencia, comprender las diversas fuentes de error se convierte en un elemento clave para perfeccionar la precisión de los modelos. Por otra parte, evaluar el rendimiento de un algoritmo se vuelve esencial para asegurar su correcto funcionamiento. Para este propósito, se emplean métricas de rendimiento de regresión. En el ámbito estadístico, se recurre a la diferencia entre dos variables continuas (datos calculados y observados) como indicador, permitiendo cuantificar la exactitud y precisión de un modelo de predicción al comparar los valores predichos con los observados.

Dentro del contexto de este proyecto de titulación, se han integrado métricas particulares de exactitud y precisión que suministran datos relevantes sobre los modelos implementados. A continuación, se procederá a establecer y describir las métricas de exactitud que servirán como base para la comparación entre los diferentes modelos propuestos.

3.1.1 Error absoluto medio (MAE)

El MAE (Mean Absolute Error) es una medida de evaluación de errores muy utilizadas comúnmente en la evaluación de modelos de predicción o regresión. Por lo tanto, para obtener el MAE, se calcula la media de los errores absolutos de toda la observación. Un MAE bajo indica que el modelo está produciendo predicciones precisas. se obtiene mediante la ecuación 8.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{8}$$

Donde:

- n: representa el número de observaciones
- y_i: representa el valor real
- $\hat{y}_i$: representa el valor predicho

3.1.2. Error cuadrático medio (MSE)

En cuanto al MSE tal como dice su nombre es la media de la diferencia entre el valor predicho y el valor real elevado al cuadrado. Entonces, a diferencia del MAE que encontramos la diferencia absoluta con este encontramos la diferencia al cuadrado. Por lo cual les da mucho más peso a los grandes errores (valores atípicos) que a los más pequeños debido a esto es de más utilidad cuando se trata de grades errores que cuando se trata de pequeños errores. Finalmente, un buen modelo tendrá un valor MSE más cercano a cero y se obtiene mediante la ecuación 9.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{9}$$

3.1.3. Raíz del error cuadrático medio (RMSE)

La Raíz del Error Cuadrático Medio conocido por sus siglas RMSE (Root Mean Squared Error). Se calcula tomando la raíz cuadrada de MSE y este mide la magnitud promedio de los errores y se ocupa de las desviaciones del valor real. Es una estadística imperfecta para la evaluación, pero es muy común. Si te importa penalizar errores grandes, no es una mala opción. Finalmente, un RMSE bajo también indica que el modelo está produciendo predicciones precisas. se obtiene mediante la ecuación 10.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{10}$$

3.1.4. Error porcentual absoluto medio (MAPE)

El error porcentual absoluto medio (MAPE) es una medida de la precisión de un sistema de pronóstico. Expresa la exactitud como un porcentaje del error. Debido a que el MAPE es un porcentaje, puede ser más fácil de entender que otras métricas de medición de exactitud. Debido a que, si el MAPE es 5, en promedio, el pronóstico está errado en un 5%. Finalmente, un MAPE mientras más bajo mejor. Este se obtiene mediante la ecuación 11.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{11}$$

3.1.4. Coeficiente Determinación (R^2)

También conocido como prueba de bondad de ajuste, es una métrica que indica el rendimiento del modelo, nos indica que tan bien se ajusta el modelo a las observaciones reales que tenemos, sin embargo, no dice nada sobre el modelo en sí, solo indica si el modelo es bueno o es malo dependiendo del valor de R^2 que varía de 0 a 1, los valores más cercanos a 1 indican ajuste fuerte mientras los valores más bajos indican un ajuste débil. Este se obtiene mediante la ecuación 12.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2} \quad (12)$$

Donde:

- $\bar{y}_i$: Es la media de y
- y_i: representa el valor real
- $\hat{y}_i$: representa el valor predicho

3.2. RESULTADOS DE PREDICCIÓN MODELO WEIBULL

Respecto al modelo de Weibull ajustado el cual genero predicciones en base a la cantidad de datos que posee el histórico de fallos, se logró estimar un MTBF esperado el cual se comparó con el TBF real del histórico de fallos del componente crítico para poder estimar la precisión de dicho modelo. A continuación, En la tabla 3-1. se puede apreciar las métricas de rendimiento obtenidas para el modelo Weibull.

Modelo	MAE	MSE	RMSE	MAPE	R^2
Weibull ajustado	755,89 [hrs]	133292,63[hrs]^2	1156,41[hrs]	2270,4	0,05

Tabla 3-1. Métricas de rendimiento modelo Weibull ajustado.
Fuente: Elaboración propia a través de Excel.

En la tabla 3-1 se pueden apreciar los errores que presenta el modelo analizado, donde el modelo de Weibull ajustado entrega un error absoluto medio con un valor de 755.89 [hrs], esto indica que en promedio los valores predichos por el modelo se encuentran alejados de los TBF reales. Por otra parte, el Coeficiente de determinación (R^2) con un valor 0,05 nos indican un ajuste débil por lo cual el modelo no se ajusta adecuadamente a las observaciones reales que tenemos. Por otra parte, estos primeros

resultados de las métricas utilizadas, en este modelo no se puede inferir si es preciso en primera instancia, ya que, se requiere de un punto de comparación y para ellos se utilizarán los otros modelos propuestos. Como consecuencia en la Figura 3-1 se aprecia el error absoluto de cada dato predicho por el modelo Weibull ajustado.

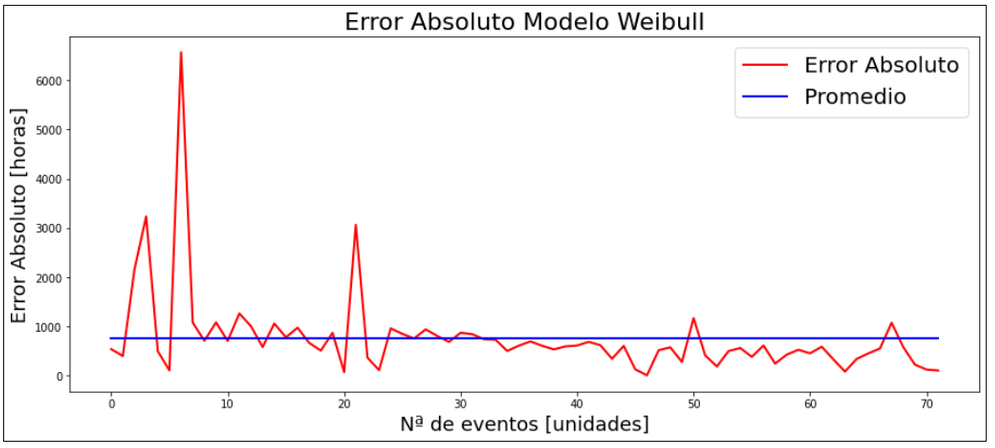

Figura 3-1. Error absoluto en la predicción de eventos con modelo Weibull.
Fuente: Elaboración propia en Python.

Al visualizar la figura 3-1. se evidencia que el error máximo absoluto se posiciona en 6570 [hrs]. Al proyectar este desfase a una escala más amplia, se revela que esta discrepancia representa una desviación significativa en la predicción del evento, llegando a aproximadamente 9 meses de diferencia. Este hallazgo resalta la importancia de ajustar y perfeccionar los modelos de predicción para lograr una mayor precisión temporal. Por otro lado, es notable mencionar que, en el extremo opuesto, el error mínimo de aproximación se registra en 7[hrs], indicando una capacidad relativamente precisa del modelo en ciertos puntos temporales con respecto al tiempo real.

3.3. RESULTADOS DE PREDICCIÓN MODELO ARIMA

En relación con el modelo ARIMA, se lograron crear dos modelos distintos para la predicción de fallos. En el primer escenario, se procedió a la estimación manual de los parámetros, culminando con un modelo ARIMA (1,0,0). Por otra parte, se implementó el método de auto ARIMA, el cual automatizó la generación de los parámetros del modelo obteniendo un modelo ARIMA (1,1,0) para la predicción de fallos. Estos dos métodos fueron posteriormente comparados con el Tiempo Entre Fallos (TBF) real del histórico de fallos del componente crítico, con el objetivo de evaluar la precisión de dichos modelos. En la tabla 3-2 se presentan las métricas de rendimiento obtenidas para ambos modelos.

Modelo	MAE	MSE	RMSE	MAPE	R^2
ARIMA (1,0,0)	407,68 [hrs]	734620,48[hrs]^2	857,1[hrs]	496,95	0,42
Auto ARIMA (1,1,0)	338,93 [hrs]	472680,01[hrs]^2	687,51[hrs]	425,09	0,62

Tabla 3-2. Métricas de rendimiento modelos ARIMA.

Fuente: Elaboración propia a través de Excel.

En la Tabla 3-2. se evidencian los errores encontrados en los modelos analizados. Al contrastar ambos modelos presentes, se observa una diferencia significativa. Para determinar el modelo óptimo, se debe optar por aquel que obtenga valores más bajos en las métricas de rendimiento, ya que esto indicaría un ajuste más preciso. En consecuencia, de este criterio podemos concluir que el modelo auto ARIMA se posiciona como la elección más adecuada. dado que presenta un error absoluto medio de 338.93 [hrs]. Este valor sugiere que, en promedio, las predicciones del modelo se distancian de los tiempos entre fallas reales.

Además, es importante destacar el Coeficiente de Determinación (R^2) en este análisis, el cual exhibe un valor de 0,62. Este coeficiente nos proporciona información sobre el grado de ajuste del modelo a las observaciones reales. En este caso, el valor de 0.62 indica un ajuste moderado, lo que significa que el modelo presenta una capacidad aceptable en los datos observados. Sin embargo, es crucial señalar que aún existen oportunidades para mejorar este ajuste.

En contraste con los resultados arrojados por el modelo Weibull, el modelo auto ARIMA presenta una mejora significativa en las métricas de rendimiento propuestas. Esta mejora se refleja claramente en la Figura 3-2, donde se visualiza el error absoluto asociado a cada predicción realizada por el modelo auto ARIMA. Este último, cabe destacar, ha demostrado un desempeño superior en comparación con el modelo Weibull, al obtener métricas de rendimiento más favorables.

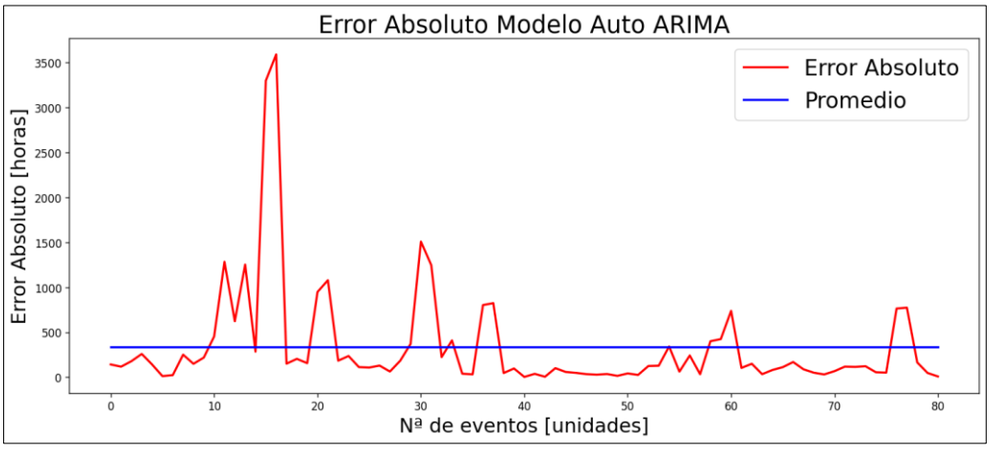

Figura 3-2. Error absoluto en la predicción de eventos con modelo Auto ARIMA.

Fuente: Elaboración propia en Python.

Al analizar detenidamente la figura 3-2, se aprecia claramente que el error absoluto máximo, en comparación con el modelo Weibull, ha experimentado una disminución de aproximadamente 3000[hrs]. Como consecuencia directa de esta mejora, el error máximo absoluto presente en el modelo auto ARIMA se sitúa ahora en 3591[hrs]. Al proyectar esta discrepancia a una escala temporal más extensa, se revela que esta diferencia implica una desviación sustancial en la predicción del evento, con un lapso de aproximadamente 5 meses. Por otro lado, es digno de mencionar que, en el extremo opuesto del espectro, se registra un error mínimo de aproximación de tan solo 2 [hrs]. Este dato indica una capacidad relativamente precisa del modelo en ciertos puntos temporales con respecto al tiempo real.

3.4. RESULTADOS DE PREDICCIÓN RANDOM FOREST

Con respecto al modelo Random Forest, se llevó a cabo la creación de dos modelos distintos destinados a la predicción de fallos. En el primer modelo, se optó por utilizar los datos predichos por el modelo de auto ARIMA, combinados con todos los datos de tiempo entre fallos (TBF) reales. La obtención de predicciones se llevó a cabo mediante la aplicación del algoritmo Random Forest. Por otro lado, se implementó un Random Forest ajustado el que considera exclusivamente los valores reales provenientes del histórico de fallos. En este caso, los datos de entrada incluyen tanto los tiempos fuera de servicio como los valores reales de TBF. Posteriormente, estos dos enfoques fueron sometidos a una comparación utilizando el conjunto de datos de prueba, que representa el último 20% de los datos de TBF reales en el historial de fallos del componente crítico. El objetivo primordial de esta evaluación es medir y analizar la precisión de ambos modelos en sus predicciones. Los resultados de este análisis se presentan detalladamente en la Tabla 3-3, donde se exhiben las métricas de rendimiento obtenidas para cada uno de los modelos, brindando una visión más completa y detallada de su desempeño.

Modelo	MAE	MSE	RMSE	MAPE	R^2
Auto ARIMA + RF	18,91 [hrs]	$3035,12[hrs]^2$	55,1[hrs]	3,47	0,97
RF Ajustado	12,21 [hrs]	$364,77[hrs]^2$	19,1[hrs]	2,61	0,99

Tabla 3-3. Métricas de rendimiento modelos Random Forest.
Fuente: Elaboración propia a través de Excel.

En la Tabla 3-3, se destacan los errores detectados en los modelos analizados. Al contrastar ambos modelos presentes, se percibe una diferencia mínima entre ellos. Para determinar el modelo óptimo, se debe optar por aquel que obtenga valores más bajos en las métricas de rendimiento, ya que esto indicaría un ajuste más preciso. En consecuencia, de este criterio podemos concluir que el modelo Random Forest ajustado se posiciona como la elección más adecuada. dado que presenta un error absoluto medio de 12.21[hrs]. Este valor sugiere que, en promedio, las predicciones del modelo se aproximan considerablemente a los valores reales.

Además, es crucial resaltar que el error porcentual absoluto medio es asombrosamente bajo, registrando tan solo un 2,61%. Por otro lado, el Coeficiente de Determinación (R^2) en este análisis destaca con un valor de 0,99. Este coeficiente nos suministra información crucial acerca del grado de ajuste del modelo a las observaciones reales. En este escenario, el valor señala un ajuste fuerte, indicando que las predicciones se adaptan de manera excepcional a las observaciones reales disponibles.

En comparación con los resultados obtenidos por los modelos mencionados anteriormente, el modelo Random Forest ajustado muestra una mejora significativa en las métricas de rendimiento propuestas. Esta mejora se evidencia de manera clara en la Figura 3-3, donde se representa el error absoluto asociado a cada predicción realizada por el modelo Random Forest ajustado. Es relevante destacar la elección de este modelo, respaldada por una evaluación minuciosa de su desempeño en relación con los pronósticos realizados. Esto lo posiciona como la opción más sólida y precisa entre todas las alternativas consideradas.

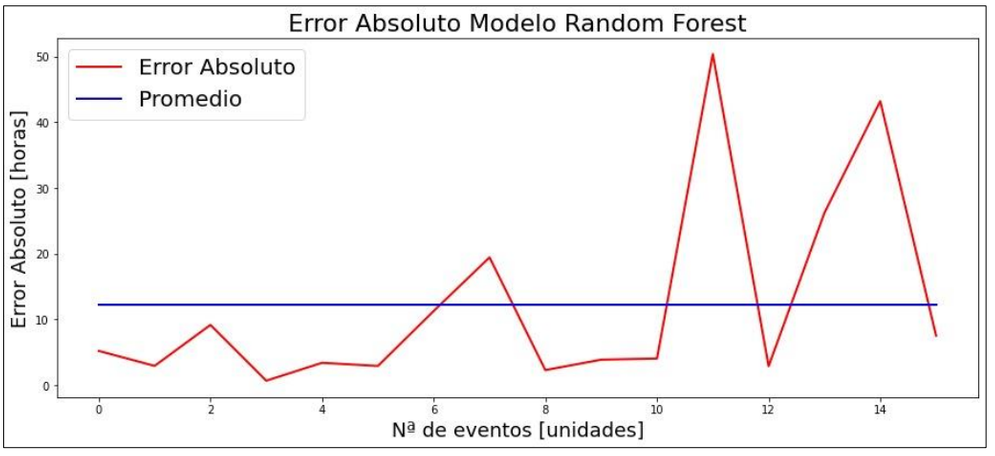

Figura 3-3. Error absoluto en la predicción de eventos con modelo Random Forest Ajustado.
Fuente: Elaboración propia en Python.

Al detenernos a analizar detalladamente la figura 3-3, se revela con evidencia palpable que el error absoluto máximo ha experimentado una disminución extraordinariamente significativa en comparación con los dos modelos previamente

desarrollados. Como consecuencia inmediata de esta mejora, el error máximo absoluto presente en el modelo Random Forest ajustado ahora se cifra en 50 [hrs]. Al proyectar esta discrepancia a lo largo de un período temporal más extenso, emerge la comprensión de que esta diferencia implica una mejora considerable en la precisión de la predicción del evento, abarcando un lapso de aproximadamente 2 días.

Por otro lado, es digno de destacar que, en el extremo opuesto del espectro, se registra un error mínimo de aproximación que apenas roza la marca de una hora. Este dato revela que, en ciertos puntos, los datos predichos exhiben un error absoluto insignificante en comparación con el tiempo real, siendo extraordinariamente bajo. Esta variación en los niveles de error aporta una perspectiva más completa sobre la eficacia y la precisión del modelo utilizados en la predicción de eventos.

3.5. MEJOR MODELO DE PREDICCIÓN DE FALLOS FUTUROS

A través de este trabajo, se lleva a cabo un exhaustivo análisis y comparación de diversos modelos de predicción y estimación, con el objetivo de lograr pronósticos más precisos en los tiempos entre fallos (TBF). Este análisis amplía la perspectiva sobre los resultados alcanzables, respaldado por métricas de rendimiento que facilitan la identificación del método de predicción con la menor cantidad de errores en sus estimaciones, como se evidencia previamente. A continuación, en la Figura 3-4. se sintetiza el orden de las metodologías empleadas en este caso de estudio, brindando una visión clara de cómo se llevaron a cabo las predicciones en este trabajo.

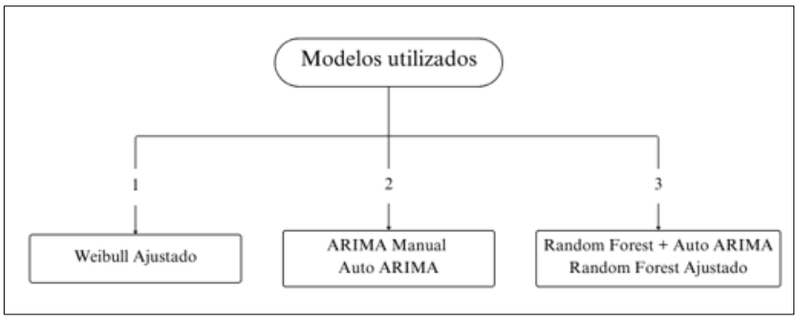

Figura 3-4. Diagrama de flujo de modelos utilizados.
Fuente: Elaboración propia a través de canva.

Siguiendo la referencia a la Figura 3-4, que exhibe los diversos modelos de predicción de fallos propuestos, se opta por seleccionar el mejor de cada uno. El propósito es llevar a cabo una comparación basada en las métricas de rendimiento, las cuales proporcionan información sobre el error de cada modelo. El objetivo primordial de esta

evaluación es medir y analizar la precisión de cada modelo en sus predicciones. Los resultados de este análisis se presentan detalladamente en la Tabla 3-4, donde se exhiben las métricas de rendimiento obtenidas para cada uno de los modelos, brindando una visión más completa y detallada de su desempeño.

Modelo	MAE	MSE	RMSE	MAPE	R^2
Weibull ajustado	755,89 [hrs]	133292,63[hrs]^2	1156,41[hrs]	2270,4	0,05
Auto ARIMA (1,1,0)	338,93 [hrs]	472680,01[hrs]^2	687,51[hrs]	425,09	0,62
RF Ajustado	12,21 [hrs]	364,77[hrs]^2	19,1[hrs]	7,61	0,99

Tabla 3-4. Métricas de rendimiento mejores modelos de cada metodología.
Fuente: Elaboración propia a través de Excel.

Al examinar y comparar las métricas de rendimiento derivadas de los diversos modelos vistos en este trabajo de título, podemos resumir que el modelo Random Forest se destaca al exhibir los valores más bajos en cada métrica propuesta para evaluar el error de cada modelo. Esta observación se fundamenta en la premisa de que a medida que los valores de las métricas disminuyen, se logra una adaptación más ajustada entre las predicciones y los valores reales, con la excepción del Coeficiente de Determinación (R^2), donde una proximidad a uno indica un ajuste más preciso de los datos predichos.

En última instancia, podemos concluir que el modelo Random Forest (RF), el cual es un algoritmo de aprendizaje supervisado dentro del machine learning, ha demostrado ser el más efectivo en la predicción de fallos en comparación con los demás modelos propuestos. Este análisis resalta la capacidad sobresaliente de Random Forest en proporcionar predicciones precisas y respalda la conclusión de que este enfoque de machine learning ha superado a sus contrapartes en términos de rendimiento predictivo.

CONCLUSIONES

En conclusión, el objetivo de este estudio fue desarrollar un enfoque metodológico destinado a potenciar de manera significativa la capacidad de predecir eventos de fallos en un componente crítico de camiones mineros. A lo largo de este estudio, nos embarcamos en la implementación de los modelos Weibull, ARIMA (Autorregresivo Integrado Media Móvil) y exploramos las posibilidades ofrecidas por las técnicas de Machine Learning.

El modelo de distribución de Weibull es ampliamente empleado en confiabilidad, especialmente para el análisis del ciclo de vida de activos. Sin embargo, es importante destacar que, mediante este modelo, se observa un error absoluto medio significativo de 755.89 [hrs], colocándolo como el que genera mayores errores en los pronósticos.

En retrospectiva, la aplicación del modelo auto ARIMA mostró resultados positivos al desempeñarse de manera eficiente en comparación con el modelo Weibull. El error absoluto medio disminuyó un 55% aproximadamente quedando en 338.93[hrs], en comparación con el valor obtenido con Weibull ajustado. A pesar de estas mejoras, persisten errores significativos en las predicciones, lo que lo descarta como la mejor opción.

Para finalizar este trabajo de título, se propone la inclusión de Random Forest en nuestro enfoque la cual fue la elección decisiva. Las predicciones excepcionales que logró destacan su superioridad en la tarea de anticipar eventos de fallos en comparación con las otras metodologías propuestas. Debido su bajo valor de error absoluto medio el cual tiene un valor de 12.21[hrs] por lo cual el error promedio de las predicciones no supera un día por lo cual presenta excepcionales predicciones.

La implementación de nuestro enfoque metodológico en el contexto de los camiones mineros. Al adoptar nuestro modelo que combina ARIMA y Random Forest, las empresas pueden anticipar de manera más precisa y proactiva eventos de fallos en componentes críticos. Esto se traduce en una serie de beneficios tangibles, como la reducción de costos asociados con reparaciones no planificadas, el aumento de la disponibilidad y la mejora general en la eficiencia de la gestión de la flota.

RECOMENDACIONES

Las estrategias de pronóstico ofrecen una visión distinta al reconocer eficazmente las particularidades de un conjunto de datos con mayor exactitud. Por tanto, al poner en práctica las metodologías previamente desarrolladas en este trabajo, se proponen los siguientes aspectos para tener en cuenta.

El Modelo Autorregresivo de Media Móvil emerge como una herramienta que proporciona estimaciones prometedoras. Es considerado uno de los modelos más prevalentes en diversas áreas industriales en tiempos recientes. Es crucial realizar un estudio exhaustivo de este modelo para su implementación exitosa. Gracias al lenguaje de programación, es factible obtener resultados de manera más rápida y automática. Se requiere la obtención y validación de sus parámetros, seguido de la ejecución y, finalmente, la realización de predicciones.

El Random Forest se destaca como un modelo de Machine Learning, especialmente cuando se dispone de una gran cantidad de datos. Es recomendable realizar una partición efectiva de los datos durante las etapas de entrenamiento y validación, ya que este aspecto juega un papel crucial en el rendimiento del modelo. Además, la mejora del modelo es siempre posible según la partición de los datos. Aprovechando la naturaleza de los árboles de decisión y aumentando su número, se logra potenciar la precisión del modelo

Los métodos previamente mencionados se basan en datos que, en este contexto, son registros realizados por personas. Es importante tener en cuenta que estos registros pueden contener errores al ser ingresados en una base de datos. Por lo tanto, se aconseja proporcionar capacitación al personal encargado para que ingresen los datos con el máximo detalle posible. Esto garantizará que la serie de tiempo introducida en el modelo se ajuste lo más fielmente posible al comportamiento real de los datos, mejorando así los resultados al emplear modelos de predicción.

BIBLIOGRAFÍA

Baptista, M., Sankararaman, S., De Medeiros, I. P., Nascimento, C. L., Prendinger, H., & Henriques, E. (2018). Forecasting fault events for predictive maintenance using data-driven techniques and ARMA modeling. *Computers & Industrial Engineering*, *115*, 41-53. https://doi.org/10.1016/j.cie.2017.10.033

Cardellino, F. (2021, 22 marzo). *Tutorial para un clasificador basado en bosques aleatorios: Cómo utilizar algoritmos basados en árboles para el aprendizaje automático*. freeCodeCamp.org. https://www.freecodecamp.org/espanol/news/random-forest-classifier-tutorial-how-to-use-tree-based-algorithms-for-machine-learning/

Joaquín Amat Rodrigo, (2020, octubre). Random *Forest Python*. https://cienciadedatos.net/documentos/py08_random_forest_python

Johanna Orellana Alvear - johanna.orellana@ucuenca.edu.ec. (s. f.). *Arboles de decision y random forest*. https://bookdown.org/content/2031/

Madrigal, E. (2022, 3 noviembre). Conoce las métricas de precisión más comunes para modelos de regresión. *Grow Up*. https://www.growupcr.com/post/metricas-precision

NTIC Máster. (2021, 8 junio). *¿Qué es el machine Learning? | Aprendizaje automático | UCM*. Máster Data Science. https://www.masterdatascienceucm.com/que-es-machine-learning/

Randall Romero Aguilar. (2021). *3.Modelos AutoRegresivos de media móvil (ARMA) — Apuntes de macroeconometría*. https://randall-romero.github.io/econometria/teoria/03-arma/chapter03.html

ANEXO A

Programación Python de métricas de evaluación

```
# Métrica de Evaluación
from sklearn import metrics
#evaluar modelo
def evaluacion_metrica(y_true, y_pred):

    def mean_absolute_percentage_error(y_true, y_pred):
        y_true, y_pred = np.array(y_true), np.array(y_pred)
        return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
    print('resultados de evaluacion:-')
    print(f'MSE is : {metrics.mean_squared_error(y_true, y_pred)}')
    print(f'MAE is : {metrics.mean_absolute_error(y_true, y_pred)}')
    print(f'RMSE is : {np.sqrt(metrics.mean_squared_error(y_true,
y_pred))}')
    print(f'MAPE is : {mean_absolute_percentage_error(y_true,
y_pred)}')
    print(f'R2 is : {metrics.r2_score(y_true, y_pred)}',end='\n\n')
```

ANEXO B

Programación Python de modelo Weibull ajustado

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import gamma
from scipy.stats import exponweib
from scipy.optimize import fmin

#Función weibull automatizada
def fitweibull(n):
    def optfun(theta):
        return -np.sum(np.log(exponweib.pdf(n, 1, theta[0],
scale=theta[1], loc=0)))
    logx = np.log(n)
    # Calculo del parametro Beta
    beta = 1.2 / np.std(logx)
    # Calculo del parametro Alpha
    alpha = np.exp(np.mean(logx) + (0.572 / beta))
    return fmin(optfun, [beta, alpha], xtol=0.01, ftol=0.01, disp=0)

#Importar Data
ruta="C:\\Users\\sergi\\Desktop\\usm\\trabajo de titulcion
2023\\TT1.xlsx"
df=pd.read_excel(ruta ,sheet_name="calculos",index_col=0)

#Definir variable a utilizar de los datos
tbf_real = df['TBF'].tolist()


# Generación de una lista vacia
mtbf_esperado = [0,0,0,0,0,0,0,0]
def mtbf(beta, alpha):
    mt_w = alpha * gamma(1 + (1 / beta))
    return mt_w
```

```
print("beta",fitweibull(tbf_real[0:81])[0])
print("alfa",fitweibull(tbf_real[0:81])[1])
print(mtbf(fitweibull(tbf_real[0:81])[0],fitweibull(tbf_real[0:81])[1])
)

# Ciclo cerrado para agregar los resultados a lista vacia
j=9
while j<82:

mtbf_esperado.append(mtbf(fitweibull(tbf_real[0:j])[0],fitweibull(tbf_r
eal[0:j])[1]))
    j+=1
print(mtbf_esperado)
#se agrega columna
df['Estimación con Weibull'] = np.array(mtbf_esperado)
```

ANEXO C

Programación Python de modelo ARIMA y Auto ARIMA

```
import warnings
warnings.filterwarnings("ignore")

# Visualización de datos
import matplotlib.pyplot as plt

import numpy as np
import pandas as pd
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf

# PREDIC ARIMA
ruta="C:\\Users\\sergi\\Desktop\\usm\\trabajo de titulcion
2023\\TT1.xlsx"
df=pd.read_excel(ruta ,sheet_name="calculos", usecols=[1])
# prueba de estacionalidad pvalue<0.05 datos estacionario
print("\n PRUEBA DE ESTACIONALIDAD")
from statsmodels.tsa.stattools import adfuller
result = adfuller(df)

print('ADF Statistic: %f' % result[0])
print("p-value: ", result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("\n Rechazar Ho - La serie temporal es estacionaria")
else:
    print ("\n No se pudo rechazar Ho: la serie temporal no es
estacionaria")

# encontrar AR valor p respecto a los resagos del grafico
print("\n encontrar AR valor p ")
plt.rcParams.update({"figure.figsize":(9,3),"figure.dpi":(120)})
fig, axes=plt.subplots(1,2,sharex=False)
axes[0].plot(df,color="orangered");axes[0].set_title("Equipo CA-73
modelo 930E-4")
axes[1].set(ylim=(-1,4))
```

```
plot_pacf(df.dropna(), ax=axes[1], color="orangered")
plt.show()

# encontrar MA valor q
print("\n encontrar MA valor q")
fig, axes=plt.subplots(1,2,sharex=False)
axes[0].plot(df,color="fuchsia");axes[0].set_title("Equipo CA-73 modelo
930E-4")
axes[1].set(ylim=(-2,4))
plot_acf(df.dropna(), ax=axes[1], color="fuchsia")
plt.show()

#contruir modelo arima
print("\n MODELO ARIMA")
import statsmodels.api as sm
# modelo arima (p,i,q)
# 1 0 1 modelo
modelo_1= sm.tsa.SARIMAX(df, order=(1,0,0))
modelo_fit1=modelo_1.fit(disp=False) #entrenar el nuevo modelo
print(modelo_fit1.summary()) # informacion del modelo arima

#PREDICCIONES
pred_y=modelo_fit1.predict(1,81)

print("\n METRICAS DE EVALUCION ")
evaluacion_metrica(df,pred_y)#modelo ARIMA Manual


# MODELO AUTO ARIMA
import pmdarima as pm
from statsmodels.tsa.arima_model import ARIMA
auto_model=pm.auto_arima(df,star_p=1,start_q=1,
                         test="adf" ,#MAS OPTIMO
                         max_p=3,max_q=3,#maximo p y q
                         m=1,
                         d=None ,#diferenciacion
                         seasonal=False,
                         star_P=0,
                         D=0,
                         trace=True,
                         erro_action="ignore",
                         suppress_warnings=True,
                         stepwise=True)
print(auto_model.summary())

#MODELO AUTO ARIMA
modelo_2= sm.tsa.SARIMAX(df, order=(1,1,0))
modelo_fit2=modelo_2.fit(disp=False) #entrenar el nuevo modelo
print(modelo_fit2.summary()) # informacion del modelo arima

#PREDICCIONES AUTO ARIMA
pred_y2=modelo_fit2.predict(1,81)

# Visualiza las predicciones y los datos reales
plt.plot(df["TBF"], label='TBF Real')
plt.plot(pred_y2, label='ARIMA (1,1,0)',color="m")
plt.xlabel("Nª de eventos [unidades]")
plt.ylabel("Tiempo de Funcionamiento [horas]")
plt.title("Modelo auto ARIMA",fontsize=20)
plt.legend()
plt.show()

print("\n METRICAS DE EVALUCION AUTO ARIMA ")
```

```
from sklearn.metrics import mean_squared_error
evaluacion_metrica(df,pred_y2)#modelo auto-ARIMA

print("\n METRICAS DE EVALUCION ARIMA MANUAL")
evaluacion_metrica(df,pred_y)#modelo ARIMA Manual
```

ANEXO D

Programación Python de modelo ARIMA + Random Forest

```
#RANDOM FOREST
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error

#Importar la data
ruta="C:\\Users\\sergi\\Desktop\\usm\\trabajo de titulcion
2023\\TT1.xlsx"
df1=pd.read_excel(ruta ,sheet_name="calculo")
df= df1["TBF"]

print("\n PIMEROS 5 DATOS")
print(df.head())
print("\n INFORMAICION DEL DF")
print(df.info())

#MODELO AUTO ARIMA
print("\n MODELO AUTO ARIMA")
import statsmodels.api as sm
modelo_1= sm.tsa.SARIMAX(df, order=(1,1,0))
modelo_fit1=modelo_1.fit(disp=False)
pred_y=modelo_fit1.predict(1,81)

df1["ARIMA"]=pred_y

#definir variables
df1=df1.dropna()
y=df1["TBF"] #dependiente

x=df1[["ARIMA","TBF"]]#independiente

#DIVISION DE DATOS
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,test_size =
0.2, random_state=0
                                                    ,shuffle=False)

#DEFINIR ALGORITMO
RFReg = RandomForestRegressor(n_estimators=50,random_state=0)

#ENTRENAR MODELO
RFReg.fit(x_train, y_train)

#PREDICCIONES
y_pred=RFReg.predict(x_test)


plt.plot(dta,)
```

```
plt.title("RF + AUTO ARIMA")
plt.ylabel("Tiempo de Funcionamiento [horas]")
plt.xlabel("Nª de eventos [unidades]")
plt.legend(["Conjunto de Prueba", "Datos Predichos"])
plt.figure()
plt.show()

#METRICAS DE PRECISION
print("\n METRICAS DE EVALUCION ")
evaluacion_metrica(y_test, y_pred)#modelo RF + auto-ARIMA
```

ANEXO E

Programación Python de modelo Random Forest ajustado

```
#RANDOM FOREST
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error

#Importar la data
ruta="C:\\Users\\sergi\\Desktop\\usm\\trabajo de titulcion
2023\\TT1.xlsx"
df=pd.read_excel(ruta ,sheet_name="calculo")

df["anual"]=df["Inicio"].apply(lambda x: x.year)
df["mes"]=df["Inicio"].apply(lambda x: x.month)
df=df.dropna()
#definir variables
y=df["TBF"] #dependiente
x=df[["TBF","TTR","mes","anual","L1","L2","L3","L4"]]#independiente

#DIVISION DE DATOS
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,test_size =
0.2,random_state=12345,shuffle=False)

#DEFINIR ALGORITMO
RFReg = RandomForestRegressor(n_estimators=50, random_state=0)

#ENTRENAR MODELO
RFReg.fit(x_train, y_train)

#PREDICCIONES
y_pred=RFReg.predict(x_test)

plt.plot(dta,)
plt.title("Random Forest Ajustado")
plt.ylabel("Tiempo de Funcionamiento [horas]")
plt.xlabel("Nª de eventos [unidades]")
plt.legend(["Conjunto de Prueba", "Datos Predichos"])
plt.figure()
plt.show()
print(dta.head())

print("\n METRICAS DE EVALUCION ")
evaluacion_metrica(y_test, y_pred)
```