BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

MAESTRÍA EN CIENCIAS DE LA ELECTRÓNICA
OPCIÓN EN AUTOMATIZACIÓN

“Desarrollo, validación numérica y experimental de un nuevo método de detección de daños en aerogeneradores Offshore” *

T E S I S

Presentada para obtener el título de:
Maestra en Ciencias de la Electrónica, Opción en Automatización

Presenta:

Ing. Gabriela Aquino González**

Directores:

Dr. José Eligio Moisés Gutiérrez Arias (FCE-BUAP, México)

Dra. Yolanda Vidal Seguí (UPC, España)

Dr. José Julián Rodellar Benedé (UPC, España)

Puebla, México **Septiembre 2020**

* Trabajo en colaboración con la Universidad Politécnica de Cataluña (UPC)

** Becario CONACYT.

Agradecimientos

A mi mamá, por ser mi motivación y mi ejemplo en la vida. Por estar presente en cada momento importante y darme el ánimo necesario para enfrentar cualquier dificultad. Por amarme desde el primer día que supo de mi existencia y darme la formación necesaria para convertirme en el ser humano que soy.

A Willy, por compartir tantas experiencias, apoyarme y creer en mí.

A mis amigos y a todos aquellos que me apoyaron, ya que sin ellos nada de esto seria posible.

A mis asesores por motivarme, por respetar mis ideas y sugerencias: al Dr. Moisés Gutiérrez, por confiar en mí y darme la oportunidad de trabajar con él en este proyecto; a la Dra. Yolanda Vidal, al Dr. Francesc Pozo y al Dr. José Rodellar por recibirme con los brazos abiertos durante mi estancia en la UPC, por dedicarme parte de su tiempo, compartir su conocimiento, darme consejos y hacer de este trabajo una grata experiencia.

A los miembros del jurado revisor Dra. Olga Félix, Dr. Jaime Cid y Dr. Gibrán Etcheverry, por sus atinados comentarios en cada revisión que me han hecho crecer académica y profesionalmente.

A la Benemérita Universidad Autónoma de Puebla, en especial a la Maestría en Ciencia de la Electrónica, por aceptarme como parte de esta gran familia durante dos años.

A la Universidad Politécnica de Cataluña y al grupo CoDAlab, por facilitarme el uso de sus instalaciones para realizar este trabajo.

A CONACYT por el apoyo económico brindado para poder desarrollar esta tesis y permitirme continuar con mi formación académica.

Esta investigación fue realizada
gracias al apoyo del Consejo de Ciencia y Tecnología
del Estado de Puebla

Dedicatoria

A G.A.D.U.

Por poner en mi camino a las personas correctas.

A mi mamá

Por todo tu amor.

Resumen

Este trabajo se enfoca en el desarrollo de un nuevo paradigma, capaz de validar de manera numérica y experimental la detección de daños en la estructura de los aerogeneradores colocados en altamar, también conocidos como aerogeneradores *Offshore*.

A lo largo de estas páginas se propone una metodología que permite detectar las fallas estructurales específicamente en las plataformas con cimentaciones tipo *jacket*. Este enfoque se ha validado de manera experimental en el laboratorio del grupo CoDAlab (Laboratorio de control, dinámica y aplicaciones), perteneciente a la Universidad Politécnica de Cataluña (UPC).

En este escrito se identifican dos partes principales: la primera, la obtención de datos a partir del banco experimental de pruebas que consta de una plataforma de aerogenerador *offshore* a escala con un soporte tipo *jacket* como cimentación. La plataforma se somete a diversas vibraciones que simulan diferentes intensidades de viento, ya que en la realidad la intensidad del viento no se puede imponer. Estas vibraciones se generan a partir de una señal de ruido.

Y la segunda, consiste en el procesamiento de los datos obtenidos, la aplicación de técnicas para la reducción de dimensión y posteriormente la utilización de los algoritmos de aprendizaje automático (*machine learning*) para la detección e identificación de fallas. Finalmente se analiza el desempeño de cada uno de los algoritmos elegidos y se obtiene una conclusión sobre cuál es el mejor para dar solución a la problemática presentada.

Índice general

Índice de figuras

Índice de tablas

Introducción

El cambio climático que estamos viviendo en la actualidad ha generado una crisis energética, que ha motivado a la humanidad a encontrar nuevas fuentes de energía. Entre las cuales se encuentra la energía eólica, que hoy en día ha recibido una enorme atención para aliviar la demanda mundial de combustibles fósiles y las preocupaciones resultantes sobre cuestiones ambientales. Algunas de las ventajas que ofrece la energía eólica son las siguientes:

- Es una fuente de energía barata y limpia.
- No emana contaminantes.
- No agrava el efecto invernadero.
- Es una fuente inagotable de energía.

La energía eólica tiene como finalidad aprovechar el viento y ha sido empleada desde hace siglos para aplicaciones diversas que van desde el transporte marítimo y aplicaciones agrícolas hasta la generación de energía eléctrica.

En la actualidad para poder aprovechar la energía cinética del viento y transformarla en energía eléctrica es necesario el uso de aerogeneradores, también conocidos como turbinas eólicas. Los aerogeneradores, si bien nos ofrecen ventajas, también se caracterizan por tener un mantenimiento costoso y crítico desde el punto de vista de la seguridad debido al gran tamaño de sus estructuras (ver Figura 1). Los costos de operación y mantenimiento de un aerogenerador oscilan entre 15 % y 35 % del costo total, de donde 80 % proviene del mantenimiento no planificado [1].

Existen dos tipos de aerogeneradores *onshore* y *offshore*. Los de tipo *offshore* se colocan en alta mar para aprovechar el gran potencial de la energía eólica marina. Por ello, cada vez se busca colocarlos en aguas más profundas y dependiendo de la profundidad, para soportarlos se utilizan diferentes tipos de fundaciones entre las cuales encontramos las plataformas con soporte tipo *jacket*.

Figura 1: Mantenimiento a aerogeneradores de Artajona, España.

En los aerogeneradores *offshore*, las tareas de mantenimiento son de 5 a 10 veces más elevadas que en tierra debido a las cambiantes condiciones climáticas del entorno. Por esta razón, es importante monitorear y detectar cuándo éstas estructuras presentan algún tipo de falla, para así poder tomar acciones antes de que ocurra un desastre económico o humano. Debido a las difíciles condiciones de acceso a los aerogeneradores *offshore*, no es tarea fácil obtener datos para realizar pruebas, por eso en algunas investigaciones se utilizan programas de simulación como el software FAST nombrado así por sus siglas en inglés *Fatigue, Aerodynamics, Structures, and Turbulence* el cual es un simulador aeroelástico completo capaz de imitar las cargas extremas y de fatiga de los aerogeneradores de eje horizontal de dos y tres palas [2]. Sin embargo, en trabajos recientes se han realizado plataformas y aerogeneradores a escala para poder realizar las pruebas pertinentes.

Estado del arte

La energía eólica se ha desarrollado rápidamente, este crecimiento también conlleva el aumento simultáneo de los accidentes de aerogeneradores con el aumento de tamaño de la turbina de viento. En [3] se presenta el aumento constante del número de proyectos de aerogeneradores *offshore*, los cuales han estimulado la investigación, entre otras, en la detección e identificación de defectos o daños.

Para realizar la detección e identificación de daños (parte de la monitorización de la integridad de la salud estructural) en aerogeneradores se han propuesto varios métodos, dentro de ellos, se pueden encontrar documentadas diversas metodologías, por ejemplo, en [4] se trabaja con el análisis de textura de imagen, en [5] se propone un nuevo esquema de detección y aislamiento de fallas basado en datos de series de tiempo y análisis de datos sin usar ningún tipo de modelado físico, en [6] se presenta un algoritmo de detección de fallas

mecánicas utilizando solo mediciones de corriente del estator del generador no estacionario.

Una de las metodologías más destacadas dentro del campo de identificación de daños estructurales está basada en el estudio de las vibraciones [7]. Estas han tenido un gran desarrollo durante los últimos años por su simplicidad, facilidad de uso y alta efectividad [8], ya que emplean señales aleatorias de excitación y/o respuesta a la vibración (series de tiempo), construcción de modelos estadísticos y esquemas de toma de decisiones estadísticas para deducir si existe algún daño en la estructura [9]. En la mayoría de estudios que hacen uso de estas metodologías se enfocan en la respuesta a la vibración y excitación de entrada medibles. No obstante, en algunas aplicaciones la excitación no se puede imponer y a menudo no se puede medir [10].

Otros estudios que también se centran en la salud estructural de los aerogeneradores *offshore* son mencionados a continuación: en [11] se realizó una revisión de los sistemas de monitoreo de salud estructural para de los aerogeneradores *offshore*, considerando el tema como un problema de reconocimiento de patrones estadísticos. En el trabajo realizado por Mieloszyk et al. [12] se establece un sistema de monitoreo de la salud estructural basado en sensores de rejilla de fibra *Bragg* dedicados a un modelo de estructura de soporte de turbina eólica marina para detectar y localizar grietas. De igual modo se destaca la investigación [13] de Fritzen et al., donde se presenta un método para la detección y localización de daños en línea acompañado de pruebas de campo de una plataforma prototipo de 5MW.

En los últimos años, algunos investigadores han discutido sobre la detección de daños en cimentaciones en alta mar. Un trabajo a destacar es el realizado por Weijtjens et al. [14], relacionado con el monitoreo de un aerogenerador *offshore* real con monopilote basado en sus frecuencias de resonancia, donde los problemas clave son la variabilidad operativa y ambiental de las frecuencias de resonancia de la turbina que pueden ocultar cualquier cambio estructural.

En otros trabajos, la plataforma tipo *jacket* es analizada, por ejemplo, en [15] se habla de un método basado en la actualización del modelo de elementos finitos y que hace uso de la lógica difusa para aplicarla a una plataforma a escala de este tipo ubicada en un laboratorio.

Para finalizar los métodos documentados para la detección de daños, en [16] se utiliza un banco de pruebas experimental similar al indicado en este trabajo, la detección de daños se realiza (pero no la localización o clasificación) mediante el indicador de daño estimado de matriz de covarianza. En este documento se propone una metodología de detección y localización de daños para un aerogenerador *offshore* con plataforma tipo *jacket* utilizando solo datos de respuesta a la vibración. El trabajo comprende de las etapas mostradas en la Figura 2.

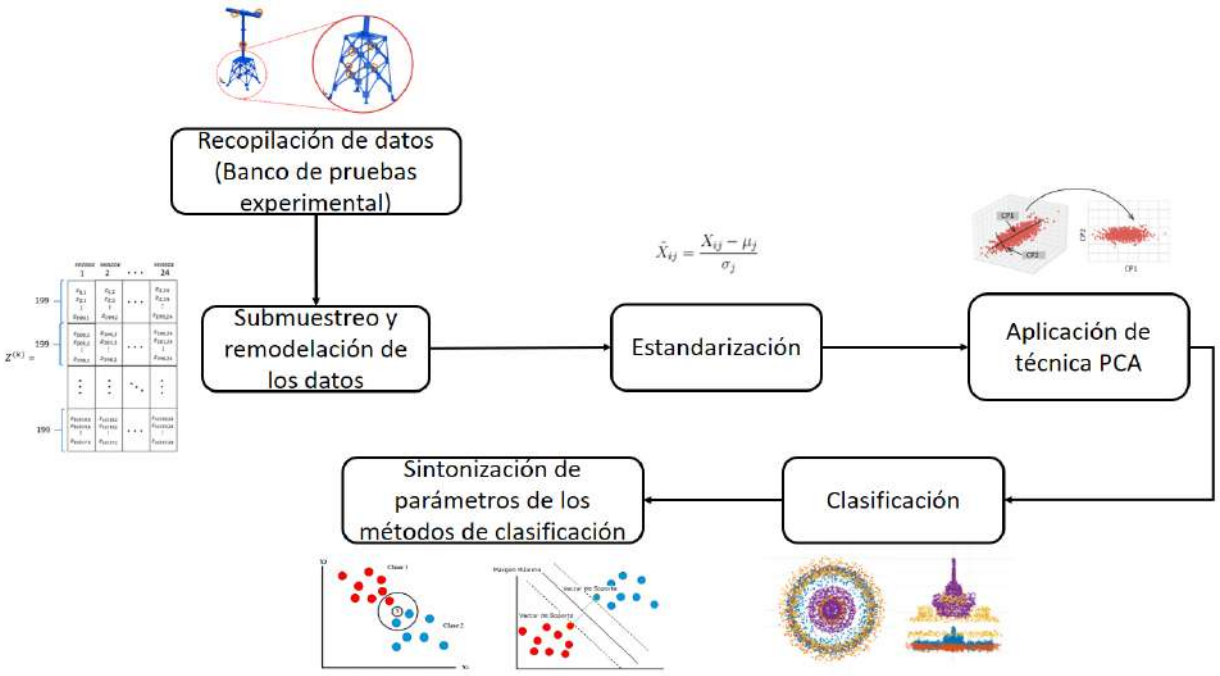


Figura 2: Etapas del proyecto.

Primero, se recopilan los datos con los que se probará la metodología propuesta. Para ello se hace uso de un banco de pruebas experimental que se explica a detalle en el Capítulo 2.

Segundo, los datos recabados y almacenados tal cuales se adquirieron del banco de pruebas (datos en crudo), se submuestrean y remodelan con el fin de aumentar la cantidad de información en cada muestra y de esta manera mantener la mayor cantidad de información posible con la mínima cantidad de datos.

Tercero, se realiza una estandarización para ajustar las magnitudes de los datos que provienen de diferentes sensores y para simplificar el cálculo de los componentes principales (CP).

Cuarto, se aplica la técnica de *PCA* para realizar una selección de componentes principales y así reducir la dimensionalidad de los datos.

Quinto, se prueban los métodos de clasificación k -NN y máquina soporte vectorial (SVM). Para estimar su rendimiento se utiliza la técnica de validación cruzada con $\kappa = 5$.

Sexto, se sintonizan los parámetros de los métodos de clasificación elegidos para finalmente poder definir cuál método y con qué parámetros nos ofrece el mejor rendimiento y una solución óptima al problema presentado.

Justificación

La viabilidad de estructuras de alto valor añadido como son las instalaciones eólicas *offshore* depende, entre otros factores, de los costos asociados a su mantenimiento. Debido a las exigentes condiciones ambientales, unido a la dificultad de acceso, se torna inviable su explotación. Por tal razón es vital la investigación en métodos de detección e identificación de daños estructurales. Esto favorecerá la adaptación de los aerogeneradores a las condiciones extremas del entorno marino alargando los períodos de funcionamiento y a su vez, minimizando los costos de mantenimiento.

La detección de daños se enmarca en el ámbito de la ingeniería llamado monitorización de la integridad de la salud estructural. En este sentido, se entiende el daño como cualquier alteración a la estructura que provoca un cambio en sus propiedades físicas, tan solo en Suecia se realizó un estudio que muestra las fallas más recurrentes en aerogeneradores [3] los cuales se muestran en la Figura 3. De forma global, se han desarrollado estrategias robustas enfocadas a aplicaciones en aerogeneradores sometidos a condiciones ambientales variables. Muchos de los trabajos en este campo están basados en principios de ondas elásticas guiadas, un campo el cual se prevé siga explorándose.

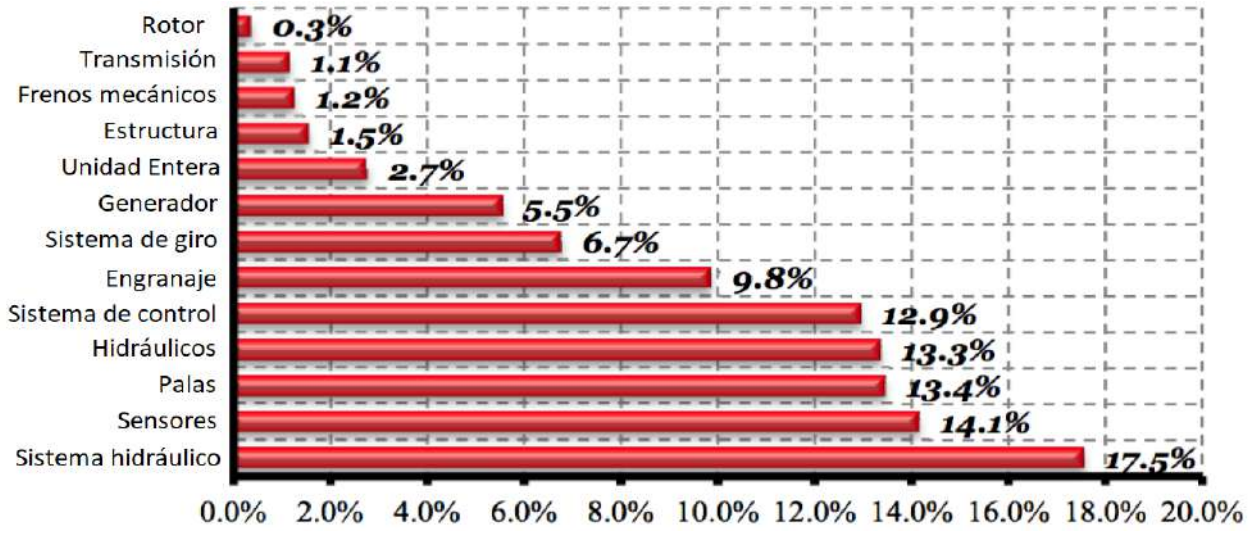


Figura 3: Ribrant & Bertling, (2007). Distribución del número de fallos para las plataformas de energía eólica de Suecia entre los años 2000 a 2004.

En la Universidad Politécnica de Cataluña se ha estado trabajando en un nuevo paradigma, en el cual ya no se fuerza una excitación predeterminada en la estructura, sino que es el propio viento incidente el que la excita. En consecuencia, el nuevo paradigma implica una

mayor complejidad en el proceso de detección de daño, por lo que se trabaja con métodos basados en aprendizaje automático (*Machine Learning*).

Objetivos

Objetivo general

Desarrollar, validar numérica y experimentalmente un nuevo método de detección e identificación de daños de los aerogeneradores offshore integrando investigación fundamental relacionada con el desarrollo de un algoritmo que tenga aplicabilidad real.

Objetivos particulares

- Estudiar la bibliografía relacionado con el tema.
- Estudiar la herramienta de simulación numérica FAST que permite simular con gran fidelidad el comportamiento de turbinas de tamaño industrial (MW).
- Estudiar el actuador y los sensores implementados en el prototipo.
- Estudiar la técnica de reducción de dimensionalidad PCA (*Principal Components Analysis*)
- Desarrollar estrategias de detección e identificación de daños utilizando métodos basados en aprendizaje autónomo (k vecinos más cercanos, árboles de decisión, máquinas de soporte vectorial o combinaciones de dichos métodos).
- Validar las técnicas de *Machine Learning* a través de ensayos experimentales en el laboratorio.

Construcción del documento

La tesis presentada a continuación ha quedado estructurada de la siguiente manera: Capítulo 1, presenta el marco teórico que habla de los conceptos relacionados a los aerogeneradores. Capítulo 2, detalla el banco de pruebas de laboratorio experimental utilizado para validar la metodología propuesta. Capítulo 3, contiene la metodología usada para la detección y clasificación de daños en los aerogeneradores con estructura *offshore*. Capítulo 4, presenta los resultados obtenidos a partir de la aplicación de la metodología propuesta. Finalmente, se muestran las conclusiones obtenidas.

Capítulo 1

Marco teórico

Aunque la gran mayoría de nosotros conocemos sobre la existencia de los aerogeneradores, pocos sabemos a detalle cómo se clasifican o cuáles son las partes que los conforman, y de igual forma sucede con el tema del aprendizaje automático. Por ello, en este capítulo se exponen los conceptos necesarios para adentrarnos en cada uno de estos temas.

1.1. Definición de aerogenerador

Una turbina de viento o aerogenerador es una estructura/sistema que transforma la energía cinética del viento en energía eléctrica.

1.1.1. Clasificación

Los aerogeneradores se pueden clasificar de acuerdo a [1] (ver Figura 1.1) según su ubicación ya sea *onshore* u *offshore*, por la posición del eje, orientación del rotor, capacidad de rotación de las palas o velocidad del rotor.

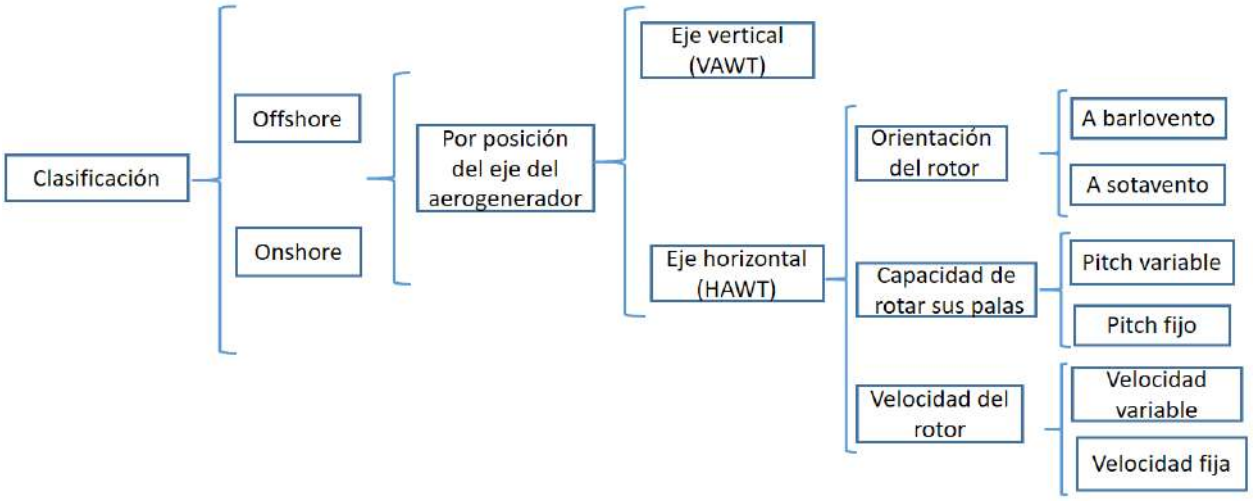


Figura 1.1: Clasificación de los aerogeneradores.

Como se muestra en la Figura 1.2, los aerogeneradores se clasifican como *onshore* y *offshore*, dependiendo del sitio dónde se coloquen.

Un **aerogenerador *onshore***, como los colocados en el parque eólico PIER II (Esperanza, Pue, México), es una turbina eólica que se encuentra ubicada físicamente en tierra firme, por lo general, en zonas costeras o de alta incidencia de vientos superficiales. Mientras que un **aerogenerador *offshore*** es una turbina eólica que se coloca en alta mar para aprovechar las corrientes de viento marinas.

(a) Parque eólico *onshore* PIER II (Esperanza, Pue, México).

(b) Parque eólico *offshore* WIKINGER (Mar Báltico, Alemania).

Figura 1.2: Parques eólicos.

En las zonas costeras la velocidad del viento es 1m/s superior respecto a la terrestre, lo que garantiza un desempeño de los parques *offshore* entre 30 % y 50 % superior al de los parques *onshore*. Además, existen una serie de ventajas al instalar aerogeneradores en alta mar [17], entre las cuales encontramos:

- Disponibilidad de áreas de instalación de mayor tamaño.
- Disminución del impacto visual y de los ruidos sobre poblaciones.
- Vientos más uniformes y con velocidades mayores, lo que se traduce a incrementar la posibilidad de obtener un mayor recurso eólico explotable.
- Menos efectos turbulentos sobre las estructuras, lo que reduce las solicitaciones de fatiga.

Actualmente estos aerogeneradores cuentan con palas que llegan a medir hasta 174m de diámetro como los que se están instalando para el parque eólico *Baltic Eagle* en el Mar Báltico en Alemania [18]. Para instalar estos colosos del mar se requieren de cimentaciones especiales como las mostradas en la Figura 1.3, las cuales han ido evolucionando a través de los años para poder colocarlos cada vez a mayor profundidad [1].

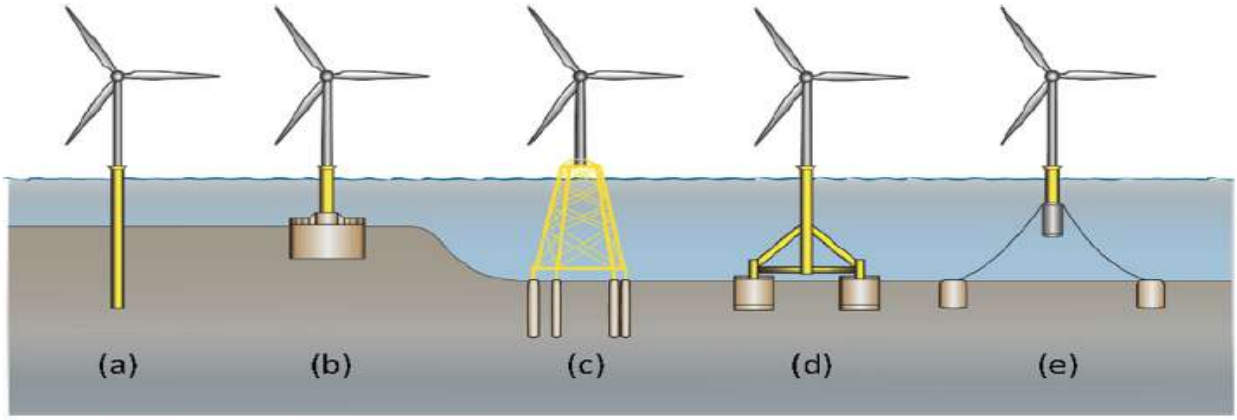


Figura 1.3: Kaynia, A.M. (2018). Tipos de cimentaciones comunes utilizadas en el diseño de aerogeneradores offshore: mono-pila (a), mono-pod (cajón con faldón) (b), estructura jacket (c), trípode (d) y aerogenerador flotante con anclajes (e).

Los cimientos más comunes son los monopilote al ser utilizados de un 81 % en los aerogeneradores. Estas son estructuras bastante simples ancladas directamente al fondo marino. Se pueden instalar en profundidades de agua de hasta 40 metros.

EL 5.7 % de aerogeneradores ocupa cimentación de gravedad e implican el uso de una gran plataforma de concreto o acero con un diámetro de aproximadamente 15 metros y un peso

de aproximadamente 3,000 toneladas. Las cimentaciones de estructura tipo *jacket* se usan para aguas más profundas debido al menor peso de las estructuras de soporte y se fijan al suelo mediante pilotes incorporando cuatro puntos de anclaje al fondo marino, de forma que aportan más seguridad a la fijación de las torres de los aerogeneradores. Para profundidades de agua más grandes, la solución económica es una cimentación flotante con anclaje al fondo marino.

Como se mencionó anteriormente, existen diversas formas de clasificar un aerogenerador además de la ubicación, estos tipos de clasificación se explican a continuación:

Por la posición del aerogenerador

En esta clasificación encontramos los aerogeneradores con eje vertical y los de eje horizontal, los cuales de acuerdo a [19] se definen como:

- **Eje vertical:** Conocido como VAWT (*Vertical Axis Wind Turbines*, por sus siglas en inglés), posee el eje del rotor en posición vertical, esto le permite generar energía eléctrica sin importar la dirección desde donde proviene el viento. Pueden trabajar incluso cuando el viento sopla a velocidades muy bajas. Los aerogeneradores más utilizados de eje vertical se observan en la Figura 1.4 y son los de tipo Savonius, tipo Darrieus y tipo Panémona [20].

Figura 1.4: Renovables Verdes (2017). Tipos de aerogeneradores VAWT (De izquierda a derecha: Tipo Daerrius, tipo Savonius, tipo Panémona.

- **Eje horizontal:** Los aerogeneradores de eje horizontal o HAWT (*Horizontal Axis Wind Turbine*), se construyen con un rotor tipo hélice con el eje dispuesto en posición horizontal, su propósito es transformar el movimiento lineal del viento en un movimiento

rotatorio que se encargará de impulsar el alternador para producir energía eléctrica (Ver Figura 1.5).

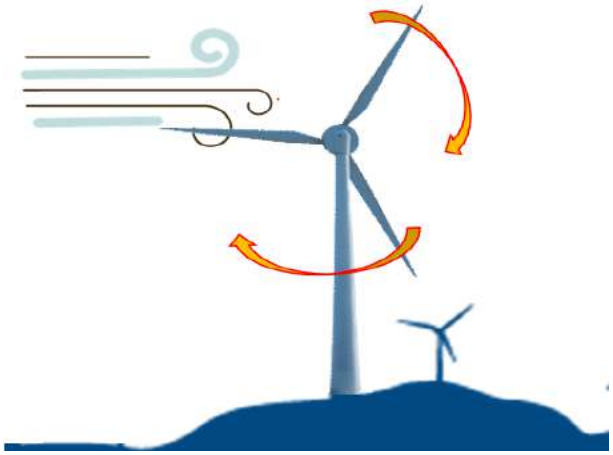


Figura 1.5: Aerogenerador HAWT.

Por la orientación con respecto del viento

Dependiendo de la orientación (ver Figura 1.6) que la turbina eólica tenga con respecto al viento se tiene la siguiente clasificación [19]:

- **A barlovento:** El rotor se encuentra de frente a la dirección del viento dominante. Es la configuración más utilizada para el diseño de aerogeneradores, con ella se obtiene un mayor aprovechamiento de la fuerza del viento. No presenta interferencias aerodinámicas con la torre. Una característica a destacar es que en esta configuración se requiere un mecanismo capaz de orientar la máquina hacia el viento.
- **A sotavento:** El rotor se encuentra enfocado en sentido contrario a la dirección del viento dominante. No son necesarios elementos de orientación automatizada.

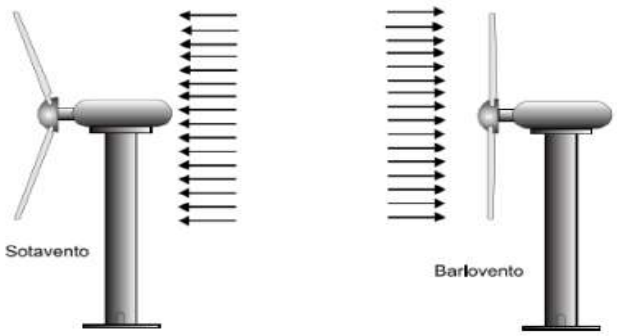


Figura 1.6: Guerrón, G. (2014). Clasificación de aerogeneradores según su orientación respecto al viento.

Por la capacidad de rotar sus palas a lo largo de los ejes longitudinales

- **Pitch fijo:** Inicialmente de menor costo, tienen capacidad reducida para controlar las cargas y cambiar el par aerodinámico. No son muy comunes para grandes aerogeneradores [1].
- **Pitch variable:** Permite el giro de las palas a lo largo del eje de paso.

Por la velocidad del rotor

- **Velocidad variable:** El rotor puede girar a distintas velocidades. Tienden a funcionar más cerca de su máxima eficiencia aerodinámica durante un mayor intervalo del tiempo.
- **Velocidad fija:** El rotor gira siempre a la misma velocidad, por lo cual no pueden aprovechar los vientos variables.

1.1.2. Componentes estructurales

A nivel estructural, los principales componentes estructurales de un aerogenerador [21] se enumeran a continuación y se pueden apreciar en la Figura 1.7:

1. **Rotor:** Formado por un conjunto de elementos que ayudan a transformar la energía del viento en energía mecánica.
2. **Góndola:** Protege a todos los componentes mecánicos de las inclemencias del tiempo. Gira en torno a la torre para poner a la turbina en dirección al viento.
3. **Palas:** Recolectan la energía cinética del viento. El diseño de estas palas son similares a las alas de un avión.

4. **Torre:** Es el elemento estructural que se encarga de transferir a la cimentación las cargas de la góndola y el rotor. La torre es la parte más visible del aerogenerador.

5. **Cimentación:** Su misión es transmitir las cargas estructurales del aerogenerador al suelo distribuyéndolas de forma que no superen su presión admisible.

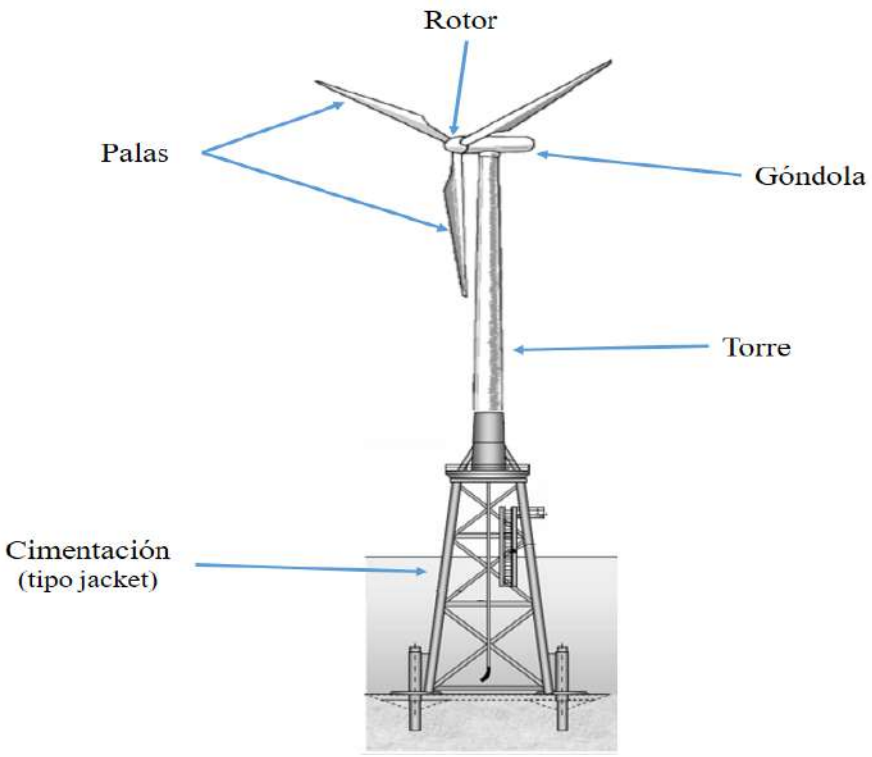


Figura 1.7: Ejemplo de la estructura de un aerogenerador *offshore*.

1.2. Descripción simulador FAST

El software FAST (*Fatigue, Aerodynamics, Structures, and Turbulence*) es un simulador aeroelástico capaz de predecir las cargas extremas y de fatiga de los aerogeneradores de eje horizontal (*HAWT*) de dos y tres palas [22], creado por el Laboratorio Nacional de Energía Renovable (NREL: *National Renewable Energy Laboratory*). Este software es usado en diferentes proyectos donde no se pueden obtener datos de aerogeneradores de tamaño real. Ha sido evaluado y ha resultado adecuado para el cálculo de las cargas de aerogeneradores tanto *onshore* y *offshore* [23] mediante los diferentes módulos de trabajo mostrados en la Figura 1.8.

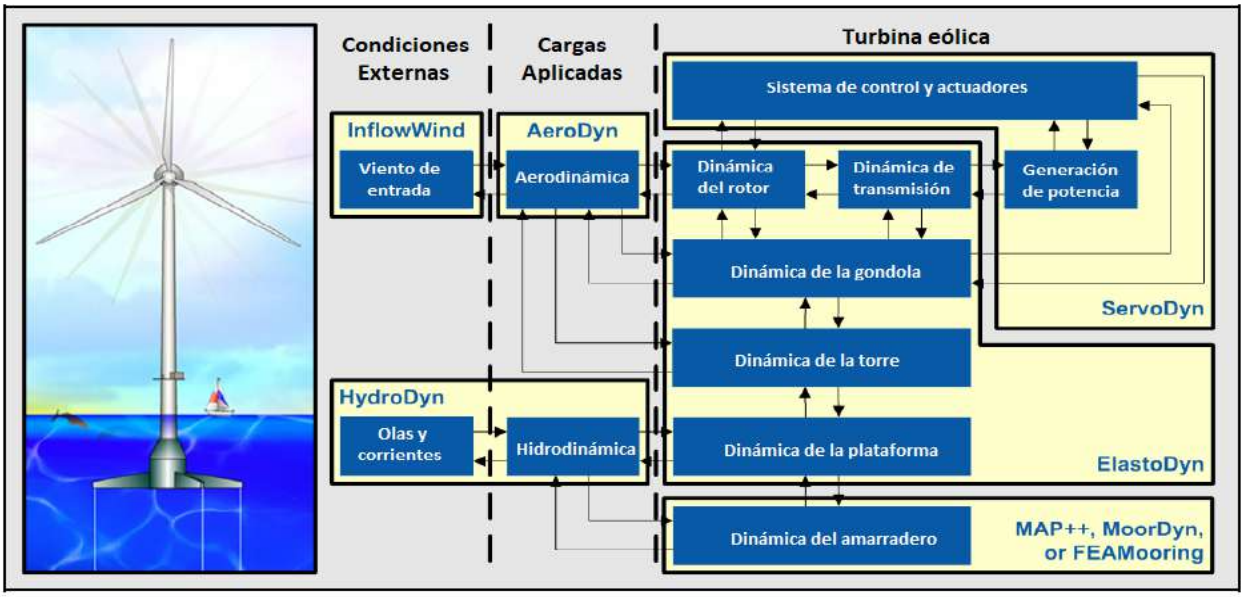


Figura 1.8: Jonkman J. & Jonkman B. (2016). Módulos de control FAST.

En FAST por ejemplo, en cuanto a los aerogeneradores *offshore*, se pueden encontrar diferentes modelos de aerogeneradores y de cimentaciones los cuales cuentan con pruebas de certificación basadas en turbinas eólicas existentes, esto es para realizar simulaciones y observar el comportamiento con respecto a diversas perturbaciones de la manera más fiel posible a un aerogenerador en condiciones reales.

Otra característica que otorga FAST es la posibilidad de poder realizar la vinculación entre FAST y Simulink mediante MATLAB©.

Como parte de los objetivos de esta tesis se encuentra el estudio de la herramienta FAST por lo cual se realizaron diversas simulaciones utilizando como objeto de estudio la "turbina eólica de referencia de 5 MW de NREL *offshore* con plataforma tipo *jacket* en condiciones saludables" [23], sus características se muestran en la Tabla 1.1. Para la simulación se tomo en cuenta un tiempo de 60 segundos y un viento turbulento.

Frecuentemente esta turbina se utiliza por los equipos de investigación de todo el mundo para estandarizar las especificaciones de referencia de las turbinas eólicas marinas y para cuantificar los beneficios de las tecnologías avanzadas de energía eólica terrestres y marinas [24].

Aerogenerador de referencia	
Potencia nominal	5MW
Número de palas	3
Diámetro del rotor, altura del rotor	126m, 3m
Velocidad de viento: corte inicial, nominal, corte final	3 m/s, 11.4 m/s, 25 m/s
Masa del rotor	110,000 kg
Masa de la góndola	240,000 kg
Orientación del rotor	barlovento

Tabla 1.1: Propiedades del aerogenerador.

Se obtuvo información sobre la velocidad angular del eje de alta velocidad (ver Figura 1.9) y la posición del ángulo *Pitch* (ver Figura 1.10).

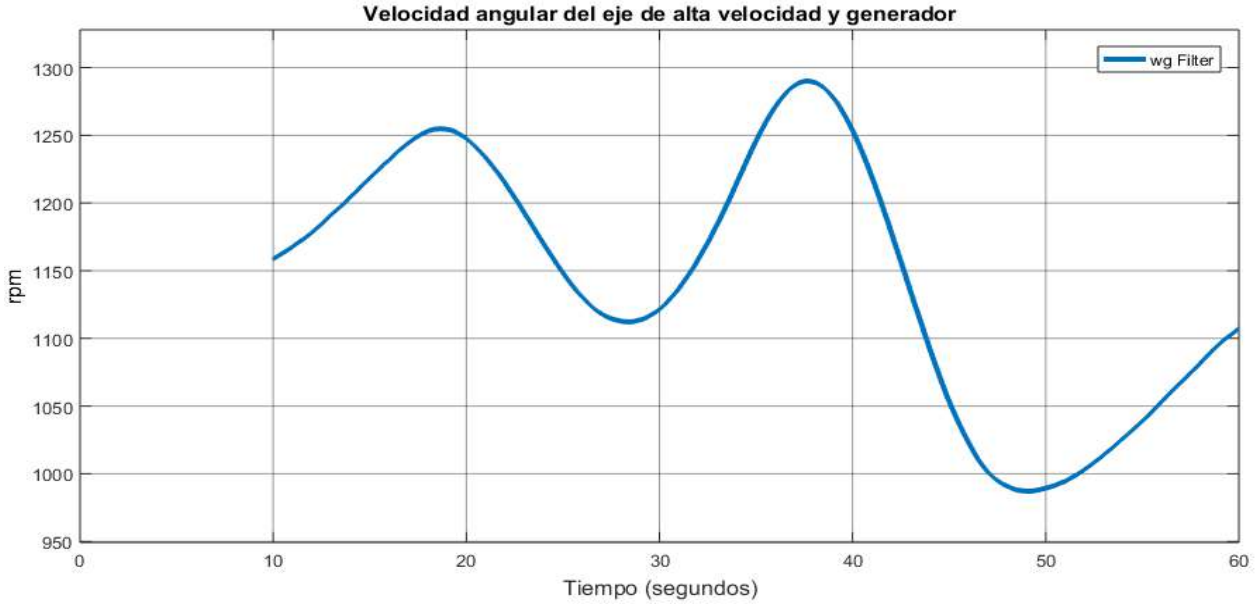


Figura 1.9: Velocidad angular del eje de alta velocidad y generador.

El objetivo de la "velocidad angular del eje de alta velocidad y generador" es mantenerse regulada cerca del valor nominal de la velocidad angular del generador, en este caso 1200rpm. Se observa que en ocasiones se sobrepasa y en otras disminuye este valor, pero se corrige. Esto se debe a que existe una relación no lineal entre el ángulo pitch y la velocidad angular del generador.

En la Figura 1.10 se muestra el ángulo de posición de una de las palas del aerogenerador, donde es posible apreciar el comportamiento de esta respecto a la velocidad del viento.

Cuando el viento es suave la pala se encuentra abierta para captar el máximo viento posible y así producir el máximo de energía, entonces la pala tiene un ángulo de cero grados. Mientras que si se encuentra en presencia de un viento fuerte el ángulo cambia su posición para cerrar las palas y evitar que el aerogenerador se acelere demasiado.

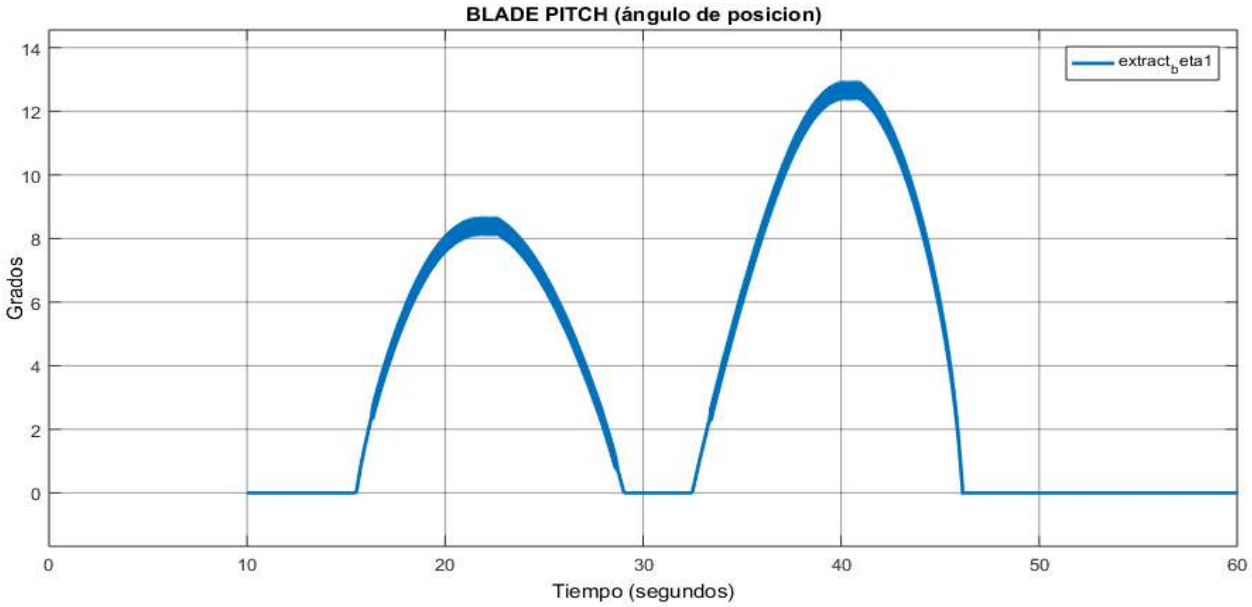


Figura 1.10: Pala 1, ángulo pitch (posición).

1.3. Aprendizaje automático (*Machine Learning*)

El aprendizaje automático o como es conocido en inglés, *machine learning*, se puede definir como un conjunto de métodos que identifican patrones y luego usa los patrones descubiertos para predecir datos futuros, o bien, para realizar otros tipos de acciones como la toma de decisiones bajo incertidumbre (por ejemplo, planificar cómo recopilar más datos) [25].

El proceso de aprendizaje automático se puede representar como una serie de pasos [26]:

- Adquisición y procesamiento de datos.
- Selección y extracción de características.
- Selección del modelo.
- Validación.
- Clasificación y predicción.

1.3.1. Clasificación

El aprendizaje automático se puede clasificar como aprendizaje supervisado y no supervisado. Dentro del aprendizaje supervisado están los algoritmos de regresión y clasificación. Mientras que en el aprendizaje no supervisado se encuentran los algoritmos de *clustering*. Esta clasificación se muestra en la Figura 1.11.

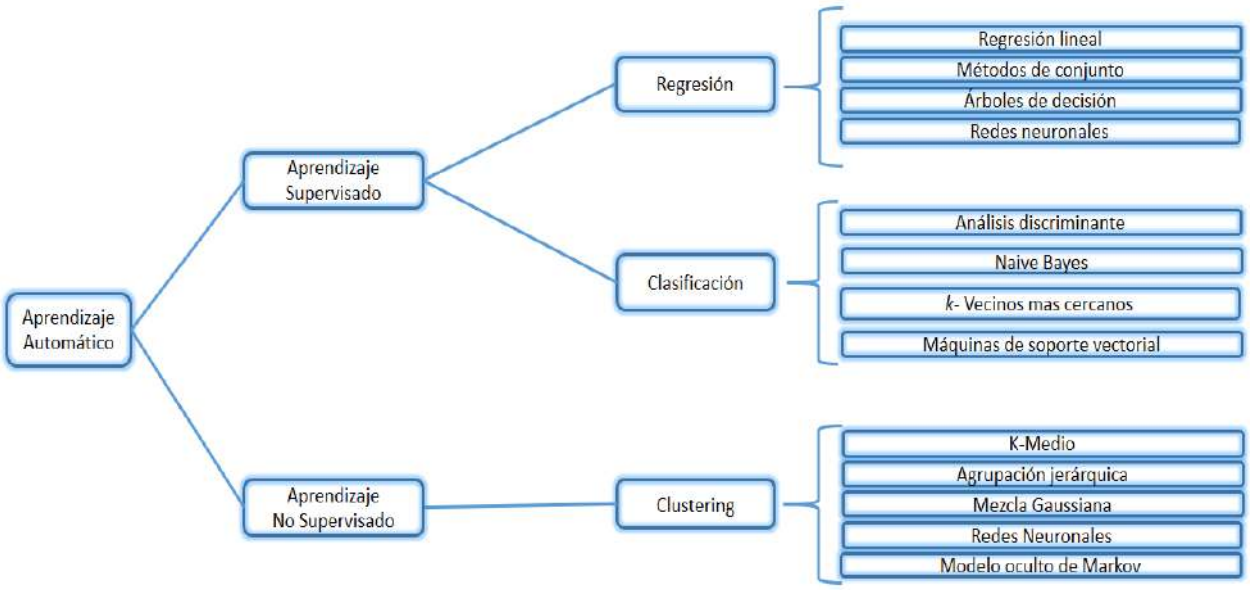


Figura 1.11: Tipos de aprendizaje automático.

Aprendizaje supervisado: Desarrolla un modelo predictivo basado en los datos de entrada y salida. El principal objetivo del aprendizaje automático supervisado es construir un modelo que haga predicciones basadas en evidencia, en presencia de incertidumbre. Para ello un algoritmo del tipo aprendizaje supervisado toma un conjunto conocido de datos de entrada y respuestas conocidas a los datos (salida) y entrena un modelo para generar predicciones razonables para la respuesta a nuevos datos [27].

En el aprendizaje automático supervisado distinguimos entre los modelos que predicen una variable numérica o una variable categórica, estas son conocidas como técnicas de regresión y clasificación respectivamente.

- Clasificación: Clasifican los datos de entrada en categorías. Predicen respuestas discretas.
- Regresión: Predicen respuestas continuas.

Aprendizaje no supervisado: En este tipo de aprendizaje no se dispone de un conocimiento previo de los datos de salida, por tanto, se pretende describir la estructura de los datos, para intentar encontrar algún tipo de organización que simplifique el análisis.

El aprendizaje no supervisado encuentra patrones ocultos o estructuras intrínsecas en los datos. Se utiliza para extraer inferencias de conjuntos de datos que consisten en datos de entrada sin respuestas etiquetadas. La agrupación (*clustering*), es la técnica de aprendizaje no supervisada más común. Se utiliza para el análisis exploratorio de datos para encontrar patrones ocultos o agrupaciones en los datos. Las aplicaciones para la agrupación incluyen el análisis de secuencias de genes, la investigación de mercado y el reconocimiento de objetos.

1.3.2. Algoritmos

A continuación, se describen algunos algoritmos que forman parte del aprendizaje automático tanto de aprendizaje supervisado como no supervisado [27]:

a) **k-Vecinos más cercanos (k-NN):** Clasifica los objetos según las clases de sus vecinos más cercanos en el conjunto de datos. Las predicciones de k-NN suponen que los objetos cercanos entre sí son similares. Este algoritmo se explica a detalle en la subsección 3.5.2 y puede ser utilizado cuando:

   - Se necesite un algoritmo simple para establecer reglas de aprendizaje de referencia.
   - El uso de memoria del modelo entrenado es una preocupación menor
   - La velocidad de predicción del modelo entrenado es una preocupación menor.

b) **Regresión logística:** Se ajusta a un modelo que puede predecir la probabilidad de que una respuesta binaria pertenezca a una clase u otra. Debido a su simplicidad, la regresión logística se usa comúnmente como punto de partida para los problemas de clasificación binaria. Normalmente se usa:

   - Cuando los datos se pueden separar claramente por un solo límite lineal.
   - Como referencia para evaluar métodos de clasificación más complejos.

c) **Máquina de soporte vectorial (SVM):** Clasifica los datos encontrando el límite de decisión lineal (hiperplano) que separa todos los puntos de datos de una clase de los de la otra clase. Este algoritmo se explica a detalle en la subsección 3.5.3. Este algoritmo usualmente es usado:

- Para datos de alta dimensión, no linealmente separables.
- Cuando se necesita un clasificador que sea simple, preciso y fácil de interpretar.

d) **Naive Bayes:** Un clasificador de Bayes asume que la presencia de una característica particular en una clase no está relacionada con la presencia de cualquier otra característica. Clasifica los nuevos datos según la probabilidad más alta de que pertenezca a una clase en particular. Este algoritmo es usado:

- Para un pequeño conjunto de datos que contiene muchos parámetros.
- Cuando necesitas un clasificador que sea fácil de interpretar.
- Cuando el modelo encontrará escenarios que no estaban en los datos de capacitación, como es el caso con muchas aplicaciones financieras y médicas.

e) **Análisis discriminante:** Clasifica los datos al encontrar combinaciones lineales de características. El análisis discriminante asume que las diferentes clases generan datos basados en distribuciones gaussianas. Entrenar un modelo de análisis discriminante implica encontrar los parámetros para una distribución gaussiana para cada clase. Los parámetros de distribución se utilizan para calcular los límites, que pueden ser funciones lineales o cuadráticas. Estos límites se utilizan para determinar la clase de datos nuevos. Este algoritmo se usa cuando:

- Se necesita un modelo simple que sea fácil de interpretar.
- El uso de la memoria durante el entrenamiento es una preocupación.
- Necesitas un modelo que sea rápido de predecir.

f) **Árboles de decisión:** Permite predecir las respuestas a los datos siguiendo las decisiones en el árbol desde la raíz (principio) hasta un nodo de hoja. Un árbol consiste en condiciones de ramificación donde el valor de un predictor se compara con un peso entrenado. Se determina el número de ramas y los valores de pesos. En el proceso de formación se puede usar una modificación adicional para simplificar el modelo. Este algoritmo se usa cuando:

- Necesita un algoritmo que sea fácil de interpretar y rápido de ajustar.
- Se requiere minimizar el uso de memoria.
- La alta precisión predictiva no es un requisito.

1.3.3. Herramientas de evaluación de desempeño de algoritmos

Una vez entrenado alguno de los algoritmos del aprendizaje automático, se suele comparar los modelos obtenidos usando herramientas como la matriz de confusión o la curva ROC las cuales se describen a continuación:

Matriz de confusión

La Figura 1.12 muestra una matriz de confusión, la cual es una herramienta de visualización y diagnóstico que permite ver qué tan bien se desempeña el modelo de clasificación que se ha entrenado [28] y nos permite identificar las clases donde el clasificador ha tenido un desempeño pobre [29].

En otras palabras, las matrices de confusión son la herramienta principal para evaluar los errores de un problema de clasificación, por esta razón el aprendizaje automático supervisado típicamente hace uso de esta herramienta [30].

Cada columna de la matriz representa las predicciones hechas por el sistema de clasificación, mientras que cada fila representa la información acerca de las clasificaciones actuales. Las celdas diagonales que se encuentran coloreadas de verde representan clasificaciones correctas, mientras que las confusiones o clasificaciones erróneas se muestran de color rojo en las celdas fuera de la diagonal [31].

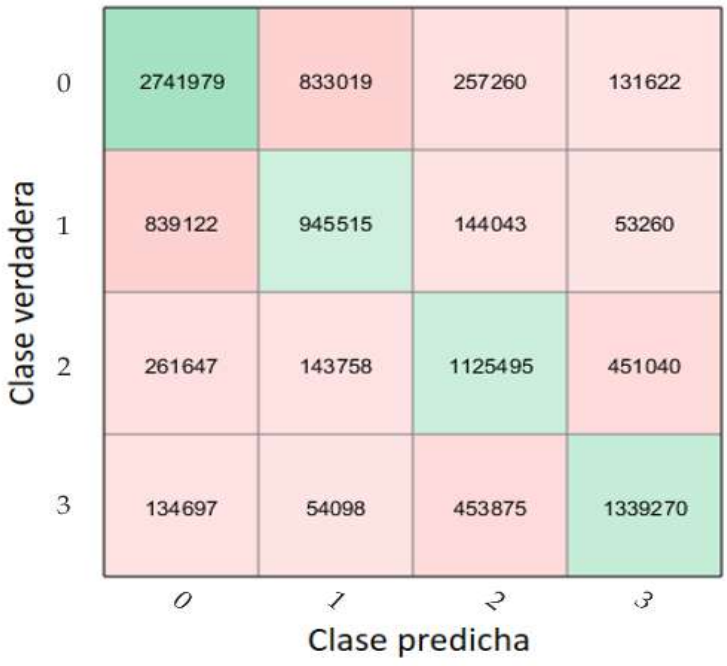


Figura 1.12: Matriz de confusión.

La información respecto a esta herramienta se complementa con la mencionada en la subsección 3.6.1.

Curva ROC

La curva ROC (*Receiver Operating Characteristic*), como se presenta en la Figura 1.13, es una herramienta visual y numérica para evaluar el rendimiento de los algoritmos de clasificación.

En ella se grafica la tasa de falsos positivos (FPR) ó 1 - especificidad (trazada en el eje horizontal) contra la tasa de verdaderos positivos ó sensibilidad (trazada en el eje vertical) [32].

Para interpretar esta curva se puede considerar como un resultado perfecto ó sin datos mal clasificados cuando se obtiene un ángulo recto (90 grados) , el marcador se encuentra en la esquina superior izquierda de la gráfica y tiene un área bajo la curva igual a 1 (AUC=1). Un resultado pobre sería una línea a 45 grados [29] que equivale a una AUC= 0.5 aproximadamente, en este caso el clasificador no es capaz de distinguir correctamente los datos.

El número área bajo curva mientras más grande sea, indicará un mejor rendimiento del clasificador.

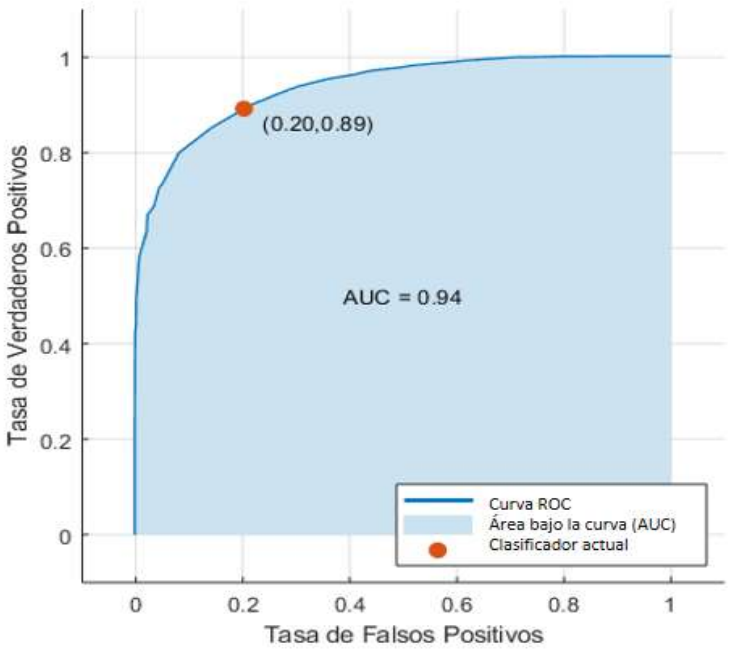


Figura 1.13: Curva ROC. La gráfica muestra una tasa de falsos positivos de 0.2, lo cual indica que el clasificador actual asigna 20 % de las muestras/observaciones incorrectamente a la clase positiva. La verdadera tasa positiva de 0.89 indica que el clasificador actual asigna 89 % de las muestras/observaciones correctamente a la clase positiva.

En este trabajo se hace uso de la matriz de confusión para evaluar el desempeño de los clasificadores.

1.4. Herramientas estadísticas

De acuerdo con [25], el aprendizaje automático tiene un enfoque probabilístico que está estrechamente relacionado con el campo de la estadística, sin embargo presenta diferencias en cuanto a terminología se refiere.

El aprendizaje automático se enfrenta a diversos problemas que son resueltos gracias a las herramientas estadísticas. Algunas de las herramientas utilizadas en este trabajo se describen a continuación.

1.4.1. Media aritmética

La media aritmética ($\overline{x}$) es un valor representativo de un conjunto de datos, se puede definir como la suma de todos los valores dividida por el número total de datos [33]. También se conoce como media ponderada o promedio y se calcula mediante (1.1):

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{1.1}$$

donde x_i representa a cada uno de los valores de la variable que van desde i hasta n, mientras que n es número total de datos.

1.4.2. Desviación estándar

La desviación estándar mide el grado de dispersión o variabilidad de los datos. Mientras mayor es la desviación estándar, mayor es la dispersión de estos.

Para obtener la desviación estándar se utiliza (1.2), en ella se calcula la desviación de cada dato respecto a su media, se elevan al cuadrado estas desviaciones para que sean positivas y se promedian, posteriormente se extrae la raíz cuadrada [34].

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}} \tag{1.2}$$

donde x_i es cada uno de los valores de la variable que van desde $i = 1$ hasta n, n es número total de datos y finalmente $\overline{x}$ representa la media aritmética.

1.4.3. Varianza

La varianza, obtenida por (1.3), representa la variabilidad de un conjunto de datos respecto a su media, en otras palabras, la varianza compara cada uno de los datos de un conjunto con la media de este.

La varianza, se calcula como el cuadrado de la desviación estándar, por lo tanto, la varianza de una muestra tiene unidades que son el cuadrado de las unidades en los datos observados mientras que la desviación estándar se encuentra en unidades lineales [35].

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1} \tag{1.3}$$

1.4.4. Covarianza

La covarianza, obtenida mediante (1.4), es un valor que indica el grado de variación conjunta de dos variables aleatorias respecto a sus medias, en otras palabras, es la media aritmética de los productos de las desviaciones de cada una de las variables respecto a sus medias [36].

$$cov(X, Y) = \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{n-1} \tag{1.4}$$

La covarianza nos permite descubrir si las variables mantienen un vínculo de dependencia.

Resumen de fin de capítulo

En este capítulo se realizó una revisión breve de los conceptos relacionados con los aerogeneradores, el aprendizaje automático y las herramientas estadísticas que nos permiten entender y encontrar solución a problemas relacionados a él. Lo anterior se usará en capítulos posteriores para realizar diversos cálculos como la normalización de los datos y el análisis de componentes principales.

Capítulo 2

Banco de pruebas experimental

En el CoDAlab (Laboratorio de Control, Dinámica y Aplicaciones), actualmente se encuentra la infraestructura mostrada en la Figura 2.1. Esta es utilizada para realizar el ensayo experimental y se describe a detalle en el presente capítulo.

Figura 2.1: Ensayo experimental aerogenerador *offshore* con soporte tipo jacket en el laboratorio.

La estructura se fabricó manteniendo similitud con la turbina eólica *offshore* de referencia [24].

Para las pruebas experimentales se ha multiplicado la amplitud de una señal eléctrica tipo ruido en un generador de funciones, éstas modificaciones en la amplitud se realizan con el fin de simular diferentes velocidades del viento las cuales serán producidas por un

agitador inercial. La señal proveniente del generador de funciones entra al amplificador que acompaña al agitador inercial. El amplificador regula la señal producida por el generador de funciones y posteriormente la señal de salida del amplificador pasa a ser la entrada del agitador inercial. Este es el encargado de generar vibraciones similares a las producidas por las rachas de vientos sobre las palas del aerogenerador a partir de una señal eléctrica (ruido blanco). El agitador inercial se encuentra colocado en la barra superior de la plataforma, de esta manera se simula la masa de la góndola y los efectos ambientales del viento sobre toda la estructura a escala, la cual es capaz de recrear situaciones que se encontrarían en una estructura de tamaño real. Existen 8 sensores triaxiales que detectan las vibraciones en la estructura, los cables de salida de los acelerómetros tienen 3 salidas (x, y y z) estas salidas se conectan al sistema de adquisición de datos. Con ayuda de NI-MAX (*National Instruments*), *Analog Input Recorder* y MATLAB© se adquieren los datos, se preprocesan, se guardan y finalmente en la herramienta *Classification Learner* se entrenan diferentes modelos del aprendizaje supervisado y evalúan los resultados obtenidos.

El banco de pruebas experimental utilizado se muestra de forma general en la Figura 2.2.

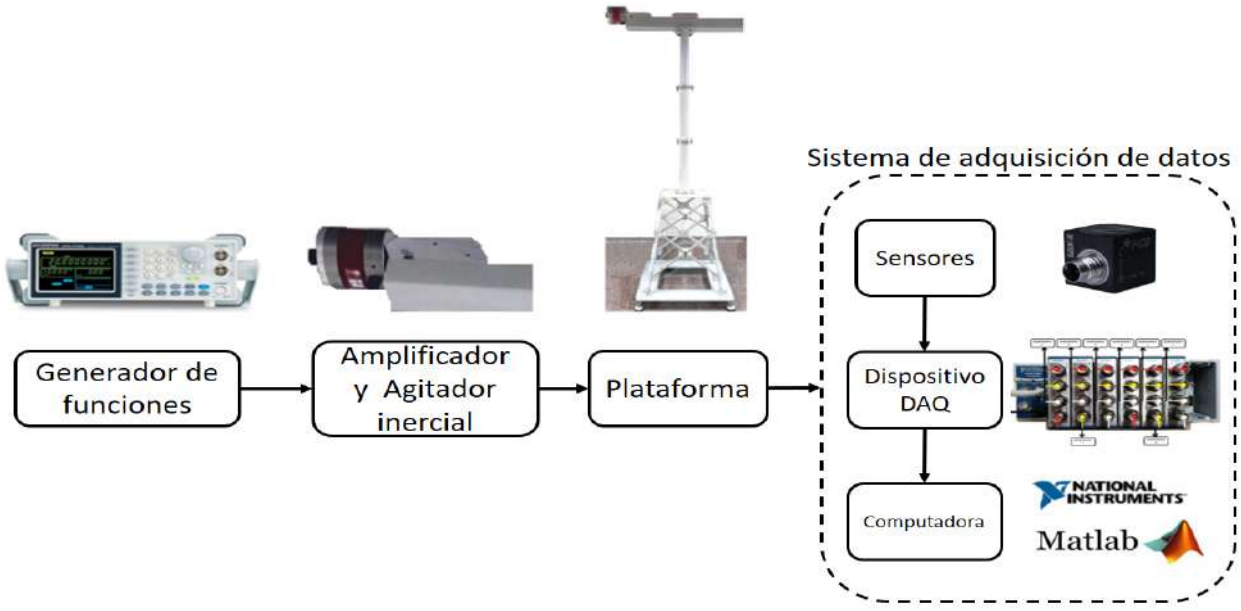


Figura 2.2: Descripción general del banco de pruebas experimental.

En las siguientes subsecciones se describen los diferentes elementos que conforman el banco de pruebas.

2.1. Generador de funciones

Los generadores de funciones son aparatos encargados de producir señales con un voltaje específico aplicado durante un tiempo específico. En este trabajo se utiliza el modelo GW INSTEKAF-2005. Para realizar las pruebas experimentales se selecciona la señal de ruido blanco, este tipo de señal es similar a las producidas por las rachas de viento sobre las palas del aerogenerador. Se simularon diferentes velocidades del viento multiplicando la amplitud de la señal de ruido blanco (en el generador de funciones) por los factores 0.5, 1, 2 y 3.

2.2. Amplificador y agitador inercial

El amplificador de potencia Modelo PA300E de Data Physics mostrado en la Figura 2.3a se utiliza con el fin de transmitir y regular la señal de ruido previamente producida por el generador de funciones, este dispone de un regulador de potencia que al no estar escalado se mantiene fijo. Por esa razón, los distintos vientos se simulan a través de multiplicarla señal del generador de funciones por un factor determinado.

Posteriormente la señal del amplificador es transmitida al agitador inercial GW-IV47 de Data Physics (ver Figura 2.3b), encargado de producir las vibraciones necesarias para excitar la plataforma al aplicar una señal eléctrica (ruido blanco) y así recrear los efectos ambientales del viento. Cabe mencionar que el eje central del agitador inercial se une a la estructura puesta a prueba.

En la Figura 2.4 es posible observar la ubicación del amplificador y el agitador inercial dentro del banco de pruebas experimental.

(a) Amplificador PA300E.

(b) Agitador inercial GW-IV47.

Figura 2.3: Amplificador y agitador inercial ambos de la marca Data Physic usados en el banco de prueba experimental.

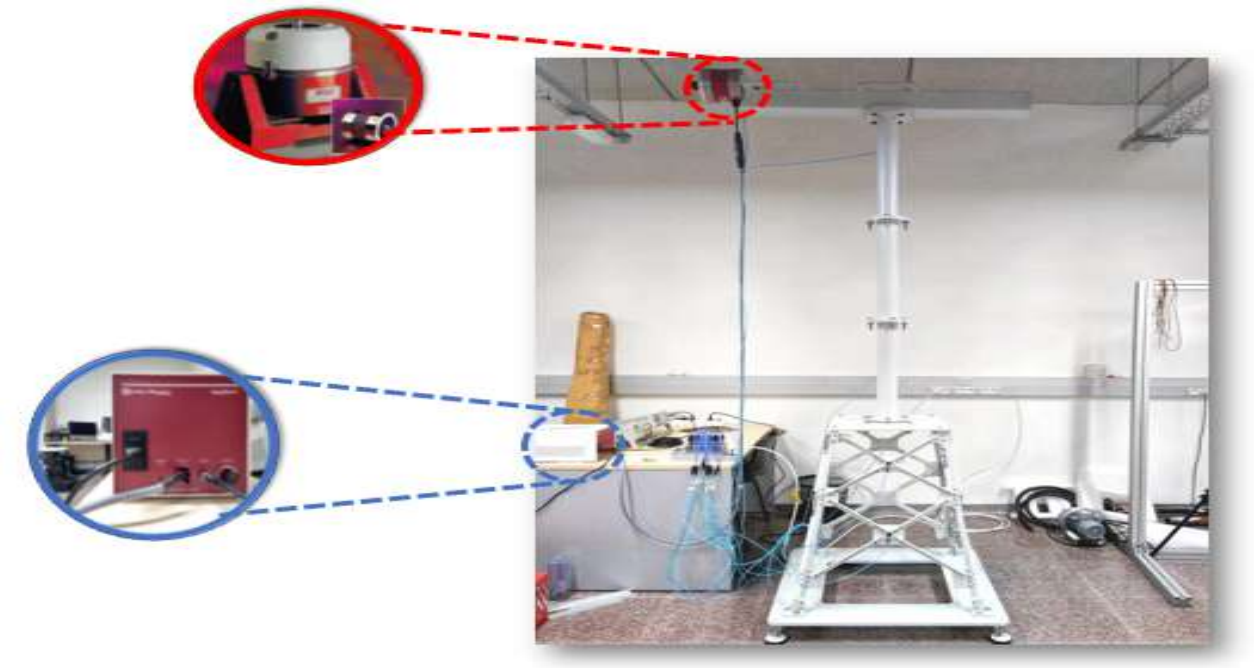


Figura 2.4: Ubicación del amplificador y agitador inercial. El circulo azul resalta el amplificador mientras que el círculo rojo resalta el agitador inercial.

2.3. Plataforma

La plataforma a escala se fabricó manteniendo de referencia la turbina eólica *offshore* fabricada por el Laboratorio Nacional de Energías Renovables [24]. Ésta tiene una altura total de 2.7m y como se muestra en la Figura 2.5 (izquierda) consta de 3 partes principales:

- **Góndola:** Representada por una viga en la parte superior de la estructura. Tiene 1m de largo y 0.6m de ancho, allí se encuentra un agitador inercial que simula la masa de la góndola.
- **Torre:** Se compone de tres secciones tubulares unidas con pernos a un torque 125Nm.
- **Soporte *jacket*:** Es la estructura piramidal que se compone de varias secciones (barras), todas ellas unidas con pernos a un torque de 12Nm. Se localiza en la parte inferior de la plataforma por lo cual, en una plataforma de tamaño real es la parte que quedaría sumergida en el océano. En particular, el soporte *jacket* tiene cuatro longitudes de barra diferentes, cada una a diferentes niveles (profundidad).

Para el caso práctico presentado, se realizaron pruebas con 4 casos: barra original saludable, barra réplica saludable, barra con *crack*/fisura y barra con perno flojo. Para dichas

pruebas se manipuló la barra localizada en el nivel 2 del soporte *jacket* como se muestra en la Fig.2.5.

Para el daño de *crack*/fisura se realizó un corte de 5mm en la barra antes mencionada. Se ha comprobado que las fisuras son uno de los principales tipos de daños encontrados en las plataformas de aerogeneradores *offshore* [37].

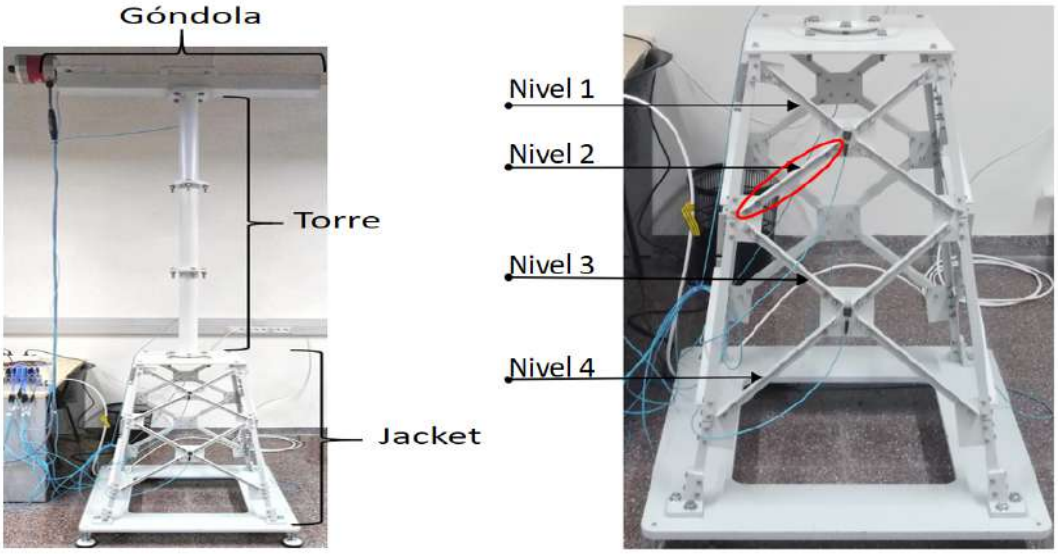


Figura 2.5: Izquierda: Modelo a escala de la plataforma de un aerogenerador *offshore* con soporte tipo *jacket* utilizada en las pruebas experimentales. Derecha: Niveles del soporte *jacket*, el círculo rojo muestra la barra manipulada para realizar las pruebas.

2.4. Sistema de adquisición de datos

La adquisición de datos (*Data Acquision*, *DAQ* por sus siglas en inglés), es el proceso de medir con una computadora un fenómeno eléctrico o físico como voltaje, corriente, temperatura, presión o sonido mediante dispositivos digitales.

Generalmente en un sistema DAQ se identifican los siguientes elementos principales: *hardware*, *software*, computadora y sensores.

A continuación, se detallan los componentes del sistema de adquisición de datos utilizado en este trabajo.

2.4.1. Sensores

Se colocaron 8 sensores triaxiales modelo 356A17 de la marca PCB Piezotronicbla en la plataforma de manera estratégica, para detectar cualquier anomalía en el comportamiento

dinámico de la estructura. Son acelerómetros cerámicos y de alta sensibilidad, que tienen una sensibilidad de voltaje fijo, independientemente del tipo de cable utilizado o su longitud y su señal de salida es de baja impedancia, por lo que puede transmitir a través de cables largos en entornos hostiles sin perder la calidad de la señal. El dispositivo está acompañado por un cable que trifurca dando una salida para cada componente espacial x, y y z, esto se observa en la Figura 2.6, por lo tanto, se obtienen datos de 24 señales de salida.

Figura 2.6: Acelerómetro triaxial PCB Piezotronics, model 356A17 y cable trifurcado complemento del acelerómetro.

La ubicación óptima de los sensores se presenta en la Figura 2.7 y se encuentra determinada de acuerdo al método de eliminación del sensor por criterio de aseguramiento modal (SEAMAC por sus siglas en inglés) [16].

Figura 2.7: Ubicación de los sensores en plataforma a escala.

2.4.2. Dispositivo DAQ

Los dispositivos que hacen posible la adquisición de datos son el chasis cDAQ-9188 y 6 módulos NI-9234 de National Instrument mostrados en la Figura 2.8.

(a) Chasis cDAQ-9188. (b) Cartucho NI-9234.

Figura 2.8: Dispositivos de National Instruments para realizar la adquisión de datos.

- **Chasis cDAQ-9188:** es un chasis Ethernet CompactDAQ, que consta de 8 ranuras de entrada, cada ranura puede recibir hasta 4 señales diferentes [38]. El chasis es capaz de controlar la temporización, la sincronización y la transferencia de datos entre los módulos de E/S de la Serie C y un servidor externo.
- **Modulo NI-9234:** pueden medir señales de sensores IEPE (piezoeléctricos electrónicos integrados) y no IEPE como acelerómetros, tacómetros y sensores de proximidad. Este cartucho proporciona un amplio rango dinámico e incorpora acoplamiento de AC/DC y acondicionamiento de señales IEPE seleccionable por software [39].

Para la adquisición de datos en este trabajo, cada una de las salidas de los sensores (x, y, z) va conectada a las entradas de los módulos NI-9234. A su vez los módulos están insertados en el chasis cDAQ-9188. Estos dispositivos captan cada una de las componentes cartesianas recibidas de los acelerómetros (ver Figura 2.9).

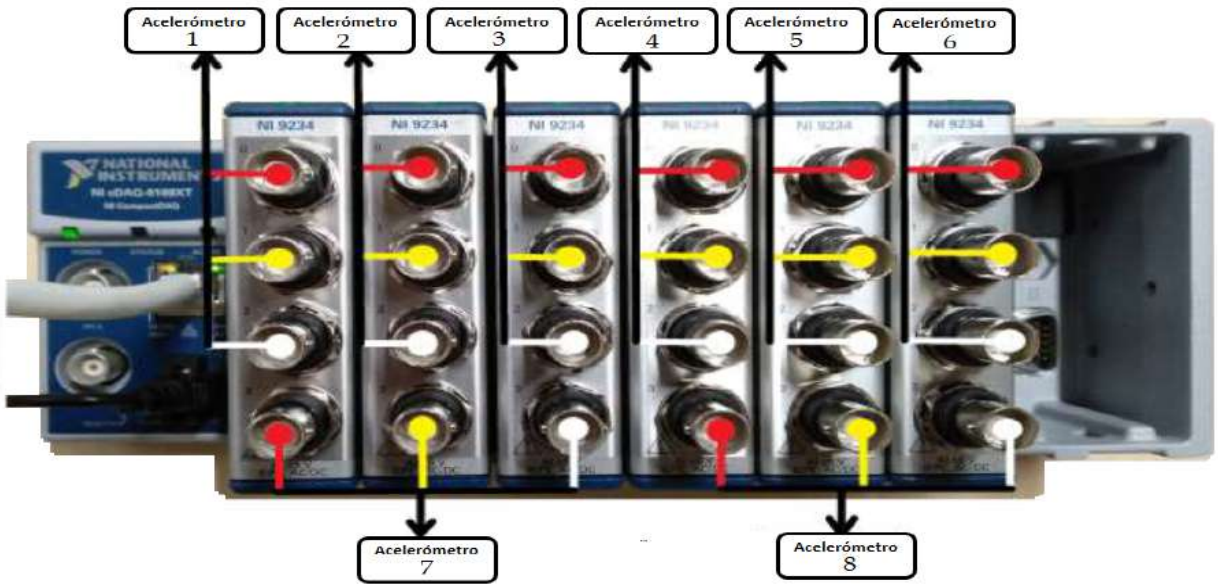


Figura 2.9: Conexión de los sensores al NI-9234 y cDAQ-9188. Las señales de salida de los sensores se señalan como x (linea roja), y (linea amarilla) y z (linea blanca).

2.4.3. *National Instruments*: NI-MAX

Measurement & Automation Explorer (NI-MAX) es un software que pertenece a *National Instruments*, transcribe las señales del dispositivo DAQ, compatibilizándolo para este trabajo con el programa MATLAB©. Una vez realizada la sincronización, obtendremos los datos directamente en la interfaz de MATLAB©.

2.4.4. MATLAB©: Analog Input Recorder

La aplicación *Analog Input Recorder* es una herramienta para adquirir señales analógicas de dispositivos de adquisición de datos directamente en MATLAB© (Ver Figura 2.10). Esta aplicación detecta automáticamente los dispositivos DAQ disponibles, permite controlar los parámetros de adquisición, muestra una vista previa en vivo de los datos y permite grabar datos directamente en el espacio de trabajo de MATLAB© para su posterior análisis y visualización. La aplicación también es capaz de generar un código MATLAB© que puede ser usado en el futuro.

Esta herramienta nos ofreció como frecuencia mínima de muestreo 1,651.6166 Hz, por lo que en este trabajo al realizar experimentos con duración de 60 segundos obtuvimos 99,097 muestras.

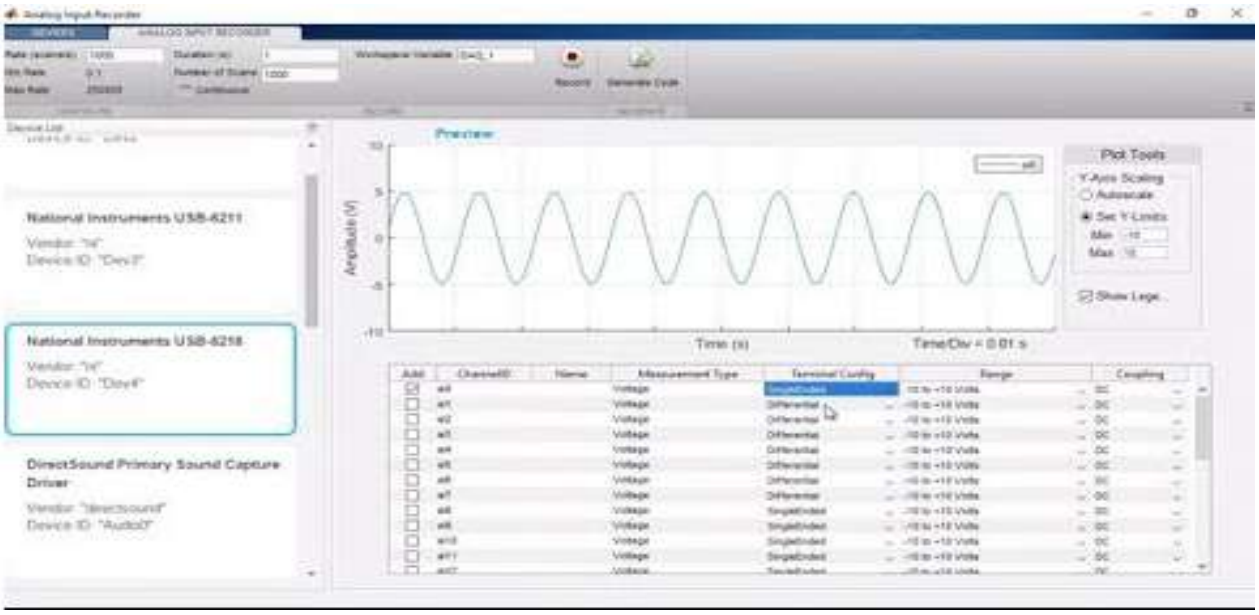


Figura 2.10: Interfaz de la herramienta *Analog Input Recorder* en MATLAB©

En el trabajo presentado, para la adquisición de datos se usaron en conjunto los programas NI-MAX de *National Instruments* y la herramienta *Analog Input Recorder* de MATLAB©, de esta manera se logró la comunicación con cDAQ-9188.

Para ello se ejecutan los siguientes pasos:

1. Abrir MATLAB© y esperar a que en la parte inferior izquierda se muestre la leyenda "*Ready*".

2. Abrir NI-MAX.

   - En la parte izquierda superior ir a "*My system*">> "*Network Devices*">>"NI cDAQ-9188XT".

   - En el menú superior ejecutar el *self-test*. Asegurarse que se muestre la leyenda "*the self-test completed succesfully*".

   - Minimizar la ventana.

3. Regresar a MATLAB© y escribir en *command window*: "daq.getDevices".

4. Abrir la app *Analog Input Recorder*. Está se encuentra en: menú Apps>>*Test and Measures*>> *Analog Input Recorder*.

   - En el lado izquierdo de la ventana se elige el dispositivo.

- Se configura el número de muestras o tiempo de muestreo.
- Elegir el botón "*Record*"

Como se mencionó en la subsección 2.4.4, se puede obtener un código generado por la herramienta *Analog Input Recorder*, para fines de este trabajo así se realizó. Una vez obtenido dicho código se realizaron los ajustes necesarios para obtener los datos a usar y guardarlos en ficheros, el código modificado cuenta con las siguientes características:

- Determina el número de muestras o tiempo de muestreo de cada experimento.
- Agrega los canales que captan las señales emitidas por cada uno de los sensores. (En nuestro caso son 3 canales por cada sensor lo cual da un total de 24 canales).
- Renombrar los canales.
- Inicia la adquisición, asignándole el nombre *"data"* a el conjunto conjunto de datos que se obtiene de registrar los datos obtenidos de los sensores.
- Limpiar la sesión y los canales para una nueva toma de datos.
- Guardar los datos en ficheros .mat

El código completo se puede ver en el apéndice A.1

2.4.5. MATLAB©: Classification Learner

La aplicación *Classification Learner* de MATLAB© (ver Figura 2.11) permite realizar tareas comunes de aprendizaje supervisado, así como explorar de forma interactiva los datos, seleccionar funciones, especificar esquemas de validación, modelos de entrenamiento y evaluar resultados.

También tiene la capacidad de generar código MATLAB© para integrar modelos a diferentes aplicaciones [40] y exportarlos al espacio de trabajo de MATLAB© para futuros entrenamientos.

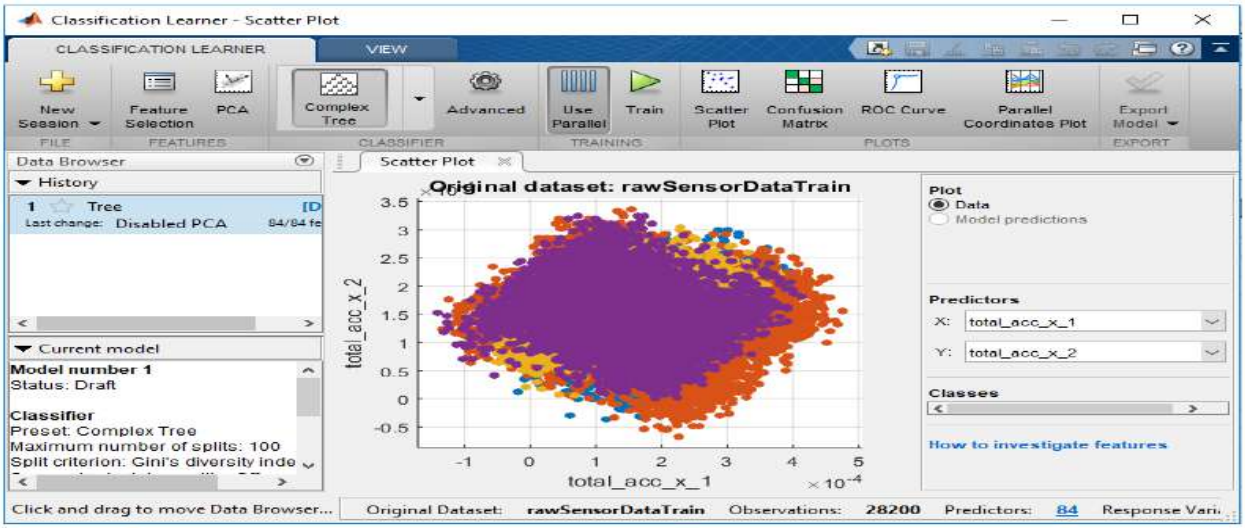


Figura 2.11: Interfaz visual de "*Classification Learner*".

Esta herramienta se puede encontrar en el menú Apps de MATLAB©. Para empezar a trabajar en ella, se elige la opción "*New Session*" que se localiza en la barra superior del lado izquierdo. Posteriormente se debe elegir la base de datos que se desean entrenar. También se elige el esquema de validación, los cuales calculan el rendimiento del modelo en los nuevos datos en comparación con los datos de entrenamiento y ayuda a poder elegir el mejor modelo, además que la validación protege contra el sobre ajuste. Los esquemas con los que *Classification Learner* cuenta son:

a) **Validación cruzada (*Cross Validation*):** Selecciona un número de divisiones para particionar un conjunto de datos. Este método proporciona una buena estimación de la precisión predictiva del modelo final entrenado con todos los datos [41] y se explica a detalle en la sección 3.5.1.

b) **Validación reservada (*Hold Validation*):** Este tipo de validación se recomienda para grandes conjuntos de datos, ya que se basa solo en una parte de todos los datos. Al igual que que en la validación cruzada, se elige un conjunto de prueba, la aplicación entrena al modelo en el conjunto de entrenamiento y evalúa el rendimiento con el punto de prueba.

c) **Sin validación (*No Validation*):** En este método no existe protección contra el sobreajuste. No toma ningún conjunto de datos de prueba para obtener una estimación de la precisión predictiva por lo que no es recomendable.

Resumen de fin de capítulo

En este capítulo se explicaron los elementos tanto de *software* como de *hardware* que conforman el banco de pruebas experimental utilizado para obtener datos de diferentes daños estructurales que un aerogenerador *offshore* con una plataforma tipo *jacket* puede presentar tales como tener un perno flojo o presentar una fisura. También se describió el funcionamiento de dicho banco de pruebas y el cómo se obtienen y guardan datos para posteriormente ser procesados. A los datos obtenidos se les realizará la aplicación de la metodología propuesta y descrita en el capítulo 3.

Capítulo 3

Metodología

3.1. Recopilación de los datos

En el banco de pruebas de laboratorio se han considerado 4 estados estructurales diferentes en las barras del soporte tipo *jacket*: barra original en estado saludable, réplica de la barra original en estado saludable, barra con un *crack* o fisura y barra con un perno flojo. Para cada uno de estos estados se hicieron pruebas con las 4 amplitudes de ruido blanco: 0.5, 1, 2 y 3 obteniendo un total de 100 experimentos/pruebas como se muestra en la Tabla 3.1, cada uno con una duración de 60 segundos.

Caso	**Amplitud**				**Total de**
	0.5	**1**	**2**	**3**	**experimentos**
Barra original	10	10	10	10	40
Barra réplica	5	5	5	5	20
Barra con *crack*/fisura	5	5	5	5	20
Barra con perno flojo	5	5	5	5	20
					100

Tabla 3.1: Experimentos/pruebas realizadas para cada estado estructural y amplitud.

Los datos recopilados en cada experimento mediante *Analog Input Recorder* fueron guardados en ficheros de MATLAB©. Con la herramienta *Analog Input Recorder* se obtienen las muestras. El paso de tiempo ("*time step*") usado para tomar los datos es aproximadamente $T_1 = 6.0547e^{-04}$ ya que está limitado por el tipo de sensor. Los ficheros tienen nombres del estilo 1_7_1A.mat. En este caso, significa que es una toma de datos asociada al caso saludable (barra original) porque el fichero empieza por 1, luego el número 7 indica que es la toma de datos número 7, y por último el 1A indica la amplitud de ruido usada. Los nombres

de los ficheros fueron asignados dependiendo del estado estructural al cual pertenece cada experimento, esto se detalla en la Tabla 3.2.

Caso	**Nombre del fichero**
Barra original	Empieza por 1
Barra réplica	Empieza por 2
Barra con *crack*/fisura	Empieza por 3
Barra con perno flojo	Empieza por 4

Tabla 3.2: Asignación de nombre a ficheros pertenecientes a cada experimento.

En cada fichero ".mat" hay una base de datos llamada "data" (ver Figura 3.1). Esta tiene 99,097 filas y 24 columnas. Cada columna está asociada a un sensor (acelerómetro triaxial). El número de filas corresponde a distintos instantes de tiempo, al haber tomado datos durante 60 segundos con un "*time step*" $T_1 = 6.0547e^{-04}$ tenemos $\frac{60}{6.0547e^{-04}} \approx 99,097$.

En la Figura 3.2 se muestra la representación gráfica los datos recopilados del primer sensor, dentro de la base de datos "data" se encuentran recopilados en la primera columna.

data

99097x24 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	3.4766e-04	2.1276e-04	1.7186e-04	2.1933e-04	2.2777e-04	1.9664e-04	1.0476e-04	1.7118e-04	3.1045e-04	6.7658e-05	3.1305e-04	2.7955e-04	1.4704e-04	8.3768e-05	3.6793e-04	2.7297e-04	1.7909e-05
2	8.1458e-05	2.4871e-04	2.0541e-04	1.2910e-04	1.3874e-04	1.4603e-04	1.1024e-04	1.6021e-04	3.3238e-04	2.3466e-04	2.8926e-04	2.7346e-04	2.2633e-04	3.3139e-05	4.1734e-04	2.2662e-04	7.5804e-06
3	2.2250e-04	3.9497e-04	2.3896e-04	1.0227e-04	2.5399e-04	2.2591e-04	1.4494e-04	2.1751e-04	2.9644e-04	1.8285e-04	2.8865e-04	2.2046e-04	1.5314e-04	4.4729e-05	3.6122e-04	1.6443e-04	7.1014e-05
4	2.2494e-04	2.9807e-04	2.2920e-04	1.7482e-04	2.7229e-04	1.1128e-04	1.5833e-04	2.9005e-04	3.5918e-04	1.0849e-04	2.7646e-04	2.2899e-04	1.1654e-04	3.0699e-05	3.5634e-04	2.4736e-04	2.6118e-07
5	1.6633e-04	2.4750e-04	2.7739e-04	1.9982e-04	1.7716e-04	1.9542e-04	1.0171e-04	1.7118e-04	3.0375e-04	9.2648e-05	3.0573e-04	3.0087e-04	2.0742e-04	3.3139e-05	3.8440e-04	2.8333e-04	1.3680e-05
6	1.5900e-04	2.6212e-04	2.0480e-04	-2.5999e-06	1.0276e-04	1.3627e-04	7.4929e-05	1.3278e-04	3.3542e-04	1.7066e-04	3.0268e-04	2.7468e-04	2.4767e-04	9.1087e-05	3.8318e-04	2.7541e-04	4.2957e-05
7	1.4496e-04	3.2184e-04	2.2859e-04	2.6628e-04	2.9363e-04	2.3749e-04	1.5833e-04	2.1020e-04	3.2324e-04	1.7676e-04	2.5450e-04	2.0828e-04	9.7025e-05	2.3379e-05	3.7159e-04	1.5162e-04	2.0999e-05
8	2.4753e-04	3.8400e-04	2.1395e-04	1.9067e-04	2.6314e-04	2.2164e-04	1.9486e-04	2.3092e-04	2.5440e-04	1.0606e-04	3.3561e-04	2.9722e-04	1.5375e-04	9.6577e-05	3.5268e-04	1.7906e-04	4.5397e-05
9	2.1700e-04	2.9868e-04	1.8955e-04	9.8610e-05	2.2472e-04	2.4115e-04	1.0658e-04	2.0410e-04	3.3116e-04	1.4811e-04	2.9353e-04	2.4909e-04	1.9095e-04	7.3398e-05	3.7891e-04	2.6199e-04	4.6007e-05
10	2.1090e-04	2.6456e-04	2.7495e-04	2.0836e-04	2.2655e-04	1.7042e-04	5.7883e-05	1.4375e-04	3.2933e-04	1.9200e-04	2.7768e-04	2.4909e-04	1.8241e-04	3.1919e-05	4.4235e-04	2.9248e-04	2.7098e-05
11	2.1151e-04	2.5054e-04	1.3770e-04	1.8580e-04	1.7350e-04	1.6798e-04	1.3824e-04	2.0959e-04	3.0070e-04	1.2007e-04	2.6609e-04	2.3935e-04	1.3179e-04	1.7889e-05	3.3377e-04	1.3333e-04	6.7354e-05
12	1.8831e-04	3.6328e-04	3.0972e-04	8.2148e-05	1.9606e-04	2.1981e-04	2.1007e-04	2.1264e-04	2.9644e-04	1.4324e-04	3.1549e-04	1.9183e-04	1.9278e-04	3.3139e-05	4.0880e-04	1.6199e-04	-9.5870e-07
13	1.1626e-04	1.9814e-04	2.0846e-04	1.7726e-04	1.9545e-04	1.0579e-04	9.0757e-05	2.0959e-04	3.1593e-04	1.5299e-04	2.6365e-04	2.6432e-04	1.4155e-04	6.8518e-05	2.9961e-04	2.2662e-04	2.0389e-05
14	3.2202e-04	2.8040e-04	2.1456e-04	1.2422e-04	2.3448e-04	2.1066e-04	6.7624e-05	1.8642e-04	3.3969e-04	1.5177e-04	2.6060e-04	2.9234e-04	1.6107e-04	3.2529e-05	3.8318e-04	3.3333e-04	-1.2548e-05
15	1.3946e-04	2.8589e-04	2.9691e-04	1.7665e-04	2.3692e-04	2.5883e-04	1.4311e-04	1.5533e-04	3.0862e-04	1.9139e-04	3.1366e-04	3.0209e-04	2.4523e-04	8.1328e-05	3.7403e-04	1.3638e-04	8.5653e-05
16	5.7490e-06	2.8284e-04	1.7491e-04	1.2361e-04	1.9789e-04	1.2225e-04	1.5042e-04	1.9252e-04	2.6476e-04	1.9443e-04	2.8439e-04	2.4239e-04	1.2752e-04	1.1789e-05	3.4475e-04	1.6687e-04	3.0758e-05
17	2.9943e-04	1.9570e-04	1.5844e-04	2.1567e-04	1.9789e-04	2.1676e-04	1.2059e-04	2.3641e-04	3.0984e-04	7.9848e-05	2.6426e-04	2.5214e-04	9.5195e-05	7.2788e-05	3.9477e-04	2.7906e-04	5.1407e-06

Figura 3.1: Visualización de los datos contenidos en la base de datos "data".

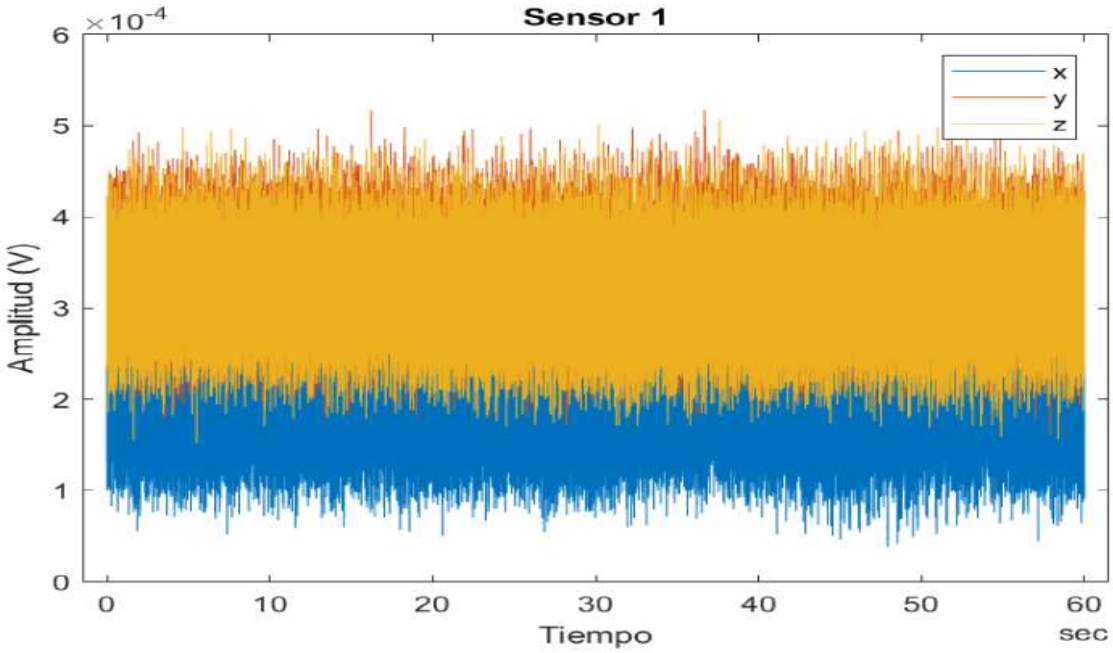


Figura 3.2: Gráfica de los datos contenidos en sensor 1.

Cabe recordar que los datos provienen de un caso no lineal y no estacionario, ya que la excitación usada (ruido blanco) simula un viento real con turbulencia.

3.2. Submuestreo y remodelación de los datos

Como se menciona anteriormente, cada experimento ($k = 1, 2, 3, \ldots, 100$) se guarda en un fichero ".mat" que contiene la base de datos "data", esta la podemos ver como una matriz (3.1) renombrada $\mathbf{Y}^{(k)}$ de dimensiones $99,097 \times 24$.

$$\mathbf{Y}^{(k)} = \begin{pmatrix} y_{1,1}^{(k)} & y_{1,2}^{(k)} & \cdots & y_{1,24}^{(k)} \\ y_{2,1}^{(k)} & y_{2,2}^{(k)} & \cdots & y_{2,24}^{(k)} \\ y_{3,1}^{(k)} & y_{3,2}^{(k)} & \cdots & y_{3,24}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ y_{99097,1}^{(k)} & y_{99097,2}^{(k)} & \cdots & y_{99097,24}^{(k)} \end{pmatrix} \tag{3.1}$$

Si se pretendiera unir la información de los 100 experimentos y formar una única matriz (3.2) para posteriormente procesar los datos, obtendríamos una matriz de dimensiones $9,909,700$ filas y 24 columnas que representan un total de $237,832,800$ datos en crudo.

$$\mathbf{Y} = \begin{pmatrix} \mathbf{Y}^{(1)} \\ \vdots \\ \mathbf{Y}^{(k)} \\ \vdots \\ \mathbf{Y}^{(100)} \end{pmatrix} \tag{3.2}$$

Esta cantidad de información resulta un problema para los equipos de cómputo comunes, que son incapaces de procesarlos y por esta razón es necesario realizar un submuestreo. El submuestreo que se describe a continuación se realizó para cada uno de los experimentos.

Para los sensores se ha considerado una frecuencia de muestreo de 275 Hz, factible para un entorno en alta mar [42]. Para lograr esto se debe tener un paso de tiempo igual a $\mathrm{T}_2 = 0.0036s$. En la toma inicial de datos el paso de tiempo $\mathrm{T}_1 = 6.0547e^{-04}$ por lo tanto el factor que buscamos para realizar el submuestreo es 6 de acuerdo a (3.3).

$$\text{factor} = \frac{\mathrm{T}_2}{\mathrm{T}_1} = \frac{0.0036\text{s}}{0.00060547\text{s}} = 5.95 \approx 6 \tag{3.3}$$

Se realiza el submuestreo para cada $\mathbf{Y}^{(k)}$ con el nuevo paso de tiempo y se forma la matriz $\mathbf{Z}^{(k)}$ (3.4) detallada a continuación:

$$\mathbf{Z}^{(k)} = \begin{pmatrix} y_{1,1}^{(k)} & y_{1,2}^{(k)} & \cdots & y_{1,24}^{(k)} \\ y_{7,1}^{(k)} & y_{7,2}^{(k)} & \cdots & y_{7,24}^{(k)} \\ y_{13,1}^{(k)} & y_{13,2}^{(k)} & \cdots & y_{13,24}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ y_{99097,1}^{(k)} & y_{99097,2}^{(k)} & \cdots & y_{99097,24}^{(k)} \end{pmatrix} = \begin{pmatrix} z_{1,1}^{(k)} & z_{1,2}^{(k)} & \cdots & z_{1,24}^{(k)} \\ z_{2,1}^{(k)} & z_{2,2}^{(k)} & \cdots & z_{2,24}^{(k)} \\ z_{3,1}^{(k)} & z_{3,2}^{(k)} & \cdots & z_{3,24}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ z_{16517,1}^{(k)} & z_{16517,2}^{(k)} & \cdots & z_{16517,24}^{(k)} \end{pmatrix} \tag{3.4}$$

Por tanto, la matriz obtenida después del submuestreo tiene dimensiones $16,517$ (instantes de tiempo) filas y 24 (sensores) columnas.

Una vez determinada $\mathbf{Z}^{(k)}$ se reorganizan los datos en muestras que contienen determinados instantes de tiempo. Los instantes de tiempo que cada muestra contendrá se determina calculando los factores de $16,517$ (estos son 199 y 83) para garantizar trabajar con números enteros y con ello usar toda la información posible.

Se eligió tener conjuntos de 199 instantes de tiempo tomados de $\mathbf{Z}^{(k)}$. Finalmente, se reorganizan los datos en una nueva matriz denominada $\tilde{\mathbf{Z}}^{(k)}$ mostrada en (3.5), que tiene 83 muestras y 199 instantes de tiempo multiplicados por cada uno de los 24 sensores, esto da

una dimensión de $\tilde{\mathbf{Z}}^{(k)}$ de dimensiones $83 \times 4,776$ [43].

$$\tilde{\mathbf{Z}}^{(k)} = \left(\begin{array}{cccc|cccc|c|cccc} \overbrace{}^{1} & & & & \overbrace{}^{2} & & & & & \overbrace{}^{3} & & & & \overbrace{}^{24} \\ z_{1,1}^{(k)} & z_{2,1}^{(k)} & \cdots & z_{199,1}^{(k)} & z_{1,2}^{(k)} & z_{2,2}^{(k)} & \cdots & z_{199,2}^{(k)} & & z_{1,3}^{(k)} & z_{2,3}^{(k)} & \cdots & z_{199,3}^{(k)} & & z_{1,24}^{(k)} & z_{2,24}^{(k)} & \cdots & z_{199,24}^{(k)} \\ z_{200,1}^{(k)} & z_{201,1}^{(k)} & \cdots & z_{398,1}^{(k)} & z_{200,2}^{(k)} & z_{201,2}^{(k)} & \cdots & z_{398,2}^{(k)} & & z_{200,3}^{(k)} & z_{201,3}^{(k)} & \cdots & z_{398,3}^{(k)} & & z_{200,24}^{(k)} & z_{201,24}^{(k)} & \cdots & z_{398,24}^{(k)} \\ z_{399,1}^{(k)} & z_{400,1}^{(k)} & \cdots & z_{597,1}^{(k)} & z_{399,2}^{(k)} & z_{400,2}^{(k)} & \cdots & z_{597,2}^{(k)} & & z_{399,3}^{(k)} & z_{400,3}^{(k)} & \cdots & z_{597,3}^{(k)} & & z_{399,24}^{(k)} & z_{400,24}^{(k)} & \cdots & z_{597,24}^{(k)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & & \vdots & \vdots & \ddots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ z_{16318,1}^{(k)} & z_{16319,1}^{(k)} & \cdots & z_{16517,1}^{(k)} & z_{16318,2}^{(k)} & z_{16319,2}^{(k)} & \cdots & z_{16517,2}^{(k)} & & z_{16318,3}^{(k)} & z_{16319,3}^{(k)} & \cdots & z_{16517,3}^{(k)} & & z_{16318,24}^{(k)} & z_{16319,24}^{(k)} & \cdots & z_{16517,24}^{(k)} \end{array} \right) \tag{3.5}$$

Finalmente obtenemos una matriz $\mathbf{X}$ (3.6) de dimensiones $8,300 \times 4,776$. Esta matriz tiene los datos con los cuales se realizará la extracción de características y posteriormente procesar los datos.

$$\mathbf{X} = \begin{pmatrix} \tilde{\mathbf{Z}}^{(1)} \\ \vdots \\ \tilde{\mathbf{Z}}^{(k)} \\ \vdots \\ \tilde{\mathbf{Z}}^{(100)} \end{pmatrix} \tag{3.6}$$

Cabe mencionar que las matrices $\tilde{\mathbf{Z}}^{(k)}$ que le corresponden a cada caso estructural se muestran a continuación:

Caso	**Matrices correspondientes**
Barra original	$\tilde{\mathbf{Z}}^{(1)}, \cdots, \tilde{\mathbf{Z}}^{(40)}$
Barra réplica	$\tilde{\mathbf{Z}}^{(41)}, \cdots, \tilde{\mathbf{Z}}^{(60)}$
Barra con *crack*/fisura	$\tilde{\mathbf{Z}}^{(61)} \cdots, \tilde{\mathbf{Z}}^{(80)}$
Barra con perno flojo	$\tilde{\mathbf{Z}}^{(81)} \cdots, \tilde{\mathbf{Z}}^{(100)}$

Los cálculos se realizaron mediante un programa en MATLAB©, el diagrama de flujo se

muestra en la Figura 3.3. En él solo se muestra el proceso para el caso saludable. Para obtener la matriz $\mathbf{X}$ se ejecuta el programa para cada estado estructural.

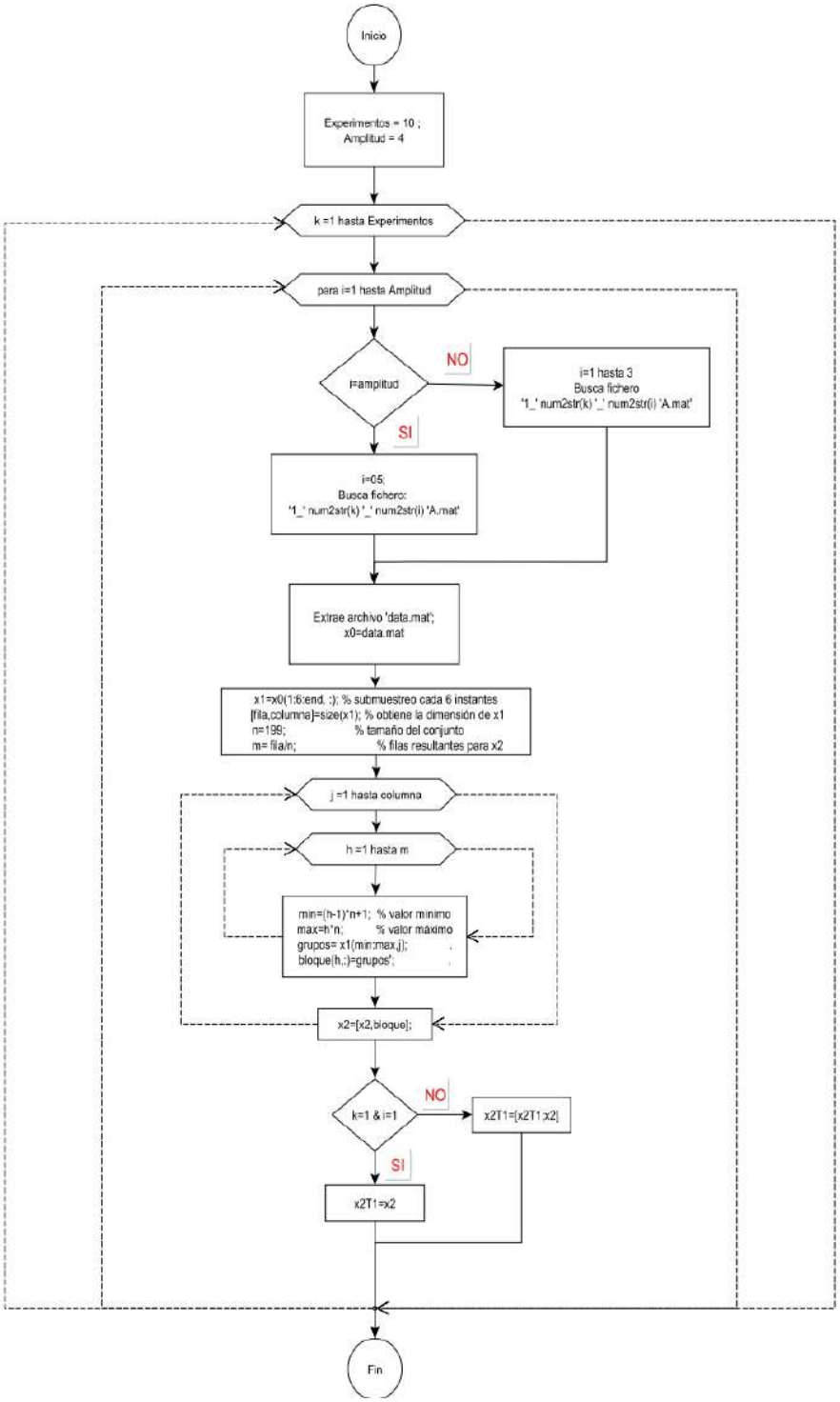


Figura 3.3: Diagrama de flujo para programa en MATLAB© estado estructural saludable. Para este caso en particular, el número de experimentos es 10 (para cada amplitud).

Otra forma de plantear la reestructuración de datos es la siguiente: dada la naturaleza tridimensional de la información recopilada en este documento (experimentos,tiempo, sensores), se puede remodelar de acuerdo al desarrollo propuesto por Westerhuis et al. [44]. Toda la información es posible representarla con una matriz bidimensional [45] $\mathbf{X} = \left(x_{i,j}^{k,l}\right) \in \mathcal{M}_{(n_1+\cdots+n_4)\times(24\cdot 199)}$ como se muestra en (3.7).

$$
\begin{aligned}
\mathbf{X} &= \left(x_{i,j}^{k,l}\right) \qquad\qquad \textit{units} \\
&= \left[\begin{array}{ccc|ccc|c|ccc}
x_{1,1}^{1,1} & \cdots & x_{1,1}^{1,199} & x_{1,1}^{2,1} & \cdots & x_{1,1}^{2,199} & \cdots & x_{1,1}^{24,1} & \cdots & x_{1,1}^{24,199} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
x_{3320,1}^{1,1} & \cdots & x_{3320,1}^{1,199} & x_{n_1,1}^{2,1} & \cdots & x_{n_1,1}^{2,199} & \cdots & x_{3320,1}^{K,1} & \cdots & x_{3320,1}^{K,L} \\
\hline
x_{1,2}^{1,1} & \cdots & x_{1,2}^{1,199} & x_{1,2}^{2,1} & \cdots & x_{1,2}^{2,199} & \cdots & x_{1,2}^{24,1} & \cdots & x_{1,2}^{24,199} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
x_{1660,2}^{1,1} & \cdots & x_{1660,2}^{1,199} & x_{1660,2}^{2,1} & \cdots & x_{1660,2}^{2,199} & \cdots & x_{1660,2}^{24,1} & \cdots & x_{1660,2}^{24,5} \\
\hline
x_{1,3}^{1,1} & \cdots & x_{1,3}^{1,199} & x_{1,3}^{2,1} & \cdots & x_{1,3}^{2,199} & \cdots & x_{1,3}^{24,1} & \cdots & x_{1,3}^{24,199} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
x_{1660,3}^{1,1} & \cdots & x_{1660,3}^{1,199} & x_{1660,3}^{2,1} & \cdots & x_{1660,3}^{2,199} & \cdots & x_{1660,3}^{24,1} & \cdots & x_{1660,3}^{24,5} \\
\hline
x_{1,4}^{1,1} & \cdots & x_{1,4}^{1,199} & x_{1,4}^{2,1} & \cdots & x_{1,4}^{2,199} & \cdots & x_{1,4}^{24,1} & \cdots & x_{1,4}^{24,199} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
x_{1660,4}^{1,1} & \cdots & x_{1660,4}^{1,199} & x_{1660,4}^{2,1} & \cdots & x_{1660,4}^{2,199} & \cdots & x_{1660,4}^{24,1} & \cdots & x_{1660,4}^{24,199}
\end{array}\right] \\
&= \left[\begin{array}{c}
\mathbf{X}_1 \\ \hline \mathbf{X}_2 \\ \hline \mathbf{X}_3 \\ \hline \mathbf{X}_4
\end{array}\right] = \left[\begin{array}{c|c|c|c} \mathbf{X}^1 & \mathbf{X}^2 & \cdots & \mathbf{X}^{24} \end{array}\right].
\end{aligned}
\tag{3.7}
$$

donde los dos subíndices $i = 1, \ldots, n_j$ y $j = 1, \ldots, J$ están relacionados con la prueba experimental y el estado estructural, mientras que los dos superíndices $k = 1, \ldots, K$ y $l = 1, \ldots, L$ representan el sensor y el instante de tiempo. J es la cantidad de diferentes estados estructurales; K es el número total de sensores y L es el número de marcas de tiempo por experimento.

Por lo tanto, de acuerdo a lo información anterior podemos considerar lo siguiente:

- $J =$ 4 estados estructurales.

- $K = 24$ sensores.
- $L = 199$ instantes de tiempo
- $n_1 + n_2 + n_3 + n_4 = 8,300$ muestras
  - $n_1 = 3,320$ (83 muestras $\times$ 40 experimentos)
  - $n_2 = 1,660$ (83 muestras $\times$ 20 experimentos)
  - $n_3 = 1,660$ (83 muestras $\times$ 20 experimentos)
  - $n_4 = 1,660$ (83 muestras $\times$ 20 experimentos)

De esta manera se puede recopilar la información relacionada con las mediciones del sensor y sus variaciones a lo largo del tiempo.

3.3. Normalización de los datos

Antes de usar un clasificador, los datos deben procesarse para obtener las características más adecuadas. En este trabajo se utilizó la normalización de los datos y el análisis de componentes principales (PCA), ya que el objetivo principal es mantener la mayor cantidad de información posible con la mínima cantidad de datos.

En general, los datos en crudo se deben normalizar principalmente por dos razones:

- Evitar trabajar con diferentes magnitudes.
- Simplificar los cálculos para la descomposición de PCA.

Por estas razones se requiere de la normalización de los datos, en este trabajo se usa el método de estandarización de los datos. Este se basa en el centrado medio en columnas seguido de la división de cada columna por la desviación estándar de esa columna de la matriz $\mathbf{X}$. El resultado de cada columna de la nueva matriz estandarizada $\tilde{\mathbf{X}}$, dada por (3.8), tiene una media de cero y una desviación estándar de uno [43]. A continuación, se detalla el cálculo de la nueva matriz:

$$\tilde{\mathbf{X}}_{ab} = \frac{\mathbf{X}_{ab} - \mu_b}{\sigma_b} \tag{3.8}$$

donde:
$a, 1, \cdots, 8300;$

b, $1, \cdots, 4776$;
$\tilde{\mathbf{X}}_{ab}$, matriz normalizada.
$\mathbf{X}_{ab}$, matriz inicial (ver ecuación (3.6)).
μ_b, media o promedio de la columna b de la matriz $\mathbf{X}$.
σ_b, desviación estándar de la columna b de la matriz $\mathbf{X}$.

Una vez obtenida la matriz con los datos normalizados $\tilde{\mathbf{X}}_{ab}$, se procede a aplicar PCA.

3.4. Análisis de Componentes Principales (PCA: *Principal Componentes Analysis*)

El análisis de componentes principales (PCA) es un método utilizado para lograr una reducción de variables al generar un nuevo conjunto conocido como componentes principales (CP) (Figura 3.4). Cada componente principal es una combinación lineal de las variables originales. Todas las componentes principales son ortogonales entre sí, por lo que no hay información redundante y forman una base para el espacio de los datos. El conjunto completo de componentes principales es tan grande como el conjunto original de variables [46].

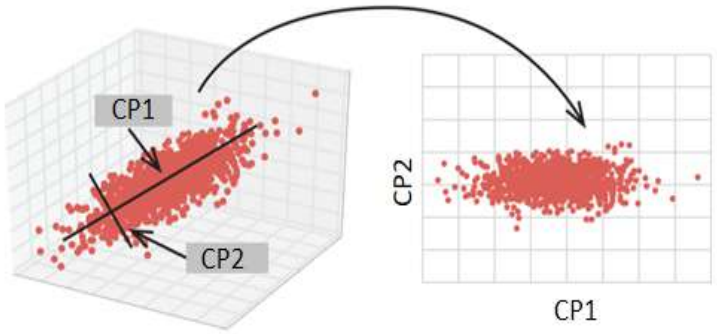


Figura 3.4: Componentes principales.

La transformación de las variables se calcula como una multiplicación de matriz a matriz [45].

$$\mathbf{T} = \tilde{\mathbf{X}}\mathbf{P}$$

donde $\tilde{\mathbf{X}}$ es la matriz normalizada, $\mathbf{P}$ es la matriz que contiene como columnas a las componentes principales de $\tilde{\mathbf{X}}$. Mientras que $\mathbf{T}$ es una matriz de transformación de dimensión

$8,300 \times 4,776$, en otras palabras, $\mathbf{T}$ es la proyección de la matriz escalada $\tilde{\mathbf{X}}$.

La reducción de dimensionalidad se realiza a través de un modelo PCA reducido mostrado en (3.9), donde ℓ representa a las componentes principales, $\mathbf{P}_\ell$ contiene las primeras ℓ componentes principales de $\tilde{\mathbf{X}}$, $\mathbf{T}_\ell$ es la matriz de transformación reducida.

$$\mathbf{T}_\ell = \tilde{\mathbf{X}}\mathbf{P}_\ell. \tag{3.9}$$

Para obtener el número de componentes principales idóneas para realizar la reducción de dimensionalidad se toma la siguiente proporción de varianza:

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_\ell}{\lambda_1 + \lambda_2 + \cdots + \lambda_n}, \tag{3.10}$$

donde $\lambda_1 + \lambda_2 + \cdots + \lambda_n$ son los valores propios asociados con las componentes principales de la matriz de varianza-covarianza, mientras que $\lambda_1 + \lambda_2 + \cdots + \lambda_\ell$ son los valores propios de las primeras ℓ componentes principales.

Para la realización de este trabajo se hizo uso de la herramienta MATLAB© para obtener el análisis de componentes principales de los datos ya preprocesados, para ello se usa el siguiente comando:

[coeff,score,latent]=pca (X);

Con este comando se aplica PCA a los datos pre-procesados, esta instrucción devuelve:

- **coeff**, componentes principales o vectores propios de la matriz de varianza, covarianza (ordenados de mayor a menor).
- **score**, proyecciones de los datos de la matriz $\tilde{\mathbf{X}}$ (datos pre procesados) en la nueva base (componentes principales).
- **latent**, valores propios de la matriz de varianza.

Al aplicarlo a los datos, se obtuvieron las dimensiones mostradas en la Figura 3.5. Las componentes principales son $4,776$, ya que el objetivo principal es reducir la dimensión total del conjunto de datos, se hace uso de (3.10) para determinar las componentes principales con las cuales se va a trabajar.

Name	Value
coeff	4776x4776 double
latent	4776x1 double
score	8300x4776 double
X	8300x4776 double
Y	8300x4776 double

Figura 3.5: Dimensiones de la variables obtenidas en MATLAB© .

En este documento, se seleccionó el número de componentes principales en función de mantener el 80 %, 85 % y 95 % de la varianza. Tal como se observa en la Figura 3.6, con 282 componentes principales se tiene el 85 % de la varianza, con 665 componentes principales se tiene el 90 % de la varianza retenida y por último con 1, 500 componentes principales se tiene el 95 % de la varianza retenida.

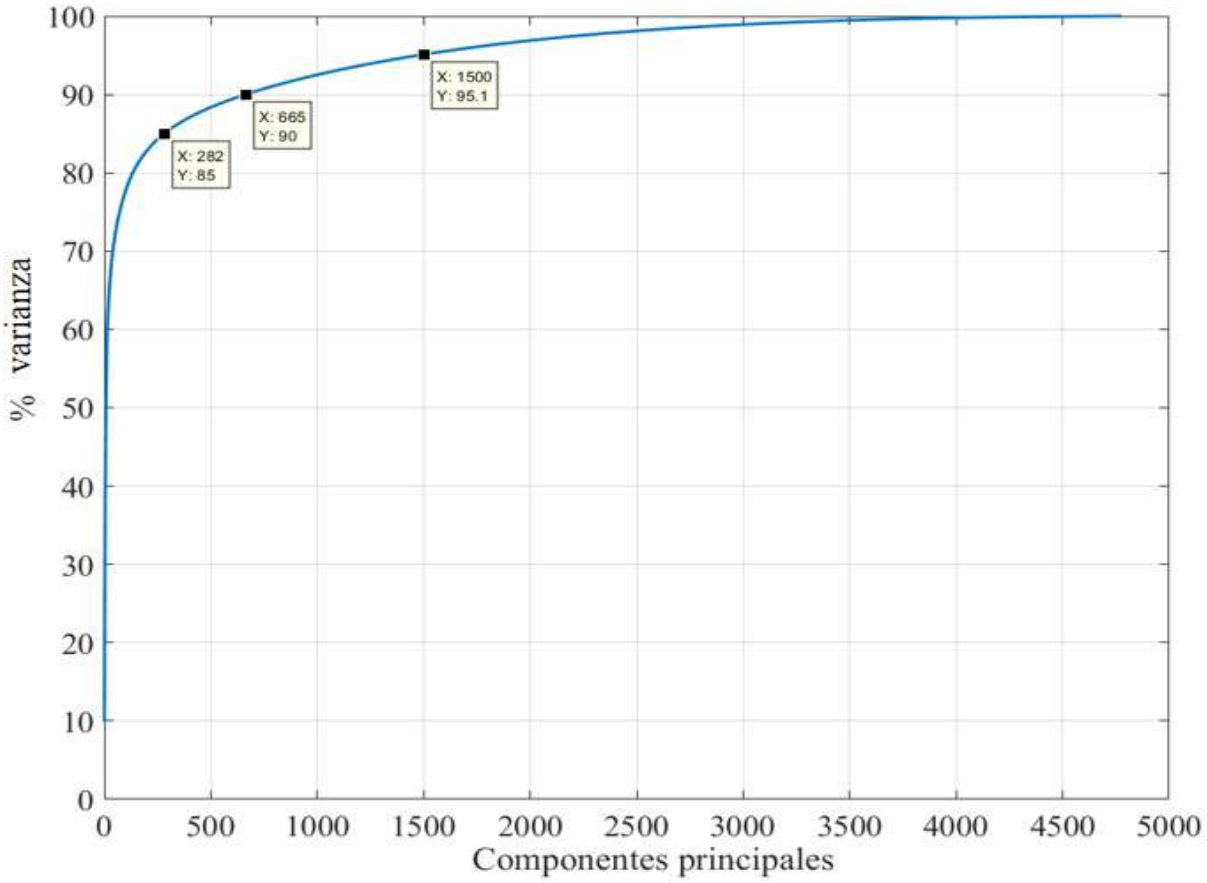


Figura 3.6: Gráfica de relación de las componentes principales con respecto al porcentaje de varianza retenida.

Una vez elegido el número de componentes con las que se va a trabajar, se usa el si-

guiente comando de MATLAB© para obtener solamente la información contenida en dichas componentes principales:

[coeff,score,latent]=pca (X,'NumComponents',CP);

donde:

- $\mathbf{X} = \tilde{\mathbf{X}}$, matriz normalizada de donde se toman las componentes principales.
- **NumComponents**, nombre del argumento.
- **CP**, número de componentes principales con las que se desea trabajar (282, 665 o $1,500$).

Al aplicarlo a nuestros datos, se obtuvo la información mostrada en la Figura 3.7:

Name	Value
coeff	4776x282 double
latent	4776x1 double
score	8300x282 double

(a) 282 componentes principales.

Name	Value
coeff	4776x665 double
latent	4776x1 double
score	8300x665 double

(b) 665 componentes principales.

Name	Value
coeff	4776x1500 double
latent	4776x1 double
score	8300x1500 double

(c) 1500 componentes principales.

Figura 3.7: Resultados obtenidos de PCA con MATLAB©.

Posteriormente se categorizan los datos de acuerdo a la Tabla 3.3:

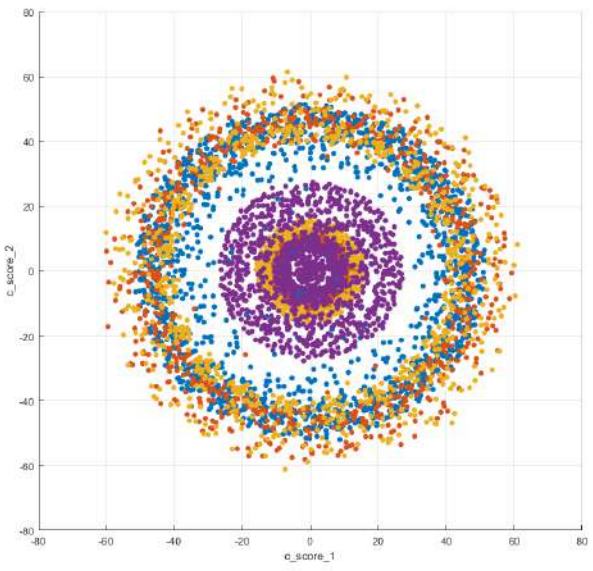
Caso	Categoría asignada	Cantidad de muestras
Barra original	0	3,320
Barra replica	1	1,660
Barra con crack\fisura	2	1,660
Barra con perno flojo	3	1,660

Tabla 3.3: Categorización de las muestras.

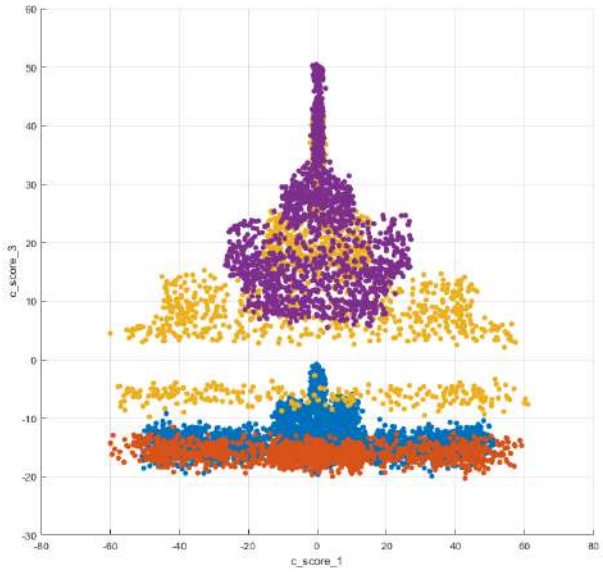
3.5. Clasificación

Habiéndoles asignado una categoría a los datos, se prueban los algoritmos de clasificación de aprendizaje automático supervisado en la herramienta *Classification Learner*.

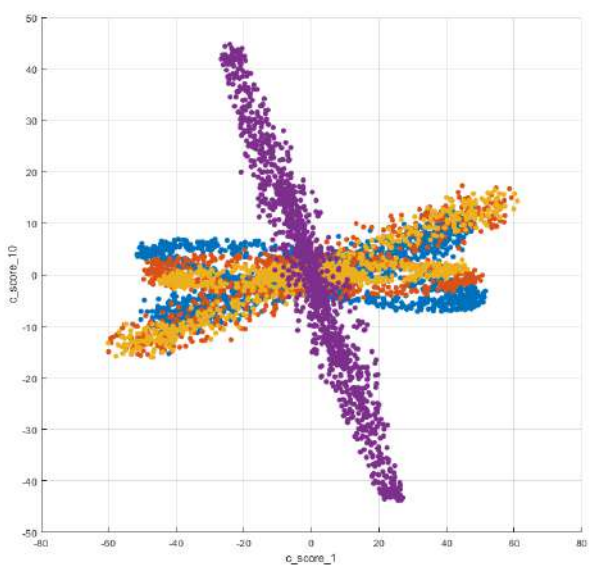
De acuerdo a la información a partir de los datos procesados se obtienen los gráficos de dispersión mostrados en la Figura 3.8.



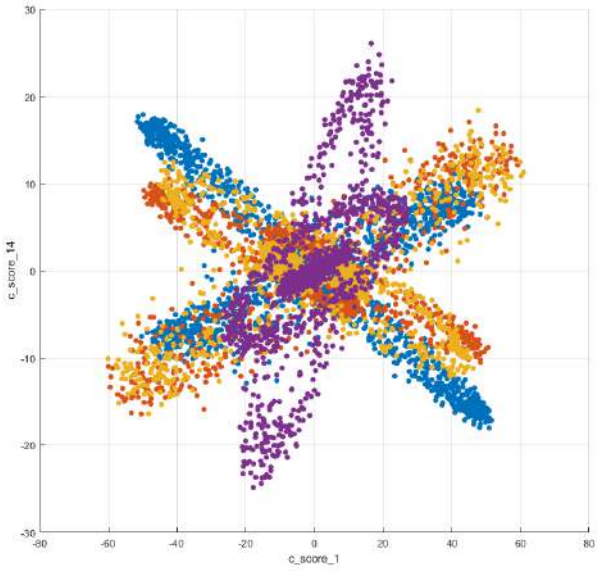
(a) Gráfica de la 1° componente principal Vs la 2° componente principal.



(b) Gráfica de la 1° componente principal Vs la 3° componente principal.



(c) Gráfica de la 1° componente principal Vs la 10° componente principal.



(d) Gráfica de la 1° componente principal Vs la 14° componente principal.

Figura 3.8: Los puntos azules representan las muestras saludables, los puntos naranjas representan las muestras de la barra con réplica, los puntos amarillos representan las muestras de la barra con *crack*/fisura y los puntos morados representan las muestras con perno flojo.

Como se mencionó anteriormente, en la herramienta *Classification Learner* es necesario elegir un método de validación, por tal motivo en este documento se utiliza una validación cruzada con $\kappa = 5$ para estimar el desempeño de la estrategia propuesta. Este método de validación se explica con más detalle a continuación.

3.5.1. Validación cruzada de κ iteraciones (κ-Fold Cross Validation)

La validación cruzada de κ iteraciones conocida en inglés como κ-*fold cross validation* es una técnica utilizada para evaluar los resultados y garantizar que sean independientes de la partición entre los datos de entrenamiento y prueba [45].

Normalmente, un clasificador basado en datos se infiere en base a datos de entrenamiento y considerando un algoritmo de aprendizaje clasificador. Un error de predicción, también conocido como error verdadero, está asociado a cada clasificador. Sin embargo, este error de predicción generalmente es desconocido, no se puede calcular y debe estimarse en función de los datos.

Este enfoque implica dividir aleatoriamente el conjunto de datos en subconjuntos κ aproximadamente el mismo tamaño. El primer grupo se trata como un conjunto de validación

[47] y el método se ajusta a los $(\kappa - 1)$ restantes (ver Figura 3.9).

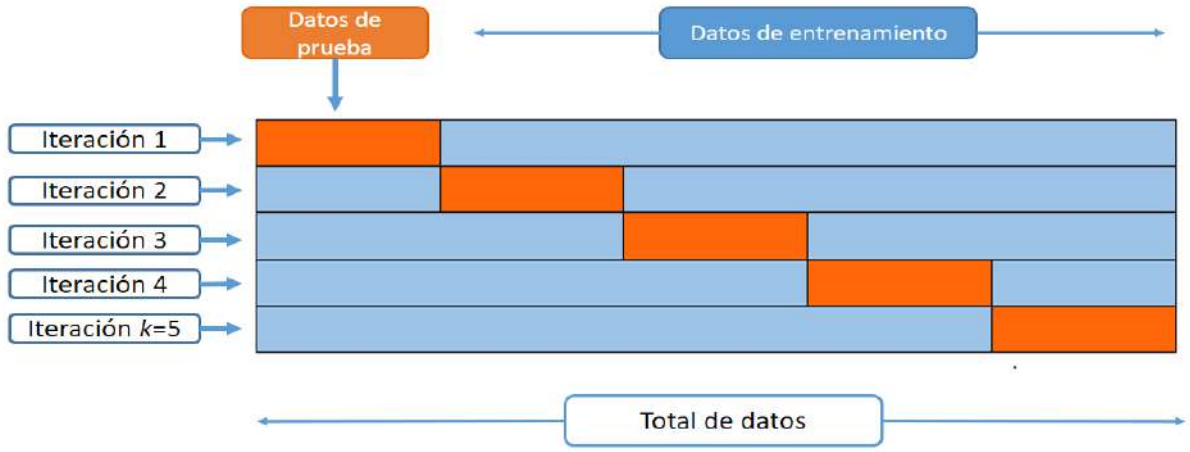


Figura 3.9: Representación esquemática de la validación cruzada con $\kappa = 5$ iteraciones.

Para fines prácticos, normalmente se eligen $\kappa = 5$ ó $\kappa = 10$. La principal ventaja de usar estos valores es computacional, ya que se ahorra recursos computacionales al ajustar el procedimiento de aprendizaje a solo 5 ó 10 veces. Empíricamente se ha demostrado que estos valores producen estimaciones más precisas [47].

Al final, la estimación del error es la media numérica de los errores cometidos en cada grupo de datos.

La problemática a resolver en este trabajo tiene que ver con el aprendizaje supervisado y específicamente con la clasificación, por eso se han elegido para trabajar los algoritmos k-vecinos más cercanos conocido como k-NN (*k-Nearest Neighborts*) y máquinas de soporte vectorial conocidas como SVM (*Support Vector Machine*).

Una vez establecido lo anterior, lo siguiente es elegir los algoritmos con los cuales serán entrenados los modelos de clasificación, por eso se han elegido para trabajar los algoritmos k-vecinos más cercanos conocido como k-NN y máquinas de soporte vectorial conocidas como SVM.

3.5.2. k-Vecinos Más Cercanos (k-NN: *k-Nearest Neighborts)*

Es uno de los algoritmos no paramétricos más simples y más conocidos en cuanto a clasificación y reconocimiento de patrones se trata. Está basado en muestras y tiene como propósito clasificar los objetos haciendo uso de las muestras más cercanas las cuales son conocidas como vecinos [48].

Este algoritmo almacena el conjunto de datos de entrenamiento y para hacer una predicción calcula los k vecinos más cercanos a la muestra a clasificar y asigna la categoría de la mayoría [45]. La variable k será la encargada de indicar cuántos vecinos se utilizarán para la clasificación (ver Figura 3.10).

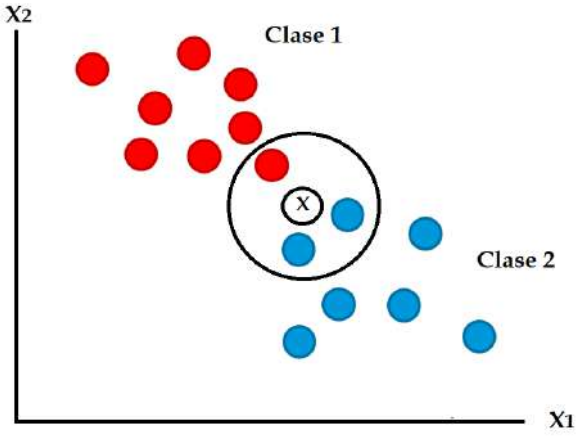


Figura 3.10: Clasificación por método k-NN.

Un valor pequeño para k proporciona el ajuste más flexible, mientras que los valores más grandes de k proporcionan un ajuste más suave y menos variable [47].

No se debe confundir la k de este algoritmo con la k usada en uno de los superíndices en el elemento genérico de la matriz $\mathbf{X}$ en la ecuación, ya que tienen un significado diferente.

3.5.3. Máquina de soporte vectorial (*SVM: Support Vector Machine)*

Las máquinas de soporte conocidas como *SVM* por sus siglas en inglés, se han aplicado con éxito a una serie de problemas del mundo real. Es un método que clasifica los datos encontrando el límite de decisión lineal (hiperplano) que separa todos los puntos de datos de una clase de los de la otra clase. El mejor hiperplano para un *SVM* es el que tiene el mayor margen entre las dos clases, cuando los datos son linealmente separables. Si los datos no son linealmente separables, se utiliza una función de pérdida para penalizar los puntos en el lado equivocado del hiperplano. *SVM* a veces usa una transformación de núcleo y así convertir los datos no linealmente separables en dimensiones más altas donde se puede encontrar un límite de decisión.

Es conveniente usar este algoritmo cuando:

- Los datos tienen exactamente dos clases.

- Los datos son de alta dimensión, no linealmente separables.
- Se necesita un clasificador simple, fácil de interpretar y preciso.

Ahora bien, la clasificación *SVM* [47] es fundamentalmente una técnica de clasificación binaria. Este algoritmo busca un hiperplano óptimo que separe los datos en dos clases (ver Figura 3.11). Para las clases separables, el hiperplano óptimo maximiza un espacio (que no contiene ninguna observación) alrededor de sí mismo, lo que crea el límite para las clases. Para las clases inseparables, el objetivo es el mismo, pero el algoritmo impone una penalización (se denota por la letra C) en la longitud del margen para cada observación que se encuentra en el lado equivocado de su límite de clase. Un hiperplano se define como un subespacio plano afín de la dimensión $p-1$ en un espacio p-dimensional.

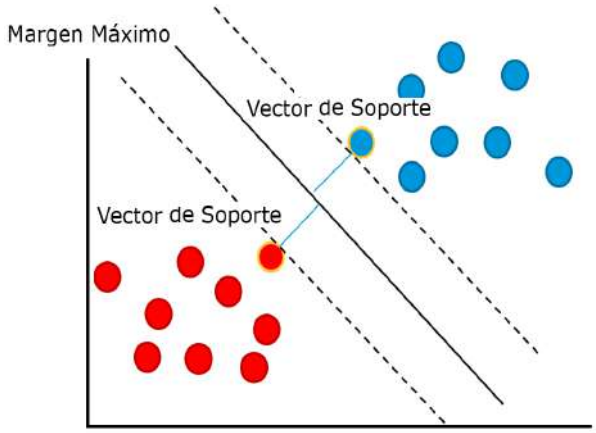


Figura 3.11: Máquina de vectores de soporte (SVM). La figura muestra datos donde una clase está representada por puntos azules y la otra por puntos rojos. El objetivo principal es encontrar el hiperplano óptimo que defina el margen más amplio para separar ambas clases.

Para entender de una mejor manera, se muestra el siguiente desarrollo matemático tomado de [43]. Consideremos un conjunto de entrenamiento $\{(x_i, y_i)\}_{i=1}^{N}$ con datos d-dimensionales $x_i \in \mathbb{R}^d$ y sus etiquetas asociadas $y_i \in \{-1, +1\}$

El hiperplano está dado por:

$$h(x) = \omega^T x + b, \tag{3.11}$$

donde x es una muestra cualquiera, ω es un vector conocido en inglés como *weight vector* y b es una constante conocida en inglés como *bias term*.

Entre todas las descripciones posibles del hiperplano, se ha elegido el llamado hiperplano canónico que satisface:

$$\omega^T x_+^{vs} + b = 1 \tag{3.12}$$

$$\omega^T x_-^{vs} + b = -1 \tag{3.13}$$

donde x_+^{vs} y x_-^{vs} simbolizan las muestras de entrenamiento representadas por los puntos azules y los puntos rojos respectivamente, es decir, los llamados vectores de soporte (vs) mostrados en la Figura 3.11.

La distancia entre un punto x y el hiperplano h viene dada por:

$$d(x, h) = \frac{\left|\omega^T x + b\right|}{\|\omega\|} \tag{3.14}$$

Pero para el hiperplano canónico, cuando x es un vector de soporte, el numerador $\left|\omega^T x + b\right|$ es igual a uno y la distancia al vector de soporte es

$$d(x_\pm^{sv}, h) = \frac{1}{\|\omega\|} \tag{3.15}$$

Es conocido que el ancho del margen es el doble de esta distancia: $\frac{2}{\|\omega\|}$, por lo tanto maximizar el margen es equivalente a minimizar la ecuación (3.15), que es equivalente al siguiente problema:

$$\max \frac{1}{\|\omega\|} = \min \|\omega\| = \min_{\omega,b} \frac{1}{2} \|\omega\|^2, \quad \text{sujeto a} \quad \begin{cases} h(x_i) \geq 1, \forall y_i = 1 \\ h(x_i) \leq 1, \forall y_i = -1 \end{cases} \tag{3.16}$$

Si se desea encontrar el extremo de una función con restricciones se puede resolver utilizando multiplicadores de Lagrange, eso nos dará una nueva expresión que podemos minimizar o maximizar sin preocuparnos por las restricciones. Entonces, esto nos lleva a:

$$\min_{\omega,b} L(\omega, b) = \frac{1}{2} \|\omega\|^2 - \sum_{i=1}^{N} \alpha_i \left[y_i \left(\omega^T x_i + b \right) - 1 \right] \tag{3.17}$$

donde α_i representa a los multiplicadores de Lagrange. Tomando derivada parcial con respecto a ω igual a cero,

$$\frac{\partial L(\omega, b)}{\partial \omega} = \omega - \sum_{i=1}^{N} \alpha_i y_i x_i = 0 \Leftrightarrow \omega = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{3.18}$$

Esta ecuación establece que ω, es una combinación lineal de las muestras de los datos.

Tomando derivada parcial con respecto a b igual a cero,

$$\frac{\partial L(\omega, b)}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3.19}$$

para finalizar, se sustituyen las ecuaciones (3.18) y (3.19) en la ecuación (3.17)

$$\min_{\alpha_i} \left[\frac{1}{2} \left(\sum_{i=1}^{N} \alpha_i y_i x_i \right)^T \left(\sum_{j=1}^{N} \alpha_j y_j x_j \right) - \sum_{i=1}^{N} \alpha_i y_i \left(\sum_{j=1}^{N} \alpha_j y_j x_j \right)^T x_i - b \underbrace{\sum_{i=1}^{N} \alpha_i y_i}_{0} + \sum_{i=1}^{N} \alpha_i \right] \tag{3.20}$$

que puede reescribir como

$$\min_{\alpha_i} \left[\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \right] \tag{3.21}$$

En caso de que los datos no admitan un hiperplano de separación, el método *SVM* puede usar un margen suave, esto significa que en lugar de buscar el mayor margen posible para que cada observación se encuentre en el lado correcto del hiperplano y del margen, vamos a permitir que algunas observaciones estén en el lado incorrecto del margen como se muestra en la Figura 3.12.

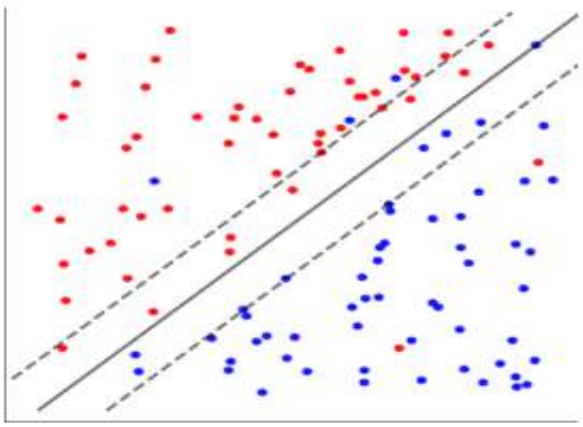


Figura 3.12: Método *SVM* con un margen suave.

En consecuencia, el problema anterior se generaliza por medio de variables de holgura ϵ_i y un parámetro de penalización C. La formulación general para el núcleo lineal es en este

caso:

$$\min_{\omega,b,\epsilon_i} \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{n}\epsilon_i \quad \text{sujeto a} \quad \begin{cases} h(x_i)y_i \geq 1-\epsilon_i, i=1,\ldots,N; \\ \epsilon_i \geq 0, i=1,\ldots,N \end{cases} \tag{3.22}$$

Usando los multiplicadores de Lagrange

$$\min_{\alpha_i}\left[\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^T x_j\right] \text{sujeto a} \quad \begin{cases} \sum_{i=1}^{N}\alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C, i=1,\ldots,N \end{cases} \tag{3.23}$$

Con frecuencia, el parámetro de penalización C se denomina en inglés *box constrain* ya que mantiene los valores admisibles de los multiplicadores de Lagrange en una región acotada.

Se observa a partir de las ecuaciones (3.21) y (3.23), que la optimización depende solo del producto interno de los pares de observaciones o muestras.

Cuando el problema de clasificación no tiene un hiperplano simple como un criterio de separación, normalmente se recurre a usar una transformación a otro espacio, $\phi(\cdot)$. De hecho, la transformación en sí no es necesaria, lo verdaderamente importante es el producto de punto, que da pie a tener la llamada función *kernel* (3.24):

$$K(x_i, x_j) = \phi(x_i)\phi(x_j) \tag{3.24}$$

Una función *kernel*, permite el cálculo del producto interno entre los vectores de diferentes espacios sin calcular expresamente la transformación de esos espacios. Se pueden usar diferentes funciones *kernel*, por ejemplo, polinomio, tangente hiperbólica o función de base radial gaussiana.

La ecuación (3.25) se conoce como función *kernel* polinomial de grado d, donde d es un entero positivo y se usa para $d > 1$. Mientras que ρ representa el *kernel scale*.

$$K(x_i, x_j) = \left(1 + \frac{1}{\rho}(x_i^T x_j)\right)^d \tag{3.25}$$

En este documento se ha elegido trabajar con una función *kernel* polinomial de segundo grado ($d = 2$) de acuerdo al comportamiento de los datos que observamos en la Figura 3.8. Estos gráficos revelan una relación cuadrática, en particular el gráfico de la Figura 3.8a ya que expone una forma de círculos concéntricos del conjunto de datos. Por esta razón optó por adoptar el clasificador cuadrático *SVM*.

3.6. Elección del modelo de clasificación adecuado

Para seleccionar cuál es el mejor algoritmo de aprendizaje automático se realiza un proceso de prueba y error, que depende de algunas características específicas de los algoritmos, tales como: velocidad de entrenamiento, uso de memoria, precisión predictiva sobre nuevos datos y transparencia o interpretabilidad (con qué facilidad puede comprender las razones por las que un algoritmo realiza sus predicciones).

Sin embargo, para elegir el modelo de clasificación adecuado para cada problema es necesario realizar la evaluación de los modelos de clasificación mediante las métricas de evaluación y sintonizar los parámetros requeridos para cada uno de los algoritmos de clasificación.

3.6.1. Métricas de evaluación

Para saber si el modelo que hemos entrenado es el ideal para la aplicación que deseamos, es importante evaluar y analizar cuidadosamente los resultados de algoritmos de aprendizaje automático elegidos.

El rendimiento de los modelos de clasificación está representado por diferentes parámetros, como exactitud, precisión, sensibilidad, F1−*score* y especificidad. Generalmente, estas métricas evalúan los problemas de clasificación binaria [49], representados en una matriz de confusión como la presentada en la Tabla 3.4:

		Predicción	
		Positivo	**Negativo**
Observación	**Positivo**	Verdadero Positivo (TP)	Falso Negativo (FN)
	Negativo	Falso Positivo (FP)	Verdadero Negativo (TN)

Tabla 3.4: Matriz de confusión binaria

- **Verdadero positivo (TP):** Es la proporción de casos positivos que se identificaron correctamente.

- **Falso positivo (FP):** Es la proporción de casos negativos que se clasificaron incorrectamente como positivos.
- **Verdadero negativo (TN):** Se define como la proporción de casos negativos que se clasificaron correctamente.
- **Falso negativo (FN):** Es la proporción de casos positivos que se clasificaron incorrectamente como negativos.

La interpretación que se le atribuya a los términos *positivo* y *negativo* puede variar de una aplicación a otra. Por ejemplo, en [50] y [4], donde se considera un modelo de aerogenerador para la detección de fallas. Si una muestra se clasifica como positiva, entonces la estructura actual es saludable, por el contrario, si una muestra se clasifica como negativa indica la existencia de algún tipo de falla. Esta interpretación se toma como referencia para el presente trabajo.

De igual manera, es necesario realizar la evaluación de los modelos de clasificación mediante las métricas de evaluación de modelos y sintonizar los parámetros requeridos por cada uno.

Los parámetros empleados para evaluar los modelos de clasificación de este trabajo se describen a continuación:

- **Exactitud (*Accuracy* - acc):** Es el número de predicciones correctas realizadas por el modelo de acuerdo a el número total de registros. La mejor exactitud es el 100 %, lo que indica que todas las predicciones son correctas.

$$\text{acc} = \frac{\text{TP} + \text{TN}}{\text{TP+FP+FN+TN}}$$

  La exactitud no funciona cuando se trabaja con un conjunto de datos desbalanceados. Se refiere a una situación en la que el número de muestras no es el mismo para todas las clases en del conjunto de datos usado para clasificación.

- **Precisión (*Precision* - ppv):** La precisión, se encuentra definida por la siguiente expresión matemática:

$$\text{ppv} = \frac{\text{TP}}{\text{TP+FP}}$$

  Este parámetro evalúa los datos por su desempeño de predicciones *positivas*, en otras palabras, es la proporción de los casos positivos que han sido predichos correctamente.

- **Sensibilidad (*Sensibility/ Recall* - tpr):** También llamada tasa de verdaderos positivos (en inglés *true positive rate*).

$$\text{tpr} = \frac{\text{TP}}{\text{TP+FN}}$$

  Cuando la clase es positiva, este parámetro responde a la pregunta ¿Qué porcentaje logró clasificar correctamente?. Dicho de otra manera, este parámetro calcula cuántos de los positivos reales ha sido capaz nuestro modelo de identificarlos como positivo (verdadero positivo).

- **F1 Score:** Es definida como la media armónica de la precisión y la sensibilidad.

$$\text{F1} = 2\frac{\text{ppv} \times \text{tpr}}{\text{ppv} + \text{tpr}}$$

  Una puntuación $F1$ alcanza su mejor valor en 1 (precisión y sensibilidad perfecta) y peor en 0.

- **Especificidad (*Specificity* - tnr):** La especificidad o tasa de verdaderos negativos, se calcula como el número de predicciones negativas correctas dividido por el número total de negativos.

$$\text{tnr} = \frac{\text{TN}}{\text{TN+FP}}$$

La especificidad es exactamente lo contrario a la sensibilidad, nos dice la proporción de los casos negativos predichos correctamente.

Estas métricas también son aplicables a problemas de clasificación múltiple como se muestra en [51]. el caso presentado en este documento es un problema de clasificación de 4 clases, por lo que de acuerdo con [52], [53] y [54] el resultado de las métricas de evaluación será el promedio obtenido al sumar el resultado de cada clase y dividir sobre el número total de clases. Las fórmulas para calcular estas métricas en un modelo de clasificación de varias clases se muestran en la Tabla 3.5 de acuerdo a [45]. En la tabla mostrada j representa a la clase individual y 4 al número total de clases.

Para identificar TP, TN, FP y FN en una matriz de confusión de clases múltiples, se debe realizar un análisis por cada clase como en [45] y [52]. Dicho lo anterior, en la Tabla 3.6 se pueden identificar estos elementos poniendo como ejemplo la clase C:

Métrica	**Formula**
Exactitud promedio ($\overline{\text{acc}}$)	$\frac{1}{4}\sum_{j=1}^{4}\frac{\text{TP}_j+\text{TN}_j}{\text{TP}_j+\text{FP}_j+\text{FN}_j+\text{TN}_j}$
Precisión promedio ($\overline{\text{ppv}}$)	$\frac{1}{4}\sum_{j=1}^{4}\frac{\text{TP}_j}{\text{TP}_j+\text{FP}_j}$
Sensibilidad promedio ($\overline{\text{tpr}}$)	$\frac{1}{4}\sum_{j=1}^{4}\frac{\text{TP}_j}{\text{TP}_j+\text{FN}_j}$
F1-Score promedio ($\overline{\text{F1}}$)	$2\frac{\overline{\text{ppv}}\times\overline{\text{tpr}}}{\overline{\text{ppv}}+\overline{\text{tpr}}}$
Especificidad promedio ($\overline{\text{tnr}}$)	$\frac{1}{4}\sum_{j=1}^{4}\frac{\text{TN}_j}{\text{TN}_j+\text{FP}_j}$

Tabla 3.5: Métricas de evaluación para modelos de clasificación múltiple.

		Clase predicha			
		Clase A	**Clase B**	**Clase C**	**Clase D**
Clase actual	**Clase A**	AA	AB	AC	AD
	Clase B	BA	BB	BC	BD
	Clase C	CA	CB	CC	CD
	Clase D	DA	DB	DC	DD

Tabla 3.6: Matriz de confusión con 4 clases.

De esta matriz de confusión podemos identificar cuatro regiones:

- **Región naranja**: relacionada con el verdadero negativo de la clase C (TN_C). Es la suma TN_C =AA+AB+BA+BB+AD+BD+DA+DB+DD.
- **Región verde**: relacionada con el verdadero positivo (TP_C). Es TP_C =CC
- **Región roja**: relacionada con el falso negativo (FN_C). Es la suma FN_C = CA+CB+CD.
- **Región café**: relacionada con el falso positivo (FP_C). Es la suma de FP_C = AC+BC+DC.

De manera general, para cada clase los elementos TN_j, TP_j, FN_j y FP_j donde $j = 1, \ldots, 4$, se identifican de la siguiente manera:

- TN_j: será la suma de los valores correspondientes a todas las columnas y filas exceptuando la columna y fila de la clase analizada.
- TP_j: es la celda donde se intersectan la clase predicha y la clase actual de la clase analizada.

- FN_j: es la suma de los valores correspondientes a la fila de la clase analizada exceptuando TP_j.
- FP_j: es la suma de los valores correspondientes a la columna de la clase analizada exceptuando TP_j.

3.6.2. Sintonización de parámetros para k-NN

El algoritmo k-NN, fue el primer clasificador con el cual se trabajó para realizar las pruebas de sintonización. Se realizó una tabla de comparación de las métricas para evaluar los modelos con los resultados obtenidos en base al 80 %, 85 % y 90 % de la varianza.

Varianza	Número de componentes principales	k	Exactitud promedio $\overline{\text{acc}}$	Precisión promedio $\overline{\text{ppv}}$	Sensibilidad promedio $\overline{\text{tpr}}$	F1 Score promedio $\overline{\text{F1}}$	Especificidad promedio $\overline{\text{tnr}}$	Promedio
85 %	282	1	0.939	0.928	0.925	0.927	0.980	0.940
		2	0.925	0.921	0.906	0.914	0.974	0.928
		3	0.943	0.939	0.929	0.934	0.98	0.945
		4	0.933	0.931	0.916	0.923	0.976	0.936
		5	0.942	0.940	0.928	0.934	0.980	0.945
		10	0.929	0.931	0.911	0.921	0.974	0.933
		15	0.909	0.915	0.886	0.900	0.966	0.915
90 %	665	1	0.938	0.929	0.925	0.927	0.980	0.940
		2	0.922	0.920	0.903	0.911	0.972	0.926
		3	0.941	0.938	0.928	0.933	0.980	0.944
		4	0.932	0.930	0.916	0.923	0.976	0.935
		5	0.939	0.938	0.925	0.931	0.978	0.942
		10	0.927	0.929	0.909	0.919	0.974	0.932
		15	0.908	0.915	0.886	0.900	0.966	0.915
95 %	1500	1	0.937	0.928	0.924	0.926	0.979	0.939
		2	0.920	0.918	0.900	0.909	0.971	0.924
		3	0.937	0.934	0.922	0.928	0.978	0.940
		4	0.926	0.925	0.908	0.916	0.974	0.930
		5	0.935	0.933	0.919	0.926	0.977	0.938
		10	0.930	0.932	0.913	0.922	0.975	0.934
		15	0.911	0.917	0.889	0.903	0.967	0.917

Tabla 3.7: Indicadores de evaluación al modelo k-NN utilizando diferentes porcentajes de varianza y número de vecinos más cercanos (k).

En la Tabla 3.7 se pueden observar los resultados de los parámetros de los métodos de clasificación. Promediando todos los parámetros obtenemos el mejor desempeño cuando tenemos el 85 % de la varianza y $k = 3$ ó $k = 5$, sin embargo, cuando se tiene $k = 3$ existe una mejor sensibilidad. Esto quiere decir que el modelo ha sido capaz de clasificar correctamente los datos en un 92.9 %. En la tabla también se observa que la exactitud no sobrepasa el 95 % para el método k-NN.

Se procede a graficar el caso con el 85 % de la varianza de $k = 1$ hasta $k = 15$ para corroborar el comportamiento de cada uno de los modelos de clasificación de acuerdo a los parámetros que los evalúan (ver Figura 3.13).

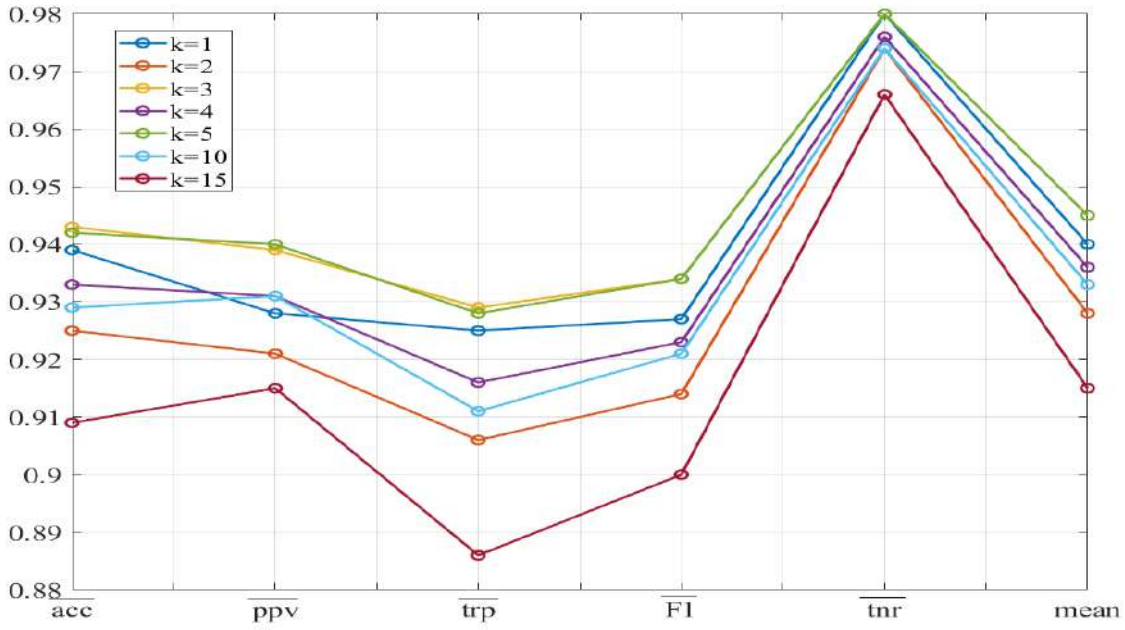


Figura 3.13: Métricas para evaluar el modelo de clasificación k-NN usando 85 % de varianza. Las métricas son: $\overline{\text{acc}}$ (exactitud promedio), $\overline{\text{ppv}}$ (precisión promedio), $\overline{\text{trp}}$ (sensibilidad promedio), $\overline{\text{F1}}$ (F1- Score promedio), $\overline{\text{tnr}}$ (especificidad promedio) y media total.

3.6.3. Sintonización de parámetros para *SVM*

De la misma manera que se obtuvieron las métricas de evaluación para el método k-NN, se obtuvieron para el método *SVM*. Por tanto, se realizaron las Tablas 3.8 y 3.9 donde se muestran los resultados obtenidos. Los parámetros requeridos específicamente por este método son *kernel Scale* y *C*.

En este trabajo se observa que en las varianzas 85 % y 95 %, a partir del *kernel scale* igual a 17 y 26 respectivamente, cuentan con una sensibilidad y una exactitud del 0.999,

visto en porcentaje equivale a un 99.9 %. Esto quiere decir que nuestro modelo ha clasificado correctamente el 99.9 % de los datos.

De acuerdo a los resultados obtenidos se considera óptimo trabajar con el 85 % de varianza, un *kernel scale* = 17 y una C= 1. Elegir trabajar con cualquier otro modelo, exceptuando los casos con 85 % y 95 % de varianza, y un *kernel scale* igual a 4 y 7 respectivamente, no afecta el desempeño y solo significa un mayor gasto de recursos computacionales.

Varianza	Número de componentes principales (p)	Kernel scale (ρ)	C	Exactitud $\overline{\text{acc}}$	Precisión $\overline{\text{ppv}}$	Sensibilidad $\overline{\text{tpr}}$	F1 Score $\overline{\text{F1}}$	Especificidad $\overline{\text{tnr}}$	Promedio
85 %	282	$\frac{\sqrt{p}}{4}$ = 4	1	0.996	0.996	0.995	0.996	0.999	0.996
			2	0.996	0.996	0.995	0.996	0.999	0.996
			3	0.996	0.996	0.995	0.996	0.999	0.996
			4	0.996	0.996	0.995	0.996	0.999	0.996
			5	0.996	0.996	0.995	0.996	0.999	0.996
			10	0.996	0.996	0.995	0.996	0.999	0.996
			15	0.996	0.996	0.995	0.996	0.999	0.996
			20	0.996	0.996	0.995	0.996	0.999	0.996
			100	0.996	0.996	0.995	0.996	0.999	0.996
		$\sqrt{p}$ = 17	1	0.999	0.999	0.999	0.999	1	0.999
			2	0.999	0.999	0.999	0.999	1	0.999
			3	0.999	0.999	0.999	0.999	1	0.999
			4	0.999	0.999	0.999	0.999	1	0.999
			5	0.999	0.999	0.999	0.999	1	0.999
			10	0.999	0.999	0.999	0.999	1	0.999
			15	0.999	0.999	0.999	0.999	1	0.999
			20	0.999	0.999	0.999	0.999	1	0.999
			100	0.999	0.999	0.999	0.999	1	0.999
		$4\sqrt{p}$ = 67	1	0.999	0.999	0.999	0.999	1	0.999
			2	0.999	0.999	0.999	0.999	1	0.999
			3	0.999	0.999	0.999	0.999	1	0.999
			4	0.999	0.999	0.999	0.999	1	0.999
			5	0.999	0.999	0.999	0.999	1	0.999
			10	0.998	0.998	0.998	0.998	0.999	0.998
			15	0.998	0.998	0.998	0.998	0.999	0.998
			20	0.997	0.997	0.997	0.997	0.999	0.997
			100	0.997	0.997	0.997	0.997	0.999	0.997

Tabla 3.8: Indicadores de evaluación al modelo *SVM* utilizando el 85 % de varianza. Se variaron los valores del parámetro ρ (Kernel scale) y C (número de restricciones).

Varianza	Número de componentes principales (p)	Kernel scale (p)	C	Exactitud $\overline{\text{acc}}$	Precisión $\overline{\text{ppv}}$	Sensibilidad $\overline{\text{tpr}}$	F1 Score $\overline{\text{F1}}$	Especificidad $\overline{\text{tnr}}$	Promedio
90 %	665	$\frac{\sqrt{p}}{4} = 7$	1	0.998	0.998	0.998	0.998	0.999	0.998
			2	0.998	0.998	0.998	0.998	0.999	0.998
			3	0.998	0.998	0.998	0.998	0.999	0.998
			4	0.998	0.998	0.998	0.998	0.999	0.998
			5	0.998	0.998	0.998	0.998	0.999	0.998
			10	0.998	0.998	0.998	0.998	0.999	0.998
			15	0.998	0.998	0.998	0.998	0.999	0.998
			20	0.998	0.998	0.998	0.998	0.999	0.998
			100	0.998	0.998	0.998	0.998	0.999	0.998
		$\sqrt{p} = 26$	1	0.999	0.999	0.999	0.999	1	0.999
			2	0.999	0.999	0.999	0.999	1	0.999
			3	0.999	0.999	0.999	0.999	1	0.999
			4	0.999	0.999	0.999	0.999	1	0.999
			5	0.999	0.999	0.999	0.999	1	0.999
			10	0.999	0.999	0.999	0.999	1	0.999
			15	0.999	0.999	0.999	0.999	1	0.999
			20	0.999	0.999	0.999	0.999	1	0.999
			100	0.999	0.999	0.999	0.999	1	0.999
		$4\sqrt{p} = 103$	1	0.998	0.998	0.998	0.998	0.999	0.998
			2	0.998	0.998	0.998	0.998	0.999	0.998
			3	0.998	0.998	0.998	0.998	0.999	0.998
			4	0.998	0.998	0.998	0.998	0.999	0.998
			5	0.998	0.998	0.998	0.998	0.999	0.998
			10	0.997	0.997	0.997	0.997	0.999	0.997
			15	0.997	0.997	0.997	0.997	0.999	0.997
			20	0.997	0.997	0.997	0.997	0.999	0.997
			100	0.996	0.996	0.996	0.996	0.999	0.997

Tabla 3.9: Indicadores de evaluación al modelo *SVM* utilizando el 90 % de varianza. Se variaron los valores del parámetro ρ (kernel scale) y C (box constrain).

Resumen de fin de capítulo

En este capítulo se ha describió la metodología propuesta para detección y localización de daños para un aerogenerador *offshore* con plataforma tipo *jacket* utilizando solo datos de

respuesta a la vibración. Parte de la metodología es el muestreo y remodelado de los datos, es destacable mencionar que es la parte más importante de este capítulo ya que el submuestreo nos va a dar una base de datos más pequeña y el remodelar los datos va a hacer que una muestra contenga más información que antes. Se estudió la importancia de estandarizar los datos y se habló sobre la técnica PCA y los modelos de aprendizaje automático y algunas técnicas para saber cómo elegir el mejor dependiendo de la aplicación. Los modelos elegidos fueron el k-NN y SVM, el primero se eligió por ser el más sencillo para trabajar y se usó con el fin de realizar una comparación con el método SVM. El segundo fue elegido debido a que pudimos observar el comportamiento cuadrático de los datos. Esta metodología se aplicó a los modelos de aprendizaje automático antes mencionados, que de acuerdo a las métricas de evaluación de modelos tienen el mejor rendimiento, y los resultados finales se muestran en el siguiente capítulo.

Capítulo 4

Resultados

A lo largo de estas páginas se han tomado datos a partir de un banco experimental ubicado en el CoDAlab de la Universidad Politécnica de Cataluña, se ha aplicado la metodología propuesta y se han entrenado los modelos de clasificación elegidos, los resultados obtenidos de los algoritmos k-NN y SVM se presentan en este capítulo.

4.0.1. Resultados para el método de clasificación k-NN

Retomando la Tabla 3.7, se observa que el mejor desempeño del método k-NN se obtiene con una varianza del 85 % lo cual representa 282 componentes principales de un total de 4,476 y una $k = 3$.

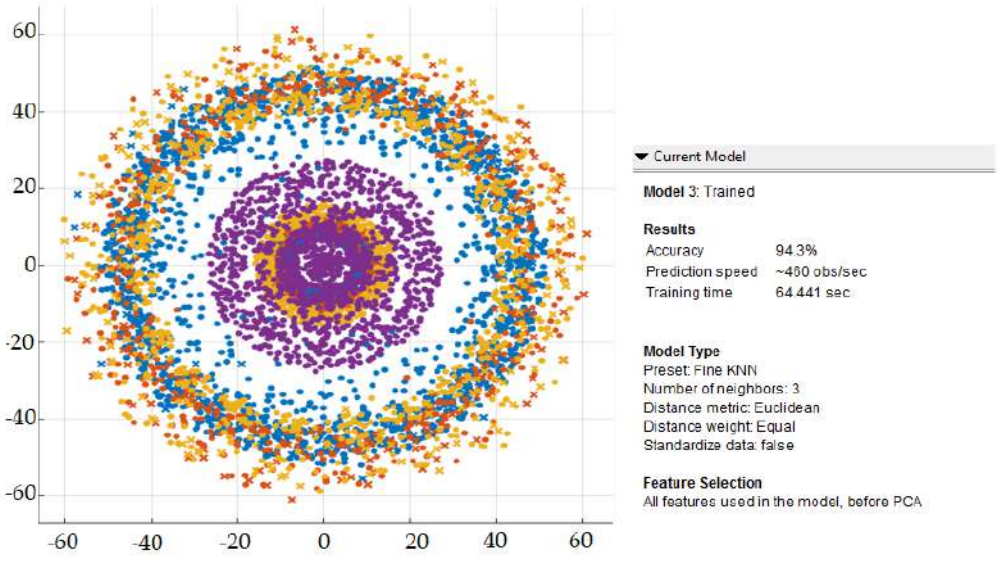


Figura 4.1: Gráfico de dispersión de predicciones obtenidas con el método k-NN.

De acuerdo a la Figura 4.1 conocemos que la exactitud obtenida con este método es del 94.3 %. Esto quiere decir que este modelo ha sido capaz de clasificar correctamente ese porcentaje de datos. Estos resultados los podemos ver de forma más detallada mediante una matriz de confusión. Por un lado, la Figura 4.2 evidencía el número de muestras clasificadas correcta e incorrectamente para cada clase. Las muestras que fueron clasificadas de forma correcta se ubican en la diagonal de la matriz y se encuentran resaltadas en tonalidades verdes mientras que las muestras que se han clasificadas incorrectamente se encuentran en tonalidades de color rojo. Por otro lado, la Figura 4.3 muestra estos mismos datos pero representados en porcentajes. También se puede apreciar el porcentaje de verdaderos negativos y falsos positivos, por lo que este modelo queda de la siguiente manera: La clase 0 que representa al caso saludable es clasificada correctamente en un 99 %. La clase 1 que representa al caso replica saludable solo fue clasificado correctamente en un 83 %. La clase 2 representa al caso con una fisura, fue clasificado correctamente en un 90 %. Por último, la clase 3 representa al caso con un perno flojo y fue clasificado correctamente en un 99 %.

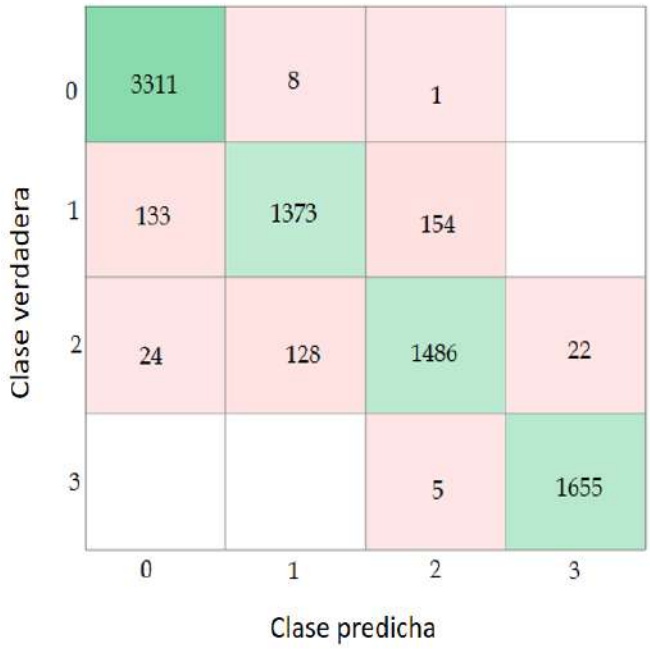


Figura 4.2: Matriz de confusión del modelo k-NN, representada por número de muestras.

Aunque los falsos negativos tienen porcentajes bajos, los resultados obtenidos pueden mejorar con el método *SVM*. Esto es debido a la forma que tienen nuestros datos (ver Figura 3.8), como se observa tenemos datos con una alta dimensión y no linealmente separables.

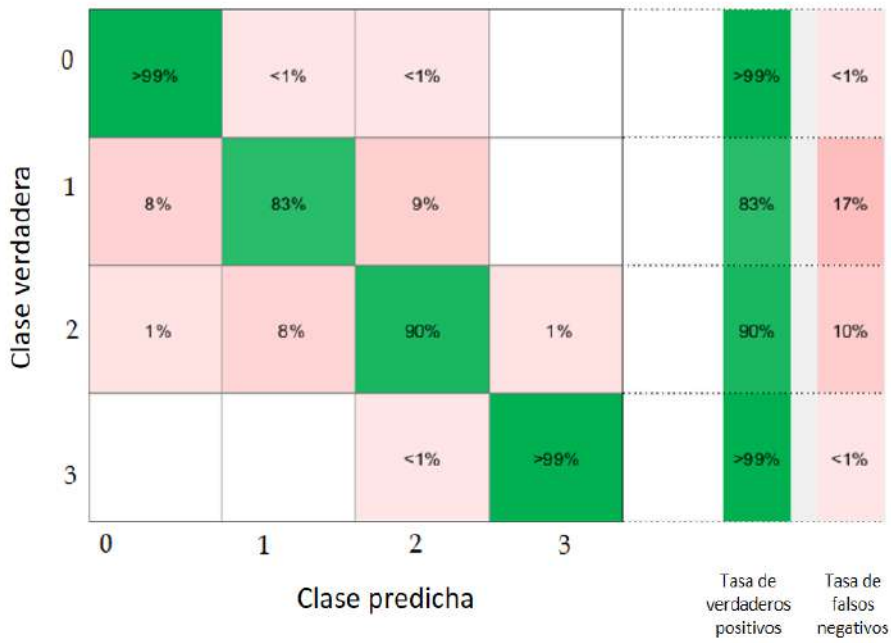


Figura 4.3: Matriz de confusión (tasa de verdaderos positivos & falsos negativos).

4.0.2. Resultados para el método de clasificación *SVM*

Los resultados presentados a continuación corresponden al método *SVM*. De acuerdo a las Tablas 3.8 y 3.9 se aprecia que el mejor desempeño se tiene cuando existe una varianza del 85 % lo cual representa 282 componentes principales de un total de 4,476, se tiene un *kernel scale* $\rho = 17$ y una $C = 1$. Con estos valores se obtiene un 99.9 % de exactitud en la clasificación (ver Figura 4.4). En las Figuras 4.5 y 4.6 tenemos la matriz de confusión para este modelo donde se aprecia el incremento en cuanto a tasa de verdaderos positivos teniendo resultados por arriba del 90 %, esto indica que las muestras se han clasificado correctamente casi en su totalidad.

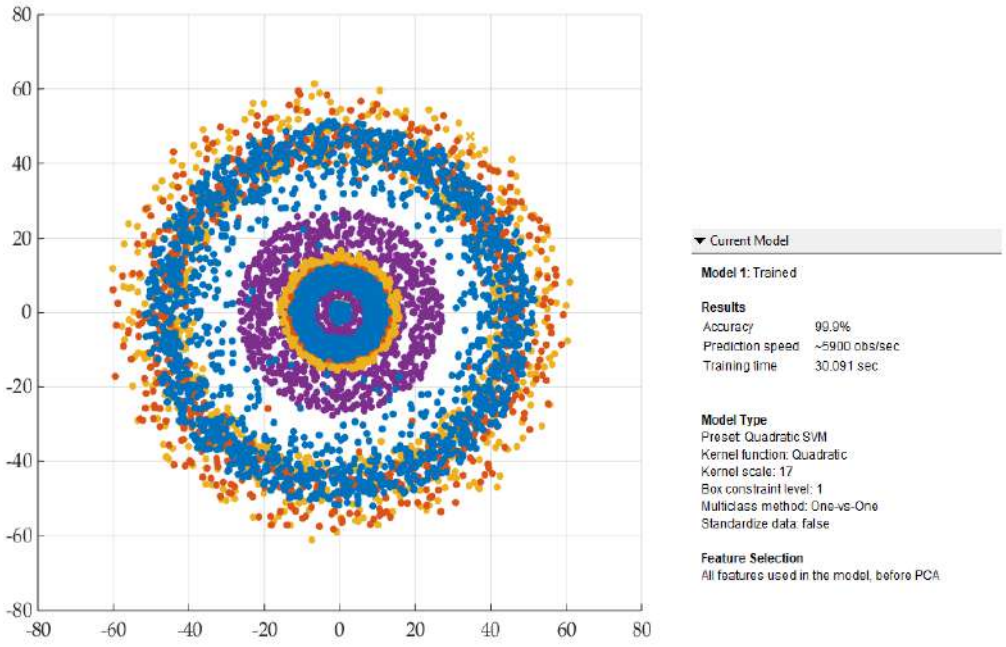


Figura 4.4: Gráfico de dispersión de predicciones obtenidas con el método *SVM*.

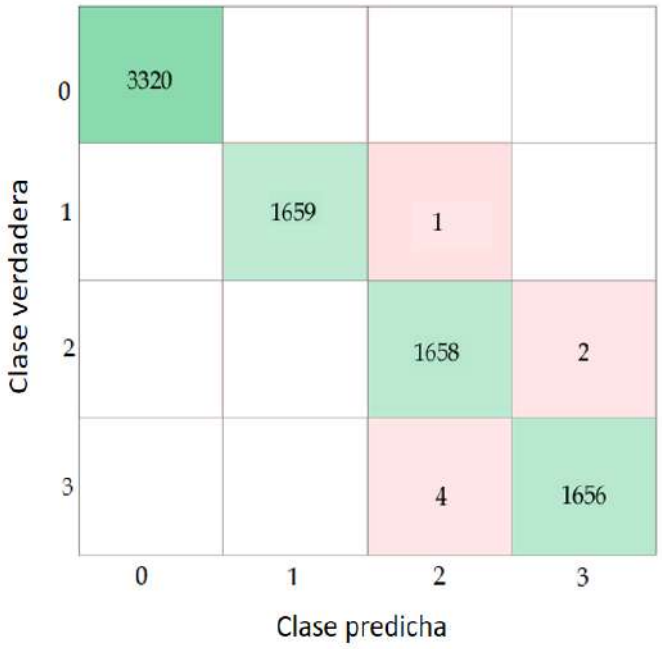


Figura 4.5: Matriz de confusión para modelo de clasificación *SVM*, representa el número de muestras.

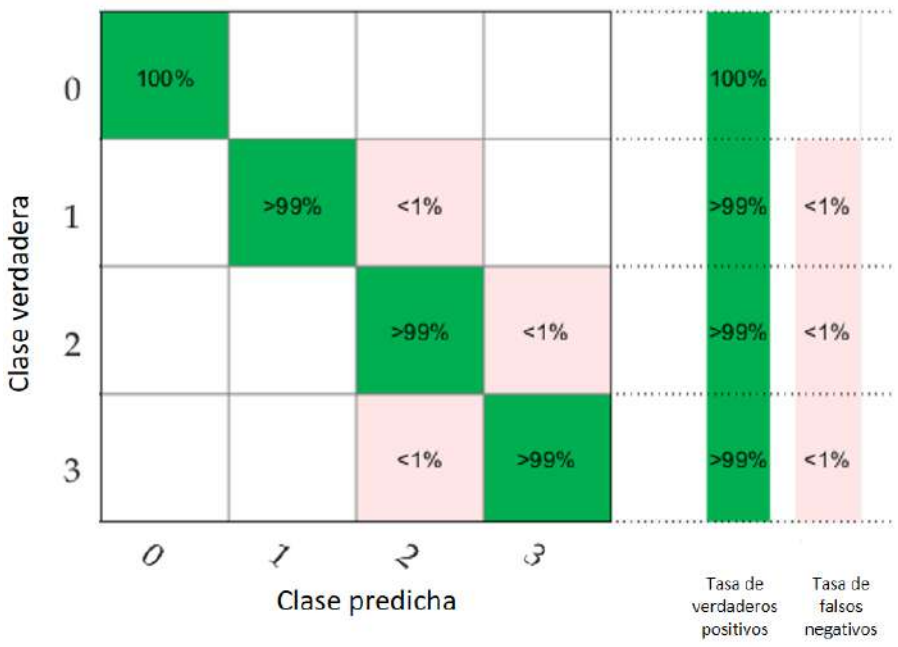


Figura 4.6: Matriz de confusión (verdaderos positivos & falsos negativos).

La exactitud que se obtiene con este método es del 99.9 %. La clase 0 que representa al caso saludable es clasificada correctamente en un 100 %, mientras el resto de clases se clasificaron correctamente en un 99 %. Los falsos negativos presentan realmente un porcentaje bajo, de menos del 1 %, lo cual es muy conveniente para resolver la problemática propuesta.

Por último, podemos observar en la Tabla 4.1 los recursos computacionales que se utilizan en cada uno de los métodos de clasificación elegidos. El método *SVM* tiene un menor tiempo de entrenamiento, categoriza más muestras por segundo y tiene una mayor exactitud en comparación con el método k-NN.

Parámetros	k-**NN**	***SVM***
Exactitud	94.3 %	99.9 %
Tiempo de predicción Aprox. (obs/seg)	460	5900
Tiempo de entrenamiento (seg)	64	30

Tabla 4.1: Tabla de comparación de parámetros computacionales.

Resumen de fin de capítulo

En este capítulo se mostraron los resultados obtenidos a partir de aplicar la metodología propuesta en el capítulo 3.

Los modelos que se entrenaron fueron k-NN con el parámetro $k = 3$ y SVM con parámetros $\rho = 17$ y $C = 17$ ambos modelos con 85 % de la varianza ya que resultaron tener el mejor redimiendo en cada uno de estos. Se presentaron detalles de cada modelo como la exactitud, la velocidad de predicción y el tiempo de entrenamiento. Para cada modelo se analizaron e interpretaron las matrices de confusión tanto la que está representada por el número de muestras como en la que se observa la tasa de verederos positivos y falsos negativos. Aunque los dos modelos presentaron resultados favorables el modelo SVM resulto tener mejor rendimiento para resolver la problemática presentada.

Conclusiones

En la tesis presentada, se ha mostrado la metodología propuesta para detectar e identificar daños en aerogeneradores *offshore*, específicamente en los que cuentan con un soporte tipo *jacket*, mediante técnicas de aprendizaje automático. Haber validado esta metodología de manera experimental en una plataforma a escala, nos brinda la posibilidad de aplicarla en un futuro a plataformas *offshore* de tamaño real que se encuentran en entornos de difícil acceso. Incluso, esta metodología podría ser adaptada a aerogeneradores *onshore* como los localizados en los parques eólicos del estado de Puebla.

Con el trabajo realizado se puede concluir lo siguiente: la recolección de los datos es importante para el desarrollo de un modelo de aprendizaje automático, cuanto más y mejores datos obtengamos, mejor será el rendimiento de nuestro modelo. Sin embargo, esto puede resultar un problema de procesamiento computacional. Por ello, una contribución destacable en este trabajo es la forma como se recopilan, organizan, escalan, transforman y reducen las dimensiones de los datos procedentes de diferentes instantes de tiempo, sensores y experimentos. Con el desarrollo propuesto, una muestra tendrá más información teniendo como referencia las muestras a las cuales no se le realiza este tratamiento.

La selección del algoritmo dependerá en gran parte del tipo de problema al que nos enfrentemos y de cómo se encuentren agrupados los datos a tratar. En este caso fueron seleccionados dos algoritmos de clasificación de aprendizaje automático supervisado: k**-NN** y *SVM*.

Es importante tener claro cómo se va a determinar el método de clasificación que resulta tener el mejor desempeño. Para este problema fue conveniente sintonizar los parámetros de cada algoritmo y obtener el valor de las métricas de evaluación de cada uno de los métodos de clasificación. Las métricas más importantes para este caso resultaron ser la sensibilidad y la exactitud.

De los dos algoritmos elegidos, el mejor desempeño se obtuvo con el clasificador *SVM* cuadrático, al contar con $\rho = 17$, $C = 1$ y 282 componentes principales (85 % de varianza). Este tuvo un rendimiento muy cercano al ideal, logrando en todos los indicadores un resultado igual o superior a 99.99 % con una velocidad de predicción de $5,900$ obs/seg y un tiempo de

entrenamiento 30 seg. Esto se atribuye a que el algoritmo fue elegido de acuerdo a la forma de los datos que se manejaron.

Apéndice A

Códigos de Matlab

Listing A.1: Adquisición de datos

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ADQUISICION DE DATOS
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
clc;
close all;
%---------------------------------------------------------------------
%% Crea una sesion de adquisicion de datos

s = daq.createSession('ni'); % Crea una sesion para un proveedor especifico,
                             % en este caso National Instruments
%---------------------------------------------------------------------
%% Conjunto de propiedades de la sesion

% Como propiedades se puede elegir el numero de muestras que se desea
% obtener o el tiempo durante el cual se desea muestrear.

% s.DurationInSeconds = 60;  % Duracion de tiempo de toma de muestras

s.NumberOfScans = 99097;     % Numero de muestras
%---------------------------------------------------------------------
%% Agregar los canales a la sesion
% Se agregan los canales y el conjunto de propiedaes

% Primer acelerometro
channelA1X = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod1','ai0','Voltage');
channelA1X.Name = 'A1X'; % Se renombra el canal para X
channelA1Y = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod1','ai1','Voltage');
channelA1Y.Name = 'A1Y'; % Se renombra el canal para Y
channelA1Z = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod1','ai2','Voltage');
channelA1Z.Name = 'A1Z'; % Se renombra el canal para Z
```

```
% Segundo acelerometro
channelA2X = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod2','ai0','Voltage');
channelA2X.Name = 'A2X';
channelA2Y = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod2','ai1','Voltage');
channelA2Y.Name = 'A2Y';
channelA2Z = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod2','ai2','Voltage');
channelA2Z.Name = 'A2Z';

% Tercer acelerometro
channelA3X = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod3','ai0','Voltage');
channelA3X.Name = 'A3X';
channelA3Y = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod3','ai1','Voltage');
channelA3Y.Name = 'A3Y';
channelA3Z = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod3','ai2','Voltage');
channelA3Z.Name = 'A3Z';

% Cuarto acelerometro
channelA4X = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod4','ai0','Voltage');
channelA4X.Name = 'A4X';
channelA4Y = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod4','ai1','Voltage');
channelA4Y.Name = 'A4Y';
channelA4Z = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod4','ai2','Voltage');
channelA4Z.Name = 'A4Z';

% Quinto acelerometro
channelA5X = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod5','ai0','Voltage');
channelA5X.Name = 'A5X';
channelA5Y = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod5','ai1','Voltage');
channelA5Y.Name = 'A5Y';
channelA5Z = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod5','ai2','Voltage');
channelA5Z.Name = 'A5Z';

% Sexto acelerometro
channelA6X = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod6','ai0','Voltage');
channelA6X.Name = 'A6X';
channelA6Y = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod6','ai1','Voltage');
channelA6Y.Name = 'A6Y';
channelA6Z = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod6','ai2','Voltage');
channelA6Z.Name = 'A6Z';

% Septimo acelerometro
channelA7X = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod1','ai3','Voltage');
channelA7X.Name = 'A7X';
channelA7Y = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod2','ai3','Voltage');
channelA7Y.Name = 'A7Y';
channelA7Z = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod3','ai3','Voltage');
channelA7Z.Name = 'A7Z';

% Octavo acelerometro
channelA8X = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod4','ai3','Voltage');
channelA8X.Name = 'A8X';
channelA8Y = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod5','ai3','Voltage');
channelA8Y.Name = 'A8Y';
channelA8Z = addAnalogInputChannel(s,'cDAQ9188XT-1BFC51FMod6','ai3','Voltage');
```

```
channelA8Z.Name = 'A8Z';

%% Adquisicion de datos
% Inicia la sesion en foreground (primer plano)
[data, timestamps, starttime] = startForeground(s);

%Data contiene los datos en el mismo orden que se han abierto los canales

%% Clean Up
%Limpia la sesion y los canales.
clear s channelA1X channelA1Y channelA1Z channelA2X channelA2Y channelA2Z ...
        channelA3X channelA3Y channelA3Z channelA4X channelA4Y channelA4Z ...
        channelA5X channelA5Y channelA5Z channelA6X channelA6Y channelA6Z ...
        channelA7X channelA7Y channelA7Z channelA8X channelA8Y channelA8Z

%Nombre dado a cada experimento
save 1_1_1A.mat
```

Listing A.2: Extracción de ficheros y submuestreo

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PROGRAMA PARA EXTRAER DATO EN CRUDO DE FICHEROS Y PROCESARLOS HASTA
%OBTENER UNA REDUCCION DE DIMENSIONES.
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
clc;
close all;
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bucle BARRA ORIGINAL (SALUDABLE) (40 EXPERIMENTOS)
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fileInfo1=dir('1_*.mat');   %Esta funci n de MATLAB enumera archivos y carpetas en la ↩
    carpeta actual.
experimentos_1=10;          %n mero de toma de muestras con la barra original
amplitudes=4;

for k=1:experimentos_1      %n mero de experimento
    for i=1:amplitudes              %toma de amplitud (1A,2A,3A,05A)
      %------------------------------------------------------------------
        if i==amplitudes            %si i=4 entonces  (PARA CASO 05A)
              i='05';               %asigna un 05 para obtener los datos de la amplitud 05A
      %------------------------------------------------------------------
        fileName=['1_' num2str(k) '_' num2str(i) 'A.mat'];
        fromMat=load(fileName);
        x0=(fromMat.data);
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
             delta_T=1/256;      %Por tanto delta T=0.0039s.
             x1=x0(1:6:end, :); %Nueva base de datos cada 0.0039 seg aproximadamente
             [fila,columna]=size(x1);
            %-------------------------------------------------------------
```

```
        n=199;          % tama o  del  conjunto
        m= fila/n;      % filas  resultantes  para  x2
       %------------------------------------------------
        x2=[];      % inicializaci  n  del  vector  x2,  vector  vacio
                for  j=1:columna          % corrimiento  sobre  columnas  que  representan ↩
                    los  datos  obtenidos  de   las  se  ales  de  los  sensores
                    for   h=1:m             % corrimiento  sobre  las  filas  para  formar  los↩
                        conjuntos  de  datos
                        min=(h-1)*n+1;   % valor  m  nimo
                        max=h*n;         % valor  m  ximo
                        grupos= x1(min:max,j);
                        bloque(h,:)=grupos';
                    end
                    x2=[x2,bloque];
                end
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
         %-------------
         i=str2num('05');     % convierte  el  string  en  un  caracter  n  merico
         %-------------
                    if  k==1 && i==1
                        x2T1=x2;
                    else
                        x2T1=[x2T1;x2];
                    end
    %----------------------------------------------------
     else               %PARA  CASOS  1A,  2A  Y  3A
    %----------------------------------------------------
     fileName=['1_' num2str(k)  '_' num2str(i)  'A.mat'];
     fromMat=load(fileName);
     x0=(fromMat.data);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        delta_T=1/256;       % Por  tanto  delta  T=0.0039s.
        x1=x0(1:6:end,  :);  % Nueva  base  de  datos  cada  0.0039  seg  aproximadamente
        [fila,columna]=size(x1);
       %------------------------------------------------
        n=199;          % tama o  del  conjunto
        m= fila/n;      % filas  resultantes  para  x2
       %------------------------------------------------
        x2=[];      % inicializaci  n  del  vector  x2,  vector  vacio
                for  j=1:columna         % corrimiento  sobre  columnas  que  representan ↩
                    los  datos  obtenidos  de   las  se  ales  de  los  sensores
                    for   h=1:m            % corrimiento  sobre  las  filas  para  formar  los ↩
                       conjuntos  de  datos
                        min=(h-1)*n+1;  % valor  m  nimo
                        max=h*n;        % valor  m  ximo
                        grupos= x1(min:max,j);
                         bloque(h,:)=grupos';
                    end
                    x2=[x2,bloque];
                end
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    if  k==1 && i==1
                        x2T1=x2;
                    else
```

```
                                x2T1=[x2T1;x2];
                            end
            end
        end
end
            disp ('Los factores de')
              disp (fila)
              disp ('son')
              fac=factor(fila);    %los factores son 83 y 199
              disp (fac)
              disp ('Se usa el factor 199,')
              disp (' ')
              disp ('Este n mero representa en segundos: ')
              seg=0.0036328*n;
              disp(seg)
              disp (' ')
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
              disp ('--------------------------------')
              disp ('N mero de filas para x2_SALUDABLE son:')
              filasT1=experimentos_1*amplitudes*m;
              disp (filasT1)

%categoria
for i=1:filasT1
    d_str1(i,1)= 0; %Barra original sana
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Bucle BARRA REPLICA (SALUDABLE) (20 EXPERIMENTOS)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fileInfo2=dir('2_*.mat'); %Esta funcion de MATLAB enumera archivos y carpetas en la ↩
    carpeta actual.
experimentos_2=5;         %numero de toma de muestras con la barra replica saludable
amplitudes=4;

for k=1:experimentos_2     %numero de experimento
    for i=1:amplitudes               %toma de amplitud (1A,2A,3A,05A)

      %-----------------------------------------------------
        if i==amplitudes           %si i=4 entonces  (PARA CASO 05A)
            i='05';                %asigna un 05 para obtener los datos de la amplitud 05A
      %-----------------------------------------------------
        fileName=['2_' num2str(k) '_' num2str(i) 'A.mat'];
        fromMat=load(fileName);
        x0=(fromMat.data);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
              delta_T=1/256;      %Por tanto delta T=0.0039s.
              x1=x0(1:6:end,:); %Nueva base de datos cada 0.0039 seg aproximadamen
              [fila,columna]=size(x1);
             %-----------------------------------------------
              n=199;         %tama o del conjunto
              m= fila/n;     %filas resultantes para x2
             %-----------------------------------------------
              x2=[];      %inicializaci n del vector x2, vector vacio
                    for j=1:columna          %corrimiento sobre columnas que representan ↩
```

```
                    los datos obtenidos de  las se ales de los sensores
                    for  h=1:m            %corrimiento sobre las filas para formar los↩
                         conjuntos de datos
                         min=(h-1)*n+1;   %valor  m nimo
                         max=h*n;         %valor  m ximo
                         grupos= x1(min:max,j);
                         bloque(h,:)=grupos';
                    end
                    x2=[x2,bloque];
                 end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          %--------------
          i=str2num('05');     %convierte el string en un caracter n merico
          %--------------

                    if k==1 && i==1
                        x2T2=x2;
                    else
                        x2T2=[x2T2;x2];
                    end
    %----------------------------------------------------
     else                 %PARA CASOS 1A, 2A Y 3A
    %----------------------------------------------------

     fileName=['2_' num2str(k) '_' num2str(i) 'A.mat'];
     fromMat=load(fileName);
     x0=(fromMat.data);
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
         delta_T=1/256;      %Por tanto delta T=0.0039s.
         x1=x0(1:6:end, :); %Nueva base de datos cada 0.0039 seg aproximadamen
         [fila,columna]=size(x1);
        %-----------------------------------------------
         n=199;          %tama o del conjunto
         m= fila/n;      %filas resultantes para x2
        %-----------------------------------------------
         x2=[];      %inicializaci n del vector x2, vector vacio
                for j=1:columna          %corrimiento sobre columnas que representan ↩
                    los datos obtenidos de  las se ales de los sensores
                    for  h=1:m           %corrimiento sobre las filas para formar los ↩
                        conjuntos de datos
                         min=(h-1)*n+1; %valor  m nimo
                         max=h*n;       %valor  m ximo
                         grupos= x1(min:max,j);
                          bloque(h,:)=grupos';
                    end
                    x2=[x2,bloque];
                 end
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                    if k==1 && i==1
                        x2T2=x2;
                    else
                        x2T2=[x2T2;x2];
                    end
```

```
        end
    end
end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            disp ('—————————————————————')
            disp ('N mero de filas para x2_REPLICA son:')
            filasT2=experimentos_2*amplitudes*m;
            disp (filasT2)

% categoria
for i=1:filasT2
  d_str2(i,1)= 1; % barra replica sana
end
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bucle BARRA CON CRACK  (20 EXPERIMENTOS)
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fileInfo3=dir('3_*.mat'); % Esta funcion de MATLAB enumera archivos y carpetas en la ↩
    carpeta actual.
experimentos_3=5;       % numero de toma de muestras con la barra CON CRACK
amplitudes=4;
for k=1:experimentos_3      % numero de experimento
    for i=1:amplitudes              % toma de amplitud (1A,2A,3A,05A)

      %—————————————————————————————————————
        if i==amplitudes           % si i=4 entonces  (PARA CASO 05A)
              i='05';              % asigna un 05 para obtener los datos de la amplitud 05A
      %—————————————————————————————————————

        fileName=['3_' num2str(k) '_' num2str(i) 'A.mat'];
        fromMat=load(fileName);
        x0=(fromMat.data);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            delta_T=1/256;      %Por tanto delta T=0.0039s.
            x1=x0(1:6:end, :); % Nueva base de datos cada 0.0039 seg aproximadamen
            [fila,columna]=size(x1);
           %————————————————————————————————
            n=199;          %tama o del conjunto
            m= fila/n;      %filas resultantes para x2
           %————————————————————————————————

            x2=[];          % inicializaci n del vector x2, vector vacio
                    for j=1:columna           % corrimiento sobre columnas que representan ↩
                        los datos obtenidos de  las se ales de los sensores
                        for  h=1:m            % corrimiento sobre las filas para formar los↩
                            conjuntos de datos
                            min=(h-1)*n+1;   % valor m nimo
                            max=h*n;         % valor m ximo
                            grupos= x1(min:max,j);
                            bloque(h,:)=grupos';
                        end
                        x2=[x2,bloque];
                    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
              %—————————
```

```
            i=str2num('05');                % convierte el string en un caracter n merico
            %------------

                        if k==1 && i==1
                            x2T3=x2;
                        else
                            x2T3=[x2T3;x2];
                        end
        %--------------------------------------------------
        else                %PARA CASOS 1A, 2A Y 3A
        %--------------------------------------------------

        fileName=['3_' num2str(k) '_' num2str(i) 'A.mat'];
        fromMat=load(fileName);
        x0=(fromMat.data);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            delta_T=1/256;      % Por tanto delta T=0.0039s.
            x1=x0(1:6:end, :); % Nueva base de datos cada 0.0039 seg aproximadamen
            [fila,columna]=size(x1);
            %--------------------------------------------
            n=199;          %tama o del conjunto
            m= fila/n;      %filas resultantes para x2
            %--------------------------------------------
            x2=[];      % inicializaci n del vector x2, vector vacio
                    for j=1:columna         % corrimiento sobre columnas que representan ↩
                        los datos obtenidos de  las se ales de los sensores
                        for  h=1:m          % corrimiento sobre las filas para formar los ↩
                            conjuntos de datos
                            min=(h-1)*n+1;  % valor m nimo
                            max=h*n;        % valor m ximo
                            grupos= x1(min:max,j);
                             bloque(h,:)=grupos';
                        end
                        x2=[x2,bloque];
                    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                        if k==1 && i==1
                            x2T3=x2;
                        else
                            x2T3=[x2T3;x2];
                        end
        end
    end
end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            disp ('------------------------------------')
            disp ('N mero de filas para x2_CRACK son:')
            filasT3=experimentos_3*amplitudes*m;
            disp (filasT3)

% categoria
for i=1:filasT3
```

```
  d_str3(i,1)= 2; %crack
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bucle BARRA PERNO FLOJO (20 EXPERIMENTOS)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fileInfo4=dir('4_*.mat'); % Esta funcion de MATLAB enumera archivos y carpetas en la ↩
    carpeta actual.
experimentos_4=5;       % n mero de toma de muestras con la barra PERNO FLOJO
amplitudes=4;


for k=1:experimentos_4      % numero de experimento
    for i=1:amplitudes              % toma de amplitud (1A,2A,3A,05A)

      %-------------------------------------------------------
        if i==amplitudes           % si i=4 entonces (PARA CASO 05A)
              i='05';              % asigna un 05 para obtener los datos de la amplitud 05A
      %-------------------------------------------------------

        fileName=['4_' num2str(k) '_' num2str(i) 'A.mat'];
        fromMat=load(fileName);
        x0=(fromMat.data);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            delta_T=1/256;      %Por tanto delta T=0.0039s.
            x1=x0(1:6:end, :); % Nueva base de datos cada 0.0039 seg aproximadamen
            [fila,columna]=size(x1);
           %---------------------------------------------
            n=199;         % tama o del conjunto
            m= fila/n;     % filas resultantes para x2
           %---------------------------------------------

            x2=[];      % inicializaci n del vector x2, vector vacio
                    for j=1:columna          % corrimiento sobre columnas que representan ↩
                        los datos obtenidos de  las se ales de los sensores
                        for h=1:m            % corrimiento sobre las filas para formar los↩
                             conjuntos de datos
                             min=(h-1)*n+1;  % valor m nimo
                             max=h*n;        % valor m ximo
                             grupos= x1(min:max,j);
                             bloque(h,:)=grupos';
                        end
                        x2=[x2,bloque];
                    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
             %-------------
             i=str2num('05');     % convierte el string en un caracter n merico
             %-------------

                        if k==1 && i==1
                            x2T4=x2;
                        else
                            x2T4=[x2T4;x2];
                        end
```

```
        %-------------------------------------------------------
         else                %PARA CASOS 1A, 2A Y 3A
        %-------------------------------------------------------

         fileName=['4_' num2str(k) '_' num2str(i) 'A.mat'];
         fromMat=load(fileName);
         x0=(fromMat.data);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
             delta_T=1/256;      %Por tanto delta T=0.0039s.
             x1=x0(1:6:end, :); %Nueva base de datos cada 0.0039 seg aproximadamen
             [fila,columna]=size(x1);
            %-------------------------------------------------
             n=199;          %tama o del conjunto
             m= fila/n;      %filas resultantes para x2
            %-------------------------------------------------
             x2=[];      %inicializaci n del vector x2, vector vacio
                     for j=1:columna          %corrimiento sobre columnas que representan ↩
                         los datos obtenidos de  las se ales de los sensores
                         for  h=1:m           %corrimiento sobre las filas para formar los ↩
                             conjuntos de datos
                              min=(h-1)*n+1; %valor m nimo
                              max=h*n;       %valor m ximo
                              grupos= x1(min:max,j);
                               bloque(h,:)=grupos';
                         end
                         x2=[x2,bloque];
                     end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                         if k==1 && i==1
                             x2T4=x2;
                         else
                             x2T4=[x2T4;x2];
                         end
        end
    end
end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
             disp ('-------------------------------------')
             disp ('N mero de filas para x2_PERNO son:')
             filasT4=experimentos_4*amplitudes*m;
             disp (filasT4)
% categoria
for i=1:filasT4
  d_str4(i,1)=3 ; % perno
end

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
             disp (' ')
             disp ('N mero total de columnas para x2Total son:')
             col=columna*n;
             disp (col)
             disp (' ')
             disp ('N mero total de filas para x2Total son:')
             tfil=filasT1+filasT2+filasT3+filasT4;
```

```
            disp (tfil)

x2Total=[x2T1;
         x2T2;
         x2T3;
         x2T4];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%CATEGORIZACION DE LOS DATOS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp (' ')
disp ('Los datos se etiquetan de la siguiente manera:')
disp ('Barra Original Saludable = 0')
disp ('Barra Replica Saludable  = 1')
disp ('Barra Crack/Fisura       = 2')
disp ('Barra Perno Flojo        = 3')



%ETIQUETAS
d_str=[d_str1;
    d_str2;
    d_str3;
    d_str4];

%categorias
d_Activity = categorical(d_str);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%NORMALIZACION DE DATOS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

a_Y=x2Total;
[a_m,a_n]=size(a_Y);     %Tama o de matriz de datos

%%
%Normalizamos cada columna (cada variable) de manera independiente. Es decir:
%A cada columna restamos la media y dividimos por la desviacion de esa columna

b_media=mean(a_Y);      %media es un vector que contiene la media de cada columna
b_desviacion=std(a_Y); %std es un vector que contiene la desviacion tipica de cada ↩
    columna

for j=1:a_n
    b_X(:,j)=(a_Y(:,j)-b_media(j))/(b_desviacion(j));   %b_X es la matriz de datos ya ↩
        normalizada
end

%Comprobamos que la media de cada columna de la matrix X es 0
b_mean=mean(b_X);

%Comprobamos que la desviacion tipica de cada columna de la matriz b_X es 1
b_desviacion=std(b_X);
```

```
%% SE APLICA PCA(obtenemos todas las componentes principales!!)
[b_coeff,b_score,b_latent] = pca(b_X); % 4776 componentes principales

% "b_coeff" son las componentes principales que de hecho son los vectores propios de la ↩
    matriz de covarianza (ordenados de mayor a menor).
% "b_score" proyecciones de los datos X en la nueva base.
% "b_latent" valores propios de la matriz de varianza
```

Listing A.3: Código Matlab para determinar el número de componentes principales.

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% REDUCCION DE DIMENSION (reduccion de columnas, eleccion de Componentes Principales a ↩
    trabajar)
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[f_latent, c_latent]=size(b_latent); % Tama o de la matriz latent (valores propios)

% Para obtener lambda_n (denominador)
for g=1:f_latent   % contador de 1 hasta 4776
     if g == 1
          lambda_n= b_latent(g,1);
     else
         lambda_n= lambda_n+b_latent(g,1);
     end
end

% Obtiene el porcentaje de exactitud para saber cuantas componentes
% principales se requieren usar.

for g=1:f_latent   % contador de 1 hasta 4776
     if g == 1
          lambda_d= b_latent(g,1);                   % Para obtener lambda_d (numerador)
          porcentaje(g,:)= (lambda_d/lambda_n)*100;
     else
         lambda_d= lambda_d+b_latent(g,1);           % Para obtener lambda_d (numerador)
         porcentaje(g,:)= (lambda_d/lambda_n)*100;
     end
end


% Grafica de porcentajes
plot(porcentaje)
title ('Grafica de porcentajes')   % titulo de la grafica
grid on
xlabel ('Componentes principales') % eje x
ylabel ('Porcentaje')              % eje y

disp (' Porcentaje de exactitud de acuerdo a las componentes principales utilizadas:')
disp ('95% --- 1500 CP')
disp ('90% ---  665 CP')
disp ('85% ---  282 CP')
```

```
disp ('.')
disp ('N mero de componentes principales utilizadas (1500,665 o 282):')
disp ('.')
Num_CP=input('  Cuntos   componentes principales desea utilizar?  ');
%1500 componentes principales para asegurar el 95%
% 665 componentes principales para asegurar el 90%
% 282 componentes principales para asegurar el 85%
[c_coeff,c_score,c_latent]=pca(b_X,'NumComponents',Num_CP);



SensorDataTrain= table(...
    c_score, d_Activity); %Esta tabla contiene  la matriz c_score y la categoria de cada ↩
        estado

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%A partir de aqui se puede utilizar la App Classification Learner.
%entrenando el modelo KNN
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Apéndice B

Publicaciones derivadas de la tesis

B.1. Revista *Sensors*

Sensors (ISSN 1424-8220; CODEN: SENSC9) es una revista internacional de acceso abierto dedicada a la publicación de los últimos logros de desarrollos tecnológicos relacionado con los sensores, que incluyen todos los aspectos del diseño, la tecnología, la prueba de concepto y la aplicación. Esta revista es de publicación semestral en línea y forma parte del editor MDPI, el cual ha sido pionero en la publicación académica de acceso abierto desde 1996 (con sede en Basel, Suiza). MDPI fomenta el intercambio científico abierto en todas las disciplinas. Cuenta con 227 revistas de acceso abierto, respaldadas por más de 35,500 editores académicos, las cuales reciben más de 5.8 millones de visitas mensuales en su página web. Todos los manuscritos enviados se someten a una rigurosa revisión antes de su publicación. Durante el año 2018 alcanzó un factor de impacto de 3.031 como se muestra en la Figura B.1, mientras que en la Figura B.2 observamos las citas por año [55].

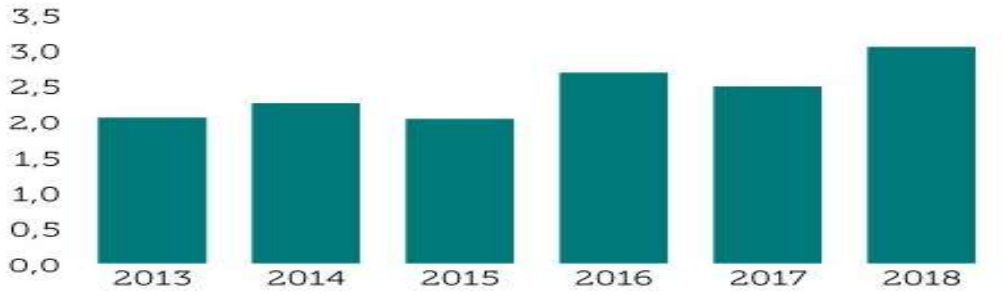


Figura B.1: Factor de Impacto revista *Sensors*

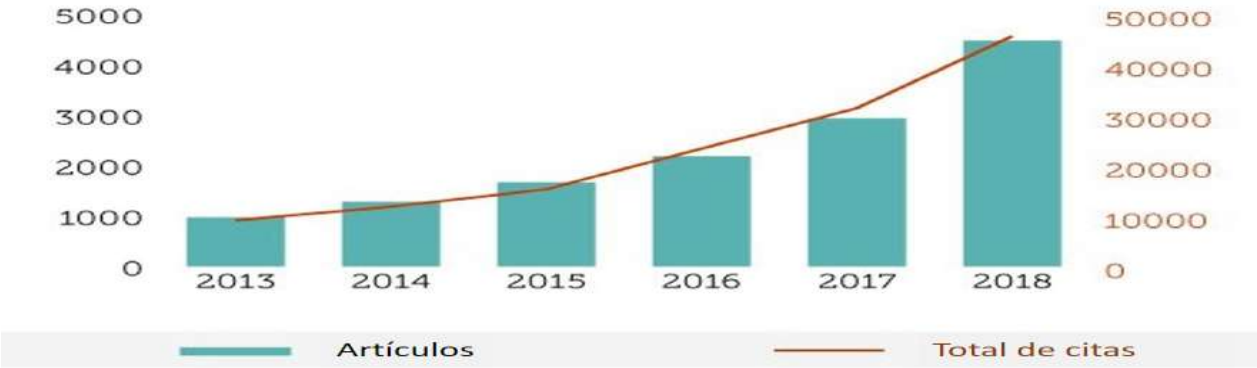


Figura B.2: Citas y artículos de la revista *Sensors* por año.

B.1.1. Artículo publicado

sensors

MDPI

Article

Structural Health Monitoring for Jacket-Type Offshore Wind Turbines: Experimental Proof of Concept

Yolanda Vidal [1,*], Gabriela Aquino [2], Francesc Pozo [1] and José Eligio Moisés Gutiérrez-Arias [2]

[1] Control, Modeling, Identification and Applications (CoDAlab), Department of Mathematics, Escola d'Enginyeria de Barcelona Est (EEBE), Universitat Politècnica de Catalunya (UPC), Campus Diagonal-Besòs (CDB), Eduard Maristany, 16, 08019 Barcelona, Spain; francesc.pozo@upc.edu

[2] Facultad de Ciencias de la Electrónica (FCE), Benemérita Universidad Autónoma de Puebla (BUAP), Av. San Claudio y 18 Sur, Ciudad Universitaria, Edificio 1FCE6/202, 72570 Puebla, Mexico; aquino.201006419@gmail.com (G.A.); arigutmses5@gmail.com (J.E.M.G.-A.)

* Correspondence: yolanda.vidal@upc.edu; Tel.: +34-934-137-309

Received: 20 February 2020; Accepted: 24 March 2020; Published: 26 March 2020

Abstract: Structural health monitoring for offshore wind turbines is imperative. Offshore wind energy is progressively attained at greater water depths, beyond 30 m, where jacket foundations are presently the best solution to cope with the harsh environment (extreme sites with poor soil conditions). Structural integrity is of key importance in these underwater structures. In this work, a methodology for the diagnosis of structural damage in jacket-type foundations is stated. The method is based on the criterion that any damage or structural change produces variations in the vibrational response of the structure. Most studies in this area are, primarily, focused on the case of measurable input excitation and vibration response signals. Nevertheless, in this paper it is assumed that the only available excitation, the wind, is not measurable. Therefore, using vibration-response-only accelerometer information, a data-driven approach is developed following the next steps: (i) the wind is simulated as a Gaussian white noise and the accelerometer data are collected; (ii) the data are pre-processed using group-reshape and column-scaling; (iii) principal component analysis is used for both linear dimensionality reduction and feature extraction; finally, (iv) two different machine-learning algorithms, *k* nearest neighbor (*k*-NN) and quadratic-kernel support vector machine (SVM), are tested as classifiers. The overall accuracy is estimated by 5-fold cross-validation. The proposed approach is experimentally validated in a laboratory small-scale structure. The results manifest the reliability of the stated fault diagnosis method being the best performance given by the SVM classifier.

Keywords: structural health monitoring; jacket-type; accelerometers; support vector machines; principal component analysis

1. Introduction

The potential of offshore wind power is enormous. In offshore wind farms, wind turbines (WTs) are erected with different types of foundations. Monopile foundations are by far the most common foundation (81%). These are quite simple structures anchored directly to the seabed. Gravity foundation systems are very rare (at 5.7% market share) as they involve using a large concrete or steel platform with a diameter of approximately 15 m and a weight of approximately 3000 tons. Finally, jackets are preferred for extreme sites with poor soil conditions as these are foundations with a lattice framework that feature three or four seabed anchoring points, which increases the levels of safety

when anchoring the towers; see Figure 1. As said previously, the potential of offshore wind power is enormous. However, it can only be exploited by diminishing operation and maintenance costs. Structural health monitoring (SHM) solutions to provide an early warning of damage are essential to accomplish this objective. Thus, this paper focuses in the problem of damage detection for jacked-type foundations.

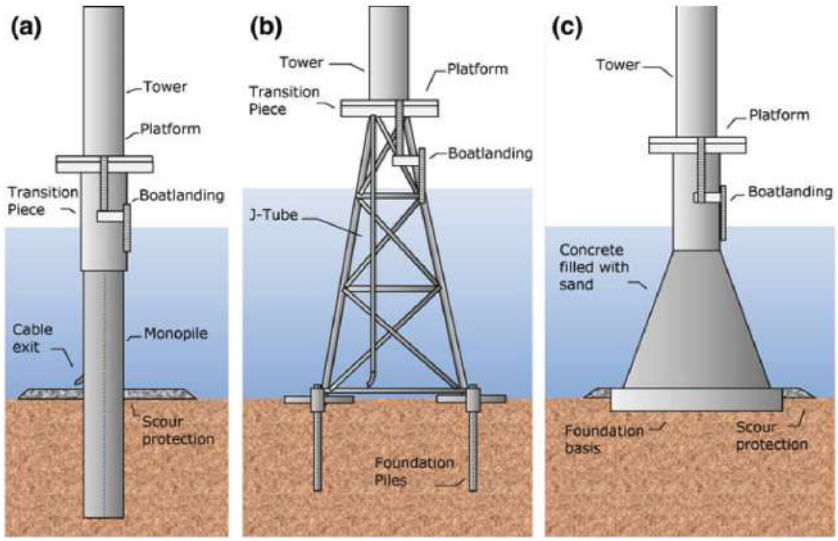


Figure 1. Fixed type WT foundations [1]. Monopile (**a**), jacket (**b**), and gravity-based (**c**).

In the literature, a lot of methodologies for damage detection can be found, among them the vibration-based methods are one of the most prolific ones, as shown in [2]. Vibration-based SHM methods are data-based approaches employing random excitation and/or vibration response signals (time series), statistical model building, and statistical decision making schemes for inferring the health state of a structure [3]. The interest in these methods has been growing in recent years, due to their simplicity, ease of use, and high effectiveness [4]. However, most studies are, primarily, focused on the case of measurable input excitation and vibration response signals, with only a few recent studies focused on the vibration-response-only case [5], the importance of which stems from the fact that in some applications the excitation cannot be imposed and is often not measurable. This work, aims to contribute in this area of vibration-response-only as the vibration excitation is given by the wind (it cannot be imposed and it is assumed to be unknown).

An overview of SHM systems for various WT components is presented, for example, in [6]. Some important studies that focus specifically on the offshore WT structure are the following. In [7] a review of SHM systems for offshore WTs has been carried out considering the topic as a statistical pattern recognition problem. In [8] health monitoring systems and operational safety evaluation techniques of the offshore wind turbine structure are systematically investigated and summarized. It is noteworthy the work of Mieloszyk et al. [9] where a SHM system is stated based on fiber Bragg grating sensors dedicated to an offshore wind turbine support structure model is presented to detect and localize crack occurrence. It is also remarkable the work of Fritzen et al. [10] where a method for online damage detection/localization is presented accompanied with on field tests of a prototype 5MW plant. In [11] a method for damage localization using finite element model updating is introduced as a subsequent step to a three tier SHM scheme for damage detection. It is also noteworthy the work of Weijtjens et al. [12] related to the foundation SHM of a real offshore monopile WT based on its resonance frequencies where the key problems are the operational and environmental variability of the resonance frequencies of the turbine that potentially conceal any structural change. Another method based on random decrement and neural networks is stated in [13]. A method based on finite element model updating and fuzzy logic is applied on a lab-scale jacket-type platform in [14]. At the lowest level of

SHM, the main objective is simply the detection of the presence of damage. In most cases, a model of normality is built [15], and data originating from the structure of interest are tested, usually after some processing, in terms of novelty (when compared to the normal model). Novel data are thus detected and can be considered indicative of damage. Although this process is generally considered less challenging than the full identification of damage, it has a great advantage: it does not need data from damaged states. Finally, in [16], where an experimental testbed similar to the one stated in this work is used, damage detection is accomplished (but not localization or classification) by means of the covariance matrix estimate damage indicator.

This paper contributes a damage detection and localization method (the latter being treated as a classification problem) for a jacket-type WT model by using only acceleration response data. As in [17], it is assumed that the available excitation is the wind, thus the input excitation is not measurable. Hence, the contributed methodology comprises the following steps. First, a Gaussian white noise is used to simulate the wind excitation. Secondly, the data coming from the WT accelerometers are acquired. Thirdly, the raw data are pre-processed using group-reshape (to increase the amount of information contained by each observation) and column-scaling (to simplify the computation of the principal components (PCs)). Fourthly, the PCA is used as a feature selection technique as well as to reduce the dimensionality of the data and the computing time. Finally, the *k*-nearest neighbor (*k*-NN) and the quadratic Support Vector Machine (SVM) classifiers are tested. To estimate their performance, the 5-fold cross-validation technique is used to advise that the SVM has the best performance. The reliability of the proposed method is verified using different bar crack locations in a small-scale structure—an experimental testbed modeling a jacket-type WT.

The structure of the paper is as follows. Section 2 details the experimental laboratory testbed used to validate the proposed approach. Section 3 states the damage detection and classification methodology. The results are presented in Section 4. Finally, the conclusions are drawn in Section 5.

2. Experimental Testbed

The general overview of the experimental testbed is given in Figure 2 and explained as follows. The experiment starts with a white noise signal given by the function generator. This signal is amplified and passed to the inertial shaker. This is responsible for generating vibrations (similar to those produced by steady state wind on the blades) to the laboratory tower structure. The shaker is placed in the upper part of the structure, thus simulating the nacelle mass. Finally, the structure is monitored by 8 triaxial accelerometers which are connected to the data acquisition system. The next subsections describe the testbed different steps and instrumentation.

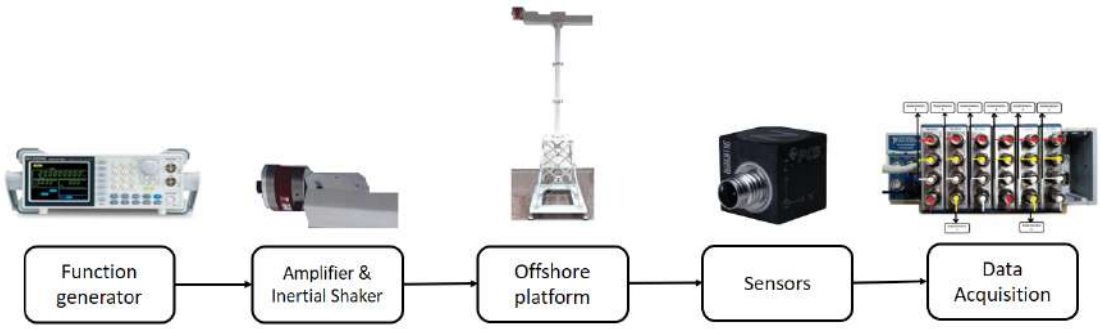


Figure 2. General overview of the experimental testbed.

2.1. Function Generator

Function generators are signal sources which provide a specifiable voltage applied over a specifiable time. In this work the GW INSTEKAF-2005 model is used. To perform the experimental

tests, the white noise signal is selected. Different wind speeds were simulated by multiplying the amplitude of the white noise signal (at the function generator) by the factors 0.5, 1, 2 and 3.

2.2. Amplifier and Shaker

When large structures need to be tested, inertial shakers provide the ideal solution. The central spigot is attached to the structure under test and the body then provides the inertial mass. In this work, the inertial shaker model GW-IV47 from Data Physics is used as well as its PA300E gain control amplifier; see Figure 3. To produce vibrations that simulate the ones obtained when the wind hits the WT blades, the white noise signal given by the function generator is amplified and this electrical signal is applied to the shaker. Thus, the vibration needed to excite the structure is created.

Figure 3. Amplifier model PA300E (left) and inertial shaker IV47 series (right) from Data Physics used in the experimental set-up.

2.3. Laboratory Tower Structure and Studied Types of Damage

The real structure used in this work is 2.7 m high and, as shown in Figure 4 (left), it has three different structural components: nacelle, tower and jacket. The top piece is a 1 m long and 0.6 m wide beam where an inertial shaker is located that simulates the nacelle mass and the environmental effects of the wind over the whole structure. The tower is formed by three tubular sections linked with bolts with a torque of 125 Nm. The jacket is a pyramidal structure formed by several steel bars of different lengths, all of them linked with bolts, with a torque of 12 Nm. The studied damage is introduced in these bars; see Figure 4 (right). In particular, the jacket has four different bar lengths, each one at different levels (depth). Level 1 is where the shortest bars are located, near the water surface. Then, greater depth leads to next levels up to level 4 where the longest bars are situated (near the see bottom). The damage will be introduced, one at a time, at the four different levels, i.e., at four different bars located at level 1, 2, 3 and 4 as illustrated in Figure 4 (right). Fatigue cracks are one of the types of damage found on offshore WT foundations. The probability of detection of a fatigue crack is low for small crack sizes. However, for larger and therefore better detectable fatigue cracks, the crack growth rate accelerates rapidly [18]. Consequently, there is only a small time window for detection and repair of this type of cracks before failure. Thus, in this work a 5 mm crack is considered located at different bars of the jacket structure, one at a time. Please note that in [16] a modal analysis and power spectral density signal processing methods were not able to detect this 5 mm crack located in the substructure using a similar laboratory tower model.

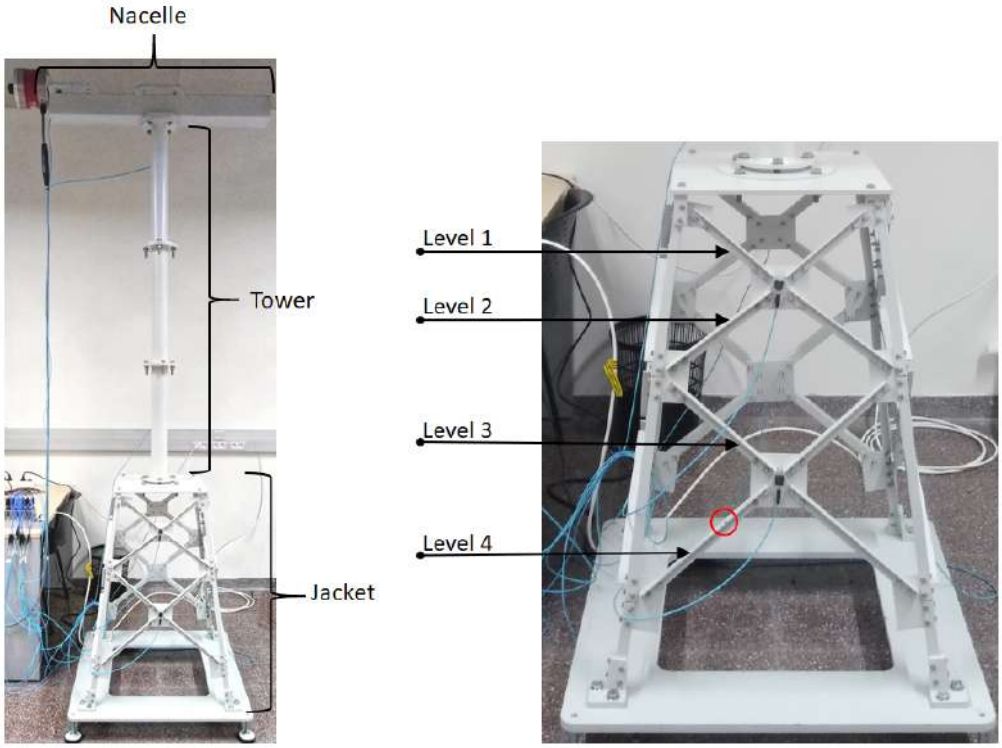


Figure 4. WT scaled offshore fixed jacket-type platform tower model used in the experimental tests (left). Bars (pointed with arrows) where the crack damage is introduced (right); note that in this picture the damage is introduced at level 4 (red circle), but it will be introduced on each level one by one.

2.4. Sensors

Eight triaxial accelerometers (model 356A17, PCB Piezotronics) (PCB® manufacturer, Depew, NY, USA) have been used strategically (placed at the tower and jacket by direct adhesive mount) to detect some anomaly in the dynamic behavior of the structure. These are high sensitivity and ceramic shear accelerometers that have a fixed voltage sensitivity, regardless of the type of cable used or its length; and its output signal is low impedance, so it can transmit over long cables in hostile environments without losing signal quality. The device is accompanied by a cable that trifurcates giving an output for each spatial component x, y, and z; see Figure 5. Hence, data from 24 sensors are acquired.

Figure 5. Triaxial accelerometers used in the testbed (PCB Piezotronics, model 356A17).

The optimal location of the sensors (see Figure 6) is determined according to the sensor elimination by modal assurance criterion (SEAMAC) ([16], Chapter 3.7, page 53). This is a sensor removal algorithm based on eliminating iteratively, one by one, the degrees of freedom that show a lower impact on MAC

matrix values. This iterative process stops when you get a default MAC matrix, high values in the diagonal terms and low values in off-diagonal terms.

Figure 6. Location of the sensors on the overall structure.

2.5. Data Acquisition System

The data acquisition system is composed by the cDAQ-9188 chassis and six NI-9234 modules from National Instruments^{TM}manufacturer (Austin, TX, USA), as shown in Figure 7. The cDAQ-9188 is a CompactDAQ Ethernet chassis, consisting of 8 input slots, and each slot can receive up to 4 different signals. The chassis is capable of controlling timing, synchronization and data transfer between the C Series I / O modules and an external server. The NI-9234 modules can measure signals from integrated electronic piezoelectric sensors (IEPE) and non-IEPE such as accelerometers (used in this work), tachometers and proximity sensors.

Figure 7. Data Acquisition System (DAQ) used in this work: cDAQ-9188 chassis and six NI-9234 modules from National Instruments.

3. Damage Detection Methodology

3.1. Data Collection and Reshape

In this work, data collection and reshape is considered with the goal of combining different response signals (measured by different sensors during multiple observations) into a single and unified view. We will present a method for data fusion, dimensionality reduction and feature extraction using a particular unfolding. We then apply a machine-learning classifier (k-NN and SVM are tested) to detect damage or structural changes in incoming collected data. It is clear that the classifiers will play a key role in the damage detection methodology. However, given the three-dimensional nature of the collected information in this paper (time, sensors, experiments), how the data are collected, arranged, scaled, transformed, and reduced may affect the overall performance of the strategy [19]. A similar problem is considered in [20], where the three-dimensional nature of the SHM data comes from location, frequency and time. In that case, tensor analysis is considered to extract the features.

One of the most widely adopted ways to deal with this kind of three-dimensional source of information is the unfolding proposed by Westerhuis et al. [21], where six alternative ways of arranging a 3-dimensional data matrix are proposed. In our case, the combination of the different response signals into a unified view will be represented by a two-dimensional matrix $\mathbf{X} = \left(x_{i,j}^{k,l}\right) \in \mathcal{M}_{(n_1+\cdots+n_E)\times(K\cdot L)}$ as follows:

$$\mathbf{X} = \left(x_{i,j}^{k,l}\right) = \left[\begin{array}{ccc:ccc:c:ccc} x_{1,1}^{1,1} & \cdots & x_{1,1}^{1,L} & x_{1,1}^{2,1} & \cdots & x_{1,1}^{2,L} & \cdots & x_{1,1}^{K,1} & \cdots & x_{1,1}^{K,L} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n_1,1}^{1,1} & \cdots & x_{n_1,1}^{1,L} & x_{n_1,1}^{2,1} & \cdots & x_{n_1,1}^{2,L} & \cdots & x_{n_1,1}^{K,1} & \cdots & x_{n_1,1}^{K,L} \\ \hdashline x_{1,2}^{1,1} & \cdots & x_{1,2}^{1,L} & x_{1,2}^{2,1} & \cdots & x_{1,2}^{2,L} & \cdots & x_{1,2}^{K,1} & \cdots & x_{1,2}^{K,L} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n_2,2}^{1,1} & \cdots & x_{n_2,2}^{1,L} & x_{n_2,2}^{2,1} & \cdots & x_{n_2,2}^{2,L} & \cdots & x_{n_1,2}^{K,1} & \cdots & x_{n_2,2}^{K,L} \\ \hdashline \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \hdashline x_{1,J}^{1,1} & \cdots & x_{1,J}^{1,L} & x_{1,J}^{2,1} & \cdots & x_{1,J}^{2,L} & \cdots & x_{1,J}^{K,1} & \cdots & x_{1,J}^{K,L} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n_J,J}^{1,1} & \cdots & x_{n_J,J}^{1,L} & x_{n_J,J}^{2,1} & \cdots & x_{n_J,J}^{2,L} & \cdots & x_{n_J,J}^{K,1} & \cdots & x_{n_J,J}^{K,L} \end{array}\right] \tag{1}$$

$$= \left[\begin{array}{c} \mathbf{X}_1 \\ \hdashline \mathbf{X}_2 \\ \hdashline \vdots \\ \hdashline \mathbf{X}_J \end{array}\right] = \left[\begin{array}{c|c|c|c} \mathbf{X}^1 & \mathbf{X}^2 & \cdots & \mathbf{X}^K \end{array}\right].$$

Matrix $\mathbf{X}$ in Equation (1) is presented for a general case, so that the proposed strategy can be easily reproduced. The two subindices i and j and the two superindices k and l are related to the experimental trial, structural state, sensor and time instant, respectively. More precisely,

- $i = 1, \ldots, n_j$ represents the $i-$th experimental trial, while n_j is the number of observations or experimental trials per structural state;
- $j = 1, \ldots, J$ is the structural state that is been measured, while J is the quantity of different structural states;
- $k = 1, \ldots, K$ indicates the sensor that is measuring, while K is the total number of sensors;

- $l = 1, \ldots, L$ identifies the time stamp, while L is the number of time stamps per experiment.

Please note that matrix $\mathbf{X}$ in Equation (1) can also be viewed as the vertical concatenation of J matrices $\mathbf{X}_j$, $j = 1, \ldots, J$, where each matrix is associated with a different structural state. Similarly, matrix $\mathbf{X}$ can also be considered to be the horizontal concatenation of K matrices $\mathbf{X}^k$, $k = 1, \ldots, K$, where each matrix is associated with a different sensor. This horizontal concatenation of matrices can also be viewed as a kind of *group-reshaping*, where we measure a sensor during $n_j \cdot L$ time instants (in the $j-$th structural state), and we finally arrange these $n_j \cdot L$ time instants in a $n_j \times L$ matrix $\mathbf{X}^j$. It is noteworthy that by this reshape—that is the key of the selected unfolding proposed by Westerhuis et al. [21]—it is increased the amount of information contained by each observation (row). Moreover, this choice facilitates the study of the variability among samples, because we compile the information related to the sensor measurements and their variations over time.

3.2. Column-Scaling and Principal Component Analysis (PCA)

The raw data in matrix $\mathbf{X}$ in Equation (1) is scaled for two main reasons: first, to process data that come from different sensors and second, to simplify the computations of the data transformation using PCA [22–25]. In this work, column-wise scaling (CS) is used. More precisely, each column vector in matrix $\mathbf{X}$ is normalized by subtracting the mean of all the elements in the column and by dividing by the standard deviation of the same set of data. Thus, each column of the new scaled matrix, $\check{\mathbf{X}}$, has a mean of zero and a standard deviation of one.

Recall that before using a classifier, the data must be processed (transformed and reduced) to obtain the most suitable features. In this work, multiway PCA is selected to accomplish this objective. On one hand, the transformation is calculated as a matrix-to-matrix multiplication

$$\mathbf{T} = \check{\mathbf{X}}\mathbf{P},$$

where $\mathbf{P}$ is the matrix that contains, written as columns, the principal components of matrix $\check{\mathbf{X}}$. $\mathbf{T}$ is a $(n_1 + \cdots + n_J) \times (K \cdot L)$ matrix. On the other hand, the dimensionality reduction is performed through the reduced PCA model $\mathbf{P}_\ell$ that contains, written as columns, the first ℓ principal components. More precisely, $\mathbf{T}_\ell$ is the projection of the scaled matrix $\check{\mathbf{X}}$ into the vectorial space spanned by the reduced PCA model through the matrix-to-matrix multiplication

$$\mathbf{T}_\ell = \check{\mathbf{X}}\mathbf{P}_\ell.$$

Since we have applied column-scaling, the trace of the variance-covariance matrix is equal to $K \cdot L$. This means that the first ℓ principal components retain a proportion of variance given by

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_\ell}{KL},$$

where λ_i are the eigenvalues associated with the eigenvectors (principal components) of the variance-covariance matrix, in decreasing order.

3.3. Machine-Learning Classifiers

Multi-class classification algorithms are used to categorize the different structural states. In particular, the supervised learning algorithms *k*-NN and SVM are used and its performance compared through different indicators. The *k*-NN classifier is an instance-based learner, i.e., it stores the training data in the memory instead of constructing a model and compares the new test data to closest saved instances to perform the prediction. On the other hand, the SVM classifier constructs a model that is supposed to generalize well. The classifiers are succinctly introduced in the following subsections.

3.3.1. *k*-Nearest Neighbor (*k*-NN)

The *k*-NN algorithm [26,27] stores the training dataset and to make a prediction computes the k nearest neighbors to the observation to be categorized and assigns the category of the majority. It only requires tuning one parameter: the number of neighbors, k. The main drawback is that as it does not train a model, the algorithm spends more time during the prediction. Please note that we use the same symbol k as one of the superindices in the generic element of matrix $\mathbf{X}$ in Equation (1), but with a different meaning. We will keep the notation *k*-NN because we think that there is no possibility of ambiguity.

3.3.2. Support Vector Machine (SVM)

It is not the purpose of this paper to give a detailed explanation of the SVM classifier. For the interested reader, an excellent detailed review is given in reference [28]. However, to hand over the background and motivation for the proposed methodology, a summary of the method is given. This recap is based on reference [29].

SVM classification is primarily a binary classification technique. Suppose a training set $\{(x_i, y_i)\}_{i=1}^{N}$ with d-dimensional data $x_i \in \mathbb{R}^d$ and their complementary binary label $y_i \in \{-1, +1\}$. Figure 8 shows a two-dimensional example of these type of data where one class is labeled as $(+)$ and the other one as $(-)$. The objective of the SVM is to find the hyperplane with the widest margin to separate both classes; see Figure 8. Conventionally, the hyperplane is given by

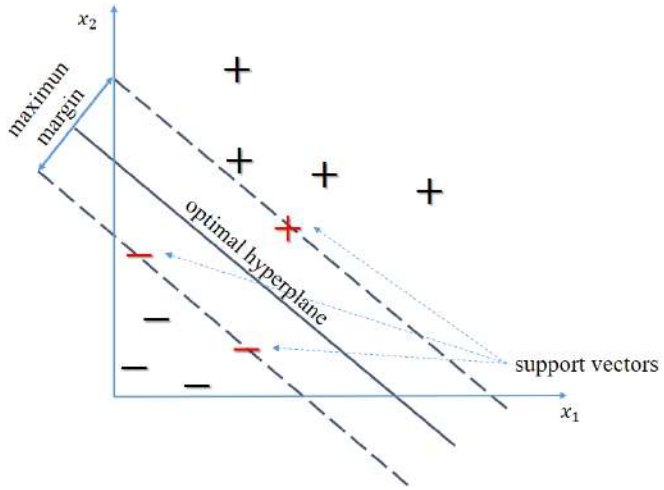


Figure 8. Linear support vector machine (SVM) in a two-dimensional example.

$$h(x) = \omega^T x + b, \tag{2}$$

where ω is the weight vector and b is the bias term. Among all the possible descriptions of the hyperplane, usually the so-called canonical hyperplane is used that satisfies

$$\omega^T x_+^{sv} + b = 1, \tag{3}$$
$$\omega^T x_-^{sv} + b = -1, \tag{4}$$

where x_+^{sv} and x_-^{sv} illustrate the (+) and (−) training samples closest to the hyperplane (the so-called support vectors); see Figure 8. Maximizing the margin is equivalent to the following minimization problem

$$\min_{\omega,b} \frac{1}{2}||\omega||^2 \quad \text{subject to} \quad h(x_i)y_i \geq 1,\ i = 1,\ldots,N. \tag{5}$$

When the data are not separable by a hyperplane, SVM can use a soft margin, meaning to find a hyperplane that separates many but not all the data. Therefore, the problem is generalized by introducing slack variables, ε_i, and a penalty parameter, C. Then, the general formulation, for the linear kernel, is,

$$\min_{\omega,b,\varepsilon_i} \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{N}\varepsilon_i \quad \text{subject to} \quad \begin{cases} h(x_i)y_i \geq 1-\varepsilon_i,\ i = 1,\ldots,N; \\ \varepsilon_i \geq 0,\ i = 1,\ldots,N. \end{cases} \tag{6}$$

In this case, using Lagrange multipliers, the problem reads

$$\min_{\alpha_i} \left[\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^T x_j\right] \quad \text{subject to} \quad \begin{cases} \sum_{i=1}^{N}\alpha_i y_i = 0; \\ 0 \leq \alpha_i \leq C,\ i = 1,\ldots,N. \end{cases} \tag{7}$$

The final set of constraints demonstrate why the penalty parameter C is called a box constraint, as it retains the values of the Lagrange multipliers in a bounded region.

From Equation (7), it is obvious that optimization depends only on dot products of pairs of samples. Plus, the decision rule depends only on the dot product. Thus, when the classification problem does not have a simple separating hyperplane, even using a soft margin, a transformation to another space can be used, $\phi(\cdot)$. Indeed, the transformation itself is not needed, but just the dot product (so-called kernel function),

$$K(x_i, x_j) = \phi(x_i)\phi(x_j). \tag{8}$$

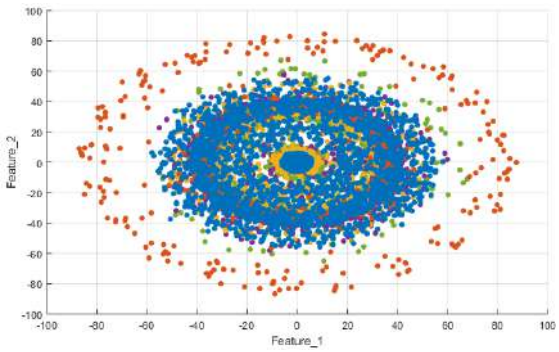
The kernel function permits the computation of the inner product between the mapped vectors without expressly calculating the mapping. This is known as the *kernel trick* [30]. Different kernels can be used, namely polynomial, hyperbolic tangent, or Gaussian radial basis function. Here, to give an insight and understand why the quadratic kernel is selected in this work, some scatter plots are shown in Figure 9. It can be seen that these plots reveal a quadratic relationship and, particularly, the first versus the second feature scatter plot exposes a concentric circles shape of the data set. Therefore, the quadratic SVM classifier is adopted, i.e., the following polynomial kernel is used

$$K(x_i, x_j) = \left(1 + \frac{1}{\rho^2}x_i^{\mathrm{T}}x_j\right)^2 \tag{9}$$

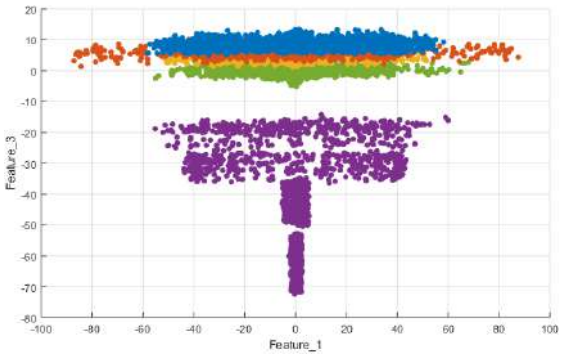
where ρ is the so-called kernel-scale parameter; and x_i and x_j denote here different observations of our data set.

As said previously, SVM classification is a binary classification technique, which must be adapted to cope with multi-classification problems. Two of the most common methods to enable this adaptation include the one-vs-one and one-vs-all approaches. The one-vs-all technique [31] represents the earliest and most common SVM multi-class approach and comprises the division of an N class dataset into N two-class cases and it designates the class which classifies the test with greatest margin. The one-vs-one strategy [32] comprises constructing a machine for each pair of classes, thus resulting in $N(N-1)/2$ machines. When this approach is applied to a test point, each classification gives one vote to the winning class and the point is labeled with the class with most votes. The one-vs-one strategy is more

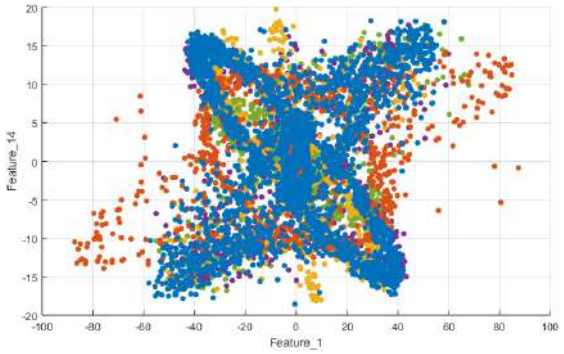
computationally demanding since the results of more SVM pairs ought to be computed. In this work, the one-vs-all approach is used.



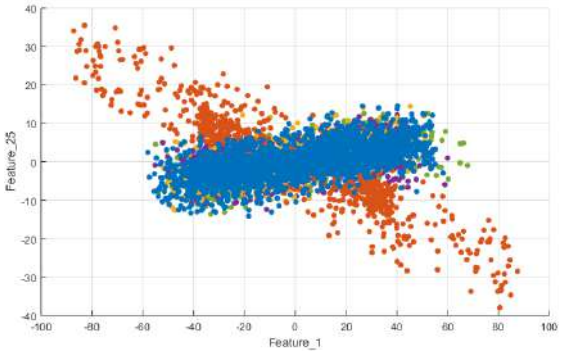
(a) Scatter plots of the first versus the second principal component.



(b) Scatter plots of the first versus the third principal component.



(c) Scatter plots of the first versus the fourteenth principal component.



(d) Scatter plots of the first versus the twenty-fifth principal component.

Figure 9. Blue dots represent healthy samples, orange dots represent samples of damage at level 1, yellow dots represent samples of damage at level 2, purple dots represent samples of damage at level 3 and green dots represent samples of damage at level 4.

3.4. κ-Fold Cross-Validation

Cross-validation is a technique used to evaluate the results and ensure that they are independent of the partition between training and test data. It comes from the improvement of the holdout method that consists of dividing the sample data into two complementary sets, performing the training in the first subset and validating with the other subset. By the holdout method, the evaluation can depend to a large extent on how the partition between training and test data is performed. Due to this deficiency the concept of cross-validation appears. In the cross-validation of κ iterations, or κ-fold cross-validation, the data is divided into κ subsets. One of the subsets is used as test data and the rest $(\kappa - 1)$ as training data. This process is repeated κ times, and at each iteration a different subset is used as test data and the others as training data. Finally, the arithmetic mean of the results of each iteration is performed to obtain a single result. This method is a more accurate estimate of model prediction performance since it evaluates from κ different combinations of training and test data. In this paper, 5-fold cross-validation is used to estimate the performance of the proposed strategy.

4. Results

4.1. Experimental Set-Up

As said in Section 2.3, we have considered $J = 5$ different structural states, the healthy WT and the structure with damage located at four different jacket levels. Moreover, a total of $n_1 + n_2 + n_3 + n_4 + n_5 = 11620$ experimental tests are conducted, which includes the four amplitudes that represent the different wind speed regions (multiplying the amplitude of the white noise signal by the factors 0.5, 1, 2 and 3). In particular:

(i) 1245 tests with the original healthy bar for each amplitude, i.e., $n_1 = 1245 \cdot 4 = 4980$ tests.
(ii) 415 tests with damage located at level 1 for each amplitude, i.e., $n_2 = 415 \cdot 4 = 1660$ tests.
(iii) 415 tests with damage located at level 2 for each amplitude, i.e., $n_3 = 415 \cdot 4 = 1660$ tests.
(iv) 415 tests with damage located at level 3 for each amplitude, i.e., $n_4 = 415 \cdot 4 = 1660$ tests.
(v) 415 tests with damage located at level 4 for each amplitude, i.e., $n_5 = 415 \cdot 4 = 1660$ tests.

For each experimental test, we measure $K = 24$ sensors (see Section 2.4), during $L = 199$ time instants. Since the accelerometers measure at a sampling frequency of about 275 Hz, the time step is $\Delta = 0.003633$ seconds, which represents a time window for each experimental test of $\frac{199}{275\,\text{Hz}} = 0.7236$ seconds. Please note that this sampling frequency is feasible in an offshore environment using accelerometers; see [8].

Therefore, with the raw matrix $\mathbf{X}$ as in Equation (1) where

$$\begin{aligned} n_1 &= 4980 \\ n_2 &= 1660 \\ n_3 &= 1660 \\ n_4 &= 1660 \\ n_5 &= 1660 \\ J &= 5 \\ L &= 199 \\ K &= 24 \end{aligned}$$

we apply the column-scaling and we compute the PCA model for the data transformation as detailed in Section 3.2. The extent of the data reduction can be measured as the rate of the number of principal components that retains a predetermined variance with respect to the number of columns in matrix $\mathbf{X}$ in Equation (1). For instance:

(i) if we retain 85% of the variance, the first 443 principal components are needed out of $K \cdot L = 4776$ columns. This represents a data reduction (leading to reduced memory requirements) of 90.72%.
(ii) if we retain 90% of the variance, the first 887 principal components are needed. The reduction in this case is 81.43%.
(iii) if we retain 95% of the variance, the first 1770 principal components are needed. This represents a still significant reduction of 62.94%.

Data reduction is of paramount importance in both the training time and the prediction speed. The results in this section are computed and assessed using MATLAB©.

4.2. Metrics for Evaluating Classification Models

In classification problems, training data is used to build a model and thus predict the class label for a new sample. To know if the model that we have trained has the best performance according to the problem presented, it is important to evaluate the classification model through metrics such as accuracy, precision, sensitivity and specificity that can be generated from a confusion matrix such as the one shown in Table 1. Regularly, these metrics evaluate binary classification problems.

Table 1. Binary confusion matrix.

		Predicted Class	
		Positive	**Negative**
Actual class	**Positive**	True positive (TP)	False negative (FN)
	Negative	False positive (FP)	True negative (TN)

From this binary confusion matrix:

- True positive (TP) is the number of positive cases that were correctly identified.
- False positive (FP) is the quantity of negative cases that were incorrectly classified as positive.
- True negative (TN) is defined as the sum of negative cases that were correctly classified.
- False negative (FN) is the total of positive cases that were incorrectly classified as negative.

The meaning of *positive* and *negative* may vary from one application to another. For instance, in [23,33], where a 5 MW high-fidelity WT model is considered for fault detection, if a new sample is categorized as positive it means that the current structure is classified as healthy. Otherwise, some kind of fault is present in the WT. In the present work, we follow the same classification with respect to positive and negative.

Table 2 shows the most common metrics for choosing the best solution to a binary classification problem.

Table 2. Metrics for evaluating binary classification models.

Metric	**Formula**	**Description**
Accuracy	$\text{acc} = \frac{\text{TP+TN}}{\text{TP+FP+FN+TN}}$	It is the number of correct predictions made by the model according to the total number of records. The best accuracy is 100%, which indicates that all predictions are correct. Accuracy alone does not tell the full story when working with a class-imbalanced data set.
Precision	$\text{ppv} = \frac{\text{TP}}{\text{TP+FP}}$	This parameter evaluates the data by its performance of *positive* predictions, in other words, it is the proportion of positive cases that have been correctly predicted.
Sensibility/Recall	$\text{tpr} = \frac{\text{TP}}{\text{TP+FN}}$	This parameter calculates the fraction of the positive cases that our model has been able to identify as positive (true positive).
F1-Score	$\text{F1} = 2\frac{\text{ppv} \times \text{tpr}}{\text{ppv} + \text{tpr}}$	It is defined as the harmonic mean of precision and sensitivity. A $F1$ score reaches its best value at 1 (accuracy and perfect sensitivity) and worse at 0.
Specificity	$\text{tnr} = \frac{\text{TN}}{\text{TN+FP}}$	The specificity or true negative rate is calculated as the proportion of correct negative predictions divided by the total number of cases that are classified as negative. Specificity is the exact opposite of sensitivity, the proportion of negative cases predicted correctly.

As shown in [34], these metrics are easy to calculate and applicable to binary and multi-class classification problems. When the classification problem is multi-class, according to [35,36] the result is the average obtained by adding the result of each class and dividing over the total number of classes. The formulas for calculating these metrics in a multi-class classification model are shown in Table 3.

Table 3. Metrics for evaluating multi-class classification models, where j refers to the individual class and J is the total number of classes.

Metric	Formula
Average Accuracy ($\overline{\text{acc}}$)	$\frac{1}{J}\sum_{j=1}^{J}\frac{\text{TP}_j+\text{TN}_j}{\text{TP}_j+\text{FP}_j+\text{FN}_j+\text{TN}_j}$
Average Precision ($\overline{\text{ppv}}$)	$\frac{1}{J}\sum_{j=1}^{J}\frac{\text{TP}_j}{\text{TP}_j+\text{FP}_j}$
Average Sensibility/Recall ($\overline{\text{tpr}}$)	$\frac{1}{J}\sum_{j=1}^{J}\frac{\text{TP}_j}{\text{TP}_j+\text{FN}_j}$
Average F1-Score ($\overline{\text{F1}}$)	$2\frac{\overline{\text{ppv}}\times\overline{\text{tpr}}}{\overline{\text{ppv}}+\overline{\text{tpr}}}$
Average Specificity ($\overline{\text{tnr}}$)	$\frac{1}{J}\sum_{j=1}^{J}\frac{\text{TN}_j}{\text{TN}_j+\text{FP}_j}$

In a multi-class confusion matrix, the classification results TP, TN, FP and FN can also be considered for each class, as shown in ([35], page 71). Table 4 summarizes a confusion matrix with 5 different classes. When we focus, for instance, on class B (that is, the second class), we can identify four regions:

- the green region is related to the true positive (TP_2);
- the magenta region is related to the false positive (FP_2). More precisely, FP_2 is the sum of the elements in the second column but BB, i.e., $\text{FP}_2 = \text{AB} + \text{CB} + \text{DB} + \text{EB}$;
- the orange region is related to the false negative (FN_2). More precisely, FN_2 is the sum of the elements in the second row but BB, i.e., $\text{FN}_2 = \text{BA} + \text{BC} + \text{BD} + \text{BE}$; and finally,
- the cyan region is related to the true negative (TN_2). More precisely, TN_2 is the sum of all the elements of the confusion matrix but the ones in the second row and the second column.

Table 4. Multi-class confusion matrix. In this case, the confusion matrix has five classes.

		Predicted Class				
		Class A	**Class B**	**Class C**	**Class D**	**Class E**
Actual class	**Class A**	AA	AB	AC	AD	AE
	Class B	BA	BB	BC	BD	BE
	Class C	CA	CB	CC	CD	CE
	Class D	DA	DB	DC	DD	DE
	Class E	EA	EB	EC	ED	EE

The classification results $\text{TP}_j, \text{FP}_j, \text{FN}_j$ and TN_j for $j = 1, \ldots, 5$ that correspond to classes A, B, C, D and E in Table 4, respectively, are computed similarly.

Recall that this paper shows a multiple classification problem that collects data grouped in five different classes: the original healthy bar, damage located at jacket level 1, damage located at jacket level 2, damage located at jacket level 3 and damage located at jacket level 4.

4.3. Results of k-NN Classification Method

First, the *k*-NN classifier is tested. The indicators introduced in Section 4.2 are employed to tune the value of the parameter k, which is the number of neighbors to be used, and also decide the best variance to be adopted when applying PCA.

Table 5 shows the results obtained when working with 85%, 90% and 95% of the variance, and also varying the number of neighbors, k. The classifiers with the best performance are highlighted in boldface font. It can be observed that the best classifiers from 90% and 95% of the variance, respectively, have both similar indicators. In this case, working only with 90% of variance is preferred as it will reduce the computational memory requirement. In terms of time, Table 6 shows the training time and prediction speed for each of the classifiers with the best performance. It can be inferred that the classifier using 90% of the variance and $k = 200$ neighbors has the best performance. In particular, for this classifier, Figure 10 graphically shows its performance indicators (pink line) and Table 7 represents its confusion matrix. Regarding Figure 10 it is observed that initially, increasing the number of neighbors leads to better indicators, however when k is greater than 200 the performance degrades for all the indicators. With regard to the confusion matrix (Table 7), each row represents the instances in a true class while each column represents the instances in a predicted class (by the classifier). In particular, the first row (and first column) is labeled as 0 and corresponds to the healthy bar. The next labels (for rows and columns) 1, 2, 3, and 4 correspond to bars damage in the corresponding levels of the jacket structure. From this confusion matrix, it can be derived all the aforementioned indicators. In particular, it is noteworthy that an average accuracy of 95%, an average precision (proportion of healthy cases predicted correctly) of 95%, and an average specificity (proportion of faulty cases predicted correctly) of 99.5% are obtained.

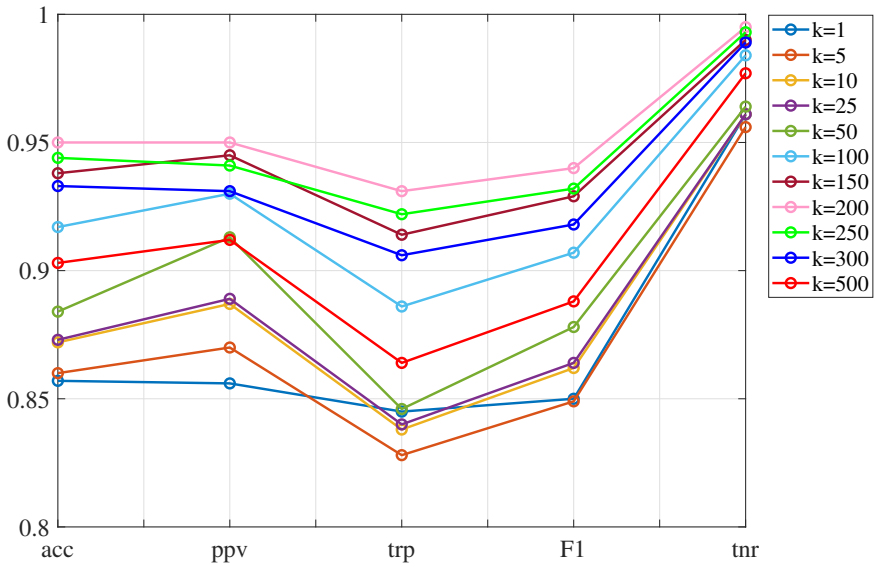


Figure 10. Indicators corresponding to the *k*-NN method using 90% of variance. The case $k = 200$, represented by the pink line, shows the best performance in each of the evaluation indicators.

Table 5. Evaluation indicators for the *k*-NN method using different percentages of variance and different number of nearest neighbors (*k*). The best classifiers for each variance are highlighted using a boldface font.

Variance	Number of PCs	Neighbors k	Accuracy $\overline{acc}$	Precision $\overline{ppv}$	Recall $\overline{tpr}$	F1 score $\overline{F1}$	Specificity $\overline{tnr}$
85%	443	1	0.857	0.860	0.846	0.853	0.950
		5	0.856	0.869	0.82	0.844	0.953
		10	0.867	0.891	0.827	0.858	0.956
		25	0.868	0.898	0.828	0.862	0.955
		50	0.874	0.912	0.831	0.87	0.958
		100	0.911	0.928	0.879	0.903	0.981
		150	0.933	0.942	0.908	0.925	0.988
		200	**0.946**	**0.947**	**0.924**	**0.936**	**0.993**
		250	0.940	0.938	0.917	0.927	0.991
		300	0.930	0.929	0.903	0.915	0.988
		500	0.899	0.909	0.859	0.884	0.976
90%	887	1	0.857	0.856	0.845	0.85	0.961
		5	0.860	0.870	0.828	0.849	0.956
		10	0.872	0.887	0.838	0.862	0.961
		25	0.873	0.889	0.840	0.864	0.961
		50	0.884	0.913	0.846	0.878	0.964
		100	0.917	0.930	0.886	0.907	0.984
		150	0.938	0.945	0.914	0.929	0.990
		200	**0.950**	**0.950**	**0.931**	**0.940**	**0.995**
		250	0.944	0.941	0.922	0.932	0.993
		300	0.933	0.931	0.906	0.918	0.989
		500	0.903	0.912	0.864	0.888	0.977
95%	1770	1	0.854	0.853	0.842	0.847	0.960
		5	0.858	0.865	0.829	0.847	0.956
		10	0.872	0.885	0.841	0.862	0.962
		25	0.873	0.888	0.841	0.864	0.962
		50	0.885	0.911	0.848	0.879	0.965
		100	0.918	0.930	0.888	0.908	0.985
		150	0.938	0.945	0.915	0.930	0.990
		200	**0.950**	**0.950**	**0.930**	**0.940**	**0.995**
		250	0.945	0.942	0.923	0.932	0.993
		300	0.934	0.931	0.908	0.919	0.989
		500	0.904	0.913	0.866	0.889	0.978

Table 6. Prediction speed and training time results for the best performance classifiers of each variance (85%, 90% and 95%) using the *k*-NN classification method on a 3GHz Intel Core i7, 16 GB RAM computer. The best classifier in terms of accuracy is highlighted using a boldface font.

Variance	Accuracy	Prediction Speed (obs/s)	Training Time (s)
85%	94.6%	210	329
90%	**95.0%**	**110**	**638**
95%	95.0%	55	1251

Table 7. Confusion matrix for the *k*-NN algorithm with k =200 neighbors. The obtained accuracy is 95%. Label 0 corresponds to the healthy state, and labels 1, 2, 3 and 4 correspond to the damage state located at the corresponding level. In this matrix, each row represents the instances in a true class while each column represents the instances in a predicted class. An empty blank square means 0%.

	0	1	2	3	4
0	> 99%		< 1%		
1	< 1%	98%	2%		
2	7%		93%		< 1%
3				75%	25%
4	1%		< 1%		99%

4.4. Results of SVM Classification Method

In this Section the results of the SVM classification method are presented. As stated in Section 3.3.2, since the scatter plots in Figure 9 reveal a quadratic relationship (particularly the first versus the second principal components), the quadratic SVM classifier is adopted. Therefore, in this case, the hyper-parameters are the box constraint C, see (6), and the kernel scale ρ, see (9), that are tuned using the indicators detailed in Section 4.2.

Table 8 summarizes the results obtained when working with 85%, 90% and 95% of the variance. Since the problem we are dealing with seems a separable problem, no changes were found in the performance of the indicators when considering different values of the box constraint C. Therefore, $C = 1$ is considered for the rest of the analysis. The cases that present the best results have been highlighted in boldface font. It can be observed that the best classifiers from 85%, 90% and 95% of the variance reach a level of accuracy, precision, recall, F1 score and specificity around 99.8% and 100%. However, the best cases are the ones that consider 85% of the variance and kernel scales $\rho = 90$ and $\rho = 100$. In these cases, the results are almost ideal reaching a specificity value of 100% and the rest of the indicators 99.9%. With respect to the time, Table 9 presents the training time and the prediction speed for each one of the classifiers with the best performance. According to this table, it is observed that the classifier with 85% of the variance and a kernel scale ρ=90 has a higher prediction speed (in terms of observations per second) and a shorter training time. Therefore, this classifier has the best performance. It may seem incongruous that a case with fewer principal components (85%, 443 PCs) behaves better than cases with more principal components (90%, 887 PCs; or even 95%, 1770). However, this can occur, since the last principal components usually collect the noise present in the measurements.

Figure 11 graphically shows the magnitude of the indicators in Table 3 with respect to the kernel scale ρ for the case with 85% of the variance. It can be seen that increasing the kernel scale from $\rho = 5$

(solid light blue) onwards improves the overall performance of the classification method up to a certain limit. The performance degradation appears for values greater than $\rho = 100$.

The confusion matrix for the best performing classifier (85% variance and kernel scale $\rho = 90$) is represented in Table 10. From this confusion matrix and according to the indicators of evaluation, we obtain an average accuracy of 99.9%, an average precision of 99.9% and an average specificity of 100%.

Table 8. Evaluation indicators for the SVM model using different percentages of variance and different values for ρ (kernel scale). The best values for each indicator are highlighted using a boldface font.

Variance/ Number of Components	Box Constraint C	Kernel Scale ρ	Accuracy $\overline{acc}$	Precision $\overline{ppv}$	Recall $\overline{tpr}$	F1 Score $\overline{F1}$	Specificity $\overline{tnr}$
85%/443	1	5	0.987	0.993	0.983	0.988	0.995
		20	0.994	0.991	0.991	0.991	**1.000**
		30	0.995	0.994	0.994	0.994	**1.000**
		40	0.996	0.995	0.995	0.995	**1.000**
		50	0.997	0.996	0.996	0.996	**1.000**
		60	0.998	0.997	0.997	0.997	**1.000**
		70	0.998	0.998	0.998	0.998	**1.000**
		80	**0.999**	0.998	0.998	0.998	**1.000**
		90	**0.999**	**0.999**	**0.999**	**0.999**	**1.000**
		100	**0.999**	**0.999**	**0.999**	**0.999**	**1.000**
		150	0.998	0.998	0.998	0.998	**1.000**
		200	0.997	0.996	0.997	0.997	0.999
		300	0.882	0.926	0.835	0.878	0.954
90%/887	1	5	0.987	0.993	0.982	0.988	0.995
		20	0.991	0.988	0.988	0.988	**1.000**
		30	0.995	0.993	0.993	0.993	**1.000**
		40	0.996	0.995	0.995	0.995	**1.000**
		50	0.997	0.997	0.997	0.997	**1.000**
		60	**0.998**	0.997	0.997	0.997	**1.000**
		70	**0.998**	0.997	0.997	0.997	**1.000**
		80	**0.998**	0.997	0.997	0.997	**1.000**
		90	**0.998**	**0.998**	**0.998**	**0.998**	**1.000**
		100	**0.998**	**0.998**	**0.998**	**0.998**	**1.000**
		150	**0.998**	**0.998**	**0.998**	**0.998**	0.999
		200	0.997	0.996	0.996	0.996	0.999
		300	0.883	0.927	0.837	0.880	0.955
95%/1770	1	5	0.986	0.993	0.981	0.987	0.994
		20	0.990	0.987	0.986	0.986	**1.000**
		30	0.994	0.992	0.992	0.992	**1.000**
		40	0.996	0.994	0.994	0.994	**1.000**
		50	0.997	0.996	0.996	0.996	**1.000**
		60	0.997	0.997	0.997	0.997	**1.000**
		70	**0.998**	0.997	0.997	0.997	**1.000**
		80	**0.998**	0.997	**0.998**	**0.998**	**1.000**
		90	**0.998**	**0.998**	**0.998**	**0.998**	**1.000**
		100	**0.998**	0.997	**0.998**	**0.998**	**1.000**
		150	**0.998**	0.997	**0.998**	0.997	0.999
		200	0.996	0.996	0.996	0.996	0.999
		300	0.997	0.997	0.997	0.997	0.999

Table 9. Prediction speed and training time results for the best performance cases of each variance (85%, 90% and 95%) using the SVM classification method. The best classifier in terms of accuracy and computational cost is highlighted using a boldface font.

Variance	Kernel scale ρ	Accuracy	Prediction speed (obs/sec)	Training time (sec)
85%	**90**	**0.999**	**1700**	**76**
	100	0.999	1600	79
90%	90	0.998	320	256
	100	0.998	580	364
95%	90	0.998	100	676

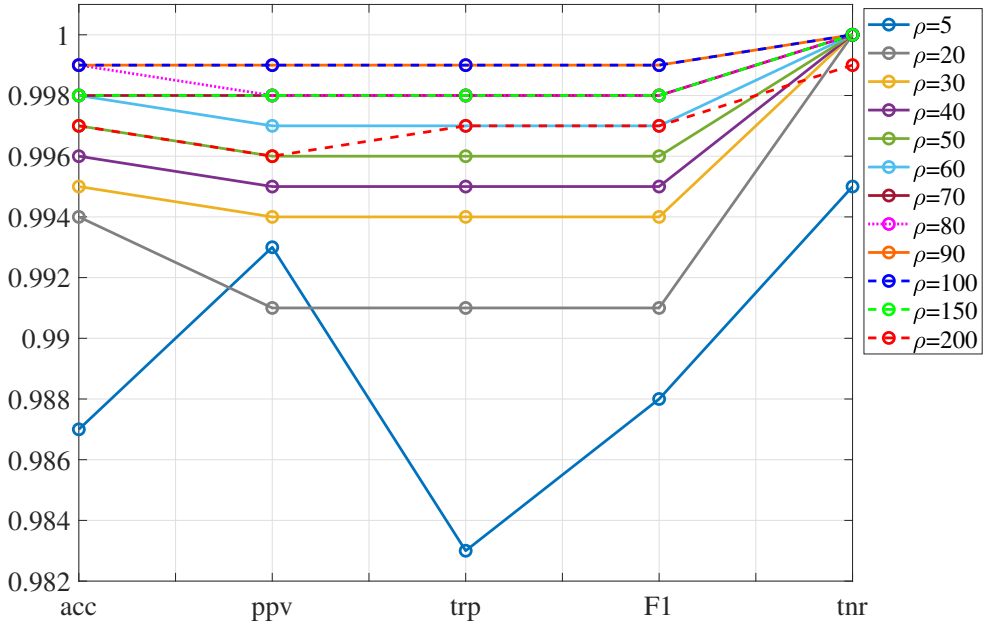


Figure 11. Indicators to evaluate the SVM classification model using 85% of variance. The cases with the best performance are $\rho = 90$ and $\rho = 100$ that are represented by the overlapped orange (solid) line and blue (dotted) line, respectively. The graph omits $\rho = 300$ since the results obtained are much worse and would be out of the y-axis scale used in this graph.

Table 10. Confusion matrix of the SVM model with $\rho = 90$ and $C = 1$, we get an accuracy of 99.9%. Label 0 corresponds to healthy, label 1 corresponds to level 1, label 2 corresponds to level 2, label 3 level 3 and label 4 corresponds to level 4. In this matrix, each row represents the instances in a true class while each column represents the instances in a predicted class. An empty blank square means 0%.

	0	1	2	3	4
0	$> 99\%$	$< 1\%$	$< 1\%$		
1	$< 1\%$	99%	$< 1\%$		
2		$< 1\%$	99%		
3				$> 99\%$	$< 1\%$
4					100%

5. Conclusions

In this work, a methodology has been stated for damage detection and localization on a laboratory-scale WT with jacket-type foundation. In particular, a crack damage is studied in four different locations of the jacket foundation. The main conclusions stressed from this work are the following:

(i) A vibration-response-only methodology has been conceived and a satisfactory experimental proof of concept has been conducted. However, future work is needed to validate the technology in a more realistic environment that takes into account the varying environmental and operational conditions.

(ii) The contribution of this work resides in how three-dimensional data (coming from different time, sensors, and experiments) is collected, arranged, scaled, transformed, and dimension reduced following a general framework stated in Sections 3.1 and 3.2, and afterwards particularized for the specific application that concerns us in Section 4.1.

(iii) The damage detection and localization methodology with the quadratic SVM classifier, kernel scale $\rho = 90$, box constraint $C = 1$, and 443 principal components (85% of variance kept) has a very close to ideal performance, achieving in all indicators a result equal o higher to 99.99% with a very fast prediction speed (1700 obs/sec) and short training time (76 sec).

Finally, it is important to note that environmental and operational conditions (EOC) play an important role when dealing with long term monitoring, because they can complicate damage detection. Large variations in EOCs make EOC monitoring almost as important as structural monitoring itself. Therefore, its influence should be compensated. Several methods for EOC compensation for WTs have been developed to make SHM possible. For example, in [37] affinity propagation clustering is used to delineate data into WT groups of similar EOC. In [38] covariance-driven stochastic subspace identification is used. Finally, in [39,40] fuzzy classification techniques are used for EOC compensation. However, as noted previously, this work is an experimental proof of concept and EOC compensation is left as future work using pattern recognition techniques in a more realistic environment.

Author Contributions: All authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially funded by the Spanish Agencia Estatal de Investigación (AEI) - Ministerio de Economía, Industria y Competitividad (MINECO), and the Fondo Europeo de Desarrollo Regional (FEDER) through the research project DPI2017-82930-C2-1-R; and by the Generalitat de Catalunya through the research project 2017 SGR 388. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

Acknowledgments: We thank the two anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

CS	column-scaling
FN	false negative
FP	false positive
k-NN	*k* nearest neighbors
IEPE	integrated electronic piezoelectric sensors
MAC	modal assurance criterion
PC	principal component
PCA	principal component analysis
SHM	structural health monitoring
SVM	support vector machine
TN	true negative
TP	true positive
WT	wind turbine

References

1. Klijnstra, J.; Zhang, X.; van der Putten, S.; Röckmann, C. *Aquaculture Perspective of Multi-Use Sites in the Open Ocean*; Technical Risks of Offshore Structures; Springer: Cham, Switzerland, 2017; pp. 115–127.
2. Fritzen, C.P. *Structural Health Monitoring*; Vibration-Based Techniques for Structural Health Monitoring; Wiley: Hoboken, NJ, USA, 2006; pp. 45–224.
3. Fassois, S.D.; Sakellariou, J.S. Time-series methods for fault detection and identification in vibrating structures. *Philos. Trans. R. Soc. A* **2006**, *365*, 411–448. [CrossRef] [PubMed]
4. Goyal, D.; Pabla, B. The vibration monitoring methods and signal processing techniques for structural health monitoring: a review. *Arch. Comput. Meth. Eng.* **2016**, *23*, 585–594. [CrossRef]
5. Vamvoudakis-Stefanou, K.J.; Sakellariou, J.S.; Fassois, S.D. Output-only statistical time series methods for structural health monitoring: A comparative study. In Proceedings of the 7th European Workshop on Structural Health Monitoring, Nantes, France, 8–11 July 2014.
6. Liu, W.; Tang, B.; Han, J.; Lu, X.; Hu, N.; He, Z. The structure healthy condition monitoring and fault diagnosis methods in wind turbines: A review. *Renew. Sustain. Energy Rev.* **2015**, *44*, 466–472. [CrossRef]
7. Martinez-Luengo, M.; Kolios, A.; Wang, L. Structural health monitoring of offshore wind turbines: A review through the Statistical Pattern Recognition Paradigm. *Renew. Sustain. Energy Rev.* **2016**, *64*, 91–105. [CrossRef]
8. Lian, J.; Cai, O.; Dong, X.; Jiang, Q.; Zhao, Y. Health monitoring and safety evaluation of the offshore wind turbine structure: a review and discussion of future development. *Sustainability* **2019**, *11*, 494. [CrossRef]
9. Mieloszyk, M.; Ostachowicz, W. An application of Structural Health Monitoring system based on FBG sensors to offshore wind turbine support structure model. *Mar. Struct.* **2017**, *51*, 65–86. [CrossRef]
10. Fritzen, C.P.; Kraemer, P.; Klinkov, M. *Structural Dynamics*; An Integrated SHM Approach for Offshore Wind Energy Plants; Springer: New York, NY, USA, 2011; pp. 727–740.
11. Schröder, K.; Gebhardt, C.; Rolfes, R. Damage Localization at Wind Turbine Support Structures Using Sequential Quadratic Programming for Model Updating. In Proceedings of the 8th European Workshop on Structural Health Monitoring, Bilbao, Spain, 5–8 July 2016.

12. Weijtjens, W.; Verbelen, T.; De Sitter, G.; Devriendt, C. Foundation structural health monitoring of an offshore wind turbine—A full-scale case study. *Struct. Health Monit.* **2016**, *15*, 389–402. [CrossRef]
13. Elshafey, A.A.; Haddara, M.R.; Marzouk, H. Damage detection in offshore structures using neural networks. *Mar. Struct.* **2010**, *23*, 131–145. [CrossRef]
14. Mojtahedi, A.; Yaghin, M.L.; Hassanzadeh, Y.; Ettefagh, M.; Aminfar, M.; Aghdam, A. Developing a robust SHM method for offshore jacket platform using model updating and fuzzy logic system. *Appl. Ocean Res.* **2011**, *33*, 398–411. [CrossRef]
15. Papatheou, E.; Dervilis, N.; Maguire, A.E.; Campos, C.; Antoniadou, I.; Worden, K. Performance monitoring of a wind turbine using extreme function theory. *Renew. Energy* **2017**, *113*, 1490–1502. [CrossRef]
16. Zugasti Uriguen, E. Design and Validation of a Methodology for Wind Energy Structures Health Monitoring. Ph.D. Thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 2014.
17. Pozo, F.; Vidal, Y. Wind turbine fault detection through principal component analysis and statistical hypothesis testing. *Energies* **2016**, *9*, 3. [CrossRef]
18. Ziegler, L.; Muskulus, M. Comparing a fracture mechanics model to the SN-curve approach for jacket-supported offshore wind turbines: Challenges and opportunities for lifetime prediction. In Proceedings of the ASME 2016 35th International Conference on Ocean, Offshore and Arctic Engineering, Busan, Korea, 18–24 June 2016.
19. Pozo, F.; Vidal, Y.; Serrahima, J. On real-time fault detection in wind turbines: Sensor selection algorithm and detection time reduction analysis. *Energies* **2016**, *9*, 520. [CrossRef]
20. Anaissi, A.; Makki Alamdari, M.; Rakotoarivelo, T.; Khoa, N. A tensor-based structural damage identification and severity assessment. *Sensors* **2018**, *18*, 111. [CrossRef] [PubMed]
21. Westerhuis, J.A.; Kourti, T.; MacGregor, J.F. Comparing alternative approaches for multivariate statistical analysis of batch process data. *J. Chemom.* **1999**, *13*, 397–413. [CrossRef]
22. Mujica, L.; Rodellar, J.; Fernandez, A.; Güemes, A. Q-statistic and T2-statistic PCA-based measures for damage assessment in structures. *Struct. Health Monit.* **2011**, *10*, 539–553. [CrossRef]
23. Pozo, F.; Vidal, Y.; Salgado, Ó. Wind turbine condition monitoring strategy through multiway PCA and multivariate inference. *Energies* **2018**, *11*, 749. [CrossRef]
24. Wang, Y.; Ma, X.; Qian, P. Wind turbine fault detection and identification through PCA-based optimal variable selection. *IEEE Trans. Sustain. Energy* **2018**, *9*, 1627–1635. [CrossRef]
25. Jolliffe, I.T.; Cadima, J. Principal component analysis: A review and recent developments. *Philos. Trans. R. Soc. A* **2016**, *374*, 20150202. [CrossRef]
26. Tan, M.; Zhang, Z. Wind turbine modeling with data-driven methods and radially uniform designs. *IEEE Trans. Ind. Informatics* **2016**, *12*, 1261–1269. [CrossRef]
27. Vitola, J.; Pozo, F.; Tibaduiza, D.A.; Anaya, M. A sensor data fusion system based on k-nearest neighbor pattern classification for structural health monitoring applications. *Sensors* **2017**, *17*, 417. [CrossRef]
28. Smola, A.J.; Schölkopf, B. A tutorial on support vector regression. *Stat. Comput.* **2004**, *14*, 199–222. [CrossRef]
29. Vidal, Y.; Pozo, F.; Tutivén, C. Wind turbine multi-fault detection and classification based on SCADA data. *Energies* **2018**, *11*, 3018. [CrossRef]
30. Sergios Theodoridis and Konstantiinos Koutroumbas. *Pattern Recognition*; Elsevier: Amsterdam, The Netherlands, 2009.
31. Scholkopf, B.; Sung, K.K.; Burges, C.J.; Girosi, F.; Niyogi, P.; Poggio, T.; Vapnik, V. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Trans. Signal Process.* **1997**, *45*, 2758–2765. [CrossRef]
32. Allwein, E.L.; Schapire, R.E.; Singer, Y. Reducing multiclass to binary: A unifying approach for margin classifiers. *J. Mach. Learn. Res.* **2000**, *1*, 113–141.
33. Ruiz, M.; Mujica, L.E.; Alferez, S.; Acho, L.; Tutiven, C.; Vidal, Y.; Rodellar, J.; Pozo, F. Wind turbine fault detection and classification by means of image texture analysis. *Mech. Syst. Sig. Process.* **2018**, *107*, 149–167. [CrossRef]
34. Hossin, M.; Sulaiman, M. A review on evaluation metrics for data classification evaluations. *IJDKP* **2015**, *5*, 1–11.
35. Krüger, F. Activity, Context, and Plan Recognition with Computational Causal Behaviour Models. Ph.D. Thesis, University of Rostock, Mecklenburg, Germany, 2016.

36. Hameed, N.; Hameed, F.; Shabut, A.; Khan, S.; Cirstea, S.; Hossain, A. An Intelligent Computer-Aided Scheme for Classifying Multiple Skin Lesions. *Computers* **2019**, *8*, 62. [CrossRef]
37. Häckell, M.W.; Rolfes, R.; Kane, M.B.; Lynch, J.P. Three-tier modular structural health monitoring framework using environmental and operational condition clustering for data normalization: Validation on an operational wind turbine system. *Proc. IEEE* **2016**, *104*, 1632–1646. [CrossRef]
38. Kraemer, P.; Friedmann, H.; Ebert, C.; Mahowald, J.; Wölfel, B. Experimental validation of stochastic subspace algorithms for structural health monitoring of offshore wind turbine towers and foundations. In Proceedings of the 8th European Workshop On Structural Health Monitoring, Bilbao, Spain, 5–8 July 2016.
39. Fritzen, C.P.; Kraemer, P.; Buethe, I. Vibration-based damage detection under changing environmental and operational conditions. *Adv. Sci. Technol. Water Resour.* **2013**, *83*, pp. 95–104. [CrossRef]
40. Ostachowicz, W.; Güemes, A. *New Trends in Structural Health Monitoring*; Springer Science & Business Media: New York, NY, USA, 2013; Volume 542.

B.2. Libro: Robótica y Computación. Nuevos Avances

Es un libro publicado como parte de las memorias del Congreso Internacional de Robótica y Computación que este año celebra su 7ma edición en la ciudad de la Paz en Baja California.

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de La Paz

Robótica y Computación.
Nuevos Avances

TECNOLOGICO
NACIONAL DE MÉXICO

Coordinadores de la Edición
Iliana Castro Liera
Mario Cortés Larrinaga

ISBN: 978-607-98174-6-6

Figura B.3: Portada del libro Robótica y Computación. Nuevos Avances

B.2.1. Artículo publicado

Robótica y Computación. Nuevos Avances

Monitorización de la integridad estructural para aerogeneradores *offshore* con estructura tipo *jacket* mediante métodos de aprendizaje autómatico.

Gabriela Aquino González
Facultad de Ciencias de la Electrónica (FCE)
Benemérita Universidad Autónoma de Puebla (BUAP)
Puebla, México.
gabriela.aquino@alumno.buap.mx

Yolanda Vidal Seguí
Control, Modeling, Identification and Applications (CoDAlab)
Universitat Politècnica de Catalunya (UPC)
Barcelona, España.
yolanda.vidal@upc.edu

José Eligió Moisés Gutiérrez Arias
Facultad de Ciencias de la Electrónica (FCE)
Benemérita Universidad Autónoma de Puebla (BUAP)
Puebla, México.
arigutmses5@gmail.com

Francesc Pozo Montero
Control, Modeling, Identification and Applications (CoDAlab)
Universitat Politècnica de Catalunya (UPC)
Barcelona, España.
francesc.pozo@upc.edu

***Resumen*—La energía eólica *offshore* tiene por objetivo aprovechar la fuerza del viento producida en alta mar. Para ello se colocan aerogeneradores, algunos de los cuales se encuentran en aguas profundas y utilizan plataformas con estructuras de soporte tipo *jacket* que hoy en día son la mejor solución para hacer frente a este entorno. Un problema al que se enfrentan las plataformas *offshore* son los daños estructurales, por lo cual la monitorización de la integridad estructural es imprescindible. En este trabajo, se establece una metodología para el diagnóstico y localización de un daño estructural en plataformas con estructuras tipo *jacket*. El enfoque propuesto se valida experimentalmente dentro del laboratorio CoDALab de la Universidad Politécnica de Cataluña en una plataforma a escala.**

***Palabras Clave*—monitorización; integridad estructural; estructura tipo *jacket*; acelerómetros; aprendizaje supervisado; máquina de soporte vectorial, análisis de componentes principales; remodelación de datos**

I. Introducción

En los parques eólicos *offshore*, las turbinas eólicas también conocidas como aerogeneradores se erigen con diferentes tipos de soportes, dependiendo de la profundidad a la que se instalará el aerogenerador. A profundidades inferiores a 15 metros se utilizan monopilotes, que son estructuras bastante simples ancladas directamente al fondo marino. Para aerogeneradores colocados a profundidades menores o iguales a 30 metros, generalmente se usa el sistema de soportes por gravedad, que implica el uso de una gran plataforma de hormigón o acero con un diámetro y un peso aproximados de 15 metros y 1,000 toneladas respectivamente. Finalmente, a profundidades de más de 30 metros, las compañías de instalación usan soportes tipo *jacket*, éstos son estructuras que presentan tres o cuatro puntos de anclaje al fondo marino, lo que aumenta los niveles de seguridad al anclar las torres como se muestra en la Fig. 1. La energía eólica *offshore* tiene un potencial enorme, no obstante, solo puede explotarse disminuyendo los costos de operación y mantenimiento. Por lo que la monitorización de la integridad estructural para proporcionar una alerta temprana de daños son esenciales para lograr este objetivo. Éste documento se enfoca en el problema de la detección de daños para soportes con estructuras tipo *jacket*.

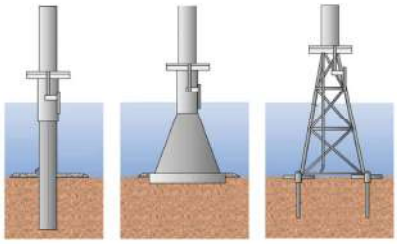


Fig. 1: Soportes para aerogeneradores *offshore* [1]. De izquierda a derecha monopilote, de gravedad y *jacket*.

En la literatura se pueden encontrar muchas metodologías para la detección de daños, entre las mas prolíficas se encuentran las que utilizan métodos basados en vibración [2]. El interés en estos métodos ha estado creciendo en los últimos años debido a su simplicidad, facilidad de uso y alta efectividad [3]. Sin embargo, la mayoría de los estudios se centran principalmente en el caso de señales de respuesta de vibración y excitación de entrada medibles con solo unos pocos estudios recientes [4] centrados en solo en la respuesta a la vibración, cuya importancia se deriva del hecho de que en algunas aplicaciones la excitación no se puede imponer y a menudo, no se puede medir. Durante las décadas de 1970 y 1980,

la industria mostró interés en los métodos de identificación de daños basados en vibraciones para plataformas *offshore*, pero debido a muchos problemas prácticos estos esfuerzos se abandonaron a principios de la década de 1990 [5]. En los últimos años, algunos investigadores han discutido sobre la detección de daños en fundaciones en alta mar. Un trabajo a destacar es el realizado por Weijtjens et al. [6] relacionado con el monitoreo de un aerogenerador *offshore* real con monopilote basado en sus frecuencias de resonancia, donde los problemas clave son la variabilidad operativa y ambiental de las frecuencias de resonancia de la turbina que pueden ocultar cualquier cambio estructural. Finalmente, en [7] se utiliza un banco de pruebas experimental similar al indicado en este trabajo, la detección de daños se realiza (pero no la localización o clasificación) por medio del indicador de daño estimado de matriz de covarianza.

Este documento aporta un método de detección y localización de daños (este último se trata como un problema de clasificación) para un modelo a escala de un aerogenerador *offshore* con estructura tipo *jacket* utilizando solo datos de respuesta de aceleración. De acuerdo a [8], la excitación disponible es el viento, por lo que la excitación de entrada no es medible. En este trabajo la excitación de la vibración es dada por el viento (no se puede imponer y se supone que es desconocida).

La metodología aportada comprende los siguientes pasos:

- Simulación del viento como un ruido blanco gaussiano sobre el aerogenerador.
- Recopilación de datos. Se adquieren los datos procedentes de los acelerómetros colocados en la plataforma del aerogenerador y se crea una base de datos.
- Remodelación de los datos. Los datos en crudo se preprocesan usando una remodelación de la información (para aumentar la cantidad de información contenida en cada observación/muestra).
- Estandarización de los datos y el escalado de columnas. Esto se realiza con el fin de simplificar el cálculo de los componentes principales.
- Extracción de características. La técnica PCA (Análisis de componentes principales por sus siglas en inglés) se utiliza para extraer características, así como para reducir la dimensionalidad de los datos y el tiempo de cálculo.
- Clasificación. Se prueban los clasificadores vecino más cercano k (k-NN) y máquina de soporte vectorial (SVM) con *kernel* cuadrático. En este punto se incluye la evaluación del modelo de clasificación para sintonizar los parámetros requeridos por cada modelo de clasificación.

La fiabilidad del método propuesto se verifica de manera experimental utilizando un banco de pruebas explicado a detalle en secciones posteriores.

La estructura del documento se describe a continuación: la Sección II presenta el banco de pruebas de laboratorio utilizado para validar el enfoque propuesto. La Sección III detalla la metodología de detección y clasificación de daños. En la Sección IV se muestran los resultados obtenidos y finalmente las conclusiones son mostradas en la Sección V.

II. BANCO DE PRUEBAS DE LABORATORIO

La Fig.2 muestra la descripción del experimento, el cual se explica de la siguiente manera:

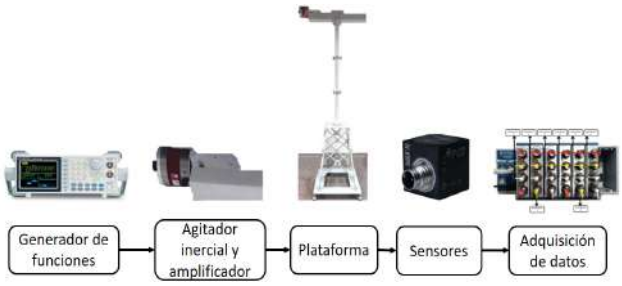


Fig. 2: Descripción general del banco de pruebas experimentales.

El experimento se inicia con una señal de ruido blanco dada por el generador de funciones. En él se simularon diferentes velocidades del viento multiplicando la amplitud de la señal de ruido blanco por los factores 0.5, 1, 2 y 3 de esta manera se crea la vibración necesaria para excitar la estructura y generar vibraciones (similares a las producidas por ráfagas de viento en las palas) a la plataforma en el laboratorio. La señal de ruido blanco emitida por el generador de funciones se amplifica y esta señal eléctrica se aplica al agitador. El agitador se coloca en la parte superior de la estructura, simulando así la masa de la góndola y los efectos ambientales del viento sobre toda la plataforma.

La plataforma a escala utilizada en este trabajo se muestra en la Fig. 3 (izquierda) y tiene tres componentes estructurales diferentes: góndola, torre y el soporte tipo *jacket*.

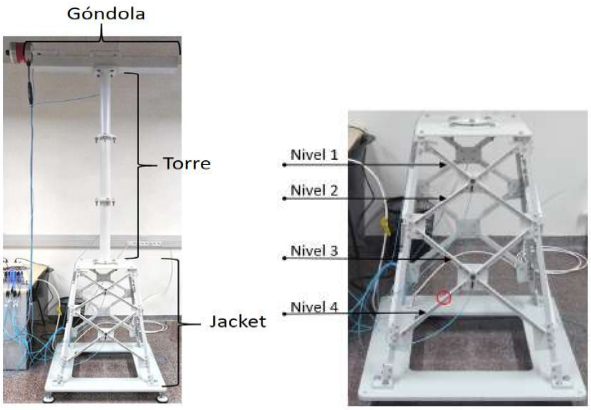


Fig. 3: Izquierda: Modelo a escala de la plataforma de un aerogenerador *offshore* con estructura tipo *jacket* utilizada en las pruebas experimentales). Derecha: Niveles de la estructura *jacket* donde se introduce el daño por grieta (círculo rojo).

La estructura tipo *jacket* tiene cuatro longitudes de barra diferentes, cada una en diferentes niveles. El daño (grieta de 5 mm) se introducirá uno a la vez en los cuatro niveles diferentes como se ilustra en la Fig.3 (derecha). Se ha demostrado que las grietas son uno de los principales tipos de daños encontrados en las plataformas de aerogeneradores *offshore* [9].

Finalmente, la plataforma es monitoreada por 8 acelerómetros triaxiales (PCB Piezotronics, modelo 356A17) que están conectados al sistema de adquisición de datos. Los acelerómetros han sido ubicados estratégicamente como se muestra en la Fig. 4. Al tener los acelerómetros una salida para cada componente espacial x, y y z, se obtienen datos de 24 señales de salida.

Fig. 4: Ubicación de los sensores en la plataforma.

III. Metodología de detección y clasificación de daños

Para este trabajo fue necesario obtener un conjunto de datos sobre el cual se pudiera trabajar y que permitiera entrenar el algoritmo de aprendizaje supervisado.

III-A. Recopilación de datos

En el banco de pruebas de laboratorio se han considerado 5 estados estructurales diferentes en las barras de la estructura tipo *jacket*: estado saludable, daño en nivel 1, daño en nivel 2, daño en nivel 3 y daño en nivel 4. Para cada uno de estos casos se hicieron pruebas con las 4 amplitudes de ruido blanco: 0.5, 1, 2 y 3 obteniendo un total de 140 experimentos/pruebas cada uno de duración 60 segundos como se muestra en la Tabla I.

Estado estructural	Amplitud				Total de pruebas
	0.5	1	2	3	
1. Estado saludable	15	15	15	15	60
2. Daño en nivel 1	5	5	5	5	20
3. Daño en nivel 2	5	5	5	5	20
4. Daño en nivel 3	5	5	5	5	20
5. Daño en nivel 4	5	5	5	5	20
	35	35	35	35	140

Tabla I: Pruebas realizadas para cada caso y amplitud.

Para los sensores se ha considerado una frecuencia de muestreo de 275 Hz, factible en un entorno *offshore* [10]. De este modo, los datos en crudo se almacenan en una matriz $\mathbf{Z}$, la cual para cada prueba almacena $16,517$ muestras de cada una de las mediciones de los 24 sensores como en (1).

$$\mathbf{Z}^{(p)} = \begin{pmatrix} z^{(p)}_{1,1} & z^{(p)}_{1,2} & \cdots & z^{(p)}_{1,24} \\ z^{(p)}_{2,1} & z^{(p)}_{2,2} & \cdots & z^{(p)}_{2,24} \\ z^{(p)}_{3,1} & z^{(p)}_{3,2} & \cdots & z^{(p)}_{3,24} \\ \vdots & \vdots & \ddots & \vdots \\ z^{(p)}_{16517,1} & z^{(p)}_{16517,2} & \cdots & z^{(p)}_{16517,24} \end{pmatrix} \tag{1}$$

donde $p = 1, 2, \cdots, 140$ es el número de prueba. Al tener un total de 140 pruebas, se obtiene (2) una matriz $\mathbf{Z}$ de datos en crudo con una dimensión total de $2,312,380 \times 24$.

$$\mathbf{Z} = \begin{pmatrix} \mathbf{Z}^{(1)} \\ \vdots \\ \mathbf{Z}^{(p)} \\ \vdots \\ \mathbf{Z}^{(140)} \end{pmatrix} \tag{2}$$

III-B. Remodelación de los datos

Con el objetivo de incrementar la cantidad de información contenida en cada muestra (fila), se hace la remodelación de los datos de la matriz $\mathbf{Z}$.

Dada la naturaleza tridimensional de la información recopilada en este documento (tiempo, sensores, experimentos), la remodelación se realizó en base al desarrollo propuesto por Westerhuis et al. [11]. Donde ahora nuestra información estará representada por una matriz bidimensional $\mathbf{X} = \left(x^{k,l}_{i,j}\right) \in \mathcal{M}_{(n_1+\cdots+n_5)\times(24\cdot 199)}$ de la siguiente manera:

En la matriz $\mathbf{X}$ mostrada en (3), los dos subíndices $i = 1, \ldots, n_j$ y $j = 1, \ldots, J$ y los dos superíndices $k = 1, \ldots, K$ y $l = 1, \ldots, L$ están relacionados con la prueba experimental, el estado estructural, el sensor y el instante de tiempo, respectivamente. Cabe mencionar que J es la cantidad de diferentes estados estructurales; K es el número total de sensores y L es el número de marcas de tiempo por experimento.

De acuerdo a la información obtenida en la Sección III-A hemos considerado:

- $n_1 + n_2 + n_3 + n_4 + n_5 = 11,620$ muestras.
  - $n_1 = 4,980$
  - $n_2 = 1,660$
  - $n_3 = 1,660$
  - $n_4 = 1,660$
  - $n_5 = 1,660$
- $J = \ 5$ estados estructurales.
- $K = 24$ sensores.
- $L = 199$ instantes de tiempo

Por tanto al remodelar nuestros datos, finalmente obtenemos la matriz $\mathbf{X}$ de datos en crudo con dimensiones $11,620 \times 4,776$.

De esta manera se ha recopilado la información relacionada con las mediciones del sensor y sus variaciones a lo largo del tiempo.

ISBN: 978-607-98174-6-6

$$\mathbf{X} = \left(x_{i,j}^{k,l}\right) = \left[\begin{array}{ccc|ccc|c|ccc} x_{1,1}^{1,1} & \cdots & x_{1,1}^{1,199} & x_{1,1}^{2,1} & \cdots & x_{1,1}^{2,199} & \cdots & x_{1,1}^{24,1} & \cdots & x_{1,1}^{24,199} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{4980,1}^{1,1} & \cdots & x_{4980,1}^{1,199} & x_{n_1,1}^{2,1} & \cdots & x_{n_1,1}^{2,199} & \cdots & x_{4980,1}^{24,1} & \cdots & x_{4980,1}^{24,199} \\ \hline x_{1,2}^{1,1} & \cdots & x_{1,2}^{1,199} & x_{1,2}^{2,1} & \cdots & x_{1,2}^{2,199} & \cdots & x_{1,2}^{24,1} & \cdots & x_{1,2}^{24,199} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{1660,2}^{1,1} & \cdots & x_{1660,2}^{1,199} & x_{1660,2}^{2,1} & \cdots & x_{1660,2}^{2,199} & \cdots & x_{1660,2}^{24,1} & \cdots & x_{1660,2}^{24,5} \\ \hline \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \hline x_{1,5}^{1,1} & \cdots & x_{1,5}^{1,199} & x_{1,5}^{2,1} & \cdots & x_{1,5}^{2,199} & \cdots & x_{1,5}^{24,1} & \cdots & x_{1,5}^{24,199} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{1660,5}^{1,1} & \cdots & x_{1660,5}^{1,199} & x_{1660,5}^{2,1} & \cdots & x_{1660,5}^{2,199} & \cdots & x_{1660,5}^{24,1} & \cdots & x_{1660,5}^{24,199} \end{array}\right] \quad (3)$$

$$= \left[\begin{array}{c} \mathbf{X}_1 \\ \hline \mathbf{X}_2 \\ \hline \vdots \\ \hline \mathbf{X}_5 \end{array}\right] = \left[\begin{array}{c|c|c|c} \mathbf{X}^1 & \mathbf{X}^2 & \cdots & \mathbf{X}^{24} \end{array}\right].$$

III-C. Estandarización

Los datos en crudo, o sea, los datos sin procesar se deben estandarizar ya que PCA es sensible a la magnitud de las variables además que simplificará el cálculo empírico de la matriz de covarianza necesario para la descomposición de PCA.

En este trabajo, se utiliza la estandarización en columnas, de esta manera cada columna de la nueva matriz estandarizada $\breve{\mathbf{X}}$, tiene una media de cero y una desviación estándar de uno [12].

III-D. PCA (Análisis de Componentes Principales)

El método PCA genera un nuevo conjunto de variables conocidas como componentes principales (CP).

La matriz de transformación se calcula como una multiplicación de matriz a matriz.

$$\mathbf{T} = \breve{\mathbf{X}}\mathbf{P},$$

donde $\mathbf{T}$ es una matriz de $11,620 \times 4,776$ y $\mathbf{P}$ es la matriz que contiene escritos como columnas las componentes principales de la matriz $\breve{\mathbf{X}}$.

La reducción de dimensionalidad se realiza a través de un modelo PCA reducido mostrado en (4), donde $\mathbf{P}_\ell$ tiene las primeras CP ℓ y $\mathbf{T}_\ell$ es la proyección de la matriz escalada $\breve{\mathbf{X}}$.

$$\mathbf{T}_\ell = \breve{\mathbf{X}}\mathbf{P}_\ell. \quad (4)$$

Al aplicar la estandarización de columnas, la matriz de varianza es igual a $K \cdot L$. Esto significa que las primeras CP ℓ conservan una proporción de varianza-covarianza dada por:

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_\ell}{\lambda_1 + \lambda_2 + \cdots + \lambda_{4,776}},$$

donde $\lambda_1+\lambda_2+\cdots+\lambda_{4,776}$ son los valores propios asociados con las componentes principales de la matriz de varianza-covarianza, en orden decreciente.

En particular, en este documento se utilizan las primeras 443, 887 y 1770 componentes principales de la descomposición de PCA ya que representan $85\,\%$, $90\,\%$ y $95\,\%$ de la varianza, esto se puede observar en la Fig. 5.

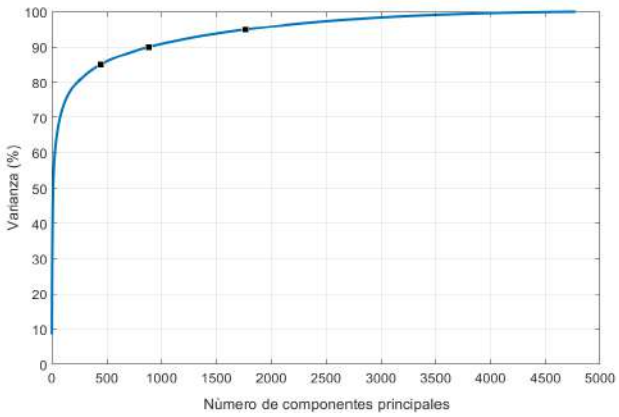


Fig. 5: Gráfica de relación de las CP con respecto al porcentaje de varianza.

III-E. Clasificación

Para poder diagnosticar y localizar un daño estructural, en este documento se prueban los algoritmos de clasificación de aprendizaje supervisado: k-NN y SVM. Después el rendimiento de cada uno se compara a través de diferentes métricas de evaluación.

El clasificador k-NN almacena los datos de entrenamiento en la memoria en lugar de construir un modelo y compara los nuevos datos de prueba con las instancias guardadas más cercanas para realizar la predicción.

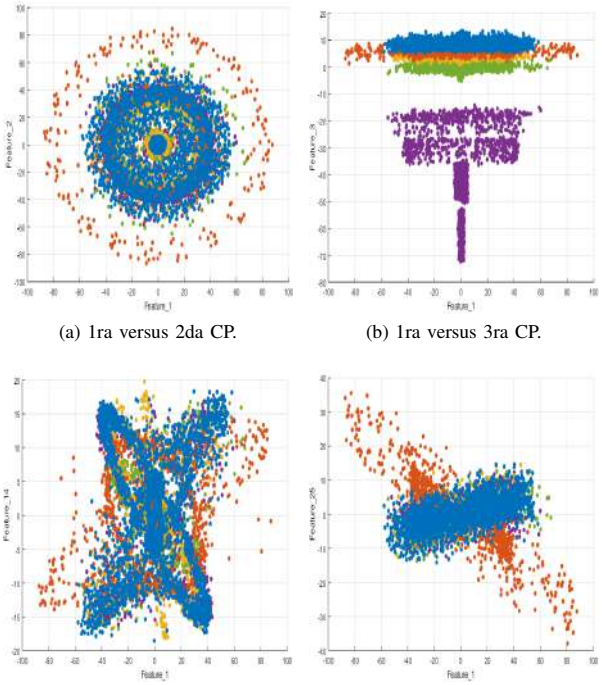
La clasificación SVM es una técnica de clasificación binaria que debe adaptarse para hacer frente a problemas de clasificación múltiple. De manera especifica se utilizó un clasificador SVM con *kernel* cuadrático representado en (5). Esta elección se realizó de acuerdo a lo observado en los diagramas de dispersión de la Fig. 6 donde se identifica una relación cuadrática.

$$K(x_i, x_j) = \left(1 + \frac{1}{\rho^2} x_i^\mathsf{T} x_j\right)^2 \quad (5)$$

donde ρ es el parámetro *kernel scale* y x_i y x_j denotan las diferentes muestras de nuestro conjunto de datos.

III-F. Validación cruzada κ-Fold

La validación cruzada es una técnica utilizada para evaluar los resultados y garantizar que sean independientes de la partición entre los datos de entrenamiento y prueba. En este documento se utiliza la validación cruzada de $\kappa = 5$.



(a) 1ra versus 2da CP. (b) 1ra versus 3ra CP.

(c) 1ra versus 14va CP. (d) 1ra versus 25va CP.

Fig. 6: Diagrama de dispersión de las muestras de acuerdo a los diferentes estados estructurales. Sanas (puntos azules), daño en nivel 1 (puntos naranjas), daño en nivel 2 (puntos amarillos), daño en nivel 3 (puntos morados) y daño en nivel 4 (puntos verdes).

III-G. Métricas para la evaluación de modelos de clasificación

En problemas de clasificación, para saber si el modelo que hemos entrenado tiene el mejor rendimiento de acuerdo con el problema presentado, es importante evaluar el modelo de clasificación a través de métricas. Las métricas que se han empleado son la exactitud, precisión, sensibilidad, F1$-$ *score* y especificidad. Como se muestra en [13], éstas son fáciles de calcular y aplicables a problemas de clasificación binaria y múltiple.

De acuerdo a [14] y [15], cuando el problema de clasificación tiene varias clases el resultado es el promedio obtenido al sumar el resultado de cada clase y dividir el número total de clases. Las fórmulas para calcular estas métricas en un modelo de clasificación de varias clases se muestran en la Tabla II.

Donde TP_j es el número de casos positivos que fueron identificados correctamente de cada clase $j = 1, \ldots, 5$; TN_j es la suma de los casos negativos que fueron clasificados correctamente; FP_j es la cantidad de casos negativos que fueron clasificados incorrectamente como positivos; FN_j es el total de casos positivos que fueron clasificados incorrectamente como negativos. Estos datos son obtenidos a partir de una matriz de confusión multiclase.

Métrica	**Formula**
Exactitud promedio ($\overline{\mathrm{acc}}$)	$\frac{1}{5}\sum_{j=1}^{5}\frac{\mathrm{TP}_j+\mathrm{TN}_j}{\mathrm{TP}_j+\mathrm{FP}_j+\mathrm{FN}_j+\mathrm{TN}_j}$
Precisión promedio ($\overline{\mathrm{ppv}}$)	$\frac{1}{5}\sum_{j=1}^{5}\frac{\mathrm{TP}_j}{\mathrm{TP}_j+\mathrm{FP}_j}$
Sensibilidad promedio ($\overline{\mathrm{tpr}}$)	$\frac{1}{5}\sum_{j=1}^{5}\frac{\mathrm{TP}_j}{\mathrm{TP}_j+\mathrm{FN}_j}$
F1-Score promedio ($\overline{\mathrm{F1}}$)	$2\frac{\overline{\mathrm{ppv}}\times\overline{\mathrm{tpr}}}{\overline{\mathrm{ppv}}+\overline{\mathrm{tpr}}}$
Especificidad promedio ($\overline{\mathrm{tnr}}$)	$\frac{1}{5}\sum_{j=1}^{5}\frac{\mathrm{TN}_j}{\mathrm{TN}_j+\mathrm{FP}_j}$

Tabla II: Métricas para evaluar modelos de clasificación de varias clases, donde j se refiere a la clase individual y 5 es el número total de clases.

IV. Resultados

Los mejores modelos obtenidos de los métodos de clasificación k-NN y SVM, se detallan en esta sección.

IV-A. Resultados para el método de clasificación k-NN

Primero, se prueba el clasificador k-NN. Los indicadores introducidos en la Sección III-G se emplean para ajustar el valor del parámetro k, es decir, el número de vecinos que se utilizarán y decidir la mejor variación que se adoptará al aplicar PCA.

Varianza	k	$\overline{\mathrm{acc}}$	$\overline{\mathrm{ppv}}$	$\overline{\mathrm{tpr}}$	$\overline{\mathrm{F1}}$	$\overline{\mathrm{tnr}}$
85 % (443 CP)	1	0.857	0.860	0.846	0.853	0.950
	5	0.856	0.869	0.82	0.844	0.953
	10	0.867	0.891	0.827	0.858	0.956
	25	0.868	0.898	0.828	0.862	0.955
	50	0.874	0.912	0.831	0.87	0.958
	100	0.911	0.928	0.879	0.903	0.981
	150	0.933	0.942	0.908	0.925	0.988
	200	**0.946**	**0.947**	**0.924**	**0.936**	**0.993**
	250	0.940	0.938	0.917	0.927	0.991
	300	0.930	0.929	0.903	0.915	0.988
	500	0.899	0.909	0.859	0.884	0.976
90 % (887 CP)	1	0.857	0.856	0.845	0.85	0.961
	5	0.860	0.870	0.828	0.849	0.956
	10	0.872	0.887	0.838	0.862	0.961
	25	0.873	0.889	0.840	0.864	0.961
	50	0.884	0.913	0.846	0.878	0.964
	100	0.917	0.930	0.886	0.907	0.984
	150	0.938	0.945	0.914	0.929	0.990
	200	**0.950**	**0.950**	**0.931**	**0.940**	**0.995**
	250	0.944	0.941	0.922	0.932	0.993
	300	0.933	0.931	0.906	0.918	0.989
	500	0.903	0.912	0.864	0.888	0.977
95 % (1770 CP)	1	0.854	0.853	0.842	0.847	0.960
	5	0.858	0.865	0.829	0.847	0.956
	10	0.872	0.885	0.841	0.862	0.962
	25	0.873	0.888	0.841	0.864	0.962
	50	0.885	0.911	0.848	0.879	0.965
	100	0.918	0.930	0.888	0.908	0.985
	150	0.938	0.945	0.915	0.930	0.990
	200	**0.950**	**0.950**	**0.930**	**0.940**	**0.995**
	250	0.945	0.942	0.923	0.932	0.993
	300	0.934	0.931	0.908	0.919	0.989
	500	0.904	0.913	0.866	0.889	0.978

Tabla III: Indicadores de evaluación para el método k -NN usando diferentes porcentajes de varianza y diferente número de vecinos más cercanos (k). Los mejores clasificadores para cada variación se resaltan en negrita.

La Tabla III muestra los resultados obtenidos al trabajar con 85 %, 90 % y 95 % de la varianza, e ir variando el número

de vecinos k. Los clasificadores con el mejor rendimiento se resaltan en negrita. Se puede observar que los mejores clasificadores son 90 % y 95 % de la varianza y una $k = 200$. Se decide trabajar con 90 % de varianza ya que reducirá el requisito de memoria computacional, lo cual se comprueba en la Tabla IV.

Varianza	Exactitud	Velocidad de predicción (obs/seg)	Tiempo de entrenamiento (seg)
85 %	94.6 %	210	329
90 %	**95.0 %**	**110**	**638**
95 %	95.0 %	55	1251

Tabla IV: Resultados de la velocidad de predicción y el tiempo de entrenamiento para los mejores clasificadores de rendimiento de cada variación (85 %, 90 % y 95 %) utilizando el método de clasificación k -NN en una computadora Intel Core i7 de 3 GB y 16 GB de RAM.

Retomando la Tabla III, en particular para este clasificador con el 90 % de varianza, podemos observar que inicialmente, al aumentar el número de vecinos los valores de todas la métricas tienden a mejorar, sin embargo, cuando $k > 200$ el rendimiento de todas las métricas decrece.

En la matriz de confusión mostrada en la Tabla V, cada fila representa las clases verdaderas mientras que cada columna representa las clases predichas (por el clasificador). En particular, la primera fila (y la primera columna) están etiquetadas como 0 y corresponden al estado saludable. Las siguientes etiquetas (para filas y columnas) 1, 2, 3 y 4 corresponden al daño en los niveles correspondientes de la estructura tipo *jacket*. La exactitud que se obtiene es de 95 %, mientras que la precisión también es de 95 %, lo cual es el porcentaje que este modelo ha sido capaz de clasificar correctamente.

		Clase predicha				
		0	**1**	**2**	**3**	**4**
Clase verdadera	**0**	> 99 %		< 1 %		
	1	< 1 %	98 %	2 %		
	2	7 %		93 %		< 1 %
	3				75 %	25 %
	4	1 %		< 1 %		99 %

Tabla V: Matriz de confusión para el algoritmo k -NN con $k = 200$ vecinos. La etiqueta 0 corresponde al estado saludable y las etiquetas 1, 2, 3 y 4 corresponden al estado de daño ubicado en el nivel correspondiente. Un cuadrado en blanco significa 0 %.

IV-B. Resultados para el método de clasificación SVM

Dado que los diagramas de dispersión en la Fig. 6 revelan una relación cuadrática (particularmente la primera frente a la segunda componente principal), se probó el clasificador SVM cuadrático. En este caso los parámetros a sintonizar son C (*box constrain*) y ρ (*kernel scale*), que se ajustan usando las métricas detalladas en la Subsección III-G.

La Tabla VI resume los resultados obtenidos al trabajar con 85 %, 90 % y 95 % de la varianza. Dado que el problema que estamos tratando parece un problema separable, no se

Varianza	C	ρ	acc	ppv	tpr	F1	tnr
85 % (443 CP)	1	5	0.987	0.993	0.983	0.988	0.995
		20	0.994	0.991	0.991	0.991	**1.000**
		30	0.995	0.994	0.994	0.994	**1.000**
		40	0.996	0.995	0.995	0.995	**1.000**
		50	0.997	0.996	0.996	0.996	**1.000**
		60	0.998	0.997	0.997	0.997	**1.000**
		70	0.998	0.998	0.998	0.998	**1.000**
		80	**0.999**	0.998	0.998	0.998	**1.000**
		90	**0.999**	**0.999**	**0.999**	**0.999**	1.000
		100	**0.999**	**0.999**	**0.999**	**0.999**	1.000
		150	0.998	0.998	0.998	0.998	**1.000**
		200	0.997	0.996	0.997	0.997	0.999
		300	0.882	0.926	0.835	0.878	0.954
90 % (887 CP)	1	5	0.987	0.993	0.982	0.988	0.995
		20	0.991	0.988	0.988	0.988	**1.000**
		30	0.995	0.993	0.993	0.993	**1.000**
		40	0.996	0.995	0.995	0.995	**1.000**
		50	0.997	0.997	0.997	0.997	**1.000**
		60	**0.998**	0.997	0.997	0.997	**1.000**
		70	**0.998**	0.997	0.997	0.997	**1.000**
		80	**0.998**	0.997	0.997	0.997	**1.000**
		90	**0.998**	**0.998**	**0.998**	**0.998**	**1.000**
		100	**0.998**	**0.998**	**0.998**	**0.998**	**1.000**
		150	**0.998**	**0.998**	**0.998**	**0.998**	0.999
		200	0.997	0.996	0.996	0.996	0.999
		300	0.883	0.927	0.837	0.880	0.955
95 % (1770 CP)	1	5	0.986	0.993	0.981	0.987	0.994
		20	0.990	0.987	0.986	0.986	**1.000**
		30	0.994	0.992	0.992	0.992	**1.000**
		40	0.996	0.994	0.994	0.994	**1.000**
		50	0.997	0.996	0.996	0.996	**1.000**
		60	0.997	0.997	0.997	0.997	**1.000**
		70	**0.998**	0.997	0.997	0.997	**1.000**
		80	**0.998**	0.997	**0.998**	**0.998**	**1.000**
		90	**0.998**	**0.998**	**0.998**	**0.998**	**1.000**
		100	**0.998**	0.997	**0.998**	**0.998**	**1.000**
		150	**0.998**	0.997	**0.998**	0.997	0.999
		200	0.996	0.996	0.996	0.996	0.999
		300	0.998	0.997	0.997	0.997	1.000

Tabla VI: Indicadores de evaluación para el modelo SVM usando diferentes porcentajes de varianza y diferentes valores para el parámetro ρ (*Kernel scale*). Como es un problema separable, tomamos el valor del parámetro C como 1. Los mejores valores para cada indicador están resaltados en negrita.

encontraron cambios en el desempeño de los indicadores al tomar diferentes valores de C. Por consiguiente se consideró $C = 1$. Los mejores casos son los que presentan 85 % de la varianza con $\rho = 90$ y $\rho = 100$.

De acuerdo con la Tabla VII, se observa que el clasificador con 85 % de la varianza y $\rho = 90$ tiene una mayor velocidad de predicción y un tiempo de entrenamiento más corto. De manera que este clasificador tiene el mejor rendimiento.

Varianza	Kernel scale ρ	Exactitud	Velocidad de predicción (obs/seg)	Tiempo de entrenamiento (seg)
85 %	**90**	**99.9 %**	**1700**	**76**
	100	99.9 %	1600	79
90 %	90	99.8 %	320	256
	100	99.8 %	580	364
95 %	90	99.8 %	100	676

Tabla VII: Resultados de velocidad de predicción y tiempo de entrenamiento para los mejores casos de rendimiento de cada variación (85 %, 90 % y 95 %) utilizando el método de clasificación SVM.

Una vez determinado el mejor clasificador, observamos en

la Tabla VI el rendimiento de las métricas para el caso con 85 % de la varianza, el incrementar el parámetro ρ mejora el rendimiento general del método de clasificación hasta cierto punto. La degradación del rendimiento aparece para valores superiores a $\rho = 100$.

La matriz de confusión para el mejor clasificador (85 % varianza y $\rho = 90$) se representa en la Tabla VIII. De esta matriz de confusión y de acuerdo con las métricas de evaluación, obtenemos una exactitud promedio de 99.9 %, una precisión promedio de 99.9 % y una especificidad promedio de 100 %.

		Clase predicha				
		0	**1**	**2**	**3**	**4**
Clase verdadera	**0**	$> 99\,\%$	$< 1\,\%$	$< 1\,\%$		
	1	$< 1\,\%$	$99\,\%$	$< 1\,\%$		
	2		$< 1\,\%$	$99\,\%$		
	3				$> 99\,\%$	$< 1\,\%$
	4					$100\,\%$

Tabla VIII: Matriz de confusión del modelo SVM con una varianza del 85 %, $\rho = 90$ y $C = 1$. La exactitud obtenida es de 99.9 %. Esta matriz se encuentra etiquetada de la siguiente manera: etiqueta 0 corresponde a saludable, etiqueta 1 corresponde al daño en el nivel 1, etiqueta 2 daño en nivel 2, etiqueta 3 daño en nivel 3 y etiqueta 4 daño en nivel 4. Un cuadrado vacío significa 0 %.

V. Conclusiones

El hecho de probar experimentalmente una metodología basada en la respuesta a la vibración para la monitorización de la integridad estructural, hace viable su aplicación a plataformas *offshore* de tamaño real, donde la excitación obviamente no puede imponerse y a menudo no es medible.

En cuanto a la contribución de este trabajo, se puede destacar la forma como se recopilan, organizan, escalan, transforman y reducen las dimensiones de los datos tridimensionales (procedentes de diferentes tiempos, sensores y experimentos) lo cual nos va a dar la oportunidad de que una muestra contenga mas información en comparación a la información que contienen las muestras a las que no se les realiza este tratamiento. En este documento, esta metodología se especializó para una aplicación específica, la cual se explica en III-A.

En este caso para realizar la detección y localización de daños fue necesario seleccionar un método de clasificación de aprendizaje automático supervisado. Se observó que la selección dependerá en gran parte del tipo de problema que al que nos enfrentemos y de como se encuentren agrupados los datos a tratar como en III-E.

De los dos métodos de clasificación de aprendizaje supervisado que fueron probados, el clasificador SVM cuadrático con $\rho = 90$, $C = 1$ y 443 componentes principales (85 % de varianza) tiene un rendimiento muy cercano al ideal, logrando en todos los indicadores un resultado igual o superior a 99.99 % con una velocidad de predicción muy rápida (1700 obs / seg) y un tiempo de entrenamiento corto (76 seg).

Referencias

[1] Klijnstra, J.; Zhang, X.; van der Putten, S.; Röckmann, C. Technical risks of offshore structures. In Aquaculture Perspective of Multi-Use Sites in the Open Ocean; Springer, Cham, 2017; pp. 115–127.

[2] Fritzen, C.P. Vibration-Based Techniques for Structural Health Monitoring.Structural Health Monitoring 2006, pp. 45–224.

[3] Goyal, D.; Pabla, B. The vibration monitoring methods and signal processing techniques for structural health monitoring: a review.Archives of Computational Methods in Engineering2016,23, 585–594.

[4] Vamvoudakis-Stefanou, K.J.; Sakellariou, J.S.; Fassois, S.D. Output-only statistical time series methods for structural health monitoring: A comparative study. Proceedings of the 7th European Workshop on Structural Health Monitoring.

[5] Farrar, C.R.; Worden, K. An introduction to structural health monitoring.Philosophical Transactions of the389Royal Society A: Mathematical, Physical and Engineering Sciences2006,365, 303–315.

[6] Weijtjens, W.; Verbelen, T.; De Sitter, G.; Devriendt, C. Foundation structural health monitoring of an offshore wind turbine—a full-scale case study.Structural Health Monitoring2016,15, 389–402.

[7] Zugasti Uriguen, E. Design and validation of a methodology for wind energy structures health monitoring. PhD thesis, Universitat Politècnica de Catalunya, 2014.

[8] Pozo, F.; Vidal, Y. Wind turbine fault detection through principal component analysis and statistical hypothesis testing.Energies2016,9, 3.

[9] Stutzmann, J.; Ziegler, L.; Muskulus, M. Fatigue crack detection for lifetime extension of monopile-based offshore wind turbines.Energy Procedia 2017,137, 143–151.

[10] Lian, J.; Cai, O.; Dong, X.; Jiang, Q.; Zhao, Y. Health monitoring and safety evaluation of the offshore wind turbine structure: a review and discussion of future development. Sustainability 2019,11, 494.

[11] Westerhuis, J.A.; Kourti, T.; MacGregor, J.F. Comparing alternative approaches for multivariate statistical analysis of batch process data.Journal of Chemometrics: A Journal of the Chemometrics Society 1999,13, 397–413.

[12] Vidal, Y.; Pozo, F.; Tutivén, C. Wind turbine multi-fault detection and classification based on SCADA data. Energies2018,11, 3018.

[13] Hossin, M.; Sulaiman, M. A review on evaluation metrics for data classification evaluations.International Journal of Data Mining Knowledge Management Process (IJDKP) 2015,5, 1–11

[14] Krüger, F. Activity, Context, and Plan Recognition with Computational Causal Behaviour Models. PhD thesis, University of Rostock, Mecklenburg, Germany, 2016.

[15] Hameed, N.; Hameed, F.; Shabut, A.; Khan, S.; Cirstea, S.; Hossain, A. An Intelligent Computer-Aided Scheme for Classifying Multiple Skin Lesions.Computers 2019,8, 62.

Apéndice C

Participación en congresos

C.1. Congreso Internacional De Robótica y Computación CIRC 2020

El Instituto Tecnológico de La Paz impulsa esta plataforma para intercambiar ideas y discutir desarrollos recientes en Robótica y Computación. CIRC 2020 ofrece un foro de alta calidad y nivel técnico formado por miembros destacados de la comunidad científica y tecnológica [56].

Este congreso se realiza desde el año 2013, realizando en el año 2020 la 7ma edición de éste. Los temas tratados durante el congreso son:

- Soft Computing
- Robótica
- Sistemas de control
- Reconocimiento de patrones
- Computo paralelo
- Visión por computadora
- Bio Informática
- Seguridad
- Redes de computadora
- Computo en la nube

C.1.1. Constancia de asistencia

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

EL TECNOLÓGICO NACIONAL DE MÉXICO

A TRAVÉS DEL INSTITUTO TECNOLÓGICO DE LA PAZ

OTORGA LA PRESENTE

CONSTANCIA

A

Gabriela Aquino González, Yolanda Vidal Seguí, José Eligió Moisés Gutiérrez Arias y Francesc Pozo Montero

Por su trabajo: **Monitorización de la integridad estructural para aerogeneradores offshore con estructura tipo jacket mediante métodos de aprendizaje automático**
presentado en el Séptimo Congreso Internacional de Robótica y Computación

La Paz, B.C.S., a 25 de mayo de 2020.

Atentamente
"CIENCIA ES VERDAD, TÉCNICA ES LIBERTAD"

EDUCACIÓN
INSTITUTO TECNOLÓGICO DE LA PAZ
DIRECCIÓN

ING. JESUS DAVID ESTRADA RUIZ
DIRECTOR

CIRC 2020

Apéndice D

Movilidad Estudiantil

Como parte de los objetivos planteados para desarrollar adecuadamente este trabajo de tesis, se realizó una estancia de seis meses en la Universidad Politécnica de Cataluña (UPC) en Barcelona, España .

La UPC es una institución pública de investigación y educación superior. Es una de las universidades politécnicas líderes de Europa y forma parte de los principales rankings internacionales.

D.1. Reporte

H. Puebla de Zaragoza, 17 de marzo de 2020.

Reporte de actividades durante la estancia de investigación en el laboratorio del grupo CoDALab en la Universidad Politécnica de Cataluña (UPC).

Estudiante: Ing. Gabriela Aquino González.

Programa de posgrado: Maestría en Ciencias de la Electrónica Opción en Automatización.

Institución de procedencia: Benemérita Universidad Autónoma de Puebla (BUAP)

Proyecto de tesis: Desarrollo, validación numérica y experimental de un nuevo método de detección de daños en aerogeneradores Offshore.

Institución receptora: Universidad Politécnica de Cataluña (UPC)

Periodo de estancia: 15 de septiembre de 2019 al 14 de marzo de 2020.

Investigador/a responsable: Dra. Yolanda Vidal Seguí (UPC). Profesora asociada del departamento de matemáticas y responsable del Laboratorio de Control, Dinámica y Aplicaciones (CoDALab) de la Universidad Politécnica de Cataluña.

Objetivo de la estancia: Realizar pruebas de campo en el laboratorio CoDALab, haciendo uso de la estructura a escala de un aerogenerador offshore para validar un nuevo método de detección e identificación de daños utilizando aprendizaje automático.

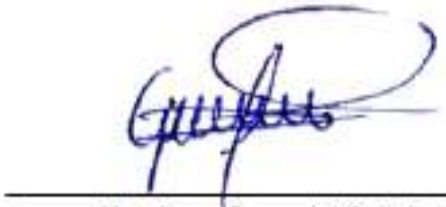
Actividades realizadas:

1. Se estudió el banco de pruebas que se encuentra en el laboratorio CoDALab, el cual incluye la estructura a escala del aerogenerador offshore y los sensores.
2. Se recopilaron los datos a través de ensayos experimentales en el laboratorio.
3. Se hizo una remodelación de los datos.
4. Se entrenaron modelos de clasificación de aprendizaje automático KNN y SVM con los datos remodelados.
5. Se estudiaron diferentes métricas de evaluación de modelos de clasificación para obtener y elegir el modelo con mejor rendimiento que se adecuara a resolver la problemática estudiada.
6. Se realizaron reuniones de trabajo con los investigadores responsables para la elaboración de un artículo científico.

Estas actividades han sido realizadas como parte necesaria y complementaria a las actividades del cronograma propuesto al inicio de la estancia, el cual se muestra a continuación:

Actividad	Octubre 2019 - Diciembre 2019	Enero 2020- Marzo 2020.
Estudio y caracterización de los sensores implementados en el prototipo.	x	
Validación de las técnicas a través de simulaciones numéricas con la herramienta FAST	x	
Validación de las técnicas a través de ensayos experimentales en el laboratorio		x
Escritura de reporte de estancia de investigación	x	x

ATENTAMENTE



Nombre y firma del Estudiante
Gabriela Aquino González

D.2. Carta de termino

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Asunto: Carta de Termino de Estancia

DRA. OLGA GUADALUPE FELIX BELTRAN
COORDINADORA DE LA
MAESTRIA EN CIENCIAS DE LA ELECTRONICA OPCION AUTOMATIZACION
BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA
P R E S E N T E

Por este medio la que suscribe Dra. Yolanda Vidal Seguí responsable del Laboratorio de Control, Dinámica y Aplicaciones (CoDAlab) de la Universidad Politécnica de Cataluña, hago constar que la estudiante Gabriela Aquino González con matricula 218470622 adscrita al programa Maestría en Ciencias de la Electrónica Opción Automatización perteneciente a la Benemérita Universidad de Puebla (Puebla, México), ha concluido satisfactoriamente la estancia de investigación como parte de la tesis "Desarrollo, validación numérica y experimental de un nuevo método de detección de daños en aerogeneradores Offshore". Dicha estancia se realizó en la Universidad Politécnica de Cataluña Campus Diagonal Besòs (Barcelona, España) durante el periodo del 15 de septiembre de 2019 al 14 de marzo de 2020.

Sin más por el momento y agradeciendo la atención prestada a la presente, me despido de usted no sin antes enviarle un cordial saludo.

ATENTAMENTE
Barcelona, España a 14 de marzo del 2020

Dra. Yolanda Vidal Seguí

Bibliografía

[1] Pao, L. Y. and Johnson, K. E. A tutorial on the dynamics and control of wind turbines and wind farms. In *2009 American Control Conference*, pages 2076–2089. IEEE, 2009.

[2] Vidal, Y., Acho, L., Ningsu, L., Zapateiro, M. and Pozo, F. Power control design for variable-speedwind turbines. *Energies*, 5(doi:10.3390/en5083033):3033–3050, aug 2012.

[3] Ribrant, J. and Bertling, L. Survey of failures in wind power systems with focus on swedish wind power plants during 1997–2005. *IEEE Transactions on industrial electronics*, 22(1):167–173, mar 2007.

[4] Ruiz, M., Mujica, L., Alférez, S., Acho, L., Tutivén, C., Vidal, Y., Rodellar, J. and Pozo, F. Wind turbine fault detection and classification by means of image texture analysis. *Elsevier*, (107):149–167, feb 2018. https://doi.org/10.1016/j.ymssp.2017.12.035.

[5] Valente, L., Martinez R., Silveira , M. and Chaves, J. Data-driven fault detection and isolation scheme for a wind turbine benchmark. *Elsevier*, (87):634–645, nov 2015.

[6] Gong, X. and Qiao, W. Current-based mechanical fault detection for direct-drive wind turbines via synchronous sampling and impulse detection. *IEEE Transactions on industrial electronics*, 62(3), mar 2015.

[7] Fritzen, C.P. Vibration-based techniques for structural health monitoring. *Structural Health Monitoring*, pages 45–224, 2006.

[8] Goyal, D. and Pabla, B.S. The vibration monitoring methods and signal processing techniques for structural health monitoring: a review. *Archives of Computational Methods in Engineering*, 23(4):585–594, 2016.

[9] Fassois, S.D. and Sakellariou, J.S. Time-series methods for fault detection and identification in vibrating structures. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1851):411–448, 2006.

[10] Vamvoudakis-Stefanou, K.J., Sakellariou, J.S. and Fassois, S.D. Output-only statistical time series methods for structural health monitoring: A comparative study. In *Proceedings of the 7th European Workshop on Structural Health Monitoring.*

[11] Martinez-Luengo, M., Kolios, A. and Wang, L. Structural health monitoring of offshore wind turbines: A review through the statistical pattern recognition paradigm. *Renewable and Sustainable Energy Reviews*, 64:91–105, 2016.

[12] Mieloszyk, M. and Ostachowicz, W. An application of structural health monitoring system based on fbg sensors to offshore wind turbine support structure model. *Marine Structures*, 51:65–86, 2017.

[13] Fritzen, C.P., Kraemer, P. and Klinkov, M. An integrated shm approach for offshore wind energy plants. In *Structural Dynamics, Volume 3*, pages 727–740. Springer, 2011.

[14] Weijtjens, W., Verbelen, T., De Sitter, G. and Devriendt, C. Foundation structural health monitoring of an offshore wind turbine—a full-scale case study. *Structural Health Monitoring*, 15(4):389–402, 2016.

[15] Mojtahedi, A., Yaghin, M.L., Hassanzadeh, Y., Ettefagh, M.M., Aminfar, M.H., and Aghdam, A.B. Developing a robust shm method for offshore jacket platform using model updating and fuzzy logic system. *Applied Ocean Research*, 33(4):398–411, 2011.

[16] Zugasti, E. *Design and validation of a methodology for wind energy structures health monitoring.* PhD thesis, Universitat Politècnica de Catalunya, 2014.

[17] Gutiérrez, J. *Desarrollo de herramientas software para el análisis de aerogeneradores offshore sometidos a cargas acopladas de viento y oleaje.* Doctorado, Universidad Politécnica de Cartagena, Colombia, may 2014.

[18] Iberdrola, S.A. Qué es la energía eólica marina, jun 2020.

[19] Ruíz, R. Diseño mecánico de un aerogenerador eólico. Master, Escuela Universitaria de Ingenieria Tecnica Industrial de Bilbao., España, nov 2015.

[20] Portillo, G. Aerogenerador vertical, jun 2017.

[21] González, A. *Algoritmos de generación de consigna de velocidad angular y ajuste del control de velocidad en aerogeneradores de media potencia.* Doctorado., Universidad del País Vasco, España, 2016.

[22] Jonkman, J.M . and Buhl, M.L. *FAST User's Guide. NREL/EL-500-38230 (previously NREL/EL-500-29798),.* National Renewable Energy Laboratory, aug 2005.

[23] Jonkman, B. and Jonkman, J. *FAST v8.16.00a-bjj*. National Renewable Energy Laboratory, jun 2016.

[24] Jonkman, J.M., Butterfield, S., Musial, W. and Scott, G. Definition of a 5-mw reference wind turbine for offshore system development. Technical Report Technical Report NREL/TP-500-38060., National Renewable Energy Laboratory: Golden, CO,, USA, 2009.

[25] Murphy, K.P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[26] Stetco, A., Dinmohammadi, F., Zhao, X., Robu, V., Flynn, D., Barnes, M., Keane, J. and Nenadic, G. Machine learning methods for wind turbine condition monitoring: A review. *Renewable energy*, 2018.

[27] Inc. The MathWorks. *Introducing Machine Learning*. The MathWorks, Inc., 2016.

[28] Chow, J., Dom. B. and Lin, D. *patente*. Yahoo! Inc.,, Sunnyvale, CA (US), dec 2017.

[29] Matlab. Assess classifier performance in classification learner, jun 2020.

[30] Beauxis-Aussalet, E. and Hardman, L. Visualization of confusion matrix for non-expert users. In *IEEE Conference on Visual Analytics Science and Technology (VAST)-Poster Proceedings*, 2014.

[31] Danhauer, J.L. and Lucks, L.E. The confusion matrix: A new model. *Canadian Journal of Speech-Language Pathology and Audiology*, 11(4):7–11, 1987.

[32] Obuchowski, N. Roc analysis. *American Journal of Roentgenology*, 184(2):364–372, feb 2005.

[33] Rondero, C. and Font, V. Articulación de la complejidad matemática de la media aritmética. *Enseñanza de las ciencias: revista de investigación y experiencias didácticas*, pages 29–49, 2015.

[34] Peña, D. *Fundamentos de estadística*. Alianza editorial, 2014.

[35] Walpole, R.E., Myers, R.H., Myers, S.L. and year=1992 publisher=McGraw-Hill México Cruz, R., volume=624. *Probabilidad y estadística*.

[36] Inc. Wolfram Research. Covarianza, may 2019.

[37] Stutzmann, J., Ziegler, L. and Muskulus, M. Fatigue crack detection for lifetime extension of monopile-based offshore wind turbines. *Energy Procedia*.

[38] National Instruments. *NI cDAQTM-9188XT. User Manual*. National Instruments, mar 2016.

[39] National Instruments. *Datasheet NI 9234*. National Instruments, oct 2015.

[40] Inc. The MathWorks. Classification learner, may 2019.

[41] Inc. The MathWorks. Select data and validation for classification problem, may 2019.

[42] Lian, J., Cai, O., Dong, X., Jiang, Q. and Zhao, Y. Health monitoring and safety evaluation of the offshore wind turbine structure: a review and discussion of future development. *Sustainability*, 11(2):494, 2019.

[43] Pozo F. Tutivén C. Vidal, Y. Wind turbine multi-fault detection and classification based on scada data. *Energies*, pages 2–18, nov 2018.

[44] Westerhuis, J.A., Kourti, T. and MacGregor, J.F. Comparing alternative approaches for multivariate statistical analysis of batch process data. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 13(3-4):397–413, 1999.

[45] Vidal, Y., Aquino, G., Pozo F. and Gutiérrez-Arias J.E.M. Structural health monitoring for jacket-type offshore wind turbines: Experimental proof of concept. *Sensors*, page 20, 2020.

[46] González, C.G. and Felpeto, A.B. *Tratamiento de datos*. Ediciones Díaz de Santos, 2006.

[47] James, G., Witten, D., Hastie, T. and Tibshirani, R. *An introduction to statistical learning*, volume 112. Springer, 2013.

[48] Yang, J.; Sun, Z.; Chen, Y. . Fault detection using the clustering-knn rule for gas sensor arrays. *Sensors*, page 21, nov 2016.

[49] Pozo, F. Data analysis & pattern recognition. performance evaluation. nov 2019.

[50] Pozo, F., Vidal, Y. and Salgado, O. Wind turbine condition monitoring strategy through multiway pca and multivariate inference. *Energies*, 11(4):749, 2018.

[51] Hossin, M. and Sulaiman, M.N. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, 5(2):1–11, mar 2015.

[52] Krüger, Frank. *Activity, Context, and Plan Recognition with Computational Causal Behaviour Models*. PhD thesis, University of Rostock, Mecklenburg, Germany, 2016.

[53] Hameed, N., Hameed, F., Shabut, A., Khan, S., Cirstea, S. and Hossain, A. An intelligent computer-aided scheme for classifying multiple skin lesions. *Computers*, 8(3):62, 2019.

[54] Sokolova, M. and Lapalme, G. A systematic analysis of performance measures for classification tasks. *Elsevier*, 2009.

[55] Sensors — open access journal, jun 2020, https://www.mdpi.com/journal/sensors.

[56] Vll congreso internacional de robótica y computación, may 2020, http://posgrado.lapaz.tecnm.mx/circ2020/http://posgrado.lapaz.tecnm.mx/circ2020/.