UNIVERSIDAD DE TALCA

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA CIVIL MECATRÓNICA

ANÁLISIS Y DETECCIÓN DE FALLAS EN ENGRANAJES UTILIZANDO APRENDIZAJE DE MÁQUINAS

Memoria para optar al Título de

Ingeniero Civil Mecatrónico

Profesor Guía: Alejandro Jofré Moreno

RODRIGO ALONSO FLORES ZURITA

CURICÓ-CHILE

2022

CONSTANCIA

La Dirección del Sistema de Bibliotecas a través de su encargado Biblioteca Campus Curicó certifica que el autor del siguiente trabajo de titulación ha firmado su autorización para la reproducción en forma total o parcial e ilimitada del mismo.

Curicó, 2023

ANÁLISIS Y DETECCIÓN DE FALLAS EN ENGRANAJE UTILIZANDO INTELIGENCIA ARTIFICIAL

Rodrigo Alonso Flores Zurita

Octubre de 2022

Resumen

Los engranajes son uno de los elementos que siempre está presente en la gran mayoría de las máquinas industriales, son muy comunes y tienen un gran impacto en la cadena de producción debido a que son los encargados de trasmitir energías por medio de las diferentes máquinas, por lo que es de mucha importancia el estudio de la tecnología de diagnóstico de fallas en los engranajes que son utilizadas en los planes de mantenimiento.

Las diferentes fallas que se pueden presentar en los engranajes son trasmitidas por vibraciones, las cuales son capaces de entregar suficiente información capaz de identificar el estado del rodamiento, pudiendo así clasificar las distintas fallas. Para poder procesar esta información, el avance de la tecnología llego a un punto donde la automatización y la programación se fusionaron, pudiendo crear lo que se conoce como machine learning [1], el cual con la suficiente información es capaz de procesar y clasificar las diferentes fallas con una exactitud muy elevada.

En el presente documento se propone la realización de tres modelos de machine learning supervisado con la finalidad de identificar sus falencias y virtudes de cada uno y ver el potencial que tienen estas herramientas para la obtención de un modelo de clasificación basado en inteligencia artificial.

En una primera instancia se explicará la importancia en el área de mantenimiento y la extracción de parámetros para posteriormente trabajar con estos datos de manera tradicional calculando su trasformada de Fourier y los parámetros característicos que los representan.

De manera más profunda se iniciará la creación de los diferentes algoritmos dando énfasis en las características de cada método utilizado, enfocándose en los diferentes parámetros y los resultados obtenidos, para luego variar los parámetros y ver si se lograban mejores resultados o peores.

Como resultado se lograron exactitudes de un 99% para el modelo de support vector machine, 98% en K-nearest neighbor y finalmente un 97% en Random forest. Todos bajo las mismas características, de un 80% de entrenamiento y un 20% de testeo, sacados de la misma base de datos.

Dedicado a mi familia, amigos y todos aquellos que confiaron en mí…

Índice

Lista de Figuras

Lista de Tablas

Abreviaciones

IA	: Inteligencia artificial
ML	: Machine learning
DNN	: Red neuronal profunda (deep neural network)
SVC	: Support vector classification
KNN	: K- nearest neighbor
KPI	: Indicador clave de rendimiento

1. Introducción

1.1 Introducción general

El avance de la tecnología es un tema que con los años cada día se va acrecentando más y más, llegando a un punto en donde la programación y sus aplicaciones han tomado un énfasis muy grande a nivel mundial, estando insertas en muchas cosas que conocemos, de las cosas que conocemos, computadoras, celulares, industrias automovilísticas, análisis mundiales, etc. Una de las ramas que está tomando un énfasis en la última década es la inteligencia artificial [2], un área basada en programación donde su principal base es llegar asimilarse lo mayormente posible al cerebro humano, creando redes neuronales reflejando el comportamiento de las neuronas de los seres humanos. En la actualidad está tan introducido en la vida cotidiana que no notamos el uso de esta tecnología, como son los motores de búsqueda de Google [3], el cual por medio de esta tecnología es capaz de comprender nuestro lenguaje escrito para facilitar la búsqueda de alguna idea que tengamos en mente, esto ocurre gracias al machine learning, el cual por medio de una base de datos la cual vamos creando nosotros con cada búsqueda logra entender nuestros comportamientos para las recomendaciones. Otra gran aplicación de esta tecnología corresponde al área de la salud, aquí es menos común de ver día a día, pero la ayuda que genera y el impacto para las personas ha sido de gran ayuda. La inteligencia artificial está aplicada en el diagnóstico de enfermedades como el covid-19, por medio del análisis de radiografía de tórax [4] esta herramienta es capaz de detectar si el paciente presenta complicaciones por medio de distintos porcentajes y probabilidades que arroja el análisis por medio de inteligencia artificial. Aún más relevante es el uso para la detección de cáncer, es sabido que el cáncer es una de las enfermedades más letales y dolorosas, porque se expande de manera ramificada y produce una muerte lenta si no se detecta a tiempo, con el avance de la tecnología fue capaz de realizar estudios donde se compara un médico experto en el área con la inteligencia artificial, donde está última genera los mismos pronósticos que el doctor que lleva años estudiando casos de cáncer.

El machine learning [5], se remonta en el año 1950, cuando el gran Alan Turing creó el ‘‘Test de Turning’’, Este consistía en una máquina debía engañar a un humano haciéndose creer que se encontraba delante de otro humano y no una máquina. Posterior a esto dos años más tarde Arthur Samuel es capaz de crear un algoritmo que tiene la habilidad de aprender, el cual consistía en un programa que jugaba damas y mejoraba partida tras partida. Posterior a esto en una conferencia nació el termino de ‘‘Inteligencia Artificial’’. Gracias a esto hubo un boom en los años 70, el cual no obtuvo grandes repercusiones debido al poco avance en el área. Pese a todo esto en 1976 nace el primer algoritmo ‘‘Nearest Neighbor’’ que es considerado como el nacimiento de los algoritmos de reconocimiento de patrones.

En el año 2003 empieza un antes y un después, de pasar a la decadencia, el machine learning cayó en las manos de Google, el gigante del internet, debido a que se empieza a trabajar con Big Data [6] con la creación de la primera plataforma de Big Data Open Source, a la cual la llaman Hadoop.

En la actualidad la inteligencia artificial se puede separar en subconjuntos como el machine learning y el Deep learning, estos grandes grupos de la misma manera que subdividen en más subgrupos creando una especia de árbol de la inteligencia artificial. Estas tecnologías se emplean en diferentes áreas, teniendo un gran desarrollo en el Big Data, reconocimiento de patrones entre otros. Pero últimamente se ha implementado en el área industrial, donde se utiliza en la detección de fallas y la clasificación de estas para la prevención y mejora en el área de mantenimiento. Los algoritmos más utilizados son aquellos de red neuronal, como el aprendizaje de máquinas. Todo esto es posible gracias a los avances tecnológicos que existen en la actualidad, la recaudación de datos en industrias antiguamente era nula, debido a que no existía la tecnología y no se pensaba que la recaudación de estos datos pudiera de servir en el futuro. Esto gracias a los sistemas de control y los diferentes artefactos como sensores se puede realizar y es una realidad, con la cual se puede alargar la vida funcional remanente [7] de un equipo. Un ejemplo de estas aplicaciones son los vehículos que se manejan solos, gracias a diferentes sensores se recopila la información en un radio específico alrededor del auto y por medio de inteligencia artificial el vehículo toma la decisión de frenar doblar o aumentar la velocidad en tiempo real.

En todas las industrias existe 3 bases que gobiernan todo el proceso productivo, estas son la reducción de costos, mejora en confiabilidad e incrementos en la productividad. Para poder lograr esto el machine learning ha sido clave debido a su gran intervención en líneas de producción con algoritmos que detectan fallas anticipadamente. Esto enfocado principalmente a equipos rotatorios como son los engranajes y rodamientos, los cuales están presentes en la mayoría de los equipos a nivel industrial siendo los más propensos a fallas debido a las grandes fuerzas que son sometidos. Estos los podemos encontrar en el caso de los engranajes en reductores, sistemas de trasmisión, molinos entre otros.

Chile es un país considerado minero debido a las grandes cantidades de cobre y otros minerales como el litio. En todos los procesos mineros siempre hay un equipo con engranaje en intervención. La gran importancia de este elemento sobre todo en la minería es fundamental, debido que la maquinaria utilizada es de gran envergadura, por lo que el reemplazo de una pieza tiene un costo de miles de millones de pesos considerando el trasporte, el cambio de la pieza, y el tiempo de la parada del equipo, haciendo que las bases de productividad se desplomen. Por esto la detección temprana de fallas y el análisis de fallas en engranajes es de vital importancia, ahorrando costos para la empresa muy grandes.

1.2 Estado del arte

En este subcapítulo se estudiará el estado del arte de las diferentes técnicas de machine learning y aplicadas principalmente a fallas en activos físicos en la industria, analizando las principales categorías que existen dentro de esta rama del ML describiendo algunos estudios previos de investigación que han sido de soporte para el desarrollo de este trabajo. Con la finalidad de generar una intervención temprana se presentarán distintas herramientas de monitoreo que existen en la actualidad para el área de mantenimiento.

1.2.1 Engranajes

Los engranajes son elementos que desde sus inicios han servido para la transmisión de movimiento, por medio de la literatura se mencionan engranajes, pero no existe mucha claridad desde cuando se utilizan estos artefactos. En la actualidad los engranajes son uno de los elementos fundamentales dentro de cualquier proceso industrial, que requiera un

movimiento debido a que su principal función es transmitir energía mecánica de un componente a otro.

Existen distintos tipos de engranajes, entre los cuales se pueden identificar:

- Rectos.
- Helicoidal.
- Cónicos.
- En V.
- Cilíndrico dentado recto interior.

Dentro de estos tipos de engranajes empiezan a salir distintas combinaciones como un cónico recto o cónico helicoidal, entre otros.

Figura 1.1. Engranaje recto, cónico y helicoidal. [8]

Como cualquier elemento, los engranajes presentan fallas y gracias a los avances tecnológicos estos cada día son más detectables. Antiguamente la falla de esta pieza implicaba el cambio o en su defecto la pérdida del equipo, hoy en día esto cuesta mucho de que ocurra, priorizando en procesos industriales la pérdida de tiempo. Es por esto por lo que existen diferentes métodos para la detección de fallas.

Estas fallas se pueden clasificar de la siguiente manera:

- Rotura.

- Desgaste.
- Fatiga superficial.
- Deformación plástica.

Estas fallas se presentan de diferentes maneras y usualmente ocurren por un mal mantenimiento o por alguna falla humana que terminó desenvolviendo tal situación, según la empresa esto se puede detectar generalmente por un análisis de vibraciones, la cual es la técnica más utilizada en la actualidad.

1.2.2 Análisis de señales

El análisis de señales es una de las herramientas más importantes en cualquier proceso industrial, el área encargada es la de mantenimiento y es aquella que tiene que procesar los datos adquiridos para la determinación del estado de una línea de producción.

Este análisis es de los más utilizados para la detección de fallas tempranas y en la actualidad se utilizan herramientas especializadas de inspección sintomática para su detección. Una de estas es el análisis de vibraciones la que se caracteriza por la utilización de un sensor, los más comunes son la utilización de transductores de vibración. Generalmente, hay dos tipos de sensores de aceleración: piezoeléctricos y capacitivos. Los primeros tienen buenas características dinámicas, pero carecen de lo físico-estático, mientras que los segundos se utilizan principalmente para la medición de características estáticas y cuasi estáticas [9]. La señal entregada en forma de ondas puede ser vista en el dominio del tiempo o frecuencia, gracias a la emisión de una señal de voltaje emitida por el sensor, el cual está conectado a un colector de datos que registra la señal. Como resultado, la señal es analizada por un personal específico en vibraciones o un algoritmo ‘‘inteligente’’. Por lo general se utiliza el dominio de la frecuencia para este análisis para una mejor interpretación de la señal, pudiendo identificar anomalías de una manera más rápida, como se puede observar en la Figura 1.2, en la cual se observan picks, que son específicos de cada elemento, si hubiera una perturbación en el sistema se vería reflejado, indicando una falla en el rodamiento, engranaje o lo que se esté analizando.

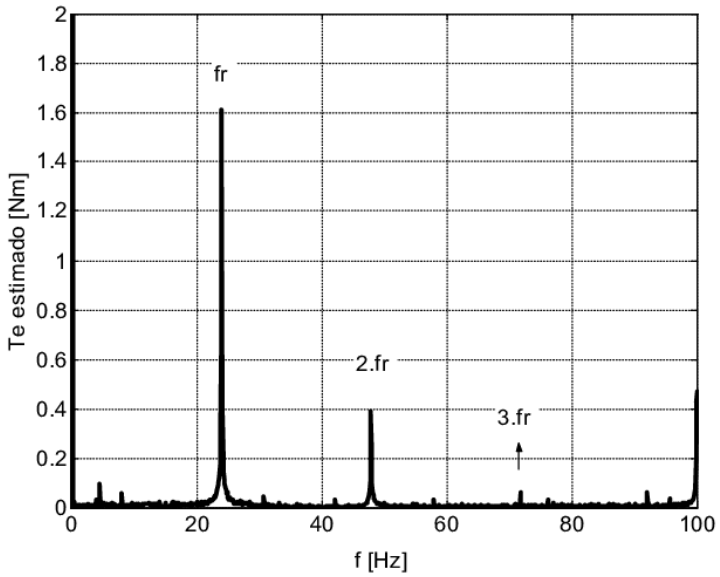


Figura 1.2. Espectro de frecuencia. Fuente: [10].

1.2.3 Resúmenes estadísticos

Los resúmenes estadísticos, son una herramienta potente para la detección de fallas, por medio de los datos y gráficos se puede analizar distintas tendencias o patrones en los componentes o maquinas. Estas técnicas son utilizadas para determinar si las condiciones normales del proceso están siendo alteradas o están declinándose a una tendencia para esta atento a una posible falla. Estas medidas se pueden clasificar en 3 tipos.

- Medidas de tendencia central:

En las medidas de tendencia central se hace referencia en el promedio de los datos, la mediana y la moda, estas características son muy características para determinar un valor numérico que identifique una n cantidad de muestras y poder trabajar bajo estos valores [11].

- Medidas de dispersión:

En las medidas de dispersión contamos con la varianza, el rango y el coeficiente de variación, los cuales son variables que nos indican si los datos que se están analizando varían mucho con la mediana, influenciando directamente en el resultado final, como nos indica Miguel en un estudio de análisis de vibraciones [11].

- Medidas de Forma:

Las medidas de forma se utilizan para determinar si los valores que tenemos tienen alguna tendencia, para esto se utiliza el coeficiente de simetría y curtosis que nos indica como la muestra de datos se comporta, siendo esta simétrica, con tendencia positiva o negativa, además de la forma de la gráfica, siendo leptocúrtica o mesocúrtica [11].

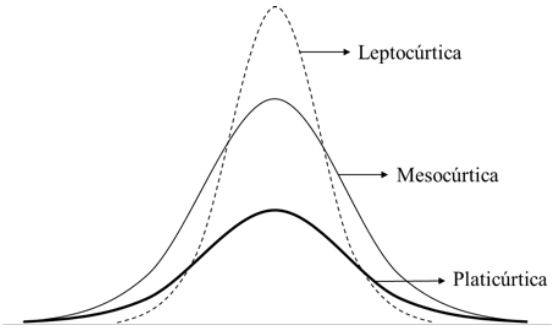


Figura 1.3. Coeficiente de curtosis. Fuente: [12]

1.2.4 Machine learning

En el transcurso de los años ML y el DNN han tomado una gran relevancia lo múltiples algoritmos que se han podido desarrollar han evolucionado al punto de detectar fallas con mucho tiempo de anticipación, cosa que con instrumentos tradicionales habría sido muy difícil de predecir, estas técnicas obtienen resultados de alta precisión como lo expone Alexander Huertas [13] enfocándose en machine learning supervisado y Deep learning. El cual mediante una base de datos de la NASA para turbinas aeronáuticas pudo plasmar diferentes tipos de modelos obteniendo un resultado muy eficiente en un modelo híbrido mezclando los dos tipos de métodos.

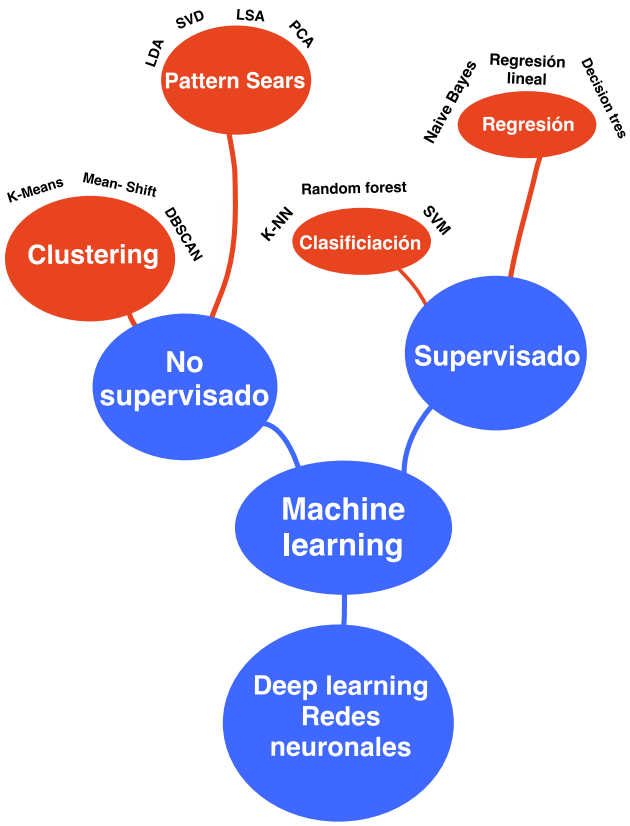


Figura 1.4. Mapa conceptual machine learning. Fuente: Elaboración propia.

Por el otro lado existe el Deep Learning una rama muy especial debido a que simula el cerebro humano con sus redes neuronales simulando el proceso de una neurona de un ser humano. Es una rama que poco a poco está tomando fuerza y cada vez está logrando cosas más interesantes como lo expresa en su tesis de grado Sebastián Montagna [14] en donde realizo un detector de fallas utilizando Deep learning en donde mediante distintas variables como imágenes térmicas, análisis de partículas, vibraciones entre otros entreno una red neuronal causal obteniendo como resultado excelentes resultados en la detección de fallas en turbinas.

1.2.5 Engranajes y el aprendizaje de máquinas

Muchas personas, han buscado la manera de poder entender mejor el proceso de falla de los engranajes u otras piezas fundamentales en los procesos industriales, las cuales están presente en la mayoría de los equipos utilizados en el día a día. Es por esto por lo que el

diagnostico de fallas se ha empezado a popularizar de la mano del machine learning, debido a su capacidad de procesamiento y versatilidad como lo presentan Pei Cao, Zhang y Tang en su documento llamado ‘‘Preprocessing-Free Gear Fault Diagnosis UsingSmall Datasets With Deep Convolutional Neural Network-Based Transfer Learning’’ [15] quienes realizaron un estudio en donde por medio de pequeños conjuntos de datos se realizó un diagnóstico de fallas sin un pre procesamiento por medio de redes neuronales convolucionales. El objetivo principal era el estudio de sistemas mecánicos complejos de cajas de cambios por el cual como resultado se obtuvo grandes resultados con un sistema robusto y viable con horizontes a otras prácticas de diagnósticos de fallas.

Por otro lado Li, Sanchez, Zurita y compañía [16] al igual que Cao realizaron un diagnóstico de fallas de la caja de cambios, pero ellos se enfocaron en otro método, este es el de bosque aleatorio o random forest, por medio de señales acústicas y vibratorias, una mezcla de ambas disciplinas que nos llegan a indicar cuando nuestro equipo está fallando. La metodología detrás de esta combinación de técnicas es trabajarlas por si solas adquiriendo datos, luego se utiliza la transformada de wavelet [17] para cada una, desarrollando dos máquinas Boltzmann profundas [18]. Finalmente, para poder realizar la conexión entre ambas, se aplicó random forest para la fusión de ambas salidas, dando como resultado un 97.68% de exactitud para 11 patrones distintos concluyendo que ambas técnicas pueden mejorar las capacidades de diagnóstico para los engranajes abriendo paso a la combinación de técnicas.

1.2.6 Resolución estado del arte

Según lo expuesto en el estado del arte podemos comprobar que dentro de la historia el hallazgo de las diferentes fallas presentes en los equipos industriales la tecnología ha ido avanzando en conjunto con el avance de los diferentes equipos, aunque estos han mejorado para proporcionar una vida más duradera y con menos fallas. En este sentido ha sido muy difícil que se cumpla en un cien por ciento, debido a que los diferentes equipos son sometidos a diferentes circunstancias en las distintas plantas industriales. Lo más utilizado en la última época son los ya mencionados análisis de vibraciones, los cuales por medios de la transformada de Fourier son capaces de analizar si un equipo presenta algún fallo. En conjunto con estos datos se agregan los resúmenes estadísticos, que analiza un equipo de

manera más prolongada que un análisis de vibraciones debido a que se utilizan métodos estadísticos para ver la tendencia o moda de los datos, curtosis, entre otros, los cuales son capaces de advertir si el equipo analizado presenta anomalías o alguna tendencia.

Por ultimo y en un rango distinto a los anteriores, en la última década se ha tratado de implementar inteligencia artificial en la industria, como lo es la clasificación para la detección de fallas en tiempo real de un equipo industrial. Esto ha sido difícil, debido a que la mayoría de las industrias no cuenta con una data de los equipos activos, trabajan con mantenciones correctivas o a lo más preventivas las cuales son muy poco bien implementadas. Es por esto por lo que se ha ido de a poco tratando de impartir estas nuevas ideologías para que dé a poco la inteligencia artificial tome más terreno.

1.3 Objetivos

1.3.1 Objetivo General

- Desarrollo de algoritmo de análisis de mantenimiento aplicado a sistemas de engranes con uso de técnicas de aprendizaje de máquina.

1.3.2 Objetivos Específicos

- Comprender los diferentes engranes y sus diferentes fallas para la detección temprana.
- Comprender los datos obtenidos del engrane, para su organización y planteamiento previo al procesamiento de datos.
- Procesar los datos a través de estadística descriptiva para la identificación de patrones y tendencias.
- Desarrollar los datos obtenidos para su análisis en el dominio del tiempo y la frecuencia.
- Desarrollar una serie de algoritmos de aprendizaje de máquinas para la identificación de fallas por medio del método de clasificación.
- Probar los diferentes algoritmos creados en diferentes circunstancias para la búsqueda óptima del mejor resultado.
- Identificar de manera detallada los diferentes algoritmos de clasificación de engranajes para una posterior utilización en el área de mantenimiento.

- Comparar los algoritmos generados a través de indicadores de desempeño pertinentes.

1.4 Alcances y limitaciones

1.4.1 Alcances

- El algoritmo se realizará por medio del servidor web Google Colaboratory en lenguaje Python.
- Los datos del engrane pertenecen al paper ‘‘Preprocessing-Free Gear Fault Diagnosis UsingSmall Datasets With Deep Convolutional Neural Network-Based Transfer Learning’’ [15].
- Se desarrollarán los algoritmos de clasificación: support vector machine (VSM), random forest y K- nearest neighbor.

1.4.2 Limitaciones

- Los datos recaudados no consideran perturbaciones en el sistema debido a que se extraen de un banco de fallas.
- El algoritmo desarrollado estará optimizado para el contexto operativo de la base de datos analizada [19].
- Se utilizará la metodología Ceteris paribus [20], para los diferentes algoritmos, variando aquellos parámetros más relevantes, dejando como constante el resto.

1.5 Metodología

Para la implementación y puesta en marcha del sistema de mantenimiento predictivo, se debe seguir una serie de pasos para lograr los objetivos propuestos y obtener las pruebas experimentales que exige el proyecto. Los pasos necesarios para lograr lo propuesto son:

1.5.1 Estudio de la estructura y fallas de los engranajes

Corresponde a la primera etapa y se caracteriza por el estudio de la estructura de los diferentes engranajes que existen en la actualidad, los más utilizados en la industria y sus cualidades, además de la identificación de las diferentes fallas que pudieran presentar dependiendo del engranaje.

1.5.2 Estudio teórico y análisis de mantenimiento predictivo

Se revisará y estudiarán las diferentes propuestas que existen a la fecha para poder realizar mantenimiento predictivo, las herramientas que se utilizan a nivel industrial y como la inteligencia artificial se puede implementar al área de mantenimiento en conjunto con herramientas tradicionales.

Para esto es necesario obtener información de documentos relacionados con mantenimiento preventivo y predictivo, además de inteligencia artificial. Esta información será obtenida de la plataforma IEEE-Xplore de la IEEE (Institute of Electrical and Electronics Engeneers), la cual posee convenio con la Universidad de Talca, siendo posible acceder a toda la información necesaria para desarrollar este tipo de proyectos, además de información de diferentes papers de temas relacionados en la web.

1.5.3 Estudio de análisis estadísticos en análisis de vibraciones

Se realizará un estudio teórico sobre las diferentes alternativas que existen en el área de estadística relacionado con el análisis de vibraciones para la identificación de patrones característicos para la clasificación de fallos en engranes dando énfasis en aquellos más utilizados que tengan un fundamento solido que los respalde en su relación con el análisis de vibraciones.

Para un aprendizaje óptimo se revisará bibliografía, documentos y revistas especializadas en el análisis de vibraciones, además de los elementos estadísticos más utilizados a la fecha que se utilicen en la industria y la detección de fallas.

1.5.4 Estudio del lenguaje de programación Python y sklearn

Se estudiará el lenguaje de programación de Python debido a la gran potencia que tiene y lo tan vinculado que se encuentra con la inteligencia artificial. Además de las librerías de sklearn [21], las cuales se caracterizan por ser las más utilizadas en machine learning.

Para esto se estudiarán diferentes artículos donde enseñen el lenguaje de programación Python, además del estudio de la web de sklearn, donde enseñan los diferentes métodos que existen como lo son el de clasificación, regresión, clustering, entre otros.

1.5.5 Implementación de las metodologías a utilizar

Se implementará las diferentes alternativas presentes en aprendizaje de máquinas, relacionadas con clasificación enfocadas a las fallas presentes en engranajes. De las cuales se seleccionaron Random forest, K-nearest neighbor y Support vector machine las cuáles serán las que se utilizaran para realizar la clasificación de las fallas del engrane, representando su efectividad mediante una matriz de confusión y análisis de sus KPI.

Para esto se estudiarán diferentes documentos en la web y en la IEEE, que detallen el algoritmo además de basarse en la base de datos que entrega sklearn donde explica los parámetros utilizados y su implementación en Python.

1.5.6 Análisis de resultados

Esta es la etapa final, en donde se analizan los resultados experimentales como la precisión, la exactitud, el recall. Además de la comparación con los resultados obtenidos de manera estadística, buscando obtener una relación entre ambas.

1.6 Organización de la memoria

En el capítulo 1 se desarrollará una introducción identificando el objetivo principal del desarrollo del documento con los objetivos principales y delimitando los alcances y limitaciones de la memoria. Se hablará del estado del arte y la metodología a utilizar a lo largo de la memoria identificando el uso de software, lenguaje de programación entre otras cosas.

En el capítulo 2, se desarrollará de manera profunda todos lo visto previamente en la introducción teniendo en cuenta el objetivo principal y los secundarios. Partiendo con la herramienta de software utilizada y el porqué de su elección, al igual que el del lenguaje de programación.

Luego de haber indagado sobre el software se trabajará con los datos obtenidos del engranaje, indicando la obtención de estos, como este compuesto para un procesamiento de datos y obtención de parámetros relevantes. Realizando por medio del software un análisis de vibraciones y las características principales presentes en los gráficos en el dominio del tiempo y de la frecuencia. Luego de haber realizado este análisis, se obtendrán lo que se le

conoce como resúmenes estadísticos, el cual trabaja con parámetros estadísticos como la moda, la mediana, el RMS, curtosis, entre otros, para ver el comportamiento de la falla en relación con sus valores normales.

Para finalizar el capítulo se explicarán los métodos de machine learning utilizados, y como estos se implementan en el algoritmo desarrollado, presentando las diferentes alternativas que presenta estas metodologías y la variación de sus parámetros para la búsqueda de los mejores resultados de clasificación de fallas.

En el capítulo 3 y final de la memoria se abordarán las ideas obtenidas en el desarrollo o capítulo 2, identificando si los resultados son aceptables o viables para los datos del engranaje y por consiguiente una comparación de los diferentes algoritmos creados y como estos se pueden llegar a implementar a otras piezas o maquinaria industriales en ayuda del área de mantenimiento y la identificación de fallas futuras. Además, se proponen trabajos futuros para le expansión de los resultados obtenidos para tener una visión más subjetiva de lo obtenido y analizar el comportamiento de diferentes datos bajo las mismas condiciones que se implementaron en el documento.

2. Análisis descriptivo

2.1 Tipos de engranajes y sus fallas.

Existen una gran variedad para trasmitir el movimiento, pero uno de los más utilizados son los engranajes, este es un elemento el cual dispone de ‘‘dientes’’ que encajan entre sí para trasmitir el movimiento. La gran ventaja de los engranajes es que ocupan un espacio muy reducido y al mismo tiempo tienen la capacidad de trasmitir una cantidad de potencia. Un engranaje está compuesto principalmente de 4 partes, un diente de engrane, una circunferencia exterior, una interior y una circunferencia primitiva.

2.1.1 Tipos de engranajes

Dentro de los engranajes se pueden identificar varias configuraciones, cada una con una finalidad en concreto, maximizando virtudes que otras configuraciones no pueden presentar, de las cuales podemos encontrar las siguientes clases:

2.1.1.1 Rectos

Se utilizan en transmisiones de ejes paralelos. Son uno de los más utilizados a nivel industrial y dentro de la mayoría de los mecanismos. Se utilizan principalmente para grandes reducciones de velocidad y trasmisión de grandes potencias.

Una de las principales ventajas son la sencillez al momento de su fabricación, ya que es un elemento de fácil construcción y siempre hay stock, dependiendo siempre del tamaño del engranaje. Suele ser más eficiente en comparación con un helicoidal y al trabajar con elementos de forma paralela no existe una fuerza axial, lo que significa menos elementos que produzcan desgaste.

Las principales desventajas de estos engranajes es la gran cantidad de estrés que presentan y la desventaja al no poder trasmitir energía en largas distancias, debido a que se necesitarían muchos engranajes de forma paralela o muy grandes.

Figura 2.1. Engranaje recto. Fuente: [8].

2.1.1.2 Helicoidal

Los engranajes helicoidales se caracterizan por su forma, en comparación con los rectos estos poseen una inclinación entre 15 y 30 grados, generando una forma oblicua. Su principal diferencia es la transmisión de potencia, debido a que este se caracteriza por poder trasmitir esta de forma paralela, cruzada o incluso perpendicular.

Al contar con una forma oblicua, sus dientes suelen estar más en contacto con el otro engrane, por lo que la transmisión de fuerza es más silenciosa, uniforme y segura.

Las principales desventajas es la cantidad de fricción que se genera debido al número de dientes que están en contacto, por lo que genera una reducción en el tiempo de mantenimiento. Al contar con una inclinación en sus dientes, se produce fuerzas axiales, por lo que produce una mejor eficiencia de trasmisión y por consiguiente genera más calor de lo normal por el roce.

Figura 2.2. Engranaje helicoidal. Fuente: [8]

2.1.1.3 Cónicos

Los engranajes cónicos se emplean para trasmitir movimiento entre ejes perpendiculares o para ejes con ángulos distintos a 90 grados, se pueden encontrar principalmente de forma cónica de los cuales pueden ser los dientes rectos o curvos, siendo muy utilizados en la industria automotriz estos últimos, debido a su gran capacidad de transmisión de potencia.

El mantenimiento que se le realiza es muy básico y el rendimiento en comparación a los demás engranes es muy superior, además de eliminar cualquier tipo de deslizamiento debido a su forma particular.

Estos engranajes pueden ser de forma helicoidal, para la trasmisión de movimiento en 90 grados. Los hipoides conformados de un piñón reductor y una rueda y los engranajes cónicos con dientes rectos, que son utilizados principalmente para la reducción de velocidad.

Figura 2.3. Engranaje cónico. Fuente:[8]

2.1.2 Tipos de fallas

Los engranajes son piezas que conforman mecanismos muy complejos, como toda pieza sufre desgaste, ya sea por condiciones internas, como externas que terminan desencadenando problemas que si no se tratan a tiempo provocan malestares más grandes. Estos fallan de varias maneras, generalmente por oxido, roturas o fatiga. Evaluar en que parte se sitúa el problema y el daño provocado es una tarea muy complicada que poco a poco se está mejorando gracias al avance de la tecnología. Principalmente esta labor la realiza un analista de vibraciones, el cual por medio de una máquina especializada analiza las vibraciones de los equipos determinando por medio de un análisis que parte está fallando. En el último tiempo se está empezando a insertar en el mundo industrial la inteligencia artificial, machine learning y el aprendizaje de máquinas, el cual se espera que para los próximos años se cuente con la información suficiente para poder afrontar estas situaciones de conflicto de manera más rápida y minimizando los costos por detención.

Las fallas se pueden clasificar de la siguiente manera:

2.1.2.1 Fallas por desgaste

Las fallas por desgaste se pueden definir como el deterioro que sufren los diferentes dientes y por el cual estos empiezan a desgastarse reduciendo su espesor, lo que provoca un cambio en el perfil del engranaje. Estas fallas se producen generalmente por el contacto entre metal y metal, el cual se produce por fallas en la película de lubricante que las protege de estas fallas, otra causante es la presencia de partículas abrasivas en el lubricante, lo que provoca un desgaste en los dientes.

Como todo elemento industrial existen normas que están para que se cumplan de manera que todos las cumplan y exista un estándar a nivel mundial. El diseño y fabricación de engranajes están normalizados por la norma AGMA [22]. En el caso del análisis de vibraciones de los engranajes no se sigue una norma específica, pero hay elementos que se deben conocer.

- Piñón: Hace referencia al engranaje más pequeño de un tren de engranes.
- Engrane o corona: Es el engranaje más lento del tren de engranajes.
- Blackash: Hace referencia a al espacio que se encuentra entre los dientes. Esta dimensión cumple la función de una buena lubricación, además de otorgar un espacio para la expansión térmica, evitando que se produzca una unión por fusión entre ambos dientes.
- Holgura: Es el espacio que se encuentra entre la punta del diente y el fondo de otro diente.

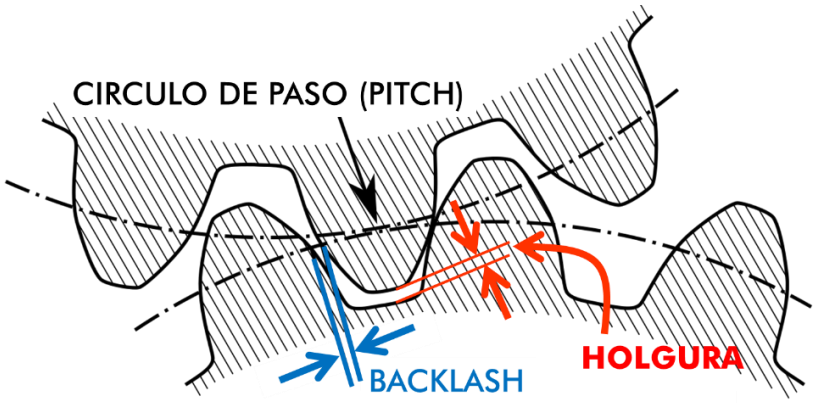


Figura 2.4. Backlash y holgura en engranajes. Fuente: [23].

Por diversos motivos los engranajes pueden sufrir algún desperfecto, ya sea por mala manipulación o por algún desperfecto de origen. Es por esto por lo que existen diferentes tipos de desgaste de los cuales se pueden nombrar los siguientes:

- Desgaste pulimentado.

  Este desgaste se caracteriza por un roce producido entre metal y metal, el cual produce un efecto de pulimiento en los dientes del engranaje. Este efecto ocurre en bajas velocidades y con una lubricación cercana al límite, produciendo una película muy delgada entre los dientes en contacto produciendo este efecto. Este defecto no trae consigo consecuencias muy graves, solo basta con cambiar el lubricante a uno nuevo.
- Desgaste moderado y excesivo.

  Un desgaste moderado ya es considerado una remoción del material, en este caso el metal de ambos engranajes. Este tipo de desgaste se produce por un trabajo con lubricación al límite o contaminación en el lubricante. Este desgaste puede llegar a ser excesivo cuando existe una gran cantidad de material removido en la superficie lo que puede llegar a provocar picaduras, además de perder el perfil original de los engranes.

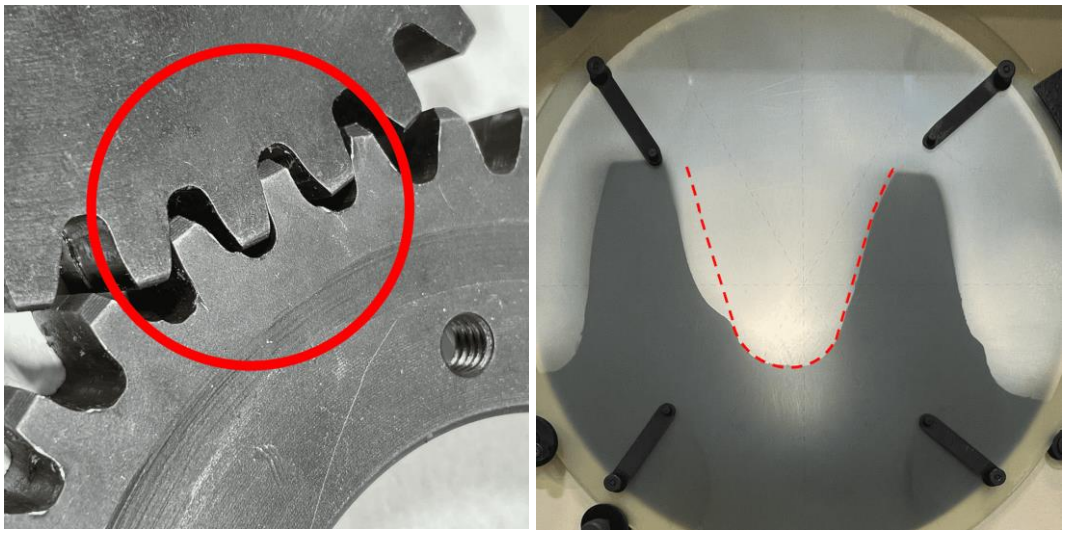


Figura 2.5. Desgaste excesivo engranaje. Fuente: [24]

- Desgaste abrasivo.

  Es un desgaste que se caracteriza por dejar zonas planas y gastadas que interrumpen el perfil del diente. Comienza en la parte inferior de contacto de un solo diente y termina en la línea de paso, comenzando nuevamente allí para terminar en el punto de contacto más alto de un solo diente. Se presenta así debido a las grandes cargas que generalmente se producen en el sector.

  Este desgaste se produce por la presencia de elementos presentes en el lubricante, por lo que identifican estas huellas es necesario revisar los filtros y eliminar los sedimentos que se encuentren en el lubricante.

Figura 2.6. Desgaste abrasivo en engranaje. Fuente: [25].

- Desgaste corrosivo.

  Es un deterioro ocurrido por una acción química. Se caracteriza por una gran cantidad de picaduras presente en la superficie de trabajo del engranaje. Esto puede producirse por el agua y la presión de trabajo ejercida en el engranaje o bien por alguna otra sustancia corrosiva.

Figura 2.7. Desgaste corrosivo engranaje. Fuente: [25].

- Desgaste adhesivo.

  Es un desgaste que ocurre de manera muy rápida en los dientes del engranaje causado por grandes fuerzas adhesivas producto del contacto entre dientes. Debido a la rugosidad en la superficie de contacto de los dientes, hay puntos que se tocan entre sí, produciendo elevadas temperaturas provocando una micro soldadura entre ambas superficies. Por efecto de movimiento estas micro soldaduras se rompen, causando diferentes daños al engranaje.

2.1.2.2 Fallas por fatiga

Una de las fallas más comunes es la falla superficial, esta ocurre incluso con una lubricación adecuada, es el resultado de repetidos esfuerzos en la superficie formando una grieta en la superficie de contacto, generando con el tiempo rotura o desprendimiento de algún diente del engranaje.

Existen diversos lugares donde son propensos estos tipos de falla. Los piñones helicoidales de dureza media y de 20 o más dientes se tienden a picar a lo largo de la línea primitiva

En la Figura 2.8, se puede observar cómo actúa el movimiento repetitivo en un solo sentido del engranaje, donde se nota un deslizamiento del metal hacia el exterior por el esfuerzo constante.

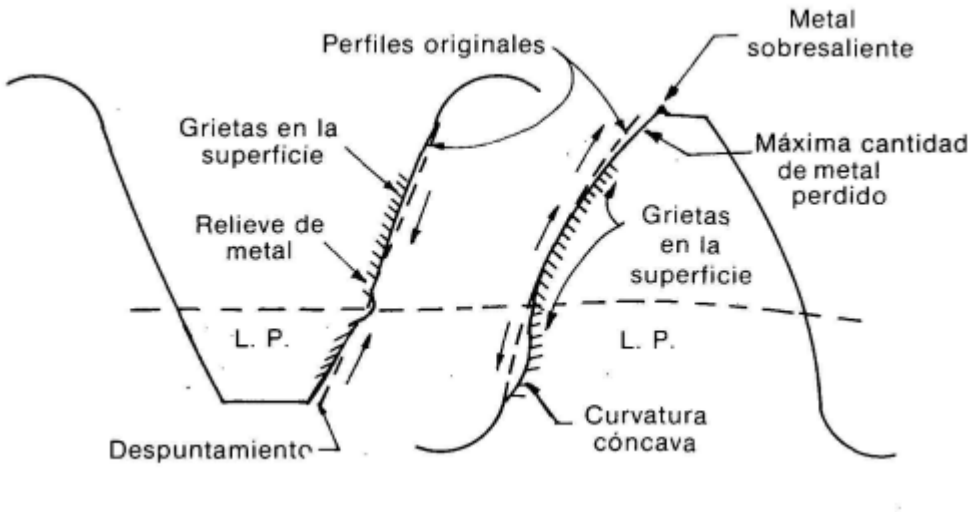


Figura 2.8. Efectos del sentido de deslizamiento sobre un diente de un engranaje. Fuente: [26].

La fatiga superficial se puede encontrar de diferentes formas las cuales son:

- Picado inicial.

  Consta de un picado en la superficie muy leve, el cual al distribuir mejor la carga la falla deja de producirse.
- Picado destructivo.

Es el grado mayor al picado inicial, ocurre cuando no se distribuye las cargas continuando con el desgaste produciendo más picaduras en la superficie. Este tipo de fallas termina produciendo una destrucción del perfil y originando grietas más grandes llevando consigo la producción de una fractura del diente.

Figura 2.9. Fractura por picaduras. Fuente: [26].

- Desprendimiento.

  Es una falla que ocurre por problemas metalúrgicos, el cual lleva consigo el desprendimiento de pedazos de metal considerables. Esta falla puede ocurrir por un problema del material, el cual pudo haber estado con imperfecciones que con las distintas cargas a las que se somete desprende un diente o parte de él. La segunda falla corresponde a que el material está sujeto a diferentes esfuerzos, ya sean de corte de compresión y tensión, el cual cualquiera de estos tres que llegue a su punto máximo generara daños al diente. Estos esfuerzos son alimentados principalmente por sobre cargas o desalineamientos produciendo el desprendimiento de un diente en el engranaje.

Figura 2.10. Desprendimiento de diente por sobretensión. Fuente: [26].

2.1.2.3 Fallas flujo plástico

Se genera principalmente en engranajes a altas revoluciones, debido que se necesita un foco de calor para generar deformaciones en los dientes. Cada diente soporta una cierta deformación, llegado a la deformación plástica permanente el diente no volverá a su orientación principal derivando consigo problemas de desalineación, ruidos y vibraciones. Lo que hace que pueda producirse un golpe martillo en la superficie del engranaje pudiendo romper algún diente. Las grandes temperaturas y las deformaciones producen que el material de la superficie se creen rebordes de altura irregulares generando una serie de alteraciones en el funcionamiento del engrane.

2.1.2.4 Fallas combinadas

Las fallas combinadas son producidas por una serie de acontecimientos que producen un efecto cadena, una mala alineación puede generar un desgaste que con el tiempo puede producir fatiga pudiendo hacer que se rompa un diente y así generar diferentes acontecimientos que terminan de una forma u otra dañando la estructura del engranaje.

2.2 Herramienta de software utilizada

2.2.1 Lenguaje de programación

Para poder realizar el algoritmo, es necesario trabajar en un lenguaje que se adapte a las necesidades del machine learning, para esto es necesario considerar algunos factores que son relevantes a la hora de poder decidir el camino a seguir.

Existen muchos lenguajes de programación, pero uno de los más versátiles es Python [27], debido a su sencillez y versatilidad convencen a los usuarios en elegir este lenguaje de programación. Es por esto por lo que cuenta con librerías dedicadas a machine learning como Scipy, Numpy, scikit-learn y Pandas relacionadas con temas como análisis de datos, depp learning, entre otros, simplificando el lenguaje entre hombre y máquina en la toma de decisiones. Además, cuenta con un gran respaldo de los usuarios enseñando de manera didáctica o por medios de textos, dedicados explícitamente al aprendizaje del lenguaje, además cuenta con un gran soporte en distintas plataformas como GitHub o distintos foros en internet.

2.2.2 Google Colaboratory

Para poder desarrollar el código es necesario un ecosistema que sea acorde al lenguaje y para esto entre los distintos software que hay en la actualidad se eligió Google colab, un entorno de programación que se diferencia de los demás debido a su gran particularidad de poder trabajar en línea con una GPU otorgada por la misma plataforma de Google, generando un vínculo además con Google drive para el almacenaje directo desde la nube, pudiendo suministrar los recursos del computador en otras tareas, además de poder trabajar con gran volumen de datos desde cualquier computador, sin tener el inconveniente que los datos se demoraran en cargarse perdiendo muchas veces el avance por errores del procesador teniendo que reiniciar el programa.

2.2.3 Comparación Google colab vs GPU tradicional

El poder con que cuenta colab se ve reflejado gracias a Jeremy DiBattista [28] quien comparó una GPU Nvidia GTX 1080 vs colab, mostrando interesantes resultados, que vamos a comentar a continuación.

La herramienta desarrollada por Google Research puede asignar distintas GPU al azar, estos recursos provienen de los servidores de Google que en esos momentos no se estén utilizando. Colab afirma que la mayoría de las GPU que se asignan son Nvidia K80s, T4s, P4s y p100s.

Para este experimento se utilizó una biblioteca de Python llamada AI Benchmark que utiliza TensorFlow para ejecutar 42 pruebas en 19 secciones diferentes, para así generar las distintas puntuaciones observadas en la Figura 2.11. Pudiendo observar que la mayor punición en las diferentes ramas es la P100 quien fue alrededor de 4 veces más eficiente que la más lenta del grupo.

La más rápida equivalía a una RTX 2070, mientras que la más lenta a una GTX 1050Ti, que no está nada de mal para una GPU que es gratuita.

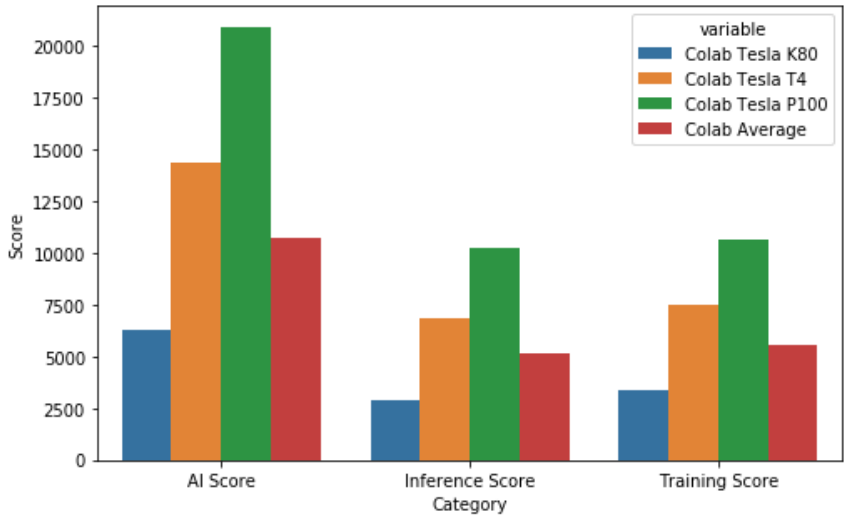


Figura 2.11. Puntuación asignación GPU Google Colab. Fuente: [29].

Ahora bien, DiBattista utilizó directamente su GPU e hizo las mismas pruebas, utilizando sus propios recursos, alcanzo resultados muy buenos comparativamente, los que se pueden observar en la Figura 2.12.

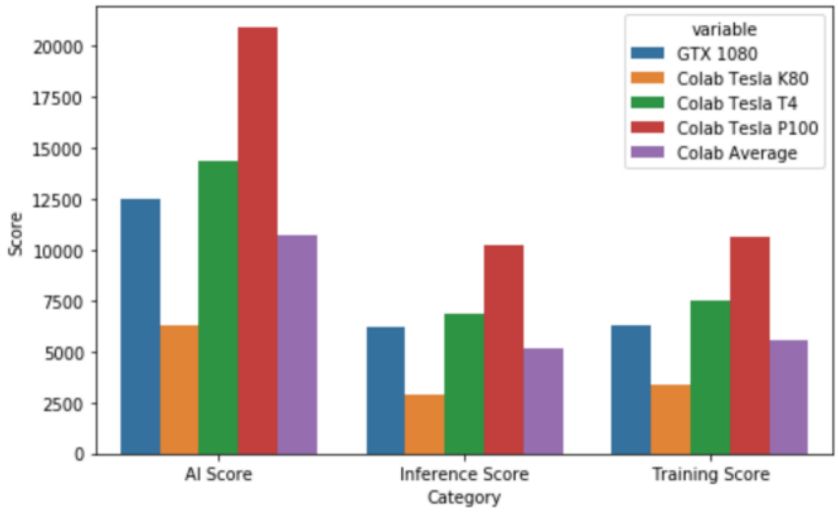


Figura 2.12. Comparación GPU GTX 1080 vs Google colab. Fuente: [29].

Como conclusión se puede observar que Google colab es una herramienta muy poderosa, enfocada principalmente a aquellas personas que recién están partiendo en el mundo de la inteligencia artificial, dando unos resultados muy interesantes para ser una herramienta gratuita.

Un aspecto negativo es que los recursos otorgados, también son desvinculados al momento de estar inactivo mucho tiempo, por lo que es necesario cargar todo de nuevo, lo que podría causar pérdidas de tiempo muy grandes cuando los códigos sean muy grandes, sobre todo cuando consumen una gran cantidad de GPU.

2.3 Adquisición de datos

2.3.1 Datos del Engrane

Los datos del engrane se obtuvieron por medio de la página web Figshare [30], una página web desarrollada sin fines de lucro en donde cualquier usuario pueda subir datos, investigaciones, entre otros para difundir de manera masiva el avance científico. Estos datos se utilizaron para el diagnóstico de fallas en engranajes sin procesamiento utilizando redes neuronales convolucionales profundas [19]. En el documento se habla lo complejo que es el diagnóstico temprano de fallas en engranajes incluso con los avances que ha tenido el aprendizaje de máquinas. Esto debido a que la extracción de características es indispensable la participación humana por lo que los resultados obtenidos pueden no ser representativos y estar sesgados de una aplicación a otra. Desarrolla un enfoque de aprendizaje de trasferencia basado en redes neuronales convolucionales profundas con un pequeño conjunto de datos de entrenamiento, lo que lo hace interesante debido a que generalmente los datos de entrenamiento requieren un conjunto sustancial, lo que dificulta su aplicación cuando se tienen datos de entrenamiento limitados. Como conclusión la precisión lograda indica que el enfoque propuesto es viable y robusto, al punto de tener un gran potencial para ser aplicable a otras prácticas de diagnóstico de fallas.

El autor de los datos nos indica que existen 9 tipos de estados, de los cuales van desde un estado sano, pasando por diferentes fallas, como, un diente roto, un diente faltante y distintas rupturas en los dientes dependiendo del tamaño de diente faltante. En total existen 104 datos por cada estado del engranaje haciendo un total de 903 tomas de muestras, mientras que cada muestra esta echa en un total de 3600 segundos a 1000 Hz, obteniendo una muestra cada 3.6 segundos, pudiendo observar de manera visual en la Tabla 2.1 y de manera ilustrativa en la Figura 2.13, las distintas fallas del engranaje que se pueden generar.

Tabla 2.1. Conjunto de datos Engranaje

Estado Engranaje	Muestras
Healthy	104
Missing	104
Crack	104
Spall	104
Chip1a	104
Chip2a	104
Chip3a	104
Chip4a	104
Chip5a	104

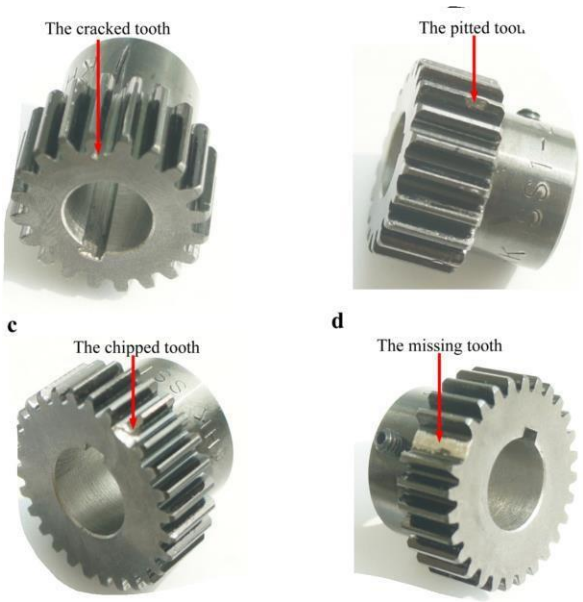


Figura 2.13. Figuras de distintos estados del engranaje. Fuente: [31].

2.4 Procesamiento de datos

Siempre que tenemos una gran cantidad de datos es necesario hacer una extracción de datos, si bien la información es relevante sobre la aparición y evolución de una falla es

necesario realizar distintos procesos para una mejor interpretación como se observa en la Figura 2.14.

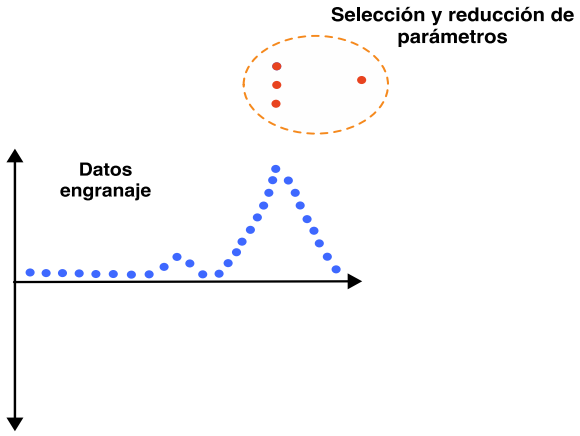


Figura 2.14. Preanálisis de datos. Fuente: Elaboración propia.

Si bien dentro del espectro en el dominio del tiempo tenemos la señal de vibración es posible analizar las longitudes de ondas o la amplitud, esto no nos va a aportar una gran información a la hora del análisis, es por esto por lo que podemos dividir una señal estacionaria en el dominio del tiempo o en el dominio de la frecuencia como se observa en la Figura 2.15.

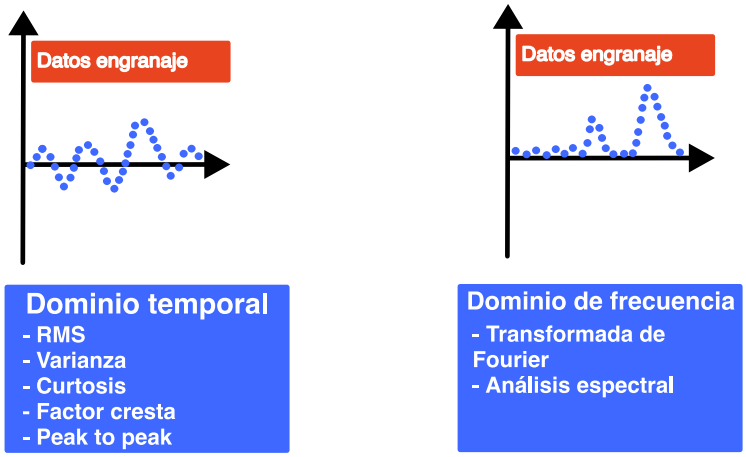


Figura 2.15. Subcategorías de la extracción de datos. Elaboración propia

De aquí se obtienen una serie de parámetros determinantes para la clasificación de fallas según los datos obtenidos, como lo son el RMS, curtosis, factor cresta, entre otros, que

se encuentran en el dominio del tiempo, mientras que, por otro lado, tenemos en el dominio de la frecuencia la transformada de Fourier o el análisis espectral, siendo el primero el que más predomina de una manera visual y que genera más aporte a la hora de un análisis y posterior clasificación.

En el caso de una señal no estacionaria es posible aplicar la transformada corta de Fourier, la cual proviene de la normal, pero se caracteriza por subdividir en N partes la señal y así poder distinguir de mejor manera los distintos cambios que se producen debido a las distintas frecuencias que se le aplican durante el tiempo de muestreo.

En conclusión, en posible analizar las distintas etapas de la señal por medio de diferentes herramientas, sacando el máximo de provecho a los datos entregados, ya sea por medio del dominio de la frecuencia como del dominio del tiempo. Todas son de gran ayuda, y entregan información considerable respecto a la señal en bruto que se obtiene mediante el análisis de vibraciones. Con esto se da el inicio a la selección de parámetros para construir un modelo que por medio de IA pueda detectar las distintas fallas que se presentan en los engranajes o rodamientos.

2.5 Parámetros relevantes

2.5.1 Procesamiento de datos engrane

Al poseer una velocidad constante, las muestras que se realizaron son muy parecidas, por lo que al comparar una o más gráficas en el dominio del tiempo y en el dominio de la frecuencia, se obtendrá será una gráfica muy parecida, pudiendo simplificar la representación de esta por medio de un solo gráfico omitiendo el resto demuestras. Para poder corroborar esta información se expondrá dos gráficas de diferentes muestras para la misma falla, con la finalidad de poder comparar esta teoría.

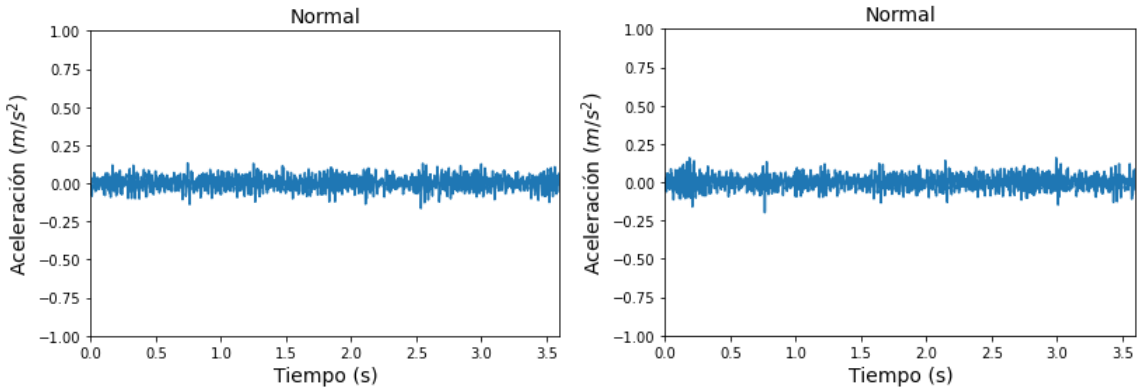


Figura 2.16. Comparación de 2 ensayos de vibraciones en estado normal. Fuente: Elaboración propia.

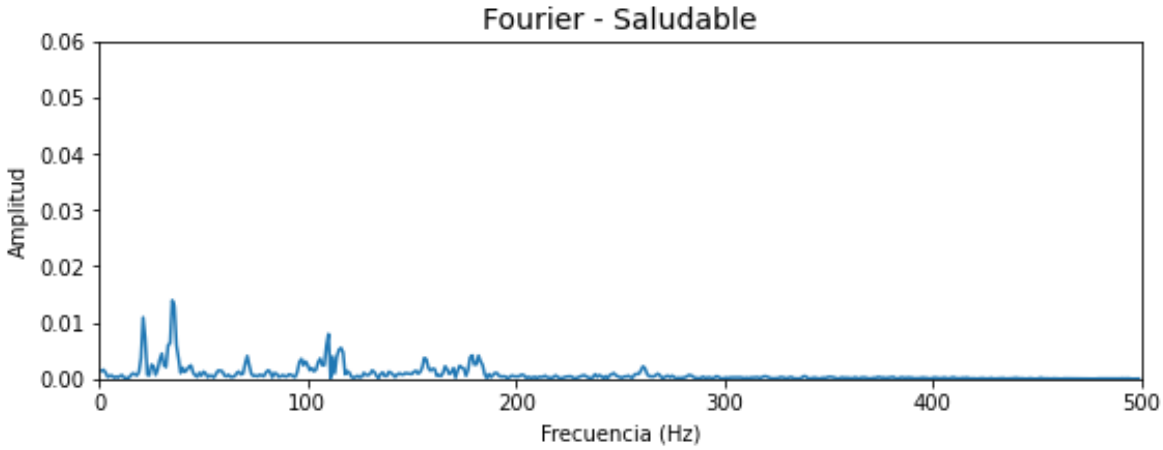


Figura 2.17. Transformada de Fourier para estado sano 1. Fuente: Elaboración propia.

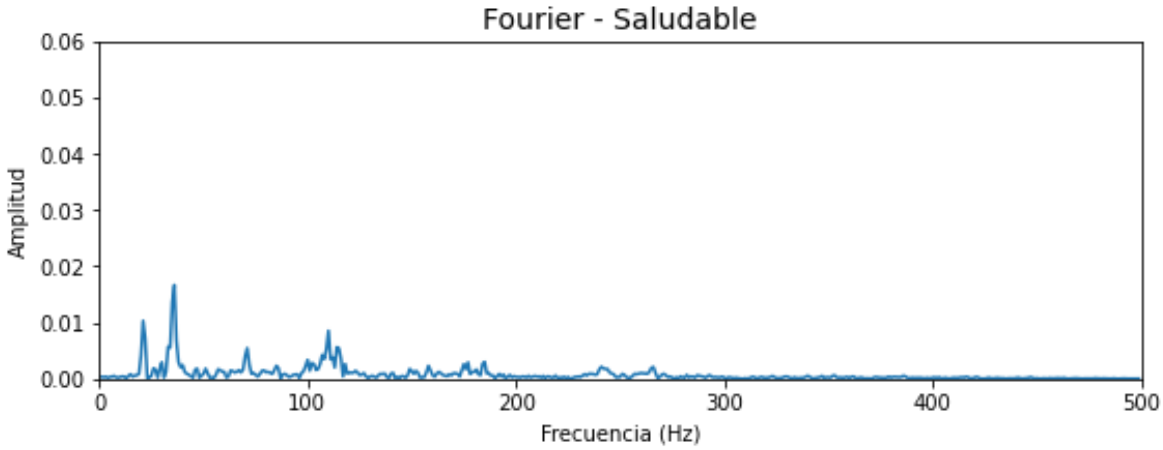


Figura 2.18. Transformada de Fourier para estado sano 2. Fuente: Elaboración propia.

Como conclusión se puede observar que en la Figura 2.16 no existe una gran diferencia dentro de las vibraciones, si se observa una mayor fluctuación en el inicio, pero no es notoria en aspectos numéricos.

Cuando observamos las Figura 2.17 y Figura 2.18, de manera detallada lo que está sucediendo con las vibraciones por medio de la transformada de Fourier, se visualiza un cambio del dominio del tiempo a un dominio predominado en la frecuencia. Esto se realiza debido a que muchas veces el dominio del tiempo crea muchas conjeturas debido a que los cambios que se generan son muy poco perceptibles al ojo humano, es por este motivo que se opta por un cambio de dominio notando una gran diferencia dentro de las ondas obtenidas en la Figura 2.16 versus la transformada de Fourier en la Figura 2.17 y Figura 2.18.

Debido a que contamos con grandes volúmenes de datos, se utilizará de manera excepcional para describir información concreta como el RMS o curtosis, un solo dato de cada estado, debido a que como se pudo comprobar no existe una diferencia notoria en cada estado en las diferentes muestras que se realizaron.

Por otro lado, para la creación del modelo de inteligencia artificial se utilizarán todos los datos, debido a que serán de gran ayuda para entrenar al modelo y así poder diferenciar los distintos fallos que puedan ocurrir.

2.6 Análisis de vibraciones

Los sensores, miden el movimiento relativo del eje, este está compuesto por un transductor que contiene una bobina plana instalada en una punta de resina que es montada en las cajeras de cojinetes para monitorizar el movimiento del eje. Esta bobina es alimentada con una corriente que genera un campo magnético que varía de acuerdo con la proximidad del eje u elemento al cual se requiere medir el comportamiento dinámico.

Principalmente el más utilizado en la industria es el piezoeléctrico que se puede observar en la Figura 2.19, debido a que se utiliza principalmente para monitorear la señal de vibración para evitar fallas en la máquina.

Figura 2.19. Sensor de aceleración piezoeléctrico. Fuente: [32].

2.6.1 Gráficos representativos en dominio del tiempo y la frecuencia para los diferentes estados del engranaje

A continuación, se mostrará para cada estado del engranaje su respectiva gráfica en el dominio del tiempo y en el dominio de la frecuencia, respectivamente con su transformada de Fourier.

2.6.1.1 Estado Sano

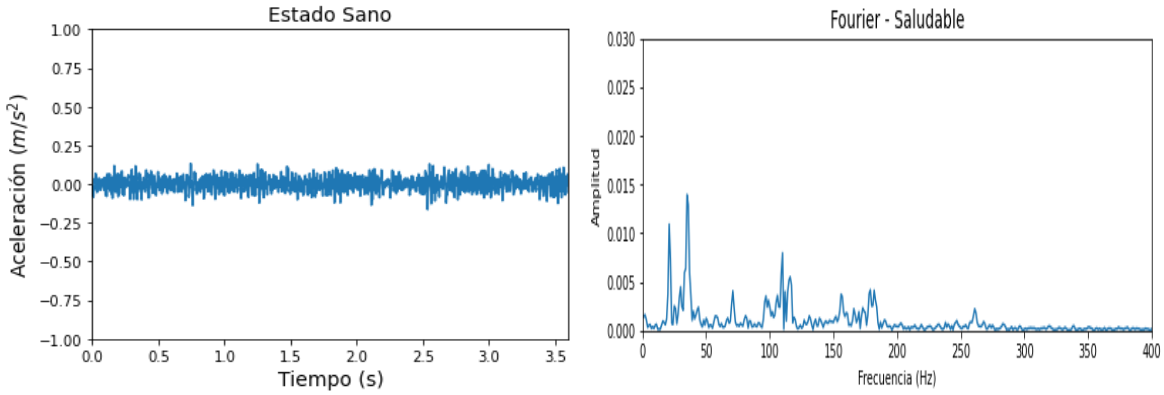


Figura 2.20. Estado sano, dominio tiempo y frecuencia. Fuente: Elaboración propia.

2.6.1.2 Diente Faltante

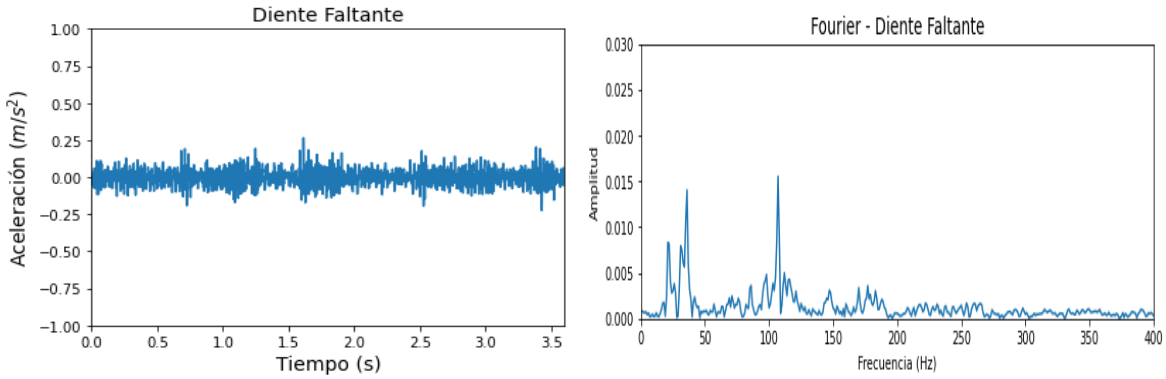


Figura 2.21. Diente faltante, dominio tiempo y frecuencia. Fuente: Elaboración propia.

2.6.1.3 Diente Roto

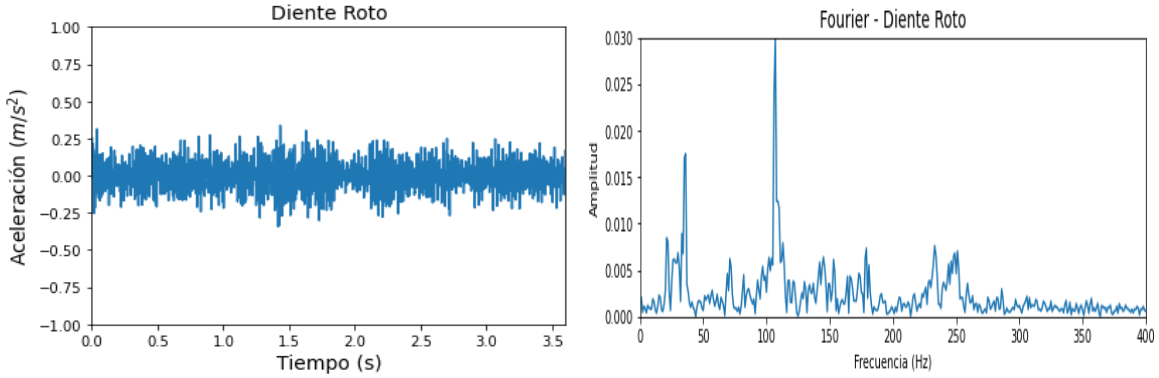


Figura 2.22. Diente roto, dominio tiempo y frecuencia. Fuente: Elaboración propia.

2.6.1.4 Diente Picado

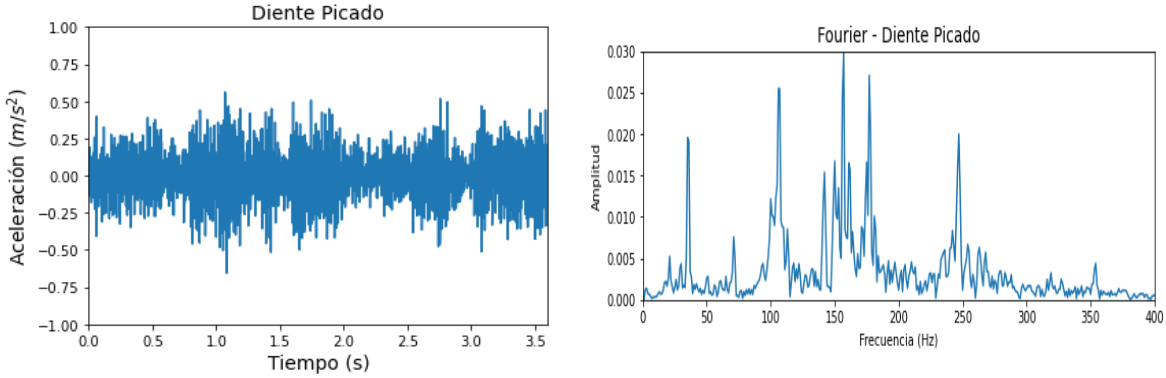


Figura 2.23. Diente picado, dominio tiempo y frecuencia. Fuente: Elaboración propia.

2.6.1.5 Diente Astillado 1

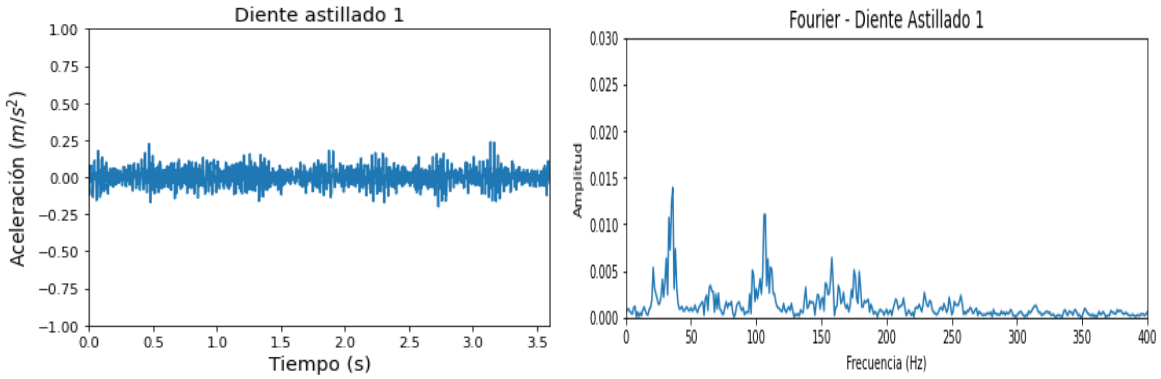


Figura 2.24. Diente astillado 1, dominio tiempo y frecuencia. Fuente: Elaboración propia.

2.6.1.6 Diente Astillado 2

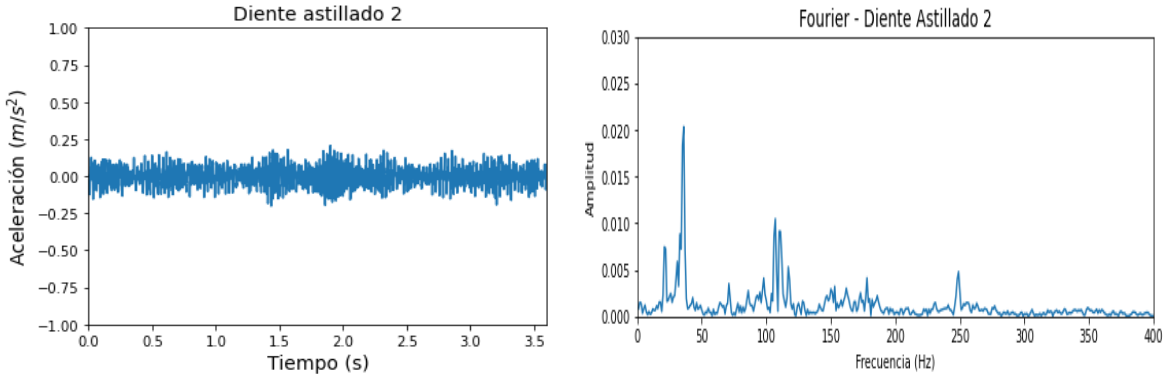


Figura 2.25. Diente astillado 2, dominio tiempo y frecuencia. Fuente: Elaboración propia.

2.6.1.7 Diente Astillado 3

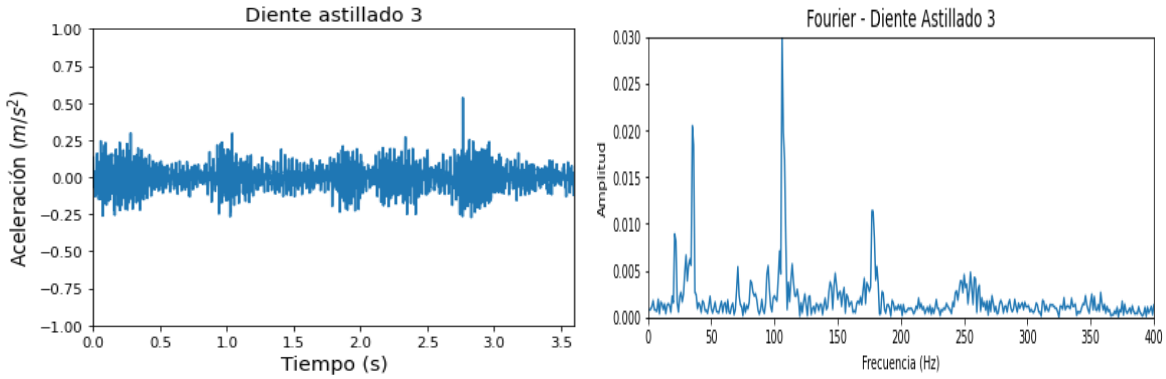


Figura 2.26. Diente astillado 3, dominio tiempo y frecuencia. Fuente: Elaboración propia.

2.6.1.8 Diente Astillado 4

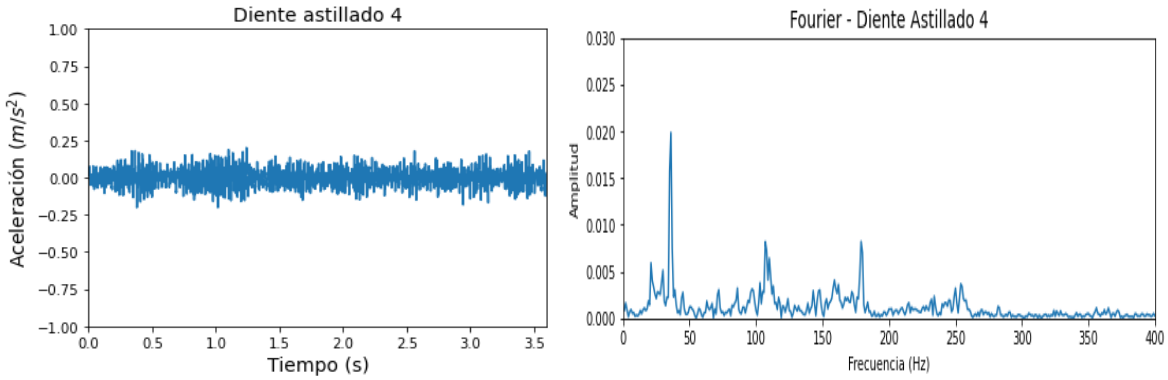


Figura 2.27. Diente astillado 4, dominio tiempo y frecuencia. Fuente: Elaboración propia.

2.6.1.9 Diente Astillado 5

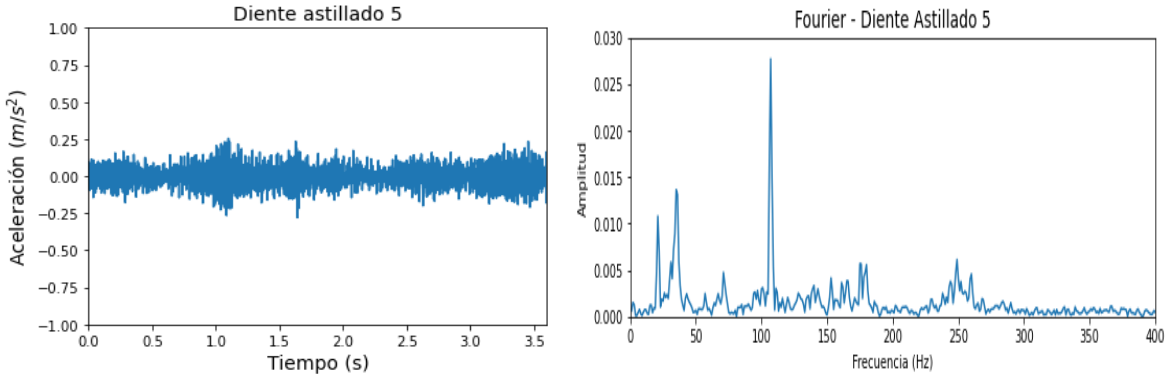


Figura 2.28. Diente astillado 5, dominio tiempo y frecuencia. Fuente: Elaboración propia.

2.6.2 Parámetros Estadísticos

2.6.2.1 Root Mean Square (RMS)

El valor RMS describe la energía contenida dentro de la señal. Se utiliza para evaluar el estado general de los componentes. Es por esto por lo que no es muy sensible a fallo incipiente, sin embargo, se utiliza para realizar un seguimiento de la progresión del fallo general de la máquina.

$$S_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(s_i)^2}$$

Donde:

N es el número de puntos de la señal S.

S_i es el punto i-ésimo en la evolución temporal de la señal S.

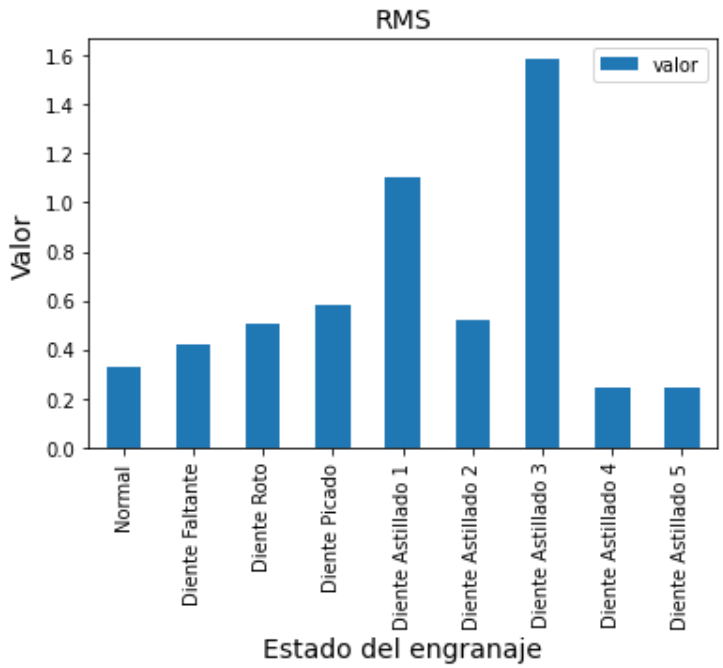


Figura 2.29. Valor RMS engranajes. Fuente: Elaboración propia.

2.6.2.2 Peak

El valor peak es la máxima amplitud de la señal en cierto intervalo de tiempo.

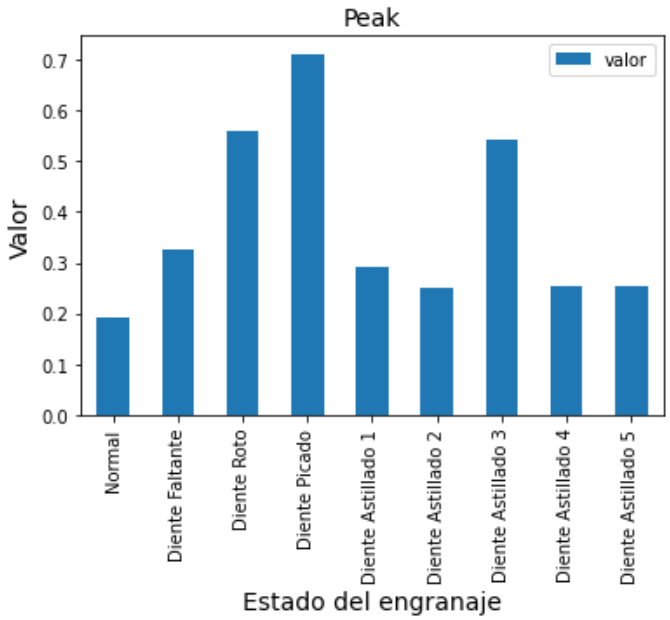


Figura 2.30. Valor Peak engranajes. Fuente: Elaboración propia.

2.6.2.3 Peak to Peak

Corresponde a la distancia entre la amplitud máxima y mínima de una señal.

$$S_{peak-peack} = S_{max} - S_{min}$$

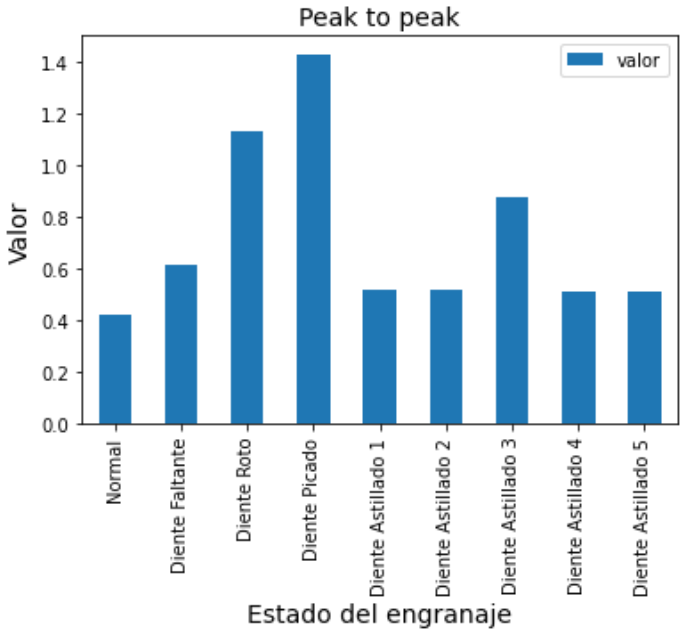


Figura 2.31. Valor Peak to Peak engranajes. Fuente: Elaboración propia. Fuente: Elaboración propia.

2.6.2.4 Valor medio

El valor medio es el primer momento estadístico. Con este valor se identifica el punto medio de la señal muestreada, tiene como principal defecto la sensibilidad a interferencias o valores atópicos que puedan haber dentro de la señal.

El valor medio se define como la suma de todas las amplitudes de la señal S, entre la longitud N, expresada algebraicamente de la siguiente manera:

$$\bar{S} = \frac{1}{N}\sum_{i=1}^{N} S_i$$

Donde:

N es el número de puntos en la historia de la señal S.

S_i es el punto i-ésimo en la evolución temporal de la señal S.

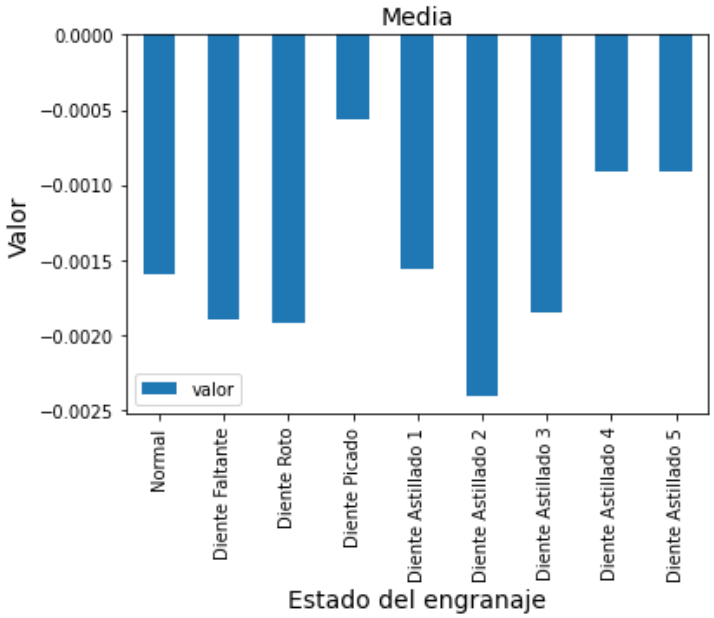


Figura 2.32. Valor de la media aritmética engranajes. Fuente: Elaboración propia.

2.6.2.5 Varianza

La varianza da cuenta de la distancia entre cada punto y el valor medio en una serie de datos. Para una señal S, la varianza se calcula como:

$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(S_i - \bar{S})^2$$

Donde:

N es el número de puntos en la historia de la señal S.

S_i es el punto i-ésimo en la evolución temporal de la señal S.

S es el valor promedio de los Si.

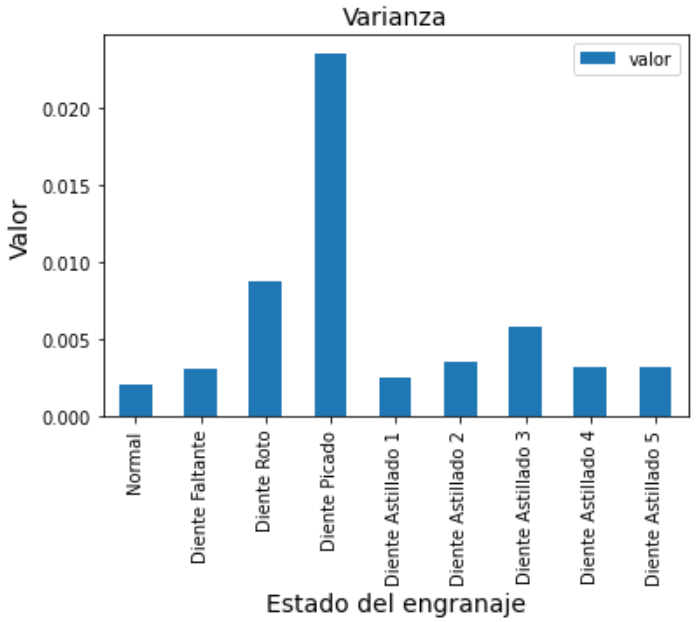


Figura 2.33. Varianza engranajes. Fuente: Elaboración propia.

2.6.2.6 Curtosis

Curtosis es un parámetro estadístico que proporciona información de cuan agudo son en promedio los peaks de una señal. Sirve para medir los picos o valles relativos de a distribución comparada a una distribución normal.

Curtosis mayores a 3 implican peaks más puntiagudos y Curtosis menores a 3 implican peaks más planos.

$$Curtosis = N\frac{\sum_{i=1}^{N}(S_i - \bar{S})^4}{\left(\sum_{i=1}^{N}(S_i - \bar{S})^2\right)^2}$$

Donde:

N es el número de puntos en la historia de la señal S.

S_i es el punto i-ésimo en la evolución temporal de la señal S.

S es el valor promedio de los Si.

Curtosis proporciona una medida del tamaño de las colas de la distribución y se utiliza como un indicador de los peaks principales en un conjunto de datos. Si un engranaje se desgasta y se rompe, esta característica debe señalar un error debido al mayor nivel de vibración.

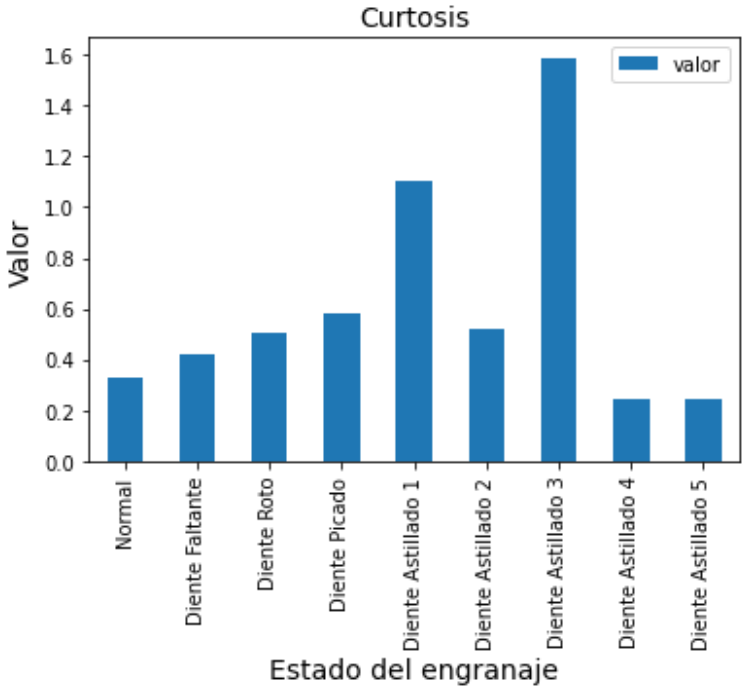


Figura 2.34. Curtosis engranajes. Fuente: Elaboración propia.

2.6.2.7 Factor Cresta

El factor cresta es la diferencia que existe entre el peak de una señal respecto a su valor RMS. Normalmente, fluctúa entre 2 y 6, un valor mayor a lo mencionado podría presentar un posible fallo en la máquina. Los rodamientos poseen un factor cresta estimado entre 2.5 y 3.5.

El factor cresta se posicionó entre uno de los mejores al ser un excelente identificador de los primeros signos de fallos, debido a la diferencia significativa entre el valor pico de corta duración y el valor RMS, haciendo que el factor cresta aumentara considerablemente.

Sin embargo, los grandes daños en los rodamientos hacían que el valor RMS en el dominio del tiempo se incrementara de manera significativa. Como el RMS es el dominador

de la proporción del factor cresta, causaba que el valor cresta se redujera, provocando un factor de cresta menor, pero con un aumento irreparable en los rodamientos.

El análisis de este bajo valor en el factor de cresta daba como resultado no saber a ciencia cierta si el rodamiento estaba en buen estado o si en realidad presentaba daños.

El Factor cresta puede ser utilizado para indicar fallos en una etapa temprana. Esta función se utiliza para detectar cambios en el patrón de señal debido a las fuentes de vibración impulsivos tales como la rotura de dientes en un engrana.

$$CF = \frac{S_{peak}}{S_{RMS}}$$

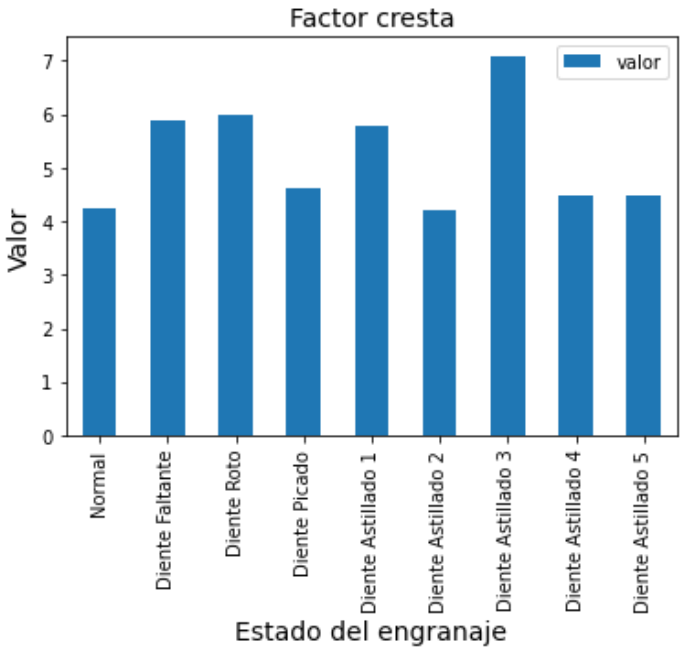


Figura 2.35. Factor cresta engranajes. Fuente: Elaboración propia.

2.6.2.8 Asimetría

La asimetría es el tercer momento estadístico normalizado, puede ser tan efectivo como la curtosis si los datos son bien identificados desde un principio. Su principal ventaja es su menor susceptibilidad a vibraciones que pueden perturbar el resultado.

La sensibilidad el cambio de cargas y velocidad es menor a la de curtosis, pudiendo expresar de manera algebraica de la siguiente manera [9]:

$$Asimetría = N\frac{\sum_{i=1}^{N}(S_i - \bar{S})^3}{\sum_{i=1}^{N}(S_i - \bar{S})^3}$$

Donde:

N es el número de puntos en la historia de la señal S.

S_i es el punto i-ésimo en la evolución temporal de la señal S.

S es el valor promedio de los Si.

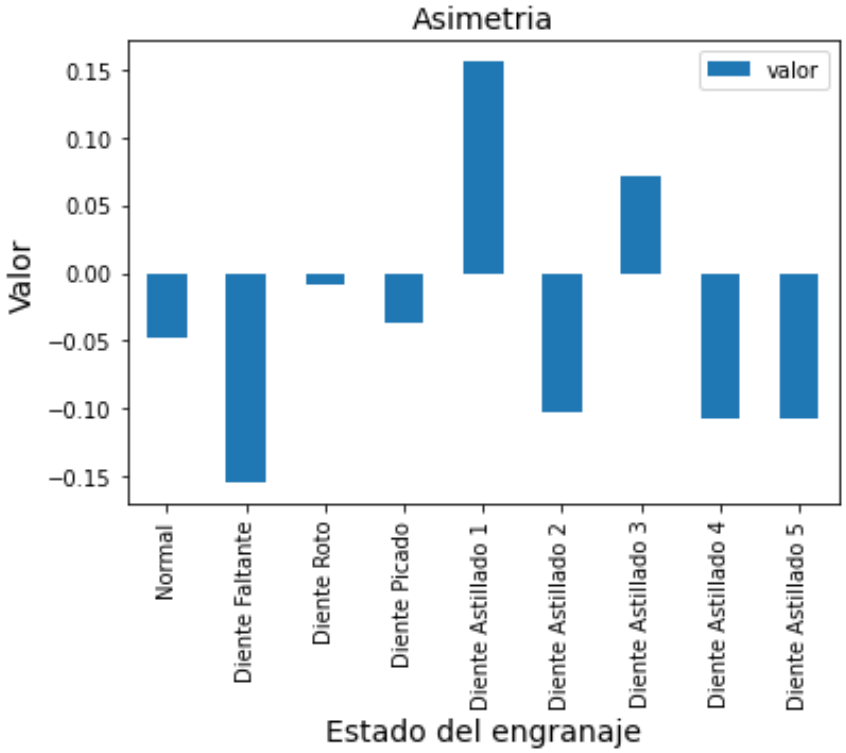


Figura 2.36. Asimetría engranajes. Fuente: Elaboración propia.

2.6.2.9 Conclusiones

A continuación, se expondrán de manera detallada los elementos más importantes del análisis descriptivo y la importancia de estos valores dentro de la detección de fallas en los

engranajes para una detección temprana por medio de análisis estadístico y como estos serán de ayuda para ratificar el modelo de IA que se determinara con posterioridad.

Como primer punto se analizará la Figura 2.29, a simple vista se puede observar el valor del estado sano del engranaje, el cual es mucho más alto que el de los demás, esto es debido a que contiene una mayor energía, esto se puede explicar debido a que este se encuentra en su estado sano, en comparación con los demás estados, los cuales todos tienen distintos problemas en los dientes de los engranajes por lo que hace que puedan generar una menor energía a la hora de medir las señales. Además, en su conjunto se puede observar en la Figura 2.20, el dominio de la frecuencia, la transformada de Fourier, en la cual nos indica en su primera instancia los peak de la corona y piñón alrededor de los 50 Hz y en los 100 Hz podemos observar otro peak que corresponde al GMF (Frecuencia de engranaje), su amplitud corresponde significativamente a la carga en el momento de lectura en el que se encuentra sometido el engranaje. Además de agregar los estados en 150 Hz y 250 Hz que corresponden a las mismas frecuencias de engranaje para un estado sano.

Luego tenemos los valores peak y peak to peak en la Figura 2.30 y Figura 2.31 donde principalmente nos indica que el diente picado o pitted tooth contiene ondas muy altas y bajas, por lo que a simple vista este genera un estado mayor que el de los demás en conjunto con el diente roto y el diente astillado 3. Respecto a los demás estados existe poca variación por lo que a simple vista se asemejan al valor normal. Con respecto a los estados en el dominio del tiempo y del estado podemos observar que en la Figura 2.22, Figura 2.23 y Figura 2.26 que corresponden al diente roto, picado y astillado 3 respectivamente, en el dominio de la frecuencia podemos observar a simple vista una mayor energía, lo que genera esos peak en las distintas muestras. En el dominio del tiempo podemos observar que en las 3 figuras a los 100 Hz la amplitud es mucho mayor que en el estado normal, por lo que el GMF presenta una mayor frecuencia, lo que explica que los valores de peak y peak to peak sean tan altos.

Una característica que no se ve en los demás estados y se presenta mucho en estas 3 figuras es la frecuencia de resonancia que se puede observar claramente en el dominio de la frecuencia, que se presenta alrededor de los 70 Hz, lo que nos está indicando una falla muy grave a nivel estructural del engranaje.

El valor medio y la varianza las cuales las podemos observar en la Figura 2.32 y Figura 2.33 van una correlacionada con respecto a la otra. Por un lado, se observa una tendencia general hacia los números negativos en el valor medio, esto debido a una desaceleración en la toma de datos con respecto a la vibración, dando como resultado una media negativa en todas las fallas. Por otro lado, la varianza que está relacionada con el valor medio denota una tendencia nuevamente en el diente picado, haciendo más fácil la identificación de este problema. Esto debido a que existe una gran diferencia en la variabilidad del conjunto del diente picado con respecto a la media. Si estos datos son comparados con la varianza con los datos sanos, se nota que estos no varían mucho de la media, concluyendo que cualquier dato muy alejado podría indicar una inminente falla en el engrane.

Curtosis, por otro lado, es un factor relevante en la toma de decisiones debido a que es uno de los más utilizados por su gran certeza dentro de los rodamientos y engranajes. En la Figura 2.34, podemos observar una tendencia que no se ha observado con anterioridad en los anteriores parámetros estadísticos, donde se observan dos estados del engranaje que no eran muy ponderados, que son el diente astillado 1 y el diente astillado 3. Como bien es sabido curtosis no da ninguna indicación acerca del diagnóstico del equipo, pero debido a que ya contamos con las distintas fallas que puede presentar el engranaje, es fácil identificar los valores atípicos que se desvían del estado sano. Este indicador es relevante debido a que si observamos la Figura 2.24, podemos identificar que en el dominio del tiempo y en el dominio de la frecuencia estos son muy parecidos al estado sano, lo que podemos concluir que genera un gran valor a la hora de la toma de decisiones por su gran aporte en estados muy parecidos al sano.

El factor cresta que se puede observar en la Figura 2.35, no aporta mucho en la decisión del posible fallo debido a que como fue mencionado, este al aumentar el valor del RMS causaba incertidumbres y esto se puede notar en la figura ya mencionada y como todos los estados del engranaje poseen un valor muy parecido.

La asimetría que se puede ver en la Figura 2.36 es un gran aporte debido a que como se observa en la imagen divide a los estados en 2 partes desde su estado sano pudiendo identificar fácilmente distintos síntomas debido a sus valores tan alejados a un estado sano.

3. Algoritmos de clasificación.

3.1 Planteamiento y desarrollo modelo de inteligencia artificial para engranajes.

3.1.1 Introducción

‘‘El aprendizaje de máquinas (machine learning) es la rama de la inteligencia artificial que tiene como objetivo desarrollar técnicas que permitan a las computadoras aprender’’[5]. De forma más concreta se trata de crear algoritmos capaces de generalizar comportamientos y reconocer patrones a partir de información suministrada en forma de ejemplos.

En este capítulo se desarrollará una serie de pasos para cumplir las condiciones establecidas y así el modelo pueda identificar por medio de patrones las diferentes fallas que pueda presentar una máquina u objeto en concreto, como lo es un engranaje en este caso. Si bien se comentó que es necesaria la utilización de ejemplos esto no aplica para todos los modelos del machine learning como se puede observar en la Figura 3.1.

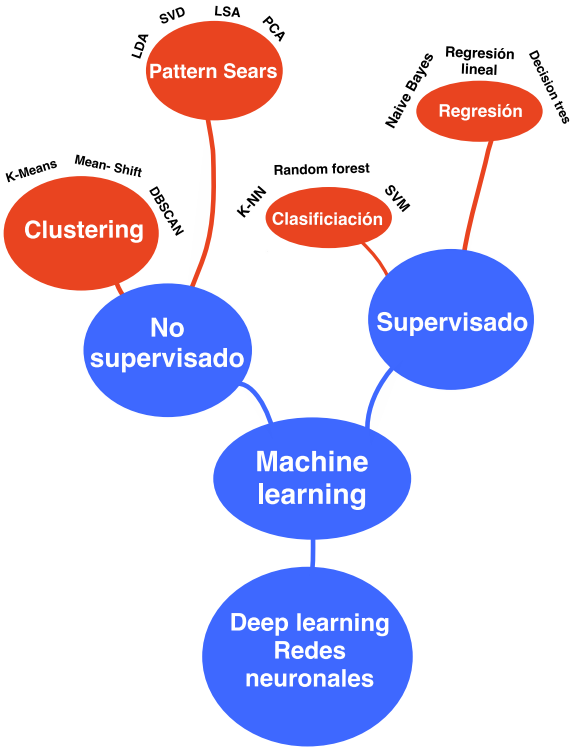


Figura 3.1. Mapa conceptual machine learning. Fuente: Elaboración propia.

Para el caso en concreto y los objetivos planteados se utilizará un método de machine learning clásico, en donde se le darán los datos de aprendizaje, por lo que nos iremos por la rama supervisada y se desarrollara un modelo de clasificación en donde se compararan distintos modelos para hacer una comparación de cuál se adapta mejor a los datos entregados, con un enfoque en el área de mantenimiento.

3.1.2 Monitoreo de la condición

Para poder monitorear el estado de salud de un equipo es posible determinar 3 niveles principales para determinar el verdadero estado en el que se encuentra.

3.1.2.1 Detección

Es el estado más básico de los tres, pero es aquel que se caracteriza por ser el más directo, y el cual trata de poder identificar el modo de operación del sistema y su estado, es decir, detectar la falla con los elementos que uno pueda identificar a simple vista, como lo puede ser desbalance muy notorio, sonidos extraños, entre otros. Pudiendo identificar de manera rápida comportamientos anómalos en el equipo.

3.1.2.2 Diagnóstico

Luego de identificar distintas anomalías presentes en el equipo, es necesario profundizar que tipo de anomalía es, en que parte se está presentando y que características específicas caracterizan a esta anomalía, pudiendo categorizar dentro de ciertos criterios que tan severa es y cuanto afecta en el proceso en el que está involucrada. Aquí es donde se aplican los distintos modelos del mantenimiento, como lo es las tablas de criticidad que son muy importantes a la hora de la evaluación de los equipos y los problemas que estos puedan tener.

3.1.2.3 Pronóstico

Para poder realizar el pronóstico es necesario contar con una base, esta debe contener la mayor cantidad de fallos que ha tenido el equipo, esta característica es muy necesaria si tenemos que enseñarle a detectar la próxima falla del equipo.

Un factor que es muy difícil de controlar son las condiciones de operación en las que se encuentra el equipo, si bien podemos obtener un modelo de predicción para un equipo, no necesariamente este pueda ser siempre igual ante diferentes operaciones, por lo que hay que tenerlo muy en cuenta a la hora de poder implementar un modelo de mantenimiento predictivo basado en inteligencia artificial.

Como bien es sabido un equipo al entrar en constantes fallos y detenciones su vida útil empieza a decrecer, es por esto por lo que además de poder identificar una posible falla, es de suma importancia para el área de mantenimiento saber cuál es el tiempo de vida útil remanente del equipo, para poder anteponerse a estos cambios que podrían costar millones a nivel de producción.

3.1.3 Detección de anomalías

La detección de anomalías en el área del aprendizaje de máquinas es una rama en la que se caracteriza debido a que un algoritmo detecta a base de patrones definidos datos fuera de lo común, clasificándolas y advirtiendo al usuario que están ocurriendo procesos anormales.

Para poder que nuestro algoritmo detecte estos patrones irregulares es necesario poder entrenarlo, esto se hace entregándole datos en condiciones normales del equipo, realizando un entrenamiento, garantizando una detección temprana de una posible falla.

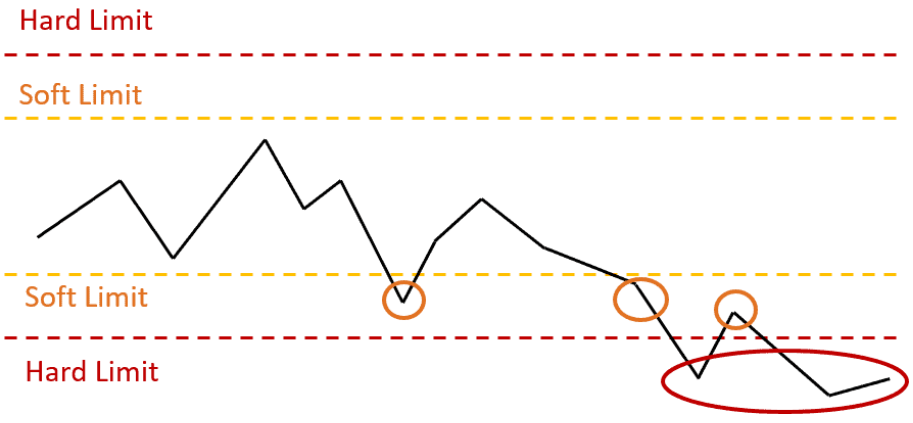


Figura 3.2. Gráfico detección de anomalías. Fuente: [33].

El modelo ya entrenado, se le entregan datos nuevos y gracias al algoritmo este nos

indicará los diferentes puntos que se han detectado y que tan grave puede ser. Como se puede observar en la Figura 3.2, donde los círculos en rojos corresponden a eventos fuera de lo común con respecto a los datos de entrenamientos que nosotros le entregamos. Es por esto por lo que es muy importante al momento de entrenar al algoritmo no sobrecargarlo de datos o solo entregar datos irrelevantes, ya que esta será la base para que a futuro se puedan detectar y clasificar los diferentes eventos que puedan hacer fallar la máquina costando horas de producción muy valiosas para una empresa.

Cuando los datos que tenemos no están etiquetados, es decir, no se cuentan con los diferentes problemas del equipo, se pueden implementar metodologías para la detección de anomalías, implementando análisis de componentes principales, la cual es una estrategia de análisis no supervisada para la detección de anomalías. ‘‘El método PCA (Análisis de componentes principales) crea una función que mapea la posición que ocupa cada observación en el espacio original con el que ocupa en el nuevo espacio generado’’[33]. El mapeo funciona bidireccionalmente, por lo que aquellas observaciones que hayan sido bien proyectadas podrán volver a la posición que ocupaban en el espacio original. ‘‘Este método principalmente se basa en simplificar la complejidad de espacios con múltiples dimensiones a la vez que conserva su información.’’[34]. En otras palabras, segmenta la información dada y crea una clasificación según los datos obtenidos y separa los datos que no pertenezcan a un subconjunto.

3.1.4 KPI

Cuando es necesario evaluar el rendimiento de la clasificación un elemento es importante elegir aquellos factores y variables que son más relevantes, en el caso en particular de los algoritmos a clasificar, es necesario obtener la precisión, el recall, F1-score y el accuracy, los cuales son elementos fundamentales en los modelos de clasificación debido a la información que nos entregan.

3.1.4.1 Precisión

La precisión hace referencia a la dispersión del conjunto de valores obtenidos a partir de mediciones repetidas de una magnitud. Si la dispersión es menor, lo que arrojará será una precisión mayor.

Esta se representa por la proporción de verdaderos positivos dividido entre todos los resultados positivos, agrupando los verdaderos positivos, como los falsos positivos. En otras palabras, corresponde al porcentaje de casos positivos detectados.

Se puede representar por medio de la siguiente ecuación.

$$\frac{V_P}{V_P + F_P}$$

Donde:

V_P: Verdadero positivo.

F_P: Falso positivo.

3.1.4.2 Recall

El recall es una métrica capaz de dar información sobre la cantidad de casos positivos que fueron correctamente identificados por el algoritmo.

Se puede representar mediante la siguiente ecuación.

$$\frac{V_P}{V_P + F_N}$$

Donde:

V_P: Verdadero positivo.

F_P: Falso positivo.

3.1.4.3 F1- score

Es una métrica que resume la variable de la precisión y el recall en una sola métrica. Muy utilizado en la determinación de los diferentes KPI, debido a que hace más fácil poder comparar el rendimiento de la precisión y el recall.

Se puede representar por medio de la siguiente ecuación.

$$F1 - score = 2 * \frac{Precisión * recall}{recisión + recall}$$

3.1.4.4 Accuracy

La exactitud o Acurracy corresponde a lo cerca que se encuentra el resultado de una medición del valor verdadero. Estadísticamente, hace relación con el sesgo de una estimación. Se puede representar como la proporción de resultados verdaderos, estos pueden ser verdaderos positivos como negativos, dividido entre el total de datos, integrando los verdaderos positivos, falsos positivos, verdaderos negativos y por último los falsos negativos.

En otras palabras, la exactitud corresponde a la cantidad de predicciones positivas que fueron correctas. Obteniendo su representación por medio de la siguiente ecuación.

$$\frac{V_P + V_N}{V_P + F_P + F_N + V_N}$$

Donde:

V_P: Verdadero positivo.

F_P: Falso positivo.

F_N: Falso negativo.

V_N: Verdadero negativo.

3.1.5 Algoritmos de clasificación

El propósito de las diferentes técnicas de clasificación se centra en agrupar los datos de la estructura requerida con base en características comunes. La clasificación de estas características permite determinar aquellos tipos o grupos de pertenencia que se desconoce. Los distintos métodos de clasificación se definen como modelos que producen resultados diferentes unos con otros, debido a que estos métodos incluyen un análisis y clasificación basado en los ejemplos de los conjuntos de datos que se le presentan para entrenarlos.

La clasificación se basa en un algoritmo de aprendizaje autónomo. El propósito del aprendizaje es crear un modelo de clasificación. Estos datos se seleccionan como datos de entrenamiento que son recogidos de una base de datos con los datos históricos del equipo, el

algoritmo opera tanto en el conjunto de datos de entrenamiento como en el de testeo, en el cual no se conoce los datos de clasificación, lo que se comparan para determinar a qué grupo pertenecen los datos de testeo en referencia a los datos de entrenamiento.

Existen diferentes métodos de clasificación entre los cuales podemos encontrar SVM (Máquina de soporte de vectores), decisión tress (Árbol de decisión), K-nearest neighbor (K-vecinos más cercanos), random forest, random forest (bosque aleatorio), entre otros. Todos tienen maneras de implementarse distintas, pero cumplen una función en específico que es la de clasificación.

A continuación, se detallará las principales herramientas de clasificación ya mencionadas y se detallarán en cada una de ellas para el uso óptimo en el área de mantenimiento y la clasificación de los diferentes modos de falla de un engranaje.

3.1.6 K-nearest neighbor (K- vecinos más cercanos)

Se utilizará la librería de sklearn.neigbors la cual proporciona una funcionalidad para métodos de aprendizaje de máquinas basados en vecinos no supervisados, como de vecinos supervisados.

3.1.6.1 Definición del modelo

El principio detrás del método es encontrar un número predefinido de muestras de entrenamiento más cercanas en distancia al nuevo punto y predecir la etiqueta a partir de ellas. El número de muestras que se utilizará puede ser una constante previamente definida por el usuario o una variación según la densidad local de puntos, en otras palabras, un radio de puntos. La distancia que uno estima por lo general puede ser cualquier medida métrica, aunque generalmente se utiliza una distancia denominada euclidiana que es utilizada en aprendizaje de máquinas, pudiendo también utilizar métricas como chebyshev, manhattan o kullback-leibler.

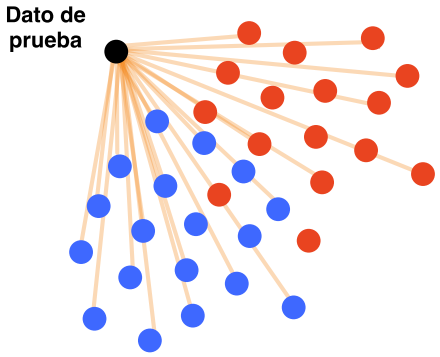


Figura 3.3. Medición de la distancia entre el dato de testeo y las muestras. Fuente: Elaboración propia.

Como se puede observar en la Figura 3.3, el dato de testeo, para poder clasificarlo, se mide la distancia entre él y sus compañeros más cercanos. Los vecinos más cercanos sirven para estimar la función de densidad $F(X/C_i)$ de las predictoras X por cada clase C_i. En otras palabras, este método estima un valor de la función de densidad de probabilidad que ante un elemento X pertenezca a una clase C_i a partir de los datos proporcionados por el usuario. Los datos obtenidos de las distancias se ordenan de la distancia más cercana a la más lejana.

El tercer paso, luego de medir las distancias y ordenarlas es estimar el valor de K, que será el rango de puntos más cercanos al punto de testeo. En el caso concreto de la Figura 3.4, se estima un K=5.

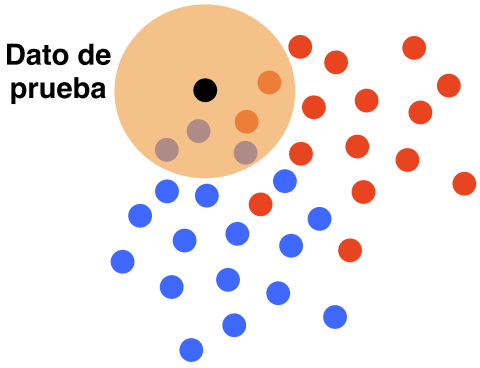


Figura 3.4. Estimación de K vecinos. Fuente: Elaboración propia.

El siguiente paso es hacer un recuento de clases, en el caso en concreto de la Figura 3.4, se puede observar que hay 5 clases rojas y 2 azules, por lo que ese dato de testeo se clasificará como una clase roja.

Este algoritmo se realiza para todos los datos de testeo que se encuentren disponible y cada uno de ellos será clasificado democráticamente por la mayoría de las clases, ya sea roja o azul.

3.1.6.2 Parámetros

K-nearest neighbor consta de 8 parámetros, los cuales cada uno cumple una función específica dentro del algoritmo de clasificación. Estos parámetros tienen la finalidad de modificar el código dependiendo de los atributos que se anden buscando.

Parámetros:

- n_neighbors: Int, por defecto = 5.
  Este parámetro se utiliza para determinar la cantidad de vecinos permitidos en el radio de estimación. Como se puede observar en la Figura 3.4.
- weights: (‘‘uniform’’, ‘‘distance’’), predeterminado = ‘‘uniforme’’
  El peso se utiliza para determinar la ponderación de cada vecindario. Si es ‘‘uniforme’’, todos ponderan el mismo resultado, mientras que por ‘‘distancia’’, la ponderación se realiza por el inverso de su distancia. Mientras más cercanos se encuentren los vecinos, recibirán una mayor ponderación. Esto se conlleva una gran relevancia a la hora de clasificar debido a que mientras más cercano se encuentre a un vecino el cual ya se sabe a qué pertenece, mayor será la probabilidad de que pertenezca a ese fallo. Por otro lado, si la ponderación se hace igualitaria para todos los vecinos, existirá una confusión a la hora de clasificar las fallas.
- Algorithm: (‘‘auto’’, ‘‘ball tree’’, ‘‘kd tree’’, ‘‘brute’’), default = ‘‘auto’’.
  ‘‘Ball tree’’ y ‘‘kd tree ’’, son algoritmos de árbol utilizados para la división espacial de puntos de datos. En otras palabras, se utilizan para estructurar en un espacio multidimensional.
  ‘‘Ball tree’’ [35] es contemplado como un árbol métrico, esto quiere decir que se organiza y estructura los puntos de datos teniendo en cuenta el espacio métrico en el

que se encuentran los puntos. El algoritmo divide los puntos de datos en dos grupos. Cada grupo está rodeado por un círculo (2D) y una esfera (3D). Cada grupo representa un nodo del árbol. Primero se establece el centroide de la nube de puntos de datos. El punto con la distancia máxima al centroide se selecciona como el centro del primer grupo, y el punto más alejado del centro del primer grupo es el punto central del segundo grupo. Es así como se clasifican los puntos asignados al grupo 1 o 2. El proceso de dividir lo puntos de datos en dos grupos y una esfera se repite dentro de cada grupo hasta que se alcanza una profundidad definida.

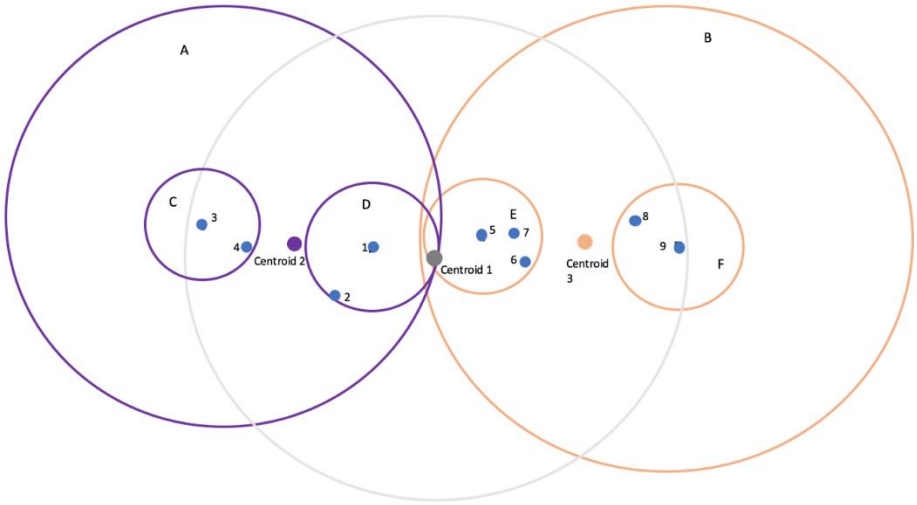


Figura 3.5. Visualización del Algoritmo ball tree. Fuente: [36].

Por otro lado ‘‘Kd tree’’ [35] divide los puntos de datos en nodos de dos conjuntos, el criterio utilizado para la división de los nodos suele ser la mediana haciendo este procedimiento varias veces hasta llegar a un consenso generando otro árbol independiente al principal de k-nearest neighbor.

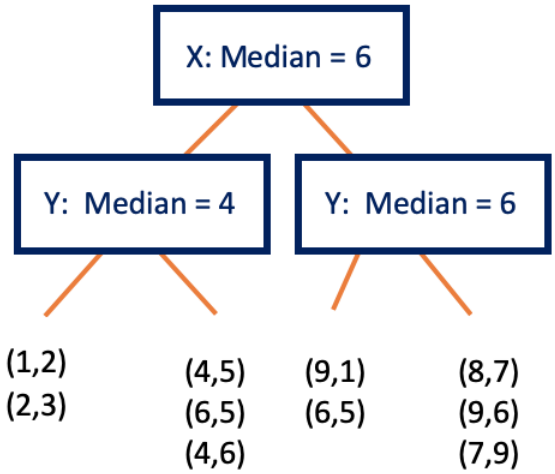


Figura 3.6. Árbol Kd tree. Fuente: [36].

‘‘Bruto’’, utilizará una búsqueda de fuerza bruta.

‘‘Auto’’, intentará decidir el algoritmo más apropiado en función de los valores pasados cuando se aplicó la librería .fit.

- Leaf_size: int , por efecto = 30.
  Este parámetro determina el tamaño de hoja que se utilizara en ‘‘Ball tree’’ o ‘‘KD tree’’, Es un parámetro el cual puede inferir en la velocidad de construcción de los algoritmos mencionados. Un valor optimo dependerá exclusivamente del problema. Por normativa general cuanto más grande sea ‘’leaf_size’’, más cercanos son los vecinos que elige el algoritmo, lo cual no representa una mayor precisión.
- P: int, por defecto = 2.
  Parámetro de potencia para la métrica de minkowski. Cuando p = 1, equivale a usar manhattan_distance y para p = 2, euclidean_distance. La distancia de Manhattan es una métrica de distancia entre dos puntos en un espacio vectorial de N dimensiones. Es la suma de las longitudes de las proyecciones del segmento de línea entre los puntos en los ejes de coordenadas. En otras palabras, es la suma de la diferencia absoluta entre las medidas en todas las direcciones de dos puntos.

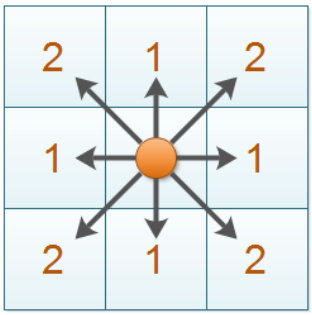


Figura 3.7. Concepto distancia de manhattan. Fuente: [37].

La distancia euclidiana [38] es un número positivo que indica la separación que tienen dos puntos en un espacio donde se cumplen los axiomas y teoremas de la geométrica de Euclides.

- Metric: Str o callable, predeterminado = ‘‘minkowski’’.

  Es aquella métrica que se utilizará para la creación del árbol. Por defecto se utiliza ‘‘minkowski’’, y va relacionada con el parámetro P. Con p = 2 es equivalente a la métrica euclidiana estándar.

- Metric_params: dict, predeterminado = Ninguno.

  Argumentos de palabras clave adicionales para la función métrica.

- N_jobs: int, predeterminado = Ninguno.

  Es aquel parámetro el cual hace referencia al número de trabajos paralelos que se ejecutarán para la búsqueda de vecinos. Si N_jobs = 1 solo se ejecutará un solo trabajo y no se iniciará ningún proceso en paralelo, siendo útil para procesos que requieran depuración. Por otro lado, si N_jobs = -1 significa usar todos los procesadores que tenga disponible el computador exigiéndolo al máximo, pudiendo traer varias consecuencias. Si se deja como N_jobs = -2 significa que se utilizarán todas menos un procesador, cumpliendo la siguiente ecuación: (n_cpus + 1 + n_jobs).

3.1.6.3 Adquisición y carga de datos

Para la adquisición de datos es necesario contar con una cantidad razonable de muestras que sean útiles. Para el engranaje los datos se obtuvieron por medio de una página

web en donde distintas personas suben contenido de distintas áreas para que sean utilizadas en proyectos científicos u otros fines, debido a que son de libre acceso.

Los datos obtenidos pertenecen a un engranaje y se pueden observar con detalle en la Tabla 2.1. Para poder cargar estos datos se utilizará el siguiente código.

```
1. import scipy.io as sio
2.
3. Datos000=sio.loadmat('/DataForClassification_TimeDomain.mat')
4. TODO=Datos000['AccTimeDomain']}
5. Datos9=sio.loadmat('/Etiquetas.mat')
6. Etiquetas=Datos9['Etiquetas']
```

Los datos se encuentran en formato *.mat*, esto quiere decir que para poder traspasar la información y poder trabajar en ella es necesario utilizar la biblioteca *scipy.io* de Python, la cual es utilizada para leer y escribir datos de una variedad de formatos de archivos. En este caso específico se utilizó *sio*, la cual es la que nos permite cargar los datos a colab y por consiguiente trabajar en ellos.

Se cargan dos datos, los cuales son los datos en bruto y, por otro lado, las etiquetas que consiste en enumerar las fallas que puede tener un engranaje.

3.1.6.4 Algoritmo de entrenamiento

En la fase de entrenamiento del algoritmo es necesario almacenar los vectores característicos y las etiquetas de las distintas clases que tenemos para los ejemplos de entrenamiento.

```
1. from sklearn.model_selection import train_test_split
2. from sklearn.neighbors import KNeighborsClassifier
3. from sklearn.metrics import plot_confusion_matrix
4.
5. #Segmentando Datos
6. X_train, X_test, y_train, y_test = train_test_split(X, Y, tes
   t_size=0.2) #Divide los datos entre entrenamiento y testeo
7.
8. #KNeighborsClassifier: Número de vecinos, función de los
   pesos (Uniforme, basada en la distancia)
   9. Model = KNeighborsClassifier(n_neighbors=1,
      weights='distance')
```

La primera parte del código se puede observar la utilización de la librería sklearn. La primera librería se utiliza para dividir los datos en 3 subconjuntos: entrenamiento, testeo y el tamaño de la prueba. En la octava línea se puede ver se llaman los datos *X*, *Y*, además de *test_size*, X corresponde a los datos en bruto, mientras que Y a las etiquetas. Se utilizó un *test_size* de un 0.2, esto quiere decir que de todos los datos disponibles se utilizara el 20% para testeo y el 80% restante para entrenar el modelo.

Para poder realizar una búsqueda del mejor K que se adapte a los datos propuestos, se aplica un código, que itera el mejor K, el cual por medio de una gráfica se puede observar en el eje Y, el porcentaje y en el eje X, que K se adapta mejor pudiendo elegir de esta manera un K óptimo para el algoritmo K-nearest neighbor.

```
1. k_range = range(1, 20)
2. scores = []
3. for k in k_range:
4.     knn = KNeighborsClassifier(n_neighbors = k)
5.     knn.fit(X_train, y_train)
6.     scores.append(knn.score(X_test, y_test))
7. plt.figure()
8. plt.xlabel('k')
9. plt.ylabel('Precisión')
10.     plt.scatter(k_range, scores)
11.     plt.xticks([0,5,10,15,20])
```

Lo que el algoritmo nos pide es introducir un rango que puede ser el que nosotros deseemos, pero siempre se va a mover a no más de 20 vecinos debido a que el código se sobre exigiría y sería contraproducente buscar vecinos que estén más alejados. Luego de determinar este valor se debe entregar los valores de entrenamiento *X_train*, *Y_train*, para evaluar los diferentes K vecinos. Esto nos dará una puntuación que corresponde al porcentaje de precisión que se verá reflejada en un gráfico como en el de la Figura 3.8

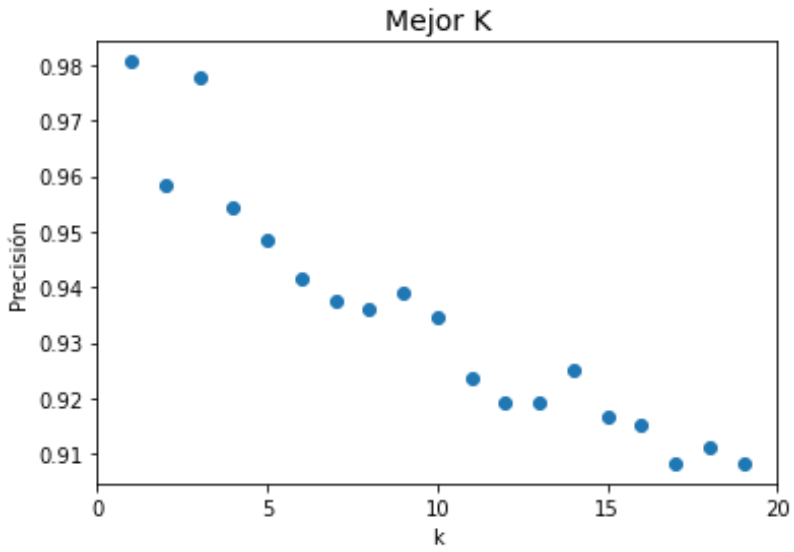


Figura 3.8. Gráfico elección de K. Fuente: Elaboración propia.

Ya elegido el mejor K que, para el caso en concreto, existen 2 opciones muy buenas, en el primer caso tenemos un K=1, esto implica que el dato de testeo se quedara con el primer dato que se encuentre, lo que implica que no habrá una discusión sobre a que falla corresponde, si no más bien se quedara con la primera que pille. Para el segundo caso, donde el K=3, es un caso muy distinto al primero, si bien el primero puede tener mejores resultados esto puede ser un poco erróneo debido a que será debido a que solo eligió a la primera falla que se encontró, sin haber un debate. Es por esto por lo que la mejor opción es un K=3, para que el algoritmo tenga sentido. Luego el modelo se debe elegir los pesos, estos son tres, el primero es un peso uniforme, todos los puntos en cada vecindario se ponderan por igual, luego está el peso por distancia, este consiste en determinar la distancia que existe entre cada vecino influyendo más aquellos que se encuentran más cercanos al punto de muestreo con respecto aquellos que estén más alejados.

Estos son los parámetros más importantes y con mayor peso dentro del modelo de clasificación K-nearest neighbor, aunque existen otros parámetros como elegir el algoritmo para calcular los vecinos más cercanos, que pueden ser ‘‘ball_tree’’, ‘‘kd_tree’’, ‘‘brute’’, esto siempre y cuando se necesite utilizar un algoritmo en específico debido a que por default el modelo busca el mejor de los tres para utilizar el más apto.

3.1.6.5 Algoritmo de clasificación

En la fase de clasificación se evalúan los distintos ejemplos generados en el entrenamiento, aquí no se conoce la clase a la que pertenece los datos debido a que no contiene etiquetas y en el proceso de comparación se le asignara la correspondiente. Esta clasificación es representada por un vector en el espacio característico. Dado los distintos vectores se calcula la distancia entre los vectores almacenados (entrenamiento) versus los vectores nuevos (testeo), aquí es donde se seleccionan los K vecinos más cercanos. Luego de esta comparación se clasificará los datos de testeo con respecto a la clase que más se repita entre los vectores seleccionados.

```
1. Model.fit(X_train, y_train)
2. Yp=Modelo.predict(X_test) #Se evalúa la red con los datos de
   testeo
```

Para realizar finalizar con el proceso de clasificación es necesario evaluar los datos de entrenamiento con los de teste y para esto lo primero que hay que realizar es un ajuste y para esto se utiliza en el código el comando .fit en donde desde el modelo se ajustan los datos con respecto a *X_train* y *y_train* para luego evaluarlos.

3.1.6.6 Resultados

Para poder ver los resultados se presentan dos maneras de poder ver estos datos. Una vez obtenido el modelo y haber aplicado todos los pasos anteriores lo que queda es ver los resultados obtenidos y analizar qué tan efectivos fue la clasificación de los diferentes fallos que existen en el engranaje.

Para esto se representará de dos maneras distintas. La primera corresponde a un KPI por modo de falla y la otra una manera más gráfica de poder ver los datos reflejados en el modelo.

Tabla 3.1. KPI por modo de falla, método K-vecinos. Fuente: Elaboración propia

	Precisión	recall	f1-score	support
Normal	0.99	0.99	0.99	88

Diente Faltante	0.94	0.98	0.96	85
Diente Roto	0.99	0.95	0.97	88
Diente Picado	1.00	1.00	1.00	74
Diente Astillado 1	1.00	1.00	1.00	63
Diente Astillado 2	1.00	0.99	0.99	75
Diente Astillado 3	0.99	1.00	0.99	76
Diente Astillado 4	0.96	1.00	0.98	79
Diente Astillado 5	1.00	0.97	0.98	92
Acurracy			0.98	720
Macro avg	0.99	0.99	0.99	720
weigted avg	0.99	0.98	0.98	720

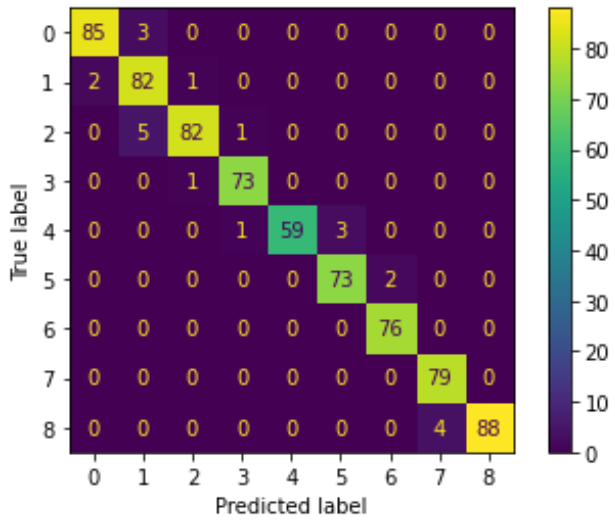


Figura 3.9. Matriz de confusión. Fuente: Elaboración propia.

En la Tabla 3.1 podemos observar el KPI por cada modo de falla en la primera columna se puede observar los diferentes modos de falla. A continuación, le continua la precisión de cada método de fallo, esto con respecto al entrenamiento en comparación con los datos de testeo. Las demás columnas representan un valor que representa valores como

el promedio entre la precisión y el recall. Por último, se contabilizan cuantos elementos de los datos de entrenamiento fueron designados al método de falla.

Luego en la Figura 3.9 se ve de manera más grafica lo que se aprecia en la Tabla 3.1 pudiendo observar algunos datos que no pudieron ser bien clasificados explicando el porqué de la precisión en algunos casos baja a un 94 %.

3.1.6.7 Conclusiones

Como se pudo observar el método utilizado que corresponde a K-nearest neighbor es una estrategia muy útil. En los resultados obteniendo en promedio una exactitud del 98%. El método es muy directo y los datos utilizados fueron de gran ayuda para poder llegar a ese porcentaje, si bien existe un error de un 2% esto sigue siendo una estimación aceptable, debido a que, si se compara con los métodos tradicionales, sumando la serie de errores que siempre puede producir una persona sigue siendo un excelente resultado.

Para poder ver en donde se centra ese 2% hay que comparar los parámetros estadísticos versus los simulados. En primera instancia en la Tabla 3.1 se puede observar que donde hubo una menor precisión fue en el método de falla llamado diente faltante que al observar el grafico correspondiente al dominio del tiempo y el de Fourier en el dominio de la frecuencia en la Figura 2.21 y lo comparamos con el estado sano de la Figura 2.20, podemos ver que se parecen mucho, es por esto que el modelo toma un dato de ejemplo y le cuesta saber a qué método de falla corresponde y es aquí donde ocurren esos pequeños errores. Esto ocurre en varios métodos de falla como se ve claramente en la Tabla 3.1 y es un error que si bien existe y está presente en el modelo de predicción es algo muy mínimo en comparación con el resto de los aciertos en los diferentes métodos de falla. Para corroborar esta información es necesario indagar más en la Tabla 3.1, la precisión nos muestra la calidad de la predicción, siendo de un 99% de efectividad, demostrando que los datos de testeo un 99% fue considerado positivo (dato correcto), de manera detallada el diente faltante obtuvo la menos clasificación con un 94%, obteniendo un 6% de falsos positivos. Recall nos indica la cantidad de elementos que fueron clasificados siendo su falla correcta obteniendo nuevamente un 99% en términos generales, pero un 95% en el diente roto.

Para concluir es un método muy eficaz, capaz de detectar en un 98% los diferentes

modos de falla recordando que existen nueve lo cual no es un número poco razonable en comparación al porcentaje de acierto.

En los valores más llamativos que se observaron en los parámetros estadísticos se encontraban el diente astillado 3 y el diente picado. Donde estos siempre daban como resultado valores muy elevados y distintos a los normales. Al observar la precisión en la Tabla 3.1, estos tienen un porcentaje muy bueno siendo un 100% o un 99% por lo que se puede concluir que mientras los datos sean más separados a los normales en el caso en concreto de un engranaje sano, el método identificará de mejor manera estas fallas.

3.1.7 Random Forest Classifer (Clasificador de bosque aleatorio)

3.1.7.1 Definición del modelo

Random forest es uno de los algoritmos de aprendizaje supervisado más populares en el aprendizaje de máquinas supervisado debido a sus características más flexibles y fáciles de usar. Es utilizado para la resolución de problemas tanto de clasificación como de regresión.

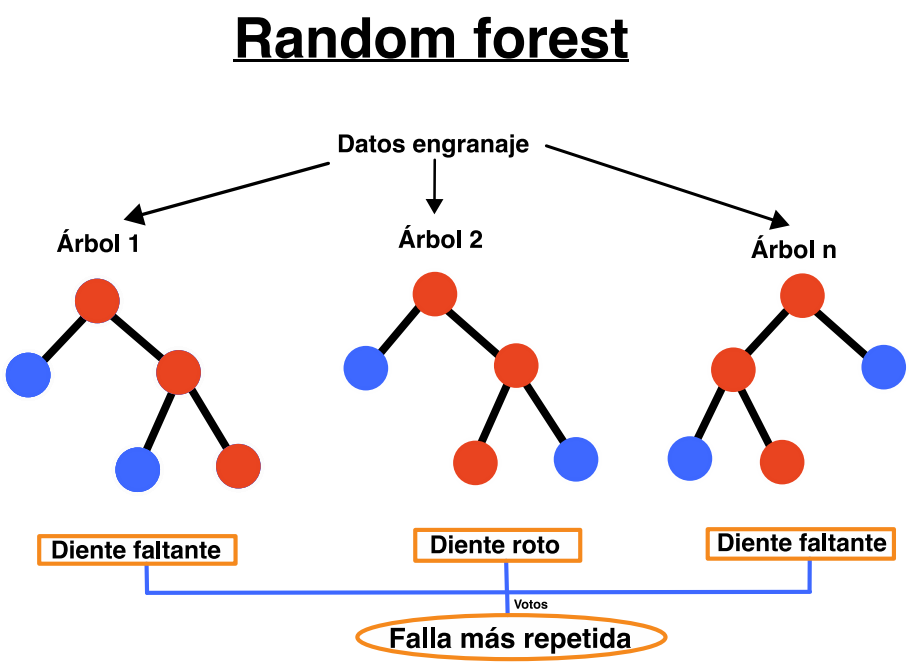


Figura 3.10. Esquema random forest. Fuente: Elaboración propia.

Para comprobar funcionamiento del algoritmo de random forest podemos observar la Figura 3.10 pudiendo ver de manera ilustrativa como se procesa cada paso para poder tomar una decisión, de manera textual quedaría de la siguiente manera:

Lo que hace random forest es que del training o los datos de entrenamiento escoge aleatoriamente un subconjunto de entrenamiento y con estos datos entrena un árbol de decisión (A_i). Donde el dato aleatorio puede ser 1, 30, 50 o 100, dependiendo de la cantidad de árboles creados.

Generalmente, lo que se hace es una selección con reemplazo, eso quiere decir que cuando uno saca un dato aleatorio y lo copia, lo vuelve a reincorporar para que cuando se saque otro dato exista la posibilidad de poder sacarlo nuevamente.

Ahora bien, cuando se desea evaluar los datos, lo que se hace es tomar 1 muestra y se va a clasificar con los diferentes árboles que se crearon. El árbol 1 lo clasificará de una manera, el árbol 2 de otra manera o igual al árbol 1 y así sucesivamente hasta ser evaluado por los n árboles creados en el training.

Al clasificar las muestras, se tomará como decisión final la mayoría de los votos, siendo esta decisión la clasificación para esa muestra.

La ventaja con respecto a los árboles independientes es que estos tienen a sobre ajustarse a los datos de entrenamiento, mientras que random forest mitiga este problema debido a que se realiza un promedio de todos los árboles recordando que cada uno tiene su propia predicción.

3.1.7.2 Parámetros

Random Forest Classifer consta de 18 parámetros, los cuales cada uno cumple una función específica dentro del algoritmo de clasificación. Estos parámetros tienen la finalidad de modificar el código dependiendo de los atributos que se anden buscando.

Parámetros:

- N_estimators : int, Por defecto = 100.
  Corresponde al indicador del número de árboles en el bosque.

- Criterion : (“gini”, “entropy”, “log_loss”), Por defecto =”gini”.

  Esta función cumple con la finalidad de medir la calidad de una división o un nuevo nodo, estos pueden ser ‘‘Gini’’, el cual mide la impureza y el otro es la ‘‘Entropy’’, que utiliza la ganancia. Mientras menos ganancia y menos impureza contenga el nodo analizado mayor será la probabilidad de ser elegido para la selección final.
- Max_depth : int, Por defecto = None.

  Es la máxima profundidad del árbol. Si este se deja como ‘‘None’’, el árbol se expandirá en los nodos que sea necesario hasta que cada uno sea puro es decir un ‘‘Gini’’ o ‘‘Entropy’’ igual a cero, dependiendo del método que se haya seleccionado.
- Min_samples_split : int or float, Por defecto = 2.

  Es aquel número mínimo de muestras que se necesitan para dividir un nodo. Es decir, si existen menos de 2 muestras en el nodo no se hará la división, poniendo fin al árbol.
- Min_samples_leaf : int or float, Por defecto = 1.

  Corresponde al número mínimo de muestras requeridas para estar en un nodo hoja (nodo externo), es decir un nodo sin hijos (sin más divisiones). En comparación con el anterior este parámetro realiza una comprobación antes de generar un nodo, decidiendo si es posible la división. Si la división propuesta da como resultado un hijo (nodo interno) con menos muestras de las permitidas se negará la división dando por termino a la subdivisión.
- Min_weight_fraction_leaf : float, Por defecto = 0.0.

  Es la fracción ponderada mínima de la suma total de pesos de todas las muestras de entrada, para estar en un nodo hoja. Todas las muestras tienen el mismo peso cuando no se proporciona sample_weight. En otras palabras este parámetro lo que quiere dar a entender, es que si se tiene una cantidad N de muestras y algunas son más confiables que otras y no se quiere deshacer de las muestras menos confiables, se debe utilizar el parámetro ‘‘sample_weight’’ el cual se debe proporcionar como un requisito previo, con esto lo que se logra es que este parámetro ‘‘min_weight_fraction_leaf’’ no sea ignorado y se pueda decidir no crear una hoja con un peso menor al indicado en ‘‘Min_weight_fraction_leaf’’.
- Max_features : (“sqrt”, “log2”, None), int or float, Por defecto = ”sqrt”.

Es el número de características a considerar al buscar la mejor división. Esta partición no se detiene hasta encontrar al menos una partición válida de las muestras de nodos.

- Max_leaf_nodes : int, Por defecto = None.

  Es un parámetro que trata de depurar lo mejor posible la creación de árboles. Los mejores nodos se definen como una reducción relativa de la impureza. Si se deja como por defecto, entonces se realiza un número ilimitado de nodos hoja. Este parámetro prioriza la calidad antes que la cantidad.

- Min_impurity_decrease : float, Por defecto = 0.0.

  Un nodo se dividirá siempre y cuando disminuya la impureza la cual sea mayor o menos a la establecida en el parámetro.

- Bootstrap : bool, Por defecto = True.

  Determina si se utiliza todos los datos para construir el árbol o una muestra. Si es False se utiliza todo el conjunto de lo contrario solo una muestra del conjunto de datos.

- Oob_score : bool, Por defecto = False.

  Ya sea para usar muestras listas para usar para estimar la puntuación de generalización. Solo disponible si bootstrap=True. En otras palabras, aquellos valores que no estén presente en un nodo son llamados elementos fuera de bolsa, estos son utilizados para una técnica de validación el cual sirve para obtener resultados de mínima varianza como se aprecia en la Figura 3.11.

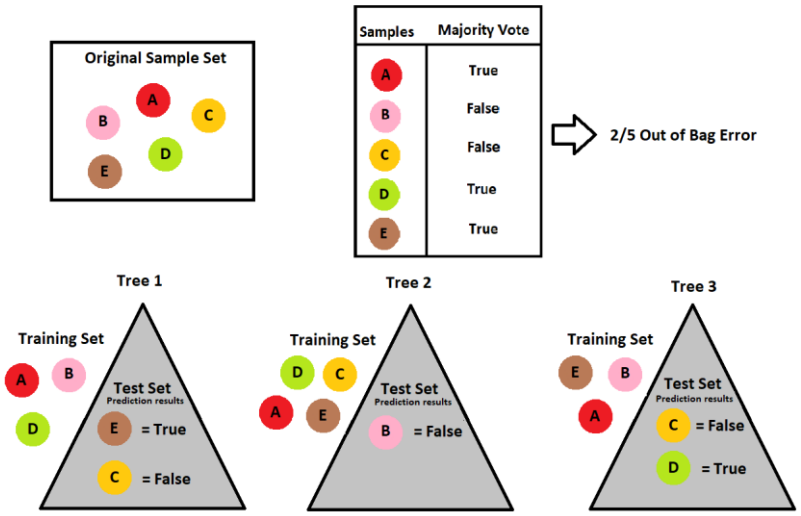


Figura 3.11. Validación de puntuación fuera de bolsa (OOB). Fuente: [39]

- N_jobs : int, Por defecto = None.

  Es aquel parámetro el cual hace referencia al número de trabajos paralelos que se ejecutarán para la búsqueda de vecinos. Si N_jobs = 1 solo se ejecutará un solo trabajo y no se iniciará ningún proceso en paralelo, siendo útil para procesos que requieran depuración. Por otro lado, si N_jobs = -1 significa usar todos los procesadores que tenga disponible el computador exigiéndolo al máximo, pudiendo traer varias consecuencias. Si se deja como N_jobs = -2 significa que se utilizarán todas menos un procesador, cumpliendo la siguiente ecuación: (n_cpus + 1 + n_jobs).

- Random_state : int, RandomState instance or None, Por defecto = None.

  Controla tanto la aleatoriedad del arranque de las muestras utilizadas al construir árboles (si bootstrap=True) como el muestreo de las características a considerar cuando se busca la mejor división en cada nodo.

- Verbose : int, Por defecto = 0.

  Controla la verbosidad al ajustar y predecir. La verbosidad significa mostrar más información ‘‘prolija’’ para la tarea. Mientras más alto sea el número más información se informará.

- Warm_start : bool, Por defecto = False.

  Cuando se establece en True, reutiliza la solución de la llamada anterior para ajustar y agregar más estimadores al conjunto; de lo contrario, solo ajusta un bosque

completamente nuevo. En otras palabras, cuando se ajusta un estimador repetidamente en el mismo conjunto de datos, pero para múltiples valores de parámetros (como para encontrar el valor que maximiza el rendimiento, puede ser posible reutilizar aspectos del modelo aprendidos del valor de parámetro anterior, ahorrando tiempo.

- Class_weight : (“balanced”, “balanced_subsample”), dict or list of dicts, Por defecto =None.

  Corresponde a los pesos asociados. Si no se proporciona, se supone que todas las clases tienen peso uno.

  El modo "equilibrado" utiliza los valores de y para ajustar automáticamente los pesos inversamente proporcionales a las frecuencias de clase en los datos de entrada.

  El modo "submuestra_equilibrada" es igual que "equilibrado", excepto que los pesos se calculan en función de la muestra de arranque para cada árbol cultivado.

- Ccp_alpha : non-negative float, Por defecto = 0.0.

  Parámetro de complejidad utilizado para la poda de complejidad de costo mínimo. En otras palabras, se utiliza para evitar que un árbol se sobreajuste. La poda de complejidad de costos proporciona otra opción para controlar el tamaño de un árbol. Mientras mayor sea ccp_alpha aumentaran el número de nodos podados.

- Max_samples : int or float, Por defecto = None.

  Si bootstrap es True, el número de muestras que se extraerán de X para entrenar cada estimador base.

  Si es ninguno, se extraerán 0 muestras.

  Si es int, se extrae max_samples muestras.

  Si es float, se extraerá muestras que estén dentro del rango de la siguiente ecuación en un intervalo de (0.0 , 1.0)

$$\max \_\text{sample} * X.shape\ [0]$$

3.1.7.3 Algoritmo de entrenamiento

```
1. from sklearn.model_selection import train_test_split
2.
3. X_train, X_test, y_train, y_test = train_test_split(X, Y, tes
   t_size=0.2) #Divide los datos entre entrenamiento y testeo
```

Para el entrenamiento de datos se utilizará de manera equitativa en los diferentes modelos los mismos datos de entrada y el mismo porcentaje de entrenamiento del modelo.

Se llamará a la librería de sklearn, utilizando *train_test_split* para poder entrenar el algoritmo.

Donde:

X: Pertenece a todos los del engranaje mencionados en la Tabla 2.1.

Y: Conjunto de datos etiquetados para cada estado del engranaje.

3.1.7.4 Algoritmo de clasificación

```
1. #RandomForestClassifier: Número de árboles, criterio de
   selección
2.
3. from sklearn.ensemble import RandomForestClassifier
4.
5. #'n_estimator': creación de número de árboles
6. #seleccion de método de criterio: 'entropy' o 'gini'
7.
8. Model=RandomForestClassifier(n_estimators=100,criterion='entr
   opy')
9. Model.fit(X_train, y_train)
10. Yp=Model.predict(X_test)# Se evalúa la red con los datos de
   testeo
```

La carga de los datos y el entrenamiento de los datos es el mismo que en el algoritmo anterior, esto para al finalizar, poder comparar las diferentes alternativas que se presentan para el aprendizaje de máquina y ver cuáles son las ventajas y desventajas de estos.

El algoritmo de clasificación trata de evaluar y crear los árboles de decisiones. Para poder realizar esto es necesario parametrizar algunos elementos. En la línea del código número 10 podemos analizar que se crea el modelo, este con un *n_estimators* igual a 100, esto quiere decir que el número de árboles en el bosque será de 100, este valor es variable, uno puede utilizar desde 1 árbol hasta los que se les ocurra, siempre y cuando tenga un tamaño de procesamiento que sea capaz de soportarlo.

El criterio por utilizar se puede elegir entre dos alternativas que nos da el modelo.

Esta función sirve para medir la calidad de una división, las cuales pueden ser por ‘‘Gini’’ o ‘‘entropía’’. La impureza Gini se estima de la siguiente fórmula.

$$Gini = 1 - \sum_i S_i^2$$

Donde:

S_i = es el punto i-ésimo en la evolución temporal de la señal S.

Gini trata de medir la frecuencia con la que un dato está etiquetado de una manera incorrecta cuando esta se etiqueta de manera aleatoria. Cada rama del random forest separa los conjuntos por cada método de fallo. El Gini es 0 cuando todos los elementos contenidos en una rama pertenecen a una sola clase. De no ser así, el nodo se seguirá dividiendo, optando por las características con menor índice de Gini.

Por otro lado, la entropía se calcula de la siguiente manera.

$$\sum_i S_i \cdot \log_2(S_i)$$

Donde:

S_i = es el punto i-ésimo en la evolución temporal de la señal S.

La entropía es una medida que indica el desorden en los diferentes árboles. De una forma similar, está al igual que el Gini, hace la separación a otra rama dependiendo del valor con menos entropía, es decir, menos desorden. Cuando su valor es cero, quiere decir que los valores y resultados corresponden a cada etiqueta dando fin a la clasificación.

Ahora bien, las dos estrategias son similares, buscando en teoría la misma definición que es que cuando los valores muestreados pertenezcan a los valores etiquetados, el árbol terminará dando en cada caso un Gini igual a cero y una entropía igual a cero.

La diferencia que hay entre ambas son los valores con los cuales se evalúa el modelo para decidir si hay que seguir ampliando el árbol o no. El Gini tiene valores entre [0, 0.5] mientras que la entropía toma valores entre [0, 0.1], es decir, la entropía puede tomar más valores que el Gini.

Si esto se lleva a la parte computacional, habrá muchos factores que influyan, primero la cantidad de árboles que se establezcan el algoritmo de clasificación, mientras que la otra corresponde en una cuestión del modelo matemático que hay de tras de cada método de criterio. En el caso del Gini tenemos una simple sumatoria que no es muy difícil de procesar computacionalmente, mientras que en el caso de la entropía hay un logaritmo, lo que en sí produce en grandes cantidades de datos un procesamiento de información y gasto de recursos computacionales importantes.

Pablo Aznar [40] en un estudio que realizó comprobó la diferencia en los tiempos de entrenamiento del modelo según para cada método de criterio, utilizando 10 características, las cuales se pueden agrupar en 4 grupos distintos dependiendo si las características eran de manera informativa, redundantes, repetidas o aleatorias. Además, cada grupo está compuesto por 5 conjuntos de datos donde el número de muestras va desde los 100, 1.000, 10.000, 100.000, 200.000 [40].

Figura 3.12. Factor del tiempo según el método de criterio. Fuente: [40].

Como se puede observar en la Figura 3.12, se nota con claridad que el utilizar el criterio de entropía (azul) nos toma un tiempo mayor al que si se utilizara el criterio de Gini (rojo). Esto no quiere decir que los resultados sean distintos o uno peor que otro, sino que aquí se toma en consideración el valor computacional para las distintas clasificaciones que se puedan realizar a futuro.

Otra característica que puede ser relevante a la hora de modelar el algoritmo para una mejor clasificación es controlar el tamaño de la muestra, por defecto el algoritmo considera toda la muestra para realizar el bosque aleatorio, pero si esto no se desea se debe parametrizar el valor de *max_samples*, que indica el valor máximo de muestras que se iteran en el algoritmo.

Para que el nodo se divida por defecto debe haber 2 muestras, esta puede ser un *int* o un *float* y se determina por el algoritmo de *min_samples_split.* Si se desean más muestras

para poder dividir el nodo, es necesario cambiar el *min_samples_split* a la cantidad de muestras que se deseen.

La división de nodos también se ve afectada por otro criterio, el *max_features* el cual considera las características al buscar la mejor división. Por defecto, la búsqueda de una división no se detiene hasta encontrar al menos una división válida de las muestras. Este criterio puede ser la raíz o el logaritmo. Por defecto, el algoritmo trabaja con la raíz (*sqrt*) pero al igual que el criterio para medir la calidad de la división se omite el logaritmo debido a lo que se conversó con anterioridad sobre los tiempos de espera.

Por último, otro factor importante es el *max_depth* el cual determina la profundidad del árbol, es decir que tan grande se hará dependiendo de las divisiones de cada nodo. Por defecto es ninguno lo que quiere decir que los nodos se expanden hasta que todas las partes sean puras o hasta que las hojas contengan menos de *min_samples_split* que se determinó con anterioridad.

3.1.7.5 Resultados

```
#matriz de confusión
from sklearn.metrics import classification_report
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(Model, X_test, y_test)
plt.show()

#KPI por modo de falla

target_names = (['Normal','Diente Faltante','Diente Roto','Diente Picado','Diente Astillado 1','Diente Astillado 2','Diente Astillado 3','Diente Astillado 4','Diente Astillado 5'])
print(classification_report(y_test, Yp, target_names=target_names))
```

Una vez ya implementado el algoritmo de entrenamiento y el de clasificación, es momento de conocer si los datos que se obtuvieron son lo suficientemente buenos y aceptables para una implementación a futuro.

A continuación, se presentarán los resultados de cuatro simulaciones, esto para determinar si existe una diferencia muy notoria al aplicar el método de criterio de entropía y el del Gini. Además de ver que tanto influye la cantidad de árboles que nosotros estimemos en el algoritmo. Se realizarán pruebas con un total de 100 árboles para el método de criterio de entropía y 100 árboles para estimar el método de Gini. Posteriormente, se utilizará los mismos criterios, pero esta vez con 1000 árboles, para determinar la influencia de estos.

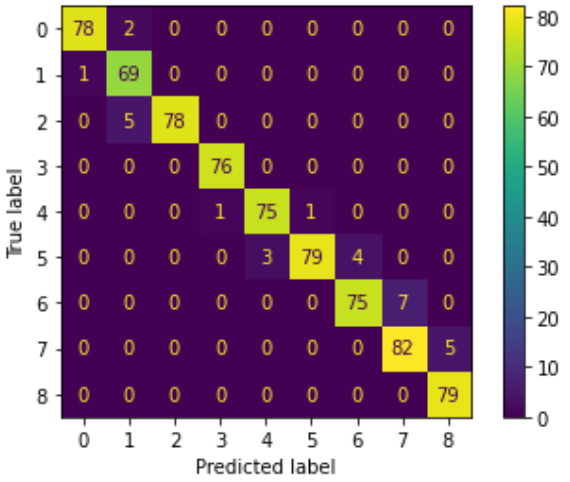


Figura 3.13. Matriz de confusión. Árboles: 1000, método: Gini. Fuente: Elaboración propia.

Tabla 3.2. KPI por método de falla. Árboles: 1000 método: Gini. Fuente: Elaboración propia.

	Precisión	recall	f1-score	support
Normal	0.99	0.97	0.98	80
Diente Faltante	0.91	0.99	0.95	70
Diente Roto	1.00	0.94	0.97	83
Diente Picado	0.99	1.00	0.99	76
Diente Astillado 1	0.96	0.97	0.97	77
Diente Astillado 2	0.99	0.92	0.95	86
Diente Astillado 3	0.95	0.91	0.93	82
Diente Astillado 4	0.92	0.94	0.93	87

Diente Astillado 5	0.94	1.00	0.97	79
Acurracy			0.96	720
Macro avg	0.96	0.96	0.96	720
weigted avg	0.96	0.96	0.96	720

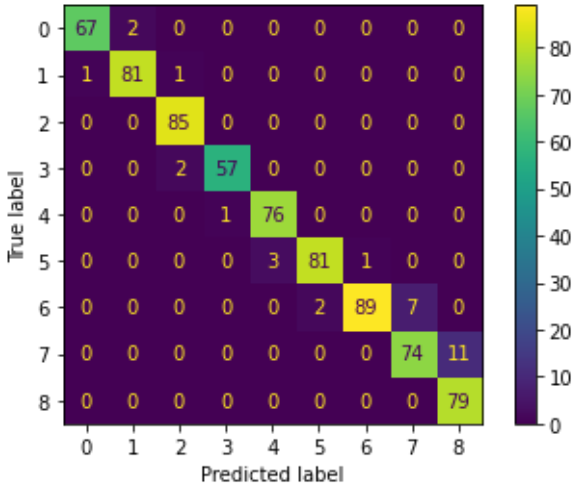


Figura 3.14. Matriz de confusión. Árboles: 1000, método: Entropía. Fuente: Elaboración propia.

Tabla 3.3. KPI por método de falla. Árboles: 1000 método: Entropía. Fuente: Elaboración propia.

	Precisión	recall	f1-score	support
Normal	0.99	0.97	0.98	69
Diente Faltante	0.94	0.98	0.98	83
Diente Roto	0.97	1.00	0.98	85
Diente Picado	0.98	0.97	0.97	59
Diente Astillado 1	0.96	0.99	0.97	77
Diente Astillado 2	0.98	0.95	0.96	85
Diente Astillado 3	0.99	0.91	0.95	98
Diente Astillado 4	0.91	0.87	0.89	85

Diente Astillado 5	0.88	1.00	0.93	79
Acurracy			0.96	720
Macro avg	0.96	0.96	0.96	720
weigted avg	0.96	0.96	0.96	720

En la Figura 3.13 y Figura 3.14 se puede observar los resultados obtenidos a través de la matriz de confusión, para ambos casos los datos obtenidos son muy parecidos donde a simple vista no se ven errores muy grandes, por lo que se debe apreciar el detalle de porcentaje que se encuentran en la Tabla 3.2 y Tabla 3.3. En ambos casos el resultado de exactitud es de un 96% por lo que no existe una diferencia notoria en la clasificación de las fallas dependiendo estrictamente del método de criterio utilizado. Si nos vamos al detalle en la Tabla 3.2, hay 2 dientes que tienen una precisión muy mala en comparación con los demás siendo del diente faltante y del diente astillado con una precisión de un 91 % y 92%. Esto nos quiere decir que hubo datos que se clasificaron de mala manera, obteniendo como resultados datos falsos positivos. Por otro lado, el recall de el diente faltante fue excelente, no así del diente astillado 2 y 3, los cuales obtuvieron un porcentaje de 91 % y 92 %. Esto quiere decir en pocas palabras que los datos positivos de esas muestras solo el porcentaje descrito pudo ser clasificado, quedando con un porcentaje más o menos del 8% sin poder ser clasificado.

Para la Tabla 3.3, en comparación con la anterior, podemos observar que la variación del parámetro se vio reflejado en la precisión siendo el más afectado el diente astillado 5, en conjunto con el diente astillado 4, siendo de un 88% y 91% respectivamente, obteniendo falsos positivos con mayor porcentaje que en el caso anterior, siendo preocupante. En el caso del recall, los más afectados fueron el diente astillado 5 con un 93% y el diente astillado 4 con un 89%, obteniendo que los datos positivos que se encontraban en ese rango un 7% no fue clasificado en el diente astillado 5 y un 11% no fue clasificado en el diente astillado 4, quedando una estimación bastante preocupante.

El f1 score nos indica un promedio entre la precisión y el recall, en ambos casos en términos generales son iguales, no que nos indica que en términos generales se comportan de

igual manera, enfocándose en unas fallas más que en otras, pero a final de cuentas obteniendo los mismos resultados. Concluyendo que el método de Gini y la entropía da resultados iguales en ambos casos para los parámetros descritos.

Lo que si se pudo observar que al ser 1000 la cantidad de árboles generados el tiempo de espera fue de un par de minutos. Haciendo la comparación de ambos criterios se puede confirmar la teoría explicada por Pablo Aznar, confirmando una diferencia muy notoria en la espera de tiempo entre el método de criterio Gini, con el de entropía, siendo el primero mucho más rápido a la hora de clasificar la información versus la entropía que demore prácticamente el doble en este caso en concreto.

Para el siguiente caso, se disminuirán la cantidad de árboles del modelo, en concreto se utilizarán 100 árboles, para poder diferenciar si existe una diferencia entre los dos modelos descritos, el método de Gini y el de entropía.

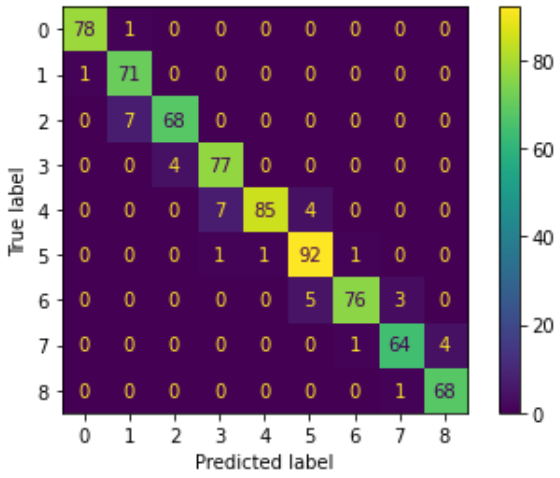


Figura 3.15. Matriz de confusión. Árboles: 100, método: Gini. Fuente: Elaboración propia.

Tabla 3.4. KPI por método de falla. Árboles : 100 método: Gini. Fuente: Elaboración propia.

	Precisión	recall	f1-score	support
Normal	0.99	0.99	0.99	79
Diente Faltante	0.90	0.99	0.94	72

Diente Roto	0.94	0.91	0.93	75
Diente Picado	0.91	0.95	0.93	81
Diente Astillado 1	0.99	0.89	0.93	96
Diente Astillado 2	0.91	0.97	0.94	95
Diente Astillado 3	0.97	0.90	0.94	84
Diente Astillado 4	0.94	0.93	0.93	69
Diente Astillado 5	0.94	0.99	0.96	69
Acurracy			0.94	720
Macro avg	0.94	0.94	0.94	720
weigted avg	0.94	0.94	0.94	720

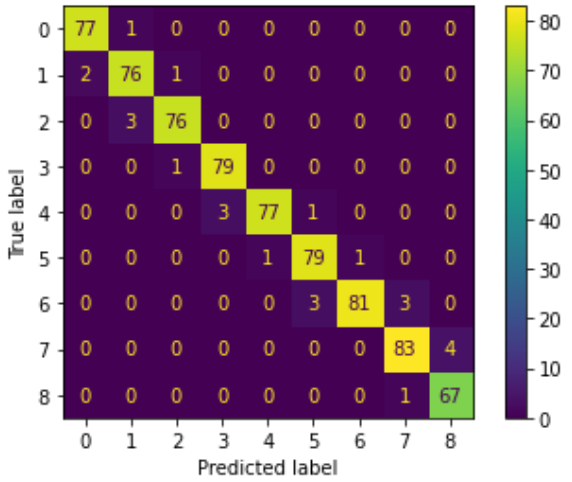


Figura 3.16. Matriz de confusión. Árboles: 100, método: Entropía. Fuente: Elaboración propia.

Tabla 3.5. KPI por método de falla. Árboles : 100 método: Entropía. Fuente: Elaboración propia.

	Precisión	recall	f1-score	support
Normal	0.97	0.99	0.98	78
Diente Faltante	0.95	0.96	0.96	79
Diente Roto	0.97	0.96	0.97	79

Diente Picado	0.96	0.99	0.98	80
Diente Astillado 1	0.99	0.95	0.97	81
Diente Astillado 2	0.95	0.98	0.96	81
Diente Astillado 3	0.99	0.93	0.96	87
Diente Astillado 4	0.95	0.95	0.95	87
Diente Astillado 5	0.94	0.99	0.96	68
Acurracy			0.97	720
Macro avg	0.97	0.97	0.97	720
weigted avg	0.97	0.97	0.97	720

Esta vez se disminuyó con creces la cantidad de árboles creados para clasificar las diferentes fallas dentro del engranaje. En la Figura 3.15 y Figura 3.16 se pueden observar la matriz de confusión de ambos métodos de criterio. Aquí se puede observar a simple vista una mejora en la Figura 3.16 pero siempre para confirmar la información se necesitan datos concretos como los presentados en la Tabla 3.4 y Tabla 3.5.

Aquí se observa algo muy diferente a los que estábamos viendo con anterioridad y que abre distintas posibilidades dentro del método de clasificación random forest. En primera instancia el método de criterio Gini nuevamente fue más rápido que el de entropía, pero en comparación con los resultados vistos con anterioridad en este caso en concreto no ocurrió que ambos criterios utilizados fueran idénticos en la exactitud del modelo. Lo que se puede apreciar es que en la Tabla 3.5 hay una exactitud del 97%, mientras que en la Tabla 3.4 su exactitud baja a un 94%. Si bien el criterio utilizado de Gini es más rápido que el criterio de entropía esto disminuye considerablemente al ser tan poca la cantidad de árboles por lo que como primera conclusión se puede estimar que para pocas cantidades de árboles creados el criterio influye muy poco y como segunda conclusión y la más importante es que a pesar de que uno supondría que los criterios pueden dar valores cercanos esto no siempre se cumple, si no que depende de la cantidad de árboles creados.

3.1.7.6 Conclusiones

Para poder finalizar el algoritmo de clasificación random forest es necesario poder agrupar las diferentes características que nos entrega esta metodología de clasificación dentro de las diferentes ramas del machine learning.

Primero que todo el algoritmo presenta resultados muy buenos siendo en el mejor de los casos un 97%. Hay que tener siempre en consideración la cantidad de árboles creados para el algoritmo debido a que como se pudo notar es una variante de las más importantes.

Para continuar se puede concluir que es un método donde no se puede buscar ‘‘la mejor cantidad’’ de árboles para un resultado optimo, por lo que es muy fácil fallar en este sentido, al igual que en la selección del método de criterio, pudiendo enfocarse a simple vista en el criterio Gini debido a que por estudios es más rápido el criterio de entropía, no siempre siendo la mejor elección para clasificar las distintas fallas ya sea de un engranaje, rodamiento u otro tipo de equipo a clasificar.

Es un método que se podría decir que es ‘‘democrático’’, en donde el mejor promedio es siempre al que eligen, además de ser un algoritmo fácil de implementar identificando los datos de entrenamiento y no sobre cargándolo.

Los resultados se realizaron con el mejor porcentaje de entrenamiento y testeo que se produjo en la iteración, es por esto por lo que todos los algoritmos se realizan con un 20% de testeo y un 80 % de entrenamiento.

3.1.8 SVM (Support vector machine)

Se utilizará para el algoritmo de SVM la librería de sklearn.svm para la creación del código de clasificación en Colaboratory de Google, donde al ser un algoritmo de clasificación de llama además a SVC en sklearn.

3.1.8.1 Definición del modelo

Máquina de vectores de soporte o SVM es un algoritmo que se puede separar en clasificación y regresión, cuando estamos hablando de una clasificación se llama SVC (C-

support vector classification), mientras que, por otro lado, tenemos SVR (support vector regresion) [41].

SVM se basa principalmente en Maximal Margin Classifier, que a su vez está basado en los conceptos del hiperplano. Suponiendo un espacio dimensional $p - dimesinal$, un hiperplano es un subespacio plano de dimensiones $p - 1$. Por ejemplo, si tenemos un espacio de dos dimensiones, el hiperplano es un subespacio de 1 dimensión, como se aprecia en la Figura 3.17.

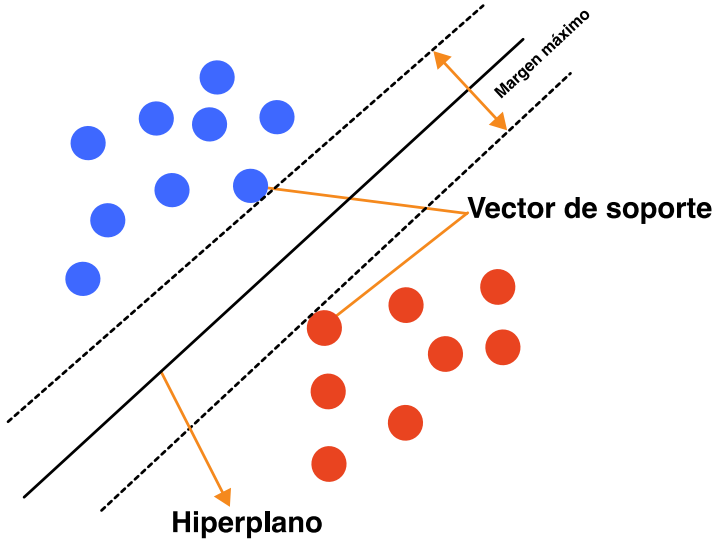


Figura 3.17. Hiperplano 2 dimensiones. Fuente: Elaboración propia.

Al momento de llevar el concepto anterior a la práctica, se encuentra que los datos no están separados perfectamente, por lo que el *Maximal Margin Classifer* es muy poco probable de aplicar en la mayoría de los casos. Por estas razones es preferible crear un clasificador basado en un hiperplano, que no va a separar las muestras en los grupos perfectamente, pero será más robusto y la capacidad predictiva sea mayor. Es aquí donde se implementan los clasificadores de vector soporte, para poder lograr esta metodología se debe disminuir el margen de clasificación permitiendo que ciertas muestras estén en el lado incorrecto del margen o incluso del hiperplano.

Las máquinas vectoriales de soporte se pueden dividir en 3 subgrupos, los cuales son:

1- Lineal con separación perfecta.

Cuando tenemos una separación perfecta, estamos hablando de un conjunto de datos de entrenamiento, como se puede observar en la Figura 3.18, donde los datos que se tienen se pueden separar perfectamente por una línea o un hiperplano.

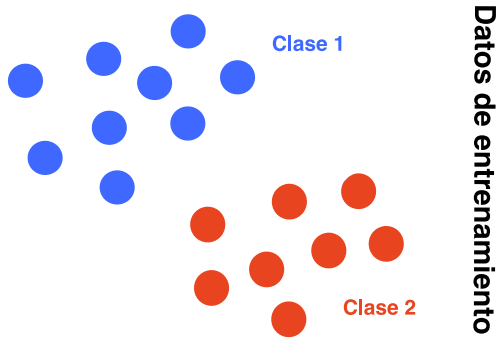


Figura 3.18. SVM: dos clases. Fuente: Elaboración propia.

La idea es poder encontrar una línea de decisión que separe los conjuntos de datos. Esta línea puede tomar distintas posibilidades para poder separar al conjunto de datos, y es aquí donde uno se pregunta. ¿Cuál es la mejor línea de decisión para este conjunto de datos?

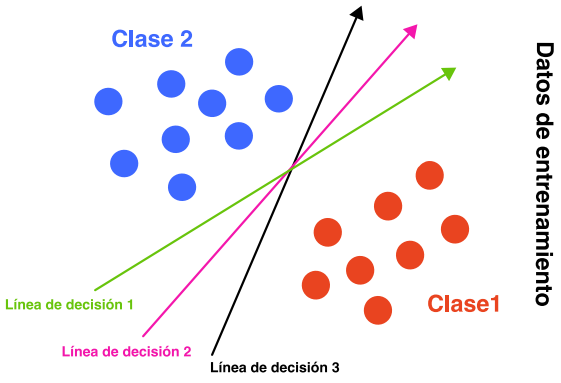


Figura 3.19. Distintas representaciones de una línea de decisión. Fuente: Elaboración propia.

Como se puede ver en la Figura 3.19, pueden existir una infinita cantidad de líneas que serán las encargadas de separar las clases perfectamente. Para poder encontrar la mejor línea de decisión es necesario optimizar la función objetivo.

Para una línea de decisión definimos el margen ‘b1’ como la distancia perpendicular de la línea de la muestra más cercana de la clase 1, esto puede ser, a la muestra o las muestras que se encuentren más cercanas. Lo mismo ocurrirá para el margen ‘b2’, siendo en este caso para la clase 2.

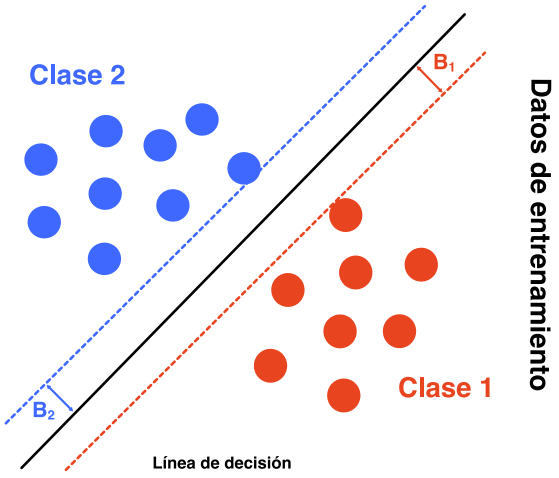


Figura 3.20. Estimación margen b1 y b2. Fuente: Elaboración propia.

El primer paso es escoger una línea de tal forma que ‘b1’ y ‘b2’ estén a la misma distancia de la línea de decisión, por lo que $b1 = b2$.

Como segundo punto, la distancia entre líneas (b) debe maximizarse, es decir que se debe elegir la línea que tenga un mayor margen, más conocido como *Maximal Margin Classifer*.

Un consejo importante es que la línea de decisión que se estimó, la cual fue máxima para los datos que se obtuvieron, va a depender única y exclusivamente de los datos que estén tocando el margen. Si por alguna razón de los datos de entrenamiento se eliminan datos que no sean los que está tocando el margen de la línea de decisión, no va a influir en esta. Estos puntos son los conocidos vectores de soporte que se pueden observar en la Figura 3.21.

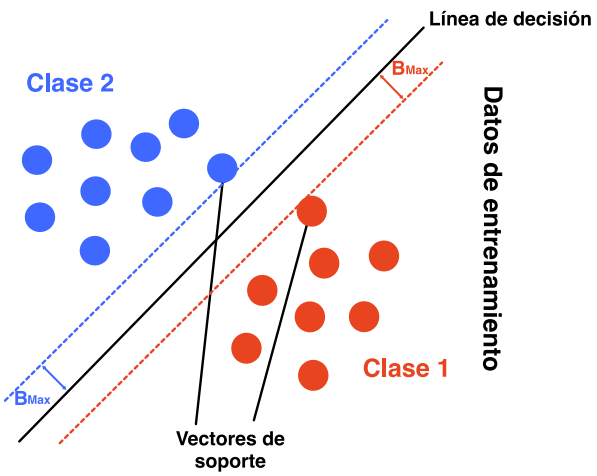


Figura 3.21. Vectores de soporte en el margen de línea de decisión. Fuente: Elaboración propia.

Planteamiento matemático:

La solución será $g(x)$, la cual corresponderá a nuestra línea de decisión, la cual dependerá únicamente de los valores de nuestros vectores de soporte. Dependiendo del valor que nos entregue $g(x)$ dependerá del lado en que se encuentre en la línea de decisión. Si $g(x) > 0$ corresponderá a la parte azul, de lo contrario si $g(x) < 0$ pertenecerá al lado azul, como se observa en la Figura 3.22.

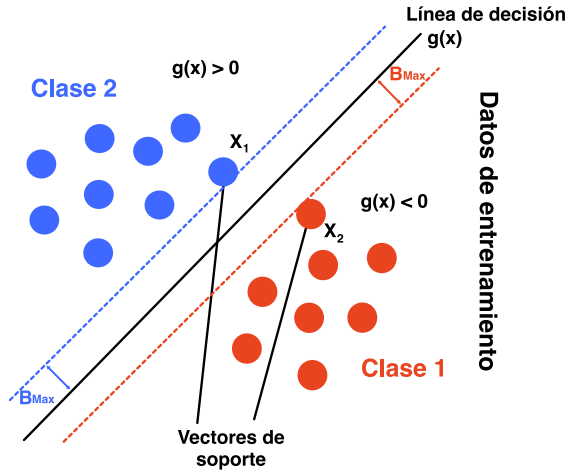


Figura 3.22. Planteamiento matemático. Fuente: Elaboración propia.

Matemáticamente, $g(x)$ se calcula de la siguiente manera:

$$g(x) = \sum_{i=1}^{N} \alpha_i z_i x_i \cdot x + a_0$$

Donde:

x_i: Vectores de soporte

x: Muestras o datos de entrenamiento

z_i: Valor perteneciente a un lado u otro de la línea de decisión

α_i: Valor de salida (constante)

a_0: Constante

En la ecuación se puede observar que hay un producto punto entre x_i y x, donde x_i pertenece a los vectores de soporte, por las muestras que uno evalúa. Los z_i son +1 o -1 dependiendo del lado que pertenezca de la línea de decisión. El valor de α y a_0 se obtendrán al final del proceso de optimización siendo el output.

2- Lineal sin separación perfecta.

En este caso los datos de la clase 1 y 2 se traslapan uno sobre otros, como se observa en la Figura 3.23.

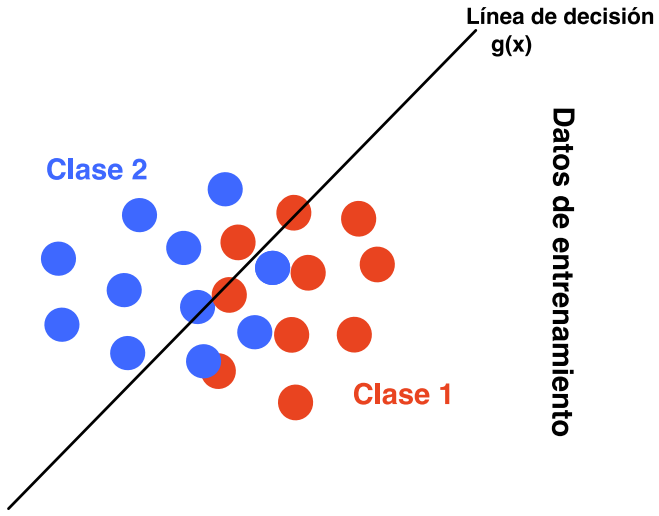


Figura 3.23. Clasificación lineal no perfecta. Fuente: Elaboración propia.

Para poder abordar este tipo de problemas lo que uno debe hacer es considerar solo las muestras clasificadas erróneamente, que quiere decir, aquellas que según la línea de decisión no se encuentren en el lado correspondiente a su clase, como se observa en la Figura 3.24.

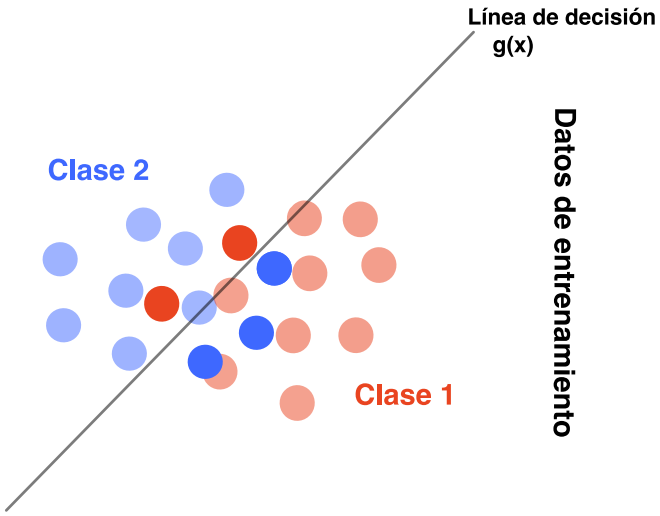


Figura 3.24. Errores de clasificación. Fuente: Elaboración propia.

Ahora bien, a diferencia del modelo anterior, los vectores de soporte corresponderán a estas muestras o errores que se obtuvieron al trazar la línea de

decisión. Midiendo a cada nuevo vector de soporte el error según corresponda a la línea de decisión como se ve en la

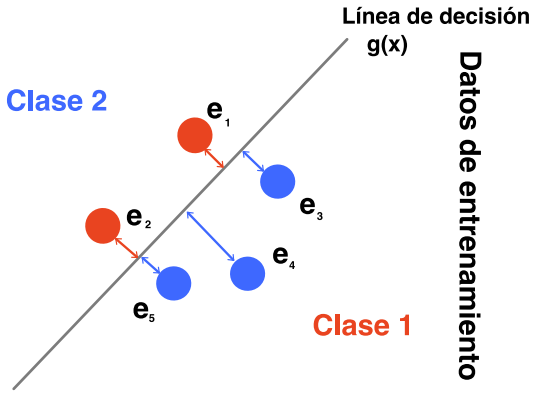


Figura 3.25. Medición del error de las diferentes clases con respecto a la línea de decisión. Fuente: Elaboración propia.

Para poder hallar la línea de decisión es necesario poder determinar el error, ya que esta va a ser aquella que minimice el error de las diferentes clases mal seleccionadas.

$$e = \sum e_i \longrightarrow \text{Mínimo}$$

Si esta línea de decisión quedara mal puesta, la consecuencia sería una exactitud menor debido a que habrá por consiguiente más elementos mal clasificados.

Planteamiento matemático:

$$g(x) = \sum_{i=1}^{N} \alpha_i z_i K(x_i \cdot x) + a_0$$

Como se mencionó con anterioridad, en este caso es posible aceptar un pequeño margen de error, este error va a ser regularizado por una contante C, esta va a decidir si el margen va a ser mínimo o máximo, pudiendo tener una mayor cantidad de elementos que no serán clasificados. Si C es grande el margen es más pequeño, en

cambio, sí C es pequeño hay una mayor tolerancia de excepciones, por lo que la exactitud no será la que se podría esperar.

La ecuación que determina la línea de decisión, como se observa en la educación, es muy parecida a la anterior, la única diferencia es que se aplica una constante K, la cual pertenece a kernel.

3- No lineal.

Si bien el trazar una línea recta en un principio resolvía gran parte de la incógnita pudiendo clasificar las distintas clases, no era la mejor solución debido a que existían muchos valores que quedaban sin poder clasificar. Es por esto por lo que bajo los mismos conceptos se creó un modelo capaz de separar los datos a través de algún elemento separador que no fuera una línea recta, como se vio con anterioridad.

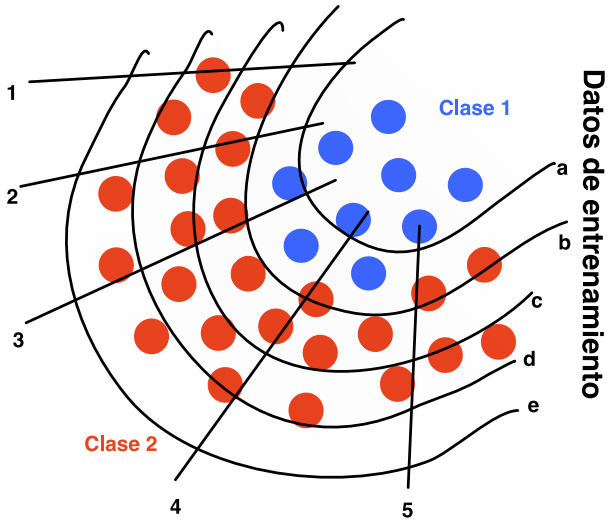


Figura 3.26. Planteamiento no lineal. Fuente: Elaboración propia.

Dada dos características y no pudiendo hacer una separación lineal, se pensó que de alguna forma las líneas curvas se pudieran estirar o moverse para que el conjunto de datos que este ubicado en una curvatura que se puede observar en la Figura 3.26 pudieran estar más alineados.

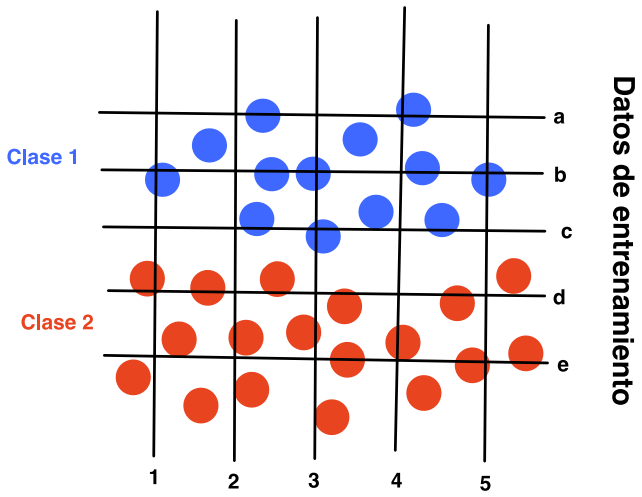


Figura 3.27. Transformación geométrica no lineal. Fuente: Elaboración propia.

Al plantearse un nuevo sistema de coordenadas donde los números correspondan a una dimensión y las letras en otra, lo que ocurre es una transformación geométrica no lineal, dando como resultado lo que se aprecia en la Figura 3.27.

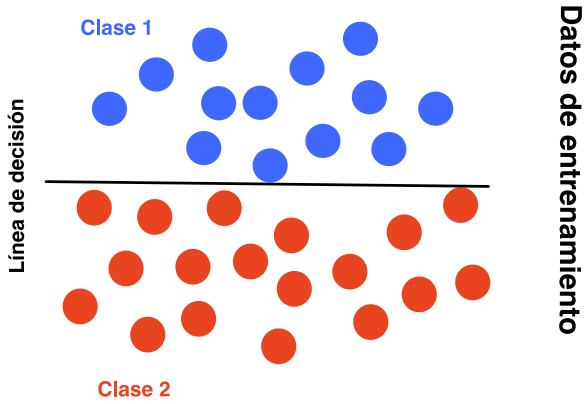


Figura 3.28. SVM lineal en nuevo sistema de coordenadas. Fuente: Elaboración propia.

Ahora en el nuevo sistema de coordenadas se puede trazar la misma línea con la cual se estuvo trabajando en los problemas anteriores, quedando el resultado como se puede observar en la Figura 3.28.

Del mismo modo uno puede ir trabajando con distintas dimensiones, si el resultado que uno obtuvo aún no se puede trabajar en 2 dimensiones es posible pasarlo a 3 o a 4 dimensiones, con tal de que el resultado final sea poder separar los datos en las diferentes clases por medio de esta línea recta.

Para poder obtener la línea de decisión lo que uno hace una vez obtenida la línea de separación es aplicar una transformación geométrica inversa, obteniendo como resultado lo obtenido en la Figura 3.29.

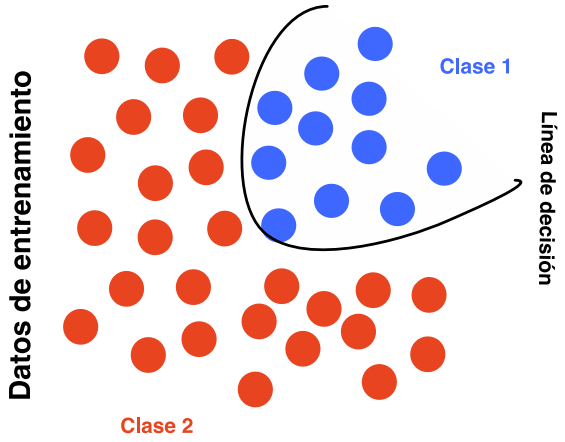


Figura 3.29. Línea de decisión aplicada el método geométrico inverso. Fuente: Elaboración propia.

Para aplicar la línea de decisión $g(x)$ y aplicar la ecuación que se obtuvo en el SVM lineal, es necesario aplicar una transformación de la variable X. Ahora se aplicará una $h(x)$ quedando como resultado un $g(h(x))$, donde $h(x)$ es la transformación, obteniendo como la nueva línea de decisión la siguiente ecuación:

$$g\big(h(x)\big) = \sum_{i=1}^{N} \alpha_i z_i h(x_i) \cdot h(x) + a_0$$

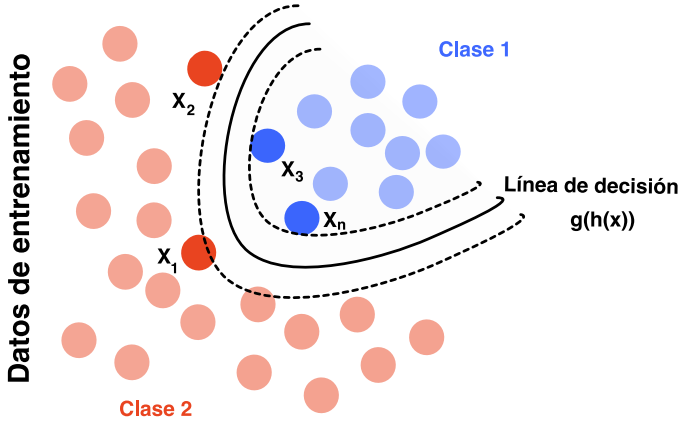


Figura 3.30. Nueva línea de decisión. Fuente: Elaboración propia.

Para hacer las transformaciones un poco más fáciles, las personas que crearon el SVM indican que lo más importante no es la transformación, sino el producto punto de los vectores de soporte transformados con la muestra x de muestra transformada.

De aquí nace Kernel, el cual es el producto punto del que se ha estado hablando. Aquellos que crearon SVM propusieron en su tiempo producto punto para las distintas formas dependiendo de cómo abordar el problema.

Alguna de estas sugerencias son:

- Lineal: $< x', x >$
- Polinomial: $< 1 + < x', x >>^n$
- Base radial: $\exp\left(-\frac{\left|\left|x', x\right|\right|^2}{c}\right)$
- Sigmoide: $tanh(K_1 < x', x > + K_2)$

3.1.8.2 Parámetros

Support vector machine consta de 15 parámetros, los cuales cada uno cumple una función específica dentro del algoritmo de clasificación. Estos parámetros tienen la finalidad de modificar el código dependiendo de los atributos que se anden buscando.

Parámetros:

- C : float, por defecto = 1.0.

Parámetro de regularización. La fuerza de la regularización es inversamente proporcional a C. Debe ser estrictamente positiva.

- Kernel : (‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’) or callable, por defecto =’rbf’.

  Parámetro utilizado para especificar el tipo de kernel que se utilizará en el algoritmo. Si no se proporciona ninguno, se utilizará 'rbf'. Si se proporciona un ‘‘callable’’, se utiliza para calcular previamente la matriz del kernel a partir de matrices de datos; esa matriz debe ser una matriz de forma .(n_samples, n_samples).

- Degree : int, por defecto = 3.

  Grado de la función kernel polinomial (‘‘poly’’).

- Gamma: (‘scale’, ‘auto’) or float, por defecto = ’scale’

  Coeficiente kernel para ‘‘rbf ’’, ‘‘poly’’ y ‘‘sigmoid’’.

  Si gamma = ‘‘scale’’ (predeterminado), entonces usa 1 / (n_features * X.var()) como valor de gamma.

  Si es 'auto', usa 1 / n_features.

- coef0 : float, por defecto = 0.0.

  Término independiente en la función kernel. Solo es significativo en 'poli' y 'sigmoide'.

- shrinking : bool, por defecto = True.

  Es un parámetro utilizado para la reducción del tiempo de entrenamiento aumentando la tolerancia.

- probability : bool, por defecto = False.

  Parámetro que cuando se activa, probability = true, calcula las estimaciones de probabilidad al crear una clase de support vector machine, esta se utiliza antes de la librería .fit.

- tol : float, por defecto = 1e-3.

  Tolerancia para el criterio de parada.

- cache_size : float, por defecto = 200.

  Especifica el tamaño de la memoria caché del kernel (en MB).

- class_weight : dict or ‘balanced’, por defecto = None.

Establece el parámetro C de la clase i en class_weight [i] * C para support vector machine. Si no se proporciona, se supone que todas las clases tienen peso uno. El modo ‘‘equilibrado’’ utiliza los valores para ajustar automáticamente los pesos, que son inversamente proporcionales a las frecuencias de clase en los datos de entrada como: n_samples / (n_classes * np.bincount(y)).

- verbose : bool, por defecto = False.
  Controla la verbosidad al ajustar y predecir. La verbosidad significa mostrar más información ‘‘prolija’’ para la tarea. Mientras más alto sea el número más información se informará.
- max_iter : int, por defecto = -1.
  Límite estricto en las iteraciones dentro del solucionador, cuando es -1 no tiene ningún límite de iteraciones.
- decision_function_shape : (‘ovo’, ‘ovr’), por defecto = ’ovr’.
  Evalúa la función de decisión para las muestras en X.
  Si decision_function_shape = 'ovo', los valores de la función son proporcionales a la distancia de las muestras X al hiperplano de separación.
  Si decision_function_shape = 'ovr', la función de decisión es una transformación monótona de la función de decisión ovo.
- break_ties : bool, por defecto = False.
  Si es verdadero, decision_function_shape='ovr'y el número de clases > 2, predecir romperá los empates de acuerdo con los valores de confianza de decision_function; de lo contrario, se devuelve la primera clase entre las clases vinculadas. Hay que tener en cuenta que romper empates tiene un costo computacional relativamente alto en comparación con una predicción simple.

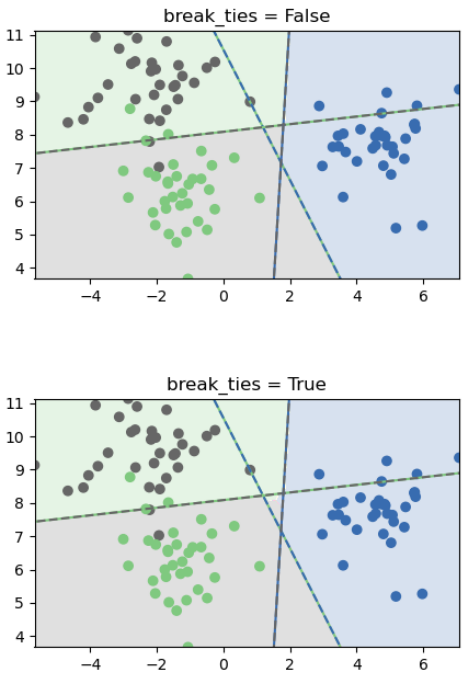


Figura 3.31. Comparación break_ties = False y break_ties = True. Fuente:[42].

Las dos parcelas difieren solo en el área del medio donde las clases están empatadas. Si break_ties=False, todas las entradas en esa área se clasificarían como una clase, mientras que si break_ties=True, el mecanismo de desempate creará un límite de decisión no convexo en esa área.

- random_state : int, RandomState instance or None, por defecto = None.
  Controla la generación de números pseudoaleatorios para mezclar los datos para estimaciones de probabilidad. Ignorado cuando probabilityes Falso. Pase un int para una salida reproducible en múltiples llamadas a funciones.

3.1.8.3 Algoritmo de entrenamiento

Al igual que en todos los algoritmos anteriores, el algoritmo de entrenamiento es el mismo para la posterior comparación de datos.

```
1. from sklearn.model_selection import train_test_split
2.
3. X_train, X_test, y_train, y_test = train_test_split(X, Y, tes
   t_size=0.2) #Divide los datos entre entrenamiento y testeo
```

Donde:

X: Pertenece a todos los datos entregados por la página web sobre el engranaje mencionados en la Tabla 2.1

Y: Conjunto de datos etiquetados para cada estado del engranaje.

3.1.8.4 Algoritmo de clasificación

Para el algoritmo de clasificación se utilizará la biblioteca que nos entrega Sklearn para el método de clasificación de SVM.

Dentro del algoritmo es posible modificar distintos parámetros para la creación de una clasificación más personalizada según los datos que tengamos a disposición.

El primer parámetro y el más importante es el *Kernel*, como ya se habló con anterioridad, este puede tomar 4 valores predeterminados, que fueron determinados para aumentar la cantidad de dimensiones y a la vez facilitar la separación de los datos, como se pudo ver en la Figura 3.29.

Dentro de las funciones de kernel, se tiene la función base, la cual es la lineal, que en sklearn se utiliza su traducción al inglés *''linear''*. Luego está la segunda función, que es la polinómica que se denomina en la librería como *''poly''*, como tercera función está aquella que dentro del código está como default que es *''rbf''* el cual hace referencia a la base radial y la que se ocupó en el ejemplo de la Figura 3.30. Por último, está la *''sigmoid''* la cual hace referencia al método de sigmoide.

Las funciones polinomiales y de base radial son útiles para el hiperplano no lineal, los núcleos polinomiales y *''rbf''* calculan las líneas de separación en la dimensión superior obteniendo resultados más precisos.

El siguiente parámetro por evaluar será C, el cual tiene como característica que es utilizado para regularizar, el cual dentro de la librería es utilizado como un parámetro de penalización. C representa una clasificación errónea o un término de error. La clasificación errónea o término de error le indica a la optimización del algoritmo, cuanto error es soportable. Esto trae como consecuencia que mientras mayor sea C, mayor será el error

aceptado, por lo que el modelo tendera a ser más perfecto, aumentando el Acurracy de este mismo. Un valor menor en C crea un hiperplano con un margen pequeño y un C mayor agrandará este margen.

Por defecto, la librería asume este C como un valor 1.0, pudiendo ser modificado por el usuario.

Estos son los más utilizados y de mayor impacto a la hora de realizar el algoritmo, pero existen otros parámetros que pueden ser modificados para resultados más específicos.

En el caso concreto del kernel, al utilizar el modelo polinomial *''poly''* y como ya vinos con anterioridad este lleva una exponencial dentro de la ecuación, esta exponencial puede ser la que el usuario desee, para esto hay que modificar el parámetro de *''grado''*, por defecto esta es 3, pudiendo ser alterada a un *n* deseado.

Por último, se tiene el parámetro *''gamma''*, este es aplicado en el coeficiente de kernel a *''rbf''*, *''poly''* y *''sigmoid''*, lo que a diferencia del *''grado''*, este es aplicado a todos los kerneles, menos al lineal.

Este parámetro *''gamma''*, es un parámetro que defina hasta donde debe llegar la influencia de un solo entrenamiento, limitando que es lo que se define como lejos o cerca.

Si *''gamma''* es demasiado grande, el radio del área de influencia de los vectores de soporte será nulo, por lo que, al aplicar la constante C, que nos ajusta, un error aceptable no servirá en absoluto debido a que se considerará al valor testeado como un solo objeto sin nada a su alrededor.

Por otro lado, si *''gamma''* es muy pequeño, lo que ocurrirá será que será muy restringido y no podrá distinguir la complejidad de los datos, esto quiere decir que no distinguirá si está cerca o lejos o si pertenece a una clase u otro debido a que incluirá a todo el conjunto de entrenamiento.

C y *''gamma''*, son valores que uno dependen del otro, esto queda demostrado en un ejemplo que realizo sklearn [43], donde explican la importancia de estos dos parámetros y que ocurre con la exactitud del modelo cuando estos valores se llevan al extremo.

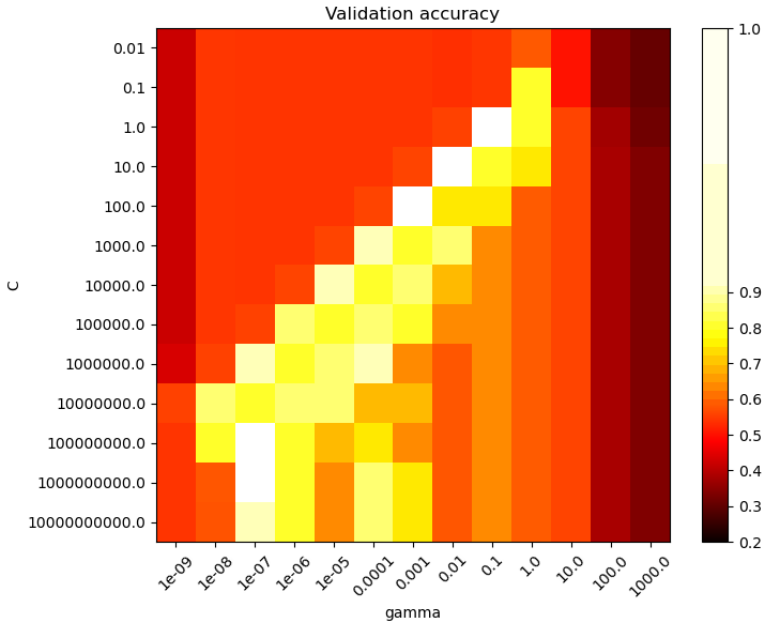


Figura 3.32. Comparación entre C y gamma. Fuente: [43].

Como se puede apreciar en la Figura 3.32, los mejores resultados sé observar para los valores medios, si se llega a ir a algún extremo por alguna de las dos partes, esta tiende a disminuir la exactitud del modelo afectando por completo todo el trabajo realizado.

3.1.8.5 Resultados

Para testear los diferentes resultados, se realizará la prueba para los diferentes kerneles y ver la diferencia de cada uno en conjunto con el parámetro C y gamma.

Caso 1:

En primera instancia se utilizará como parámetros kernel predeterminado, o sea *‘‘rbf’’’*, un C=5 y gamma por defecto (gamma = ‘‘scale’’), esto quiere decir, para calcular gamma se manejará la siguiente ecuación.

$$gamma\ (scale) = \frac{1}{n_{features} * X.var}$$

Donde:

$n_{features}$: cantidad de etiquetas o posibles fallas del engranaje.

$X.var$: La varianza de la matriz X (donde X es la matriz de entrada de todas las vibraciones)

Tabla 3.6. KPI por método de falla.

	Precisión	recall	f1-score	support
Normal	0.99	0.96	0.98	84
Diente Faltante	0.95	1.00	0.98	81
Diente Roto	1.00	0.97	0.98	86
Diente Picado	0.98	1.00	0.99	87
Diente Astillado 1	1.00	1.00	1.00	77
Diente Astillado 2	1.00	1.00	1.00	83
Diente Astillado 3	1.00	1.00	1.00	74
Diente Astillado 4	1.00	1.00	1.00	74
Diente Astillado 5	1.00	1.00	1.00	74
Acurracy			0.99	720
Macro avg	0.99	0.99	0.99	720
weigted avg	0.99	0.99	0.99	720

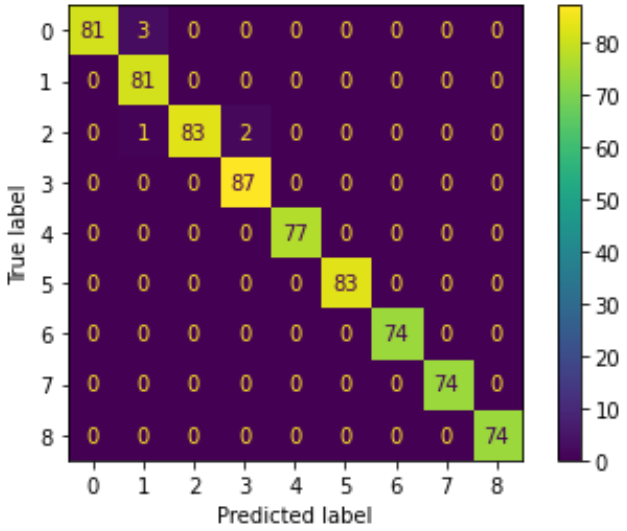


Figura 3.33. Matriz de confusión. Datos predeterminados, con un C=5. Fuente: Elaboración propia.

Ahora bien , para los mismos parámetros se variará C a un número muy grande a ver qué es lo que pasa con los resultados, para esto se utilizará un C = 1000. Esto con la finalidad de poder comprobar la teoría que fue explicada con anterioridad poniendo en práctica lo visto en la Figura 3.32, comprando los parámetros de gamma y C. Como gamma, depende de los valores establecidos, se dejará constante y solo se variará el parámetro C.

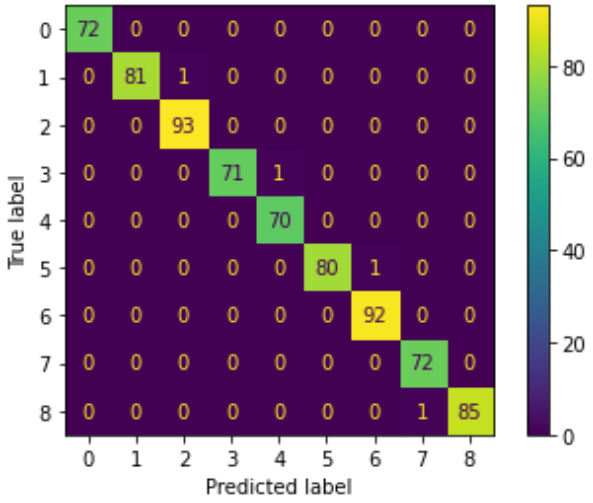


Figura 3.34. Matriz de confusión. Datos predeterminados, con un C=1000. Fuente: Elaboración propia.

Tabla 3.7. KPI por método de falla.

	Precisión	recall	f1-score	support
Normal	1.00	1.00	1.00	72
Diente Faltante	1.00	0.99	0.99	82
Diente Roto	0.99	1.00	0.99	93
Diente Picado	1.00	0.99	0.99	72
Diente Astillado 1	0.99	1.00	0.99	70
Diente Astillado 2	1.00	0.99	0.99	81
Diente Astillado 3	0.99	1.00	0.99	92
Diente Astillado 4	0.99	1.00	0.99	72
Diente Astillado 5	1.00	0.99	0.99	86
Acurracy			0.99	720
Macro avg	0.99	0.99	0.99	720
weigted avg	0.99	0.99	0.99	720

Como conclusión se esperaba una mejora con respecto a la exactitud del método debido a que, al aumentar el C, habría una mayor permisibilidad a la hora de determinar a qué falla pertenece. Si entramos al detalle de la Tabla 3.6 y Tabla 3.7 con respecto a los KPI, podemos encontrar que en ambos casos son perfectos la precisión es de un 99% en términos generales para ambos y el recall igualmente, por lo que quiere decir que existen muy pocos datos falsos positivos y aquellos positivos son clasificados con un 99% de efectividad, lo que nos indica una nula diferencia entre ambos.

Caso 2:

Para el segundo caso se conservará los valores predeterminados como se hizo en el caso 1. En el caso número 2 se variarán los distintos kernel partiendo por un kernel lineal hasta llegar al sigmoide, para ver si cambiando los kernel varían la exactitud del modelo.

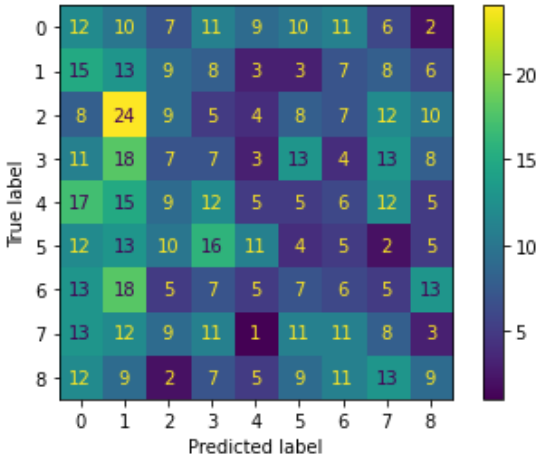


Figura 3.35. Matriz de confusión. Datos predeterminados, kernel lineal. Fuente: Elaboración propia.

Tabla 3.8. KPI por método de falla.

	Precisión	recall	f1-score	support
Normal	0.11	0.15	0.13	78
Diente Faltante	0.10	0.18	0.13	72
Diente Roto	0.13	0.10	0.12	87
Diente Picado	0.08	0.08	0.08	84
Diente Astillado 1	0.11	0.06	0.08	86
Diente Astillado 2	0.06	0.05	0.05	78
Diente Astillado 3	0.09	0.108	0.08	79
Diente Astillado 4	0.10	0.10	0.10	79
Diente Astillado 5	0.15	0.12	0.13	77
Acurracy			0.10	720
Macro avg	0.10	0.10	0.10	720
weigted avg	0.10	0.10	0.10	720

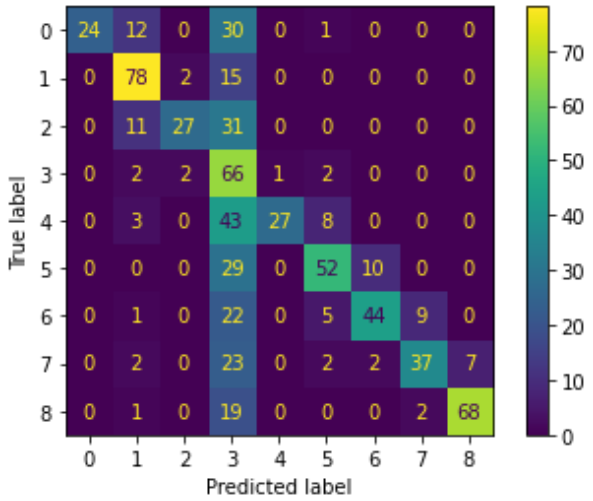


Figura 3.36. Matriz de confusión. Datos predeterminados, kernel polinomial. Fuente: Elaboración propia.

Tabla 3.9. KPI por método de falla.

	Precisión	recall	f1-score	support
Normal	1.00	0.36	0.53	67
Diente Faltante	0.10	0.18	0.13	95
Diente Roto	0.13	0.10	0.12	69
Diente Picado	0.08	0.08	0.08	73
Diente Astillado 1	0.11	0.06	0.08	81
Diente Astillado 2	0.06	0.05	0.05	91
Diente Astillado 3	0.09	0.108	0.08	81
Diente Astillado 4	0.10	0.10	0.10	73
Diente Astillado 5	0.15	0.12	0.13	90
Acurracy			0.59	720
Macro avg	0.78	0.58	0.60	720
weigted avg	0.78	0.59	0.61	720

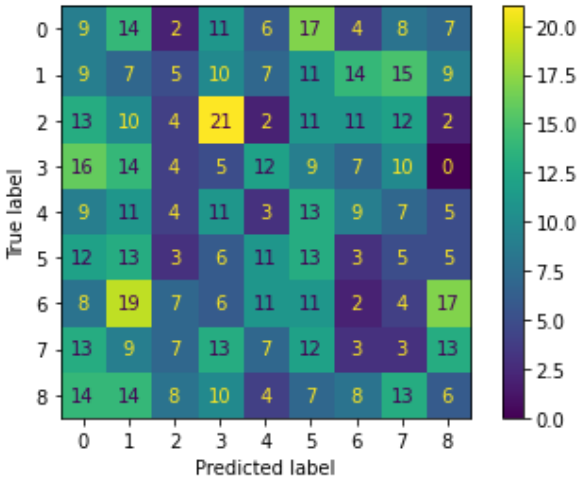


Figura 3.37. Matriz de confusión. Datos predeterminados, kernel sigmoide. Fuente: Elaboración propia.

Tabla 3.10. KPI por método de falla.

	Precisión	recall	f1-score	support
Normal	0.09	0.12	0.10	78
Diente Faltante	0.06	0.08	0.07	87
Diente Roto	0.09	0.05	0.06	86
Diente Picado	0.05	0.06	0.06	77
Diente Astillado 1	0.05	0.04	0.04	72
Diente Astillado 2	0.12	0.18	0.15	71
Diente Astillado 3	0.03	0.02	0.03	85
Diente Astillado 4	0.04	0.04	0.04	80
Diente Astillado 5	0.09	0.07	0.08	84
Acurracy			0.07	720
Macro avg	0.07	0.07	0.07	720
weigted avg	0.07	0.07	0.07	720

Para las diferentes situaciones se pudo observar cómo cambian los resultados dependiendo de las distintas alternativas que nos entrega el algoritmo de SVC, pudiendo ver

que hay muchos problemas a la hora de la exactitud del algoritmo dependiendo de que kernel se utilice.

Dentro de las diferentes simulaciones que se realizó en el caso número 2, se pudo encontrar que la mejor alternativa fue la que se presentó con anterioridad con un kernel de *‘‘rbf’’*, el cual se puede observar en la Figura 3.33, el cual según los datos arrojados en la Tabla 3.6, dio como resultado de un 99% de efectividad, con una precisión y recall de un 99% cada uno, lo que lo convierte en la mejor alternativa si se compara con los diferentes kernel que se simularon. Los casos con un kernel de ‘‘*sigmoide’’* y ‘‘*polinomial’’* ninguno llego a una exactitud parecida como la que se simulo con un kernel con el método de*‘‘rbf’’*.

3.1.8.6 Conclusiones

Tal y como hemos podido comprobar SVM, es un método muy amplio con una gran variedad de procesos previos según los datos que uno dispone. Como primer parámetro tenemos una estimación lineal, la cual separa los datos de manera equitativa por medio de un margen creando los vectores de soporte. Este método es muy poco práctico, debido a que por lo general solo sirve para dos muestras y siempre y cuando estas estén bien separadas, por lo que no es un método que se recomiende para una gran cantidad de datos y etiquetas. En el caso del engranaje al ser una gran cantidad de datos y fallas, este método muy poco práctico como quedo expresado en la Tabla 3.8, pudiendo observar una exactitud del 10%.

Luego se pudo ver que los datos se podían separar mediante curvas y en diferentes planos pudiendo trabajar con datos muy complejos, siendo la variable más importante el kernel, el cual proponía diferentes métodos para calcular estas curvas y los diferentes planos tridimensionales para poder facilitar la separación de los datos. Esta fue la mejor alternativa que se le pudo dar a la clasificación de los engranajes, debido a que se pudo obtener uno de los mejores resultados posibles, siendo este un 99% de exactitud. Este porcentaje se debió a los diferentes parámetros obteniendo el más optimo dentro del grupo de los diferentes kernel.

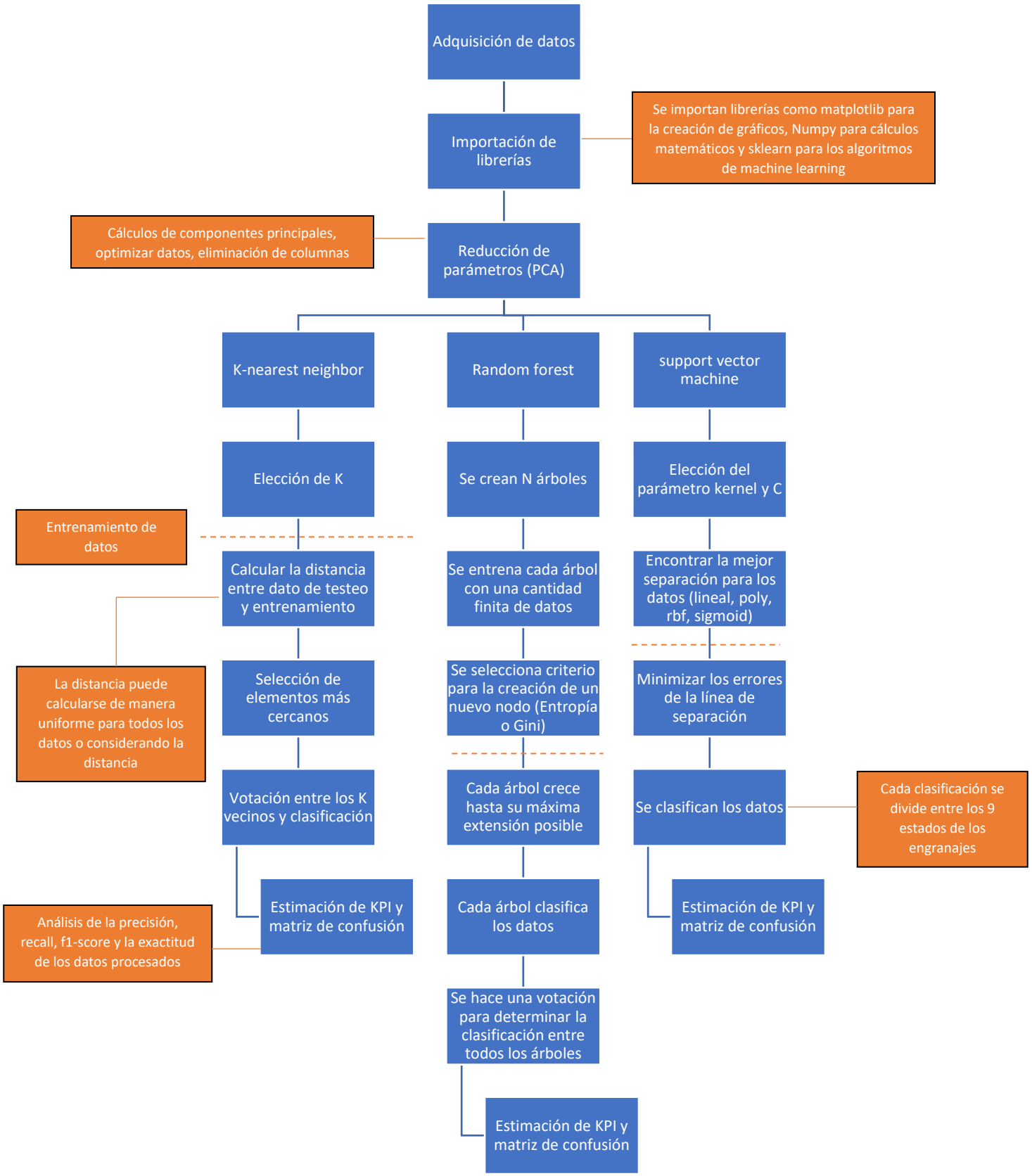
También se pudo observar la relación entre el gamma y la variable C, obteniendo un equilibrio entre las dos, observando que gamma dependía estrictamente de la varianza de los datos y la cantidad de fallos por clasificar. Por esta razón la única variable que había que

variar era C, donde se pudo observar que, aunque se aumentara la cantidad de C, ósea el error aceptable nunca era de un 100%, por lo que no era un factor muy relevante dentro del código.

Para finalizar este algoritmo de SVM es uno de los más utilizados y con gran razón debido al gran resultado que entrega de exactitud. Es un método que se compara estrictamente con redes neuronales debido a las buenas prestaciones del algoritmo [44], el cual sirve para usarse en clasificación de engranajes.

4. Diagrama de flujo

Para una mejor visualización de los procesos que componen los diferentes algoritmos ya vistos, se creará un diagrama de flujo con la finalidad de poder exponer de manera más compacta lo mostrado. Dando como inicio a la adquisición de datos, la importación de las librerías y la reducción de parámetros para posteriormente elegir el método a utilizar para el desarrollo del algoritmo de clasificación.


Adquisición de datos
Importación de librerías
Se importan librerías como matplotlib para la creación de gráficos, Numpy para cálculos matemáticos y sklearn para los algoritmos de machine learning
Cálculos de componentes principales, optimizar datos, eliminación de columnas
Reducción de parámetros (PCA)
K-nearest neighbor
Random forest
support vector machine
Elección de K
Se crean N árboles
Elección del parámetro kernel y C
Entrenamiento de datos
Calcular la distancia entre dato de testeo y entrenamiento
Se entrena cada árbol con una cantidad finita de datos
Encontrar la mejor separación para los datos (lineal, poly, rbf, sigmoid)
La distancia puede calcularse de manera uniforme para todos los datos o considerando la distancia
Selección de elementos más cercanos
Se selecciona criterio para la creación de un nuevo nodo (Entropía o Gini)
Minimizar los errores de la línea de separación
Votación entre los K vecinos y clasificación
Cada árbol crece hasta su máxima extensión posible
Se clasifican los datos
Cada clasificación se divide entre los 9 estados de los engranajes
Estimación de KPI y matriz de confusión
Cada árbol clasifica los datos
Estimación de KPI y matriz de confusión
Análisis de la precisión, recall, f1-score y la exactitud de los datos procesados
Se hace una votación para determinar la clasificación entre todos los árboles
Estimación de KPI y matriz de confusión

5. Conclusión

5.1 Introducción

En el presente trabajo, se ha presentado distintas maneras de enfrentar un problema que en el área de mantenimiento siempre tiene grandes repercusiones, que son la identificación de los diferentes fallos que puede presentar un equipo en concreto, en específico se hizo hincapié en los engranajes, los cuales tienen se subvinieron en 9 posibles fallos.

5.2 Conclusiones

Al iniciar este trabajo de investigación y desarrollo, el objetivo fue la elaboración de un código capaz de clasificar los distintos fallos que puede presentar un equipo crítico, como lo es un engranaje. Los diferentes algoritmos propuestos para la clasificación de fallas para un engranaje fueron muy asertivos a la hora de evaluar el modelo. Llegando a la conclusión que cualquiera de los 3 modelos presentados en la memoria es aptos para clasificar las diferentes fallas, recalcando que esta herramienta se diseñó con el objetivo de poder ser de ayuda en el área de mantenimiento y no un reemplazo definitivo a las diferentes alternativas que se utilizan en la industria. Esto debido a que las condiciones presentadas en el documento son realizadas experimentalmente bajo condiciones ideales.

Los análisis realizados en la memoria de título permitieron llegar a las siguientes conclusiones:

- Google Colaboratory, la herramienta en donde se desarrolló los diferentes algoritmos, tanto del procesamiento de datos y desarrollo de los diferentes métodos presentados en la memoria, dieron muy buenos resultados, siendo una interfaz amigable, entendible y por sobre todo, la utilización de RAM y GPU externa, el cual determinó que en ningún momento hubiera problemas de carga de datos o de procesamiento, ahorrando mucho tiempo con respecto a si se hubiera implementado en un programa local del computador.

  El aspecto menos recomendable y el cual hace mucha repercusión, es que las imágenes desarrolladas en Colab, son de muy mala resolución perdiendo mucha

calidad a la hora de descargarlas para poder utilizarlas. Pudiendo ser imágenes vectorizadas.

- Si bien Fourier es una herramienta útil, es una trasformada que representa los datos en el dominio del tiempo a un dominio donde prevalece la frecuencia, en si es lo mismo, pero aquí se puede observar datos que por otro medio como en el caso del tiempo no se logran apreciar. Esta trasformada por lo general necesita de una persona apta para la interpretación de los resultados. Un plus que tiene la transformada de Fourier es que se puede completar esta información con los parámetros estadísticos, pero aun así esto no garantiza que con estas herramientas se logren diferenciar las distintas fallas del engranaje en comparación con la exactitud que entregan los 3 modelos descritos.
- Machine learning es una herramienta super poderosa a la hora de desarrollar inteligencia artificial, y queda expresamente demostrado con los porcentajes de exactitud que presentaron los modelos desarrollados. Si bien redes neuronales es el siguiente paso por desarrollar es una muy buena base para trabajos posteriores lo desarrollado en la memoria de título.

5.3 Trabajos futuros

Como bien fue señalado en la conclusión, los trabajos futuros relacionados a la memoria de título van relacionados explícitamente con el desarrollo de otras áreas del machine learning, como lo es explícitamente redes neuronales, las cuales tienen la capacidad de clasificar los distintos fallos, aunque estos no estén etiquetados como fue en el caso desarrollado durante el documento.

Si bien existen muchas áreas dentro del extenso mundo del machine learning las más interesantes a investigar son el Deep learning o poder seguir dentro de la misma rama procesando otros tipos de elementos críticos bajo los mismos conceptos para una posible comparación a futuro, identificando virtudes y falencias dentro de los distintos códigos o métodos señalados, debido a que solo se desarrollaron 3 algoritmos de clasificación, pero existen otros capaces de poder hacer el mismo trabajo.

Una posible investigación muy interesante es aplicar el mismo algoritmo, pero con datos obtenidos en la universidad de Talca, en el simulador de fallas que se encuentra en el campus los niches. Pudiendo abrir diferentes debates sobre la adquisición y verificación del código para comprobar si es compatible con otras fallas de diferentes elementos críticos que se desee investigar. Pudiendo así verificar los datos obtenidos y la efectividad del código de clasificación, además de poder realizar comparaciones de la transformada de Fourier y los parámetros estadísticos.

De esta misma manera los datos obtenidos se trabajaron en el dominio del tiempo, si bien es una herramienta útil, se pudiera abrir la opción de poder realizar el mismo estudio, pero esta vez tomando el dominio de la frecuencia, investigar y ver principalmente los cambios generados a la hora de poder implementar los algoritmos. Como su nombre lo indica la transformada de Fourier es un cambio en el dominio del tiempo a la frecuencia, pero no se sabe si implementando este cambio pueden ocurrir distintos cambios que pudieran afectar de manera positiva o negativa a los algoritmos, por lo que abre un abanico de puertas a la hora de investigación.

Bibliografía

[1] O. Simeone, “A Very Brief Introduction to Machine Learning with Applications to Communication Systems,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 4, pp. 648–664, Dec. 2018, doi: 10.1109/TCCN.2018.2881442.

[2] “Funcionamiento del Cerebro e Inteligencia Artificial - IEEE Capítulo Argentino de la CIS.” https://r9.ieee.org/argentina-cis/inicio/jb1day/20211016_42/ (accessed Aug. 06, 2022).

[3] “IA en Google: ¿cómo funcionan los motores de búsqueda? - Jorge Pérez Colín.” https://blog.jorgeperezcolin.mx/google-inteligencia-artificial-motores-busqueda/ (accessed Aug. 06, 2022).

[4] “El uso de la Inteligencia artificial para mitigar los efectos de la pandemia del Covid-19 | CAF.” https://www.caf.com/es/conocimiento/visiones/2021/08/el-uso-de-la-inteligencia-artificial-para-mitigar-los-efectos-de-la-pandemia-del-covid19/ (accessed Mar. 27, 2022).

[5] “El machine learning (aprendizaje automático) - Arrizabalagauriarte Consulting.” https://arrizabalagauriarte.com/el-machine-learning-aprendizaje-automatico/ (accessed Jul. 23, 2022).

[6] S. Sagiroglu and D. Sinanc, “Big data: A review,” *Proc. 2013 Int. Conf. Collab. Technol. Syst. CTS 2013*, pp. 42–47, 2013, doi: 10.1109/CTS.2013.6567202.

[7] P. Ing and G. Sigüenza, “PRONOSTICO Y ESTIMACIÓN DE LA VIDA ÚTIL REMANENTE,” 2017.

[8] “Engranajes: Tipos y Calculos.” https://www.areatecnologia.com/mecanismos/engranajes.html (accessed Aug. 04, 2022).

[9] O. C. Morales, “Análisis tiempo-frecuencia de señales,” 2011.

[10] E. Giraldo, C. Verucchi, G. Acosta, and M. Ferrari, “Detection of misalignment in elastic couplings through fuzzy logic,” *2017 17th Work. Inf. Process. Control. RPIC 2017*, vol. 2017-Janua, pp. 1–6, Dec. 2017, doi: 10.23919/RPIC.2017.8214313.

[11] M. CONTRA VECINO, “Análisis de vibraciones de un wheelset con diferentes niveles de defecto,” 2015.

[12] “Vista de Diferencias en la estimación del coeficiente de curtosis en diferentes softwares estadísticos | e-Agronegocios.” https://revistas.tec.ac.cr/index.php/eagronegocios/article/view/4456/4046 (accessed Jul. 23, 2022).

[13] A. Huertas Mora, “Algoritmos de aprendizaje supervisado utilizando datos de monitoreo de condiciones: Un estudio para el pronóstico de fallas en máquinas,” pp. 1–77, 2020, [Online]. Available: https://repository.usta.edu.co/bitstream/handle/11634/29886/2020alexanderhuertas.p

df?sequence=1&isAllowed=y.

[14] SEBASTIÁN MONTAGNA PUGA, “DETECCIÓN DE FALLAS EN EQUIPOS UTILIZANDO MODELOS EN BASE A DEEP LEARNING,” vol. 7, no. 5, pp. 1–2, 2018, [Online]. Available: https://repositorio.uchile.cl/handle/2250/170010.

[15] P. Cao Graduate Research Assistant, S. Zhang Stanley Black, J. Tang, P. Cao, and S. Zhang, “Pre-Processing-Free Gear Fault Diagnosis Using Small Datasets with Deep Convolutional Neural Network-Based Transfer Learning.”

[16] C. Li, R. V. Sanchez, G. Zurita, M. Cerrada, D. Cabrera, and R. E. Vásquez, “Gearbox fault diagnosis based on deep random forest fusion of acoustic and vibratory signals,” *Mech. Syst. Signal Process.*, vol. 76–77, pp. 283–293, Aug. 2016, doi: 10.1016/J.YMSSP.2016.02.007.

[17] J. R. A. Montoya, “La transformada wavelet,” *Rev. la Univ. Mendoza*, 2001, [Online]. Available: //www.um.edu.ar/ojs2019/index.php/RUM/article/view/22.

[18] M. Graña *et al.*, “Experimentos de aprendizaje con Máquinas de Boltzmann de alto orden.”

[19] P. Cao, S. Zhang, and J. Tang, “Preprocessing-Free Gear Fault Diagnosis Using Small Datasets with Deep Convolutional Neural Network-Based Transfer Learning,” *IEEE Access*, vol. 6, pp. 26241–26253, 2018, doi: 10.1109/ACCESS.2018.2837621.

[20] “Ceteris paribus - Qué es, definición y concepto | 2022 | Economipedia.” https://economipedia.com/definiciones/ceteris-paribus.html (accessed Aug. 09, 2022).

[21] “scikit-learn: machine learning in Python — scikit-learn 1.1.1 documentation.” https://scikit-learn.org/stable/ (accessed Jul. 25, 2022).

[22] L. A. Norma and A. Agma, “Engranajes Conforme la norma ANSI/AGMA,” 2015.

[23] “Análisis de vibraciones en cajas de engranajes | Power-MI.” https://power-mi.com/es/content/análisis-de-vibraciones-en-cajas-de-engranajes-0 (accessed Aug. 22, 2022).

[24] “Engranes Gastados Causan Problemas en Envolvedoras Horizontales - Greener Corporation Knowledge Center - P3 Solutions Blog.” https://p3solutionsblog.com/es/engranes-gastados-causan-problemas-en-envolvedoras-horizontales/ (accessed Aug. 05, 2022).

[25] C. D. E. L. Curso, “Engranajes y sus fallas.”

[26] L. Eduardo and B. Hernandez, “Falla en los engranajes.”

[27] Raúl González Duque, “Para todos.”

[28] “Aprendizaje profundo: GPU frente a Google Colab | Hacia la ciencia de datos.” https://towardsdatascience.com/deep-learning-on-a-budget-450-egpu-vs-google-colab-494f9a2ff0db (accessed May 07, 2022).

[29] “Deep Learning: GPU vs Google Colab | Towards Data Science.” https://towardsdatascience.com/deep-learning-on-a-budget-450-egpu-vs-google-colab-494f9a2ff0db (accessed Jul. 23, 2022).

[30] “Figshare.” https://knowledge.figshare.com/about (accessed May 29, 2022).

[31] M. Zhang, D. Wei, and K. Wang, “An Order Analysis Based Second-Order Cyclic Function Technique for Planetary Gear Fault Detection,” *Proc. - 2017 Int. Conf. Sensing, Diagnostics, Progn. Control. SDPC 2017*, vol. 2017-December, pp. 678–682, Dec. 2017, doi: 10.1109/SDPC.2017.134.

[32] A. Gomes Pereira, J. V. Scanavini Chiaradia, A. Í. Sette Antonialli, and S. B. Shiki, “INVESTIGAÇÃO NUMÉRICA DO CONTROLE PASSIVO DE CHATTER NO TORNEAMENTO DE LIGAS TI-6AL-4V A PARTIR DO DISPOSITIVO SHUNT-PIEZOELÉTRICO,” Jun. 2021, doi: 10.26678/ABCM.COBEF2021.COB21-0170.

[33] “Detección de anomalías con PCA y python.” https://www.cienciadedatos.net/documentos/py21-deteccion-anomalias-pca-python.html (accessed Jul. 23, 2022).

[34] “Reducción de la dimensionalidad: Análisis de Componentes Principales (PCA) | profesorDATA.com.” https://profesordata.com/2020/09/01/reduccion-de-la-dimensionalidad-analisis-de-componentes-principales-pca/ (accessed Jul. 23, 2022).

[35] H. Munaga and V. Jarugumalli, “Performance Evaluation : Ball-Tree and KD-Tree in the context of MST.”

[36] “Tree algorithms explained: Ball Tree Algorithm vs. KD Tree vs. Brute Force | by Hucker Marius | Towards Data Science.” https://towardsdatascience.com/tree-algorithms-explained-ball-tree-algorithm-vs-kd-tree-vs-brute-force-9746debcd940 (accessed Jul. 31, 2022).

[37] “Manhattan distance.” https://iq.opengenus.org/manhattan-distance/ (accessed Jul. 31, 2022).

[38] “Distancia euclidiana: concepto, fórmula, cálculo, ejemplo.” https://www.lifeder.com/distancia-euclidiana/ (accessed Jul. 31, 2022).

[39] “Out of Bag Score | OOB Score Random Forest Machine Learning.” https://www.analyticsvidhya.com/blog/2020/12/out-of-bag-oob-score-in-the-random-forest-algorithm/ (accessed Aug. 01, 2022).

[40] “Árboles de decisión: Gini vs Entropía | Quantdare.” https://quantdare.com/decision-trees-gini-vs-entropy/ (accessed Jul. 01, 2022).

[41] T. Liu, L. Jin, C. Zhong, and F. Xue, “Study of thermal sensation prediction model based on support vector classification (SVC) algorithm with data preprocessing,” *J. Build. Eng.*, vol. 48, p. 103919, May 2022, doi: 10.1016/J.JOBE.2021.103919.

[42] “SVM Tie Breaking Example — scikit-learn 1.1.1 documentation.” https://scikit-learn.org/stable/auto_examples/svm/plot_svm_tie_breaking.html (accessed Aug. 01,

2022).

[43] “RBF SVM parameters — scikit-learn 1.1.1 documentation.” https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html (accessed Jul. 07, 2022).

[44] E. A. Galindo, J. A. Perdomo, J. C. Figueroa-García, E. A. Galindo, J. A. Perdomo, and J. C. Figueroa-García, “Estudio comparativo entre máquinas de soporte vectorial multiclase, redes neuronales artificiales y sistema de inferencia neuro-difuso auto organizado para problemas de clasificación,” *Inf. tecnológica*, vol. 31, no. 1, pp. 273–286, Feb. 2020, doi: 10.4067/S0718-07642020000100273.

6. Anexos

6.1 Código análisis de vibraciones

```
#vector de tiempo
Fs=1000 #Hz
dt=1/Fs #paso de tiempo
N=len(Normal)
t=np.linspace(0,dt*(N-1),N)

#Graficar caso Normal respecto al tiempo
plt.plot(t,Normal[:,1])
plt.xlabel('Tiempo (s)', fontsize=14)
    plt.ylabel('Aceleración ' "$(m/s^2)$", fontsize=14)
    plt.title('Estado Sano', fontsize=14)
    plt.xlim(0,3.6)
    plt.ylim(-1,1)
    plt.show()


    #Graficar caso Diente faltante respecto al tiempo
    plt.plot(t,Diente_Faltante[:,1])
    plt.xlabel('Tiempo (s)', fontsize=14)
    plt.ylabel('Aceleración ' "$(m/s^2)$", fontsize=14)
    plt.title('Diente Faltante', fontsize=14)
    plt.xlim(0,3.6)
    plt.ylim(-1,1)
    plt.show()

    #Graficar caso Diente faltante respecto al tiempo
    plt.plot(t,Diente_Roto[:,1])
    plt.xlabel('Tiempo (s)', fontsize=14)
    plt.ylabel('Aceleración ' "$(m/s^2)$", fontsize=14)
    plt.title('Diente Roto', fontsize=14)
    plt.xlim(0,3.6)
    plt.ylim(-1,1)
    plt.show()

    #Graficar caso Diente faltante respecto al tiempo
    plt.plot(t,Diente_Picado[:,1])
    plt.xlabel('Tiempo (s)', fontsize=14)
    plt.ylabel('Aceleración ' "$(m/s^2)$", fontsize=14)
    plt.title('Diente Picado', fontsize=14)
    plt.xlim(0,3.6)
    plt.ylim(-1,1)
    plt.show()

    #Graficar caso Diente faltante respecto al tiempo
```

```
    plt.plot(t,Diente_Astillado_1[:,1])
    plt.xlabel('Tiempo (s)', fontsize=14)
    plt.ylabel('Aceleración ' "$(m/s^2)$", fontsize=14)
    plt.title('Diente astillado 1', fontsize=14)
    plt.xlim(0,3.6)
    plt.ylim(-1,1)
    plt.show()

    #Graficar caso Diente faltante respecto al tiempo
    plt.plot(t,Diente_Astillado_2[:,1])
    plt.xlabel('Tiempo (s)', fontsize=14)
    plt.ylabel('Aceleración ' "$(m/s^2)$", fontsize=14)
    plt.title('Diente astillado 2', fontsize=14)
    plt.xlim(0,3.6)
    plt.ylim(-1,1)
    plt.show()

    #Graficar caso Diente faltante respecto al tiempo
    plt.plot(t,Diente_Astillado_3[:,1])
    plt.xlabel('Tiempo (s)', fontsize=14)
    plt.ylabel('Aceleración ' "$(m/s^2)$", fontsize=14)
    plt.title('Diente astillado 3', fontsize=14)
    plt.xlim(0,3.6)
    plt.ylim(-1,1)
    plt.show()

    #Graficar caso Diente faltante respecto al tiempo
    plt.plot(t,Diente_Astillado_4[:,1])
    plt.xlabel('Tiempo (s)', fontsize=14)
    plt.ylabel('Aceleración ' "$(m/s^2)$", fontsize=14)
    plt.title('Diente astillado 4', fontsize=14)
    plt.xlim(0,3.6)
    plt.ylim(-1,1)
    plt.show()

    #Graficar caso Diente faltante respecto al tiempo
    plt.plot(t,Diente_Astillado_5[:,1])
    plt.xlabel('Tiempo (s)', fontsize=14)
    plt.ylabel('Aceleración ' "$(m/s^2)$", fontsize=14)
    plt.title('Diente astillado 5', fontsize=14)
    plt.xlim(0,3.6)
    plt.ylim(-1,1)
    plt.show()
```

6.2 Código de transformada de Fourier

```
#importar librerías
import scipy.io as sio
```

```
3. import numpy as np
4. import matplotlib.pyplot as plt
5. import math
6. from scipy.fftpack import fft, fftfreq
7. from numpy import mean, sqrt, square
8. from scipy import signal
9. import matplotlib.pyplot as plt
10.     import numpy as np
11.     from numpy import pi
12.
13.     #Transformada de Fourier
14.     Fs=1000 #Hz
15.     dt=1/Fs #paso de tiempo
16.     N=3600
17.     t=np.linspace(0,dt*(N-1),N)
18.
19.     #Dividir por segmentos
20.     L=1000 #largo de los segmentos
21.     l=0 #overlap
22.     Nt=math.floor((N-l)/(L-l)) #total de segmentos
23.
24.
25.
26.     ventana=np.hanning(L-1) #otras ventanas: hamming(L-1),
   blackman(L-1),bartlett(L-1), kaiser(L-1, beta)
27.
28.     for i in range(1,Nt+1):
29.         inicio=(i-1)*L-(i-1)*l+1
30.         fin=i*L-(i-1)*l
31.
32.
33.         Fn = fft(Normal_1[inicio:fin,0]*ventana)[1:int(L/2)]
   /(L/2)
34.         Ff = fft(Diente_Faltante_1[inicio:fin,0]*ventana)[1:
   int(L/2)]/(L/2)
35.         Fr = fft(Diente_Roto_1[inicio:fin,0]*ventana)[1:int(
   L/2)]/(L/2)
36.         Fp = fft(Diente_Picado_1[inicio:fin,0]*ventana)[1:in
   t(L/2)]/(L/2)
37.         Fa = fft(Diente_Astillado_11[inicio:fin,0]*ventana)[
   1:int(L/2)]/(L/2)
38.         Fa2 = fft(Diente_Astillado_22[inicio:fin,0]*ventana)
   [1:int(L/2)]/(L/2)
39.         Fa3 = fft(Diente_Astillado_33[inicio:fin,0]*ventana)
   [1:int(L/2)]/(L/2)
40.         Fa4 = fft(Diente_Astillado_44[inicio:fin,0]*ventana)
   [1:int(L/2)]/(L/2)
```

```
41.         Fa5 = fft(Diente_Astillado_55[inicio:fin,0]*ventana)
   [1:int(L/2)]/(L/2)
42.         frq = fftfreq(L,dt)[1:int(L/2)]
43.
44.     f, ax1 = plt.subplots(sharex='col',figsize=(9,3))
45.     ax1.plot(frq,abs(Fn))
46.     ax1.set(xlabel='Frecuencia (Hz)')
47.     ax1.set(ylabel='Amplitud')
48.     ax1.set_title('Fourier - Saludable', fontsize=14)
49.     plt.xlim(0,500)
50.     plt.ylim(0,0.06)
51.     plt.show()
52.
53.     f, ax2 = plt.subplots(sharex='col',figsize=(9,3))
54.     ax2.plot(frq,abs(Ff))
55.     ax2.set(xlabel='Frecuencia (Hz)')
56.     ax2.set(ylabel='Amplitud')
57.     ax2.set_title('Fourier - Diente Faltante', fontsize=14)
58.     plt.xlim(0,500)
59.     plt.ylim(0,0.06)
60.     plt.show()
61.
62.     f, ax3 = plt.subplots(sharex='col',figsize=(9,3))
63.     ax3.plot(frq,abs(Fr))
64.     ax3.set(xlabel='Frecuencia (Hz)')
65.     ax3.set(ylabel='Amplitud')
66.     ax3.set_title('Fourier - Diente Roto', fontsize=14)
67.     plt.xlim(0,500)
68.     plt.ylim(0,0.06)
69.     plt.show()
70.
71.     f, ax4 = plt.subplots(sharex='col',figsize=(9,3))
72.     ax4.plot(frq,abs(Fp))
73.     ax4.set(xlabel='Frecuencia (Hz)')
74.     ax4.set(ylabel='Amplitud')
75.     ax4.set_title('Fourier - Diente Picado', fontsize=14)
76.     plt.xlim(0,500)
77.     plt.ylim(0,0.06)
78.     plt.show()
79.
80.     f, ax5 = plt.subplots(sharex='col',figsize=(9,3))
81.     ax5.plot(frq,abs(Fa))
82.     ax5.set(xlabel='Frecuencia (Hz)')
83.     ax5.set(ylabel='Amplitud')
84.     ax5.set_title('Fourier - Diente Astillado
   1', fontsize=14)
85.     plt.xlim(0,500)
86.     plt.ylim(0,0.06)
```

```
87.     plt.show()
88.
89.     f, ax6 = plt.subplots(sharex='col',figsize=(9,3))
90.     ax6.plot(frq,abs(Fa2))
91.     ax6.set(xlabel='Frecuencia (Hz)')
92.     ax6.set(ylabel='Amplitud')
93.     ax6.set_title('Fourier - Diente Astillado
   2', fontsize=14)
94.     plt.xlim(0,500)
95.     plt.ylim(0,0.06)
96.     plt.show()
97.
98.     f, ax7 = plt.subplots(sharex='col',figsize=(9,3))
99.     ax7.plot(frq,abs(Fa3))
100.    ax7.set(xlabel='Frecuencia (Hz)')
101.    ax7.set(ylabel='Amplitud')
102.    ax7.set_title('Fourier - Diente Astillado
   3', fontsize=14)
103.    plt.xlim(0,500)
104.    plt.ylim(0,0.06)
105.    plt.show()
106.
107.    f, ax8 = plt.subplots(sharex='col',figsize=(9,3))
108.    ax8.plot(frq,abs(Fa4))
109.    ax8.set(xlabel='Frecuencia (Hz)')
110.    ax8.set(ylabel='Amplitud')
111.    ax8.set_title('Fourier - Diente Astillado
   4', fontsize=14)
112.    plt.xlim(0,500)
113.    plt.ylim(0,0.06)
114.    plt.show()
115.
116.    f, ax9 = plt.subplots(sharex='col',figsize=(9,3))
117.    ax9.plot(frq,abs(Fa5))
118.    ax9.set(xlabel='Frecuencia (Hz)')
119.    ax9.set(ylabel='Amplitud')
120.    ax9.set_title('Fourier - Diente Astillado
   5', fontsize=14)
121.    plt.xlim(0,500)
122.    plt.ylim(0,0.06)
123.    plt.savefig("Correlaciones
   Pearson.jpg", bbox_inches='tight')
124.    plt.show()
```

6.3 Código de parámetros estadísticos

```
1.
   #calcular parámetros por tramos
2. L=1800 #largo de los segmentos
3. l=0
4. Nt=math.floor((N-l)/(L-l)) #total de tramos
5.
6.
7. #inicializar matrices con parámetros
8. Pn=np.zeros((Nt,8))
9. Pf=np.zeros((Nt,8))
10.      Pr=np.zeros((Nt,8))
11.      Pp=np.zeros((Nt,8))
12.      Pa=np.zeros((Nt,8))
13.      Pa2=np.zeros((Nt,8))
14.      Pa3=np.zeros((Nt,8))
15.      Pa4=np.zeros((Nt,8))
16.      Pa5=np.zeros((Nt,8))
17.
18.
19.      for i in range(1,Nt+1):
20.          inicio=(i-1)*L-(i-1)*l+1
21.          fin=i*L-(i-1)*l
22.
23.      Pn[i-1,0]=sqrt(mean(square(Normal[inicio:fin]))) #RMS
24.      Pn[i-1,1]=np.amax(Normal[inicio:fin]) #Peak
25.      Pn[i-1,2]=np.amax(Normal[inicio:fin])-
   np.amin(Normal[inicio:fin]) #peak-peak
26.      Pn[i-1,3]=Pn[i-1,1]/Pn[i-1,0] #crest
27.      Pn[i-1,4]=np.mean(Normal[inicio:fin]) #Media
28.      Pn[i-1,5]=np.var(Normal[inicio:fin]) #var
29.      Pn[i-1,6]=skew(Normal[inicio:fin])[0] #asimetria
30.      Pn[i-1,7]=kurtosis(Normal[inicio:fin])[0] #curtosis
31.
32.      Pf[i-
   1,0]=sqrt(mean(square(Diente_Faltante[inicio:fin]))) #RMS
33.      Pf[i-1,1]=np.amax(Diente_Faltante[inicio:fin]) #Peak
34.      Pf[i-1,2]=np.amax(Diente_Faltante[inicio:fin])-
   np.amin(Diente_Faltante[inicio:fin]) #peak-peak
35.      Pf[i-1,3]=Pf[i-1,1]/Pf[i-1,0] #crest
36.      Pf[i-1,4]=np.mean(Diente_Faltante[inicio:fin]) #Media
37.      Pf[i-1,5]=np.var(Diente_Faltante[inicio:fin]) #var
38.      Pf[i-
   1,6]=skew(Diente_Faltante[inicio:fin])[0] #asimetria
39.      Pf[i-
   1,7]=kurtosis(Diente_Faltante[inicio:fin])[0] #curtosis
40.
```

```
41.      Pr[i-
   1,0]=sqrt(mean(square(Diente_Roto[inicio:fin]))) #RMS
42.      Pr[i-1,1]=np.amax(Diente_Roto[inicio:fin]) #Peak
43.      Pr[i-1,2]=np.amax(Diente_Roto[inicio:fin])-
   np.amin(Diente_Roto[inicio:fin]) #peak-peak
44.      Pr[i-1,3]=Pr[i-1,1]/Pr[i-1,0] #crest
45.      Pr[i-1,4]=np.mean(Diente_Roto[inicio:fin]) #Media
46.      Pr[i-1,5]=np.var(Diente_Roto[inicio:fin]) #var
47.      Pr[i-1,6]=skew(Diente_Roto[inicio:fin])[0] #asimetria
48.      Pr[i-1,7]=kurtosis(Diente_Roto[inicio:fin])[0] #curtosis
49.
50.      Pp[i-
   1,0]=sqrt(mean(square(Diente_Picado[inicio:fin]))) #RMS
51.      Pp[i-1,1]=np.amax(Diente_Picado[inicio:fin]) #Peak
52.      Pp[i-1,2]=np.amax(Diente_Picado[inicio:fin])-
   np.amin(Diente_Picado[inicio:fin]) #peak-peak
53.      Pp[i-1,3]=Pp[i-1,1]/Pp[i-1,0] #crest
54.      Pp[i-1,4]=np.mean(Diente_Picado[inicio:fin]) #Media
55.      Pp[i-1,5]=np.var(Diente_Picado[inicio:fin]) #var
56.      Pp[i-1,6]=skew(Diente_Picado[inicio:fin])[0] #asimetria
57.      Pp[i-
   1,7]=kurtosis(Diente_Picado[inicio:fin])[0] #curtosis
58.
59.      Pa[i-
   1,0]=sqrt(mean(square(Diente_Astillado_1[inicio:fin]))) #RMS
60.      Pa[i-1,1]=np.amax(Diente_Astillado_1[inicio:fin]) #Peak
61.      Pa[i-1,2]=np.amax(Diente_Astillado_1[inicio:fin])-
   np.amin(Diente_Astillado_1[inicio:fin]) #peak-peak
62.      Pa[i-1,3]=Pa[i-1,1]/Pa[i-1,0] #crest
63.      Pa[i-1,4]=np.mean(Diente_Astillado_1[inicio:fin]) #Media
64.      Pa[i-1,5]=np.var(Diente_Astillado_1[inicio:fin]) #var
65.      Pa[i-
   1,6]=skew(Diente_Astillado_1[inicio:fin])[0] #asimetria
66.      Pa[i-
   1,7]=kurtosis(Diente_Astillado_1[inicio:fin])[0] #curtosis
67.
68.      Pa2[i-
   1,0]=sqrt(mean(square(Diente_Astillado_2[inicio:fin]))) #RMS
69.      Pa2[i-1,1]=np.amax(Diente_Astillado_2[inicio:fin]) #Peak
70.      Pa2[i-1,2]=np.amax(Diente_Astillado_2[inicio:fin])-
   np.amin(Diente_Astillado_2[inicio:fin]) #peak-peak
71.      Pa2[i-1,3]=Pa2[i-1,1]/Pa2[i-1,0] #crest
72.      Pa2[i-
   1,4]=np.mean(Diente_Astillado_2[inicio:fin]) #Media
73.      Pa2[i-1,5]=np.var(Diente_Astillado_2[inicio:fin]) #var
74.      Pa2[i-
   1,6]=skew(Diente_Astillado_2[inicio:fin])[0] #asimetria
```

```
75.      Pa2[i-
   1,7]=kurtosis(Diente_Astillado_2[inicio:fin])[0] #curtosis
76.
77.      Pa3[i-
   1,0]=sqrt(mean(square(Diente_Astillado_3[inicio:fin]))) #RMS
78.      Pa3[i-1,1]=np.amax(Diente_Astillado_3[inicio:fin]) #Peak
79.      Pa3[i-1,2]=np.amax(Diente_Astillado_3[inicio:fin])-
   np.amin(Diente_Astillado_3[inicio:fin]) #peak-peak
80.      Pa3[i-1,3]=Pa3[i-1,1]/Pa3[i-1,0] #crest
81.      Pa3[i-
   1,4]=np.mean(Diente_Astillado_3[inicio:fin]) #Media
82.      Pa3[i-1,5]=np.var(Diente_Astillado_3[inicio:fin]) #var
83.      Pa3[i-
   1,6]=skew(Diente_Astillado_3[inicio:fin])[0] #asimetria
84.      Pa3[i-
   1,7]=kurtosis(Diente_Astillado_3[inicio:fin])[0] #curtosis
85.
86.      Pa4[i-
   1,0]=sqrt(mean(square(Diente_Astillado_4[inicio:fin]))) #RMS
87.      Pa4[i-1,1]=np.amax(Diente_Astillado_4[inicio:fin]) #Peak
88.      Pa4[i-1,2]=np.amax(Diente_Astillado_4[inicio:fin])-
   np.amin(Diente_Astillado_4[inicio:fin]) #peak-peak
89.      Pa4[i-1,3]=Pa4[i-1,1]/Pa4[i-1,0] #crest
90.      Pa4[i-
   1,4]=np.mean(Diente_Astillado_4[inicio:fin]) #Media
91.      Pa4[i-1,5]=np.var(Diente_Astillado_4[inicio:fin]) #var
92.      Pa4[i-
   1,6]=skew(Diente_Astillado_4[inicio:fin])[0] #asimetria
93.      Pa4[i-
   1,7]=kurtosis(Diente_Astillado_4[inicio:fin])[0] #curtosis
94.
95.      Pa5[i-
   1,0]=sqrt(mean(square(Diente_Astillado_4[inicio:fin]))) #RMS
96.      Pa5[i-1,1]=np.amax(Diente_Astillado_4[inicio:fin]) #Peak
97.      Pa5[i-1,2]=np.amax(Diente_Astillado_4[inicio:fin])-
   np.amin(Diente_Astillado_4[inicio:fin]) #peak-peak
98.      Pa5[i-1,3]=Pa4[i-1,1]/Pa5[i-1,0] #crest
99.      Pa5[i-
   1,4]=np.mean(Diente_Astillado_4[inicio:fin]) #Media
100.     Pa5[i-1,5]=np.var(Diente_Astillado_4[inicio:fin]) #var
101.     Pa5[i-
   1,6]=skew(Diente_Astillado_4[inicio:fin])[0] #asimetria
102.     Pa5[i-
   1,7]=kurtosis(Diente_Astillado_4[inicio:fin])[0] #curtosis
103.
104.     np.save('Pn.npy', Pn)
105.     np.save('Pf.npy', Pf)
106.     np.save('Pr.npy', Pr)
```

```
    np.save('Pp.npy', Pp)
    np.save('Pa.npy', Pa)
    np.save('Pa2.npy', Pa2)
    np.save('Pa3.npy', Pa3)
    np.save('Pa4.npy', Pa4)
    np.save('Pa5.npy', Pa5)

    #graficar RMS
    f, ax1 = plt.subplots( sharex='col',figsize=(7,6))
    ax1.plot(Pn[:,0])
    ax1.plot(Pf[:,0])
    ax1.plot(Pr[:,0])
    ax1.plot(Pp[:,0])
    ax1.plot(Pa[:,0])
    ax1.plot(Pa2[:,0])
    ax1.plot(Pa3[:,0])
    ax1.plot(Pa4[:,0])
    ax1.plot(Pa5[:,0])
    ax1.legend(['Normal','Diente Faltante','Diente
  Roto','Diente Picado','Diente Astillado 1','Diente Astillado
  2','Diente Astillado 3','Diente Astillado 4','Diente
  Astillado 5'])
    ax1.set_title('RMS', fontsize=14)

    RMS = pd.DataFrame({
        'Etiquetas': ['Normal','Diente Faltante','Diente
  Roto','Diente Picado','Diente Astillado 1','Diente Astillado
  2','Diente Astillado 3','Diente Astillado 4','Diente
  Astillado 5'],
        'valor' : [0.32592991,0.4213465,0.50755609,0.5815712
  9,1.10222899,0.52107871,1.58734613,0.24391928,0.24391928]
    })

    # Grafico
    RMS.plot(x="Etiquetas", y='valor', kind="bar")
    plt.title('RMS', fontsize=14)
    plt.ylabel('Valor', fontsize=14)
    plt.xlabel('Estado del engranaje', fontsize=14)

    #-----------------------------------------------------
  ---------------------
    #graficar Valor peak
    f, ax2 = plt.subplots( sharex='col',figsize=(7,6))
    ax2.plot(Pn[:,1])
    ax2.plot(Pf[:,1])
    ax2.plot(Pr[:,1])
    ax2.plot(Pp[:,1])
    ax2.plot(Pa[:,1])
```

```
147.     ax2.plot(Pa2[:,1])
148.     ax2.plot(Pa3[:,1])
149.     ax2.plot(Pa4[:,1])
150.     ax2.plot(Pa5[:,1])
151.     ax2.legend(['Normal','Diente Faltante','Diente
   Roto','Diente Picado','Diente Astillado 1','Diente Astillado
   2','Diente Astillado 3','Diente Astillado 4','Diente
   Astillado 5'])
152.     ax2.set_title('Valor peak', fontsize=14)
153.
154.
155.     Peak = pd.DataFrame({
156.         'Etiquetas': ['Normal','Diente Faltante','Diente
   Roto','Diente Picado','Diente Astillado 1','Diente Astillado
   2','Diente Astillado 3','Diente Astillado 4','Diente
   Astillado 5'],
157.         'valor' : [0.19137762,0.32677926,0.56071031,0.711112
   46,0.290659,0.25080326,0.54213499,0.25285756,0.25285756]
158.     })
159.
160.     # Grafico
161.     Peak.plot(x="Etiquetas", y='valor', kind="bar")
162.     plt.ylabel('Valor', fontsize=14)
163.     plt.xlabel('Estado del engranaje', fontsize=14)
164.     plt.title('Peak', fontsize=14)
165.
166.
167.     #-----------------------------------------------------
   ---------------------
168.
169.     #graficar Peak to Peak
170.     f, ax3 = plt.subplots( sharex='col',figsize=(7,6))
171.     ax3.plot(Pn[:,2])
172.     ax3.plot(Pf[:,2])
173.     ax3.plot(Pr[:,2])
174.     ax3.plot(Pp[:,2])
175.     ax3.plot(Pa[:,2])
176.     ax3.plot(Pa2[:,2])
177.     ax3.plot(Pa3[:,2])
178.     ax1.plot(Pa4[:,2])
179.     ax3.plot(Pa5[:,2])
180.     ax3.legend(['Normal','Diente Faltante','Diente
   Roto','Diente Picado','Diente Astillado 1','Diente Astillado
   2','Diente Astillado 3','Diente Astillado 4','Diente
   Astillado 5'])
181.     ax3.set(xlabel='Segmento')
182.     ax3.set_title('Valor peak-peak', fontsize=14)
183.
```

```
    PtP = pd.DataFrame({
        'Etiquetas': ['Normal','Diente Faltante','Diente Roto','Diente Picado','Diente Astillado 1','Diente Astillado 2','Diente Astillado 3','Diente Astillado 4','Diente Astillado 5'],
        'valor' : [0.41845175, 0.61534257, 1.13613883, 1.43132087, 0.51531394, 0.51560918, 0.8784165, 0.50944436, 0.50944436]
    })

    # Grafico
    PtP.plot(x="Etiquetas", y='valor', kind="bar")
    plt.ylabel('Valor', fontsize=14)
    plt.xlabel('Estado del engranaje', fontsize=14)
    plt.title('Peak to peak', fontsize=14)


    #-------------------------------------------------------
    #graficar Factor de cresta
    f, ax4 = plt.subplots( sharex='col',figsize=(7,6))
    ax4.plot(Pn[:,3])
    ax4.plot(Pf[:,3])
    ax4.plot(Pr[:,3])
    ax4.plot(Pp[:,3])
    ax4.plot(Pa[:,3])
    ax4.plot(Pa2[:,3])
    ax4.plot(Pa3[:,3])
    ax4.plot(Pa4[:,3])
    ax4.plot(Pa5[:,3])
    ax4.legend(['Normal','Diente Faltante','Diente Roto','Diente Picado','Diente Astillado 1','Diente Astillado 2','Diente Astillado 3','Diente Astillado 4','Diente Astillado 5'])
    ax4.set(xlabel='Segmento')
    ax4.set_title('Factor de cresta', fontsize=14)


    cresta = pd.DataFrame({
        'Etiquetas': ['Normal','Diente Faltante','Diente Roto','Diente Picado','Diente Astillado 1','Diente Astillado 2','Diente Astillado 3','Diente Astillado 4','Diente Astillado 5'],
        'valor' : [ 4.23878538, 5.88386284, 5.98452791, 4.63206865, 5.78380521, 4.20013436, 7.08922732, 4.49241344, 4.49241344]
    })

```

```
    # Grafico
    cresta.plot(x="Etiquetas", y='valor', kind="bar")
    plt.ylabel('Valor', fontsize=14)
    plt.xlabel('Estado del engranaje', fontsize=14)
    plt.title('Factor cresta', fontsize=14)


    #-------------------------------------------------------
    ---------------------
    #graficar Media
    f, ax5 = plt.subplots( sharex='col',figsize=(7,6))
    ax5.plot(Pf[:,4])
    ax5.plot(Pr[:,4])
    ax5.plot(Pp[:,4])
    ax5.plot(Pa[:,4])
    ax5.plot(Pa2[:,4])
    ax5.plot(Pa3[:,4])
    ax5.plot(Pa4[:,4])
    ax5.plot(Pa5[:,4])
    ax5.legend(['Normal','Diente Faltante','Diente
    Roto','Diente Picado','Diente Astillado 1','Diente Astillado
    2','Diente Astillado 3','Diente Astillado 4','Diente
    Astillado 5'])
    ax5.set(xlabel='Segmento')
    ax5.set_title('Media', fontsize=14)


    media = pd.DataFrame({
        'Etiquetas': ['Normal','Diente Faltante','Diente
    Roto','Diente Picado','Diente Astillado 1','Diente Astillado
    2','Diente Astillado 3','Diente Astillado 4','Diente
    Astillado 5'],
        'valor' : [ -0.00159601, -0.00189514, -0.00191527, -
    0.00056088, -0.0015633, -0.00240038, -0.00184775, -
    0.00091444, -0.00091444]
    })

    # Grafico
    media.plot(x="Etiquetas", y='valor', kind="bar")
    plt.ylabel('Valor', fontsize=14)
    plt.xlabel('Estado del engranaje', fontsize=14)
    plt.title('Media', fontsize=14)


    #-------------------------------------------------------
    ---------------------
    #graficar Varianza
```

```
256.     f, ax6 = plt.subplots( sharex='col',figsize=(7,6))
257.     ax6.plot(Pn[:,5])
258.     ax6.plot(Pf[:,5])
259.     ax6.plot(Pr[:,5])
260.     ax6.plot(Pp[:,5])
261.     ax6.plot(Pa[:,5])
262.     ax6.plot(Pa2[:,5])
263.     ax6.plot(Pa3[:,5])
264.     ax6.plot(Pa4[:,5])
265.     ax6.plot(Pa5[:,5])
266.     ax6.legend(['Normal','Diente Faltante','Diente
   Roto','Diente Picado','Diente Astillado 1','Diente Astillado
   2','Diente Astillado 3','Diente Astillado 4','Diente
   Astillado 5'])
267.     ax6.set(xlabel='Segmento')
268.     ax6.set_title('Varianza', fontsize=14)
269.
270.
271.     Var = pd.DataFrame({
272.         'Etiquetas': ['Normal','Diente Faltante','Diente
   Roto','Diente Picado','Diente Astillado 1','Diente Astillado
   2','Diente Astillado 3','Diente Astillado 4','Diente
   Astillado 5'],
273.         'valor' : [ 0.0020359, 0.0030809, 0.00877477, 0.0235
   679, 0.00252301, 0.0035599, 0.00584472, 0.00316722, 0.0031672
   2]
274.     })
275.
276.     # Grafico
277.     Var.plot(x="Etiquetas", y='valor', kind="bar")
278.     plt.ylabel('Valor', fontsize=14)
279.     plt.xlabel('Estado del engranaje', fontsize=14)
280.     plt.title('Varianza', fontsize=14)
281.
282.
283.     #-----------------------------------------------------
   ---------------------
284.     #graficar Asimetría
285.     f, ax7 = plt.subplots( sharex='col',figsize=(7,6))
286.     ax7.plot(Pn[:,6])
287.     ax7.plot(Pf[:,6])
288.     ax7.plot(Pr[:,6])
289.     ax7.plot(Pp[:,6])
290.     ax7.plot(Pa[:,6])
291.     ax7.plot(Pa2[:,6])
292.     ax7.plot(Pa3[:,6])
293.     ax7.plot(Pa4[:,6])
294.     ax7.plot(Pa5[:,6])
```

```
295.    ax7.legend(['Normal','Diente Faltante','Diente
   Roto','Diente Picado','Diente Astillado 1','Diente Astillado
   2','Diente Astillado 3','Diente Astillado 4','Diente
   Astillado 5'])
296.    ax7.set(xlabel='Segmento')
297.    ax7.set_title('Asimetria', fontsize=14)
298.
299.
300.    Asi = pd.DataFrame({
301.        'Etiquetas': ['Normal','Diente Faltante','Diente
   Roto','Diente Picado','Diente Astillado 1','Diente Astillado
   2','Diente Astillado 3','Diente Astillado 4','Diente
   Astillado 5'],
302.        'valor' : [ -0.04704743, -0.1544306, -0.00848705, -
   0.0364813, 0.15573989, -0.10285059, 0.07111518, -
   0.10700252, -0.10700252]
303.    })
304.
305.    # Grafico
306.    Asi.plot(x="Etiquetas", y='valor', kind="bar")
307.    plt.ylabel('Valor', fontsize=14)
308.    plt.xlabel('Estado del engranaje', fontsize=14)
309.    plt.title('Asimetria', fontsize=14)
310.
311.
312.    #-------------------------------------------------------
   ---------------------
313.    #graficar Curtosis
314.    f, ax8 = plt.subplots( sharex='col',figsize=(7,6))
315.    ax8.plot(Pn[:,7])
316.    ax8.plot(Pf[:,7])
317.    ax8.plot(Pr[:,7])
318.    ax8.plot(Pp[:,7])
319.    ax8.plot(Pa[:,7])
320.    ax8.plot(Pa2[:,7])
321.    ax8.plot(Pa3[:,7])
322.    ax8.plot(Pa4[:,7])
323.    ax8.plot(Pa5[:,7])
324.    ax8.legend(['Normal','Diente Faltante','Diente
   Roto','Diente Picado','Diente Astillado 1','Diente Astillado
   2','Diente Astillado 3','Diente Astillado 4','Diente
   Astillado 5'])
325.    ax8.set(xlabel='Segmento')
326.    ax8.set_title('Curtosis', fontsize=14)
327.
328.
329.    PtP = pd.DataFrame({
```

```
330.         'Etiquetas': ['Normal','Diente Faltante','Diente
   Roto','Diente Picado','Diente Astillado 1','Diente Astillado
   2','Diente Astillado 3','Diente Astillado 4','Diente
   Astillado 5'],
331.         'valor' : [ 0.32592991, 0.4213465, 0.50755609, 0.581
   57129, 1.10222899, 0.52107871, 1.58734613, 0.24391928, 0.2439
   1928]
332.     })
333.
334.     # Grafico
335.     PtP.plot(x="Etiquetas", y='valor', kind="bar")
336.     plt.ylabel('Valor', fontsize=14)
337.     plt.xlabel('Estado del engranaje', fontsize=14)
338.     plt.title('Curtosis', fontsize=14)
```

6.4 Código de K-nearest neighbor

```
1. from sklearn.model_selection import train_test_split
2. from sklearn.neighbors import KNeighborsClassifier
3. from sklearn.model_selection import RandomizedSearchCV
4. from sklearn.model_selection import GridSearchCV
5. from sklearn.metrics import plot_confusion_matrix
6.
7. #Adquisición de datos
8. X=TODO
9. Y=Etiquetas
10.
11.      #Preanalaitica - Segmentando Datos
12.      X_train, X_test, y_train, y_test = train_test_split(X, Y
   , test_size=0.2) #Divide los datos entre entrenamiento y
   testeo
13.
14.      #KNeighborsClassifier: Número de vecinos, función de los
   pesos (Uniforme, basada en la distancia)
15.      parameter_space = {
16.      'n_neighbors': [1,2,3,4,5,6,7,8,9,10], #Alternativas de
   parámetros a evaluar
17.      'weights': ['uniform', 'distance'],}
18.      Model = KNeighborsClassifier(n_neighbors=1, weights='dis
   tance')
19.      Model.fit(X_train, y_train)
20.      Yp=Model.predict(X_test)# Se evalúa la red con los datos
   de testeo
21.
22.      from sklearn.metrics import classification_report
23.      target_names = (['Normal','Diente Faltante','Diente
   Roto','Diente Picado','Diente Astillado 1','Diente Astillado
```

```
2','Diente Astillado 3','Diente Astillado 4','Diente
Astillado 5'])
    print(classification_report(y_test, Yp, target_names=tar
get_names))

    # matriz de confusión
    plot_confusion_matrix(Model, X_test, y_test)
    plt.show()

    #KPI por modo de falla
    target_names = (['Normal','Diente Faltante','Diente
Roto','Diente Picado','Diente Astillado 1','Diente Astillado
2','Diente Astillado 3','Diente Astillado 4','Diente
Astillado 5'])
    print(classification_report(y_test, Yp, target_names=tar
get_names))

    #Elección mejor K
    k_range = range(1, 20)
    scores = []
    for k in k_range:
        knn = KNeighborsClassifier(n_neighbors = k)
        knn.fit(X_train, y_train)
        scores.append(knn.score(X_test, y_test))
    plt.figure()
    plt.xlabel('k')
    plt.ylabel('Precisión')
    plt.title('Mejor K', fontsize=14)
    plt.scatter(k_range, scores)
    plt.xticks([0,5,10,15,20])

    #Eficiencia KNN
    n_neighbors = 1

    knn = KNeighborsClassifier(n_neighbors)
    knn.fit(X_train, y_train)
    print('Accuracy of K-NN classifier on training set:
{:.2f}'
         .format(knn.score(X_train, y_train)))
    print('Accuracy of K-NN classifier on test set: {:.2f}'
         .format(knn.score(X_test, y_test)))
```

6.5 Código de Random forest

```
#RandomForestClassifier

from sklearn.model_selection import train_test_split
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV

#adquisición de datos
X=TODO
    Y=Etiquetas

    X_train, X_test, y_train, y_test = train_test_split(X, Y
, test_size=0.2) #Divide los datos entre entrenamiento y
testeo

    Model=RandomForestClassifier(n_estimators=100,criterion=
'entropy')
    Model.fit(X_train, y_train)
    Yp=Model.predict(X_test)# Se evalúa la red con los datos
de testeo

    # matriz de confusión
    from sklearn.metrics import plot_confusion_matrix
    plot_confusion_matrix(Model, X_test, y_test)
    plt.show()

    # KPI por modo de falla
    from sklearn.metrics import classification_report
    target_names = (['Normal','Diente Faltante','Diente
Roto','Diente Picado','Diente Astillado 1','Diente Astillado
2','Diente Astillado 3','Diente Astillado 4','Diente
Astillado 5'])
    print(classification_report(y_test, Yp, target_names=tar
get_names))
```

6.6 Código de SVM

```
#Support Vector Machines
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import classification_report

#adquisición de datos
X=TODO
Y=Etiquetas

    X_train, X_test, y_train, y_test = train_test_split(X, Y
, test_size=0.2) #Divide los datos entre entrenamiento y
testeo

```

```
13.     Model = SVC(kernel='sigmoid',C=5)
14.     Model.fit(X_train, y_train)
15.     Yp=Model.predict(X_test)# Se evalúa la red con los datos
   de testeo
16.
17.
18.     # matriz de confusión
19.     plot_confusion_matrix(Model, X_test, y_test)
20.     plt.show()
21.
22.     # KPI por modo de falla
23.     target_names = (['Normal','Diente Faltante','Diente
   Roto','Diente Picado','Diente Astillado 1','Diente Astillado
   2','Diente Astillado 3','Diente Astillado 4','Diente
   Astillado 5'])
24.     print(classification_report(y_test, Yp, target_names=tar
   get_names))
```