UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA MECÁNICA

TESIS:

"MODELOS PREDICTIVOS BASADOS EN MACHINE LEARNING PARA OPTIMIZAR EL DIAGNÓSTICO DEL ÍNDICE DE SALUD EN LOS INTERRUPTORES DE POTENCIA DE UNA EMPRESA DE TRANSMISIÓN ELÉCTRICA"

PARA OBTENER EL GRADO ACADÉMICO DE MAESTRO EN INGENIERÍA CON MENCIÓN EN GERENCIA E INGENIERÍA DE MANTENIMIENTO

ELABORADO POR:
ANGEL HUAMÁN SARZO

ASESOR:
DR. JORGE ENRIQUE ORTIZ PORRAS

LIMA – PERÚ

2023

DEDICATORIA

Dedico esta tesis a mi esposa e hijos: Kely, Leonardo y Sebastián por su amor, cariño y comprensión durante todo el tiempo que me indispuso estar con ellos, desde el inicio de la maestría hasta culminar con esta meta.

AGRADECIMIENTOS

Agradezco a mi asesor el Dr. Jorge Ortiz, por su amable disponibilidad y apoyo brindado. También al Dr. Gilberto Becerra, al Dr. Juan Vargas Machuca, y al MSc. Dheybi Cervan por sus oportunas observaciones y sugerencias que ayudaron a mejorar y enriquecer la presente investigación.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

RESUMEN

A través de esta investigación se pretende sentar las bases del uso de modelos predictivos con inteligencia artificial basados en machine learning aplicado al diagnóstico técnico del índice de salud en los interruptores de potencia de una empresa de transmisión de energía eléctrica, con el fin de obtener un modelo predictivo con inteligencia artificial que sea efectivo y eficiente, el cual será entrenado con todo el conocimiento experto que se tiene de estudios anteriores para obtener un modelo predictivo óptimo basado en machine learning, que a diferencia del método tradicional usado en la empresa (lógica fuzzy) donde se le debe indicar a la máquina cuáles son las reglas de parametrización y tener un criterio de inferencia lógica para cada regla, el modelo predictivo utilizó todos los parámetros técnicos (variables predictoras) que intervinieron en el diagnóstico del índice de salud (variable objetivo) y diseño su propio modelo matemático en base a un análisis computacional de correlaciones y componentes principales. La aplicación del machine learning al diagnóstico del índice de salud en los interruptores de potencia optimizó todo el proceso que se sigue actualmente consiguiendo diagnósticos más certeros y con mejor oportunidad.

El algoritmo propuesto obtuvo resultados en el diagnóstico del estado de salud de 99.27% respecto a su efectividad y de una disminución de horas hombre del 76.19% respecto a la metodología antigua.

PALABRAS CLAVE: Interruptores de potencia, Índice de salud, Machine learning, Humedad, Pureza, Dióxido de Azufre, Resistencia de contacto, Edad.

ABSTRACT

The purpose of this research is to lay the foundations for the use of predictive models with artificial intelligence based on machine learning applied to the technical diagnosis of the health index in the power circuit breakers of an electric power transmission company, in order to obtain a predictive model with artificial intelligence that is effective and efficient, which will be trained with all the expert knowledge that we have from previous studies to obtain an optimal predictive model based on machine learning, Unlike the traditional method used in the company (fuzzy logic) where the machine must be told which are the parameterization rules and have a logical inference criterion for each rule, the predictive model used all the technical parameters (predictor variables) involved in the diagnosis of the health index (target variable) and designed its own mathematical model based on a computational analysis of correlations and principal components. The application of machine learning to the diagnosis of the health index in circuit breakers optimized the entire process that is currently followed, achieving more accurate diagnoses and with better timing.

With the design of the predictive model, an effectiveness of 99.27% was achieved with a decrease in man hours of 76.19% with respect to the old methodology.

KEY WORDS: Power breakers, Health index, Machine learning, Moisture, Purity, SO2, Contact resistance, age.

INTRODUCCIÓN

La presente investigación pretende sentar las bases de la inteligencia artificial aplicado al diagnóstico técnico de los interruptores de potencia en una empresa de transmisión de energía eléctrica en Perú que por motivos de confidencialidad en adelante denominaremos ETEP. Utilizaremos para este propósito algoritmos predictivos supervisados de machine learning a fin de conseguir un aprendizaje con inteligencia artificial de todo el conocimiento experto que se tiene de estudios de índice de salud anteriores, los cuales fueron realizados con metodología fuzzy, metodología que necesita una parametrización triangular, trapezoidal o gaussiana por cada variable a usar y sendas inferencias lógicas por cada combinación de variables. El modelo predictivo optimiza este proceso aprendiendo con los datos ya existentes y creando sus propias reglas y criterios en base a un análisis computacional de componentes principales y de correlaciones entre variables que intervinieren en el diagnóstico de la salud del interruptor.

Para seguir toda la estructura de un proceso de investigación se desarrollaron 04 capítulos los cuales resumimos a continuación:

Primer capítulo, en esta sección analizaremos los antecedente bibliográficos encontrados y describiremos la realidad problemática que conllevo a la realización de la presente investigación, formularemos el problema general y específicos para luego dar a conocer la justificación e importancias de la presente investigación, plantearemos el objetivo general y específicos y las correspondientes hipótesis que demostraremos en el desarrollo, para ello estableceremos la variable independiente y la variable dependiente así como los indicadores con los cuales mediremos estas variables.

Segundo capítulo, en esta sección nos centraremos en dar el marco teórico necesario para entender sobre los interruptores de potencia, como estos se clasifican, que parámetros se les controla, cuáles son sus principales modos de falla, como se debe entender el índice de salud para un activo, que es machine learning, que son modelos predictivos, cuáles son los principales algoritmos de machine learning y además el marco conceptual de algunos términos que tendrán un significado particular en esta investigación.

Tercer capítulo, en esta sección realizaremos un análisis de datos univariante de cada una de las variables predictoras (histogramas y boxplot estadísticos) y un análisis bivariante de cómo cambian estas variables con respecto a la variable objetivo (Índice de Salud), también realizaremos un análisis de matriz de correlaciones y un análisis para determinar las variables principales para finalmente modelar y entrenar usando algoritmos predictivos de machine learning

Cuarto capítulo, en esta última sección realizaremos la evaluación de los modelos predictivos en función a sus métricas, pare ello usaremos el F1-Score en 10 sorteos de datos y seleccionaremos el modelo predictivo óptimo con mejor desempeño y menor variabilidad. Concluiremos este capítulo contrastando las hipótesis planteadas al inicio de la presente investigación.

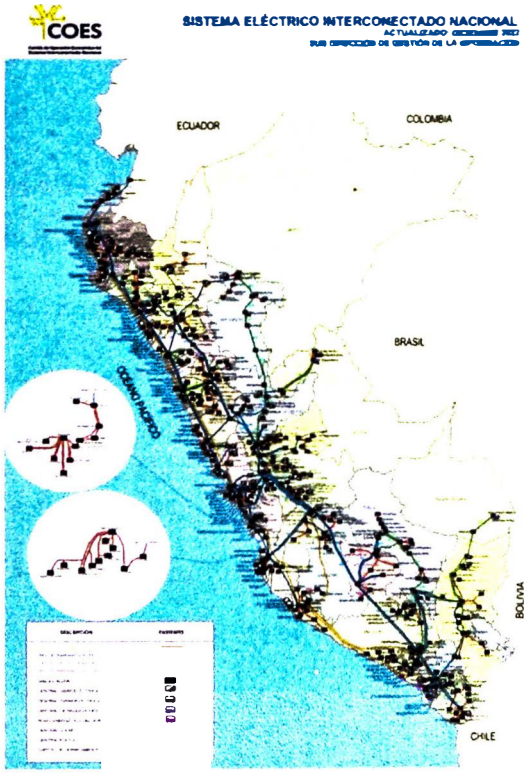
La tesis finaliza exponiendo las conclusiones principales llegadas durante el desarrollo de esta investigación y las recomendaciones más oportunas para investigaciones futuras que quieran ahondar en el uso del machine learning como herramienta para predecir la salud de interruptores de potencia u otros activos importantes presentes en subestaciones eléctricas.

CAPÍTULO I. DESCRIPCIÓN Y ASPECTOS METODOLÓGICOS

1.1. ÁMBITO DEL DESARROLLO DE LA INVESTIGACIÓN

La presente investigación se desarrolló con los interruptores de potencia que se encuentran instalados a nivel nacional en las subestaciones eléctricas de alta tensión que pertenecen a la empresa de transmisión de energía eléctrica peruana ETEP. En la figura 1 se muestra el mapa de líneas y subestaciones del SEIN – PERÚ

Figura 1*.Mapa de Líneas y Subestaciones del SEIN-PERÚ*



Nota: Tomada de página web del COES. Disponible en: https://www.coes.org.pe/Portal/Operacion/CaractSEIN/MapaSEIN

1.2. ANTECEDENTES BIBLIOGRÁFICOS

El estudio de modelos predictivos basados en inteligencia artificial como el machine learning u otras herramientas más avanzadas como el Deep learning, están siendo hoy en día muy estudiado y aplicado en las diferentes disciplinas de la ciencia. Para el desarrollo de la presente tesis se ha investigado los estudios más recientes en revistas indexadas sobre aplicaciones en industrias energéticas, minera y de petróleo en tópicos como el diagnóstico y predicción de fallas o la predicción de la vida útil de equipos. Los principales desarrollos encontrados fueron de China, Estados unidos y Europa y entre estos autores tenemos a:

Angelopoulos et al. (2019)[1] realizaron un estudio sobre las industrias modernas o industrias 4.0 y como abordan las fallas en plena era de la transformación digital, la utilización de herramientas de inteligencia artificial y el internet de las cosas. En su investigación los autores comenzaron realizando un mapeo estadístico de diferentes bibliografías que abordan el tema del uso de machine learning para aplicaciones industriales en la detección de fallas y como implementaron el proceso de obtención de datos online y offline , el tratamiento de la calidad de estos datos y como en función a estos datos han tratado la clasificación de los modos de fallas, procediendo luego a describir de manera detallada cada una de las técnicas del machine learning más utilizadas para finalmente mostrarnos los puntos débiles de las

[1] Angelopoulos, A., Michailidis, E. T., Nomikos, N., Trakadas, P., Hatziefremidis, A., Voliotis, S., & Zahariadis, T. (2019). Tackling Faults in the Industry 4.0 Era—A Survey of Machine-Learning Solutions and Key Aspects. http://dx.doi.org/10.3390/s20010109

soluciones actuales y los posibles problemas con la ciberseguridad. Concluyendo que la mayoría de las investigaciones al respecto han utilizado modelos supervisados de machine learning para detectar modos de fallas ya conocidos, modelos no supervisados de machine learning para conseguir una clasificación de modos de falla desconocidos y el Deep learning (aprendizaje profundo) en datos no estructurados (imágenes, sonidos, vibraciones, etc.)

Ellefsen et al. (2019)[2] investigaron sobre la predicción de la vida útil en motores turboventiladores, utilizando un modelo de arquitectura profunda semisupervisado con algoritmos genéticos para seleccionar los hiperparametros óptimos que caracterizan las fallas en estos motores. Los autores sugieren que las técnicas de aprendizaje profundo resultan efectivas para diagnosticar y predecir el estado de un equipo, sin embargo, mencionan que estos modelos tienen el problema de necesitar una cantidad importante de datos para el entrenamiento y validación, siendo aquello un factor limitante en su aplicación. En consecuencia, los autores plantean el uso previo de algoritmos no supervisados para caracterizar los datos no etiquetados, sirviendo esto como un preentrenamiento inicial para el modelo semisupervisado. Finalmente, los autores comparan sus modelos supervisados y sus modelos mixtos (semi supervisados + supervisados) con el C-MAPSS (Datos Públicos de la NASA para Simulación de Sistemas de Aéreo-propulsores comerciales), obteniendo resultados superiores para los modelos mixtos.

[2] Ellefsen, L., Bjørlykhaug, E., Æsøy, V., Ushakov, S., & Zhang, H. (2019). Remaining useful life predictions for turbofan engine degradation using semi-supervised Deep architecture. https://doi.org/10.1016/j.ress.2018.11.027.

Gao et al. (2019)[3] realizaron un estudio sobre la predicción de fallas mecánicas en interruptores de media tensión, cuyo medio de extinción es el vacío, usando un sensor de vibración e implementando un modelo predictivo de aprendiza profundo denominado IELM (máquina de aprendiza extremo integrado). Modelo que usa un conjunto de redes neuronales para poder desagregar la señal de vibración en subseñales usando un esquema de descomposición de ruido adaptativo. Luego adicionándole un filtro pasa banda y mediante la transformada de Hilbert consiguen una matriz tiempo frecuencia de estas señales y las clasifican usando vectores característicos de falla, teniendo como referencia la entropía tiempo- frecuencia y la entropía singular de un interruptor sin problemas mecánicos. El 76% de las muestras se utilizaron para entrenar el modelo clasificador, y el 24% de las muestras se utilizaron para probar la precisión de la clasificación. Con este modelo de aprendiza extremo integrado, los autores consiguieron una efectividad en la exactitud de hasta el 100% resultando este método superior a los modelos tradicionales de aprendiza máquina.

Li et al. (2020) [4] realizaron un estudio sobre la aplicación de una máquina de soporte vectorial (SVM) que combina el algoritmo de optimización de enjambre de partículas (PSO) para diagnosticar y predecir fallas mecánicas en interruptores de media tensión modelo ZN12 usando un acelerómetro instalado en un interruptor de ensayo. Para implementar este modelo

---

[3] Gao, W., Wai, S., Qiao, R., & Guo, M. (2019). Mechanical Faults Diagnosis of High-Voltage Circuit Breaker via Hybrid Features and Integrated Extreme Learning Machine. https://doi: 10.1109/ACCESS.2019.2915252.

[4] Li, X., Wu, S., Li, X. et al. (2020). Particle Swarm Optimization-Support Vector Machine Model for Machinery Fault Diagnoses in High-Voltage Circuit Breakers. https://doi.org/10.1186/s10033-019-0428-5

predictivo los autores usaron un algoritmo SVM de cuatro capas en cascada (SVM1, SVM2, SVM3, SVM4) donde cada máquina de soporte vectorial fue entrenada en reconocer un tipo específico de falla mecánica, teniendo como input inicial, la descomposición vectorial de la onda principal de vibración, luego en cada etapa se iba recibiendo como input, la salida del análisis anterior. El algoritmo PSO se aplica para buscar el parámetro del núcleo óptimo "δ" y el parámetro de penalización "c" para la SVM. El tiempo de entrenamiento de SVM-PSO es más largo que el de SVM normal. Sin embargo, la precisión total mejoró drásticamente del 80% al 98% . En conclusión, el modelo de cuatro capas de SVM-PSO diagnosticaba con una alta tasa de efectividad, si el interruptor tenía algún problema de desgaste en su resorte, desplazamiento de su eje, atasco del engranaje o estaba en una condición normal.

Yi et al. (2020) [5] Ensayaron un modelo predictivo para el diagnóstico de fallas de aislamiento en subestaciones con celdas aisladas en gas (GIS) de 110kV, colocando para este fin un lector externo de campo eléctrico transitorio (STEF) captando los campos eléctricos generados en las subestaciones GIS durante su operación normal y en operación anormal, recreando situaciones anormales como descargas parciales internas, presencia de partículas metálicas suspendidas dentro de los compartimientos de gas y hasta problemas de flashover en los bujes de la subestaciones GIS. Estos campos magnéticos transitorios lo analizaron mediante el método de transformación de paquetes de ondas (WPT), obteniendo una matriz de caracterización

[5] Yi, T., Xie, Y., H. Zhang & X. Kong. (2020). "Insulation Fault Diagnosis of Disconnecting Switches Based on Wavelet Packet Transform and PCA-IPSO-SVM of Electric Fields." https://doi: 10.1109/ACCESS.2020.3026932.

multidimensional. Finalmente, los autores redujeron la matriz con el método de Análisis de parámetro principales (PCA) y esta matriz reducida lo modelaron en una máquina de soporte vectorial (SVM) y un algoritmo mejorado de optimización de enjambre de partículas (IPSO) consiguiendo un modelo predictivo compuesto que lo denominaron SVM-PCA-IPSO, modelo entrenado para interpretar los STEF y clasificar los modos de falla previamente aprendidos. En comparación con otros métodos la combinación de SVM-PCA-IPSO obtuvo la mayor precisión de clasificación para el conjunto de entrenamiento y el conjunto de prueba: 99,46% y 97,58%, respectivamente.

También se han encontrado investigaciones con aplicaciones muy interesantes del machine learning en otras ramas de la ciencia como el análisis médico, análisis físico y análisis quimico. Se traen en cuenta estas investigaciones como antecedentes bibliográficos dado que el planteamiento y estructura lógica están muy relacionados al objetivo de la presente investigación: "Encontrar un modelo predictivo optimo".

Uddin et al. (2022)[6] aplicaron algoritmos de machine learning para predecir el índice de calidad del agua costera (ICA) en el puerto de Cork- Irlanda, en su estudio utilizaron once variables de calidad del agua: temperatura (TEMP), nitrógeno orgánico total (TON), amoníaco (AMN), oxígeno disuelto (DOX), nitrógeno amoniacal (AMN), pH, salinidad (SAL) ,molibdato, fósforo reactivo (MRP), demanda biológica de oxígeno (DBO), transparencia (TRAN) y clorofila a (CHL) y compararon ocho algoritmos de aprendizaje supervisado

[6] Uddin, M. G., Nash, S., Mahammad Diganta, M. T., Rahman, A., & Olbert, A. I. (2022). Robust machine learning algorithms for predicting coastal water quality index. https://doi.org/10.1016/j.jenvman.2022.115923

para predecir esta variable objetivo adimensional tipo continua (ICA). Se incluyeron en este análisis: Random Forest (RF), Decisión Tree (DT), K-Nearest Neighbors(KNN), Extreme Gradient Boosting (XGB), Extra Tree (ExT), Support Vector Machine (SVM), Linear Regression (LR) y Gaussian Naive Bayes(GNB). Optimizaron los hiperparametros de los modelos con el algoritmo de búsqueda en cuadriculas "GridSearch". Dividieron sus datos en 70% para entrenamiento y 30% para las pruebas y los modelos se validaron mediante el método de CV (validación cruzada) por diez veces, para evaluar el desempeño de los modelos se utilizaron las métricas: error cuadrático medio (MSE), error cuadrático medio (RMSE), error medio absoluto (MAE) y coeficiente de determinación (R2) y el porcentaje de índice de error relativo (PREI). En los resultados de CV, el XGB mostró el mejor rendimiento, para el entrenamiento (RMSE = 3,3, MSE = 10,91, MAE = 1,67 y R2 = 1,0) y para la prueba (RMSE = 0,0, MSE = 0,0, MAE = 0,02 y R2 = 1,0). Los errores de predicción más bajos se encontraron para DT (PREI = 0), ExT (PREI = 0) y XGB (PERI = + 0,1 a - 0,1). Estos algoritmos funcionaron mejor a la hora de predecir los ICA en cada uno de los sitios de seguimiento del puerto de Cork.

Omar, A et al., (2023)[7] aplicaron un método basado en datos para predecir la falla de las tuberías principales de agua en la ciudad de Kitchener. El primer conjunto de datos usado considera todas las redes de agua instaladas desde junio de 1889 hasta enero de 2021. El segundo conjunto de datos es el historial de interrupciones desde enero de 1985 hasta enero de 2021. El

[7] Omar, A., Delnaz, A., & Nik-Bakht, M. (2023). Comparative analysis of machine learning techniques for predicting water main failures in the City of Kitchener. https://doi.org/10.1016/j.iintel.2023.100044

tercer conjunto de datos incluye información relacionada con segmentos de carreteras como: el tráfico diario promedio anual (AADT) y la subcategoría de carreteras. Se desarrollaron seis modelos de predicción de aprendizaje automático : Naive Bayes (NB), Decisión Tree (DT), k -vecinos más cercanos (k-NN), regresión logística (LR), red neuronal artificial (ANN) y Random Forest (RF) bajo dos escenarios: modelos globales, que consideran los tres tipos de materiales dominantes en la red; y el modelo homogéneo, que considera únicamente tuberías de hierro fundido. En este estudio, el proceso de ajuste de hiperparametros se llevó a cabo de forma iterativa mediante prueba y error (para modelos más simples) y el método de búsqueda aleatoria "RandomSearch" (para modelos ANN).Se utilizó una validación cruzada de 10 veces con muestreo aleatorio estratificado para dividir el conjunto de datos preparado en conjunto de entrenamiento y prueba. Todos los modelos muestran un buen rendimiento de predicción, así como capacidades de generalización; sin embargo, el modelo RF también es el mejor en términos de puntuación F1 (86. 0%) y AUC (86,2%). Por otro lado, NB identifica exitosamente el mayor número de fallas (es decir, mejor recuperación: 83,6%); pero a expensas de las tuberías mal clasificadas (es decir, menor precisión de NB: 85,2%, frente al 90,1% de RF). Además, el modelo k -NNs tiene el menor rendimiento. En general, todos los modelos homogéneos de CI podían identificar más instancias de falla que los modelos globales; pero también clasificaban erróneamente una mayor cantidad de tuberías además de ser menos estables (es decir, tener mayores desviaciones estándar (SD) para las métricas relacionadas con la precisión). La superioridad de los modelos desarrollados radicaba en su capacidad para predecir fallas en las tuberías con el menor número de falsas alarmas. El

modelo de bosque aleatorio superó a todos los demás modelos y fue el algoritmo más robusto. Los resultados mostraron que más del 72% de las interrupciones se podrían haber evitado monitoreando y actualizando solo el 8% de la red.

Ahmad G.N. et al., (2022)[8]. En este estudio los autores entrenaron diferentes tipos de algoritmos clasificadores de aprendizaje automático, incluida la regresión logística (LR), K-vecinos más cercanos (K-NN), máquina de vectores de soporte (SVM) y Clasificador de aumento de gradiente (GBC) con y sin "GridSearchCV", para seleccione el mejor modelo predictivo para la detección precisa de enfermedades cardíacas en una etapa temprana. El conjunto de datos de predicción de enfermedades cardíacas tiene 12 columnas, con 11 componentes independientes y una variable objetivo dependiente. La variable objetivo se separa en dos grupos: los que tienen enfermedades cardíacas y los que no. Cada algoritmo se analiza utilizando diferentes hiperparámetros: Los resultados de la búsqueda aleatoria "RandomSearch", la búsqueda en cuadrícula "GridSearchCV" y otros procedimientos de optimización de hiperparámetros. Se tomaron datos para entrenamiento y prueba en proporciones del 70% y 30%, respectivamente. Emplearon validación cruzada quíntuple (CV=5) para la verificación. Ofreciendo un estudio comparativo para estas cuatro metodologías. Los conjuntos de datos de Cleveland, Hungría, Suiza y Long Beach V y UCI Kaggle se utilizan para analizar el rendimiento de los modelos. En el análisis

[8] Ahmad, G.N., Fatima, H., Ullah, S., Saidi, A.S. & Imdadullah (2022), "Efficient Medical Diagnosis of Human Heart Diseases Using Machine Learning Techniques With and Without GridSearchCV,". doi: 10.1109/ACCESS.2022.3165792.

se encuentra que el Clasificador de aumento de gradiente extremo (GBC) con "GridSearchCV" proporciona las precisiones de prueba y entrenamiento más altas y casi comparables: 100 % y 99,03 % para ambos conjuntos de datos (Hungría, Suiza &; Long Beach V y UCI Kaggle).

Madushani et al (2023)[9] Estudiaron el empleo varias técnicas de aprendizaje automático, incluidas RF (Random Forest), DT (Decisión Tree), XGB (Extreme Gradient Boosting), KNN (K-vecinos más cercanos) y un modelo LR tradicional (regresión logística), para predecir y clasificar la gravedad de los accidentes en las autopistas. Los datos incluyen la condición de la superficie de la carretera (seca o mojada), las condiciones climáticas (despejadas, nubladas/con niebla o lluviosas), las condiciones de iluminación (luz diurna, luz diurna parcial, noche con buen alumbrado público o noche sin alumbrado público), alineación de la vía (curva o recta) y ubicación (en un cruce o no en un cruce). La gravedad del accidente se clasificó en dos niveles: accidentes mortales/graves y todos los demás accidentes. El estudio utilizó 223 instancias para entrenamiento y 56 para pruebas, con dos niveles de gravedad de choques: muy severo para fatales/graves y poco severo para no graves/daños a la propiedad únicamente. Para evaluar los modelos se utilizaron métricas de rendimiento como sensibilidad, precisión, exactitud, puntuación F1, tasa de falsos positivos (FPR) y AUC. El modelo LR logró una sensibilidad de 0.83, una precisión de 0,78, una exactitud de 0,77 y un AUC de 0,73 en la predicción de la gravedad de los accidentes. El estudio reveló

[9] Madushani, J. P. S. S., Sandamal, R. M. K., Meddage, D. P. P., Pasindu, H. R., & Gomes, P. I. A. (2023). Evaluating expressway traffic crash severity by using logistic regression and explainable & supervised machine learning classifiers. https://doi.org/10.1016/j.treng.2023.10019

que los métodos de aprendizaje automático superaron al modelo LR en la predicción y clasificación de la gravedad de los accidentes. Específicamente, el modelo XGB demostró el mayor rendimiento de predicción con una sensibilidad de 0.94, una precisión de 0,80 una exactitud de 0.82 , una puntuación F1 de 0.87 y AUC de 0.917.

1.3. DESCRIPCIÓN DE LA REALIDAD PROBLEMÁTICA

La empresa de transmisión de energía eléctrica peruana ETEP tiene a la fecha marcadas brechas con el uso y aprovechamiento de herramientas de inteligencia artificial, lo cual es una realidad de la mayoría de las empresas en Perú. Según una encuesta realizada el por el comité de Estrategia Nacional de Inteligencia Artificial solo el 7% de las instituciones públicas afirmaron que usan la Inteligencia Artificial en sus labores (PCM, 2021)[10]. En el caso de Latinoamérica, según el Global AI Adoption Index 2022 (Morning Consult) el 29% de las empresas han implementado proyectos de IA (Telefónica, 2023)[11]. A nivel mundial un 47% de las grandes empresas están invirtiendo en esta tecnología. Según Cloudera, plataforma de machine learning y análisis de datos. Encuesta realizada entre 200 grandes corporaciones de España, Francia, Alemania y Reino Unido (Hostelsur, 2019)[12].

[10] PCM (18 de mayo del 2021) Estrategia Nacional de Inteligencia Artificial (ENIA). https://www.gob.pe/institucion/pcm/informes-publicaciones/1929011-estrategia-nacional-de-inteligencia-artificial.

[11] Telefónica (23 de marzo del 2023). Inteligencia artificial: Cinco claves para el avance en las empresas peruanas.https://telefonica.com.pe/inteligencia-artificial-cinco-claves-para-el-avance-en-las-empresas-peruanas.

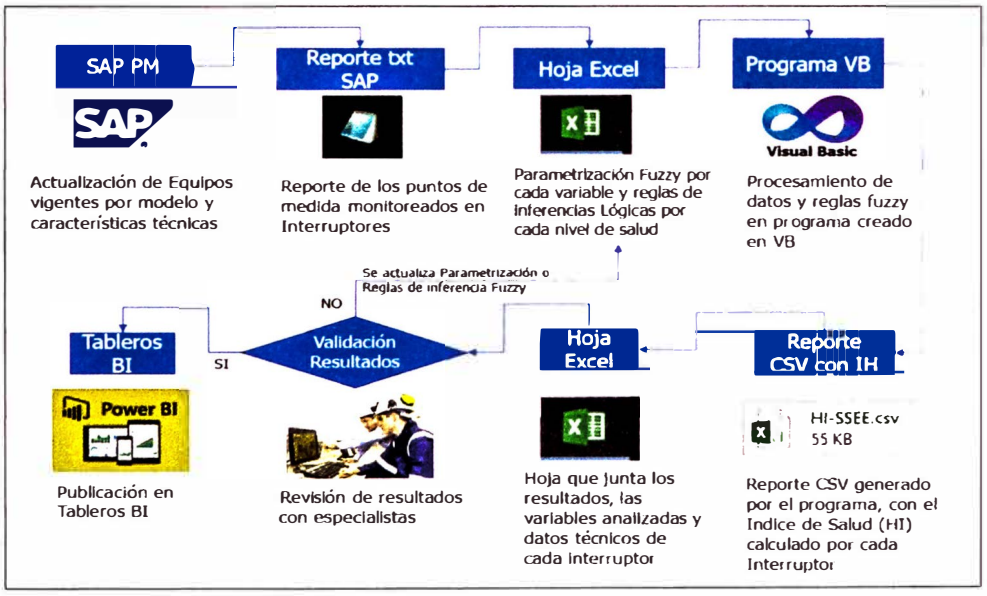
[12] Hostelsur (03 de enero del 2019). Nueve de cada diez grandes empresas están invirtiendo en machine learning.https://www.hosteltur.com/125780_nueve-de-cada-diez-grandes-empresas-estan-invirtiendo-en-machine-learning.

La realidad problemática identificada es que actualmente se vienen realizando procesos matemáticos engorrosos de aplicación de múltiples reglas, para cada variable monitoreada en los interruptores de potencia y en función a estas entradas se usan criterios de inferencias mediante lógicas Fuzzy para establecer salidas (diagnósticos). Todo ello conlleva a un arduo proceso y en la práctica al contrastar los resultados con los especialistas técnicos no se logra una efectividad del 100% con los diagnósticos. Esto debido a diferentes particularidades no mapeadas en la parametrización de variables o inferencias lógicas establecidas previamente, lo que conlleva cada vez a un reproceso en la actualización de los umbrales de cada parámetro y en los criterios de inferencias lógicas predefinidas. En consecuencia, hace necesario una revisión minuciosa de los resultados para validar los diagnósticos obtenidos. Todo este proceso de cálculo, diagnóstico y validación puede tomar hasta 4 semanas en realizarse, representando una gran brecha para hacer un diagnóstico continuo y de manera más oportuna para las decisiones en la gestión del mantenimiento.

En la figura 2 se esquematiza la complejidad del proceso actual donde podemos observar que se parte de una actualización del listado de equipos y del reporte de puntos de medida monitoreados en los interruptores, los cuales vienen hacer el registro de variables eléctricas y mecánicas monitoreadas en estos equipos luego con el diseño de una parametrización fuzzy triangular por cada variable usada (05 Variables en total) y las reglas de inferencia lógicas por cada nivel de salud (04 Niveles de salud), son procesados en un programa desarrollado en Visual Basic el cual toma toda esta información y emite un reporte en formato "CSV" con el diagnostico de salud por cada interruptor, este reporte luego es ampliado con información de características técnicas

del interruptor que ayudan a un mejor entendimiento y análisis posterior de los resultados tales como tipo de fabricante, modelo, tipo de accionamiento y otros datos característica de placa, con todos estos datos los resultados son revisados con nuestro equipo de especialistas en interruptores de encontrar observaciones en los resultados se actualiza las fronteras de las parametrizaciones fuzzy y/o las reglas de inferencia lógicas, repitiéndose todo el proceso anterior, finalmente al ser validados todos los diagnósticos, estos son publicados en un tablero de Power BI para conocimiento de todo el área de Operación y mantenimiento.

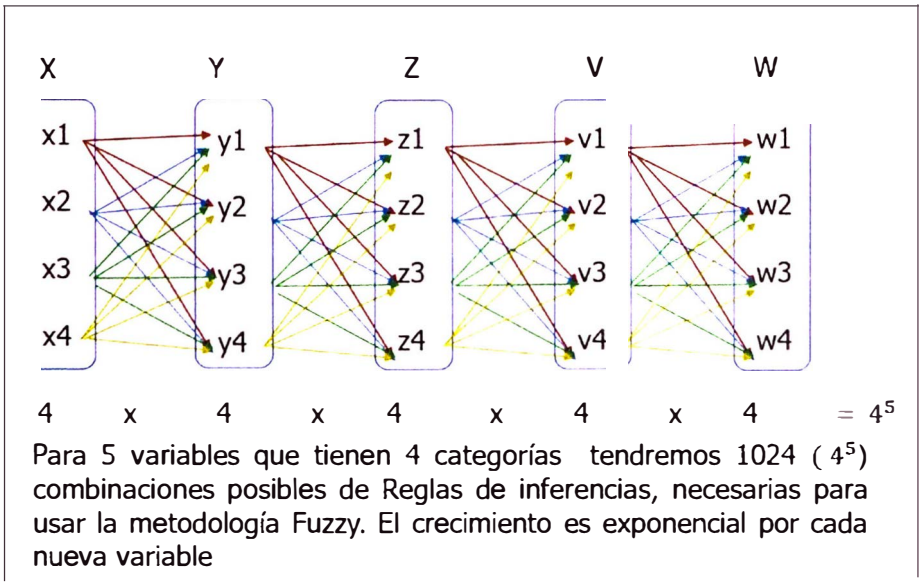
Figura 2. *Proceso para determinar el Índice de Salud con Metodología Fuzzy en interruptores de potencia*



Nota: Elaboración propia

En la figura 3 se cuantifica la magnitud de las reglas de inferencias fuzzy que se debe implementar por cada variable y categoría, al tener 05 variables en la metodología actual: Humedad en gas SF6 (ppm), Pureza en gas SF6 (%), Dióxido de azufre en gas SF6 (ppm), Resistencia de contactos (μΩ) y Edad del interruptor (años) y cada una de estas variables tiene 04 categorías (Muy Bueno, Bueno, Pobre y Muy Pobre) entonces existen 4^5 combinaciones posibles entre las categorías de cada variable, lo que resulta en 1024 reglas de inferencia necesarias para determinar un diagnóstico de salud final. El tiempo necesario para procesar y validar todo el proceso puede ser variable teniendo en un escenario pesimista un tiempo de 04 semanas como se mencionó anteriormente.

Figura 3. *Cantidad de Reglas de inferencias fuzzy que resultan difícil su revisión*



Nota: Elaboración propia

1.4. FORMULACIÓN DEL PROBLEMA.

1.4.1. PROBLEMA GENERAL:

¿En qué medida el diseño de modelos predictivos basados en machine learning se constituye en una herramienta para optimizar el diagnóstico del índice de salud en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP?

1.4.2. PROBLEMAS ESPECÍFICOS:

1. ¿En qué grado el diseño de modelos predictivos basados en machine learning me permitirá aumentar la efectividad del diagnóstico del índice de salud en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP?
2. ¿Cómo impacta el diseño de modelos predictivos basados en machine learning en el tiempo empleado para el diagnóstico del índice de salud en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP?
3. ¿Hasta qué punto el diseño de modelos predictivos basados en machine learning me permitirá optimizar los costos incurridos por fallas imprevista en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP?

1.5. JUSTIFICACIÓN

El diseño de un modelo predictivo para evaluar la salud de los interruptores en sistemas eléctricos presenta una relevancia fundamental en la optimización de la operación y el mantenimiento de infraestructuras críticas. Los interruptores desempeñan un papel esencial en la gestión y distribución de la energía eléctrica, y su funcionamiento incorrecto o fallos inesperados pueden acarrear costosos tiempos de inactividad y riesgos de seguridad significativos.

Justificación Técnica:

La aplicación de técnicas de Machine Learning permite el análisis avanzado de datos operativos y registros históricos de los interruptores de potencia. La construcción de modelos predictivos precisos mediante algoritmos de aprendizaje automático identifica patrones y comportamientos en los datos que están más allá de la capacidad humana de detección. Esto facilita la predicción temprana de potenciales problemas y degradaciones en los interruptores, permitiendo una programación eficiente de los mantenimientos preventivos y minimizando los tiempos de inactividad no planificados.

Justificación Económica:

La implementación de modelos predictivos basados en Machine Learning para el diagnóstico de la salud de los interruptores conlleva beneficios económicos significativos. La reducción de los costos asociados con interrupciones no programadas y la optimización de los programas de mantenimiento se traducen en ahorros financieros sustanciales. Además, la

capacidad para prever problemas y realizar acciones preventivas evita los gastos imprevistos en reparaciones de emergencia y contribuye a una gestión más eficiente de los recursos.

Justificación de Seguridad:

La seguridad operativa en el ámbito de la transmisión eléctrica es primordial. La operación incorrecta o el fallo de los interruptores de potencia pueden dar lugar a apagones, daños a equipos y riesgos para el personal. Al utilizar modelos predictivos basados en Machine Learning, se mejora la capacidad de anticipar problemas y se evitan situaciones peligrosas, garantizando un sistema eléctrico más seguro y confiable.

1.6. IMPORTANCIA

Aplicar el machine learning para el diagnóstico del índice de salud de los interruptores de potencia repercutirá en una sustancial mejora en la oportunidad y efectividad de todo el proceso que se sigue actualmente y en consecuencia también de la gestión de otras áreas que depende de este indicador para la toma de decisiones como la de planificar compras de repuestos, nuevos interruptores o la de planificar mantenimientos mayores.

1.7. OBJETIVOS

1.7.1. OBJETIVO GENERAL:

Diseñar modelos predictivos basados en machine learning para optimizar el diagnóstico del Índice de Salud en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP.

1.7.2. OBJETIVOS ESPECÍFICOS:

1. Aumentar la efectividad del diagnóstico del índice de salud diseñando modelos predictivos basados en machine learning en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP.
2. Disminuir el tiempo empleado para el diagnóstico del índice de salud diseñando modelos predictivos basados en machine learning en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP.
3. Optimizar los costos incurridos por fallas imprevistas diseñando modelos predictivos basados en machine learning en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP.

1.8. HIPÓTESIS

1.8.1. HIPÓTESIS GENERAL:

El diseño de un modelo predictivo basado en machine learning influye significativamente en la optimización del diagnóstico del índice de salud en interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP.

1.8.2. HIPÓTESIS ESPECIFICAS:

1. Se aumentará la efectividad del diagnóstico del índice de salud diseñando de modelos predictivos basado en machine learning en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP.
2. Se disminuirá el tiempo empleado para el diagnóstico del índice de salud diseñando modelos predictivos basado en machine learning en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP.
3. Se optimizará los costos incurridos por fallas imprevistas diseñando modelos predictivos basado en machine learning en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP.

1.9. VARIABLES, DIMENSIONES E INDICADORES

En esta sección describiremos la variable independiente y dependiente, propondremos los indicadores y desarrollaremos la operacionalización de las variables. En la tabla 1 podemos observar el desarrollo de una matriz con la operacionalización de las variables.

***Tabla 1**. Matriz de operacionalización de las variables*

Variable	Definición	Dimensiones	Indicador	Escala
Independiente: "Modelos predictivos basados en machine learning"	Algoritmos matemáticos modelados por inteligencia artificial que aprenden y se entrenan con datos cualitativos y/o cuantitativos intentando predecir una variable objetivo	Calidad del Modelo	-Efectividad del diagnóstico del Índice de Salud	(0-100%)
Dependiente: "Diagnóstico del Índice de salud en interruptores de potencia"	Pronóstico del estado de deterioro irreversible de un interruptor, usado como indicador de reemplazo del equipo o de necesidad de un mantenimiento mayor	Eficiencia del Proceso	-Tiempo para diagnosticar el Índice de Salud -Costos por fallas imprevistas	(días) (USD)

Nota: Elaboración propia

1.10. DISEÑO METODOLÓGICO

En esta sección se busca explicar el proceso que se siguió en el desarrollo de la investigación, en el cual se consideraron los aspectos que a continuación se indican.

1.10.1. UNIDAD DE ANÁLISIS

La unidad de análisis de la presente investigación comprende los interruptores de potencia que se encuentran operando a nivel nacional en las

subestaciones eléctricas de alta tensión de la empresa de transmisión de energía eléctrica peruana ETEP. Cuyo nivel de tensión es mayor o igual a 60kV y cuyo medio de extinción del arco eléctrico es el gas SF6.

1.10.2. FUENTES DE INVESTIGACIÓN

Las fuentes de información primaria provienen principalmente del sistema informático de la empresa ETEP que utiliza el módulo PM del software SAP el cual permite la planificación, el procesamiento y el reporte de tareas de mantenimiento. Las fuentes de información secundaria usadas como referencia para el planteamiento del modelo predictivo provienen principalmente de revistas indexadas, tesis de maestría y libros relacionado al machine learning. Estas se describen con mayor detalle a continuación:

- Fuentes de información primaria:
    1. Historial de parámetros monitoreados en los interruptores de potencia de los últimos 6 años (2017-2022)
    2. Historial de Índices de Salud de los interruptores de potencia de los últimos 3 años. (2020-2022)
    3. Manuales de Operación según modelo y fabricante de los Interruptores potencia involucrados en el presente estudio.
    4. Estrategias de Mantenimiento para interruptores potencia
    5. Guías de aplicación normalizada para interruptores potencia
- Fuentes de información secundaria:
    1. Revistas indexadas
    2. Tesis de maestría.
    3. Libros

1.10.3. TIPO Y NIVEL DE LA INVESTIGACIÓN

La presente investigación es del tipo "Aplicada" puesto que se busca aplicar una nueva tecnología (machine learning) para optimizar un proceso en el área de mantenimiento (diagnóstico de salud en interruptores). Este tipo de investigaciones están orientadas a mejorar, perfeccionar u optimizar el funcionamiento de los sistemas, los procedimientos, normas, reglas tecnológicas actuales a la luz de los avances de la ciencia y la tecnología; (Ñaupas, 2013).[13]

Respecto al nivel de investigación es principalmente "Explicativo", este nivel de investigación se enfoca en explicar por qué ocurre un fenómeno y en qué condiciones se manifiesta (Hernández et al., 2014).[14]. En la presente tesis se explica como las variables monitoreadas en los interruptores de potencia inciden en el diagnostico final de la salud de los interruptores y como se logra optimizar este diagnóstico usando el machine learning.

1.10.4. MÉTODO DE DISEÑO DE LA INVESTIGACIÓN

El diseño de la presente investigación tiene un enfoque "Cuantitativo" dado que se esta trabajando con datos numéricos y estadísticos de las variables eléctricas, mecánica y físicas que se monitorean en interruptores de potencia y es del tipo "No experimental" dado que no se está recreando alguna

[13] Ñaupas, H. (2013). Metodología de la investigación científica y elaboración de tesis. Lima: Universidad Nacional Mayor de San Marcos.

[14] Hernández, S., Fernández, R., & Baptista, P. (2014). Metodología de la investigación (6° Ed.). México, D.F., México: McGraw Hill Interamericana.

situación en los interruptores, sino se observan y analizan el historial de datos ya existentes de sus variables monitoreadas.

1.10.5. PERÍODO DE ANÁLISIS

Para el desarrollo de la presente investigación se usó la última información histórica disponible de los últimos 6 años (2017-2022) de cada uno de los parámetros controlados en los interruptores de potencia durante sus mantenimiento preventivos, predictivos y correctivos.

1.10.6. TÉCNICAS DE RECOLECCIÓN Y PROCESAMIENTO DE DATOS

La información es recolectada directamente consultando en el sistema SAP-PM y extrayendo un reporte de todos los interruptores que cumplen con las características técnicas definidas en el alcance de la presente investigación (Tensión: >=60kV y medio de extinción: SF6) los cuales resultan en 530 interruptores al cierre del año 2022 (31/12/2022), existiendo los niveles de 60kV, 138kV, 220kV y 500kV.

Finalmente se crea una base de datos con las siguientes características por cada interruptor:

- Nivel de Voltaje en kV
- Tipo de Fabricante
- Humedad en gas SF6 en ppmv
- Pureza en gas SF6 en %
- Dióxido de azufre en gas SF6 en ppmv
- Resistencia de Contacto en μΩ
- Edad del Interruptor en años

Luego se hará un preprocesamiento para adecuar la data en filas (equipos) y columnas (Parámetros) de manera que pueda ser entendida como tal, por

el lenguaje de programación Python, el cual tiene bibliotecas especializadas de análisis de datos tales como: pandas, sklearn, matplotlib, scikitplot.

Finalmente debemos tener un archivo plano donde cada fila debe representar a un interruptor y su último documento de medida por cada punto de medida existente (los cuales será los datos de las variables predictoras), así como el diagnóstico vigente del índice de salud por cada interruptor (variable objetivo). Luego se usará el programa Python para revisar la calidad de los datos, tratar los outliers y tratar los datos nulos.

Sistemas y programas informáticos de recolección y procesamiento de datos:

- SAP-PM
- Python
- Excel

CAPITULO II. MARCO TEÓRICO Y CONCEPTUAL

Para poder entender el desarrollo de la presente tesis debemos conocer conceptos previos en torno a 03 temas en específicos: "interruptores de potencia", "índice de salud de equipos" y "Machine learning". Teniendo claro todo el marco teórico de estos títulos estaremos en la capacidad de responder las siguientes preguntas: ¿Qué es un interruptor de potencia? ¿Por qué es importante para una subestación eléctrica de alta tensión? ¿Cuáles son sus principales parámetros controlados? ¿Cuáles son sus principales modos de falla? ¿Qué es el índice de salud de un interruptor de potencia? ¿Qué es el machine learning? ¿Cuáles la diferencia entre un modelo supervisado y un modelo no supervisado? ¿Cuáles son los principales algoritmos de clasificación y regresión? ¿Cuáles son las principales métricas que validan la efectividad de un modelo de aprendizaje máquina? .En ese sentido durante el desarrollo del presente capitulo iremos respondiendo uno a uno las preguntas planteadas.

2.1. INTERRUPTORES DE POTENCIA

Los interruptores de potencia son equipos diseñados para resistir un alto voltaje de trabajo, flujo de altas corrientes en su estado normal de operación y corrientes de corto circuito durante un estado de falla en el circuito eléctrico que protegen. Tienen la capacidad de poder abrir o cerrar un circuito eléctrico con carga, a diferencia de un seccionador que solo está diseñado para

hacerlo sin carga. (Mejia, 2003) [15]. Estos equipos pueden ser clasificados o agrupados de distintas maneras que a continuación vamos a detallar:

a) Por su nivel de Tensión. - Pueden ser de Media tensión: >1kV & <=35kV, Alta Tensión: >35kV & <= 230kV y de muy alta Tensión: >230kV. (MINEM, 2011) [11]. En Perú tenemos Interruptores de potencia hasta 500kV. En la figura 4 se muestra un interruptor típico de muy alta tensión.

***Figura 4**.Interruptores de Potencia 500kV*

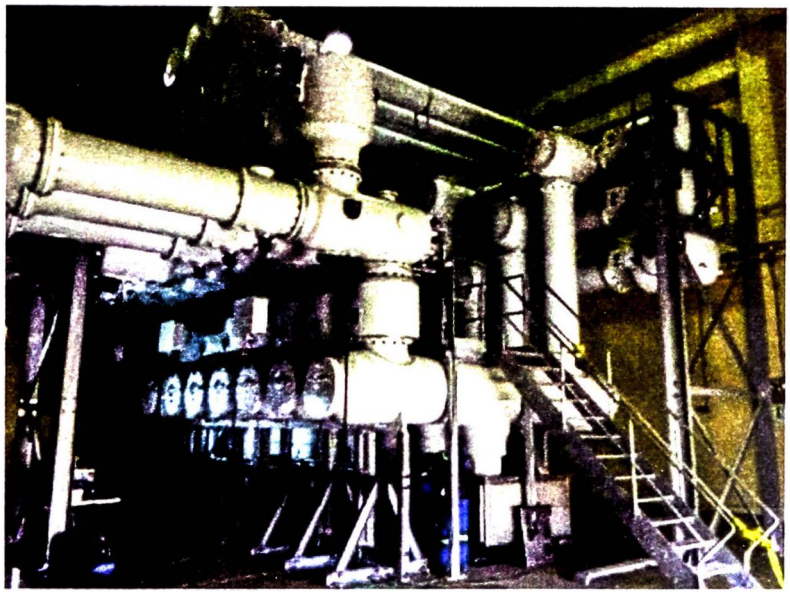
Nota: Tomada de página web de Docplayer. Disponible en: https://docplayer.es/docs-images/44/14380956/images/page_1.jpg

b) Por su lugar de instalación. – Indoor: ubicados dentro de cubículos o ambientes bajo techo y Outdoor: expuestos al medio ambiente, están

[15] Mejia, V. (2003). Subestaciones de Alta y Extra Alta Tensión. Editorial Impresiones graficas Ltda. Colombia. pp822

diseñados para resistir las diferentes condiciones climáticas. (Mejia, 2003)[16]. En la figura 5 se muestra un interruptor tipo GIS en caseta Indoor.

Figura 5*.Interruptores de Potencia Indoor*



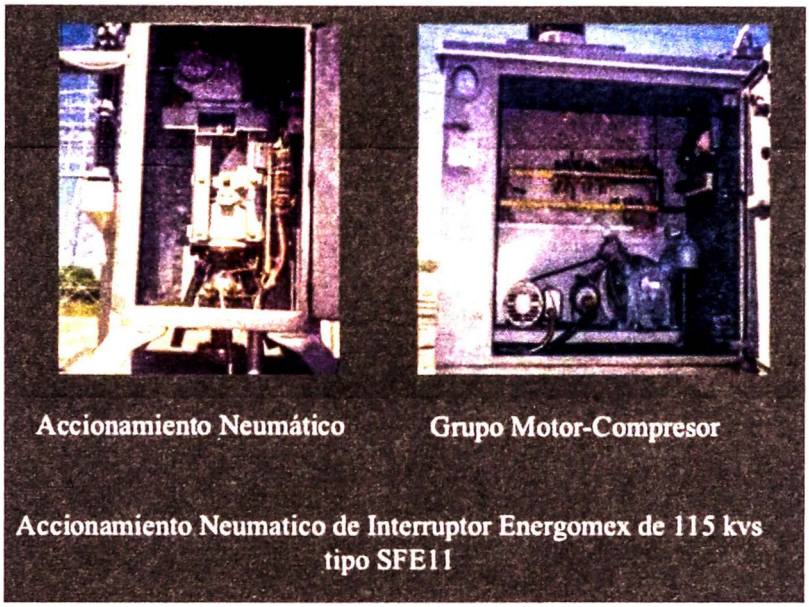
Nota: Tomada de página web de Sector Electricidad. Disponible en: http://www.sectorelectricidad.com/wp-content/uploads/2019/04/interruptor-gis-500-kv.jpg

c) Por su mecanismo de accionamiento. – Tenemos los de accionamiento tipo mecánico (uso de resortes como medio de almacenamiento de energía y es necesario un motor para su carga), tenemos los de accionamiento tipo hidráulico (uso de aceite a presión para almacenar la energía, se necesita de bomba), tenemos los de accionamiento tipo neumático (uso de aire comprimido para almacenamiento de energía, es necesario una moto compresora para mantener la presión

[16] Mejia, V. (2003). Subestaciones de Alta y Extra Alta Tensión. Editorial Impresiones graficas Ltda. Colombia. pp822

constante). En la figura 6 se muestra un tablero de mando de accionamiento tipo neumático.

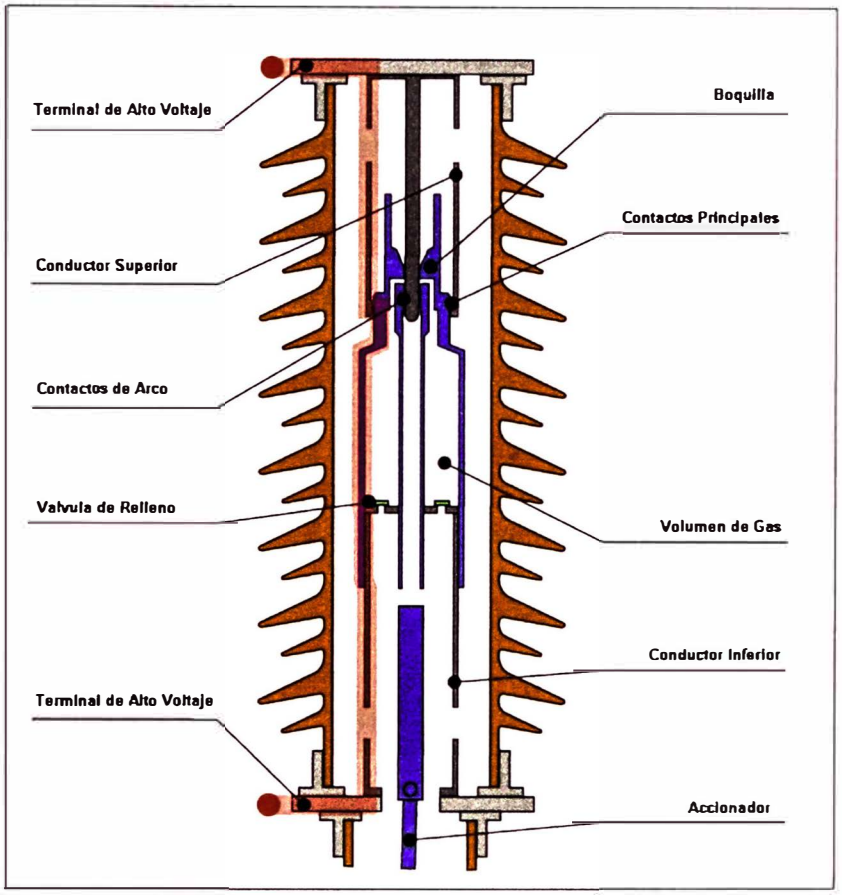
***Figura 6**.Interruptores de Potencia Accionamiento Neumatico*



Nota: Tomada de página web de slideshare. Disponible en: https //image.slidesharecdn.com/interruptoresdepotencia-090609140916-phpapp01/95/interruptores-de-potencia-19-728.jpg?cb=1244556631

d) Por su tipo de aislamiento interno. - tenemos los interruptores de vacío (usan el aire comprimido para el soplado del arco), interruptores de SF6 (usan el hexafluoruro de azufre como medio de extinción del arco) y los interruptores de aceite (usan el aceite como medio de extinción del arco). En la figura 7 se muestra un corte típico de la cámara de corte de un interruptor de potencia en SF6.

***Figura 7**.Camara de Interruptor con aislamiento en SF6*



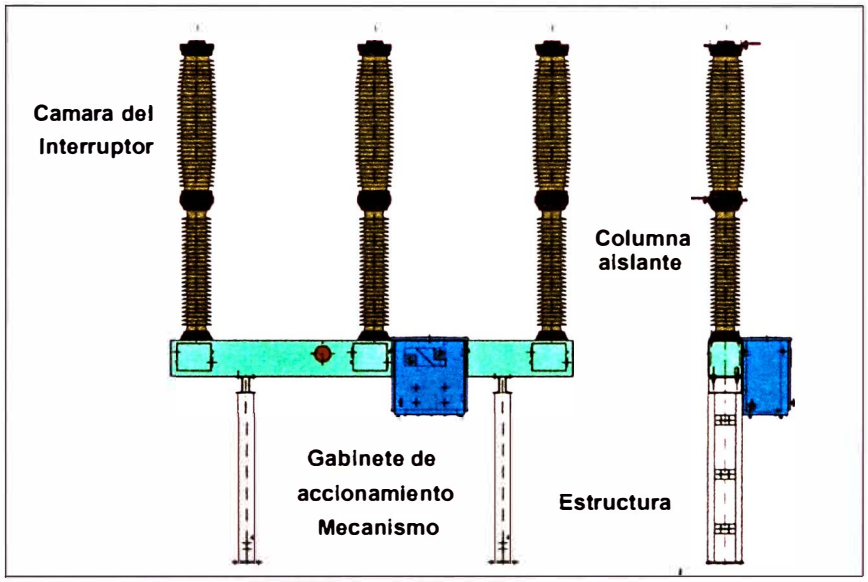
Nota: Tomada de página web de Megas. Disponible en: http://megas.com.do/wp-content/uploads/2017/08/Camara-de-Extincion.png

e) Por su tipo de aislamiento externo. - Tenemos interruptores con aislamiento natural de porcelana, interruptores de aislamiento natural tipo polimérico e interruptores con aislamiento mixto de porcelana recubierta con grasa silicona e incluso de porcelana recubierta con goma silicona)

f) Por su tipo de mando de apertura. - Tenemos los interruptores de mando tripolar (las 03 faces abren y cierran al mismo tiempo) y los de mando monopolar cada fase es independiente durante su apertura y

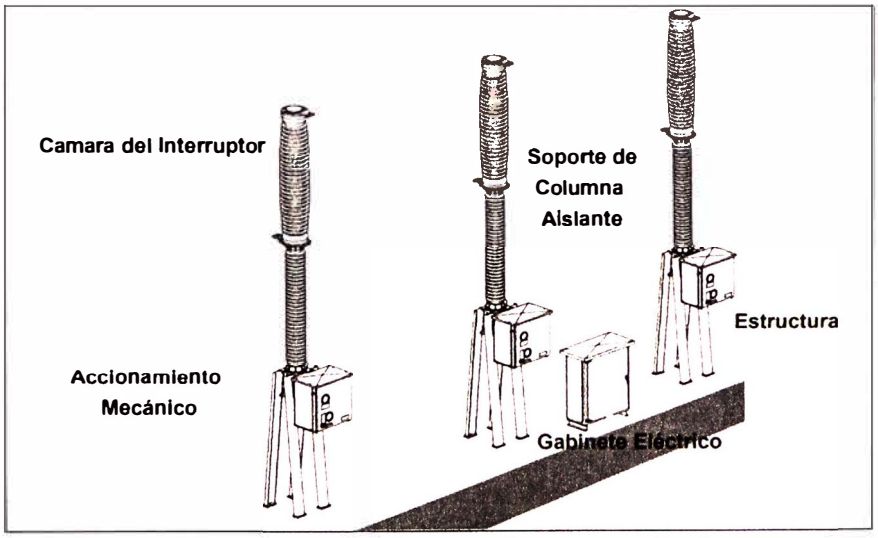
cierre). En las figuras 8 y 9 se muestra el diseño y distribución de un interruptor tipo tripolar y monopolar respectivamente.

***Figura 8**. Interruptor de Mando Tripolar*



Nota: Tomada de página web de Cigre. Disponible en: https://www.cigre.cl/wp-content/uploads/2017/03/Perturbaciones-transitorias-v1_2.pdf

***Figura 9**. Interruptor de Mando Monopolar*



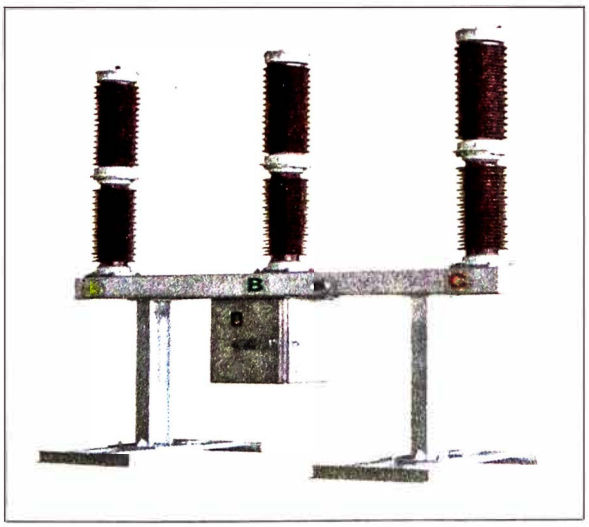
Nota: Tomada de página web de Cirgre. Disponible en: https://www.cigre.cl/wp-content/uploads/2017/03/Perturbaciones-transitorias-v1_2.pdf

g) Por la ubicación de sus contactos principales. - Tenemos a los de tanque muerto (contactos principales en la base del interruptor) y los de tanque vivo (contactos principales en la parte superior del interruptor). En la figura 10 se muestra un interruptor típico de tanque muerto y en la figura 11 se muestra un interruptor típico de tanque vivo.

Figura 10. *Interruptor de Tanque Muerto*

Nota: Tomada de página web de Sites Google. Disponible en: https://sites.google.com/site/subestacioneseléctricasdsu/_/rsrc/1526001181302/recursos-1/descarga.jpg?height=231&width=320

Figura 11. *Interruptor de Tanque Vivo*



Nota: Tomada de página web de Tiendas equipo Industriales. Disponible en: http://tienda.equiposindustriales.com/web/image/product.template/27709/image_1024?unique=fb7f2b7

h) Por su cantidad de cámaras de extinción. - Tenemos por ejemplo a los de una cámara de extinción por fase o a los de dos cámaras de extinción por fase. En la figura 12 se muestra un interruptor típico de doble cámara de extinción.

Figura 12. *Interruptores de Doble camara de Extinción*

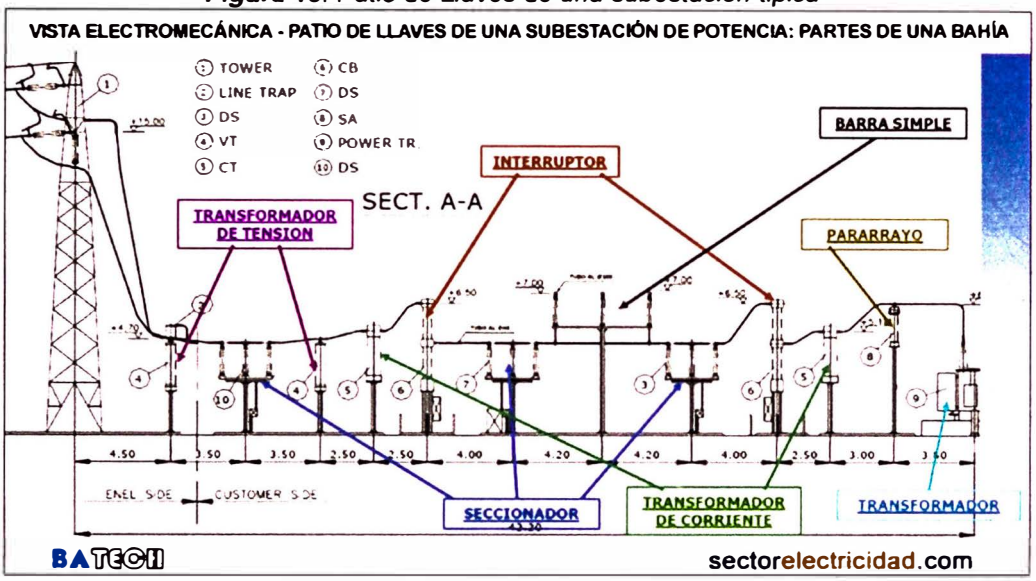
Nota: Tomada de página web de Research Gate. Disponible en: https://www.researchgate.net/profile/Sergio-Yanez-5/publication/293010339/figure/fig1/AS:580553373753344@1515426118647/Figura-2-Interruptores-de-potencia-tipo-FL245.png

2.1.1. IMPORTANCIA DE LOS INTERRUPTORES DE POTENCIA

Una subestación eléctrica de alta tensión está compuesta por varias celdas o también conocidas como bahías de transmisión las cuales a su vez esta compuestas por diferentes equipos de patio como: Transformadores de potencia, interruptores de potencia, Transformadores de corriente, transformadores de tensión, seccionadores de línea, seccionadores de barra, seccionadores de enlace, Pararrayos y trampas de ondas.(Harper, 2012)[17] Cada una cumpliendo un rol muy importante en el proceso del transporte de la energía, sin embargo dentro del estudio de criticidad que se le puede hacer a estos equipos de patio, considerando su riesgo operativo, su costos de reparación, su impacto en la reputación de la empresa, su impacto en la seguridad del personal y hasta su impacto en el medio ambiente, sale a relucir en buena cuenta que los equipos más críticos en una subestación son los Transformadores e interruptores de potencia en ese orden. Para la presente investigación se optó por analizar los interruptores de potencia debido a su superioridad numérica en cantidades de equipos y de información de datos respecto a lo transformadores de potencia, según el mapeo estadístico realizado por cada transformador de potencia se tiene hasta nueve interruptores de potencia en una subestación lo que nos da una mayor opción de datos para el análisis predictivo. En la figura 13 se muestra la ubicación y distribución de los interruptores dentro de una celda de subestación de alta tensión.

[17] Harper, E. (2012) Manual del Técnico en Subestaciones Eléctricas Industriales y Comerciales. Editorial Limusa S.A. España.pp428

***Figura 13**. Patio de Llaves de una subestación tipica*



Nota: Tomada de página web de Sector Electricidad. Disponible en: https://www.sectorelectricidad.com/wp-content/uploads/2014/11/patio-de-llaves.png

2.1.2. PARTES PRINCIPALES DE LOS INTERRUPTORES DE POTENCIA

Como ya hemos visto los interruptores de potencia dependiendo del tipo accionamiento y del medio de extinción tienen diferentes características sin embargo la mayoría coincide en almenos estas 04 partes principales: su cámara de extinción, su aislador soporte, su circuito de control y su mecanismo de operación.

a) Cámara de extinción. – Aquí podemos encontrar los contactos de cobre del interruptor y el medio de extinción que sirve para soplado del arco producido durante la apertura o cierre del interruptor (SF6, aceite, aire comprimido)

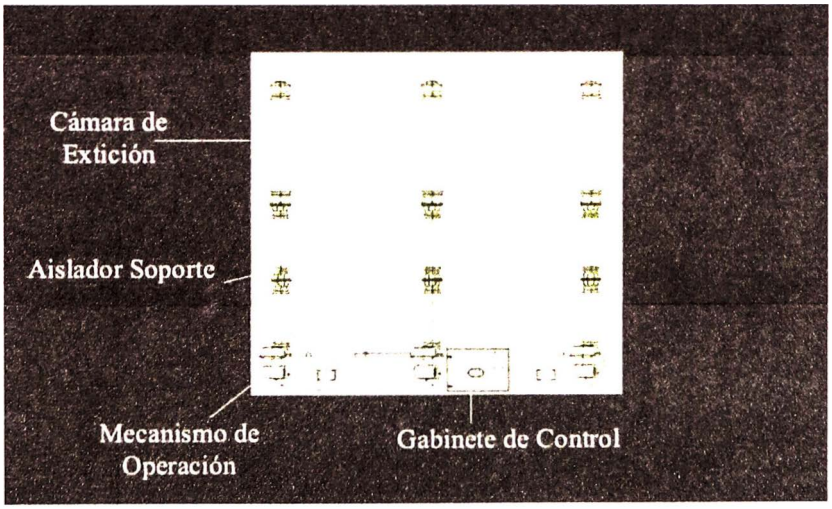
b) Aislador Soporte. - Sirve para que los interruptores tengan la distancia de seguridad mínima necesario contra tierra, generalmente de material de porcelana y en algunos casos adicionados con grasa

silicona, para mejorar el aislamiento y ayuda también para prolongar las tareas de limpieza de aislamiento.

c) Circuito de control. - contiene todos los circuitos y componentes del control del interruptor como, por ejemplo: Selectores de posición remoto, local y apagado, pulsadores de apertura y cierre, interruptores termomagnéticos, contactores, guardamotores, resistencias de calefacción, temporizadores, manómetro.

d) mecanismo de operación. – para un interruptor de accionamiento mecánico está conformado por el motor, los resortes mecánicos que almacenan la energía del motor, y cada uno de los engranajes mecánicos necesarios para la apertura o cierre del interruptor. Adicionalmente para los de accionamiento hidráulico o neumáticos se debe considerar la bomba o motocompesora y sus tuberías o ductos. En la figura 14 se muestra las partes típicas de un interruptor de potencia.

Figura 14. *Partes de un Interruptor de Potencia*



Nota: Tomada de página web de slideshare. Disponible en: https://image.slidesharecdn.com/interruptoresdepotencia-090609140916-phpapp01/95/interruptores-de-potencia-11-728.jpg?cb=1244556631

2.1.3. PARÁMETROS MEDIBLES EN LOS INTERRUPTORES DE POTENCIA

Hablar de los parámetros medibles en un interruptor es hablar de sus variables de control y estos están orientados a cada una de las partes principales del interruptor, estas pueden ser del tipo numéricas y del tipo categóricas, en ese sentido empezaremos a listar cada uno de los principales parámetros de control que se podría monitorear en un interruptor de potencia

a) Para la cámara de extinción tenemos:
   - La resistencia estática de contactos (uhm)
   - La resistencia dinámica de contactos (uhm)
   - Presencia de Humedad en gas SF6 (ppm)
   - Presencia de SO2 en gas SF6 (ppm)
   - Grado de Pureza del gas SF6 (%)
   - Tiempo de apertura de cámara (ms)
   - Tiempo de cierre de cámara (ms)
   - Discrepancia máxima de apertura (ms)
   - Discrepancia máxima al cierre (ms)
   - Cantidad de operaciones en fallas (UND)
   - Corrientes acumulada de falla (kA)
   - Tiempo acumulado de falla (ms)

b) Para el mecanismo de operación
   - Corriente de apertura del motor (A)
   - Corriente de cierre del motor (A)
   - Resistencia de aislamiento del motor (Mohm)
   - Componente mecánico averiado (SI/NO)

- Contador de operaciones (UND)
- Fuga Aceite actuador hidráulico (SI/NO)
- Fuga Aire actuador neumático (SI/NO)
- Resorte descargado (SI/NO)

c) Para el circuito de control

- Componente eléctrico con avería (SI/NO)
- Resistencia de calefacción con avería (SI/NO)

d) Para el aislador soporte

- Estado de aislamiento (BUENO, MALO)

En la figura 15 se muestra un ejemplo de pruebas eléctricas que se realiza en interruptores de potencia para determinar sus parámetros eléctricos, el alcance dependerá del tipo de maleta usada.

Figura 15. *Pruebas eléctricas en Interruptores de Potencia*

Nota: Tomada de página web de Omicron Energy. Disponible en: https://www.omicronenergy.com/fileadmin/_processed_/2/6/csm_2020-09-Coverstory-quote_243561351b.jpg

2.1.4. MODOS DE FALLA EN LOS INTERRUPTORES DE POTENCIA

Pueden ocurrir múltiples modos de falla en un interruptor presentándose de manera aislada cada una o incluso pueden concurrir varios modos de falla en simultaneo, en esta sección describiremos estos modos de fallas de acuerdo con la parte del interruptor donde se presenta:

a) Para la cámara de extinción tenemos:
   - Alta resistencia de contacto (Por ejemplo > 2 veces su nominal)
   - Alta humedad en gas SF6 (Por ejemplo > 200ppm)
   - Alta concentración de SO2 (Por ejemplo > 2ppm)
   - Baja pureza de SF6 (por ejemplo <97%)
   - Alto tiempo de apertura de cámara (Por ejemplo > 1.5 veces su nominal)
   - Alto tiempo de cierre de cámara (Por ejemplo > 1.5 veces su nominal)
   - Alta discrepancia máxima de apertura (por ejemplo > 2.5ms)
   - Alta discrepancia máxima al cierre (por ejemplo > 5ms)
   - Alta cantidad de Operaciones (por ejemplo >10000 maniobras acumuladas)
   - Alta cantidad de corriente acumulada de falla (por ejemplo, la sumatorias de corriente de fallas al cuadro >20000kA^2)
   - Fuga de gas SF6
   - Falla en manómetro

b) Para el mecanismo de operación
   - Alta corriente de apertura del motor (Por ejemplo >2 veces su nominal)

- Alta corriente de cierre del motor (Por ejemplo >2 veces su nominal)
- Baja resistencia de aislamiento del motor (Por ejemplo < 1 Mohm)
- Falla en el contador de maniobras
- Falla en el motor

c) Para el circuito de control

- Falla en el mando eléctrico

d) Para el aislador soporte

- Presencia de Efluvios

En la figura 16 se muestra un mantenimiento correctivo típico en interruptores de potencia.

Figura 16. *Mantenimiento Correctivo en Interruptores de Potencia*

Nota: Tomada de página web de Metromecanica. Disponible en: https://metromecanica.com.pe/wp-content/uploads/2019/10/Screenshot_23-1.jpg

2.2. ÍNDICE DE SALUD DE EQUIPOS

El índice de salud es una herramienta útil para cuantificar el deterioro irreversible a lo largo de la vida útil de un activo (Restrepo et al., 2020)[18], para nuestro caso el activo es el interruptor de potencia. Para tal efecto se analizan parámetros que afectan directa o indirectamente las características de funcionamiento y envejecimiento del interruptor con el propósito de estimar y dar un diagnóstico de la salud del activo. Usando criterios relacionados con la degradación a largo plazo, que acumulativamente resulta en el final de la vida del interruptor, esto incluye necesariamente la identificación de los activos que están cerca del final de su vida de operación y activos en alto riesgo de falla, que requieren gastos para su reemplazo, mantenimientos mayores (Overhaul) u otra decisión de gestión. (Deloitte, 2014)[19]

Para calcular el índice de salud de un activo existen muchos métodos y criterios que dependen exclusivamente de cada empresa que lo aplica y parte transcendental de este diagnóstico son los datos que se pudieran tener de cada activo en particular. (Durán et al., 2020)[20]. El procedimiento actual que se sigue para realizar un diagnóstico del índice de salud en los interruptores de potencia en la empresa ETEP, consiste en evaluar todas las

[18] Restrepo L., Calderón L., Cortes E. et al. (2020). Herramienta para estimación de índices de salud en activos de subestaciones de TyD Celsia. XXII congreso internacional de mantenimiento y gestión de activos.

[19] Deloite. (2014). Asset Health Indices A utility industry necessity. Canadian Electricity Association.

[20] Durán, O., Orellana, F., Perez, P., & Hidalgo, T. (2020). Incorporating an Asset Health Index into a Life Cycle Costing: A Proposition and Study Case. Mathematics, 8(10), 1787. MDPI AG.

variables que se monitorean en un interruptor durante sus mantenimientos preventivos, predictivos y/o correctivos, estableciéndose reglas y criterios mediante lógicas Fuzzy, considerando mayores pesos en aquellos variables que de estar fuera de rango o de producirse reiteradamente representan modos de falla irreparables en el interruptor de potencia, finalmente se establece un índice de salud del "01" a "04" para determinar el grado de degradación del interruptor y su posible tiempo de reemplazo. Donde el índice "01" denota un equipo con un diagnóstico de salud "muy pobre" requiriendo un tiempo de reemplazo que va desde una atención urgente y hasta máximo 2 años, el índice "02" representa un equipo con un diagnóstico de salud "pobre" requiriendo un tiempo de reemplazo de 2 a 5 años, el índice "03" representa un diagnóstico de salud "bueno" el cual no quiere decir que el equipo no tenga problemas, sino que son tolerables y por eso para esta escala se tiene un horizonte de reemplazo de 5 a 10 años y finalmente el índice de "04" que denota un diagnóstico de salud "muy bueno" y por ello tiene un horizonte de reemplazo mayor a los 10 años.

2.3. MACHINE LEARNING

Las industrias modernas o industrias 4.0 están valiéndose de la transformación digital, las herramientas de inteligencia artificial y del internet de las cosas para abordar sus fallas críticas (Angelopoulos et al., 2019)[21]. En

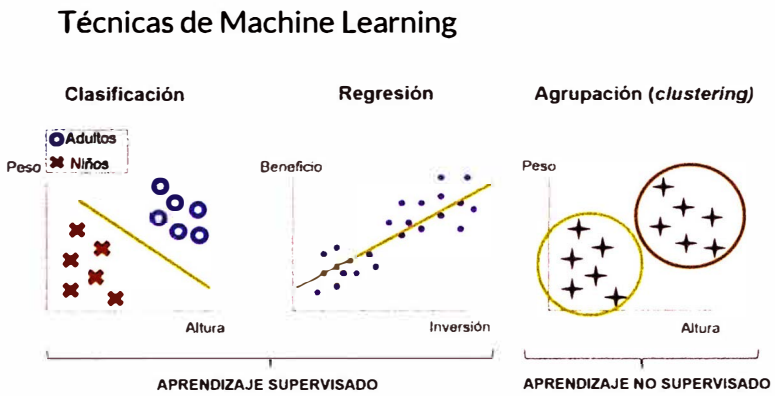
[21] Angelopoulos, A., Michailidis, E. T., Nomikos, N., Trakadas, P., Hatziefremidis, A., Voliotis, S., & Zahariadis, T. (2019). Tackling Faults in the Industry 4.0 Era A Survey of Machine-Learning Solutions and Key Aspects. Sensors, 20(1), 109. MDPI AG. Retrieved from

este contexto es sumamente importante entender entonces que es el machine learning y cómo funciona.

El machine learning o aprendiza de maquina es una técnica moderna de la inteligencia artificial que utiliza la computadora para detectar automáticamente patrones en los datos y usarlos para hacer predicciones o decisiones. (Hurwitz, J. & Kirsch, D., 2018)[22]. En el machine learning tenemos los siguientes tipos de aprendizajes: Aprendizaje Supervisado, Aprendizaje No Supervisado, Aprendizaje Reforzado y el aprendizaje profundo. Para el propósito de la presente tesis daremos el marco teórico de los dos primeros tipos de aprendizaje maquina descritos, dado que son los que encajan más con el propósito de la presente investigación.

En la figura 17 se muestran las diferentes técnicas usadas en machine learning para entrenar algoritmos de manera supervisada y No supervisada

Figura 17. *Tecnicas del Machine Learning*



Nota: Tomado de página web de Open Webinar. Disponible en: https://dc722jrlp2zu8.cloudfront.net/media/cache/1d/da/1ddab8ae80c940a9fe6449cf4bad6329.webp

[22] Hurwitz, J. & Kirsch, D (2018). Machine Learning For Dummies. Editorial John wiley & Sons, Inc. USA.pp75

2.3.1. APRENDIZAJES SUPERVISADOS Y NO SUPERVISADOS

a) Aprendizaje Supervisado. - Un aprendizaje supervisado es aquel que usa datos recopilados en un tiempo pasado de cualquier sistema del mundo real y que cada uno de esos registros se etiquetan de acuerdo con el resultado esperado, donde este nuevo conjunto de datos de entrada se denominan datos de entrenamiento del modelo. En concreto con esta técnica evaluamos la correlación entre un registro de datos y la salida objetivo donde para nuevos datos de entrada el algoritmo intenta dar el mejor resultado basado en esos datos aprendidos previamente. (Jahnke, 2015)[23]

b) Aprendizaje No Supervisado. - El aprendizaje no supervisado es aquel que tiene por objetivo agrupar o clasificar los datos de entrada buscando o descubriendo características similares en ellos. Menciona también que con esta técnica podemos descubrir la distribución de los datos de entrada en un proceso llamado estimación de la densidad. (Jahnke, 2015)[21]

2.3.2. TÉCNICAS DE REGRESIÓN Y CLASIFICACIÓN

Con las técnicas de regresión lo que buscamos es predecir un valor continúo utilizando datos de entrenamiento, es decir tener un objetivo como variable dependiente y los atributos de ese objetivo como variables independientes; en cambio, con las técnicas de clasificación lo que buscamos es predecir un

[23] Jahnke, P. (2015). Machine Learning Approaches for Failure Type Detection and Predictive Maintenance. Master's Thesis, Technische Universität Darmstadt, Darmstadt, Germany.pp83.

valor categórico ósea una clase o escenario, utilizando datos de entrenamiento o atributos. (Bishop, 2006)[24]. A continuación, daremos a conocer los principales métodos que se usan en cada una es estas técnicas de modelado.

a) Modelo de Regresión Lineal. – también llamados Bivariados es el modelo de regresión más simple, donde tenemos una correspondencia única entre el atributo y su salida, por ejemplo, podríamos afirmar de una relación única entre el grado de educación de una persona y el sueldo que llega a conseguir:

$$X(educación) - - \rightarrow y\,(Sueldo)$$

El modelo de regresión es de la forma:

$$y = \alpha_0 + \alpha_1 x + \varepsilon \quad (1)$$

Donde la pendiente de la línea esta dado por α_1 y la intercepción esta dado por α_0 y ε es el error del modelo

Dada la muestra de valores de x e y, las estimaciones b_0 de β_0 y b_1 de β_1 se obtienen utilizando los mínimos cuadrados u otro método.

La estimación resultante del modelo es:

$$y' = \alpha_0 + \alpha_1 x + \varepsilon \quad (2)$$

[24] Bishop, M.C. (2006). Pattern Recognition and Machine Learning. Editorial Springer. USA.pp758

Donde el símbolo y' se denomina: "y prima" y se refiere a los valores predichos de la variable dependiente y que están relacionados con los valores x, dado el modelo lineal.

b) Modelo de Regresión Múltiple. – también llamado multivariado, donde intervienen varios atributos para explicar una salida por ejemplo que el sueldo de una persona dependa no solamente de sus educaciones, sino también este asociado a su edad y experiencia relacionada:

$$X1(educación), X2(edad), X3(experiencia) - - \rightarrow y\ (Sueldo)$$

y existirá un coeficiente de regresión para cada variable independiente, el modelo de regresión en su forma general será de la forma:

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \cdots \alpha_q x_q + \varepsilon \quad (3)$$

El modelo tiene varias variables independientes "x" que explican la variable dependiente y. Donde α_i determina la contribución de la variable independiente x_i. Igual que el modelo lineal ε es el error del modelo.

$$y' = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \cdots \alpha_q x_q + \varepsilon \quad (4)$$

c) Modelo de Regresión Logística. – este modelo busca una regresión de los atributos "x" a una forma de onda logarítmica, teniendo la siguiente ecuación:

$$\ln\left[\frac{\rho}{1-\rho}\right] = \alpha + \beta x + \varepsilon \quad (5)$$

Donde:

- ρ: $es\ la\ probabilidad\ de\ que\ el\ evento$ "y" ocurra, $\rho\ (y = 1)$

- $\frac{\rho}{1-\rho}$: *es la proporción de probabilidad*
- $\ln\left[\frac{\rho}{1-\rho}\right]$: *es el logaritmo de la proporción de probabilidades*
- *La distribución logistica limita las probabilidades estimadas entre* 0 *y* 1
- *La probabilidad estimada es:* $\rho = \left[\frac{1}{1+\exp(-\alpha-\beta x)}\right]$
- *a medida que* $\alpha + \beta x$ *se vuelve grande,* ρ *se acerca a* 1
- *a medida que* $\alpha + \beta x$ *se vuelve pequeño,* ρ *se acerca a* 0

d) Modelo de Regresión Polinomial. – si reemplazamos un modelo lineal estándar con una función polinomial conseguimos un modelo de grado "n" suficientemente grande que nos permite producir una curva extremadamente no lineal.

Para lograrlo bastara con crear nuevas variables $x_1 = x$, $x_2 = x^2$, $x_3 = x^3, \ldots x_n = x^n$ y luego lo trataremos como regresión lineal múltiple.

$$f'(x_0) = \beta'_0 + \beta'_1 x_0 + \beta'_2 x_0^2 + \beta'_3 x_0^3 + \beta'_4 x_0^4 + \cdots \beta'_n x_0^n \quad (6)$$

Es inusual usar un grado "n" mayor a 3 ó 4 porque para valores grandes de "n", la curva polinomial se vuelve muy flexible y puede adoptar formas extrañas.

e) Modelo de Regresión con Soporte Vectorial. – conocido en inglés como el "Support Vector Regression", usa una función de perdida insensitiva "ε" dado un conjunto de datos $\{x_1, x_2, x_3, \ldots x_n\}$ con valores objetivo $\{u_1, u_2, u_3, \ldots u_n\}$, la función de error para la regresión del vector de soporte se puede escribir como:

$$Min\frac{1}{2}\|w\|^2 + C\sum_{i=1}(\varepsilon_i + \varepsilon_i^*) \quad (7)$$

$$donde \begin{cases} u_i - w^t x_i - b \leq \epsilon + \varepsilon_i \\ w^t x_i + b - u_i \leq \epsilon + \varepsilon_i^* \\ \varepsilon_i \geq 0, \varepsilon_i^* \geq 0 \end{cases}$$

f) Modelo de Regresión con Arboles de Decisión. – En este modelo dividimos el espacio del predictor en "J" regiones distintas que no se superponen, luego hacemos la misma predicción para todas las observaciones en la misma región; utilizando la media de las respuestas para todas las observaciones de entrenamiento que se encuentran en la región, estas regiones pueden tener cualquier forma. El modelo busca encontrar las regiones $R_1, R_2, \dots R_j$ que minimicen el RSS. (Bishop, 2006)[25]

$$RSS = \sum_{j=1,\, i \in R_j} \sum \left(y_i - y'_{R_j}\right)^2 \quad (8)$$

g) Modelo de Clasificación por Vecinos Cercanos. – Conocido con el termino inglés: K-nearest neighbor, este modelo consiste en clasificar un valor de prueba con la clase de la mayoría de los valores "k" más cercanos al valor de consulta. (Bishop, M.C., 2006)[23]

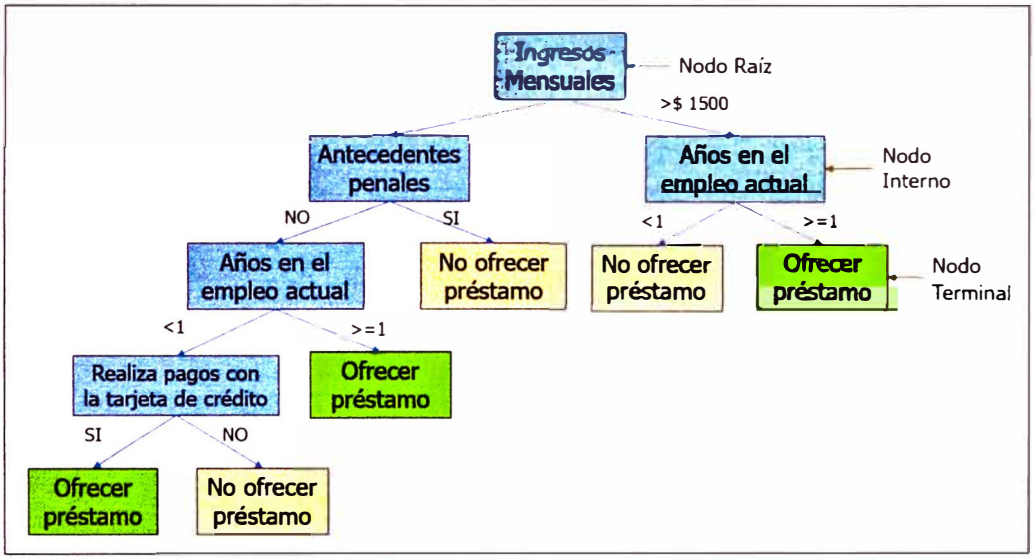
- $KNN = k\ vecinos\ más\ cercanos$
- $Para\ k = 1 (1NN), asignamos\ el\ valor\ de\ prueba\ al\ valor\ mas\ cercano\ en\ el\ conjunto\ de\ entrenamiento$
- $1NN\ no\ es$ tan $robusto, ya\ que\ un\ valor\ puede\ ser\ etiquetado\ de\ forma\ erronea\ o\ puede\ ser\ un\ documento\ atipico$
- $Para\ k > 1, asignamos\ el\ documento\ a\ la\ clase\ de\ la\ mayoria\ de\ los\ k\ valores\ mas\ cercanos\ en\ el\ conjunto\ de\ entrenamiento$

[25] Bishop, M.C. (2006). Pattern Recognition and Machine Learning. Editorial Springer. USA.pp758

h) Modelo de Clasificación por Arboles de decisión. – El aprendizaje mediante el árbol de decisión es una de la técnica de clasificación más utilizadas, su precisión es competitiva con otros métodos, y es muy eficiente, el modelo C.4.5 de Ross Quinlan es quizás el sistema más conocido. (Jahnke, 2015) [26]

En la figura 18 se muestra un modelo típico de un esquema de árbol de decisión.

Figura 18.*Esquema de un Modelo de Arbol de Desición*



Nota: Tomada de página web grupo Davia. Disponible en: https://www.grupodabia.com/post/2020-05-19-arbol-de-decision/arbol-decision.png

i) Modelo de Clasificación Lineal. – el objetivo de este modelo es encontrar una función lineal para separar las clases existentes

$$g(x) = w^T x + b \quad (9)$$

[26] Jahnke, P. (2015). Machine Learning Approaches for Failure Type Detection and Predictive Maintenance. Master's Thesis, Technische Universität Darmstadt, Darmstadt, Germany.pp83

Donde:

$$g(x)\begin{cases} w^T x + b > 0 \\ w^T x + b = 0 \\ w^T x + b < 0 \end{cases}$$

j) Modelo de Clasificación por Soporte Vectorial. – también llamados SMV, existen dos tipos los lineales y no lineales. El SMV lineal usa una función de discriminante lineal con el margen máximo, el margen lo definimos como el ancho que podría aumentar el límite antes de llegar a un punto de datos (Bishop, 2006)[27]. Tiene la siguiente formulación:

$$Minimo\ \frac{1}{2}\|w\|^2 \quad (10)$$

Tal que:

Para $y_i = +1,\ \ w^T x_i + b \geq 1$

Para $y_i = -1,\ \ w^T x_i + b \leq -1$

La función discriminante lineal se basa en un producto escalar entre el punto de prueba "x" y los vectores de soporte "x_i" teniedo la siguiente forma:

$$g(x) = W^T x + b = \sum_{i \in SV} \alpha_i x_i^T x + b \qquad (11)$$

Los SMV no lineales en vez de usar una función lineal para separar los datos de entrada usan un plano, donde "x" va a estar dado por

[27] Bishop, M.C. (2006). Pattern Recognition and Machine Learning. Editorial Springer. USA.pp758

una nueva función "Ø" que depende de esta variable y denominaremos $\emptyset(x)$, entonces la discriminante se convierte en:

$$g(x) = W^T \emptyset(x) + b = \sum_{i \in SV} \alpha_i \emptyset(x_i^T) \emptyset(x) + b \quad (12)$$

No es necesario conocer este mapeo explícitamente, porque solo usamos el producto escalar de los vectores de características tanto en el entrenamiento como en la prueba, a estas funciones también se le conoce como funciones de kernel (producto escalar de dos vectores de características en un espacio de características ampliado)

k) Modelo de Clasificación por Bosques aleatorios. - Durante la fase de aprendizaje un bosque aleatorio trabaja mediante el diseño de un conjunto de árboles de decisión, y el resultado es la clase elegida de la mayoría de los árboles de decisión. Se crea entonces un bosque aleatorio con un conjunto aleatorio de características y un conjunto aleatorio de datos de entrenamiento, (Jahnke, 2015)[28]

A continuación, describiremos las posibles ventajas de un bosque aleatorio:

- Se puede reducir el tiempo de aprendizaje porque este modelo aprende en paralelo de todos los árboles de decisiones.
- Se puede usar para el análisis de datos grande dado que su tiempo de evaluación puede ser más corto respecto al de un solo árbol.

[28] Jahnke, P. (2015). Machine Learning Approaches for Failure Type Detection and Predictive Maintenance. Master's Thesis, Technische Universität Darmstadt, Darmstadt, Germany.pp83

l) Modelo de Clasificación por Redes Neuronales. – este modelo utiliza un concepto denominado neuronas artificiales el cual está inspirado en las neuronas biológicas, donde cada neurona artificial puede conectarse con otras neuronas, teniendo capas de entrada, capas ocultas y capas de salida. Cada neurona oculta o de salida tiene conexiones de entrada ponderadas de cada una de las unidades de la capa anterior. La unidad realiza una suma ponderada de sus entradas y resta su valor umbral para dar su nivel de activación, El nivel de activación pasa a través de una función de activación sigmoidea para determinar su salida. (Jahnke, 2015)[29].Tenemos entonces lo siguientes características en todas la rede neuronales:

- La capa de entrada:
  - ✓ Introduce valores de entrada en la red.
  - ✓ Sin función de activación u otro procesamiento.
- Las capas ocultas:
  - ✓ Realiza clasificación de características
  - ✓ Dos capas ocultas son suficientes para solucionar cualquier problema
- La capa de salida:
  - ✓ Funcionan como las capas ocultas
  - ✓ Las salidas se transmiten al mundo fuera de la red neuronal.
- Funciones de Activación:

[29] Jahnke, P. (2015). Machine Learning Approaches for Failure Type Detection and Predictive Maintenance. Master's Thesis, Technische Universität Darmstadt, Darmstadt, Germany.pp83

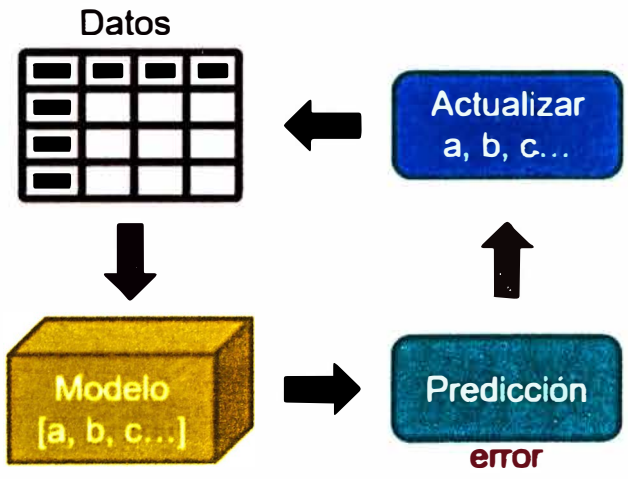
- ✓ Transforma la entrada de la neurona en salida.
- ✓ Simple y fácil de calcular

- La función de umbral de limitación:
  - ✓ Corresponde al paradigma biológico: ¿dispara o no?
- Funciones sigmoideas:
  - ✓ La función logística
  - ✓ La tangente hiperbólica (simétrica)

Finalmente, cada modelo de machine Learnig debe cumplir un ciclo de mejora continua hasta minimizar el error del modelo en la predicción esperada (Domingos, 2012)[30]. La forma de medir estos errores lo revisaremos en el siguiente subcapítulo.

En la figura 19 se muestra el ciclo de mejora continua de un modelo predictivo de machine learning.

Figura 19. *Ciclo de un modelo de Machine Learning*



Nota: Tomado de página web de Open Webinar. Disponible en: https://dc722jrlp2zu8.cloudfront.net/media/cache/49/01/4901b4e3140f9bd6eb6535a3f1bafcbf.webp

[30] Domingos, P. (2012). A few useful things to know about machine learning. Commun. ACM 55, 10 (October 2012), 78–87

2.3.3. PRINCIPALES MÉTRICAS USADAS EN MODELOS PREDICTIVOS

Se necesita uno o varios parámetros de referencia que sirva de control de calidad al modelo predictivo. Hay muchas formas habituales de comprobar esta calidad a continuación detallaremos las más utilizadas: (Bishop, 2006)[31]

a) Error absoluto medio (MAE). – Es una métrica de las diferencias entre valores predichos por un modelo o un estimador y los valores observados. Esta métrica es usada para modelos de regresión numérica.

$$MAE = \frac{(\sum_{i=1}^{n} |\varepsilon_i|)}{n} \qquad (13)$$

b) Error cuadrático medio (RMSE). - es una métrica de uso frecuente de las diferencias entre valores predichos por un modelo o un estimador y los valores observados. Igual al MAE esta métrica es usada para modelos de regresión numérica

$$RMSE = \sqrt{\left(\sum_{i=1}^{n} (\varepsilon_i)^2\right)/n} \qquad (14)$$

c) Precisión. - es una métrica que evalúa la relación entre las predicciones positivas correctas respecto al total de casos predichos positivos. Esta métrica es muy usada en modelos de clasificación.

$$\frac{tp}{tp + fp} \qquad (15)$$

[31] Bishop, M.C. (2006). Pattern Recognition and Machine Learning. Editorial Springer. USA.pp758

Donde:

- tp: $verdaderos\ positivos$
- fp: $falsos\ positivos$

d) Sensibilidad (recall). - es una métrica que evalúa la relación entre las predicciones positivas correctas respecto al total de casos positivos reales. Esta métrica es muy usada en modelos de clasificación.

$$\frac{tp}{tp + fn} \quad (16)$$

Donde:

- tp: $verdaderos\ positivos$
- fn: $falsos\ negativos$

e) Exactitud (accuracy). - es una métrica que evalúa la relación entre las predicciones positivas y negativas correctas respecto al total casos positivos y negativos reales. Esta métrica es muy usada en modelos de clasificación.

$$\frac{tp + tn}{tp + tn + fp + fn} \quad (17)$$

Donde:

- tp: $verdaderos\ positivos$
- tn: $verdaderos\ negativos$
- fn: $falsos\ negativos$
- fp: $falsos\ positivos$

f) F1-score .- es una métrica que combina la precisión y el recall (sensibilidad) de un modelo en una sola medida. Esta métrica es muy usada en modelos de clasificación con datos desbalanceados dado que combina 2 métricas importantes:

$$F1 - Score = \frac{2\ x\ precisión}{precisión\ + recall} \quad (18)$$

g) Curva ROC. - Curva característica de funcionamiento del receptor, esta curva grafica la tasa positiva verdadera (sensibilidad) frente a la tasa de falsos positivos: $fp/(fp + tn)$, donde una curva por debajo de la diagonal significa alta aleatoriedad en los datos y una curva optima estaría en la esquina superior izquierda. Por las características mismas de esta métrica solo es válido para problema de "clasificación binaria"

h) Área bajo la curva (AUC). - Es el área bajo la curva ROC, refleja el rendimiento para diferentes umbrales posibles de probabilidad. Igual que el ROC, esta métrica es válida para problema de "clasificación binaria"

2.4. MARCO CONCEPTUAL.

- Interruptores. - Para la presente tesis cuando hablemos de Interruptores o interruptores de potencia nos referiremos exclusivamente aquellos con un nivel de tensión mayor o igual a 60kV
- Variables. – Cualquier parámetro medible en los interruptores de potencia que pueden ser del tipo categórico o numérico.
- Variables Predictoras. - Variables seleccionadas de un interruptor de potencia para el análisis de un modelo predictivo.
- Modelo. – Algoritmo matemático resultado de un análisis exploratorio de variables predictoras y que tiene una cierta precisión, sensibilidad y exactitud respecto a su objetivo.
- Variable objetivo. - Para nuestro caso el Índice de salud, el cual se busca correlacionar con las variables predictoras
- Valor atípico (outliers). - Datos anómalos registrados en un interruptor que alteran las variables predictoras y que deben ser identificados y procesadas para evitar una distorsión en el modelo predictivo.
- Celda o Bahía. - Conjunto de equipos de patio de una Subestacion eléctrica, predispuesto bajo un diseño de ingeniería cumpliendo requisitos técnicos y de seguridad con el objetivo de transmitir de la energía eléctrica
- Salud. - Diagnóstico de un interruptor que nos indicará su estado de deterioro.

- Envejecimiento.- Edad nominal del interruptor.
- Efectividad. - Es el porcentaje en el que se logra acertar con el correcto diagnóstico del índice de salud.
- Robustes.- Capacidad de un modelo para mantener un buen rendimiento y generalización incluso cuando se enfrenta a datos atípicos, ruidosos o perturbaciones en los datos de entrada
- Aprendizaje.- Rendimiento de un modelo (por ejemplo, en precisión) a medida que se aumenta la cantidad de datos de entrenamiento, lo que puede indicar si el modelo se beneficia de más datos o si pudiera estar sufriendo de sobreajuste.
- Sobreajuste .- Situación en la que un modelo de machine learning se ajusta demasiado a los datos de entrenamiento, capturando ruido y detalles irrelevantes, lo que resulta en un rendimiento deficiente en datos no vistos, como los de prueba.
- Subajuste.- Situación en la que un modelo de machine learning es demasiado simple para capturar las relaciones en los datos de entrenamiento, lo que resulta en un rendimiento deficiente tanto en los datos de entrenamiento como en los de prueba.

CAPÍTULO III. ANÁLISIS DE DATOS Y MODELADO DE ALGORITMOS PREDICTIVOS

3.1. RECOLECCIÓN Y LIMPIEZA DE DATOS

La recolección de datos se detalla en el acápite 1.10.5 "Técnicas de recolección y procesamiento de datos" por lo que en este apartado trataremos como se efectuó el proceso de limpieza de datos lo cuales comprende básicamente el tratamiento de datos nulos y el tratamiento de datos atípicos o también conocidos como datos outliers (ver definición en marco conceptual).

En ese sentido primero identificaremos todas las variables y las clasificaremos por su tipo: Categóricas y Numéricas, dentro de las numérica tenemos también la subclasificación entre variables discretas y las continuas

Variables categóricas:

- Tipo fabricante

Variable Numéricas:

- Discretas: Nivel de kV
- Continuas: Humedad en gas SF6, Pureza de gas SF6, Dióxido de azufre en gas SF6, Resistencia de contactos.

Para la variable categórica (tipo de fabricante), no se tuvo problemas con datos nulos o datos atípicos por lo que todos los interruptores contaban con este dato y más bien lo que se hizo fue convertir esta variable categórica a una variable numérica discreta, asignando un valor correlativo a cada tipo de fabricante lo cual se puede apreciar en el análisis Univariante y Bivariante.

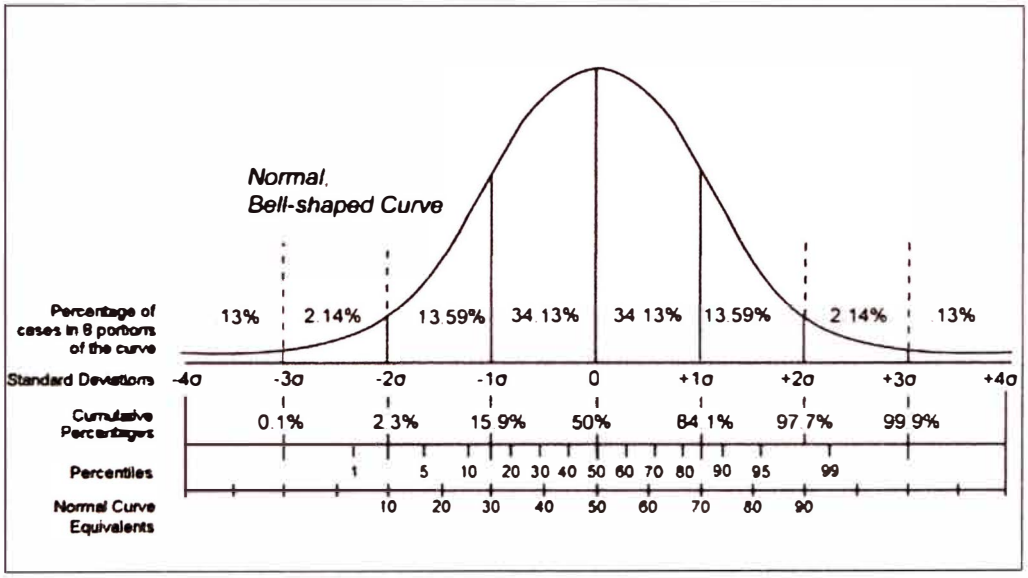
Para la variable discreta (Nivel de kV) no se tuvo problemas con datos nulos, pero si con datos atípicos, dado que algunos interruptores contaban con datos fuera del estándar en el nivel de kV por ejemplo existían interruptores de 66kV que fueron agrupados junto con los de 60kV que eran la mayoría y también existían interruptores de 132kV que fueron agrupados con los interruptores de 138kV que eran la mayoría, el nivel de 220kV también se encontró algunos interruptores de 210kV que fueron agrupados en esta misma categoría. En el nivel de 500kV no se encontró valore atípicos para esta variable.

Para las variables numérica tipo continua si se encontró ambos problemas: datos nulos y datos atípicos, la forma de tratar estos problemas es muy variable y depende del conocimiento técnico y experiencia de cada experto. Una buena técnica es realizar un análisis de percentiles estadísticos y cubrir por ejemplo los datos nulos con un percentil estadístico, usar la moda de los datos o la media aritmética o la media armónica u otro valor estadístico. En la presente investigación se optó por usar el percentil "50" para reemplazar todos los datos nulos de las variables numéricas tipo continuas. Para tratar los datos atípicos que generalmente son valores de mala calidad por ser errores durante el registro de datos al sistema, ya sea por tener valores demasiado altos o bajos y que pueden introducir ruido durante el entrenamiento de modelos predictivos. Se optó por filtrar estos valores realizando un análisis de percentiles estadísticos y desviaciones estándar, de tal manera que aquellos valores por encima del percentil "50" más 3 desviaciones estándar (Percentil "99.9") se consideraron como datos atípicos y se reemplazó por la media (percentil "50") y de igual manera aquellos valores por debajo del percentil "50" menos 3 desviaciones

estándar (Percentil "0.1") se consideraron como datos atípicos y se reemplazó por la media (percentil "50").

Para mejor entendimiento y esquematización del análisis realizado para el tratamiento de datos atípicos y nulos mediante herramientas estadísticas se muestra el grafico de la figura 20, donde se puede observar los diferentes percentiles estadísticos en una curva de distribución, las desviaciones estándar que explican la variabilidad de los datos y los diferentes umbrales que cada experto puede usar para determinar los potenciales datos atípicos (outliers).

Figura 20. *Analisis de percentiles estadisticos para el tratamiento de datos atipicos y nulos*



Nota: Tomada de psicologiaymente.com Disponible en: https://psicologiaymente.com/miscelanea/como-calcular-percentiles

3.2. ANÁLISIS UNIVARIANTE DE DATOS

El objetivo de este análisis es caracterizar a las variables por sus parámetros estadísticos principales: cantidad de datos, el valor promedio, la desviación estándar, el valor mínimo, el valor máximo, los percentiles y cuartiles estadísticos de cada categoría).

Como primer paso para el análisis y entendimiento de nuestros datos, debemos tomar en cuenta que la población estudiada en la presente investigación es de 530 interruptores y que los parámetros en común y disponibles que tenemos de estos interruptores son: "Nivel voltaje" , "Tipo fabricante", "Humedad en gas SF6", "Pureza de gas SF6", "Concentración de dióxido de azufre en gas SF6" , "Resistencias de contactos del interruptor" y "Edad del interruptor", los cuales serán parte del presente estudio y conoceremos en adelante como nuestras variables predictoras. En la tabla 2, se muestra un cuadro resumen del análisis estadístico de variables usadas.

Tabla 2. Cuadro resumen de análisis estadístico de variables

	count	mean	std	min	0.1%	1%	25%	50%	75%	99%	99.9%	max
KV	530.00	190.88	104.49	60.00	60.00	60.00	138.00	220.00	220.00	500.00	500.00	500.00
FAB	530.00	2.18	1.27	1.00	1.00	1.00	1.00	2.00	3.00	6.71	7.00	7.00
HUM	530.00	70.33	104.03	0.00	1.06	10.00	25.00	44.00	78.75	653.90	939.48	946.75
PUR	530.00	99.24	1.24	86.36	88.23	95.79	99.00	99.90	99.90	100.00	100.00	100.00
SO2	530.00	1.34	21.95	0.00	0.00	0.00	0.00	0.00	0.00	9.71	271.53	499.00
RCC	530.00	46.30	49.13	0.00	0.00	18.22	29.70	38.00	46.00	243.38	610.38	714.80
ENV	530.00	16.90	8.64	1.00	1.00	2.00	11.00	14.00	23.00	42.42	43.00	43.00

Nota: Elaboración propia

Donde:

- KV : Nivel de Voltaje en kV
- FAB : Tipo de Fabricante
- HUM : Humedad en gas SF6 en ppmv

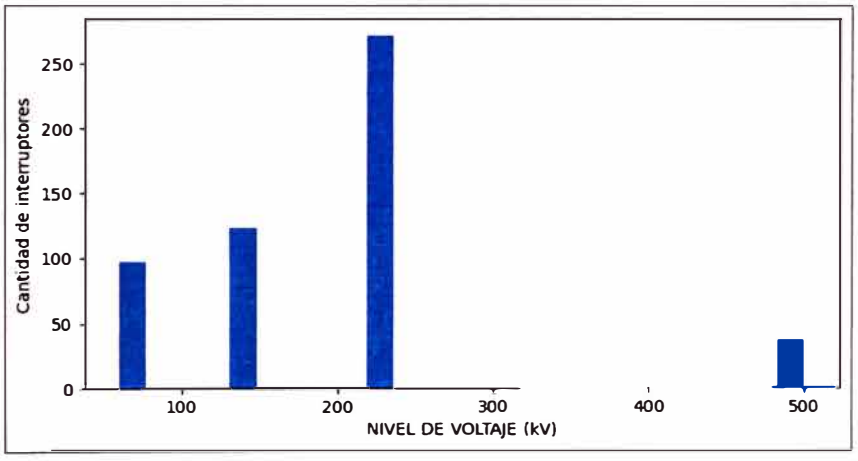
- PUR : Pureza en gas SF6 en %
- SO2 : Dióxido de azufre en gas SF6 en ppmv
- RCC : Resistencia de Contacto en μΩ
- ENV : Edad del Interruptor en años

Para un mejor análisis y comprensión de nuestros datos generaremos gráficos de histogramas y diagrama de Boxplot (diagrama estadístico de cajas y bigotes) por cada variable predictora definida. Para este fin nos ayudaremos de las herramientas de Python usando la biblioteca "Matplotlib".

En las siguientes paginas mostraremos cada una de las gráficas estadísticas obtenidas y realizaremos comentarios u observaciones pertinentes producto del análisis estadístico univariante.

En la figura 21 se muestra un Histograma de la cantidad de interruptores vs Nivel de voltaje.

Figura 21. Histograma de Variable: Nivel de voltaje

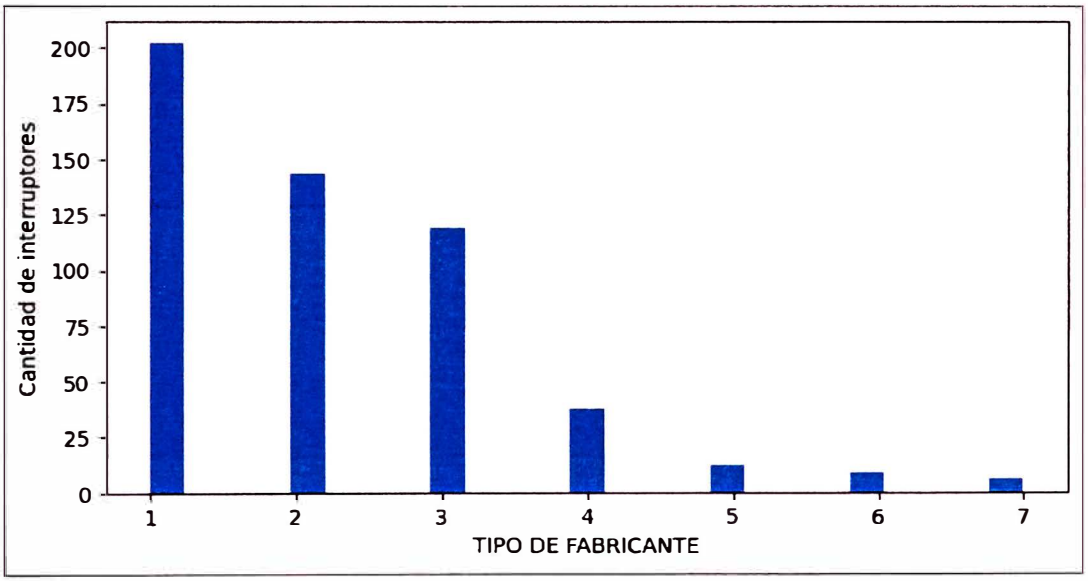


Nota: Elaboración propia

Del histograma para la variable "Nivel de voltaje" podemos observar que la mayoría de los interruptores de nuestra población estudiada están concentrados en el nivel de 220kV y el de menor concentración son los interruptores de 500kV.

En la figura 22 se muestra un Histograma de la Variable: Tipo de Fabricante.

Figura 22. Histograma de variable: Tipo de fabricante

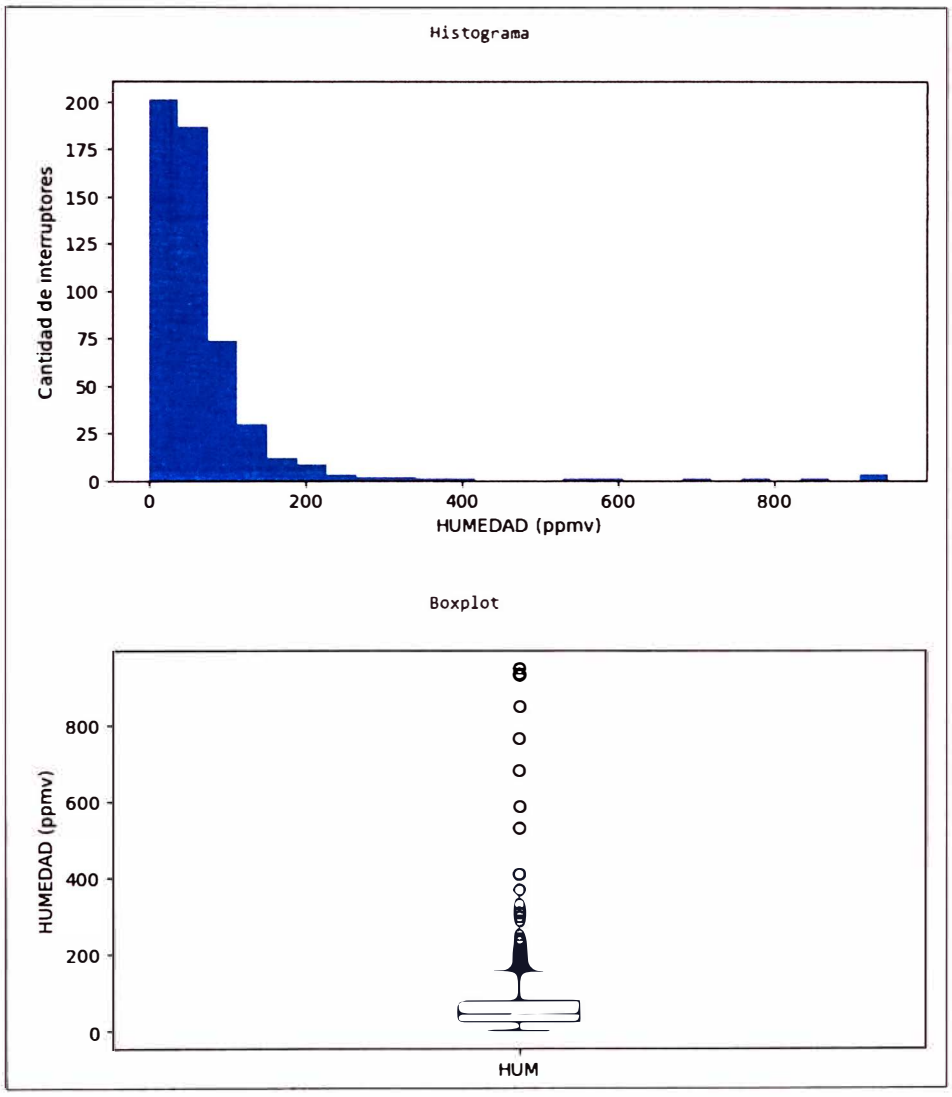


Nota: Elaboración propia

Del Histograma para la variable "Tipo de fabricante" podemos observar que la mayor cantidad de interruptores se concentran en fabricante identificado con el código "1" y el de menor cantidad el fabricante identificado con el código "7".

En la figura 23 se muestra el Histograma y Boxplot de la Variable: Humedad de gas SF6.

Figura 23. Histograma y Boxplot de Variable: Humedad



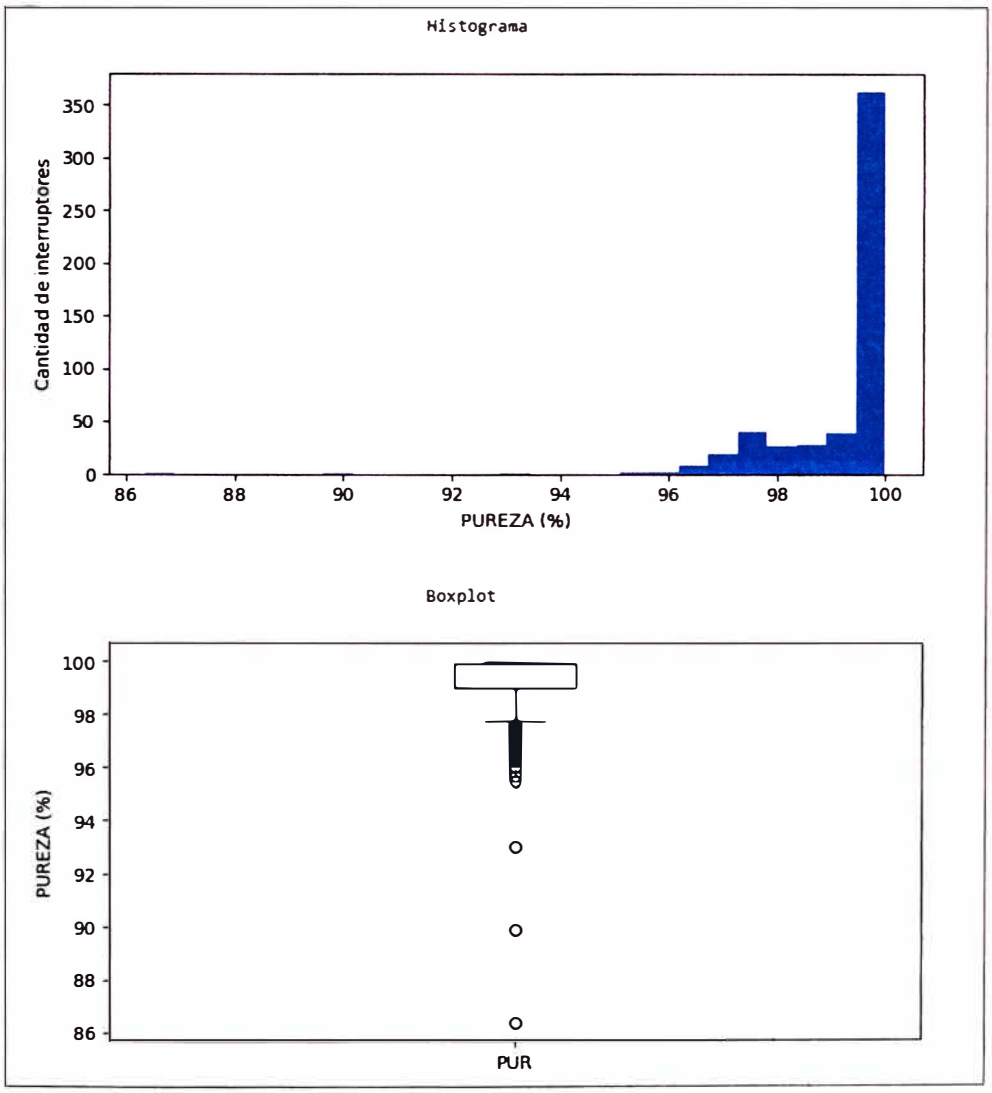
Nota: Elaboración propia

Del histograma para la variable "humedad de gas SF6" podemos observar que la mayoría de los interruptores se concentran en un valor cercano a los 200ppmv y del Boxplot estadístico podemos observar que el 50% de toda la población estudiada

(línea verde del cuadro) se concentran en un valor cercano a los 100ppmv (partes por millón de volumen).

En la figura 24 se muestra el Histograma y Boxplot de la Variable: Pureza de gas SF6.

Figura 24. Histograma y Boxplot de Variable: Pureza de SF6



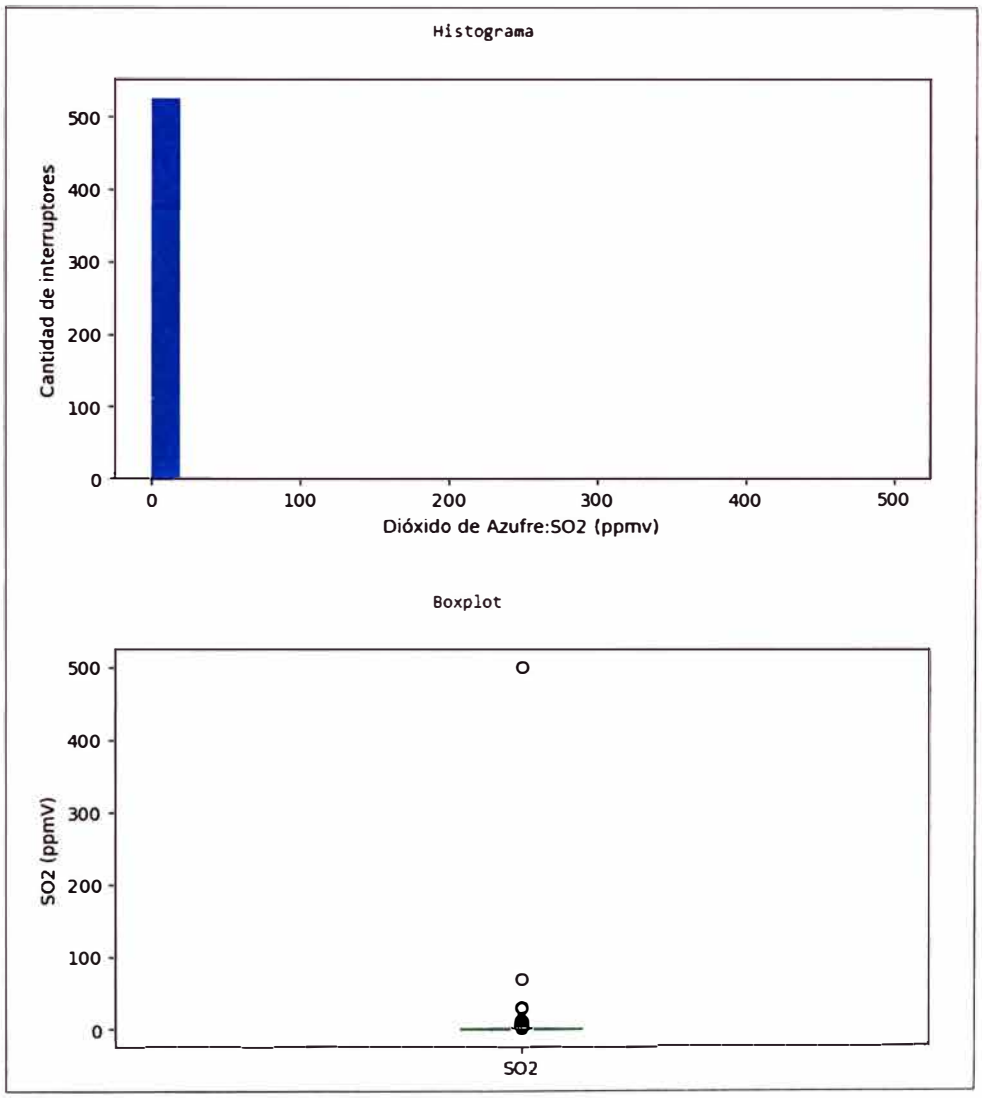
Nota: Elaboración propia

Del histograma para la variable "pureza de gas SF6" podemos observar que la mayoría de la población estudiada se concentran en un valor superior al 96% y del

Boxplot estadístico podemos observar que el 75% de toda la población estudiada (borde inferior del cuadro) se concentran en un valor cercano al 99% (porcentaje de pureza del gas).

En la figura 25 se muestra el Histograma y Boxplot de la Variable: Presencia de SO2 en gas SF6.

Figura 25. Histograma y Boxplot de Variable: SO2 en SF6

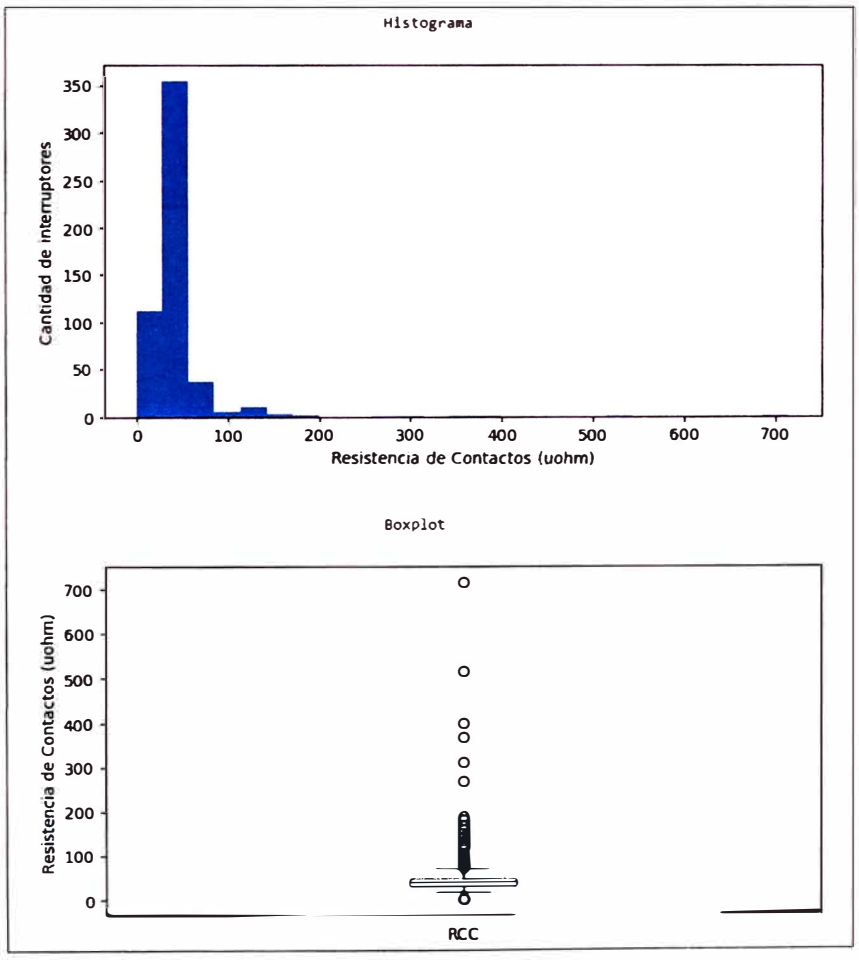


Nota: Elaboración propia

Del histograma para la variable "pureza de gas SF6" podemos observar que la mayoría de la población estudiada se concentran en un valor cercano a 0 ppmv (partes por millón de volumen) y del Boxplot estadístico podemos observar que si bien es cierto la mayoría de la población tiene valores cercanos a 0 ppmv, también existen algunos interruptores con valores outliers que incluso llegan a tener valores entre 100 a 500 ppmv

En la figura 26 se muestra el Histograma y Boxplot de la Variable: Resistencia de Contactos.

Figura 26. Histograma y Boxplot de Variable: Resistencia de Contactos

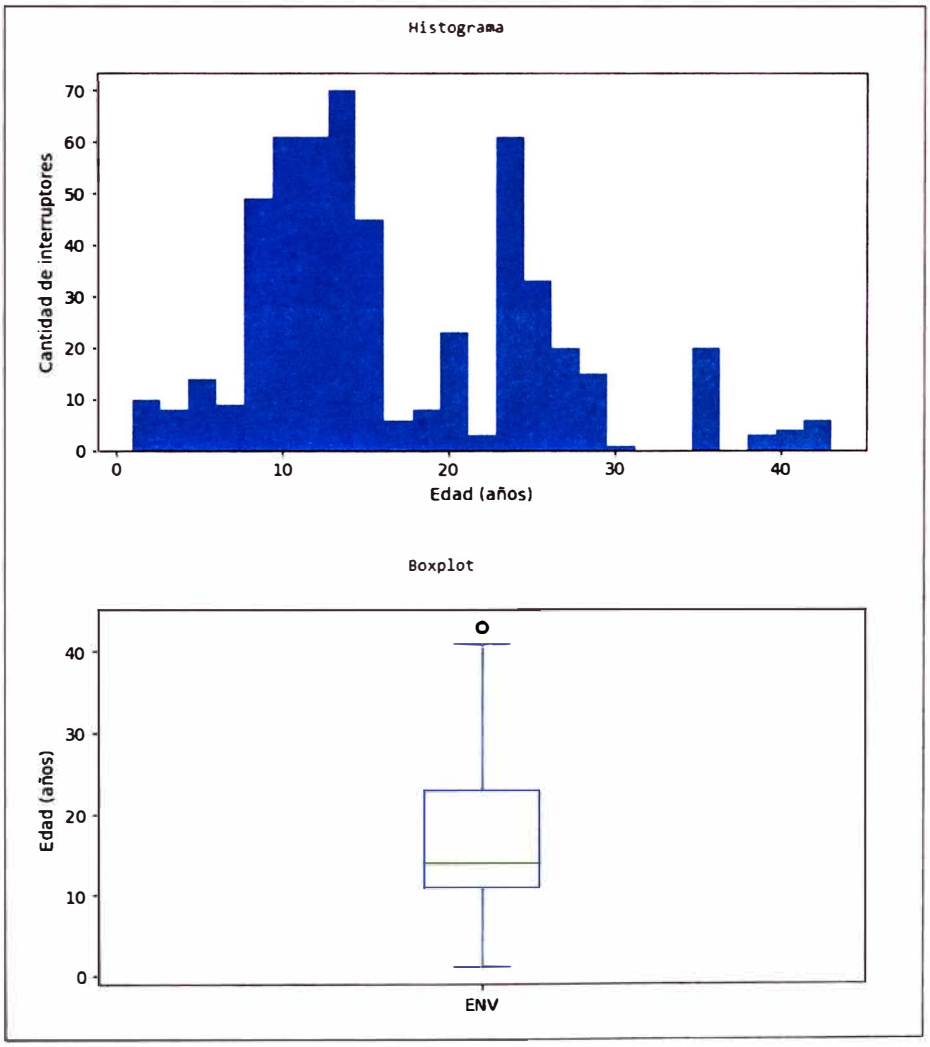


Nota: Elaboración propia

Del histograma para la variable "Resistencia de Contactos" podemos observar que la mayoría de la población estudiada se concentran en un valor próximo a los 100 μΩ y del Boxplot estadístico podemos observar que el 50% de toda la población estudiada se concentran en valores cercanos a los 50 μΩ, existiendo valores outliers que van desde los 200 a 700 μΩ.

En la figura 27 se muestra el Histograma y Boxplot de la Variable: Edad del interruptor.

Figura 27. Histograma y Boxplot de Variable: Edad



Nota: Elaboración propia

Del histograma para la variable "Edad del interruptor en años" podemos observar que no hay una mayoría absoluta dado que la población de interruptores se encuentra distribuidos en diferentes grupos de edades, sin embargo, en el Boxplot estadístico podemos verificar que el 75% de toda la población estudiada (borde superior del cuadro) se concentran en edades de hasta 25 años aproximadamente y el 50% de los interruptores analizados (línea central verde) se concentra en edades de 15 años aproximadamente.

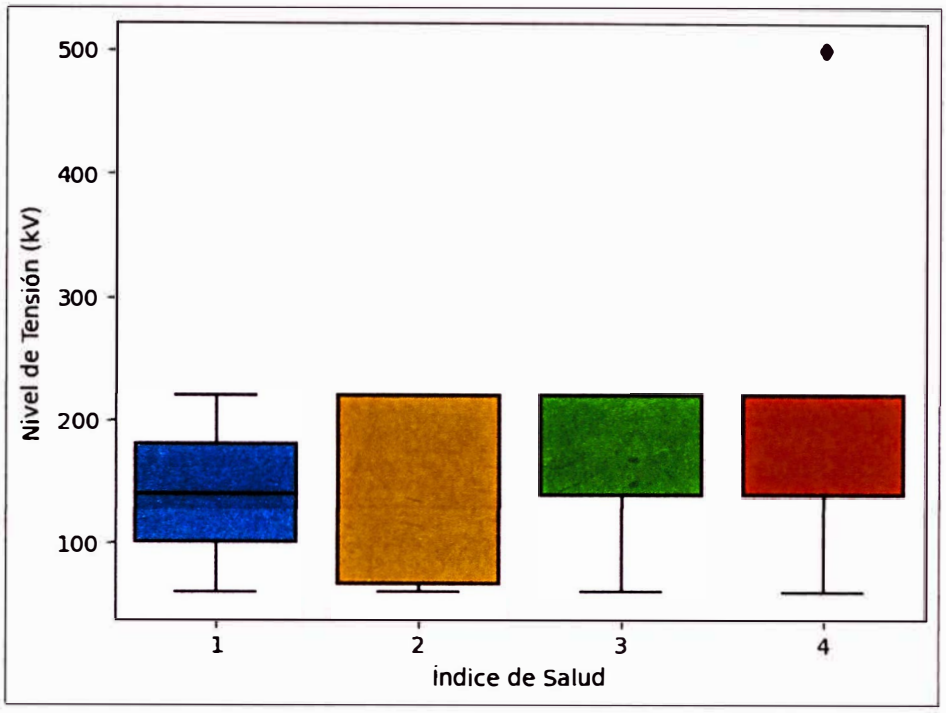
En síntesis, todos estos gráficos realizados de histogramas y boxplot nos muestra visualmente cómo se distribuyen y concentran estadísticamente los datos de cada variable con el fin de ayudar en el entendimiento del comportamiento típico de cada variable y el tratamiento de sus valores atípicos (outlier)

3.3. ANÁLISIS BIVARIANTE DE DATOS

El objetivo de este análisis Bivariante es comparar cada una de las variables predictoras con la variable objetivo con el fin de encontrar patrones de tendencia o correspondencia entre ciertos valores de las variables predictoras y cada una de las categorías de la variable objetivo.

En ese sentido analizaremos las variables seleccionadas en diagramas estadísticos Boxplot, comparándolas con cada uno de los Índices de Salud (definidos y detallados en el capítulo II, sección 2.2 Índice de salud de equipos). Generaremos los gráficos usando el programa Python. En la figura 28 se muestra el análisis bivariante entre el nivel de voltaje y el estado de salud de los interruptores de potencia.

Figura 28. Analisis Bivariante: "Nivel de Voltaje" vs "Índice de Salud"

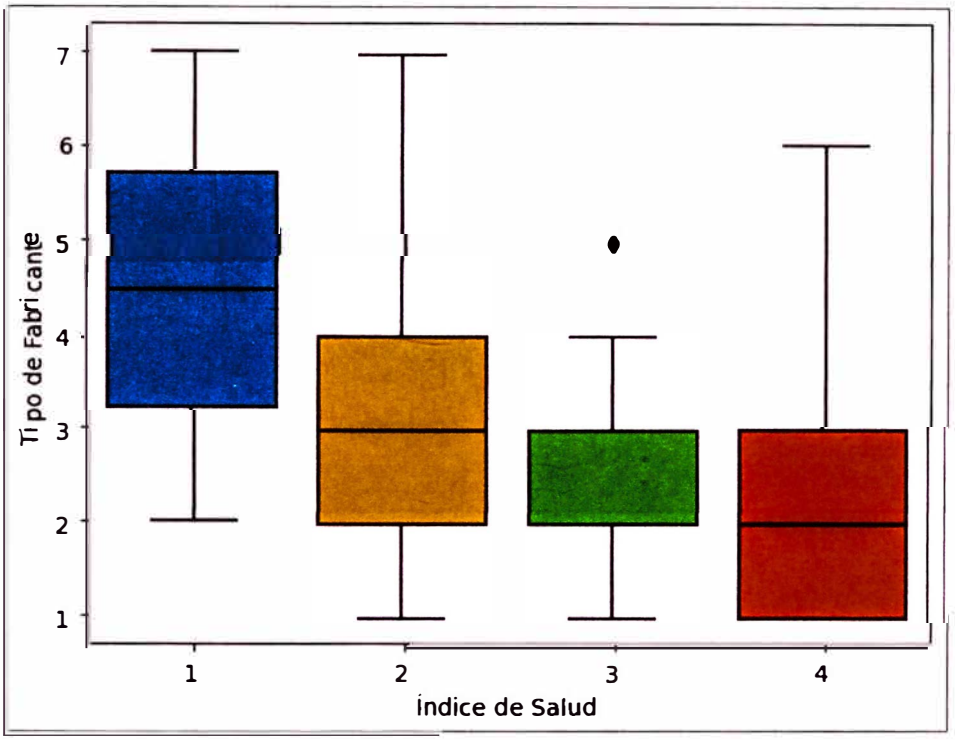


Nota: Elaboración propia

De este análisis Bivariante podemos observar que para el índice de salud tipo 1, el nivel de tensión principalmente involucrado en los interruptores es de 138kV, para el índice de salud tipo 2 los niveles de tensión involucrados son 60kV, 138kV y 220kV para el índice de salud tipo 3 el nivel tensión involucrado principalmente es de 220kV y para el índice de salud tipo "4" lo niveles de tensión involucrados son 220kV y 500kV

En la figura 29 se muestra el análisis bivariante entre el tipo de fabricante y el índice de salud de los interruptores de potencia.

Figura 29. Analisis Bivariante: "Tipo de Fabricante" vs "Índice de Salud"

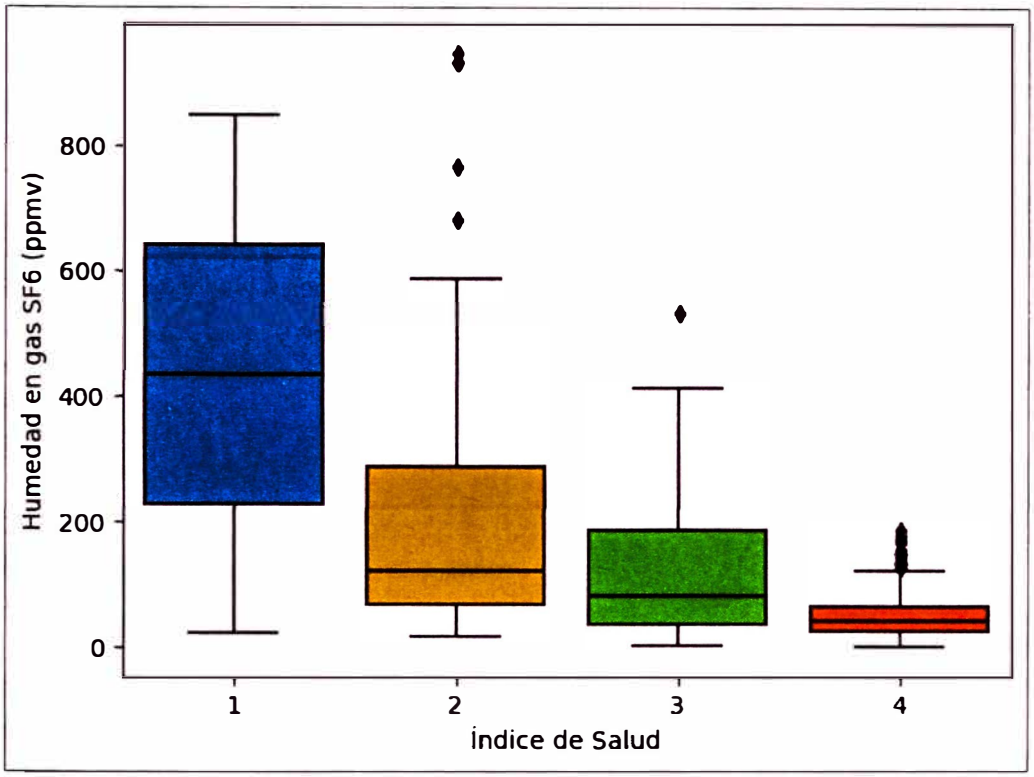


Nota: Elaboración propia

De este análisis Bivariante podemos observar que para el índice de salud tipo 1, Los fabricantes de interruptores involucrados principalmente son los que tienen el código 4 y 5, para el índice de salud tipo 2, los fabricantes involucrados son aquellos identificados con los códigos: 2, 3 y 4, para el índice de salud tipo 3, los fabricantes involucrados principalmente son aquellos con los códigos: 2 y 3. Finalmente para el índice de salud tipo 4 los fabricantes involucrados son principalmente aquellos con los códigos: 1, 2 y 3

En la figura 30 se muestra el análisis bivariante entre la humedad del gas SF6 y el índice de salud los interruptores de potencia.

Figura 30.. Analisis Bivariante: "Humedad" vs "Índice de Salud"

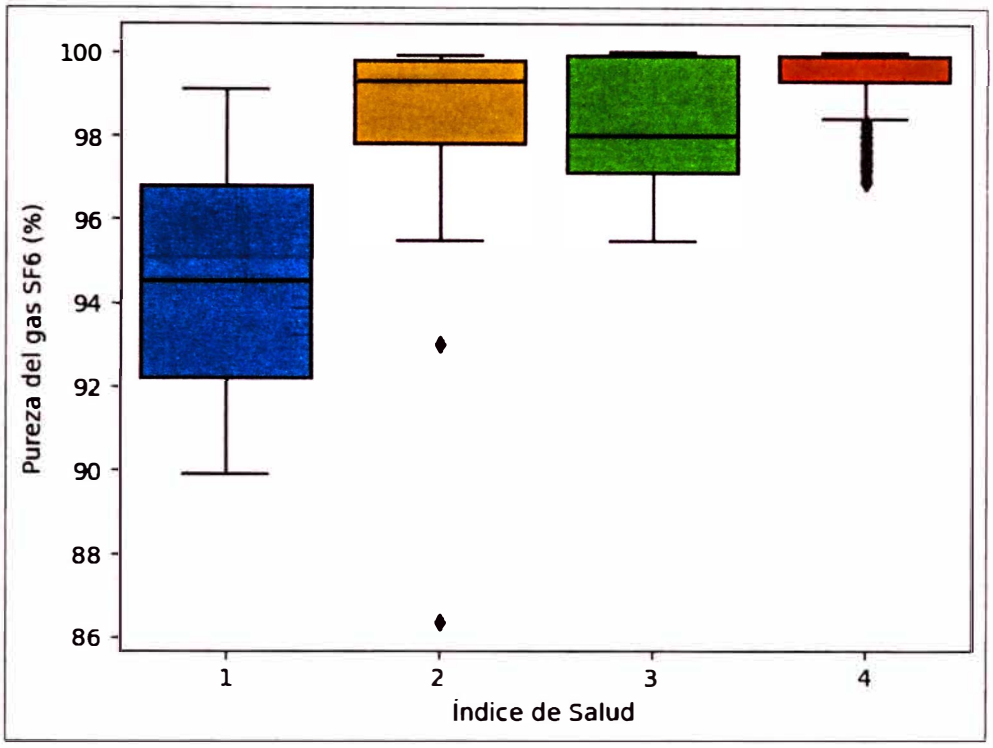


Nota: Elaboración propia

De este análisis Bivariante podemos observar que para el índice de salud tipo 1, la humedad en los interruptores se concentra principalmente en el rango de 200ppmv a 600ppmv, para el índice de salud tipo 2, la humedad se concentra principalmente en el rango de 100ppmv a 300ppmv, para el índice de salud tipo 3, la humedad se concentra principalmente en el rango de 50ppmv a 200ppmv y finalmente para el índice de salud tipo 4, la humedad se concentra principalmente en el rango de 30ppmv a 100ppmv. Debemos tener en cuenta que existen también en los índices de salud tipo 2, 3 y 4 interruptores que escapan de estos parámetros.

En la figura 31 se muestra el análisis bivariante entre la pureza del gas SF6 y el índice de salud de los interruptores de potencia.

Figura 31.. Analisis Bivariante: "Pureza" vs "Índice de Salud"

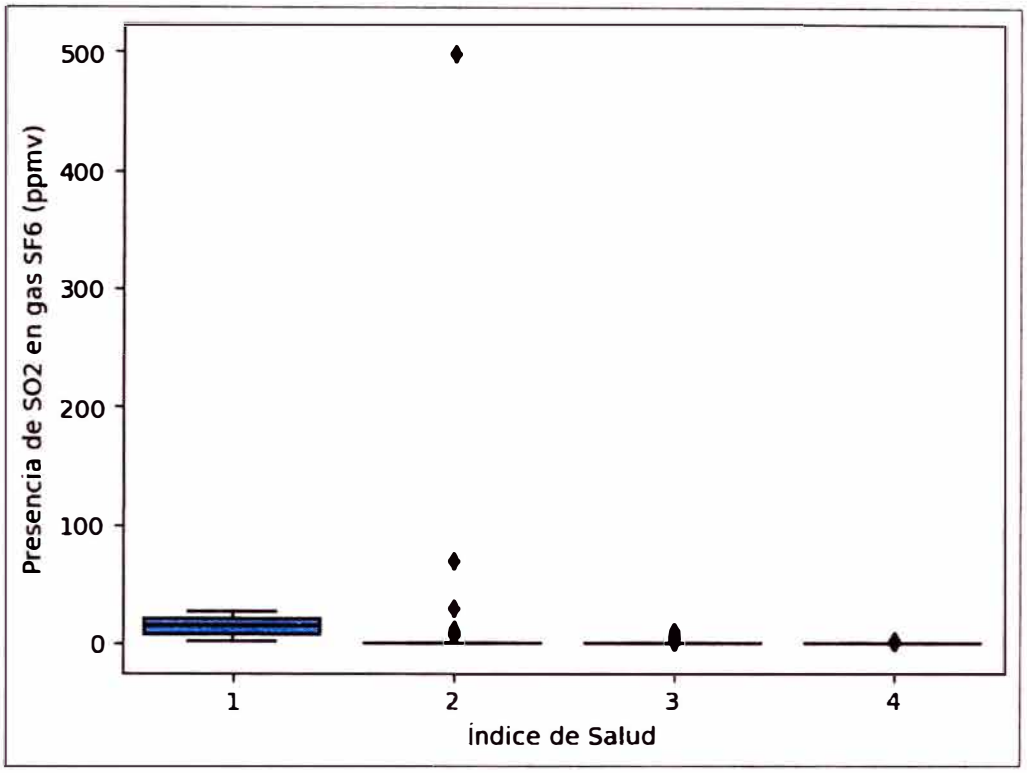


Nota: Elaboración propia

De este análisis Bivariante podemos observar que para el índice de salud tipo 1, la pureza en los interruptores se concentra principalmente en el rango del 92% al 96%, para el índice de salud tipo 2, la pureza se concentra principalmente en el rango del 98% al 100% existiendo outliers de 93% y del 87% incluso, para el índice de salud tipo 3, la pureza se concentra principalmente en el rango del 97% al 100% y finalmente para el índice de salud tipo 4 la pureza se concentra principalmente en el rango del 99% al 100%, existiendo outliers entre el 96% al 98%.

En la figura 32 se muestra el análisis bivariante entre la presencia de SO2 y el índice de salud de los interruptores de potencia.

Figura 32. Analisis Bivariante: "SO2" vs "Índice de Salud"

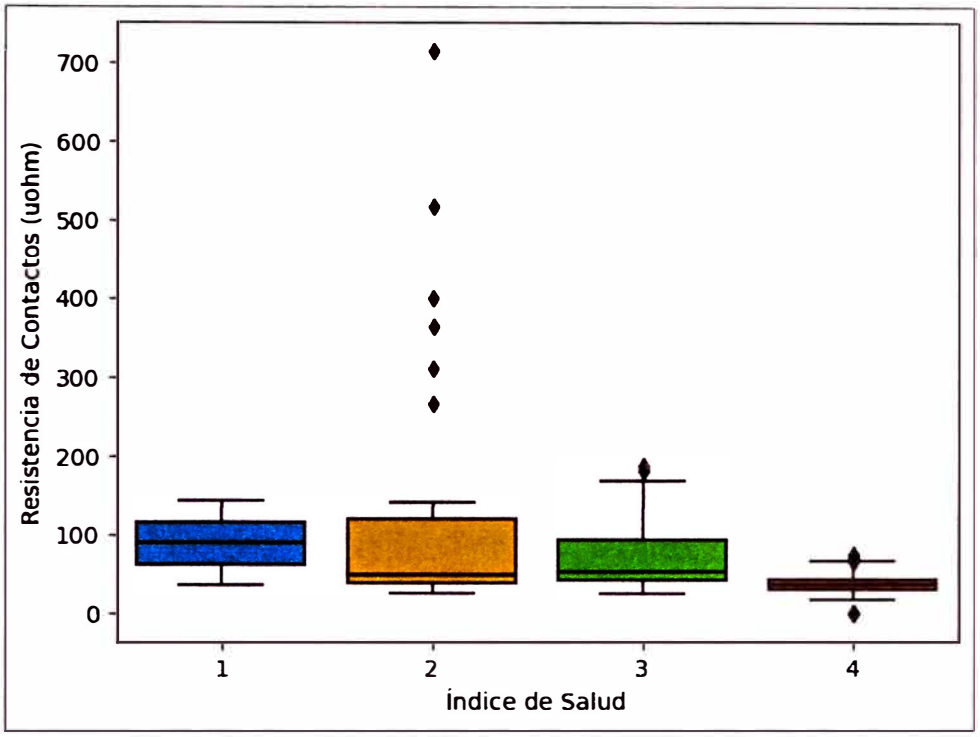


Nota: Elaboración propia

De este análisis Bivariante podemos observar que los equipos con existencia de SO2 en el gas SF6, están concentrados principalmente en el índice de salud tipo 1, y que para las demás categorías de salud existen casos de interruptores con presencia de SO2, pero a manera de outliers (casos muy puntuales y aislados)

En la figura 33 se muestra el análisis bivariante entre la resistencia de contactos y el índice de salud de los interruptores de potencia.

Figura 33. Analisis Bivariante: "Resistencia de Contacto" vs "Índice de salud"



Nota: Elaboración propia

De este análisis Bivariante podemos observar que para el índice de salud tipo 1, la resistencia de contactos de los interruptores se concentra en valores del rango de 80μΩ a 110μΩ, para el índice de salud tipo 2, la resistencia de contactos se concentra principalmente en el rango de 50 μΩ a 110μΩ, para el índice de salud tipo 3, la resistencia de contactos se concentra principalmente en el rango de 50 μΩ a 90 μΩ, y finalmente para el índice de salud tipo 4 la resistencia de contactos se concentra principalmente en el rango de 30 μΩ a 50 μΩ. Debemos tener en cuenta también que los índices de salud tipo 2, 3 y 4 existen interruptores que escapan de estos parámetros.

En la figura 34 se muestra el análisis bivariante entre la edad del interruptor y el índice de salud de los interruptores de potencia.

Figura 34. Analisis Bivariante: "Edad" vs "Índice de Salud"

Nota: Elaboración propia

De este análisis Bivariante podemos observar que para el índice de salud tipo 1, la edad de los interruptores se concentra principalmente entre los 20 a 35. Para el índice de salud tipo 2, la edad de los interruptores se concentra principalmente entre los 15 a 28 años. Para el índice de salud tipo 3, la edad de los interruptores se concentra principalmente entre los 10 a 28 años. Finalmente, para el índice de salud tipo 4 la edad de los interruptores se concentra principalmente entre los 10 a 25 años.

En síntesis, en esta sección logramos analizar y visualizar gráficamente como se correlaciona cada variable predictora con cada una de las categorías de la variable objetivo (Índice de Salud) encontrando ciertos patrones que ayudan a entender el grado de influencia de cada variable respecto al índice de Salud.

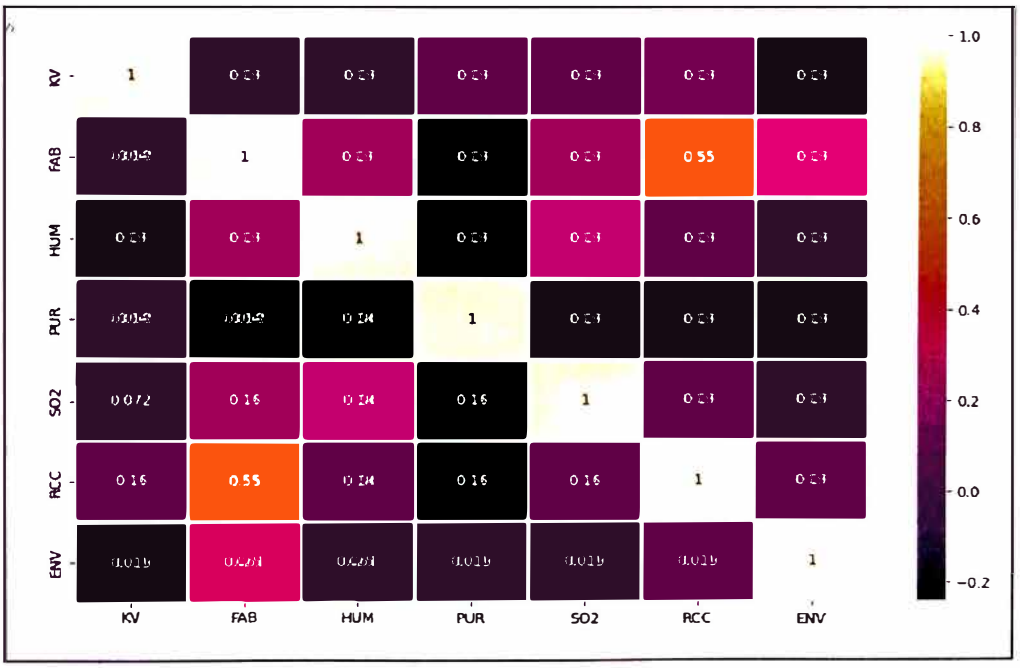
3.4. ANÁLISIS DE MATRIZ DE CORRELACIONES

Se realizo un analis de matriz de correlaciones entre las variables predictoras para determinar el grado de correlacion (positiva o negativa) entre todas estas variables involucradas. Usaremos para este fin el metodo de la matriz de Spearman.

El método de Spearman se utiliza para hallar el grado de correlación entre 2 o más variables correlacionadas no linealmente o que tienen distribuciones no normales, si las variables tuvieran una correlación lineal y distribuciones normales se podría usar el método de Person. Tanto el metodo de Spearman y Person estan disponible en python a travez del la biblioteca Seaborn.

En la figura 35 se muestra la Matriz de Correlaciones con el método de Spearman obtenida mediante Python.

Figura 35. Matriz de Correlaciones con el metodo de Spearman



Nota: Elaboración propia

Las correlaciones entre variables con valor cercano a "1" indican que las variables están 100% correlacionadas (Matriz diagonal) y en consecuencia están aportando la misma información. Al contrario, las variables con una correlación muy cercana a "0" indican que no existe mayor correlación entre ellas y en consecuencia cada una aportara diferente información. Las correlaciones intermedias sean positivas o negativas nos indica que el par de variables analizadas tienen cierto grado de correlación de manera directa o inversa, dicho de otra forma, una variable podría explicar el comportamiento de otra en cierto grado, por lo tanto, cada una de estas variables aporta un valor diferente a un futuro modelo predictivo.

En síntesis, de esta sección podemos afirmar lo siguiente de los resultados de matriz de Sperman desarrollada: Las variables con mayor correlación entre ellas es el "Tipo de fabricante" (FAB) y la "Resistencia de contactos" (RCC) con una correlación directa de 0.55, seguido de la "Humedad (HUM)" y la "Pureza del gas SF6" (PUR) con una correlación inversa de -0.24 y en tercer lugar la "Pureza del gas SF6" (PUR) con la "Presencia de SO2" (SO2) con una correlación inversa de -0.16.

Dado los resultados de que ninguna variable tiene una correlación con otra variable cercana a 1, entonces no se puede pensar en eliminar alguna de ellas a este nivel de análisis, por lo que usaremos otras técnicas para jerarquizar la importancia de cada una de las variables y pensar recién en eliminar alguna de ellas, lo cual veremos en la siguiente sección.

3.5. DETERMINACIÓN DE VARIABLES PRINCIPALES

Para determinar con métodos numérico las variables más importantes en un modelo predictivo el método más sencillo es realizar un análisis de correlación entre las variables predictoras y la variable objetivo, las variables más importantes serán las que tenga el valor más alto de correlación. Otro método muy utilizado es modelar los datos en algoritmos de machine learning de árboles de decisión ("Random Forest Classifier") el cual resulta más efectivo para este fin, dado que puede capturar relaciones no lineales y complejas entre las variables predictoras y la variable objetivo y además puede descubrir combinaciones de características que son importantes en conjunto, pero podrían no ser detectadas mediante correlaciones lineales simples .

"Random Forest Classifier" determina las variables que más se repiten en los nodos para la toma de decisiones y, en consecuencia, estas variables serán las más importantes. Para usar este método primero debemos determinar en función a nuestros datos cuales son los mejores hiperparametros a configurar en nuestro algoritmo, esta búsqueda de mejores hiperparametros se puede realizar de manera manual probando diferentes valores hasta conseguir métricas buenas o usando alguna técnica de búsqueda automática del machine learning como "GridSearch" o "RandomizedSearch". Para todo el desarrollo de la presente investigación usaremos la técnica de "RandomizedSearch" por haber demostrado mejor eficiencia para encontrar los hiperparametros en los diferentes ensayos realizados. Estas evidencias lo podemos apreciar en la matriz de confusión, métricas y curva de aprendizaje probados para los hiperparametros obtenido con GridSearch, el cual se encuentra en el Anexo 2 de la presente investigación.

Según lo descrito, primero determinaremos las variables importantes en función al grado de correlación lineal entre las variables predictoras y la variable objetivo (Índice de Salud). En la figura 36 se muestran el código implementado en Python para este fin y el reporte de variables principales determinadas mediante este método.

Figura 36. Metodo 1 - Correlación lineal entre variables predictoras y el Índice de Salud.

```
import pandas as pd

# Cargar el conjunto de datos desde Google Sheets
link = 'https://docs.google.com/spreadsheets/d/e/2PACX-1vRftR0vMrgdgLzR08wxGM8kiYupxCHUH9AP9
dfx = pd.read_csv(link)
dataset = dfx.iloc[:, 1:]

# Separar características (X) y etiquetas (y)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

# Calcular las correlaciones entre cada variable predictora y la variable objetivo
correlations = X.corrwith(y)

# Ordenar los resultados por valor absoluto de la correlación
correlations = correlations.abs().sort_values(ascending=False)

print("Correlaciones entre variables predictoras y la variable objetivo:\n", correlations)
```

```
Correlaciones entre variables predictoras y la variable objetivo:
 HUM    0.522575
RCC     0.478248
PUR     0.390355
FAB     0.286841
ENV     0.233256
SO2     0.190911
KV      0.106007
dtype: float64
```

Nota: Elaboración propia

De acuerdo con este análisis de correlaciones, las variables predictoras más importantes debido al grado de correlación lineal con la variable objetivo (Índice de Salud) son: “La Humedad del gas SF6” (HUM), “La resistencia de contactos del interruptor” (RCC) y la “Pureza del gas SF6” (PUR).

Ahora realizaremos un procedimiento similar, pero usando el segundo método descrito, el cual será finalmente el que utilizaremos para seleccionar nuestras variables predictoras que se implementaran en los diferentes modelos de machine learning que necesitamos evaluar. Como ya se mencionó antes, para usar este método primero debemos encontrar los mejores hiperparametros a configurar en el algoritmo de "Random Forest". En la figura 37 se muestran el código implementado en Python para este fin y el reporte de variables principales determinadas mediante este método

Figura 37. Busqueda de mejores hiperparametros con "RandomizedSearch"

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, RandomizedSearchCV
import pandas as pd
import numpy as np

link = 'https://docs.google.com/spreadsheets/d/e/2PACX-1vRftR0vMrgdgLzR08wxGMBkiYupxCHUH9AP9LC2gwiiTZVP5Hw
dfx = pd.read_csv(link)
dataset = dfx.iloc[:, 1:]

X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

# Separar datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Definir los hiperparámetros y sus valores a explorar
param_dist = {
    'n_estimators': np.arange(25, 251, 25),
    'max_depth': [None, 5, 10, 15, 20],
    'min_samples_split': np.arange(2, 9),
    'min_samples_leaf': [0.01, 0.1, 1, 2, 3, 4, 5]


# Crear el modelo de Random Forest Classifier
model = RandomForestClassifier(random_state=42)

# Crear el objeto RandomizedSearchCV
random_search = RandomizedSearchCV(
    model, param_distributions=param_dist, n_iter=100, cv=5, scoring='f1', n_jobs=-1, random_state=42


# Ajustar el objeto RandomizedSearchCV con los datos de entrenamiento
random_search.fit(X_train, y_train)

# Obtener el mejor modelo y sus hiperparámetros
best_model = random_search.best_estimator_
best_params = random_search.best_params_

print("Mejores hiperparámetros:", best_params)
```

```
c:\Users\ahueman\python\lib\site-packages\sklearn\model_selection\_split.py:680: UserWarning: The least popul
  UserWarning,
Mejores hiperparámetros: {'n_estimators': 25, 'min_samples_split': 5, 'min_samples_leaf': 3, 'max_depth': 10}
```

Nota: Elaboración propia

Es preciso aclarar que “RandomizedSearch” como todo algoritmo de machine learning usa también hiperparametros, en ese sentido en el Anexo 3, se explican los hiperparametros usados en la configuración de este algoritmo de búsqueda. Una vez obtenido los mejores hiperparametros con “RandomizedSearch” configuraremos estos valores al algoritmo de “Random Forest Classifier”. En la figura 38 se muestra la jerarquización de variables, determinadas mediante el método de modelado de datos en arboles de decisión.

Figura 38. Metodo 2 - Modelado de datos en arboles de decisión

```
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Cargar el conjunto de datos desde Google Sheets
link = 'https://docs.google.com/spreadsheets/d/e/2PACX-1vRftR0vMrgdgLzR08wxGMBkiYupxCHUH9AP9LC2gwiiTZVP5HnESaCMTFvcLgy2k
dfx = pd.read_csv(link)
dataset = dfx.iloc[:, 1:]


# Separar características (X) y etiquetas (y)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

# Escalar características
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Convertir X_scaled a un DataFrame
X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)

# separar datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_scaled_df, y, test_size=0.3, random_state=42)

# crear modelo de random forest classifier
model = RandomForestClassifier(max_depth=10, min_samples_leaf=3, min_samples_split=5, n_estimators=25, random_state=42)

# ajustar modelo con datos de entrenamiento
model.fit(X_train, y_train)

# Obtener la importancia de las variables
importance = model.feature_importances_

# Crear un dataframe para visualizar los resultados
df_importance = pd.DataFrame({'variable': X.columns, 'importancia': importance})
df_importance = df_importance.sort_values('importancia', ascending=False)
df_importance['importancia_acumulada'] = df_importance['importancia'].cumsum()
print("Importancia de variables:\n", df_importance)
```

```
✓ 2.9s

Importancia de variables:
   variable  importancia  importancia_acumulada
      RCC     0.358130              0.358130
      HUM     0.321078              0.679208
      PUR     0.121163              0.800371
4     SO2     0.073428              0.873799
      ENV     0.059756              0.933555
1     FAB     0.048685              0.982240
0      KV     0.017760              1.000000
```

Nota: Elaboración propia

En síntesis, de acuerdo con los resultados de este método, el cual usaremos para jerarquizar las variables predictoras, concluimos que las variables que más aportan al modelo predictivo son: "La Resistencia de Contactos" (RCC), "La Humedad del gas SF6" (HUM), "La Pureza del gas SF6" (PUR), "La presencia de SO2 en el gas SF6" (SO2) y" La edad del interruptor" (ENV). También se pude observar de los resultados que las variables que menos aportan son: El Tipo de Fabricante (FAB) y El nivel de voltaje de Operación (KV), por lo que para efectos de la presente investigación se estará descartando estas variables durante el proceso entrenamiento de los modelos predictivos.

3.6. MODELADO Y ENTRENAMIENTO DE ALGORITMOS PREDICTIVOS

Para continuar con esta etapa debemos previamente seleccionar los algoritmos predictivos a usar, en ese sentido tomando en cuenta los algoritmos de clasificación más populares, existentes por ejemplo en softwares especializados en analítica como la plataforma Azure de Microsoft[32] y también considerando el uso y rendimiento de estos algoritmos en la revisión de los antecedentes bibliográficos (Capitulo 1.2).Se optó por seleccionar los siguientes 05 algoritmos:

1. Random Forest Classifier
2. Gradient Boosting Classifier
3. Support Vector Machines
4. K-Nearest Neighbors
5. Redes Neuronales Artificiales

[32] Azure (2023), "Algoritmos de aprendizaje automático: Una introducción a las matemáticas y la lógica subyacentes en el aprendizaje automático". https://azure.microsoft.com/es-mx/resources/cloud-computing-dictionary/what-are-machine-learning-algorithms

Una vez seleccionado los algoritmos, para propósitos de ser comparables lo resultados de sus métricas, todos recibieron la misma data de entrenamiento balanceada[33] y escalada[34], lo cual no fue necesario en el proceso anterior (selección de variables), dado que se usó una función del algoritmo "Random Forest Classifier" y este algoritmo en particular ya realiza el escalado de datos internamente, es robusto para casos de datos desbalanceados y además porque en esa etapa, no se utilizó el algoritmo con la intención de compararse con otros algoritmos.

Una vez obtenido los mejores hiperparametros con "RandomizedSearch", se calculará la Matriz de confusión de cada modelo y las principales métricas globales por cada categoría salud tales como: precisión, exactitud, sensibilidad, F1-Score. Con el fin de analizar el resultado de cada uno de ellos, compararlos y evaluarlos.

Modelado 1: Algoritmo de Random Forest Classifier. - Este algoritmo es muy popular debido a su capacidad para manejar datos no lineales y tiene buen desempeño en conjuntos de datos grandes. En la figura 39 se muestra los hiperparametros óptimos encontrados para este modelo en función a nuestros datos. En la figura 40 se muestra el Modelado de datos mediante el algoritmo de "Random Forest Classifier", usando los hiperparametros óptimos y en la figura 41 la matriz de confusión y las métricas obtenidas.

[33] Proceso de modificar la distribución de las clases en un conjunto de datos para evitar desequilibrios, mejorando la capacidad del modelo para tratar con clases minoritarias

[34] Ajustar las características numéricas para que tengan una escala similar, previniendo que características con magnitudes diferentes tengan un impacto desproporcionado en modelos de machine learning

Figura 39. Búsqueda de hiperparametros óptimos para Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from sklearn.preprocessing import StandardScaler
import pandas as pd
from imblearn.over_sampling import SMOTE
import numpy as np

link = 'https://docs.google.com/spreadsheets/d/e/2PACX-1vInVafIx6fRGlzlsVytZMWJnFvstgdIobhRa54IMBG-NknxE5z6
dfx = pd.read_csv(link)
dataset = dfx.iloc[:, 1:]

X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

# Realizar oversampling de las clases con menor cantidad de datos usando SMOTE
max_samples = max(y.value_counts())
k_neighbors = min(y.value_counts()) - 1
smote = SMOTE(random_state=42, k_neighbors=k_neighbors)
X_balanced, y_balanced = smote.fit_resample(X, y)

# Escalar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_balanced)

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Definir los hiperparámetros y sus valores a explorar
param_dist = {
    'n_estimators': np.arange(25, 251, 25),
    'max_depth': [None] + list(np.arange(5, 21, 5)),
    'min_samples_split': np.arange(2, 9),
    'min_samples_leaf': [0.01, 0.1, 1, 2, 3, 4, 5]
}

# Crear el modelo de Random Forest Classifier
model = RandomForestClassifier(random_state=42)

# Crear el objeto RandomizedSearchCV
random_search = RandomizedSearchCV(model, param_distributions=param_dist, n_iter=100,
                                   cv=5, scoring='f1', n_jobs=-1, random_state=42)

# Ajustar el objeto RandomizedSearchCV con los datos de entrenamiento escalados y balanceados
random_search.fit(X_scaled, y_balanced)

# Obtener el mejor modelo y sus hiperparámetros
best_model = random_search.best_estimator_
best_params = random_search.best_params_

print("Mejores hiperparámetros:", best_params)
```

```
Mejores hiperparámetros: {'n_estimators': 25, 'min_samples_split': 5, 'min_samples_leaf': 3, 'max_depth': 10}
```

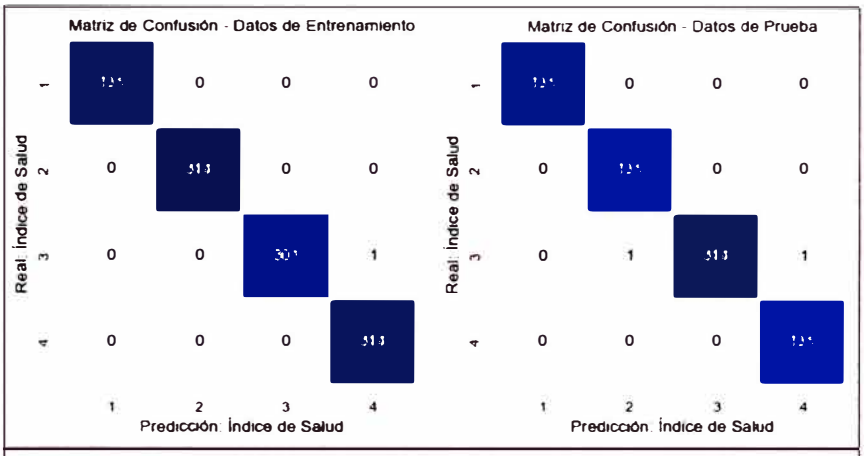
Nota: Elaboración propia

Figura 40. Modelado por Random Forest Classifier con parametros óptimos

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import pandas as pd
from imblearn.over_sampling import SMOTE
from pprint import

# Cargar el conjunto de datos desde Google Sheets
link = 'https://docs.google.com/spreadsheets/d/e/2PACX-1vInVaf1x6fRGlzlsVytZNWJnfvstgdIobhRa64IMBG-MknxE5z6nCGBRtF6xzdPX
dfx = pd.read_csv(link)
dataset= dfx.iloc[:,1:]

# Separar caracteristicas (X) y etiquetas (y)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

# Realizar oversampling de las clases con menor cantidad de datos usando SMOTE
max_samples = max(y.value_counts())
k_neighbors = min(y.value_counts()) - 1
smote = SMOTE(random_state=42, k_neighbors=k_neighbors)
X_balanced, y_balanced = smote.fit_resample(X, y)

# Escalar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_balanced)

# separar datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_balanced, test_size=0.3, random_state=42)

# Obtener las etiquetas de clase únicas en el conjunto de entrenamiento y prueba
labels = sorted(list(set(y_train.values.ravel())))

# crear modelo de random forest classifier
model = RandomForestClassifier(max_depth=10, min_samples_leaf=3,min_samples_split=5, n_estimators=25,random_state=42)

# ajustar modelo con datos de entrenamiento
model.fit(X_train, y_train)

# hacer predicciones para conjunto de datos entrenamiento y prueba
train_predictions = model.predict(X_train)
test_predictions = model.predict(X_test)

# matriz de confusion del train y el test
from sklearn.metrics import  , confusion_matrix
print('\nVisualizando la matriz de confusion del train:\n')
print(confusion_matrix(y_train, train_predictions))
print('\nVisualizando la matriz de confusion del test:\n')
print(confusion_matrix(y_test, test_predictions))

# calcular metricas para conjunto de entrenamiento
train_accuracy = accuracy_score(y_train, train_predictions)
global_train_precision = precision_score(y_train, train_predictions, average='macro')
train_precision = precision_score(y_train, train_predictions, average=None, labels=labels)
global_train_recall = recall_score(y_train, train_predictions, average='macro')
train_recall = recall_score(y_train, train_predictions, average=None, labels=labels)
global_train_f1 = f1_score(y_train, train_predictions, average='macro')
train_f1 = f1_score(y_train, train_predictions, average=None, labels=labels)

# calcular metricas para conjunto de prueba
test_accuracy = accuracy_score(y_test, test_predictions)
global_test_precision = precision_score(y_test, test_predictions, average='macro')
test_precision = precision_score(y_test, test_predictions, average=None, labels=labels)
global_test_recall = recall_score(y_test, test_predictions, average='macro')
test_recall = recall_score(y_test, test_predictions, average=None, labels=labels)
global_test_f1 = f1_score(y_test, test_predictions, average='macro')
test_f1 = f1_score(y_test, test_predictions, average=None, labels=labels)
```

Nota: Elaboración propia

Figura 41. Matriz de confusión y metricas de Random Forest Classifier




```
Métricas de conjunto de entrenamiento con Random Forest Classifier:
Exactitud: 0.9992
Precisión: 0.9992
Precisión (individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 1.0000
  - Para el Índice de salud 4: 0.9968
Sensibilidad: 0.9992
Sensibilidad (Individual):
  -  Para el Índice de salud 1: 1.0000
  -  Para el Índice de salud 2: 1.0000
  -  Para el Índice de salud 3: 0.9967
  -  Para el Índice de salud 4: 1.0000
F1-Score: 0.9992
F1-Score (individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 0.9983
  - Para el Índice de salud 4: 0.9984
Métricas de conjunto de prueba con Random Forest Classifier:
Exactitud: 0.9963
Precisión: 0.9961
Precisión (Individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 0.9921
  - Para el Índice de salud 3: 1.0000
  - Para el Índice de salud 4: 0.9924
Sensibilidad: 0.9965
Sensibilidad (Individual):
  -  Para el Índice de salud 1: 1.0000
  -  Para el Índice de salud 2: 1.0000
  -  Para el Índice de salud 3: 0.9860
  -  Para el Índice de salud 4: 1.0000
F1-Score: 0.9963
F1-Score (individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 0.9960
  - Para el Índice de salud 3: 0.9930
  - Para el Índice de salud 4: 0.9962
```

Nota: Elaboración propia

De los resultados de las métricas calculadas podemos manifestar lo siguiente para el conjunto de datos de entrenamiento:

- La precisión global tiene el valor de 99.92% y la precisión individual de todas las categorías de salud tiene un valor superior al 99%, siendo la mínima de 99.68% y la máxima del 100%, lo cual se puede corroborar en la matriz de confusión.
- La sensibilidad global tiene el valor de 99.92% y la sensibilidad individual de cada categoría de salud también obtienen métricas mayores al 99%, siendo la mínima del 99.67% y la máxima del 100%, lo cual se puede corroborar también en la matriz de confusión.
- El F1-Score global tiene el valor de 99.92% y en el F1-Score de cada categoría de salud tienen un valor superior al 99%, siendo la mínima del 99.83% y la máxima del 100%

Respecto a las métricas para el conjunto de datos de prueba podemos manifestar lo siguiente:

- La precisión global tiene el valor de 99.61% y la precisión individual de todas las categorías de salud tiene un valor mayor a 99%, siendo la mínima de 99.21% y la máxima del 100%, lo cual se puede corroborar en la matriz de confusión.
- La sensibilidad global tiene el valor de 99.65% y la sensibilidad individual de cada categoría de salud también obtienen métricas del 100% a excepción de la categoría de salud tipo 3 que logra un 98.60%, lo cual se puede corroborar también en la matriz de confusión.

- El F1-Score global tiene el valor de 99.63% y en el F1-Score de cada categoría de salud tienen un valor superior al 99%, siendo la mínima del 99.30% y la máxima del 100%

Se debe recordar que la precisión es la relación entre las predicciones correctas (verdaderos positivos) respecto al total de casos predichos (verdaderos positivos + falsos positivos). También se debe recordar que la sensibilidad es relación entre las predicciones positivas correctas respecto al total de casos positivos reales (verdaderos positivos + falsos negativos) y que el F1-Score es la combinación de la precisión y la sensibilidad en una sola medida (promedio armónico), esta métrica es muy importante dado que logra medir el equilibrio entre ambas métricas y nos sirve para comparar este equilibrio con otros modelos. Ver formulación y mayor detalle de cada una de las métricas en el ítem 2.3.3 "Principales métricas usadas en modelos predictivos".

Modelado 2: Gradient Boosting Classifier.-Este modelo construye árboles de decisión de manera secuencial, mejorando el modelo con cada nuevo árbol. En la figura 42 se muestra los hiperparametros óptimos encontrados para este modelo en función a nuestros datos. En la figura 43 se muestra el Modelado de datos mediante el algoritmo de "Gradient Boosting Classifier", usando los hiperparametros óptimos y en la figura 44 la matriz de confusión y las métricas obtenidas,

Figura 42. Búsqueda de hiperparametros óptimos para Gradient Boosting Classifier

```
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from sklearn.preprocessing import StandardScaler
import pandas as pd
from imblearn.over_sampling import SMOTE
import numpy as np

# Carga de datos y preprocesamiento
link = 'https://docs.google.com/spreadsheets/d/e/2PACX-1vInVafIx6fRGlzlsVytZN4WJnfvstgdIobhRa64IMBG-NknxE5z6
dfx = pd.read_csv(link)
dataset = dfx.iloc[:, 1:]

# Separar caracteristicas (X) y etiquetas (y)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1] - 1

# Realizar oversampling de las clases con menor cantidad de datos usando SMOTE
max_samples = max(y.value_counts())
k_neighbors = min(y.value_counts()) - 1
smote = SMOTE(random_state=42, k_neighbors=k_neighbors)
X_balanced, y_balanced = smote.fit_resample(X, y)

# Escalar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_balanced)

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Definir los hiperparámetros y sus valores a explorar
param_dist = {
    'n_estimators': [25, 50, 100, 150, 200, 250],
    'max_depth': [None, 5, 10, 15, 20],
    'min_child_weight': [1, 2, 3, 4, 5],  # Hiperparámetro especifico de XGBoost
    'learning_rate': [0.01, 0.1, 0.2, 0.3],  # Hiperparámetro especifico de XGBoost
}

# Crear el modelo de XGBClassifier
model = XGBClassifier(random_state=42)

# Crear el objeto RandomizedSearchCV
random_search = RandomizedSearchCV(model, param_distributions=param_dist, n_iter=100, cv=5,
                                   scoring='f1', n_jobs=-1, random_state=42)

# Ajustar el objeto RandomizedSearchCV con los datos de entrenamiento escalados y balanceados
random_search.fit(X_scaled, y_balanced)

# Obtener el mejor modelo y sus hiperparámetros
best_model = random_search.best_estimator_
best_params = random_search.best_params_

print("Mejores hiperparámetros:", best_params)
```

```
c:\Users\ahuaman\python\lib\site-packages\sklearn\model_selection\_search.py:972: UserWarning: One or more of t
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan]
  category=UserWarning,
Mejores hiperparámetros: {'n_estimators': 100, 'min_child_weight': 4, 'max_depth': 15, 'learning_rate': 0.01}
```

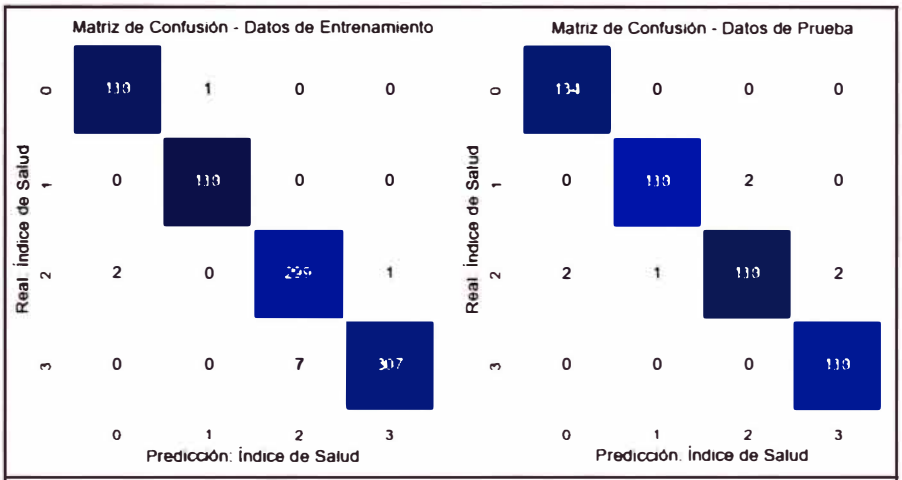
Nota: Elaboración propia

Figura 43. Modelado por Gradient Boosting Classifier parametros óptimos

```
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import pandas as pd
from imblearn.over_sampling import SMOTE
from pprint import pprint

# Cargar el conjunto de datos desde Google Sheets
link = 'https://docs.google.com/spreadsheets/d/e/2PACX-1vTnVafIx6fRGlzlsVytZMWJnFvstgdlobhRa64IMBG-NknxE5z6nC
dfx = pd.read_csv(link)
dataset= dfx.iloc[:,1:]

# Separar caracteristicas (x) y etiquetas (y)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1] - 1

# Realizar oversampling de las clases con menor cantidad de datos usando SMOTE
max_samples = max(y.value_counts())
k_neighbors = min(y.value_counts()) - 1
smote = SMOTE(random_state=42, k_neighbors=k_neighbors)
X_balanced, y_balanced = smote.fit_resample(X, y)

# Escalar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_balanced)

# separar datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_balanced, test_size=0.3, random_state=42)

# Obtener las etiquetas de clase únicas en el conjunto de entrenamiento y prueba
labels = sorted(list(set(y_train.values.ravel())))

# crear modelo de XGBoost classifier
model = XGBClassifier(learning_rate=0.01,max_depth=15,min_child_weight=4,n_estimators= 100,random_state=42)

# ajustar modelo con datos de entrenamiento
model.fit(X_train, y_train)

# hacer predicciones para conjunto de datos entrenamiento y prueba
train_predictions = model.predict(X_train)
test_predictions = model.predict(X_test)

# matriz de confusion del train y el test
from sklearn.metrics import [illegible], confusion_matrix
print('\nVisualizando la matriz de confusion del train:\n')
print(confusion_matrix(y_train, train_predictions))
print('\nVisualizando la matriz de confusion del test:\n')
print(confusion_matrix(y_test, test_predictions))

# calcular metricas para conjunto de entrenamiento
train_accuracy = accuracy_score(y_train, train_predictions)
global_train_precision = precision_score(y_train, train_predictions, average='macro')
train_precision = precision_score(y_train, train_predictions, average=None, labels=labels)
global_train_recall = recall_score(y_train, train_predictions, average='macro')
train_recall = recall_score(y_train, train_predictions, average=None, labels=labels)
global_train_f1 = f1_score(y_train, train_predictions, average='macro')
train_f1 = f1_score(y_train, train_predictions, average=None, labels=labels)

# calcular métricas para conjunto de prueba
test_accuracy = accuracy_score(y_test, test_predictions)
global_test_precision = precision_score(y_test, test_predictions, average='macro')
test_precision = precision_score(y_test, test_predictions, average=None, labels=labels)
global_test_recall = recall_score(y_test, test_predictions, average='macro')
test_recall = recall_score(y_test, test_predictions, average=None, labels=labels)
global_test_f1 = f1_score(y_test, test_predictions, average='macro')
test_f1 = f1_score(y_test, test_predictions, average=None, labels=labels)
```

Nota: Elaboración propia

Figura 44. Resultados de metricas Gradient Boosting Classifier




```
Métricas de conjunto de entrenamiento con Gradient Boosting Classifier:
Exactitud: 0.9912
Precisión: 0.9911
Precisión (individual):
  - Para el índice de salud 1: 0.9936
  - Para el índice de salud 2: 0.9969
  - Para el índice de salud 3: 0.9771
  - Para el índice de salud 4: 0.9968
Sensibilidad: 0.9911
Sensibilidad (Individual):
  -  Para el índice de salud 1: 0.9968
  -  Para el índice de salud 2: 1.0000
  -  Para el índice de salud 3: 0.9901
  -  Para el índice de salud 4: 0.9777
F1-Score: 0.9911
F1-Score (individual):
  - Para el índice de salud 1: 0.9952
  - Para el índice de salud 2: 0.9984
  - Para el índice de salud 3: 0.9836
  - Para el índice de salud 4: 0.9871

Métricas de conjunto de prueba con Gradient Boosting Classifier:
Exactitud: 0.9869
Precisión: 0.9870
Precisión (Individual):
  - Para el índice de salud 1: 0.9853
  - Para el índice de salud 2: 0.9920
  - Para el índice de salud 3: 0.9857
  - Para el índice de salud 4: 0.9850
Sensibilidad: 0.9873
Sensibilidad (Individual):
  -  Para el índice de salud 1: 1.0000
  -  Para el índice de salud 2: 0.9841
  -  Para el índice de salud 3: 0.9650
  -  Para el índice de salud 4: 1.0000
F1-Score: 0.9871
F1-Score (individual):
  - Para el índice de salud 1: 0.9926
  - Para el índice de salud 2: 0.9880
  - Para el índice de salud 3: 0.9753
  - Para el índice de salud 4: 0.9924
```

Nota: Elaboración propia

De los resultados de las métricas calculadas podemos manifestar lo siguiente para el conjunto de datos de entrenamiento:

- La precisión global tiene el valor de 99.11% y la precisión individual de todas las categorías de salud es alta (>99%) a excepción de la categoría de salud tipo 3 (97.71%) lo cual se puede corroborar también en la matriz de confusión.
- La sensibilidad global tiene el valor de 99.11% y la sensibilidad individual de cada categoría de salud también obtienen métricas altas (>99%) a excepción de la categoría de salud tipo 3 (97.77%) lo cual se puede corroborar también en la matriz de confusión.
- El F1-Score global tiene el valor de 99.11% y en el F1-Score de cada categoría de salud en las categorías de salud 1 y 2 tiene un valor mayor a 99% mientras que para las categorías 3 y 4 tienen un valor mayor a 98%.

Respecto a las métricas para el conjunto de datos de prueba podemos manifestar lo siguiente:

- La precisión global tiene el valor de 98.70% y la precisión individual de todas las categorías de salud están en el orden del 98% a excepción de la categoría de salud tipo 2 (99.20%) lo cual se puede corroborar también en la matriz de confusión.
- La sensibilidad global tiene el valor de 98.73% y la sensibilidad individual de cada categoría de salud obtienen métricas variadas siendo la más baja del 96.50% para categoría de salud tipo 3 y la más alta del 100% para las categorías de salud 1 y 4.
- El F1-Score global tiene el valor de 98.71% y el F1-Score individual de cada categoría de salud obtienen métricas variadas siendo la más baja del 97.53%

para categoría de salud tipo 3 y la más alta del 99.26% para las categorías de salud tipo 1.

- Ver formulación y mayor detalle de cada una de las métricas en el ítem 2.3.3 "Principales métricas usadas en modelos predictivos".

Modelado 3: Support Vector Machines (SVM).- Este modelo es útil en problemas de clasificación lineal y no lineal, y es especialmente efectivo en datos de alta dimensionalidad. En la figura 45 se muestra los hiperparametros óptimos encontrados para este modelo en función a nuestros datos, En la figura 46 se muestra el Modelado de datos mediante el algoritmo de "Support Vector Machines" usando los hiperparametros óptimos, en la figura 47 la matriz de confusión y las métricas obtenidas.

Figura 45. *Búsqueda de hiperparametros óptimos para Support Vector*

```
from sklearn.svm import SVC
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from sklearn.preprocessing import StandardScaler
import pandas as pd
from imblearn.over_sampling import SMOTE
import numpy as np

# Cargar el conjunto de datos desde Google Sheets
link = 'https://docs.google.com/spreadsheets/d/e/2PACX-1vTnVafIx6fRGlzlsVytZMwJnFvstgdIobhRa641MBG-NknxE5z6nCGBRtf
dfx = pd.read_csv(link)
dataset = dfx.iloc[:, 1:]

# Separar características (X) y etiquetas (y)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

# Realizar oversampling de las clases con menor cantidad de datos usando SMOTE
max_samples = max(y.value_counts())
k_neighbors = min(y.value_counts()) - 1
smote = SMOTE(random_state=42, k_neighbors=k_neighbors)
X_balanced, y_balanced = smote.fit_resample(X, y)

# Escalar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_balanced)

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Definir los hiperparámetros y sus valores a explorar
param_dist = {
    'C': [0.1, 1, 5, 10, 20],  # Parámetro de regularización
    'kernel': ['linear', 'rbf', 'poly'],  # Funciones de kernel
    'degree': [2, 3, 4],  # Grado del polinomio (para kernel 'poly')
    'class_weight': [None, 'balanced'],  # Opciones de peso de clase
    'gamma': ['scale', 'auto', 0.1, 1, 5],  # Parámetro gamma
    'coef0': [0.0, 0.1, 1, 2]  # Coeficiente de la función de pérdida
}

# Crear modelo de SVC
model = SVC(random_state=42)

# Crear el objeto RandomizedSearchCV
random_search = RandomizedSearchCV(model, param_distributions=param_dist, n_iter=100, cv=5,
                                   scoring='f1', n_jobs=-1, random_state=42)

# Ajustar el objeto RandomizedSearchCV con los datos de entrenamiento escalados y balanceados
random_search.fit(X_scaled, y_balanced)

# Obtener el mejor modelo y sus hiperparámetros
best_model = random_search.best_estimator_
best_params = random_search.best_params_

print("Mejores hiperparámetros:", best_params)
```

```
Mejores hiperparámetros: {'kernel': 'rbf', 'gamma': 'scale', 'degree': 3, 'coef0': 2, 'class_weight': None, 'C': 20}
```

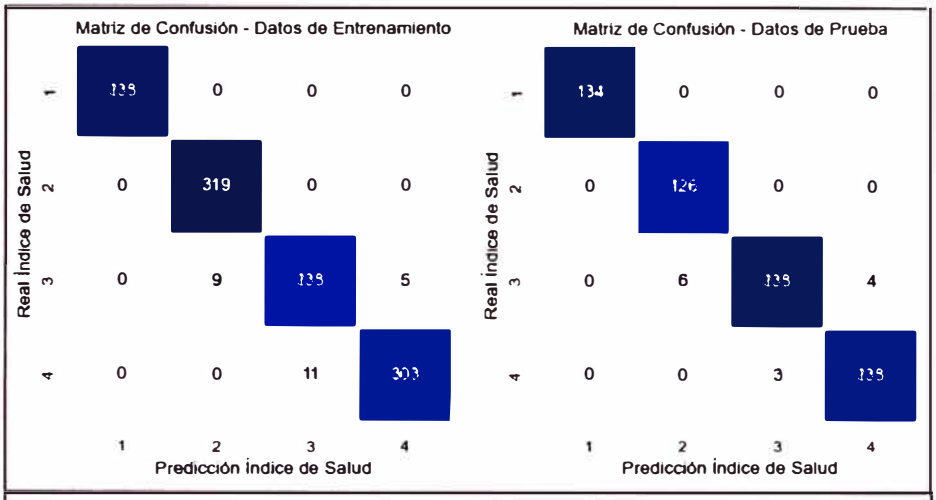
Nota: Elaboración propia

Figura 46. Modelado por Support Vector Machines con parametros óptimos

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import pandas as pd
from imblearn.over_sampling import SMOTE
from pprint import pprint

# Cargar el conjunto de datos desde Google Sheets
link = 'https://docs.google.com/spreadsheets/d/e/2PACX-1vInVafIx6fRGlzlsVytZMWJnfvstgdIobhRa64IMBG-MknxE5z6
dfx = pd.read_csv(link)
dataset = dfx.iloc[:, 1:]

# Separar caracteristicas (X) y etiquetas (y)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

# Realizar oversampling de las clases con menor cantidad de datos usando SMOTE
max_samples = max(y.value_counts())
k_neighbors = min(y.value_counts()) - 1
smote = SMOTE(random_state=42, k_neighbors=k_neighbors)
X_balanced, y_balanced = smote.fit_resample(X, y)

# Escalar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_balanced)

# Separar datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_balanced, test_size=0.3, random_state=42)

# Obtener las etiquetas de clase únicas en el conjunto de entrenamiento y prueba
labels = sorted(list(set(y_train.values.ravel())))

# Crear modelo de SVM
model = SVC(C=20,degree=3,kernel='rbf',gamma= 'scale',coef0= 2)

# Ajustar modelo con datos de entrenamiento
model.fit(X_train, y_train)

# hacer predicciones para conjunto de datos entrenamiento y prueba
train_predictions = model.predict(X_train)
test_predictions = model.predict(X_test)

# matriz de confusion del train y el test
from sklearn.metrics import [illegible], confusion_matrix
print('\nVisualizando la matriz de confusion del train:\n')
print(confusion_matrix(y_train, train_predictions))
print('\nVisualizando la matriz de confusion del test:\n')
print(confusion_matrix(y_test, test_predictions))

# calcular métricas para conjunto de entrenamiento
train_accuracy = accuracy_score(y_train, train_predictions)
global_train_precision = precision_score(y_train, train_predictions, average='macro')
train_precision = precision_score(y_train, train_predictions, average=None, labels=labels)
global_train_recall = recall_score(y_train, train_predictions, average='macro')
train_recall = recall_score(y_train, train_predictions, average=None, labels=labels)
global_train_f1 = f1_score(y_train, train_predictions, average='macro')
train_f1 = f1_score(y_train, train_predictions, average=None, labels=labels)

# calcular métricas para conjunto de prueba
test_accuracy = accuracy_score(y_test, test_predictions)
global_test_precision = precision_score(y_test, test_predictions, average='macro')
test_precision = precision_score(y_test, test_predictions, average=None, labels=labels)
global_test_recall = recall_score(y_test, test_predictions, average='macro')
test_recall = recall_score(y_test, test_predictions, average=None, labels=labels)
global_test_f1 = f1_score(y_test, test_predictions, average='macro')
test_f1 = f1_score(y_test, test_predictions, average=None, labels=labels)
```

Nota: Elaboración propia

Figura 47. Resultados de metricas de Support Vector Machines




```
Métricas de conjunto de entrenamiento con Support Vector Machines:
Exactitud: 0.9799
Precisión: 0.9799
Precisión (individual):
  - Para el índice de salud 1: 1.0000
  - Para el índice de salud 2: 0.9726
  - Para el índice de salud 3: 0.9632
  - Para el índice de salud 4: 0.9838
Sensibilidad: 0.9797
Sensibilidad (Individual):
  -  Para el índice de salud 1: 1.0000
  -  Para el índice de salud 2: 1.0000
  -  Para el índice de salud 3: 0.9536
  -  Para el índice de salud 4: 0.9650
F1-Score: 0.9797
F1-Score (individual):
  - Para el índice de salud 1: 1.0000
  - Para el índice de salud 2: 0.9861
  - Para el índice de salud 3: 0.9584
  - Para el índice de salud 4: 0.9743
Métricas de conjunto de prueba con Support Vector Machines:
Exactitud: 0.9757
Precisión: 0.9755
Precisión (Individual):
  - Para el índice de salud 1: 1.0000
  - Para el índice de salud 2: 0.9545
  - Para el índice de salud 3: 0.9779
  - Para el índice de salud 4: 0.9697
Sensibilidad: 0.9768
Sensibilidad (Individual):
  -  Para el índice de salud 1: 1.0000
  -  Para el índice de salud 2: 1.0000
  -  Para el índice de salud 3: 0.9301
  -  Para el índice de salud 4: 0.9771
F1-Score: 0.9759
F1-Score (individual):
  - Para el índice de salud 1: 1.0000
  - Para el índice de salud 2: 0.9767
  - Para el índice de salud 3: 0.9534
  - Para el índice de salud 4: 0.9734
```

Nota: Elaboración propia

De los resultados de las métricas calculadas podemos manifestar lo siguiente para el conjunto de datos de entrenamiento:

- La precisión global tiene el valor de 97.99% y la precisión individual de todas las categorías de salud es variada teniendo un mínimo de 96.32% para la categoría de salud tipo 3 y un valor máximo del 100% para la categoría de salud tipo 1, lo cual se puede corroborar en la matriz de confusión.
- La sensibilidad global tiene el valor de 97.97% y la sensibilidad individual de todas las categorías de salud es variada teniendo un mínimo de 95.36% para la categoría de salud tipo 3 y un valor máximo del 100% para la categoría de salud tipo 1 y 2, lo cual se puede corroborar también en la matriz de confusión.
- El F1-Score global tiene el valor de 97.97% y el F1-Score individual de todas las categorías de salud es variada teniendo un mínimo de 95.84% para la categoría de salud tipo 3 y un valor máximo del 100% para la categoría de salud tipo 1.

Respecto a las métricas para el conjunto de datos de prueba podemos manifestar lo siguiente:

- La precisión global tiene el valor de 97.55% y la precisión individual de cada categoría de salud obtienen métricas variadas siendo la más baja del 95.45% para categoría de salud tipo 2 y la más alta del 100% para las categorías de salud tipo 1
- La sensibilidad global tiene el valor de 97.68% y la sensibilidad individual de cada categoría de salud obtienen métricas variadas siendo la más baja del 93.01% para categoría de salud tipo 3 y la más alta del 100% para las categorías de salud tipo 1 y 2.

- El F1-Score global tiene el valor de 97.59% y el F1-Score individual de cada categoría de salud obtienen métricas variadas siendo la más baja del 95.34% para categoría de salud tipo 3 y la más alta del 100% para las categorías de salud tipo 1.

Ver formulación y mayor detalle de cada una de las métricas en el ítem 2.3.3 "Principales métricas usadas en modelos predictivos".

Modelado 4: K-Nearest Neighbors (KNN).-Este modelo clasifica nuevos puntos de datos basándose en los datos de entrenamiento más cercanos a ellos. Es un modelo puede tener dificultades en conjuntos de datos de alta dimensionalidad y grandes tamaños. En la figura 48 se muestra los hiperparametros óptimos encontrados para este modelo en función a nuestros datos, En la figura 49 se muestra el Modelado de datos mediante el algoritmo de "K-Nearest Neighbors" usando los hiperparametros óptimos, en la figura 50 la matriz de confusión y las métricas obtenidas.

Figura 48. Búsqueda de hiperparametros óptimos para K-Nearest Neighbors

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, RandomizedSearchCV
import pandas as pd
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE
import numpy as np

# Cargar el conjunto de datos desde Google Sheets
link = 'https://docs.google.com/spreadsheets/d/e/2PACX-1vInVaf1x6fRGlzlsVytZMwJnFvstgdIobhRa64IMB
dfx = pd.read_csv(link)
dataset = dfx.iloc[:, 1:]

# Separar caracteristicas (X) y etiquetas (y)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

# Realizar oversampling de las clases con menor cantidad de datos usando SMOTE
max_samples = max(y.value_counts())
k_neighbors = min(y.value_counts()) - 1
smote = SMOTE(random_state=42, k_neighbors=k_neighbors)
X_balanced, y_balanced = smote.fit_resample(X, y)

# Escalar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_balanced)

# Separar datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Definir los hiperparámetros y sus valores a explorar
param_dist = {
    'n_neighbors': np.arange(1, 10),  # Valores entre 1 y 9
    'weights': ['uniform', 'distance'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
    'p': [1, 2]
}

# Crear modelo de K-Nearest Neighbors
model = KNeighborsClassifier()

# Crear el objeto RandomizedSearchCV
random_search = RandomizedSearchCV(model, param_distributions=param_dist, n_iter=100, cv=5,
                                   scoring='f1', n_jobs=-1, random_state=42)

# Ajustar el objeto RandomizedSearchCV con los datos de entrenamiento escalados y balanceados
random_search.fit(X_scaled, y_balanced)

# Obtener el mejor modelo y sus hiperparametros
best_model = random_search.best_estimator_
best_params = random_search.best_params_

print("Mejores hiperparámetros:", best_params)
```

```
Mejores hiperparámetros: {'weights': 'distance', 'p': 1, 'n_neighbors': 3, 'algorithm': 'brute'}
```

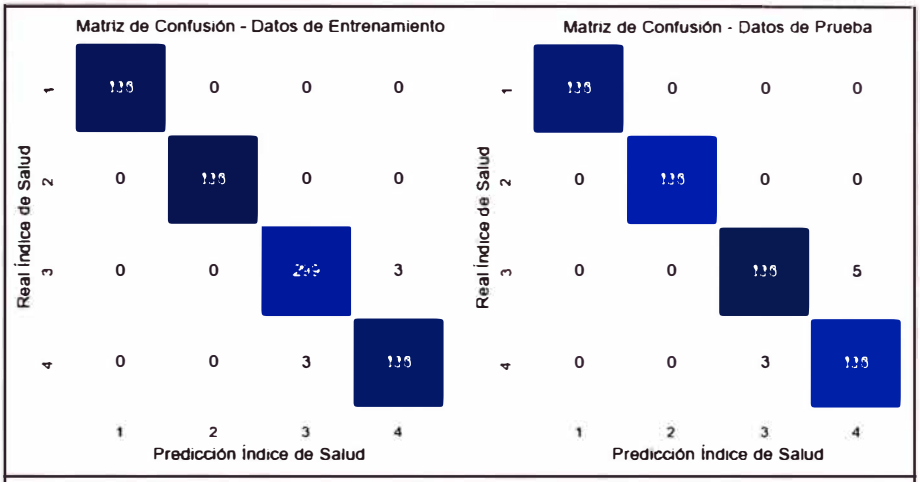
Nota: Elaboración propia

Figura 49. Modelado por K-Nearest Neighbors con parametros óptimos

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import pandas as pd
from imblearn.over_sampling import SMOTE
from pprint import pprint

# Cargar el conjunto de datos desde Google Sheets
link = 'https://docs.google.com/spreadsheets/d/e/2PACX-1vInVafIx6fRGlzlsVytZN8JnFvstgdIobhRa54IMBG-NknxE5z6
dfx = pd.read_csv(link)
dataset = dfx.iloc[:, 1:]

# Separar caracteristicas (X) y etiquetas (y)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

# Realizar oversampling de las clases con menor cantidad de datos usando SMOTE
max_samples = max(y.value_counts())
k_neighbors = min(y.value_counts()) - 1
smote = SMOTE(random_state=42, k_neighbors=k_neighbors)
X_balanced, y_balanced = smote.fit_resample(X, y)

# Escalar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_balanced)

# Separar datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_balanced, test_size=0.3, random_state=42)

# Obtener las etiquetas de clase únicas en el conjunto de entrenamiento y prueba
labels = sorted(list(set(y_train.values.ravel())))

# Crear modelo de K-Nearest Neighbors
model = KNeighborsClassifier(algorithm='brute',n_neighbors=3,p=1,weights='distance')

# Ajustar modelo con datos de entrenamiento
model.fit(X_train, y_train)

# hacer predicciones para conjunto de datos entrenamiento y prueba
train_predictions = model.predict(X_train)
test_predictions = model.predict(X_test)

# matriz de confusion del train y el test
from sklearn.metrics import [illegible], confusion_matrix
print('\nVisualizando la matriz de confusion del train:\n')
print(confusion_matrix(y_train, train_predictions))
print('\nVisualizando la matriz de confusion del test:\n')
print(confusion_matrix(y_test, test_predictions))

# calcular métricas para conjunto de entrenamiento
train_accuracy = accuracy_score(y_train, train_predictions)
global_train_precision = precision_score(y_train, train_predictions, average='macro')
train_precision = precision_score(y_train, train_predictions, average=None, labels=labels)
global_train_recall = recall_score(y_train, train_predictions, average='macro')
train_recall = recall_score(y_train, train_predictions, average=None, labels=labels)
global_train_f1 = f1_score(y_train, train_predictions, average='macro')
train_f1 = f1_score(y_train, train_predictions, average=None, labels=labels)

# calcular métricas para conjunto de prueba
test_accuracy = accuracy_score(y_test, test_predictions)
global_test_precision = precision_score(y_test, test_predictions, average='macro')
test_precision = precision_score(y_test, test_predictions, average=None, labels=labels)
global_test_recall = recall_score(y_test, test_predictions, average='macro')
test_recall = recall_score(y_test, test_predictions, average=None, labels=labels)
global_test_f1 = f1_score(y_test, test_predictions, average='macro')
test_f1 = f1_score(y_test, test_predictions, average=None, labels=labels)
```

Nota: Elaboración propia

Figura 50. Resultados de metricas K-Nearest Neighbors




```
Métricas de conjunto de entrenamiento con K-Nearest Neighbors:
Exactitud: 0.9952
Precisión: 0.9951
Precisión (individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 0.9901
  - Para el Índice de salud 4: 0.9904
Sensibilidad: 0.9951
Sensibilidad (Individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 0.9901
  - Para el Índice de salud 4: 0.9904
F1-Score: 0.9951
F1-Score (individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 0.9901
  - Para el Índice de salud 4: 0.9904

Métricas de conjunto de prueba con K-Nearest Neighbors:
Exactitud: 0.9850
Precisión: 0.9853
Precisión (Individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 0.9787
  - Para el Índice de salud 4: 0.9624
Sensibilidad: 0.9855
Sensibilidad (Individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 0.9650
  - Para el Índice de salud 4: 0.9771
F1-Score: 0.9854
F1-Score (individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 0.9718
  - Para el Índice de salud 4: 0.9697
```

Nota: Elaboración propia

De los resultados de las métricas calculadas podemos manifestar lo siguiente para el conjunto de datos de entrenamiento:

- La precisión global tiene el valor de 99.51% y la precisión individual de todas las categorías de salud es mayor al 99% teniendo un mínimo de 99.01% para la categoría de salud tipo 3 y un valor máximo del 100% para la categoría de salud tipo 1 y 2, lo cual se puede corroborar en la matriz de confusión.
- La sensibilidad global tiene el valor de 99.51% y la sensibilidad individual de todas las categorías de salud es mayor al 99% teniendo un mínimo de 99.01% para la categoría de salud tipo 3 y un valor máximo del 100% para la categoría de salud tipo 1 y 2, lo cual se puede corroborar también en la matriz de confusión.
- El F1-Score global tiene el valor de 99.51% y el F1-Score individual de todas las categorías de salud es mayor al 99% teniendo un mínimo de 99.01% para la categoría de salud tipo 3 y un valor máximo del 100% para la categoría de salud tipo 1 y 2, lo cual se puede corroborar también en la matriz de confusión.

Respecto a las métricas para el conjunto de datos de prueba podemos manifestar lo siguiente:

- La precisión global tiene el valor de 98.53% y la precisión individual de cada categoría de salud obtienen métricas variadas, siendo la más baja del 96.24% para categoría de salud tipo 4 y la más alta del 100% para las categorías de salud tipo 1 y 2.
- La sensibilidad global tiene el valor de 98.55% y la sensibilidad individual de cada categoría de salud obtienen métricas variadas, siendo la más baja del 96.50% para categoría de salud tipo 3 y la más alta del 100% para las categorías de salud tipo 1 y 2.

- El F1-Score global tiene el valor de 98.54% y el F1-Score individual de cada categoría de salud obtienen métricas variadas, siendo la más baja de 96.97% para categoría de salud tipo 4 y la más alta del 100% para las categorías de salud tipo 1 y 2.

Ver formulación y mayor detalle de cada una de las métricas en el ítem 2.3.3 "Principales métricas usadas en modelos predictivos".

Modelado 5: Redes Neuronales Artificiales (ANN).- es un modelo de aprendizaje automático inspirado en el funcionamiento del cerebro humano. Están compuestas por capas de nodos interconectados que procesan la información de entrada y generan una salida a través de una función de activación. En la figura 51 se muestra los hiperparametros óptimos encontrados para este modelo en función a nuestros datos, En la figura 52 se muestra el Modelado de datos mediante el algoritmo de Redes Neuronales Artificiales usando los hiperparametros óptimos, en la figura 53 la matriz de confusión y las métricas obtenidas.

Figura 51. Búsqueda de hiperparametros óptimos con Redes Neuronales Artificiales

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import pandas as pd
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import RandomizedSearchCV
from scipy import stats

# Cargar el conjunto de datos desde Google Sheets
link = 'https://docs.google.com/spreadsheets/d/e/2PACX-1vInVafIx6fRGlzlsVytZNWJnFvstgdlobhRa54IMBG-NknxE5z6nCGBRtf6xzdPXT9dfi7d5

dfx = pd.read_csv(link)
dataset = dfx.iloc[:, 1:]

# Separar caracteristicas (X) y etiquetas (y)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

# Realizar oversampling de las clases con menor cantidad de datos usando SMOTE
max_samples = max(y.value_counts())
k_neighbors = min(y.value_counts()) - 1
smote = SMOTE(random_state=42, k_neighbors=k_neighbors)
X_balanced, y_balanced = smote.fit_resample(X, y)

# Escalar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_balanced)

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_balanced, test_size=0.3, random_state=42)

# Definir las distribuciones para los hiperparametros
param_dist = {
    'hidden_layer_sizes': [(100,), (50, 50), (50, 100), (100, 50), (50,)],
    'max_iter': stats.randint(100, 300),  # Distribucion uniforme entre 100 y 300
    'alpha': [0.0001, 0.001, 0.01],  # Valores especificos
    'solver': ['adam', 'sgd'],
}

# Crear modelo de Redes Neuronales Artificiales
model = MLPClassifier(random_state=42)

# Crear el objeto RandomizedSearchCV
random_search = RandomizedSearchCV(model, param_distributions=param_dist, n_iter=100, cv=5, scoring='f1',
                                   n_jobs=-1, random_state=42)

# Ajustar el objeto RandomizedSearchCV con los datos de entrenamiento escalados y balanceados
random_search.fit(X_train, y_train)

# Obtener el mejor modelo y sus hiperparametros
best_model = random_search.best_estimator_
best_params = random_search.best_params_

print("Mejores hiperparametros con RandomizedSearchCV:", best_params)
```

```
c:\Users\ahuaman\python\lib\site-packages\sklearn\model_selection\_search.py:972: UserWarning: One or more of the test scores are n
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan]
  category=UserWarning,
Mejores hiperparametros con RandomizedSearchCV: {'alpha': 0.01, 'hidden_layer_sizes': (100, 50), 'max_iter': 192, 'solver': 'adam'}
```

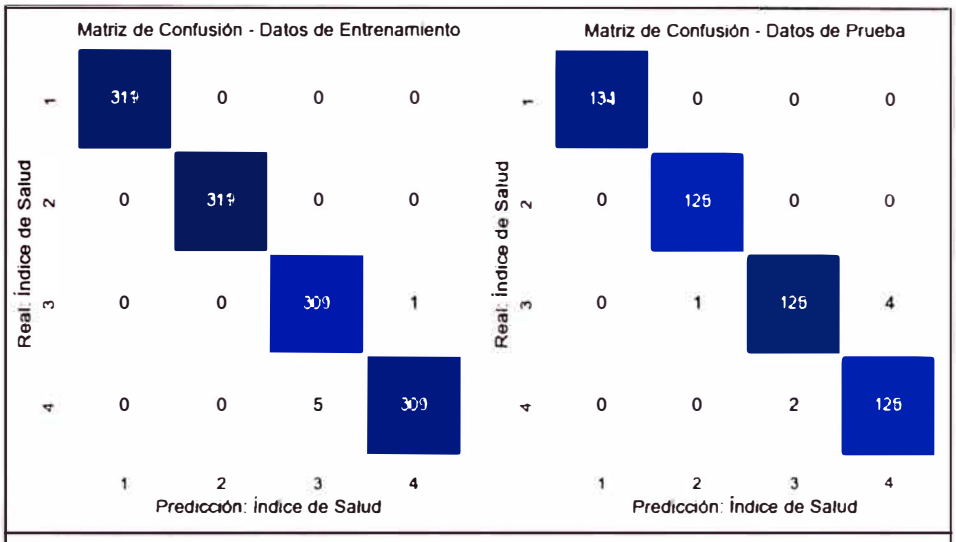
Nota: Elaboración propia

Figura 52. Modelado por Redes Neuronales Artificiales con parametros óptimos

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import pandas as pd
from imblearn.over_sampling import SMOTE
from pprint import pprint

# Cargar el conjunto de datos desde Google Sheets
link = 'https://docs.google.com/spreadsheets/d/e/2PACX-1vInVaf1x6fRGlzlsVytZMWJnfvstgdIobhRa64IMBG-NkrxE5z6nCGE
dfx = pd.read_csv(link)
dataset = dfx.iloc[:, 1:]

# Separar caracteristicas (X) y etiquetas (y)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

# Realizar oversampling de las clases con menor cantidad de datos usando SMOTE
max_samples = max(y.value_counts())
k_neighbors = min(y.value_counts()) - 1
smote = SMOTE(random_state=42, k_neighbors=k_neighbors)
X_balanced, y_balanced = smote.fit_resample(X, y)

# Escalar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_balanced)

# Separar datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_balanced, test_size=0.3, random_state=42)

# Obtener las etiquetas de clase únicas en el conjunto de entrenamiento y prueba
labels = sorted(list(set(y_train.values.ravel())))

# Crear modelo de Redes Neuronales Artificiales
model = MLPClassifier(alpha=0.01, hidden_layer_sizes=(100,50), max_iter=192, solver='adam', random_state=42)

# Ajustar modelo con datos de entrenamiento
model.fit(X_train, y_train)

# hacer predicciones para conjunto de datos entrenamiento y prueba
train_predictions = model.predict(X_train)
test_predictions = model.predict(X_test)

# matriz de confusion del train y el test
from sklearn.metrics import [illegible], confusion_matrix
print('\nVisualizando la matriz de confusión del train:\n')
print(confusion_matrix(y_train, train_predictions))
print('\nVisualizando la matriz de confusión del test:\n')
print(confusion_matrix(y_test, test_predictions))

# calcular métricas para conjunto de entrenamiento
train_accuracy = accuracy_score(y_train, train_predictions)
global_train_precision = precision_score(y_train, train_predictions, average='macro')
train_precision = precision_score(y_train, train_predictions, average=None, labels=labels)
global_train_recall = recall_score(y_train, train_predictions, average='macro')
train_recall = recall_score(y_train, train_predictions, average=None, labels=labels)
global_train_f1 = f1_score(y_train, train_predictions, average='macro')
train_f1 = f1_score(y_train, train_predictions, average=None, labels=labels)

# calcular métricas para conjunto de prueba
test_accuracy = accuracy_score(y_test, test_predictions)
global_test_precision = precision_score(y_test, test_predictions, average='macro')
test_precision = precision_score(y_test, test_predictions, average=None, labels=labels)
global_test_recall = recall_score(y_test, test_predictions, average='macro')
test_recall = recall_score(y_test, test_predictions, average=None, labels=labels)
global_test_f1 = f1_score(y_test, test_predictions, average='macro')
test_f1 = f1_score(y_test, test_predictions, average=None, labels=labels)
```

Nota: Elaboración propia

Figura 53. Resultados de metricas Redes Neuronales Artificiales




```
Métricas de conjunto de entrenamiento con Redes Neuronales Artificiales:
Exactitud: 0.9952
Precisión: 0.9951
Precisión (individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 0.9837
  - Para el Índice de salud 4: 0.9968
Sensibilidad: 0.9952
Sensibilidad (Individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 0.9967
  - Para el Índice de salud 4: 0.9841
F1-Score: 0.9951
F1-Score (individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 0.9901
  - Para el Índice de salud 4: 0.9904

Métricas de conjunto de prueba con Redes Neuronales Artificiales:
Exactitud: 0.9869
Precisión: 0.9869
Precisión (Individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 0.9921
  - Para el Índice de salud 3: 0.9857
  - Para el Índice de salud 4: 0.9699
Sensibilidad: 0.9874
Sensibilidad (Individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 0.9650
  - Para el Índice de salud 4: 0.9847
F1-Score: 0.9871
F1-Score (individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 0.9960
  - Para el Índice de salud 3: 0.9753
  - Para el Índice de salud 4: 0.9773
```

Nota: Elaboración propia

De los resultados de las métricas calculadas podemos manifestar lo siguiente para el conjunto de datos de entrenamiento:

- La precisión global tiene el valor de 99.51% y la precisión individual de todas las categorías de salud es variada teniendo un mínimo de 98.37% para la categoría de salud tipo 3 y un valor máximo del 100% para la categoría de salud tipo 1 y 2, lo cual se puede corroborar en la matriz de confusión.
- La sensibilidad global tiene el valor de 99.52% y la sensibilidad individual de todas las categorías de salud es variada teniendo un mínimo de 98.41% para la categoría de salud tipo 4 y un valor máximo del 100% para la categoría de salud tipo 1 y 2, lo cual se puede corroborar también en la matriz de confusión.
- El F1-Score global tiene el valor de 99.51% y el F1-Score individual de todas las categorías de salud es mayor al 99%, teniendo un mínimo de 99.01% para la categoría de salud tipo 3 y un valor máximo del 100% para la categoría de salud tipo 1 y 2.

Respecto a las métricas para el conjunto de datos de prueba podemos manifestar lo siguiente:

- La precisión global tiene el valor de 98.69% y la precisión individual de cada categoría de salud obtienen métricas variadas siendo la más baja de 96.99% para categoría de salud tipo 4 y la más alta del 100% para las categorías de salud tipo 1
- La sensibilidad global tiene el valor de 98.74% y la sensibilidad individual de cada categoría de salud obtienen métricas variadas siendo la más baja de 96.50% para categoría de salud tipo 3 y la más alta del 100% para las categorías de salud tipo 1 y 2.

- El F1-Score global tiene el valor de 98.71% y el F1-Score individual de cada categoría de salud obtienen métricas variadas siendo la más baja del 97.53% para categoría de salud tipo 3 y la más alta del 100% para las categorías de salud tipo 1.

Ver formulación y mayor *detalle* de cada una de las métricas en el ítem 2.3.3 "Principales métricas usadas en modelos predictivos".

CAPÍTULO IV. EVALUACIÓN DE LOS MODELOS PREDICTIVOS Y CONTRASTACIÓN DE LAS HIPÓTESIS

4.1. EVALUACIÓN DE LOS MODELOS PREDICTIVOS

De acuerdo con las métricas obtenidas durante el proceso de entrenamiento y prueba de cada uno modelos desarrollados se elaboró el siguiente cuadro resumen de la Tabla 3, considerando el indicador F1-Score, dado que esta combina en una sola métrica la precisión y sensibilidad tomando en cuenta tanto los falsos positivos y falsos negativos en su cálculo.

Tabla 3. *Cuadro Resumen del "F1-Score" de cada modelo predictivo en los conjuntos de Entrenamiento y Prueba.*

Modelo	Metrica F1-Score "global" Conjunto de Entrenamiento	Metrica F1-Score "global" Conjunto de Prueba
1. Random Forest Classifier	99.92%	99.63%
2. Gradient Boosting Classifier	99.11%	98.71%
3. Support Vector Machines	97.97%	97.59%
4. K-Nearest Neighbors	99.51%	98.54%
5. Redes Neuronales Artificiales	99.51%	98.71%

Nota: Elaboración propia

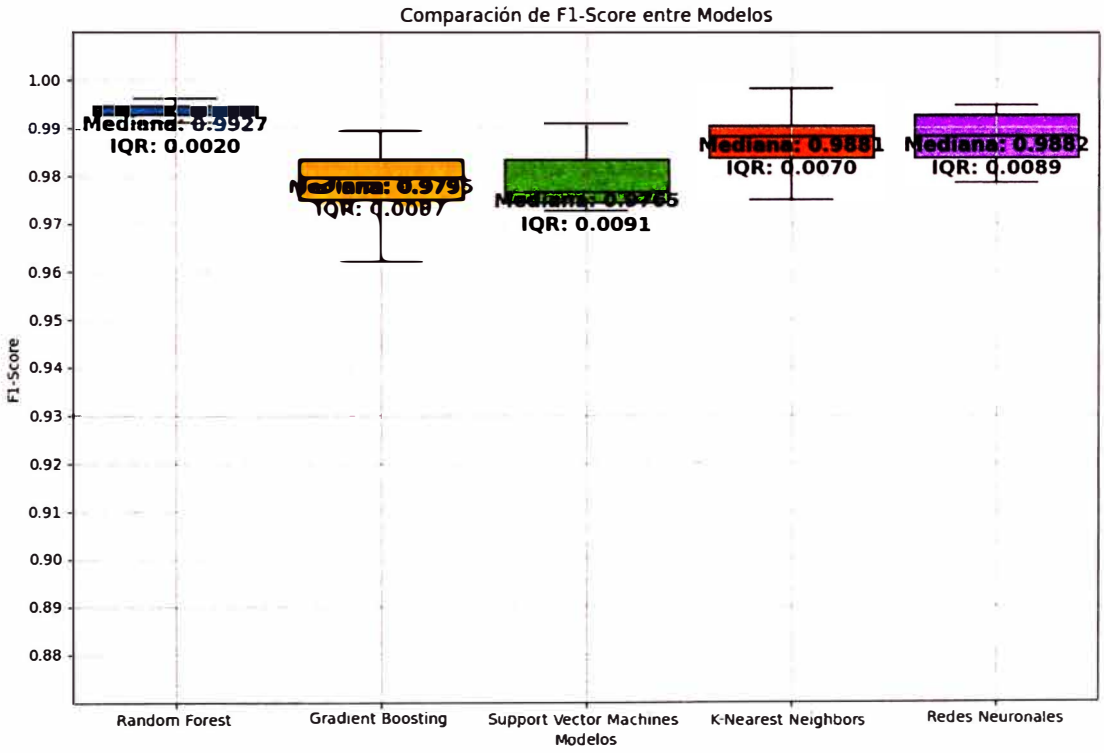
De la tabla podemos afirmar que el modelo "Random Forest Classifier" es el que mejor rendimiento ha mostrado tanto con los datos de entrenamiento y los datos de Prueba, pero seguido muy de cerca de los otros modelos desarrollados y el modelo que más se aleja de ese rendimiento es el de Support Vector Machines.

Dado que la evaluación de esta manera si limita a un conjunto de datos único, el cual fue definido durante el entrenamiento y prueba de cada modelo, no se puede medir la variabilidad de este indicador en diferentes conjuntos de datos, siendo en ese sentido conveniente probar el rendimiento de los

modelos en diferentes sorteos de datos, para tener mayor certeza de la efectividad de los modelos.

En ese sentido se evaluará los modelos predictivos analizando la variabilidad de la métrica "F1-Score" en 10 sorteos de datos. Para fines del presente análisis se realizó un graficó de boxplot estadístico con los resultados de la métrica "F1-Score". En la figura 54 se muestra los resultados obtenidos: valores de la mediana y el intercuartil estadístico "IQR" (diferencia entre el percentil 75 y el percentil 25, el cual mide la desviación de los resultados enmarcado en ese 50% central).

Figura 54. Comparación del "F1-Score" entre modelos en 10 sorteos de datos



Nota: Elaboración propia

Adicionalmente para mayor análisis en la evaluación de los modelos se elaboró un cuadro estadístico con la evolución de la métrica F1-Score en diferentes percentiles estadísticos, además se calculó la desviación estándar de todos los resultados (std) respecto a la media (mean) y los valores mínimos y máximos alcanzados por cada modelo predictivo en los 10 sorteos de datos realizados. Ver resultados en la tabla 4

Tabla 4. *Cuadro de percentiles estadístico del "F1-Score" de cada modelo predictivo en 10 sorteos de datos*

	count	mean	std	min	10%	30%	50%	70%	90%	max
Random Forest	10.0	0.9934	0.0015	0.9912	0.9922	0.9926	0.9927	0.9945	0.9949	0.9961
Gradient Boosting	10.0	0.9789	0.0081	0.9619	0.9727	0.9758	0.9795	0.9823	0.9888	0.9893
Support Vector Machines	10.0	0.9789	0.0060	0.9727	0.9735	0.9747	0.9765	0.9822	0.9853	0.9908
K-Nearest Neighbors	10.0	0.9875	0.0075	0.9748	0.9784	0.9856	0.9881	0.9895	0.9982	0.9982
Redes Neuronales	10.0	0.9877	0.0059	0.9786	0.9795	0.9853	0.9882	0.9917	0.9945	0.9945

Nota: Elaboración propia

El código en Python implementado para realizar el proceso de sortear los datos en 10 agrupamientos y evaluar el rendimiento del F1-Score en cada sorteo podemos visualizarlo en la figura 60

Figura 55. *Proceso de sortear los datos y evaluar el rendimiento del "F1-Score"*

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler, LabelEncoder

# Cargar el conjunto de datos desde Google Sheets
link = 'https://docs.google.com/spreadsheets/d/e/2PACX-1vInVafIxbFRGlzlsVytZMWJnFvstgdIobbRab4IMBG-NknxE5z6nCGBRtI6kzdPXIR0fi7u5tneuy
dfx = pd.read_csv(link)
dataset = dfx.iloc[:, 1:]

# Separar caracteristicas (X) y etiquetas (y)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

# Convertir etiquetas de clase a valores numericos consecutivos comenzando desde 0
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Realizar Balanceo de dato con oversampling de las clases con menor cantidad de datos
k_neighbors = min(pd.Series(y_encoded).value_counts()) - 1
smote = SMOTE(random_state=42, k_neighbors=k_neighbors)
X_balanced, y_balanced = smote.fit_resample(X, y_encoded)

# Escalar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_balanced)

# Inicializar modelos
models = {
    "Random Forest ": RandomForestClassifier(max_depth=10,min_samples_leaf=3,min_samples_split=5, n_estimators=25,random_state=42),
    "Gradient Boosting ": XGBClassifier(learning_rate=0.01,max_depth=15,min_child_weight=4,n_estimators= 100,random_state=42),
    "Support Vector Machines ": SVC(C=20,degree=3,kernel='rbf',gamma= 'scale',coef0= 2),
    "K-Nearest Neighbors ": KNeighborsClassifier(algorithm='brute',n_neighbors=3,p=1,weights='distance'),
    "Redes Neuronales": MLPClassifier(alpha=0.01, hidden_layer_sizes=(100,50), max_iter=192, solver='adam', random_state=42)
}

# Repetir el proceso de entrenamiento y evaluacion en 10 sorteos
num_draws = 10
results_list = []

for draw in range(num_draws):
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_balanced, test_size=0.3, random_state=draw)

    results = {}
    for model_name, model in models.items():
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        f1 = f1_score(y_test, y_pred, average="macro")  # Macro F1-score
        results[model_name] = f1

    results_list.append(results)

# Crear un DataFrame con los resultados y graficar
results_df = pd.DataFrame(results_list)
plt.figure(figsize=(12, 8))
ax = sns.boxplot(data=results_df)
```

Nota: Elaboración propia

4.2. SELECCIÓN DEL MODELO PREDICTIVO ÓPTIMO

Para seleccionar el modelo predictivo óptimo analizaremos los resultados estadístico anteriores y en ese sentido de acuerdo con los datos representados en el gráfico del boxplot de la Figura 59, podemos observar que el modelo con una mediana mejor ubicada para la métrica del F1-Score en los 10 sorteos de datos realizados, es el modelo de Random Forest, con un valor de 0.9927, luego el modelo de Redes Neuronales con un valor de 0.9882, muy de cerca el modelo K-Nearest Neighbord con un valor de 0.9881, continuando el modelo predictivo de Gradiente de Boosting con una mediana de 0.9795 y finalmente seguido muy cerca el modelo de Support Vector Machines con una mediana de 0.9765.

Del mismo gráfico del boxplot de la Figura 59, podemos observar que el modelo predictivo con mejor intercuartil estadístico (IQR) es el algoritmo de Random Forest con un valor de 0.0020, este algoritmo obtiene una clara ventaja sobre los otros modelos que casi cuadriplican esta métrica de dispersión, empezando con el modelo de K-Nearest Neighbord teniendo un valor de 0.0070, Luego el modelo de Gradiente de Boosting con un valor de 0.0087, seguido del modelo de Redes Neuronales con un valor de 0.0089 y finalmente el modelo de Support Vector Machines con un valor de 0.0091.

De acuerdo con los valores mostrados en el cuadro estadístico de la tabla 2 podemos observar que el modelo con menor desviación estándar, el cual diferencia del intercuartil estadístico, en que este parámetro considera la dispersión del total de datos analizados y en ese sentido es sensible a resultados atípicos que se pudieran presentar. Para esta métrica se observa que también el Random Forest obtiene el mejor valor con 0.0015, luego el

modelo de Redes Neuronales con un valor de 0.0059, seguido del modelo de Support Vector Machines con un valor de 0.0060, continuando el modelo K-Nearest Neighbord con un valor de 0.0075 y finalmente la desviación estándar mayor del modelo Gradiente de Boosting con 0.0081.

También podemos observar del cuadro estadístico de la tabla 2 que el modelo predictivo con mejor valor promedio y mejor valor mínimo respecto a la métrica del F1-Score, es el Randon Forest con el valor de 0.9934 y 0.9912 respectivamente. El Modelo predictivo con mejor máximo resultó ser el modelo de K-Nearest Neighbord con el valor de 0.9982 pero muy de cerca el modelo de Randon Forest con el valor de 0.9961.

A partir de estos resultados, se puede concluir que, para los datos analizados, el modelo predictivo más óptimo es el de Random Forest Classifier. Sin embargo, los demás modelos también exhiben métricas de rendimiento favorables y se encuentran en proximidad al rendimiento del modelo Random Forest.

4.3. ANÁLISIS DE TIEMPOS, COSTOS Y BENEFICIOS

Hasta este punto de la investigación se ha centrado los esfuerzos en el desarrollado del objetivo de aumentar la efectividad del diagnóstico del Índice de Salud, en base a modelos predictivos, debido al sustento metodológico y científico que esto conlleva, sin embargo, en esta sección, desarrollaremos los otros objetivos planteados para finalmente contrastar todas las hipótesis específicas planteadas.

4.3.1. ANÁLISIS DE TIEMPOS

Durante la etapa del desarrollo de los modelos predictivos con machine learning se empleó tiempos para la investigación, análisis y tratamiento de datos, modelado de algoritmos, pruebas y análisis de métricas, el cual podemos ver en la tabla 5.

Tabla 5. *Cuadro resumen de tiempo empleado para el desarrollo de algoritmo predictivo.*

ítem	Tarea	Tiempo invertido (meses)
1	Investigación	2
2	Análisis y tratamiento de datos	1
3	Modelado de algoritmos	2
4	Pruebas y análisis de métricas	1
	Total	6

Cabe mencionar que los tiempos mostrados no involucraron una dedicación exclusiva sino a tiempo parcial y que se realiza por única vez y en ese sentido ya teniendo los modelos predictivos desarrollados y probados, el tiempo para su utilización es mínimo. Básicamente comprende la actualización de la base de datos de interruptores, extracción de sus variables monitoreadas, ejecución del programa y revisión de resultados este tiempo puede tomar de 3 a 7 días en base a la experiencia del autor. La variación es por si se necesita reentrenar el modelo ante alguna situación específica como por ejemplo algún resultado no coherente producto de una combinación de variables que escapa al conocimiento con el cual fue entrenado inicialmente el modelo. El reentrenamiento en modelos de machine learning supervisados es

parte de su mejora continua, debido a que mientras más combinaciones de casos tenga en su aprendizaje (datos etiquetados) este será más robusto aumentando su efectividad, precisión y sensibilidad antes nuevos casos.

Justamente esta es la principal ventaja de modelos predictivos de machine learning comparadas con modelos de lógicas fuzzy mientras que al primero basta con darle nuevos datos etiquetados y reentrenarlo con el algoritmo ya desarrollado, al segundo ante algún resultado no coherente se debe revisar y analizar cada de una las lógicas de inferencia existentes (1024 sentencias para las 5 variables) e incluso también involucra revisar y modificar las reglas de parametrización triangular fuzzy definidas para clasificar cada variable (función de membresía). En ese sentido en base a la experiencia del autor durante los años 2021, 2022 y 2023 que se han venido realizando los diagnósticos de Salud con el modelo fuzzy se ha incurrido en un tiempo de 4, 3 y 2 semanas respectivamente. La disminución de tiempo se logró en base a la misma experiencia adquirida año tras año y las mejoras que se pudieron plantear para optimizar este proceso.

4.3.2. ANÁLISIS DE COSTOS Y BENEFICIOS

Para analizar el impacto en los costos de la empresa debido a un diagnóstico oportuno y certero del índice de salud de los interruptores de potencia usando modelos predictivos de machine learning, debemos considerar lo siguiente:

- Inversión
- Costos
- Beneficios

En ese sentido se realiza primero un análisis completo de costos directos e indirecto asociado a la falla de un interruptor que pertenece al sistema eléctrico de potencia, el cual se puede visualizar en el Tabla 6

Tabla 6. *Análisis de costos por falla de un interruptor.*

Análisis de costos y afectaciones por falla de un interruptor de Potencia	USD	
Costo promedio de un Interruptor de Potencia	$35,000.00	
Costo de traslado, montaje y pruebas	$15,000.00	
Costo total Directo por falla de interruptor	$50,000.00	
Costos Indirectos por falla de interruptor*	$25,000.00	50%
Costo Total por Falla	$75,000.00	

* Compesaciones a los agentes del SEIN por Energia no suministrada debido a corte imprevisto, sanciones del OSINERGMIN por indisponibilidad prolongada del interruptor, Multas de la OEFA por afectación al medio ambiente debido a Fuga de SF6, Multas del ministerio del Trabajo por afectación a la vida o salud del personal, Daño a la imagen y reputación empresarial.

Del monto total mostrado, se considerará solo los costos indirectos: 25 000USD, como posible ahorro, dado que el diagnostico por sí solo, no evitará la avería y su necesidad de reemplazo, pero si evitará que esto suceda de manera intempestiva provocando otras afectaciones, es decir podremos tener un mantenimiento correctivo planificado en vez de un mantenimiento correctivo de emergencia.

Para el análisis técnico y económico se definió una ventana de 10 años. Y se proyectó el parque de interruptores en ese horizonte con una tasa de crecimiento del 5%, se

proyectó también la probabilidad de falla de los interruptores con la misma tasa, partiendo de un valor de referencia de 0.38% que corresponde a un análisis estadístico del año 2022.

La efectividad de los diagnósticos también se proyecta, dado que los modelos deben ajustarse o reentrenarse al menos una vez al año, mejorando sus métricas. Se tomó como línea base los valores calculados y se asumió para los 2 siguientes años un aumento de 1% para el modelo "Fuzzy" y un aumento de solo el 0.1% para el modelo "Machine Learning" (debido a que ya se encuentra en un valor superior al 99%) y luego hasta el año 5 un aumento del 0.05% para ambos modelos y del año 6 hacia adelante un aumento mínimo del 0.01%

*E*l ahorro *anual* se *estimó* de la siguiente manera:

$$Ahorro\ anual = P \times N \times C \times E \qquad (19)$$

Donde:

- P: Probabilidad de falla
- *N*: Cantidad de Interruptores
- C: Costo *Indirecto por falla*
- E: Efectividad del diagnostico

Los resultados de las proyecciones y los cálculos *de ahorro se* pueden visualizar en la tabla *7*.

Tabla 7. *Cuadro resumen de Proyecciones y cálculos de ahorro.*

			Efectividad Diagnóstico		Ahorro de O&M		
Año	Cantidad de interruptores	Probabilidad de falla	Modelado con Lógica Fuzzy	Modelado con Machine learning	Modelado con Lógica Fuzzy	Modelado con Machine learning	
Año 0	530	0.38%	96.27%	99.27%	-	-	Calculado
Año 1	557	0.40%	97.27%	99.37%	$53,569	$54,726	Proyectado
Año 2	584	0.42%	98.27%	99.47%	$59,667	$60,396	
Año 3	614	0.44%	98.32%	99.52%	$65,817	$66,620	
Año 4	644	0.46%	98.37%	99.57%	$72,600	$73,485	
Año 5	676	0.48%	98.42%	99.62%	$80,082	$81,058	
Año 6	710	0.51%	98.43%	99.63%	$88,299	$89,376	
Año 7	746	0.53%	98.44%	99.64%	$97,360	$98,546	
Año 8	783	0.56%	98.45%	99.65%	$107,350	$108,658	
Año 9	822	0.58%	98.46%	99.66%	$118,365	$119,808	
Año 10	863	0.61%	98.47%	99.67%	$130,511	$132,101	

Con miras a realizar un análisis económico de flujo de caja se establecen los costos de inversión por cada modelo, los costos periódicos de revisión y ajuste de los modelos y los beneficios anuales, que podemos observar en la tabla 8

Tabla 8. *Cuadro resumen de inversión, costos y beneficios.*

		Metodologia		
Item	Analisis de inversión, costos y beneficios	Logica Fuzzy	Machine Learning	frecuencia
1	Inversión para el desarrollo e implementación de modelo de diagnostico	3,500.00	5,000.00	Unica vez
2	Costos directos asociados a la revisión, ajuste y reentranamientro del modelo	1,000.00	500.00	Anualmente
3	Ahorro de Costos por Compensaciones, sanciones, multas y afectaciones	$53,569	$54,726	Año 1
4	Ahorro de Costos por Compensaciones, sanciones, multas y afectaciones	$59,667	$60,396	Año 2
5	Ahorro de Costos por Compensaciones, sanciones, multas y afectaciones	$65,817	$66,620	Año 3
6	Ahorro de Costos por Compensaciones, sanciones, multas y afectaciones	$72,600	$73,485	Año 4
7	Ahorro de Costos por Compensaciones, sanciones, multas y afectaciones	$80,082	$81,058	Año 5
8	Ahorro de Costos por Compensaciones, sanciones, multas y afectaciones	$88,299	$89,376	Año 6
9	Ahorro de Costos por Compensaciones, sanciones, multas y afectaciones	$97,360	$98,546	Año 7
10	Ahorro de Costos por Compensaciones, sanciones, multas y afectaciones	$107,350	$108,658	Año 8
11	Ahorro de Costos por Compensaciones, sanciones, multas y afectaciones	$118,365	$119,808	Año 9
12	Ahorro de Costos por Compensaciones, sanciones, multas y afectaciones	$130,511	$132,101	Año 10

4.4. CONTRASTACIÓN DE LAS HIPÓTESIS

En este apartado se contrastará cada una de las hipótesis especificas planteadas en ítem 1.8.2 de la presente investigación, para finalmente validar la hipótesis general planteada.

Hipótesis 1: "Se aumentará la efectividad del diagnóstico del índice de salud diseñando modelos predictivos basado en machine learning en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP".

Para contrastar esta hipótesis planteada se ha elaborado la tabla 9 con el escenario analizado que podemos ver a continuación:

Tabla 9. *Cuadro contrastación Hipótesis 1.*

		Escenario Conservador (%)
Metodología actual	Efectividad del diagnóstico del índice de salud usando: "reglas de parametrización por cada variable y lógicas de inferencias Fuzzy con todas las combinaciones posible". ** En los últimos 3 años (2023,2022 y 2021) el estudio de índice de salud mediante lógica fuzzy se ha tenido una efectividad en los resultados del 95%, 97% y 98% respectivamente (Promedio = 96.66%)*	*96.66%
Metodología propuesta	Efectividad del diagnóstico del índice de salud usando: "modelos predictivos de machine learning con el algoritmo de Randon Forest Clasifier". **De acuerdo con el percentil 50 (mediana) de los resultados de la métrica F1-Score registrada en los 10 sorteos de datos*	*99.27%
	Diferencia	2.61%

Hipótesis 2: "Se disminuirá el tiempo empleado para el diagnóstico del índice de salud diseñando modelos predictivos basado en machine learning en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP".

Para contrastar esta hipótesis planteada se ha elaborado la tabla 10 con los distintos escenarios analizados que podemos ver a continuación.

Tabla 10. *Cuadro contrastación Hipótesis 2.*

		Escenario Optimista (días)	Escenario Conservador (días)	Escenario Pesimista (días)
Metodología actual	Tiempo necesario para diagnosticar el índice de salud usando: "reglas de parametrización por cada variable y lógicas de inferencias Fuzzy con todas las combinaciones posible".	14	21	28
Metodología propuesta	Tiempo necesario para diagnosticar el índice de salud usando: "modelos predictivos de machine learning con aprendizaje supervisado"	3	5	7
	Diferencia	11	16	21

Nota: Elaboración propia

Del cuadro comparativo podemos observar que en todos los escenarios analizados hay una disminución de los tiempos empleados para diagnosticar el índice de Salud respecto a la actual metodología. Por lo tanto, queda validado la primera hipótesis.

Hipótesis 3: "Se optimizará los costos incurridos por fallas imprevistas diseñando modelos predictivos basado en machine learning en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP".

Para contrastar esta hipótesis planteada, se establece una tasa de descuento del 12% y se calcula el Valor Actual Neto (VAN) en un horizonte de 10 años. Bajo los considerandos y supuestos descritos en el ítem 4.3.2 Análisis de

costos y beneficios . Donde se demuestra que ambos proyectos son muy beneficios para la empresa, sin embargo, implementar el Proyecto con la metodología propuesta de machine learning generará más beneficios económicos a la empresa por los mayores ahorros antes fallas imprevistas de interruptores, tal como se observa en la tabla 11 con los flujos de caja de cada de cada proyecto

Tabla 11. *Cuadro contrastación Hipótesis 3.*

	Flujo de caja										
Caso 1: Metodologia actual	**Año 0**	**Año 1**	**Año 2**	**Año 3**	**Año 4**	**Año 5**	**Año 6**	**Año 7**	**Año 8**	**Año 9**	**Año 10**
INVERSIÓN (USD)	$3,500										
COSTOS (USD)		$1,000	$1,000	$1,000	$1,000	$1,000	$1,000	$1,000	$1,000	$1,000	$1,000
BENEFICIOS (USD)		$53,569	$59,667	$65,817	$72,600	$80,082	$88,299	$97,360	$107,350	$118,365	$130,511
FLUJO NETO (USD)	$3,500	$52,569	$58,667	$64,817	$71,600	$79,082	$87,299	$96,360	$106,350	$117,365	$129,511

Tasa de descuento	12%
VAN , Caso 1 (USD)	$441,509

	Flujo de caja										
Caso 2: Metodologia Propuesta	**Año 0**	**Año 1**	**Año 2**	**Año 3**	**Año 4**	**Año 5**	**Año 6**	**Año 7**	**Año 8**	**Año 9**	**Año 10**
INVERSIÓN (USD)	$5,000										
COSTOS (USD)		$500	$500	$500	$500	$500	$500	$500	$500	$500	$500
BENEFICIOS (USD)		$54,726	$60,396	$66,620	$73,485	$81,058	$89,376	$98,546	$108,658	$119,808	$132,101
FLUJO NETO (USD)	$5,000	$54,226	$59,896	$66,120	$72,985	$80,558	$88,876	$98,046	$108,158	$119,308	$131,601

Tasa de descuento	12%
VAN , Caso 2 (USD)	$448,779

VAN2 > VAN1 (USD)	$7,270

Por lo tanto, queda validado la tercera hipótesis. Contrastadas las hipótesis especificas validamos la hipótesis general planteada: "El diseño de un modelo predictivo basado en machine learning influye significativamente en la optimización del diagnóstico del índice de salud en interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP".

CONCLUSIONES

1. De acuerdo con el análisis de efectividad, se concluye que el algoritmo de machine learning más efectivo para diagnosticar el índice de salud de los interruptores de potencia con los datos de la presente investigación fue: "Random Forest Classifier" con un 99.27% estando por encima del modelo de "Lógica Fuzzy" en 2.61%
2. De acuerdo con el análisis de tiempos, se concluye que el diagnóstico del índice de salud de los interruptores de potencia usando modelos predictivos de machine learning disminuye el tiempo total usado para este proceso como mínimo en 11 días.
3. De acuerdo con el análisis económico de costos, se concluye que usar modelos predictivos con machine learning resultará en un ahorro de USD 448 779, según el cálculo del valor actual neto (VAN) en un horizonte de 10 años, estando por encima del ahorro estimado para el modelo Fuzzy en USD 7 270.

RECOMENDACIONES

1. Para poder determinar los mejores hiperparametros que debemos aplicar a un determinado modelo de machine learning, una buena técnica es usar el algoritmo de "RandomizedSearch", sin embargo no se debe limitar solo a esta técnica y se debe buscar comparar con resultados de otros algoritmos de búsqueda como el "GridSearch", "BayesSearch" o incluso realizar una búsqueda manual.
2. Para casos cuando no se dispone de datos etiquetados, se recomienda usar modelos "No Supervisados" con parámetros comunes que se tengan disponibles en todos los interruptores. Estos modelos utilizan algoritmos de agrupamiento o "clusterizaciones" por características semejantes.
3. Finalmente, para futuras investigaciones se recomienda explorar otras técnicas de machine learning como por ejemplo los algoritmos Semi-Supervisados que combina técnicas de algoritmos Supervisados y No Supervisados, también se recomienda explorar técnicas de "Deep Learning" y comparar sus métricas de efectividad frente a un modelo puramente Supervisado.

BIBLIOGRAFÍA

Ahmad, G.N., Fatima, H., Ullah, S., Saidi, A.S. & Imdadullah (2022), "Efficient Medical Diagnosis of Human Heart Diseases Using Machine Learning Techniques With and Without GridSearchCV,". doi: 10.1109/ACCESS.2022.3165792.

Angelopoulos, A., Michailidis, E. T., Nomikos, N., Trakadas, P., Hatziefremidis, A., Voliotis, S., & Zahariadis, T. (2019). Tackling Faults in the Industry 4.0 Era A Survey of Machine-Learning Solutions and Key Aspects. http://dx.doi.org/10.3390/s20010109

Azure (2023), "Algoritmos de aprendizaje automático: Una introducción a las matemáticas y la lógica subyacentes en el aprendizaje automático". https://azure.microsoft.com/es-mx/resources/cloud-computing-dictionary/what-are-machine-learning-algorithms

Bishop, M.C. (2006). Pattern Recognition and Machine Learning. Editorial Springer. USA.pp758

Deloite. (2014). Asset Health Indices A utility industry necessity. Canadian Electricity Association.pp36.

Domingos, P. (2012). A few useful things to know about machine learning. Commun. ACM 55, 10 (October 2012). https://doi.org/10.1145/2347736.2347755

Durán, O., Orellana, F., Perez, P., & Hidalgo, T. (2020). Incorporating an Asset Health Index into a Life Cycle Costing: A Proposition and Study Case. Mathematics, 8(10), 1787. MDPI AG. http://dx.doi.org/10.3390/math8101787

Ellefsen, L., Bjørlykhaug, E., Æsøy, V., Ushakov, S., & Zhang, H. (2019). Remaining useful life predictions for turbofan engine degradation using semi-supervised Deep architecture, Reliability Engineering & System Safety. https://doi.org/10.1016/j.ress.2018.11.027.

Gao, W., Wai, S., Qiao, R., & Guo, M. (2019). Mechanical Faults Diagnosis of High-Voltage Circuit Breaker via Hybrid Features and Integrated Extreme Learning Machine. https://doi: 10.1109/ACCESS.2019.2915252.

Harper, E. (2012) Manual del Técnico en Subestaciones Eléctricas Industriales y Comerciales. Editorial Limusa S.A. España.pp428

Hernández, S., Fernández, R., & Baptista, P. (2014). Metodología de la investigación (6° Ed.). México, D.F., México: McGraw Hill Interamericana.

Hostelsur (2019). Nueve de cada diez grandes empresas están invirtiendo en machine learning.https://www hosteltur.com/125780_nueve-de-cada-diez-grandes-empresas-estan-invirtiendo-en-machine-learning.

Hurwitz, J. & Kirsch, D (2018). Machine Learning For Dummies. Editorial John wiley & Sons, Inc. USA.pp75

Jahnke, P. (2015). Machine Learning Approaches for Failure Type Detection and Predictive Maintenance. Master's Thesis, Technische Universität Darmstadt, Darmstadt, Germany.pp83.

Li, X., Wu, S., Li, X. et al. (2020). Particle Swarm Optimization-Support Vector Machine Model for Machinery Fault Diagnoses in High-Voltage Circuit Breakers. https://doi.org/10.1186/s10033-019-0428-5

Madushani, J. P. S. S., Sandamal, R. M. K., Meddage, D. P. P., Pasindu, H. R., & Gomes, P. I. A. (2023). Evaluating expressway traffic crash severity by using logistic regression and explainable & supervised machine learning classifiers. https://doi.org/10.1016/j.treng.2023.100190

Mejia, V. (2003). Subestaciones de Alta y Extra Alta Tensión. Editorial Impresiones graficas Ltda. Colombia. pp822

MINEM. (2011). Código Nacional de Electricidad Suministro. Editorial El Peruano. Perú.pp326

Ñaupas, H. (2013). Metodología de la investigación científica y elaboración de Tesis. Lima: Universidad Nacional Mayor de San Marcos.

Omar, A., Delnaz, A., & Nik-Bakht, M. (2023). Comparative analysis of machine learning techniques for predicting water main failures in the City of Kitchener. https://doi.org/10.1016/j.iintel.2023.100044

PCM (2021) Estrategia Nacional de Inteligencia Artificial (ENIA). https://www.gob.pe/institucion/pcm/informes-publicaciones/1929011-estrategia-nacional-de-inteligencia-artificial.

Restrepo L., Calderón L., Cortes E. et al. (2020). Herramienta para estimación de índices de salud en activos de subestaciones de TyD Celsia.XXII congreso internacional de mantenimiento y gestión de activos.

Telefónica (2023). Inteligencia artificial: Cinco claves para el avance en las empresas peruanas.https://telefonica.com.pe/inteligencia-artificial-cinco-claves-para-el-avance-en-las-empresas-peruanas.

Uddin, M. G., Nash, S., Mahammad Diganta, M. T., Rahman, A., & Olbert, A. I. (2022). Robust machine learning algorithms for predicting coastal water quality index. https://doi.org/10.1016/j.jenvman.2022.115923

Yi, T., Xie, Y., H. Zhang & X. Kong. (2020). "Insulation Fault Diagnosis of Disconnecting Switches Based on Wavelet Packet Transform and PCA-IPSO-SVM of Electric Fields. https://doi: 10.1109/ACCESS.2020.3026932.

ANEXOS

ANEXO 1: MATRIZ DE CONSISTENCIA

"MODELOS PREDICTIVOS BASADOS EN MACHINE LEARNING PARA OPTIMIZAR EL DIAGNÓSTICO DEL ÍNDICE DE SALUD EN LOS INTERRUPTORES DE POTENCIA DE UNA EMPRESA DE TRANSMISIÓN ELECTRICA"

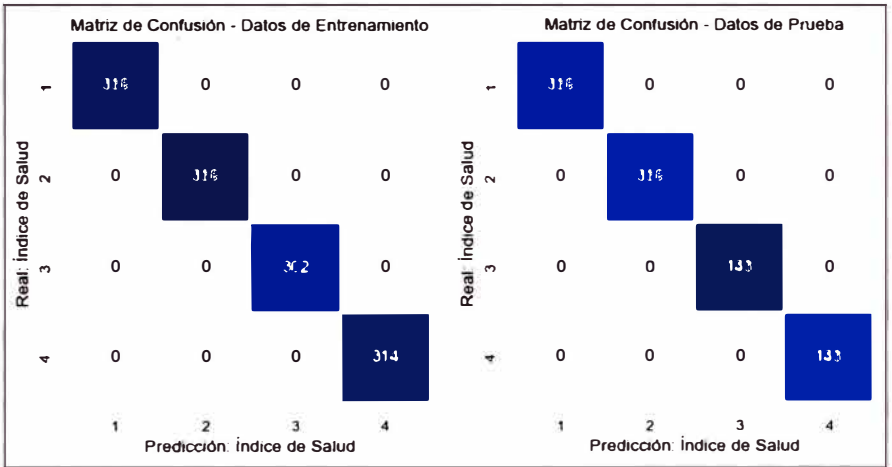
PROBLEMA GENERAL	OBJETIVO GENERAL	HIPÓTESIS GENERAL	VARIABLES	MARCO METODOLÓGICO
¿En qué medida el diseño de modelos predictivos basados en machine learning se constituye en una herramienta para optimizar el diagnóstico del Índice de salud en los interruptores de potencia de la empresa de transmisión de energía electrica peruana ETEP?	Diseñar modelos predictivos basados en machine learning para optimizar el diagnóstico del Índice de Salud en los interruptores de potencia de la empresa de transmisión de energía electrica peruana ETEP	El diseño de un modelo predictivo basado en machine learning influye significativamente en la optimización del diagnóstico del Índice de salud en interruptores de potencia de la empresa de transmisión de energía electrica peruana ETEP	**Variable independiente:** X: Modelos predictivos basados en machine learning **Variable dependiente:** Y: Diagnóstico del Índice de salud en interruptores de Potencia	**Tipo y nivel de la investigación:** La presente investigación es del tipo "Aplicada" de nivel principalmente "Explicativo". **Método de diseño de la investigación:** El diseño de la presente investigación tiene un enfoque "Cuantitativo" del tipo "No experimental"
PROBLEMAS ESPECIFICOS	**OBJETIVOS ESPECIFICOS**	**HIPÓTESIS ESPECIFICOS**	**INDICADORES**	
1.¿En qué grado el diseño de modelos predictivos basados en machine learning me permitirá aumentar la efectividad del diagnóstico del índice de salud en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP? 2.¿Cómo impacta el diseño de modelos predictivos basados en machine learning en el tiempo empleado para el diagnóstico del índice de salud en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP? 3.¿Hasta qué punto el diseño de modelos predictivos basados en machine learning me permitirá optimizar los costos incurridos por fallas imprevista en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP?	1.Aumentar la efectividad del diagnóstico del índice de salud diseñando modelos predictivos basados en machine learning en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP. 2.Disminuir el tiempo empleado para el diagnóstico del índice de salud diseñando modelos predictivos basados en machine learning en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP. 3.Optimizar los costos incurridos por fallas imprevistas diseñando modelos predictivos basados en machine learning en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP.	1.Se aumentará la efectividad del diagnóstico del índice de salud diseñando de modelos predictivos basado en machine learning en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP. 2.Se disminuirá el tiempo empleado para el diagnóstico del índice de salud diseñando modelos predictivos basado en machine learning en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP. 3.Se optimizará los costos incurridos por fallas imprevistas diseñando modelos predictivos basado en machine learning en los interruptores de potencia de la empresa de transmisión de energía eléctrica peruana ETEP..	**Variable independiente:** - Efectividad del diagnóstico del Índice de Salud **Variable dependiente:** - Tiempo para diagnosticar el Índice de Salud -Costos por fallas imprevistas	**Unidad de Analisis** Interruptores de potencia >= 60kV, cuyo medio de extinción es el gas SF6 y que se encuentran operando a nivel nacional en las subestaciones eléctricas de la empresa de transmisión de energía electrica peruana ETEP **Tecnicas de recolección y procesamiento de datos** Se usaron los siguientes sistemas y programas informáticos de recolección y procesamiento de datos: • SAP-PM • • Python • • Excel •

ANEXOS 2: EXPLORACIÓN CON "GRID SEARCH"

El método de GridSearch se descartó para esta investigación porque los valores sugeridos para los hiperparametros generaban un alto sobreajuste del modelo (overfiting), es decir el modelo con estos hiperparametro se ajustaba tanto a los datos que, en vez de aprender de los datos, los memoriza. El sobreajuste suele ocurrir cuando un modelo es demasiado complejo en relación con la cantidad de datos de entrenamiento disponibles. El modelo puede aprender incluso el ruido o las fluctuaciones aleatorias en los datos de entrenamiento, en lugar de capturar solo las relaciones genuinas y significativas. Esto puede resultar en un rendimiento deficiente cuando el modelo se enfrenta a nuevos datos.

Como se puede apreciar en la matriz de confusión y sus métricas, los hiperparametros sugeridos por el algoritmo de Gridsearch ajustaron tanto al modelo que todo sale perfecto y cuando estamos ante estos casos siempre debemos concluir que los hiperparametro no son los adecuados y buscar otras alternativas de búsqueda hiperparametro óptimos.

MATRIZ DE CONFUSIÓN Y MÉTRICAS CON "GRID SEARCH"




```
Métricas de conjunto de entrenamiento con Random Forest Classifier:
Exactitud: 1.0000
Precisión: 1.0000
Precisión (individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 1.0000
  - Para el Índice de salud 4: 1.0000
Sensibilidad: 1.0000
Sensibilidad (Individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 1.0000
  - Para el Índice de salud 4: 1.0000
F1-Score: 1.0000
F1-Score (individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 1.0000
  - Para el Índice de salud 4: 1.0000
Métricas de conjunto de prueba con Random Forest Classifier:
Exactitud: 1.0000
Precisión: 1.0000
Precisión (Individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 1.0000
  - Para el Índice de salud 4: 1.0000
Sensibilidad: 1.0000
Sensibilidad (Individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 1.0000
  - Para el Índice de salud 4: 1.0000
F1-Score: 1.0000
F1-Score (individual):
  - Para el Índice de salud 1: 1.0000
  - Para el Índice de salud 2: 1.0000
  - Para el Índice de salud 3: 1.0000
  - Para el Índice de salud 4: 1.0000
```

ANEXO 3: HIPERPARAMETROS USADOS CON "RANDOM SEARCH"

- "n_iter=100".- especifica la cantidad de iteraciones o muestras aleatorias de hiperparámetros que serán evaluadas durante la búsqueda aleatoria de hiperparámetros. Esto implica que se probarán 100 configuraciones de hiperparámetros de forma aleatoria antes de concluir el proceso de búsqueda. El valor de "n_iter" puede ser adaptado de acuerdo con las necesidades computacionales y el tiempo que el usuario esté dispuesto a dedicar a la búsqueda de hiperparámetros. Usualmente, un valor mayor de "n_iter" incrementa la probabilidad de hallar una combinación de hiperparámetros efectiva, aunque también puede extender el tiempo de ejecución del proceso.
- "cv=5".- Se especifica este valor para indicar que se deben dividir los datos en 5 conjuntos para realizar una validación cruzada con cada conjunto durante la búsqueda de hiperparámetros con "RandomizedSearch". Normalmente se utiliza 5 ó 10 divisiones de datos y debido la limitada cantidad de datos que se tiene, se optó solo por 5 divisiones.
- "scoring"='F1'.- Se utiliza para especificar el tipo de métrica a usar durante la evaluación del rendimiento del modelo en cada combinación de hiperparámetros. Se optó por la métrica F1-Score dado que esta combina a su vez las métricas Precisión y Sensibilidad en un cálculo de media armónica.
- "n_jobs = -1".- Se utiliza para especificar la cantidad de núcleos del procesador que debe usar el algoritmo, al poner: "-1" se le está diciendo al modelo que utilice todos los núcleos disponibles para paralelizar el proceso de búsqueda de hiperparámetros y acelerar el cálculo.
- "random_state = 42".- solo establece una semilla para la generación de números aleatorios. Utilizar un valor fijo para "random_state" garantiza la reproducibilidad de los resultados. En este caso particular, la elección de 42 es un número arbitrario y podría haberse optado por cualquier otro número. El número 42 es a menudo utilizado como una especie de "semilla estándar" en la comunidad de programación debido a su relación con la respuesta a la "Pregunta Fundamental de la Vida, el Universo y Todo", en la serie de libros "The Hitchhiker's Guide to the Galaxy" de Douglas Adams.

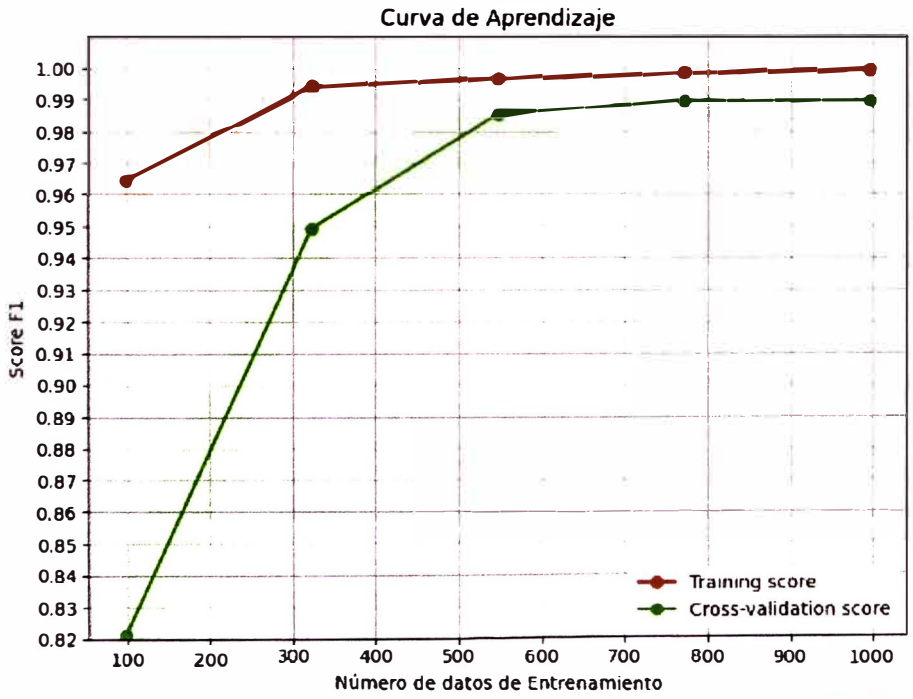
ANEXO 4: CURVAS DE APRENDIZAJE DE ALGORITMOS MODELADOS

Las curvas de aprendizaje muestran como un modelo va aprendiendo a medida que se entrena con más datos. La curva de rojo muestra el rendimiento durante el entrenamiento y la de verde el rendimiento durante pruebas de validación cruzada.

Curva de aprendiza para Random Forest Classifier

```
import scikitplot as skplt
import matplotlib.pyplot as plt

# Trazar la curva de aprendizaje con scikitplot
skplt.estimators.plot_learning_curve(model, X_train, y_train, figsize=(8, 6),scoring='f1_macro')

# Ajustar el eje y con incrementos de 0.01
plt.gca().set_ylim([0.82, 1.01])
plt.gca().set_yticks([i/100 for i in range(82, 101, 1)])

# Ajustar el eje x para incrementos de 100 en 100
x_ticks = [i for i in range(100, min(len(X_train)+1, 1100), 100)]
plt.xticks(x_ticks)

plt.title('Curva de Aprendizaje')
plt.xlabel('Número de datos de Entrenamiento')
plt.ylabel('Score F1')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```

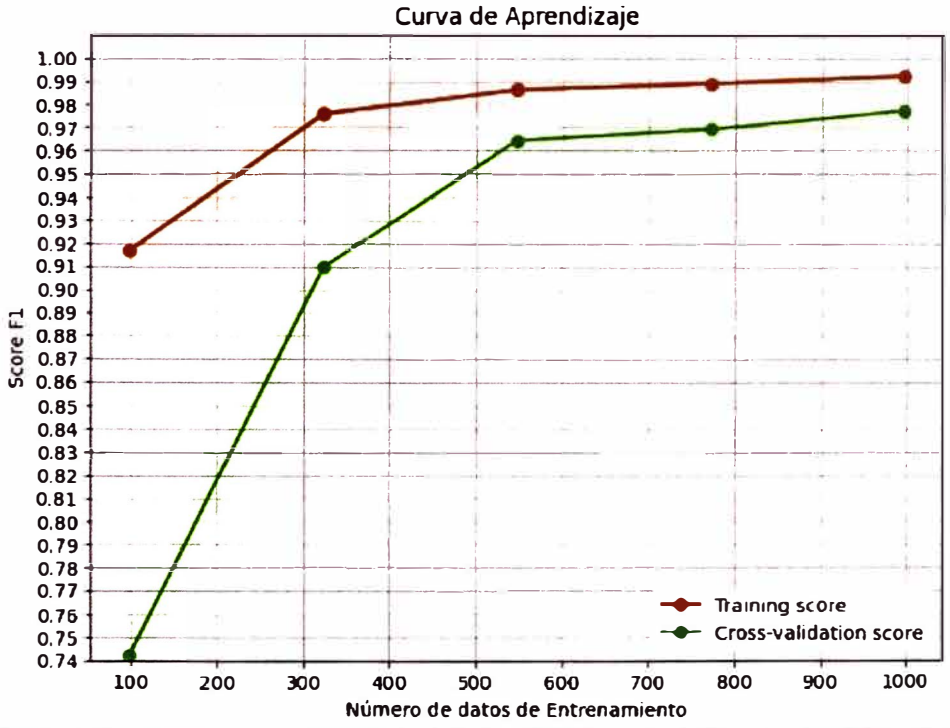
✓ 2.5s



Curva de aprendiza para Gradient Boosting Classifier

```
import scikitplot as skplt
import matplotlib.pyplot as plt

# Trazar la curva de aprendizaje con scikitplot
skplt.estimators.plot_learning_curve(model, X_train, y_train, figsize=(8, 6),scoring='f1_macro')

# Ajustar el eje y con incrementos de 0.01
plt.gca().set_ylim([0.74, 1.01])
plt.gca().set_yticks([i/100 for i in range(74, 101, 1)])

# Ajustar el eje x para incrementos de 100 en 100
x_ticks = [i for i in range(100, min(len(X_train)+1, 1100), 100)]
plt.xticks(x_ticks)

plt.title('Curva de Aprendizaje')
plt.xlabel('Número de datos de Entrenamiento')
plt.ylabel('Score F1')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```

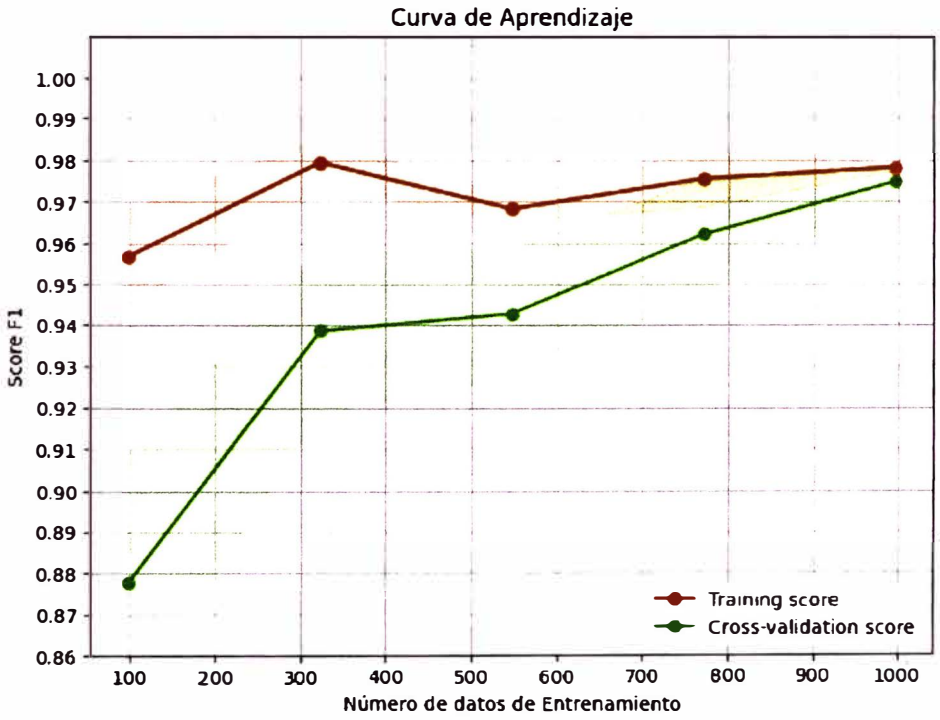
✓ 7.3s



Curva de aprendiza para Support Vector Machines

```
import scikitplot as skplt
import matplotlib.pyplot as plt

# Trazar la curva de aprendizaje con scikitplot
skplt.estimators.plot_learning_curve(model, X_train, y_train, figsize=(8, 6),scoring='f1_macro')

# Ajustar el eje y con incrementos de 0.01
plt.gca().set_ylim([0.86, 1.01])
plt.gca().set_yticks([i/100 for i in range(86, 101, 1)])

# Ajustar el eje x para incrementos de 100 en 100
x_ticks = [i for i in range(100, min(len(X_train)+1, 1100), 100)]
plt.xticks(x_ticks)

plt.title('Curva de Aprendizaje')
plt.xlabel('Número de datos de Entrenamiento')
plt.ylabel('Score F1')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```

✓ 1.4s



Curva de aprendiza para K-Nearest Neighbors

```
import scikitplot as skplt
import matplotlib.pyplot as plt

# Trazar la curva de aprendizaje con scikitplot
skplt.estimators.plot_learning_curve(model, X_train, y_train, figsize=(8, 6), scoring='f1_macro')

# Ajustar el eje y con incrementos de 0.01
plt.gca().set_ylim([0.85, 1.01])
plt.gca().set_yticks([i/100 for i in range(85, 101, 1)])

# Ajustar el eje x para incrementos de 100 en 100
x_ticks = [i for i in range(100, min(len(X_train)+1, 1100), 100)]
plt.xticks(x_ticks)

plt.title('Curva de Aprendizaje')
plt.xlabel('Número de datos de Entrenamiento')
plt.ylabel('Score F1')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```

✓ 0.6s