Universitat Oberta de Catalunya

uoc.edu

Modelos de clasificación para incidencias en entornos industriales con datos no balanceados

José Manuel Martínez Raya
Máster Universitario en Ciencia de Datos
Minería de datos y machine learning

Raúl Parada Medina *(Profesor Colaborador)*
Jordi Casas Roma *(Profesor Responsable)*

Junio de 2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Modelos de clasificación para incidencias en entornos industriales con datos no balanceados*
Nombre del autor:	*José Manuel Martínez Raya*
Nombre del consultor/a:	*Raúl Parada Medina*
Nombre del PRA:	*Jordi Casas Roma*
Fecha de entrega (mm/aaaa):	06/2019
Titulación::	*Máster Universitario en Ciencia de Datos*
Área del Trabajo Final:	*Minería de datos y machine learning*
Idioma del trabajo:	*Español*
Palabras clave	*Machine Learning, Anomaly Detection, Imbalanced Dataset, Industrial Automation*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

En el momento actual desde el sector industrial y tecnológico se habla de la cuarta revolución industrial. Un nuevo paradigma en donde las fábricas del futuro serán automatizadas, digitales, inteligentes, flexibles, sostenibles y más humanas. [1]

Un punto importante será el de predecir los posibles errores y fallas que se puedan encontrar a lo largo de todo el proceso de producción, desde la recepción de las materias primas hasta el producto acabado.

La tendencia es que el proceso se llegue automatizar en su totalidad, en donde ni el componente humano tenga un papel supervisor. Serán los propios dispositivos, autómatas, controlados por una IA capaz de predecir los errores y mantener la producción en niveles de excelencia.

Pero antes, para llegar hasta ese punto, se debe comprender e identificar el porqué de algunos errores en el proceso de fabricación, con tal de evitarlos y mejorar la calidad del producto final.

La creación de un modelo de clasificación nos ayudará a optimizar el rendimiento de las máquinas y es el primer paso para un mantenimiento predictivo que evite futuros fallos.

Abstract (in English, 250 words or less):

At the present time from the industrial and technological sector we speak of the fourth industrial revolution. A new paradigm where the factories of the future will be automated, digital, intelligent, flexible, sustainable and more humane.

An important point will be to predict the possible errors and failures that can be found throughout the production process, from the receipt of raw materials to the finished product.

The tendency is for the process to be automated in its entirety, where even the human component does not have a supervisor role. They will be the devices themselves, automatons, controlled by an AI capable of predicting errors and maintaining production at levels of excellence.

But first, to reach that point, you must understand and identify the reason for some errors in the manufacturing process, in order to avoid them and improve the quality of the final product.

The creation of a classification model will help us to optimize the performance of the machines and is the first step for predictive maintenance to avoid future failures.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación

"Si funciona, no lo toques" Dice el viejo dicho, pero el auge de los análisis predictivos está haciendo que esa idea se vaya diluyendo al ayudar a las empresas a corregir los problemas antes de que surjan. Ahora estamos ante una nueva era donde "Más vale prevenir que curar". [2]

Dentro de las motivaciones en la elección del trabajo, destacar su carácter innovador dentro del sector y el prometedor futuro de esta tecnología. Tal como se explicaba en el anterior resumen, es previsible la tendencia a la automatización de una gran mayoría de las actividades realizadas hoy en día por el hombre. Siendo la industria de los manufacturados un sector que incluye una gran cantidad de procesos mecánicos, es firme candidata a la automatización integral, con el considerable aumento de la rentabilidad que ello supone.

Las fábricas inteligentes recopilan cada vez más datos de los sensores distribuidos en toda la cadena de producción, utilizando posteriormente diferentes algoritmos para que detecten señales de advertencia.

Un parte importante dentro del proceso de automatización es la del mantenimiento y control de las máquinas. De forma que se pueda prevenir futuras averías para evitar costes y recursos innecesarios.

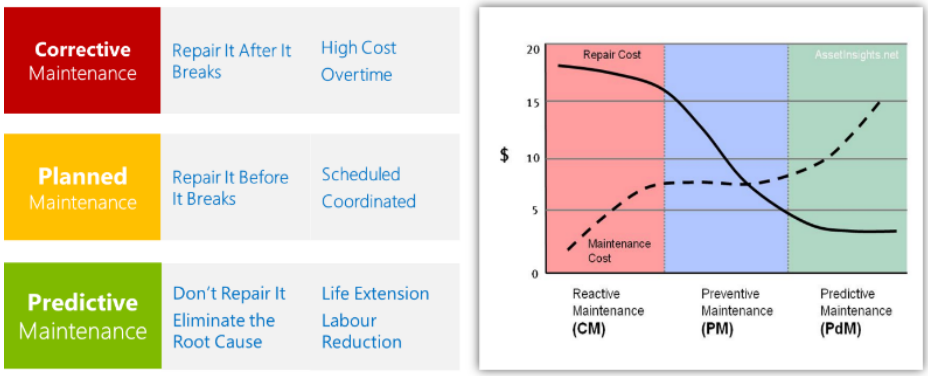


Ilustración 1 Costes según el tipo de mantenimiento
https://www.marutitech.com/problems-solved-machine-learning/

Desde un lado más personal y profesionalmente hablando, la organización para la cual ofrezco mis servicios tiene como actividad principal, la automatización de procesos industriales mediante autómatas programables. Todos los eventos durante el proceso de fabricación se monitorizan creando un volumen de registros que se almacena en una base de datos centralizada por cada fábrica.
Los datos tienen diversos orígenes según el autómata, por ejemplo, alarmas, procesos asignados, motores, sensores, personal, etc. De todo este contenido se puede extraer información muy valiosa, útil para la mejora de la organización. Pero existe un problema y es que el volumen de datos es bastante alto lo que impide que un operario pueda manejar y localizar patrones que sean del interés de la dirección.

La creación de un modelo de aprendizaje automático es una de las opciones más interesantes y nos permitirá entender los datos generados, transformándolos en información útil para la organización.

1.2 Objetivos del Trabajo

- Visualizar el efecto de una variable de entrada sobre otras de salidas (Simulación) y si proporciona valores óptimos de funcionamiento (Optimización)

- La fuente de datos está altamente desbalanceada por lo que no se espera que los resultados, con respecto a la precisión y sensibilidad, sean muy buenos. Se habrá de aplicar un tipo de tratamiento para solventarlo.

- Creación de un modelo supervisado de clasificación con técnicas de árboles de decisión, de redes neuronales, máquinas de soporte vectorial o de k-NN.

- Creación de un modelo supervisado de predicción con técnicas de análisis de regresión, de árboles de regresión, de redes neuronales o de k-NN.

- Resolver si existe algún patrón o tendencia que nos advierta de un posible mal funcionamiento o falla inminente de las máquinas con el fin de evitarlo.

- ¿Cuál es la probabilidad de que una máquina falle en un futuro cercano debido a una falla de un componente determinado?

1.3 Enfoque y método seguido

Se seguirá las etapas de un proyecto de Minería de datos y ML, pues esta planificación está ampliamente consolidada y sigue a su vez las fases típicas del ciclo de vida de los datos.

Comenzaremos con el análisis previo de los datos y su preparación. Qué cantidad de atributos y registros disponemos, de qué tipo son, de dónde provienen y en qué condiciones nos llegan, la estructura presentada y si tienen errores.

Posteriormente seleccionaremos aquellos datos que nos permitan alcanzar mejor nuestros objetivos, descartando así otros. Para ello se evaluarán su relevancia, calidad y limitaciones de los mismos.

Una vez tengamos los datos con los que vamos a trabajar se procederá a prepararlos aplicando técnicas de transformación, discretización y reducción de la dimensionalidad, si fuera el caso.

Al tratarse de un problema de clasificación puede aceptar técnicas de análisis discriminante, de árboles de decisión, de redes neuronales, máquinas de soporte vectorial (SVMs) o de k-NN. [3]

El problema se puede resolver con varias técnicas por lo que se realizará una búsqueda del mejor modelo posible tal y como se muestra en la siguiente figura:

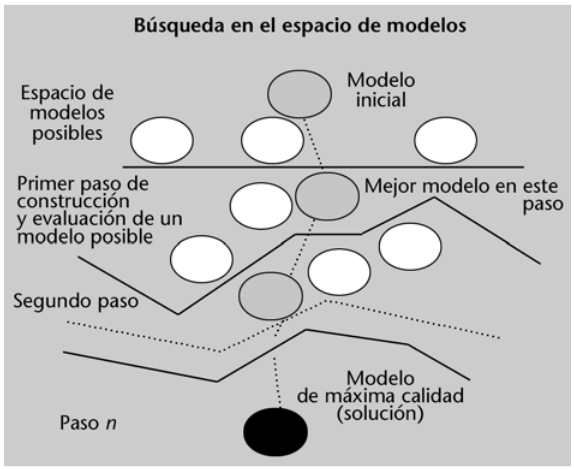


Ilustración 2 Sangüesa i Solé (2018)
El proceso de descubrimiento de los conocimientos a partir de los datos IFUOC.

Observamos como desde un modelo inicial, crearemos varias posibilidades en repetidos pasos ajustándolos a medida que vayamos evaluando su comportamiento y hasta llegar al mejor modelo que explique los atributos que se encuentran en los datos. [4]

En lenguaje de programación para la escritura del código será Python en un entorno Jupyter Notebook ampliamente usado en proyecto de ciencia de datos por sus características [5]. Además, Python es actualmente es uno de los lenguajes con más éxito y con una comunidad ampliamente extendida [6].

Cuando se disponga del modelo adecuado se procederá a su representación con algún tipo de herramienta que también nos permita explorarlos. Una opción, sería Tableau que está posicionada como uno de los mejores [7] y permite la generación de dashboards.

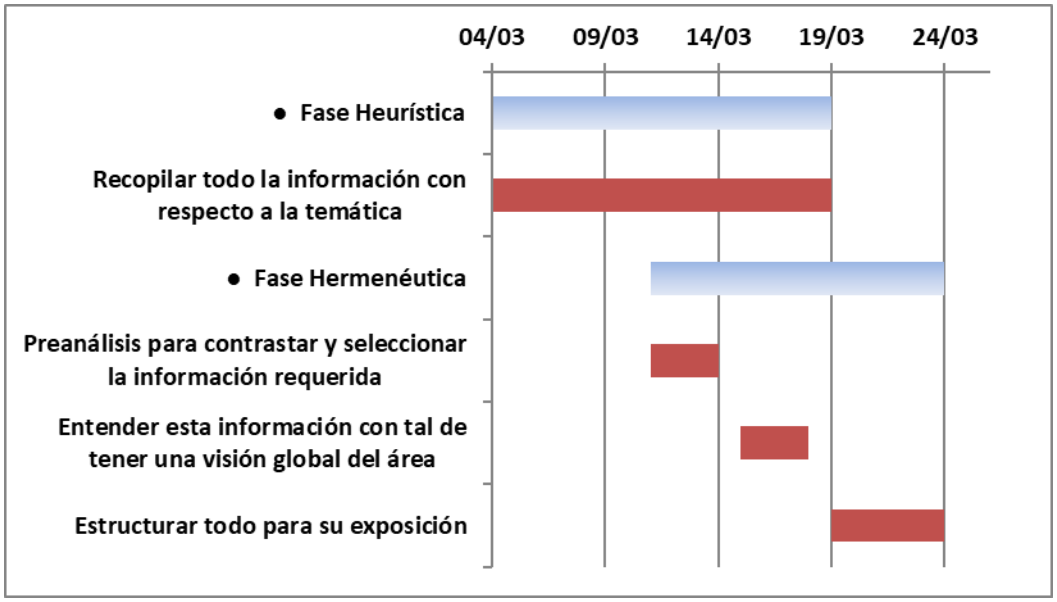
1.4 Planificación del Trabajo

Para la realización de este proyecto se hará uso de los siguientes recursos:

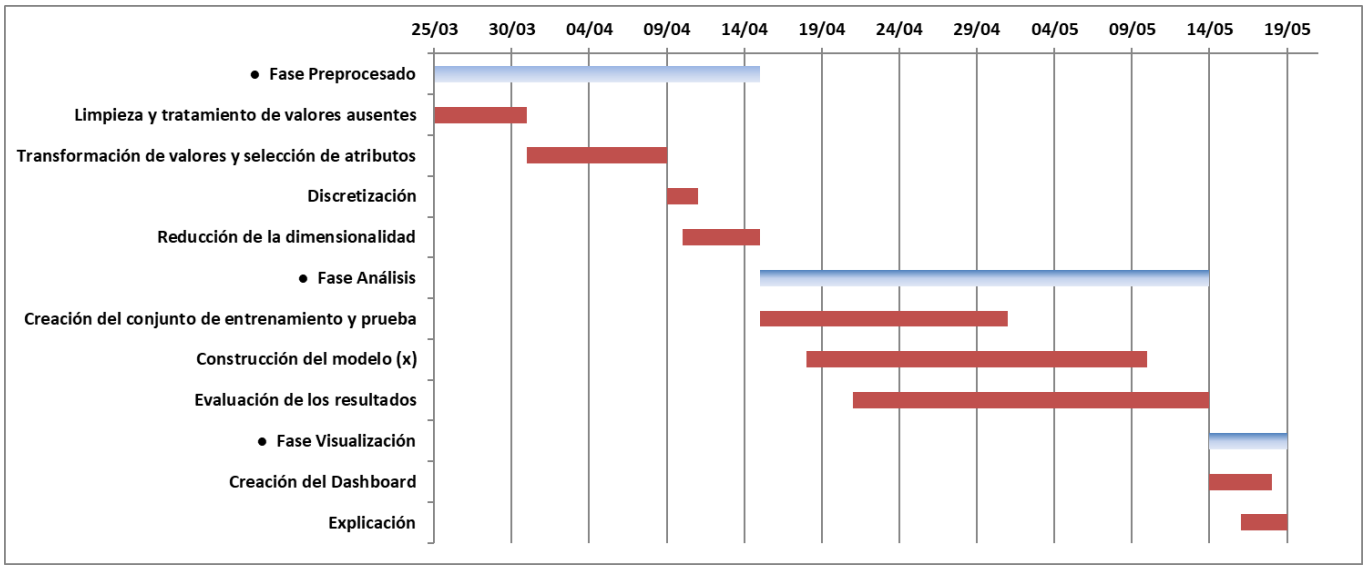
- Hardware
  - Torre: Thermaltake Toughpower Grand RGB 750W 80 Plus Gold Modular
  - Fuente de alimentación: Nfortec Scutum PSU 750W Modula
  - Procesador: Intel i7-7700 (4Ghz, 12MB)
  - Ventilador: Cooler Master Hyper 212X
  - Placa Base: Asus Prime Z270-K
  - Disco duro:
    - SAMSUNG 970 250 SSD EVO NVMe M.2
    - Kingston SSDNow UV400 240GB SATA3
    - 2TB SATA3 64MB
  - Memoria: DDR4 3000 PC4-24000 16GB 2x8GB
  - Gráfica: Asus Dual GTX 1060 OC 6GB GDDR5 (CUDA)

- Software
  - Windows 10
  - Jupyter Notebook
  - TensorFlow-GPU & Keras
  - Python 3.6
  - Librerías especializadas ML
  - Microsoft Office

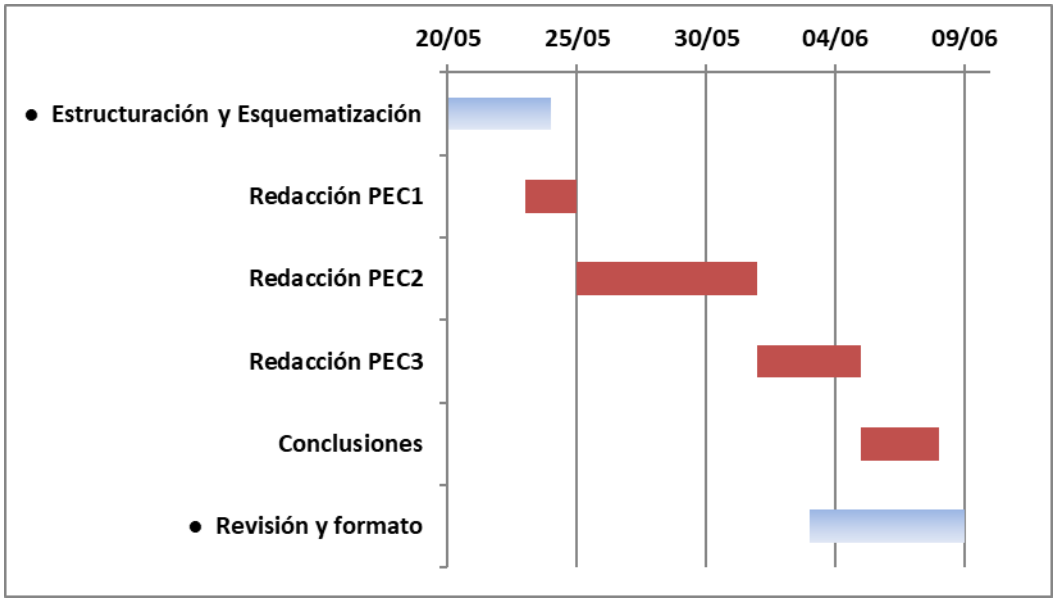
Tareas	Inicio	Días	Fin
Estado del arte o análisis de mercado del proyecto	**04/03**	**20**	**24/03**
● **Fase Heurística**	04/03	15	19/03
Recopilar todo la información con respecto a la temática	04/03	15	19/03
● **Fase Hermenéutica**	11/03	13	24/03
Preanálisis para contrastar y seleccionar la información requerida	11/03	3	14/03
Entender esta información con tal de tener una visión global del área	15/03	3	18/03
Estructurar todo para su exposición	19/03	5	24/03
Diseño e implementación del trabajo	**25/03**	**55**	**19/05**
● **Fase Preprocesado**	25/03	21	15/04
Limpieza y tratamiento de valores ausentes	25/03	6	31/03
Transformación de valores y selección de atributos	31/03	9	09/04
Discretización	09/04	2	11/04
Reducción de la dimensionalidad	10/04	5	15/04
● **Fase Análisis**	15/04	29	14/05
Creación del conjunto de entrenamiento y prueba	15/04	16	01/05
Construcción de los modelos	18/04	22	10/05
Evaluación de los resultados	21/04	23	14/05
● **Fase Visualización**	14/05	5	19/05
Creación del Dashboard	14/05	4	18/05
Explicación	16/05	3	19/05
Redacción de la memoria	**20/05**	**20**	**09/06**
● **Estructuración y Esquematización**	20/05	4	24/05
Redacción PEC1	23/05	2	25/05
Redacción PEC2	25/05	7	01/06
Redacción PEC3	01/06	4	05/06
Conclusiones	05/06	3	08/06
● **Revisión y formato**	03/06	6	09/06
Presentación y defensa del proyecto	**10/06**	**6**	**16/06**
Creación de diapositivas	10/06	3	13/06
Grabación del video	13/06	2	15/06
Montaje	14/06	2	16/06



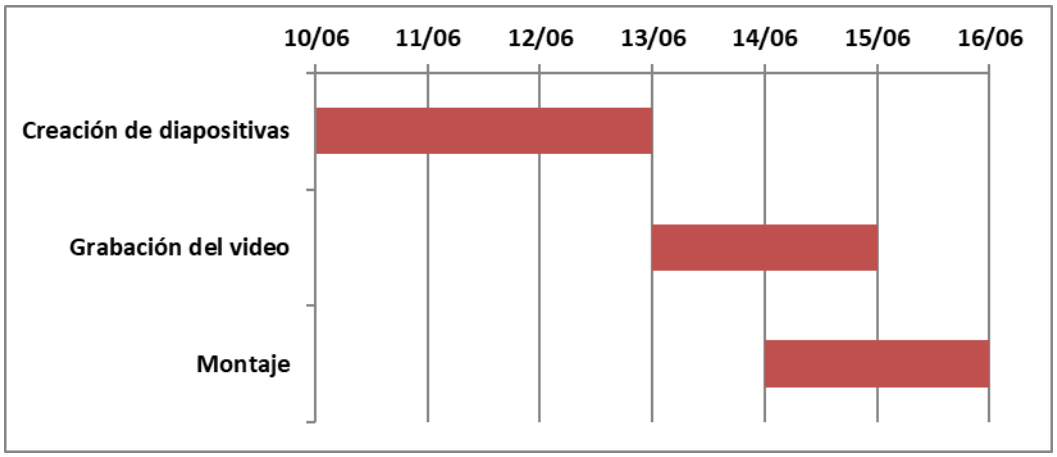
PEC2 (Inicio: 04/03/19 – Entrega: 24/03/19)



PEC3 (Inicio: 25/03/19 – Entrega: 19/05/19)



PEC4 (Inicio: 20/05/19 – Entrega: 09/06/19)



PEC5 (Inicio: 10/06/19 – Entrega: 16/06/19)

1.5 Breve sumario de productos obtenidos

En la realización de este proyecto se han creado lo siguientes productos:

- Memoria (este documento)
- Conjunto de datos (archivo CSV)
- Código fuente (archivo IPYNB)
- Presentación con diapositivas (PowerPoint)

1.6 Breve descripción de los otros capítulos de la memoria

En los sucesivos capítulos se desarrolla los procedimientos para cumplir los objetivos que se han marcado.

En el capítulo denominado *Estado del Arte*, se realiza un breve repaso del origen de la industria 4.0 y su definición. Seguidamente, se explica cuál es la situación actual del aprendizaje automático en la automatización industrial, las técnicas y metodologías que se están implementado, también se muestra algunos casos de éxitos.

El siguiente capítulo *Preparación de los datos,* se trabaja en adecuar el conjunto de datos que se ha facilitado para la creación de los algoritmos. Dividido en tres partes. Lectura de los datos, de dónde provienen y cómo son introducidos al sistema. Limpieza de los datos, cómo se transforman para su correcto tratamiento. Por último, la reducción del conjunto para facilitar su procesado.

En el apartado *Algoritmos de Machine Learning*, se desarrolla los distintos modelos como los parámetros que se escogieron para cada uno de ellos. También se trata la división del conjunto de datos y el problema del desbalanceo.

En la *Evaluación del Modelo* se hace una breve introducción sobre las métricas y formas de evaluar la eficacia de cada modelo. Acto seguido se muestran los resultados obtenidos y se realiza una comparativa. Al final se presenta una serie de gráficos a modo de visualización para su mejor comprensión.

Por último, la sección de *Conclusiones*, se realiza una síntesis de los resultados obtenidos, si son los que se esperaban y si se cumplieron los objetivos marcados al principio del proyecto.

2. Estado del arte

2.1 Introducción

El termino de Industria 4.0 se originó en la feria de Hannover en Alemania durante el año 2011, cuando el gobierno y el sector empresarial, liderados por Bosch, formaron un grupo de investigación para encontrar un marco común que permitiera la aplicación de nuevas tecnologías y digitalización de la industria.

En esta misma feria, durante el año 2013, se presentó al público el informe final en donde se hacía referencia al paradigma que ahora se conoce como la cuarta revolución industrial. Actualmente, diferentes países se refieren a este concepto en diferentes términos y se aplica dentro del ecosistema industrial tanto a nivel multinacional como de PYMES [8]

En el año 2016 solo una quinta parte de las empresas industriales habían digitalizado sus procesos clave a lo largo de la cadena de valor; pero la tendencia es que en el 2021 el 85% de estas compañías habrán implementado soluciones de la Industria 4.0 en todas las divisiones comerciales importantes. [9].

La industrial 4.0 se encuentra en su etapa inicial, es esencial definir claramente su estructura, metodologías y desafíos como marco para su implementación en la industria. La digitalización en muchos de los sectores viene de la mano gracias sobre todo, a la irrupción de nuevas tecnologías, como por ejemplo la Industria de Internet de las cosas (IIoT), el Big Data, redes de sensores inalámbricas (WSN), la computación en la nube y los sistemas cibernéticos (CPS).

Un ejemplo podría ser el de autómatas conectados conjuntamente formando una gran red, muchos de ellos de forma inalámbrica y que generarían una gran cantidad de datos, posteriormente se almacenarían de forma constante en la nube y sería analizado en tiempo real para tomar decisiones de la producción según las condiciones.

Son diversos los autores que coinciden en que estos aspectos serán los pilares en la nueva organización industrial [10-20].

Todos estos elementos nos permiten tener sistemas de aprendizaje automático, pero dentro de ellos el Big Data tiene un papel fundamental en la forma en que se tomarán las decisiones combinando varias áreas como son las matemáticas, estadística avanzada e informática, todo ello englobado bajo el Machine Learning. El objetivo final será la detección de patrones en los conjuntos de datos y las relaciones que existen entre estos.

Las condiciones básicas que deben cumplirse para determinar que una organización tiene un esquema de mantenimiento automatizado son las siguientes: [21]

- La monitorización y medición de un componente, debe hacerse de manera no intrusiva, bajo condiciones de normalidad.

- La variable que debe medirse para hacer las predicciones, debe cumplir con las condiciones de: repetibilidad, análisis, parametrización y diagnóstico.

- Los resultados y los valores de las medidas se pueden expresar en unidades físicas o índices correlacionados.

2.2 Enfoque general

Comenzando por lo esencial, diferentes investigaciones trabajan con tres enfoques a la hora de iniciar un proyecto de aprendizaje automático. [22-24]

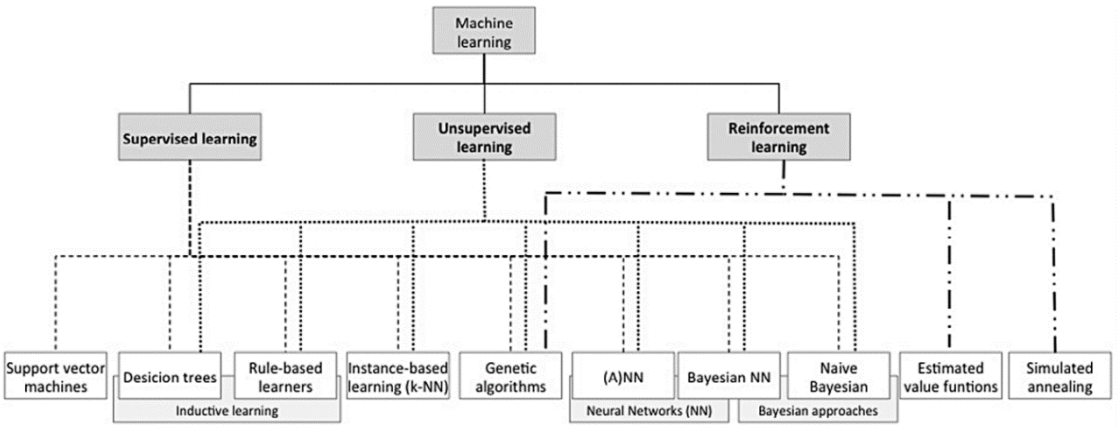


Ilustración 3. Estructuración de técnicas y algoritmos de aprendizaje automático. https://www.researchgate.net/figure/Structuring-of-Machine-Learning-techniques-and-algorithms_fig1_304355797

Hacer especial mención a los modelos, no tan conocidos, como son los de aprendizaje por refuerzo (Reinforcement learning) o también denominados semisupervisados. Estos modelos aprenden de la experiencia y son una combinación de los modelos supervisados y no supervisados. El sistema recibe entradas mientras interactúa con los procesos de fabricación y toma decisiones para maximizar las recompensas futuras. La idea es que en lugar de un conjunto de datos de entrenamiento que indica el resultado correcto para una entrada determinada, solo se hará una indicación de si una acción es correcta o no. Si una acción no es correcta, entonces habrá que continuar hasta encontrarla [25].

No obstante, la mayoría de los problemas de fabricación pertenecen a problemas de clasificación, donde se tiene que determinar una etiqueta de la clase para un objeto específico o una situación basada en un gran conjunto de datos [26]. Los problemas de clasificación no tienen que estar relacionados solo con la fabricación, sino que pueden relacionarse con toda la industria. Por ejemplo, la resolución de problemas y el control de calidad [26]. El resultado de la investigación científica ha demostrado que las técnicas de aprendizaje profundo se consideran herramientas potentes para la mejora permanente de la calidad en procesos grandes y complejos, como es la fabricación de semiconductores. Además de aplicación en sistemas de fabricación LEAN y también mediante el uso de herramientas *just-in-time* (JIT) y kanban. Los resultados de la implementación de ML en los sistemas de fabricación LEAN muestran que las redes neuronales y los árboles de decisión representan dos algoritmos muy potentes para ajustar el número de kanbans en entornos dinámicos de fabricación JIT [26].

2.3 Aprendizaje Profundo

Tanto el aprendizaje profundo como el automático están basados en técnicas que modelan la compleja relación entre los datos. En el caso del aprendizaje profundo (DL) se integra el aprendizaje de características y la construcción de modelos ajustando los parámetros para su optimización desde su inicio hasta su final. Además, si nuestro conjunto de datos contiene numerosos datos sin etiquetar podemos utilizar el aprendizaje profundo con autoencoders para realizar un preentrenamiento y luego entrenar el modelo con los datos etiquetados.

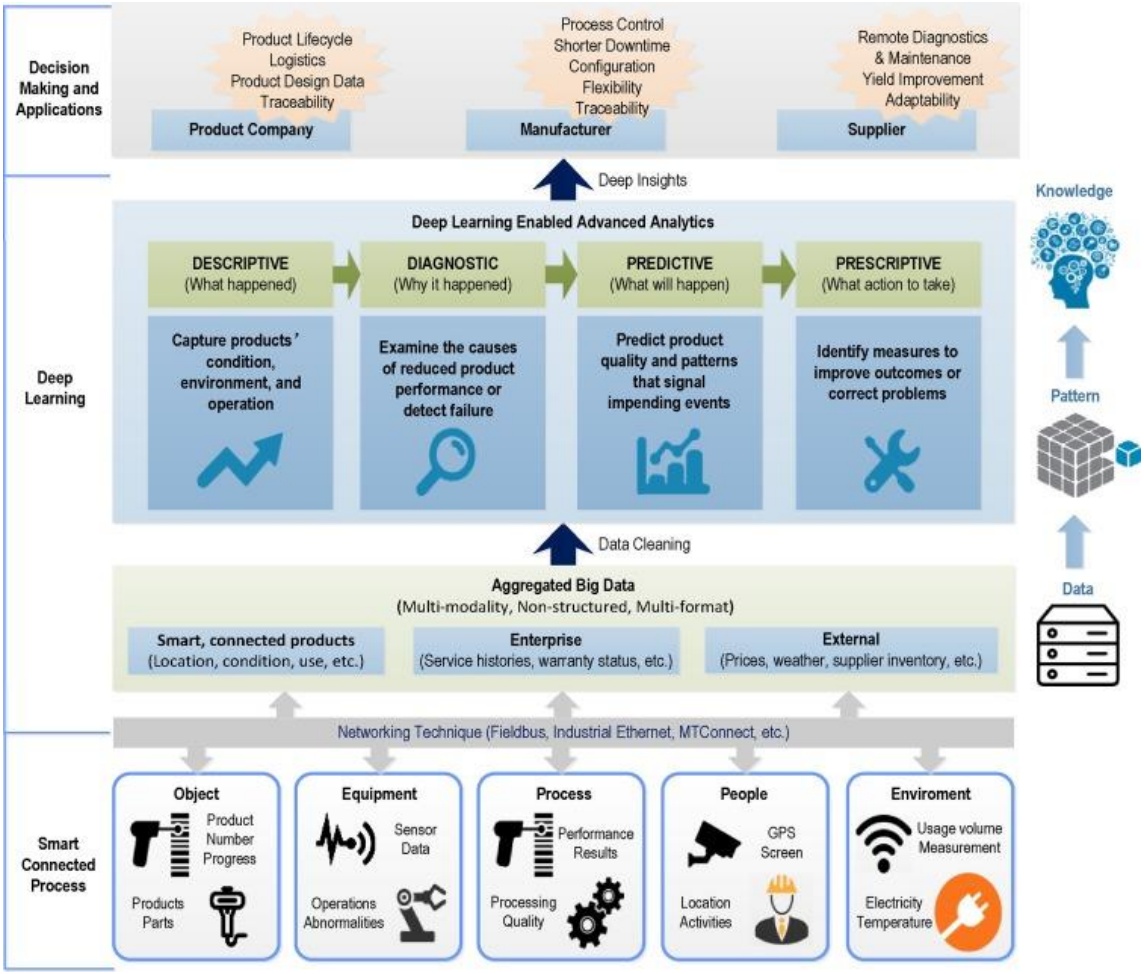


Ilustración 4

De cómo el aprendizaje profundo usa el análisis avanzado para la fabricación inteligente.
https://www.researchgate.net/publication/322325843_Deep_learning_for_smart_manufacturing_Methods_and_applications

La arquitectura de las redes neuronales integra muchas capas ocultas lo que permite crear de forma automática una jerarquía de conceptos, empezando por los más simples, e ir mezclándolos para crear otros de más complejos. Atributos como un borde, esquina, contorno y objeto, se abstraen capa por capa de una imagen. Estas representaciones de entidades acaban en una última capa que realizará tareas de clasificación y regresión. En general, el aprendizaje profundo es una estructura de aprendizaje con una mínima inferencia humana,

Por el contrario, el aprendizaje automático tradicional realiza la extracción de atributos y la construcción de modelos de manera separada, y cada módulo se construye paso a paso. En primer lugar, los atributos se extraen transformando los datos sin procesar en un dominio representativo. A continuación, se realiza la selección de atributos para

mejorar la relevancia y reducir la redundancia. Las técnicas tradicionales de aprendizaje automático generalmente tienen estructuras poco profundas. Por lo tanto, el rendimiento del modelo construido no solo se basa en la optimización de los algoritmos adoptados, sino que también se ve muy afectado por la ingeniería de características. En general, la extracción y selección de atributos requieren mucho tiempo y dependen en gran medida del conocimiento del dominio.

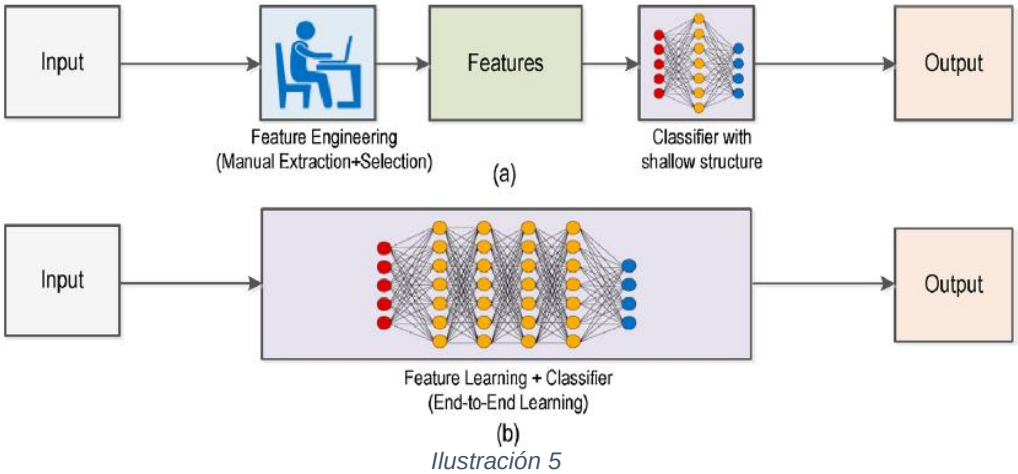


Ilustración 5

Comparación de un modelo de aprendizaje automático tradicional (a) con otro de profundo (b)
https://www.researchgate.net/publication/322325843_Deep_learning_for_smart_manufacturing_Methods_and_applications

2.3 Casos de estudio

- CASO 1

Los sistemas de calefacción, ventilación y aire acondicionado (HVAC, por sus siglas en inglés) controlan el clima interior, la temperatura del aire, la humedad y la presión, creando un ambiente óptimo de producción en edificios industriales. Estos equipos son cruciales para el funcionamiento de una fábrica en el contexto de la Industria 4.0. No obstante, el mantenimiento rutinario no siempre identifica posibles fallas. El objetivo del mantenimiento es extender la viuda útil del equipo, reconociendo patrones de posibles fallas como son la vibración, temperatura o equilibrio.
En este estudio se usó los datos generados por los sistemas HVAC de 20 edificios.

Con un total de 8000 registros cuyos cuales contienen la temperatura óptima y los valores reales medidos por sensores en los distintos edificios. Se utilizaron para analizar el comportamiento del sistema HVAC y determinar si el equipo lograba mantener las temperaturas.

Diferentes autores realizaban selección de atributos y categorizaban algunos atributos, como son la temperatura y las alarmas [27][28]. Posteriormente se dividió el dataset en conjunto de entrenamiento y prueba. Para la creación del modelo se usaron dos algoritmos de aprendizaje supervisado la ***regresión logística*** y el ***RandomForest*.**

La regresión logística es un tipo de análisis de regresión utilizado para predecir el resultado de una variable categórica que únicamente puede tomar un número limitado de categorías. En el caso del estudio citado se aplicó un modelo binomial para las temperaturas. La precisión con este modelo fue de 0.65. Por otro lado, los resultados con el *RandomForest* fueron muy similares. [8]

- CASO 2

En este segundo caso los autores realizan un mantenimiento predictivo del motor de un avión en donde el objeto de estudio tiene un componente de degradación progresivo. El algoritmo revisa los valores de los sensores y los cambios en los valores del históricos de alarmas. [29]
El lenguaje propuesto por los autores en la creación del modelo es R. Además, se han utilizado tres soluciones diferentes para su resolución:

- **Regresión**: predice la vida útil restante o el tiempo hasta el fallo.
- **Clasificación binaria**: predice si un activo fallará dentro de un período de tiempo determinado.
- **Clasificación de múltiples clases:** predice si un activo fallará en diferentes ventanas de tiempo.

Los datos de entrenamiento consistían en múltiples series de tiempo multivariadas en ciclos como unidad de tiempo, junto con 21 lecturas de sensores por cada ciclo.
De las 21 mediciones se realizaron una derivación de datos para crear un atributo nuevo que los agrupaba a todos.

Los resultados obtenidos para cada enfoque y modelo fueron:

Algorithms	Negative Log Likelihood	Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
Decision Forest Regression	462.724302192819	21.141062	30.147662	0.574998	0.526317	0.473683
Boosted Decision Tree Regression	Infinity	21.283579	29.615866	0.578874	0.507913	0.492087
Poisson Regression	NA	23.243022	29.973356	0.632167	0.520249	0.479751
Neural Network Regression	NA	75.505023	86.185055	2.053597	4.301346	-3.301346

Ilustración 6 Resultados modelo de regresión

Vemos como el modelo *Decision Forest Regression* y Boosted Tree Regression tienen un mejor desempeño con respecto a las métricas de *Error absoluto medio* y *raíz del error cuadrático medio.*

Algorithms	Accuracy	Precision	Recall	F-Score
Logistic Regression	0.92	0.947368	0.72	0.818182
Boosted Decision Tree	0.91	0.9	0.72	0.8
Decision Forest	0.92	0.947368	0.72	0.818182
Neural Network	0.94	0.952381	0.8	0.869565

Ilustración 7. Resultados modelo de clasificación binaria

En esta segunda fase de clasificación binaria observamos como las redes neuronales destacan sobre los otros modelos superándolos en todas las métricas.

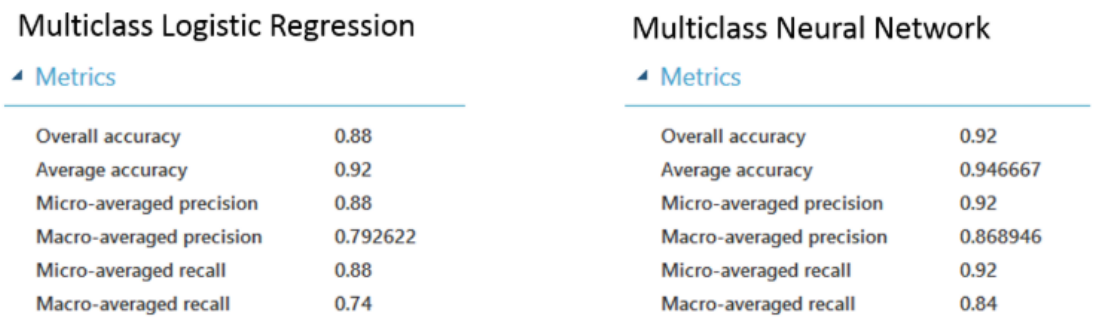


Ilustración 8 Resultado modelo clasificación multiclase

Por último, en la fase de multiclase también se observa como la red neuronal tiene unos mejores valores que la regresión logística.

- CASO 3

En este caso se utiliza técnicas de fusión de datos que permiten el diseño de un modelo de mantenimiento predictivo. La fusión de datos supone un problema cuando se trata de datos generados por sensores distribuidos.

Los autores [31] utilizan un conjunto de datos en donde se registran la temperatura óptima y los valores detectados por los sensores, de esta manera se detectaba si el equipo estaba fallando o no.

En este aspecto tenemos un caso similar al primero, pero a diferencia de aquel, aquí se utiliza el algoritmo *Naive Bayes.* Se trata de un algoritmo ya utilizado en procesos secuenciales de mantenimiento para modelos probabilísticos, la ventaja es que no se modifica la representación en problemas de fusión de datos.

El algoritmo considera que a los valores correctos se les asigna un 0 y los considerados como alarmas o fuera de rango, serían positivos catalogados como 1.

```
#Implementación de Naive Bayes
from sklearn.cross_validation import train_test_split
from sklearn.naive_bayes      import GaussianNB
from matplotlib               import pyplot as plt
from sklearn.metrics          import accuracy_score
datos=[]                      # Lista que recibe las columnas del archivo csv
datos_entren=[]               # Lista que recibe los datos de entrenamiento (temp.objeto y temp. actual)
etiqueta_entren=[]            # Lista que recibe las etiquetas de entrenamiento (temp.actual)
filas = open('temperaturas.csv','r').read().splitlines() # Se abre el archivo csv de temperaturas
filas.pop(0)                                             # Se descarta la fila de encabezados
for f in filas:
    fila = f.split(',')                                  # Se dividen las columnas utiliznado comas
    datos.append([int(fila[0]), int(fila[1]),int(fila[2]),int(fila[3]), # Se cargan las columnas a las listas
                  int(fila[4]),int(fila[5])])
    datos_entren.append([int(fila[0]),int(fila[1])])
    etiqueta_entren.append([int(fila[1])])
#*******************************************************************************************
datos_entren, datos_prueba, etiqueta_entren, etiqueta_prueba =  train_test_split(datos_entren,
                                                                etiqueta_entren,test_size=.2)
#*******************************************************************************************
clasf = GaussianNB()                    #Se activa el clasificador
clasf.fit(datos_entren, etiqueta_entren)  #Se envian los datos para entrenamiento del algoritmo
val_pred = clasf.predict(datos_prueba)    #Se realiza la predicción con datos de prueba
print (val_pred)                          #Impresión de los valores de predicción
print (etiqueta_prueba)                   #Impresión de los valores de la lista etiquetas (temp. actual)
exactitud = accuracy_score(val_pred, etiqueta_prueba) # Cálculo de la exactidud de la predicción
print(exactitud)
#*******************************************************************************************
plt.plot(val_pred,marker='o', color='r',linewidth=3.5,linestyle='--')       # Impresión valores de predicción
plt.plot(etiqueta_prueba,marker='o',color = 'b',linewidth=1.5,linestyle='-') # Impresión valores reales
plt.show
```

Ilustración 9. Algortimo Naive Bayes empleado en el CASO nº 3

Los resultados de este modelo de clasificación supervisada, fueron excelentes siendo cercanos al 100% en exactitud.

2.4 Conclusión

Tanto los casos mostrados como la literatura nos indica que se debe primero, realizar un preprocesamiento de los datos, pues nos encontraremos con numerosas variables de mediciones que se deben seleccionar y reducir.

Segundo, como se ha observado, se debe realizar una búsqueda del mejor modelo probando varios enfoques, aunque las redes neuronales son un firme candidato a tener en cuenta por sus buenos resultados, no debemos descartar otros modelos que se puedan adaptar mejor al tipo concreto de datos.

3. Preparación de los datos

Lo más normal en un proyecto de minería de datos, es que los datos no estén en el formato adecuado y, por lo tanto, tengan que ser organizados de forma que puedan ser tratados por los modelos seleccionados y así obtener el mejor resultado posible del conjunto de datos.

Dividiremos la preparación de los datos en tres fases:

- **Lectura de los datos**. Se cargarán los datos en bruto al programa.
- **Limpieza de los datos**. Se corregirán errores, inconsistencias, etc.
- **Transformación de los datos**. Adaptación de los datos para su posterior análisis.

3.1 Lectura de los datos

Carga de datos

El conjunto de datos nos viene dado en un único archivo con extensión .csv (*comma-separated values*) lo que significa que los valores vendrán separados por comas. El tamaño del archivo es de aproximadamente un 1 Mb.

Antes de cargar el archivo comprobaremos que no haya problema con los decimales si estos existieran. Según el idioma con el que esté configurado la aplicación Excel el tratamiento del separador decimal será diferente. Por ejemplo, para el idioma inglés la coma (,) se utiliza para separar los valores y el punto (.) como separador decimal. En cambio, en el idioma español el separador punto y coma (;) se utiliza para los valores y la coma (,) para los decimales.

dataset.csv

```
Date,Temperature,Humidity,Operator,Measure1,Measure2,Measure3,Measure4,Measure5,Measure6,Measure7,Measure8,Measure9,Measure10,
2016-01-01 00:00:00,67,82,Operator1,291,1,1,1041,846,334,706,1086,256,1295,766,968,1185,1355,1842,90,No,2016,1,1,5,0,0,0
2016-01-01 01:00:00,68,77,Operator1,1180,1,1,1915,1194,637,1093,524,919,245,403,723,1446,719,748,91,No,2016,1,1,5,1,0,0
2016-01-01 02:00:00,64,76,Operator1,1406,1,1,511,1577,1121,1948,1882,1301,273,1927,1123,717,1518,1689,92,No,2016,1,1,5,2,0,0
2016-01-01 03:00:00,63,80,Operator1,550,1,1,1754,1834,1413,1151,945,1312,1494,1755,1434,502,1336,711,93,No,2016,1,1,5,3,0,0
2016-01-01 04:00:00,65,81,Operator1,1928,1,2,1326,1082,233,1441,1736,1033,1549,802,1819,1616,1507,507,94,No,2016,1,1,5,4,0,0
2016-01-01 05:00:00,67,84,Operator1,398,1,2,1901,1801,1153,1085,1547,2005,477,1217,1632,1324,1854,1739,95,No,2016,1,1,5,5,0,0
```

Ilustración 10. Datos en bruto

Procedemos a la carga del fichero en nuestro entorno de Jupyter Notebook y vemos como se ha dado formato por campos:

	Date	Temperature	Humidity	Operator	Measure1	Measure2	Measure3	Measure4	Measure5	Measure6	...	Measure15
0	2016-01-01 00:00:00	67	82	Operator1	291	1	1	1041	846	334	...	1842
1	2016-01-01 01:00:00	68	77	Operator1	1180	1	1	1915	1194	637	...	748
2	2016-01-01 02:00:00	64	76	Operator1	1406	1	1	511	1577	1121	...	1689
3	2016-01-01 03:00:00	63	80	Operator1	550	1	1	1754	1834	1413	...	711
4	2016-01-01 04:00:00	65	81	Operator1	1928	1	2	1326	1082	233	...	507

Ilustración 11. Datos cargados en Jupyter

Una primera verificación es comprobar que el número de variables y registros coinciden con los del archivo.

```
In [6]: dataset.shape
Out[6]: (8784, 28)
```

Ilustración 12 Función para determinar tamaño del conjunto de datos

Nuestro conjunto de datos tiene un total de 8784 observaciones y 28 variables. La carga ha resultado satisfactoria y coincide con los datos que tenemos en el archivo.

Una vez que tenemos nuestros datos cargado procederemos a comprobar si las variables tienen el tipo estadístico adecuado.

Tipos de variables estadísticas

Las variables se pueden clasificar según si son:

- Categóricas (objetos)
    - Nominal
    - Ordinal (se puede ordenar)
- Cuantitativas (valores numéricos)
    - Discreta (enteros y naturales)
    - Continua

En la siguiente tabla se detalla cómo son las variables de nuestro dataset:

Nombre	Tipo de variable	Tipo estadístico	Descripción
Date	Factor	Cuantitativa	Fecha completa del evento.
Temperature	Numérico	Cuantitativa	Temperatura
Humidity	Numérico	Cuantitativa	Humedad
Operator	Factor	Categórica	Operario
Measure1	Numérico	Cuantitativa	Medida
Measure2	Numérico	Categórica	Medida
Measure3	Numérico	Categórica	Medida
Measure4	Numérico	Cuantitativa	Medida
Measure5	Numérico	Cuantitativa	Medida
Measure6	Numérico	Cuantitativa	Medida
Measure7	Numérico	Cuantitativa	Medida
Measure8	Numérico	Cuantitativa	Medida
Measure9	Numérico	Cuantitativa	Medida
Measure10	Numérico	Cuantitativa	Medida
Measure11	Numérico	Cuantitativa	Medida
Measure12	Numérico	Cuantitativa	Medida
Measure13	Numérico	Cuantitativa	Medida
Measure14	Numérico	Cuantitativa	Medida
Measure15	Numérico	Cuantitativa	Medida
Hours Since Previous Failure	Numérico	Cuantitativa	Horas transcurridas sin incidencias
Failure	Factor	Categórica	Incidencia
Date.year	Numérico	Categórica	Año
Date.month	Numérico	Categórica	Mes
Date.day-of-month	Numérico	Categórica	Día
Date.day-of-week	Numérico	Categórica	Día de la semana
Date.hour	Numérico	Categórica	Hora
Date.minute	Numérico	Cuantitativa	Minutos
Date.second	Numérico	Cuantitativa	Segundos

Se observan como algunas variables no corresponden con el tipo estadístico.

Tenemos el caso de Measure1 y Measure2 que representan un estado o posición y no una medida por lo que son variables categóricas ocultas como variables cuantitativas.

```
dataset['Measure2'] = dataset['Measure2'].astype(object)
dataset['Measure3'] = dataset['Measure3'].astype(object)
```

Ilustración 13 Conversión de variables a tipo objeto

El tiempo es una magnitud física y corresponde a la una variable cuantitativa continua, en este caso la fecha (*Date)* se encuentra como objeto *string,* por lo tanto, deberemos pasarlo al tipo de *Datatime64.*

```
dataset['Date'] =  pd.to_datetime(dataset['Date'])
```

Ilustración 14 Conversión de variable a tipo fecha

Por último, los campos también relacionados con el tiempo: *Date.year,Date.month, Date.day-of-month, Date.day-of-week, Date.hour* han sido clasificados como numéricos pero corresponden a los de una variable categórica.

Revisión descriptiva

Continuaremos realizando una inspección rápida de las variables. Una forma de hacerlo es con una estadística descriptiva simple. Además, también revisaremos si el contenido de las variables que estamos utilizando coincide con los de origen.

- Variables cuantitativas

En el caso de variables cuantitativas buscaremos los mínimos, máximos, media y cuartiles. Básicamente es la representación de un diagrama de caja.
En cambio, para las variables cualitativas utilizaremos un diagrama de frecuencia.

```
summary = dataset.describe()
summary = summary.transpose()
```

Ilustración 15 Creación y transposición del resumen

Resumen descriptivo de las variables cuantitativas:

	count	mean	std	min	25%	50%	75%	max
Temperature	8784.0	64.026412	2.868833	5.0	62.00	64.0	66.0	78.0
Humidity	8784.0	83.337090	4.836256	65.0	80.00	83.0	87.0	122.0
Measure1	8784.0	1090.900387	537.097769	155.0	629.00	1096.0	1555.0	2011.0
Measure4	8784.0	1071.629895	536.518466	155.0	608.75	1058.0	1533.0	2011.0
Measure5	8784.0	1075.822860	533.158826	155.0	606.00	1077.0	1541.0	2011.0
Measure6	8784.0	1076.023793	534.004966	155.0	623.00	1072.0	1537.0	2011.0
Measure7	8784.0	1086.897086	538.195156	155.0	621.00	1089.0	1558.0	2011.0
Measure8	8784.0	1077.277209	537.187671	155.0	612.00	1074.0	1541.0	2011.0
Measure9	8784.0	1082.014572	532.983115	155.0	631.00	1078.0	1532.0	2011.0
Measure10	8784.0	1082.403005	537.582829	155.0	619.00	1080.0	1547.0	2011.0
Measure11	8784.0	1088.719148	534.995992	155.0	627.00	1093.0	1550.0	2011.0
Measure12	8784.0	1088.329349	533.299486	155.0	627.00	1082.0	1552.0	2011.0
Measure13	8784.0	1076.755806	535.111353	155.0	609.00	1067.0	1539.0	2011.0
Measure14	8784.0	1088.307377	537.264847	155.0	617.00	1088.5	1560.0	2011.0
Measure15	8784.0	1082.392304	537.527604	155.0	614.00	1076.0	1550.0	2011.0
Hours Since Previous Failure	8784.0	217.341872	151.751750	1.0	90.00	195.0	324.0	666.0
Date.minute	8784.0	0.000000	0.000000	0.0	0.00	0.0	0.0	0.0
Date.second	8784.0	0.000000	0.000000	0.0	0.00	0.0	0.0	0.0

Ilustración 16 Resumen estadístico de variables cuantitativas

Rápidamente se puede observar varias variables de medidas (*Measure)* con valores muy similares y que siguen una distribución prácticamente igual. Esto se debe tener en cuenta más adelante cuando tengamos que reducir datos y escoger que variables no aportan nueva información.

Además, observamos como las variables minutos y segundos no contiene ningún tipo de valor. Esto encaja con el origen de los datos que disponíamos.

- Variables cualitativas

Para las tablas de frecuencia de variables cualitativas podemos destacar:

Operator

En el caso del operario no está aportando información relevante pues parece ser que simplemente se ha distribuido los registros por número de operarios e incluso el segundo es justo el doble de todos los demás, puede tratarse de un error o caso excepcional.

col_0	count
Operator	
Operator1	976
Operator2	1952
Operator3	976
Operator4	976
Operator5	976
Operator6	976
Operator7	976
Operator8	976

Ilustración 17 Recuento de operadores

Si cruzamos los datos con los de incidencia o fallo en la máquina, el *operador nº 2* tiene una mayor representación debido a su mayor número de registros. Por lo que esto nos puede llevar a una falsa conclusión. Este hecho se debe tener en cuenta en el momento de selección de variables.

Measure 2 y 3

Lo mismo ocurre con las variables *de medida nº 2 y 3* en donde sospechosamente, se deja ver cierta distribución equitativa con respecto al número de apariciones de un valor. Esta tendencia también es observada en tablas cruzadas con respecto la clase objetivo.

Failure	No	Yes
Measure2		
0	2200	27
1	2154	14
2	2226	22
3	2123	18

Ilustración 18 Recuento de Measure2 vs Failure

Failure	No	Yes
Measure3		
0	2907	24
1	2897	32
2	2899	25

Ilustración 19 Recuento de Measure3 vs Failure

Esto nos deja la posibilidad que no sean variables explicativas.

Date.year

Todos los valores corresponden al año 2016 por lo que no resulta útil para la creación del modelo, de modo que se procede a su descarte.

Date.Month

Para este atributo directamente tenemos que realizar un cruzamiento con la variable objetivo, para verificar su distribución y si hay alguna posible tendencia.

Failure	No	Yes
Date.month		
1	739	5
2	692	4
3	735	9
4	715	5
5	738	6
6	716	4
7	737	7
8	742	2
9	705	15
10	738	6
11	708	12
12	738	6

Ilustración 20 Recuento de los fallos por meses

Se observa algún repunte de los fallos en determinados meses, aunque a simple vista no parece que sea atribuido al propio mes. Los registros por cada mes son muy similares en número y tampoco se vislumbra patrón alguno.

Date.day-of-month

Esta variable hace referencia al mismo día de todos los meses, al igual que en la característica anterior, directamente tenemos que realizar una tabla de frecuencias cruzada pues, por ejemplo, los días 31 tienen muchas menos apariciones debido a que no todos los meses lo tienen.

Failure	No	Yes
Date.day-of-month		
1	287	1
2	288	0
3	288	0
4	282	6
5	285	3
6	281	7
7	286	2
8	277	11
9	284	4
10	281	7
11	284	4
12	282	6
13	287	1
14	287	1
15	286	2
16	287	1
17	282	6
18	288	0
19	288	0
20	288	0
21	288	0
22	284	4
23	288	0
24	287	1
25	278	10
26	288	0
27	287	1
28	287	1
29	288	0
30	264	0
31	166	2

Ilustración 21 Recuento de fallos por el día del mes

Con tantos valores la información no está muy clara, esta variable se deberá tratar más adelante para su discretización y con posible gráfica de distribución.

Date.day-of-week

Al igual que las dos variables anteriores, realizaremos una representación con tabla cruzada:

Failure	No	Yes
Date.day-of-week		
1	1237	11
2	1237	11
3	1242	6
4	1234	14
5	1266	6
6	1259	13
7	1228	20

Ilustración 22 Recuento de fallos por día de la semana

Se observa como dos días de la semana suelen haber menos averías.

Date.minute y Date.second

Estas variables no contienen valores o son ceros por lo que se descartan.

Failure

Nuestra clase objetivo Failure representa si se produjo fallo o no. Tan solo en el **0.08%** de las apariciones se registraron incidencia. Por lo que la variable se encuentra muy desbalanceada, con pocos casos en los cuales el modelo pueda entrenarse. Debemos ser consciente de las limitaciones que esto puede ocasionar.

col_0	count
Failure	
No	8703
Yes	81

Ilustración 23 Recuento de fallos con respecto a todas las observaciones

3.2 Limpieza de los datos

En este apartado trataremos de detectar los errores cometidos durante el proceso de obtención de datos. Los problemas más frecuentes que se pueden encontrar en esta fase son:

- Errores sintácticos y de normalización.
- En variables cuantitativas errores en la unidad de medida
- Valores atípicos
- Valores ausentes.

Errores sintácticos y de normalización.

No se encuentran errores sintácticos con respecto a las variables cualitativas y tampoco tenemos casos de categorías erróneas.

Se verifica un error comentado con anterioridad con respecto la variable *Operador,* en donde el valor *Operator2* tiene justo el doble de registros que el resto de valores. Podemos suponer que se debe a un doble turno, sería una explicación válida, aunque este hecho afectará por probabilidad a que existan un número mayor de fallos con respecto a los otros valores.

Para tratar este error tenemos varias opciones:

1. Descartar la variable sería el caso más drástico porque perdemos información que puede ser significativa a la hora de realizar el modelo.

2. Dividir el total de observaciones de este valor y crear uno que represente al nuevo conjunto de forma que no se pierda información.

3. No hacer nada.

Para la opción número 2 deberíamos identificar si existe algún patrón con respecto las horas, días del mes o días de la semana.

De todos ellos se visualiza una serie con respecto a las horas y turnos del operador.

Date.hour	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Operator																								
Operator1	122	122	122	122	122	122	122	122	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Operator2	122	122	122	122	122	122	122	122	122	122	122	122	122	122	122	122	0	0	0	0	0	0	0	0
Operator3	0	0	0	0	0	0	0	0	122	122	122	122	122	122	122	122	0	0	0	0	0	0	0	0
Operator4	0	0	0	0	0	0	0	0	122	122	122	122	122	122	122	122	0	0	0	0	0	0	0	0
Operator5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	122	122	122	122	122	122	122	122
Operator6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	122	122	122	122	122	122	122	122
Operator7	122	122	122	122	122	122	122	122	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Operator8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	122	122	122	122	122	122	122	122

Ilustración 24 Observaciones por operador y hora

En el cuadro marcado en rojo se observa un comportamiento diferente al resto de valores.

Se podría dividir creando un nuevo valor *Operator9* con un intervalo de 8 a 15h. Tal y como se muestra en la siguiente imagen:

Date.hour	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Operator																								
Operator1	122	122	122	122	122	122	122	122	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Operator2	122	122	122	122	122	122	122	122	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Operator3	0	0	0	0	0	0	0	0	122	122	122	122	122	122	122	122	0	0	0	0	0	0	0	0
Operator4	0	0	0	0	0	0	0	0	122	122	122	122	122	122	122	122	0	0	0	0	0	0	0	0
Operator5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	122	122	122	122	122	122	122	122
Operator6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	122	122	122	122	122	122	122	122
Operator7	122	122	122	122	122	122	122	122	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Operator8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	122	122	122	122	122	122	122	122
Operator9	0	0	0	0	0	0	0	0	122	122	122	122	122	122	122	122	0	0	0	0	0	0	0	0

Ilustración 25. Observaciones por operador y hora -modificado-

Con respecto a las variables cuantitativas podemos tener problemas con los separadores de decimales, pero en este conjunto de datos no existen valores *float* ni tampoco es utilizado el separador de miles.

Por otro lado, otro típico problema, es no utilizar las mismas unidades por cada variable. En nuestro caso desconocemos que unidades se están utilizando pues no se indica y tampoco podemos averiguarlo.

Por último, queda la estandarización de las que contengan fecha o/y tiempo que son variables cuantitativas con un formato especial. En nuestro conjunto de datos tan sólo la variable *Date* tiene este formato pues las otras variables están descompuestas y discretizadas.
Nuestro variable tiempo conserva el formato en todos los registros por lo que no es necesario una estandarización.

Transformación de las variables

La primera tarea de transformación consistirá en normalizar los datos, de esta forma las variables cuantitativas serán comparables entre sí y evitaremos que atributos con valores altos sesguen los resultados del modelo.

Utilizaremos el método más común denominado normalización *z-score* o por desviación estándar en donde tiene como propiedad que su valor medio es 0 y su desviación típica es 1.

$$Z_i = \frac{x_i - \mu}{\sigma}$$

	Temperature	Humidity	Measure1	Measure4	Measure5	Measure6	Measure7	Measure8
0	1.036574	-0.276488	-1.489386	-0.057093	-0.431083	-1.389624	-0.707771	0.016239
1	1.385168	-1.310404	0.165900	1.572021	0.221667	-0.822181	0.011340	-1.030010
2	-0.009207	-1.517188	0.586704	-1.045000	0.940068	0.084229	1.600074	1.498114
3	-0.357801	-0.690054	-1.007137	1.271921	1.422128	0.631072	0.119114	-0.246254
4	0.339387	-0.483271	1.558650	0.474139	0.011587	-1.578771	0.657983	1.226313

Measure9	Measure10	Measure11	Measure12	Measure13	Measure14	Measure15	Hours Since Previous Failure
-1.549883	0.395491	-0.603252	-0.225645	0.202295	0.496418	1.413231	-0.839194
-0.305871	-1.557808	-1.281801	-0.685075	0.690072	-0.687423	-0.622129	-0.832604
0.410891	-1.505720	1.566981	0.065015	-0.672339	0.799824	1.128579	-0.826014
0.431531	0.765687	1.245465	0.648211	-1.074148	0.461051	-0.690966	-0.819423
-0.091968	0.868003	-0.535958	1.370172	1.007781	0.779348	-1.070503	-0.812833

Ilustración 26. Normalización de variables cuantitativas

En siguiente paso consiste en la discretización(binning), en donde los valores de una variable los dividiremos en intervalos para que haya un número limitado de estados y los trataremos como categorías.
Las variables susceptibles a descretización son:

Date.hour. Dividiremos las horas en tres intervalos correspondientes a los turnos vistos con anterioridad.

Date.hour	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Turno																								
Noche	366	366	366	366	366	366	366	366	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mañana	0	0	0	0	0	0	0	0	366	366	366	366	366	366	366	366	0	0	0	0	0	0	0	0
Tarde	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	366	366	366	366	366	366	366	366

Ilustración 27 Recuento de horas según el turno

Date.day-of-month. Realizaremos dos intervalos correspondientes a la primera y segunda quincena del mes.

Date.day-of-month	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Quincena															
Primera	288	288	288	288	288	288	288	288	288	288	288	288	288	288	288
Segunda	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
288	288	288	288	288	288	288	288	288	288	288	288	288	288	264	168

Ilustración 28 Recuento de días según quincena

Seguidamente buscaremos si existen valores duplicados en los registros.

```
duplicateRowsDF = dataset[dataset.duplicated()]
print(duplicateRowsDF)
```

```
Empty DataFrame
```

Ilustración 29 Comprobación de valores duplicados

Podemos comprobar que no existen valores duplicados en el conjunto de datos.

Inconsistencias, valores atípicos, nulos o perdidos.

Para realizar estas comprobaciones haremos uso de los diagramas de caja. También de determinadas funciones para encontrar valores nulos o asuntes.

```
dataset.isnull().values.any()
```

```
False
```

Ilustración 30 Comprobación de valores nulos

Podemos comprobar que no existen valores nulos.

Ahora se realizará los diagramas de caja y se identificará posibles valores atípicos. En consecuencia, se decidirá si mantenerlos o eliminarlos:

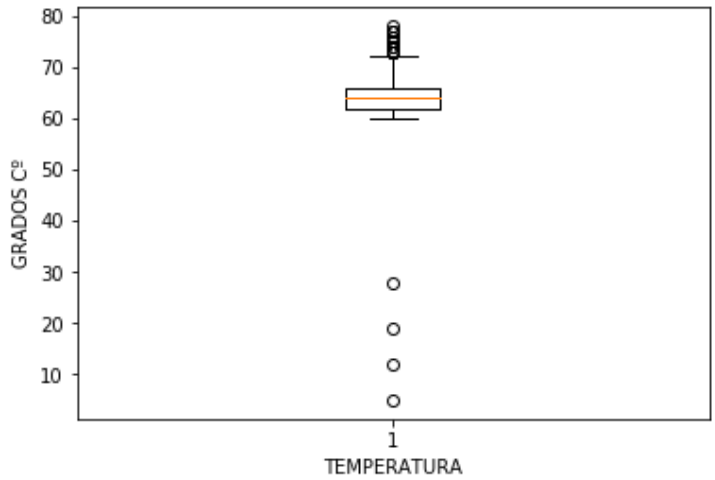


Ilustración 31 Diagrama de caja. Temperatura

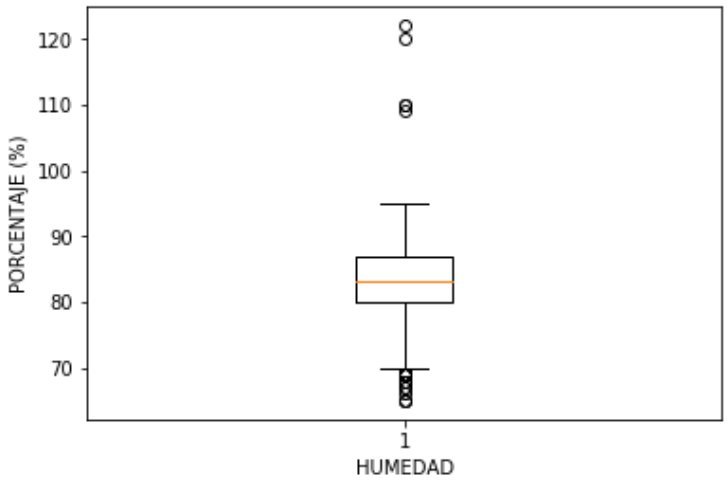


Ilustración 32 Diagrama de caja. Humedad

La variable *Temperatura* y *Humedad*, contiene valores atípicos, debemos comprobar si estos se han dado por la propia naturaleza de los datos o se trata de un error. Para comenzar veremos la secuencia de como varia la temperatura.

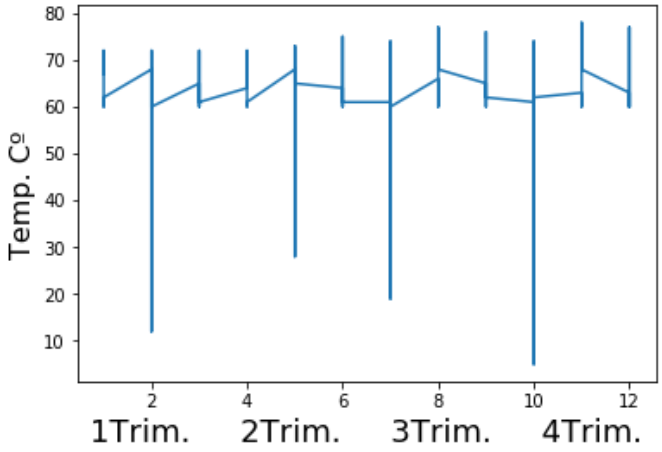


Ilustración 33 Variación de la temperatura respecto al tiempo en trimestre

Se observan cuatro picos en donde parece que los valores están fuera de rango y tampoco siguen una tendencia.

Date	Temperature
07/10/2016 0:00	5
28/02/2016 0:00	12
25/07/2016 0:00	19
18/05/2016 4:00	28

Después de observar los valores cercanos se constata que ha sido un problema de error de lectura del dato, pues la tendencia no era un descenso tan brusco, además las bajadas repentinas no coinciden con los fallos o parada de la máquina.

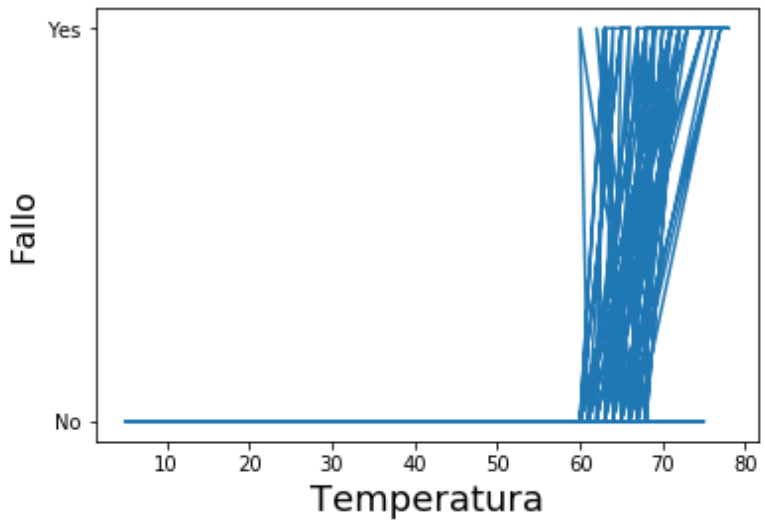


Ilustración 34 Correlación entre temperatura e incidencia

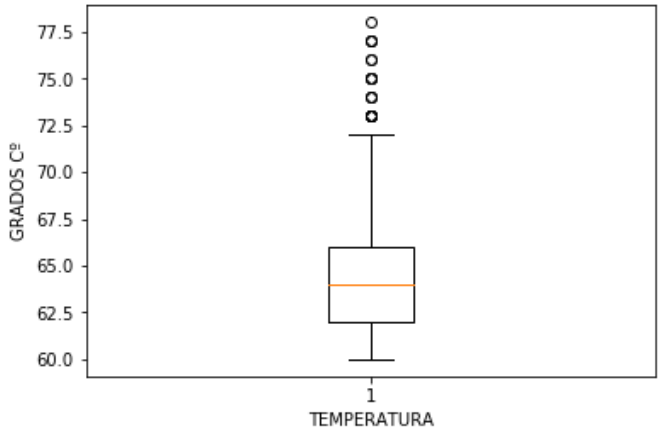


Ilustración 35. Diagrama de caja Temperatura sin valores atípicos

Una vez aplicado los cambios podemos constatar como ya no aparecen los valores atípicos por debajo del umbral de 60. Los valores por encima de 70, sí que son propios del funcionamiento de la máquina y se mantendrán.

Con respecto a la humedad, fácilmente se identifica valores superiores a 100% y que no son posibles, deben tratarse como errores de lectura. Se procede a sustituir estos valores por 95 que es el valor máximo dentro de los segmentos llamados “bigotes”.

Date	Humidity
06/07/2016 13:00	122
01/12/2016 15:00	120
03/01/2016 12:00	110
04/10/2016 0:00	110
17/04/2016 9:00	109

El resultado después de aplicar los cambios se puede visualizar en la siguiente caja de diagrama:

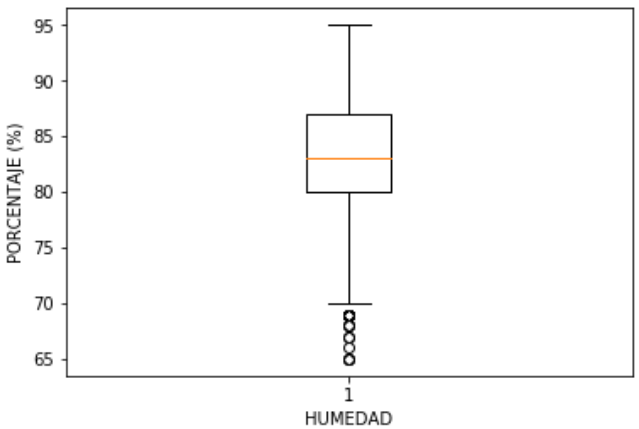


Ilustración 36. Diagrama de cajas Humedad sin valores atípicos

Por lo que respecta a la variable “Horas desde la última incidencia” y el resto de mediciones (13). Ninguna presenta valores atípicos.

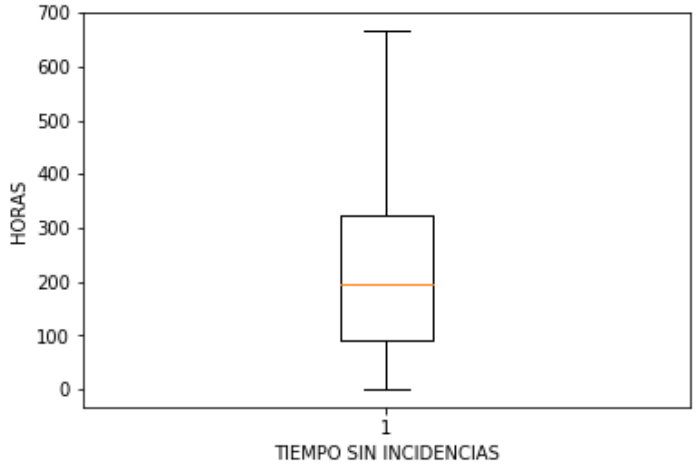


Ilustración 37. Diagrama de cajas Tiempo en horas sin incidències

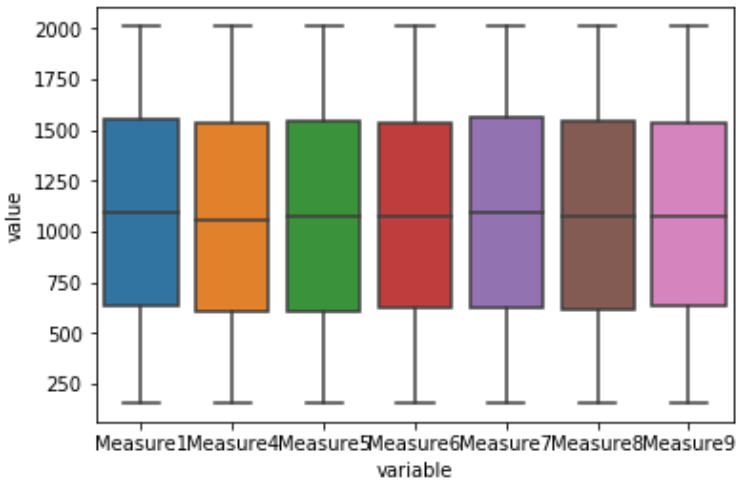


Ilustración 38 Diagramas de caja. Mediciones 1-9

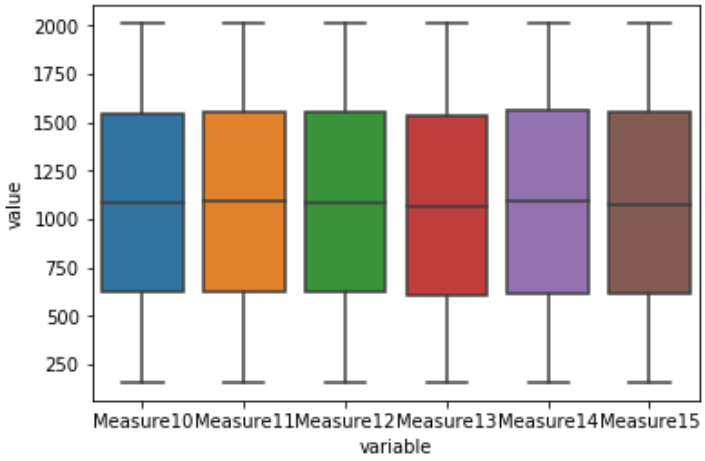


Ilustración 39 Diagrama de cajas. Mediciones 10-15

3.3 Reducción de dimensionalidad

La reducción es una operación que se lleva a cabo con tal de que nuestro modelo pueda tratar la cantidad de datos de una forma óptima, tanto por coste computacional y por tiempo de procesado.

Existen dos estrategias básicas a la hora de reducir nuestros datos:

- Reducción del número de registros.
- Reducción del número de variables.

Nuestro conjunto de datos no dispone de demasiados registros (8784), por lo que no es recomendable reducirlo así que este aspecto lo mantendremos. Aunque como se verá más adelante, el hecho de tener un clase objetivo desbalanceada propiciará el uso de resampleo.

En el caso del número de variables sí, que podríamos aplicar algún tipo de reducción, pues ahora mismo, disponemos de 30 atributos. La técnica más popular es la de análisis de componentes principales (ACP) o principal component analysis (PCA), en inglés.

Primero, buscaremos los componentes principales mediante una matriz de covarianza que nos de la dispersión entre variables, a partir de ahí calcularemos los autovectores que son las direcciones donde la varianza de los datos es mayor.

Autovalores en orden descendiente:

```
303626.4821174431
300444.6688593852
298865.87836350757
297005.815340992
290335.2168403182
288634.3116094872
286046.9434753663
282524.4890495928
281041.995766932
279957.81482023885
277199.0829297144
273684.65905428247
```

```
272258.19938582357
23002.411930163016
22.856738580617098
7.114499655753117
```

Para reducir la dimensionalidad sin perder demasiada información, debemos descartar los autovectores con valores bajos que no aportan mucha información. Para ello después de su ordenación, veremos la varianza explicada y se decidirá cuáles son los componentes principales.

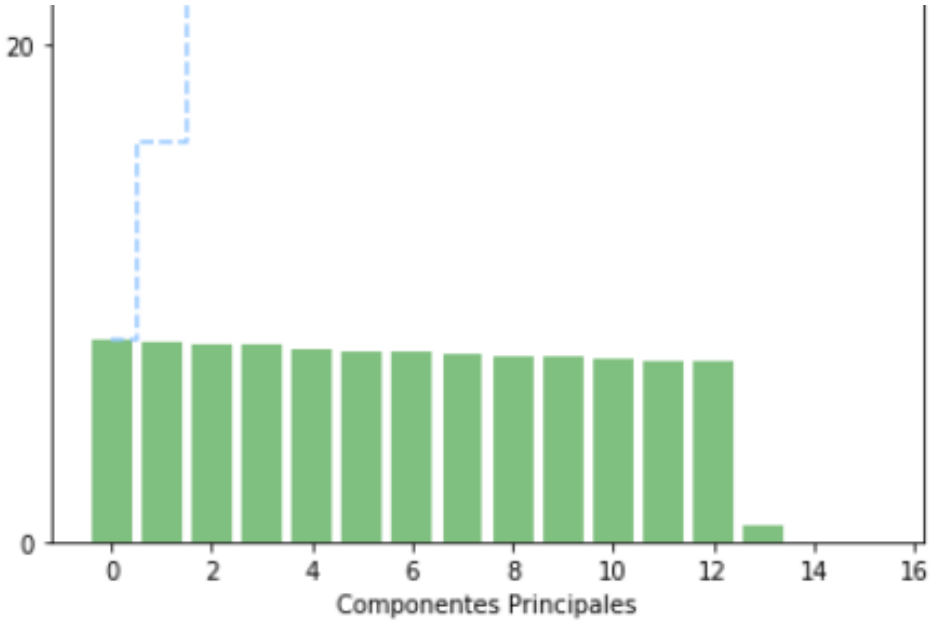


Ilustración 40 Gráfica de barra para componentes principales

Lamentablemente, se observa como ningún componente aporta demasiada información, tampoco destaca ninguno sobre los demás.

Otra metodología para medir la importancia de las variables es la de utilizar el algoritmo *RamdonForest*, que trabaja de forma que cada uno de los arboles es entrenado con un subconjunto diferente de variables.

Se realizará indicando que *Failure* es la clase objetivo.

```
clf = ensemble.RandomForestClassifier(n_estimators=80)
clf.fit(X, y)

# cálculo de la importancia de las variables y representación gráfica:
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1]
feature_names = X.columns
f, ax = plt.subplots(figsize=(11, 9))
plt.bar(range(X.shape[1]), importances[indices], color="b", align="center")
plt.xticks(range(X.shape[1]), indices)
plt.xlim([-1, X.shape[1]])
plt.ylabel("Importancia", fontsize = 18)
plt.xlabel("Índice de variable", fontsize = 18)
plt.show()

important_features = pd.Series(data=clf.feature_importances_,index=X.columns)
important_features.sort_values(ascending=False,inplace=True)
```

Ilustración 41 Algortimo Random Forest para PCA

Obtenemos la siguiente lista con la importancia de cada variable:

```
Humidity                        0.314660
Temperature                     0.257007
Hours Since Previous Failure    0.149133
Measure10                       0.027369
Measure14                       0.026284
Measure1                        0.024966
Measure15                       0.023924
Measure8                        0.021765
Measure11                       0.021082
Measure9                        0.021003
Measure13                       0.020159
Measure7                        0.019743
Measure5                        0.019326
Measure4                        0.018681
Measure6                        0.018527
Measure12                       0.016371
```

En este caso sí, que se observa que las variables *Humedad* y *Temperatura* destacan por encima de las otras. En un tercer puesto también se encuentra la variable de *Horas Desde el Último Fallo*, tiene su lógica pues cuanto más tiempo transcurrido mayor probabilidad de que se vuelve a dar un error.

Esta diferencia todavía es más visible en su representación gráfica:

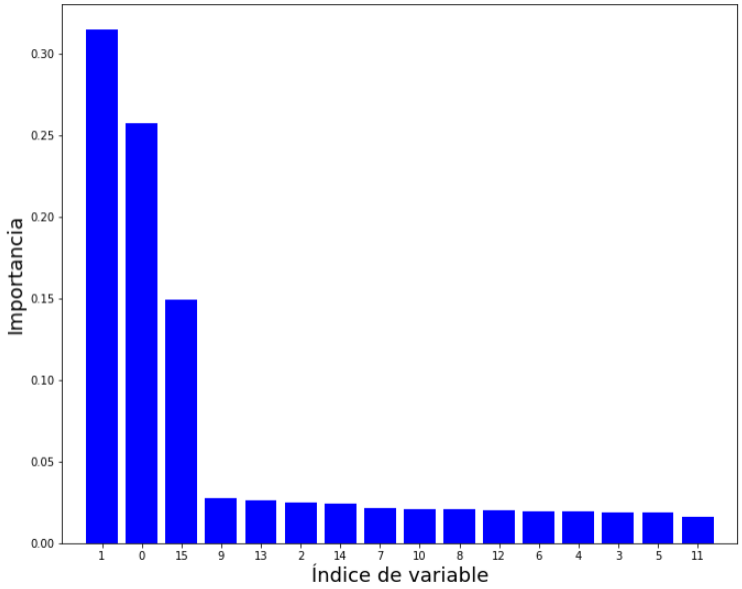


Ilustración 42 Gráfico de barras según el peso de cada variable

Visto los resultados se crearán los modelos sin las variables de mediciones, aunque no se eliminarán y se tendrán en cuenta para una segunda propuesta y así compararlos ambos modelos.

Por último, ordenaremos el dataset y renombraremos la selección de variables.

	Operador	Temperatura	Humedad	Sensor_A	Sensor_B	Horas sin incidencia	Mes	Quincena	Día semana	Turno	Falla
0	Operator1	67	82	1	1	90	1	Primera	5	Noche	No
1	Operator1	68	77	1	1	91	1	Primera	5	Noche	No
2	Operator1	64	76	1	1	92	1	Primera	5	Noche	No
3	Operator1	63	80	1	1	93	1	Primera	5	Noche	No
4	Operator1	65	81	1	2	94	1	Primera	5	Noche	No

Ilustración 43 las cinco primeras filas del conjunto modificado

Por último, debemos convertir las variables categóricas a una matriz de enteros para que puedan ser tratadas por los algoritmos.
Esto no altera el concepto que representa dichas variables.

	Operador	Temperatura	Humedad	Sensor_A	Sensor_B	Horas sin incidencia	Mes	Quincena	Día semana	Turno	Falla
0	1	67	82	1	1	90	1	1	5	0	0
1	1	68	77	1	1	91	1	1	5	0	0
2	1	64	76	1	1	92	1	1	5	0	0
3	1	63	80	1	1	93	1	1	5	0	0
4	1	65	81	1	2	94	1	1	5	0	0

Ilustración 44 Transformación de variables categóricas a matrices de enteros

4. Algoritmos de Machine Learning

4.1 Conjuntos de entrenamiento y test.

Un punto en común en la mayoría de modelos, es la creación de los conjuntos de entrenamiento y test. Esto es sumamente importante para evitar el sobreentrenamiento que sucede cuando nuestro modelo predice a la perfección nuestro conjunto actual de datos, pero en cambio no va generalizar nada bien ante nuevos datos por lo que no acaba siendo un modelo válido.

La forma de evitar que ocurre esto, procede separando el conjunto de datos en dos subconjuntos. Uno mayor que corresponde aproximadamente al 80% del original y que será el conjunto que usaremos para entrenar el modelo. El 20 % restante se utilizará para medir la precisión del modelo creado.

Además, se procederá también a separar la variable objetivo del resto para el entrenamiento.

Debemos recordar que nuestra clase objetivo está fuertemente desbalanceada por lo que debemos utilizar la opción *stratify* de la librería scikit-learn. De manera que dividirá nuestro conjunto de entrenamiento y test de manera que se mantenga la proporción de valores negativos y positivo.

```
y=df.Falla
x=df.drop('Falla',axis=1)
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=2019, stratify=y)
```

Ilustración 45 División del dataset en clase objectivo, conjuntos de entrenamiento y test

```
(7027, 10) → conjunto de entrenamiento X
(1757, 10) → conjunto de test X

(7027,) → conjunto de entrenamiento clase objetivo
(1757,) → conjunto de test clase objetivo
```

Cada modelo tendrá unos hiperparámetros que se deberán ajustar de forma que obtengamos el mejor resultado posible. Para ello se utilizará la búsqueda en rejilla (GridSearch) que nos proporciona también la librería. De forma que se entrena cada modelo con la combinación de parámetros posibles y posteriormente se evalúa con validación cruzada, obteniéndose la mejor combinación.

Por otra parte, la validación cruzada divide el conjunto de datos en K subconjuntos del mismo tamaño, usando k-1 de los conjuntos para entrenar el modelo y otro para evaluarlo, este proceso se repite según el valor escogido de k y promediando el error estimado.

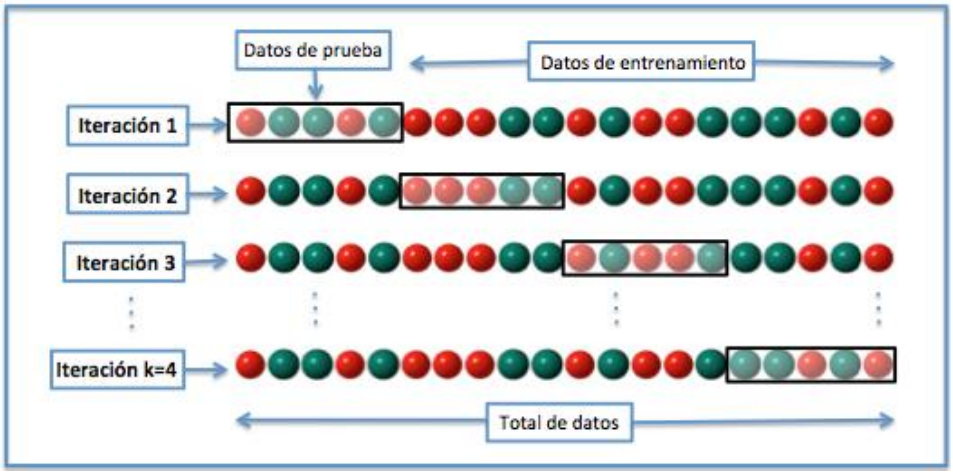


Ilustración 46 Iteraciones en la validación cruzada
Wikimedia Commons

4.2 Problema con datos desbalanceados

El conjunto de datos está fuertemente desbalanceado, esto es un problema para la evaluación del modelo pues la exactitud tendrá puntuaciones altas pero que como veremos serán engañosas.

Los algoritmos clasificadores son sensibles a las proporciones de clases y tienden a favorecer aquellas que están fuertemente representadas.

En nuestro estudio, la clase que representa *Falla* supone el 99,07% de los casos.

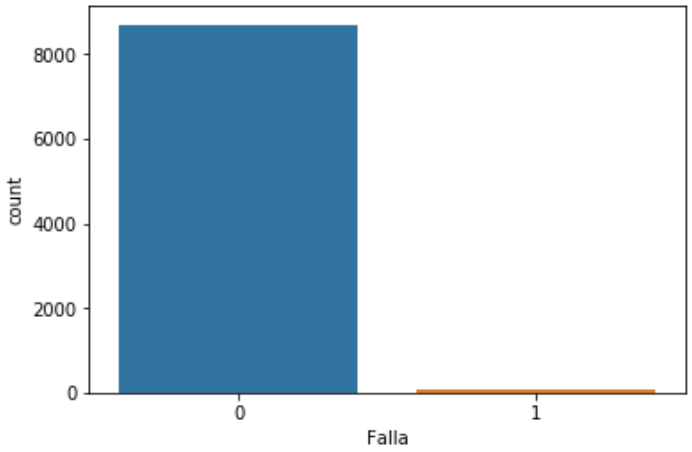


Ilustración 47 Gráfico de barra variable objetivo

Si evaluamos nuestro modelo con la métrica de *Accuracy (exactitud)* tendremos resultados también del 99% pues verdaderamente, nuestro modelo clasificará correctamente la clase mayoritaria que es un reflejo de su distribución, pero esto no será de utilidad. Debido a que, gran parte de los clasificadores, minimizan la función que representa el error que se produce, por lo que el algoritmo tenderá a minimizar esta clase negativa que representa la casi totalidad de los datos.

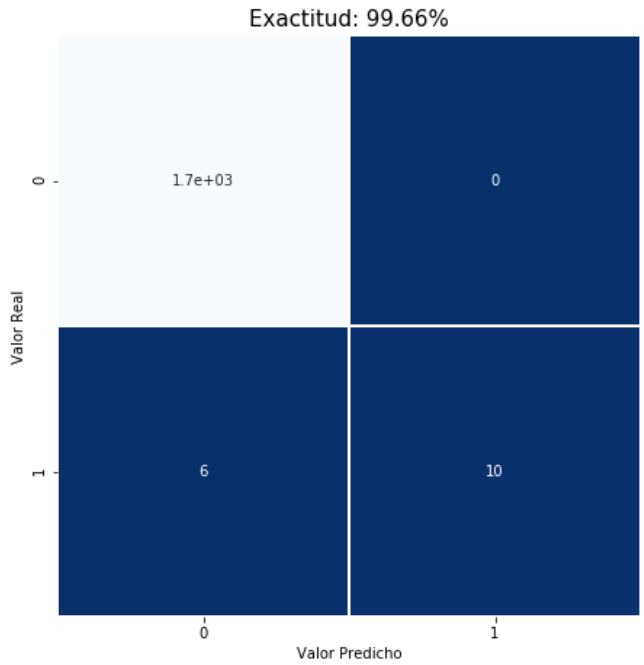


Ilustración 48 Matriz de confusión modelo regresión logística

Para solucionar este problema contamos con varias estrategias:

- Tratamientos con métodos de resampleo: submuestreo, sobremuestreo, generación de datos fabricados y aprendizaje sensible al costo.

- Uso de otras métricas como la precisión, sensibilidad, F-Score o curvas ROC son preferibles en contraposición a la métrica exactitud.

Utilizaremos la técnica de sobremuestreo (oversampling) que se emplea para incrementar el número de observaciones positivas que corresponde a la clase minoritaria. Existen varios métodos para realizar dicha tarea, aunque aquí se utilizará el muestreo con remplazo. En concreto se utilizará el algoritmo SMOTE debido a que está ampliamente estudiado, además de ser el más utilizado.
No obstante, también presenta algunos inconvenientes como son la generación de ruido o un menor rendimiento para variables discretas. [30]

Por ello, se usará una variante del algoritmo que realiza una limpieza utilizando los enlaces Tomek:

```
smt = SMOTETomek(ratio='auto')
x_smt, y_smt = smt.fit_sample(x, y)
```

Ilustración 49 Algoritmo SMOTE

Los datos de entrenamiento iniciales eran de 7027 registros, pero después de aplicar el algoritmo SMOTE estos aumentaron hasta los 13916, es decir casi hemos doblado los números de registros.
Y sin duda hemos aumentado equiparando la clase minoritaria a la clase mayoritaria haciendo ambas equivalentes con un total de 6958 registros para cada una.

```
import collections
contar = y_train
collections.Counter(contar)
```

Ilustración 50 Nuevo recuento de observaciones

Se debe tener en cuenta que, aunque se habló en un primer momento de la división de los datos en los conjuntos de entrenamiento y test, es necesario primero aplicar las técnicas de resampling y luego ya con el nuevo conjunto crear las divisiones.

Antes de proceder aplicar los algoritmos, tenemos que normalizar todos los datos de las variables categóricas que se habían transformado a matriz de enteros:

```
X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)
```

Ilustración 51. Normalización del resto de variables.

4.3 Modelo Regresión Logística

El modelo de regresión logística es similar al de regresión lineal con la diferencia que este actúa clasificando las instancias según la probabilidad de pertenecer alguno de los valores de nuestra variable independiente, siempre encontrándose en la franja de 0 al 1.

```
hyperparameters = dict(C=[0.001,0.01,0.1,1,10,100], penalty=['l1', 'l2'])

clf = GridSearchCV(LogisticRegression(), hyperparameters, cv=5, n_jobs=-1, scoring='f1')

clf.fit(X_train, y_train)
```

Ilustración 52 Modelo Regresión Logística

Después de aplicar el GridSearch obtenemos que los mejores hiperparámetros para el modelo han sido con una regularización L1 más penalización de 0.01 aunque también se han obtenidos valores de penalización de 100.

```
print('Best Penalty:', clf.best_estimator_.get_params()['penalty'])
print('Best C:', clf.best_estimator_.get_params()['C'])
```

```
Best Penalty: l1
Best C: 0.01
```

Ilustración 53. Mejores parámetros para el modelo de Regresión Logística

4.4 Modelo k vecinos más cercanos (*k*-NN)

Este algoritmo abreviado como k-NN a diferencia del resto no se genera después del entrenamiento con el conjunto homónimo, sino que lo hace en el mismo instante que se está evaluando la clasificación de una nueva instancia. Es el tipo de aprendizaje denominado vago (lazy learning)

Los parámetros idóneos para este modelo son la adecuada elección de K, que representa al número de instancias más cercanas y, por otro lado, la forma en cómo se medirá los pesos para la distancia entre instancias.

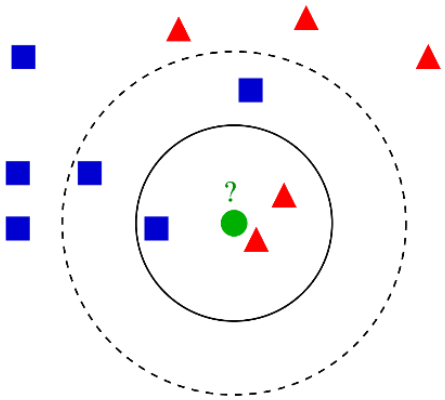


Ilustración 54 Ejemplo del algoritmo Knn.
El ejemplo que se desea clasificar es el círculo verde

En este caso los hiperparámetros obtenido son para *K: 2* y para la distancia: *uniform* o de pesos uniformes, es decir, todos los puntos en cada barrio se ponderan por igual.

```
hyperparameters = {"n_neighbors": range(1, 11), "weights": ["uniform", "distance"]}

clf = GridSearchCV(KNeighborsClassifier(), param_grid=hyperparameters, cv=5,n_jobs=-1, scoring='f1')

clf.fit(X_train, y_train)
```

Ilustración 55 Modelo k-NN

4.5 Modelo Random Forest (Bosques Aleatorios)

El Random Forest funciona como una combinación de árboles predictores, en donde cada árbol actuará bajo un vector independiente y manteniendo la distribución por cada uno de estos.

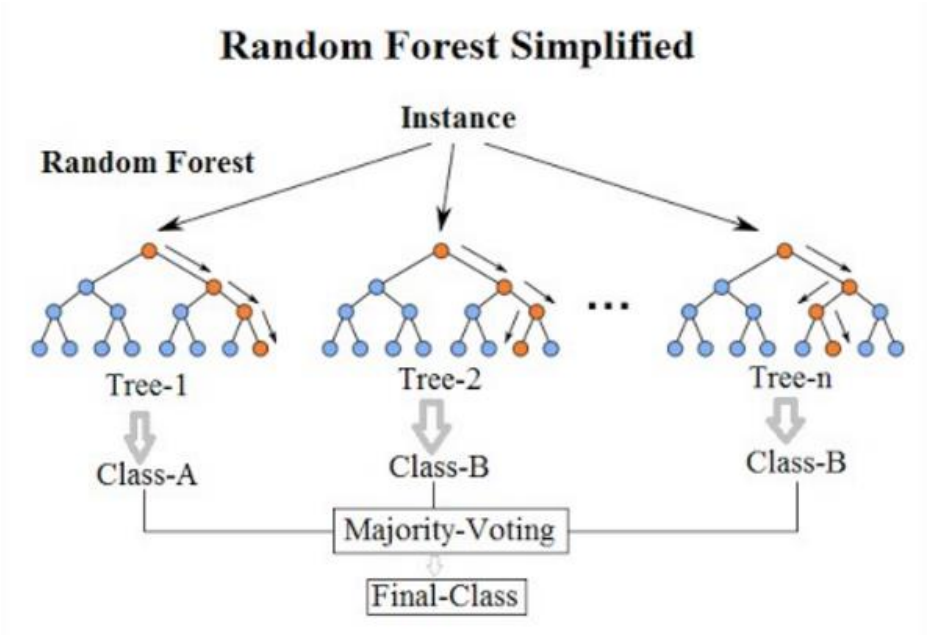


Ilustración 56 Ejemplo de algoritmo RandomForest
Wikimedia Commons

Se trata de unos de los algoritmos más precisos que existen actualmente si se dispone de un conjunto de datos lo suficientemente grande.

```
hyperparameters = {
    'n_estimators'      : [50,100,200],
    'max_depth'         : [8, 10, 15, 20],
    'random_state'      : [2019],
}

clf = GridSearchCV(RandomForestClassifier(), hyperparameters, cv=5, n_jobs=-1,scoring='f1'

clf.fit(X_train, y_train)
```

Ilustración 57 Modelo RandomForest

Después de aplicar GridSearch, crearemos un bosque aleatorio con 100 árboles, con una profundidad máxima de 15.

```
clf.best_params_
```

```
{'max_depth': 15, 'n_estimators': 100, 'random_state': 2019}
```

Ilustración 58. Mejores parámetros del modelo RandomForest

4.6 Modelo Máquina de Soporte Vectorial (SVM)

Las Máquinas de soporte vectorial son algoritmos capaces de resolver problemas lineales como no lineales. Las observaciones se representan en el espacio y se separan por clases a dos espacios lo más distantes posibles mediante un hiperplano. Posteriormente, para las nuevas observaciones se clasificarán según al espacio que pertenezca.

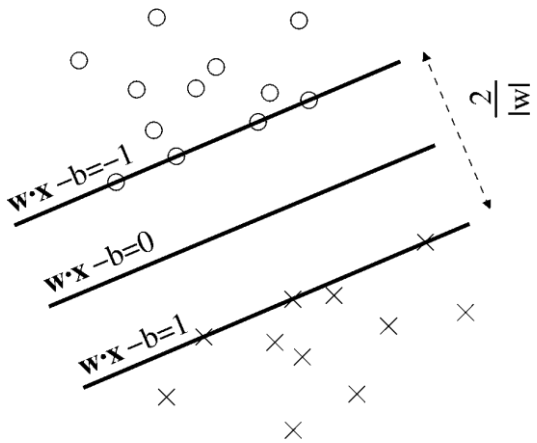


Ilustración 59 El hiperplano de margen máximo y los márgenes para un SVM entrenado con muestras de dos clases. Wikimedia Commons

```
hyperparameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
                    'C': [1, 10, 100, 1000]},
                   {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]
clf = GridSearchCV(SVC(), hyperparameters, cv=5,n_jobs=-1,scoring='f1')

clf.fit(X_train, y_train)
```

Ilustración 60 Modelo SVM

Los hiperparámetros ideales para el modelo son kernel *RBF, gamma: 0.001 y C: 1000*

```
clf.best_params_
```

```
{'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
```

Ilustración 61. Mejores parámetros para el model SVM

4.7 Modelo Naive Bayes

Basado en el teorema de Bayes, conocido como teorema de clasificación condicionada. Es un clasificador ingenuo y asume que la presencia o no de una característica no está relacionada con la también existencia o no de otra característica distinta. La ventaja de este algoritmo es que no requiere de un conjunto grande para poder entrenar.

```
clf = GaussianNB()
clf.fit(X_train, y_train)

predGNB = clf.predict(X_test)
```

Ilustración 62. Modelo Naive Bayes

En este modelo no se han utilizado hiperparámetros para afinar.

4.8 Modelo Red Neuronal Artificial

Las redes neuronales son un símil a las neuronas biológicas, donde un conjunto de nodos (neuronas) agrupadas en capas, se conectan unas con otras a través de sus enlaces y cada una con diferentes pesos.

El algoritmo se va autoajustando calculando el error en la salida e intentando minimizar la función de perdida.

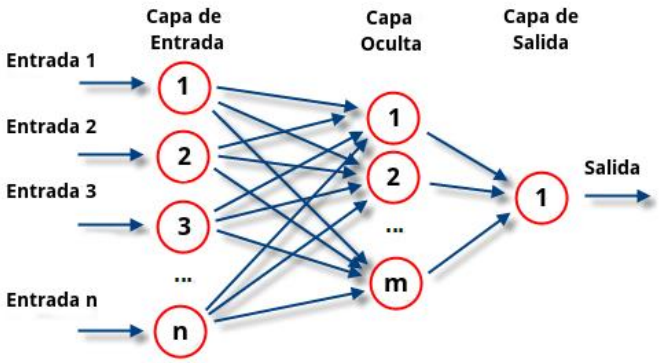


Ilustración 63 Diseño de una Red Neuronal de 3 capas
Wikimedia Commons

Para este modelo existen varios parámetros a tener en cuenta:

- Número de neuronas por capa: **50 (entrada), 10 (oculta), 1 (salida)**
- Número de capas: **3**
- Número de épocas: **10**
- Tamaño del lote: **15**
- Optimizador: **Adam**
- Función de perdida: **binary crossentropy**
- Tasa de aprendizaje: **0.01**
- Función de activación: **relu** y **sigmoid**

```
optimizer = keras.optimizers.Adam(lr=0.01)

model = Sequential()
model.add(Dense(50, input_dim=X_train.shape[1], kernel_initializer='normal', activation='relu'))
model.add(Dense(10, kernel_initializer='normal', activation='relu'))
model.add(Dense(1, kernel_initializer='normal', activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
```

Ilustración 64 Modelo de Red Neuronal Artificial

4.9 Combinación de Modelos (Stacking)

Para finalizar, se procederá a realizar una combinación secuencial de todos los clasificadores utilizados y que son de base diferente. Se generará un nuevo modelo *Stacking*, que será entrenado a partir de las combinaciones predichas por los modelos anteriores.

```
X_test_stacking = np.column_stack((predNNC, predSVM, predRFC, predKNN, predLR, predGNB))

clf = ensemble.RandomForestClassifier(n_estimators=100)

cvscores = cross_val_score(clf, X_test_stacking, y_test, cv=10,scoring='f1')
```

Ilustración 65 Ensamblaje de varios modelos

5. Evaluación de los modelos

5.1 Métricas

Como se comentó anteriormente, es común para la evaluación de modelos en los algoritmos de clasificación, utilizar la métrica de *exactitud*. Pero en los casos donde la clase objetivo está muy desbalanceada esta métrica no es válida y nos pueda llevar a error.

Para calcular la exactitud utilizaríamos la siguiente ecuación:

$$exactitud = \frac{VP + VN}{VP + FP + FN + VN}$$

Donde,

- VP corresponden a los verdaderos positivos.
- VN corresponden a los verdaderos negativos.
- FP corresponden a los falsos positivos.
- FN corresponden a los falsos negativos.

Como los modelos priorizarán en minimizar el error en alguna de las clases, lo harán propiamente en la negativa pues es la mayoritaria, como es en este caso de estudio. El clasificador que predice siempre 0 tendrá una exactitud aproximadamente al 100% pues prácticamente, el total de las observaciones pertenecen a esta clase.

Por lo que para conjuntos asimétricos es mejor usar métricas como la precisión y sensibilidad o una métrica combinada de ambas como es el valor-F.

$$precisión = \frac{VP}{VP + FP}$$

$$sensibilidad = \frac{VP}{VP + FN}$$

$$F1 = \frac{precisión \cdot sensibilidad}{precisión + sensibilidad}$$

5.2 Resultados

Resultados sin técnicas de remuestreo:

Modelo	Exactitud	Precisión	Sensibilidad	F1 score
Reg. Logística	0.99	0.92	0.69	0.79
k-NN	0.99	0.90	0.56	0.69
RandomForest	0.99	0.85	0.69	0.76
SVM	0.99	0.67	0.62	0.65
Naive Bayes	0.99	1	0.62	0.77
NNC	0.99	1	0.32	0.47
Stacking	0.99	-	-	-

Como se puede observar la exactitud en todos los modelos es del 99% pero ya se ha comentado que esta métrica es engañosa debido al fuerte desbalanceo que encontramos en los datos.

Sin haber realizado un resampleado, nos encontramos que nuestra variable objetivo tiene muy pocos casos positivos comparados con la totalidad de las observaciones. Una métrica más realista sería la precisión y sensibilidad, pero, sobre todo, F1 score que en este caso sí muestra valores más realistas. Observamos que el modelo que mejor ha funcionado ha sido la regresión logística. En contraposición, las redes neuronales han tenido un desempeño muy bajo no llegando al 50%.

Otra métrica alternativa es el uso de curvas ROC. Una curva ROC representa la tasa de verdaderos positivos frente a falsos positivos en diferentes umbrales de clasificación. Al reducir ese umbral se clasificará más elementos como positivos.

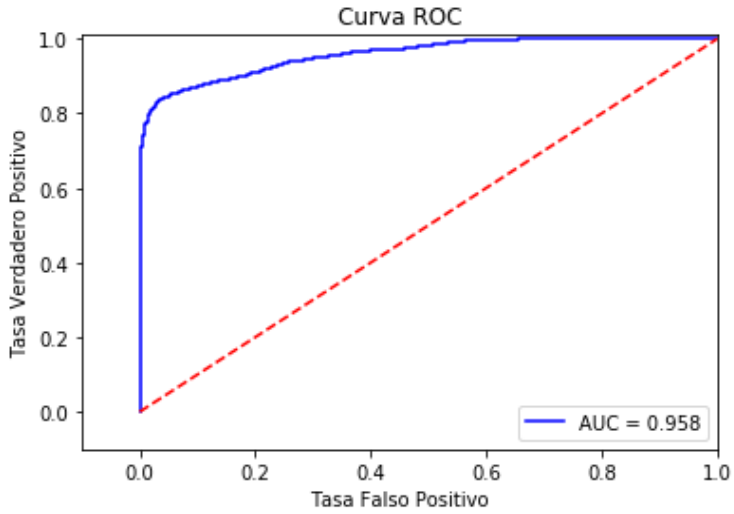


Ilustración 66 Curva ROC de la Regresión Logística

Como se observa, en el modelo regresión logística la curva ROC ha obtenido un valor de 0.958 en una de las pruebas, por lo que sería catalogado como un test muy bueno.

Ahora veamos, los resultados aplicando técnicas de sobremuestreo:

Modelo	Exactitud	Precisión	Sensibilidad	F1 score
Reg. Logística	0.88	0.91	0.86	0.88
k-NN	0.98	0.98	1	0.99
RandomForest	0.99	1	1	1
SVM	0.99	0.99	1	1
Naive Bayes	0.89	0.95	0.84	0.89
NNC	1	0.99	1	1
Stacking	1	-	-	-

También veamos el detalle de los resultados de la red neuronal y como se observa que ya en la primera época, la exactitud se encuentra por encima del 99% y el conjunto de entrenamiento y test siguen la misma tendencia.

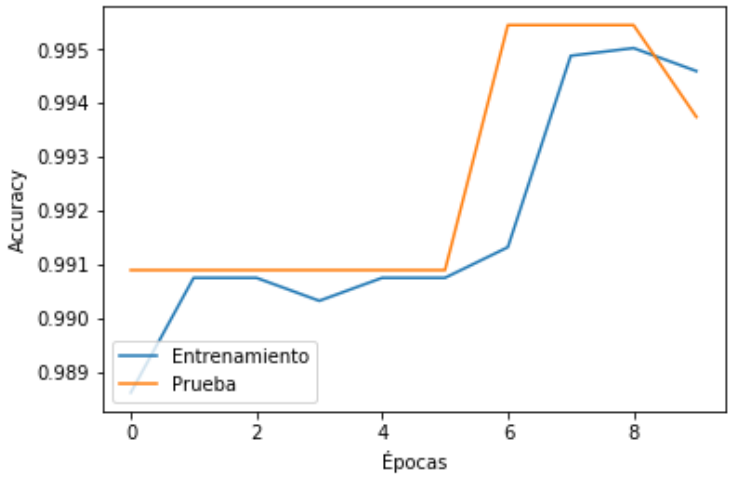


Ilustración 67 Exactitud según Épocas

Aplicando la técnica SMOTE se obtienen resultados muy satisfactorios, excepto para la Regresión Logística y Naive Bayes que fueron mejores, pero sin llegar a la excelencia.

Por otro lado, unos resultados tan buenos, puede hacernos pensar que la técnica de remuestreo no está realizando el procedimiento todo lo correcto que podría esperarse y que las nuevas observaciones en donde la variable objetivo es positiva es fácilmente de clasificar.

Para evitar que puede ser el caso utilizaremos una técnica diferente de resampling que trabaja con submuestreo pero antes, veamos la comparativa utilizando el grueso de variables de mediciones que se habían descartado en un principio.

Resultados incorporando todas las mediciones y con técnicas de sobremuestreo:

Modelo	Exactitud	Precisión	Sensibilidad	F1 score
Reg. Logística	0.91	0.93	0.90	0.91
k-NN	0.98	0.97	1	0.98
RandomForest	0.92	0.87	1	0.93
SVM	0.99	0.99	1	1
Naive Bayes	0.92	0.97	0.88	0.92
NNC	1	0.99	1	1
Stacking	1	-	-	-

Este caso es igual que al anterior, pero se incluyeron todas las mediciones que se descartaron en el análisis de componentes principales, éstas correspondían al resto de medidas del tipo cuantitativas.

Vemos como efectivamente, el hecho de no utilizarlas no ha repercutido en la calidad de los modelos, incluso algunas métricas tienen una puntuación ligeramente inferior.

Por último, tenemos los resultados para dos técnicas de submuestreo diferentes, en donde se ha reducido el número de observaciones para equiparar los valores de la clase objetivo. La primera técnica es por generación de prototipos, mientras que la segunda es por selección.

Como las observaciones son tan escasas los hiperparámetros escogidos para los modelos anteriores no son válidos aquí, por lo que de nuevo se hará uso de la herramienta GridSearch.

Resultados con técnica de submuestreo generación de prototipos:

Modelo	Exactitud	Precisión	Sensibilidad	F1 score
Reg. Logística	0.93	1	0.88	0.94
k-NN	0.96	1	0.94	0.97
RandomForest	0.97	1	0.94	0.97
SVM	0.97	1	0.94	0.97
Naive Bayes	1	1	1	1
NNC	0.97	1	0.94	0.97
Stacking	0.97	-	-	-

Esta técnica de generación de prototipos reduce el número de muestras en las clases seleccionadas, pero las muestras restantes se generan y no se seleccionan del conjunto original.

El conjunto de entrenamiento se ha visto reducido a 129 observaciones. Puede que, debido a la generación de muestras y lo escaso de las observaciones, todas las métricas

obtengan unos resultados muy similares. En general las puntuaciones oscilan entorno a las observadas con la técnica del sobremuestreo.

En cambio, con técnica de submuestreo por selección de prototipos, obtenemos los siguientes resultados:

Modelo	Exactitud	Precisión	Sensibilidad	F1 score
Reg. Logística	0.90	1	0.82	0.90
k-NN	0.90	1	0.82	0.90
RandomForest	0.93	0.94	0.94	0.94
SVM	0.90	1	0.82	0.90
Naive Bayes	0.90	1	0.82	0.90
NNC	0.93	1	0.88	0.93
Stacking	0.92	-	-	-

El algoritmo ha reducido aleatoriamente los valores negativos de la clase objetivo para equipararlos a los positivos. Es comprensible que, en ciertos casos, esta selección aleatoria no haya sido muy favorable para los conjuntos de entrenamiento y prueba.

5.3 Visualización

En el siguiente apartado se procede a visualizar la comparativa de modelos vista en las tablas anteriores, pero ahora en formato de diagrama de cajas. Estas representaciones gráficas nos describen características tan significativas como son el mínimo, máximo y los tres cuartiles. Los diagramas de cajas son muy útiles a la hora de comparar distribuciones, en este caso se trata de las puntuaciones en las métricas F1 llevadas a cabo en diferentes pruebas.

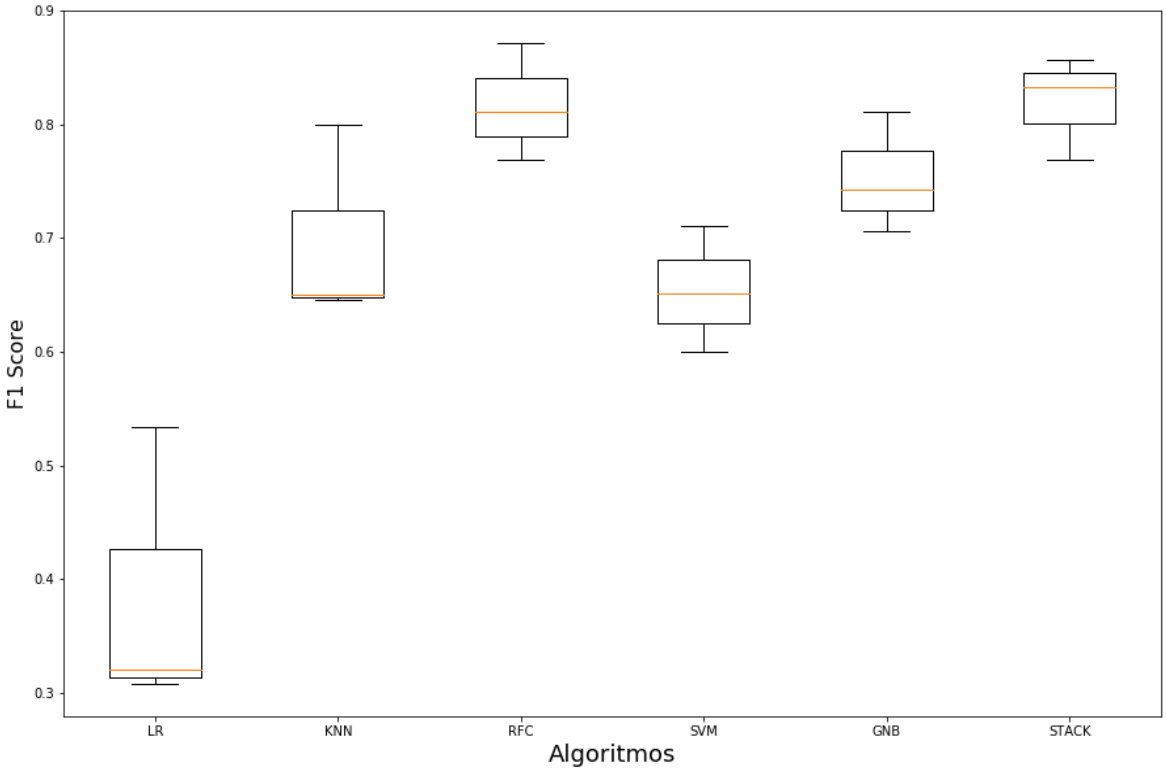


Ilustración 68. Comparación de modelos sin técnicas para desbalanceo

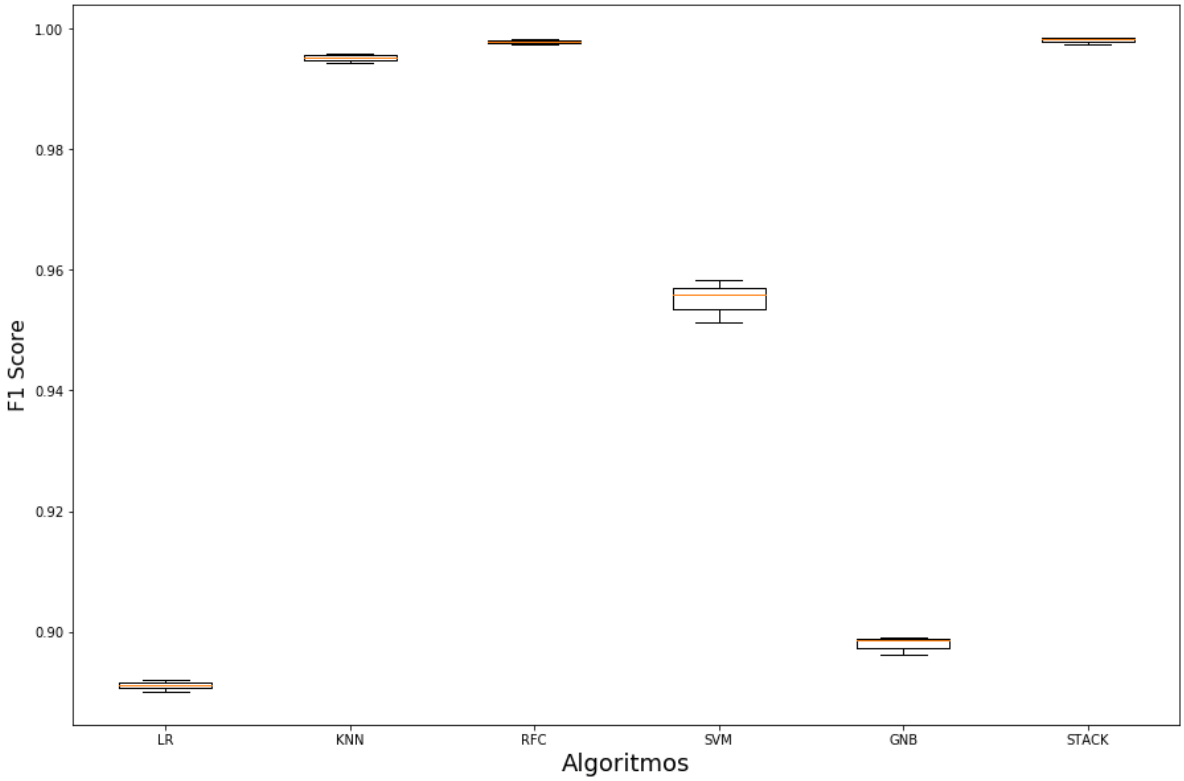


Ilustración 69. Comparación de modelos con sobremuestreo

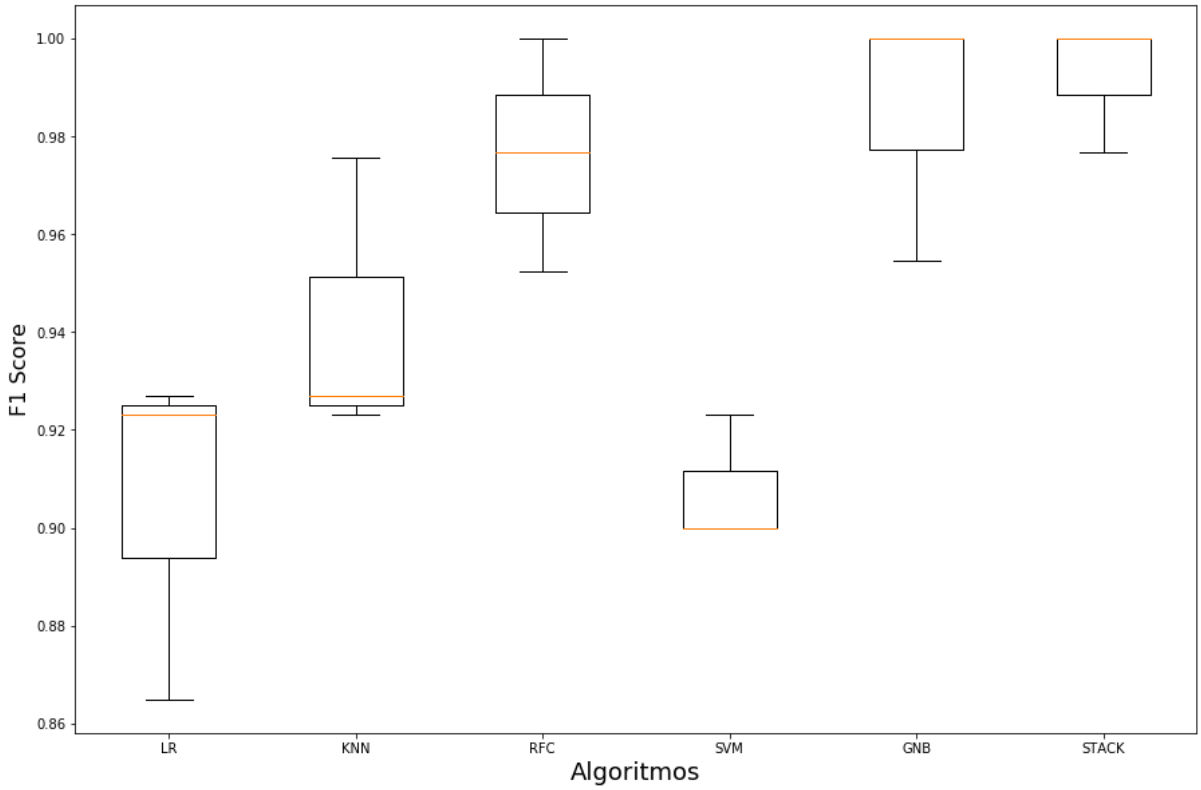


Ilustración 70. Comparación de modelos con submuestreo de generación

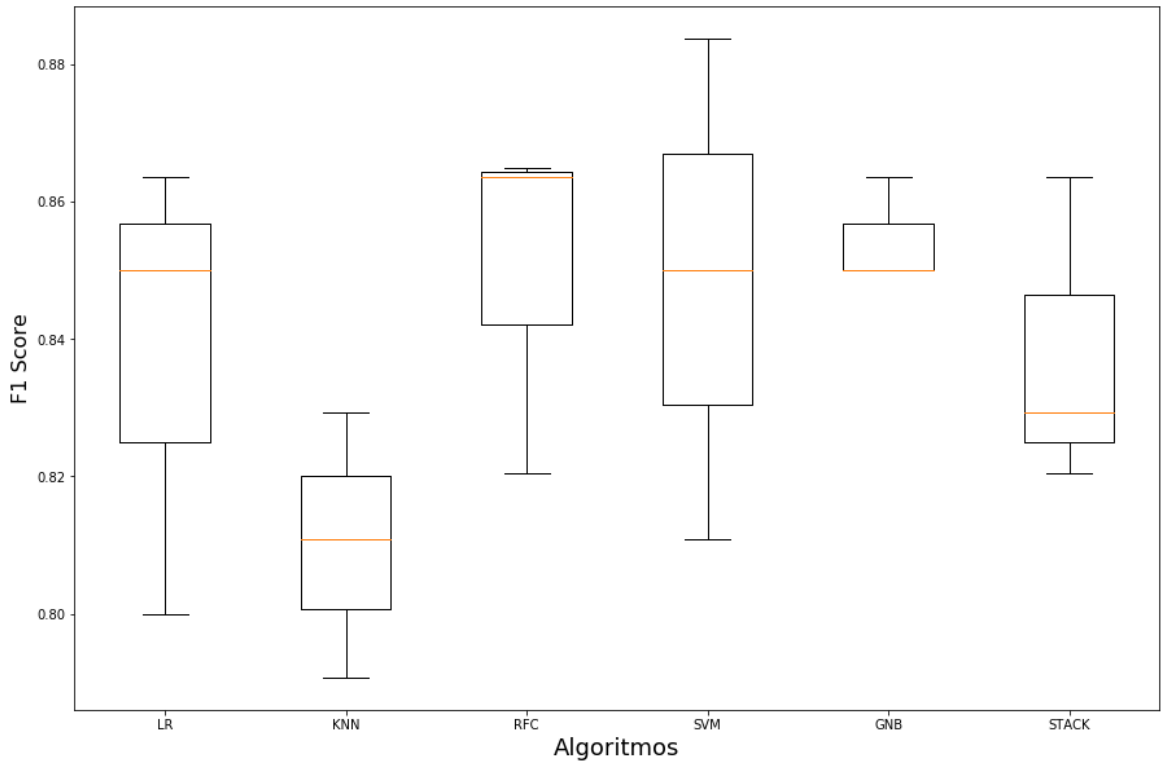


Ilustración 71. Comparación de modelos con submuestreo de selección

6. Conclusiones

6.1 Idoneidad de los objetivos

Como en la mayoría de proyectos de gran envergadura, durante la realización de éste, se ha tenido que revisar la planificación para modificarla pues no siempre los procesos marcados se pueden determinar con exactitud en ejecución y duración. Además, se debe tener en cuenta que pueden surgir contratiempos que ralenticen la realización del trabajo, como también que algunas de las expectativas como objetivos no hayan sido del todo viables. En la medida de lo posible, se ha procurado que la desviación fuera la menor posible y manteniendo, esto sí, las fechas límites de finalización.

Los objetivos marcados eran ambiciosos para el escaso tiempo de realización que se ha dispuesto por lo que la mitad de ellos no se han podido afrontar. De los 6 objetivos marcados 3 han podido ser resueltos, estos objetivos han sido:

- Creación de varios modelos supervisados de clasificación.
- Tratamiento de conjuntos de datos fuertemente desbalanceados.
- Realización un estudio de las variables de ingeniería de características como de selección.

Por el contrario, los objetivos no alcanzados han sido los siguientes:

- Identificación de variables o conjunción de éstas que anticipen un positivo en nuestra clase objetivo.
- Predecir el posible tiempo de aparición de una nueva incidencia.
- Creación de un modelo predictivo.

6.2 Planificación y metodología

La preparación de los datos ha consumido gran parte del tiempo planificado, aunque sin duda es un aspecto que se agradece en las posteriores etapas, pues reduce notablemente tiempo de ejecución de los algoritmos y nos evita posibles problemas de mal interpretaciones.

Este tiempo que no se ha dispuesto era el necesario para concluir el resto de objetivos, una vez realizada la primera etapa de modelos de clasificación. Sin duda, hubiera sido muy interesante disponer de los resultados de esta segunda fase pues es donde, se aporta el grosso de la información útil.

Revisando otros aspectos de la planificación se sobrevaloraron ciertos procedimientos dejando otros en un segundo plano que eran igual de importantes. Por ejemplo, se asignó una duración demasiada extensa para el análisis de los resultados y en cambio, el tiempo de procesado y limpieza de datos ha sido de menor duración teniendo éste, una carga de trabajo mucho más importante.

El hecho de no conseguir todos los objetivos ha inferido también en el apartado de visualización. En su planteamiento original consistía en la realización de un *dashboard* con Tableau y acceso web.

Sin duda, para la realización de proyectos es de suma importancia tener una buena planificación y conocer lo mejor posible la carga de trabajo de las diferentes fases.
Estos aspectos se tendrán en cuenta para posibles futuros trabajos.

6.3 Lecciones aprendidas y líneas de trabajo para el futuro

Se ha constatado que no hay una única forma de afrontar el problema del desbalanceo en el conjunto de datos, existen varias técnicas cada una de ellas con su pros y contras.

Además, el tener un conjunto fuertemente desequilibrado con respecto a los valores de la clase objetivo, nos fuerza a realizar un postprocesado del dataset como también el uso de otras métricas de lo contrario podemos caer en el error de pensar que nuestro modelo está clasificando muy bien cuando no es así.

No obstante, se obtuvieron muy buenos resultados en el conjunto de métricas. Esto no lleva reflexionar si con las técnicas de sobremuestreo los nuevos casos originados han sido de forma errónea y por lo tanto fácilmente clasificables, por ello se ha utilizado una variante de la técnica SMOTE ejecutándose en numerosas ocasiones.
Y en el caso, de submuestreo apenas se llegaba a tener un mínimo de observaciones, en donde el conjunto test apenas contenía cuarenta valores.

También se ha constatado como muchas de las variables no aportaban apenas información y en cambio perjudicaban en el tiempo de entrenamiento, por lo que fue una correcta decisión realizar una reducción de la dimensionalidad.

Prácticamente, la totalidad de los modelos obtuvieron valores parecidos y con diferentes métricas, no siendo posible destacar ninguno por encima. Sí que se ha detectado que en varias ocasiones, ha habido un peor de desempeño en los modelos de Regresión Logística y Naive Bayes.

7. Glosario

Autoencoders		tipo especial de redes neuronales que funcionan intentando reproducir los datos de entrada en la salida de la red.
Big Data	*Datos masivos*	conjunto de estrategias, tecnologías y sistemas para el almacenamiento, procesamiento, análisis y visualización de conjuntos de datos complejos
CPS	*Sistema ciberfísico*	mecanismo controlado o monitoreado por algoritmos basados en computación y estrechamente integrados con internet
CUDA	*Arquitectura Unificada de Dispositivos de Cómputo*	plataforma de computación en paralelo incluyendo un compilador y un conjunto de herramientas de desarrollo
DL	*Deep Learning / Aprendizaje profundo*	subconjunto, dentro del aprendizaje automático especializado en la creación de modelos más complejos y abstractos
IA	*Inteligencia Artificial*	Máquina flexible que percibe su entorno y lleva a cabo acciones que maximicen sus posibilidades de éxito en algún objetivo o tarea
IIot	*La Industria del Internet de las cosas*	concepto que se refiere a una interconexión digital de procesos industriales con internet
JIT	*Just-in-time*	sistema de organización de la producción para las fábricas, de origen japonés
Jupyter Notebook		es un entorno computacional interactivo basado en la web para crear documentos, especialmente Python.
kanban	*sistema de tarjetas*	sistema que controla la fabricación de los productos necesarios en la cantidad y tiempo en cada uno de los procesos que tienen en la fábrica
K-NN	*k vecinos más cercanos*	algoritmo de aprendizaje supervisado de clasificación no paramétrico, que estima el valor de la función de densidad de probabilidad
L1	*L1 Regularization*	técnica utilizada para evitar el sobreentrenamiento penalizando el valor absoluto de todas las ponderaciones
Aprendizaje vago	*Lazy Learning*	método de aprendizaje en el que la generalización más allá de los datos de entrenamiento es demorada hasta que se hace una pregunta al sistema.

ML	*Machine Learning / Aprendizaje Automático*	rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan
PCA	*Principal Component Analysis*	es una técnica utilizada para describir un conjunto de datos en términos de nuevas variables no correlacionadas.
scikit-learn		biblioteca para aprendizaje de máquina de software libre para el lenguaje de programación Python.
SMOTE	*Syntetic Minority Over-sampling Technique*	algoritmo de oversampling que genera instancias "sintéticas" o artificiales para equilibrar la muestra de datos basado en la regla del vecino más cercano
SVMs	*Máquinas de Soporte Vectorial*	algoritmo de aprendizaje supervisado capaz de resolver problemas de clasificación, tanto lineales como no lineales
WSN	*Red de sensores inalámbricas*	son sensores autónomos espacialmente distribuidos para monitorizar condiciones físicas o ambientales

8. Bibliografía

[1] **Fran Yañez** "*La meta es la industria 4.0."* Independently published. p131, 2017

[2] **William Trotman** "*5 Use Cases for Predictive Maintenance and Big Data"* ORACLE, 2017
https://blogs.oracle.com/bigdata/predictive-maintenance-big-data-use-cases

[3] **Sangüesa i Solé** "*El proceso de descubrimiento de los conocimientos a partir de los datos"* IFUOC. Fundació per a la Universitat Oberta de Catalunya, 2018

[4] **Gironés, Casas, Minguillón, Caihuelas "***Minería de datos Modelos y algoritmes."* Editorial UOC, 2017

[5] **Helen Shen** "*Interactive notebooks: Sharing the code"* NATURE, 2014
https://www.nature.com/news/interactive-notebooks-sharing-the-code-1.16261

[6] **TIOBE Index for March 2019**
https://www.tiobe.com/tiobe-index/

[7] **Victor Pascual** "*Introducción a Tableau"* IFUOC. Fundació per a la Universitat Oberta de Catalunya, 2018

[8] **Žilina, Slovakia.** *"Machine Learning Predictive Model for Industry 4.0: 13th International Conference",* KMO 2018, 08/2018

[9] **Ferreiro, Konde, Fernández, Prado** (2016) *INDUSTRY 4.0: Predictive Intelligent Maintenance for Production Equipment*

[10] **García Coria, J. A., Castellanos-Garzón, J. A., Corchado, J. M**, *"Intelligent business pro-cesses composition based on multi-agent systems"*.
Expert Systems with Applications, Vol.41(4 PART 1), pp.1189–1205, 2014

[11] **De La Prieta, F., Navarro, M., García, J. A., González, R., Rodríguez, S**. *"Multi-agent system for controlling a cloud computing environment*".
Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinfor-matics), Vol. 8154 LNAI, 2013.

[12] **Lee J, Bagheri B, Kao HA.** *"A cyber-physical systems architecture for industry 4.0-based manufacturing Systems"* 2015;3:18–23.

[13] **Da Xu L, He W, Li S**. *"Internet of things in industries: A survey. IEEE Trans Ind Inf"* 2014;10(4):2233–43.

[14] **Lee J, Lapira E, Bagheri B, Kao HA.** *"Recent advances and trends in predictive manufacturing systems in big data environament".*

[15] **Lee J, Ardakani HD, Yang S, Bagheri B**. *"Industrial big data analytics and cyberphysical systems for future maintenance & service innovation."* Procedia CIRP
2015;38:3–7.

[16] **Zhang L, Luo Y, Tao F, Li BH, Ren L, Zhang X, et al**. *"Cloud manufacturing: a new manufacturing paradigm."* Enterprise Inf Syst 2014;8(2):167–87.

[17] **Yang S, Bagheri B, Kao HA, Lee J. A** *"unified framework and platform for*

designing of cloud-based machine health monitoring and manufacturing systems." J Sci Eng 2015;137(4):040914.

[18] **Baheti R, Gill H.** *"Cyber-physical systems. Impact Control Technol"* 2011;12 (1):161–6.

[19] **Leitao P, Karnouskos S, Ribeiro L, Lee J, Strasser T, Colombo AW.** *"Smart agents in industrial cyber–physical systems."* Proc IEEE 2016;104(5):1086–101.

[20] **Kagerman, H., Anderl, R., Gausemeier J., Schuh G., and Wahlster W.** (2016) *Industrie 4.0 in a Global Context: Strategies for Cooperating with International Partners***,** Acatech Study https://www.acatech

[21] **Ballesteros, F**., *"La Estrategia Predictiva en el mantenimiento industrial".* Grupo Álava, España, Predictécnico Vol. 23, pp. 36-45, 2017.

[22] **T. Wuest; D. Weimer; C. Irgens; K.-D. Thoben**, "Machine learning in manufacturing: advantages, challenges, and applications. Production & Manufacturing" Research 4(1) (2016) pp. 23-45.

[23] **L. Monostori,** "Ai and machine learning techniques for managing complexity, changes and uncertainties in manufacturing."
IFAC Proceedings 35(1) (2002) pp. 119-130.

[24] . **K. Bharatkumar**, "Intelligent system's design approaches: a review. International Journal of Engineering and Management Research 5" (4) (2015), pp. 208-213.

[25] **M.I. Jordan; T.M. Mitchell**, *"Machine learning: trends, perspectives, and Prospects"*. Science 349(6245) (2015), pp. 255-260.

[26] **D.T. Pham; A.A. Afify**, *"Machine-learning techniques and their applications in Manufacturing"*, Journal of Engineering Manufactureol 219(5) (2005), pp. 395-412.

[27] **Fernández-Riverola, F., Diaz, F., Corchado, J. M.**, *"Reducing the memory size of a Fuzzy case-based reasoning system applying rough set techniques*". IEEE Transactions on Sys-tems, Man and Cybernetics Part C: Applications and Reviews, Vol. *37*(1), pp. 138–146, 2007.

[28] **Mata, A., Corchado, J. M.** *"Forecasting the probability of finding oil slicks using a CBR system*. In: Expert Systems with Applications, Vol. *36*(4), pp. 8239–8246, 2009.

[29] **AzureML Team** Predictive Maintenance: Step 1 of 3, data preparation and feature engineering April 27, 2015
https://gallery.azure.ai/Experiment/df7c518dcba7407fb855377339d6589f

[30] **Á Sicilia, D.Rodríguez, J.Riquelme** SMOTE-I: mejora del algoritmo SMOTE para balanceo de clases minoritarias. January 2009
https://www.researchgate.net/publication/229045207_SMOTE-I_mejora_del_algoritmo_SMOTE_para_balanceo_de_clases_minoritarias

[31] **Grupo de Investigación BISITE** Diseño de un modelo predictivo en el contexto Industria 4.0". Universidad Salamanca, España. 2018
https://knepublishing.com/index.php/KnE-Engineering/article/view/1458/3520

9. Anexos

En los anexos se va incluir el código utilizado para la realización de este trabajo.
El lenguaje de programación es Python en su versión 3.6 y el editor Jupyter Notebook.

Carga de las librerías requeridas.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from keras.optimizers import Adam
from keras.models import Sequential
from keras.utils import to_categorical
from keras.layers import Dense, Activation

from imblearn.combine import SMOTETomek
from imblearn.under_sampling import (RandomUnderSampler,ClusterCentroids)

from sklearn.svm import SVC
from sklearn.utils import resample
from sklearn.decomposition import PCA
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score, train_test_split, GridSearchCV
from sklearn.metrics import classification_report, roc_curve, confusion_matrix, auc
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

Importación del conjunto de datos.

```
dataset = pd.read_csv('dataset.csv', sep='\s*,\s*', delimiter=',', encoding="utf-8-sig", skipinitialspace=True)
```

Tamaña y tipo de datos.

```
dataset.shape
```

```
(8784, 28)
```

```
dataset.dtypes
```

```
Date                          object
Temperature                    int64
Humidity                       int64
```

Cambio de tipo para algunas variables.

```
dataset['Date'] =  pd.to_datetime(dataset['Date'])
dataset['Measure2'] = dataset['Measure2'].astype(object)
dataset['Measure3'] = dataset['Measure3'].astype(object)
dataset['Date.year'] = dataset['Date.year'].astype(object)
dataset['Date.hour'] = dataset['Date.hour'].astype(object)
dataset['Date.month'] = dataset['Date.month'].astype(object)
dataset['Date.day-of-week'] = dataset['Date.day-of-week'].astype(object)
dataset['Date.day-of-month'] = dataset['Date.day-of-month'].astype(object)
```

Resumen estadístico.

```
summary = dataset.describe()
summary = summary.transpose()
summary
```

Tablas de frecuencias de distintas variables.

```
pd.crosstab(index=dataset["Failure"], columns="count")

pd.crosstab(index=dataset["Operator"], columns="count")
pd.crosstab(index=dataset["Operator"], columns=dataset["Failure"])

pd.crosstab(index=dataset["Measure2"], columns="count")
pd.crosstab(index=dataset["Measure2"], columns=dataset["Failure"])

pd.crosstab(index=dataset["Measure3"], columns="count")
pd.crosstab(index=dataset["Measure3"], columns=dataset["Failure"])


pd.crosstab(index=dataset["Date.month"], columns=dataset["Failure"])
pd.crosstab(index=dataset["Date.day-of-month"], columns=dataset["Failure"])
pd.crosstab(index=dataset["Date.day-of-week"], columns=dataset["Failure"])
```

Tratamiento de la variable *Operator.*

```
pd.crosstab(index=dataset["Operator"], columns=dataset["Date.hour"])
```

```
dataset.loc[(dataset["Date.hour"] >= 8) & (dataset["Operator"] == "Operator2") , 'Operator'] = "Operator9"
```

```
pd.crosstab(index=dataset["Operator"], columns=dataset["Date.hour"])
```

Verificación de la existencia de valores nulos y separación de la variable objetivo.

```
dataset.isnull().values.any()
```

```
features = ['Temperature','Humidity','Measure1','Measure4','Measure5','Measure6',
            'Measure7','Measure8','Measure9','Measure10','Measure11','Measure12',
            'Measure13','Measure14','Measure15','Hours Since Previous Failure']
# Separamos los atributos
x = dataset.loc[:, features].values
# Separamos la clase objetivo
y = dataset.loc[:,['Failure']].values
# Estandarizamos los atributos
x = StandardScaler().fit_transform(x)
standartDataset = pd.DataFrame(data = x, columns = features)
```

Discretización de las variables que miden el tiempo.

```
bins = [-1, 7, 15, 23]
labels = [0,1,2]
dataset['Turno'] = pd.cut(dataset['Date.hour'], bins=bins, labels=labels)
pd.crosstab(index=dataset["Turno"], columns=dataset["Date.hour"])
```

```
bins = [-1, 15, 31]
labels = [1,2]
dataset['Quincena'] = pd.cut(dataset['Date.day-of-month'], bins=bins, labels=labels)
pd.crosstab(index=dataset["Quincena"], columns=dataset["Date.day-of-month"])
```

Comprobación y eliminación de valores atípicos en la Temperatura.

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.boxplot(dataset['Temperature'])
ax.set_xlabel('TEMPERATURA')
ax.set_ylabel('GRADOS Cº')
```

```
dataset.loc[(dataset.Temperature == 28), 'Temperature'] = 68
```

```
dataset.loc[(dataset.Temperature == 12), 'Temperature'] = 62
```

```
dataset.loc[(dataset.Temperature == 19), 'Temperature'] = 69
```

```
dataset.loc[(dataset.Temperature == 5), 'Temperature'] = 65
```

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.boxplot(dataset['Temperature'])
ax.set_xlabel('TEMPERATURA')
ax.set_ylabel('GRADOS Cº')
```

Comprobación y eliminación de valores atípicos en la Humedad.

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.boxplot(dataset['Humidity'])
ax.set_xlabel('HUMEDAD')
ax.set_ylabel('PORCENTAJE (%)')
```

```
dataset.loc[(dataset.Humidity > 100), 'Humidity'] = 95
```

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.boxplot(dataset['Humidity'])
ax.set_xlabel('HUMEDAD')
ax.set_ylabel('PORCENTAJE (%)')
```

Búsqueda de componentes principales.

```
featuresPCA = ['Temperature','Humidity','Measure1','Measure4','Measure5','Measure6',
               'Measure7','Measure8','Measure9','Measure10','Measure11','Measure12',
               'Measure13','Measure14','Measure15','Hours Since Previous Failure']
features = ['Date','Operator','Measure2', 'Measure3','Failure','Date.year',
            'Date.month','Date.day-of-month','Date.day-of-week','Date.hour',
            'Date.minute','Date.second','Turno','Quincena']
cov_mat = np.cov(dataset[featuresPCA].T)

eig_vals, eig_vecs = np.linalg.eig(cov_mat)

print('Eigenvectors \n%s' %eig_vecs)
print('\nEigenvalues \n%s' %eig_vals)
```

```
#  Hacemos una lista de parejas (autovector, autovalor)
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]

# Ordenamos estas parejas en orden descendiente con la función sort
eig_pairs.sort(key=lambda x: x[0], reverse=True)

# Visualizamos la lista de autovalores en orden desdenciente
print('Autovalores en orden descendiente:')
for i in eig_pairs:
    print(i[0])
```

```
autovalores=len(featuresPCA)

# A partir de los autovalores, calculamos la varianza explicada
suma_Auto = sum(eig_vals)
var_exp = [(i / suma_Auto)*100 for i in sorted(eig_vals, reverse=True)]
cum_var_exp = np.cumsum(var_exp)

# Representamos en un diagrama de barras
with plt.style.context('seaborn-pastel'):
    plt.figure(figsize=(6, autovalores))
    plt.bar(range(autovalores), var_exp, alpha=0.5, align='center',
            label='Varianza individual explicada', color='g')
    plt.step(range(autovalores), cum_var_exp, where='mid', linestyle='--',label='Varianza explicada acumulada')
    plt.ylabel('Ratio de Varianza Explicada')
    plt.xlabel('Componentes Principales')
    plt.legend(loc='best')
    plt.tight_layout()
```

Selección de las variables finales y conversión de la variable objetivo.

```
dataset.loc[(dataset.Failure == 'Yes'), 'Failure'] = 'Sí'
dataset.loc[(dataset.Failure == 'No'), 'Failure'] = 'No'

df= dataset[['Operator','Temperature','Humidity','Measure2','Measure3','Hours Since Previous Failure',
             'Date.month','Quincena','Date.day-of-week','Turno','Failure']]

df.columns = ['Operador','Temperatura','Humedad','Sensor_A','Sensor_B','Horas_sin_incidencia',
              'Mes','Quincena','Día semana','Turno','Falla']

for i in range(1, 10):
    df.loc[(df.Operador == 'Operator'+str(i)), 'Operador'] = i

df.loc[(df.Falla == 'No'), 'Falla'] = 0
df.loc[(df.Falla == 'Sí'), 'Falla'] = 1

y=df.Falla
x=df.drop('Falla',axis=1)
```

Diferentes técnicas de resampleo y las dimensiones resultantes.

```
sm = SMOTE(random_state=2019)
x_smt, y_smt = sm.fit_sample(x, y)

smt = SMOTETomek(random_state=2019)
x, y = smt.fit_sample(x, y)

cc = ClusterCentroids(random_state=2019)
x, y = cc.fit_resample(x, y)

rus = RandomUnderSampler(random_state=2019)
x, y = rus.fit_resample(x, y)

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, stratify=y)

X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)
```

```
print (X_train.shape)
print (X_test.shape)
print (y_train.shape)
print (y_test.shape)
```

Stacking o ensamblado de varios modelos.

```
for name, model in models:
    cv_results = model_selection.cross_val_score(clf, X_train, y_train,scoring='f1')
    results.append(cv_results)
    names.append(name)
```

```
X_test_stacking = np.column_stack((predNNC, predSVM, predRFC, predKNN, predLR, predGNB))

clf = RandomForestClassifier(n_estimators=100)

cvscores = cross_val_score(clf, X_test_stacking, y_test, cv=10,scoring='f1')

cv_results = model_selection.cross_val_score(clf, X_train, y_train,scoring='f1')
results.append(cv_results)
names.append('STACK')

print("Dimensiones de la matriz con las predicciones de todos los clasificadores: {}".format(np.shape(X_test_stacking)))
print("Precisión media obtenida con CV: {:.2f} +/- {:.2f} %".format(np.mean(cvscores)*100, np.std(cvscores)*100))
```

```
fig = plt.figure(figsize=(15,10))
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.xlabel('Algoritmos', fontsize=18)
plt.ylabel('F1 Score', fontsize=16)
plt.show()
```