ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA MANTENIMIENTO INDUSTRIAL

“DETECCIÓN DE FALLAS EN UN SISTEMA HIDRÁULICO UTILIZANDO REDES NEURONALES ARTIFICIALES”

Trabajo de Integración Curricular

Tipo: Proyecto de Investigación

Presentado para optar por el grado académico de:

INGENIERO EN MANTENIMIENTO INDUSTRIAL

AUTOR:

HUGO SEBASTIÁN MERA CRUZ

Riobamba-Ecuador

2022

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA MANTENIMIENTO INDUSTRIAL

“DETECCIÓN DE FALLAS EN UN SISTEMA HIDRÁULICO UTILIZANDO REDES NEURONALES ARTIFICIALES”

Trabajo de Integración Curricular

Tipo: Proyecto de Investigación

Presentado para optar por el grado académico de:

INGENIERO EN MANTENIMIENTO INDUSTRIAL

AUTOR: HUGO SEBASTIÁN MERA CRUZ

DIRECTOR: Ing. FÉLIX ANTONIO GARCÍA MORA

Riobamba-Ecuador

2022

Yo, Hugo Sebastián Mera Cruz, declaro que el presente trabajo de integración curricular es de mi autoría y los resultados de este son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor asumo la responsabilidad legal y académica de los contenidos de este trabajo de integración curricular; el patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 11 de mayo del 2022

Hugo Sebastián Mera Cruz

C.I: 180485648-0

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA MANTENIMIENTO INDUSTRIAL

El Tribunal del Trabajo de Integración Curricular certifica que: El Trabajo de Integración Curricular; Tipo: Proyecto de Investigación, "**DETECCIÓN DE FALLAS EN UN SISTEMA HIDRÁULICO UTILIZANDO REDES NEURONALES ARTIFICIALES**.", realizado por el señor **HUGO SEBASTIÁN MERA CRUZ**, ha sido minuciosamente revisado por los Miembros del Trabajo de Integración Curricular, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal Autoriza su presentación.

	FIRMA	**FECHA**
Ing. Marco Ordóñez Viñán Msc. **PRESIDENTE DE TRIBUNAL**	espoch Facultad de Mecánica Carrera de Mantenimiento Industrial MARCO ANTONIO ORDONEZ VINAN	2022-05-11
Ing. Félix Antonio García Mora **DIRECTOR DE TRABAJO DE INTEGRACIÓN CURRICULAR**	Firmado electrónicamente por: FÉLIX ANTONIO GARCÍA MORA	2022-05-11
Ing. Edisson Fernando Calderón Freire **MIEMBRO DE TRIBUNAL**	Firmado electrónicamente por: EDISSON FERNANDO CALDERON FREIRE	2022-05-11

DEDICATORIA

Este trabajo está dedicado principalmente a Dios por la fortaleza y salud, a mis padres Jorge Mera y Cecilia Cruz quienes han sido el pilar fundamental y apoyo emocional durante mi preparación profesional. A mi hermano Adrián Mera quien me ha motivado a ser mejor cada día y a mis amigos quienes me brindan su apoyo incondicional, impulsándome a seguir adelante sin desistir de mis sueños por alcanzar.

Sebastián.

AGRADECIMIENTO

A la Escuela Superior Politécnica de Chimborazo, en especial a la carrera en Mantenimiento Industrial quien me abrió sus puertas para continuar la preparación profesional, a los docentes que han demostrado su aptitud, paciencia y dedicación en su gran labor. A mis tutores: Ing. Félix García e Ing. Edisson Calderón por confiar en mí y guiarme a culminar este objetivo propuesto.

Sebastián.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE ANEXOS

RESUMEN

El presente proyecto investigativo tuvo como objetivo detectar fallas y condiciones en un sistema hidráulico utilizando redes neuronales artificiales. Se inició con la indagación de los factores que influyen en modelos inteligentes para la detección de fallos, además se analizó la base de datos de monitoreo de condición de sistemas hidráulicos del repositorio de la Universidad de Carolina, consecuentemente se utilizó lenguaje de programación Python para el planteamiento de un modelo inteligente basado en el aprendizaje profundo (Deep Learning), con redes neuronales artificiales creando una estructura de visualización y limpieza de datos, a continuación, se dimensionaron y normalizaron los datos para un set de prueba y entrenamiento 80-20 de aprendizaje para la ANN, se ejecutó una red neuronal artificial basado en la librería de Theano-Keras de tipo dense como clasificador juntamente con el optimizador Adam basado en gradientes de caída estocásticas con el activador de función sigmoid-softmax. Se evaluó la predicción de las condiciones del sistema hidráulico que se muestran en varias matrices de confusión, dando como resultado con una exactitud del 97% de efectividad mostrando el desenvolvimiento y el nivel de épocas o iteraciones de aprendizaje, demostrando un nivel de confianza aceptable para la detección de fallos en sistemas hidráulicos. Se concluyó que acuraccy es la métrica final usada para medir la exactitud del modelo inteligente. Se recomienda antes de iniciar el análisis de los datos con el método inteligente, obtener información adecuada confiable acerca de Machine Learning y Deep Learning ya que todos los métodos de inteligencia artificial no pueden aplicarse a todos los modelos inteligentes.

Palabras clave: <SISTEMA HIDRAÚLICO> <REDES NERONALES ARTIFICIALES> <MODELOS INTELIGENTES> <DETECCIÓN DE FALLOS> <MATRIZ DE CONFUSIÓN>.

SUMMARY

The objective of this research project was to detect faults and conditions in a hydraulic system using artificial neural networks. It began with the investigation of the factors that influenced intelligent models for fault detection. In addition, the condition monitoring database of hydraulic systems from the repository of Carolina University was analyzed. Consequently the Python programming language was used for the approach of an intelligent model based on Deep Learning, with artificial neural networks creating a structure of visualization and data cleansing. Then, the data is dimensioned and normalized for a test set and training 80-20 of learning to ANN. An artificial neural network based on the dense Theano-Keras library was implemented as a classifier together with the Adam optimizer based on stochastic decay gradients with the sigmoid-softmax function activator. The prediction of the conditions of the hydraulic system that is shown in several confusion matrices was evaluated, resulting in an accuracy of 97% efficiency showing the development and the level of times or iterations of learning, demonstrating an acceptable level of confidence for fault detection in hydraulic systems. It was concluded that accuracy is the final metric used to measure the accuracy of the intelligent model. Before starting the data analysis with the intelligent method, it is recommended to obtain adequate and reliable information about Machine Learning and Deep Learning since all the artificial intelligence methods cannot be applied to all the intelligent models.

Keywords: <HYDRAULIC SYSTEMS> <ARTIFICIAL NEURAL NETWORKS> <INTELLIGENT MODELS> <FAULT DETECTION> <CONFUSION MATRIX>.

INTRODUCCIÓN

Conocer el avance tecnológico es de gran importancia para la industria ya que permite mejorar su productividad, los procesos y lo más importante la vida útil de los equipos con la correcta aplicación del mantenimiento industrial. La industrialización de las empresas a gran a escala hoy en día ha permitido que adquieran sistemas hidráulicos, ya que su diseño y fabricación de máquinas es de muy alta precisión realizando trabajos muy pesados siendo aplicadas en varios sectores.

Las máquinas que usan la hidráulica poseen características específicas y cumplen algunos parámetros, la cual el funcionamiento de estos sistemas es muy sencillo, un fluido a presión genera una energía lo suficiente para realizar un trabajo, son sistemas económicos que poseen desventajas. Al ser sistemas sencillos poseen problemas en la fácil detección de fallos en sus sensores ya que a simple visualización no se puede detectar fallos en su totalidad, necesitando de herramientas sofisticadas de monitoreo para la detección de fallos. Con el progreso tecnológico en la actualidad, se ha empezado a implementar la inteligencia artificial en la industria para la detección acertada de fallos en sistemas hidráulicos.

Las Redes Neuronales Artificiales son una parte de la inteligencia artificial, siendo un modelo matemático que simula el comportamiento biológico de las neuronas y la estructura del cerebro, que es utilizado para resolver un amplio rango de problemas. Este sistema puede ser visto como un sistema inteligente que dirige varias tareas como las realiza una computadora actual. Estas últimas son muy rápidas en el procesamiento de información, existiendo tareas muy complejas, como el reconocimiento y clasificación de modelos que demandan mucho tiempo y no son capaces de procesar, pero el cerebro humano es el más apto para resolverlas convirtiéndose en sistemas altamente complejos.

La aplicación de redes neuronales artificiales en sistemas hidráulicos se los realiza mediante una base de datos obtenida durante el proceso de trabajo y el historial de fallos que ha presentado la maquinaria, permitiendo al algoritmo simular una neurona cerebral que pueda ser capaz de predecir fallos futuros y así mejorar la disponibilidad operacional de los sistemas.

CAPÍTULO I

1. MARCO TEORICO Y MARCO CONCEPTUAL

1.1. Antecedentes

Los sistemas hidráulicos son uno de los principales subsistemas ampliamente utilizados en las industrias. Pueden transmitir cargas elevadas con menores esfuerzos (Prakash & Kankar, 2020, p.1). El mantenimiento a través de la historia ha evolucionado de forma exponencial, siendo la principal fuente de mejora en la industria mecánica, eléctrica y civil. De esta forma se ha mejorado la producción y preservación, mejorando así la disponibilidad y vida útil de un activo físico. Sin embargo, todo esto se ve reflejado en la afectación al medio ambiente por las malas aplicaciones del mantenimiento para la detección de fallas en diferentes sistemas que son aplicados en la industria, siendo uno de ellos los sistemas hidráulicos.

> *Los cuales se utilizan ampliamente en la industria moderna, que suele desempeñar un papel insustituible en la mayoría de los sistemas industriales. Una consecuencia desastrosa y enormes pérdidas económicas ocurrirían una vez que fracasara. Para garantizar que todo el sistema hidráulico esté en funcionamiento normal, es fundamental contar con un método de control del estado del sistema hidráulico fiable y preciso. Los métodos de monitorización de la condición actual se desarrollan basándose en algoritmos clásicos de aprendizaje automático, por ejemplo, redes neuronales, máquina de soporte vectorial (SVM), análisis discriminante lineal (LDA). Sin embargo, cuando elegimos un método para la clasificación o regresión de los sistemas hidráulicos, generalmente no podemos obtener una alta precisión y es difícil elegir un algoritmo si el rendimiento de todos los métodos no es deseable.* (Guo et al., 2019, p.1)

Los estudios realizados con aplicación de redes neuronales artificiales son muy escasos, se utilizaron por primera vez para modelado del sistema nervioso humano. Sin embargo, hoy en día se pueden implementar como una técnica numérica universal para la resolución de problemas aplicados en el campo de la ingeniería, el diseño de sistemas de control, así como la optimización de los procesos de fabricación. Cada vez, dichos métodos se van acoplando y mejorando industria, Desde este punto de vista, la ANN se puede aplicar como un modelo simplificado y modificado de la actividad cerebral. El amplio uso de estos sistemas es una mejora grande y simplificada ya que ayuda al área de mantenimiento a predecir fallos y monitorear todos los sistemas dentro de una empresa, Sin embargo, no existen avances significativos en el uso de sistemas de inteligencia artificial para la resolución de problemas aplicados en el campo de la ingeniería mecánica debido a la ausencia de enfoques unificados para su implementación.(Pavlenko et al., 2018, p.19)

1.2. Planteamiento del problema

La afectación por fallos en sistemas hidráulicos ha sido un factor importante para la productividad en la industria, la falta de estudio y de aprendizaje ha conllevado a no innovar nuevos métodos de mantenimiento para la detección de fallos en los sistemas hidráulicos. Las fugas internas son una falla típica en los sistemas hidráulicos, que pueden ser causadas por daños en los sellos y resultar en un rendimiento deteriorado del sistema. (Yao, Yu y Yao, 2018, pp.1-2). Siendo así, que la inteligencia artificial vaya tomando espacio en el sector productivo para imitar tareas generadas por los sistemas. La cual las industrias modernas poseen un alto nivel de tecnología para que la producción pueda tener altos estándares de calidad; pero, si se produjese deficiencias y averías en la maquinaria, el funcionamiento normal de la producción se vería alterado, generando grandes pérdidas económicas y posibles accidentes catastróficos. Por lo tanto, existe la necesidad de diagnosticar las fallas de manera efectiva y precisa de un sistema hidráulico para mantener una alta confiabilidad de la producción. (Liu et al., 2019, p.2050)

1.3. Justificación del problema

La escasez de conocimiento en la aplicación de métodos modernos de detección de fallas en sistemas hidráulicos ha sido visto como un problema ya que los métodos tradicionales que se utilizan son muy demandantes existiendo paros innecesarios para la producción en la industria, se ha demostrado que aplicando algunos métodos de inteligencia artificial se pueden detectar fallas en diversos tipos de equipos o componentes.

En este estudio se desea demostrar que, mediante la utilización de las redes neuronales artificiales, se pueden tener buenos resultados para la detección de fallos de un sistema hidráulico utilizando los datos de Machine Learning Repository de la Universidad de California, Irvine.

En el Ecuador se han realizado trabajos de investigación utilizando inteligencia artificial en varias aplicaciones, pero pocas investigaciones enfocadas específicamente al mantenimiento industrial. En la Carrera de Mantenimiento Industrial de la Facultad de Mecánica de la Escuela Superior Politécnica de Chimborazo, se ha visto la necesidad de realizar investigaciones aplicando la tecnología, que permitan mejorar el desenvolvimiento de las aplicaciones del avance tecnológico existente de la inteligencia artificial enfocadas al mantenimiento industrial, por lo que el presente trabajo de integración curricular busca plantear una idea innovadora que aporte al desarrollo cognitivo de los estudiantes y profesionales de la Carrera de Mantenimiento Industrial.

1.4. Objetivos

1.4.1. Objetivo general

Detectar fallas en un sistema hidráulico utilizando redes neuronales artificiales

1.4.2. Objetivos específicos

Dividir los datos para entrenamiento y prueba en un sistema hidráulico.

Encontrar las mejores funciones que correlacionan a los datos.

Aplicar un algoritmo de redes neuronales artificiales con aprendizaje profundo para los datos en sistemas hidráulicos.

Comprobar la precisión del método de redes neuronales artificiales utilizados para los datos de sistemas hidráulicos.

1.5. Hipótesis

Utilizando las redes neuronales artificiales se detectan fallos en un sistema hidráulico

1.5.1. Variable dependiente

Detección de fallos

1.5.2. Variables independientes

Redes Neuronales Artificiales (ANN)
Precisión (F1-score)
Matriz de confusión

1.6. Fundamentación teórica

La fundamentación teórica hace mención a citas bibliográficas, Libros, Papers, Tesis, Maestrías y Doctorados que mencionan la aplicación de la Inteligencia Artificial a la industria.

1.6.1. Introducción a Python con Anaconda

Python es uno de los lenguajes de programación que se ha convertido en el más usado para la investigación. Es gratuita y de código abierto, encontrando varios innumerables ejemplos de aumento de la productividad para la investigación gracias a Python, a través de una abundancia de propietarios en línea incluyendo la ciencia de datos, la inteligencia artificial y la investigación científica.(Rolon-Mérette et al., 2020, p.S3).

En el mundo de la internet existe una gran demanda para el estudio y el avance tecnológico, usando todos los esfuerzos de la comunidad para la investigación en el desarrollo y avance de la aplicación de un software para la resolución de problemas de ciencia de datos. Anaconda es una de las muchas plataformas de código abierto que facilitan el uso de lenguajes de programación (Python, R) para gran procesamiento de datos, análisis predictivo y computación científica. Utilizando la plataforma de Anaconda con su paquete de biblioteca en Python con Jupyter Notebook o Spyder para el modelado de modelos matemáticos.(Kadiyala y Kumar, 2017, p.1).

1.6.1.1. Instalación de Python con Anaconda

La plataforma que más se asemeja para utilizar el sistema de lenguaje de programación en Python es anaconda Python. Anaconda se hace llamar “meta distribuciones”, siendo un sistema operativo sobre otro sistema operativo. Se basa en la investigación científica por lo que su Python viene con varios módulos sencillos de instalar para muchas aplicaciones científicas, es posible instalar varios entornos en el ordenador gracias a su navegador con diferentes lenguajes para programación además de Python.(Zadka, 2019, p.5).

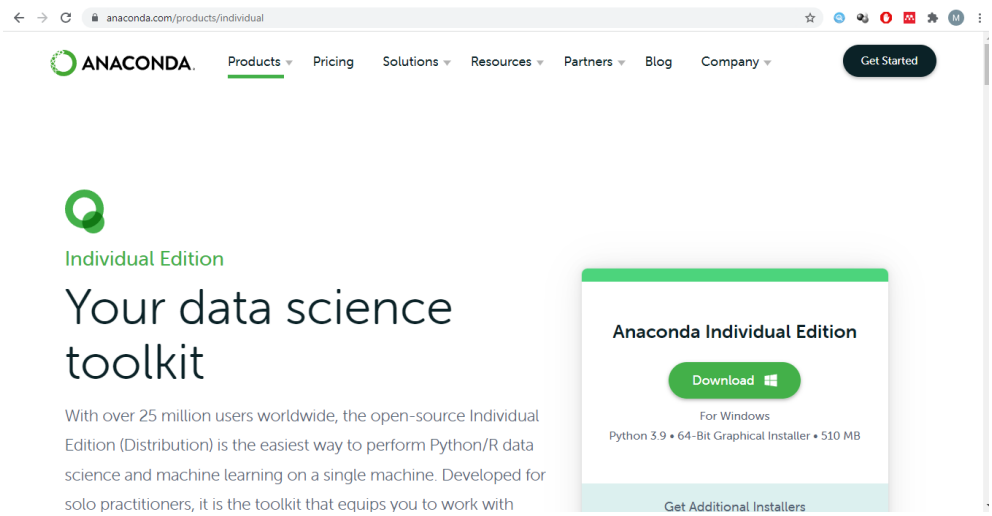


Figura 1-0: Instalador Python Anaconda.

Fuente:(© Anaconda Inc., 2021)

La instalación del navegador Anaconda es proporcionada del sitio web propio de manera gratuito en Mac, Linux y Windows. Se inicia descargando el instalador escogiendo el sistema operativo el cual se posee con un instalador gráfico de 32 y 64bits. Es importante notar que Python se va actualizando automáticamente, en las opciones de instalación se recomienda no seleccionar nada. Una vez cumplida la instalación se puede buscar Anaconda Navigator en su dispositivo y está listo para su ejecución. (Rolon-Mérette et al., 2020, p.S5).

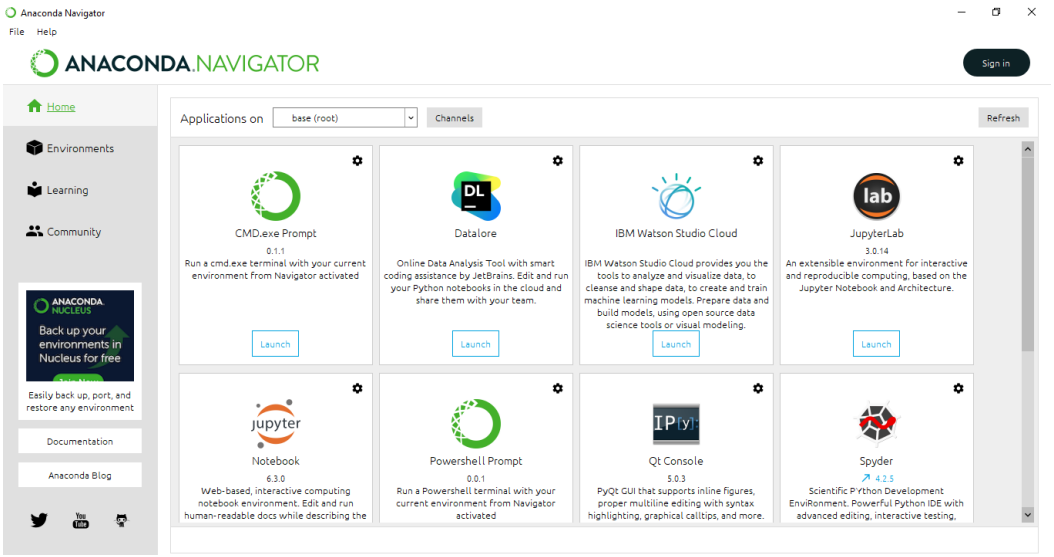


Figura 2-0: Navegador Anaconda

Fuente: (Rolon-Mérette et al., 2020)

1.6.2. Detección de fallas.

La detección de fallas es un diagnóstico y monitoreo de resultados obtenido por tres clases de categorías: basado en modelos matemáticos, en señales y conocimiento. Los modelos matemáticos describen la forma de funcionamiento del equipo, los métodos basados en señales son usados con cuatros diferentes técnicas para el procesamiento de señales: análisis en el dominio con el tiempo, análisis en el dominio de la frecuencia, análisis de frecuencia mejorado y técnicas de análisis de tiempo-frecuencia, éstos no necesitan modelos completos del sistema. Los métodos basados en el conocimiento se aplican en métodos cuantitativos que se basa en inteligencia de aprendizaje automático (Machine Learning) y cualitativos sobre base de inteligencia simbólica.(Ince et al., 2016, p.1).

1.6.3. Detección de fallas en sistemas hidráulicos.

La detección de fallas en un sistema hidráulico es aquel que mantiene datos de sensores de series de tiempo multivalentes; que, en los entornos industriales se puede llegar a sufrir con datos

ruidosos como resultados erróneos en la detección por medio de hardware o interrupciones externas. Existiendo tres desafíos para la formulación de una detección de fallos en sistema hidráulicos. Primero, realizar una detección automatizada y en tiempo real sin intervención del operario ya que es complicado debido a los requisitos sensibles al tiempo para mantener la correcta funcionalidad del sistema. Segundo, los datos de los sensores son más abundantes en cantidad y dimensión, por lo que la identificación de errores es un desafío debido a la complejidad no lineal que se relacionan entre datos. En tercer lugar, por lo general los sensores se encuentran en lugares ruidosos, por lo que los datos suelen ser ruidosos y poco confiables debido a errores producidos en las configuraciones del sensor o interferencias en el entorno externo. (Fawwaz y Chung, 2020, pp.1-2).

1.6.4. Análisis de fallas

Aplicar un análisis de fallas es muy importante para la prevención de averías o daños que se puedan producir o repetir a futuro. Se previene o se reduce el tiempo de inactividad, garantizando la confiabilidad de los activos físicos. Los sistemas hidráulicos que se usan para la producción requieren herramientas de monitoreo para la condición del equipo, el factor humano toma decisiones según el análisis y la obtención de información adquirida visualmente o verificada en hardware inteligente optimizado para la monitorización de máquinas.(Zinnikus et al., 2017, p.207).

1.6.5. Principales defectos en sistemas de hidráulicos

Los sistemas hidráulicos están siempre expuestos a diversos ambientes, siendo poco considerados los daños que puedan ocurrir o puedan existir. Los mayores fallos en los sistemas hidráulicos se relacionan con la violación de la operabilidad de acoplamientos de precisión como consecuencia del desgaste mecánico abrasivo y corrosivo en las superficies de trabajo (Kozyreva, Kozyrev y Chupyatov, 2018, p.985).

El sistema puede presentar daños incluso si está operando, estos defectos se deben tomar en consideración para que exista una operación segura y confiable, siendo varios los más comunes: Bombas y motores hidráulicos que toman la energía mecánica o eléctrica, la energía mecánica es la base de los subconjuntos de toda la máquina definiendo sus velocidades, frecuencias y de forma general la productividad en la que se trabaja, en el aumento de presión el sistema hidráulico sigue funcionando pero con ritmo lento y reduciendo su rendimiento, los distribuidores hidráulicos y las válvulas pueden alterar el flujo del fluido aumentando las pérdidas totales de caudal. (Drumea et al. 2016, pp.212-213).

1.6.6. Feature extraction

Feature extraction o extracción de características se aplica en el aprendizaje automático (Machine Learning) para una investigación científica aplicando una parte de la inteligencia artificial basados en modelos matemáticos. La extracción de características, por ejemplo, cuando se habla de un contexto de la clasificación de dígitos escritos a mano, las características extraídas corresponden a la alimentación de datos de señales para aplicarlas a un entrenador o clasificador como, por ejemplo: una máquina de soporte vectorial (SVM), Redes Neuronales Artificiales (ANN), etc. A su vez la clasificación de dígitos escritos a mano lo haríamos, pero necesitamos que el extractor sea robusto por esa razón se emplea el clasificador automático para simular la toma de datos manuales, dándole las características y límites deseados. El éxito en tareas prácticas de aprendizaje automático es generado por redes neuronales que se los simula como modelos matemáticos.(Wiatowski y Bolcskei, 2018, p.1).

1.6.7. Machine Learning

Machine learning conocido como el aprendizaje automático es una parte que conforma la inteligencia artificial, utilizando métodos algebraicos de matrices para el aprendizaje automático, incluyendo selección de características, un análisis de componentes y el análisis de correlación de datos; constando de tres aprendizajes: aprendizaje supervisado, no supervisado y el aprendizaje activo.(Zhang, 2020, p.223). El aprendizaje automático repasa continuamente conocimientos pasados para luego utilizarlos en el futuro que serán destinados a la resolución de problemas, su base fundamental de ejecución de aprendizaje dominante actual se aprende de forma apartada, formando un conjunto de datos de entrenamiento, ejecuta el algoritmo de aprendizaje automático en el grupo de datos para después aplicarse un modelo que se lo utiliza en la aplicación prevista. No intenta retener el conocimiento aprendido y utilizarlo en el aprendizaje posterior.(Chen y Liu, 2018, p.viii).

1.6.8. Deep Learning

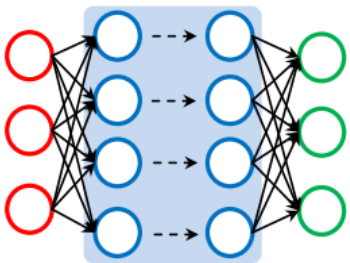


Figura 3-0: Modelo básico de una Red Profunda

Fuente: (Hao, Zhang y Ma, 2016)

Deep Learning o conocido al español como Aprendizaje profundo, una rama del Machine Learning que modela datos de alto nivel, Utilizando varias capas de neuronas formando estructuras complejas o a su vez transformadas no lineales. Dependiendo a la cantidad de datos y el poder computacional la Redes Neuronales con estructuras más complejas atraído una amplia aplicación a diferentes campos de análisis, incluyendo algoritmos de arquitectura y algoritmos de entrenamiento, todo el análisis es gracias al poder que posee la red neuronal con un análisis múltiple de diferentes modelos inteligentes que se desea aplicar.(Hao, Zhang y Ma, 2016, p.417)

1.6.9. Matriz de correlación

En el aprendizaje automático la matriz de correlación juega un papel muy importante ya que el conjunto de datos de posibles medidas más utilizadas dependientes entre todas las variables aleatorias. Las correlaciones aumentan el fenómeno a un colapso que se hace referencia a la diversificación de datos, porque la mayoría de datos son tomados empíricamente. (Wied, 2015, p.2). Las matrices son la principal acción del análisis multivariado, ya que este avance es muy importante para la aplicación de varios métodos componentes de escala, evaluación de modelos gráficos, análisis discriminante y análisis factorial. la matriz de correlación recibe mucha atención en la estadística de alta dimensión, se debe a que representa la dimensionalidad y el tamaño de la muestra que se va aplicar en el aprendizaje automático.(Han y Liu, 2017, p.23).

1.6.10. Matriz de Confusión

La Matriz de Confusión se aplica después del entrenamiento de los datos que se utilizaron en el algoritmo de inteligencia artificial para demostrar la evaluación utilizada en el Machine Learning. Una matriz de confusión o una matriz de error, crea varias predicciones y resultados de prueba, también ayuda a obtener un análisis en profundidad de los datos estadísticos más rápido a través de una visualización de datos.(Babel, Kumar Singh y Kumar Jangir, 2019. p.2). Las columnas de una Matriz de confusión son los resultados del tipo de predicción y las filas los resultados reales, son enumerados todos los resultados posibles, tomando el problema como una clasificación binaria.(Xu, Zhang y Miao, 2020, p.775).

La matriz de confusión es una tabla de frecuencias bidireccionales con dos variables binarias reales (bueno o malo) y predicho (bueno o malo) los valores son cuatro elementos que se denominan: Verdadero Negativo (TN), Verdadero Positivo (TP), Falso Negativo (FN) y Falso Positivo (FP), no necesariamente deben ir en orden.

- Verdadero negativo es el número de negativos (reales) que se clasifican correctamente como negativos.

- Falso negativo es el número de positivos (reales) que se clasifican incorrectamente como negativos.
- Verdadero positivo es el número de positivos (reales) que se clasifican correctamente como positivos
- Falso positivo es el número de negativos (reales) que se clasifican incorrectamente como positivos.

Desde el punto de vista de la lingüística, podemos interpretar Falso como "no actual".(Zeng, 2019, pp.3-4).

1.6.10.1. Precisión

Con el método podemos medir la calidad del modelo de machine learning en formas de clasificación. (Zeng, 2019, p.9). Se lo calcula

$$\text{Precisión} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

1.6.10.2. Recall (Exhaustividad)

Con el método informar sobre la cantidad que el modelo de machine learning es capaz de determinar. (Zeng, 2019, p.9). Se lo calcula

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

1.6.10.3. Sensibilidad

Es la precisión de los positivos o la tasa de verdaderos positivos. (Zeng, 2019, p.9). Definida como:

$$\text{Sensibilidad} = \frac{\text{TP}}{\text{Positivos reales totales}} \tag{3}$$

1.6.10.4. Especificidad

Es la precisión de los negativos o la tasa de verdaderos negativos, Representando la proporción de casos negativos (0) que se clasificaron correctamente. (Zeng, 2019, p.9). Definida como:

$$\text{Especificidad} = \frac{\text{TN}}{\text{Total de Negativos Reales}} \tag{4}$$

1.6.10.5. Exactitud (Accuracy)

Denominada también Exactitud general, es la proporción de verdadero positivo y verdadero negativo al número total de cuentas. (Zeng, 2019, p.9). Se lo calcula:

$$\text{Exactitud} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{5}$$

1.6.11. F1 Score (Precisión)

La precisión o F1 score como se lo conoce en el Machine Learning se utiliza como frecuencia que evalúa problemas de clasificación binaria, de diferentes clases y etiquetas. (Opitz y Burst, 2019, p.1). Esta medida se llama precisión, por concepto, también cuando funciona como etiqueta son más de dos llamado multiclase. Cuando el conjunto de datos se encuentra desequilibrado el número de muestras es mucho mayor que en otras clases, pero no puede denominarse una medida confiable, porque proporciona una aproximación de la capacidad del clasificador o algoritmo inteligente aplicado. (Chicco y Jurman, 2020, p.3).

F1 score combina las medidas de la precisión y el recall transformándolas en un solo valor, haciéndolo practico para demostrar el rendimiento combinado en varias soluciones. Se calcula haciendo la media armónica entre la precisión y la exhaustividad. (Zeng, 2019, p.9). Se la calcula:

$$\text{F1 Score} = 2\left(\frac{\text{Precisión x Recall}}{\text{Precisión + Recall}}\right) \tag{6}$$

1.6.12. Overfitting y Underfitting

Overfitting y Underfitting traducido al español como Sobreajuste y desajuste, aparece finalmente cuando se habla del grado polinomial, dicho grado se representa cuanta flexibilidad en el modelo de machine learning existe, con una potencia alta que permite que el modelo tenga facilidad para alcanzar los puntos de datos como sea lo posible.(Follow, 2018, p.6). Sobreajuste y desajuste son constantes y omnipresentes en el aprendizaje automático, el objetivo del aprendizaje es aproximar o ajustar señales verdaderas que relacionan las características X con su respectiva propuesta en Y, que se pueda interpretar como una función f: $X \rightarrow Y$ o una distribución $P(Y \mid X)$.(Bashir et al, 2020, p.347).

1.6.13. Redes Neuronales Artificiales (ANN)

Las Redes Neuronales Artificiales (ANN) se han convertido en una herramienta computacional popular, ya que es una de las más potentes y versátiles instrumentos de la inteligencia artificial. Las Redes Neuronales Artificiales son mecanismo de computo que brindan un modelo matemático minimalista simulando una neurona cerebral, con una base de datos amplia y un algoritmo de aprendizaje, pueden aplicar modelado de datos para la clasificación y predicción.(Panerati et al. 2019, p.1).

En los sistemas hidráulicos la aplicación de Redes Neuronales Artificiales (ANN) se acoplan como herramienta de prevención y predicción, ya que los sistemas poseen una compleja detección de fallas, que ocultan potentes señales no lineales que varían con el tiempo y el complejo mecanismo de transmisión de las vibraciones; por los tanto aplicar la inteligencia artificial para la detección es una de las mejores alternativas para las industrias que mejorarán la productividad operacional de los equipos. (Dai et al., 2019, p.1).

El modelo matemático consta de:

- X e Y son entradas
- W_1 y W_2 son pesos sinápticos correspondientes a cada entrada
- *b* es un término aditivo
- *f* es una función de activación
- *z* una salida

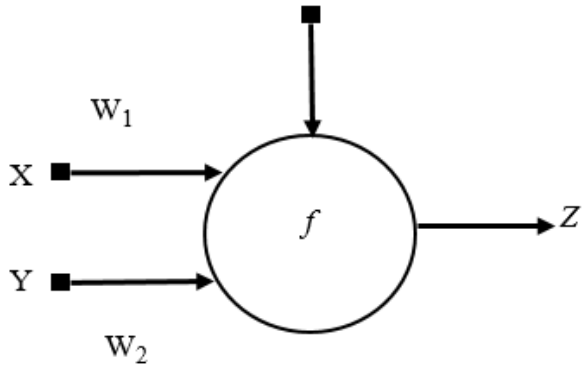


Figura 4-0: Modelo de una Red Neuronal

Fuente: (Tablada -Germán y Torres, 2021)

Las entradas x e y son el estímulo que la neurona artificial recibe del entorno que la rodea, y la salida z es la respuesta a tal estímulo. La neurona se adapta al medio circundante y aprende de él modificando el valor de sus pesos sinápticos w1 y w2 y su término aditivo b. Éstos son conocidos como los parámetros libres del modelo, pues los mismos pueden ser modificados y adaptados para realizar una tarea determinada.(Tablada -Germán y Torres, 2021, p.24). En este modelo, la salida neuronal z está dada por:

$$z = f(w_1 x + w_2 x + b) \tag{7}$$

1.6.13.1. Redes de neuronales de una capa

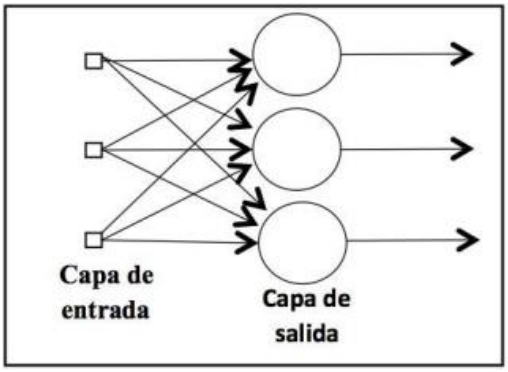


Figura 5-0: Red Neuronal monocapa

Fuente: (Rivas, Bertha y Olivo, 2017)

La red neuronal de una sola capa es la más sencilla de todas ya que posee simplemente una sola capa de entrada lo cual su función es hacer una proyección hacia la capa de salida en la cual existen cálculos necesarios.(Rivas, Bertha y Olivo, 2017, p.19)

1.6.13.2. Redes de neuronales Multicapa

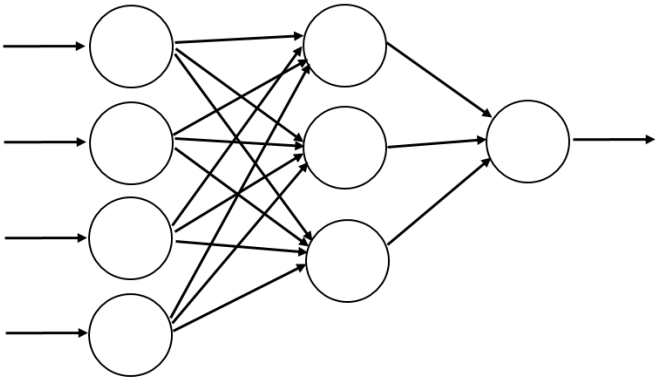


Figura 6-0: Red Neuronal multicapa

Fuente: (Rivas, Bertha y Olivo, 2017)

Es una complemento o aumento de la red mono capa existiendo un conjunto de capas, entre la entrada y la salida qué dichas capas se las denomina capas ocultas, forman una secuencia en la entrada hasta la proyección de la salida mediante los cálculos que se efectúan atreves de dicho proceso. (Rivas, Bertha y Olivo, 2017, p.19)

1.6.14. Librerías de Python

1.6.14.1. TensorFlow

Biblioteca de Python que crea, construye y ejecuta algoritmos de Machine Learning, son ejecutados en un navegador entorno a nodos, la biblioteca es la parte del ecosistema de tensorflow y proporciona un conjunto de API (Interfaz de Programación de Aplicaciones), siendo compatibles con Python, lo que se transfieren a los ecosistemas tanto de Python como JavaScript.(© MLSys Proceedings, 2022)

1.6.14.2. Theano

Biblioteca de Python que optimiza, define y evalúa las expresiones matemáticas que involucran matrices multidimensionales de manera eficiente, usándose como uno de los compiladores matemáticos de CPU (Unidad central de procesamiento) y GPU (Unidad de procesamiento gráfico) más utilizados en la comunidad Machine Learning.(The Theano Development Team et al, 2016, p.1)

1.6.14.3. Keras

Biblioteca de Python de alto nivel que compacta y es sencillo de aprender para emplear en el aprendizaje que puede ejecutarse sobre TensorFlow (Theano o CNTK: kit de herramientas cognitivas de Microsoft). Ésto permite a que los programadores se centren en los conceptos principales del Deep Learning, como la creación de capas en Redes Neuronales Artificiales existiendo dos tipos principales de estructuras: la API secuencial y la API funcional, el secuencial API se basa en la idea de una secuencia de capas; este es el más común. El modelo secuencial se considera como un grupo lineal de capas. (Manaswi, 2018, p.31)

La librería keras importa métricas importantes cuando se construye la red neuronal, a continuación, se denominará las utilizadas:

- Optimizador “Adam”: método que se basa en el descenso de gradiente estocástico que es la estimación que se adaptan momentos de primer y segundo orden. (Manaswi, 2018, p.31)
- Activador “Relu”: Selecciona la cantidad de neuronas ocultas para que no exista un sobreajuste, el activador no causa problema de gradiente de fuga, no se congestiona y es fácil de computar, evitando que el peso cambie su valor.(Alhassan y Zainon, 2021, p.5)
- Activador “Sigmoid”: es una función que tiene una tasa de activación a las neuronas junto con “Relu”, dicha función se utiliza con categorías binarias de predicción. (Pan et al, 2018, p.2)
- Activador “Softmax”: utilizado generalmente para la clasificación de multicategorias en la capa de salida de la red neuronal.(Wei et al, 2020, p.45)

CAPÍTULO II

2. MARCO METODOLÓGICO Y DISEÑO DE LA INVESTIGACIÓN

El presente capítulo se basa en la aplicación de Deep Learning para la detección de fallos, que se utiliza datos previamente obtenidos del repositorio de la Universidad de Carolina, Irvine. Basado en el estudio realizado por la investigación de (Guo et al, 2019), que se pondrá a prueba los datos para realizar un entrenamiento y así aplicar el modelo de Redes Neuronales Artificiales siguiendo pasos del siguiente flujograma:

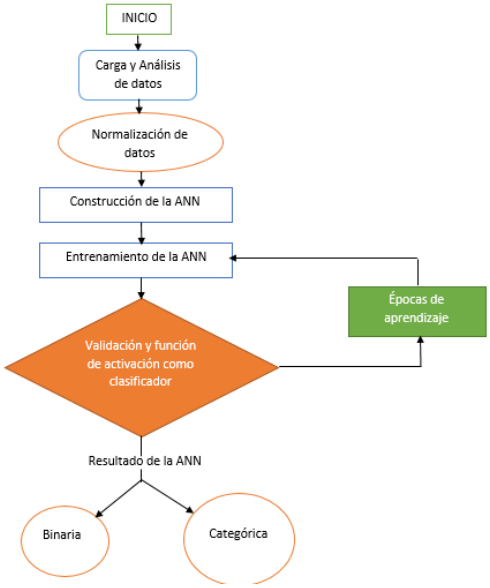


Figura 1-2: Flujograma de aplicación del modelo de ANN

Realizado por: Mera H. 2022

Según la figura 1-2 resume el proceso que se realiza en la aplicación de un modelo inteligente basado en ANN para la detección de fallos en un sistema hidráulico, con la carga y análisis de datos, siguiendo con una normalización para el entrenamiento de los datos para continuar con la construcción de la red neuronal, seguido de su entrenamiento y ejecución para su aprendizaje midiendo métricas y numero de iteraciones o épocas de aprendizaje obteniendo un resultado según la condición de manera binaria y categórica

2.1. Análisis de datos

Se realiza la obtención de datos de Machine Learning Repository de la Universidad de California, Irvine, para la elaboración de la programación y simulación de la aplicación del algoritmo

matemático de las Redes Neuronales Artificiales. La base de datos está destinada a la necesidad de aplicar nuevos métodos de detección o predicción de fallas aplicando métodos de algoritmos de aprendizaje profundo (Deep Learning) para mejorar la productividad de las industrias.

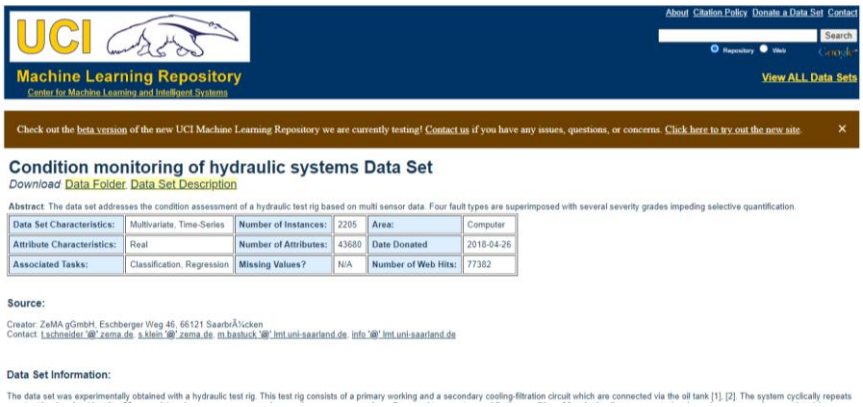


Figura 2-2: Pagina de la toma de datos para el Machine Learning.

Fuente: (Dua, 2017)

2.1.1. Información de atributos del sistema hidráulico.

Tabla 1-2: Características de datos de los sensores del sistema hidráulico

Sensor	Cantidad física	Unidad	Tasa de muestreo (Hz)	Atributos
PS1 - Sensor de presión	Presión	bar	100	6000
PS2 - Sensor de presión	Presión	bar	100	6000
PS3 - Sensor de presión	Presión	bar	100	6000
PS4 - Sensor de presión	Presión	bar	100	6000
PS5 - Sensor de presión	Presión	bar	100	6000
PS6 - Sensor de presión	Presión	bar	100	6000
EPS1 - Sensor de potencia del motor	Potencia de Motor	W	100	6000
FS1 - Sensor de caudal volumétrico	Volumen bajo	1/min	10	600
FS2 - Sensor de caudal volumétrico	Volumen bajo	1/min	10	600
TS1 - Sensor de temperatura	Temperatura	°C	1	60
TS2 - Sensor de temperatura	Temperatura	°C	1	60
TS3 - Sensor de temperatura	Temperatura	°C	1	60
TS4 - Sensor de temperatura	Temperatura	°C	1	60
VS1 - Sensor de vibración	Vibración	mm/s	1	60
CE - Sensor de eficiencia de enfriamiento virtual	Eficiencia de Enfriamiento (virtual)	%	1	60
CP - Sensor de potencia de refrigeración virtual	Poder de Enfriamiento (virtual)	kW	1	60
SE - Factor de eficiencia	Factor de eficiencia	%	1	60

Fuente: (Guo et al, 2019)

Realizado por: Mera H. 2022

Según la tabla 1-2. Los atributos son datos del sensor (todos numéricos y continuos) de mediciones tomadas en el mismo momento, respectivamente, del ciclo de trabajo de un banco de

pruebas hidráulico. Los sensores se leyeron con diferentes velocidades de muestreo, lo que dio lugar a diferentes números de atributos por sensor a pesar de que todos estaban expuestos al mismo ciclo de trabajo. El conjunto de datos contiene datos de sensor de proceso sin procesar (es decir, sin extracción de características) que están estructurados como matrices (delimitadas por tabulaciones) con las filas que representan los ciclos y las columnas los puntos de datos dentro de un ciclo. (Guo et al., 2019, p.3).

2.1.2. Librerías aplicadas para el modelado de datos.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas_profiling
import os
from sklearn.preprocessing import StandardScaler
```

Figura 3-2: Librerías de aplicación de Python

Realizado por: Mera H. 2022

Las librerías son la base de los comandos que ofrece Python para poder realizar la programación y desarrollo del Machine Learning, algunas librerías ya vienen por defecto instaladas y otras se debe realizar la descarga para su correcto funcionamiento según la acción requerida.

2.1.3. Exploración de datos

Para iniciar la exploración se debe descargar la base de datos. Una vez obtenida la base de dato se procede a la carga y la creación de una data set para realizar el desarrollo en el software de lenguaje de programación Python.

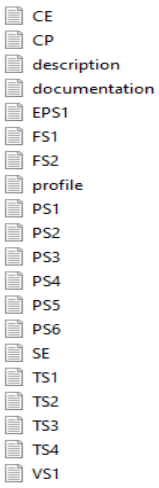


Figura 4-2: Base de datos del sistema hidráulico

Realizado por: Mera H. 2022

2.1.4. Carga de datos del Data Set

La visualización de datos en Python requiere llamar a la carpeta que previamente se descarga con los datos, dichos datos se encuentran en el ordenador, en una ubicación específica, siguiendo dicha ubicación se los importa a Python, los archivos tienen una extensión tipo ".txt". con el comando get_files se realiza el llamado de forma conjunta, junto con lector de archivos ".csv" (el archivo con extensión "csv" o Comma Separeted Values o archivo separado por comas) para el llamado de los archivos a Python.

```
datos= 'C:/Users/Elpnk/Desktop/DATOS DE PROGRAMACION'
```

```
def get_files(datos, filename):
    return pd.read_csv(os.path.join(datos, filename), sep='\t', header=None)
```

Figura 5-2: Llamado de datos a Python

Realizado por: Mera H. 2022

2.1.5. Importación de los datos

La base de datos tiene múltiples variables de datos ya que se pretende hacer un desarrollo de varias clases de implementos censados en el sistema hidráulico, a continuación, se importa los datos agrupados según su función:

- Datos de sensores de presión

```
presion_1 = get_files(datos=datos, filename='PS1.txt')
presion_2 = get_files(datos=datos, filename='PS2.txt')
presion_3 = get_files(datos=datos, filename='PS3.txt')
presion_4 = get_files(datos=datos, filename='PS4.txt')
presion_5 = get_files(datos=datos, filename='PS5.txt')
presion_6 = get_files(datos=datos, filename='PS6.txt')
```

Figura 6-2: Datos de sensores de presión

Realizado por: Mera H. 2022

```
In [5]: presion_1
```

	0	1	2	3	4	5	6	7	8	9	...	5990	5991	5992	5993	5994	5995	5996	5997	5998	5999
0	151.47	151.45	151.52	151.27	150.80	150.69	153.89	154.67	152.88	153.82	...	151.16	151.19	151.25	151.16	151.10	151.16	151.14	151.10	151.21	151.19
1	151.11	151.12	151.16	150.92	150.70	150.62	152.40	153.21	152.81	153.53	...	150.82	150.82	150.86	150.80	150.73	150.79	150.84	150.79	150.80	150.86
2	150.81	150.79	150.84	150.65	150.35	150.23	152.03	152.81	152.44	153.27	...	150.49	150.44	150.47	150.46	150.38	150.47	150.50	150.43	150.54	150.62
3	150.48	150.47	150.52	150.31	150.04	149.98	151.63	152.48	152.24	152.94	...	150.34	150.30	150.28	150.38	150.41	150.33	150.31	150.31	150.25	150.28
4	150.41	150.35	150.24	150.12	149.87	149.71	151.64	152.37	151.78	152.68	...	150.31	150.20	150.17	150.28	150.31	150.25	150.27	150.22	150.13	150.19
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2200	151.70	151.83	151.90	151.75	151.62	151.63	153.78	154.73	153.83	154.43	...	151.84	151.75	151.67	151.76	151.81	151.78	151.78	151.83	151.84	151.86
2201	151.90	151.83	151.81	151.77	151.65	151.63	153.97	154.84	153.67	154.37	...	151.81	151.81	151.84	151.83	151.79	151.87	151.87	151.75	151.77	151.78
2202	151.73	151.78	151.84	151.62	151.46	151.49	153.91	154.88	153.69	154.24	...	151.73	151.66	151.73	151.71	151.68	151.79	151.84	151.73	151.73	151.83
2203	151.77	151.77	151.75	151.56	151.47	151.57	154.09	154.80	153.45	154.21	...	151.75	151.71	151.68	151.71	151.73	151.75	151.71	151.68	151.71	151.78
2204	151.83	151.78	151.77	151.65	151.49	151.48	154.00	154.77	153.48	154.29	...	151.70	151.71	151.63	151.64	151.67	151.64	151.70	151.73	151.71	151.70

Figura 7-2: Visualización de datos de sensores de presión

Realizado por: Mera H. 2022

Se denomina presion_1 a la carpeta que contiene los datos del primer sensor PS1.txt para realizar la extracción de datos del primer sensor. Se repite la misma acción con los demás sensores como se muestra la figura 6-2. Después se comprueba si los datos fueron subidos correctamente como se los muestra en la figura 7-2.

- Datos de sensores de flujo de volumen

```
Flujo_1 = get_files(datos=datos, filename='FS1.txt')
Flujo_2 = get_files(datos=datos, filename='FS2.txt')
```

Figura 8-2: Datos de sensores de flujo de volumen

Realizado por: Mera H. 2022

Se denomina Flujo_1 a la carpeta que contiene los datos del primer sensor FS1.txt para realizar la extracción de datos del primer sensor. Se repite la misma acción con los demás sensores como se muestra la figura 8-2. Después se comprueba si los datos fueron subidos correctamente como se los muestra en la figura a continuación:

```
Flujo_2
```

	0	1	2	3	4	5	6	7	8	9	...	590	591	592	593	594	595	596	597	598
0	10.179	10.174	10.151	10.149	10.172	10.176	10.169	10.176	10.174	10.171	...	10.413	10.399	10.397	10.384	10.401	10.407	10.395	10.374	10.379
1	10.408	10.429	10.415	10.418	10.401	10.403	10.408	10.416	10.398	10.417	...	10.438	10.411	10.419	10.414	10.407	10.391	10.427	10.411	10.434
2	10.392	10.386	10.404	10.391	10.387	10.422	10.414	10.414	10.441	10.434	...	10.320	10.352	10.356	10.336	10.338	10.327	10.337	10.350	10.356
3	10.329	10.328	10.349	10.363	10.359	10.328	10.333	10.341	10.371	10.329	...	10.299	10.296	10.283	10.256	10.270	10.272	10.280	10.285	10.267
4	10.276	10.279	10.292	10.288	10.266	10.271	10.284	10.283	10.294	10.267	...	10.199	10.199	10.233	10.245	10.233	10.211	10.205	10.214	10.227
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2200	10.196	10.189	10.180	10.201	10.176	10.200	10.185	10.193	10.182	10.190	...	10.193	10.174	10.170	10.176	10.170	10.170	10.194	10.187	10.154
2201	10.182	10.194	10.189	10.181	10.186	10.167	10.166	10.198	10.171	10.166	...	10.172	10.182	10.191	10.185	10.176	10.170	10.171	10.190	10.187
2202	10.171	10.169	10.178	10.182	10.203	10.171	10.177	10.187	10.171	10.183	...	10.189	10.174	10.169	10.150	10.162	10.183	10.182	10.185	10.185
2203	10.166	10.194	10.189	10.180	10.173	10.163	10.162	10.185	10.189	10.181	...	10.187	10.198	10.185	10.185	10.189	10.189	10.172	10.165	10.184
2204	10.193	10.174	10.172	10.178	10.203	10.203	10.170	10.179	10.166	10.176	...	10.182	10.180	10.163	10.167	10.194	10.193	10.171	10.167	10.173

2205 rows × 600 columns

Figura 9-2: Visualización de datos de sensores de flujo de volumen

Realizado por: Mera H. 2022

- Datos de sensores de temperatura

```
temperatura1 = get_files(datos=datos, filename='TS1.txt')
temperatura2 = get_files(datos=datos, filename='TS2.txt')
temperatura3 = get_files(datos=datos, filename='TS3.txt')
temperatura4 = get_files(datos=datos, filename='TS4.txt')
```

Figura 10-2: Datos de sensores de temperatura

Realizado por: Mera H. 2022

Se denomina temperatura_1 a la carpeta que contiene los datos del primer sensor TS1.txt para realizar la extracción de datos del primer sensor. Se repite la misma acción con los demás sensores

como se muestra la figura 10-2. Después se comprueba si los datos fueron subidos correctamente como se los muestra en la figura a continuación:

```
temperatura1
```

	0	1	2	3	4	5	6	7	8	9	...	50	51	52	53	54	55	56	57	58
0	35.570	35.492	35.469	35.422	35.414	35.320	35.227	35.242	35.160	35.176	...	36.008	35.984	35.996	36.039	36.008	36.008	36.094	36.102	36.090
1	36.156	36.094	35.992	36.008	35.992	35.902	35.824	35.820	35.727	35.727	...	37.328	37.324	37.340	37.332	37.316	37.410	37.418	37.422	37.488
2	37.488	37.391	37.340	37.312	37.223	37.145	37.059	36.973	36.898	36.879	...	38.457	38.461	38.457	38.469	38.469	38.555	38.527	38.543	38.527
3	38.633	38.535	38.469	38.379	38.297	38.223	38.125	38.062	37.977	37.969	...	39.441	39.363	39.367	39.457	39.461	39.461	39.473	39.441	39.453
4	39.461	39.461	39.375	39.281	39.203	39.113	39.043	38.969	38.875	38.883	...	40.324	40.320	40.312	40.340	40.320	40.387	40.391	40.391	40.387
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2200	35.414	35.348	35.254	35.262	35.168	35.172	35.105	35.094	35.000	35.008	...	35.426	35.516	35.437	35.426	35.410	35.414	35.414	35.426	35.414
2201	35.434	35.332	35.266	35.195	35.262	35.195	35.082	35.105	34.988	35.008	...	35.434	35.434	35.449	35.414	35.422	35.414	35.426	35.437	35.402
2202	35.434	35.355	35.262	35.187	35.266	35.102	35.105	35.094	35.016	35.016	...	35.500	35.434	35.410	35.434	35.426	35.437	35.441	35.410	35.434
2203	35.449	35.359	35.277	35.184	35.195	35.184	35.105	35.016	35.012	35.008	...	35.437	35.531	35.426	35.516	35.422	35.441	35.461	35.414	35.426
2204	35.422	35.336	35.250	35.176	35.195	35.094	35.102	35.094	35.012	35.012	...	35.437	35.453	35.520	35.437	35.437	35.441	35.437	35.434	35.434

2205 rows × 60 columns

Figura 11-2: Visualización de datos de sensores de temperatura

Realizado por: Mera H. 2022

- Datos de Sensores de eficiencia de la bomba, vibraciones, eficiencia de enfriamiento, potencia de enfriamiento, factor de eficiencia.

```
bomba = get_files(datos=datos, filename='EPS1.txt')
vibracion1 = get_files(datos=datos, filename='VS1.txt')
enfriamiento_CE = get_files(datos=datos, filename='CE.txt') # Sensor de eficiencia de enfriamiento
enfriamiento_CP = get_files(datos=datos, filename='CP.txt') # Sensor de potencia de refigeracion
factor_efi = get_files(datos=datos, filename='SE.txt')
```

Figura 12-2: Datos de sensores de: Bomba, Vibraciones, Enfriamiento y Eficiencia

Realizado por: Mera H. 2022

Se denomina bomba a la carpeta que contiene los datos del sensor EPS1.txt para realizar la extracción de datos del sensor de la bomba. Se repite la misma acción con los demás sensores como se muestra la figura 12-2. Después se comprueba si los datos fueron subidos correctamente como se los muestra en la figura a continuación:

```
bomba
```

	0	1	2	3	4	5	6	7	8	9	...	5990	5991	5992	5993	5994	5995	5996	5997	5998
0	2411.6	2411.6	2411.6	2411.6	2411.6	2411.6	2411.6	2411.6	2411.6	2409.6	...	2409.6	2409.2	2409.6	2409.4	2409.6	2409.4	2409.6	2409.6	2409.6
1	2409.6	2409.6	2409.6	2409.6	2409.6	2409.6	2409.6	2409.6	2409.6	2409.6	...	2398.8	2398.2	2398.2	2398.0	2398.0	2398.0	2398.0	2397.8	2397.8
2	2397.8	2397.8	2397.8	2397.8	2397.8	2397.8	2397.8	2397.8	2397.8	2395.8	...	2383.8	2383.8	2383.8	2383.8	2383.8	2383.8	2383.8	2383.8	2383.8
3	2383.8	2383.8	2383.8	2383.8	2383.8	2383.8	2383.8	2383.8	2382.8	2382.8	...	2373.2	2372.8	2372.6	2372.4	2372.2	2372.0	2372.0	2372.0	2372.0
4	2372.0	2372.0	2372.0	2372.0	2372.0	2372.0	2372.0	2372.0	2372.0	2373.0	...	2370.0	2370.0	2369.8	2369.8	2369.8	2369.8	2369.6	2369.6	2369.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2200	2416.4	2416.4	2416.4	2416.4	2416.4	2416.4	2416.4	2416.4	2416.4	2416.4	...	2415.6	2415.2	2415.6	2415.4	2415.6	2415.4	2415.6	2415.6	2415.6
2201	2415.6	2415.6	2415.6	2415.6	2415.6	2415.6	2415.6	2415.6	2415.6	2415.6	...	2414.6	2414.6	2414.0	2414.0	2413.8	2413.8	2413.6	2413.6	2413.6
2202	2413.6	2413.6	2413.6	2413.6	2413.6	2413.6	2413.6	2413.6	2413.6	2413.6	...	2413.6	2413.6	2412.6	2413.6	2413.2	2413.6	2413.4	2413.6	2413.4
2203	2413.6	2413.6	2413.6	2413.6	2413.6	2413.6	2413.6	2413.6	2413.6	2413.6	...	2413.8	2413.8	2413.8	2414.8	2415.8	2415.2	2415.6	2415.4	2415.6
2204	2415.8	2415.6	2415.6	2415.6	2415.6	2415.6	2415.6	2415.6	2415.6	2415.6	...	2413.8	2413.8	2413.8	2414.8	2415.8	2416.2	2416.6	2416.8	2417.0

2205 rows × 6000 columns

Figura 13-2: Visualización de datos de la eficiencia de la bomba

Realizado por: Mera H. 2022

- Datos de la etiqueta o perfil

```
perfil = get_files(datos=datos, filename='profile.txt')
```

Figura 14-2: Datos de sensores del Perfil o Parámetro de predicción

Realizado por: Mera H. 2022

Se denomina perfil a la carpeta que contiene los datos del perfil profile.txt para realizar la extracción de datos, los datos del perfil son las condiciones que presenta la base de datos. Se comprueba si los datos fueron subidos correctamente como se los muestra en la figura 15-2, en la figura marca las condiciones en números enteros, que se representan de la siguiente manera: la fila 1 pertenece a la Condición de enfriamiento, la fila 2 a la Condición de la válvula, la fila 3 a la Fuga interna de la bomba, la fila 4 al Acumulador hidráulico y la fila 5 la Estabilidad del sistema hidráulico.

```
perfil
```

	0	1	2	3	4
0	3	100	0	130	1
1	3	100	0	130	1
2	3	100	0	130	1
3	3	100	0	130	1
4	3	100	0	130	1
...	...	...	...	...	...
2200	100	100	0	90	0
2201	100	100	0	90	0
2202	100	100	0	90	0
2203	100	100	0	90	0
2204	100	100	0	90	0

2205 rows × 5 columns

Figura 15-2: Datos para la predicción

Realizado por: Mera H. 2022

2.1.6. Procesamientos de datos

2.1.6.1. Definición de variables

```
coolerCondition = pd.DataFrame(perfil.iloc[:, 0])
valveCondition = pd.DataFrame(perfil.iloc[:, 1])
pumpLeak = pd.DataFrame(perfil.iloc[:, 2])
hydraulicAcc = pd.DataFrame(perfil.iloc[:, 3])
stableFlag = pd.DataFrame(perfil.iloc[:, 4])
```

Figura 16-2: Datos para la predicción

Realizado por: Mera H. 2022

Se define variables “y” para la predicción según la condición que se muestra la figura 16-2, definiéndose como: “Condición de enfriamiento”, “Condición de la válvula”, “Eficiencia de la

bomba", "Acumulador hidráulico" y "Estabilidad", llamándolos con la función ".iloc" para la denominación de la fila a la que pertenecen en los datos del perfil figura 15-2. Cada condición posee sus propias condiciones convirtiéndolas en categóricas y solo la estabilidad de manera binaria que ayudara para el análisis y la implementación de la red neuronal. Como se muestra a continuación la descripción de cada condición a evaluar:

```
Condicion de enfriamiento / %:
    3: cerca del fallo total (732 instancias)
    20: eficiencia reducida (732 instancias)
    100: full eficiencia (741 instancias)

Condicion de la valvula / %:
    100: comportamiento de conmutación óptimo (1125 instancias)
    90: pequeño retraso (360 instancias)
    80: severo retraso (360 instancias)
    73: close to total failure (360 instancias)

Fuga interna de la bomba:
    0: ninguna fuga (1221 instancias)
    1: fuga débil (492 instancias)
    2: fuga severa (492 instancias)

acumulador hidraulico / bar:
    130: presión óptima (599 instancias)
    115: presión ligeramente reducida (399 instancias)
    100: presión severamente reducida (399 instancias)
    90: cerca de la falla total(808 instancias)

Estabilidad:
    0: condiciones eran estables (1449 instancias)
    1: Es posible que aún no se hayan alcanzado las condiciones estáticas. (756 instancias)
```

Figura 17-2: Condiciones de parámetros de predicción

Realizado por: Mera H. 2022

Una vez planteando como se realiza el análisis para el modelo se continúa con el procesamiento de los datos.

2.1.6.2. Promediar datos

Se promedia todos los datos ya que existen varios ciclos de ingreso utilizando la función "Mean".

```
def Mean(DataFrame):
    DataFrame=DataFrame.mean(axis=1)
    return DataFrame
```

Figura 18-2: Función Mean

Realizado por: Mera H. 2022

2.1.6.3. Conversión de datos

Se organiza todos los datos para su respectiva media, por cada clase de sensor se los denomina según su función y se los va llamando uno por uno, iniciando con los datos de presión, flujo de volumen, temperatura, enfriamiento y los datos finales.

```
# datos de presión

presion1=Mean(presion_1)
presion2=Mean(presion_2)
presion3=Mean(presion_3)
presion4=Mean(presion_4)
presion5=Mean(presion_5)
presion6=Mean(presion_6)

# datos de flujo de volumen

Flujo1=Mean(Flujo_1)
Flujo2=Mean(Flujo_2)
```

```
# datos de temperatura
temperatura1=Mean(temperatura_1)
temperatura2=Mean(temperatura_2)
temperatura3=Mean(temperatura_3)
temperatura4=Mean(temperatura_4)

#datos de enfriamiento

enfriamientoCE=Mean(enfriamiento_CE)
enfriamientoCP=Mean(enfriamiento_CP)

# Otros datos

bomba_eficiencia=Mean(bomba)
vibracion_1=Mean(vibracion1)
factorefi=Mean(factor_efi)
```

Figura 19-2: Conversión de datos según su función.

Realizado por: Mera H. 2022

2.1.6.4. Agrupación de datos

Todos los datos son agrupados y asignados un nombre según su función o variable al que pertenecen, todos los datos son almacenados en una variable general llamada Data convirtiéndose en la concatenación de todos los datos para un nuevo Dataset.

```
Data=pd.DataFrame()

# datos de presión

Data['PS1']=presion1
Data['PS2']=presion2
Data['PS3']=presion3
Data['PS4']=presion4
Data['PS5']=presion5
Data['PS6']=presion6

# datos de flujo de volumen

Data['FS1']=Flujo1
Data['FV2']=Flujo2

# datos de temperatura

Data['TS1']=temperatura1
Data['TS2']=temperatura2
Data['TS3']=temperatura3
Data['TS4']=temperatura4

#datos de enfriamiento

Data['CE']=enfriamientoCE # Sensor de eficiencia de enfriamiento
Data['CP']=enfriamientoCP # Sensor de potencia de refigeracion

# Otros datos

Data['EPS1']=bomba_eficiencia
Data['VS1']=vibracion_1
Data['SE']=factorefi

# datos de Prediccion

Data['PL']=pumpLeak
Data['CD']=coolerCondition
Data['VC']=valveCondition
Data['HA']=hydraulicAcc
Data['Stable']=stableFlag
```

Figura 20-2: Agrupación de datos.

Realizado por: Mera H. 2022

Una vez agrupados los datos se procede a la visualización en una sola tabla llamada “Data”.

```
In [19]: Data
```

PS2	PS3	PS4	PS5	PS6	FS1	FV2	TS1	TS2	...	CE	CP	EPS1	VS1	SE	PL	CD	VC	HA	Stab
6914	1.991475	0.000000	9.842170	9.728097	6.709815	10.304592	35.621983	40.978767	...	39.601350	1.862750	2538.929167	0.576950	59.157183	0	3	100	130	
4890	1.976234	0.000000	9.635142	9.529488	6.715315	10.403098	36.676967	41.532767	...	25.786433	1.255550	2531.498900	0.565850	59.335617	0	3	100	130	
8845	1.972224	0.000000	9.530548	9.427949	6.718522	10.366250	37.880800	42.442450	...	22.218233	1.113217	2519.928000	0.576533	59.543150	0	3	100	130	
4807	1.946575	0.000000	9.438827	9.337430	6.720565	10.302678	38.879050	43.403983	...	20.459817	1.062150	2511.541633	0.569267	59.794900	0	3	100	130	
1434	1.922707	0.000000	9.358762	9.260636	6.690308	10.237750	39.803917	44.332750	...	19.787017	1.070467	2503.449500	0.577367	59.455267	0	3	100	130	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9581	2.001438	10.202473	9.972037	9.850361	6.689930	10.184515	35.313783	40.874800	...	46.628517	2.160600	2543.911033	0.550833	59.033100	0	100	100	90	
7481	1.998781	10.197919	9.966184	9.844864	6.692182	10.177767	35.321600	40.868883	...	46.689817	2.151450	2543.411333	0.547483	59.068000	0	100	100	90	
6174	1.993436	10.196824	9.964329	9.842628	6.693277	10.176172	35.319183	40.875950	...	46.472300	2.143300	2542.729767	0.545233	59.132350	0	100	100	90	
3884	2.007077	10.198588	9.968232	9.846690	6.684128	10.178363	35.324767	40.876067	...	46.544967	2.148483	2544.046333	0.537017	58.970800	0	100	100	90	
2177	2.002690	10.203126	9.973838	9.851949	6.692302	10.183393	35.322233	40.869400		46.647933	2.157050	2543.818300	0.546583	59.053900	0	100	100	90	

Figura 21-2: Agrupación de datos en una sola tabla.

Realizado por: Mera H. 2022

2.1.6.5. Información de variables

```
Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2205 entries, 0 to 2204
Data columns (total 22 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   PS1     2205 non-null   float64
 1   PS2     2205 non-null   float64
 2   PS3     2205 non-null   float64
 3   PS4     2205 non-null   float64
 4   PS5     2205 non-null   float64
 5   PS6     2205 non-null   float64
 6   FS1     2205 non-null   float64
 7   FV2     2205 non-null   float64
 8   TS1     2205 non-null   float64
 9   TS2     2205 non-null   float64
 10  TS3     2205 non-null   float64
 11  TS4     2205 non-null   float64
 12  CE      2205 non-null   float64
 13  CP      2205 non-null   float64
 14  EPS1    2205 non-null   float64
 15  VS1     2205 non-null   float64
 16  SE      2205 non-null   float64
 17  PL      2205 non-null   int64
 18  CD      2205 non-null   int64
 19  VC      2205 non-null   int64
 20  HA      2205 non-null   int64
 21  Stable  2205 non-null   int64
dtypes: float64(17), int64(5)
memory usage: 379.1 KB
```

Figura 22-2: Información de los datos según las variables.

Realizado por: Mera H. 2022

Según la figura 22-2, se observa un total de 21 variables incluidas las variables de predicción, las 17 varias son los sensores que conforman el sistema hidráulico de tipo float64(números reales o con comas) y los 5 sobrantes son las condiciones de predicción de tipo int64(números enteros).

2.1.6.6. Observaciones y números faltantes

Se debe observar los números faltantes del conjunto de datos para realizar una correcta aplicación del Machine Learning, ya que podrían verse afectado en reducir el rendimiento del modelo.

```
Data.isnull().sum()
```

```
PS1       0
PS2       0
PS3       0
PS4       0
PS5       0
PS6       0
FS1       0
FV2       0
TS1       0
TS2       0
TS3       0
TS4       0
CE        0
CP        0
EPS1      0
VS1       0
SE        0
PL        0
CD        0
VC        0
HA        0
Stable    0
dtype: int64
```

```
Data.shape
```

```
(2205, 22)
```

Figura 23-2: Números faltantes y observaciones del conjunto de datos

Realizado por: Mera H. 2022

Se observa la figura 23-2, que no existen observaciones ni valores faltantes y existen 2205 filas de datos con sus respectivos 22 variables en columnas, lo cual no existe la necesidad de anular o degradar valores para la aplicación del modelo.

2.1.6.7. Visualización de variables

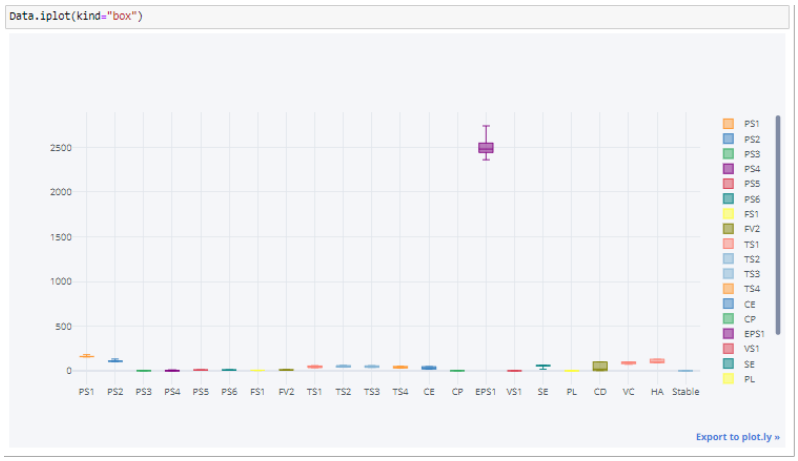


Gráfico 1-2: Tabla general de datos agrupados

Realizado por: Mera H. 2022

Según el gráfico 1-2 se puede observar como los datos se posicionan, pero no siguen ninguna distribución, a continuación, se demuestra en un histograma para demostrar la irregularidad y que no siguen la forma de la distribución normal.

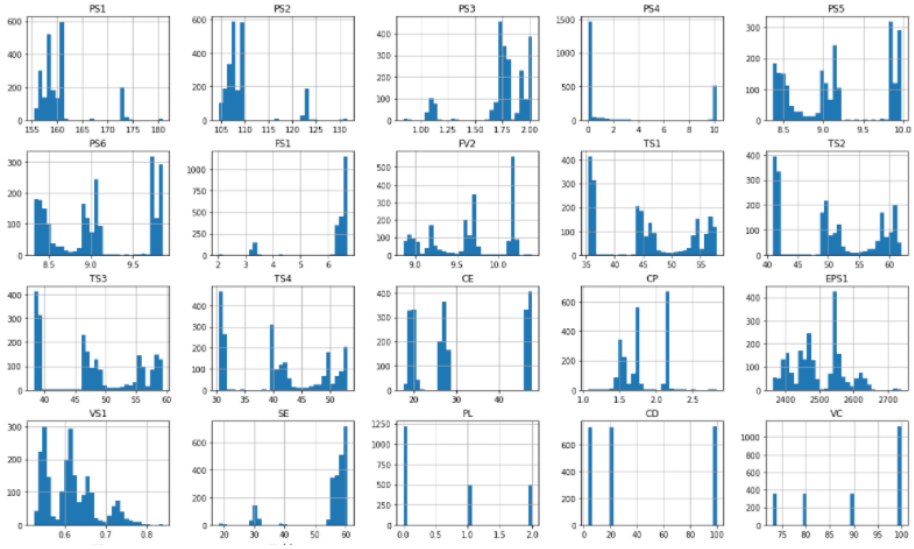


Gráfico 2-2: Histograma de distribución de datos.

Realizado por: Mera H. 2022

2.1.6.8. Correlación de datos

Se grafica un mapa de calor que se lo analiza como correlación de datos que demuestra qué variables son las que más correlacionan entre sí, según la posición que poseen se los denomina entre 1 y -1, los variables que llegan a 1 se correlacionan entre sí positivamente, si se aproxima a -1 la correlación se convierte en negativa. siguiendo el gráfico 3-2, a continuación, se observa que existe un posicionamiento bajo en correlación ya sea positiva o negativamente.

Gráfico 3-2: Mapa de calor

Realizado por: Mera H. 2022

2.1.7. Análisis de correlación de datos según su condición

Antes de construir la red neuronal artificial se debe seleccionar las mejores funciones de correlación aplicadas a los sensores o variables para demostrar la influencia que aportan a las condiciones de predicción, la figura 17-2.

2.1.7.1. Selección de funciones de correlación de datos

Para realizar la selección de funciones de correlación de datos se inicia eliminando todas las variables de predicción, para la cual se aplica la función Feature_seleccion con la librería Sklearn, que mide la importancia de contribución de cada variable para cada condición.

```
from sklearn.feature_selection import mutual_info_classif
importances=mutual_info_classif(X2,y2)
```

Figura 24-2: Selección de features

Realizado por: Mera H. 2022

A continuación, se demuestra la contribución que posee cada variable para cada condición:

- Selección de las mejores features (funciones) para la Estabilidad (Stable Flag)

```
X=Data.drop(['Stable','PL','CD','VC','HA'],1)
y=Data['Stable']
```

Figura 25-2: Asignación de variable "y"

Realizado por: Mera H. 2022

Una vez eliminadas las variables de predicción de las variables "x" con la función ".drop", se las denomina con una nueva variable "y". la nueva denominación de variables será primordial para que el modelo de inteligencia artificial pueda entrenarse y entregar los resultados deseados. se aplica la función de selección de features, según la figura 24-2, comenzando con la estabilidad (stable) como se observa la figura 25-2, primera condición de la predicción demostrando cual es la contribución que ofrece para las variables "x", sino caso contrario podrían ser eliminadas ya que se convierten en valores irrelevantes para la predicción de la estabilidad que pueden ser eliminados para el análisis de aplicación del modelo de inteligencia artificial.

Según se observa el gráfico 4-2, que CP (Poder de enfriamiento) y PS4 (Sensor de presión 4) no existe una contribución sustancial siendo menores a 15% y convirtiéndose en valores despreciables que no afectan para la evaluación de la estabilidad

```
PS1     0.279429
PS2     0.269172
PS3     0.189773
PS4     0.099495
PS5     0.392179
PS6     0.390753
FS1     0.299570
FV2     0.294233
TS1     0.408156
TS2     0.374132
TS3     0.390146
TS4     0.392441
CE      0.196252
CP      0.123257
EPS1    0.271929
VS1     0.171554
SE      0.285827
dtype: float64
```

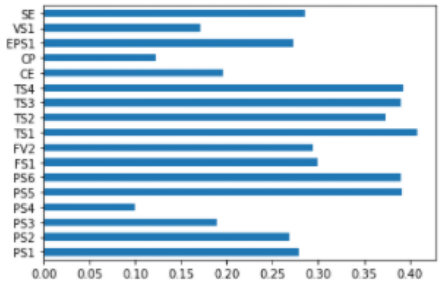


Gráfico 4-2: Features para la Estabilidad

Realizado por: Mera H. 2022

- Selección de las mejores features (funciones) para la condición de la válvula (Valve Condition)

```
X2=Data.drop(['Stable','PL','CD','VC','HA'],1)
y2=Data['VC']
```

Figura 26-2: Asignación de variable "y2"

Realizado por: Mera H. 2022

```
PS1     0.580035
PS2     0.591373
PS3     0.140037
PS4     0.088636
PS5     0.353748
PS6     0.354115
FS1     0.371377
FV2     0.252797
TS1     0.436819
TS2     0.406083
TS3     0.439246
TS4     0.376826
CE      0.129169
CP      0.062581
EPS1    0.254181
VS1     0.120241
SE      0.415622
dtype: float64
```

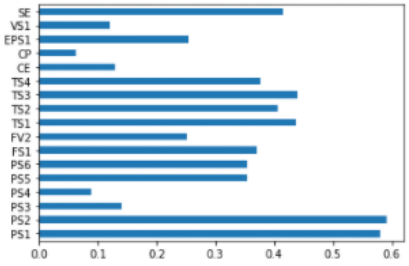


Gráfico 5-2: Features para la Condición de la Válvula

Realizado por: Mera H. 2022

La segunda variable “y2” es de la condición de la válvula (VC) como se muestra la figura 26-2, demostrando la contribución que las variables ofrecen para el análisis de la condición. a continuación, según el gráfico 5-2, se observa que CP (Poder de enfriamiento), CE (Eficiencia de enfriamiento) y PS3, PS4 (Sensor de presión 3 y 4) no existe una contribución sustancial siendo menores a 15% y convirtiéndose en valores despreciables que no afectan para la evaluación de Condición de la válvula que pueden ser eliminados para el análisis de aplicación del modelo de inteligencia artificial.

- Selección de las mejores features (funciones) para la condición de la bomba (Pumbleak)

```
X3=Data.drop(['Stable','PL','CD','VC','HA'],1)
y3=Data['PL']
```

Figura 27-2: Asignación de variable “y3”

Realizado por: Mera H. 2022

La tercera variable “y3” es la condición de la bomba (PL) como se muestra la figura 27-2, demostrando la contribución que las variables ofrecen para el análisis de la condición. a continuación, según el gráfico 6-2, se observa que CP (Poder de enfriamiento), CE (Eficiencia de enfriamiento) PS4 (Sensor de presión 4) no existe una contribución sustancial siendo menores a 15% y convirtiéndose en valores despreciables que no afectan para la evaluación de condición de la Bomba que pueden ser eliminados para el análisis de aplicación del modelo de inteligencia artificial.

```
PS1     0.578333
PS2     0.443895
PS3     0.571101
PS4     0.130461
PS5     0.426075
PS6     0.427733
FS1     0.845264
FV2     0.321487
TS1     0.475361
TS2     0.470623
TS3     0.472574
TS4     0.489811
CE      0.136940
CP      0.149822
EPS1    0.630895
VS1     0.188347
SE      0.847806
dtype: float64
```

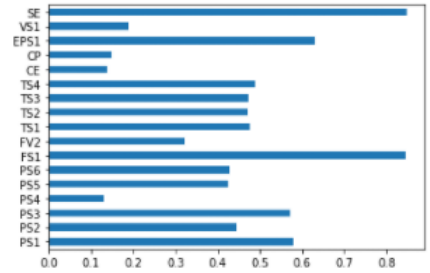


Gráfico 6-2: Features para la Condición de la Bomba

Realizado por: Mera H. 2022

- Selección de las mejores features (funciones) para la Condición de Enfriamiento (Coolercondition)

```
X4=Data.drop(['Stable','PL','CD','VC','HA'],1)
y4=Data['CD']
```

Figura 28-2: Asignación de variable "y4"

Realizado por: Mera H. 2022

La cuarta variable "y4" la Condición de enfriamiento (CD) como se muestra la figura 28-2, demostrando la contribución que las variables ofrecen para el análisis de la condición. a continuación, según el gráfico 7-2, se observa que todas las variables poseen una contribución sustancial mayor al 15%, es decir que no hay necesidad de eliminar ninguna variable "x" ya que influyen para el análisis de la condición de enfriamiento aplicando el modelo.

```
PS1     0.992029
PS2     0.962200
PS3     0.843577
PS4     0.572369
PS5     1.032893
PS6     1.032207
FS1     0.709197
FV2     0.999847
TS1     1.017553
TS2     1.005907
TS3     1.003467
TS4     1.037901
CE      1.088549
CP      1.083736
EPS1    0.979578
VS1     0.987400
SE      0.597018
dtype: float64
```

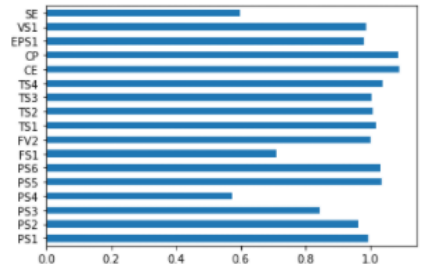


Gráfico 7-2: Features para la Condición de Enfriamiento

Realizado por: Mera H. 2022

- Selección de las mejores features (funciones) para la condición de la bomba (Pumbleak)

```
X5=Data.drop(['Stable','PL','CD','VC','HA'],1)
y5=Data['HA']
```

Figura 29-2: Asignación de variable "y5"

Realizado por: Mera H. 2022

La quinta variable "y" Acumulador hidráulico (ha) como se muestra en la figura 29-2, demostrando la contribución que las variables ofrecen para el análisis de la condición. a

continuación, según el gráfico 8-2, se observa que todas las variables poseen una contribución sustancial mayor al 15%, es decir que no hay necesidad de eliminar ninguna variable “x” ya que influyen para el análisis de la condición de enfriamiento aplicando el modelo.

```
PS1     0.472861
PS2     0.455874
PS3     0.307737
PS4     0.245625
PS5     0.779220
PS6     0.762187
FS1     0.311277
FV2     0.739195
TS1     0.841404
TS2     0.814709
TS3     0.824351
TS4     0.834841
CE      0.418430
CP      0.329156
EPS1    0.549550
VS1     0.329290
SE      0.325855
dtype: float64
```

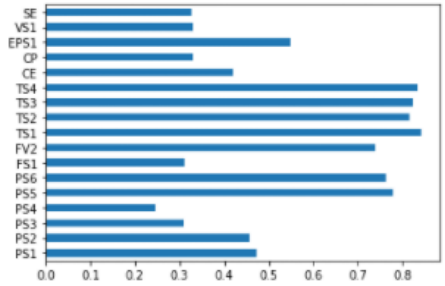


Gráfico 8-2: Features para la Condición de Enfriamiento

Realizado por: Mera H. 2022

2.1.7.2. Eliminación de variables según la condición

Una vez aplicado prueba y error se escoge las mejores funciones para todas las condiciones del sistema para una mejora en el aprendizaje y la predicción de la red neuronal, con la función “feat_imp” figura se escoge las funciones mayores al 40% en las 4 condiciones menos en la estabilidad ya que no presenta una mejora tan significativa

```
1  ### features mejor correlacionados
2  feat_imp[feat_imp>0.4].index
```

```
Index(['PS1', 'PS2', 'TS1', 'TS2', 'TS3', 'SE'], dtype='object')
```

Figura 30-2: Función de mejores correlaciones

Realizado por: Mera H. 2022

2.1.7.3. Transformación a binarios las condiciones

Para que exista una predicción se necesita que los datos “y” de cada una de la condición se transformen a binarios, en este caso la condición de la estabilidad como es una categoría binaria

no se necesita realizar dicha transformación, pero en cambio con las demás condiciones si ya que poseen subcondiciones y se convierten en variables categóricas.

```
y02 = pd.get_dummies(y2.values.ravel())
# print(y02)
clases = list(y02.columns)
y02
```

Figura 31-2: Función get_dummies

Realizado por: Mera H. 2022

Con la función "get_dummies", figura 31-2, se transforma a código binario las condiciones categóricas antes de ser puestas en marcha en el algoritmo inteligente, se denomina clases sin alterar el número de columnas ni de filas, figura 32-2, para ordenar de manera secuencial las subcategorías y pueda existir un enteramiento correcto de la red neuronal.

	73	80	90	100
0	0	0	0	1
1	0	0	0	1
2	0	0	0	1
3	0	0	0	1
4	0	0	0	1
...	...	...	...	...
2200	0	0	0	1
2201	0	0	0	1
2202	0	0	0	1
2203	0	0	0	1
2204	0	0	0	1

2205 rows × 4 columns

Figura 32-2: Función get_dummies

Realizado por: Mera H. 2022

2.1.8. División de datos de entrenamiento, validación y prueba

2.1.8.1. Datos de entrenamiento y prueba

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=0)
```

Figura 33-2: Variables de entrenamiento y prueba.

Realizado por: Mera H. 2022

Es muy importante considerar datos propuestos, que proponga un porcentaje de prueba y entrenamiento para que el modelo tenga la probabilidad de iniciar con el máximo error y terminar

con el mínimo, es decir el entrenamiento del modelo ayuda a que pueda tener un desenvolvimiento previo antes de ser aplicado a la realidad, simulando la menor cantidad de errores cometidos en el aprendizaje del entrenamiento, en este caso se determinará con un 80:20 es decir la prueba se realiza primero un 20% de entrenamiento y 80% de testeo lo cual se tomara como guía para aplicar testeo, para alcanzar la mejor precisión posible del modelo aplicado a la detección de fallos en sistemas hidráulicos.

Según la figura 33-2, se utiliza la librería de skaearn, el train_test_split que automáticamente simula la declaración de las variables que serán sometidas al modelado de la red neuronal artificial, se declara la variable de la primera condición, que se aplica el mismo procedimiento para las demás condiciones de predicción, únicamente cambiando de variables según la condición en este caso X e y que se esté modelando. Siendo variables de entrenamiento (X_train) y prueba (y_test), utilizados para cuando se aplique el modelo de Machine Learning, además se observa que con la función test_size se denomina el porcentaje de entrenamiento del modelo y random_state la cantidad de semillas aleatorias que se crearán según el aprendizaje del algoritmo.

Se imprime los datos de entrenamiento que serán destinados como la variable independiente como se observa en la figura 34-2, es decir que son los datos que va analizar la red neuronal para combinar con la variable dependiente con los datos que se observa en la figura 35-2, los que serían utilizados para realizar la predicción según la condición la cual se esté aplicando, dicho proceso aplica para las 5 condiciones a obtener, ya que lo único que cambia es la condición de predicción(y_test) del sistema hidráulico.

```
datos de entrenamiento            PS1        PS2      PS3       PS4      PS5       PS6  \
572   156.259090  105.109720  1.686528   0.000000  8.409337  8.359900
117   157.029422  106.361600  1.729071   0.000000  8.592279  8.533665
1161  159.114915  107.267203  1.803048   0.000000  9.136608  9.051796
789   158.176940  107.214707  1.786352   0.000000  8.970626  8.898110
1322  158.989445  107.983810  1.806357   0.000158  9.146062  9.061882
...          ...         ...       ...        ...       ...       ...
1033  158.867803  107.103200  1.789223   0.000000  9.054415  8.976108
1731  160.771032  109.054252  1.935512  10.112827  9.880726  9.761596
763   157.909970  106.973936  1.779226   0.000000  8.924975  8.852112
835   158.262197  107.266625  1.803428   0.000000  8.985091  8.909845
1653  161.047945  109.579380  2.008050   0.785936  9.937681  9.819300

          FS1        FV2        TS1        TS2        TS3        TS4  \
572   6.572717   8.932982  57.330050  61.493300  58.674333  52.562750
117   6.601323   9.239225  53.610750  58.093367  55.153417  49.000933
1161  6.632630   9.690735  44.593617  49.634633  46.804117  40.021017
789   6.634915   9.604517  47.109183  52.262500  49.325900  42.666800
1322  6.672445   9.723228  44.389533  49.417183  46.637700  39.878583
...        ...        ...        ...        ...        ...        ...
1033  6.643458   9.662503  45.757833  50.665533  47.919200  41.195717
1731  6.500505  10.175213  35.897400  41.499200  38.796267  30.888767
763   6.626108   9.544212  47.881850  53.009617  50.051850  43.319533
835   6.636710   9.607130  46.754067  51.858083  48.984550  42.363117
1653  6.687085  10.195733  35.514217  41.111717  38.441000  30.552333

           CE        CP         EPS1       VS1        SE
572   18.882850  1.471883  2367.347967  0.727367  59.950200
117   20.267500  1.531833  2405.333733  0.653417  60.262533
1161  27.786933  1.770233  2460.764700  0.612783  58.852000
789   26.399917  1.724267  2447.128100  0.597333  59.569083
1322  27.348517  1.772367  2464.307600  0.602217  59.857767
...         ...       ...          ...       ...        ...
1033  28.058167  1.751483  2460.162400  0.616033  58.991450
1731  47.345067  2.168317  2553.740300  0.542833  56.692333
763   26.283400  1.731600  2436.215500  0.595333  59.623317
835   26.282333  1.713750  2443.892400  0.603817  59.742533
1653  47.129133  2.166700  2542.090533  0.562700  58.897267
```

Figura 34-2: Datos de entrenamiento (X_test)

Realizado por: Mera H. 2022

```
datos de prueba 572     0
117     1
1161    0
789     1
1322    0
       ..
1033    0
1731    0
763     1
835     1
1653    1
Name: Stable, Length: 1764, dtype: int64
```

Figura 35-2: Datos de prueba (y_test)

Realizado por: Mera H. 2022

2.1.8.2. Estandarización o Escalado de datos

Para que exista un correcto análisis de datos, ya que no poseen el mismo rango de valores entre las variables X e Y, se necesitaría mucho calculo y eso requiere un alto número de tiempo, es decir para que éste inconveniente para que no afecte en el análisis de datos cuando se inserte en el modelo de inteligencia artificial, se realiza un escalado de variables tanto, los valores de "X" como para valores de "y" ya que se necesita que el rango entre valores leídos no sean demasiado amplios convirtiéndose en un rango particular normalizando todos los valores, utilizando la librería de sklearn, StandardScaler se los transforma de manera automática, preparando los datos para ser introducidos en la red neuronal, dicho proceso se lo observa a continuación en la figura 36-2.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Figura 36-2: Datos de prueba (y_test)

Realizado por: Mera H. 2022

El proceso de escalado se realiza para todas las condiciones de predicción sin excepción o para las condiciones que plantee el problema al cual se vaya aplicar una red neuronal para predecir de X variable.

2.1.9. Arquitectura de la Red Neuronal Artificial

```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

Figura 37-2: Librería Keras

Realizado por: Mera H. 2022

Se inicia importando la librería keras, como se visualiza en la figura 37-2, además se importa "dense" que es el tipo de red que se utiliza para la detección de fallos y la métrica "sequential" para darle una secuencia entre capa y capa a la red.

2.1.9.1. Empezar la Red Neuronal Artificial

```
red_neuronal = Sequential()
```

Figura 38-2: denominación de la red

Realizado por: Mera H. 2022

Antes de mencionar como estarán construidas las capas se denomina el nombre de la red, dicha red se la denomina como una variable independiente de clasificación, en este caso se llama red_neuronal, figura 38-utilizando la función "Sequential", demostrando la secuencia entre capa y capa ocultas.

2.1.9.2. Añadir la capa de entrada y la primera capa oculta

```
# capa de entrada
red_neuronal.add(Dense(units=64, kernel_initializer='uniform', activation = 'relu', input_dim=17))
# 1era capa oculta
red_neuronal.add(Dense(units=64, kernel_initializer='uniform', activation = 'relu'))
```

Figura 39-2: Capa de entrada y Primera oculta

Realizado por: Mera H. 2022

Como se observa en la figura 39-2, con la función "Dense" que forma una clase de tensorflow que implemanta Python en la librería Keras para agregar y conectar las capas de la red neuronal, agregando valores unitarios según las variables que dispongan, no existe ninguna regla para determinar el número exacto de cuantas variables de entrada deben ir. Se activa la red con la función "activation" por medio del rectificador estándar "relu", en "input_dim" se incluyen las entradas de todas las variables independientes que fueron analizadas que son optimas para el análisis de predicción.

2.1.9.3. Segunda capa oculta

```
#2da capa oculta
red_neuronal.add(Dense(units=32, kernel_initializer='uniform', activation = 'relu'))
```

Figura 40-2: Segunda Capa Oculta

Realizado por: Mera H. 2022

Aquí reiteradamente agregamos entradas y el activador de la red, será el mismo proceso con el cual se construyó la primera capa oculta que tendrá la misma función de manera secuencial.

2.1.9.4. Capa de salida binaria

```
#capa de salida
red_neuronal.add(Dense(units=1, kernel_initializer='uniform', activation = 'sigmoid'))
```

Figura 41-2: Capa de salida

Realizado por: Mera H. 2022

Se denomina capa de salida con 1 neurona porque necesitamos predecir de manera binaria si es 0 o 1, por esa razón se requiera una neurona de salida, se usa la función de activación "sigmoid" que no solo sirve para predecir sino también para proporcionar la probabilidad, en este caso de que sea un fallo en el sistema hidráulico. Con esto se concluye la construcción de la red neuronal artificial, figura 42-2.

```
# capa de entrada
red_neuronal.add(Dense(units=64, kernel_initializer='uniform', activation = 'relu', input_dim=17))
# 1era capa oculta
red_neuronal.add(Dense(units=64, kernel_initializer='uniform', activation = 'relu'))
#2da capa oculta
red_neuronal.add(Dense(units=32, kernel_initializer='uniform', activation = 'relu'))
#capa de salida
red_neuronal.add(Dense(units=1, kernel_initializer='uniform', activation = 'sigmoid'))
```

Figura 42-2: Red neuronal Artificial

Realizado por: Mera H. 2022

2.1.9.5. Capa de salida categórica

La capa de salida para las condiciones categóricas, cambia los parámetros ya que la base de datos ofrece varias condiciones con subcondiciones y se debe cambiar las métricas de la salida por categorías, es decir en "units" se escribe la cantidad de subcategorías o respuestas que nos ofrece la condición y en "activatio" se utiliza la función de activación "softmax" ya que sirve para realizar predicciones multicategóricas, figura 43-2.

```
#capa de salida
red_neuronal.add(Dense(units=4, kernel_initializer='uniform', activation = 'softmax'))
```

Figura 43-2: Capa de salida

Realizado por: Mera H. 2022

2.1.10. Parámetros de la Red Neuronal

En los parámetros hay que tener en cuenta todos los optimizadores que existen en la librería keras para realizar la compilación o aprendizaje de la red neuronal.

2.1.10.1. Compilación de la Red Neuronal Artificial

```
red_neuronal.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Figura 44-2: Copilar la ANN

Realizado por: Mera H. 2022

"adam" método de tensorflow, es el optimizador que se basa o realiza el descenso de gradiente estocástico, éste optimizador analiza y actualiza los pesos de las entradas en el entrenamiento y reduce su perdida, usando la función binary_cossentropy se analiza de manera binaria la predicción, figura 44-2. Para la evaluación del modelo se usa la métrica de precisión "accuracy".

```
#parametros de la capa
red_neuronal.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Figura 45-2: Copilar la ANN categórica

Realizado por: Mera H. 2022

En el análisis de categórico se reemplaza la función de binary_cossentropy por categorical_cossentropy, para que se analice varias respuestas en las capas de salidas, permace la métrica de la precisión, figura 45-2.

2.1.10.2. Revisión de general de la Neurona

Se imprime un reporte general para observar las métricas y construcción de la red

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                1152
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_2 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 7,425
Trainable params: 7,425
Non-trainable params: 0
_________________________________________________________________
None
```

Figura 46-2: Reporte de la configuración de la red

Realizado por: Mera H. 2022

2.1.10.3. Ajuste de la Red Neuronal a los datos de entrenamiento

Se denomina una historia con la función ".fit" la cual contiene todos los datos o repeticiones de entrenamiento que tendrá el modelo. Con batch_side compara la cantidad de lotes por segundo que realizara la red al momento de entrenamiento, el número de enteramiento o iteraciones que

debe realizarse, se la plantea con la función Epoch. El Epoch es el número de iteraciones en las cual realiza todo el proceso que se planteó en la construcción de la red que aplica métricas de análisis, según la figura 47-2.

```
history= red_neuronal.fit(X_train, y_train, batch_size=200, epochs=2000)
```

Figura 47-2: Ajuste de la red

Realizado por: Mera, H, 2022

2.1.10.4. Épocas de precisión

```
Epoch 1/4000
12/12 [==============================] - 0s 1ms/step - loss: 0.6901 - accuracy: 0.6485
```

Figura 48-2: Época inicial

Realizado por: Mera H. 2022

Al ejecutarse la historia inicia la ejecución de las iteraciones, figura 47-2, según las épocas que se plantee el aprendizaje inicia con un máximo error medido por "loss", ya que es el primer intento de aprendizaje de la red y mediante la métrica "accuracy" inicia con una baja precisión, figura 48-2,

```
Epoch 4000/4000
12/12 [==============================] - 0s 1ms/step - loss: 0.0159 - accuracy: 0.9943
```

Figura 49-2: Época final

Realizado por: Mera H. 2022

Se observa en la figura 49-2, como la precisión sube y los valores perdidos bajan es decisión que la red está realizando bien su aprendizaje.

CAPÍTULO III

3. RESULTADOS Y DISCUSIÓN DE RESULTADOS

La aplicación de Redes Neuronales Artificiales para la detección de fallas en Sistemas hidráulicos se divide en la obtención de los siguientes resultados por condiciones:

3.1. Matriz de confusión

Antes de comprobar los resultados con la matriz de confusión, se inicia prediciendo los resultados que se puedan existir, con la función de "y_pred", se declara la predicción que arroja la red neuronal en el test de la base de datos

```
y_pred = red_neuronal.predict(X_test)
y_pred = (y_pred>0.5)
```

Figura 1-3: Predecir resultados

Realizado por: Mera H. 2022

Una vez con ejecutado la función de predicción se procede a dibujar la matriz de confusión, con la librería sklearn.metrics se importa la función de matriz de confusión, a continuación se denomina con un nombre a la matriz y se la ejecuta con y_test e y_pred cuyas variables son las que se enlazan para la predicción de datos y se imprime la matriz

```
from sklearn.metrics import confusion_matrix, classification_report

mc = confusion_matrix(y_test, y_pred)
print(confusion_matrix(y_test, y_pred))
```

Figura 2-3: Código de matriz de confusión

Realizado por: Mera, H, 2022

Tabla 1-3: Matriz de confusión

		Positivo (1)	Negativo (0)
Valores predecidos	Positivo (1)	Verdaderos positivos	Falsos Positivos
	Negativo (0)	Falsos negativos	Verdaderos negativos

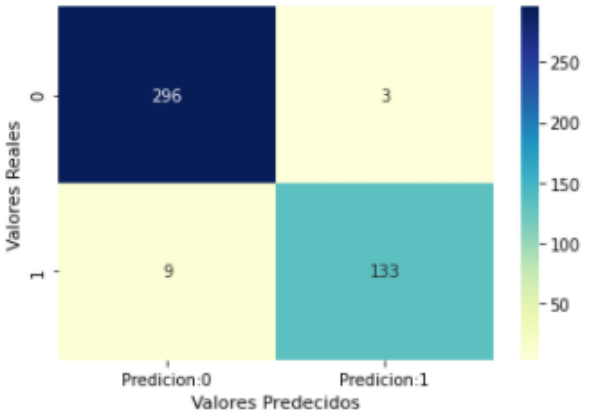
Fuente: (Babel, Kumar Singh y Kumar Jangir, 2019)

Realizado por: Mera H. 2022

A continuación, se demuestra cómo se lee la matriz de confusión en la tabla 1-3 para evaluar los resultados del Recall, f1 score, precisión y exactitud, dichas métricas son calculadas con las formulas nombradas en el capítulo I, la matriz de confusión dispone de valores predecidos que se convierten en positivos y negativos según la posición en la se encuentren se denominaran como en la tabla mencionada.

Una vez hecho el análisis de como leer la matriz de confusión se procede a observar las matrices de confusión y porcentajes de resultados por cada condición:

3.2. Estabilidad



	precision	recall	f1-score	support
0	0.97	0.99	0.98	299
1	0.98	0.94	0.96	142
accuracy			0.97	441
macro avg	0.97	0.96	0.97	441
weighted avg	0.97	0.97	0.97	441

Gráfico 1-3: Resultados de Estabilidad

Realizado por: Mera H. 2022

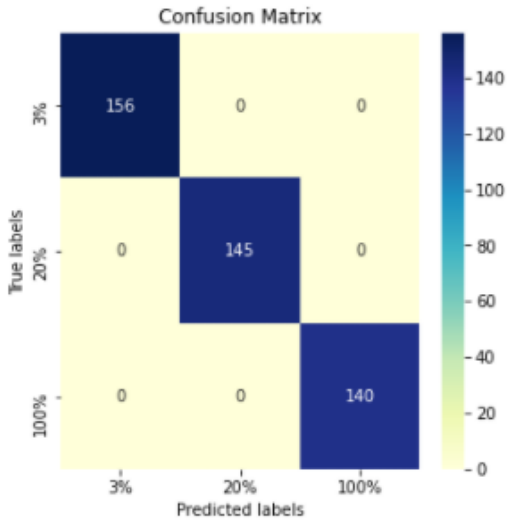
La estabilidad es una condición general del sistema hidráulico, que demuestra una predicción binaria de 0 cuando el sistema no posee ninguna falla en su sistema y 1 cuando su sistema como se muestra en el gráfico 1-3 la matriz de confusión con su respectivo reporte de clasificación que demuestran la efectividad del sistema, en la matriz de confusión de 2 por 2 se observa que demuestra el desenvolvimiento que tuvo la aplicación de la red neuronal para la predicción de fallos, la suma total de sus datos de testeo son 441, mostrados en la tabla de resultados, separados de la siguiente manera, 296 puntos en predicciones verdaderas cuando la condición de estabilidad esta en 0 y 133 puntos de predicciones preventivas cuando está en fallo o 1, se debe tomar en cuenta que cuando tenemos falsos positivos (posición superior derecha) basarse en la tabla 1-3, en éste caso poseemos 3, hay que tener en cuenta que son valores que alertan una posible falla y

se debería aplicar un mantenimiento predictivo al sistema, cuando se posee falsos negativos, en éste caso 9 se procede aplicar una reparación urgente ya que son fallos críticos que podrían dañar el sistema.

A continuación, con la matriz de confusión de la estabilidad se procede a calcular las métricas que miden la eficiencia de la red neuronal, las métricas nombradas a continuación se las calcula con las formulas mencionadas en el capítulo I, se inicia con la precisión en la condición 0 se obtiene un 97% y en la condición 1 un 98% aquí nos damos cuenta que el sistema si posee un equilibrio, se procede a calcular el recall, en 0 un 99% y en 1 un 94% aquí vemos que varía por un 6%, se puede decir que la red posee caídas en neuronas irrelevantes y elimina valores, siendo un poco la variación en el recall, utilizando las métricas de la precisión y el recall se procede a calcular el f1-score quien predominara para la factibilidad de detección de fallos, en 0 posee un 98 y en 1 un 96%, es decir que tiene un desenvolvimiento optimo la red neuronal para predecir fallos y por último se demuestra el accuracy o exactitud es del 97% de la red neuronal aplicado a esta condición para predicción de fallos de la estabilidad.

-

3.3. Condición de enfriamiento



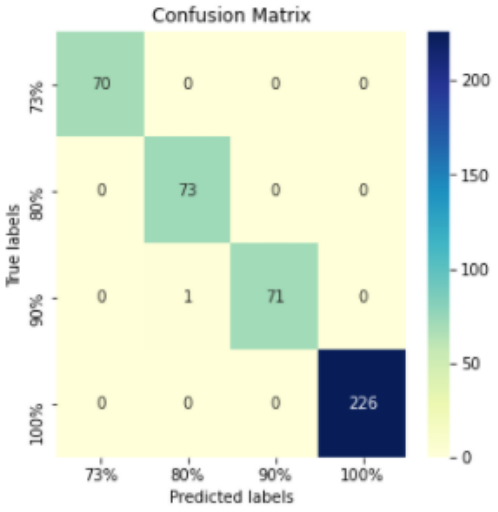
	precision	recall	f1-score	support
3	1.00	1.00	1.00	156
20	1.00	1.00	1.00	145
100	1.00	1.00	1.00	140
accuracy			1.00	441
macro avg	1.00	1.00	1.00	441
weighted avg	1.00	1.00	1.00	441

Gráfico 2-3: Resultados de Enfriamiento

Realizado por: Mera H. 2022

La segunda condición de análisis de resultados es la condición de enfriamiento se basa en una predicción categórica formando una matriz de confusión de tres por tres junto con su reporte de clasificación, que posee tres subcondiciones las cuales son: al 3% cerca del fallo total, al 20% eficiencia reducida y al 100% full eficiencia, como se observa en el gráfico 2-3, en éste caso la eficiencia de la predicción de la matriz es excelente sin existir valores falsos en sus intersecciones, es decir que para el 3% tenemos 156 valores para el 20% tenemos 145 valores y para el 100% tenemos 140 predecidos sumando 441 datos de testeo con una exactitud del 100%, se calcula la precisión y como se observa en la gráfico 2-3 la tabla de reporte de métricas, la precisión en las 3 condiciones es el 100% en el recall de igual manera el 100% y usando estas 2 métricas se calcula el f1-score dándose el 100%, la exactitud de la red o accuracy es del 100%, demostrando que la red neuronal tiene un desempeño excelente en la detección de fallos del enfriamiento del sistema.

3.4. Condición de la válvula



	precision	recall	f1-score	support
73	1.00	1.00	1.00	70
80	0.99	1.00	0.99	73
90	1.00	0.99	0.99	72
100	1.00	1.00	1.00	226
accuracy			1.00	441
macro avg	1.00	1.00	1.00	441
weighted avg	1.00	1.00	1.00	441

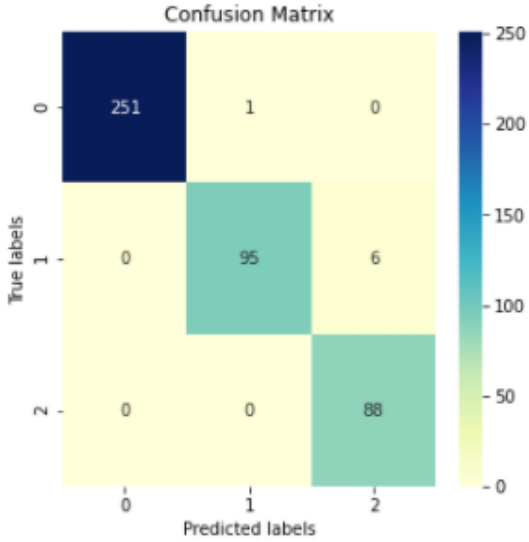
Gráfico 3-3: Resultados de la Válvula

Realizado por: Mera H. 2022

La tercera condición de análisis es la condición de la válvula poseyendo una matriz de predicción categórica de cuatro por cuatro junto con su reporte de clasificación ya que posee cuatro subcondiciones, gráfico 3-3, 73% cerca del fallo total, 80% severo retraso, 90% pequeño retraso y 100% comportamiento optimo, como se observa el gráfico 3-3, se mira un falso positivo en la

condición del 90% en conjunto con la 80% es decir que existe un valor falso positivo que se lo debe analizar en una sola condición para descartarlo en el análisis de la otra condición y se calcula, para la presicion observando el reporte de clasificación, solo la condición del 80% tiene un 99% y los demás 100%, para el recall solo el 99% posee al 90%, se calcula el f1-score y se plantea de la siguiente manera 100% para cuando la condición de la válvula está al 73% y 100% y 99% cuando esta al 80% y 90% y por último la exactitud o accuracy de la red es del 100%, demostrando que la red posee un desempeño alto para detectar fallos en la válvula.

3.5. Fuga interna de la bomba



	precision	recall	f1-score	support
0	1.00	1.00	1.00	252
1	0.99	0.94	0.96	101
2	0.94	1.00	0.97	88
accuracy			0.98	441
macro avg	0.98	0.98	0.98	441
weighted avg	0.98	0.98	0.98	441

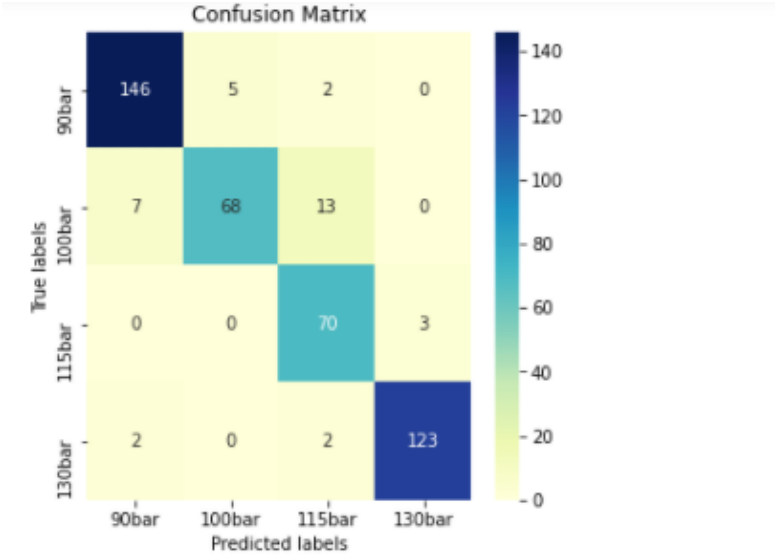
Gráfico 4-3: Resultados de bomba

Realizado por: Mera H. 2022

La cuarta condición de análisis es la fuga interna de la bomba poseyendo una matriz de confusión categórica de tres por tres junto con el reporte de clasificación ya que posee tres subcondiciones, gráfico 4-3, 0: ninguna fuga, 1: fuga de débil, 2: fuga severa, como se observa en el gráfico 4-3 la matriz posee varios puntos de predicción de falsos positivos, en especial para la subcondición 1 que posee 6 puntos a la derecha y 1 hacia arriba eso quiere decir que se asemejan a la subcondición 0, tomándolo como una alerta, en la condición 0 con 251 puntos, condición 1 con 95 puntos y 86 puntos de predicción reales o verdaderos sumando 441 datos usados para el testeo

del modelo inteligente. En la subcondición 0 posee un 100% en la precisión, recall y f1-score es decir que posee una predicción excelente, en cambio para la subcondición 1 y 2 los porcentajes varian ya que los falsos positivos en la matriz se asemejan entre valores de condición es decir que están equilibrados por condición, en la condición 1: 99% precisión, 94% recall y 96% f1-score; en la condición 2: 99% precisión, 94% recall y 96% f1-score y por último el accuracy de la red es del 100%, demostrando que la red posee un desempeño alto para detectar fallos en la bomba del sistema.

3.6. Acumulador hidráulico



	precision	recall	f1-score	support
90	0.94	0.95	0.95	153
100	0.93	0.77	0.84	88
115	0.80	0.96	0.88	73
130	0.98	0.97	0.97	127
accuracy			0.92	441
macro avg	0.91	0.91	0.91	441
weighted avg	0.93	0.92	0.92	441

Gráfico 5-3: Resultados del Acumulador hidráulico

Realizado por: Mera H. 2022

La quinta condición de análisis es la condición del acumulador hidráulico poseyendo una matriz de predicción categórica de cuatro por cuatro junto con su reporte de clasificación, ya que posee cuatro subcondiciones, gráfico 5-3, 90 bar cerca de la falla total, 100 bar presión severamente reducida, 115 bar presión ligeramente reducida y 130 bar presión optima, como se observa el gráfico 5-3, existen varios falsos positivos en cada condición en todas las condiciones: para 90 bar de presión existe 146 puntos de predicción verdaderos y 7 falsos positivos con una precisión

del 94%, un recall del 95% y un f1-score del 95% , para 100 bar de presión existe 68 puntos de predicción verdaderos y 20 falsos positivos con una precisión del 94%, un recall del 95% y un f1-score del 95%, para 115 bar de presión existe 70 puntos de predicción verdaderos y 3 falsos positivos con una precisión del 80%, un recall del 96% y un f1-score del 88%, para 130 bar de presión existe 123 puntos de predicción verdaderos y 4 falsos positivos con una precisión del 98%, un recall del 97% y un f1-score del 97%, sumando todos los valores verdaderos con un total de 407 de los 441 ingresados con una accuracy o exactitud de predicción del 92% demostrando que la red posee un desempeño bueno para detectar fallos en el acumulador hidráulico.

3.7. Tabla resumida de resultados

Tabla 2-3: Resultados finales de la Red Neuronal en detección de fallos

Condiciones	Accuracy (%)	F1-score (%)		Épocas de aprendizaje
Estabilidad	97	Estable: No estable:	98 96	4000
Condición de enfriamiento	100	Full eficiencia: Eficiencia reducida: Cerca del fallo total:	100 100 100	500
Condición de la válvula	100	Comportamiento optimo: Pequeño retraso: Severo retraso: Cerca total del fallo:	100 99 99 100	500
Fuga interna de la bomba	100	Ninguna fuga: Fuga débil: Fuga severa:	100 96 97	500
Acumulador hidráulico	88	Presión optima: Presión ligeramente reducida: Presión severamente reducida: Cerca de la falla total:	97 88 84 95	1500
Resultados finales en la predicción de fallas del sistema hidráulico	97			

Realizado por: Mera H. 2022

Según la tabla 2-3 la exactitud demuestra como fue el aprendizaje de la red neuronal en la aplicación del modelo para el entrenamiento de predicción de valores aleatorios, es decir por cada condición la Red muestra cómo se desenvuelve y el nivel de Épocas o Iteraciones de aprendizaje que necesitan para que la exactitud de la red sea aceptable y poder llegar a la predicción con un total del 97% en todo el sistema hidráulico.

El f1-score demuestra la aceptación y credibilidad que posee el modelo inteligente para la detección de fallos en un sistema hidráulico, como se observa en la tabla 2-3 posee un resumen general de los resultados obtenidos a través de la investigación, demostrando porcentajes elevados en la predicción de datos por cada subcondición, planteándose las siguientes conclusiones.

CONCLUSIONES

Se detectó fallos utilizando redes neuronales artificiales definas por el modelo basado en Deep Learning. Examinándose las mejores condiciones de la base de datos que brinda el Learning Repository de la Universidad de California, Irvine, utilizado para el mantenimiento predictivo en sistemas hidráulicos con la aplicación de la inteligencia artificial.

Se propuso el modelo de Redes Neuronales artificiales que demuestra potencial al momento de realizar la predicción deseada ya que su amplio desenvolvimiento en el Machine Learning y Deep Learning mejoran su valor de precisión de aprendizaje que utilizando métricas como “adam” y “sigmoid” de manera binaria y “softmax” de manera categórica, basándose en gradientes que realizan una rectificación y extracción automática de características por época de aprendizaje.

Se demostró que el Accuracy es la métrica final usada para medir la exactitud del modelo inteligente que se aplica en el análisis del estudio y el f1-Score determina de manera directa cada condición. Según la tabla 2-3 la predicción al detectar fallos en un sistema hidráulico es aceptable y se demuestra que la hipótesis es verdadera para la predicción de fallos usando el aprendizaje profundo.

Se comparó el método tradicional de Machine Learning, realizado en el estudio de Guo et al,2019, en la detección de fallos usando ANN, demuestra una media final de accuracy del 88.6% de exactitud y según la tabla 2-3 usando Deep Learning se obtiene un 97% de exactitud en el sistema, mejorando el estudio realizado en un 8,4% de efectividad y demostrando el desenvolvimiento de las Redes Neuronales Artificiales con aprendizaje profundo para la detección de fallos que se aplican al mantenimiento predictivo en sistemas hidráulicos.

En la investigación se demostró que los modelos inteligentes basados en redes neuronales artificiales como clasificadores tienen mucho desempeño a la hora de predecir condiciones específicas para demostrar fallos, mejorando la capacidad de productividad para la industria, utilizando métodos de inteligencia artificial en la aplicación del mantenimiento.

RECOMENDACIONES

Se recomienda antes de iniciar el análisis de los datos con el método inteligente que se quiera aplicar, obtener información adecuada, confiable acerca de Machine Learning y Deep Learning ya que todos los métodos de inteligencia artificial no pueden aplicarse a un aprendizaje profundo Deep Learning.

Analizar el concepto de aplicación y de utilización para cada red cuando se quiera aplicar un modelo inteligente con Redes Neuronales Artificiales, ya que existen varios tipos de redes neuronales basados en la misma fórmula matemática estadística para el aprendizaje, pero con mejor optimización.

Entender cómo funcionan las métricas de activación de las redes neuronales artificiales ya que existen varios tipos de iniciadores como lo es “softmax” y “sigmoid” junto con el optimizador Adam, variando su aplicación al momento de aplicar un modelo inteligente, ya que las Redes Neuronales Artificiales no solamente sirven como clasificador, sino que se aplican también para regresiones o predictores específicos.

Realizar varias pruebas de comparación con el número de épocas y neuronas que se aplica en la ANN ya que un exceso podría verse afectado existiendo perdidas grandes y significativas que ocasionarían un sobreajuste y desajuste de la red ocasionando una caída y bajando el porcentaje de predicción

Plantear como trabajo futuro un mismo análisis con la base de datos de monitoreo en sistemas hidráulicos del repositorio de la Universidad de California, Irvine. Utilizando varios tipos de Redes Neuronales como: Convolucionales, LSTM o redes recurrentes entre otras. Para la comparación y demostración de análisis, de que red es más efectiva en la detección de fallos en sistemas hidráulicos.

BIBLIOGRAFÍA

ANACONDA INC., *Your data science toolkit*. [blog]. [Consulta: 26 noviembre 2021]. Disponible en: **https://www.anaconda.com/products/individual.**

ALHASSAN, A.M. & ZAINON, W.M.N.W. "Brain tumor classification in magnetic resonance image using hard swish-based RELU activation function-convolutional neural network. Neural Computing and Applications 2021 33:15" [en línea], 2021, vol. 33, no. 15, pp. 9075-9087. [Consulta: 2 marzo 2022]. ISSN 1433-3058. Disponible en: **https://link.springer.com/article/10.1007/s00521-020-05671-3.**

BABEL, V., KUMAR SINGH, B. & KUMAR JANGIR, S., "Journal of Analysis and Computation (JAC) EVALUATION METHODS FOR MACHINE LEARNING." [en línea], 2019, [Consulta: 15 diciembre 2021]. Disponible en: **www.ijaconline.com.**

BASHIR, D., MONTAÑEZ, G.D., SEHRA, S., SEGURA, P.S. & LAUW, J., "An Information-Theoretic Perspective on Overfitting and Underfitting. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)"* [en línea], 2020, vol. 12576 LNAI, pp. 347-358. [Consulta: 21 diciembre 2021]. ISSN 16113349. Disponible en: **https://link.springer.com/chapter/10.1007/978-3-030-64984-5_27.**

CHEN, Z. & LIU, B., "Lifelong Machine Learning, Second Edition. *Synthesis Lectures on Artificial Intelligence and Machine Learning"* [en línea], 2018, vol. 12, no. 3, pp. 1-207. [Consulta: 29 noviembre 2021]. ISSN 1939-4608. Disponible en: **https://www.morganclaypool.com/doi/abs/10.2200/S00832ED1V01Y201802AIM037.**

CHICCO, D. & JURMAN, G., "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics"* [en línea], 2020, vol. 21, no. 1, pp. 1-13. [Consulta: 16 diciembre 2021]. ISSN 14712164. Disponible en: **https://link.springer.com/articles/10.1186/s12864-019-6413-7.**

DAI, J., TANG, J., HUANG, S. & WANG, Y., "Signal-Based Intelligent Hydraulic Fault Diagnosis Methods: Review and Prospects. *Chinese Journal of Mechanical Engineering (English Edition)"* [en línea], 2019, vol. 32, no. 1. [Consulta: 16 diciembre 2021]. ISSN 21928258. Disponible en: **https://doi.org/10.1186/s10033-019-0388-9.**

DRUMEA, P., DUMITRESCU, I.C., HRISTEA, A. & CHIRITA, C., "METHODS OF DIAGNOSING MALFUNCTIONS IN HYDRAULIC ACTUATIONS. *Proceedings of 2016 International Conference on Hydraulics and Pneumatics - HERVEX "* [en línea], 2016, pp. 212-217. [Consulta: 25 diciembre 2021]. Disponible en: **https://fluidas.ro/hervex/proceedings2016/pp._212-217.PDF**

DUA, D. & G., "UCI Machine Learning Repository. *Condition monitoring of hydraulic systems Data Set"* [en línea], 2017, [Consulta: 17 diciembre 2021]. Disponible en: **https://archive.ics.uci.edu/ml/datasets/Condition+monitoring+of+hydraulic+systems.**

FAWWAZ, D.Z. & CHUNG, S.H., "Real-Time and Robust Hydraulic System Fault Detection via Edge Computing. *Applied Sciences 2020, Vol. 10, Page 5933"* [en línea], 2020, vol. 10, no. 17, pp. 5933. [Consulta: 25 noviembre 2021]. ISSN 20763417. Disponible en: **https://www.mdpi.com/2076-3417/10/17/5933/htm.**

FOLLOW, W.K., "Overrtting vs. Underrtting: A Complete Example." [en línea], 2018, [Consulta: 21 diciembre 2021]. Disponible en: **https://towardsdatascience.com/overfitting-vs-underfitting-a-complete-example-d05dd7e19765.**

GUO, P., WU, J., XU, X., CHENG, Y. & WANG, Y., "Health condition monitoring of hydraulic system based on ensemble support vector machine. *2019 Prognostics and System Health Management Conference, PHM-Qingdao 2019"* [en línea], 2019, [Consulta: 25 noviembre 2021]. Disponible en: **https://ieeexplore.ieee.org/abstract/document/8942981.**

HAN, F. & LIU, H., "Statistical analysis of latent generalized correlation matrix estimation in transelliptical distribution. *Bernoulli : official journal of the Bernoulli Society for Mathematical Statistics and Probability"* [en línea], 2017, vol. 23, no. 1, pp. 23. [Consulta: 29 noviembre 2021]. ISSN 13507265. Disponible en: **/pmc/articles/PMC5360110/.**

HAO, X., ZHANG, G. & MA, S., "Deep Learning. *https://doi.org/10.1142/S1793351X16500045"* [en línea], 2016, vol. 10, no. 3, pp. 417-439. [Consulta: 28 febrero 2022]. ISSN 17937108. Disponible en: **https://www.worldscientific.com/doi/abs/10.1142/S1793351X16500045.**

INCE, T., KIRANYAZ, S., EREN, L., ASKAR, M. & GABBOUJ, M., "Real-Time Motor Fault Detection by 1-D Convolutional Neural Networks. *IEEE Transactions on Industrial Electronics"* [en línea], 2016, vol. 63, no. 11, pp. 7067-7075. [Consulta: 25 noviembre 2021]. ISSN 02780046. Disponible en: **https://ieeexplore.ieee.org/abstract/document/7501527.**

KADIYALA, A. & KUMAR, A., "Applications of Python to evaluate environmental data science problems. *Environmental Progress & Sustainable Energy"* [en línea], 2017, vol. 36, no. 6, pp. 1580-1586. [Consulta: 22 noviembre 2021]. ISSN 1944-7450. Disponible en: **https://onlinelibrary.wiley.com/doi/full/10.1002/ep.12786.**

KOZYREVA, L. V., KOZYREV, V. V. & CHUPYATOV, N.N., "Chemical Vapor Deposition of Wear-Resistant Iron-Nickel Coating onto Precision Parts of Hydraulic Systems. *Inorganic Materials: Applied Research 2018 9:5"* [en línea], 2018, vol. 9, no. 5, pp. 985-989. [Consulta: 25 noviembre 2021]. ISSN 2075-115X. Disponible en: **https://link.springer.com/article/10.1134/S2075113318050167.**

LIU, X., TIAN, Y., LEI, X., LIU, M., WEN, X., HUANG, H. & WANG, H., "Deep forest based intelligent fault diagnosis of hydraulic turbine. *Journal of Mechanical Science and Technology"*, [en línea] 2019, vol. 33, no. 5, pp. 2049-2058. [Consulta: 25 noviembre 2021]. ISSN 19763824. Disponible en: **https://link.springer.com/article/10.1007/s12206-019-0408-9**

MANASWI, N.K., "Understanding and Working with Keras. *Deep Learning with Applications Using Python"* [en línea], 2018, pp. 31-43. [Consulta: 28 febrero 2022]. Disponible en: **https://link.springer.com/chapter/10.1007/978-1-4842-3516-4_2.**

MLSYS PROCEEDINGS, "TensorFlow.js: Machine Learning For The Web and Beyond. *Machine Learning For The Web and Beyond"* [en línea], 2022, [Consulta: 28 febrero 2022]. Disponible en: **https://proceedings.mlsys.org/paper/2019/hash/1d7f7abc18fcb43975065399b0d1e48e-Abstract.html.**

OPITZ, J. & BURST, S., "Macro F1 and Macro F1. *arXiv"* [en línea], 2019, pp. arXiv:1911.03347. [Consulta: 16 diciembre 2021]. ISSN 2331-8422. Disponible en: **https://ui.adsabs.harvard.edu/abs/2019arXiv191103347O/abstract.**

PAN, S.P., LI, Z.F., HUANG, Y.J. & LIN, W.C., "FPGA realization of activation function for neural network. *Proceedings - 2018 7th International Symposium on Next-Generation Electronics, ISNE 2018*" [en línea], 2018, pp. 1-2. [Consulta: 2 marzo 2022]. Disponible en: **https://ieeexplore.ieee.org/abstract/document/8394695.**

PANERATI, J., SCHNELLMANN, M.A., PATIENCE, C., BELTRAME, G. & PATIENCE, G.S., "Experimental methods in chemical engineering: Artificial neural networks–ANNs. *The Canadian Journal of Chemical Engineering"* [en línea], 2019, [Consulta: 25 noviembre 2021]. Disponible en: **https://onlinelibrary.wiley.com/doi/full/10.1002/cjce.23507.**

PAVLENKO, I., TROJANOWSKA, J., IVANOV, V. & LIAPOSHCHENKO, O., "Parameter Identification of Hydro-Mechanical. *International Journal of Mechatronics and Applied Mechanics"* [en línea], 2018, no. 5, pp. 19-26. [Consulta: 25 noviembre 2021]. Disponible en: **https://www.semanticscholar.org/paper/PARAMETER-IDENTIFICATION-OF-HYDRO-MECHANICAL-USING-Pavlenko-Trojanowska/6cf24042617f4572dc7e7b8d4b461fb55c92ccba**

PRAKASH, J. & KANKAR, P.K., "Health prediction of hydraulic cooling circuit using deep neural network with ensemble feature ranking technique. *Measurement"* [en línea], 2020, vol. 151, pp. 107225. [Consulta: 30 agosto 2021]. ISSN 0263-2241. Disponible en: **https://www.sciencedirect.com/science/article/abs/pii/S0263224119310905.**

RIVAS, W., BERTHA, A. & OLIVO, M., "Redes neuronales artificiales aplicadas al reconocimiento de patrones." [en línea], 2017, [Consulta: 3 febrero 2022]. Disponible en: **www.utmachala.edu.ec.**

ROLON-MÉRETTE, D., ROSS, M., ROLON-MÉRETTE, T. & CHURCH, K., "Python for Research in Psychology Introduction to Anaconda and Python: Installation and setup." [en línea], 2020, vol. 16, no. 5. [Consulta: 22 noviembre 2021]. Disponible en: **https://www.spyder-ide.org/.**

TABLADA -GERMÁN, C.J. & TORRES, A., "Redes Neuronales Artificiales." [en línea], 2021, [Consulta: 16 diciembre 2021]. Disponible en: **https://revistas.unc.edu.ar/index.php/REM/article/view/10280.**

THE THEANO DEVELOPMENT TEAM., "Theano: A Python framework for fast computation of mathematical expressions." [en línea], 2016, [Consulta: 28 febrero 2022]. Disponible en: **http://arxiv.org/abs/1605.02688.**

WEI, Z., ARORA, A., PATEL, P. & JOHN, L., "Design space exploration for softmax implementations. *Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors"* [en línea], 2020, vol. 2020-July, pp. 45-52. [Consulta: 2 marzo 2022]. ISSN 10636862. Disponible en: **https://ieeexplore.ieee.org/abstract/document/9153236.**

WIATOWSKI, T. & BOLCSKEI, H., "A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction. *IEEE Transactions on Information Theory*" [en línea], 2018, vol. 64, no. 3, pp. 1845-1866. [Consulta: 25 noviembre 2021]. ISSN 00189448. Disponible en: **https://ieeexplore.ieee.org/abstract/document/8116648.**

WIED, D., "A nonparametric test for a constant correlation matrix. *"* [en línea], 2015, vol. 36, no. 10, pp. 1157-1172. [Consulta: 29 noviembre 2021]. ISSN 15324168. Disponible en: **https://www.tandfonline.com/doi/abs/10.1080/07474938.2014.998152.**

XU, J., ZHANG, Y. & MIAO, D., "Three-way confusion matrix for classification: A measure driven view. *Information Sciences"* [en línea], 2020, vol. 507, pp. 772-794. [Consulta: 16 diciembre 2021]. ISSN 0020-0255. Disponible en: **https://www.sciencedirect.com/science/article/abs/pii/S0020025519306024.**

YAO, Z., YU, Y. & YAO, J., "Artificial neural network–based internal leakage fault detection for hydraulic actuators: An experimental investigation. *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering"* [en línea], 2018, vol. 232, no. 4, pp. 369-382. [Consulta: 16 diciembre 2021]. ISSN 20413041. Disponible en: **https://journals.sagepub.com/doi/abs/10.1177/0959651816678502**

ZADKA, M., "Installing Python. *DevOps in Python"* [en línea], 2019, pp. 1-6. [Consulta: 25 noviembre 2021]. Disponible en: **https://link.springer.com/chapter/10.1007/978-1-4842-4433-3_1.**

ZENG, G., "On the confusion matrix in credit scoring and its analytical properties." [en línea], 2019, vol. 49, no. 9, pp. 2080-2093. [Consulta: 17 diciembre 2021]. ISSN 1532415X. Disponible en: **https://www.tandfonline.com/doi/abs/10.1080/03610926.2019.1568485.**

ZHANG, X.-D., "Machine Learning. *A Matrix Algebra Approach to Artificial Intelligence "* [en línea], 2020, pp. 223-440. [Consulta: 29 noviembre 2021]. Disponible en: **https://link.springer.com/chapter/10.1007/978-981-15-2770-8_6.**

ZINNIKUS, I., ANTAKLI, A., KAPAHNKE, P., KLUSCH, M., KRAUSS, C., NONNENGART, A. & SLUSALLEK, P., "Integrated semantic fault analysis and worker support for cyber-physical production systems. *Proceedings - 2017 IEEE 19th Conference on Business Informatics, CBI 2017* " [en línea], 2017, vol. 1, pp. 207-216. [Consulta: 25 noviembre 2021]. Disponible en: **https://ieeexplore.ieee.org/document/8010724.**

ANEXOS

ANEXO A: ANÁLISIS DE DATOS

El análisis de datos se realiza en todas las condiciones, porque se las realiza de manera individual para que no exista ninguna confusión

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import pandas_profiling
import seaborn as sns
import cufflinks as cf
cf.go_offline()
import plotly.express as px
from sklearn.preprocessing import StandardScaler
```

CREACION DE UN DATA SET

llamar a carpeta de base de datos

In [3]:
```
datos= 'C:/Users/Elpnk/Desktop/DATOS DE PROGRAMACION'
```

In [4]:
```
def get_files(datos, filename):
    return pd.read_csv(os.path.join(datos, filename), sep='\t', header=None)
```

Importar todos los datos de los sensores de presión

In [5]:
```
presion_1 = get_files(datos=datos, filename='PS1.txt')
presion_2 = get_files(datos=datos, filename='PS2.txt')
presion_3 = get_files(datos=datos, filename='PS3.txt')
presion_4 = get_files(datos=datos, filename='PS4.txt')
presion_5 = get_files(datos=datos, filename='PS5.txt')
presion_6 = get_files(datos=datos, filename='PS6.txt')
```

In [6]:
```
presion_1
```

Out[6]:

	0	1	2	3	4	5	6	7	8	9	...	5990	5991	5992	5993	5994
0	151.47	151.45	151.52	151.27	150.80	150.69	153.89	154.67	152.88	153.82	...	151.16	151.19	151.25	151.16	151.10
1	151.11	151.12	151.16	150.92	150.70	150.62	152.40	153.21	152.81	153.53	...	150.82	150.82	150.86	150.80	150.73
2	150.81	150.79	150.84	150.65	150.35	150.23	152.03	152.81	152.44	153.27	...	150.49	150.44	150.47	150.46	150.38
3	150.48	150.47	150.52	150.31	150.04	149.98	151.63	152.48	152.24	152.94	...	150.34	150.30	150.28	150.38	150.41
4	150.41	150.35	150.24	150.12	149.87	149.71	151.64	152.37	151.78	152.68	...	150.31	150.20	150.17	150.28	150.31
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2200	151.70	151.83	151.90	151.75	151.62	151.63	153.78	154.73	153.83	154.43	...	151.84	151.75	151.67	151.76	151.81
2201	151.90	151.83	151.81	151.77	151.65	151.63	153.97	154.84	153.67	154.37	...	151.81	151.81	151.84	151.83	151.79
2202	151.73	151.78	151.84	151.62	151.46	151.49	153.91	154.88	153.69	154.24	...	151.73	151.66	151.73	151.71	151.68
2203	151.77	151.77	151.75	151.56	151.47	151.57	154.09	154.80	153.45	154.21	...	151.75	151.71	151.68	151.71	151.73
2204	151.83	151.78	151.77	151.65	151.49	151.48	154.00	154.77	153.48	154.29	...	151.70	151.71	151.63	151.64	151.67

Importar archivos de flujo de volumen

In [7]:
```
Flujo_1 = get_files(datos=datos, filename='FS1.txt')
Flujo_2 = get_files(datos=datos, filename='FS2.txt')
```

In [8]:
```
Flujo_2
```

Out[8]:

	0	1	2	3	4	5	6	7	8	9	...	590	591	592	593	594
0	10.179	10.174	10.151	10.149	10.172	10.176	10.169	10.176	10.174	10.171	...	10.413	10.399	10.397	10.384	10.401
1	10.408	10.429	10.415	10.418	10.401	10.403	10.408	10.416	10.398	10.417	...	10.438	10.411	10.419	10.414	10.407
2	10.392	10.386	10.404	10.391	10.387	10.422	10.414	10.414	10.441	10.434	...	10.320	10.352	10.356	10.336	10.338
3	10.329	10.328	10.349	10.363	10.359	10.328	10.333	10.341	10.371	10.329	...	10.299	10.296	10.283	10.256	10.270
4	10.276	10.279	10.292	10.288	10.266	10.271	10.284	10.283	10.294	10.267	...	10.199	10.199	10.233	10.245	10.233
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2200	10.196	10.189	10.180	10.201	10.176	10.200	10.185	10.193	10.182	10.190	...	10.193	10.174	10.170	10.176	10.170
2201	10.182	10.194	10.189	10.181	10.186	10.167	10.166	10.198	10.171	10.166	...	10.172	10.182	10.191	10.185	10.176

In [9]:
```
temperatura_1 = get_files(datos=datos, filename='TS1.txt')
temperatura_2 = get_files(datos=datos, filename='TS2.txt')
temperatura_3 = get_files(datos=datos, filename='TS3.txt')
temperatura_4 = get_files(datos=datos, filename='TS4.txt')
```

In [10]:
```
temperatura_1
```

Out[10]:

	0	1	2	3	4	5	6	7	8	9	...	50	51	52	53	54
0	35.570	35.492	35.469	35.422	35.414	35.320	35.227	35.242	35.160	35.176	...	36.008	35.984	35.996	36.039	36.008
1	36.156	36.094	35.992	36.008	35.992	35.902	35.824	35.820	35.727	35.727	...	37.328	37.324	37.340	37.332	37.316
2	37.488	37.391	37.340	37.312	37.223	37.145	37.059	36.973	36.898	36.879	...	38.457	38.461	38.457	38.469	38.469
3	38.633	38.535	38.469	38.379	38.297	38.223	38.125	38.062	37.977	37.969	...	39.441	39.363	39.367	39.457	39.461
4	39.461	39.461	39.375	39.281	39.203	39.113	39.043	38.969	38.875	38.883	...	40.324	40.320	40.312	40.340	40.320
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2200	35.414	35.348	35.254	35.262	35.168	35.172	35.105	35.094	35.000	35.008	...	35.426	35.516	35.437	35.426	35.410

importar el resto de los archivos de datos: eficiencia de la bomba, vibraciones, eficiencia de enfriamiento, potencia de enfriamiento, factor de eficiencia

In [11]:
```
bomba = get_files(datos=datos, filename='EPS1.txt')
vibracion1 = get_files(datos=datos, filename='VS1.txt')
enfriamiento_CE = get_files(datos=datos, filename='CE.txt') # Sensor de eficiencia de enfriamiento
enfriamiento_CP = get_files(datos=datos, filename='CP.txt') # Sensor de potencia de refigeracion
factor_efi = get_files(datos=datos, filename='SE.txt')
```

In [12]:
```
bomba
```

Out[12]:

	0	1	2	3	4	5	6	7	8	9	...	5990	5991	5992	5993	5994
0	2411.6	2411.6	2411.6	2411.6	2411.6	2411.6	2411.6	2411.6	2411.6	2409.6	...	2409.6	2409.2	2409.6	2409.4	2409.6
1	2409.6	2409.6	2409.6	2409.6	2409.6	2409.6	2409.6	2409.6	2409.6	2409.6	...	2398.8	2398.2	2398.2	2398.0	2398.0
2	2397.8	2397.8	2397.8	2397.8	2397.8	2397.8	2397.8	2397.8	2397.8	2395.8	...	2383.8	2383.8	2383.8	2383.8	2383.8
3	2383.8	2383.8	2383.8	2383.8	2383.8	2383.8	2383.8	2383.8	2382.8	2382.8	...	2373.2	2372.8	2372.6	2372.4	2372.2
4	2372.0	2372.0	2372.0	2372.0	2372.0	2372.0	2372.0	2372.0	2372.0	2373.0	...	2370.0	2370.0	2369.8	2369.8	2369.8

Importar datos de etiqueta

In [13]:
```
perfil = get_files(datos=datos, filename='profile.txt')
```

In [14]:
```
perfil
```

Out[14]:

	0	1	2	3	4
0	3	100	0	130	1
1	3	100	0	130	1
2	3	100	0	130	1
3	3	100	0	130	1
4	3	100	0	130	1
...	...	...	...	...	...
2200	100	100	0	90	0
2201	100	100	0	90	0
2202	100	100	0	90	0

Condiciones para la predicción

In [15]:
```
#Condicion de enfriamiento / %:
    #3: cerca del fallo total (732 instancias)
    #20: eficiencia reducida (732 instancias)
    #100: full eficiencia (741 instancias)

#Condicion de la valvula / %:
    #100: comportamiento de conmutación óptimo (1125 instancias)
    #90: pequeño retraso (360 instancias)
    #80: severo retraso (360 instancias)
    #73: close to total failure (360 instancias)

#Fuga interna de la bomba:
    #0: ninguna fuga (1221 instancias)
    #1: fuga débil (492 instancias)
    #2: fuga severa (492 instancias)

#acumulador hidraulico / bar:
    #130: presión óptima (599 instancias)
    #115: presión ligeramente reducida (399 instancias)
    #100: presión severamente reducida (399 instancias)
    #90: cerca de la falla total(808 instancias)

#Estabilidad:
    #0: condiciones eran estables (1449 instancias)
    #1: Es posible que aún no se hayan alcanzado las condiciones estáticas. (756 instancias)
```

In [16]:
```
coolerCondition = pd.DataFrame(perfil.iloc[:, 0])
valveCondition = pd.DataFrame(perfil.iloc[:, 1])
pumpLeak = pd.DataFrame(perfil.iloc[:, 2])
hydraulicAcc = pd.DataFrame(perfil.iloc[:, 3])
stableFlag = pd.DataFrame(perfil.iloc[:, 4])
```

In [17]:
```
# promediar los datos del ciclo, como se nos ha entregado varios datos ciclos de entrada
def Mean(DataFrame):
    DataFrame=DataFrame.mean(axis=1)
    return DataFrame
```

In [18]:
```
#convertir todos los datos a su respectiva media

# datos de presión

presion1=Mean(presion_1)
presion2=Mean(presion_2)
presion3=Mean(presion_3)
presion4=Mean(presion_4)
presion5=Mean(presion_5)
presion6=Mean(presion_6)

# datos de flujo de volumen

Flujo1=Mean(Flujo_1)
Flujo2=Mean(Flujo_2)

# datos de temperatura
temperatura1=Mean(temperatura_1)
temperatura2=Mean(temperatura_2)
temperatura3=Mean(temperatura_3)
temperatura4=Mean(temperatura_4)

#datos de enfriamiento

enfriamientoCE=Mean(enfriamiento_CE)
enfriamientoCP=Mean(enfriamiento_CP)

# Otros datos

bomba_eficiencia=Mean(bomba)
vibracion_1=Mean(vibracion1)
factorefi=Mean(factor_efi)
```

In [19]:

```
# fusionar o concatener en un solo marco de datos (dataFrame)

Data=pd.DataFrame()

# datos de presión

Data['PS1']=presion1
Data['PS2']=presion2
Data['PS3']=presion3
Data['PS4']=presion4
Data['PS5']=presion5
Data['PS6']=presion6

# datos de flujo de volumen

Data['FS1']=Flujo1
Data['FV2']=Flujo2

# datos de temperatura

Data['TS1']=temperatura1
Data['TS2']=temperatura2
Data['TS3']=temperatura3
Data['TS4']=temperatura4

#datos de enfriamiento

Data['CE']=enfriamientoCE # Sensor de eficiencia de enfriamiento
Data['CP']=enfriamientoCP # Sensor de potencia de refigeracion

# Otros datos

Data['EPS1']=bomba_eficiencia
Data['VS1']=vibracion_1
Data['SE']=factorefi

# datos de Prediccion

Data['PL']=pumpLeak
Data['CD']=coolerCondition
```

In [20]:

```
Data
```

Out[20]:

	PS1	PS2	PS3	PS4	PS5	PS6	FS1	FV2	TS1	TS2	...	CE	CP	EPS1	V
0	160.673492	109.466914	1.991475	0.000000	9.842170	9.728097	6.709815	10.304592	35.621983	40.978767	...	39.601350	1.862750	2538.929167	0.5769
1	160.603320	109.354890	1.976234	0.000000	9.635142	9.529488	6.715315	10.403098	36.676967	41.532767	...	25.786433	1.255550	2531.498900	0.5658
2	160.347720	109.158845	1.972224	0.000000	9.530548	9.427949	6.718522	10.366250	37.880800	42.442450	...	22.218233	1.113217	2519.928000	0.5765
3	160.188088	109.064807	1.946575	0.000000	9.438827	9.337430	6.720565	10.302678	38.879050	43.403983	...	20.459817	1.062150	2511.541633	0.5692
4	160.000472	108.931434	1.922707	0.000000	9.358762	9.260636	6.690308	10.237750	39.803917	44.332750	...	19.787017	1.070467	2503.449500	0.5773
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2200	161.227572	109.779581	2.001438	10.202473	9.972037	9.850381	6.689930	10.184515	35.313783	40.874800	...	46.628517	2.160600	2543.911033	0.5508
2201	161.206070	109.787481	1.998781	10.197919	9.966184	9.844854	6.692182	10.177767	35.321600	40.868883	...	46.689817	2.151450	2543.411333	0.5474
2202	161.192120	109.756174	1.993436	10.196824	9.964329	9.842628	6.693277	10.176172	35.319183	40.875950	...	46.472300	2.143300	2542.729767	0.5452
2203	161.208917	109.793884	2.007077	10.198588	9.968232	9.846690	6.684128	10.178353	35.324767	40.876067	...	46.544967	2.148483	2544.046333	0.5370
2204	161.217128	109.792177	2.002690	10.203126	9.973638	9.851949	6.692302	10.183393	35.322233	40.859400	...	46.647933	2.157050	2543.818300	0.5465

2205 rows × 22 columns

In [21]:

```
Data.head()
```

Out[21]:

	PS1	PS2	PS3	PS4	PS5	PS6	FS1	FV2	TS1	TS2	...	CE	CP	EPS1	VS1	
0	160.673492	109.466914	1.991475	0.0	9.842170	9.728097	6.709815	10.304592	35.621983	40.978767	...	39.601350	1.862750	2538.929167	0.576950	59.1
1	160.603320	109.354890	1.976234	0.0	9.635142	9.529488	6.715315	10.403098	36.676967	41.532767	...	25.786433	1.255550	2531.498900	0.565850	59.3
2	160.347720	109.158845	1.972224	0.0	9.530548	9.427949	6.718522	10.366250	37.880800	42.442450	...	22.218233	1.113217	2519.928000	0.576533	59.5
3	160.188088	109.064807	1.946575	0.0	9.438827	9.337430	6.720565	10.302678	38.879050	43.403983	...	20.459817	1.062150	2511.541633	0.569267	59.7
4	160.000472	108.931434	1.922707	0.0	9.358762	9.260636	6.690308	10.237750	39.803917	44.332750	...	19.787017	1.070467	2503.449500	0.577367	59.4

5 rows × 22 columns

In [22]:
```
Data.describe(include="all")
```

Out[22]:

	PS1	PS2	PS3	PS4	PS5	PS6	FS1	FV2	TS1	TS2	...	CE	22
count	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	...	2205.000000	
mean	160.485315	109.379906	1.753227	2.600266	9.163320	9.079363	6.198549	9.649453	45.424567	50.365979	...	31.299077	
std	4.699425	4.986585	0.251902	4.279355	0.576296	0.549589	1.032883	0.449246	7.991933	7.396254	...	11.575330	
min	155.391547	104.406307	0.840252	0.000000	8.366800	8.321527	2.018572	8.857513	35.313783	40.859400	...	17.555983	
25%	158.100195	106.962382	1.729733	0.000000	8.547239	8.487167	6.391670	9.203397	36.237150	41.864183	...	20.084650	
50%	158.960895	107.730169	1.779631	0.000000	9.115781	9.031516	6.576673	9.692270	44.836650	49.780583	...	27.392533	
75%	161.000735	109.421612	1.932047	3.503266	9.844351	9.729275	6.857508	10.155008	54.104317	58.584467	...	46.677383	
max	180.922708	131.589089	2.023398	10.207068	9.978510	9.856591	6.722707	10.403098	57.899283	61.958467	...	47.903667	

8 rows × 22 columns

In [23]:
```
Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2205 entries, 0 to 2204
Data columns (total 22 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   PS1     2205 non-null   float64
 1   PS2     2205 non-null   float64
 2   PS3     2205 non-null   float64
 3   PS4     2205 non-null   float64
 4   PS5     2205 non-null   float64
 5   PS6     2205 non-null   float64
 6   FS1     2205 non-null   float64
 7   FV2     2205 non-null   float64
 8   TS1     2205 non-null   float64
 9   TS2     2205 non-null   float64
 10  TS3     2205 non-null   float64
 11  TS4     2205 non-null   float64
 12  CE      2205 non-null   float64
 13  CP      2205 non-null   float64
 14  EPS1    2205 non-null   float64
 15  VS1     2205 non-null   float64
 16  SE      2205 non-null   float64
 17  PL      2205 non-null   int64  
 18  CD      2205 non-null   int64  
 19  VC      2205 non-null   int64  
 20  HA      2205 non-null   int64  
 21  Stable  2205 non-null   int64  
dtypes: float64(17), int64(5)
memory usage: 379.1 KB
```

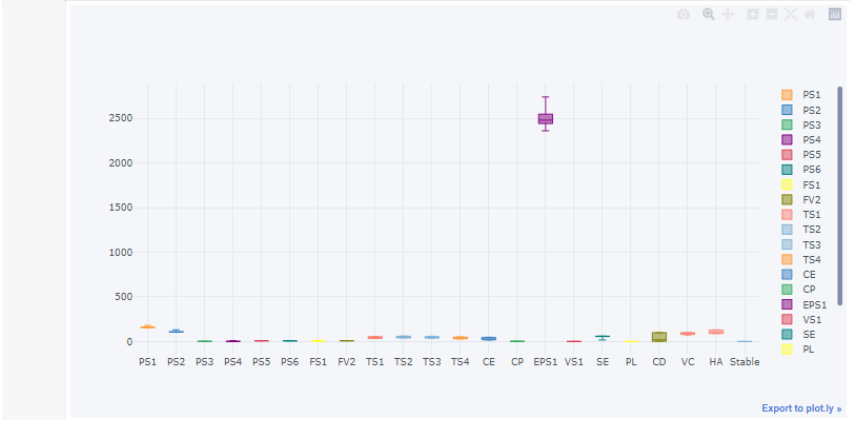
In [24]:
```
#comprobar el número de valores nulos.
Data.isnull().sum()
```

Out[24]:
```
PS1       0
PS2       0
PS3       0
PS4       0
PS5       0
PS6       0
FS1       0
FV2       0
TS1       0
TS2       0
TS3       0
TS4       0
CE        0
CP        0
EPS1      0
VS1       0
SE        0
PL        0
CD        0
VC        0
HA        0
Stable    0
dtype: int64
```

In [25]:
```
Data.shape
```

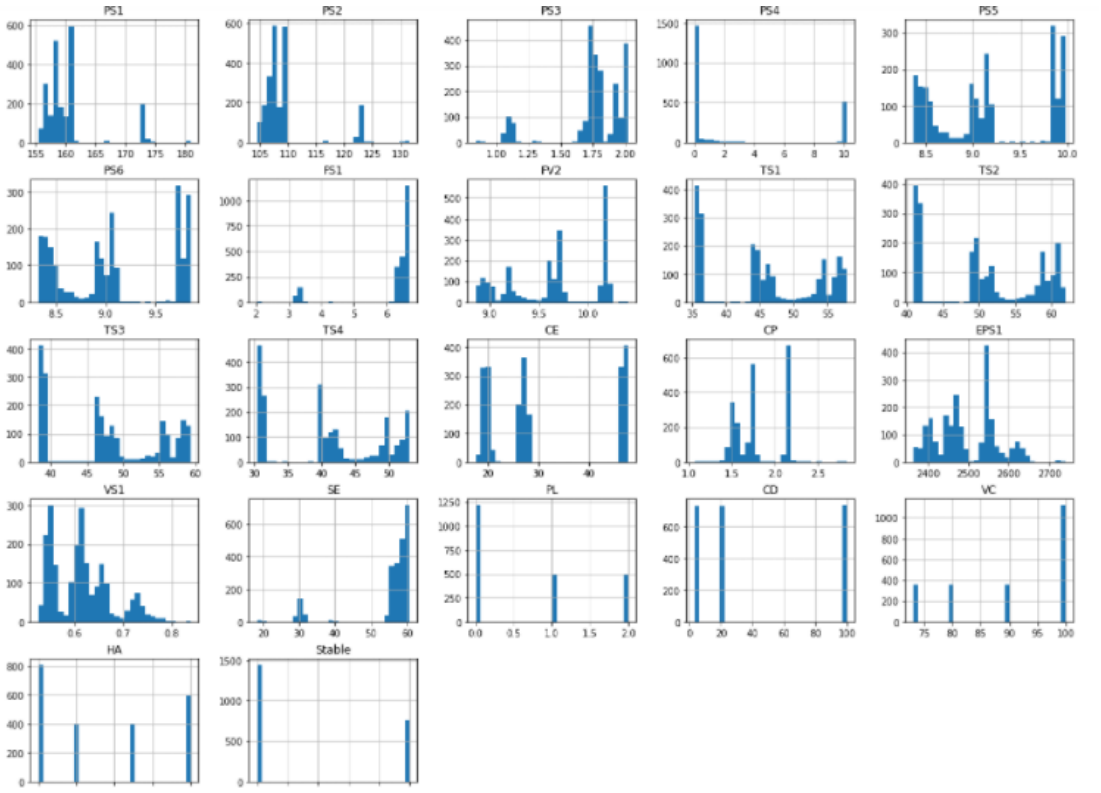
Out[25]:
```
(2205, 22)
```

In [26]:
```
1 Data.iplot(kind="box")
```



In [27]:
```
1 #histogramas de cada valor
2
3 Data.hist(bins=30, figsize=(20, 15))
```

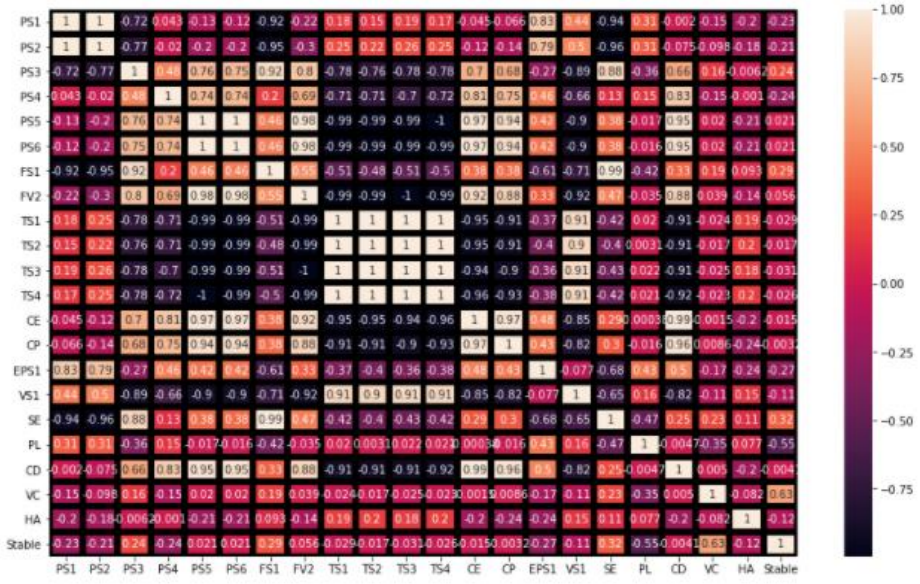
Out[27]:
```
array([[<AxesSubplot:title={'center':'PS1'}>,
        <AxesSubplot:title={'center':'PS2'}>,
        <AxesSubplot:title={'center':'PS3'}>,
        <AxesSubplot:title={'center':'PS4'}>,
        <AxesSubplot:title={'center':'PS5'}>],
       [<AxesSubplot:title={'center':'PS6'}>,
        <AxesSubplot:title={'center':'FS1'}>,
        <AxesSubplot:title={'center':'FV2'}>,
        <AxesSubplot:title={'center':'TS1'}>,
        <AxesSubplot:title={'center':'TS2'}>],
       [<AxesSubplot:title={'center':'TS3'}>,
        <AxesSubplot:title={'center':'TS4'}>,
        <AxesSubplot:title={'center':'CE'}>,
        <AxesSubplot:title={'center':'CP'}>,
        <AxesSubplot:title={'center':'EPS1'}>],
       [<AxesSubplot:title={'center':'VS1'}>,
        <AxesSubplot:title={'center':'SE'}>,
        <AxesSubplot:title={'center':'PL'}>,
        <AxesSubplot:title={'center':'CD'}>,
        <AxesSubplot:title={'center':'VC'}>],
       [<AxesSubplot:title={'center':'HA'}>,
        <AxesSubplot:title={'center':'Stable'}>, <AxesSubplot:>,
        <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



In [28]:

```
#Trazado de la correlación entre todos los conjuntos de datos de sensores de entrada

fig, ax= plt.subplots(figsize=(15,9))
sns.heatmap(Data.corr(), annot=True, linewidths=5, linecolor="black")
```

Out[28]: <AxesSubplot:>



solo algunos características están extremadamente correlacionadas. Necesitamos descartar todas las variables de destino para nuestro modelo de entrenamiento

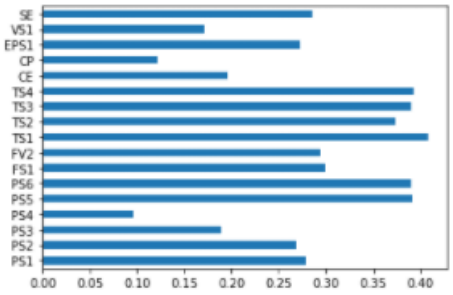
ANEXO B: PRIMERA CONDICIÓN, ESTABILIDAD

Selección de las mejores features para Stable flag

In [30]:
```
X=Data.drop(['Stable','PL','CD','VC','HA'],1)
y=Data['Stable']
```

In [31]:
```
#Selección de features utilizando la ganancia de información.
from sklearn.feature_selection import mutual_info_classif
importances=mutual_info_classif(X,y)
feat_imp=pd.Series(importances, X.columns[0:len(X.columns)])
print(feat_imp)
feat_imp.plot(kind='barh')
plt.show()
```

```
PS1     0.279421
PS2     0.269172
PS3     0.189913
PS4     0.096109
PS5     0.392179
PS6     0.390753
FS1     0.300041
FV2     0.294120
TS1     0.408156
TS2     0.374246
TS3     0.390152
TS4     0.392432
CE      0.196006
CP      0.121875
EPS1    0.271929
VS1     0.171108
SE      0.285755
dtype: float64
```



División datos de entrenamiento, validación y prueba.

In [32]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=0)
```

In [33]:
```
print(X_train)
print(y_test)
```

```
            PS1         PS2       PS3       PS4       PS5       PS6  \
572   156.259090  105.109720  1.686528  0.000000  8.409337  8.359900
117   157.029422  106.361600  1.729071  0.000000  8.592279  8.533665
1161  159.114915  107.267203  1.803048  0.000000  9.136608  9.051796
789   158.176940  107.214707  1.786352  0.000000  8.970626  8.898110
1322  158.989445  107.983810  1.806357  0.000158  9.146062  9.061882
...          ...         ...       ...       ...       ...       ...
1033  158.867803  107.103200  1.789223  0.000000  9.054415  8.976108
1731  160.771032  109.054252  1.935512  10.112827  9.880726  9.761596
763   157.909970  106.973936  1.779226  0.000000  8.924975  8.852112
835   158.262197  107.266625  1.803428  0.000000  8.985091  8.909845
1653  161.047945  109.579380  2.008050  0.785936  9.937681  9.819300

            FS1        FV2        TS1        TS2        TS3        TS4  \
572   6.572717   8.932982  57.330050  61.493300  58.674333  52.562750
117   6.601323   9.239225  53.610750  58.093367  55.153417  49.000933
1161  6.632630   9.690735  44.593617  49.634633  46.804117  40.021017
789   6.634915   9.604517  47.109183  52.262500  49.325900  42.666800
1322  6.672445   9.723228  44.389533  49.417183  46.637700  39.878583
...        ...        ...        ...        ...        ...        ...
1033  6.643458   9.662503  45.757833  50.665533  47.919200  41.195717
1731  6.500505  10.175213  35.897400  41.499200  38.796267  30.888767
763   6.626108   9.544212  47.881850  53.009617  50.051850  43.319533
835   6.636710   9.607130  46.754067  51.858083  48.984550  42.363117
1653  6.687085  10.195733  35.514217  41.111717  38.441000  30.552333

            CE        CP         EPS1       VS1        SE
572   18.882850  1.471883  2367.347967  0.727367  59.950200
117   20.267500  1.531833  2405.333733  0.653417  60.262533
1161  27.786933  1.770233  2460.764700  0.612783  58.852000
789   26.399917  1.724267  2447.128100  0.597333  59.569083
```

Estandarización de los datos

In [35]:
```
# valores escalares para x e y
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

la arquitectura de la red esta basado en tres capas 1 de entrada, 2 ocultas y una de salida

- para las capas de entrada y docultas utilizaremos nuestro activador RELU que actua como retificador de datos
- para el activador de la capa de salida tendremos al funcion sigmoid la salida de curva de aprendizaje basada en el tiempo mostrando como una progresion temporal en la salida de resultados
- Se utiliza un optimizador ADAM (SGD) es una extensión del descenso de gradiente estocástico es una extensión del descenso de gradiente estocástico Adam es un algoritmo de optimización que se puede utilizar en lugar del procedimiento de descenso de gradiente estocástico clásico para actualizar los pesos de red de forma iterativa en función de los datos de entrenamiento.
- La metrica para evaluación sera la precisión accuracy

In [36]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

In [37]:
```
red_neuronal = Sequential()
```

In [38]:
```
# capa de entrada
red_neuronal.add(Dense(units=128, kernel_initializer='uniform', activation = 'relu', input_dim=17))
# 1era capa oculta
red_neuronal.add(Dense(units=90, kernel_initializer='uniform', activation = 'relu'))
#2da capa oculta
red_neuronal.add(Dense(units=40, kernel_initializer='uniform', activation = 'relu'))
#capa de salida
red_neuronal.add(Dense(units=1, kernel_initializer='uniform', activation = 'sigmoid'))
#parametros de la capa
red_neuronal.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
print(red_neuronal.summary())
```

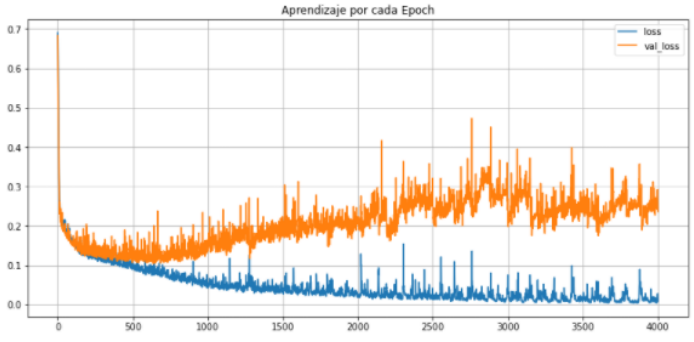
Ejecucion del apredizaje de la red neuronal

In [39]:
```
history= red_neuronal.fit(X_train, y_train, validation_data=(X_test, y_test),batch_size=150, epochs=4000)
```

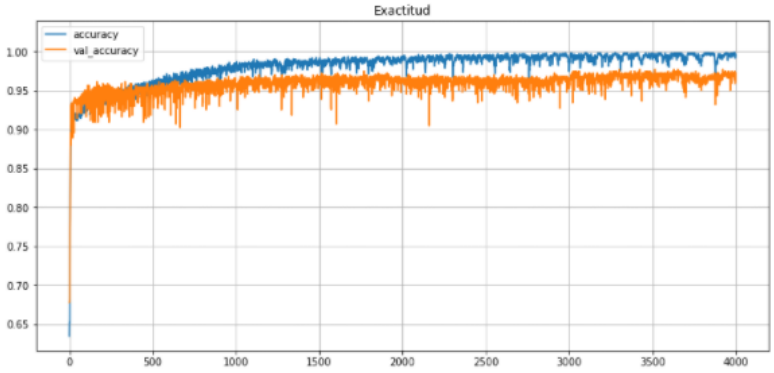
```
Epoch 1/4000
12/12 [==============================] - 0s 21ms/step - loss: 0.6904 - accuracy: 0.6344 - val_loss: 0.6832 - val_accuracy:
0.6780
Epoch 2/4000
12/12 [==============================] - 0s 3ms/step - loss: 0.6693 - accuracy: 0.6519 - val_loss: 0.6320 - val_accuracy: 0.
6780
Epoch 3/4000
12/12 [==============================] - 0s 4ms/step - loss: 0.6093 - accuracy: 0.6519 - val_loss: 0.5732 - val_accuracy: 0.
6780
Epoch 4/4000
12/12 [==============================] - 0s 4ms/step - loss: 0.5720 - accuracy: 0.6519 - val_loss: 0.5445 - val_accuracy: 0.
6848
Epoch 5/4000
12/12 [==============================] - 0s 4ms/step - loss: 0.5273 - accuracy: 0.6859 - val_loss: 0.4914 - val_accuracy: 0.
7687
Epoch 6/4000
12/12 [==============================] - 0s 4ms/step - loss: 0.4745 - accuracy: 0.8129 - val_loss: 0.4352 - val_accuracy: 0.
7868
Epoch 7/4000
```

visualizacion de desenvolviemiento de la red que no exista desajuste y sobreajuste

In [40]:
```
plt.figure(figsize=(13,6))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['loss', 'val_loss'])
plt.title("Aprendizaje por cada Epoch")
plt.grid()
plt.show()
```



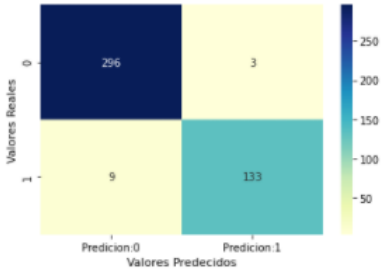
In [41]:
```
plt.figure(figsize=(13,6))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['accuracy', 'val_accuracy'])
plt.title("Exactitud")
plt.ylim([None, 1.04])
plt.grid()
plt.show()
```



parametros de predicción

In [42]:
```
y_pred = red_neuronal.predict(X_test)
y_pred = (y_pred>0.5)
```

In [43]:
```
from sklearn.metrics import confusion_matrix, classification_report

cm = confusion_matrix(y_test, y_pred)

cm_matrix = pd.DataFrame(data=cm, columns=['Predicion:0', 'Predicion:1'],  index=['0', '1'])

sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
#polynomial
plt.ylabel('Valores Reales', size = 11)
plt.xlabel('Valores Predecidos', size = 11)
plt.show()

print(classification_report(y_test, y_pred))
```




```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       299
           1       0.98      0.94      0.96       142

    accuracy                           0.97       441
   macro avg       0.97      0.96      0.97       441
weighted avg       0.97      0.97      0.97       441
```

ANEXO C: SEGUNDA CONDICIÓN, CONDICIÓN DE ENFRIAMIENTO

Seleccion del label

In [37]:
```
X2=Data.drop(columns=['Stable','PL','CD','HA','VC'])
y2=Data['CD']
```

In [38]:
```
X2
```

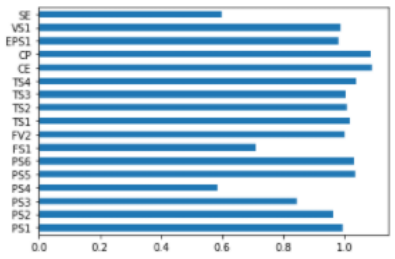
Out[38]:

	PS1	PS2	PS3	PS4	PS5	PS6	FS1	FV2	TS1	TS2	TS3	TS4	CE	CP
0	160.673492	109.466914	1.991475	0.000000	9.842170	9.728097	6.709815	10.304592	35.621983	40.978767	38.471017	31.745250	39.601350	1.862750
1	160.603320	109.354890	1.976234	0.000000	9.635142	9.529488	6.715315	10.403098	36.676967	41.532767	38.978967	34.493867	25.786433	1.255550
2	160.347720	109.158845	1.972224	0.000000	9.530548	9.427949	6.718522	10.366250	37.880800	42.442450	39.631950	35.646150	22.218233	1.113217
3	160.188088	109.064807	1.946575	0.000000	9.438827	9.337430	6.720565	10.302678	38.879050	43.403983	40.403383	36.579467	20.459817	1.062150
4	160.000472	108.931434	1.922707	0.000000	9.358762	9.260636	6.690308	10.237750	39.803917	44.332750	41.310550	37.427900	19.787017	1.070467
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2200	161.227572	109.779581	2.001438	10.202473	9.972037	9.850381	6.689930	10.184515	35.313783	40.874800	38.269267	30.404733	46.628517	2.160600
2201	161.206070	109.787481	1.998781	10.197919	9.966184	9.844854	6.692182	10.177767	35.321600	40.868883	38.268250	30.416233	46.689817	2.151450
2202	161.192120	109.756174	1.993436	10.196824	9.964329	9.842628	6.693277	10.176172	35.319183	40.875950	38.246367	30.426250	46.472300	2.143300
2203	161.208917	109.793884	2.007077	10.198588	9.968232	9.846690	6.684128	10.178353	35.324767	40.876067	38.245733	30.414283	46.544967	2.148483
2204	161.217128	109.792177	2.002690	10.203126	9.973638	9.851949	6.692302	10.183393	35.322233	40.859400	38.248917	30.390800	46.647933	2.157050

2205 rows × 17 columns

In [39]:
```
#Selección de features utilizando la ganancia de información.
from sklearn.feature_selection import mutual_info_classif
importances=mutual_info_classif(X2,y2)
feat_imp=pd.Series(importances, X2.columns[0:len(X2.columns)])
print(feat_imp)
feat_imp.plot(kind='barh')
plt.show()
```

```
PS1     0.992026
PS2     0.962200
PS3     0.843691
PS4     0.582346
PS5     1.032893
PS6     1.032207
FS1     0.709647
FV2     0.999847
TS1     1.017553
TS2     1.005907
TS3     1.003467
TS4     1.037901
CE      1.088549
CP      1.083736
EPS1    0.979578
VS1     0.986878
SE      0.597110
dtype: float64
```



In [40]:
```
### features mejor correlacionados
feat_imp[feat_imp>0.4].index
```

Out[40]:
```
Index(['PS1', 'PS2', 'PS3', 'PS4', 'PS5', 'PS6', 'FS1', 'FV2', 'TS1', 'TS2',
       'TS3', 'TS4', 'CE', 'CP', 'EPS1', 'VS1', 'SE'],
      dtype='object')
```

In [41]:
```
# X2 = Data.drop(columns=['Stable','PL','CD','HA','VC']) ### todas las columnas
X2 = Data[['PS1', 'PS2', 'PS3', 'PS4', 'PS5', 'PS6', 'FS1', 'FV2', 'TS1', 'TS2', 'TS3', 'TS4', 'CE', 'CP', 'EPS1', 'VS1', 'S
X2
```

Out[41]:

	PS1	PS2	PS3	PS4	PS5	PS6	FS1	FV2	TS1	TS2	TS3	TS4	CE	CP
0	160.673492	109.466914	1.991475	0.000000	9.842170	9.728097	6.709815	10.304592	35.621983	40.978767	38.471017	31.745250	39.601350	1.862750
1	160.603320	109.354890	1.976234	0.000000	9.635142	9.529488	6.715315	10.403098	36.676967	41.532767	38.978967	34.493867	25.786433	1.255550
2	160.347720	109.158845	1.972224	0.000000	9.530548	9.427949	6.718522	10.366250	37.880800	42.442450	39.631950	35.646150	22.218233	1.113217
3	160.188088	109.064807	1.946575	0.000000	9.438827	9.337430	6.720565	10.302678	38.879050	43.403983	40.403383	36.579467	20.459817	1.062150
4	160.000472	108.931434	1.922707	0.000000	9.358762	9.260636	6.690308	10.237750	39.803917	44.332750	41.310550	37.427900	19.787017	1.070467

Transformar a binario y ordenar la condicion

In [43]:
```
y02 = pd.get_dummies(y2.values.ravel())
# print(y02)
clases = list(y02.columns)
print(clases)
y02
```

```
[3, 20, 100]
```

Out[43]:

	3	20	100
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
...	...	...	...
2200	0	0	1
2201	0	0	1
2202	0	0	1
2203	0	0	1
2204	0	0	1

2205 rows × 3 columns

Estandarización o escalado de los datos

In [49]:
```
# valores escalares para x
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X_train.shape, X_test.shape
```

Out[49]: ((1764, 17), (441, 17))

Arquitectura de la red neuronal

In [50]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

In [51]:
```
red_neuronal2 = Sequential()
# capa de entrada
red_neuronal2.add(Dense(units=64, kernel_initializer='uniform', activation = 'relu', input_dim=17))
# 1era capa oculta
red_neuronal2.add(Dense(units=64, kernel_initializer='uniform', activation = 'relu'))
#2da capa oculta
red_neuronal2.add(Dense(units=32, kernel_initializer='uniform', activation = 'relu'))
#capa de salida
red_neuronal2.add(Dense(units=3, kernel_initializer='uniform', activation = 'softmax'))
#parametros de la capa
red_neuronal2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
red_neuronal2.summary()
```

División datos de entrenamiento, validación y prueba.

In [45]:
```
y2.unique()
```

Out[45]: array([3, 20, 100], dtype=int64)

In [46]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X2, y02, test_size=0.2,random_state=0)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[46]: ((1764, 17), (441, 17), (1764, 3), (441, 3))

In [47]:
```
# print(X_train)
# print(y_test)
```

In [48]:
```
print(X_train.shape)
print(y_train.shape)
```

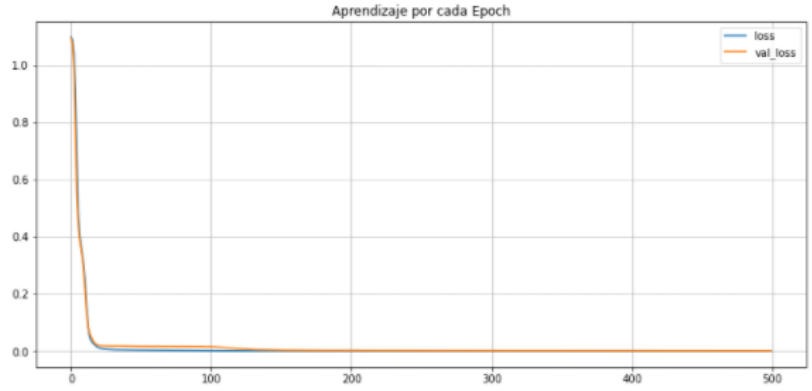
```
(1764, 17)
(1764, 3)
```

In [52]:
```
history = red_neuronal2.fit(X_train, y_train,
                            validation_data=(X_test, y_test),
                            batch_size=200,
                            epochs=500)
```

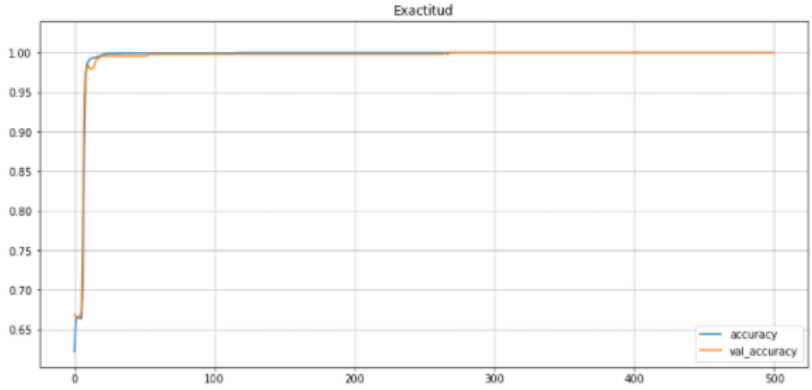
```
cy: 1.0000
Epoch 495/500
9/9 [==============================] - 0s 5ms/step - loss: 3.1706e-06 - accuracy: 1.0000 - val_loss: 8.0246e-04 - val_accura
cy: 1.0000
Epoch 496/500
9/9 [==============================] - 0s 7ms/step - loss: 3.1432e-06 - accuracy: 1.0000 - val_loss: 8.0422e-04 - val_accura
cy: 1.0000
Epoch 497/500
```

visualizacion de desenvolviemiento de la red que no exista desajuste y sobreajuste

In [53]:
```
plt.figure(figsize=(13,6))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['loss', 'val_loss'])
plt.title("Aprendizaje por cada Epoch")
plt.grid()
plt.show()
```



In [54]:
```
plt.figure(figsize=(13,6))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['accuracy', 'val_accuracy'])
plt.title("Exactitud")
plt.ylim([None, 1.04])
plt.grid()
plt.show()
```



Parametros de predicción

In [55]:
```
y_pred = red_neuronal2.predict(np.array([X_test[5]]))
print(y_pred.astype(np.float64))
```

```
[[9.16248677e-09 5.54484408e-21 1.00000000e+00]]
```

In [56]:
```
Y_test = [y_test.columns[i] for i in y_test.to_numpy().argmax(axis=1)]
# Y_test
```

In [57]:
```
y_pred=red_neuronal2.predict(X_test)
y_pred=np.argmax(y_pred, axis=1)
# Y_test=np.argmax(y_test, axis=1)
# y_pred#.shape#, Y_test.shape
```

In [58]:
```
Y_pred = [y_test.columns[i] for i in y_pred]
# Y_pred
```

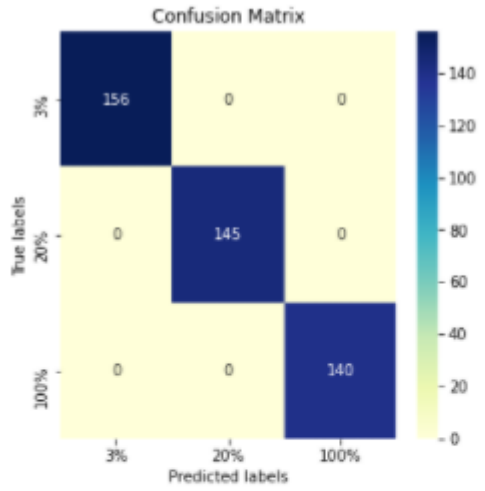
In [59]:
```
from sklearn.metrics import confusion_matrix, classification_report

mc = confusion_matrix(Y_test, Y_pred)
print(confusion_matrix(Y_test, Y_pred))
```

```
[[156   0   0]
 [  0 145   0]
 [  0   0 140]]
```

Resultados de predicción

In [61]:
```
clases2 = [str(i)+'%' for i in clases]
```

In [63]:
```
fig, ax = plt.subplots(figsize=(5,5))
sns.heatmap(mc,
            xticklabels=clases2,
            yticklabels=clases2,
            linecolor='red',
            fmt='.0f',
            annot=True,
            cmap='YlGnBu',
            ax=ax)
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
# plt.ylabel('y preds')
plt.show()

print(classification_report(Y_test, Y_pred))
```



	precision	recall	f1-score	support
3	1.00	1.00	1.00	156
20	1.00	1.00	1.00	145
100	1.00	1.00	1.00	140
accuracy			1.00	441
macro avg	1.00	1.00	1.00	441
weighted avg	1.00	1.00	1.00	441

ANEXO D: TERCERA CONDICIÓN, CONDICIÓN DE LA VÁLVULA

Selección de las mejores features para Valve condition

In [32]:
```
X2=Data.drop(columns=['Stable','PL','CD','HA','VC'])
y2=Data['VC']
```

In [33]:
```
X2
```

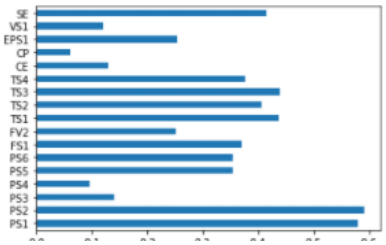
Out[33]:

	PS1	PS2	PS3	PS4	PS5	PS6	FS1	FV2	TS1	TS2	TS3	TS4	CE	CP
0	160.673492	109.466914	1.991475	0.000000	9.842170	9.728097	6.709815	10.304592	35.621983	40.978767	38.471017	31.745250	39.601350	1.862750
1	160.603320	109.354890	1.976234	0.000000	9.635142	9.529488	6.715315	10.403098	36.676967	41.532767	38.978967	34.493867	25.786433	1.255550
2	160.347720	109.158845	1.972224	0.000000	9.530548	9.427949	6.718522	10.366250	37.880800	42.442450	39.631950	35.646150	22.218233	1.113217
3	160.188088	109.064807	1.946575	0.000000	9.438827	9.337430	6.720565	10.302678	38.879050	43.403983	40.403383	36.579467	20.459817	1.062150
4	160.000472	108.931434	1.922707	0.000000	9.358762	9.260636	6.690308	10.237750	39.803917	44.332750	41.310550	37.427900	19.787017	1.070467
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2200	161.227572	109.779581	2.001438	10.202473	9.972037	9.850361	6.689930	10.184515	35.313783	40.874800	38.269267	30.404733	46.628517	2.160600
2201	161.206070	109.787481	1.998781	10.197919	9.966184	9.844854	6.692182	10.177767	35.321600	40.868883	38.268250	30.416233	46.689817	2.151450
2202	161.192120	109.756174	1.993436	10.196824	9.964329	9.842628	6.693277	10.176172	35.319183	40.875950	38.246367	30.426250	46.472300	2.143300
2203	161.208917	109.793884	2.007077	10.198588	9.968232	9.846690	6.684128	10.178353	35.324767	40.876067	38.245733	30.414283	46.544967	2.148483
2204	161.217128	109.792177	2.002690	10.203126	9.973638	9.851949	6.692302	10.183393	35.322233	40.859400	38.248917	30.390800	46.647933	2.157050

2205 rows × 17 columns

In [34]:
```
#Selección de features utilizando la ganancia de información.
from sklearn.feature_selection import mutual_info_classif
importances=mutual_info_classif(X2,y2)
feat_imp=pd.Series(importances, X2.columns[0:len(X2.columns)])
print(feat_imp)
feat_imp.plot(kind='barh')
plt.show()
```

```
PS1     0.580035
PS2     0.591373
PS3     0.139891
PS4     0.096276
PS5     0.353833
PS6     0.354115
FS1     0.371456
FV2     0.252868
TS1     0.436830
TS2     0.405953
TS3     0.439500
TS4     0.376898
CE      0.129019
CP      0.061524
EPS1    0.254181
VS1     0.119862
SE      0.415515
dtype: float64
```



In [35]:
```
### features mejor correlacionados
feat_imp[feat_imp>0.4].index
```

Out[35]:
```
Index(['PS1', 'PS2', 'TS1', 'TS2', 'TS3', 'SE'], dtype='object')
```

In [36]:
```
# X2 = Data.drop(columns=['Stable','PL','CD','HA','VC']) ### todas las columnas
X2 = Data[['PS1', 'PS2', 'TS1', 'TS2', 'TS3', 'SE']] ### columnas de mayor relacion
X2
```

Out[36]:

	PS1	PS2	TS1	TS2	TS3	SE
0	160.673492	109.466914	35.621983	40.978767	38.471017	59.157183
1	160.603320	109.354890	36.676967	41.532767	38.978967	59.335617
2	160.347720	109.158845	37.880800	42.442450	39.631950	59.543150
3	160.188088	109.064807	38.879050	43.403983	40.403383	59.794900
4	160.000472	108.931434	39.803917	44.332750	41.310550	59.455267
...	...	...	...	...	...	...
2200	161.227572	109.779581	35.313783	40.874800	38.269267	59.033100

In [38]:
```
y02 = pd.get_dummies(y2.values.ravel())
# print(y02)
clases = list(y02.columns)
y02
```

Out[38]:

	73	80	90	100
0	0	0	0	1
1	0	0	0	1
2	0	0	0	1
3	0	0	0	1
4	0	0	0	1
...	...	...	...	...
2200	0	0	0	1
2201	0	0	0	1
2202	0	0	0	1
2203	0	0	0	1
2204	0	0	0	1

2205 rows × 4 columns

División datos de entrenamiento, validación y prueba.

In [40]:
```
y2.unique()
```

Out[40]: array([100, 73, 80, 90], dtype=int64)

In [41]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X2, y02, test_size=0.2,random_state=0)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[41]: ((1764, 6), (441, 6), (1764, 4), (441, 4))

In [42]:
```
# print(X_train)
# print(y_test)
```

In [43]:
```
print(X_train.shape)
print(y_train.shape)
```

```
(1764, 6)
(1764, 4)
```

Estandarización o escalado de los datos

In [44]:
```
# valores escalares para x
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X_train.shape, X_test.shape
```

Out[44]: ((1764, 6), (441, 6))

Arquitectura de la red neuronal

In [45]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

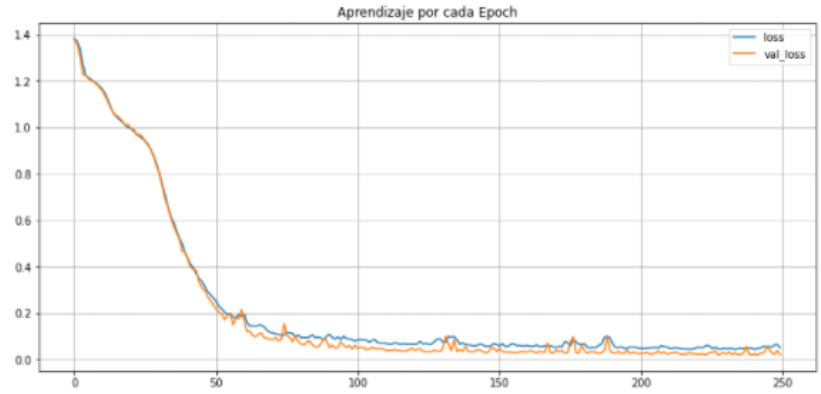
In [46]:
```
red_neuronal2 = Sequential()
# capa de entrada
red_neuronal2.add(Dense(units=64, kernel_initializer='uniform', activation = 'relu', input_dim=6)) ### n de columnas
# 1era capa oculta
red_neuronal2.add(Dense(units=64, kernel_initializer='uniform', activation = 'relu'))
#2da capa oculta
red_neuronal2.add(Dense(units=32, kernel_initializer='uniform', activation = 'relu'))
#capa de salida
red_neuronal2.add(Dense(units=4, kernel_initializer='uniform', activation = 'softmax')) ### n de labels
#parametros de la capa
red_neuronal2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
red_neuronal2.summary()
```

In [47]:
```
history = red_neuronal2.fit(X_train, y_train,
                            validation_data=(X_test, y_test), ### set de validacion | ayuda a gestionar el overfitting
                            batch_size=200,
                            epochs=250)
```

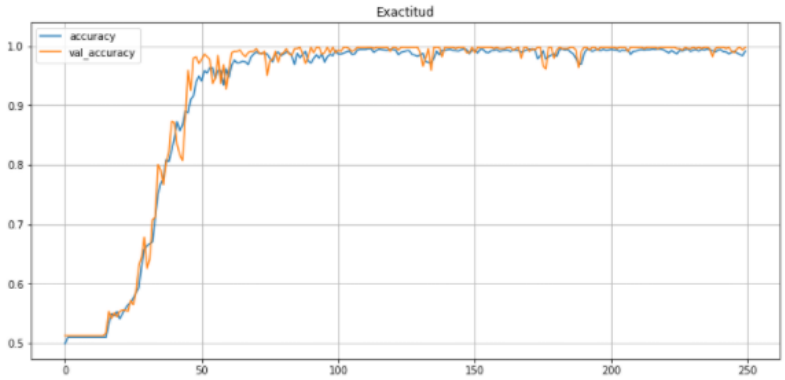
```
Epoch 1/250
9/9 [==============================] - 1s 56ms/step - loss: 1.3833 - accuracy: 0.4989 - val_loss: 1.3780 - val_accuracy: 0.5
125
Epoch 2/250
9/9 [==============================] - 0s 3ms/step - loss: 1.3711 - accuracy: 0.5096 - val_loss: 1.3584 - val_accuracy: 0.51
25
Epoch 3/250
9/9 [==============================] - 0s 3ms/step - loss: 1.3403 - accuracy: 0.5096 - val_loss: 1.3091 - val_accuracy: 0.51
25
Epoch 4/250
9/9 [==============================] - 0s 3ms/step - loss: 1.2734 - accuracy: 0.5096 - val_loss: 1.2289 - val_accuracy: 0.51
25
```

visualizacion de desenvolviemiento de la red que no exista desajuste y sobreajuste

In [48]:
```
plt.figure(figsize=(13,6))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['loss', 'val_loss'])
plt.title("Aprendizaje por cada Epoch")
plt.grid()
plt.show()
```



In [49]:
```
plt.figure(figsize=(13,6))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['accuracy', 'val_accuracy'])
plt.title("Exactitud")
plt.ylim([None, 1.04])
plt.grid()
plt.show()
```



Parametros de predicción

In [50]:
```
y_pred = red_neuronal2.predict(np.array([X_test[5]]))
print(y_pred.astype(np.float64))
```

```
[[9.99674320e-01 3.25535395e-04 1.04440382e-07 1.09825982e-20]]
```

In [51]:
```
y_pred#.to_numpy().argmax(axis=1)
```

Out[51]:
```
array([[9.9967432e-01, 3.2553539e-04, 1.0444038e-07, 1.0982598e-20]],
      dtype=float32)
```

In [52]:
```
### lista de los labels originales
Y_test = [y_test.columns[i] for i in y_test.to_numpy().argmax(axis=1)]
Y_test
```

```
100,
100,
100,
73,
73,
73,
90,
```

In [53]:
```
red_neuronal2.predict(X_test)
```

Out[53]:
```
array([[1.14877196e-31, 4.28435485e-16, 4.60263254e-06, 9.99995351e-01],
       [6.55311006e-26, 1.08781731e-14, 2.36705137e-06, 9.99997616e-01],
       [5.98752828e-21, 1.11530195e-10, 5.17320528e-04, 9.99482632e-01],
       ...,
       [1.04639013e-19, 1.27506172e-09, 2.27183662e-03, 9.97728169e-01],
       [4.45122894e-17, 8.41208675e-05, 9.97673929e-01, 2.24194536e-03],
       [2.62219560e-27, 1.04574544e-14, 5.48662592e-06, 9.99994516e-01]],
      dtype=float32)
```

In [54]:
```
### set de predicciones
y_pred=red_neuronal2.predict(X_test) ### arreglo bidimensional de predicciones
y_pred=np.argmax(y_pred, axis=1) ### arreglo unidimensional de indices
# Y_test=np.argmax(y_test, axis=1)
# y_pred#.shape#, Y_test.shape

Y_pred = [y_test.columns[i] for i in y_pred] ### se obtienen los labels como resultado de los indices
Y_pred
```

```
100,
80,
100,
80,
100,
100,
80,
100,
100,
90,
73,
100,
100,
100,
100,
100,
73,
73,
100,
100,
```

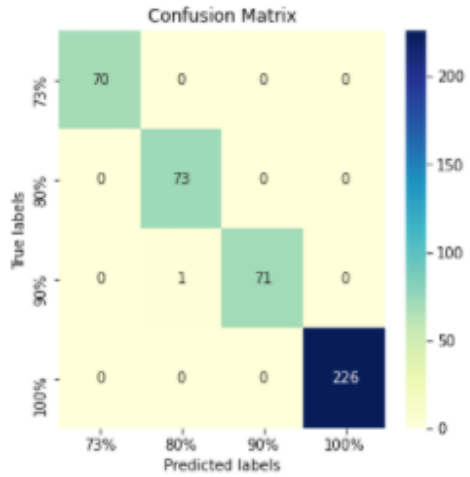
In [55]:
```
from sklearn.metrics import confusion_matrix, classification_report

mc = confusion_matrix(Y_test, Y_pred)
print(confusion_matrix(Y_test, Y_pred))
```

```
[[ 70   0   0   0]
 [  0  73   0   0]
 [  0   1  71   0]
 [  0   0   0 226]]
```

Resultados de predicción

In [56]:
```
clases2 = [str(i)+'%' for i in clases]
```

In [57]:
```
fig, ax = plt.subplots(figsize=(5,5))
sns.heatmap(mc,
            xticklabels=clases2,
            yticklabels=clases2,
            linecolor='red',
            fmt='.0f',
            annot=True,
            cmap='YlGnBu',
            ax=ax)
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
# plt.ylabel('y preds')
plt.show()

print(classification_report(Y_test, Y_pred))
```



	precision	recall	f1-score	support
73	1.00	1.00	1.00	70
80	0.99	1.00	0.99	73
90	1.00	0.99	0.99	72
100	1.00	1.00	1.00	226
accuracy			1.00	441
macro avg	1.00	1.00	1.00	441
weighted avg	1.00	1.00	1.00	441

ANEXO E: CUARTA CONDICIÓN, FUGA INTERNA DE LA BOMBA

Selección de las mejores features para PumbLeak

Seleccion del label

In [36]:
```
X2=Data.drop(columns=['Stable','PL','CD','HA','VC'])
y2=Data['PL']
```

In [37]:
```
X2
```

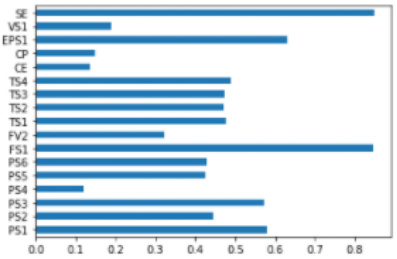
Out[37]:

	PS1	PS2	PS3	PS4	PS5	PS6	FS1	FV2	TS1	TS2	TS3	TS4	CE	CP	
0	160.673492	109.466914	1.991475	0.000000	9.842170	9.728097	6.709815	10.304592	35.621983	40.978767	38.471017	31.745250	39.601350	1.862750	
1	160.603320	109.354890	1.976234	0.000000	9.635142	9.529488	6.715315	10.403098	36.676967	41.532767	38.978967	34.493867	25.786433	1.255550	
2	160.347720	109.158845	1.972224	0.000000	9.530548	9.427949	6.718522	10.386250	37.880800	42.442450	39.631950	35.646150	22.218233	1.113217	
3	160.188088	109.064807	1.946575	0.000000	9.438827	9.337430	6.720565	10.302678	38.879050	43.403983	40.403383	36.579467	20.459817	1.062150	
4	160.000472	108.931434	1.922707	0.000000	9.358762	9.260636	6.690308	10.237750	39.803917	44.332750	41.310550	37.427900	19.787017	1.070467	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2200	161.227572	109.779581	2.001438	10.202473	9.972037	9.850361	6.689930	10.184515	35.313783	40.874800	38.269267	30.404733	46.628517	2.160600	
2201	161.206070	109.787481	1.998781	10.197919	9.966184	9.844854	6.692182	10.177767	35.321600	40.868883	38.268250	30.416233	46.689817	2.151450	
2202	161.192120	109.756174	1.993436	10.196824	9.964329	9.842628	6.693277	10.176172	35.319183	40.875950	38.246367	30.426250	46.472300	2.143300	
2203	161.208917	109.793884	2.007077	10.198588	9.968232	9.846690	6.684128	10.178353	35.324767	40.876067	38.245733	30.414283	46.544967	2.148483	
2204	161.217128	109.792177	2.002690	10.203126	9.973638	9.851949	6.692302	10.183393	35.322233	40.859400	38.248917	30.390800	46.647933	2.157050	

2205 rows × 17 columns

In [38]:
```
#Selección de features utilizando la ganancia de información.
from sklearn.feature_selection import mutual_info_classif
importances=mutual_info_classif(X2,y2)
feat_imp=pd.Series(importances, X2.columns[0:len(X2.columns)])
print(feat_imp)
feat_imp.plot(kind='barh')
plt.show()
```

```
PS1     0.578333
PS2     0.443895
PS3     0.570897
PS4     0.119627
PS5     0.426131
PS6     0.427733
FS1     0.845172
FV2     0.320885
TS1     0.475143
TS2     0.470948
TS3     0.472445
TS4     0.489502
CE      0.136499
CP      0.149726
EPS1    0.630895
VS1     0.188786
SE      0.848033
dtype: float64
```



In [39]:
```
### features mejor correlacionados
feat_imp[feat_imp>0.4].index
```

Out[39]:
```
Index(['PS1', 'PS2', 'PS3', 'PS5', 'PS6', 'FS1', 'TS1', 'TS2', 'TS3', 'TS4',
       'EPS1', 'SE'],
      dtype='object')
```

In [40]:
```
# X2 = Data.drop(columns=['Stable','PL','CD','HA','VC']) ### todas las columnas
X2 = Data[['PS1', 'PS2', 'PS3', 'PS5', 'PS6', 'FS1', 'TS1', 'TS2', 'TS3', 'TS4', 'EPS1', 'SE']] ### columnas de mayor relaci
X2
```

Out[40]:

	PS1	PS2	PS3	PS5	PS6	FS1	TS1	TS2	TS3	TS4	EPS1	SE
0	160.673492	109.466914	1.991475	9.842170	9.728097	6.709815	35.621983	40.978767	38.471017	31.745250	2538.929167	59.157183
1	160.603320	109.354890	1.976234	9.635142	9.529488	6.715315	36.676967	41.532767	38.978967	34.493867	2531.498900	59.335617
2	160.347720	109.158845	1.972224	9.530548	9.427949	6.718522	37.880800	42.442450	39.631950	35.646150	2519.928000	59.543150
3	160.188088	109.064807	1.946575	9.438827	9.337430	6.720565	38.879050	43.403983	40.403383	36.579467	2511.541633	59.794900
4	160.000472	108.931434	1.922707	9.358762	9.260636	6.690308	39.803917	44.332750	41.310550	37.427900	2503.449500	59.455267

Transformar a binario y ordenar la condicion

In [42]:
```
y02 = pd.get_dummies(y2.values.ravel())
# print(y02)
clases = list(y02.columns)
print(clases)
y02
```

```
[0, 1, 2]
```

Out[42]:

	0	1	2
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
...	...	...	...
2200	1	0	0
2201	1	0	0
2202	1	0	0
2203	1	0	0
2204	1	0	0

2205 rows × 3 columns

División datos de entrenamiento, validación y prueba.

In [44]:
```
y2.unique()
```

Out[44]: array([0, 2, 1], dtype=int64)

In [45]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X2, y02, test_size=0.2,random_state=0)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[45]: ((1764, 12), (441, 12), (1764, 3), (441, 3))

In [46]:
```
# print(X_train)
# print(y_test)
```

In [47]:
```
print(X_train.shape)
print(y_train.shape)
```

```
(1764, 12)
(1764, 3)
```

Estandarización o escalado de los datos

In [48]:
```
# valores escalares para x
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X_train.shape, X_test.shape
```

Out[48]: ((1764, 12), (441, 12))

Estandarización o escalado de los datos

In [48]:
```
# valores escalares para x
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X_train.shape, X_test.shape
```

Out[48]: ((1764, 12), (441, 12))

Arquitectura de la red neuronal

In [49]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

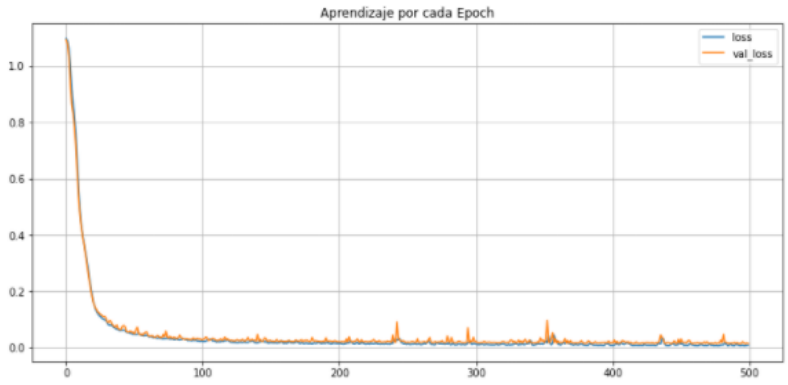
In [50]:
```
red_neuronal2 = Sequential()
# capa de entrada
red_neuronal2.add(Dense(units=64, kernel_initializer='uniform', activation = 'relu', input_dim=12))
# 1era capa oculta
red_neuronal2.add(Dense(units=64, kernel_initializer='uniform', activation = 'relu'))
#2da capa oculta
red_neuronal2.add(Dense(units=32, kernel_initializer='uniform', activation = 'relu'))
#capa de salida
red_neuronal2.add(Dense(units=3, kernel_initializer='uniform', activation = 'softmax'))
#parametros de la capa
red_neuronal2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
red_neuronal2.summary()
```

In [51]:
```
history = red_neuronal2.fit(X_train, y_train,
                            validation_data=(X_test, y_test),
                            batch_size=200,
                            epochs=500)
```

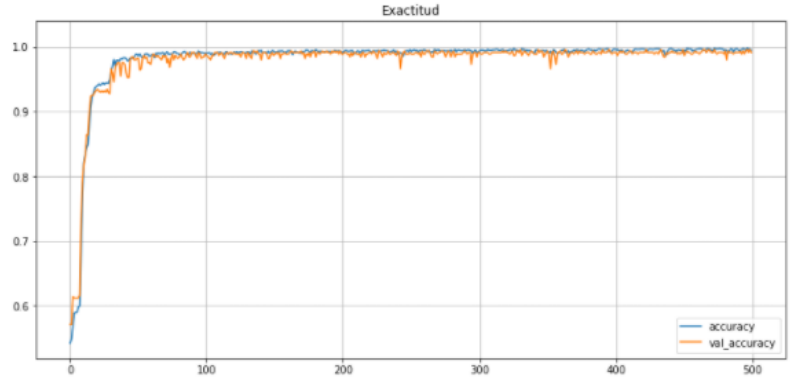
```
32
Epoch 495/500
9/9 [==============================] - 0s 3ms/step - loss: 0.0087 - accuracy: 0.9960 - val_loss: 0.0148 - val_accuracy: 0.98
87
Epoch 496/500
9/9 [==============================] - 0s 3ms/step - loss: 0.0093 - accuracy: 0.9949 - val_loss: 0.0212 - val_accuracy: 0.99
32
Epoch 497/500
9/9 [==============================] - 0s 3ms/step - loss: 0.0071 - accuracy: 0.9977 - val_loss: 0.0167 - val_accuracy: 0.99
55
Epoch 498/500
9/9 [==============================] - 0s 3ms/step - loss: 0.0082 - accuracy: 0.9966 - val_loss: 0.0144 - val_accuracy: 0.99
09
Epoch 499/500
9/9 [==============================] - 0s 3ms/step - loss: 0.0089 - accuracy: 0.9960 - val_loss: 0.0166 - val_accuracy: 0.99
55
Epoch 500/500
9/9 [==============================] - 0s 3ms/step - loss: 0.0094 - accuracy: 0.9949 - val_loss: 0.0152 - val_accuracy: 0.99
09
```

visualizacion de desenvolviemiento de la red que no exista desajuste y sobreajuste

In [52]:
```
plt.figure(figsize=(13,6))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['loss', 'val_loss'])
plt.title("Aprendizaje por cada Epoch")
plt.grid()
plt.show()
```



In [53]:
```
plt.figure(figsize=(13,6))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['accuracy', 'val_accuracy'])
plt.title("Exactitud")
plt.ylim([None, 1.04])
plt.grid()
plt.show()
```



Parametros de predicción

In [54]:
```
y_pred = red_neuronal2.predict(np.array([X_test[5]]))
print(y_pred.astype(np.float64))
```

```
[[9.99999762e-01 2.01448429e-07 0.00000000e+00]]
```

In [55]:
```
Y_test = [y_test.columns[i] for i in y_test.to_numpy().argmax(axis=1)]
# Y_test
```

In [56]:
```
y_pred=red_neuronal2.predict(X_test)
y_pred=np.argmax(y_pred, axis=1)
# Y_test=np.argmax(y_test, axis=1)
# y_pred#.shape#, Y_test.shape
```

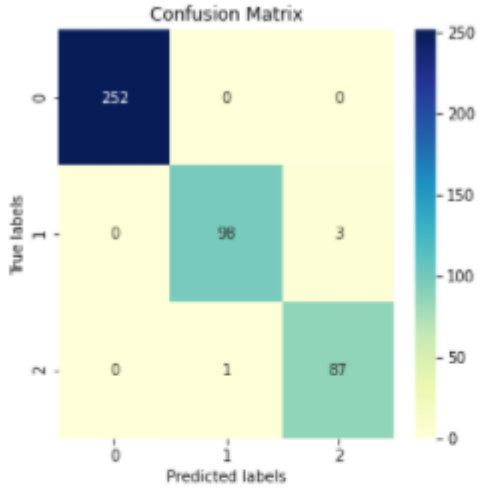
In [57]:
```
Y_pred = [y_test.columns[i] for i in y_pred]
# Y_pred
```

In [58]:
```
from sklearn.metrics import confusion_matrix, classification_report

mc = confusion_matrix(Y_test, Y_pred)
print(confusion_matrix(Y_test, Y_pred))
```

```
[[252   0   0]
 [  0  98   3]
 [  0   1  87]]
```

Resultados de predicción

In [59]:
```
fig, ax = plt.subplots(figsize=(5,5))
sns.heatmap(mc,
            xticklabels=clases,
            yticklabels=clases,
            linecolor='red',
            fmt='.0f',
            annot=True,
            cmap='YlGnBu',
            ax=ax)
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
# plt.ylabel('y preds')
plt.show()

print(classification_report(Y_test, Y_pred))
```




```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       252
           1       0.99      0.97      0.98       101
           2       0.97      0.99      0.98        88

    accuracy                           0.99       441
   macro avg       0.99      0.99      0.99       441
weighted avg       0.99      0.99      0.99       441
```

ANEXO F: QUINTA CONDICIÓN, ACUMULADOR HIDRÁULICO

Selección de las mejores features para Acumulador Hidraulico

Seleccion del label

In [37]:
```
X2=Data.drop(columns=['Stable','PL','CD','HA','VC'])
y2=Data['HA']
```

In [38]:
```
X2
```

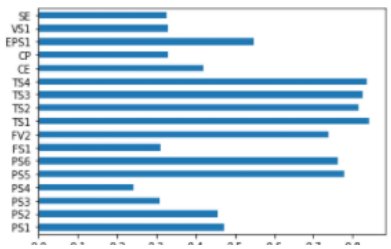
Out[38]:

	PS1	PS2	PS3	PS4	PS5	PS6	FS1	FV2	TS1	TS2	TS3	TS4	CE	CP	
0	160.673492	109.466914	1.991475	0.000000	9.842170	9.728097	6.709815	10.304592	35.621983	40.978767	38.471017	31.745250	39.601350	1.862750	
1	160.603320	109.354890	1.976234	0.000000	9.635142	9.529488	6.715315	10.403098	36.676967	41.532767	38.978967	34.493867	25.786433	1.255550	
2	160.347720	109.158845	1.972224	0.000000	9.530548	9.427949	6.718522	10.366250	37.880800	42.442450	39.631950	35.646150	22.218233	1.113217	
3	160.188088	109.064807	1.946575	0.000000	9.438827	9.337430	6.720565	10.302678	38.879050	43.403983	40.403383	36.579467	20.459817	1.062150	
4	160.000472	108.931434	1.922707	0.000000	9.358762	9.260636	6.690308	10.237750	39.803917	44.332750	41.310550	37.427900	19.787017	1.070467	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2200	161.227572	109.779581	2.001438	10.202473	9.972037	9.850361	6.689930	10.184515	35.313783	40.874800	38.269267	30.404733	46.628517	2.160600	
2201	161.206070	109.787481	1.998781	10.197919	9.966184	9.844854	6.692182	10.177767	35.321600	40.868883	38.268250	30.418233	46.689817	2.151450	
2202	161.192120	109.756174	1.993436	10.196824	9.964329	9.842628	6.693277	10.176172	35.319183	40.875950	38.246367	30.426250	46.472300	2.143300	
2203	161.208917	109.793884	2.007077	10.198588	9.968232	9.846690	6.684128	10.178353	35.324767	40.876067	38.245733	30.414283	46.544967	2.148483	
2204	161.217128	109.792177	2.002690	10.203126	9.973638	9.851949	6.692302	10.183393	35.322233	40.859400	38.248917	30.390800	46.647933	2.157050	

2205 rows × 17 columns

In [39]:
```
#Selección de features utilizando la ganancia de información.
from sklearn.feature_selection import mutual_info_classif
importances=mutual_info_classif(X2,y2)
feat_imp=pd.Series(importances, X2.columns[0:len(X2.columns)])
print(feat_imp)
feat_imp.plot(kind='barh')
plt.show()
```

```
PS1     0.472893
PS2     0.455874
PS3     0.307743
PS4     0.242838
PS5     0.779196
PS6     0.762187
FS1     0.311139
FV2     0.739168
TS1     0.841094
TS2     0.814852
TS3     0.824732
TS4     0.834793
CE      0.418250
CP      0.328641
EPS1    0.549436
VS1     0.329377
SE      0.326198
dtype: float64
```



In [40]:
```
### features mejor correlacionados
feat_imp[feat_imp>0.4].index
```

Out[40]:
```
Index(['PS1', 'PS2', 'PS5', 'PS6', 'FV2', 'TS1', 'TS2', 'TS3', 'TS4', 'CE',
       'EPS1'],
      dtype='object')
```

In [41]:
```
# X2 = Data.drop(columns=['Stable','PL','CD','HA','VC']) ### todas las columnas
X2 = Data[['PS1', 'PS2', 'PS5', 'PS6', 'FV2', 'TS1', 'TS2', 'TS3', 'TS4', 'CE','EPS1']] ### columnas de mayor relacion
X2
```

Out[41]:

	PS1	PS2	PS5	PS6	FV2	TS1	TS2	TS3	TS4	CE	EPS1
0	160.673492	109.466914	9.842170	9.728097	10.304592	35.621983	40.978767	38.471017	31.745250	39.601350	2538.929167
1	160.603320	109.354890	9.635142	9.529488	10.403098	36.676967	41.532767	38.978967	34.493867	25.786433	2531.498900
2	160.347720	109.158845	9.530548	9.427949	10.366250	37.880800	42.442450	39.631950	35.646150	22.218233	2519.928000
3	160.188088	109.064807	9.438827	9.337430	10.302678	38.879050	43.403983	40.403383	36.579467	20.459817	2511.541633
4	160.000472	108.931434	9.358762	9.260636	10.237750	39.803917	44.332750	41.310550	37.427900	19.787017	2503.449500
...	...	...	...	...	...	...	...	...	...	...	...

Transformar a binario y ordenar la condicion

In [43]:
```
y02 = pd.get_dummies(y2.values.ravel())
# print(y02)
clases = list(y02.columns)
print(clases)
y02
```

```
[90, 100, 115, 130]
```

Out[43]:

	90	100	115	130
0	0	0	0	1
1	0	0	0	1
2	0	0	0	1
3	0	0	0	1
4	0	0	0	1
...	...	...	...	...
2200	1	0	0	0
2201	1	0	0	0
2202	1	0	0	0
2203	1	0	0	0
2204	1	0	0	0

2205 rows × 4 columns

División datos de entrenamiento, validación y prueba.

In [45]:
```
y2.unique()
```

Out[45]: array([130, 115, 100, 90], dtype=int64)

In [46]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X2, y02, test_size=0.2,random_state=0)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[46]: ((1764, 11), (441, 11), (1764, 4), (441, 4))

In [47]:
```
# print(X_train)
# print(y_test)
```

In [48]:
```
print(X_train.shape)
print(y_train.shape)
```

```
(1764, 11)
(1764, 4)
```

Estandarización o escalado de los datos

In [49]:
```
# valores escalares para x
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X_train.shape, X_test.shape
```

Out[49]: ((1764, 11), (441, 11))

Arquitectura de la red neuronal

In [50]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

In [51]:
```
red_neuronal2 = Sequential()
# capa de entrada
red_neuronal2.add(Dense(units=64, kernel_initializer='uniform', activation = 'relu', input_dim=11))
# 1era capa oculta
red_neuronal2.add(Dense(units=64, kernel_initializer='uniform', activation = 'relu'))
#2da capa oculta
red_neuronal2.add(Dense(units=32, kernel_initializer='uniform', activation = 'relu'))
#capa de salida
red_neuronal2.add(Dense(units=4, kernel_initializer='uniform', activation = 'softmax'))
#parametros de la capa
red_neuronal2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
red_neuronal2.summary()
```

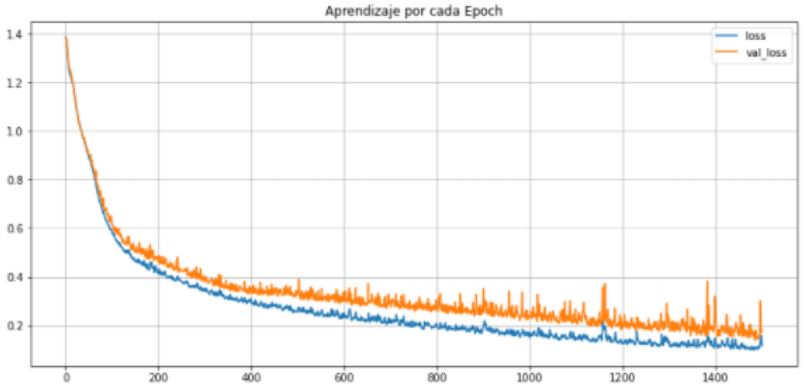
```
Model: "sequential"
```

In [52]:
```
history = red_neuronal2.fit(X_train, y_train,
                            validation_data=(X_test, y_test),
                            batch_size=200,
                            epochs=1500)
```

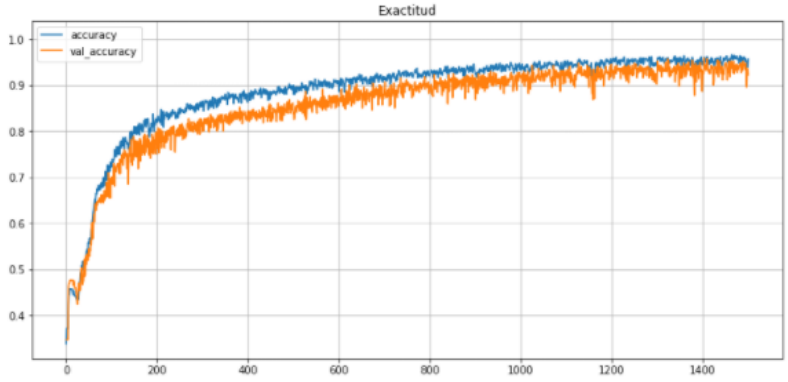
```
Epoch 1/1500
9/9 [==============================] - 1s 92ms/step - loss: 1.3848 - accuracy: 0.3379 - val_loss: 1.3828 - val_accuracy: 0.3
469
Epoch 2/1500
9/9 [==============================] - 0s 3ms/step - loss: 1.3802 - accuracy: 0.3713 - val_loss: 1.3763 - val_accuracy: 0.34
69
Epoch 3/1500
9/9 [==============================] - 0s 4ms/step - loss: 1.3688 - accuracy: 0.3713 - val_loss: 1.3614 - val_accuracy: 0.34
```

visualizacion de desenvolviemiento de la red que no exista desajuste y sobreajuste

In [53]:
```
plt.figure(figsize=(13,6))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['loss', 'val_loss'])
plt.title("Aprendizaje por cada Epoch")
plt.grid()
plt.show()
```



In [54]:
```
plt.figure(figsize=(13,6))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['accuracy', 'val_accuracy'])
plt.title("Exactitud")
plt.ylim([None, 1.04])
plt.grid()
plt.show()
```



Parametros de predicción

In [55]:
```
y_pred = red_neuronal2.predict(np.array([X_test[5]]))
print(y_pred.astype(np.float64))
```

```
[[3.59778829e-09 4.02452424e-03 1.07261717e-01 8.88713717e-01]]
```

In [56]:
```
Y_test = [y_test.columns[i] for i in y_test.to_numpy().argmax(axis=1)]
# Y_test
```

In [57]:
```
y_pred=red_neuronal2.predict(X_test)
y_pred=np.argmax(y_pred, axis=1)
# Y_test=np.argmax(y_test, axis=1)
# y_pred#.shape#, Y_test.shape
```

In [58]:
```
Y_pred = [y_test.columns[i] for i in y_pred]
# Y_pred
```

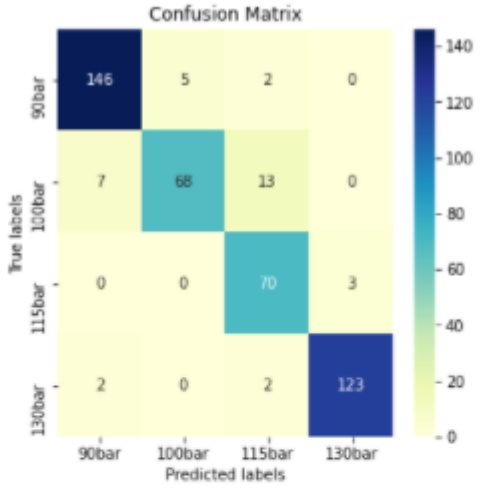
In [59]:
```
from sklearn.metrics import confusion_matrix, classification_report

mc = confusion_matrix(Y_test, Y_pred)
print(confusion_matrix(Y_test, Y_pred))
```

```
[[146   5   2   0]
 [  7  68  13   0]
 [  0   0  70   3]
 [  2   0   2 123]]
```

Resultados de predicción

In [63]:
```
clases2 = [str(i)+'bar' for i in clases]
```

In [64]:
```
fig, ax = plt.subplots(figsize=(5,5))
sns.heatmap(mc,
            xticklabels=clases2,
            yticklabels=clases2,
            linecolor='red',
            fmt='.0f',
            annot=True,
            cmap='YlGnBu',
            ax=ax)
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
# plt.ylabel('y preds')
plt.show()

print(classification_report(Y_test, Y_pred))

```



	precision	recall	f1-score	support
90	0.94	0.95	0.95	153
100	0.93	0.77	0.84	88
115	0.80	0.96	0.88	73
130	0.98	0.97	0.97	127
accuracy			0.92	441
macro avg	0.91	0.91	0.91	441
weighted avg	0.93	0.92	0.92	441