ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA MANTENIMIENTO INDUSTRIAL

“DETECCIÓN DE FALLOS EN MANTENIMIENTO PREDICTIVO UTILIZANDO EL MÉTODO DE APRENDIZAJE DE MÁQUINA RANDOM FOREST”

Trabajo de Integración Curricular

Tipo: Proyecto de Investigación

Presentado para optar al grado académico de:

INGENIERO EN MANTENIMIENTO INDUSTRIAL

AUTOR:

PABLO HERNÁN VILEMA LARA

Riobamba – Ecuador

2022

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA MANTENIMIENTO INDUSTRIAL

“DETECCIÓN DE FALLOS EN MANTENIMIENTO PREDICTIVO UTILIZANDO EL MÉTODO DE APRENDIZAJE DE MÁQUINA RANDOM FOREST”

Trabajo de Integración Curricular

Tipo: Proyecto de Investigación

Presentado para optar al grado académico de:

INGENIERO EN MANTENIMIENTO INDUSTRIAL

AUTOR: PABLO HERNÁN VILEMA LARA

DIRECTOR: Ing. FÉLIX ANTONIO GARCÍA MORA

Riobamba – Ecuador

2022

Yo, Pablo Hernán Vilema Lara, declaro que el presente trabajo de integración curricular es de mi autoría y los resultados del mismo son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

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

Riobamba, 25 de mayo de 2022

Pablo Hernán Vilema Lara
060437039-5

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA MANTENIMIENTO INDUSTRIAL

El Tribunal del Trabajo de Integración Curricular certifica que: El Trabajo de Integración Curricular; Tipo: Proyecto de Investigación, **DETECCIÓN DE FALLOS EN MANTENIMIENTO PREDICTIVO UTILIZANDO EL MÉTODO DE APRENDIZAJE DE MÁQUINA RANDOM FOREST**, realizado por el señor: **PABLO HERNÁN VILEMA LARA**, ha sido minuciosamente revisado por los Miembros del Tribunal del Trabajo de Integración Curricular, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal Autoriza su presentación.

	FIRMA	FECHA
Ing. Marco Antonio Ordoñez Viñán **PRESIDENTE DEL TRIBUNAL**		2022-05-25
Ing. Félix Antonio García Mora **DIRECTOR DE TRABAJO DE INTEGRACIÓN CURRICULAR**		2022-05-25
Ing. Eduardo Segundo Hernández Dávila **MIEMBRO DEL TRIBUNAL**		2022-05-25

DEDICATORIA

Este trabajo está dedicado a mis padres Pablo Efraín Vilema Chuiza y Rosario Victoria Lara Lara, quienes gracias a su esfuerzo y sacrificio han sabido apoyarme tanto en la parte moral como económica durante mi formación como profesional, así como en todo el transcurso de mi vida. A mis hermanas, a mi esposa y amigos que siempre estuvieron presentes con un mensaje de apoyo durante el transcurso de toda la carrera universitaria.

Pablo

AGRADECIMIENTO

En primer lugar, agradecer a la Carrera de Mantenimiento Industrial, a todos los docentes y personal administrativo que la conforman, agradecer también al director del presente trabajo de integración curricular Ing. Félix García, así como al Ing. Eduardo Hernández en calidad de miembro, ya que gracias a su ayuda y guía se ha podido culminar con éxito el presente trabajo.

A toda mi familia y a mi esposa que siempre han estado presentes para brindarme palabras de aliento en el momento preciso.

Pablo

TABLA DE CONTENIDO

ÍNDICE DE TABLAS xi

ÍNDICE DE FIGURAS xii

ÍNDICE DE GRÁFICOS xiv

ÍNDICE DE ANEXOS xv

RESUMEN xvi

SUMMARY xvii

INTRODUCCIÓN 1

CAPÍTULO I

1. MARCO TEÓRICO REFERENCIAL 5

1.1. Inteligencia artificial 5

1.2. Machine learning 6

1.2.1. *Tipos de aprendizaje en machine learning* 6

1.2.1.1. *Aprendizaje supervisado* 6

1.2.1.2. *Aprendizaje no supervisado* 7

1.2.1.3. *Aprendizaje de refuerzo* 8

1.3. Selección del lenguaje de programación para aprendizaje de máquina 8

1.3.1. *Python* 9

1.3.1.1. *Principales características* 9

1.3.1.2. *Principales librerías y herramientas* 10

1.3.2. *R Studio* 11

1.3.2.1. *Principales características* 11

1.3.3. *Matlab* 12

1.3.3.1. *Principales características* 12

1.3.4. *Julia* 12

1.3.4.1. *Principales características* 12

1.3.5. ***Análisis comparativo de los lenguajes de programación*** 13

1.4. **Detección de fallos en mantenimiento predictivo** 14

1.5. **Pasos principales de un modelo de machine learning supervisado** 14

1.5.1. ***Definición del problema*** 15

1.5.2. ***Preparación de datos*** 15

1.5.2.1. *División de datos* 16

1.5.2.2. *Datos desequilibrados* 16

1.5.3. ***Ingeniería de características*** 16

1.5.3.1. *Extracción de características* 17

1.5.3.2. *Reducción de dimensionalidad* 17

1.5.3.3. *Selección de características* 17

1.5.3.4. *Escalado/ Normalización de características* 19

1.5.4. ***Entrenamiento del modelo*** 19

1.5.5. ***Evaluación del modelo*** 20

1.5.6. ***Análisis post hoc*** 20

1.5.6.1. *Significancia estadística* 20

1.6. **Métodos para machine learning** 20

1.6.1. ***Árboles de decisión*** 21

1.6.2. ***Random Forest*** 22

1.6.2.1. *Explicación matemática de Random forest* 23

1.7. **Diferencias entre árboles de decisión y Random Forest** 24

1.8. **Hiperparámetros** 24

1.8.1. ***Ajuste de hiperparámetros y ajustes predeterminados*** 24

1.8.2. ***Estrategias de ajuste de hiperparámetros*** 24

1.8.3. ***Principales hiperparámetros en Random Forest*** 25

1.9. **Métricas utilizadas para evaluar el rendimiento del modelo** 26

1.9.1. ***Matriz de confusión*** 26

1.9.2. ***Precisión*** 27

1.9.3. ***Exactitud*** 27

1.9.4. ***Sensibilidad (recall)*** 27

1.9.5. ***Especificidad*** 27

1.9.6. ***Puntaje F1*** 27

1.9.7. ***Curva característica operativa del receptor (ROC)*** 27

1.10. **Overfitting y underfitting (sobreajuste y desajuste)** 28

1.10.1. ***Overfitting*** 28

1.10.2. ***Underfitting*** 29

1.10.3. ***Detección de overfitting y underfitting*** 29

CAPÍTULO II

2. **MARCO METODOLÓGICO** 31

2.1. **Definición del problema** 31

2.2. **Preparación de datos** 31

2.2.1. ***Descripción del conjunto de datos*** 32

2.2.2. ***Librerías utilizadas para realizar el modelo de machine learning*** 33

2.2.3. ***Creación de carpetas para almacenar datos y resultados*** 34

2.2.4. ***Análisis exploratorio de datos*** 34

2.2.4.1. *Carga de datos* 34

2.2.4.2. *Información de variables* 35

2.2.4.3. *Valores faltantes y filas duplicadas* 35

2.2.4.4. *Análisis gráfico de la variable objetivo* 36

2.2.4.5. *Análisis de correlación* 36

2.2.4.6. *Análisis gráfico de variables de proceso* 37

2.2.4.7. *Gráficos de dispersión para algunos pares de variables* 39

2.2.5. ***Limpieza de datos*** 40

2.2.5.1. *Eliminación de variables* 40

2.2.6. ***Sobremuestreo*** 41

2.2.7. ***División del conjunto de datos para entrenamiento y prueba*** 42

2.3. **Ingeniería de características** 43

2.3.1. ***Extracción de características*** 44

2.3.1.1. *Extracción de características del conjunto de entrenamiento* 44

2.3.1.2. *Extracción de características del conjunto de prueba* 49

2.3.2. ***Selección de características*** 50

2.3.2.1. *Selección de características para el conjunto de entrenamiento* 50

2.3.2.2. *Selección de características para el conjunto de prueba* 51

2.3.3. ***Escalado/normalización de características*** 51

2.4. **Entrenamiento del modelo** 52

2.4.1. ***Optimización de hiperparámetros*** 52

2.4.1.1. *Creación de un bosque aleatorio para examinar los hiperparámetros* 52

***2*.4.1.2.** *Cuadrícula de hiperparámetros aleatorios* 53

2.4.1.3. *Entrenamiento de búsqueda aleatoria* 54

2.4.1.4. *Mejores hiperparámetros* 54

2.4.1.5. *Mejor modelo entrenado* 55

2.5. **Evaluación del modelo** 55

2.5.1. ***Predicción sobre el conjunto de características seleccionadas*** 55

2.5.2. ***Predicción sobre el conjunto original de características*** 55

2.5.3. ***Métricas para la evaluación del modelo*** 56

2.5.3.1. *Matriz de confusión aplicando el conjunto de características seleccionadas* 56

2.5.3.2. *Matriz de confusión aplicando el conjunto original de características* 56

2.6. **Análisis post hoc** 57

2.6.1. ***Ranking del conjunto de características seleccionadas*** 57

2.6.2. ***Ranking del conjunto original de características*** 58

2.6.3. ***Significancia estadística*** 58

CAPÍTULO III

3. **MARCO DE RESULTADOS Y DISCUSIÓN DE RESULTADOS** 60

3.1. **Análisis de métricas con características seleccionadas e hiperparámetros optimizados** 60

3.1.1. ***Matriz de confusión*** 60

3.1.2. ***Exactitud, precisión, sensibilidad, especificidad, puntaje f1*** 62

3.2. **Análisis de métricas con el conjunto original de características e hiperparámetros optimizados** 63

3.2.1. ***Matriz de confusión*** 63

3.2.2. ***Exactitud, precisión, sensibilidad, especificidad, puntaje f1*** 64

3.3. **Resumen de métricas de evaluación con hiperparámetros optimizados y conjunto de características seleccionadas y originales** 65

3.3.1. ***Resumen de las matrices de confusión*** 65

3.3.2. ***Resumen de métricas de evaluación restantes*** 66

3.4. **Comparación del modelo con hiperparámetros predeterminados y optimizados** 66

3.4.1. ***Matriz de confusión con hiperparámetros predeterminados y características seleccionadas*** 67

3.4.2. ***Matriz de confusión con hiperparámetros predeterminados y características originales*** 67

3.4.3. ***Análisis de la influencia de hiperparámetros sobre el modelo*** 68

3.5. **Curva ROC-AUC** 69

3.6. **Curva de aprendizaje** 70

3.7. **Comparación del modelo con otros métodos de machine learning** 71

3.8. **Constatación de la hipótesis** 72

CONCLUSIONES 74

RECOMENDACIONES 76

BIBLIOGRAFÍA

ANEXOS

ÍNDICE DE TABLAS

Tabla 1-1: Factores a tomar en cuenta para la selección del lenguaje de programación 13
Tabla 2-1: Principales hiperparámetros en Random Forest 25
Tabla 3-1: Matriz de confusión 26
Tabla 1-2: Características en el dominio del tiempo 44
Tabla 1-3: Matriz de confusión con características seleccionadas 60
Tabla 2-3: Matriz de confusión con características originales 63
Tabla 3-3: Resumen de los valores de las dos matrices de confusión 65
Tabla 4-3: Resumen de métricas de evaluación restantes 66
Tabla 5-3: Métricas de evaluación en base a hiperparámetros 68
Tabla 6-3: Comparación de métodos de machine learning 71

ÍNDICE DE FIGURAS

Figura 1-1: Tendencia de lenguajes de programación a nivel mundial en los últimos 5 años... 9
Figura 2-1: Submuestreo y sobremuestreo 16
Figura 3-1: Árbol de decisión 21
Figura 4-1: Flujo de proceso de bosque aleatorio 23
Figura 5-1: Curva ROC 28
Figura 6-1: Curva de aprendizaje 29
Figura 1-2: Librerías utilizadas para el modelo de machine learning 33
Figura 2-2: Creación de carpetas 34
Figura 3-2: Carga de datos 34
Figura 4-2: Información de variables 35
Figura 5-2: Valores faltantes 35
Figura 6-2: Filas duplicadas 36
Figura 7-2: Eliminación de variables 41
Figura 8-2: Sobremuestreo de datos 41
Figura 9-2: Definición de variables y aplicación del código de división de datos 42
Figura 10-2: Conjunto de datos de entrenamiento 45
Figura 11-2: Datos para la clase 0 45
Figura 12-2: Características extraídas para la clase 0 46
Figura 13-2: Información de características extraídas 46
Figura 14-2: Datos para la clase 1 47
Figura 15-2: Características extraídas para la clase 1 47
Figura 16-2: Variable objetivo añadida a las características de clase 0 48
Figura 17-2: Variable objetivo añadida a las características de clase 1 48
Figura 18-2: Conjunto final de características de entrenamiento 49
Figura 19-2: Ranking de características 50
Figura 20-2: Características seleccionadas 51
Figura 21-2: Características eliminadas 51
Figura 22-2: Escalado y normalización de los conjuntos de características 52
Figura 23-2: Hiperparámetros predeterminados 53
Figura 24-2: Cuadrícula de hiperparámetros 53
Figura 25-2: Entrenamiento de búsqueda aleatoria 54
Figura 26-2: Mejores hiperparámetros 54
Figura 27-2: Mejor modelo entrenado de Random Forest 55
Figura 28-2: Ingreso del conjunto de características seleccionadas 55

Figura 29-2: Ingreso del conjunto original de características 55
Figura 1-3: Cálculo de métricas con características seleccionadas 62
Figura 2-3: Cálculo de métricas con características originales 64

ÍNDICE DE GRÁFICOS

Gráfico 1-1: Tipos de aprendizaje en machine learning 6
Gráfico 2-1: Pasos para realizar un modelo de machine learning supervisado..... 15
Gráfico 3-1: Modelos de machine learning..... 21
Gráfico 1-2: Observaciones para la variable Falla_maquina 36
Gráfico 2-2: Mapa de calor 37
Gráfico 3-2: Histograma y diagrama de cajas para temperatura del aire y de proceso..... 37
Gráfico 4-2: Histograma y diagrama de cajas para velocidad, torque y desgaste..... 38
Gráfico 5-2: Datos para cada variante de calidad de producto 38
Gráfico 6-2: Dispersión entre temperatura del aire vs temperatura de proceso 39
Gráfico 7-2: Dispersión entre velocidad rotacional vs torque 39
Gráfico 8-2: Dispersión entre torque y desgaste de herramienta 40
Gráfico 9-2: Puntos de datos para cada clase con sobremuestreo. 42
Gráfico 10-2: Datos para entrenamiento y prueba 43
Gráfico 11-2: Datos finales para cada conjunto 49
Gráfico 12-2: Matriz de confusion con características seleccionadas..... 56
Gráfico 13-2: Matriz de confusión con características originales..... 56
Gráfico 14-2: Ranking de características seleccionadas 57
Gráfico 15-2: Ranking del conjunto original de características..... 58
Gráfico 1-3: Matriz de confusión con hiperparámetros predeterminados y características seleccionadas..... 67
Gráfico 2-3: Matriz de confusión con hiperparámetros predeterminados y características originales 67
Gráfico 3-3: Curva de ROC 69
Gráfico 4-3: Curva de aprendizaje..... 70
Gráfico 5-3: Matriz de confusión SVM 72

ÍNDICE DE ANEXOS

ANEXO A: PROGRAMACIÓN FINALIZADA

RESUMEN

El objetivo de la presente investigación fue crear un modelo predictivo de aprendizaje de máquina de tipo supervisado, mediante el método de *Random Forest* y el software libre Python como lenguaje de programación, con la finalidad de detectar fallos en una máquina. Para ello se utilizó el conjunto de datos de mantenimiento predictivo ai4i2020 obtenido del repositorio de *machine learning* de la Universidad de California, Irvine (UCI). La creación del modelo se la realizó en 6 pasos: definición del problema (identificación de características, tarea a realizar y variable objetivo), preparación de datos (análisis exploratorio, limpieza, sobremuestreo debido al desequilibrio en la variable objetivo y división de datos para entrenamiento y prueba, donde los datos fueron divididos 75% y 25% respectivamente), ingeniería de características (extracción de características estadísticas en el dominio del tiempo con la ayuda de la librería de Python TSFEL, selección de características mediante la eliminación recursiva de atributos RFE), entrenamiento del modelo (optimización de hiperparámetros), evaluación del modelo (utilizando el conjunto de características originales y conjunto de características seleccionadas), análisis post hoc (ranking de características que contribuyeron en mayor y menor proporción a la predicción de fallos, significancia estadística). Obteniendo como resultado que el modelo mostró un mejor rendimiento al utilizar el conjunto de características seleccionadas e hiperparámetros optimizados, con un 99,26% en exactitud y 98,95% en precisión. Se concluye que el modelo funcionó con un rendimiento elevado para la detección de fallos y generaliza de forma correcta para nuevos datos. Se recomienda que para la obtención de buenos resultados se preste mucha atención al paso de preparación de datos, ya que depende en gran parte de ello.

Palabras clave: <TECNOLOGÍA Y CIENCIAS DE LA INGENIERÍA> <APRENDIZAJE DE MÁQUINA> <BOSQUE ALEATORIO> <DETECCIÓN DE FALLOS> <MANTENIMIENTO PREDICTIVO>.

SUMMARY

The objective of this research was to create a predictive model of supervised machine learning, using the Random Forest method and the free software Python as a programming language, in order to detect faults in a machine. For this, the predictive maintenance data set ai4i2020 obtained from the machine learning repository of California University, Irvine (UCI) was obtained. The creation of the model was carried out in 6 steps: problem definition (identification of characteristics, task to be carried out and objective variable), data preparation (exploratory analysis, cleaning, oversampling due to the imbalance in the objective variable and division of data for training and test, where the data was split 75% and 25% respectively), feature engineering (extraction of statistical features in the time domain with the help of the Python library TSFEL, selection of features by recursive elimination of RFE attributes) , model training (hyperparameter optimization), model evaluation (using the original feature set and selected feature set), post hoc analysis (ranking of features that contributed most and least to failure prediction, statistical significance) . Obtaining as a result, the model showed a better performance when using the set of selected characteristics and optimized hyperparameters, with 99,26% accuracy and 98,95% precision. It is concluded that the model fails with a high performance for fault detection and generalizes correctly for new data. It is recommended that to obtain good results, you pay close attention to the data preparation step, since it largely depends on it.

Keywords: <TECHNOLOGY AND ENGINEERING SCIENCE> <MACHINE LEARNING> <RANDOM FOREST> <FAULT DETECTION> <PREDICTIVE MAINTENANCE>.

Sandra Paulina Porras Pumalema
C.I. 0603357062

INTRODUCCIÓN

Con el transcurrir de los años la tecnología ha ido evolucionando notablemente en varios campos, como el campo industrial, la medicina, agricultura, entre otros, hoy en día no es la excepción ya que el mundo está siendo testigo de la cuarta revolución industrial o también conocida como industria 4.0. Hablando del campo industrial y específicamente del mantenimiento, la llegada de estas tecnologías ha permitido lograr una mejora significativa dentro de esta área, principalmente en la toma de decisiones; ya que se requiere que las máquinas y procesos operen sin interrupción alguna. Una de las estrategias de mantenimiento que está tomando mayor auge en la actualidad es el mantenimiento predictivo, ya que las acciones se llevan a cabo solo cuando son necesarias y antes que ocurran los fallos, logrando de esa manera la optimización de recursos. Gracias al internet de las cosas con la ayuda de sensores, así como también de técnicas predictivas (termografía, análisis de vibraciones, ultrasonido, entre otros), se puede obtener de las máquinas y procesos una gran cantidad de datos, los cuales son de mucha importancia para el enfoque de mantenimiento de la presente investigación, como es el aprendizaje de máquina.

Las paradas no planificadas a causa de fallos en la maquinaria afectan directamente a los procesos productivos lo cual se traduce en pérdidas económicas muy cuantiosas, bajo este análisis es de suma importancia la detección temprana de fallos, actividad que es llevada a cabo mediante la estrategia de mantenimiento predictivo. Una de las herramientas utilizadas actualmente para estos fines es el *machine learning* (ML), o también conocido como aprendizaje de máquina mismo que pertenece a una de las ramas de la inteligencia artificial y que es aplicado en varios campos. *Machine learning* se fundamenta en extraer conocimiento de los datos y generar predicciones, las cuales sirven para la toma de decisiones dentro del mantenimiento industrial. Para llevar a cabo este proceso, machine learning utiliza algoritmos como la máquina de soporte vectorial (SVM), redes neuronales, árboles de decisión, bosque aleatorio (*Random Forest*), entre otros; de todos los algoritmos mencionados el de bosque aleatorio es uno de los más utilizado para el desarrollo de aplicaciones de mantenimiento predictivo debido a su precisión relativamente alta.

El presente trabajo de integración curricular consiste en crear un modelo de aprendizaje de máquina para la detección de fallos en mantenimiento predictivo, para ello se hará uso del método de aprendizaje de máquina *Random Forest* y un conjunto de datos disponible en el repositorio de la Universidad de California, Irvine (UCI). El trabajo de integración curricular se encuentra estructurado en tres capítulos, donde el primero hace referencia a los fundamentos teóricos, el segundo describe la metodología a utilizar, el tercero menciona el análisis de los resultados, y finalmente se describe un apartado donde se da a conocer las conclusiones y recomendaciones.

Justificación

La detección tardía de fallos incide directamente sobre las máquinas y procesos lo cual se ve reflejado en pérdidas económicas para una empresa, en vista de ello se trata de buscar nuevos métodos de detección basados en aprendizaje de máquina que ayuden a evitar dichas pérdidas mediante la detección temprana de fallos. Gracias al constante desarrollo tecnológico y a la industria 4.0 se puede obtener de los procesos y de las máquinas una gran variedad y cantidad de datos que serán de mucha utilidad para el presente estudio.

La presente investigación consiste en la creación de un modelo de detección de fallos de mantenimiento predictivo utilizando el método de aprendizaje de máquina *Random Forest*, y el uso de datos que fueron tomados del repositorio de *Machine Learning* de la Universidad de California, Irvine (UCI).

Gracias a la creación de este modelo de detección de fallos en conjunto con una estrategia de mantenimiento no tan usual y no muy aplicado en las industrias como es el mantenimiento predictivo, se pretende detectar fallos de manera temprana y así evitar tiempos de inactividad en las máquinas y los procesos lo que repercute directamente en los costos. Para maximizar el tiempo de actividad de la máquina los problemas deben detectarse y corregirse antes de que las máquinas lleguen al punto de fallo (Fernandes et al., 2020).

Una buena estrategia de mantenimiento debe mejorar la condición del equipo, reducir las tasas de fallos del equipo y minimizar los costos de mantenimiento, mientras maximiza la vida útil del equipo. De entre las 3 principales estrategias de mantenimiento (correctiva, preventiva y predictiva), la estrategia predictiva es la que más se destaca. Eh ahí la importancia de la aplicación de estrategias no tan usuales, a su vez que la misma permite realizar las actividades de mantenimiento solo cuando sean necesarias y antes que ocurra el fallo (Carvalho et al., 2019).

Debido al desconocimiento, la falta de aprendizaje y al ser la carrera de Mantenimiento Industrial única en el país, no se ha evidenciado muchas investigaciones de inteligencia artificial enfocadas al mantenimiento dentro del Ecuador, a su vez en la carrera de Mantenimiento Industrial de la Escuela Superior Politécnica de Chimborazo no existen registros de investigaciones de inteligencia artificial con este enfoque, por lo que el desarrollo del presente trabajo será una base potencial para futuras investigaciones, al mismo tiempo que motivará a estudiantes y profesionales dedicados al mantenimiento industrial a buscar nuevos métodos para la resolución de problemas dentro de esta área.

Problema

La detección tardía de fallos genera paros imprevistos en las máquinas y en consecuencia en los procesos productivos, repercutiendo directamente a los costos asociados a la productividad y al mantenimiento, creando grandes pérdidas económicas para la empresa. Entre los principales factores que inciden sobre una detección tardía de fallos se tiene: el desconocimiento, la falta de capacitación y aprendizaje, el poco apoyo por parte de las gerencias y la mínima cantidad de asignación de recursos para el mantenimiento, estos factores han conllevado a no innovar nuevas formas de detección de fallos que ayuden a prevenir tiempos de inactividad en las máquinas y en los procesos a causa de fallos inesperados. Los fallos de una máquina no solo pueden traer como consecuencia un elevado tiempo de inactividad, en comparación con el tiempo de inactividad requerido para la ejecución de actividades preventivas, no obstante, también tienen la capacidad de generar un mayor daño en las máquinas y de la misma forma a los artículos que se estén elaborando en el instante en que se produce la falla (Selcuk, 2017). Además, la implementación de estrategias tradicionales de mantenimiento como son; las correctivas y preventivas, generan algunos problemas dentro de las máquinas y procesos. Por ejemplo al utilizar una estrategia correctiva se trabaja hasta el fallo, es decir la intervención se realiza una vez que ocurrieron los fallos, lo que desencadena en tiempos de inactividad y costos muy representativos; por otra parte si se emplea una estrategia preventiva que se ejecuta de acuerdo a un cronograma planificado se genera un uso ineficiente de recursos y se producen mayores gastos operativos, esto debido a que a veces se actúa de forma innecesaria, ya que, un elemento o pieza sustituida aún puede presentar un tiempo de vida útil restante considerable y aun así ser reemplazado.

Hipótesis

Utilizando el método de aprendizaje de máquina *Random Forest* se detectan fallos en mantenimiento predictivo.

Variable dependiente

Detección de fallos

Variables independientes

Random Forest
Matriz de confusión

Precisión
Sensibilidad

Objetivos

Objetivo general

Detectar fallos en mantenimiento predictivo utilizando el método de aprendizaje de máquina *Random Forest*.

Objetivos específicos

Realizar el preprocesamiento de datos de mantenimiento predictivo del repositorio de la Universidad de California, Irvine (UCI).

Dividir los datos de mantenimiento predictivo para entrenamiento y prueba del modelo.

Detectar las características de extracción estadísticas que correlacionan a los datos de mantenimiento predictivo.

Diseñar y comprobar la precisión del algoritmo de *Random Forest* para la detección de fallos en mantenimiento predictivo.

CAPÍTULO I

1. MARCO TEÓRICO REFERENCIAL

1.1. Inteligencia artificial

La inteligencia artificial (IA) es la capacidad que tienen las máquinas para utilizar algoritmos, educarse de los datos y emplear lo aprendido en la toma de decisiones, así como lo realizaría un humano. No obstante, a diferencia de los individuos, los dispositivos basados en inteligencia artificial no requieren reposar y brindan la posibilidad de examinar enormes volúmenes de información a la vez. Además, la cantidad de errores es significativamente más baja en las máquinas en comparación con la realización de la misma tarea por un ser humano. Las tecnologías fundamentadas en inteligencia artificial se encuentran siendo usadas en beneficio de los seres humanos, ya que pueden contribuir con mejoras significativas y gozar de una elevada eficiencia en casi todos los entornos de la vida (Rouhiainen, 2018, p.17).

A continuación, se dará a conocer varias aplicaciones técnicas de la inteligencia artificial, las cuales se encuentran en un rápido crecimiento en la actualidad:

- Reconocimiento de imágenes fijas, categorización y etiquetado: herramientas que resultan ser útiles para una extensa gama de industrias.
- Mejoras del desenvolvimiento de la táctica algorítmica comercial: implementada de diversas formas en el área financiera.
- En el campo de la medicina, mediante el procesamiento eficaz de datos históricos de los pacientes: con la finalidad que la atención médica sea más eficiente y confiable.
- Mantenimiento predictivo: herramienta que está siendo extensamente aplicada en varios sectores industriales.
- Detección y categorización de objetos: aplicación que puede verse en la industria automovilística autónoma, a pesar de que también tiene potencial para otros campos industriales.
- Repartición de contenido en las redes sociales: se trata principalmente de un instrumento de marketing usada en las redes sociales, pero además se puede utilizar para generar conciencia entre organizaciones sin fin de lucro o para dar a conocer información velozmente como servicio público.
- Protección de seguridad cibernética: aplicación importante en contra de amenazas hacia entidades y sistemas que envían y reciben pagos online (Rouhiainen, 2018, p.18).

1.2. Machine learning

También conocido como aprendizaje de máquina, es una de las disciplinas principales en el campo de la inteligencia artificial, en el que las máquinas u ordenadores, son capaces de aprender sin la necesidad de estar programados para aquello (Rouhiainen, 2018, p.19). Para Müller y Guido (2017, p.1) el aprendizaje de máquina se fundamenta en extraer conocimiento de los datos y generar predicciones, mediante la intersección de la informática, la inteligencia artificial y la estadística.

Machine learning utiliza algoritmos con la finalidad de aprender de los patrones de los datos. Un ejemplo muy sencillo es el spam de correo electrónico, donde se detecta que correos son no deseados para posterior a ello separarlos de los que no lo son. Este es un ejemplo práctico del aprendizaje de máquina, donde los algoritmos pueden utilizarse para aprender patrones y emplear dicho conocimiento en la toma de decisiones (Rouhiainen, 2018, p.19).

1.2.1. Tipos de aprendizaje en machine learning

A continuación, el gráfico 1-1 muestra los tipos de aprendizaje en *machine learning*

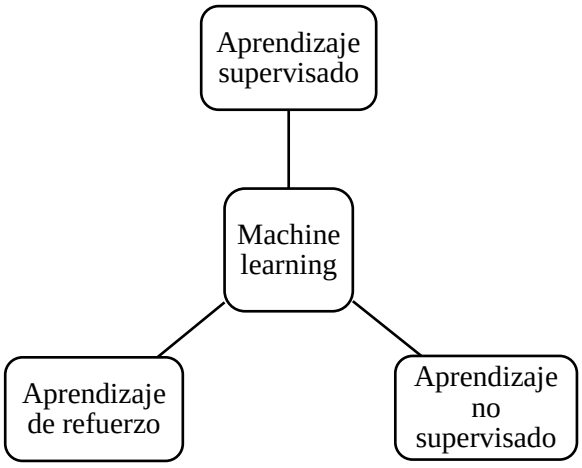


Gráfico 1-1: Tipos de aprendizaje en machine learning

Fuente: Rouhiainen, 2018, p.20

Realizado por: Vilema Pablo, 2021

1.2.1.1. Aprendizaje supervisado

Este tipo de aprendizaje se caracteriza porque el usuario brinda al algoritmo pares de entradas y salidas deseadas, gracias a ello el algoritmo busca una manera de generar la salida deseada a partir de una entrada proporcionada (Müller y Guido, 2017: p.2).

Los autores Müller y Guido (2017, p.3) proporcionan algunos ejemplos de tareas de aprendizaje supervisado como:

- Partiendo de dígitos escritos a mano en un sobre, identificar el código postal: para este ejemplo la entrada es el escaneo de la escritura a mano y la salida que se desea son los dígitos reales del código postal. Para crear una base de datos que posteriormente servirá para generar un modelo de aprendizaje de máquina, se necesita recopilar una gran cantidad de sobres y luego leer los dígitos de los diferentes códigos postales, mismos que serán almacenados como resultados deseados.
- En base a una imagen médica, comprobar si un tumor es benigno: para este caso la entrada es la imagen médica y la salida deseada es si el tumor es benigno o no. Para generar un conjunto de datos que servirá para el modelado de aprendizaje de máquina, se requiere de una gran cantidad de imágenes médicas, además de la opinión de un experto el cual una vez examinadas las imágenes determinará cual tumor es benigno y cual no. Inclusive se analizaría la posibilidad de hacer un diagnóstico adicional que va más allá de la examinación del contenido de la imagen, con la finalidad de determinar si el tumor es canceroso o no.
- En transacciones realizadas con tarjeta de crédito, detectar algún tipo de actividad fraudulenta: en este ejemplo la entrada es un registro de las transacciones de la tarjeta de crédito y la salida deseada es si la transacción fue fraudulenta o no. Generar una base de datos para este ejemplo significa, almacenar todas las transacciones realizadas y registrar si en algún momento un usuario reporta alguna transacción como fraudulenta.

1.2.1.2. Aprendizaje no supervisado

Para este tipo de aprendizaje solamente se proporciona los datos de entrada, por lo cual no se brinda al algoritmo datos de salida conocidos, convirtiéndolo en un método más difícil de evaluar y comprender, a pesar de que se conoce de la existencia de muchas aplicaciones exitosas con este enfoque (Müller y Guido, 2017: p.3).

Ejemplos de aprendizaje no supervisado son:

- Partiendo de un conjunto de publicaciones de blog, distinguir temas: si se dispusiera de una extensa colección de datos de texto, tal vez se desee encontrar temas predominantes en dicha colección. Por lo que se convierte en una posibilidad que no se sepa cuáles son estos temas o cuantos existirán, donde se puede evidenciar que no existen salidas conocidas.
- Dividir a los clientes en grupos con preferencias similares: a partir de una base de registros de clientes, es posible que se quiera determinar cuáles clientes son similares y

si existen grupos de clientes que tengan preferencias similares. Por ejemplo, para un lugar de compras estos pueden ser "jugadores" o "padres", puesto que no se sabe con certeza cuales podrían ser estos grupos o inclusive cuantos hay, no se tiene salidas conocidas.

- Encontrar patrones de acceso anormal a un sitio web: para detectar este tipo de abusos o errores, con frecuencia es muy útil hallar patrones de acceso que sean distintos a la norma. Los patrones anormales pueden ser muy diferentes entre sí y hay la posibilidad que no tenga ningún caso asociado de comportamiento anormal. En vista a que solo se observa el tráfico de patrones y no se conoce que constituye un comportamiento normal y anormal, el ejemplo se convierte en un problema no supervisado (Müller y Guido, 2017: p.4).

1.2.1.3. Aprendizaje de refuerzo

En este tipo de aprendizaje se dice que los algoritmos aprenden de la experiencia, es decir en cada vez que aciertan hay que proporcionarles un refuerzo positivo, la forma como estos algoritmos aprenden es análogo como cuando se premia con una recompensa a un perro por aprender algún tipo de habilidad, por ejemplo, a sentarse (Rouhiainen, 2018, p.21).

1.3. Selección del lenguaje de programación para aprendizaje de máquina

En la actualidad, varios son los lenguajes de programación que están siendo empleados para el desarrollo de aplicaciones de inteligencia artificial y también para una de sus ramas principales como es el *machine learning*, cada uno de estos lenguajes poseen ventajas, así como desventajas y algunos resultan ser mejores en comparación con otros, de acuerdo a la aplicación en la cual se lo utilizará o al problema a solucionar.

Para el análisis de la selección del lenguaje de programación, se ha tomado como referencia el artículo de Manrique (2020) donde se aborda los lenguajes Python y R Studio debido a que son los más comunes para el desarrollo de estas aplicaciones, Julia debido a que es un nuevo lenguaje para inteligencia artificial y Matlab debido a su orientación estadística y matemática.

La figura 1-1 muestra la tendencia de estos lenguajes de programación a través del tiempo, donde se puede apreciar que los lenguajes de programación más utilizados son Python y R Studio, seguido se encuentra Matlab y el menos utilizado es Julia, debido a que es un lenguaje nuevo. A su vez también se aprecia que Python con el transcurrir de los últimos años ha venido en constante crecimiento, debido a la sencillez en la curva de aprendizaje, facilidad en el desarrollo de aplicaciones de distinto propósito y la gran sociedad que aporta continuamente (Aimacaña y Columba, 2021).

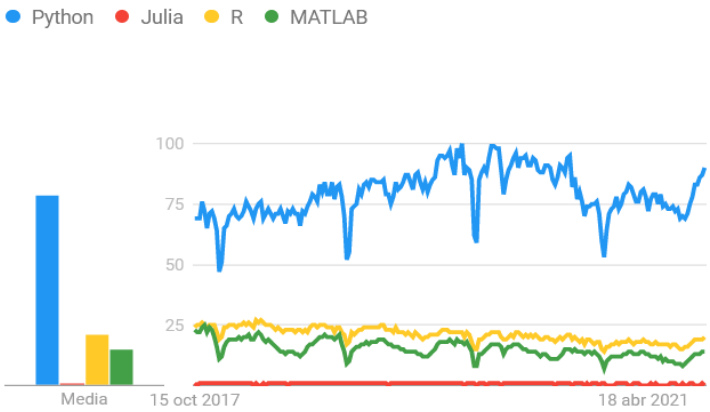


Figura 1-1: Tendencia de lenguajes de programación a nivel mundial en los últimos 5 años

Fuente: Google Trends (https://n9.cl/g7bpg)

Realizado por: Vilema Pablo, 2021

A continuación, se procederá a describir estos 4 tipos de lenguajes, para posteriormente seleccionar uno con el cual se llevará a cabo el presente trabajo de investigación.

1.3.1. Python

Python fundamentalmente es un lenguaje de programación de elevado nivel, multipropósito e interpretado. Año tras año su utilización ha ido aumentando considerablemente, por lo que en la actualidad se posiciona como uno de los lenguajes de programación mayormente utilizados para el desarrollo de software. Python brinda la posibilidad de ser empleado en varios sistemas operativos y plataformas, entre los que destacan los más populares como Windows, Mac Os X, y Linux. Gracias a que el lenguaje de programación Python no tiene un ámbito específico como por ejemplo PHP para desarrollar aplicaciones web, permite el desarrollo de software para crear juegos, para aplicaciones científicas, comunicaciones de red, aplicaciones de escritorio con interfaz gráfica de usuario, para aplicaciones web y teléfonos inteligentes (Fernández, 2013). Cabe acotar que Python es un tipo de software libre, por lo que se tiene acceso al código fuente de un programa, concediendo su libertad de ejecución, uso, distribución y alteración. En otras palabras, un nuevo software creado bajo este enfoque puede ejecutarse en cualquier ambiente, podría ser utilizado para un fin cualquiera, ser distribuido sin que se entere el usuario y modificarse de así creer conveniente (Challenger et al., 2014: p.2).

1.3.1.1. Principales características

- Es interpretado, esto quiere decir que no es necesario compilar el código para la ejecución, debido a que hay un intérprete encargado de leer el fichero fuente y ejecutarlo.

- Es multiplataforma, en otras palabras, Python brinda la posibilidad de ejecutar el mismo código en varias plataformas y sistemas operativos, sin que haya la necesidad de modificar el código fuente, eso si es necesario tener instalado el intérprete.
- Simplicidad en la sintaxis del lenguaje, es decir se hace sencillo escribir un código que sea fácil de leer, esto es un factor muy importante a tener en cuenta ya que, encima de que facilita el aprendizaje del lenguaje de programación, también ayuda a que el código sea más fácil de mantener (Fernández, 2013).

1.3.1.2. Principales librerías y herramientas

Dentro de las principales librerías y herramientas, utilizadas para una mejor compresión y desarrollo de algoritmos de aprendizaje de máquina en Python se tiene las siguientes.

- Scikit-learn: la librería scikit-learn es la más destacada y utilizada para aplicaciones de aprendizaje de máquina, debido a que contiene varios algoritmos de última generación para desarrollar machine learning, además también dispone la documentación completa de cada uno de los algoritmos (Müller y Guido, 2017: p.6).
- Numpy: el paquete numpy es uno de los más elementales en Python para la computación científica, ya que se puede generar matrices multidimensionales, también contiene funciones matemáticas de elevado nivel como transformadas de Fourier, operaciones con algebra lineal, y generadores de dígito pseudoaleatorios. En scikit-learn se utiliza la matriz numpy como estructura de datos fundamental, en otras palabras, cualquier dato que se esté utilizando para el análisis en scikit-learn debe convertirse en una matriz numpy (Müller y Guido, 2017: p.7).
- Scipy: otra de las librerías que contiene funciones para computación científica, entre sus principales funciones se destaca la optimización de funciones matemáticas, funciones matemáticas especiales, funciones avanzadas de algebra lineal, distribuciones estadísticas y el procesamiento de señales. Algo muy importante que hay que tener en cuenta es que scikit-learn se basa en las funciones de scipy para la implementación de algoritmos (Müller y Guido, 2017: p.8).
- Matplotlib: esta es la librería principal para el trazado científico en Python, contiene funciones que permiten realizar visualizaciones con gran calidad, como por ejemplo histogramas, gráficos de dispersión, gráficos de líneas, etc. La visualización de datos puede aportar información muy importante y valiosa, para una mejor comprensión del comportamiento de los mismos (Müller y Guido, 2017: p.9).
- Pandas: es una librería dedicada a la gestión y análisis de datos en Python. Su construcción se fundamenta en una estructura de datos llamada *DataFrame*, dicho de otra

forma, un *DataFrame* es una tabla parecida a una hoja de cálculo en Excel. Pandas brinda una enorme variedad de métodos para la operación y modificación de estas tablas, particularmente permite que se realicen consultas de tipo SQL y uniones de tablas. En contraste con numpy, que necesita que absolutamente todas las entradas de una matriz pertenezcan al mismo tipo, pandas permite que cada columna de una tabla cualesquiera puedan manejar diferentes tipos de entradas, como por ejemplo fechas, cadenas, números enteros, y números de punto flotante. Otra gran ventaja que ofrece esta librería es la capacidad de lectura de datos a partir de varios formatos de archivo, como formato Excel, txt, y valores separados por comas (csv) (Müller y Guido, 2017: p.10).

- Jupyter notebook: básicamente es una herramienta de código abierto fundamentado en navegador, funciona como un cuaderno de laboratorio virtual que admite código, datos, flujo de trabajo, y visualizaciones que detallan y permiten una mejor comprensión del proceso de investigación. El cuaderno de jupyter es legible por humanos y máquinas lo cual facilita la comunicación académica y la interoperabilidad, a su vez son un medio para que la ciencia sea más abierta (Randles et al., 2017). Por otra parte, para (Müller y Guido, 2017: p.7) jupyter notebook es un entorno interactivo que sirve para generar y ejecutar código en el navegador, es muy utilizado por los científicos de datos y resulta ser una herramienta de gran ayuda para el análisis exploratorio de datos. Una de las grandes ventajas del cuaderno de jupyter es que permite la incorporación de código, imágenes y texto por lo que el resultado final es una investigación más detallada y comprensible.

1.3.2. R Studio

Este tipo de lenguaje de programación está especialmente enfocado al análisis estadístico y a la representación gráfica de los resultados que se obtuvieron una vez realizado el análisis, es de tipo software libre y multiplataforma, lo cual ha sido muy importante para tener una gran acogida y permitir el crecimiento de la comunidad que lo utiliza, constantemente se añaden nuevas funcionalidades y se crean versiones mejoradas a las ya existentes (Manrique, 2020).

1.3.2.1. Principales características

- Creación de gráficos fundamentados en LaTeX
- Existencia de gran variedad de herramientas estadísticas (algoritmos de clasificación y agrupación, pruebas estadísticas, modelos lineales y no lineales).
- Gracias a que su programación es de tipo POO (orientada a objetos), se puede definir sus propias funciones y objetos.
- Se puede utilizar con un enfoque matemático y ser sustito de Matlab (Manrique, 2020).

Gracias a que el lenguaje de programación R Studio contiene varias características especiales, hace que se vuelva muy versátil el manejo de elementos estadísticos permitiendo una manipulación de datos súper rápida. En comparación con Python la curva de aprendizaje es más lenta y complicada (Manrique, 2020).

1.3.3. Matlab

Este lenguaje se lo utiliza mayormente para la resolución de problemas científicos y de ingeniería, se encuentra fundamentado en matrices y es la manera más común para representar y expresar las matemáticas computacionales alrededor del mundo entero, una gran desventaja es que es distribuido bajo licencia pagada (Manrique, 2020).

1.3.3.1. Principales características

- Permite optimizar el rendimiento de los modelos mediante el ajuste de hiperparámetros y la selección automática de funciones.
- Gracias al plugin MathWorks for *Machine Learning* se puede encontrar todos los algoritmos de regresión, agrupación y clasificación para aprendizaje supervisado y no supervisado.
- Mayor rapidez en la ejecución comparado con el código abierto (Manrique, 2020).

1.3.4. Julia

El lenguaje de programación Julia fue creado para aplicaciones de cálculo científico que se utilizan en entornos matemáticos, presenta un alto rendimiento debido a que desde un principio fue diseñado con ese fin, hace uso de un compilador JIT (*just in time*) envés de un intérprete como es el caso de Python, por lo que podría ser una desventaja versus otros lenguajes de programación ya que la primera vez que se ejecuta el código lo hará de manera lenta (Manrique, 2020).

1.3.4.1. Principales características

- Mayor velocidad versus otros lenguajes de programación como por ejemplo Python.
- Permite gestionar automáticamente la memoria.
- Sintaxis enfocada a las matemáticas.
- Gestión de recursos mediante paralelismo (Manrique, 2020).

1.3.5. Análisis comparativo de los lenguajes de programación

La tabla 1-1 muestra la comparación de los lenguajes de programación analizados, en base a 4 factores.

Tabla 1-1: Factores a tomar en cuenta para la selección del lenguaje de programación

Factores	Descripción
Velocidad	Cuando de seleccionar el lenguaje de programación se trata, la velocidad es un factor muy importante a tener en cuenta, R Studio fue diseñado como un lenguaje estadístico por lo que presenta un gran soporte en esta área, por otra parte, Python depende de las librerías, por lo que para este tipo de aplicaciones estadísticas R Studio resulta ser un poco más rápido que Python.
Curva de aprendizaje	Si de la perspectiva funcional se trata, el lenguaje de programación idóneo es R Studio, a su vez si se está orientado a objetos se recomienda utilizar Python. Python será mucho más fácil de aprender en comparación con R Studio, siempre y cuando se pertenezca al grupo de programadores funcionales. Por otra parte, Julia y Matlab presentan similitudes que permiten que el aprendizaje sea fácil, ya que los dos están enfocados a entornos matemáticos.
Costo	Matlab es el único lenguaje de programación de los 4 que fueron analizados, por el cual se paga y se necesita de una licencia para ser utilizado, es por ello que no es tan utilizado para desarrollar aplicaciones de inteligencia artificial.
Comunidad para soporte	Todos los lenguajes de programación analizados son conocidos en el mercado y presentan apoyo de la comunidad, pero hay que recalcar que Python presenta la comunidad internauta más grande, por lo que cuando se trata de solucionar algún tipo de problema relacionado al desarrollo, existe gran cantidad de información disponible para hacerlo.

Fuente: Manrique, 2020.

Realizado por: Vilema Pablo, 2021.

Una vez realizado el análisis de los 4 lenguajes de programación, se concluye que para la realización de la presente investigación se utilizará el lenguaje Python, principalmente porque es de código abierto, presenta facilidad en el aprendizaje y por ser el lenguaje mayormente utilizado

por la comunidad a nivel mundial, lo cual permitirá encontrar respuestas a diversos problemas que surjan al llevar a cabo la investigación, ya que existe una gran cantidad de información en la red.

1.4. Detección de fallos en mantenimiento predictivo

En el entorno industrial, el mantenimiento de las máquinas es un factor muy importante a tener en cuenta, ya que afecta el tiempo de operación de las mismas y su eficiencia. Es por ello, que los fallos deben ser detectados y solucionados, evitando de esta manera paradas en los procesos productivos (Carvalho et al., 2019).

Para Silva y Capretz (2019), el mantenimiento predictivo (PdM), ejecuta el mantenimiento fundamentado en los indicadores del estado de salud del equipo. Los sensores con capaces de medir un patrón inusual de estos indicadores, como por ejemplo un mayor nivel de vibración en un motor o un mayor consumo de energía y, en casi todos los casos, los fallos están precedidos por un patrón anormal de estas medidas.

Carvalho et al. (2019) menciona que el PdM, usa herramientas predictivas para determinar en qué momento son necesarias las acciones de mantenimiento. PdM se fundamenta en el monitoreo continuo de una máquina o de la integridad de un proceso, gracias a ello el mantenimiento se llevará a cabo solo cuando es necesario y antes que ocurra el fallo. A su vez, permite la detección temprana de fallos, gracias a herramientas predictivas basadas en datos históricos (como, por ejemplo técnicas para aprendizaje de máquina), factores de integridad (como desgaste, aspectos visuales, ruidos anómalos, entre otros), métodos de inferencia estadística y enfoques de ingeniería.

Los fallos de la máquina no solo pueden resultar en un mayor tiempo de inactividad, comparado con el tiempo de inactividad requerido por las intervenciones preventivas, sino que además pueden causar un mayor daño en las máquinas, así como en los artículos en producción en el momento en que ocurre la falla (Fernandes et al., 2020). Es por ello que el PdM es una técnica muy prometedora, ya que puede reducir significativamente los costos asociados al mantenimiento y el tiempo de inactividad de la máquina, mientras se hace que la frecuencia de mantenimiento sea lo más baja posible, algo que conlleva a una optimización adecuada de recursos (Lee et al., 2019).

1.5. Pasos principales de un modelo de machine learning supervisado

Vieira et al. (2019a) menciona 6 pasos principales para un modelo de aprendizaje de máquina

supervisado, los mismos que se muestran en el gráfico 2-1.

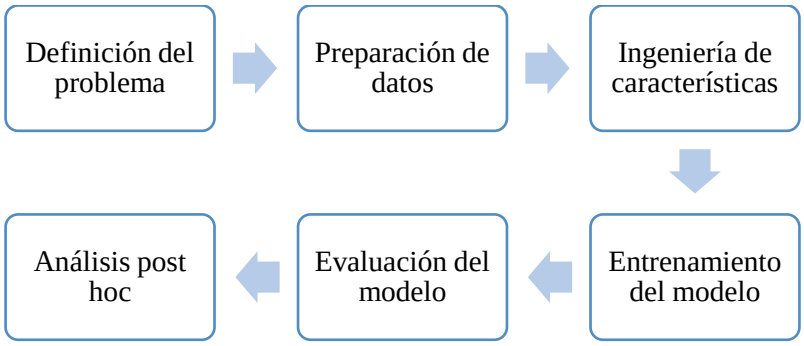


Gráfico 2-1: Pasos para realizar un modelo de machine learning supervisado

Fuente: Vieira et al., 2019a

Realizado por: Vilema Pablo, 2021

1.5.1. Definición del problema

En aprendizaje supervisado, una pregunta de investigación bien definida necesita de una variable objetivo, un conjunto de características y una tarea bien definidos. Juntar estos elementos permite formular un enunciado sucinto al problema; por ejemplo, utilizar datos de conectividad funcional (características) con la finalidad de categorizar (tarea) que pacientes se beneficiarán y que pacientes no se beneficiarán de un determinado tratamiento (objetivo) (Vieira et al., 2019a). Es de suma importancia en la realización de cualquier tipo de proyecto y más aún en *machine learning*, tener un problema bien definido, ya que, un mismo conjunto de datos puede ser analizado de diferentes formas (Vieira et al., 2019b).

1.5.2. Preparación de datos

Machine learning ayuda a descubrir patrones en los datos de entrada y utilizar estos patrones para hacer predicciones sobre nuevos datos. Para lograr aquello es muy importante limpiar, explorar y preparar los datos para mejorar la calidad general del conjunto de datos. Para este paso generalmente se empela histogramas y diagramas de dispersión para explorar los datos y la utilización de una variedad de estrategias para minimizar el impacto de los valores atípicos y los valores faltantes (Vieira et al., 2019a).

Por otra parte Brownlee (2020) menciona que la preparación de datos consiste en la transformación de datos sin procesar en una forma más adecuada para el modelado.

1.5.2.1. División de datos

Machine learning aprende de los datos con los que fue entrenado, para posteriormente tratar de hallar el patrón que le permita predecir el resultado de un nuevo caso. Debido a ello, para poder evaluar un modelo de aprendizaje de máquina, se requiere de un conjunto de datos que el algoritmo no ha visto. Por eso los datos se deben dividir en dos conjuntos: conjunto de entrenamiento y conjunto de prueba. No existe una regla clara para saber el porcentaje de los datos que deben ser destinados para cada conjunto, entre algunos ejemplos se utiliza 70% para entrenamiento y 30% para prueba, en otras ocasiones 75% y 25%, 80% y 20%, entre otros. La proporción de la división puede variar de acuerdo a diferentes casos de estudio.

1.5.2.2. Datos desequilibrados

El desequilibrio de datos en *machine learning* hace referencia a una distribución desigual de clases dentro de un conjunto de datos. Principalmente este problema se da en tareas de clasificación en las que la distribución de clases o etiquetas en un conjunto de datos no es uniforme. La forma de solucionar este problema es mediante el método de remuestreo, añadiendo registros a la clase minoritaria (sobremuestreo) o eliminando registros de la clase mayoritaria (submuestreo). Los datos desequilibrados afectan directamente al rendimiento del modelo, ya que se dará mayor atención a la clase mayoritaria y se eliminará la clase minoritaria; un claro ejemplo es la aplicación en la detección de fraudes, ya que de un conjunto de datos existirán muy pocos registros de una clase y abundantes registros de la otra (Mohammed et al., 2020). Entre algunas técnicas de submuestreo se tiene: enlaces tomeks, centroides clúster, entre otros; por otra parte, una de las técnicas de sobremuestreo de minoría sintética más empleada es SMOTE, la cual para Viloria et al. (2020) crea nuevas observaciones sintéticas interpoladas y las añade a la clase minoritaria.

Figura 2-1: Submuestreo y sobremuestreo

Fuente: Mohammed et al., 2020

1.5.3. Ingeniería de características

La ingeniería de características es el proceso de transformar datos en características que

representan mejor el problema subyacente, lo que conlleva a un rendimiento mejorado del aprendizaje de máquina (Ozdemir y Susarla, 2018: p.12). Por otra parte, el conjunto de características para un enfoque de aprendizaje de máquina se denomina vector de características (Jahnke, 2015). Vieira et al. (2019a) menciona algunos métodos de ingeniería de características: extracción de características, reducción de dimensionalidad y selección de características.

1.5.3.1. Extracción de características

Las características se pueden extraer tanto en el dominio del tiempo como en el dominio de la frecuencia. La extracción de características hace referencia a las transformaciones realizadas a los datos sin procesar, con la finalidad de extraer un vector de características apropiado para los modelos de aprendizaje de máquina, la calidad y el tamaño del conjunto de características son factores muy importantes a tener en cuenta, ya que tendrán gran influencia sobre el rendimiento del modelo (Vieira et al., 2019a).

1.5.3.2. Reducción de dimensionalidad

La reducción de dimensionalidad ayuda en gran parte a mitigar el sobreajuste del modelo, el mismo que se da cuando el modelo aprende detalles en los datos, en lugar de propiedades generalizables globales; aquello da como resultado un elevado rendimiento en los datos que se utilizaron para construir el modelo (datos de entrenamiento), pero un bajo rendimiento cuando se aplica a nuevos datos (datos de prueba). Mientras más pequeño sea el conjunto de datos, es menos probable que el modelo aprenda patrones generalizables, por lo tanto, el riesgo de sobreajuste es mayor. La reducción de dimensionalidad genera un nuevo conjunto de características intentando no perder mucha información, manteniendo así o mejorando el rendimiento del modelo. Un conjunto de características más pequeño ayudará a disminuir el tiempo y recursos computacionales, así como también eliminará las características redundantes si las hubiera (Vieira et al., 2019a). Una de las técnicas más utilizadas para llevar a cabo la reducción de dimensionalidad, es el análisis de componentes principales (PCA), el cual usa una transformación ortogonal para convertir un conjunto de observaciones que incluyen variables posiblemente correlacionadas, en un conjunto más pequeño de variables no correlacionadas conocidas como componentes principales (Vieira et al., 2019a).

1.5.3.3. Selección de características

La selección de características es una técnica muy común y ampliamente utilizada con la finalidad de reducir el costo computacional y aumentar la precisión del modelo. Se basa en la selección de

un subconjunto de características que se definen como las más representativas (Jahnke, 2015). Cabe destacar que la selección de características difiere de la reducción de la dimensionalidad, claro está que los dos métodos buscan reducir el número de características, pero en la reducción de dimensionalidad se crean nuevas y se usan en lugar de las características originales, por otro lado, el método de selección elimina alguna de las características originales. Se pueden seleccionar características de forma manual y automatizada, la selección manual se da cuando existe un conocimiento a priori acerca de cuan informativas son ciertas características, por otra parte, para la selección automatizada, no se necesita de un conocimiento a priori sobre cuales características contribuirán o no a la tarea, debido a que el conjunto óptimo de características es elegido a través de un método estadístico (Vieira et al., 2019a).

Vieira et al. (2019a) indica 3 métodos para seleccionar las características de manera automatizada:

- Métodos de filtro: estos métodos clasifican las características en base a puntuaciones estadísticas simples, como la varianza, la media y los coeficientes de correlación; posterior a ello las características se mantienen o eliminan según su clasificación. Los métodos de filtrado generalmente son univariados, por lo que, consideran cada una de las características de manera independiente o en términos de su asociación con la variable de destino. Entre algunas de las técnicas de filtrado se incluyen el coeficiente de correlación de Pearson, y las estadísticas a nivel de grupo como ANOVA y las pruebas t. Los métodos de filtrado son menos propensos al sobreajuste, pero no toman en cuenta las interacciones multivariadas entre características, lo que puede conllevar a la pérdida de información relevante o a la selección de características redundantes.
- Métodos de envoltura: para estos métodos el algoritmo de aprendizaje de máquina se encuentra envuelto en un algoritmo de selección de características, el cual se encarga de filtrar el conjunto original de características para la combinación de las mismas que produce el mayor rendimiento. Los métodos de envoltura se pueden clasificar en métodos de selección hacia adelante y eliminación hacia atrás; para el método de selección hacia adelante, las características más relevantes se añaden iterativamente a un conjunto que de forma inicial se encuentra vacío, en cambio en la eliminación hacia atrás la búsqueda comienza inicialmente con todas las características y van siendo eliminadas iterativamente. Para los dos casos la búsqueda finaliza cuando se encuentra un número óptimo de características. En comparación con los métodos de filtro, los de envoltura son multivariados por lo que pueden tener en cuenta las interacciones entre características, a su vez tienden a seleccionar características de buen rendimiento ya que la selección depende del rendimiento del algoritmo.
- Métodos integrados: de manera análoga a los métodos de envoltura, los métodos integrados también seleccionan características en función del rendimiento del algoritmo,

pero a diferencia de los métodos de envoltura, los enfoques integrados son una parte intrínseca del proceso de aprendizaje, esto quiere decir que aprenden que características contribuyen más al rendimiento del modelo, durante la etapa de entrenamiento. El método de regularización es una de las técnicas de selección integradas más común, esta técnica se emplea generalmente cuando hay una gran cantidad de características y se cree que todas contribuyen a la tarea. Las técnicas de regularización seleccionan características reduciendo el peso, es decir la importancia de algunas características a cero, este proceso es conocido como penalización de regularización, donde solo las características con mayor peso discriminatorio tendrán un valor diferente a cero y serán las que aporten a la predicción. Entre los métodos más comunes con penalización de regularización son el operador de selección y contracción absoluta mínima, regresión de crestas y red elástica.

1.5.3.4. Escalado/ Normalización de características

En algunos conjuntos de datos se tiene características que varían en unidades y rangos, para modelar los datos de forma correcta, la mayoría de los algoritmos de machine learning necesitan que los datos se encuentren en la misma escala, ya que de no ser así algunas características puede ser dominadas por otras, dificultando el aprendizaje de los algoritmos. La normalización son un conjunto de operaciones que ayudan a lidiar con estos problemas, ya que están destinadas a alinear y transformar columnas y filas en un conjunto coherente de reglas. Las técnicas de normalización garantizan que todas las filas y columnas se traten igual bajo los ojos del *machine learning* (Ozdemir y Susarla, 2018: p.90). Dos de los métodos más utilizados son el escalado mínimo-máximo y la normalización (o estandarización) del puntaje z. El primero transforma los datos a un rango deseado y el segundo cambia los datos para que se asemejen a una distribución normal (Vieira et al., 2019a).

1.5.4. Entrenamiento del modelo

Para una mejor compresión del entrenamiento del modelo se necesita definir la función $y = f(X)$, ya que, en el aprendizaje supervisado, el entrenamiento del modelo hace referencia al proceso en el que un algoritmo de aprendizaje de máquina encuentra una función f que mejor mapea algunas características de entrada X y la variable de destino y o también conocida como variable dependiente. No obstante, hay que destacar que el objetivo primordial del aprendizaje de máquina es encontrar una función f capaz de obtener un alto rendimiento cuando se aplica a nuevos datos no vistos o datos que no fueron utilizados para el entrenamiento (Vieira et al., 2019a).

1.5.5. Evaluación del modelo

Una vez que el algoritmo fue entrenado y posterior a ello se aplicó el conjunto de prueba, se debe buscar la manera de cuantificar el desempeño del modelo. Por ejemplo, para problemas de clasificación, el rendimiento es calculado mediante una matriz de confusión la cual contrasta las etiquetas predichas con las reales, otra de las métricas más utilizadas es la precisión que se refiere a la proporción de las predicciones correctas. Por otro lado, en un problema de regresión, el rendimiento del modelo se estima, calculando cuanto se desvían las puntuaciones pronosticadas del valor objetivo real en el conjunto de prueba, alguna de las métricas que se utilizan esta el error absoluto medio, error cuadrático medio y r-cuadrado (Vieira et al., 2019a).

1.5.6. Análisis post hoc

Este paso consiste en identificar las características que hicieron la mayor y menor contribución a la predicción y a su vez determinar la significancia estadística del modelo (Vieira et al., 2019a).

1.5.6.1. Significancia estadística

Lo más común en la realización de modelos de aprendizaje de máquina es estimar algunas métricas que ayuden a medir el rendimiento del modelo, a su vez se considera muy importante verificar la significancia estadística de alguna de las métricas estimadas anteriormente, con la finalidad de comprobar si el rendimiento del modelo es superior al nivel de probabilidad (Vieira et al., 2019a).

Fundamentalmente, una prueba de permutación mide la probabilidad de que el rendimiento del modelo se obtenga por pura casualidad. Donde para el aprendizaje de máquina de tipo supervisado, la variable objetivo se mezcla aleatoriamente entre todos los casos posibles hasta que se pierda cualquier tipo de relación estadística entre ellos y las características de entrada. Posteriormente se calcula el rendimiento de validación cruzada para cada permutación, obteniendo como resultado una distribución estadística de medidas, la cual refleja la hipótesis nula de que el modelo muestra un rendimiento debido a la casualidad. Para estimar un valor p, el número de ocasiones que el rendimiento es mayor o igual que el rendimiento original se divide por el número de permutaciones (Good, 1994).

1.6. Métodos para machine learning

Dhall et al. (2019) da a conocer algunos métodos utilizados para desarrollar aplicaciones de

aprendizaje de máquina, las cuales se muestran en el grafico 3-1.

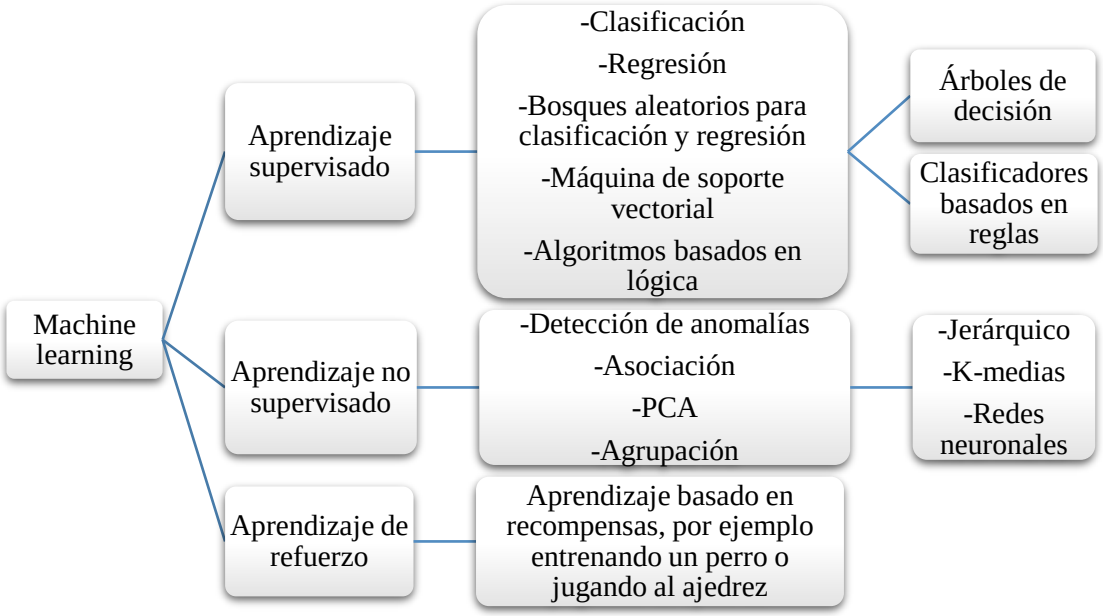


Gráfico 3-1: Modelos de machine learning

Fuente: Dhall et al., 2019

Realizado por: Vilema Pablo, 2021

A continuación, solo se hablará sobre el método de *Random Forest* (bosque aleatorio), debido a que este es el método que se utilizará para el desarrollo de la presente investigación. Para una mejor compresión del modelo de bosque aleatorio, en primer lugar, se debe conocer acerca de los árboles de decisión.

1.6.1. Árboles de decisión

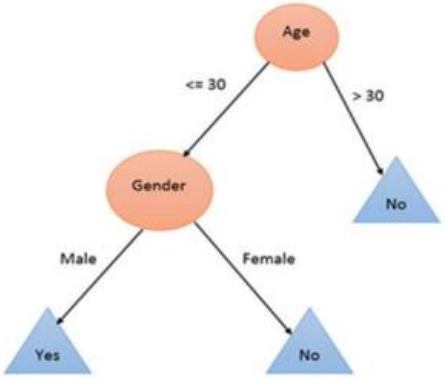


Figura 3-1: Árbol de decisión

Fuente: Dey, 2016

Los árboles de decisión son aquellos que reúnen atributos ordenándolos según sus valores. El árbol de decisión se lo usa principalmente para fines de clasificación, donde cada árbol está compuesto de nodos y ramas. Cada nodo representa atributos en un grupo que se va a clasificar y cada rama representa un valor que el nodo puede tomar (Dey, 2016).

1.6.2. Random Forest

Random Forest (bosque aleatorio), es un método de aprendizaje de máquina de conjunto para clasificación y regresión, está compuesto por varios modelos de árboles de decisión aleatorios DT. La decisión final tomada por el modelo de bosque aleatorio se basa en la predicción media de los modelos de DT individuales primarios involucrados. En cambio, el modelo de árboles de decisión se crea en función a un conjunto de características dado, donde el nodo original del DT se nombra raíz del árbol de decisión y los nodos divididos consecutivos se denominan ramas, mientras que los nodos finales se denominan hojas e indican el final de los datos. Los datos son ingresados por el nodo raíz y fluyen hacia los nodos hoja a través de las ramas, la salida de un DT se obtiene por la media de la regresión lineal de los puntos de datos de entrenamiento obtenidos del nodo hoja. Los conjuntos de datos de entrenamiento tienen la forma de árbol de decisiones, algo que hay que destacar es que los árboles no solo se entrenan en varios conjuntos de datos, sino que también utilizan diferentes características para tomar decisiones. Los modelos de DT se entrenan mediante la generación de múltiples conjuntos de datos, a través de la técnica de remuestreo *bootstraping*. Esta técnica es un proceso que se lleva a cabo para seleccionar arbitrariamente puntos de datos del conjunto de datos original, dicho proceso se repite varias veces donde un punto de datos del conjunto de datos original puede seleccionarse más de una vez o no ser seleccionado en lo absoluto. El proceso de *bootstraping* crea conjuntos de datos exclusivos que conforman un bosque de varios árboles de decisión (Vidyarthi et al., 2020).

Bosque aleatorio es considerado uno de los modelos de aprendizaje de máquina más fuertes y rápidos, debido a la arquitectura simple y robusta del árbol de datos original (Vidyarthi et al., 2020).

En la investigación realizada por Mohammady et al. (2019) menciona que el método de bosque aleatorio tiene una precisión relativamente alta, por lo que puede manejar de forma eficaz los datos faltantes tanto en los pasos de entrenamiento como de validación. Además, debido a su diseño de grupo, el método de bosque aleatorio puede predecir incluso cuando faltan algunos de los valores de entrada. Por otra parte Carvalho et al. (2019) dice que el modelo de bosque aleatorio es el más utilizado y comparado en el desarrollo de aplicaciones de mantenimiento predictivo.

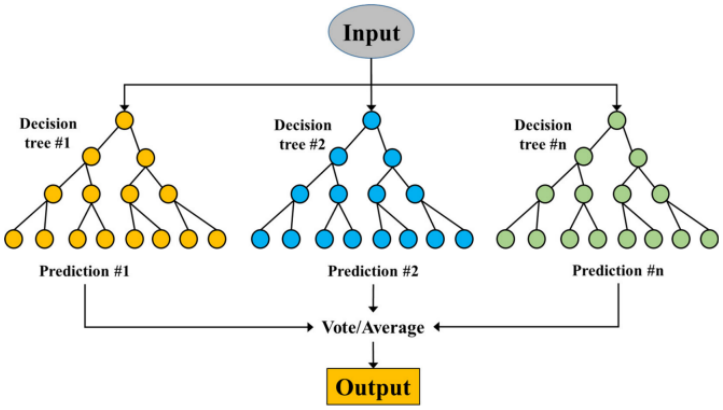


Figura 4-1: Flujo de proceso de bosque aleatorio

Fuente: Vidyarthi et al., 2020

1.6.2.1. Explicación matemática de Random forest

Como se mencionó anteriormente, el bosque aleatorio es un modelo clasificador conformado por conjuntos de clasificadores de árbol de decisión (Li et al., 2018), el cual se encuentra estructurado de la siguiente forma:

$$\{h(x, \theta_k), k=1,2,3.....n\}$$

Donde:

- θ_k son vectores aleatorios independientes e identicamente distribuidos
- x representa al vector de entrada

Al proporcionar un vector de entrada, cada árbol de decisión tiene un voto para seleccionar el resultado de clasificación óptimo.

Para Li et al. (2018) la generación de un bosque aleatorio incluye los 3 pasos siguientes:

- Primero, mediante la tecnica de *bootstrap* k conjuntos de muestras de entrenamiento se extraen del conjunto de muestras de entrenamiento original S, donde el tamaño de la muestra de cada conjunto de entrenamiento es consistente con el de S.
- Segundo, se aprenden k conjuntos de entrenamiento para posteriormente generar k modelos de árboles de decisión. Durante el proceso de generación de árboles de decisión, se asume que hay un total de M variables de entrada, y N variables se seleccionan al azar de M variables. Cada nodo interno se divide por el método de división óptima en las variables características N, y el valor N es constante durante la formación del modelo de bosque aleatorio.
- Tercero, los resultados de los k árboles de decisión se combinan para formar el resultado final. Para los problemas de clasificación, el método de combinación es un método de

votación por mayoría simple.

1.7. Diferencias entre árboles de decisión y Random Forest

Cabe mencionar que, aunque *Random Forest* es un conjunto de árboles de decisión, existen algunas diferencias que deben ser enfatizadas: los árboles de decisión generan reglas y nodos a partir del cálculo de la ganancia de información y el índice Gini, mientras que los bosques aleatorios generan árboles de decisión al azar. Además, aunque los árboles de decisión profundos pueden sufrir de sobreajuste, los bosques aleatorios evitan el sobreajuste en la mayoría de los casos de análisis, debido a que trabajan con subconjuntos aleatorios de características y construyen árboles más pequeños a partir de dichos subconjuntos (Carvalho et al., 2019).

1.8. Hiperparámetros

Los hiperparámetros son parámetros los cuales deben ser configurados antes de ejecutar un algoritmo de aprendizaje de máquina, a diferencia de los parámetros normales de un algoritmo que no se fijan antes de la ejecución, sino que se optimizan durante el entrenamiento del algoritmo. En todos los métodos de aprendizaje de máquina existen hiperparámetros que deben configurarse previamente; por ejemplo, el número de variables que se consideran en cada división en un bosque aleatorio, el número de pasos de impulso en el aumento de gradiente, el número de k en k vecinos más cercanos, el kernel en máquinas de vectores de soporte (Probst, 2019).

1.8.1. Ajuste de hiperparámetros y ajustes predeterminados

En los paquetes de software, generalmente hay hiperparámetros predeterminados, que para un problema dado posiblemente proporcionen buenos resultados si se configuran adecuadamente. La mayoría de las veces, ajustar los hiperparámetros, que significa encontrar un valor óptimo para ellos, puede conllevar a un mejor rendimiento que si se utilizara el valor predeterminado. Para algunos algoritmos, por ejemplo, para la máquina de vectores de soporte, el ajuste de los hiperparámetros es importante, ya que al establecerlos en valores óptimos se puede obtener grandes ganancias en el rendimiento del modelo (Probst, 2019).

1.8.2. Estrategias de ajuste de hiperparámetros

Existen varias estrategias para ajustar los hiperparámetros, idealmente la estrategia utilizada debería encontrar los mejores valores de hiperparámetros posibles en un tiempo muy reducido. Generalmente, se usa un método de evaluación para comparar diferentes hiperparámetros donde

el más conocido es la validación cruzada. Para evaluar todo el algoritmo, que incluye el procedimiento de ajuste, se debe realizar una validación cruzada anidada (Probst, 2019).

Una estrategia simple empleada para ajuste de hiperparámetros es la búsqueda de cuadrícula, la cual consiste en que para cada hiperparámetro se debe definir una cantidad finita de valores posibles y posteriormente se evalúan todas las combinaciones posibles de hiperparámetros. Otra estrategia sencilla es la búsqueda aleatoria, y consiste en que los hiperparámetros se extraen aleatoriamente de un espacio de hiperparámetros dado, por ejemplo, haciendo uso de la distribución uniforme (Probst, 2019).

Otros enfoques más sofisticados determinan iterativamente las especificaciones de los hiperparámetros, como por ejemplo la optimización bayesiana o también conocida como optimización basada en modelos. Para esta estrategia, un modelo sustituto se entrena con el rendimiento de los hiperparámetros ya ejecutados como salida y los hiperparámetros como entrada. Gracias a este modelo sustituto, se proponen nuevas especificaciones de hiperparámetros que cumplen dos requisitos: proporcionar buenos resultados de acuerdo con el modelo sustituto entrenado y deben estar en regiones del espacio hiperparámetro que aún no se han explorado (Probst, 2019).

1.8.3. Principales hiperparámetros en Random Forest

La tabla 2-1 muestra los principales hiperparámetros según Sun et al. (2020).

Tabla 2-1: Principales hiperparámetros en Random Forest

Hiperparámetro	Descripción
n_estimators	Número de árboles de decisión
criterion	Muestra de criterios de segmentación, incluidos Gini y entropía
min_samples_split	Número mínimo de muestras necesarias para dividir un nodo
max_depths	Profundidad máxima del árbol, por defecto, hasta que las muestras en todas las hojas sean muestras puras o el número de muestras sea menor que min_samples_split
max_features	Cantidad de características a considerar al buscar la mejor división.
min_samples_leaf	Número mínimo de puntos de datos permitidos en un nodo hoja.
bootstrap	Método para muestrear puntos de datos (con o sin reemplazo).

Fuente: Sun et al., 2020

Realizado por: Vilema Pablo, 2021

1.9. Métricas utilizadas para evaluar el rendimiento del modelo

Solo tomar en cuenta el valor de la precisión que fue obtenido por cualquier clasificador, como la medida de evaluación del desempeño del modelo de clasificación, no es la forma correcta; ya que la precisión del clasificador hace referencia solo al valor de las instancias clasificadas como pertenecientes a su clase real. La precisión no introduce las otras especificaciones del clasificador como: relación entre los atributos de los datos, la medida de la distribución correcta de las instancias de datos para todas y cada una de las clases posibles, el número de resultados positivos de entre todos los resultados positivos recibidos, entre otros, todos estos parámetros son muy necesarios durante la evaluación del desempeño de cualquier clasificador (Gianey y Choudhary, 2018).

1.9.1. Matriz de confusión

La matriz de confusión o también conocida como matriz de error, es un diseño de tabla específico que posibilita la visualización del rendimiento del algoritmo, y es mayormente empleada en el aprendizaje supervisado; cada fila de la matriz de confusión representa las instancias en una clase real, mientras que cada columna representa las instancias en una clase de predicción (o viceversa) (Haghighi et al., 2018). Por otra parte, desde el punto de vista de Gianey y Choudhary (2018) la matriz de confusión es una especie de tabla, que define el número de instancias de datos que están mal clasificadas y que están correctamente clasificadas.

Tabla 3-1: Matriz de confusión

		Predicho	
		0	1
Actual	0	TN	FP
	1	FN	TP

Fuente: Gianey y Choudhary, 2018

Realizado por: Vilema Pablo, 2021

- Verdadero negativo (TN): representa el número de predicciones negativas (0) que se clasificaron correctamente.
- Falso positivo (FP): representa el número de predicciones positivas (1) que fueron clasificadas de manera incorrecta.
- Falso negativo (FN): representa el número de predicciones negativas que fueron clasificadas incorrectamente.
- Verdadero positivo (TP): representa el número de predicciones positivas que se clasificaron correctamente (Gianey y Choudhary, 2018).

1.9.2. Precisión

La precisión es la tasa de positivos que se predijeron como positivos y en realidad también fueron positivos (Gianey y Choudhary, 2018). Se lo calcula como:

$$\frac{TP}{TP+FP} \tag{1}$$

1.9.3. Exactitud

Representa el número de predicciones correctas realizadas por el clasificador, se lo calcula como:

$$\frac{TP+TN}{TP+FP+FN+TN} \tag{2}$$

1.9.4. Sensibilidad (recall)

Representa la proporción de casos positivos reales que fueron clasificados correctamente, se lo calcula como:

$$\frac{TP}{TP+FN} \tag{3}$$

1.9.5. Especificidad

Representa la proporción de casos negativos reales que se clasificaron correctamente, se lo calcula como:

$$\frac{TN}{TN+FP} \tag{4}$$

1.9.6. Puntaje F1

Representa el promedio ponderado de la precisión y la sensibilidad y se lo calcula como:

$$\frac{2TP}{2TP+FP+FN} \tag{5}$$

1.9.7. Curva característica operativa del receptor (ROC)

Los modelos además de realizar una predicción categórica sobre la pertenencia a una clase también son capaces de generar la probabilidad de pertenecer a un determinado grupo. Para los

problemas de clasificación binaria, la mayoría de los clasificadores predicen observaciones a una clase u otra en función de un umbral de probabilidad, también conocido como umbral de clasificación que es igual a 0,5. Es viable definir el umbral de categorización y con base a ello realizar el cálculo de las métricas de sensibilidad y especificidad para cada valor de umbral posible (0 a 1). Posteriormente se puede graficar 1-especificidad (tasa de falsos positivos) versus la sensibilidad (tasa de verdaderos positivos) para cada nivel de umbral de clasificación posible, esto genera una curva conocida como curva de operación receptora (ROC). El rendimiento del clasificador se lo puede representar mediante un número, calculando el área bajo la curva ROC (AUC-ROC). El área bajo la curva para un clasificador ideal es 1, ya que la tasa de verdaderos positivos y la tasa de falsos negativos son 1 y 0 respectivamente. Un AUC-ROC de 0,5 significa que no hay discriminación, 0,7-0,8 se considera aceptable, 0,8-0,9 excelente y mayor a 0,9 se considera sobresaliente (Vieira et al., 2019a).

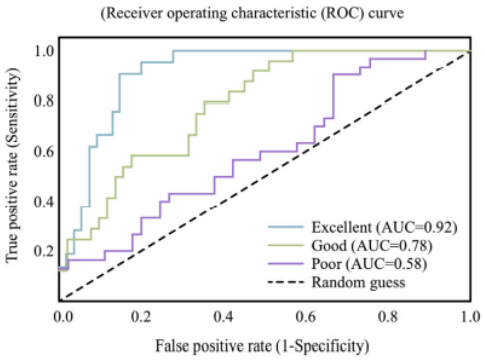


Figura 5-1: Curva ROC

Fuente: Vieira et al., 2019a

1.10. Overfitting y underfitting (sobreajuste y desajuste)

El sobreajuste y el desajuste son peligros constantes y omnipresentes en el desarrollo de modelos de machine learning. Uno de los tipos de aprendizaje en machine learning es el supervisado, donde el objetivo es aproximar o ajustar una señal verdadera que relaciona las características X con las respuestas Y (Bashir et al., 2020).

1.10.1. Overfitting

El sobreajuste (*overfitting* en inglés) se da cuando un algoritmo disminuye el error mediante la memorización de ejemplos de entrenamiento, con características ruidosas o irrelevantes, en lugar de aprender la verdadera relación general entre X e Y (Bashir et al., 2020). En palabras más sencillas el autor Chaoji et al. (2016) menciona que el sobreajuste se da cuando el modelo de *machine learning* se ajusta perfectamente a los datos de entrenamiento, pero no es capaz de generalizar a

datos de prueba o no vistos, el sobreajuste se puede combatir aumentando la cantidad de datos o haciendo que el modelo sea menos complejo.

1.10.2. Underfitting

El desajuste (*underfitting*) se da cuando un algoritmo carece de la capacidad de modelo suficiente o del entrenamiento suficiente para aprender completamente la verdadera relación entre X e Y, ya sea a través de la memorización o no (Bashir et al., 2020). Chaoji et al. (2016) menciona que el desajuste se lo puede combatir aumentando la complejidad del modelo.

1.10.3. Detección de overfitting y underfitting

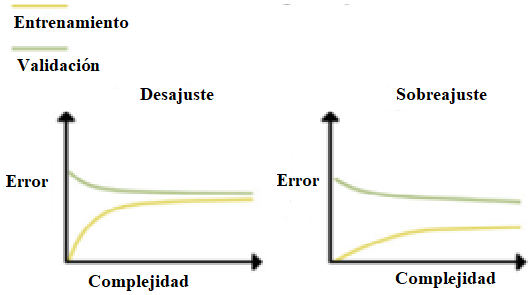


Figura 6-1: Curva de aprendizaje

Fuente: Vieira et al., 2019a

Uno de los métodos para detectar el sobreajuste y desajuste es la curva de aprendizaje. Para comprender dicha curva en primer lugar se debe conocer sobre el sesgo y la varianza, dos problemas a los cuales se enfrentan los modelos cuando aumentan su grado de complejidad. El sesgo se da cuando el modelo descubre una suposición errónea en los datos, en presencia de un sesgo muy elevado, el algoritmo no estará en la capacidad de aprender correctamente de los datos, esto se conoce como desajuste. Por otro lado, la varianza es el resultado de modelar fluctuaciones muy detalladas en los datos, un algoritmo con una varianza demasiado elevada aprenderá aspectos específicos de los datos de entrenamiento, los cuales no se pueden generalizar de una forma correcta en el conjunto de prueba, lo que se conoce como sobreajuste (Vieira et al., 2019a). Una curva de aprendizaje indica la variación del error del modelo, con respecto al tamaño del conjunto de datos de entrenamiento; es decir en el eje x del gráfico de la curva de aprendizaje se representa varios tamaños del conjunto de datos de entrenamiento y en el eje y se representa una métrica de error o de evaluación como la exactitud. Para un caso de elevada varianza o sobreajuste la gráfica muestra la existencia de una gran brecha entre los datos de prueba y los datos de entrenamiento, por lo que el error en el conjunto de entrenamiento es muy bajo debido a un buen ajuste, pero existe un error mucho mayor en el conjunto de prueba debido a que el modelo no es capaz de

generalizar. Por otro lado, cuando se tiene un sesgo muy elevado o desajuste se tiene una brecha muy pequeña, y el error tanto en el conjunto de entrenamiento como en el de prueba son elevados.

CAPÍTULO II

2. MARCO METODOLÓGICO

El presente capítulo describe la metodología que se empleó para la creación del modelo predictivo de detección de fallos, para lo cual se hizo uso del conjunto de datos de mantenimiento predictivo que se encuentra disponible en el repositorio de *machine learning* de la Universidad de California, Irvine (UCI). El modelo de detección de fallos es de tipo supervisado por lo que se dividirá el conjunto de datos para entrenamiento y prueba del modelo; para obtener las predicciones de los fallos se utilizará el método de aprendizaje de máquina *Random Forest*. La creación del modelo de *machine learning* para detectar fallos en mantenimiento predictivo, se la realizará de acuerdo a los 6 pasos mencionados en la investigación de Vieira et al. (2019a), los cuales fueron revisados en el capítulo anterior del presente trabajo de integración curricular.

2.1. Definición del problema

La base de datos de mantenimiento predictivo con la cual se llevará a cabo el trabajo de integración curricular puede ser analizada desde dos enfoques diferentes. Como primer enfoque se pueden realizar predicciones de la falla de la máquina, es decir si la máquina ha fallado o no; y como segundo enfoque se puede realizar predicciones sobre el modo de falla por el cual la máquina ha fallado, si se desearía abordar la investigación con los dos enfoques se debería realizar dos modelos de aprendizaje de máquina diferentes.

La presente investigación únicamente se basará en el primer enfoque, donde el problema es el siguiente: clasificar si la máquina ha fallado o no, utilizando datos de mantenimiento predictivo. Dada la formulación se puede identificar los elementos principales del problema y son:

- Características: variables de proceso, no se incluyen las variables de modos de fallo debido a que puede existir fugas de datos.
- Tarea: clasificación binaria
- Objetivo: falla de la máquina (estado 0 y 1, lo que significa que la máquina no ha fallado y si ha fallado respectivamente).

2.2. Preparación de datos

Para la presente investigación este paso se lo denominará preprocesamiento de datos, en primer lugar, se describe el conjunto de datos de mantenimiento predictivo.

Como pasos posteriores se realizará: análisis exploratorio de datos, limpieza de los datos, sobremuestreo y división de los datos para entrenamiento y prueba del modelo.

2.2.1. Descripción del conjunto de datos

El conjunto de datos de mantenimiento predictivo ai4i2020, es un conjunto de datos sintético que refleja datos reales de mantenimiento predictivo encontrados en la industria. (Dua y Graff, 2019). El conjunto de datos consta de 10000 puntos de datos almacenados como filas con 14 características en columnas, las cuales se describen a continuación:

- UDI: identificador único que va de 1 a 10000.
- ID de producto: consta de una letra L, M o H para baja (50% de todos los productos), media (30%) y alta (20%) respectivamente, como variantes de calidad del producto y un número de serie específico de la variante.
- Tipo: establece la letra correspondiente a la variante de calidad del producto.
- Temperatura del aire [°K]: generada mediante un proceso de recorrido aleatorio, que luego se normalizó a una desviación estándar de 2°K alrededor de 300 °K.
- Temperatura del proceso [°K]: generada mediante un proceso de recorrido aleatorio normalizado a una desviación estándar de 1 °K, sumado a la temperatura del aire más 10 °K.
- Velocidad de rotación [rpm]: calculada a partir de una potencia de 2860 W, superpuesta con un ruido distribuido normalmente.
- Torque [Nm]: los valores de par se distribuyen normalmente alrededor de 40 Nm con un Ïf = 10 Nm y sin valores negativos.
- Desgaste de la herramienta [min]: las variantes de calidad H/M/L añaden 5/3/2 minutos de desgaste de la herramienta a la herramienta utilizada en el proceso.
- Falla de la máquina: determina si la máquina ha fallado o no (Dua y Graff, 2019).

La falla de la máquina consta de cinco modos de falla independientes:

- Falla por desgaste de la herramienta (TWF): la herramienta será reemplazada o fallada en un tiempo de desgaste de la herramienta seleccionado al azar entre 200 y 240 minutos (46 veces en el conjunto de datos).
- Falla por disipación de calor (HDF): la disipación de calor provoca una falla en el proceso, si la diferencia entre la temperatura del aire y la del proceso es inferior a 8,6 °K y la velocidad de rotación de la herramienta es inferior a 1380 rpm. Este es el caso de 115 puntos de datos.
- Falla debido a la potencia (PWF): el producto del par y la velocidad de rotación (en rad / s) es igual a la potencia requerida para el proceso. Si esta potencia es inferior a 3500 W

o superior a 9000 W, el proceso falla, que es el caso 95 veces en nuestro conjunto de datos.

- Falla por sobre esfuerzo (OSF): si el producto del desgaste de la herramienta y el par supera los 11.000 minNm para la variante de producto L, 12.000 para M y 13.000 para H, el proceso falla debido al sobre esfuerzo. Esto es cierto para 98 puntos de datos.
- Fallos aleatorios (RNF): este es el caso de 19 puntos de datos (Dua y Graff, 2019).

Si al menos uno de los modos de falla anteriores es verdadero, el proceso falla y la etiqueta "falla de la máquina" se establece en 1, caso contrario se establece en 0 (Dua y Graff, 2019).

2.2.2. Librerías utilizadas para realizar el modelo de machine learning

Las librerías utilizadas para todo el desarrollo del modelo de *machine learning* no se cargan de manera predeterminada, por lo que se debe importarlas, existen algunas que vienen incluidas en el paquete de Python y otras que hay que instalarlas.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from warnings import filterwarnings
import sklearn
from sklearn.model_selection import train_test_split
import tsfel
from sklearn.feature_selection import RFE
from sklearn.preprocessing import RobustScaler
from sklearn.metrics import roc_auc_score,classification_report,
confusion_matrix,accuracy_score,precision_score, recall_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn import model_selection
from imblearn.over_sampling import SMOTE
from sklearn.inspection import permutation_importance
from pprint import pprint
from sklearn.model_selection import learning_curve
from sklearn.model_selection import StratifiedKFold, KFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn import metrics
from pathlib import Path
```

Figura 1-2: Librerías utilizadas para el modelo de machine learning

Realizado por: Vilema Pablo, 2021

2.2.3. Creación de carpetas para almacenar datos y resultados

Con la finalidad de contar con una buena organización al momento de elaborar el modelo de aprendizaje de máquina, se crean carpetas mediante la librería pathlib en las cuales se almacenarán los datos y los resultados de la investigación, lo cual se muestra en la figura 2-2.

```
progra_dir = Path('./Resultados')
progra_dir.mkdir(exist_ok=True)

experiment_name = 'Rf_mtto'
experiment_dir = progra_dir / experiment_name
experiment_dir.mkdir(exist_ok=True)

permutacion_dir = experiment_dir / 'Permutacion'
permutacion_dir.mkdir(exist_ok=True)

datos_dir= experiment_dir / 'Datos'
datos_dir.mkdir(exist_ok=True)
```

Figura 2-2: Creación de carpetas

Realizado por: Vilema Pablo, 2021

2.2.4. Análisis exploratorio de datos

2.2.4.1. Carga de datos

Los datos están almacenados como un archivo de valores separado por comas (csv), por lo que se utiliza la función read_csv () de pandas para cargar el archivo. Esta función permite cargar los archivos en un tipo de objeto llamado DataFrame, y para este caso se lo ha denominado Mpd.

```
Mpd= pd.read_csv(experiment_dir / 'Datos'/'ai4i2020.csv', header=0)
Mpd
```

	UDI	ID_Producto	Tipo	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina	TWF	HDF
0	1	M14860	M	298.1	308.6	1551	42.8	0	0	0	0
1	2	L47181	L	298.2	308.7	1408	46.3	3	0	0	0
2	3	L47182	L	298.1	308.5	1498	49.4	5	0	0	0
3	4	L47183	L	298.2	308.6	1433	39.5	7	0	0	0
4	5	L47184	L	298.2	308.7	1408	40.0	9	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	M24855	M	298.8	308.4	1604	29.5	14	0	0	0
9996	9997	H39410	H	298.9	308.4	1632	31.8	17	0	0	0
9997	9998	M24857	M	299.0	308.6	1645	33.4	22	0	0	0
9998	9999	H39412	H	299.0	308.7	1408	48.5	25	0	0	0
9999	10000	M24859	M	299.0	308.7	1500	40.2	30	0	0	0

10000 rows × 14 columns

Figura 3-2: Carga de datos

Realizado por: Vilema Pablo, 2021

2.2.4.2. Información de variables

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   UDI                   10000 non-null  int64
 1   ID_Producto           10000 non-null  object
 2   Tipo                  10000 non-null  object
 3   Temperatura_aire      10000 non-null  float64
 4   Temperatura_proceso   10000 non-null  float64
 5   Velocidad_rotacional  10000 non-null  int64
 6   Torque                10000 non-null  float64
 7   Desgaste_herramienta  10000 non-null  int64
 8   Falla_maquina         10000 non-null  int64
 9   TWF                   10000 non-null  int64
 10  HDF                   10000 non-null  int64
 11  PWF                   10000 non-null  int64
 12  OSF                   10000 non-null  int64
 13  RNF                   10000 non-null  int64
dtypes: float64(3), int64(9), object(2)
memory usage: 1.1+ MB
```

Figura 4-2: Información de variables

Realizado por: Vilema Pablo, 2021

La figura 4-2 muestra que existen un total de 14 variables, 3 de tipo float64 (números reales o con comas), 9 de tipo int64 (números enteros) y 2 de tipo object (contienen texto).

2.2.4.3. Valores faltantes y filas duplicadas

Es muy importante conocer si el conjunto de datos cuenta con valores faltantes y filas duplicadas, ya que los mismos podrían mermar el rendimiento del modelo.

```
# Número de datos faltantes por variable
Mpd.isna().sum().sort_values()
```

```
UDI                     0
ID_Producto             0
Tipo                    0
Temperatura_aire        0
Temperatura_proceso     0
Velocidad_rotacional    0
Torque                  0
Desgaste_herramienta    0
Falla_maquina           0
dtype: int64
```

Figura 5-2: Valores faltantes

Realizado por: Vilema Pablo, 2021

En la figura 5-2 se puede evidenciar que no existen valores faltantes en el conjunto de datos, por lo que no se hará uso de métodos de imputación, lo cual sería un procedimiento obligatorio en caso de existir valores faltantes.

```
print('Filas duplicadas : ',Mpd.duplicated().sum())
```

```
Filas duplicadas :  0
```

Figura 6-2: Filas duplicadas

Realizado por: Vilema Pablo, 2021

A su vez en la figura 6-2 se puede apreciar que la base de datos no cuenta con filas duplicadas, en caso de existirlas se las tiene que eliminar ya que sería información redundante para el modelo.

2.2.4.4. Análisis gráfico de la variable objetivo

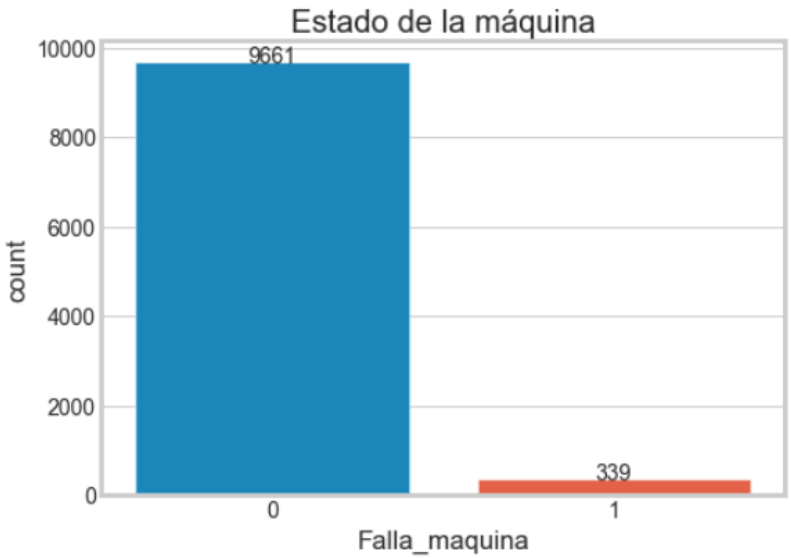


Gráfico 1-2: Observaciones para la variable Falla_maquina

Realizado por: Vilema Pablo, 2021

El gráfico 1-2 muestra los puntos de datos para cada una de las clases de la variable objetivo, donde se aprecia que para la clase 0 (representa cuando la máquina no ha fallado) hay en total 9661 puntos de datos y para la clase 1 (representa cuando la máquina ha fallado) hay 339, por lo que se puede concluir que los datos están desequilibrados, ya que hay muy pocos registros para la clase 1, para solucionar este problema más adelante se hará uso de un método de sobremuestreo para equilibrar los datos.

2.2.4.5. Análisis de correlación

A continuación, el gráfico 2-2 muestra el mapa de calor en el cual se puede evidenciar la correlación de las variables.

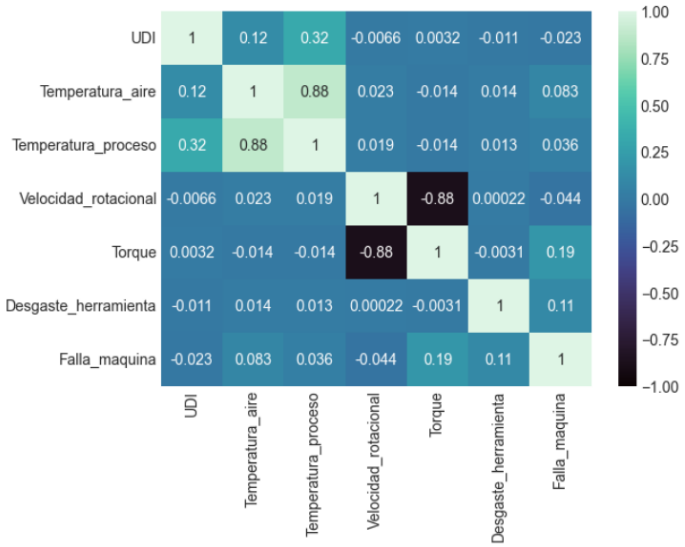


Gráfico 2-2: Mapa de calor

Realizado por: Vilema Pablo, 2021

Para poder visualizar el mapa de calor se debe comprender que existe correlación positiva cuando el valor se acerca a 1 y correlación negativa cuando el valor se acerca a -1. El grado de correlación en el gráfico 2-2 se lo puede visualizar de acuerdo a la intensidad del color guiándose en la escala de colores del lado derecho del gráfico y a su vez por el valor que se encuentra dentro de cada celda del mapa de calor. Por lo que se puede evidenciar que los pares de variables correlacionados son: Temperatura_proceso con Temperatura_aire y Velocidad_rotacional con Torque.

2.2.4.6. Análisis gráfico de variables de proceso

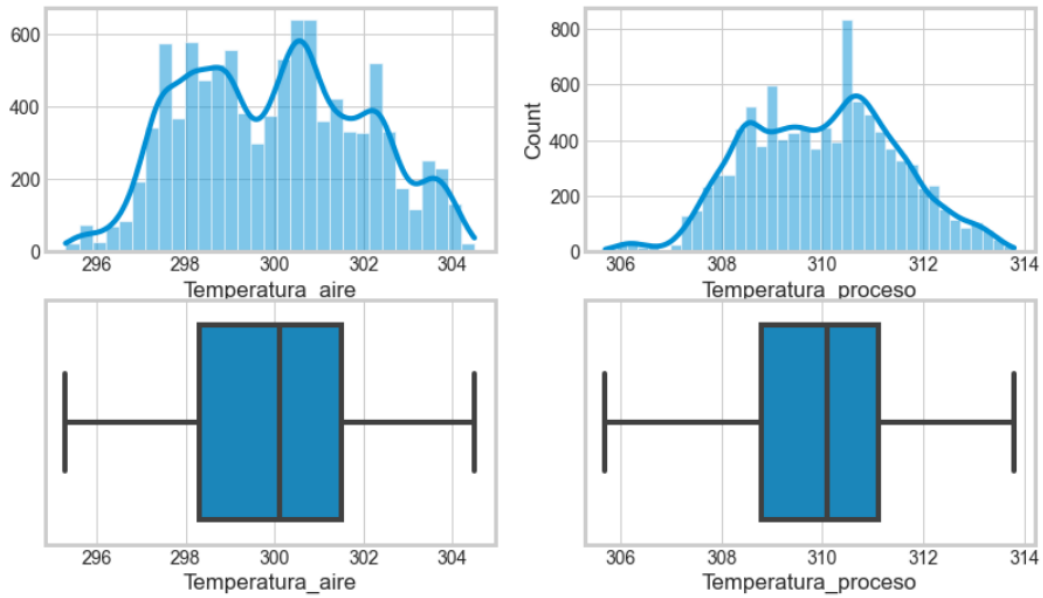


Gráfico 3-2: Histograma y diagrama de cajas para temperatura del aire y de proceso

Realizado por: Vilema Pablo, 2021

El gráfico 3-2 mediante los histogramas muestra que las dos variables analizadas no se ajustan a una distribución normal y de acuerdo al diagrama de cajas tampoco presentan valores atípicos.

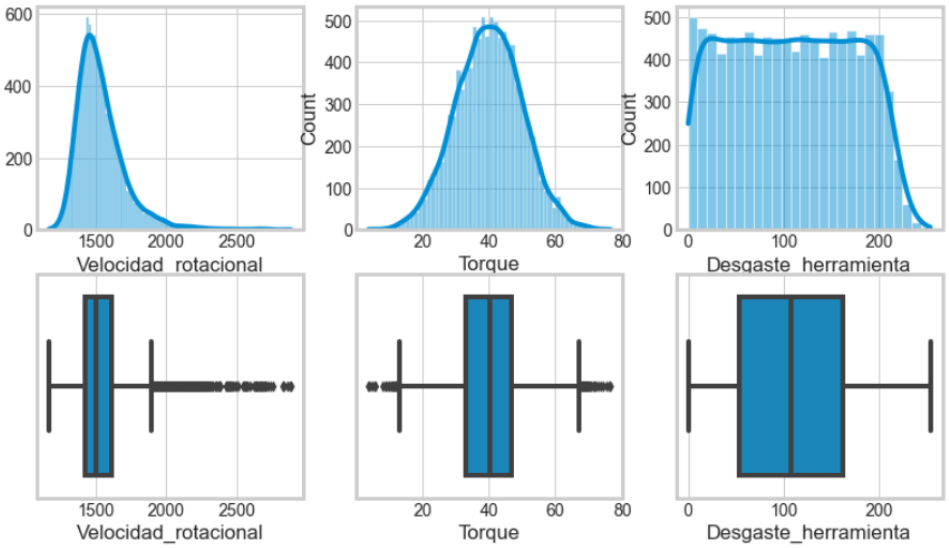


Gráfico 4-2: Histograma y diagrama de cajas para velocidad, torque y desgaste

Realizado por: Vilema Pablo, 2021

Del gráfico 4-2 se puede concluir que la única variable que se ajusta a una distribución normal es la de Torque, y de acuerdo a los diagramas de cajas las variables Velocidad_rotacional y Torque contienen algunos valores atípicos. Debido a que se utilizará el método de *Random Forest* y no es sensible a valores atípicos, no se eliminará ninguno de esos valores, ya que se perderían dichos datos los cuales pueden ser muy valiosos para el análisis. A su vez antes de ingresar los datos en el clasificador se los escalarán y normalizarán mediante la función RobustScaler, la cual ayuda a minimizar el impacto de los valores atípicos, pero sin eliminarlos.

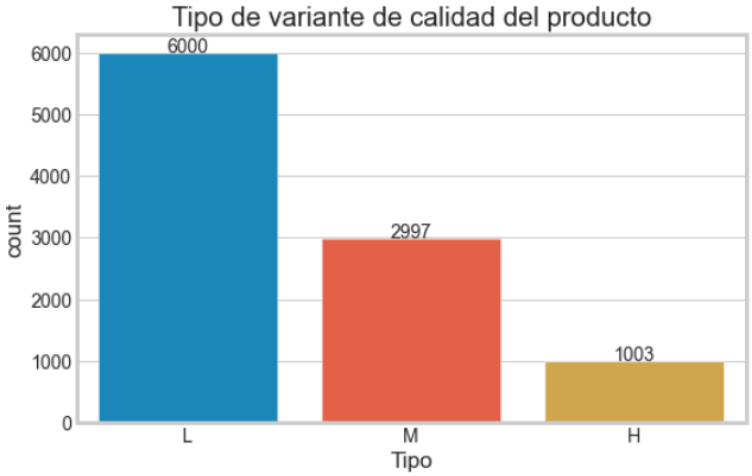


Gráfico 5-2: Datos para cada variante de calidad de producto

Realizado por: Vilema Pablo, 2021

Del gráfico 5-2 se puede concluir que la variante L es la que mayor predomina en el conjunto de datos, seguido de la variable M y finalmente la variable H, con 6000/2997/1003 puntos de datos respectivamente.

2.2.4.7. Gráficos de dispersión para algunos pares de variables

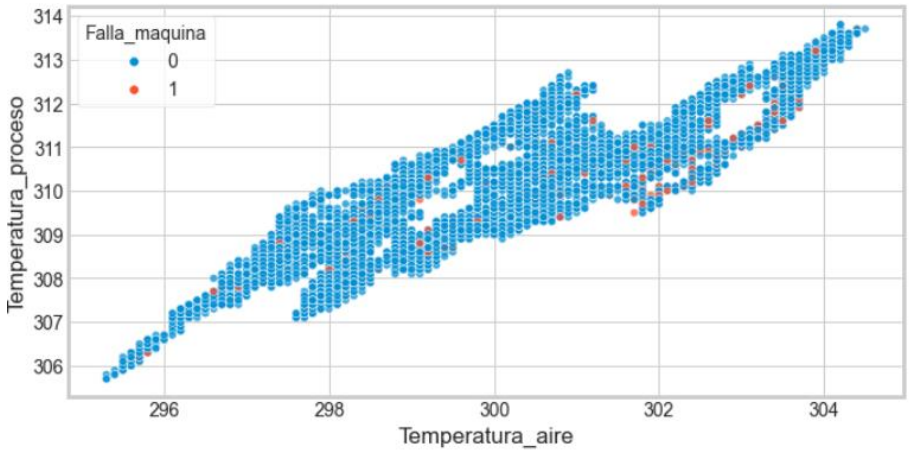


Gráfico 6-2: Dispersión entre temperatura del aire vs temperatura de proceso

Realizado por: Vilema Pablo, 2021

Del gráfico 6-2 se puede concluir que existe una relación lineal positiva, ya que al aumentar la temperatura del aire aumenta la temperatura de proceso, por otro lado, se aprecia que la máquina falla en la mayoría de las veces a partir de los 309 °K en la temperatura de proceso y a partir de los 298 °K en la temperatura del aire.

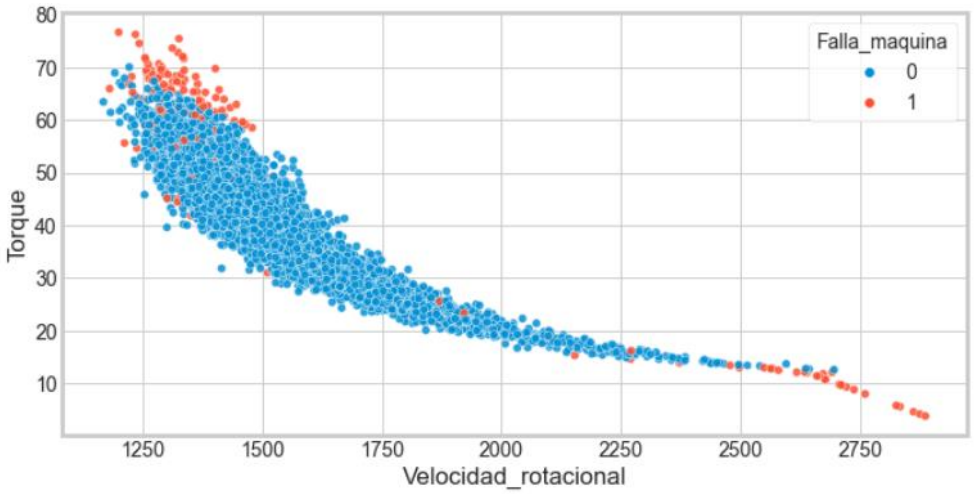


Gráfico 7-2: Dispersión entre velocidad rotacional vs torque

Realizado por: Vilema Pablo, 2021

En el gráfico 7-2 se aprecia que existe una relación lineal negativa entre los pares de variables analizados, ya que al aumentar la velocidad rotacional disminuye el torque. Por otra parte, se

aprecia que la máquina falla en la mayoría de las veces cuando tiene una velocidad rotacional baja y un torque elevado, aunque también se aprecia una proporción minina de fallos a una velocidad rotacional elevada y un torque bajo.

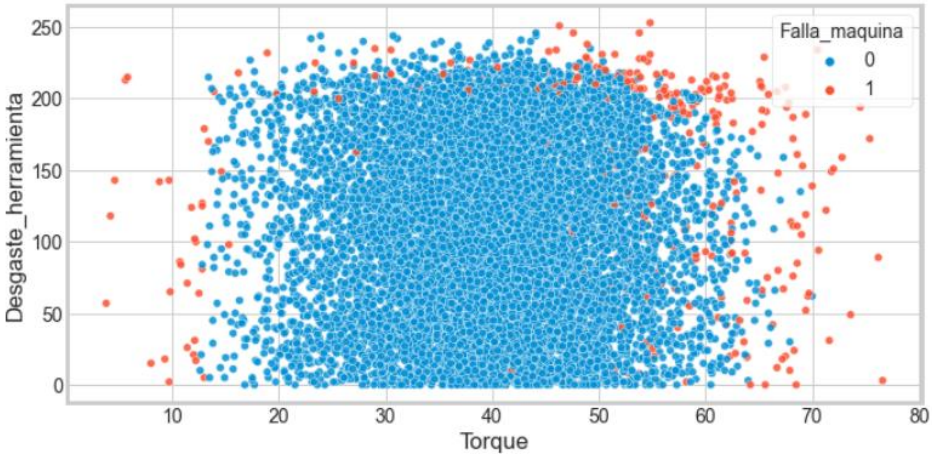


Gráfico 8-2: Dispersión entre torque y desgaste de herramienta

Realizado por: Vilema Pablo, 2021

El gráfico 8-2 muestra que no existe ningún tipo de relación entre los pares de variables analizados, por otra parte, se aprecia que la máquina falla en mayor número de veces cuando el torque es mayor a 40 Nm y el desgaste de herramienta es mínimo, medio y elevado; pero también la máquina falla en menos ocasiones cuando el torque es inferior a 40 Nm.

2.2.5. Limpieza de datos

La limpieza de los datos es un paso muy importante dentro de la preparación de los mismos, ya que gracias a este proceso se puede identificar errores y corregirlos en caso de existirlos, y a su vez se pueden eliminar variables que no contenga buena información para el desarrollo del modelo, esto con la finalidad de obtener un buen rendimiento en la predicción.

2.2.5.1. Eliminación de variables

Como se mencionó en el paso de definición del problema, las variables referentes a los modos de fallo no serán empleadas para el modelado, así como también las variables UDI, ID_Producto y Tipo, ya que se considera que no aportan en la predicción y el trabajar con las mismas para el desarrollo del modelo puede conllevar a un sobreajuste del mismo. La figura 7-2 muestra la base de datos una vez que las variables fueron eliminadas, mostrando únicamente las variables de proceso y la variable objetivo.

	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
0	298.1	308.6	1551	42.8	0	0
1	298.2	308.7	1408	46.3	3	0
2	298.1	308.5	1498	49.4	5	0
3	298.2	308.6	1433	39.5	7	0
4	298.2	308.7	1408	40.0	9	0
...	...	...	...	...	...	...
9995	298.8	308.4	1604	29.5	14	0
9996	298.9	308.4	1632	31.8	17	0
9997	299.0	308.6	1645	33.4	22	0
9998	299.0	308.7	1408	48.5	25	0
9999	299.0	308.7	1500	40.2	30	0

10000 rows × 6 columns

Figura 7-2: Eliminación de variables

Realizado por: Vilema Pablo, 2021

2.2.6. Sobremuestreo

De acuerdo al análisis gráfico de la variable objetivo Falla_maquina realizado en el proceso de análisis exploratorio de datos, se pudo evidenciar claramente que existe desequilibrio de datos en las clases, por lo que para solucionar dicho problema se hará uso de la técnica de sobremuestreo SMOTE, la cual consiste en crear nuevos datos sintéticos con la finalidad de tratar de equilibrar las clases.

	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
0	298.100000	308.600000	1551	42.800000	0	0
1	298.200000	308.700000	1408	46.300000	3	0
2	298.100000	308.500000	1498	49.400000	5	0
3	298.200000	308.600000	1433	39.500000	7	0
4	298.200000	308.700000	1408	40.000000	9	0
...	...	...	...	...	...	...
19123	301.965492	310.556760	1340	55.320145	204	1
19124	302.393361	310.776660	1375	47.346681	19	1
19125	300.650682	309.813855	1311	65.053050	191	1
19126	297.532248	308.489886	1374	54.735573	215	1
19127	300.623357	311.129840	1321	67.231172	18	1

19128 rows × 6 columns

Figura 8-2: Sobremuestreo de datos

Realizado por: Vilema Pablo, 2021

Una vez aplicado el código de sobremuestreo al conjunto de datos, se puede evidenciar en la esquina inferior izquierda de figura 8-2, que se pasa de 10000 puntos de datos a 19128 es decir se crearon 9128 datos nuevos.

A continuación, el gráfico 9-2 muestra los puntos de datos para cada clase de la variable Falla_maquina una vez hecho el sobremuestreo.

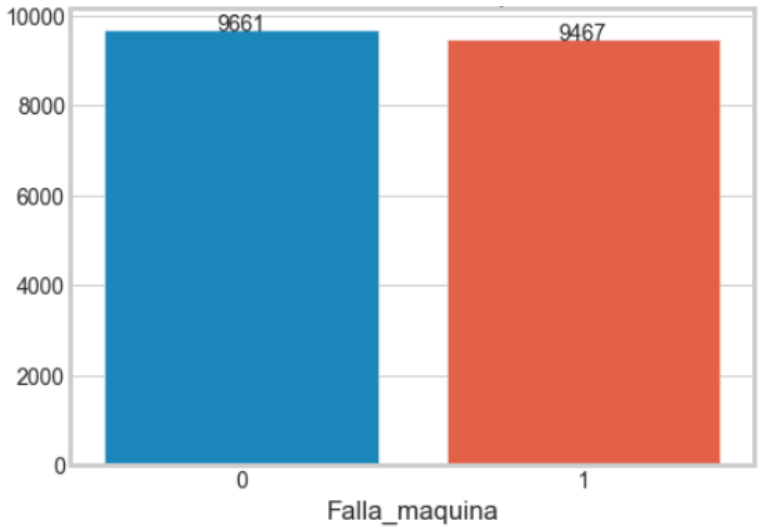


Gráfico 9-2: Puntos de datos para cada clase con sobremuestreo.

Realizado por: Vilema Pablo, 2021

En el gráfico 9-2 se evidencia que se incrementó de 339 puntos de datos a 9467, correspondientes a la clase 1. De esta forma ya se cuenta con un conjunto de datos equilibrado, por otra parte, se aprecia que para la clase 0 se mantuvieron los mismos datos, es decir el sobremuestreo se aplicó solo a la clase minoritaria.

2.2.7. División del conjunto de datos para entrenamiento y prueba

```
X1= Mpd1_new.copy() #Definiendo variables
y1= X1.pop('Falla_maquina')
```

```
X1_train, X1_test, y1_train, y1_test = train_test_split(X1,y1, random_state=0, test_size= 0.25)
```

```
y1_train.value_counts() # Observaciones de entrenamiento para cada clase
```

```
0    7252
1    7094
Name: Falla_maquina, dtype: int64
```

```
y1_test.value_counts() # Observaciones de prueba para cada clase
```

```
0    2409
1    2373
```

Figura 9-2: Definición de variables y aplicación del código de división de datos

Realizado por: Vilema Pablo, 2021

El conjunto de datos se lo debe dividir en 2: conjunto de datos para entrenamiento y conjunto de datos para prueba del modelo, una de las formas más comunes para dividir los datos es 75% para

entrenamiento y 25% para prueba, por lo que los datos de mantenimiento predictivo empleados para la presente investigación serán divididos de esa manera. Cabe acotar que para la división de los datos se aplicará el conjunto de datos equilibrado.

La figura 9-2 muestra la forma en cómo se definió las variables, donde la variable objetivo es Falla_maquina denotada por y1 y la independiente está compuesta por todas las variables de proceso y está denotada por X1. A su vez al aplicar el código se generan los datos tanto para entrenamiento y prueba de las dos variables. También se aprecia que para el entrenamiento se designó 7252 datos referentes a la clase 0 y 7094 referentes a la clase 1, por otra parte, para la prueba se designó 2409 para la clase 0 y 2373 para la clase 1; lo que concuerda a un 75% y 25% respectivamente del total de datos.

Posterior a ello se concatenan los archivos X1_train con y1_train y X1_test con y1_test con la finalidad de tener el conjunto de entrenamiento y el de prueba; la cantidad final de datos que contiene cada conjunto se lo puede visualizar en el gráfico 10-2.

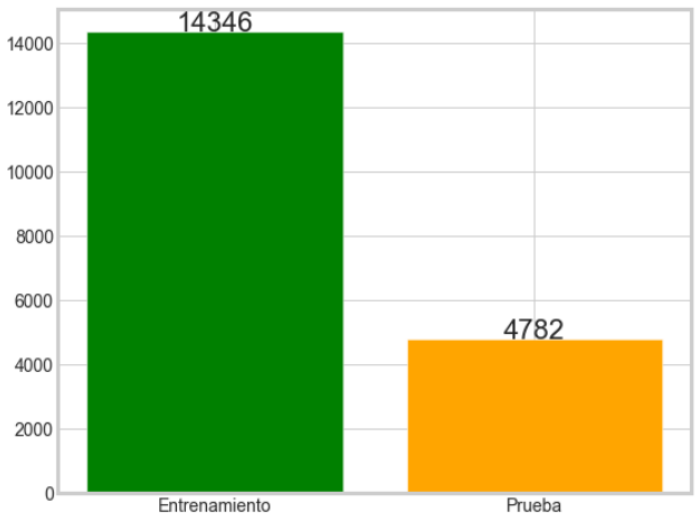


Gráfico 10-2: Datos para entrenamiento y prueba

Realizado por: Vilema Pablo, 2021

2.3. Ingeniería de características

El conjunto de datos de mantenimiento predictivo ai4i2020 únicamente contiene variables que están en el dominio de tiempo es decir series temporales, por ello es que solo se extraerán ese tipo de características y no se extraerán características en el dominio de la frecuencia.

2.3.1. Extracción de características

Una vez creados anteriormente el conjunto de datos para entrenamiento y prueba se procede a extraer las características de cada uno de ellos. Para este fin se hizo uso de la librería de Python TSFEL, ya que permite calcular automáticamente una gran cantidad de características basadas en el dominio del tiempo. Para profundizar el conocimiento sobre esta librería se recomienda revisar Barandas et al. (2020). Las características estadísticas en el dominio del tiempo que fueron extraídas se dan a conocer en la tabla 1-2.

Tabla 1-2: Características en el dominio del tiempo

Característica	Fórmula
Energía absoluta	$t1=\sum_{n=1}^{N}(x(n))^2$
Media	$t2=\frac{1}{N}\sum_{n=1}^{N}x(n)$
Raíz media cuadrada	$t3=\sqrt{\frac{1}{N}\sum_{n=1}^{N}(x(n))^2}$
Valor máximo	$t4=\max(x(n))$
Valor mínimo	$t5=\min(x(n))$
Varianza	$t6=\frac{1}{N}\sum_{n=1}^{N}(x_n-t2)^2$
Desviación estándar	$t7=\sqrt{\frac{1}{N}\sum_{n=1}^{N}(x_n-t2)^2}$
Curtosis	$t8=\frac{N\sum_{n=1}^{N}(x_n-t2)^4}{\left[\sum_{n=1}^{N}(x_n-t2)^2\right]^2}$
Asimetría	$t9=\frac{N\sum_{n=1}^{N}(x_n-t2)^3}{(t7)^3}$
Rango intercuartil	$t10=Q_3-Q_1$
Distancia pico-pico	$t11=\max(x_n)-\min(x_n)$
Mediana	$t12=x_{\frac{N+1}{2}}$

Realizado por: Vilema Pablo, 2021

Donde x(n) es una serie temporal con N puntos de datos

2.3.1.1. Extracción de características del conjunto de entrenamiento

El primer paso hacia la extracción de características consiste en cargar el conjunto de datos de

entrenamiento, en la realización de este proceso se añade por defecto una columna la cual debe ser eliminada, la figura 10-2 muestra el conjunto de datos de entrenamiento una vez que la variable creada por defecto fue eliminada.

	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
0	296.982889	307.558022	2720	9.346418	18	1
1	297.520958	308.211643	1403	61.481371	109	1
2	298.000000	308.900000	1607	37.700000	49	0
3	300.535212	311.544600	1271	58.839436	197	1
4	300.400000	311.100000	1405	54.100000	193	0
...	...	...	...	...	...	...
14341	298.000000	309.100000	1820	23.600000	120	0
14342	300.400799	309.635297	1299	65.368996	48	1
14343	298.300000	309.100000	1421	47.400000	33	0
14344	301.925735	310.100000	1360	55.311397	66	1
14345	299.700000	309.200000	1346	57.400000	138	0

14346 rows × 6 columns

Figura 10-2: Conjunto de datos de entrenamiento

Realizado por: Vilema Pablo, 2021

Luego de cargar los datos y eliminar la variable creada por defecto, se llama únicamente a las observaciones correspondientes a la clase 0 de la variable objetivo Falla_maquina, tal y como se aprecia en la figura 11-2.

	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
2	298.0	308.9	1607	37.7	49	0
4	300.4	311.1	1405	54.1	193	0
6	297.4	308.8	1640	34.2	142	0
8	301.6	310.7	1679	29.5	10	0
13	297.7	307.5	1379	43.3	111	0
...	...	...	...	...	...	...
14330	300.6	309.5	1530	40.2	93	0
14336	301.7	311.0	1298	58.1	168	0
14341	298.0	309.1	1820	23.6	120	0
14343	298.3	309.1	1421	47.4	33	0
14345	299.7	309.2	1346	57.4	138	0

7252 rows × 6 columns

Figura 11-2: Datos para la clase 0

Realizado por: Vilema Pablo, 2021

En la figura 11-2 se aprecia que efectivamente las observaciones para la clase 0 del conjunto de entrenamiento son 7252 tal y como se lo dio a conocer en el paso la división de los datos.

Posteriormente se definen variables con la finalidad separar la variable objetivo de las variables proceso, ya que las características únicamente se extraen de las variables predictoras que para en este caso son las variables de proceso. Seguido de ello se extraen 12 características por cada variable y al contar con 5 el total de características extraídas es de 60; para extraer características mediante la librería TSFEL se debe modificar un parámetro llamado tamaño de ventana, el cual determina la cantidad de datos a agruparse para extraer las características. En este caso específicamente se definió un tamaño de ventana de 5 datos, es decir TSFEL realiza una división del total de observaciones (7252) y las divide para 5, dando como resultado 1450,4 observaciones para la clase 0, pero la librería únicamente toma el valor entero sin redondear, es decir 1450.

	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Kurtosis	4_Max	4_Mean
0	447078.77	2.7	-1.489525	301.6	299.02	298.0	297.4	4.2	299.024671	0.522093	...	-1.331958	193.0	101.0
1	443969.87	0.6	-0.154622	300.6	297.98	297.5	296.6	4.0	297.983177	1.146014	...	-1.473279	156.0	92.4
2	452585.85	0.5	-0.785088	301.8	300.86	301.0	299.8	2.0	300.860715	-0.256011	...	-1.286580	208.0	117.6
3	450871.94	4.5	-1.609320	303.6	300.28	299.8	297.1	6.5	300.290506	0.119880	...	-0.926514	229.0	124.8
4	447966.00	1.4	-1.517590	300.5	299.32	299.1	298.2	2.3	299.321232	0.146392	...	-0.298353	223.0	157.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1445	453023.96	2.2	-0.987101	303.4	301.00	300.8	297.8	5.6	301.006299	-0.408224	...	-0.577177	160.0	77.6
1446	447678.71	3.2	-1.544148	301.8	299.22	298.8	297.3	4.5	299.225236	0.294745	...	-0.687819	112.0	67.6
1447	446361.73	3.7	-1.763285	300.6	298.78	299.2	296.8	3.8	298.784782	-0.179426	...	-1.623260	222.0	120.8
1448	452000.71	1.1	-0.491272	304.1	300.66	300.6	298.3	5.8	300.666164	0.739175	...	-0.260252	219.0	99.2
1449	451509.67	0.5	-0.080571	301.7	300.50	301.1	298.0	3.7	300.502802	-1.232228	...	-1.389254	202.0	136.8

1450 rows × 60 columns

Figura 12-2: Características extraídas para la clase 0

Realizado por: Vilema Pablo, 2021

En la figura 12-2 se puede apreciar algunas de las características extraídas, pero en la parte inferior izquierda de la figura se comprueba que efectivamente se cuenta con 60 columnas (características) y 1450 observaciones producto de la aplicación del tamaño de ventana.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1450 entries, 0 to 1449
Data columns (total 60 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   0_Absolute energy       1450 non-null   float64
 1   0_Interquartile range   1450 non-null   float64
 2   0_Kurtosis              1450 non-null   float64
 3   0_Max                   1450 non-null   float64
 4   0_Mean                  1450 non-null   float64
 5   0_Median                1450 non-null   float64
 6   0_Min                   1450 non-null   float64
 7   0_Peak to peak distance 1450 non-null   float64
 8   0_Root mean square      1450 non-null   float64
 9   0_Skewness              1450 non-null   float64
 10  0_Standard deviation    1450 non-null   float64
 11  0_Variance              1450 non-null   float64
```

Figura 13-2: Información de características extraídas

Realizado por: Vilema Pablo, 2021

La figura 13-2 muestra información sobre las características extraídas referente a la variable Temperatura_aire, no se muestra la información de todo el conjunto de características debido a que la información es extensa, pero las mismas pueden ser revisadas en la sección de anexos. Cabe acotar que el número 0 hace referencia a la variable Temperatura_aire, 1 a Temperatura_proceso, 2 a Velocidad_rotacional, 3 a Torque y 4 a Desgaste_herramienta.

Una vez culminada la extracción de características de la clase 0, se procede a realizar los mismos pasos descritos anteriormente con la finalidad de extraer las características para la clase 1, por lo que la figura 14-2 muestra los datos únicamente para esta clase donde se comprueba que son 7094 tal y como se lo definió en la división de datos.

	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
0	296.982889	307.558022	2720	9.346418	18	1
1	297.520958	308.211643	1403	61.481371	109	1
3	300.535212	311.544600	1271	58.839436	197	1
5	300.300700	310.587514	1296	62.381915	204	1
7	302.578528	311.219516	1353	45.618948	194	1
...	...	...	...	...	...	...
14338	297.969231	308.365898	1332	72.125128	152	1
14339	300.821667	310.278422	2533	12.895497	67	1
14340	302.462686	311.985684	1263	69.748709	232	1
14342	300.400799	309.635297	1299	65.368996	48	1
14344	301.925735	310.100000	1360	55.311397	66	1

7094 rows × 6 columns

Figura 14-2: Datos para la clase 1

Realizado por: Vilema Pablo, 2021

Posteriormente se define variables y luego se extrae la misma cantidad y las mismas características que para la clase 0.

	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Kurtosis
0	452422.945719	1.513788	-0.835553	303.328547	300.800946	300.525158	297.799403	5.529144	300.806564	-0.306394	...	-0.197081
1	455883.326875	1.888365	-1.368508	303.400000	301.952190	301.627143	300.095770	3.304230	301.954741	-0.139856	...	-1.763425
2	455012.703803	0.168048	-0.033441	303.476237	301.660710	302.277812	298.127141	5.349096	301.666274	-1.221004	...	-1.572947
3	455064.102854	1.537655	-0.316731	303.200000	301.678024	302.001024	298.328927	4.871073	301.683312	-1.070155	...	-1.823420
4	451345.928631	2.954081	-1.791925	302.300000	300.444843	299.662867	299.050766	3.249234	300.448308	0.353160	...	-0.727256
...	...	...	...	...	...	...	...	...	...	...	...	...
1413	452329.146274	2.467379	-0.684972	302.627149	300.767075	302.121132	296.729446	5.897703	300.775380	-0.949393	...	-0.487775
1414	453458.968710	1.026337	-0.916127	302.269229	301.149279	301.034075	299.550389	2.718840	301.150782	-0.501981	...	-1.569647
1415	450090.026505	0.865367	-0.090233	301.140970	300.026874	300.544388	297.380536	3.760434	300.030007	-1.255092	...	-1.639312
1416	454332.044830	1.939130	-1.331747	303.700000	301.437143	300.622443	299.821276	3.878724	301.440556	0.494651	...	0.239320
1417	454171.266248	0.576698	0.140102	302.487025	301.382706	302.196860	298.121028	4.365997	301.387215	-1.422661	...	-1.144438

1418 rows × 60 columns

Figura 15-2: Características extraídas para la clase 1

Realizado por: Vilema Pablo, 2021

El tamaño de ventana sigue siendo de 5 por lo que el número de observaciones se reduce a 1418, esto se puede comprobar en figura 15-2.

Una vez culminada con la extracción de características para la clase 0 y para la clase 1 se procede añadir la variable objetivo a cada conjunto de características extraídas, es decir se crea una columna nueva con el nombre de la variable objetivo Falla_maquina.

0_Skewness	...	4_Max	4_Mean	4_Median	4_Min	4_Peak to peak distance	4_Root mean square	4_Skewness	4_Standard deviation	4_Variance	Falla_maquina
0.522093	...	193.0	101.0	111.0	10.0	183.0	120.195674	-0.032941	65.161338	4246.00	0
1.146014	...	156.0	92.4	72.0	23.0	133.0	104.856092	0.038961	49.568538	2457.04	0
-0.256011	...	208.0	117.6	125.0	48.0	160.0	131.454935	0.233179	58.742148	3450.64	0
0.119880	...	229.0	124.8	103.0	60.0	169.0	138.780402	0.693153	60.703871	3684.96	0
0.146392	...	223.0	157.6	165.0	45.0	178.0	168.802844	-0.997704	60.470158	3656.64	0
...	...	...	...	...	...	...	...	...	...	...	...
-0.408224	...	160.0	77.6	68.0	19.0	141.0	90.631120	0.672322	46.821363	2192.24	0
0.294745	...	112.0	67.6	62.0	29.0	83.0	72.781866	0.303743	26.971096	727.44	0
-0.179426	...	222.0	120.8	89.0	13.0	209.0	146.247735	0.129558	82.436400	6795.76	0
0.739175	...	219.0	99.2	88.0	28.0	191.0	118.164292	1.000152	64.204050	4122.16	0
-1.232228	...	202.0	136.8	120.0	93.0	109.0	143.023075	0.475839	41.729606	1741.36	0

Figura 16-2: Variable objetivo añadida a las características de clase 0

Realizado por: Vilema Pablo, 2021

La figura 16-2 muestra la variable objetivo añadida al conjunto de características extraídas para la clase 0 y a continuación la figura 17-2 muestra la variable objetivo añadida al conjunto de características extraídas para la clase 1.

0_Skewness	...	4_Max	4_Mean	4_Median	4_Min	4_Peak to peak distance	4_Root mean square	4_Skewness	4_Standard deviation	4_Variance	Falla_maquina
0.038924	...	204.0	144.4	194.0	18.0	186.0	161.410037	-0.844715	72.123782	5201.84	1
-0.395591	...	201.0	125.0	107.0	97.0	104.0	130.888502	1.352164	38.817522	1506.80	1
0.197918	...	227.0	179.2	200.0	62.0	165.0	188.949729	-1.339403	59.911268	3589.36	1
-0.410828	...	193.0	67.2	24.0	0.0	193.0	97.539735	0.874080	70.697666	4998.16	1
-0.922897	...	234.0	153.8	157.0	12.0	222.0	171.976161	-0.951209	76.950374	5921.36	1
...	...	...	...	...	...	...	...	...	...	...	...
-0.706806	...	205.0	156.8	179.0	58.0	147.0	165.221064	-1.148840	52.074562	2711.76	1
1.035544	...	200.0	167.4	180.0	119.0	81.0	169.783980	-0.654354	28.352072	803.84	1
-0.596765	...	216.0	112.6	91.0	65.0	151.0	124.713271	1.244283	53.615669	2874.64	1
-0.950318	...	217.0	132.2	173.0	6.0	211.0	152.170299	-0.621062	75.358875	5678.96	1
-0.100553	...	207.0	148.6	188.0	4.0	203.0	166.238985	-1.295802	74.521406	5553.44	1

Figura 17-2: Variable objetivo añadida a las características de clase 1

Realizado por: Vilema Pablo, 2021

Como último paso para concluir con la extracción de características para el conjunto de entrenamiento, se concatenan los dos conjuntos de características extraídas (clase 0 y clase 1) los cuales ya se encuentran agregados la variable objetivo, el conjunto final de entrenamiento se

muestra en la figura 18-2 donde se aprecia que el número total de observaciones es de 2868, lo que representa la suma de observaciones de la clase 0 (1450) y de la clase 1 (1418).

	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Max	4_Mean
0	447078.770000	2.700000	-1.489525	301.600000	299.020000	298.000000	297.400000	4.200000	299.024671	0.522093	...	193.0	101.0
1	443969.870000	0.600000	-0.154622	300.600000	297.980000	297.500000	296.600000	4.000000	297.983177	1.146014	...	156.0	92.4
2	452585.850000	0.500000	-0.785088	301.800000	300.860000	301.000000	299.800000	2.000000	300.860715	-0.256011	...	208.0	117.6
3	450871.940000	4.500000	-1.609320	303.600000	300.280000	299.800000	297.100000	6.500000	300.290506	0.119880	...	229.0	124.8
4	447966.000000	1.400000	-1.517590	300.500000	299.320000	299.100000	298.200000	2.300000	299.321232	0.146392	...	223.0	157.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1413	452329.146274	2.467379	-0.684972	302.627149	300.767075	302.121132	296.729446	5.897703	300.775380	-0.949393	...	210.0	80.4
1414	453458.968710	1.026337	-0.916127	302.269229	301.149279	301.034075	299.550389	2.718840	301.150782	-0.501981	...	219.0	154.8
1415	450090.026505	0.865367	-0.090233	301.140970	300.026874	300.544388	297.380536	3.760434	300.030007	-1.255092	...	222.0	194.2
1416	454332.044830	1.939130	-1.331747	303.700000	301.437143	300.622443	299.821276	3.878724	301.440556	0.494651	...	213.0	185.0
1417	454171.266248	0.576698	0.140102	302.487025	301.382706	302.196860	298.121028	4.365997	301.387215	-1.422661	...	228.0	175.2

2868 rows × 61 columns

Figura 18-2: Conjunto final de características de entrenamiento

Realizado por: Vilema Pablo, 2021

2.3.1.2. Extracción de características del conjunto de prueba

La extracción de características para el conjunto de prueba sigue el mismo procedimiento que para el conjunto de entrenamiento, por lo que no se lo describirá. Cabe acotar que las características extraídas para este conjunto siguen siendo 12 y son las mismas que para el conjunto de entrenamiento, a su vez el tamaño de ventana aplicado sigue siendo de 5 por lo que el número de observaciones se reduce a 955. A continuación, el gráfico 11-2 muestra el número final de observaciones para cada conjunto de características, tanto para el de entrenamiento como para el de prueba.

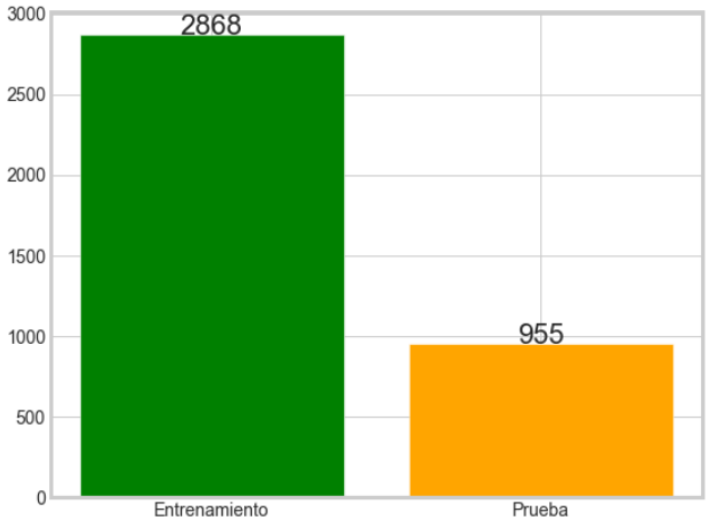


Gráfico 11-2: Datos finales para cada conjunto

Realizado por: Vilema Pablo, 2021

El número de datos que se aprecia en el gráfico 11-2, corresponden a los datos finales una vez aplicado el tamaño de ventana de 5 en el procedimiento de extracción de características. Es decir, se empleará 2868 datos para ingresar en el clasificador en la etapa de entrenamiento y 955 para la etapa de prueba. Posterior a la extracción de características se concateno los dos archivos (características de entrenamiento y prueba), ya que más adelante se lo empleará para determinar la significancia estadística del modelo.

2.3.2. Selección de características

Para la selección de características se aplicó el método de envoltura RFE (eliminación recursiva de características), para aplicar dicho método se necesita de un algoritmo de machine learning, por lo que se utilizó *Random Forest*, RFE permite definir qué cantidad de características más relevantes se seleccionarán por el algoritmo, para el presente caso de estudio se seleccionaron el 67% de las características.

2.3.2.1. Selección de características para el conjunto de entrenamiento

Una vez que se cuenta con el conjunto de características de entrenamiento, se procede aplicar el código para la selección de las mismas, RFE muestra los resultados de las características seleccionadas mediante un ranking y las características seleccionadas son aquellas que tienen un valor de 1.

	Feature	Ranking
0	0_Absolute energy	1
30	2_Min	1
31	2_Peak to peak distance	1
32	2_Root mean square	1
33	2_Skewness	1
34	2_Standard deviation	1
35	2_Variance	1
36	3_Absolute energy	1
37	3_Interquartile range	1
39	3_Max	1
40	3_Mean	1
41	3_Median	1
42	3_Min	1
43	3_Peak to peak distance	1

Figura 19-2: Ranking de características

Realizado por: Vilema Pablo, 2021

La figura 19-2 muestra algunas de las características que fueron seleccionadas por el método RFE, en ranking completo del conjunto de características puede apreciarse en la sección de anexos.

A continuación, la figura 20-2 muestra el *DataFrame* de las características que fueron seleccionadas por el método RFE, donde se evidencia que ahora existen un total de 41 características (siendo 40 las estadísticas y una hace referencia a la variable objetivo), es decir se eliminaron 20 ya que en total eran 60.

	0_Absolute energy	0_Max	0_Mean	0_Median	0_Min	0_Root mean square	1_Interquartile range	1_Max	1_Peak to peak distance	1_Root mean square	...	3_Skewness
0	447078.770000	301.600000	299.020000	298.000000	297.400000	299.024671	1.900000	311.100000	3.600000	309.402844	...	0.570505
1	443969.870000	300.600000	297.980000	297.500000	296.600000	297.983177	0.700000	309.700000	2.300000	308.301006	...	-0.544804
2	452585.850000	301.800000	300.860000	301.000000	299.800000	300.860715	1.500000	312.100000	2.800000	310.481677	...	-0.136341
3	450871.940000	303.600000	300.280000	299.800000	297.100000	300.290506	1.600000	312.000000	4.000000	309.843025	...	-0.762782
4	447966.000000	300.500000	299.320000	299.100000	298.200000	299.321232	1.300000	311.300000	2.600000	309.901400	...	0.268476
...	...	...	...	...	...	...	...	...	...	...	...	...
1413	452329.146274	302.627149	300.767075	302.121132	296.729446	300.775380	2.353021	311.561641	4.571196	309.898752	...	0.063412
1414	453458.968710	302.269229	301.149279	301.034075	299.550389	301.150782	0.632489	311.430742	2.609552	310.636296	...	0.446278
1415	450090.026505	301.140970	300.026874	300.544388	297.380536	300.030007	1.276731	311.551154	3.545323	310.005796	...	-0.225727
1416	454332.044830	303.700000	301.437143	300.622443	299.821276	301.440556	1.232353	312.100000	1.880611	311.043290	...	-0.769545
1417	454171.266248	302.487025	301.382706	302.196860	298.121028	301.387215	0.850138	311.232728	2.590672	310.445730	...	0.276014

2868 rows × 41 columns

Figura 20-2: Características seleccionadas

Realizado por: Vilema Pablo, 2021

Las características que fueron eliminadas se aprecian en la figura 21-2.

```
(['4_Standard deviation','0_Skewness','2_Kurtosis','1_Absolute energy',
  '1_Min','1_Mean','4_Variance','3_Kurtosis','0_Peak to peak distance',
  '0_Variance','4_Interquartile range','1_Skewness',
  '4_Min','1_Median','4_Peak to peak distance','0_Kurtosis',
  '4_Kurtosis','0_Standard deviation','0_Interquartile range',
  '1_Kurtosis'])
```

Figura 21-2: Características eliminadas

Realizado por: Vilema Pablo, 2021

2.3.2.2. Selección de características para el conjunto de prueba

Para el conjunto de prueba se seleccionaron las mismas características que fueron seleccionadas por el método de RFE para el conjunto de entrenamiento, es decir un total de 40 características.

2.3.3. Escalado/normalización de características

La mayoría de los algoritmos de *machine learning* requieren que todas las características estén en la misma escala, con la finalidad de que ninguna característica este dominada por otra. Para ello

se hace uso de la función RobustScaler la cual ayuda a minimizar el impacto de los valores atípicos. En primer lugar, se crea un objeto escalador y se ajusta sus parámetros (media y desviación estándar) usando el conjunto de características de entrenamiento, posteriormente se transforma los conjuntos de características de entrenamiento y prueba, tal y como muestra la figura 22-2.

```
scaler = RobustScaler()
scaler.fit(X_trains)
features_train_norm = scaler.transform(X_trains)
features_test_norm = scaler.transform(X_tests)
```

Figura 22-2: Escalado y normalización de los conjuntos de características

Realizado por: Vilema Pablo, 2021

2.4. Entrenamiento del modelo

Una vez escalados y transformados los conjuntos de características, estos se encuentran listos para ser ingresados en el clasificador.

2.4.1. Optimización de hiperparámetros

Antes de entrenar el modelo es muy importante ajustar los hiperparámetros del clasificador *Random Forest*, con la finalidad de optimizar el rendimiento del modelo, esto se lo realiza automáticamente mediante las funciones de la librería scikit-learn. Para ello se empleará el ajuste de búsqueda aleatoria mediante validación cruzada.

2.4.1.1. Creación de un bosque aleatorio para examinar los hiperparámetros

La figura 23-2 muestra los hiperparámetros disponibles para el bosque aleatorio, esto se realiza con la finalidad de apreciar los valores predeterminados.

Para el presente estudio se optimizó los hiperparámetros que se consideran más importantes según Sun et al. (2020), los cuales son:

- n_estimators
- criterion
- max_features
- max_depth
- min_samples_split

- min_samples_leaf
- bootstrap

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

Figura 23-2: Hiperparámetros predeterminados

Realizado por: Vilema Pablo, 2021

2.4.1.2. Cuadrícula de hiperparámetros aleatorios

Para determinar que hiperparámetros son los mejores se utiliza la función RandomizedSearchCV, por lo que en primer lugar se debe crear una cuadrícula de hiperparámetros para muestrear durante la optimización.

```
# Creación de cuadrícula aleatoria
random_grid = {'n_estimators': n_estimators,
               'criterion': criterion,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}

pprint(random_grid)
```

```
{'bootstrap': [True, False],
 'criterion': ['gini', 'entropy'],
 'max_depth': [10, 20, 30, 40, 50, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 3, 6],
 'n_estimators': [150, 250, 350, 450, 550]}
```

Figura 24-2: Cuadrícula de hiperparámetros

Realizado por: Vilema Pablo, 2021

Los valores de hiperparámetros establecidos que se observan en la figura 24-2, fueron

determinados aleatoriamente, y son los que mediante la validación cruzada se determinan cuáles son los que mejor rendimiento proporcionan.

2.4.1.3. Entrenamiento de búsqueda aleatoria

```
# Use la cuadrícula aleatoria para buscar los mejores hiperparámetros
rf = RandomForestClassifier()
# Búsqueda aleatoria de hiperparámetros, usando 3 veces la validación cruzada,
rf_random = RandomizedSearchCV(estimator=rf, param_distributions=random_grid,
                               n_iter = 100, scoring='f1_weighted',
                               cv = 3, verbose=2, random_state=42, n_jobs=-1,
                               return_train_score=True)

# Ajustar el modelo de búsqueda aleatoria
rf_random.fit(features_train_norm, y_trains);
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
```

Figura 25-2: Entrenamiento de búsqueda aleatoria

Realizado por: Vilema Pablo, 2021

Con la finalidad de determinar los mejores hiperparámetros, se hace uso de la cuadrícula aleatoria y se ingresa el conjunto de características de entrenamiento como muestra la figura 25-2.

Los parámetros más importantes de RandomizedSearchCV, son n_iter y cv mismos que controlan el número de combinaciones para probar y el número de pliegues para validación cruzada respectivamente. Cabe acotar que más iteraciones cubren un espacio de búsqueda más amplio y más pliegues de validación cruzada minimizan las posibilidades de sobreajuste, pero al aumentar cada uno de estos valores significa mayor tiempo de ejecución.

2.4.1.4. Mejores hiperparámetros

Una vez ejecutado el entrenamiento de búsqueda aleatoria, la figura 26-2 muestra los mejores hiperparámetros para el modelo.

```
{'n_estimators': 250,
 'min_samples_split': 3,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 20,
 'criterion': 'entropy',
 'bootstrap': False}
```

Figura 26-2: Mejores hiperparámetros

Realizado por: Vilema Pablo, 2021

2.4.1.5. Mejor modelo entrenado

Como último paso dentro del entrenamiento se procede a guardar el mejor modelo entrenado de *Random Forest* en una nueva variable, como muestra la figura 27-2.

```
rf_best= rf_random.best_estimator_
rf_best
```

```
RandomForestClassifier(bootstrap=False, criterion='entropy', max_depth=20,
                       min_samples_split=3, n_estimators=250)
```

Figura 27-2: Mejor modelo entrenado de Random Forest

Realizado por: Vilema Pablo, 2021

2.5. Evaluación del modelo

La evaluación del modelo se la llevará a cabo con los mejores hiperparámetros del clasificador *Random Forest*, haciendo uso del conjunto de características seleccionadas (40 características) y también se evaluará utilizando el conjunto de características originales (60 características), es decir el conjunto de características sin haber empleado el método de selección, con la finalidad de averiguar con qué conjunto de características se logran mejores resultados.

2.5.1. Predicción sobre el conjunto de características seleccionadas

Una vez entrenado el modelo se procede a realizar las predicciones sobre el conjunto de características seleccionadas, cabe mencionar que el conjunto seleccionado de características se lo denotó por features_test_norm y el conjunto original de características por features_test_normo.

```
# Predicción en el conjunto de prueba
y_predict = rf_best.predict(features_test_norm)
```

Figura 28-2: Ingreso del conjunto de características seleccionadas

Realizado por: Vilema Pablo, 2021

2.5.2. Predicción sobre el conjunto original de características

```
# Prediccion en el conjunto de prueba
y_predict1 = classifier.predict(features_test_normo)
```

Figura 29-2: Ingreso del conjunto original de características

Realizado por: Vilema Pablo, 2021

2.5.3. Métricas para la evaluación del modelo

Las métricas con las que se evaluarán el modelo son: matriz de confusión, exactitud, precisión, puntaje f1, sensibilidad, especificidad, curva de ROC y curva de aprendizaje; pero en el presente capítulo solamente se mostrará brevemente la matriz de confusión; el resto de métricas, así como el resultado e interpretación de la matriz de confusión serán analizadas en el capítulo III del presente trabajo de integración curricular.

2.5.3.1. Matriz de confusión aplicando el conjunto de características seleccionadas

A continuación, el gráfico 12-2 muestra la matriz de confusión que se obtuvo al momento de ingresar el conjunto de características seleccionadas en el clasificador.

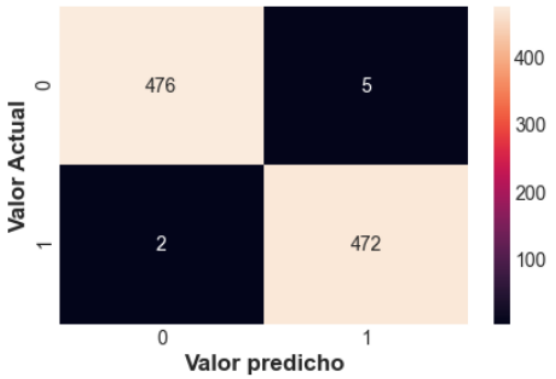


Gráfico 12-2: Matriz de confusion con características seleccionadas

Realizado por: Vilema Pablo, 2021

2.5.3.2. Matriz de confusión aplicando el conjunto original de características

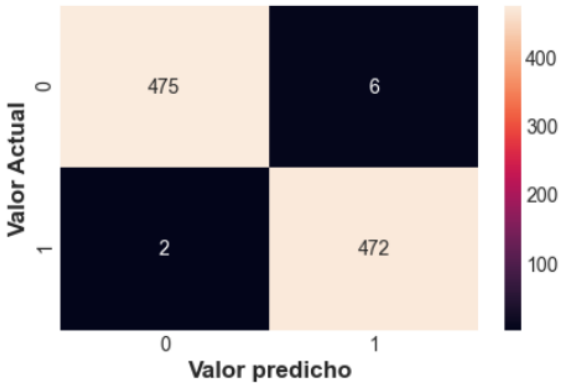


Gráfico 13-2: Matriz de confusión con características originales

Realizado por: Vilema Pablo, 2021

El gráfico 13-2 muestra la matriz de confusión que se obtuvo al aplicar el conjunto completo de características de prueba.

A simple vista se puede apreciar que el conjunto de características seleccionadas mostró un rendimiento levemente superior, ya que se logró disminuir en la predicción una instancia en los falsos positivos.

2.6. Análisis post hoc

Para finalizar con la creación del modelo de detección de fallos en mantenimiento predictivo, se da a conocer el ranking de características que aportaron en mayor y en menos proporción a la predicción de fallos de la máquina, a su vez de determina la significancia estadística del modelo.

2.6.1. Ranking del conjunto de características seleccionadas

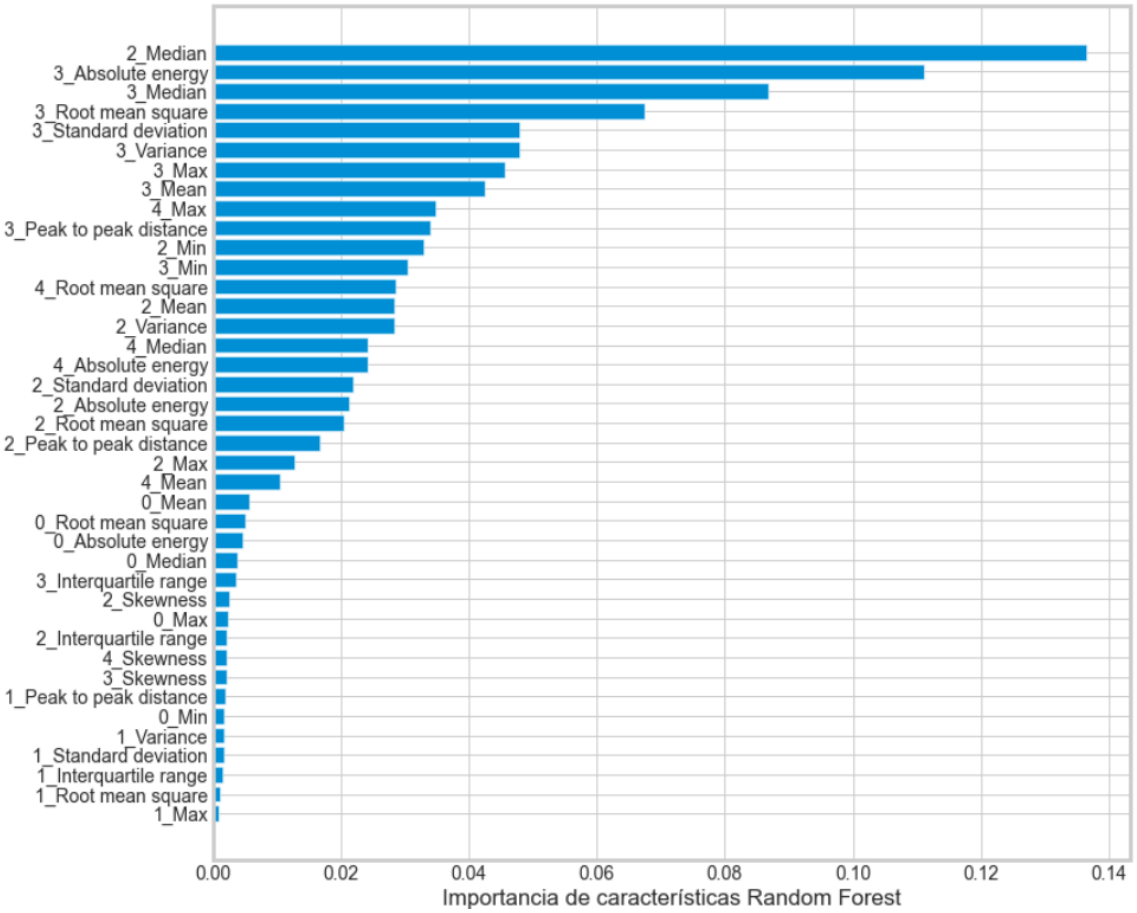


Gráfico 14-2: Ranking de características seleccionadas

Realizado por: Vilema Pablo, 2021

El gráfico 14-2 muestra el ranking de todas las características seleccionadas que se emplearon en

el modelo, donde se aprecia que las siguientes características fueron las que aportaron en mayor proporción a la predicción: 2_Median, 3_Absolute energy, 3_Median, 3_Root mean square y por otra parte las que menos aportaron son: 1_Max, 1_Root mean square, 1_Interquantile range, 1_Standar deviation, 1_Variance, 0_Min, entre otros.

2.6.2. Ranking del conjunto original de características

El gráfico 15-2 muestra el ranking del conjunto original de características que se emplearon en el modelo.

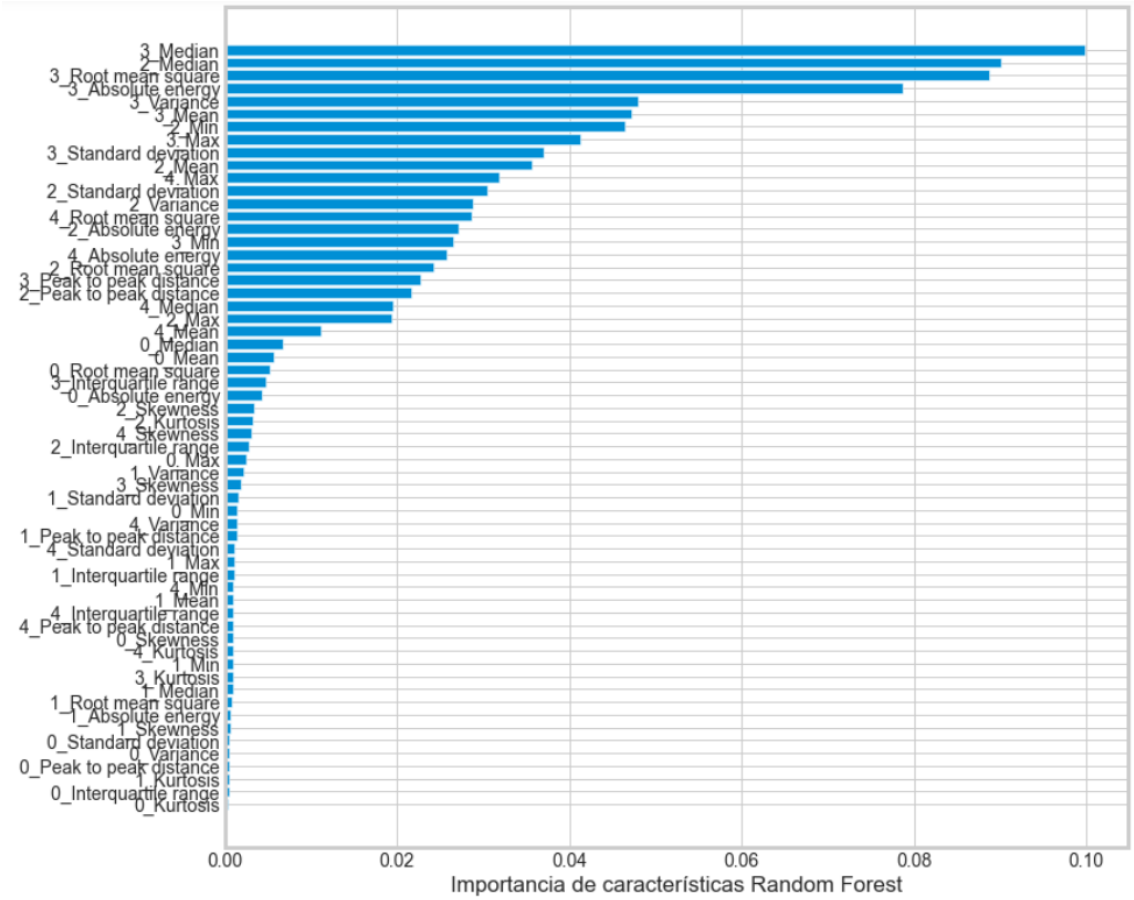


Gráfico 15-2: Ranking del conjunto original de características

Realizado por: Vilema Pablo, 2021

2.6.3. Significancia estadística

La significancia estadística se la llevó a cabo realizando 700 permutaciones, a su vez se utilizó 4 pliegues de validación cruzada y se empleó los hiperparámetros predeterminados de Random Forest, con la finalidad de reducir la carga y el tiempo computacional que tomaría realizar la prueba si se hiciera uso de los hiperparámetros optimizados. Si se desea revisar de manera más

detallada como se llevó a cabo este procedimiento se recomienda revisar la sección de anexos. Una vez finalizada la prueba de permutación se obtuvo un p-value igual a 0,001; la explicación detalla de la prueba de permutación para determinar la significancia estadística del modelo se la dará a conocer en el apartado de contraste de hipótesis del capítulo III del presente trabajo de integración curricular.

CAPÍTULO III

3. MARCO DE RESULTADOS Y DISCUSIÓN DE RESULTADOS

El presente capítulo se enfoca en analizar e interpretar los resultados obtenidos tras haber culminado con la realización del modelo de detección de fallos para mantenimiento predictivo. Este análisis está enfocado en determinar bajo que parámetros el modelo muestra el mejor rendimiento, por lo que se realizará comparaciones de las métricas de evaluación con características originales y seleccionadas tanto para hiperparámetros optimizados como para los predeterminados; una vez que se identifica los parámetros que proporcionan el mejor rendimiento del modelo se procede a elaborar la curva ROC y la curva de aprendizaje bajo dichos parámetros. Finalmente, como un paso extra se aplican otros métodos de aprendizaje de máquina con la finalidad de indagar si existe alguno que proporcione un mejor rendimiento que *Random Forest*.

3.1. Análisis de métricas con características seleccionadas e hiperparámetros optimizados

3.1.1. Matriz de confusión

Con la finalidad de lograr una buena interpretación de la matriz de confusión, se replicarán todos los valores de las celdas en una nueva matriz, como muestra la tabla 1-3.

Tabla 1-3: Matriz de confusión con características seleccionadas

		Predicho		
		0	1	
Actual	0	TN= 476	FP= 5	Total= 481
	1	FN= 2	TP= 472	Total= 474
		Total= 478	Total= 477	

Realizado por: Vilema Pablo, 2022

A continuación, se da a conocer la interpretación de la matriz de confusión:

- TN: conocido como verdadero negativo, cuenta para este caso en específico con 476 instancias correspondientes a la variable objetivo falla de la máquina en un estado de 0

(máquina no ha fallado), en otras palabras, el clasificador predijo 476 instancias como si la máquina no ha fallado y en efecto no falló, es decir, la predicción es completamente verdadera.

- TP: verdadero positivo, cuenta con 472 instancias correspondientes a la variable objetivo falla de la máquina en un estado de 1 (máquina ha fallado), en otras palabras, el clasificador predijo 472 instancias como si la máquina ha fallado y en efecto si falló, es decir se trata de una predicción completamente verdadera.
- FN: falso negativo, cuenta con 2 instancias correspondiente a la variable objetivo falla de la máquina en un estado de 0, pero para este caso el clasificador predijo mal, ya que esas 2 instancias en realidad pertenecen a cuando la máquina si falló, es decir un estado de 1; por lo que se puede decir que la predicción es falsa.
- FP: falso positivo, cuenta con 5 instancias correspondientes a la variable objetivo falla de la máquina en un estado de 1, pero la predicción realizada por el clasificador es errónea, ya que esas 5 instancias en realidad pertenecen a cuando la máquina no falló, es decir un estado de 0, por lo que se puede concluir que la predicción es falsa.

Finalmente se puede concluir que para el estado de máquina en 0 el clasificador predijo en total 478 instancias, de las cuales 476 son correctas y 2 son erróneas. Por otra parte, para el estado de máquina en 1, el clasificador predijo en total 477 instancias, de las cuales 472 son correctas y 5 son incorrectas. Lo ideal sería que los valores FN Y FP sean cero, pero es muy difícil crear un modelo que brinde dicho rendimiento, sin embargo, a simple vista se aprecian buenos resultados.

Relacionando de manera real a la matriz de confusión con una aplicación de mantenimiento predictivo, lo que más interesa es tratar de que en lo posible el clasificador sea capaz de predecir la mayor cantidad de instancias verdaderas positivas (TP) disminuyendo así las instancias de FN, ya que de esta forma se evitaría paros inesperados en las máquinas y en consecuencia en los procesos productivos lo cual generaría cuantiosas pérdidas económicas para una determinada empresa. Por otra parte analizando el valor de las instancias de la celda de falsos positivos (FP), dicho valor significaría realizar un sobre mantenimiento, ya que el clasificador con ese valor interpreta como si la máquina haya fallado, cuando en realidad no falló; por lo que en la toma de decisiones se realizaría un mantenimiento a una máquina cuando no es necesario, es decir algún elemento en específico puede aun contar con un tiempo de vida útil considerable y aun así ser reemplazado, lo que provoca una mala optimización de los recursos. Este análisis está enfocado netamente a comparar los costos, es decir analizar si los costos por un sobre mantenimiento superan a los costos por una parada inesperada o viceversa, tomando en cuenta todos los factores posibles que pueden generar pérdidas económicas (no producción, horas hombre, daños en otros componentes de la máquina, entre otros). Una vez realizado el análisis se puede tomar la decisión

de cual celda de la matriz de confusión es la que se desea que alcance la mayor cantidad de predicciones correctas.

3.1.2. Exactitud, precisión, sensibilidad, especificidad, puntaje f1

La figura 1-3 muestra el cálculo de las métricas de evaluación, para lo cual se aplicó las fórmulas descritas en el capítulo I del presente trabajo de integración curricular. A continuación, se da a conocer la interpretación de estas métricas:

- La exactitud del modelo es del 99,26%, es decir dicho valor representa la proporción de predicciones que el modelo clasificó de forma correcta, tanto para la clase 0 como para la clase 1.
- La precisión del modelo es del 98,95%, es decir dicho valor representa la proporción de predicciones positivas (falla de máquina = 1), que fueron clasificadas correctamente.
- La sensibilidad del modelo es del 99,57%, este valor representa la proporción de predicciones positivas reales, que fueron clasificadas correctamente.
- La especificidad del modelo es del 98,96%, dicho valor representa la proporción de predicciones negativas reales, que fueron clasificadas correctamente, este valor es opuesto a la sensibilidad.
- El puntaje F1 es del 99,26%, y representa en general una medida de la precisión y robustez del modelo.

```
Exactitud= ((TP+TN)/(TP+FP+FN+TN))*100
print(Exactitud)
```

```
99.26701570680628
```

```
Precisión= (TP/(TP+FP))*100
print(Precisión)
```

```
98.9517819706499
```

```
Sensibilidad= (TP/(TP+FN))*100
print(Sensibilidad)
```

```
99.57805907172997
```

```
Especificidad= (TN/(TN+FP))*100
print(Especificidad)
```

```
98.96049896049897
```

```
PuntajeF1= (2*TP/(2*TP+FP+FN))*100
print(PuntajeF1)
```

```
99.26393270241851
```

Figura 1-3: Cálculo de métricas con características seleccionadas

Realizado por: Vilema Pablo, 2022

3.2. Análisis de métricas con el conjunto original de características e hiperparámetros optimizados

3.2.1. Matriz de confusión

De la misma forma que para el anterior análisis, se replicarán los valores de la matriz en la tabla 2-3.

Tabla 2-3: Matriz de confusión con características originales

		Predicho		
		0	1	
Actual	0	TN= 475	FP= 6	Total= 481
	1	FN= 2	TP= 472	Total= 474
		Total= 477	Total= 478	

Realizado por: Vilema Pablo, 2022

Donde el análisis de la matriz es el siguiente:

- TN: cuenta para este caso en específico con 475 instancias correspondientes a la variable objetivo falla de la máquina en un estado de 0, en otras palabras, el clasificador predijo 475 instancias como si la máquina no ha fallado y en efecto no falló, es decir, la predicción es completamente verdadera.
- TP: verdadero positivo, cuenta con 472 instancias correspondientes a la variable objetivo falla de la máquina en un estado de 1, en otras palabras, el clasificador predijo 570 instancias como si la máquina ha fallado y en efecto si falló, es decir se trata de una predicción completamente verdadera.
- FN: falso negativo, cuenta con 2 instancias correspondientes a la variable objetivo falla de la máquina en un estado de 0, pero para este caso el clasificador predijo mal, ya que las 2 instancias en realidad pertenecen a cuando la máquina si falló, es decir un estado de 1; por lo que se puede decir que la predicción es falsa.
- FP: falso positivo, cuenta con 6 instancias correspondientes a la variable objetivo falla de la máquina en un estado de 1, pero la predicción realizada por el clasificador es errónea, ya que dichas 6 instancias en realidad pertenecen a cuando la máquina no falló, es decir un estado de 0, por lo que se puede concluir que la predicción es falsa.

Finalmente se puede concluir que, para el estado de máquina en 0, el clasificador predijo en total 477 instancias, de los cuales 475 son correctas y 2 son erróneas. Por otra parte, para el estado de

máquina en 1, el clasificador predijo en total 478 instancias, de las cuales 472 son correctos y 6 son incorrectas.

3.2.2. Exactitud, precisión, sensibilidad, especificidad, puntaje f1

El cálculo de las métricas se puede apreciar en la figura 2-3 y la interpretación es la siguiente:

- La exactitud del modelo es del 99,16%, es decir dicho valor representa la proporción de predicciones que el modelo clasificó de forma correcta, tanto para la clase 0 como para la clase 1.
- La precisión del modelo es del 98,74%, es decir dicho valor representa la proporción de predicciones positivas (falla de máquina = 1), que fueron clasificadas correctamente.
- La sensibilidad del modelo es del 99,57%, este valor representa la proporción de predicciones positivas reales, que fueron clasificadas correctamente.
- La especificidad del modelo es del 98,75%, donde dicho valor representa la proporción de predicciones negativas reales, que fueron clasificadas correctamente, este valor es opuesto a la sensibilidad.
- El puntaje F1 es del 99.15%, y representa en general una medida de la precisión y robustez del modelo.

```
Exactitud= ((TP1+TN1)/(TP1+FP1+FN1+TN1))*100
print(Exactitud)
```

```
99.16230366492147
```

```
Precisión= (TP1/(TP1+FP1))*100
print(Precisión)
```

```
98.74476987447699
```

```
Sensibilidad= (TP1/(TP1+FN1))*100
print(Sensibilidad)
```

```
99.57805907172997
```

```
Especificidad= (TN1/(TN1+FP1))*100
print(Especificidad)
```

```
98.75259875259876
```

```
PuntajeF1= (2*TP1/(2*TP1+FP1+FN1))*100
print(PuntajeF1)
```

```
99.15966386554622
```

Figura 2-3: Cálculo de métricas con características originales

Realizado por: Vilema Pablo, 2022

3.3. Resumen de métricas de evaluación con hiperparámetros optimizados y conjunto de características seleccionadas y originales

A continuación, se realizará dos tablas para resumir la interpretación de las métricas de evaluación del modelo, una tabla está enfocada a determinar cuál es el grado de mejora dentro de las matrices de confusión y la siguiente tabla está destinada a observar el porcentaje de mejora del resto de métricas de evaluación.

3.3.1. Resumen de las matrices de confusión

Tabla 3-3: Resumen de los valores de las dos matrices de confusión

	Valor por cada celda			
	TN	TP	FN	FP
Características seleccionadas	476	472	2	5
Características originales	475	472	2	6
Diferencia	1	0	0	-1

Realizado por: Vilema Pablo, 2022

A continuación, la tabla 3-3 muestra un resumen de los valores de cada celda correspondiente a cada matriz de confusión y se concluye que:

- Para el valor de TN, el trabajar con características seleccionadas proporciona un incremento de predicción de una instancia, es decir, el clasificador elevó las instancias predichas de forma correcta, por lo que es muy beneficioso debido a que se predijo de manera correcta una instancia más que si se trabajara con el conjunto de características originales.
- Para el valor de TP se aprecia que el trabajar con características seleccionadas no mejora las instancias predichas por el clasificador, es decir las instancias predichas son las mismas que si se trabajara con características originales.
- Para el valor de FN se aprecia que el trabajar con características seleccionadas tampoco mejora las instancias predichas por el clasificador, es decir las instancias predichas son las mismas que si se trabajara con características originales.
- Para el valor de FP, el trabajar con características seleccionadas proporciona un decremento de predicción de una instancia en comparación con características originales, en otras palabras, el clasificador redujo las instancias predichas de forma incorrecta, lo cual es positivo ya que el valor FP mientras más se acerque a 0, al igual que FN, el modelo mostrará un mejor rendimiento.

3.3.2. Resumen de métricas de evaluación restantes

Para finalizar con el resumen de las métricas de evaluación, a continuación, en la tabla 4-3 se compara los porcentajes de las métricas, con la finalidad de determinar si existe un porcentaje de mejora al trabajar con el conjunto seleccionado de características.

Tabla 4-3: Resumen de métricas de evaluación restantes

	Conjunto de características		
Métrica	Características seleccionadas	Características originales	Porcentaje de mejora
Exactitud	99,26%	99,16%	0,10%
Precisión	98,95%	98,74%	0,21%
Sensibilidad	99,57%	99,57%	0%
Especificidad	98.96%	98,75%	0,21%
Puntaje F1	99,26%	99,15%	0,11%

Realizado por: Vilema Pablo, 2022

De la tabla 4-3 se concluye que:

- Al trabajar con características seleccionadas, el modelo obtiene un rendimiento levemente superior que, al trabajar con el conjunto de características originales, esto se lo puede evidenciar en el porcentaje de mejora en cada una de las métricas evaluadas, sin embargo, en la métrica de sensibilidad no existe un porcentaje de mejora.
- Si bien es cierto que el porcentaje de mejora para todas las métricas no es tan elevado, para la mayoría de los modelos esos pequeños porcentajes de mejora significan una gran contribución para elevar las predicciones correctas y disminuir las predicciones incorrectas.

Una vez realizado este análisis se concluye que en efecto la selección de características contribuye a mejorar el rendimiento del modelo y a reducir el costo computacional, lo cual se vería significativamente afectado en bases de datos de gran dimensión.

3.4. Comparación del modelo con hiperparámetros predeterminados y optimizados

Para poder realizar la comparación y determinar la influencia de los hiperparámetros, en primer lugar, se mostrará las dos matrices de confusión, resultado de trabajar con hiperparámetros predeterminados en conjunto con las características seleccionadas y las características originales, ya que las dos matrices con los hiperparámetros optimizados ya fueron analizadas anteriormente.

3.4.1. Matriz de confusión con hiperparámetros predeterminados y características seleccionadas

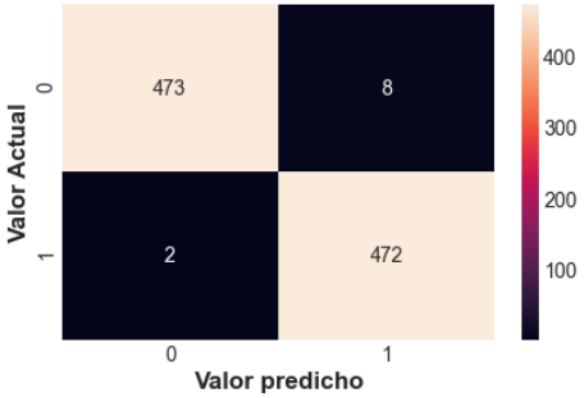


Gráfico 1-3: Matriz de confusión con hiperparámetros predeterminados y características seleccionadas

Realizado por: Vilema Pablo, 2022

Del gráfico 1-3 se concluye que:

- En comparación con la matriz del gráfico 12-2, la matriz del gráfico 1-3 muestra un rendimiento inferior en los valores de las celdas TN Y FP; ya que en el gráfico 12-2 el clasificador logra predecir 476 y 5 instancias para TN y FP respectivamente, versus 473 y 8 instancias del gráfico 1-3, el valor de TP y FN se mantienen en ambos casos.

3.4.2. Matriz de confusión con hiperparámetros predeterminados y características originales

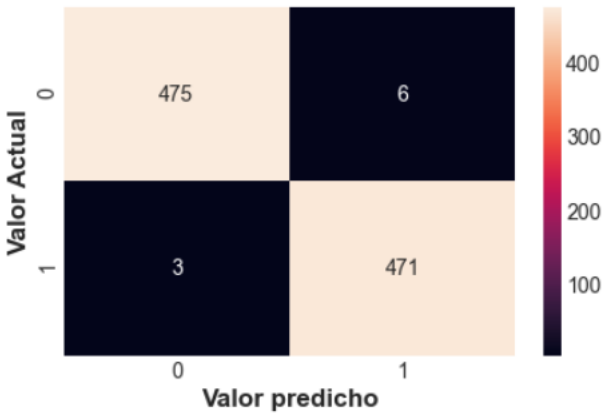


Gráfico 2-3: Matriz de confusión con hiperparámetros predeterminados y características originales

Realizado por: Vilema Pablo, 2022

Del gráfico 2-3 se concluye que:

- En comparación con la matriz del gráfico 13-2, la matriz del gráfico 2-3 muestra un rendimiento inferior en los valores de las celdas TP Y FN; ya que en el gráfico 13-2 el clasificador logra predecir 472 y 2 instancias para TP y FN respectivamente, versus 471 y 3 instancias del gráfico 2-3, el valor de TN y FP se mantienen en ambos casos.

Para los dos análisis realizados se puede concluir que las matrices de confusión que brindan mejores resultados son las que fueron producto de emplear optimización de hiperparámetros, lo cual se puede evidenciar más adelante en la comparación de métricas de evaluación.

3.4.3. Análisis de la influencia de hiperparámetros sobre el modelo

Tabla 5-3: Métricas de evaluación en base a hiperparámetros

		Conjunto de características	
		Características seleccionadas	Características originales
Hiperparámetros predeterminados	Exactitud	98,95%	99,05%
	Precisión	98,33%	98,74%
	Sensibilidad	99,57%	99,36%
	Especificidad	98,33%	98,75%
	Puntaje F1	98,95%	99,05%
Hiperparámetros optimizados	Exactitud	99,26%	99,16%
	Precisión	98,95%	98,74%
	Sensibilidad	99,57%	99,57%
	Especificidad	98,96%	98,75%
	Puntaje F1	99,26%	99,15%

Realizado por: Vilema Pablo, 2022

La tabla 5-3 muestra la comparación del modelo, con hiperparámetros predeterminados e hiperparámetros optimizados, tanto para el conjunto de características seleccionadas y para el conjunto original de características. El presente análisis está enfocado a determinar con qué conjunto de características el modelo tiene un mejor rendimiento, a su vez también se analiza si el ajuste de hiperparámetros permite mejorar el rendimiento.

Analizando los porcentajes de las métricas obtenidas aplicando características seleccionadas tanto para hiperparámetros optimizados como para predeterminados, se puede apreciar que la optimización de hiperparámetros ayuda a mejorar el rendimiento de todas las métricas excepto de la sensibilidad que se mantiene en ambos casos; en primera instancia se descarta trabajar el modelo con características seleccionadas e hiperparámetros predeterminados. A continuación, se

contrasta las características originales con los dos casos de hiperparámetros, donde se aprecia que con hiperparámetros optimizados se obtiene un mejor rendimiento en 3 métricas de evaluación y se mantiene el rendimiento en la precisión y especificad, descartando de esta forma las características originales con hiperparámetros predeterminados. Finalmente para determinar bajo que parámetros el modelo muestra el mejor rendimiento, se compara las métricas entre las características seleccionadas y originales con hiperparámetros optimizados, ya que estos dos no fueron descartados anteriormente; dando como resultado que el mejor rendimiento del modelo se alcanza utilizando características seleccionadas e hiperparámetros optimizados, ya que se aprecia una mejoría en casi todas las métricas de evaluación, excepto en la sensibilidad que se mantiene igual.

Una vez determinado bajo que parámetros el modelo muestra el mejor rendimiento, se procede a realizar la curva de ROC y la curva de aprendizaje con características seleccionadas e hiperparámetros optimizados.

3.5. Curva ROC-AUC

La curva de ROC es otra de las métricas que se emplean para evaluar un modelo de clasificación binaria, donde el rendimiento del clasificador se puede resumir en un número calculando el área bajo la curva ROC. A continuación, el gráfico 3-3 muestra la curva de ROC utilizando la optimización de hiperparámetros y el conjunto seleccionado de características.

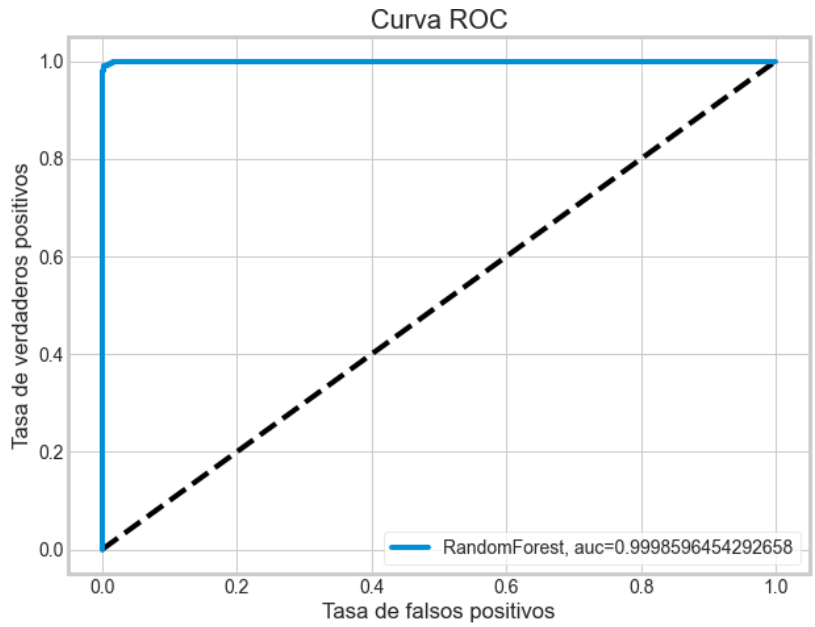


Gráfico 3-3: Curva de ROC

Realizado por: Vilema Pablo, 2022

Se puede apreciar en el gráfico 3-3 que el valor de AUC-ROC está muy cercano a 1, por lo que se concluye que el clasificador distingue de manera óptima que valores pertenecen a la clase positiva (Falla_maquina = 1) y a clase negativa (Falla_maquina = 0).

3.6. Curva de aprendizaje

La curva de aprendizaje ayuda a determinar si el agregar más datos en el entrenamiento ayuda a mejorar el rendimiento del modelo, por otra parte, permite evaluar si el modelo está sobreajustado o desajustado.

Del gráfico 4-3 se concluye que el modelo aprendió de forma correcta en el entrenamiento, por otra parte analizando la curva de validación cruzada, se aprecia que el modelo muestra un rendimiento deficiente hasta un valor cercano igual a los 1250 puntos de datos, a partir de dicho punto se observa que la exactitud se incrementa al aumentar el número de datos; las dos curvas convergen entre un rango de 2000-2250 puntos de datos, donde claramente se evidencia que permanecen estables, por lo que se concluye que agregar más datos no mejorará el rendimiento del modelo.

Además, analizando la gráfica se puede concluir que el modelo no está desajustado ya que presente una buena puntuación en el entrenamiento, y tampoco se encuentra sobreajustado ya que la puntuación de validación cruzada se encuentra muy cerna a 1, lo que indica una muy buena generalización en datos nuevos o no vistos.

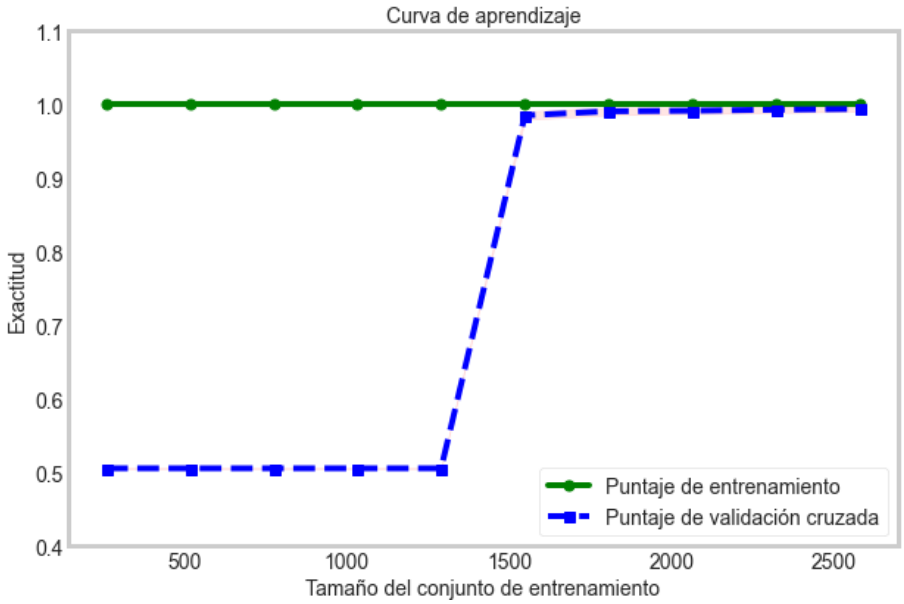


Gráfico 4-3: Curva de aprendizaje

Realizado por: Vilema Pablo, 2022

3.7. Comparación del modelo con otros métodos de machine learning

La tabla 6-3 muestra los diferentes métodos empleados para comparar el modelo, a su vez muestra 3 métricas de evaluación con sus diferentes porcentajes.

Tabla 6-3: Comparación de métodos de machine learning

Método	Exactitud	Precisión	Sensibilidad
Random Forest	98,95%	98,33%	99,57%
SVM	99,37%	98,95%	99,78%
Gradient Boosting	99,05%	98,74%	99,36%
XGBoost	99,26%	98,95%	99,57%
KNN	99,05%	98,53%	99,57%
Naive Bayes	98,21%	96,91%	99,57%
Árbol de decisión	98,01%	99,13%	96,83%

Realizado por: Vilema Pablo, 2022

Con la finalidad de determinar si existe algún método de aprendizaje de máquina que proporcione mejor rendimiento en el modelo, se probó 6 métodos los cuales son: máquina de vectores de soporte (SVM), *Gradient Boosting*, *XGBoost*, k vecinos más cercanos (KNN), *Naive Bayes* y árbol de decisión. Cabe acotar que la comparación de los métodos se la realizará con hiperparámetros predeterminados, haciendo uso del conjunto seleccionado de características y solamente se tomará en cuenta las métricas de evaluación exactitud, precisión y sensibilidad.

Realizando un análisis muy minucioso sobre la aplicación del aprendizaje de máquina para la detección de fallos en mantenimiento predictivo, lo que más importancia tiene es que se logren predecir de forma correcta la mayor cantidad de fallos posibles de una máquina y al mismo tiempo disminuir el valor de FP Y FN. La métrica de evaluación que determina la cantidad de predicciones positivas reales (falla de la máquina = 1) que fueron clasificadas correctamente es la sensibilidad, por lo que para determinar que método de aprendizaje de máquina muestra un mejor rendimiento para el mantenimiento predictivo se utilizará esta métrica de evaluación, a su vez se debe tener mucho cuidado en los falsos positivos ya que estos representan un sobre mantenimiento; las matrices de confusión para todos los métodos pueden revisarse en la sección anexos del presente trabajo de integración curricular.

Finalmente se concluye que el método que muestra el mejor rendimiento bajo el análisis realizado anteriormente es SVM, ya que solo comete una instancia mal predicha de verdaderos positivos (FN), lo cual se refleja en el porcentaje de la métrica alcanzada (99,78%), siendo el más sobresaliente de todos los métodos comparados.

En comparación con el método de *Random Forest* bajo los parámetros que proporcionaron el mejor rendimiento, el método de SVM reduce una instancia predicha de forma incorrecta en el valor de FN. Cabe acotar que los hiperparámetros de SVM no se optimizaron, por lo que quizá el optimizarlos podría ayudar a reducir un poco más las instancias predichas de forma incorrecta. A continuación, el gráfico 5-3 muestra la matriz de confusión del método SVM.

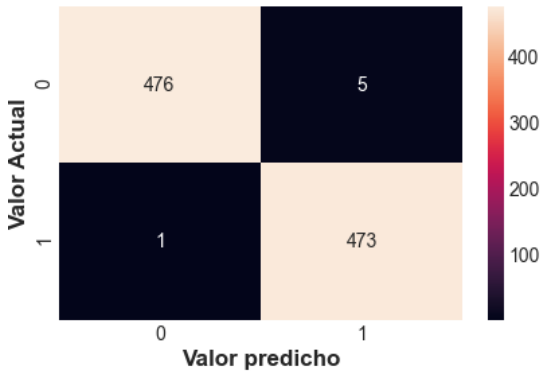


Gráfico 5-3: Matriz de confusión SVM

Realizado por: Vilema Pablo, 2022

3.8. Constatación de la hipótesis

Para constatar la hipótesis propuesta al inicio de la investigación, se utilizó la prueba de permutación para determinar si el modelo es estadísticamente significativo.

La investigación de Good (1994) menciona que para una prueba de permutación se establezca la hipótesis nula siguiente:
Ho= El rendimiento del modelo es obtenido por casualidad

Por lo que como hipótesis alternativa se establece:
Hi= El rendimiento del modelo no es obtenido por casualidad, es decir el modelo encontró una conexión real entre las características y las etiquetas de clase de la variable objetivo.

Para dar una conclusión sobre la prueba de permutación Ojala y Garriga (2010) mencionan que, si el valor p es lo suficientemente pequeño, generalmente bajo un cierto umbral (regularmente $\alpha = 0,05$), se puede decir que el error en los datos originales es significativamente pequeño bajo la hipótesis nula dada, es decir se rechaza la hipótesis nula dando como resultado un modelo de aprendizaje de máquina significativo.

Como hipótesis principal se determinó: utilizando el método de aprendizaje de máquina Random Forest se detectan fallos en mantenimiento predictivo, por lo que la métrica de evaluación a la

cual se le calculó el valor p para determinar la significancia estadística es la sensibilidad. Ya que dicha métrica está enfocada en predecir las instancias reales positivas que se clasificaron correctamente.

Una vez ejecutada la prueba de permutación se obtuvo un valor p igual a 0,001, lo cual permite rechazar la hipótesis nula y con ello aceptar la hipótesis alternativa, determinando así que el rendimiento del modelo no se obtuvo por casualidad, obteniendo un modelo estadísticamente significativo.

CONCLUSIONES

En el presente trabajo de integración curricular se detectó fallos de una máquina empleando datos de mantenimiento predictivo, para lo cual se realizó un modelo utilizando el método de aprendizaje de máquina *Random Forest*, mismo que mostró un buen rendimiento en la fase de entrenamiento y a su vez una buena generalización en los datos de prueba, lo cual se evidencia en las métricas de evaluación analizadas, a su vez se comprobó la significancia estadística la cual respalda que las predicciones obtenidas por el modelo se apegan a la realidad y no son debidas a la casualidad.

La realización del pre procesamiento de datos de mantenimiento predictivo del repositorio de la Universidad de California, Irvine (UCI); contribuyó en gran parte a obtener un buen rendimiento en el modelo, debido a que gracias al análisis exploratorio de datos se pudo comprender cuál es el comportamiento de los mismos, a su vez permitió detectar algún tipo de anomalía en la base datos para posteriormente ser corregida; por otra parte la limpieza de datos ayudó a reducir el número de variables, manteniendo las que verdaderamente contribuyen a la predicción en base al problema que se quiere resolver, finalmente gracias al sobremuestreo de datos se mejoró notoriamente el rendimiento del modelo, ya que en la fase de prueba y error al trabajar con la base de datos desequilibrada se obtuvo métricas de evaluación muy deficientes.

La división de los datos de mantenimiento predictivo para entrenamiento y prueba, fue un paso de vital importancia para realizar el modelo de forma correcta, debido a que el algoritmo se debe entrenar con una cierta cantidad de datos (esto en base al criterio del investigador, ya que no existe una regla definida que mencione cual es el porcentaje de datos que se debe designar para cada conjunto), y para realizar la prueba del modelo se emplean los datos restantes que no fueron tomados en cuenta para el entrenamiento; esto se realiza con la finalidad de evaluar cuál es el rendimiento del modelo y a su vez determinar la generalización de los patrones de los datos aprendidos durante el entrenamiento, en datos que no fueron presentados al algoritmo o datos no vistos.

Las características estadísticas extraídas en el dominio del tiempo permitieron que el algoritmo de *Random Forest* pueda aprender más patrones de los datos de mantenimiento predictivo, lo que conllevó a un mejor aprendizaje y a una mayor generalización en los datos de prueba, cabe recalcar que mientras más aumenta la cantidad de características extraídas, el costo computacional es más elevado mayormente en la etapa de entrenamiento, esto principalmente para bases de datos de gran dimensión. Un aspecto muy importante a tener en cuenta es que el mostrar más

características al algoritmo no siempre va a proporcionar un gran rendimiento, ya que se podría entorpecer el aprendizaje del algoritmo debido a la gran cantidad de información existente.

Una vez diseñado el modelo de aprendizaje de máquina, se pudo comprobar varias métricas de evaluación del algoritmo de clasificación *Random Forest*, entre una de las principales se tiene la precisión, misma que mostró un porcentaje elevado al momento de detectar fallos en mantenimiento predictivo, pudiendo de esta forma comprobar lo citado por (Mohammady et al., 2019), el cual menciona que el algoritmo de *Random Forest* presenta una precisión relativamente alta.

RECOMENDACIONES

En una investigación futura, realizar el modelo de aprendizaje de máquina utilizando el segundo enfoque de la investigación, el cual consiste en detectar cual es el modo de fallo que ocasionó la falla de la máquina, convirtiéndose el modelo en una clasificación multiclase.

Determinar la significancia estadística del método SVM, con la finalidad de comprobar si en realidad es mejor que el método de *Random Forest*.

Antes de realizar un modelo de aprendizaje de máquina es recomendable tener en cuenta las características computacionales del ordenador en el que se trabajará, ya que mientras más aumenta el número de datos, se requerirá una mayor memoria RAM y un procesador más eficiente, o a su vez se recomienda trabajar con el entorno de Google Colab.

Aplicar el aprendizaje de máquina a otros datos de mantenimiento predictivo, como por ejemplo la termografía, ya que los análisis termográficos tradicionales se los realiza de forma manual, por otra parte, mediante el aprendizaje de máquina también se puede realizar aplicaciones mediante el análisis de imágenes que para este caso serían termogramas, con la finalidad de extraer características de los mismos e identificar algún patrón anómalo lo cual conllevaría a la detección de una falla.

Se recomienda seguir realizando investigaciones sobre el aprendizaje de máquina enfocadas al mantenimiento, ya que hoy en día mediante el internet de las cosas, se están extrayendo de las máquinas y procesos una gran cantidad de datos que necesitan ser analizados para posterior a ello realizar la toma de decisiones, de esta forma se logra obtener modelos de detección de fallos automáticos más precisos y exactos en comparación con los análisis tradicionales; incluso con la finalidad de seguir mejorando en la creación de modelos de aprendizaje de máquina se recomienda adentrarse en la solución de problemas dentro de otras áreas del conocimiento, como por ejemplo la medicina, el marketing, la ciberseguridad, entre otros.

BIBLIOGRAFÍA

AIMACAÑA, Jefferson; & COLUMBA, Alexander. Análisis comparativo de algoritmos de Machine Learning para la detección de plagas en los cultivos representativos de la sierra ecuatoriana (Trabajo de titulación). [En línea] Universidad Central del Ecuador, Facultad de Ingeniería y Ciencias Aplicadas, Carrera de Ingeniería Informática. Quito-Ecuador. 2021. p.26 [Consulta: 16 de Octubre 2021]. Disponible en: http://www.dspace.uce.edu.ec/handle/25000/24228

BARANDAS, Marília; et al. "TSFEL: Time Series Feature Extraction Library". *SoftwareX* [en línea], 2020, 11. [Consulta: 9 de Noviembre 2021]. ISSN: 2352-7110. Disponible en: https://doi.org/10.1016/j.softx.2020.100456

BASHIR, Daniel; et al. "An Information-Theoretic Perspective on Overfitting and Underfitting". *AI 2020: Advances in Artificial Intelligence. AI 2020. Lecture Notes in Computer Science* [en línea], 2020, 12576, pp. 347-358. [Consulta: 10 de Noviembre 2021]. 16113349. Disponible en: https://doi.org/10.1007/978-3-030-64984-5_27

BRONWLEE, Jason. Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python [en línea]. Machine Learning Mastery, 2020. [Consulta: 13 de Noviembre 2021]. Disponible en: https://b-ok.lat/book/16370100/0383c0

CARVALHO, Thyago P.; et al. "A systematic literature review of machine learning methods applied to predictive maintenance". *Computers & Industrial Engineering* [en línea], 2019, 137. [Consulta: 26 de Agosto 2021]. Disponible en: https://doi.org/10.1016/j.cie.2019.106024

CHALLENGER, Ivet; et al. "El lenguaje de programación Python". *Ciencias Holguín* [en línea], 2014, pp. 1-13. [Consulta: 6 de Octubre 2021]. ISSN: 1027-2127. Disponible en: https://www.redalyc.org/articulo.oa?id=181531232001

CHAOJI, Vineet; et al. "Machine learning in the real world". *Proceedings of the VLDB Endowment* [en línea], 2016, 9, pp. 1597-1600. [Consulta: 10 de Noviembre 2021]. Disponible en: https://doi.org/10.14778/3007263.3007318

CHOUDHARY, Rishabh; & GIANEY, Hemant K. "Comprehensive Review On Supervised Machine Learning Algorithms". *2017 International Conference on Machine Learning and Data*

Science (MLDS) [en línea], 2018, pp. 37-43. [Consulta: 29 de Octubre 2021]. Disponible en: https://doi.org/10.1109/MLDS.2017.11

DEY, Ayon. "Machine Learning Algorithms: A Review". *International Journal of Computer Science and Information Technologies* [en línea], 2016, 7(3), pp. 1174-1179. [Consulta: 27 de Octubre 2021]. ISSN: 0975-9646. Disponible en: https://www.semanticscholar.org/paper/Machine-Learning-Algorithms-%3A-A-Review-Dey/56e8863838b4dcc4790108cd1e7e680a104a7c30

DHALL, Devanshi; et al. "Machine Learning: A Review of the Algorithms and Its Applications". *Proceedings of ICRIC 2019* [en línea], 2019, pp. 47-63. [Consulta: 27 de Octubre 2021]. Disponible en: https://doi.org/10.1007/978-3-030-29407-6_5

DUA, Dheeru; & GRAFF, Casey. *UCI Machine Learning Repository*. 2019. [Consulta: 9 de Noviembre 2021]. Disponible en: https://archive.ics.uci.edu/ml/datasets/AI4I+2020+Predictive+Maintenance+Dataset

FERNANDES, Marta; et al. "Fault Detection Mechanism of a Predictive Maintenance System Based on Autoregressive Integrated Moving Average Models". *Distributed Computing and Artificial Intelligence, 16th International Conference. DCAI 2019. Advances in Intelligent Systems and computing* [en línea], 2020, 1003. [Consulta: 25 de Agosto 2021]. Disponible en: https://doi.org/10.1007/978-3-030-23887-2_20

FERNÁNDEZ, Arturo. *Python 3 al descubierto* [en línea]. 2ª ed. México DF-México: Alfaomega Grupo Editor, 2013. [Consulta: 6 de Octubre 2021]. Disponible en: https://docer.com.ar/doc/n00x1xn

GOOD, Phillip. *Permutation tests: A practical guide to resampling methods for testing hypotheses* [en línea]. 1ª ed. New York, NY: Springer, 1994. [Consulta: 12 de Febrero 2021]. Disponible en: https://doi.org/10.1007/978-1-4757-2346-5

HAGHIGHI, Sepand; et al. "PyCM: Multiclass confusion matrix library in Python". *Journal of Open Source Software* [en línea], 2018, 3(25). [Consulta: 29 de Octubre 2021]. Disponible en: https://doi.org/10.21105/joss.00729

JAHNKE, Patrick. Machine Learning Approaches for Failure Type Detection and Predictive Maintenance (Tesis) (Maestría). [En línea] TECHNISCHE UNIVERSITAT DARMSTADT,

Department of Computer Science. Darmstadt-Germany, 2015, p.16. [Consulta: 27 de Octubre 2021]. Disponible en: http://www.ke.tu-darmstadt.de/lehre/arbeiten/master/2015/Jahnke_Patrick.pdf

LEE, Wo Jae; et al. "Predictive maintenance of machine tool systems using artificial intelligence techniques applied to machine condition data". *Procedia CIRP* [en línea], 2019, 80, pp. 506-511. [Consulta: 27 de Octubre 2021]. Disponible en: https://doi.org/10.1016/j.procir.2018.12.019

LI, Xiang; et al. "Design and Implementation of Equipment Maintenance Predictive Model Based on Machine Learning". *IOP Conference Series: Materials Science and Engineering* [en línea], 2018, 466. [Consulta: 12 de Noviembre 2021]. Disponible en: https://doi.org/10.1088/1757-899x/466/1/012001

MANRIQUE, Esperanza; et al. "Machine Learning: análisis de lenguajes de programación y herramientas para desarrollo". *Revista Ibérica de Sistemas e Tecnologías de Informação* [en línea], 2020, pp. 586-599. [Consulta: 16 de Octubre 2021]. Disponible en: https://www.proquest.com/openview/c7e24c997199215aa26a39107dd2fe98/1?pq-origsite=gscholar&cbl=1006393

MOHAMMADY, Majid; et al. "Land subsidence susceptibility assessment using random forest machine learning algorithm". *Environmental Earth Sciences* [en línea], 2019, 78, pp.1-12. [Consulta: 27 de Octubre 2021]. Disponible en: https://doi.org/10.1007/s12665-019-8518-3

MOHHAMED, Roweida; et al. "Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results". *2020 11th International Conference on Information and Communication Systems (ICICS)* [en línea], 2020, pp. 243-248, [Consulta: 27 de Diciembre 2021]. Disponible en: https://doi.org/10.1109/ICICS49469.2020.239556

MÜLLER, Andreas C, & GUIDO, Sarah. Introduction to Machine Learning with Python: A Guide for Data Scientists [en línea]. United States Of América-Gravenstein Highway North, Sebastopol: O'Reilly Media, 2017. [Consulta: 4 de Octubre 2021]. Disponible en: https://pdflife.one/download/4423583-introduction-to-machine-learning-with-python

OJALA, Markus; & GARRIGA, Gemma C. "Permutation tests for studying classifier performance". *Journal of Machine Learning Research* [en línea], 2010, 11, pp. 1833-1886, [Consulta: 10 de Febrero 2022]. Disponible en: https://www.jmlr.org/papers/volume11/ojala10a/ojala10a

OZDEMIR, Sinan; & SUSARLA, Divya. *Feature engineering made easy.* Birmingham-UK: Packt Publishing Ltd., 2018. [Consulta: 27 de Octubre 2021]. Disponible en: https://www.packtpub.com/product/feature-engineering-made-easy/9781787287600

PROBST, Phillip. Hyperparameters, Tuning and Meta-Learning for Random Forest and Other Machine Learning Algorithms (Thesis dissertation). [En línea] LMU München, Faculty of Mathematics, Computer Science and Statistics. München. 2019. [Consulta: 8 de Noviembre 2021]. Disponible en: https://edoc.ub.uni-muenchen.de/24557/

RANDLES, Bernadette M; et al. "Using the Jupyter Notebook as a Tool for Open Science: An Empirical Study". *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)* [en línea], 2017, pp. 1-2. [Consulta: 11 de Octubre 2021]. Disponible en: https://doi.org/10.1109/JCDL.2017.7991618

ROUHIAINEN, Lasse. Inteligencia artificial 101 cosas que debes saber hoy sobre nuestro futuro [en línea]. España-Barcelona: Alienta Editorial, 2018. [Consulta: 1 de Octubre 2021]. Disponible en: https://www.planetadelibros.com/libros_contenido_extra/40/39307_Inteligencia_artificial.pdf

SELCUK, Sule. "Predictive maintenance, its implementation and latest trends*". Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* [en línea], 2017, 231(9), pp. 1670–1679. [Consulta: 29 de Agosto 2021]. Disponible en: https://doi.org/10.1177/0954405415601640

SILVA, Willamos; & CAPRETZ, Miriam. "Assets Predictive Maintenance Using Convolutional Neural Networks". *2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)* [en línea], 2019, pp.59-66. [Consulta: 27 de Octubre 2021]. Disponible en: https://doi.org/10.1109/SNPD.2019.8935752

SUN, Deliang; et al. "A random forest model of landslide susceptibility mapping based on hyperparameter optimization using Bayes algorithm". *Geomorphology* [en línea], 2020, 362. [Consulta: 8 de Noviembre 2021]. Disponible en: https://doi.org/10.1016/j.geomorph.2020.107201

VIDYARTHI, Sriram K; et al. "Prediction of size and mass of pistachio kernels using random Forest machine learning". *Journal of Food Process Engineering* [en línea], 2020, 43. [Consulta:

27 de Octubre 2021]. Disponible en: https://doi.org/10.1111/jfpe.13473

VIEIRA, Sandra; et al. "Chapter 19 - A step-by-step tutorial on how to build a machine learning model". *Machine Learning: Methods and Applications to Brain Disorders* [en línea], 2019, pp. 343-370. [Consulta: 5 de Noviembre 2021]. Disponible en: https://doi.org/10.1016/B978-0-12-815739-8.00019-5

VIEIRA, Sandra; et al. "Main concepts in machine learning". *Machine Learning: Methods and Applications to Brain Disorders* [en línea], 2019, pp. 21-44. [Consulta: 27 de Octubre 2021]. Disponible en: https://doi.org/10.1016/B978-0-12-815739-8.00002-X

VILORIA, Amelec; et al. "Unbalanced data processing using oversampling: Machine learning". *Procedia Computer Science* [en línea], 2020, 175, pp. 108-113. [Consulta: 27 de Diciembre 2021]. ISSN: 1877-0509. Disponible en: https://doi.org/10.1016/j.procs.2020.07.018

ANEXOS

ANEXO A: PROGRAMACIÓN FINALIZADA

Detección de fallos en mantenimiento predictivo

Para la realización del modelo de detección de fallos en mantenimiento predictivo, se lo hará en los siguientes 6 pasos:

- Definición del problema
- Preparación de datos
- Ingeniería de características
- Entrenamiento del modelo
- Evaluación del modelo
- Análisis post hoc

Los datos utilizados para el desarrollo del modelo fueron tomados del repositorio de la Universidad de California, Irvine UCI.

https://archive.ics.uci.edu/ml/datasets/AI4I+2020+Predictive+Maintenance+Dataset

Definición del problema

La base de datos con la cual se realizará el modelo puede ser analizada de dos maneras, se puede realizar predicciones de si la máquina ha fallado o no y por otro lado se puede realizar predicciones sobre el modo de falla por el que la máquina ha fallado, por lo que se pueden realizar dos modelos diferentes.

Para el presente estudio se utilizará el enfoque únicamente de si la máquina ha fallado o no.

El problema es el siguiente:

Clasificar si la máquina ha fallado o no, utilizando los datos de mantenimiento predictivo del repositorio UCI

Dada la formulación del problema se pueden identificar los elementos del problema y son:

- Características: Variables de proceso, no se incluye las variables de modos de fallo debido a que puede existir fugas de datos.
- Tarea: Clasificación binaria
- Objetivo: Falla de la máquina (estado 0 y 1, lo que significa que la máquina no ha fallado y si ha fallado respectivamente).

Preparación de datos

Dentro del paso de preparación de datos se incluye:

- Análisis exploratorio de los datos
- Limpieza de los datos
- Oversampling
- División de los datos para entrenamiento y prueba

Análisis exploratorio de datos

Importando librerías

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from warnings import filterwarnings
import sklearn
from sklearn.model_selection import train_test_split
import tsfel
from sklearn.feature_selection import RFE
from sklearn.preprocessing import RobustScaler
from sklearn.metrics import roc_auc_score,classification_report,confusion_matrix,accuracy_score,precision_score,
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn import model_selection
from imblearn.over_sampling import SMOTE
from sklearn.inspection import permutation_importance
from pprint import pprint
from sklearn.model_selection import learning_curve
from sklearn.model_selection import StratifiedKFold, KFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn import metrics
from pathlib import Path
```

Creando carpetas para guardar resultados y datos

In [2]:
```
progra_dir = Path('./Resultados')
progra_dir.mkdir(exist_ok=True)

experiment_name = 'Rf_mtto'
experiment_dir = progra_dir / experiment_name
experiment_dir.mkdir(exist_ok=True)

permutacion_dir = experiment_dir / 'Permutacion'
permutacion_dir.mkdir(exist_ok=True)

datos_dir= experiment_dir / 'Datos'
datos_dir.mkdir(exist_ok=True)
```

Definiendo estilos

In [3]:
```
plt.style.use('fivethirtyeight') # Definiendo estilos para seaborn y matplotlib
sns.set_style('whitegrid')
filterwarnings('ignore')
```

Lectura de datos

In [4]:
```
Mpd= pd.read_csv(experiment_dir / 'Datos'/'ai4i2020.csv', header=0)
Mpd
```

Out[4]:

	UDI	ID_Producto	Tipo	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina	TI
0	1	M14860	M	298.1	308.6	1551	42.8	0	0	
1	2	L47181	L	298.2	308.7	1408	46.3	3	0	
2	3	L47182	L	298.1	308.5	1498	49.4	5	0	
3	4	L47183	L	298.2	308.6	1433	39.5	7	0	
4	5	L47184	L	298.2	308.7	1408	40.0	9	0	
...	...	...	...	...	...	...	...	...	...	...
9995	9996	M24855	M	298.8	308.4	1604	29.5	14	0	
9996	9997	H39410	H	298.9	308.4	1632	31.8	17	0	
9997	9998	M24857	M	299.0	308.6	1645	33.4	22	0	
9998	9999	H39412	H	299.0	308.7	1408	48.5	25	0	
9999	10000	M24859	M	299.0	308.7	1500	40.2	30	0	

10000 rows × 14 columns

Información de la base de datos

In [5]:
```
Mpd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   UDI                   10000 non-null  int64
 1   ID_Producto           10000 non-null  object
 2   Tipo                  10000 non-null  object
 3   Temperatura_aire      10000 non-null  float64
 4   Temperatura_proceso   10000 non-null  float64
 5   Velocidad_rotacional  10000 non-null  int64
 6   Torque                10000 non-null  float64
 7   Desgaste_herramienta  10000 non-null  int64
 8   Falla_maquina         10000 non-null  int64
 9   TWF                   10000 non-null  int64
 10  HDF                   10000 non-null  int64
 11  PWF                   10000 non-null  int64
 12  OSF                   10000 non-null  int64
 13  RNF                   10000 non-null  int64
```

```
dtypes: float64(3), int64(9), object(2)
memory usage: 1.1+ MB
```

Eliminación de variables referentes a modos de fallo

In [6]:
```
Mpd= Mpd.drop(['TWF','HDF','PWF','OSF','RNF'],axis=1)
Mpd
```

Out[6]:

	UDI	ID_Producto	Tipo	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
0	1	M14860	M	298.1	308.6	1551	42.8	0	0
1	2	L47181	L	298.2	308.7	1408	46.3	3	0
2	3	L47182	L	298.1	308.5	1498	49.4	5	0
3	4	L47183	L	298.2	308.6	1433	39.5	7	0
4	5	L47184	L	298.2	308.7	1408	40.0	9	0
...	...	...	...	...	...	...	...	...	...
9995	9996	M24855	M	298.8	308.4	1604	29.5	14	0
9996	9997	H39410	H	298.9	308.4	1632	31.8	17	0
9997	9998	M24857	M	299.0	308.6	1645	33.4	22	0
9998	9999	H39412	H	299.0	308.7	1408	48.5	25	0
9999	10000	M24859	M	299.0	308.7	1500	40.2	30	0

10000 rows × 9 columns

In [7]:
```
# El código siguiente ayuda a comprender los recuentos de los valores para cada una de las variables
# El recuento ayuda a comprender si los valores de la columna tienen buena información o no
for i in Mpd.columns:
    print("Value Count for ",i," is", Mpd[str(i)].value_counts())
    print('\n')
```

```
Value Count for  UDI  is 1        1
6671     1
6664     1
6665     1
6666     1
        ..
3334     1
3335     1
3336     1
3337     1
10000    1
Name: UDI, Length: 10000, dtype: int64


Value Count for  ID_Producto  is M14860    1
L53850    1
L53843    1
L53844    1
L53845    1
         ..
M18193    1
M18194    1
L50515    1
L50516    1
M24859    1
Name: ID_Producto, Length: 10000, dtype: int64


Value Count for  Tipo  is L    6000
M    2997
H    1003
Name: Tipo, dtype: int64


Value Count for  Temperatura_aire  is 300.7    279
298.9    231
297.4    230
300.5    229
298.8    227
        ...
304.4      7
296.0      6
295.4      3
295.3      3
```

```
304.5      1
Name: Temperatura_aire, Length: 93, dtype: int64


Value Count for  Temperatura_proceso  is 310.6    317
310.8    273
310.7    266
308.6    265
310.5    263
        ...
306.9      4
313.7      4
305.8      3
305.7      2
313.8      2
Name: Temperatura_proceso, Length: 82, dtype: int64


Value Count for  Velocidad_rotacional  is 1452    48
1435    43
1447    42
1429    40
1469    40
        ..
2197     1
2211     1
1905     1
1893     1
2450     1
Name: Velocidad_rotacional, Length: 941, dtype: int64


Value Count for  Torque  is 40.2    52
38.5    50
42.4    50
35.8    50
37.7    49
        ..
65.7     1
67.8     1
16.0     1
4.6      1
15.5     1
Name: Torque, Length: 577, dtype: int64


Value Count for  Desgaste_herramienta  is 0      120
2       69
5       63
7       58
59      58
       ...
237      1
239      1
241      1
251      1
253      1
Name: Desgaste_herramienta, Length: 246, dtype: int64


Value Count for  Falla_maquina  is 0    9661
1     339
Name: Falla_maquina, dtype: int64
```

De la ejecución del código anterior se concluye que las variables UDI Y ID_Producto no proporcionan buena información, por lo que en la parte de limpieza de datos serán eliminadas

Observación de valores faltantes

In [8]:
```
# Número de datos faltantes por variable
Mpd.isna().sum().sort_values()
```

Out[8]:
```
UDI                    0
ID_Producto            0
Tipo                   0
Temperatura_aire       0
Temperatura_proceso    0
Velocidad_rotacional   0
Torque                 0
Desgaste_herramienta   0
Falla_maquina          0
```

```
dtype: int64
```

Se puede apreciar que no existen valores faltantes por lo que no se hará uso de métodos de imputación.

Filas duplicadas

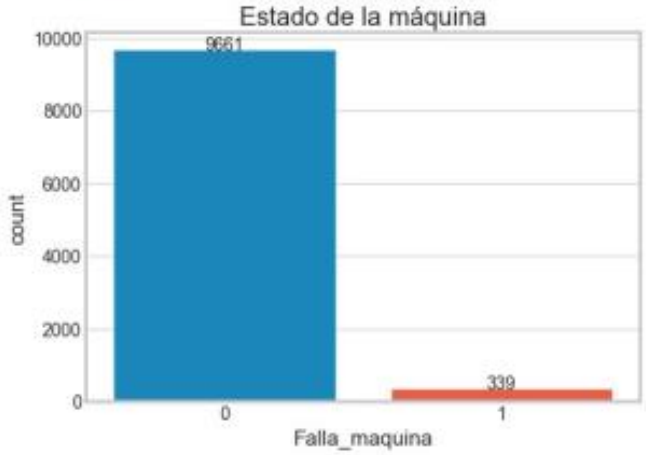
In [9]:
```
print('Filas duplicadas : ',Mpd.duplicated().sum())
```

```
Filas duplicadas :  0
```

No existen filas duplicadas

Análisis gráfico de la variable objetivo

In [10]:
```
plt.figure(figsize=(7,5))
g=sns.countplot(x='Falla_maquina', data=Mpd)
for i, u in enumerate(Mpd['Falla_maquina'].value_counts().values):
    g.text(i, u, str(u), ha= 'center')
plt.title('Estado de la máquina')
plt.show()
```

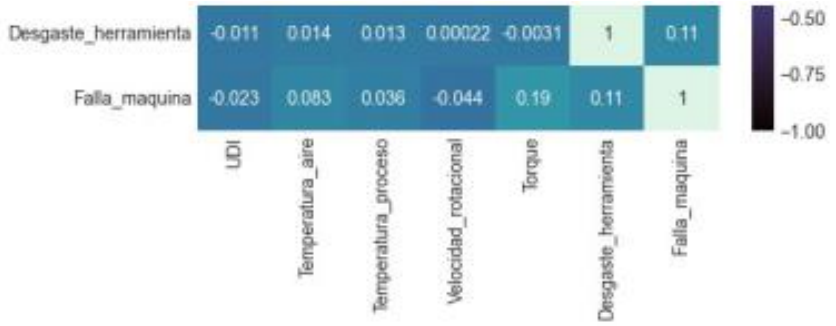


Del gráfico se puede concluir que la máquina ha fallado en 339 ocasiones de un total de 10000, mientras que no falló en 9661 ocasiones. Claramente se aprecia un desequilibrio en las clases de la variable objetivo, por lo que mas adelante se hará uso de un metodo de Oversampling (sobremuestreo)

Correlación

Mapa de calor

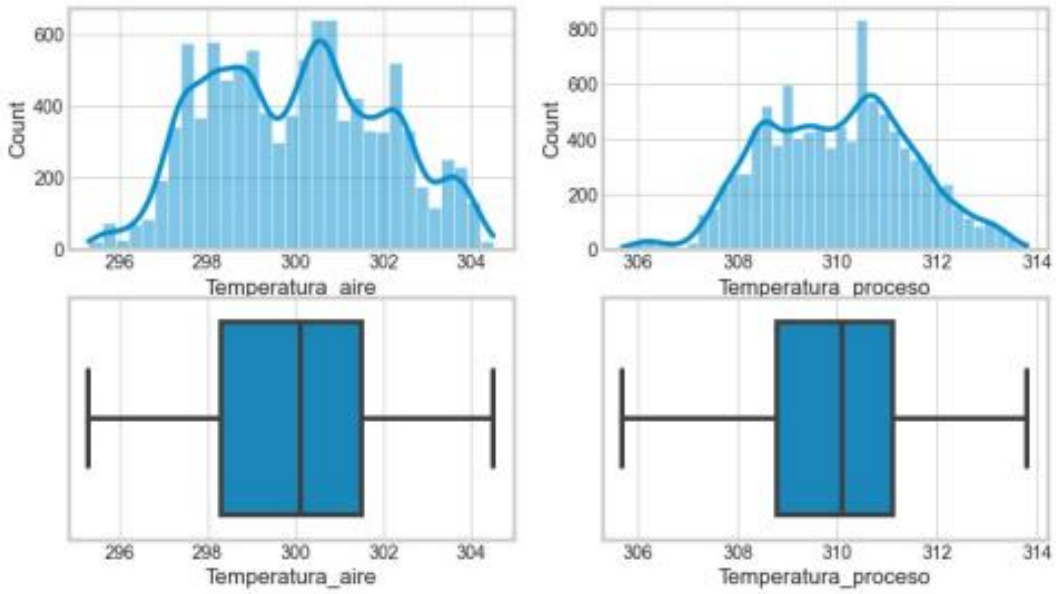
In [11]:
```
plt.figure(figsize = (8,6))
sns.heatmap(Mpd.corr(), vmin = -1.0, vmax = 1.0, center = 0, cmap = 'mako', annot = True)
plt.show()
```



Se puede apreciar que las variables Temperatura_aire y Temperatura_proceso se encuentran correlacionadas , a su vez Velocidad_rotacional y Torque también lo están

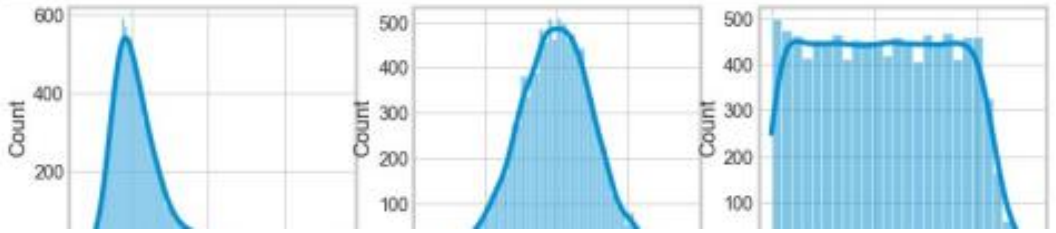
Análisis gráfico de variables

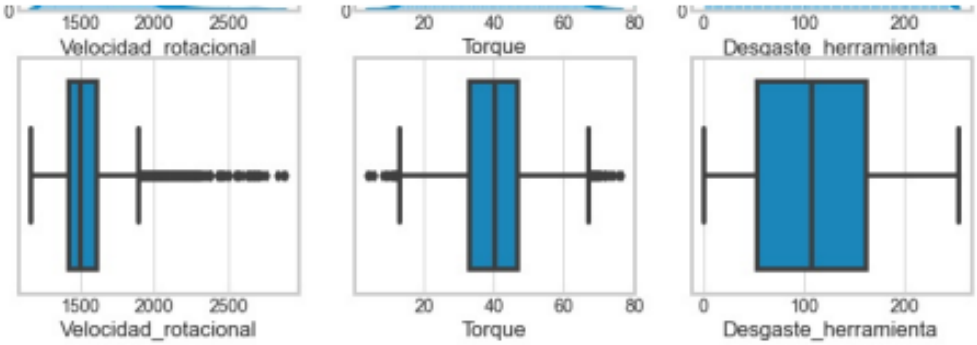
In [12]:
```
fig, ax = plt.subplots(2, 2, figsize=(12,7))
sns.histplot(data=Mpd, x='Temperatura_aire', kde=True, ax=ax[0,0])
sns.histplot(data=Mpd, x='Temperatura_proceso', kde=True, ax=ax[0,1])
sns.boxplot(data=Mpd, x='Temperatura_aire', ax=ax[1,0])
sns.boxplot(data=Mpd, x='Temperatura_proceso', ax=ax[1,1])
plt.show()
```



Del gráfico se puede concluir que las dos variables no se ajustan a una distribución normal, y tampoco tiene valores atípicos (outliers)

In [13]:
```
fig1, ax1 = plt.subplots(2, 3, figsize=(12,7))
sns.histplot(data=Mpd, x='Velocidad_rotacional', kde=True, ax=ax1[0,0])
sns.histplot(data=Mpd, x='Torque', kde=True, ax=ax1[0,1])
sns.histplot(data=Mpd, x='Desgaste_herramienta', kde=True, ax=ax1[0,2])
sns.boxplot(data=Mpd, x='Velocidad_rotacional', ax=ax1[1,0])
sns.boxplot(data=Mpd, x='Torque', ax=ax1[1,1])
sns.boxplot(data=Mpd, x='Desgaste_herramienta', ax=ax1[1,2])
plt.show()
```

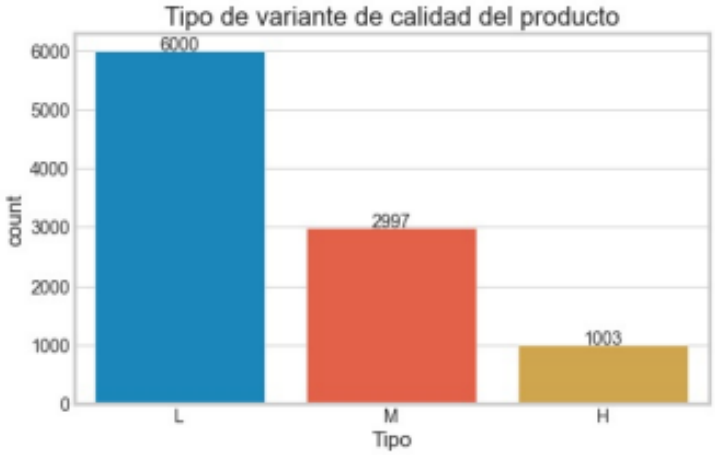




La variable Torque se ajusta a una distribución normal, pero se evidencia que junto con la variables Velocidad_rotacional, presentan valores atipicos, mientras que Desgaste_herramienta no los presenta

Debido a que se utilizará el metodo de Random Forest y se sabe que no es sensible a outliers, no se eliminará ninguno de los valores atipicos, ya que se perderían datos que pueden ser valiosos para el análisis; pero a su vez se los tratará mas adelante mediante la función RobustScaler

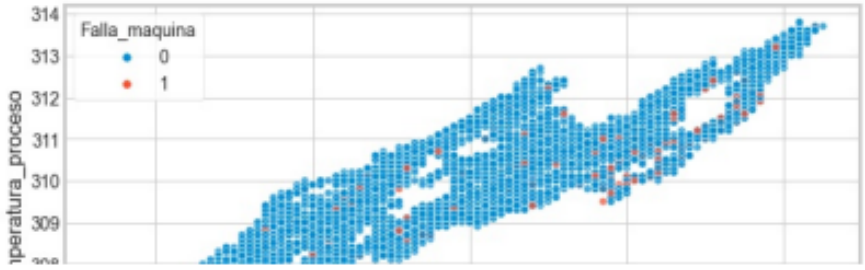
In [14]:
```
plt.figure(figsize=(8,5))
h=sns.countplot(x='Tipo', data=Mpd, order=['L','M','H'])
for i, u in enumerate(Mpd['Tipo'].value_counts().values):
    h.text(i, u, str(u), ha= 'center')
plt.title('Tipo de variante de calidad del producto')
plt.show()
```

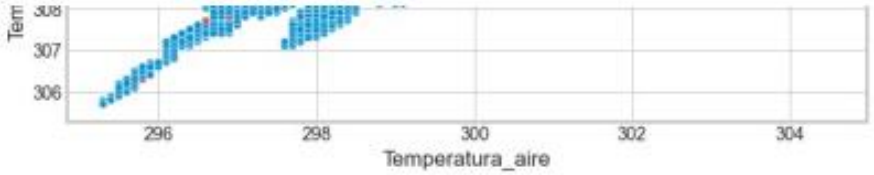


La variante L es la que mayor predomina en el conjunto de datos, seguido de M y H

Gráficos de dispersión para algunos pares de variables

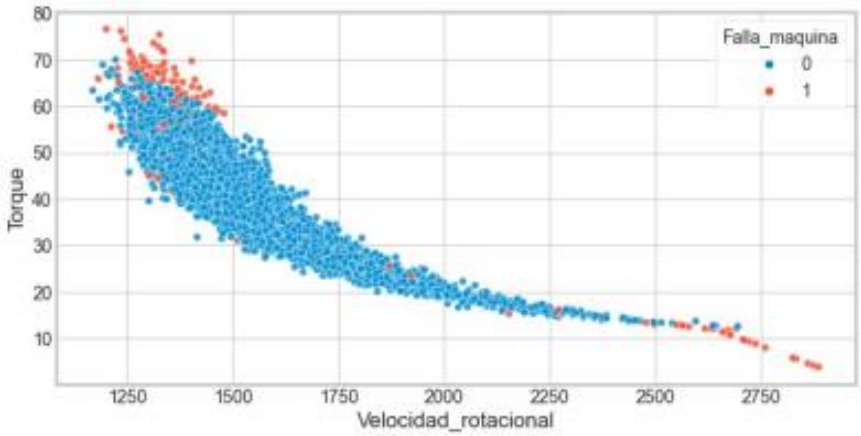
In [15]:
```
plt.figure(figsize=(10,5))
sns.scatterplot(x='Temperatura_aire', y='Temperatura_proceso',
                hue='Falla_maquina', alpha=0.75, data=Mpd)
plt.xlabel('Temperatura_aire')
plt.ylabel('Temperatura_proceso')
plt.show()
```





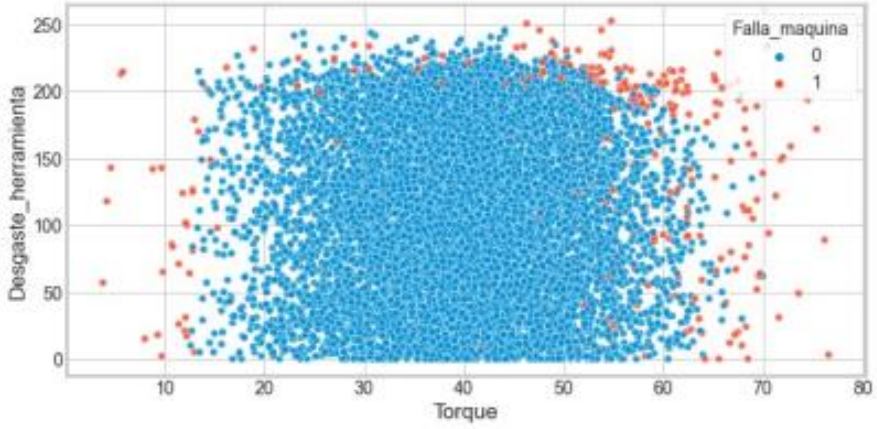
Del gráfico se observa que existe una relación lineal positiva, ya que al aumentar la temperatura del aire aumenta la temperatura del proceso

In [16]:
```
plt.figure(figsize=(10,5))
sns.scatterplot(x='Velocidad_rotacional', y='Torque', hue='Falla_maquina', alpha=0.85, data=Mpd)
plt.show()
```



Del gráfico se observa que existe relación lineal negativa, ya que al aumentar la velocidad rotacional disminuye el torque

In [17]:
```
plt.figure(figsize=(10,5))
sns.scatterplot(x='Torque', y='Desgaste_herramienta', hue='Falla_maquina', alpha=0.85, data=Mpd)
plt.show()
```



No existe relación entre los pares de variables analizados

Limpieza de los datos

Se procede a eliminar las variables anteriormente mencionadas, como son UDI Y ID_Producto; a su vez también se eliminará la variable Tipo ya que se considera que no aporta para la predicción

In [18]:
```
Mpd1= Mpd.drop(['UDI','ID_Producto','Tipo'],axis=1)
Mpd1
```

Out[18]:

	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
0	298.1	308.6	1551	42.8	0	0
1	298.2	308.7	1408	46.3	3	0
2	298.1	308.5	1498	49.4	5	0
3	298.2	308.6	1433	39.5	7	0
4	298.2	308.7	1408	40.0	9	0
...	...	...	...	...	...	...
9995	298.8	308.4	1604	29.5	14	0
9996	298.9	308.4	1632	31.8	17	0
9997	299.0	308.6	1645	33.4	22	0
9998	299.0	308.7	1408	48.5	25	0
9999	299.0	308.7	1500	40.2	30	0

10000 rows × 6 columns

Oversampling

In [19]:
```
Mpd1
```

Out[19]:

	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
0	298.1	308.6	1551	42.8	0	0
1	298.2	308.7	1408	46.3	3	0
2	298.1	308.5	1498	49.4	5	0
3	298.2	308.6	1433	39.5	7	0
4	298.2	308.7	1408	40.0	9	0
...	...	...	...	...	...	...
9995	298.8	308.4	1604	29.5	14	0
9996	298.9	308.4	1632	31.8	17	0
9997	299.0	308.6	1645	33.4	22	0
9998	299.0	308.7	1408	48.5	25	0
9999	299.0	308.7	1500	40.2	30	0

10000 rows × 6 columns

In [20]:
```
Xs= Mpd1.copy() # Definiendo variables
ys= Xs.pop('Falla_maquina')
```

In [21]:
```
sm = SMOTE(sampling_strategy= 0.98) # Aplicando código de sobremuestreo
X_res,y_res=sm.fit_resample(Xs,ys)
```

In [22]:
```
Mpd1_new= pd.concat([X_res,y_res], axis=1) #Concatenando características y variable objetivo sobremuestreados
Mpd1_new
```

Out[22]:

	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
0	298.100000	308.600000	1551	42.800000	0	0
1	298.200000	308.700000	1408	46.300000	3	0
2	298.100000	308.500000	1498	49.400000	5	0
3	298.200000	308.600000	1433	39.500000	7	0
4	298.200000	308.700000	1408	40.000000	9	0
...	...	...	...	...	...	...
19123	301.965492	310.556760	1340	55.320145	204	1
19124	302.393361	310.776660	1375	47.346681	19	1
19125	300.650682	309.813855	1311	65.053050	191	1
19126	297.532248	308.489886	1374	54.735573	215	1
19127	300.623357	311.129840	1321	67.231172	18	1

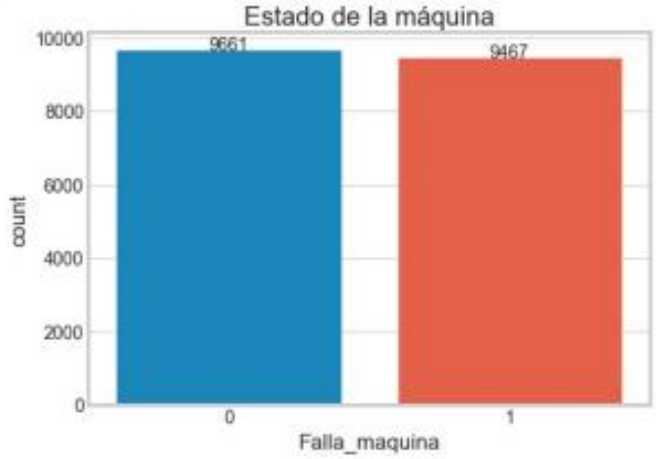
19128 rows × 6 columns

Una vez realizado el proceso de oversamplig, se paso de tener 10000 puntos de datos a 19128.

In [23]:
```
Mpd1_new.to_csv(experiment_dir / 'Datos'/'ai4ires.csv') #Guardando nuevo dataset (equilibrado)
```

Analisis gráfico de la variable objetivo una vez hecho el oversampling

In [24]:
```
plt.figure(figsize=(7,5))
g1=sns.countplot(x='Falla_maquina', data=Mpd1_new)
for i, u in enumerate(Mpd1_new['Falla_maquina'].value_counts().values):
    g1.text(i, u, str(u), ha= 'center')
plt.title('Estado de la máquina')
plt.show()
```



La gráfica muestra que se paso de 339 datos a 9467 en la clase 1 (datos cuando la máquina ha fallado).

División de datos

Se dividirá los datos 75% para entrenamiento y 25% para prueba, cabe acotar que la división se la realizará con la base de datos equilibrada

In [25]:
```
X1= Mpd1_new.copy() #Definiendo variables
y1= X1.pop('Falla_maquina')
```

In [26]:
```
X1_train, X1_test, y1_train, y1_test = train_test_split(X1,y1, random_state=0, test_size= 0.25) #Aplicando código
```

In [27]:
```
y1_train.value_counts() # Observaciones de entrenamiento para cada clase
```

Out[27]:
```
0    7252
1    7094
Name: Falla_maquina, dtype: int64
```

In [28]:
```
y1_test.value_counts() # Observaciones de prueba para cada clase
```

Out[28]:
```
0    2409
1    2373
Name: Falla_maquina, dtype: int64
```

Concatenamos los archivos de train

In [29]:
```
df_train= pd.concat([X1_train,y1_train], axis=1)
df_train.to_csv(experiment_dir / 'Datos'/'Mpd1_train.csv')
```

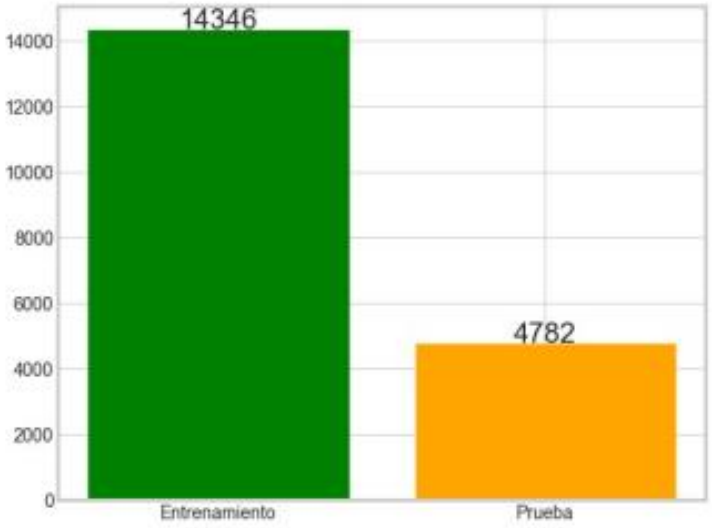
Concatenamos los archivos de test

In [30]:
```
df_test= pd.concat([X1_test,y1_test], axis=1)
df_test.to_csv(experiment_dir / 'Datos'/'Mpd1_test.csv')
```

Gráfico de la división de datos

In [31]:
```
División = ['Entrenamiento','Prueba']
Datos = [14346,4782]
colores = ['green','orange']
fig, ax3= plt.subplots(figsize=(8,6))
ax3.bar(División,Datos,width=0.8,color=colores )
for index, data in enumerate (Datos):
    plt.text(x=index, y=data+1, s=f"{data}", fontdict= dict(fontsize=22),ha='center')
plt.tight_layout()
plt.show
```

Out[31]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Ingeniería de características

Extracción de características de entrenamiento

Llamar a los datos de entrenamiento

In [32]:
```
DfT= pd.read_csv(experiment_dir / 'Datos'/'Mpd1_train.csv') # Leyendo datos de entrenamiento
DfT
```

Out[32]:

	Unnamed: 0	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
0	16510	296.982889	307.558022	2720	9.346418	18	1
1	15043	297.520958	308.211643	1403	61.481371	109	1
2	9196	298.000000	308.900000	1607	37.700000	49	0
3	15217	300.535212	311.544600	1271	58.839436	197	1
4	7587	300.400000	311.100000	1405	54.100000	193	0
...	...	...	...	...	...	...	...
14341	9225	298.000000	309.100000	1820	23.600000	120	0
14342	13123	300.400799	309.635297	1299	65.368996	48	1
14343	9845	298.300000	309.100000	1421	47.400000	33	0
14344	10799	301.925735	310.100000	1360	55.311397	66	1
14345	2732	299.700000	309.200000	1346	57.400000	138	0

14346 rows × 7 columns

Eliminación de la columna creada por defecto

In [33]:
```
DfT= DfT.drop(['Unnamed: 0'], axis= 1)
DfT
```

Out[33]:

	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
0	296.982889	307.558022	2720	9.346418	18	1
1	297.520958	308.211643	1403	61.481371	109	1
2	298.000000	308.900000	1607	37.700000	49	0
3	300.535212	311.544600	1271	58.839436	197	1
4	300.400000	311.100000	1405	54.100000	193	0
...	...	...	...	...	...	...
14341	298.000000	309.100000	1820	23.600000	120	0
14342	300.400799	309.635297	1299	65.368996	48	1
14343	298.300000	309.100000	1421	47.400000	33	0
14344	301.925735	310.100000	1360	55.311397	66	1
14345	299.700000	309.200000	1346	57.400000	138	0

14346 rows × 6 columns

LLamamos unicamente a los datos de la clase 0

In [34]:
```
DfT0= DfT[DfT['Falla_maquina']==0]
DfT0
```

Out[34]:

	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
2	298.0	308.9	1607	37.7	49	0
4	300.4	311.1	1405	54.1	193	0
6	297.4	308.8	1640	34.2	142	0
8	301.6	310.7	1679	29.5	10	0
13	297.7	307.5	1379	43.3	111	0
...	...	...	...	...	...	...
14330	300.6	309.5	1530	40.2	93	0
14336	301.7	311.0	1298	58.1	168	0
14341	298.0	309.1	1820	23.6	120	0
14343	298.3	309.1	1421	47.4	33	0
14345	299.7	309.2	1346	57.4	138	0

7252 rows × 6 columns

Definimos variables

In [35]:
```
X0T= DfT0.copy()
y0T= X0T.pop('Falla_maquina')
```

Extraemos características para la clase 0

In [36]:
```
direc = experiment_dir / 'Datos'/'caracteristicas.json'
cfg_file = tsfel.load_json(direc)
features_train0 = tsfel.time_series_features_extractor(cfg_file ,X0T, window_size=5)
features_train0
```

```
*** Feature extraction started ***
```

Progress: 100% Complete

```
*** Feature extraction finished ***
```

Out[36]:

	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Kurtosis	4_Max	4_M
0	447078.77	2.7	-1.489525	301.6	299.02	298.0	297.4	4.2	299.024671	0.522093	...	-1.331958	193.0	1
1	443969.87	0.6	-0.154622	300.6	297.98	297.5	296.6	4.0	297.983177	1.146014	...	-1.473279	156.0	
2	452585.85	0.5	-0.785088	301.8	300.86	301.0	299.8	2.0	300.860715	-0.256011	...	-1.286580	208.0	1
3	450871.94	4.5	-1.609320	303.6	300.28	299.8	297.1	6.5	300.290506	0.119880	...	-0.926514	229.0	1
4	447966.00	1.4	-1.517590	300.5	299.32	299.1	298.2	2.3	299.321232	0.146392	...	-0.298353	223.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1445	453023.96	2.2	-0.987101	303.4	301.00	300.8	297.8	5.6	301.006299	-0.408224	...	-0.577177	160.0	
1446	447678.71	3.2	-1.544148	301.8	299.22	298.8	297.3	4.5	299.225236	0.294745	...	-0.687819	112.0	
1447	446361.73	3.7	-1.763285	300.6	298.78	299.2	296.8	3.8	298.784782	-0.179426	...	-1.623260	222.0	1
1448	452000.71	1.1	-0.491272	304.1	300.66	300.6	298.3	5.8	300.666164	0.739175	...	-0.260252	219.0	
1449	451509.67	0.5	-0.080571	301.7	300.50	301.1	298.0	3.7	300.502802	-1.232228	...	-1.389254	202.0	1

1450 rows × 60 columns

Las características se extraen mediante la librería TSFEL y se aplica un tamaño de ventana de 5, esto quiere decir que del total de datos de entrenamiento para la clase 0 se dividen para 5 (7252/5), dando como resultado 1450 observaciones

Información de características extraidas

In [37]:
```
features_train0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1450 entries, 0 to 1449
Data columns (total 60 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   0_Absolute energy        1450 non-null   float64
 1   0_Interquartile range    1450 non-null   float64
 2   0_Kurtosis               1450 non-null   float64
 3   0_Max                    1450 non-null   float64
 4   0_Mean                   1450 non-null   float64
 5   0_Median                 1450 non-null   float64
 6   0_Min                    1450 non-null   float64
 7   0_Peak to peak distance  1450 non-null   float64
 8   0_Root mean square       1450 non-null   float64
 9   0_Skewness               1450 non-null   float64
 10  0_Standard deviation     1450 non-null   float64
 11  0_Variance               1450 non-null   float64
 12  1_Absolute energy        1450 non-null   float64
 13  1_Interquartile range    1450 non-null   float64
 14  1_Kurtosis               1450 non-null   float64
 15  1_Max                    1450 non-null   float64
 16  1_Mean                   1450 non-null   float64
 17  1_Median                 1450 non-null   float64
 18  1_Min                    1450 non-null   float64
 19  1_Peak to peak distance  1450 non-null   float64
 20  1_Root mean square       1450 non-null   float64
 21  1_Skewness               1450 non-null   float64
 22  1_Standard deviation     1450 non-null   float64
 23  1_Variance               1450 non-null   float64
 24  2_Absolute energy        1450 non-null   float64
 25  2_Interquartile range    1450 non-null   float64
 26  2_Kurtosis               1450 non-null   float64
 27  2_Max                    1450 non-null   float64
 28  2_Mean                   1450 non-null   float64
 29  2_Median                 1450 non-null   float64
 30  2_Min                    1450 non-null   float64
 31  2_Peak to peak distance  1450 non-null   float64
 32  2_Root mean square       1450 non-null   float64
 33  2_Skewness               1450 non-null   float64
 34  2_Standard deviation     1450 non-null   float64
 35  2_Variance               1450 non-null   float64
 36  3_Absolute energy        1450 non-null   float64
 37  3_Interquartile range    1450 non-null   float64
 38  3_Kurtosis               1450 non-null   float64
 39  3_Max                    1450 non-null   float64
 40  3_Mean                   1450 non-null   float64
```

```
 41  3_Median                  1450 non-null   float64
 42  3_Min                     1450 non-null   float64
 43  3_Peak to peak distance   1450 non-null   float64
 44  3_Root mean square        1450 non-null   float64
 45  3_Skewness                1450 non-null   float64
 46  3_Standard deviation      1450 non-null   float64
 47  3_Variance                1450 non-null   float64
 48  4_Absolute energy         1450 non-null   float64
 49  4_Interquartile range     1450 non-null   float64
 50  4_Kurtosis                1450 non-null   float64
 51  4_Max                     1450 non-null   float64
 52  4_Mean                    1450 non-null   float64
 53  4_Median                  1450 non-null   float64
 54  4_Min                     1450 non-null   float64
 55  4_Peak to peak distance   1450 non-null   float64
 56  4_Root mean square        1450 non-null   float64
 57  4_Skewness                1450 non-null   float64
 58  4_Standard deviation      1450 non-null   float64
 59  4_Variance                1450 non-null   float64
dtypes: float64(60)
memory usage: 679.8 KB
```

Donde 0 representa la variable Temperatura_aire, 1 Temperatura_proceso, 2 Velocidad_rotacional, 3 Torque y 4 Desgaste_herramienta

Llamamos únicamente a los datos de la clase 1

In [38]:
```
DfT1= DfT[DfT['Falla_maquina']==1]
DfT1
```

Out[38]:

	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
0	296.982889	307.558022	2720	9.346418	18	1
1	297.520958	308.211643	1403	61.481371	109	1
3	300.535212	311.544600	1271	58.839436	197	1
5	300.300700	310.587514	1296	62.381915	204	1
7	302.578528	311.219516	1353	45.618948	194	1
...	...	...	...	...	...	...
14338	297.969231	308.365898	1332	72.125128	152	1
14339	300.821667	310.278422	2533	12.895497	67	1
14340	302.462686	311.985684	1263	69.748709	232	1
14342	300.400799	309.635297	1299	65.368996	48	1
14344	301.925735	310.100000	1360	55.311397	66	1

7094 rows × 6 columns

Definimos variables

In [39]:
```
X1T= DfT1.copy()
y1T= X1T.pop('Falla_maquina')
```

Extraemos características para la clase 1

In [40]:
```
cfg_file = tsfel.load_json(direc)
features_train1 = tsfel.time_series_features_extractor(cfg_file ,X1T, window_size=5)
features_train1
```

```
*** Feature extraction started ***
```

Progress: 100% Complete

```
*** Feature extraction finished ***
```

Out[40]:

	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Kurt
0	448773.245999	3.014253	-1.413802	302.578528	299.583657	300.300700	296.982889	5.595639	299.590803	0.038924	...	-0.907
1	451838.567611	2.459972	-1.171554	302.825359	300.606734	300.958855	297.708516	5.116843	300.612231	-0.395591	...	0.038
2	451436.527965	1.407331	-0.964822	302.750528	300.474929	300.111694	298.401333	4.349194	300.478461	0.197918	...	0.041

3	454453.155280	1.954608	-1.705123	302.612192	301.478668	302.205801	299.912813	2.699379	301.480731	-0.410828	...	-0.767
4	454486.431868	1.419210	-0.392404	303.966999	301.483587	302.051771	297.404179	6.562820	301.491768	-0.922897	...	-0.404
...	...	...	...	...	...	...	...	...	...	...	...	
1413	455268.109505	1.760287	-0.738220	303.526142	301.746365	301.793707	298.837610	4.688532	301.750927	-0.706806	...	-0.229
1414	457073.076018	0.344394	-0.235447	303.964314	302.347270	302.055376	301.406053	2.558261	302.348500	1.035544	...	-0.910
1415	457364.563627	1.505611	-0.962917	303.630080	302.442771	302.588304	300.524453	3.105627	302.444892	-0.596765	...	-0.082
1416	454929.854765	1.304574	-0.384823	303.700000	301.632690	301.932887	298.089843	5.610157	301.638809	-0.950318	...	-1.048
1417	450589.525402	2.697740	-1.520300	302.443235	300.191768	300.691839	297.969231	4.474004	300.196444	-0.100553	...	-0.050

1418 rows × 60 columns

De la misma forma que en el anterior caso se aplica un tamaño de ventana de 5, (7094/5) dando un total de 1418 observaciones

Información de características extraidas

In [41]:
```
features_train1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1418 entries, 0 to 1417
Data columns (total 60 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   0_Absolute energy        1418 non-null   float64
 1   0_Interquartile range    1418 non-null   float64
 2   0_Kurtosis               1418 non-null   float64
 3   0_Max                    1418 non-null   float64
 4   0_Mean                   1418 non-null   float64
 5   0_Median                 1418 non-null   float64
 6   0_Min                    1418 non-null   float64
 7   0_Peak to peak distance  1418 non-null   float64
 8   0_Root mean square       1418 non-null   float64
 9   0_Skewness               1418 non-null   float64
 10  0_Standard deviation     1418 non-null   float64
 11  0_Variance               1418 non-null   float64
 12  1_Absolute energy        1418 non-null   float64
 13  1_Interquartile range    1418 non-null   float64
 14  1_Kurtosis               1418 non-null   float64
 15  1_Max                    1418 non-null   float64
 16  1_Mean                   1418 non-null   float64
 17  1_Median                 1418 non-null   float64
 18  1_Min                    1418 non-null   float64
 19  1_Peak to peak distance  1418 non-null   float64
 20  1_Root mean square       1418 non-null   float64
 21  1_Skewness               1418 non-null   float64
 22  1_Standard deviation     1418 non-null   float64
 23  1_Variance               1418 non-null   float64
 24  2_Absolute energy        1418 non-null   float64
 25  2_Interquartile range    1418 non-null   float64
 26  2_Kurtosis               1418 non-null   float64
 27  2_Max                    1418 non-null   float64
 28  2_Mean                   1418 non-null   float64
 29  2_Median                 1418 non-null   float64
 30  2_Min                    1418 non-null   float64
 31  2_Peak to peak distance  1418 non-null   float64
 32  2_Root mean square       1418 non-null   float64
 33  2_Skewness               1418 non-null   float64
 34  2_Standard deviation     1418 non-null   float64
 35  2_Variance               1418 non-null   float64
 36  3_Absolute energy        1418 non-null   float64
 37  3_Interquartile range    1418 non-null   float64
 38  3_Kurtosis               1418 non-null   float64
 39  3_Max                    1418 non-null   float64
 40  3_Mean                   1418 non-null   float64
 41  3_Median                 1418 non-null   float64
 42  3_Min                    1418 non-null   float64
 43  3_Peak to peak distance  1418 non-null   float64
 44  3_Root mean square       1418 non-null   float64
 45  3_Skewness               1418 non-null   float64
 46  3_Standard deviation     1418 non-null   float64
 47  3_Variance               1418 non-null   float64
 48  4_Absolute energy        1418 non-null   float64
 49  4_Interquartile range    1418 non-null   float64
 50  4_Kurtosis               1418 non-null   float64
 51  4_Max                    1418 non-null   float64
```

```
 52  4_Mean                    1418 non-null   float64
 53  4_Median                  1418 non-null   float64
 54  4_Min                     1418 non-null   float64
 55  4_Peak to peak distance   1418 non-null   float64
 56  4_Root mean square        1418 non-null   float64
 57  4_Skewness                1418 non-null   float64
 58  4_Standard deviation      1418 non-null   float64
 59  4_Variance                1418 non-null   float64
dtypes: float64(60)
memory usage: 664.8 KB
```

Añadimos la variable objetivo Falla_maquina al conjunto de características de clase 0

In [42]:
```
new_features_train0= features_train0.assign(Falla_maquina=0)
new_features_train0
```

Out[42]:

	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Max	4_Mean	4_Med
0	447078.77	2.7	-1.489525	301.6	299.02	298.0	297.4	4.2	299.024671	0.522093	...	193.0	101.0	11
1	443969.87	0.6	-0.154622	300.6	297.98	297.5	296.6	4.0	297.983177	1.146014	...	156.0	92.4	7
2	452585.85	0.5	-0.785088	301.8	300.86	301.0	299.8	2.0	300.860715	-0.256011	...	208.0	117.6	12
3	450871.94	4.5	-1.609320	303.6	300.28	299.8	297.1	6.5	300.290506	0.119880	...	229.0	124.8	10
4	447966.00	1.4	-1.517590	300.5	299.32	299.1	298.2	2.3	299.321232	0.146392	...	223.0	157.6	16
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1445	453023.96	2.2	-0.987101	303.4	301.00	300.8	297.8	5.6	301.006299	-0.408224	...	160.0	77.6	6
1446	447678.71	3.2	-1.544148	301.8	299.22	298.8	297.3	4.5	299.225236	0.294745	...	112.0	67.6	6
1447	446361.73	3.7	-1.763285	300.6	298.78	299.2	296.8	3.8	298.784782	-0.179426	...	222.0	120.8	8
1448	452000.71	1.1	-0.491272	304.1	300.66	300.6	298.3	5.8	300.666164	0.739175	...	219.0	99.2	8
1449	451509.67	0.5	-0.080571	301.7	300.50	301.1	298.0	3.7	300.502802	-1.232228	...	202.0	136.8	12

1450 rows × 61 columns

Añadimos la variable objetivo Falla_maquina al conjunto de características de clase 1

In [43]:
```
new_features_train1= features_train1.assign(Falla_maquina=1)
new_features_train1
```

Out[43]:

	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Max
0	448773.245999	3.014253	-1.413802	302.578528	299.583657	300.300700	296.982889	5.595639	299.590803	0.038924	...	204.0
1	451838.567611	2.459972	-1.171554	302.825359	300.606734	300.958855	297.708516	5.116843	300.612231	-0.395591	...	201.0
2	451436.527965	1.407331	-0.964822	302.750528	300.474929	300.111694	298.401333	4.349194	300.478461	0.197918	...	227.0
3	454453.155280	1.954608	-1.705123	302.612192	301.478668	302.205801	299.912813	2.699379	301.480731	-0.410828	...	193.0
4	454486.431868	1.419210	-0.392404	303.966999	301.483587	302.051771	297.404179	6.562820	301.491768	-0.922897	...	234.0
...	...	...	...	...	...	...	...	...	...	...	...	...
1413	455268.109505	1.760287	-0.738220	303.526142	301.746365	301.793707	298.837610	4.688532	301.750927	-0.706806	...	205.0
1414	457073.076018	0.344394	-0.235447	303.964314	302.347270	302.055376	301.406053	2.558261	302.348500	1.035544	...	200.0
1415	457364.563627	1.505611	-0.962917	303.630080	302.442771	302.588304	300.524453	3.105627	302.444892	-0.596765	...	216.0
1416	454929.854765	1.304574	-0.384823	303.700000	301.632690	301.932887	298.089843	5.610157	301.638809	-0.950318	...	217.0
1417	450589.525402	2.697740	-1.520300	302.443235	300.191768	300.691839	297.969231	4.474004	300.196444	-0.100553	...	207.0

1418 rows × 61 columns

Concatenamos archivos de caracteristicas de entrenamiento (clase 0 y clase 1)

In [44]:
```
Features_train_fin= pd.concat([new_features_train0,new_features_train1], axis=0)
Features_train_fin.to_csv(experiment_dir / 'Datos'/'Features_train_fin.csv')
```

In [45]:

In [45]:
```
Features_train_fin
```

Out[45]:

	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Max
0	447078.770000	2.700000	-1.489525	301.600000	299.020000	298.000000	297.400000	4.200000	299.024671	0.522093	...	193.0
1	443969.870000	0.600000	-0.154622	300.600000	297.980000	297.500000	296.600000	4.000000	297.983177	1.146014	...	156.0
2	452585.850000	0.500000	-0.785088	301.800000	300.860000	301.000000	299.800000	2.000000	300.860715	-0.256011	...	208.0
3	450871.940000	4.500000	-1.609320	303.600000	300.280000	299.800000	297.100000	6.500000	300.290506	0.119880	...	229.0
4	447966.000000	1.400000	-1.517590	300.500000	299.320000	299.100000	298.200000	2.300000	299.321232	0.146392	...	223.0
...	...	...	...	...	...	...	...	...	...	...	...	...
1413	455268.109505	1.760287	-0.738220	303.526142	301.746365	301.793707	298.837610	4.688532	301.750927	-0.706806	...	205.0
1414	457073.076018	0.344394	-0.235447	303.964314	302.347270	302.055376	301.406053	2.558261	302.348500	1.035544	...	200.0
1415	457364.563627	1.505611	-0.962917	303.630080	302.442771	302.588304	300.524453	3.105627	302.444892	-0.596765	...	216.0
1416	454929.854765	1.304574	-0.384823	303.700000	301.632690	301.932887	298.089843	5.610157	301.638809	-0.950318	...	217.0
1417	450589.525402	2.697740	-1.520300	302.443235	300.191768	300.691839	297.969231	4.474004	300.196444	-0.100553	...	207.0

2868 rows × 61 columns

Extraccion de caracteristicas de prueba

Llamar a los datos de prueba

In [46]:
```
Dft= pd.read_csv(experiment_dir / 'Datos'/'Mpd1_test.csv')
Dft
```

Out[46]:

	Unnamed: 0	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
0	12469	303.317915	311.508958	1371	47.228340	60	1
1	12361	301.321087	310.673708	1447	59.515775	145	1
2	10393	298.791116	308.906354	1345	58.884762	202	1
3	1707	298.200000	307.800000	1439	41.600000	62	0
4	4747	303.300000	311.200000	1763	27.300000	27	0
...	...	...	...	...	...	...	...
4777	11160	301.518775	310.629494	1332	54.420639	201	1
4778	10418	298.965774	309.575233	1473	56.809896	185	1
4779	4840	303.400000	311.900000	1298	63.900000	59	1
4780	18680	298.496070	309.161885	1430	53.319262	30	1
4781	8001	300.800000	312.000000	1374	50.200000	154	0

4782 rows × 7 columns

Eliminación de la columna creada por defecto

In [47]:
```
Dft= Dft.drop(['Unnamed: 0'],axis=1)
Dft
```

Out[47]:

	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
0	303.317915	311.508958	1371	47.228340	60	1
1	301.321087	310.673708	1447	59.515775	145	1
2	298.791116	308.906354	1345	58.884762	202	1
3	298.200000	307.800000	1439	41.600000	62	0
4	303.300000	311.200000	1763	27.300000	27	0
...	...	...	...	...	...	...
4777	301.518775	310.629494	1332	54.420639	201	1
4778	298.965774	309.575233	1473	56.809896	185	1
4779	303.400000	311.900000	1298	63.900000	59	1
4780	298.496070	309.161885	1430	53.319262	30	1

4781	300.800000	312.000000	1374	50.200000	154	0

4782 rows × 6 columns

Llamamos únicamente a los datos de la clase 0

In [48]:
```
Dft0= Dft[Dft['Falla_maquina']==0]
Dft0
```

Out[48]:

	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
3	298.2	307.8	1439	41.6	62	0
4	303.3	311.2	1763	27.3	27	0
5	300.7	310.7	1416	40.4	214	0
6	300.7	309.4	1589	35.0	52	0
9	301.6	311.0	1431	42.3	129	0
...	...	...	...	...	...	...
4771	300.7	310.8	1475	42.0	147	0
4772	300.0	311.4	1927	24.6	132	0
4773	299.7	310.3	1495	43.1	101	0
4775	298.6	309.1	1537	35.7	198	0
4781	300.8	312.0	1374	50.2	154	0

2409 rows × 6 columns

Definimos variables

In [49]:
```
X0t= Dft0.copy()
y0t= X0t.pop('Falla_maquina')
```

Extraemos características para la clase 0

In [50]:
```
cfg_file = tsfel.load_json(direc)
features_test0 = tsfel.time_series_features_extractor(cfg_file ,X0t, window_size=5)
features_test0
```

```
*** Feature extraction started ***
```
Progress: 100% Complete

```
*** Feature extraction finished ***
```

Out[50]:

	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Kurtosis	4_Max	4_Me
0	452717.67	0.9	-0.666767	303.3	300.90	300.7	298.2	5.1	300.904526	-0.246094	...	-0.905331	214.0	9
1	451460.26	1.8	-0.900100	303.7	300.48	299.8	297.9	5.8	300.486359	0.428214	...	-1.178424	219.0	11
2	447195.75	3.0	-1.774981	300.7	299.06	299.8	297.2	3.5	299.063789	-0.302690	...	-1.716749	184.0	10
3	447320.91	3.2	-1.629334	301.3	299.10	298.3	296.7	4.6	299.105637	0.137015	...	-1.046648	199.0	13
4	454647.43	3.4	-1.745976	303.7	301.54	300.7	299.5	4.2	301.545164	0.266768	...	-0.050307	137.0	10
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
476	447790.05	0.8	-0.630982	300.9	299.26	299.4	297.2	3.7	299.262443	-0.473655	...	-1.372402	208.0	13
477	448275.31	2.5	-1.294927	302.0	299.42	299.1	297.5	4.5	299.424551	0.375985	...	-1.423638	211.0	13
478	452103.77	0.6	-0.758264	301.6	300.70	300.7	300.2	1.4	300.700439	0.707726	...	-1.298271	209.0	12
479	447370.44	0.2	-0.120378	301.3	299.12	298.8	297.9	3.4	299.122196	1.119443	...	-1.116011	200.0	12
480	450369.76	1.3	-0.889517	302.3	300.12	300.1	298.1	4.2	300.123228	0.143835	...	-0.100673	191.0	13

481 rows × 60 columns

Al aplicar el tamaño de ventana de 5 (2409/5) se obtiene 481 observaciones

Información de características extraídas

In [51]:
```
features_test0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 481 entries, 0 to 480
Data columns (total 60 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   0_Absolute energy        481 non-null    float64
 1   0_Interquartile range    481 non-null    float64
 2   0_Kurtosis               481 non-null    float64
 3   0_Max                    481 non-null    float64
 4   0_Mean                   481 non-null    float64
 5   0_Median                 481 non-null    float64
 6   0_Min                    481 non-null    float64
 7   0_Peak to peak distance  481 non-null    float64
 8   0_Root mean square       481 non-null    float64
 9   0_Skewness               481 non-null    float64
 10  0_Standard deviation     481 non-null    float64
 11  0_Variance               481 non-null    float64
 12  1_Absolute energy        481 non-null    float64
 13  1_Interquartile range    481 non-null    float64
 14  1_Kurtosis               481 non-null    float64
 15  1_Max                    481 non-null    float64
 16  1_Mean                   481 non-null    float64
 17  1_Median                 481 non-null    float64
 18  1_Min                    481 non-null    float64
 19  1_Peak to peak distance  481 non-null    float64
 20  1_Root mean square       481 non-null    float64
 21  1_Skewness               481 non-null    float64
 22  1_Standard deviation     481 non-null    float64
 23  1_Variance               481 non-null    float64
 24  2_Absolute energy        481 non-null    float64
 25  2_Interquartile range    481 non-null    float64
 26  2_Kurtosis               481 non-null    float64
 27  2_Max                    481 non-null    float64
 28  2_Mean                   481 non-null    float64
 29  2_Median                 481 non-null    float64
 30  2_Min                    481 non-null    float64
 31  2_Peak to peak distance  481 non-null    float64
 32  2_Root mean square       481 non-null    float64
 33  2_Skewness               481 non-null    float64
 34  2_Standard deviation     481 non-null    float64
 35  2_Variance               481 non-null    float64
 36  3_Absolute energy        481 non-null    float64
 37  3_Interquartile range    481 non-null    float64
 38  3_Kurtosis               481 non-null    float64
 39  3_Max                    481 non-null    float64
 40  3_Mean                   481 non-null    float64
 41  3_Median                 481 non-null    float64
 42  3_Min                    481 non-null    float64
 43  3_Peak to peak distance  481 non-null    float64
 44  3_Root mean square       481 non-null    float64
 45  3_Skewness               481 non-null    float64
 46  3_Standard deviation     481 non-null    float64
 47  3_Variance               481 non-null    float64
 48  4_Absolute energy        481 non-null    float64
 49  4_Interquartile range    481 non-null    float64
 50  4_Kurtosis               481 non-null    float64
 51  4_Max                    481 non-null    float64
 52  4_Mean                   481 non-null    float64
 53  4_Median                 481 non-null    float64
 54  4_Min                    481 non-null    float64
 55  4_Peak to peak distance  481 non-null    float64
 56  4_Root mean square       481 non-null    float64
 57  4_Skewness               481 non-null    float64
 58  4_Standard deviation     481 non-null    float64
 59  4_Variance               481 non-null    float64
dtypes: float64(60)
memory usage: 225.6 KB
```

Llamamos únicamente a los datos de la clase 1

In [52]:
```
Dft1= Dft[Dft['Falla_maquina']==1]
Dft1
```

Out[52]:

	Temperatura_aire	Temperatura_proceso	Velocidad_rotacional	Torque	Desgaste_herramienta	Falla_maquina
0	303.317915	311.508958	1371	47.228340	60	1
1	301.321087	310.673708	1447	59.515775	145	1
2	298.791116	308.906354	1345	58.884762	202	1
7	297.080087	307.633108	2722	9.249106	17	1
8	299.128257	309.766132	2599	12.331463	95	1
...	...	...	...	...	...	...
4776	298.483494	309.666426	1375	54.310016	215	1
4777	301.518775	310.629494	1332	54.420639	201	1
4778	298.965774	309.575233	1473	56.809896	185	1
4779	303.400000	311.900000	1298	63.900000	59	1
4780	298.496070	309.161885	1430	53.319262	30	1

2373 rows × 6 columns

Definimos variables

In [53]:
```
X1t= Dft1.copy()
y1t= X1t.pop('Falla_maquina')
```

Extraemos características para la clase 1

In [54]:
```
cfg_file = tsfel.load_json(direc)
features_test1 = tsfel.time_series_features_extractor(cfg_file ,X1t, window_size=5)
features_test1
```

```
*** Feature extraction started ***
```
Progress: 100% Complete

```
*** Feature extraction finished ***
```

Out[54]:

	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Kurto
0	449806.578551	2.529971	-1.151752	303.317915	299.927692	299.128257	297.080087	6.237828	299.935519	0.326377	...	-1.205
1	452679.937773	3.041264	-1.367265	302.601570	300.885823	302.085855	297.800000	4.801570	300.891987	-0.599852	...	-1.821
2	452989.323675	3.078722	-1.585476	302.858655	300.988979	302.190281	298.200000	4.658655	300.994792	-0.471591	...	-1.712
3	450946.255891	0.566869	-0.300025	301.872362	300.312210	300.577720	297.812762	4.059600	300.315253	-0.935500	...	0.003
4	453447.572441	3.253383	-1.464649	303.280977	301.139775	302.354747	297.916536	5.364441	301.146998	-0.505490	...	-0.906
...	...	...	...	...	...	...	...	...	...	...	...	
469	454432.051615	1.922313	-1.223207	302.740134	301.470825	302.200000	299.290596	3.449538	301.473731	-0.640366	...	-0.966
470	452336.829017	1.947779	-0.629441	302.305405	300.773163	301.264656	297.718077	4.587328	300.777934	-0.887938	...	-1.458
471	453920.433445	0.377159	-0.056961	303.329983	301.297099	301.988277	297.381527	5.948455	301.303977	-1.202958	...	-1.456
472	450016.451379	0.716976	0.005756	303.291537	300.000739	299.144217	298.647683	4.643854	300.005484	1.323876	...	-0.153
473	452007.497456	0.350024	0.103543	301.518775	300.666378	301.089208	298.483494	3.035281	300.668421	-1.385645	...	-1.631

474 rows × 60 columns

Al aplicar el tamaño de ventana de 5 (2373/5) se obtiene 474 observaciones

Información de características extraídas

In [55]:
```
features_test1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 474 entries, 0 to 473
Data columns (total 60 columns):
```

```
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   0_Absolute energy      474 non-null    float64
 1   0_Interquartile range  474 non-null    float64
 2   0_Kurtosis             474 non-null    float64
 3   0_Max                  474 non-null    float64
 4   0_Mean                 474 non-null    float64
 5   0_Median               474 non-null    float64
 6   0_Min                  474 non-null    float64
 7   0_Peak to peak distance 474 non-null   float64
 8   0_Root mean square     474 non-null    float64
 9   0_Skewness             474 non-null    float64
 10  0_Standard deviation   474 non-null    float64
 11  0_Variance             474 non-null    float64
 12  1_Absolute energy      474 non-null    float64
 13  1_Interquartile range  474 non-null    float64
 14  1_Kurtosis             474 non-null    float64
 15  1_Max                  474 non-null    float64
 16  1_Mean                 474 non-null    float64
 17  1_Median               474 non-null    float64
 18  1_Min                  474 non-null    float64
 19  1_Peak to peak distance 474 non-null   float64
 20  1_Root mean square     474 non-null    float64
 21  1_Skewness             474 non-null    float64
 22  1_Standard deviation   474 non-null    float64
 23  1_Variance             474 non-null    float64
 24  2_Absolute energy      474 non-null    float64
 25  2_Interquartile range  474 non-null    float64
 26  2_Kurtosis             474 non-null    float64
 27  2_Max                  474 non-null    float64
 28  2_Mean                 474 non-null    float64
 29  2_Median               474 non-null    float64
 30  2_Min                  474 non-null    float64
 31  2_Peak to peak distance 474 non-null   float64
 32  2_Root mean square     474 non-null    float64
 33  2_Skewness             474 non-null    float64
 34  2_Standard deviation   474 non-null    float64
 35  2_Variance             474 non-null    float64
 36  3_Absolute energy      474 non-null    float64
 37  3_Interquartile range  474 non-null    float64
 38  3_Kurtosis             474 non-null    float64
 39  3_Max                  474 non-null    float64
 40  3_Mean                 474 non-null    float64
 41  3_Median               474 non-null    float64
 42  3_Min                  474 non-null    float64
 43  3_Peak to peak distance 474 non-null   float64
 44  3_Root mean square     474 non-null    float64
 45  3_Skewness             474 non-null    float64
 46  3_Standard deviation   474 non-null    float64
 47  3_Variance             474 non-null    float64
 48  4_Absolute energy      474 non-null    float64
 49  4_Interquartile range  474 non-null    float64
 50  4_Kurtosis             474 non-null    float64
 51  4_Max                  474 non-null    float64
 52  4_Mean                 474 non-null    float64
 53  4_Median               474 non-null    float64
 54  4_Min                  474 non-null    float64
 55  4_Peak to peak distance 474 non-null   float64
 56  4_Root mean square     474 non-null    float64
 57  4_Skewness             474 non-null    float64
 58  4_Standard deviation   474 non-null    float64
 59  4_Variance             474 non-null    float64
dtypes: float64(60)
memory usage: 222.3 KB
```

Añadimos la variable objetivo Falla_maquina a la clase 0

In [56]:
```
new_features_test0= features_test0.assign(Falla_maquina=0)
new_features_test0
```

Out[56]:

	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Max	4_Mean	4_Media
0	452717.67	0.9	-0.666767	303.3	300.90	300.7	298.2	5.1	300.904526	-0.246094	...	214.0	96.8	62
1	451460.26	1.8	-0.900100	303.7	300.48	299.8	297.9	5.8	300.486359	0.428214	...	219.0	110.2	116
2	447195.75	3.0	-1.774981	300.7	299.06	299.8	297.2	3.5	299.063789	-0.302690	...	184.0	105.2	139
3	447320.91	3.2	-1.629334	301.3	299.10	298.3	296.7	4.6	299.105637	0.137015	...	199.0	130.4	148
4	454647.43	3.4	-1.745976	303.7	301.54	300.7	299.5	4.2	301.545164	0.266768	...	137.0	105.2	114
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
476	447790.05	0.8	-0.630982	300.9	299.26	299.4	297.2	3.7	299.262443	-0.473655	...	208.0	139.2	172

477	448275.31	2.5	-1.294927	302.0	299.42	299.1	297.5	4.5	299.424551	0.375985	...	211.0	132.2	163
478	452103.77	0.6	-0.758264	301.6	300.70	300.7	300.2	1.4	300.700439	0.707726	...	209.0	125.8	113
479	447370.44	0.2	-0.120378	301.3	299.12	298.8	297.9	3.4	299.122196	1.119443	...	200.0	129.0	125
480	450369.76	1.3	-0.889517	302.3	300.12	300.1	298.1	4.2	300.123228	0.143835	...	191.0	135.8	149

481 rows × 61 columns

Añadimos la variable objetivo Falla_maquina a la clase 1

In [57]:
```
new_features_test1= features_test1.assign(Falla_maquina=1)
new_features_test1
```

Out[57]:

	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Max
0	449806.578551	2.529971	-1.151752	303.317915	299.927692	299.128257	297.080087	6.237828	299.935519	0.326377	...	202.0
1	452679.937773	3.041264	-1.367265	302.601570	300.885823	302.085855	297.800000	4.801570	300.891987	-0.599852	...	206.0
2	452989.323675	3.078722	-1.585476	302.858655	300.988979	302.190281	298.200000	4.658655	300.994792	-0.471591	...	208.0
3	450946.255891	0.566869	-0.300025	301.872362	300.312210	300.577720	297.812762	4.059600	300.315253	-0.935500	...	234.0
4	453447.572441	3.253383	-1.464649	303.280977	301.139775	302.354747	297.916536	5.364441	301.146998	-0.505490	...	208.0
...	...	...	...	...	...	...	...	...	...	...	...	...
469	454432.051615	1.922313	-1.223207	302.740134	301.470825	302.200000	299.290596	3.449538	301.473731	-0.640366	...	166.0
470	452336.829017	1.947779	-0.629441	302.305405	300.773163	301.264656	297.718077	4.587328	300.777934	-0.887938	...	235.0
471	453920.433445	0.377159	-0.056961	303.329983	301.297099	301.988277	297.381527	5.948455	301.303977	-1.202958	...	207.0
472	450016.451379	0.716976	0.005756	303.291537	300.000739	299.144217	298.647683	4.643854	300.005484	1.323876	...	229.0
473	452007.497456	0.350024	0.103543	301.518775	300.666378	301.089208	298.483494	3.035281	300.668421	-1.385645	...	215.0

474 rows × 61 columns

Concatenamos archivos de características de prueba (clase 0 y clase 1)

In [58]:
```
Features_test_fin= pd.concat([new_features_test0,new_features_test1], axis=0)
Features_test_fin.to_csv(experiment_dir / 'Datos'/'Features_test_fin.csv')
```

In [59]:
```
Features_test_fin
```

Out[59]:

	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Max
0	452717.670000	0.900000	-0.666767	303.300000	300.900000	300.700000	298.200000	5.100000	300.904526	-0.246094	...	214.0
1	451460.260000	1.800000	-0.900100	303.700000	300.480000	299.800000	297.900000	5.800000	300.486359	0.428214	...	219.0
2	447195.750000	3.000000	-1.774981	300.700000	299.060000	299.800000	297.200000	3.500000	299.063789	-0.302690	...	184.0
3	447320.910000	3.200000	-1.629334	301.300000	299.100000	298.300000	296.700000	4.600000	299.105637	0.137015	...	199.0
4	454647.430000	3.400000	-1.745976	303.700000	301.540000	300.700000	299.500000	4.200000	301.545164	0.266768	...	137.0
...	...	...	...	...	...	...	...	...	...	...	...	...
469	454432.051615	1.922313	-1.223207	302.740134	301.470825	302.200000	299.290596	3.449538	301.473731	-0.640366	...	166.0
470	452336.829017	1.947779	-0.629441	302.305405	300.773163	301.264656	297.718077	4.587328	300.777934	-0.887938	...	235.0
471	453920.433445	0.377159	-0.056961	303.329983	301.297099	301.988277	297.381527	5.948455	301.303977	-1.202958	...	207.0
472	450016.451379	0.716976	0.005756	303.291537	300.000739	299.144217	298.647683	4.643854	300.005484	1.323876	...	229.0
473	452007.497456	0.350024	0.103543	301.518775	300.666378	301.089208	298.483494	3.035281	300.668421	-1.385645	...	215.0

955 rows × 61 columns

Concatenacion de archivos de features

In [60]:
```
Features_fin= pd.concat([Features_train_fin,Features_test_fin], axis=0, ignore_index= True)
Features_fin.to_csv(experiment_dir / 'Datos'/'Features_fin.csv')
```

In [61]:
```
Features_fin
```

Out[61]:

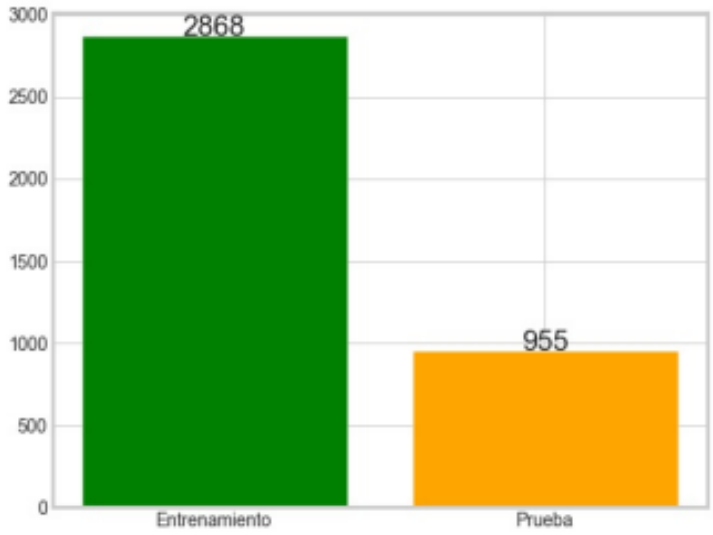
	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Max
0	447078.770000	2.700000	-1.489525	301.600000	299.020000	298.000000	297.400000	4.200000	299.024671	0.522093	...	193.0
1	443969.870000	0.600000	-0.154622	300.600000	297.980000	297.500000	296.600000	4.000000	297.983177	1.146014	...	156.0
2	452585.850000	0.500000	-0.785088	301.800000	300.860000	301.000000	299.800000	2.000000	300.860715	-0.256011	...	208.0
3	450871.940000	4.500000	-1.609320	303.600000	300.280000	299.800000	297.100000	6.500000	300.290506	0.119880	...	229.0
4	447966.000000	1.400000	-1.517590	300.500000	299.320000	299.100000	298.200000	2.300000	299.321232	0.146392	...	223.0
...	...	...	...	...	...	...	...	...	...	...	...	...
3818	454432.051615	1.922313	-1.223207	302.740134	301.470825	302.200000	299.290596	3.449538	301.473731	-0.640366	...	166.0
3819	452336.829017	1.947779	-0.629441	302.305405	300.773163	301.264656	297.718077	4.587328	300.777934	-0.887938	...	235.0
3820	453920.433445	0.377159	-0.056961	303.329983	301.297099	301.988277	297.381527	5.948455	301.303977	-1.202958	...	207.0
3821	450016.451379	0.716976	0.005756	303.291537	300.000739	299.144217	298.647683	4.643854	300.005484	1.323876	...	229.0
3822	452007.497456	0.350024	0.103543	301.518775	300.666378	301.089208	298.483494	3.035281	300.668421	-1.385645	...	215.0

3823 rows × 61 columns

Gráfico del total de datos para entrenamiento y prueba una vez aplicado la ventana de 5

In [62]:
```
División1 = ['Entrenamiento','Prueba']
Datos1 = [2868,955]
colores = ['green','orange']
fig, ax4= plt.subplots(figsize=(8,6))
ax4.bar(División1,Datos1,width=0.8,color=colores )
for index, data in enumerate (Datos1):
    plt.text(x=index, y=data+1, s=f"{data}", fontdict= dict(fontsize=22),ha='center')
plt.tight_layout()
plt.show
```

Out[62]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Selección de características por metodo RFE

Selección de caracteristicas de entrenamiento

Para la selección de características se hará uso de Random Forest

In [63]:
```
X_train= Features_train_fin.copy() # Definiendo variables
y_train= X_train.pop('Falla_maquina')
```

In [64]:
```
model = RandomForestClassifier()
rfe = RFE(estimator= model,n_features_to_select=0.67, step= 1) # Seleccionando el 67% de características más rele
fit = rfe.fit(X_train, y_train)
```

In [65]:
```
selected_rfe_features_train= pd.DataFrame({'Feature':list(X_train.columns),
                                           'Ranking': rfe.ranking_})
selected_rfe_features_train.sort_values(by= 'Ranking') # Mostrando ranking de características
```

Out[65]:

	Feature	Ranking
0	0_Absolute energy	1
30	2_Min	1
31	2_Peak to peak distance	1
32	2_Root mean square	1
33	2_Skewness	1
34	2_Standard deviation	1
35	2_Variance	1
36	3_Absolute energy	1
37	3_Interquartile range	1
39	3_Max	1
40	3_Mean	1
41	3_Median	1
42	3_Min	1
43	3_Peak to peak distance	1
44	3_Root mean square	1
45	3_Skewness	1
46	3_Standard deviation	1
47	3_Variance	1
48	4_Absolute energy	1
51	4_Max	1
52	4_Mean	1
53	4_Median	1
56	4_Root mean square	1
57	4_Skewness	1
28	2_Mean	1
27	2_Max	1
29	2_Median	1
25	2_Interquartile range	1
24	2_Absolute energy	1
23	1_Variance	1
22	1_Standard deviation	1
3	0_Max	1
4	0_Mean	1
19	1_Peak to peak distance	1
13	1_Interquartile range	1
5	0_Median	1
8	0_Root mean square	1
6	0_Min	1
15	1_Max	1
58	4_Standard deviation	2
9	0_Skewness	3
26	2_Kurtosis	4
12	1_Absolute energy	5
18	1_Min	6
16	1_Mean	7
59	4_Variance	8

38	3_Kurtosis	9
7	0_Peak to peak distance	10
11	0_Variance	11
49	4_Interquartile range	12
20	1_Root mean square	13
21	1_Skewness	14
54	4_Min	15
17	1_Median	16
55	4_Peak to peak distance	17
2	0_Kurtosis	18
50	4_Kurtosis	19
10	0_Standard deviation	20
1	0_Interquartile range	21
14	1_Kurtosis	22

Las características que tienen un puntaje de 1 en el Ranking son las seleccionadas para ingresar en el clasificador y se muestran en el dataframe siguiente, el resto de características se eliminan

In [66]:
```
features_filteredT= Features_train_fin.drop(['4_Standard deviation','0_Skewness','2_Kurtosis','1_Absolute energy',
                                             '1_Min','1_Mean','4_Variance','3_Kurtosis','0_Peak to peak distance',
                                             '0_Variance','4_Interquartile range','1_Skewness',
                                             '4_Min','1_Median','4_Peak to peak distance','0_Kurtosis',
                                             '4_Kurtosis','0_Standard deviation','0_Interquartile range',
                                             '1_Kurtosis'],axis= 1)

features_filteredT
```

Out[66]:

	0_Absolute energy	0_Max	0_Mean	0_Median	0_Min	0_Root mean square	1_Interquartile range	1_Max	1_Peak to peak distance	1_Root mean square	...	3_Skew
0	447078.770000	301.600000	299.020000	298.000000	297.400000	299.024671	1.900000	311.100000	3.600000	309.402844	...	0.57
1	443969.870000	300.600000	297.980000	297.500000	296.600000	297.983177	0.700000	309.700000	2.300000	308.301006	...	-0.54
2	452585.850000	301.800000	300.860000	301.000000	299.800000	300.860715	1.500000	312.100000	2.800000	310.481677	...	-0.13
3	450871.940000	303.600000	300.280000	299.800000	297.100000	300.290506	1.600000	312.000000	4.000000	309.843025	...	-0.76
4	447966.000000	300.500000	299.320000	299.100000	298.200000	299.321232	1.300000	311.300000	2.600000	309.901400	...	0.26
...	...	...	...	...	...	...	...	...	...	...	...	
1413	452329.146274	302.627149	300.767075	302.121132	296.729446	300.775380	2.353021	311.561641	4.571196	309.898752	...	0.06
1414	453458.968710	302.269229	301.149279	301.034075	299.550389	301.150782	0.632489	311.430742	2.609552	310.636296	...	0.44
1415	450090.026505	301.140970	300.026874	300.544388	297.380536	300.030007	1.276731	311.551154	3.545323	310.005796	...	-0.22
1416	454332.044830	303.700000	301.437143	300.622443	299.821276	301.440556	1.232353	312.100000	1.880611	311.043290	...	-0.76
1417	454171.266248	302.487025	301.382706	302.196860	298.121028	301.387215	0.850138	311.232728	2.590672	310.445730	...	0.27

2868 rows × 41 columns

Selección de características de prueba

Las características del conjunto de prueba serán las mismas que fueron seleccionadas para el conjunto de entrenamiento

In [67]:
```
features_filteredt= Features_test_fin.drop(['4_Standard deviation','0_Skewness','2_Kurtosis','1_Absolute energy',
                                            '1_Min','1_Mean','4_Variance','3_Kurtosis','0_Peak to peak distance',
                                            '0_Variance','4_Interquartile range','1_Skewness',
                                            '4_Min','1_Median','4_Peak to peak distance','0_Kurtosis',
                                            '4_Kurtosis','0_Standard deviation','0_Interquartile range',
                                            '1_Kurtosis'], axis= 1)

features_filteredt
```

Out[67]:

	0_Absolute energy	0_Max	0_Mean	0_Median	0_Min	0_Root mean square	1_Interquartile range	1_Max	1_Peak to peak distance	1_Root mean square	...	3_Skewn
0	452717.670000	303.300000	300.900000	300.700000	298.200000	300.904526	1.600000	311.200000	3.400000	310.022622	...	-0.883
1	451460.260000	303.700000	300.480000	299.800000	297.900000	300.486359	1.800000	312.800000	3.800000	310.363335	...	-0.620
2	447195.750000	300.700000	299.060000	299.800000	297.200000	299.063789	1.500000	311.700000	3.100000	309.762083	...	0.842

3	447320.910000	301.300000	299.100000	298.300000	296.700000	299.105637	2.800000	310.400000	2.900000	308.882690	...	0.311
4	454647.430000	303.700000	301.540000	300.700000	299.500000	301.545164	1.600000	313.000000	2.900000	311.441879	...	0.249
...	...	...	...	...	...	...	...	...	...	...	...	
469	453016.766208	303.473426	300.997801	301.813411	298.594498	301.003909	1.799417	311.929135	3.228861	310.250996	...	-1.316
470	451417.427591	302.695658	300.467903	300.971568	298.025893	300.472104	0.284163	310.847829	1.760514	310.316189	...	-0.990
471	451960.176882	302.489698	300.647735	301.526778	297.594017	300.652682	0.145687	311.404907	2.669472	310.477479	...	-0.877
472	451079.892280	301.936347	300.356857	301.124139	298.694615	300.359748	0.992919	311.084975	2.230140	310.292546	...	-0.388
473	451930.170505	302.329531	300.638772	300.416437	298.305131	300.642702	1.943570	311.281877	2.281193	310.078487	...	-0.605

955 rows × 41 columns

Definimos variables para realizar la normalización de características

Características originales (train y test)

In [68]:
```
X_traino= Features_train_fin.copy()
y_traino= X_traino.pop('Falla_maquina')
```

In [69]:
```
X_testo= Features_test_fin.copy()
y_testo= X_testo.pop('Falla_maquina')
```

In [70]:
```
X_traino
```

Out[70]:

	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Kurt
0	447078.770000	2.700000	-1.489525	301.600000	299.020000	298.000000	297.400000	4.200000	299.024671	0.522093	...	-1.331
1	443969.870000	0.600000	-0.154622	300.600000	297.980000	297.500000	296.600000	4.000000	297.983177	1.146014	...	-1.473
2	452585.850000	0.500000	-0.785088	301.800000	300.860000	301.000000	299.800000	2.000000	300.860715	-0.256011	...	-1.286
3	450871.940000	4.500000	-1.609320	303.600000	300.280000	299.800000	297.100000	6.500000	300.290506	0.119880	...	-0.926
4	447966.000000	1.400000	-1.517590	300.500000	299.320000	299.100000	298.200000	2.300000	299.321232	0.146392	...	-0.296
...	...	...	...	...	...	...	...	...	...	...	...	
1413	452329.146274	2.467379	-0.684972	302.627149	300.767075	302.121132	296.729446	5.897703	300.775380	-0.949393	...	-0.487
1414	453458.968710	1.026337	-0.916127	302.269229	301.149279	301.034075	299.550389	2.718840	301.150782	-0.501981	...	-1.566
1415	450090.026505	0.865367	-0.090233	301.140970	300.026874	300.544388	297.380536	3.760434	300.030007	-1.255092	...	-1.635
1416	454332.044830	1.939130	-1.331747	303.700000	301.437143	300.622443	299.821276	3.878724	301.440556	0.494651	...	0.235
1417	454171.266248	0.576698	0.140102	302.487025	301.382706	302.196860	298.121028	4.365997	301.387215	-1.422661	...	-1.144

2868 rows × 60 columns

In [71]:
```
X_testo
```

Out[71]:

	0_Absolute energy	0_Interquartile range	0_Kurtosis	0_Max	0_Mean	0_Median	0_Min	0_Peak to peak distance	0_Root mean square	0_Skewness	...	4_Kurto
0	452717.670000	0.900000	-0.666767	303.300000	300.900000	300.700000	298.200000	5.100000	300.904526	-0.246094	...	-0.9053
1	451460.260000	1.800000	-0.900100	303.700000	300.480000	299.800000	297.900000	5.800000	300.486359	0.428214	...	-1.1784
2	447195.750000	3.000000	-1.774981	300.700000	299.060000	299.800000	297.200000	3.500000	299.063789	-0.302690	...	-1.7167
3	447320.910000	3.200000	-1.629334	301.300000	299.100000	298.300000	296.700000	4.600000	299.105637	0.137015	...	-1.0466
4	454647.430000	3.400000	-1.745976	303.700000	301.540000	300.700000	299.500000	4.200000	301.545164	0.266768	...	-0.0503
...	...	...	...	...	...	...	...	...	...	...	...	
469	453016.766208	3.292327	-1.630935	303.473426	300.997801	301.813411	298.594498	4.878928	301.003909	-0.157720	...	-1.5848
470	451417.427591	1.674444	-1.073011	302.695658	300.467903	300.971568	298.025893	4.669765	300.472104	-0.199092	...	-1.0435
471	451960.176882	1.630064	-0.736271	302.489698	300.647735	301.526778	297.594017	4.895681	300.652682	-0.813679	...	-1.0897
472	451079.892280	2.282361	-1.710505	301.936347	300.356857	301.124139	298.694615	3.241732	300.359748	-0.258158	...	0.1185
473	451930.170505	2.484441	-1.347572	302.329531	300.638772	300.416437	298.305131	4.024400	300.642702	-0.204786	...	-1.2178

955 rows × 60 columns

Características seleccionadas (train y test)

In [72]:
```
X_trains= features_filteredT.copy()
y_trains= X_trains.pop('Falla_maquina')
```

In [73]:
```
X_tests= features_filteredt.copy()
y_tests= X_tests.pop('Falla_maquina')
```

In [74]:
```
X_trains
```

Out[74]:

	0_Absolute energy	0_Max	0_Mean	0_Median	0_Min	0_Root mean square	1_Interquartile range	1_Max	1_Peak to peak distance	1_Root mean square	...	3_Ro mea squa
0	447078.770000	301.600000	299.020000	298.000000	297.400000	299.024671	1.900000	311.100000	3.600000	309.402844	...	40.6518
1	443969.870000	300.600000	297.980000	297.500000	296.600000	297.983177	0.700000	309.700000	2.300000	308.301006	...	36.0193
2	452585.850000	301.800000	300.860000	301.000000	299.800000	300.860715	1.500000	312.100000	2.800000	310.481677	...	41.1439
3	450871.940000	303.600000	300.280000	299.800000	297.100000	300.290506	1.600000	312.000000	4.000000	309.843025	...	33.4938
4	447966.000000	300.500000	299.320000	299.100000	298.200000	299.321232	1.300000	311.300000	2.600000	309.901400	...	47.7043
...	...	...	...	...	...	...	...	...	...	...	...	
1413	452329.146274	302.627149	300.767075	302.121132	296.729446	300.775380	2.353021	311.561641	4.571196	309.898752	...	63.8612
1414	453458.968710	302.269229	301.149279	301.034075	299.550389	301.150782	0.632489	311.430742	2.609552	310.636296	...	59.9099
1415	450090.026505	301.140970	300.026874	300.544388	297.380536	300.030007	1.276731	311.551154	3.545323	310.005796	...	52.2779
1416	454332.044830	303.700000	301.437143	300.622443	299.821276	301.440556	1.232353	312.100000	1.880611	311.043290	...	55.4208
1417	454171.266248	302.487025	301.382706	302.196860	298.121028	301.387215	0.850138	311.232728	2.590672	310.445730	...	53.8514

2868 rows × 40 columns

In [75]:
```
X_tests
```

Out[75]:

	0_Absolute energy	0_Max	0_Mean	0_Median	0_Min	0_Root mean square	1_Interquartile range	1_Max	1_Peak to peak distance	1_Root mean square	...	3_Roo mea squar
0	452717.670000	303.300000	300.900000	300.700000	298.200000	300.904526	1.600000	311.200000	3.400000	310.022622	...	37.74201
1	451460.260000	303.700000	300.480000	299.800000	297.900000	300.486359	1.800000	312.800000	3.800000	310.363335	...	41.27767
2	447195.750000	300.700000	299.060000	299.800000	297.200000	299.063789	1.500000	311.700000	3.100000	309.762083	...	41.52536
3	447320.910000	301.300000	299.100000	298.300000	296.700000	299.105637	2.800000	310.400000	2.900000	308.882690	...	46.96905
4	454647.430000	303.700000	301.540000	300.700000	299.500000	301.545164	1.600000	313.000000	2.900000	311.441879	...	44.29320
...	...	...	...	...	...	...	...	...	...	...	...	
469	453016.766208	303.473426	300.997801	301.813411	298.594498	301.003909	1.799417	311.929135	3.228861	310.250996	...	50.51472
470	451417.427591	302.695658	300.467903	300.971568	298.025893	300.472104	0.284163	310.847829	1.760514	310.316189	...	57.26492
471	451960.176882	302.489698	300.647735	301.526778	297.594017	300.652682	0.145687	311.404907	2.669472	310.477479	...	58.46444
472	451079.892280	301.936347	300.356857	301.124139	298.694615	300.359748	0.992919	311.084975	2.230140	310.292546	...	51.28000
473	451930.170505	302.329531	300.638772	300.416437	298.305131	300.642702	1.943570	311.281877	2.281193	310.078487	...	64.36004

955 rows × 40 columns

Escalado y normalización de características seleccionadas

Se utiliza la función RobustScaler ya que esta ayuda a reducir el impacto de valores atípicos

In [76]:
```
scaler = RobustScaler()
scaler.fit(X_trains)
features_train_norm = scaler.transform(X_trains)
features_test_norm = scaler.transform(X_tests)
```

Escalado y normalización de características originales

In [77]:
```
scaler = RobustScaler()
scaler.fit(X_traino)
features_train_normo = scaler.transform(X_traino)
features_test_normo = scaler.transform(X_testo)
```

Entrenamiento del modelo

Ajuste de hiperparámetros

Creamos un bosque aleatorio para examinar los hiperparámetros

In [78]:
```
rf = RandomForestClassifier()
print('Parámetros actualmente en uso:\n')
pprint(rf.get_params())
```

```
Parámetros actualmente en uso:

{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

Ajustamos los hiperparámetros: bootstrap, criterion, max_depth, max_features, min_samples_leaf, min_samples_split, n_estimators

Creamos cuadrícula de hiperparametros

In [79]:
```
# Número de árboles en el bosque aleatorio
n_estimators = [int(x) for x in np.linspace(start = 150, stop = 550, num = 5)]
# Criterion
criterion= ['gini','entropy']
# Número de características a considerar en cada división
max_features = ['auto', 'sqrt']
# Número máximo de niveles en el árbol
max_depth = [int(x) for x in np.linspace(10, 50, num = 5)]
max_depth.append(None)
# Número mínimo de muestras necesarias para dividir un nodo
min_samples_split = [2, 3, 6]
# Número mínimo de muestras requeridas en cada nodo hoja
min_samples_leaf = [1, 2, 4]
# Método de selección de muestras para entrenar cada árbol
bootstrap = [True, False]

# Creación de cuadrícula aleatoria
random_grid = {'n_estimators': n_estimators,
               'criterion': criterion,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}

pprint(random_grid)
```

```
{'bootstrap': [True, False],
 'criterion': ['gini', 'entropy'],
 'max_depth': [10, 20, 30, 40, 50, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 3, 6],
 'n_estimators': [150, 250, 350, 450, 550]}
```

Entrenamiento de busqueda aleatoria

In [80]:
```
# Use la cuadrícula aleatoria para buscar los mejores hiperparámetros
rf = RandomForestClassifier()
# Búsqueda aleatoria de hiperparámetros, usando 3 veces la validación cruzada,
rf_random = RandomizedSearchCV(estimator=rf, param_distributions=random_grid,
                               n_iter = 100, scoring='f1_weighted',
                               cv = 3, verbose=2, random_state=42, n_jobs=-1,
                               return_train_score=True)

# Ajustar el modelo de búsqueda aleatoria
rf_random.fit(features_train_norm, y_trains);
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
```

Mejores hiperparámetros

In [81]:
```
rf_random.best_params_
```

Out[81]:
```
{'n_estimators': 250,
 'min_samples_split': 3,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 20,
 'criterion': 'entropy',
 'bootstrap': False}
```

Mejor modelo entrenado

In [82]:
```
rf_best= rf_random.best_estimator_
rf_best
```

Out[82]:
```
RandomForestClassifier(bootstrap=False, criterion='entropy', max_depth=20,
                       min_samples_split=3, n_estimators=250)
```

Evaluación del modelo

Utilizando características seleccionadas e hiperparámetros optimizados

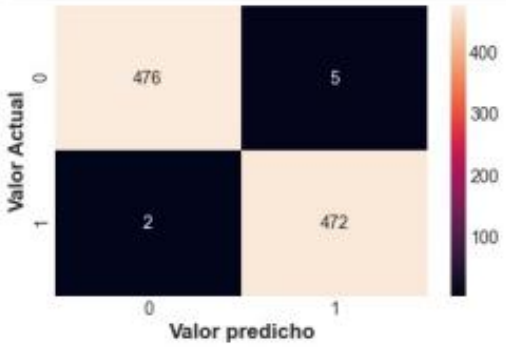
In [83]:
```
rf_best.fit(features_train_norm, y_trains)
# Predicción en el conjunto de prueba
y_predict = rf_best.predict(features_test_norm)
accuracy = accuracy_score(y_tests, y_predict) * 100
precision= precision_score(y_tests, y_predict)*100
print(classification_report(y_tests, y_predict))
print("Accuracy: " + str(accuracy) + '%')
print("Precision: " + str(precision) + '%')
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       481
           1       0.99      1.00      0.99       474

    accuracy                           0.99       955
   macro avg       0.99      0.99      0.99       955
weighted avg       0.99      0.99      0.99       955

Accuracy: 99.26701570680628%
Precision: 98.9517819706499%
```

Matriz de confusión

In [84]:
```
cm = confusion_matrix(y_tests,y_predict)
sns.heatmap(cm, annot=True, fmt='d')
plt.ylabel('Valor Actual', fontweight = 'bold', fontsize = 17)
```

```
plt.xlabel('Valor predicho', fontweight = 'bold', fontsize = 17)
plt.show()
```



Utilizando características originales e hiperparámetros optimizados

In [85]:
```
classifier =  RandomForestClassifier(bootstrap=False, criterion='entropy', max_depth=20,
                      min_samples_split=3, n_estimators=250)

classifier.fit(features_train_normo, y_traino)
# Prediccion en el conjunto de prueba
y_predict1 = classifier.predict(features_test_normo)
accuracy = accuracy_score(y_testo, y_predict1) * 100
precision= precision_score(y_testo, y_predict1)*100
print(classification_report(y_testo, y_predict1))
print("Accuracy: " + str(accuracy) + '%')
print("Precision: " + str(precision) + '%')
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       481
           1       0.99      1.00      0.99       474

    accuracy                           0.99       955
   macro avg       0.99      0.99      0.99       955
weighted avg       0.99      0.99      0.99       955

Accuracy: 99.16230366492147%
Precision: 98.74476987447699%
```

Matriz de confusión

In [86]:
```
cm1 = confusion_matrix(y_testo,y_predict1)
sns.heatmap(cm1, annot=True, fmt='d')
plt.ylabel('Valor Actual', fontweight = 'bold', fontsize = 17)
plt.xlabel('Valor predicho', fontweight = 'bold', fontsize = 17)
plt.show()
```

Utilizando características seleccionadas e hiperparámetros predeterminados

In [87]:
```
classifier1 =  RandomForestClassifier()
```

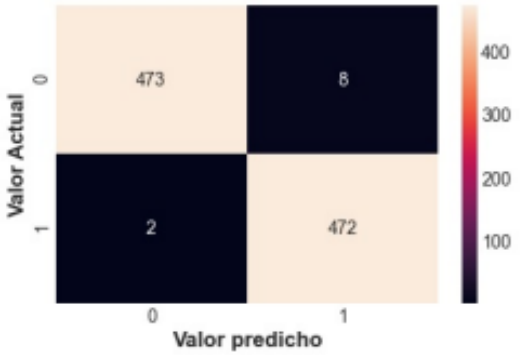
```
classifier1.fit(features_train_norm, y_trains)
# Predicción en el conjunto de prueba
y_predict2 = classifier1.predict(features_test_norm)
accuracy = accuracy_score(y_tests, y_predict2) * 100
precision= precision_score(y_tests, y_predict2)*100
print(classification_report(y_tests, y_predict2))
print("Accuracy: " + str(accuracy) + '%')
print("Precision: " + str(precision) + '%')
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       481
           1       0.98      1.00      0.99       474

    accuracy                           0.99       955
   macro avg       0.99      0.99      0.99       955
weighted avg       0.99      0.99      0.99       955

Accuracy: 98.95287958115183%
Precision: 98.33333333333333%
```

Matriz de confusión

In [88]:

```
cm2 = confusion_matrix(y_tests,y_predict2)
sns.heatmap(cm2, annot=True, fmt='d')
plt.ylabel('Valor Actual', fontweight = 'bold', fontsize = 17)
plt.xlabel('Valor predicho', fontweight = 'bold', fontsize = 17)
plt.show()
```



Utilizando características originales e hiperparámetros predeterminados

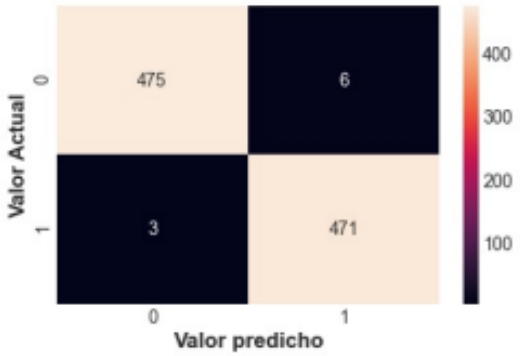
In [89]:

```
classifier2 =  RandomForestClassifier()
classifier2.fit(features_train_normo, y_traino)
# Predicción en el conjunto de prueba
y_predict3 = classifier2.predict(features_test_normo)
accuracy = accuracy_score(y_testo, y_predict3) * 100
precision= precision_score(y_testo, y_predict3)*100
print(classification_report(y_testo, y_predict3))
print("Accuracy: " + str(accuracy) + '%')
print("Precision: " + str(precision) + '%')
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       481
           1       0.99      0.99      0.99       474

    accuracy                           0.99       955
   macro avg       0.99      0.99      0.99       955
weighted avg       0.99      0.99      0.99       955

Accuracy: 99.05759162303664%
Precision: 98.74213836477988%
```

In [90]:

```
cm3 = confusion_matrix(y_testo,y_predict3)
sns.heatmap(cm3, annot=True, fmt='d')
plt.ylabel('Valor Actual', fontweight = 'bold', fontsize = 17)
plt.xlabel('Valor predicho', fontweight = 'bold', fontsize = 17)
plt.show()
```



Valores por celda de cada matriz de confusión y cálculo de métricas

Características seleccionadas e hiperparámetros optimizados

In [91]:
```
TN, FP, FN, TP= cm.ravel()
display(
TN,
FP,
FN,
TP,
)
```

```
476
5
2
472
```

In [92]:
```
Exactitud= ((TP+TN)/(TP+FP+FN+TN))*100
print(Exactitud)
```

```
99.26701570680628
```

In [93]:
```
Precisión= (TP/(TP+FP))*100
print(Precisión)
```

```
98.9517819706499
```

In [94]:
```
Sensibilidad= (TP/(TP+FN))*100
print(Sensibilidad)
```

```
99.57805907172997
```

In [95]:
```
Especificidad= (TN/(TN+FP))*100
print(Especificidad)
```

```
98.96049896049897
```

In [97]:
```
PuntajeF1= (2*TP/(2*TP+FP+FN))*100
print(PuntajeF1)
```

```
99.26393270241851
```

Características originales e hiperparámetros optimizados

In [98]:
```
TN1, FP1, FN1, TP1= cm1.ravel()
display(
TN1,
FP1,
FN1,
TP1,
)
```

```
475

6

2

472
```

In [99]:
```
Exactitud= ((TP1+TN1)/(TP1+FP1+FN1+TN1))*100
print(Exactitud)
```

```
99.16230366492147
```

In [100…
```
Precisión= (TP1/(TP1+FP1))*100
print(Precisión)
```

```
98.74476987447699
```

In [101…
```
Sensibilidad= (TP1/(TP1+FN1))*100
print(Sensibilidad)
```

```
99.57805907172997
```

In [102…
```
Especificidad= (TN1/(TN1+FP1))*100
print(Especificidad)
```

```
98.75259875259876
```

In [103…
```
PuntajeF1= (2*TP1/(2*TP1+FP1+FN1))*100
print(PuntajeF1)
```

```
99.15966386554622
```

Características seleccionadas e hiperparámetros predeterminados

In [104…
```
TN2, FP2, FN2, TP2= cm2.ravel()
display(
TN2,
FP2,
FN2,
TP2,
)
```

```
473
8
2

472
```

In [105…
```
Exactitud= ((TP2+TN2)/(TP2+FP2+FN2+TN2))*100
print(Exactitud)
```
```
98.95287958115183
```

In [106…
```
Precisión= (TP2/(TP2+FP2))*100
print(Precisión)
```
```
98.33333333333333
```

In [107…
```
Sensibilidad= (TP2/(TP2+FN2))*100
print(Sensibilidad)
```
```
99.57805907172997
```

In [108…
```
Especificidad= (TN2/(TN2+FP2))*100
print(Especificidad)
```
```
98.33679833679834
```

In [109…
```
PuntajeF1= (2*TP2/(2*TP2+FP2+FN2))*100
print(PuntajeF1)
```
```
98.9517819706499
```

Características originales e hiperparámetros predeterminados

In [110…
```
TN3, FP3, FN3, TP3= cm3.ravel()
display(
TN3,
FP3,
FN3,
TP3,
)
```
```
475
6
3
471
```

In [111…
```
Exactitud= ((TP3+TN3)/(TP3+FP3+FN3+TN3))*100
print(Exactitud)
```
```
99.05759162303664
```

In [112…
```
Precisión= (TP3/(TP3+FP3))*100
print(Precisión)
```
```
98.74213836477988
```

In [113…
```
Sensibilidad= (TP3/(TP3+FN3))*100
print(Sensibilidad)
```

```
99.36708860759494
```

In [114…
```
Especificidad= (TN3/(TN3+FP3))*100
print(Especificidad)
```

```
98.75259875259876
```

In [115…
```
PuntajeF1= (2*TP3/(2*TP3+FP3+FN3))*100
print(PuntajeF1)
```

```
99.05362776025235
```

Análisis post hoc

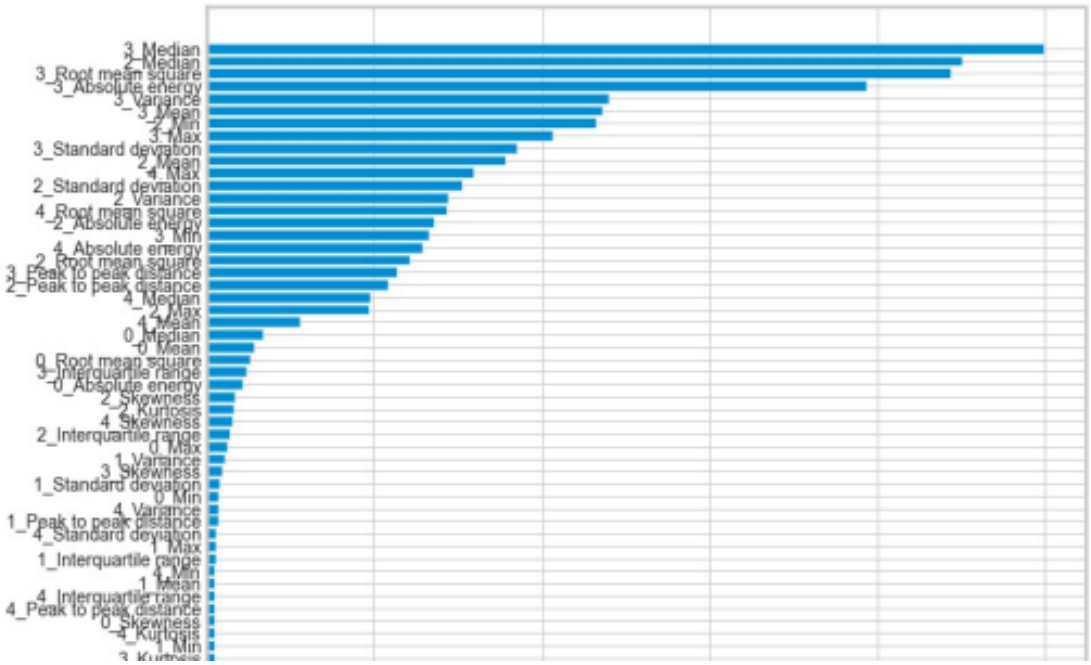
Características que más contribuyeron al modelo (características originales)

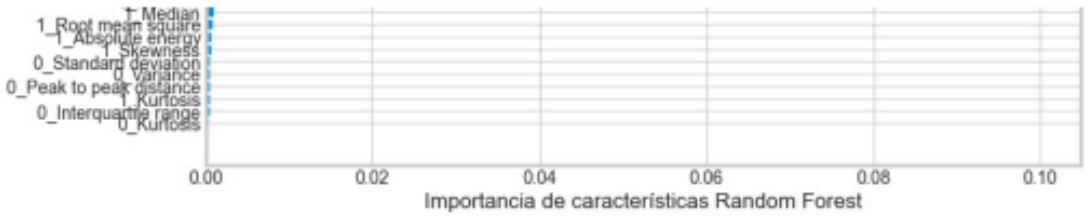
In [116…
```
classifier.feature_importances_
```

Out[116…
```
array([0.00427606, 0.00048561, 0.00042367, 0.00248483, 0.0057674 ,
       0.00681707, 0.00148266, 0.00051267, 0.00520694, 0.00095231,
       0.00057987, 0.00054308, 0.00063421, 0.00110288, 0.00051224,
       0.00112452, 0.00101405, 0.00092594, 0.0009381 , 0.00137924,
       0.00075959, 0.0006052 , 0.00161516, 0.00217067, 0.02711533,
       0.00284918, 0.00325675, 0.01944348, 0.03560347, 0.09011956,
       0.04646152, 0.02173211, 0.02425816, 0.00339705, 0.03052746,
       0.0287762 , 0.07877073, 0.00479761, 0.00093129, 0.0412347 ,
       0.04715858, 0.09980957, 0.02660477, 0.02279856, 0.0886848 ,
       0.00182266, 0.03704375, 0.04804372, 0.02573442, 0.00098703,
       0.0009482 , 0.03181157, 0.0111254 , 0.01952378, 0.00102754,
       0.0009846 , 0.02866411, 0.00304459, 0.00113432, 0.00145947])
```

In [117…
```
sorted_idx = classifier.feature_importances_.argsort()
fig= plt.figure(figsize= (11,11))
ax1 = plt.subplot(111)
ax1.barh(X_testo.columns[sorted_idx], classifier.feature_importances_[sorted_idx])
plt.xlabel("Importancia de características Random Forest")
```

Out[117…
```
Text(0.5, 0, 'Importancia de características Random Forest')
```





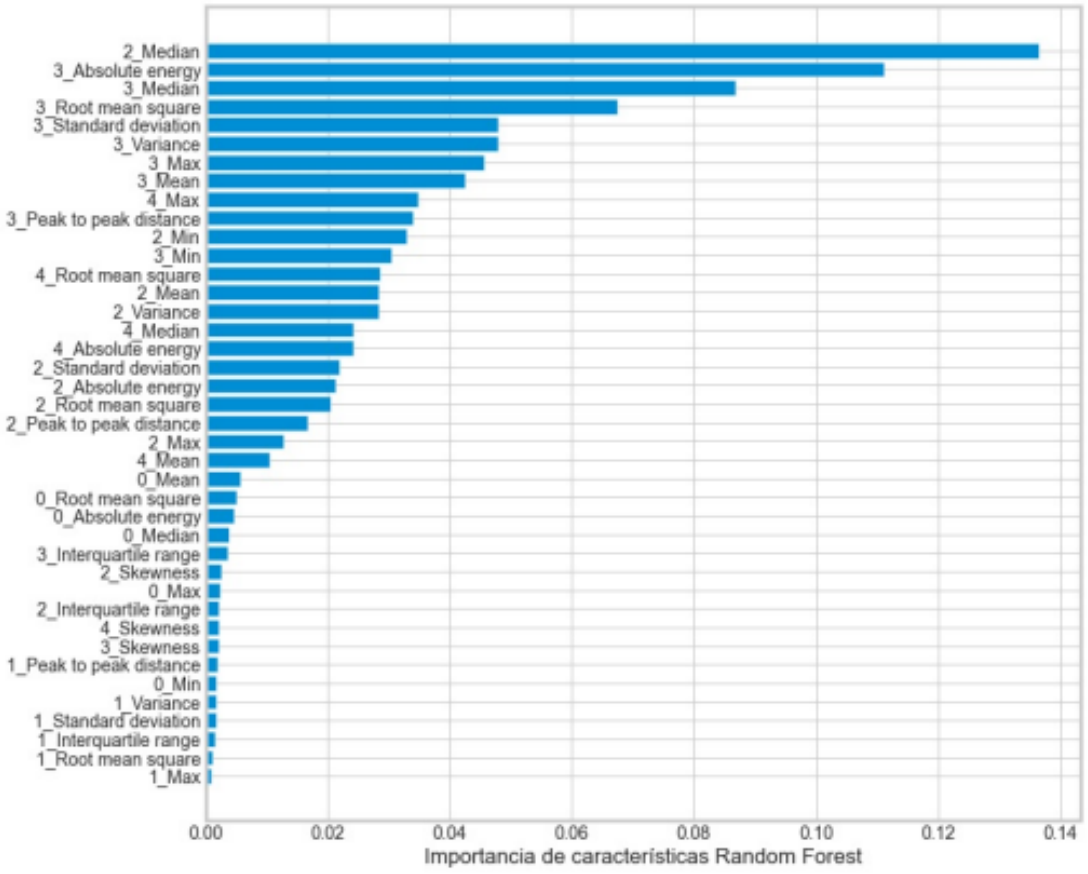
Características que más contribuyeron al modelo (características seleccionadas)

In [118…
```
rf_best.feature_importances_
```

Out[118…
```
array([0.00467033, 0.00240832, 0.00561467, 0.0037778 , 0.00177346,
       0.00517257, 0.00149975, 0.00101097, 0.00188359, 0.00106851,
       0.0017199 , 0.00173432, 0.02135104, 0.00221521, 0.01275326,
       0.02839481, 0.1364187 , 0.03302793, 0.0166868 , 0.02044142,
       0.00253944, 0.02187858, 0.02832197, 0.11101614, 0.00354183,
       0.04551258, 0.04247473, 0.08687168, 0.030393  , 0.0339665 ,
       0.06742332, 0.00211902, 0.04798144, 0.04786886, 0.0242821 ,
       0.03473887, 0.01039781, 0.02429468, 0.02856585, 0.00218822])
```

In [119…
```
sorted_idx = rf_best.feature_importances_.argsort()
fig= plt.figure(figsize= (11,11))
ax1 = plt.subplot(111)
ax1.barh(X_tests.columns[sorted_idx], rf_best.feature_importances_[sorted_idx])
plt.xlabel("Importancia de características Random Forest")
```

Out[119…
```
Text(0.5, 0, 'Importancia de características Random Forest')
```



Significancia estadística

Se define el número de permutaciones a realizar

In [120…
```
n_permutations = 700 # Definiendo número de permutaciones
sensibilidad_perm= np.zeros((n_permutations, 1)) # Creamos objeto vacío para completar luego de la permutación
```

In [121…
```
X= Features_fin.copy() # Definiendo variables del archivo Features_fin
y= X.pop('Falla_maquina')
```

In [122…
```
y = y.values.astype('int') # Transformación de características y variable objetivo
X = X.values.astype('float32')
```

In [123…
```
for i_perm in range(n_permutations): # Configuración de blucle for


    np.random.seed(i_perm) # Configurando semilla aleatoria
    y_permuted = np.random.permutation(y) # Mezcla (barajamiento) de la variable objetivo


    n_folds = 4 # Definiendo validación cruzada
    skf = StratifiedKFold(n_splits=n_folds, shuffle= True)


    sensibilidad_cv = np.zeros((n_folds, 1)) # Creamos objeto vacío para guardar la sensibilidad de validación cr


    for i_fold, (train, test) in enumerate(skf.split(X, y_permuted)): # Definimos bucle for para iterar sobre los
        X_trainp, X_testp = X[train], X[test] # Definiendo archivos de train y test
        y_trainp, y_testp = y_permuted[train], y_permuted[test]

        scaler = RobustScaler() # Escala/normalización
        scaler.fit(X_trainp)
        features_train_norml = scaler.fit_transform(X_trainp)
        features_test_norml = scaler.transform(X_testp)


        Rf =  RandomForestClassifier() # Definiendo algoritmo de Random Forest (hiperparámetros predeterminados)

        Rf.fit(features_train_norml, y_trainp) # Entrenamiento

        y_predicted = Rf.predict(features_test_norml) # Predicción

        cmp = confusion_matrix(y_testp, y_predicted) # Matriz de confusión

        tn, fp, fn, tp = cmp.ravel() # valores por cada celda de la matriz de confusión


        sensibilidad_test = tp / (tp + fn) # Cálculo de la sensibilidad


        sensibilidad_cv[i_fold, :] = sensibilidad_test


    # Estimamos la sensibilidad para los 4 cv y lo guardamos en el directorio permutacion_dir
    np.save(permutacion_dir / ('perm_test_sensibilidad_%03d.npy' % i_perm), sensibilidad_cv.mean())


    sensibilidad_perm[i_perm, :] = sensibilidad_cv.mean()
```

In [124…
```
sensibilidad_from_model = sensibilidad_cv.mean() # Sensibilidad original
```

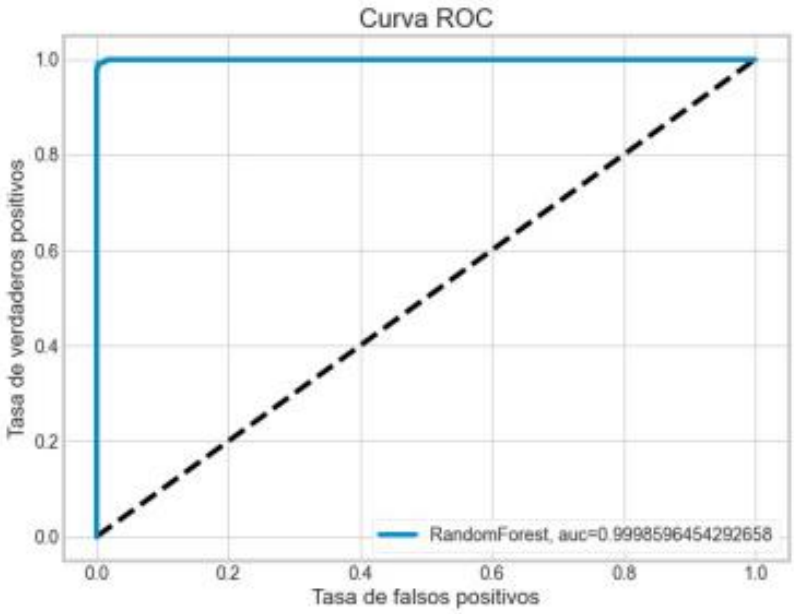
In [125…
```
sensibilidad_p_value = (np.sum(sensibilidad_perm >= sensibilidad_from_model) + 1) / (n_permutations + 1) # Cálcul

print('sensibilidad: p-value = %.3f' % sensibilidad_p_value)
```

```
sensibilidad: p-value = 0.001
```

Curva ROC

In [126…
```
rf_best.fit(features_train_norm, y_trains)
y_pred_proba = rf_best.predict_proba(features_test_norm)[::,1]
fpr, tpr, thresholds = metrics.roc_curve(y_tests,  y_pred_proba)
auc = metrics.roc_auc_score(y_tests, y_pred_proba)

plt.figure(figsize=(9,7))
plt.plot([0,1],[0,1], 'k--')
```

```
plt.plot(fpr, tpr, label= "RandomForest, auc="+str(auc))
plt.legend()
plt.xlabel("Tasa de falsos positivos")
plt.ylabel("Tasa de verdaderos positivos")
plt.title('Curva ROC')
plt.show()
```



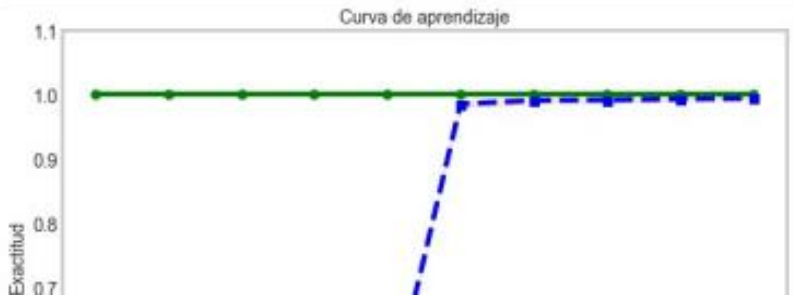
Curva de aprendizaje

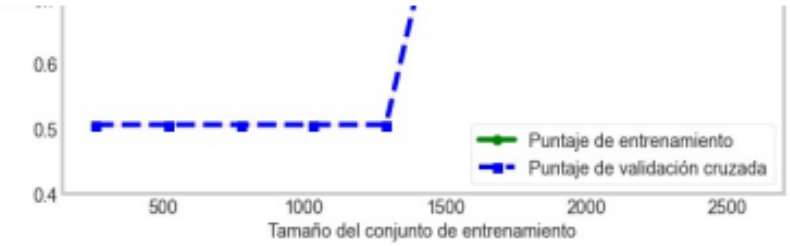
In [127…

```
clf1= RandomForestClassifier(bootstrap=False, criterion='entropy', max_depth=20,
                             min_samples_split=3, n_estimators=250)
train_sizes, train_scores, test_scores = learning_curve(estimator= clf1, X= features_train_norm, y= y_trains,
                                                        cv=10, scoring='accuracy',
                                                        n_jobs=-1, train_sizes=np.linspace(0.1, 1.0, 10))
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)
```

In [128…

```
# graficando las curvas
plt.figure(figsize=(9,6))
plt.plot(train_sizes, train_mean, color='g', marker='o', markersize=7,
         label='Puntaje de entrenamiento')
plt.fill_between(train_sizes, train_mean + train_std,
                 train_mean - train_std, alpha=0.1, color='b')
plt.plot(train_sizes, test_mean, color='b', linestyle='--',
         marker='s', markersize=7, label='Puntaje de validación cruzada')
plt.fill_between(train_sizes, test_mean + test_std,
                 test_mean - test_std, alpha=0.1, color='r')
plt.gca().set_ylim([0.4, 1.1])
plt.grid()
plt.title('Curva de aprendizaje',fontsize=14)
plt.legend(loc='lower right')
plt.xlabel('Tamaño del conjunto de entrenamiento',fontsize=14)

plt.ylabel('Exactitud',fontsize=14)
plt.show()
```





Comparación con otros métodos de machine learning

Para cada método de machine learning se hará uso de los hiperparámetros predeterminados, a su vez se empleará el conjunto seleccionado de características y solo se tomará en cuenta las métricas de evaluación de accuracy ,precisión y recall (sensibilidad)

SVM

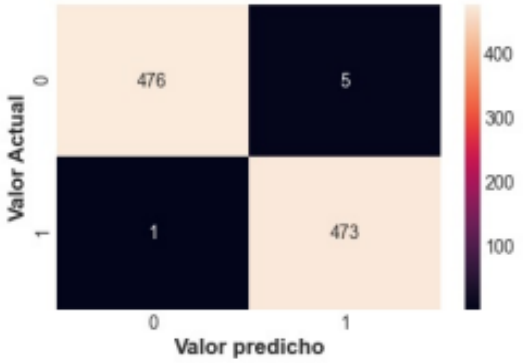
In [129…

```
svclassifier = SVC()
svclassifier.fit(features_train_norm, y_trains) # Entrenamiento SVM
y_predict4 = svclassifier.predict(features_test_norm) # Predicción
accuracy = accuracy_score(y_tests, y_predict4) * 100 # Métricas
precision= precision_score(y_tests, y_predict4)*100
sensibilidad= metrics.recall_score(y_tests, y_predict4)*100
print(classification_report(y_tests, y_predict4)) # Reporte de clasificación
print("Exactitud: " + str(accuracy) + '%')
print("Precision: " + str(precision) + '%')
print("Sensibilidad: " + str(sensibilidad) + '%')
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       481
           1       0.99      1.00      0.99       474

    accuracy                           0.99       955
   macro avg       0.99      0.99      0.99       955
weighted avg       0.99      0.99      0.99       955

Exactitud: 99.3717277486911%
Precision: 98.9539748953975%
Sensibilidad: 99.78902953586498%
```

In [130…

```
cm4 = confusion_matrix(y_tests,y_predict4) # Matriz de confusión SVM
sns.heatmap(cm4, annot=True, fmt='d')
plt.ylabel('Valor Actual', fontweight = 'bold', fontsize = 17)
plt.xlabel('Valor predicho', fontweight = 'bold', fontsize = 17)
plt.show()
```



Gradient boosting

In [131…

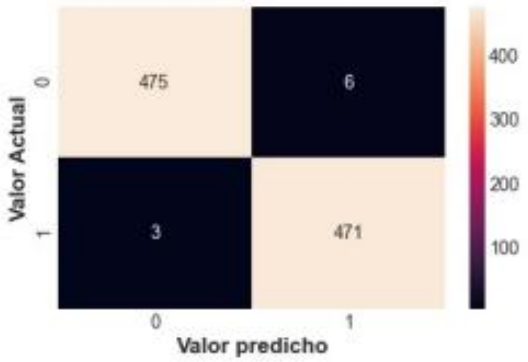
```
gb= GradientBoostingClassifier()
gb.fit(features_train_norm, y_trains) # Entrenamiento Gradient boosting
y_predict5 = gb.predict(features_test_norm) # Predicción
accuracy = accuracy_score(y_tests, y_predict5) * 100 # Métricas
precision= precision_score(y_tests, y_predict5)*100
```

```
sensibilidad= metrics.recall_score(y_tests, y_predict5)*100
print(classification_report(y_tests, y_predict5)) # Reporte de clasificación
print("Exactitud: " + str(accuracy) + '%')
print("Precision: " + str(precision) + '%')
print("Sensibilidad: " + str(sensibilidad) + '%')
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       481
           1       0.99      0.99      0.99       474

    accuracy                           0.99       955
   macro avg       0.99      0.99      0.99       955
weighted avg       0.99      0.99      0.99       955

Exactitud: 99.05759162303664%
Precision: 98.74213836477988%
Sensibilidad: 99.36708860759494%
```

In [132…

```
cm5 = confusion_matrix(y_tests,y_predict5) # Matriz de confusión Gradient boosting
sns.heatmap(cm5, annot=True, fmt='d')
plt.ylabel('Valor Actual', fontweight = 'bold', fontsize = 17)
plt.xlabel('Valor predicho', fontweight = 'bold', fontsize = 17)
plt.show()
```



XGBoost

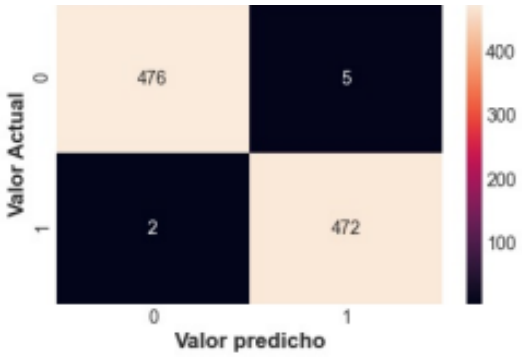
In [133…

```
XGBC = XGBClassifier()
XGBC.fit(features_train_norm, y_trains) # Entrenamiento XGBoost
y_predict6 = XGBC.predict(features_test_norm) # Predicción
accuracy = accuracy_score(y_tests, y_predict6) * 100 # Métricas
precision= precision_score(y_tests, y_predict6)*100
sensibilidad= metrics.recall_score(y_tests, y_predict6)*100
print(classification_report(y_tests, y_predict6)) # Reporte de clasificación
print("Exactitud: " + str(accuracy) + '%')
print("Precision: " + str(precision) + '%')
print("Sensibilidad: " + str(sensibilidad) + '%')
```

```
[12:59:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in
XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' t
o 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       481
           1       0.99      1.00      0.99       474

    accuracy                           0.99       955
   macro avg       0.99      0.99      0.99       955
weighted avg       0.99      0.99      0.99       955

Exactitud: 99.26701570680628%
Precision: 98.9517819706499%
Sensibilidad: 99.57805907172997%
```

In [134…

```
cm6 = confusion_matrix(y_tests,y_predict6) # Matriz de confusión XGBoost
sns.heatmap(cm6, annot=True, fmt='d')
plt.ylabel('Valor Actual', fontweight = 'bold', fontsize = 17)
plt.xlabel('Valor predicho', fontweight = 'bold', fontsize = 17)
plt.show()
```



KNN

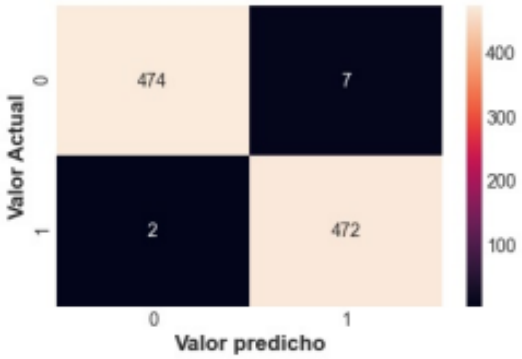
In [135…

```
KNN = KNeighborsClassifier()
KNN.fit(features_train_norm, y_trains) # Entrenamiento KNN
y_predict7 = KNN.predict(features_test_norm) # Predicción
accuracy = accuracy_score(y_tests, y_predict7) * 100 # Métricas
precision= precision_score(y_tests, y_predict7)*100
sensibilidad= metrics.recall_score(y_tests, y_predict7)*100
print(classification_report(y_tests, y_predict7)) # Reporte de clasificación
print("Exactitud: " + str(accuracy) + '%')
print("Precision: " + str(precision) + '%')
print("Sensibilidad: " + str(sensibilidad) + '%')
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       481
           1       0.99      1.00      0.99       474

    accuracy                           0.99       955
   macro avg       0.99      0.99      0.99       955
weighted avg       0.99      0.99      0.99       955

Exactitud: 99.05759162303664%
Precision: 98.53862212943632%
Sensibilidad: 99.57805907172997%
```

In [136…

```
cm7 = confusion_matrix(y_tests,y_predict7) # Matriz de confusión KNN
sns.heatmap(cm7, annot=True, fmt='d')
plt.ylabel('Valor Actual', fontweight = 'bold', fontsize = 17)
plt.xlabel('Valor predicho', fontweight = 'bold', fontsize = 17)
plt.show()
```



Naive Bayes

In [137…

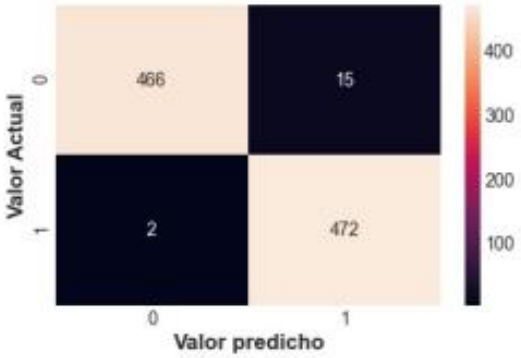
```
GNB= GaussianNB()
GNB.fit(features_train_norm, y_trains) # Entrenamiento Naive Bayes
y_predict8 = GNB.predict(features_test_norm) # Predicción
accuracy = accuracy_score(y_tests, y_predict8) * 100 # Métricas
precision= precision_score(y_tests, y_predict8)*100
sensibilidad= metrics.recall_score(y_tests, y_predict8)*100
```

```
print(classification_report(y_tests, y_predict8)) # Reporte de clasificacion
print("Exactitud: " + str(accuracy) + '%')
print("Precision: " + str(precision) + '%')
print("Sensibilidad: " + str(sensibilidad) + '%')
```

```
              precision    recall  f1-score   support

           0       1.00      0.97      0.98       481
           1       0.97      1.00      0.98       474

    accuracy                           0.98       955
   macro avg       0.98      0.98      0.98       955
weighted avg       0.98      0.98      0.98       955

Exactitud: 98.219895287958115%
Precision: 96.91991786447639%
Sensibilidad: 99.57805907172997%
```

In [138...

```
cm8 = confusion_matrix(y_tests,y_predict8) # Matriz de confusión Naive Bayes
sns.heatmap(cm8, annot=True, fmt='d')
plt.ylabel('Valor Actual', fontweight = 'bold', fontsize = 17)
plt.xlabel('Valor predicho', fontweight = 'bold', fontsize = 17)
plt.show()
```



Arbol de decisión

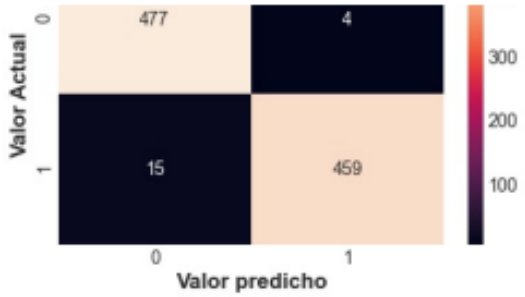
In [139...

```
DTC= DecisionTreeClassifier()
DTC.fit(features_train_norm, y_trains) # Entrenamiento Árbol de decisión
y_predict9 = DTC.predict(features_test_norm) # Predicción
accuracy = accuracy_score(y_tests, y_predict9) * 100 # Métricas
precision= precision_score(y_tests, y_predict9)*100
sensibilidad= metrics.recall_score(y_tests, y_predict9)*100
print(classification_report(y_tests, y_predict9)) # Reporte de clasificación
print("Exactitud: " + str(accuracy) + '%')
print("Precision: " + str(precision) + '%')
print("Sensibilidad: " + str(sensibilidad) + '%')
```

```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       481
           1       0.99      0.97      0.98       474

    accuracy                           0.98       955
   macro avg       0.98      0.98      0.98       955
weighted avg       0.98      0.98      0.98       955

Exactitud: 98.01047120418848%
Precision: 99.13606911447084%
Sensibilidad: 96.83544303797468%
```

In [140...

```
cm9 = confusion_matrix(y_tests,y_predict9) # Matriz de confusión árbol de decisión
sns.heatmap(cm9, annot=True, fmt='d')
plt.ylabel('Valor Actual', fontweight = 'bold', fontsize = 17)
plt.xlabel('Valor predicho', fontweight = 'bold', fontsize = 17)
plt.show()
```



Resumen de la comparación de métodos

Método	Exactitud	Precisión	Sensibilidad
Random Forest	98.95%	98.33%	99.57%
SVM	99.37%	98.96%	99.78%
Gradient Boosting	99.05%	98.74%	99.36%
XGBoost	99.26%	98.96%	99.57%
KNN	99.05%	98.53%	99.57%
Naive Bayes	98.21%	96.91%	99.57%
Árbol de decisión	98.01%	99.13%	96.83%

De acuerdo a la comparación de métodos se puede apreciar que SVM es el que mejor rendimiento proporciona en todas las métricas evaluadas, a simple vista muestra excelentes resultados pero sería muy importante determinar la significancia estadística de ese algoritmo para concluir que en efecto es mejor que Random Forest. A su vez sería muy interesante optimizar los hiperparámetros de todos los métodos comparados y determinar si se puede obtener una mejoría en las métricas de evaluación, la optimización de hiperparámetros de los métodos comparados no se encuentra dentro del alcance de esta investigación