UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

MANTENIMIENTO PREDICTIVO EN GENERADORES DE AIRBUS, UTILIZANDO APRENDIZAJE PROFUNDO

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MECÁNICO

BASTIÁN IGNACIO FUENZA MÉNDEZ

PROFESOR GUÍA:
ENRIQUE LÓPEZ DROGUETT

MIEMBROS DE LA COMISIÓN:
VIVIANA MERUANE NARANJO
RODRIGO PASCUAL JIMÉNEZ

SANTIAGO, CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL MECÁNICO
POR: **BASTIÁN IGNACIO FUENZA MÉNDEZ**
FECHA: 2020
PROF. GUÍA: ENRIQUE LÓPEZ DROGUETT

MANTENIMIENTO PREDICTIVO EN GENERADORES DE AIRBUS, UTILIZANDO APRENDIZAJE PROFUNDO

La industria avanza a la llamada Industria 4.0, donde se integran las partes físicas y digitales de las maquinarias, y el mantenimiento predictivo, encargado de determinar cuándo es necesario realizar reemplazos y reparaciones, puede hacer uso de esto. El Aprendizaje profundo, una técnica potente y novedosa, es ideal para trabajar las grandes cantidades de datos que se producen de esta forma, por lo que se utilizará en el mantenimiento predictivo de generadores IDG (Integrated Drive Generator) de aviones Airbus.

Para esto, se analizan los datos recibidos, y procesan para su mejor utilización, luego se entrenan clasificadores dedicados a identificar la existencia de una falla incipiente, y finalmente, se entrenan regresores que intentan predecir el tiempo de vida útil remanente (RUL). Esto se realiza utilizando técnicas de Aprendizaje Profundo; y en particular, en el lenguaje de programación Python, con librerías como Numpy, Pandas, Sklearn, Tensorflow(Keras), entre otras.

Se tienen 17.1GB de datos, correspondiente al funcionamiento por 24 semanas aproximadamente, de 5 IDG distintos, 2 de estos con una falla funcional al final de la grabación, confirmada por la empresa; éstos tienen una tasa de muestreo de aproximadamente 7Hz, pero con delta de tiempo no constante, lo que dificulta el análisis frecuencial. El cálculo de la diferencia de temperatura entre la entrada y salida del aceite, nombrada DtTemp, muestra claros cambios hacia el final de las grabaciones de los 2 IDG con falla, y resulta ser útil en los resultados.

Se logra obtener un modelo clasificador de Aprendizaje Profundo capaz de detectar la falla incipiente en los 2 generadores con una anticipación de 2 a 3 semanas antes de que la falla funcional ocurra. También se entrenan modelos regresores, pero estos no obtienen resultados útiles; se teoriza que esto puede deberse a que los modelos siguen una forma similar a DtTemp, el cual no cambia de forma paulatina; o a que el método de construcción de RUL utilizado no es suficiente para esto.

Finalmente, se concluye que 2 IDG (Nombres AHE2 y AJG1) presentan anomalías que no corresponden a una falla detectable con los datos, un 3ro (AJF1) no presenta cambios considerables, y los otros 2 IDG (AHD1 y AHE1), con falla confirmada, probablemente presentaron una falla del mismo tipo, la cual puede ser detectada con 2 a 3 semanas de anticipación por el clasificador; el cual puede ser utilizado como un detector de anomalías para equipos IDG, entregando un aviso cuando se detecte una falla incipiente para la que fue entrenado.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Antecedentes generales

El mantenimiento es una actividad muy importante dentro de la industria, ya que busca asegurar que los sistemas mecánicos y maquinarias funcionen correctamente, una falla en el sistema puede resultar en una baja de rendimiento o destrucción de maquinaria, lo que conlleva una pérdida de recursos materiales, y de ganancias por el detenimiento de producción [1].

Ya que todas las maquinarias tienen vida limitada, se busca estrategias de mantenimiento que aumenten la vida útil y reduzcan las perdidas por fallas. Una de las formas de categorizar las técnicas de mantenimiento es la siguiente [1]:

- Mantenimiento Correctivo: El componente se deja funcionar hasta la falla, y luego se reemplaza.
- Mantenimiento Preventivo: Se realizan reemplazos o reparaciones programadas en base al tiempo, para prevenir la falla.
- Mantenimiento Predictivo: Se utilizan herramientas predictivas para determinar cuándo es necesario realizar reemplazos o reparaciones.

Mantenimiento Predictivo es el método que pretende prolongar la vida útil de los componentes, al tomar acciones solo cuando es necesario. Y una de las herramientas predictivas potentes y novedosas corresponde al uso de datos de funcionamiento en modelos de Aprendizaje de Maquinas o Aprendizaje Profundo [1].

En esta oportunidad se cuenta con datos del funcionamiento hasta la falla de un generador de avión Airbus, los cuales serán utilizados en este trabajo de título.

1.2. Motivación

La industria avanza a lo que algunos expertos llaman la 'Cuarta revolución industrial' o Industria 4.0, donde se integran las partes físicas y digitales de maquinarias, resultando en una gran recolección de datos de funcionamiento [1]. El Aprendizaje Profundo es capaz de grandes cantidades datos, y entregar modelos útiles que podrían ser usados para mejorar las estrategias predictivas en distintos sistemas, potencialmente reduciendo costos asociados a la pérdida o reemplazo innecesario de elementos mecánicos, que resulta en una ineficiencia de utilización de recursos. La búsqueda de modelos capaces de mejorar los métodos actuales de mantenimiento es lo que motiva este trabajo.

1.3. Objetivos

Los objetivos de este trabajo pueden subdividirse en dos categorías:

1.3.1. Objetivo general

- Utilizar los datos de funcionamiento de IDG entregados por la empresa para proponer un modelo de Aprendizaje Profundo, en el contexto de mantenimiento predictivo, que haga uso de ellos.

1.3.2. Objetivos específicos

- Encontrar relaciones en el comportamiento de los datos a lo largo de las grabaciones.
- Obtener modelos de Aprendizaje Profundo que utilicen los datos.
- Concluir respecto al desempeño del mejor modelo creado.

1.4. Alcances

Se utilizarán los datos entregados del generador de Airbus para crear un modelo de Aprendizaje Profundo en el contexto de mantenimiento predictivo.

Sí la calidad y usabilidad de los datos es limitante, o si estos resultan ser inútiles, no será posible la creación de un modelo útil para el mantenimiento predictivo del generador, por lo que no se puede garantizar la creación de uno. Además, la evaluación del desempeño del modelo se limitará al modelo de Aprendizaje Profundo en sí, sobre los datos; no incluirá de ninguna forma las estrategias de mantenimiento de la empresa involucrada.

Capítulo 2

Antecedentes y discusión bibliográfica

2.1. Aprendizaje de Máquinas y Aprendizaje Profundo

El Aprendizaje de Máquinas puede considerarse como un subgrupo de la inteligencia artificial, este se diferencia o caracteriza por agrupar técnicas de programación que permiten la aproximación paulatina a una función objetivo a través del entrenamiento con datos, en otras palabras, el aprendizaje a través del entrenamiento; se dice que: 'un computador aprende de la experiencia E con respecto a alguna clase de tareas T y desempeño P, si su desempeño en tareas en T, medido por P, mejora con la experiencia E' [2]. Estas técnicas tienen la gran ventaja de no requerir una programación explicita para el trabajo numérico que realiza el programa con los datos, sino que se le da un marco o arquitectura al programa, el cual se adaptara a través de la experiencia.

Dentro de la categoría de Aprendizaje de Máquinas, existen distintos subgrupos, a la vez que existen diferentes formas de dividirlos; las categorías más relevantes son aprendizaje *supervisado*, *semi-supervisado*, *no supervisado* y *por refuerzo*[3]; se describe la primera a continuación:

- **Aprendizaje supervisado:** Se puede entender el programa como una caja negra que recibe una entrada 'x' y la trabaja numéricamente hasta una salida 'y'; el objetivo del aprendizaje supervisado es que la caja negra logre aprender una relación entre 'x' e 'y' de los datos del entrenamiento, que la lleve a ser capaz de recibir 'x' y producir un valor 'y' con el menor error posible respecto al 'y' deseado. Matemáticamente, se disponen de un set de datos de entrada-salida $\{(x_i, y_i)\}_{i=1}^{N}$ que se utilizaran para encontrar una función $f : x \longrightarrow y$ tal que el error $E(f(x_i), y_i)$ sea el menor posible. En este contexto, 'x' se conoce como 'características' o 'features', e 'y' se conoce como 'etiqueta' o 'label' [3]; a priori, tanto las características como las etiquetas pueden ser datos en cualquier formato, sea numérico o no, pero en la práctica estos deben ser ingresados como números al algoritmo, por lo que son codificados como vectores o matrices numéricas.

  Ejemplos orientados a mantenimiento, podrían ser: la utilización de termografías para determinar una condición de funcionamiento de una máquina, donde las imágenes serían las características, y la condición, la etiqueta; o también, la utilización de señales de vibración mecánica para pronosticar la vida útil remanente de una maquina; distintos sistemas mecánicos se han estudiado de estas maneras, como: turbofans, bombas hidráulicas, cajas de cambio de automóviles, etc [1].

El Aprendizaje Profundo es un subgrupo de Aprendizaje de Máquinas que se caracteriza por utilizar modelos capaces de aprender jerarquías de conceptos, donde cada concepto está definido por conceptos más simples, los cuales pueden graficarse como capas construidas sobre otras; esta forma de aprendizaje no requiere la integración de todo el conocimiento humano al modelo [4]. El aprendizaje profundo utiliza una familia de modelos específica, llamadas redes neuronales artificiales, que se describirá a continuación.

2.2. Redes neuronales artificiales

Las redes neuronales artificiales (ANN o NN) son una serie de modelos usados en aprendizaje de máquinas que se basan en la biología del cerebro, intentan reproducir artificialmente las funciones básicas de las neuronas y sus interconexiones, donde cada neurona artificial actúa como una celda que recibe información de otras neuronas, la procesa, y la entrega. Las NN son utilizadas por ser versátiles, poderosas y escalables, haciéndolas ideales para problemas complejos y con alta cantidad de datos [7].

La unidad básica de una ANN es la neurona artificial, la cual esta descrita por lo siguiente:

$$f(x) = g(x^T w + b) \tag{2.1}$$

Donde $f(x)$ es una función escalar, $x = \{x_1, x_2, ..., x_n\}$ es un vector con los datos de entrada, b es el valor 'bias' que no depende de la entrada, $w = \{w_1, w_2, ..., w_n\}$ son los 'pesos' de cada conexión de la neurona, y g es la 'función de activación' que se discutirá más adelante. Los pesos w y b son las únicas variables en la neurona, y por ello son las variables que se optimizan en el entrenamiento de la red [7].

Estas neuronas se ordenan en capas de neuronas paralelas; además, desde ahora el termino bias b será incluido dentro de w como w_0 al agregar $x_0 = 1$ a x. De esta forma, la ecuación anterior se extiende a una capa:

$$f^{(i)}(x) = g^{(i)}({x^{(i)}}^T w^{(i)}) \tag{2.2}$$

Donde $f^{(i)}(x)$ es una función vectorial, i indica el numero de la capa a la que corresponde; además, en este caso $x^{(i)}$ solo es la entrada externa para el caso $i = 1$, para los demás, $x^{(i)}$ es el vector resultante de $f^{(i-1)}$, a veces llamado $z^{(i)}$.

Las ANN están compuestas de distintas capas, y el flujo de información comienza en una capa inicial que recibe una entrada x del entorno, pasa a la siguiente capa, luego a la siguiente, y así hasta la salida; esto se conoce como 'Propagación hacia adelante' o 'Forward propagation'. Para una ANN de 3 capas, esto sería:

$$f_{ANN}(x) = f^{(3)}(f^{(2)}(f^{(1)}(x))) \tag{2.3}$$

Finalmente, las capas se distinguen de la siguiente manera: La primera capa de una ANN es llamada 'capa de entrada'; la última capa, 'capa de salida'; y todas las capas intermedias son llamadas 'capas ocultas'.

2.2.1. Funciones de activación

Para que el algoritmo funcione apropiadamente se requiere de funciones de activación, si estas fuesen la función identidad, la ANN resultaría en una simple función lineal [4], lo cual puede comprobarse de las ecuaciones (2) y (3); pero la función de activación además requiere del cálculo del gradiente (ver 'Entrenamiento de una red neuronal'), por lo que no puede ser una función como la de 'escalón' [7]. Existen diversas funciones de activación, cada una con utilidades particulares; a continuación, se mencionarán algunas relevantes, agrupadas en 'funciones de capas ocultas' y 'funciones de capa de salida', debido a sus diferencias funcionales.

2.2.1.1. Funciones de capas ocultas

Estas funciones de activación son las responsables de que la ANN sea capaz de aprender funciones no-lineales complejas. Existen numerosas funciones en la literatura, y todas son compatibles con el entrenamiento de una ANN, sin embargo, cada una tiene pequeñas particularidades [7]. A continuación, se describen 3 de las funciones más relevantes:

- Logística o Sigmoid: Nace de un enfoque probabilístico, al calcular la densidad posterior utilizando el Teorema de Bayes en clasificación binaria [6]. Es una de las funciones más básicas y clásicas, tiene la cualidad de ser continua y diferenciable en todo su dominio, converger en los extremos, y estar acotada entre 0 y 1.

$$g(a) = \frac{1}{1 + e^{-a}} \tag{2.4}$$

- Tangente hiperbólica: Similar a Sigmoid, también es continua, diferenciable en todo el dominio, converge en los extremos, pero esta acotada entre -1 y 1. Este rango suele hacer que las salidas de capas tiendan a estar cerca del 0, lo que ayuda a apresurar la convergencia [7].

$$g(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}} \tag{2.5}$$

- ReLU: Rectified Linear Unit, es continua como las anteriores, pero no diferenciable en $a = 0$, además de tener derivada 0 para $a < 0$, y solo esta acotada inferiormente en 0. A pesar de parecer poco útil, funciona bien y tiene la gran ventaja de ser rápida de calcular computacionalmente, lo cual resulta en entrenamientos menos demorosos, por lo que se ha convertido en el estándar [7].

$$g(a) = max(0, a) \tag{2.6}$$

2.2.1.2. Funciones de capa de salida

Estas funciones de activación determinan la salida final de la red, y por esto son muy dependientes del tipo de problema (clasificación o regresión), como de si existe un rango de valores deseado.

En problemas de regresión el estándar es utilizar la función identidad en la última capa, ya que normalmente esta pretende entregar la predicción de uno o más valores reales; pero existen casos donde se quieren limitar los valores de salida, por ejemplo [7]:

- Limite $(0, \infty)$: Se puede utilizar la función ReLU u otras similares, para prevenir que la red entregue valores negativos; esto es útil en situaciones donde un numero negativo no tenga sentido, como una predicción de n^o de habitaciones, peso, tiempo hasta la falla (RUL), etc.
- Limite $(0, n)$: Se puede utilizar la función Sigmoid o similares, ya que puede escalarse; puede ser particularmente útil para la predicción de probabilidades con $n = 1$, o también para representar cualquier positivo limitado.
- Limite $(-n, n)$: Se puede utilizar la función tangente hiperbólica o una similar, ya que al igual que la anterior, puede escalarse a cualquier rango; es útil para representar un rango definido, pero que incluya números negativos.

2.3. Entrenamiento de una red neuronal

El objetivo del entrenamiento de una red neuronal que aproxime la función que relaciona 'x' e 'y' de la mejor manera posible, esto es medido a través de alguna clase de métrica de 'error'; para esto se deben encontrar los valores de las variables, es decir, los pesos de la red, que minimicen el error de esta:

$$W_{op} = \underset{W}{argmin} E(W, X) \tag{2.7}$$

Donde W son los pesos de la red completa, X es la matriz que contiene todas las características para todos los datos, E la función de error, y W_{op} son los pesos óptimos que minimizan el error.

Debido a que la red neuronal es, en la práctica, una función muy no-lineal, la búsqueda de los W que minimicen el error no es sencilla, y se debe recurrir a métodos iterativos [4]; y el más básico y característico, es el Gradiente Descendente[7].

2.3.1. Gradiente Descendente

El Gradiente Descendente corresponde a un método estocástico de optimización que utiliza el gradiente de una función para encontrar la dirección de mayor descenso, y la utiliza para acercarse al mínimo [3]. Cada actualización del método estocástico se realiza de acuerdo a la siguiente ecuación:

$$W_{(i+1)} = W_{(i)} - \eta \nabla E(W_{(i)}, X) \tag{2.8}$$

Donde en este caso se ha utilizado los índices entre paréntesis para mostrar el número de iteración en la optimización; además, η corresponde a la taza de aprendizaje, la cual es una constante.

En este caso, el gradiente puede ser descrito como:

$$\nabla E = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, ..., \frac{\partial E}{\partial w_n} \right]^T \tag{2.9}$$

Pero esto muestra que, para obtener el gradiente, y actualizar todos los pesos de la red, se requiere de los gradientes para cada uno de estos. Esto se puede lograr con el método conocido como 'propagación hacia atrás' o 'back-propagation'.

2.3.2. Propagación hacia atrás (Back-Propagation)

El método de back-propagation permite calcular la derivada parcial del error al peso de interés, y es equivalente a realizar una regla de la cadena desde el error en la salida hasta todo el resto de la red [4].

Tal como el nombre lo sugiere, back-propagation es inverso a forward propagation; si esta última corresponde a tomar la entrada, y pasarla por los cálculos de cada capa, hasta la salida; back-propagation consiste el tomar el error $E(W, X)$ de la red en la salida, y pasarla por los cálculos de cada etapa, hasta la entrada. El proceso exacto se encuentra en [5].

2.3.3. Métodos de optimización modernos

El método de Gradiente Descendente es uno de los más antiguos y básicos, en la actualidad existen otros métodos que son capaces de realizar entrenamientos mucho más rápidos y alcanzando mejores resultados [7]. A continuación, se presentan brevemente los más utilizados, puede encontrarse más detalles en [7]:

- Momentum: Introduce el concepto de 'momentum' en Gradiente Descendente, que genera un aumento en la distancia de cada paso si la dirección del gradiente es relativamente constante.
- AdaGrad: Introduce la taza de aprendizaje adaptable, que acumula las dimensiones del gradiente, y disminuye la taza de aprendizaje en las dimensiones más empinadas.
- RMSProp: Mejora problemas de AdaGrad al darle más importancia a los gradientes más recientes.
- Adam: Combina RMSProp con Momentum, generando una rápida convergencia con taza de aprendizaje adaptable. Se ha convertido en el método de optimización estándar [7].

2.4. Métricas de desempeño

Las métricas de desempeño permiten obtener un valor que puede ser utilizado para evaluar la calidad de los resultados de una red, y conocer qué tan buena es. Existen métricas distintas para clasificación y para regresión, a continuación se destacan algunas de estas.

Dos métricas de clasificación utilizadas son:

- Accuracy: Corresponde a la fracción de etiquetas predichas que aciertan a la etiqueta real. Suele no ser la métrica preferida para clasificación, ya que puede entregar resultados engañosos en algunos casos [7].

$$Accuracy(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} 1(y_i = \hat{y}_i) \tag{2.10}$$

- F1 score: Corresponde a la media armónica de las medidas precision y recall, las cuales representan el accuracy de las predicciones positivas, y la fracción de instancias positivas correctamente detectadas, respectivamente. Suele ser la métrica preferida en clasificación, ya que elimina varios problemas de accuracy [7].

$$F1 - score(y, \hat{y}) = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.11}$$

En problemas de regresión, dos de las más utilizadas son [7]:

- RMSE: Corresponde a la norma $\|\cdot\|_2$ o 'Euclidiana', y es la función de costo estándar para problemas de regresión [7].

$$RMSE(x, z) = \sqrt{\frac{\sum_{i=1}^{N}(x_i - z_i)^2}{N}} \tag{2.12}$$

- MAE: Corresponde a la norma $\|\cdot\|_1$ o 'Manhattan', y se diferencia del RMSE al ser menos afectado por valores atípicos extremos, minimizando diferencias pequeñas y grandes de manera más pareja [7].

$$MAE(x, z) = \frac{\sum_{i=1}^{N}|x_i - z_i|}{N} \tag{2.13}$$

2.5. Perceptrón multicapa (MLP)

Multilayer Perceptron (MLP) es la familia más básica de modelos de ANN, estas consisten en una capa de entrada, una capa de salida, y una o más capas ocultas; estas llevan este nombre debido a estar compuestas de múltiples capas de perceptrones (y no por tener perceptrones de varias capas), la unidad que actúa de neurona artificial [7].

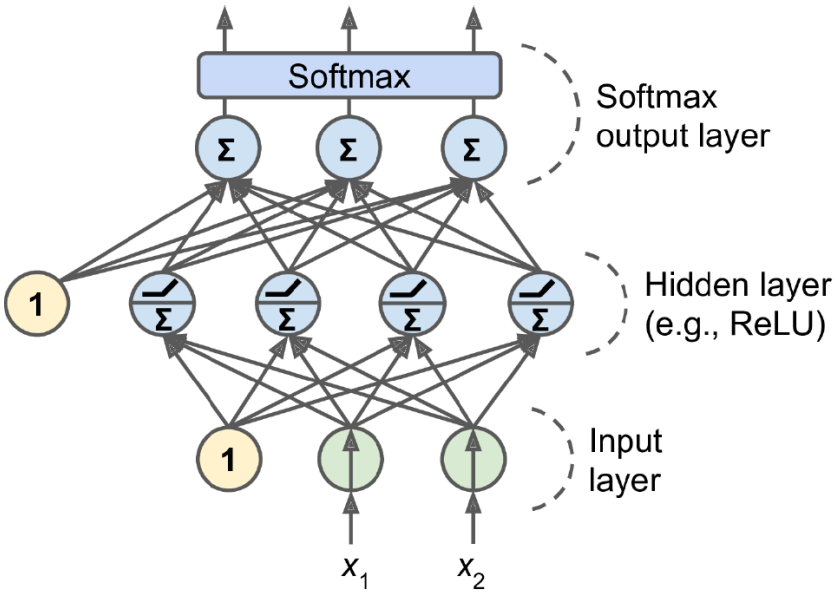

Figura 2.1: Ejemplo de arquitectura de una MLP. *Imagen tomada de [7]*

Los modelos MLP reciben un vector como entrada de datos, pero ya que en esta memoria se requiere trabajar con matrices donde una de las dimensiones es 'tiempo', se puede entregar cada instante de tiempo como un vector a la red, y luego entregar un solo resultado para todos los instantes de tiempo; esto se logra utilizando capas 'time distributed', por lo que el modelo que las utilice será llamado Time distributed MLP o TD-MLP, es esta memoria.

2.6. Redes neuronales convolucionales (CNN)

Convolutional Neural Networks (CNN) es una familia de ANN que se caracteriza por utilizar capas compuestas de operaciones convolucionales, las cuales tienen la ventaja de trabajar segmentos pequeños de una matriz; logrando extraer características locales primero, y luego construyendo características más generales en capas siguientes. Estas redes se caracterizan por su utilidad en reconocimiento de imágenes, pero también logran buenos resultados en aplicaciones como el análisis de datos temporales, o procesamiento de lenguaje natural [7].

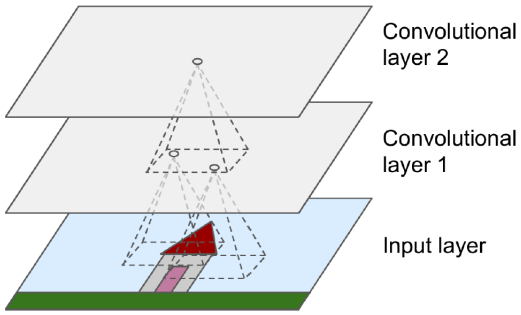

Figura 2.2: Ejemplo de capas convolucionales en una imagen. *Imagen tomada de [7]*

2.7. Redes neuronales recurrentes (RNN)

Recurrent Neural Networks (RNN) es una familia de ANN que se caracteriza por utilizar capas con memoria, y por poder trabajar con secuencias temporales de largo arbitrario; estas llevan el nombre 'recurrente' debido a utilizar capas especiales que reciben un vector a la vez por cada instante temporal, que luego utilizan su propio output (o uno de sus outputs) como input para el siguiente instante, en otras palabras, se 'alimentan' a sí mismas [7].

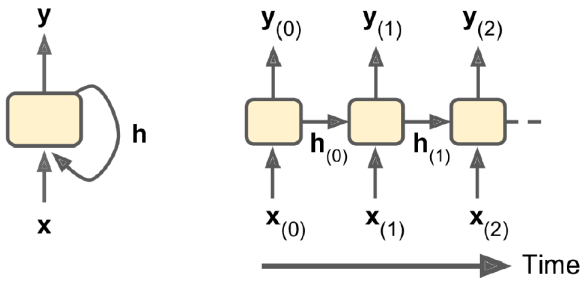

Figura 2.3: Ejemplo de capa recurrente en el tiempo. *Imagen tomada de [7]*

Capítulo 3

Metodología

Se utilizará el lenguaje de programación Python, usando con librerías nativas y terceras de éste; tales como: Numpy, Pandas, Sklearn, Tensorflow(Keras), entre otras. Y la metodología para lograr el trabajo consiste en la siguiente:

3.1. Análisis de datos

Se debe comprender apropiadamente los datos para tomar decisiones sobre el procesamiento de estos; información como: variables medidas, tasa de muestreo, cantidad de datos, etc. son de gran importancia a la hora de tomar decisiones. Se buscará información que pueda resultar útil para decidir qué sensores utilizar, calcular relaciones entre sensores, etc.

3.2. Procesamiento

Se evaluarán alternativas de 'Extracción de características', para extraer información considerada relevante de los datos; y qué sensores serán utilizados para los entrenamientos.

3.3. Clasificación

Se utilizarán los datos procesados en distintos modelos para definir los tipos de procesamiento ideales para el resto de los entrenamientos. Luego, estos se utilizarán para obtener una detección temprana de la falla incipiente, y los mejores resultados de clasificación posibles.

3.4. Regresión

Se intentará mejorar la utilidad de los resultados anteriores a través de la realización de regresión; para esto se construirá una etiqueta de RUL para los datos. Predicción con regresión es más útil a la hora de tomar decisiones de mantenimiento, pero resulta más difícil obtener buenos resultados de entrenamiento para regresión, por lo que se realizará o no dependiendo de los resultados de clasificación.

Capítulo 4

Desarrollo

En esta sección se presenta el desarrollo de todo el trabajo realizado, el cual sigue la metodología mostrada anteriormente.

4.1. Análisis de datos

4.1.1. Descripción de datos

Los datos que se poseen para el trabajo son 5 archivos con datos del funcionamiento de 5 generadores IDG distintos, donde el dispositivo de grabación se enciende junto al avión, haciendo un total de datos de 17,1GB.

Cada generador tiene una etiqueta de nombre: AHD1, AHE1, AHE2, AJF1 y AJG1, pero debido a la similitud de los nombres, se les llamará G1, G2, G3, G4 y G5, respectivamente para evitar confusiones. Además, cada archivo de generador tiene una cantidad de datos diferente, pero con 24 semanas de vuelos en promedio.

De acuerdo a lo informado por la empresa, los generadores G1 y G2 presentaron una falla funcional al final de la grabación de datos, mientras que para los demás generadores, no está confirmado.

Los datos tienen una frecuencia de muestreo de aproximadamente 7Hz (ya que en promedio existen 7 datos por segundo), pero al realizar una inspección en detalle, se nota que el lapso de tiempo entre medidas no es constante, de la siguiente manera:

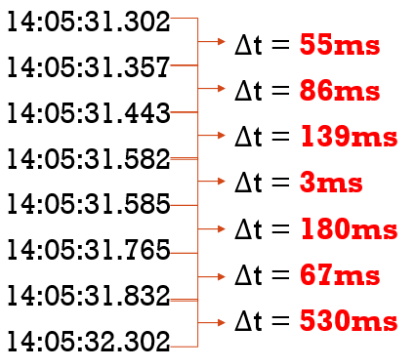

Figura 4.1: Delta tiempo no constante.

El delta tiempo no constante en los datos hace dificulta de gran manera el realizar análisis frecuencial, lo que limita el tipo de preprocesamiento que puede realizarse.

En cuanto a los atributos de datos que se poseen, son los 8 siguientes:

- Timestamp -> Fecha y tiempo del instante de medición
- InTemp -> T° de entrada de aceite
- OutTemp -> T° de salida de aceite
- Altitude -> Altitud del avión
- VibrationN2 -> Vibración RMS de Sistema acoplado a IDG
- N2Speed -> Velocidad de giro de eje
- Frequency -> Frecuencia AC de generador
- Voltage -> Voltaje AC de generador

A continuación, se grafica todo el tiempo grabado para G1, separado en cada una de las clases de datos (excepto Timestamp), con la finalidad de analizar la forma de estos. Las siguientes figuras incluyen un gráfico de todos los datos, y dos gráficos pequeños con un zoom en 2 vuelos al inicio y final de la grabación, para más detalle; en estos gráficos de zoom, la línea vertical blanca denota aproximadamente la separación entre ambos vuelos, la cual se mantiene para todas estas figuras.

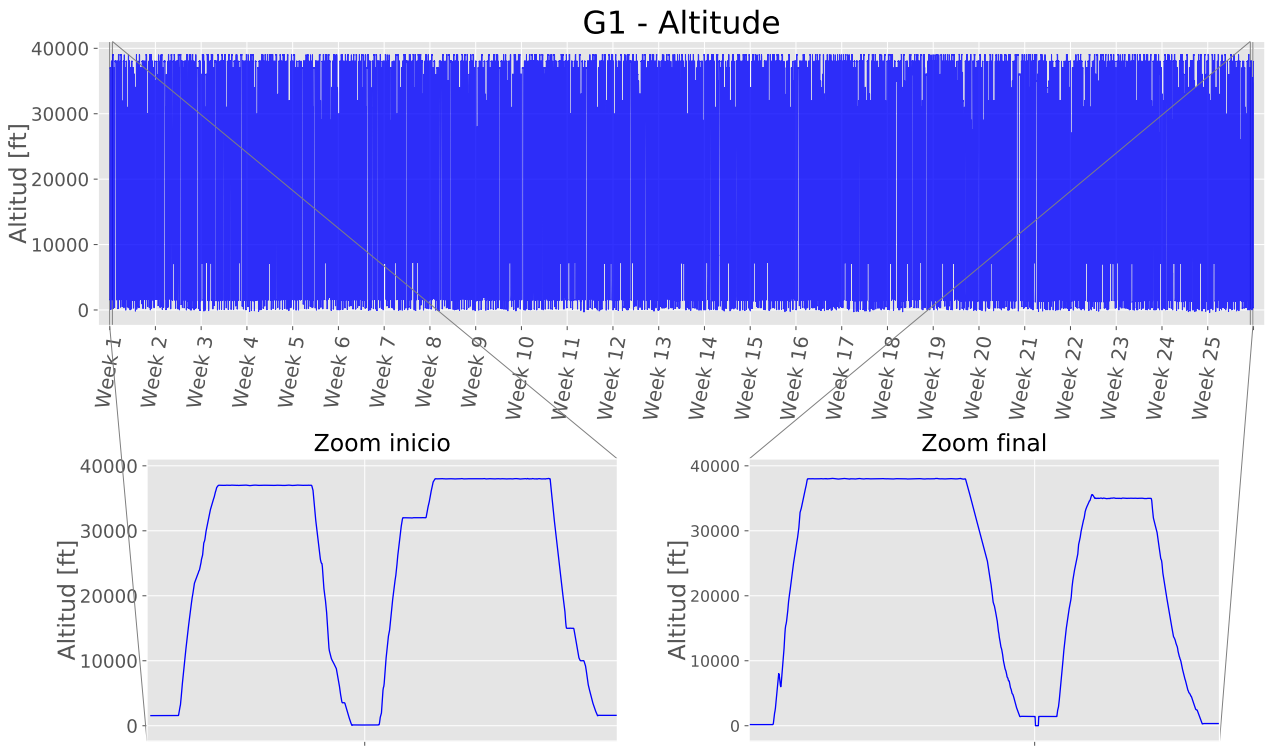

Figura 4.2: Gráfico de datos de altura de avión.

En la Figura 4.2 se puede notar levemente que existen vuelos a más altura que otros, pero que la mayoría llega a una altura de al menos 30000 pies; en los gráficos de zoom se nota que el tiempo de vuelo también es variable.

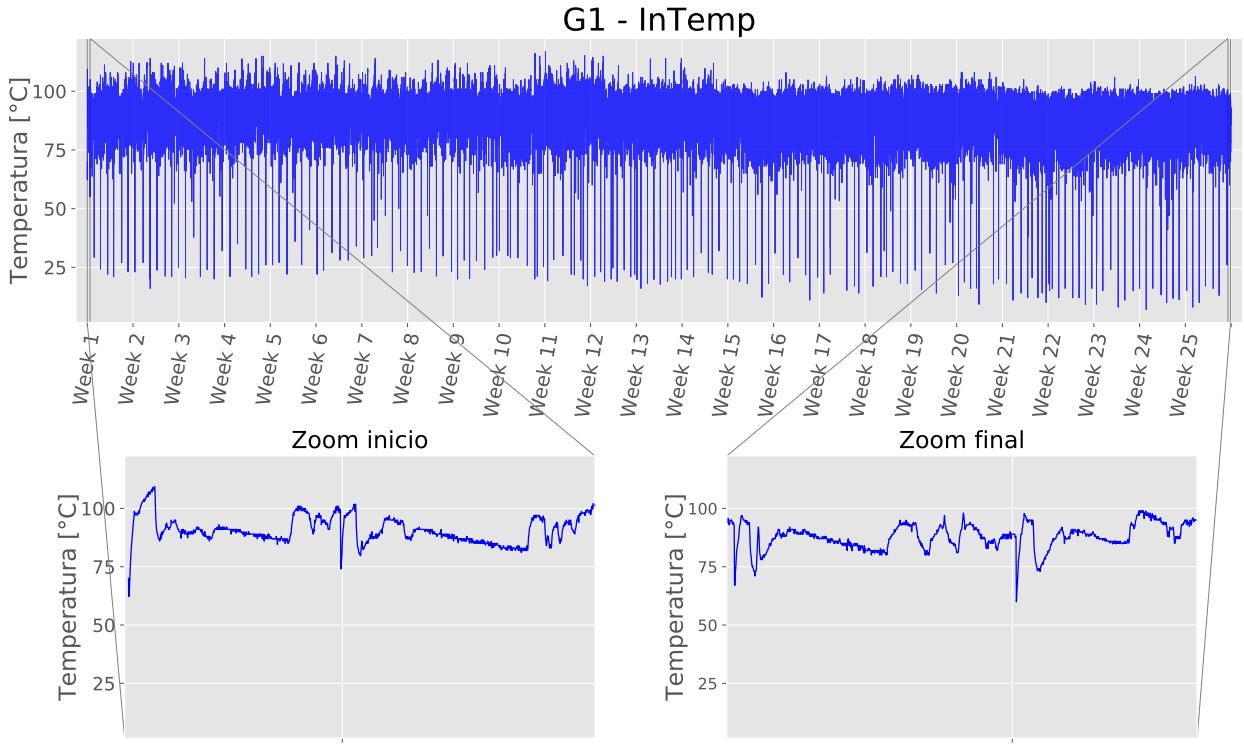

Figura 4.3: Gráfico de datos de temperatura de entrada de aceite.

Para la temperatura de entrada del aceite, se nota una baja repentina justo en la línea que separa los vuelos, lo que tiene sentido ya que el generador estuvo en descanso antes de iniciar el vuelo; los peaks bajos que pueden verse en el gráfico de todos los datos probablemente corresponden al inicio de grabaciones donde el avión estuvo en descanso por un tiempo prolongado, por lo que la temperatura inicial es cercana a la del ambiente. Además, se nota una pequeña tendencia donde los peaks superiores son más altos al inicio de los datos, y que la temperatura decae lenta y establemente cuando el avión está a altura constante (teniendo en cuenta la gráfica anterior).

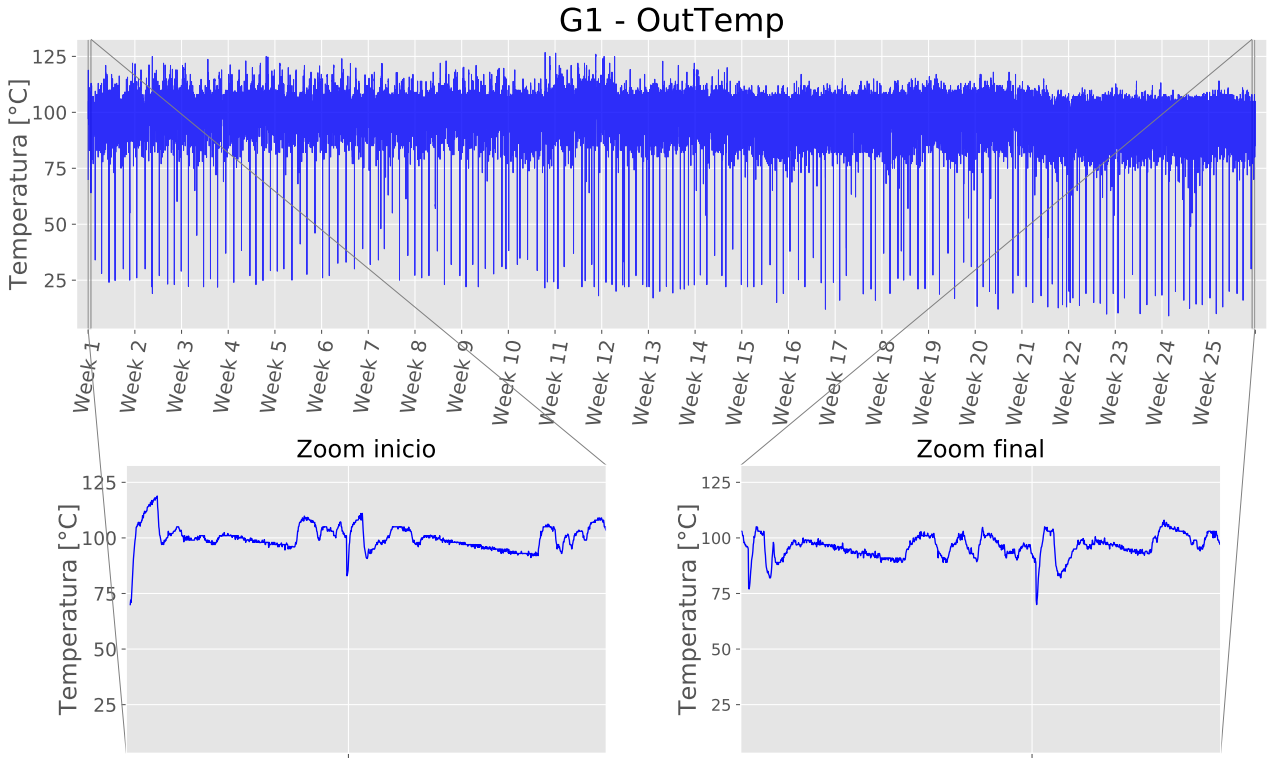

Figura 4.4: Gráfico de datos de temperatura de salida de aceite.

En el caso de la temperatura de salida, se notan las mismas tendencias anteriores, pero además, se ve una pequeña baja de temperatura en las últimas 4 semanas de datos, aunque no son tan claras en este gráfico.

Debido a que las temperaturas de entrada y salida varían similarmente, puede ser de utilidad calcular la diferencia de temperatura y analizarla. Se calcula este delta y se le nombra DtTemp, la siguiente gráfica muestra los resultados.

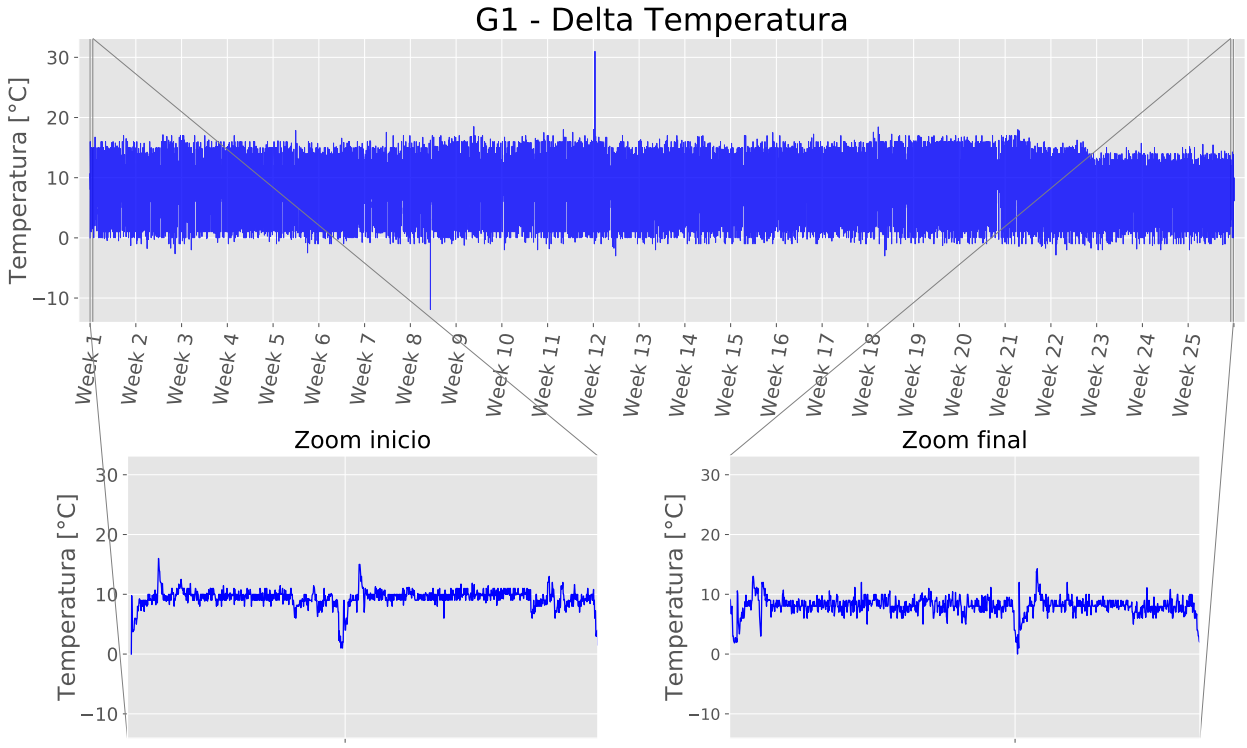

Figura 4.5: Gráfico de datos de delta de temperatura de aceite.

En la Figura 4.5 se puede notar que el delta de temperatura puede ser de utilidad, los datos muestran una alta estabilidad en comparación con las temperaturas individuales, pero en particular, se nota una baja considerable en las últimas semanas, tanto en los peaks como en los valores intermedios estables.

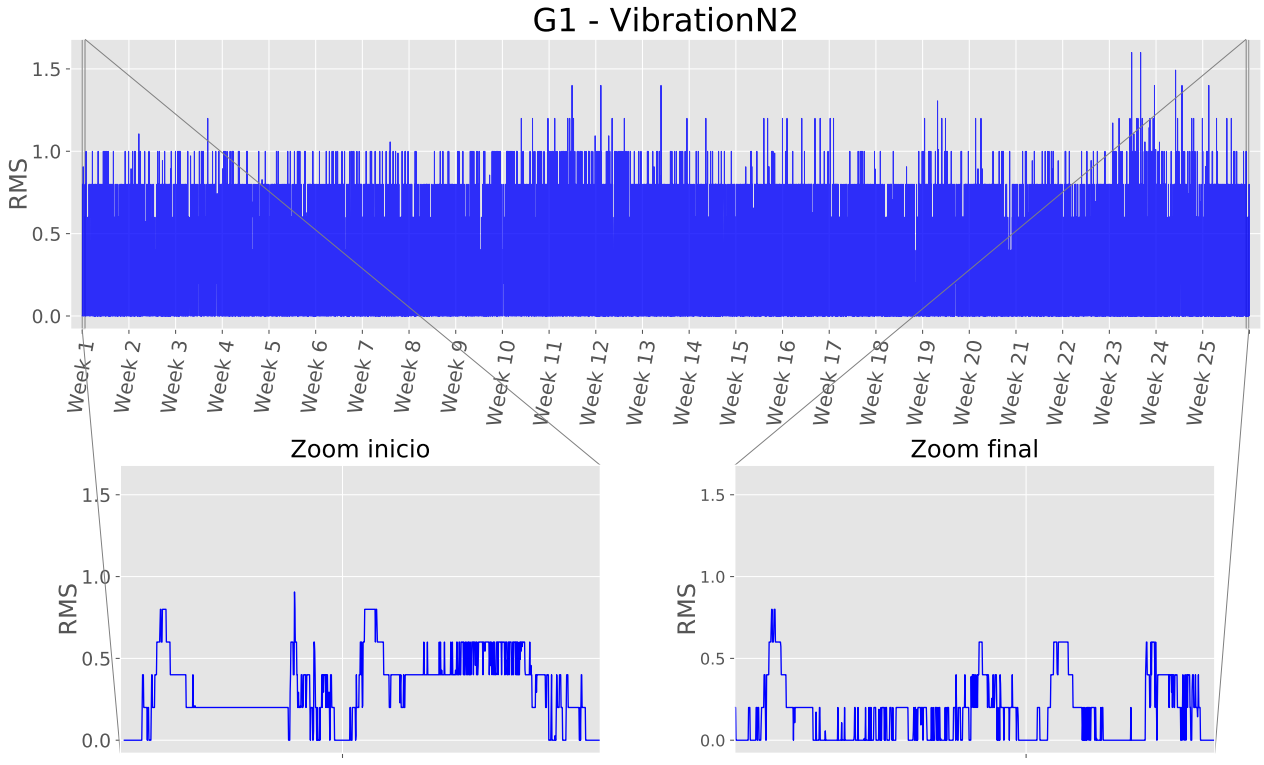

Figura 4.6: Gráfico de datos de vibración de equipo acoplado.

En el caso de la vibración del equipo acoplado, se ve que los datos no son exactamente continuos, pareciera ser que están definidos solamente en múltiplos de 0.2; además, no siguen una tendencia muy clara, pero se notan levemente estables cuando la altura es constante; también hay peaks un poco más altos en las últimas semanas de datos, pero también ocurre en otras semanas.

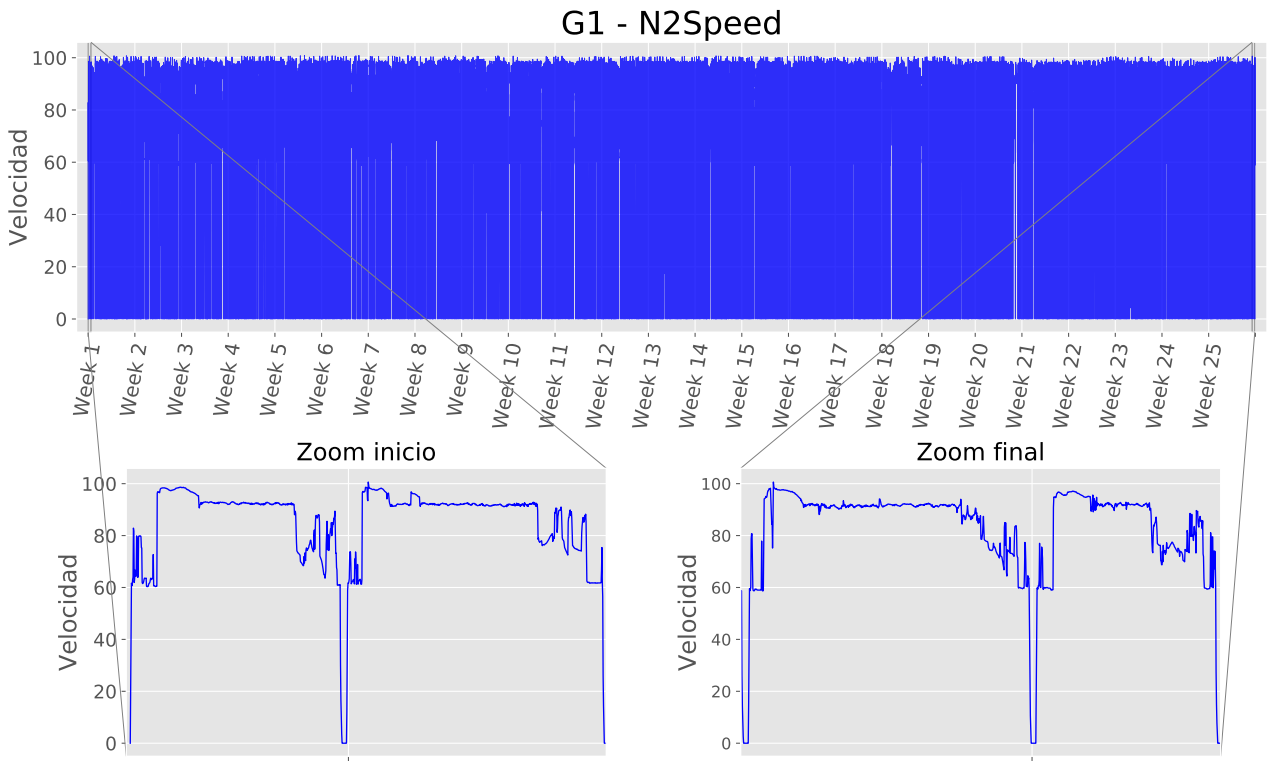

Figura 4.7: Gráfico de datos de velocidad de giro de equipo acoplado.

En el caso de la velocidad de giro del sistema acoplado, se nota estable cuando el avión está a altura constante, y tiene cambios bruscos en el ascenso y descenso, además de eso, no se nota nada especial.

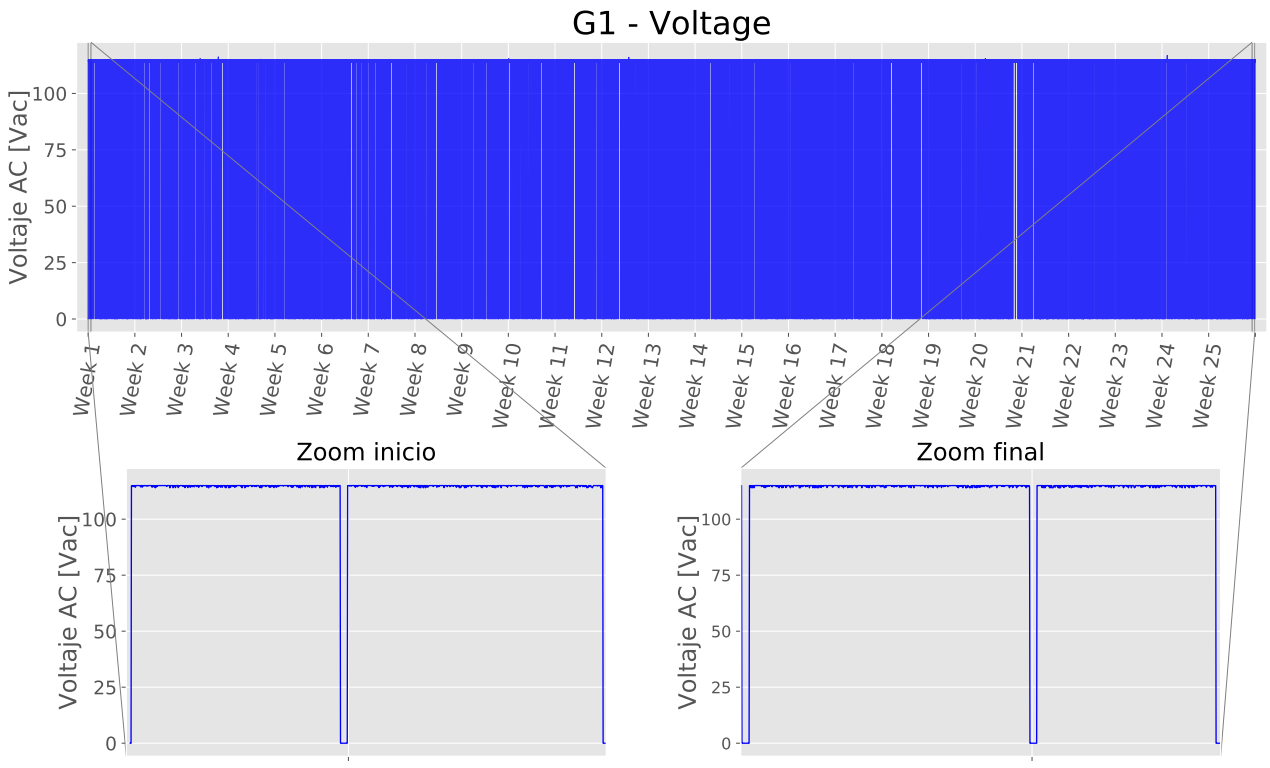

Figura 4.8: Gráfico de datos de voltaje AC de generador.

El voltaje del generador es prácticamente constante en 115Vac, salvo cuando el avión está en tierra, probablemente con el generador apagado.

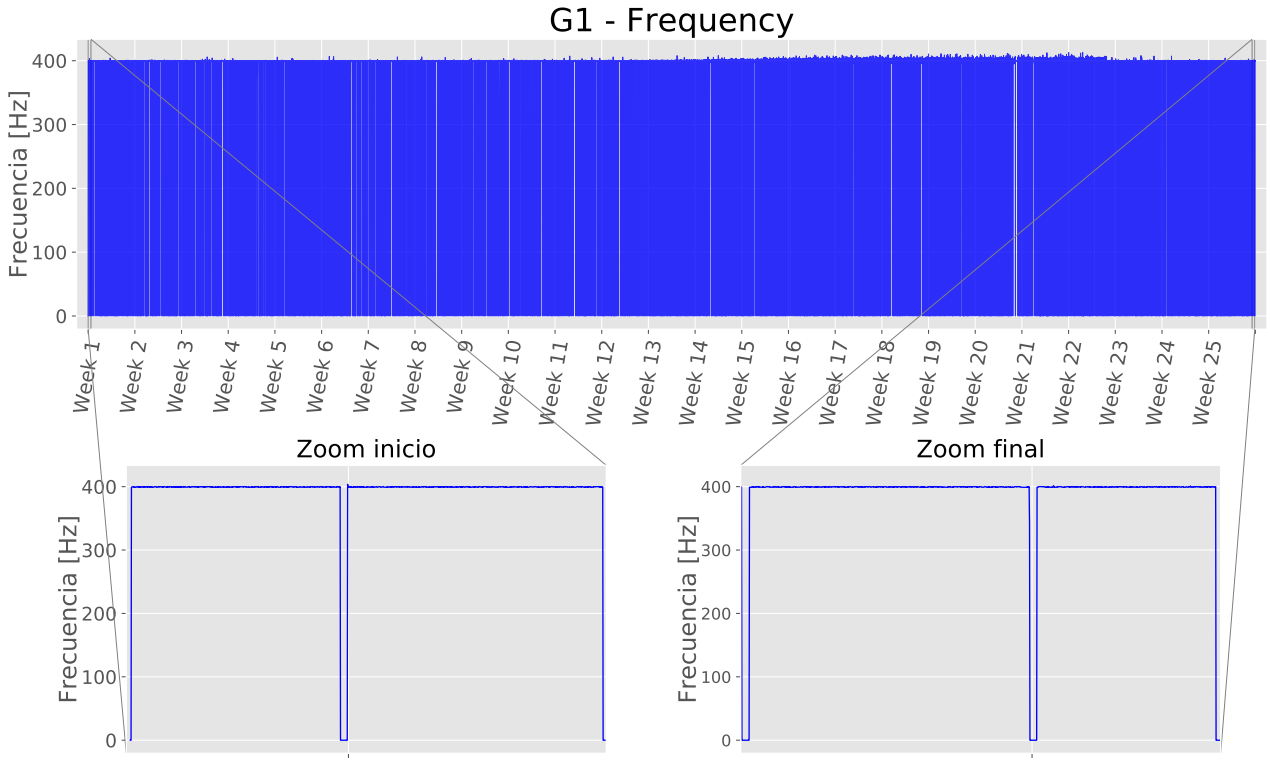

Figura 4.9: Gráfico datos de frecuencia AC de generador.

Al igual que el voltaje, la frecuencia AC del generador también es prácticamente constante.

4.1.2. Aislamiento de estado estacionario

Debido a que la mayoría de los datos varía considerablemente en el ascenso y descenso del avión, se busca aislar secciones estables del funcionamiento, esto significa aislar la fase crucero del viaje, marcado con el número 5 en la siguiente figura.

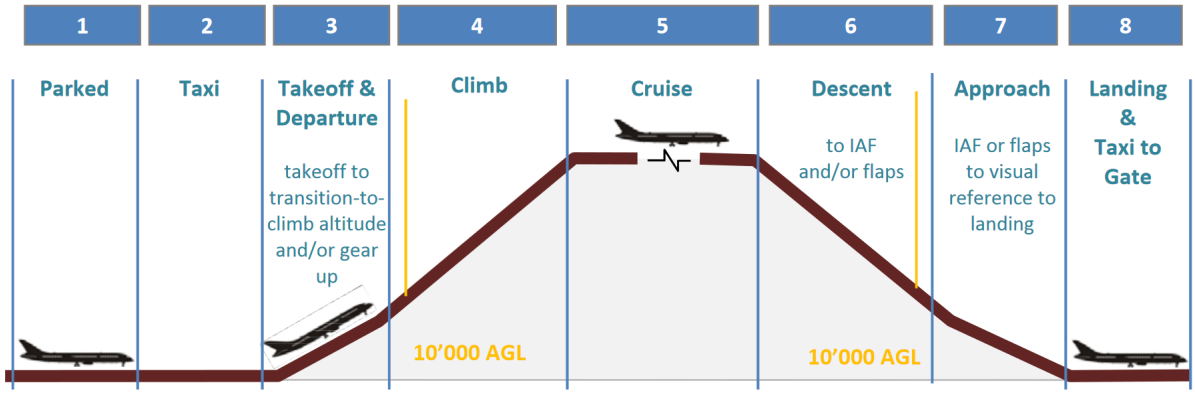

Figura 4.10: Gráfico de etapas de un vuelo común. *Imagen tomada de [8]*

A continuación, se muestran los datos estacionarios aislados de cada generador, donde las líneas verticales delimitan aproximadamente cada semana de datos.

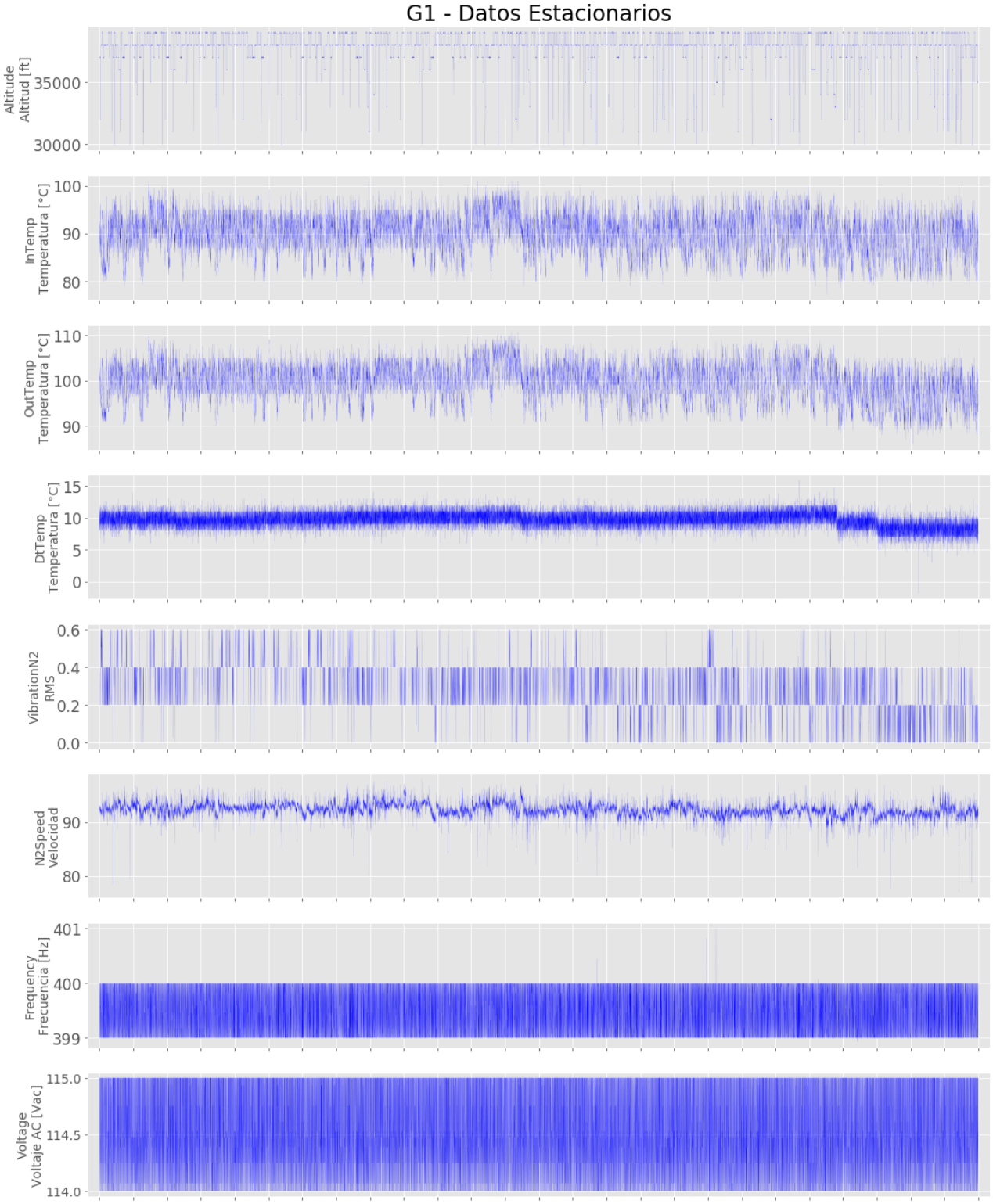

Figura 4.11: Gráfico de datos estacionarios de G1.

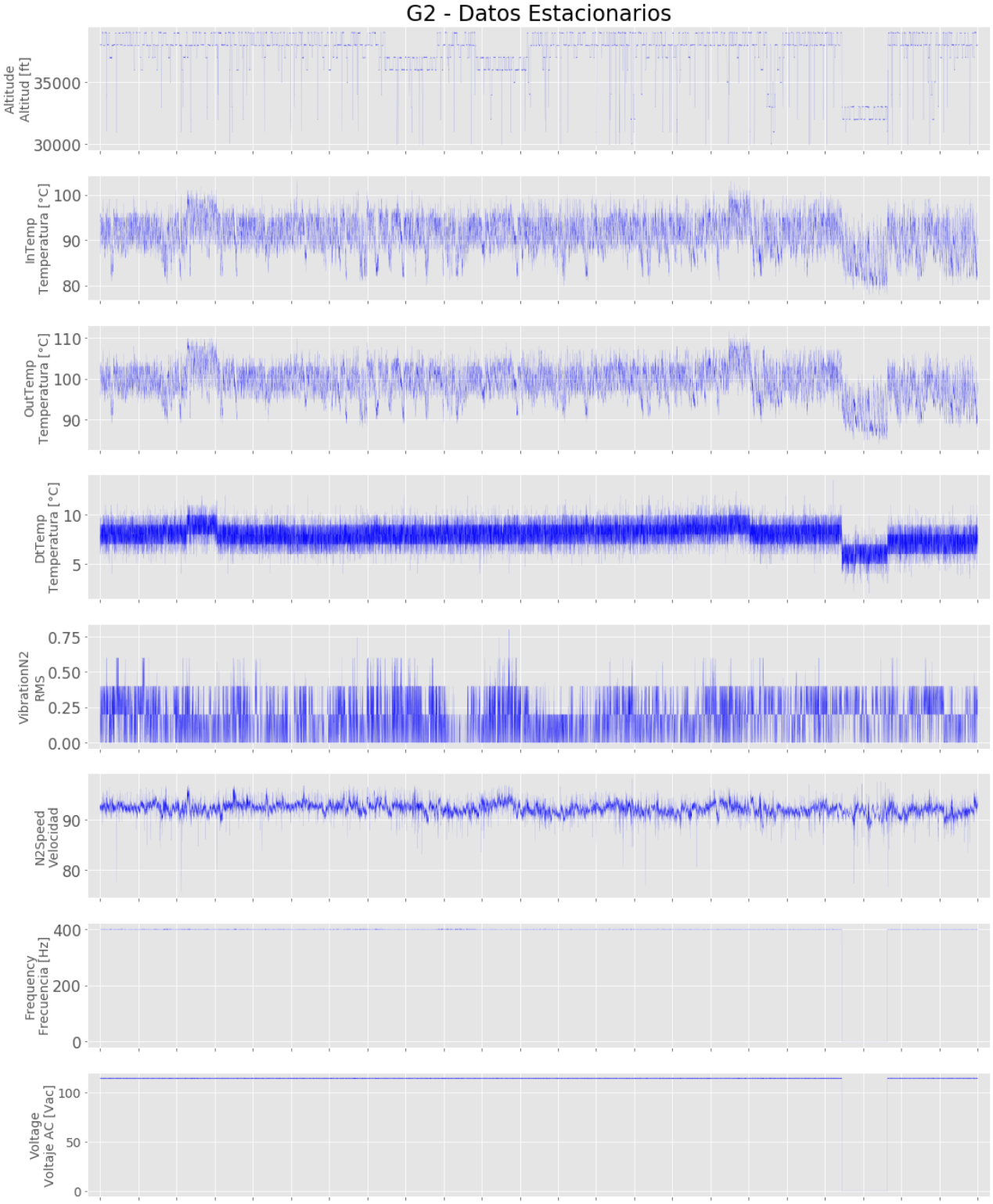

Figura 4.12: Gráfico de datos estacionarios de G2.

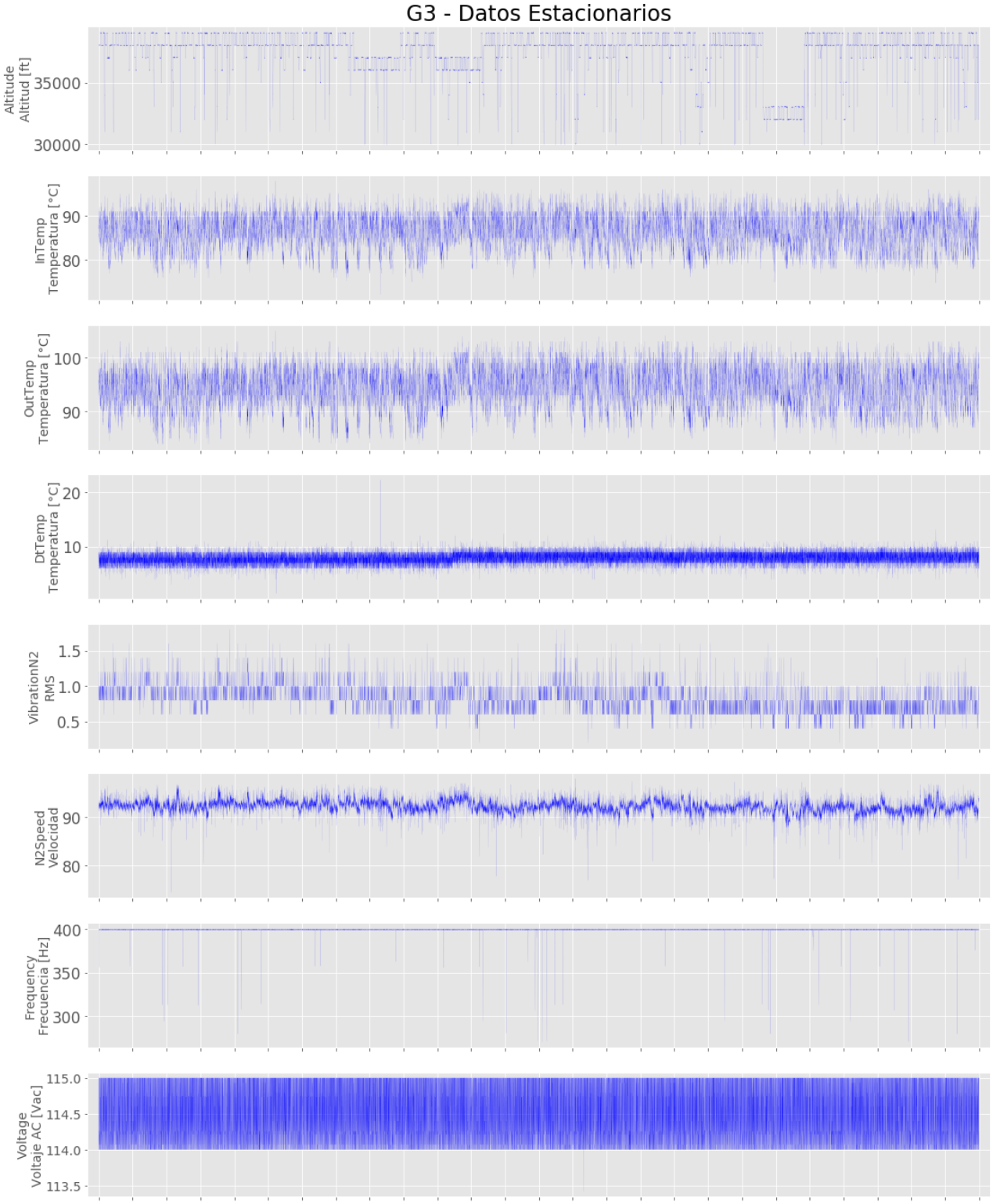

Figura 4.13: Gráfico de datos estacionarios de G3.

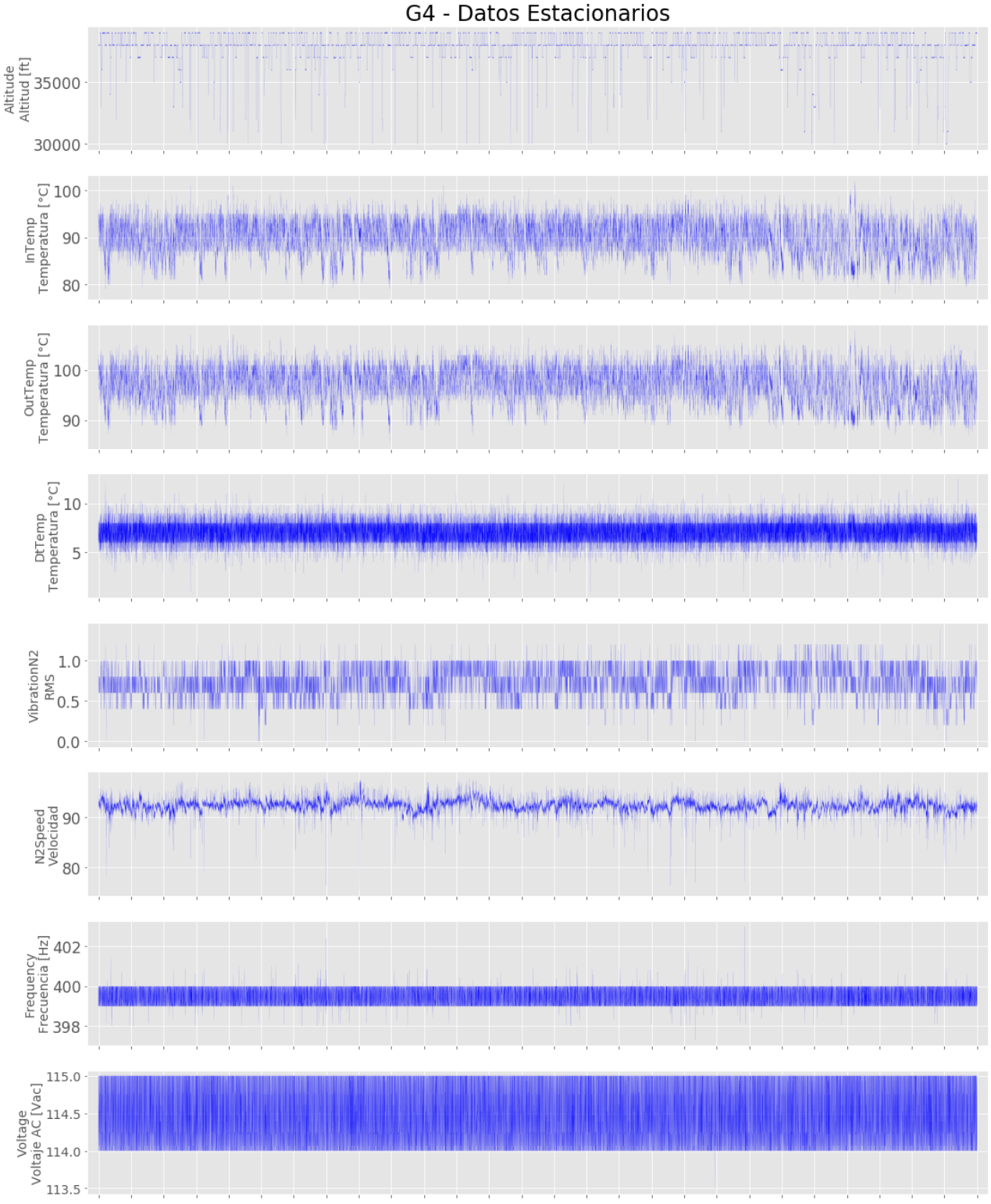

Figura 4.14: Gráfico de datos estacionarios de G4.

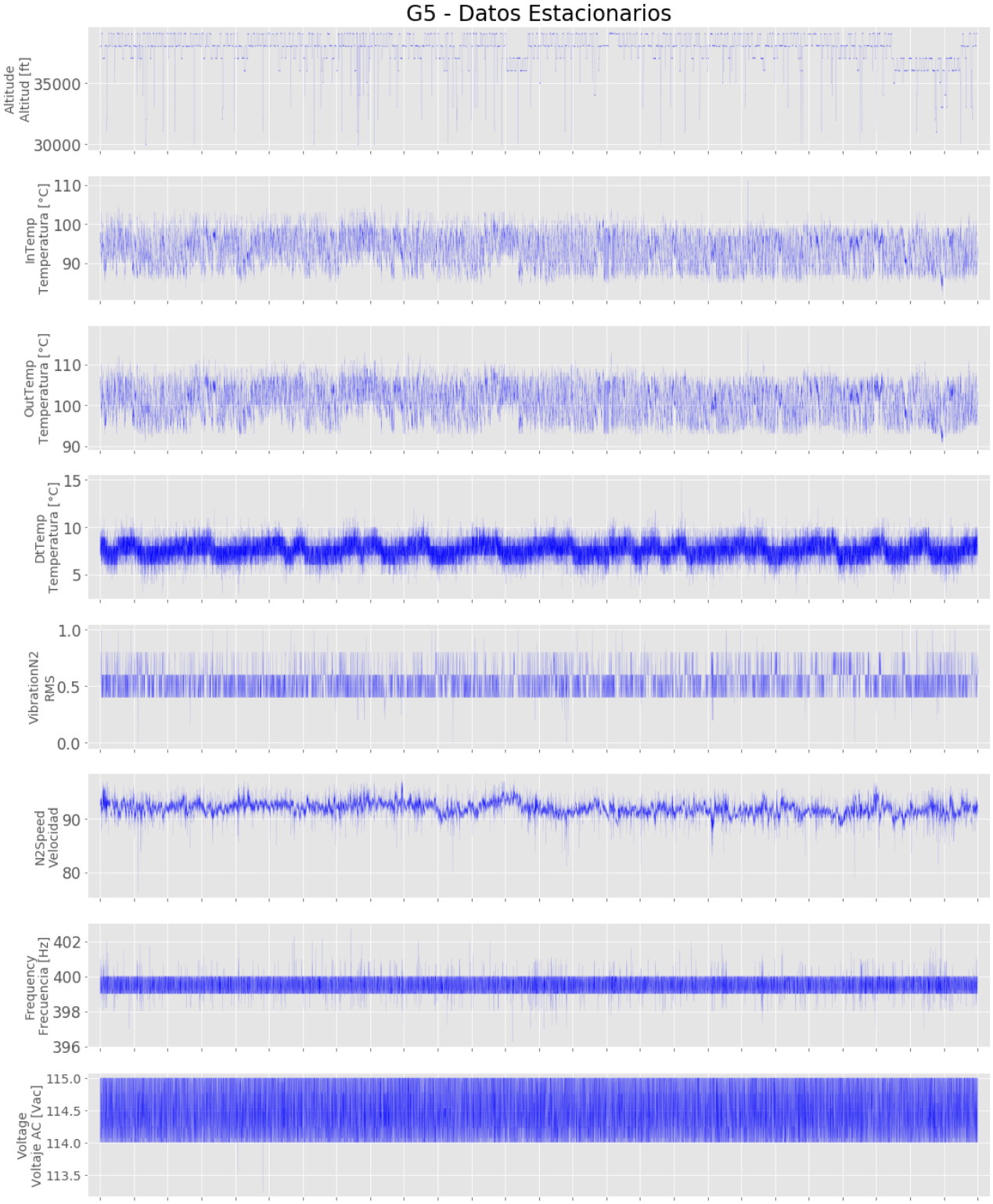

Figura 4.15: Gráfico de datos estacionarios de G5.

De las figuras anteriores se puede destacar lo siguiente:

- G1 y G2 muestran una disminución de delta de temperatura hacia el final de los datos
- G2 muestra saltos bruscos tanto del delta como de las temperaturas individuales, las cuales coinciden entre sí, pero solo coinciden en un solo caso con el cambio de altura usual de vuelo, además, en G1 los cambios de temperaturas individuales no influyen de gran manera al delta; lo cual sugiere que existen factores externos influyendo en los datos.
- En G2, la baja repentina de DtTemp hacia el final de los datos va acompañada de Frequency y Voltage en 0, lo que probablemente significa que el generador no se encontraba en funcionamiento por algún motivo. DtTemp se mantiene bajo luego de esto, por lo que es improbable que ocurriera una reparación.
- G5 tiene un delta de temperatura muy variable, con forma casi cíclica, pero sin periodo constante.
- G1 y G3 muestran una disminución de RMS de vibración hacia el final de los datos, pero el RMS de G3 es aproximadamente el doble de G1 para toda la grabación.
- La velocidad de giro del equipo acoplado no parece entregar información útil en ningún generador
- Las frecuencias y voltajes AC son casi invariables para todos los generadores, mostrando lo que pareciese ser pequeñas diferencias por la sensibilidad del sensor.

4.1.3. Conclusiones preliminares

Se posee una gran cantidad de datos de funcionamiento, lo que es útil para modelos de Aprendizaje Profundo; pero el hecho de que el delta de tiempo no sea constante, elimina muchos métodos de procesamiento que se basan en frecuencia; aun así, Aprendizaje Profundo no requiere de estos para funcionar, por lo que solo resultaría en mayor dificultad para extraer información de calidad desde los datos, potencialmente entregando resultados más débiles.

De acuerdo a lo encontrado, se espera que DtTemp, y VibrationN2 sean los atributos de mayor importancia, InTemp y OutTemp podrían añadir robustez a la información de DtTemp, Altitude solo debiese ser útil para el aislamiento de estado estacionario, y los demás sensores no parecieran mostrar información útil, sobre todo Frequency y Voltage.

Finalmente, se espera que G1, G2 y G3 obtengan mejores resultados en clasificación o regresión. G1 muestra una baja de DtTemp y VibrationN2 hacia el final de los datos, mientras que G2 muestra una baja de DtTemp, y G3, una de VibrationN2; G4 y G5 no muestran tendencias claras.

4.2. Procesamiento

Debido a que usar cada instante temporal por si solo probablemente no contenga suficiente información, se utilizan ventanas de tiempo; basándose en el trabajo hecho previamente por otro alumno con estos datos, se escoge utilizar una ventana de 560 datos, equivalente a 80 segundos (ya que obtuvo buenos resultados).

Como fue mencionado anteriormente, en teoría es inútil realizar un procesamiento basado en el cálculo de frecuencias, pero existen distintos indicadores que no requieren de estas. Para este caso, también se calcularán bandas de Fourier, ya que es posible que los tiempos en TimeStamp pueden solo estar mal grabados; además de que los modelos a utilizar pueden ignorar fácilmente la información no útil, y hacer uso de ella en caso de ser de ayuda. Por esto, se utilizan los siguientes indicadores:

- RMS
- Peak
- Peak to peak
- Crest
- Mean
- Variance
- Asymmetry
- Kurtosis
- 5to momento
- 10 Bandas de Fourier

De modo de utilizar de mayor manera los datos temporales, también se calcula la integral y derivada en el tiempo. De esta forma, se calcula la integral y derivada para cada sensor utilizado, en los 560 datos de ventana, y luego a estos se les calcula los indicadores anteriores, resultando en 57 atributos por sensor.

Además de lo anterior, también se utilizarán las ventanas de 560 datos sin ningún procesamiento, es decir como matrices de 560 datos temporales * n° sensores.

Finalmente, se utilizarán 3 sets distintos de sensores con el objetivo de comprobar cuales de estos son útiles:

- Simple: InTemp, OutTemp, VibrationN2
- Mid: InTemp, OutTemp, VibrationN2, DtTemp
- Full: InTemp, OutTemp, VibrationN2, DtTemp, N2Speed, Frequency, Voltage

De esta manera se puede comprobar que tan útil es agregar DtTemp; y si N2Speed, Frequency y Voltage ayudan a la clasificación.

4.3. Clasificación

4.3.1. Objetivo de clasificación

El objetivo de realizar clasificación es entrenar un modelo que intente ser capaz de detectar la existencia de una falla incipiente en los datos, resultando en la predicción temprana de la falla, y con esto, la posibilidad de planear apropiadamente el mantenimiento de la máquina. Para esto, es necesario un modelo que distinga los datos de falla incipiente de los datos sanos, e idealmente, capaz de predecir la falla con anticipación.

Como primer paso, se supondrá que el inicio de la grabación de funcionamiento del generador contiene datos sanos, mientras que el final de esta contiene datos con falla incipiente; de esta manera, se entrenaran modelos de clasificación que intentaran distinguir datos no usados para el entrenamiento como datos de condición 'Normal' o de 'Falla'. Mas adelante, se entrenará considerando semanas anteriores al final de datos como Falla (ejemplo: considerando la semana 22 como datos de Falla, de 25 en total), para comprobar que los modelos son capaces de distinguir la falla con anticipación, además, luego se puede comprobar con datos futuros (i.e. semanas 23 a 25) que los modelos aprendieron información de utilidad para la predicción temprana.

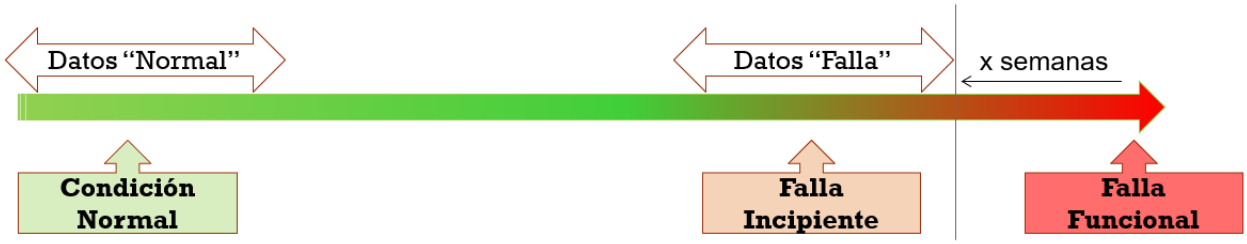

Figura 4.16: Diagrama de entrenamiento para clasificación.

4.3.2. Análisis preliminar

En primer lugar, se quiere determinar cuál procesamiento produce los mejores resultados, para continuar el resto del trabajo utilizando este; por esto, se entrenan distintos modelos de Machine Learning con los procesamientos mencionados, para los sets de sensores Simple, Mid y Full, de la siguiente manera:

- Feature extraccion: corresponde al cálculo de indicadores desde la ventana de 560 datos temporales, resulta en vectores de 57 multiplicado por el número de sensores. Utilizando los siguientes modelos que pueden trabajar con vectores:
  - Decision Tree: max_leaf_nodes=32 (Libreria Sklearn)
  - Random Forest: n_estimators=1000, max_leaf_nodes=64 (Libreria Sklearn)
  - XGBoost: (Libreria xgboost)
  - MLP: dense=5, num_neurons=256, dropout=0.3 (Libreria Tensorflow/Keras)

- Raw time window: corresponde a la ventana de 560 datos temporales sin procesar, resulta en matrices de 560 datos temporales por cada sensor. Utilizando los siguientes modelos que pueden trabajar con matrices de datos temporales.
  - Time distributed MLP: Ver en Anexo B (Libreria Tensorflow/Keras)
  - CNN: Ver en Anexo B (Libreria Tensorflow/Keras)
  - RNN: Ver en Anexo B (Libreria Tensorflow/Keras)

Nota: Los hiperparámetros fueron definidos utilizando GridSearch y Cross-Validation para los primeros 3 modelos; mientras que para los demás, se probaron distintos parámetros y se conservó los que obtuvieron mejores resultados en Val (de sets Train, Val y Test).

Se utiliza la métrica F1 score para comparar las clasificaciones, ya que esta entrega una medida robusta; la cual se muestra en porcentaje.

Tabla 4.1: Resultados de entrenamiento de selección de procesamiento.

	G1 - F1 score						
	Feature extraction				Raw time window		
	DTree	Rforests	XGBoost	MLP	TD-MLP	CNN	RNN
Simple	84,93	88,68	94,96	96	97,5	98,0	98,2
Mid	96,63	96,87	97,63	98	98,5	98,5	97,6
Full	96,63	96,88	97,87	97,9	96,7	98,2	94,4

De la tabla anterior se puede notar como los resultados mejoran considerablemente al pasar de Simple a Mid en Feature extraction, evidenciando el beneficio de agregar el delta de temperatura DtTemp; pero no hay una mejora considerable al pasar desde Mid a Full, lo que muestra que N2Speed, Frequency y Voltaje no proveen información útil para la clasificación. En cuanto a Raw time window, RNN empeora al agregar más sensores, pero esto puede deberse al aumento de overfit del modelo, a priori, Mid parece ser el mejor. Además, se puede apreciar con claridad que utilizar la ventana de datos sin extracción de características entrega mejores resultados.

Por esto, se continuará utilizando solo la ventana de datos cruda (Raw time window); y momentáneamente ambos sets de sensores Mid y Full, ya que los sensores extra podrían eventualmente ser de utilidad para otros generadores.

Pasando a un análisis distinto, también es importante conocer cómo se relacionan los datos de los distintos generadores entre sí; por esto, a continuación se muestra una tabla donde los modelos fueron entrenados con datos de un generador, y fueron probados con datos de cada uno de los generadores.

Nota: Se muestran los promedios de las clasificaciones de cada modelo, ya que los resultados individuales son similares; y que mostrar tablas para cada uno de los modelos es innecesario para este análisis. Además, los siguientes resultados fueron estandarizados con datos del generador respectivo (ej: Test G1 estandarizado con datos de Train G1, Test de G2, con datos de Train G2, etc), ya que de no ser así, no hay generalización entre G1 y G2 (Lo que se puede encontrar en Anexo A).

Tabla 4.2: Resultados promedio de clasificación entre generadores, set Mid.

	F1 score - Mean - Mid				
Train\Test	G1	G2	G3	G4	G5
G1	97,4	82,7	32,8	45,4	27,4
G2	93,7	95	20,3	48,4	25,5
G3	43,9	23,1	92,2	53,6	57,1
G4	62,5	63,2	57,4	60	52,9
G5	77,7	72,9	44,5	56,6	44,1

Tabla 4.3: Resultados promedio de clasificación entre generadores, set Full.

	F1 score - Mean - Full				
Train\Test	G1	G2	G3	G4	G5
G1	95,9	76	36	44,6	30,5
G2	90,4	95,8	24	52,5	29,8
G3	50,1	22,1	93,1	52,8	59
G4	56,3	68,4	48,6	66,4	54,2
G5	74,4	68,9	53,2	53,3	45,2

Como puede apreciarse en las tablas anteriores, entrenar con datos de G1 entrega buenos resultados tanto en el Test de G1 como de G2, y viceversa; también, entrenar en G3, entrega buenos resultados en G3, no así para el resto de los generadores. Esto muestra la posible capacidad de generalizar de los modelos entrenados en G1 y G2, ya que pueden clasificar correctamente datos de generadores distintos; no así los datos de G3, que solo funcionan con sí mismo, lo que podría indicar la existencia de una condición distinta a las de G1 y G2. Los resultados en los sets Mid y Full son similares, pero entrenar en G1 y hacer test en G2 es considerablemente mejor en el set Mid, por lo que se continuará utilizando solamente Mid.

Como los resultados muestran que los modelos entrenados en G4 y G5 no pueden clasificar bien, no se seguirán trabajando. Los resultados de G4 y G5 pueden indicar que los datos de entrenamiento son insuficientes para encontrar una falla de esa clase, o que no existe una falla en ellos.

4.3.3. Detección temprana de Falla incipiente

Una vez comprobado que los modelos son capaces de distinguir datos del inicio y final de las grabaciones de un generador (lo cual se asocia a la detección de falla), se busca cuando es lo más temprano que puede detectarse la falla incipiente.

Para esto, se busca tener una clara idea de cuál es la semana en donde los datos cambian lo suficiente para ser posible la detección de la falla incipiente; y también, conocer si la clasificación en semanas futuras es acertada, validando los resultados del entrenamiento. Para esto, se entrenan distintas instancias de los modelos en diferentes semanas como Falla cada uno, por ejemplo, se entrena con semana 25 como Falla, otra instancia con 24 como Falla, otra con 23 como Falla, etc; esto con el objetivo de verificar cómo clasifican los modelos entrenados en estas semanas, al realizar tests en todas las ultimas semanas de la grabación.

Nota: Se varía la semana para Falla, pero la semana para Normal se mantiene constante como la primera semana de datos, ya que cambiarla solo añadiría ruido y complicaría la comparación entre resultados.

Los resultados para G1, G2 y G3, son los siguientes:

Tabla 4.4: Resultados de variación de semana Falla, G1.

	G1 - F1 score - Mean							
Train\Test	wk 17	wk 19	wk 20	wk 21	wk 22	wk 23	wk 24	wk 25
week 25	25,7	28,2	33,2	40,9	40,7	96,5	94,3	97,0
week 24	26,2	28,0	34,2	43,9	44,1	97,6	97,5	98,3
week 23	25,8	28,0	34,4	44,2	44,8	94,9	94,2	92,9
week 22	34,3	31,1	34,9	72,9	82,1	91,2	91,8	90,8
week 21	44,5	54,6	76,9	75,7	66,3	79,7	76,5	76,8
week 20	42,8	61,9	81,0	51,8	32,4	24,6	24,7	26,9
week 19	48,2	60,2	71,3	48,1	34,8	32,1	32,9	36,7
week 17	48,3	56,3	59,7	41,9	36,0	17,1	16,2	17,5

Nota: La grabación de datos de G1 terminan en la semana 25, por lo que la Tabla 4.4 muestra las últimas semanas de datos. Los resultados de cada modelo individual se encuentran en Anexo A.

En la tabla anterior se puede ver como los resultados de las 3 últimas semanas son buenos para los entrenamientos en las semanas entre 21 y 25; también son relativamente buenos para las semanas 21 y 22 si se entrena con las mismas semanas 21 y 22; y finalmente, la clasificación de las últimas semanas es deficiente si se entrena con semanas anteriores a la 21. Esto muestra que la falla incipiente puede detectarse tempranamente en la semana 21, pero debido a que la calidad de clasificación disminuye cuando se entrena con la semana 21, puede ser conveniente utilizar el entrenamiento de la semana 22.

En la Tabla 4.4 también se puede notar que el comportamiento de los datos en las últimas 3 semanas es similar, ya que tienen resultados parecidos; lo que también podría ser el caso de las semanas 21 y 22. Esto coincide con el gráfico de DtTemp en la Figura 4.11, donde se muestra que el comportamiento sigue precisamente la misma tendencia encontrada en la Tabla 4.4.

Tabla 4.5: Resultados de variación de semana Falla, G2.

	G2 - F1 score - Mean							
Train\Test	wk 16	wk 17	wk 18	wk 19	wk 20	wk 21	wk 22	wk 23
week 23	42,3	32,6	37	38,7	54,1	97,9	97,5	97,8
week 22	42	32,6	36	37,7	54,6	97	95,5	96,5
week 21	42,5	32,9	36,6	38	54,7	96,9	92,3	93
week 20	48	64,4	63,9	63,2	69,9	74,8	66	51,9
week 19	55,2	51,8	58,9	66,1	65,2	47,8	60	53,4
week 18	41,7	66,6	71,6	63,2	72,4	79,9	70,8	78,3
week 17	53,4	74,7	50,1	40,7	47,2	28,4	28,5	25,3
week 16	62,6	66,9	50,3	49,8	54,9	34,2	35,6	34,4

Nota: La grabación de datos de G2 terminan en la semana 23, por lo que la Tabla 4.5 muestra las últimas semanas de datos. Los resultados de cada modelo individual se encuentran en Anexo A.

En la Tabla 4.5 se puede ver un comportamiento similar a la Tabla 4.4 en las últimas 3 semanas de datos; pero esto cambia en las 3 semanas anteriores, donde se nota una brusca disminución de la calidad de resultados entre los entrenamientos de semanas 21 y 20, resultados que mejoran levemente en el entrenamiento de semana 18.

Nuevamente, estos resultados tienen sentido cuando se considera la evolución del delta de temperatura DtTemp; la Figura 4.12 muestra una repentina disminución de DtTemp que corresponde a parte de las semanas 20 y 21, lo que resulta en que el entrenamiento en la semana 21 tenga una clasificación peor en la últimas semanas, y que al entrenar con datos anteriores, los resultados empeoran porque DtTemp es muy similar a los valores iniciales, habiendo una pequeña mejoría en el entrenamiento de semana 18, ya que esta posee un DtTemp levemente menor a las semanas adyacentes.

Tabla 4.6: Resultados de variación de semana Falla, G3.

	G3 - F1 score - Mean							
Train\Test	wk 6	wk 7	wk 8	wk 9	wk 10	wk 11	wk 12	wk 13
week 13	50,3	48,3	48,3	64,6	62,4	88,6	89	93,6
week 12	48,7	49,2	46,5	60,4	56,2	86,8	90,2	92,8
week 11	48,9	56,6	48,5	60,6	56,1	92,5	84,7	83
week 10	55,3	37,5	56,5	61	66,3	57,6	66,9	70,4
week 9	52,1	38,8	59	66,8	67	57,3	67,9	69,6
week 8	49,3	49	60,3	59,8	60,9	65,8	61,6	64,7
week 7	44	47,3	40,2	39,2	35,4	42,2	43,2	39,1
week 6	51,4	60,3	54,1	48,7	39,3	58,1	44,7	41

Nota: La grabación de datos de G3 terminan en la semana 26, por lo que la Tabla 4.6 NO muestra las últimas semanas de datos, debido a que las semanas futuras tienen resultados similares a las semanas 11-13. Los resultados de cada modelo individual se encuentran en Anexo A.

En la Tabla 4.6 se nota un cambio más suave que los anteriores, pero los mejores resultados de clasificación son notoriamente los entrenamientos con semana 11 en adelante. Como fue mencionado, los resultados para semanas futuras son similares a las ultimas semanas, lo que muestra que puede detectarse un cambio de estado con 15 semanas de anticipación.

En esta ocasión, la Figura 4.13 muestra solo un pequeño cambio en DtTemp en las semanas 10-11 que pueden explicar los resultados de la Tabla 4.6, pero este se trata de un aumento de DtTemp, contrario a la tendencia de los generadores G1 y G2, por lo que probablemente no se trata de la misma condición.

Debido a que la clasificación entre generadores (Tablas 4.2 y 4.3) no muestra generalización para G3; el que los resultados parecieran estar conectados con DtTemp, pero DtTemp no sigue una tendencia similar a G1 y G2; y que la falla incipiente puede ser realizada con 15 semanas de anticipación, en comparación a las 3-4 semanas de G1 y G2; es posible que esta clasificación solo detecte una anomalía; no se puede comprobar que se trate de una falla. Por esto, no se continuará trabajando con G3 en el siguiente análisis, aunque se entrenará en regresión para verificar.

4.3.4. Mejoras finales

De la sección anterior ya puede obtenerse un clasificador útil, pero existen formas de mejorar los resultados:

La primera se trata de no realizar clasificación sobre ventanas de tiempo individuales, si no que clasificar vuelos completos a través de las ventanas; es decir, tomar los vuelos, separarlos en ventanas, clasificar esas ventanas, y luego tomar el resultado más repetido (Normal o Falla) como el resultado del vuelo; de esta manera, solo se necesita que el 50 % + 1 de las clasificaciones de ventana sean correctas. Para clasificar vuelos completos es necesario que los datos de entrenamiento sean ventanas de vuelos completamente distintos a los de Test, por lo que el código se modifica apropiadamente para garantizar esto.

La segunda se trata de entrenar tomando más de una semana de datos de Falla y Normal, ya que tener más datos de entrenamiento podrá ser beneficioso:

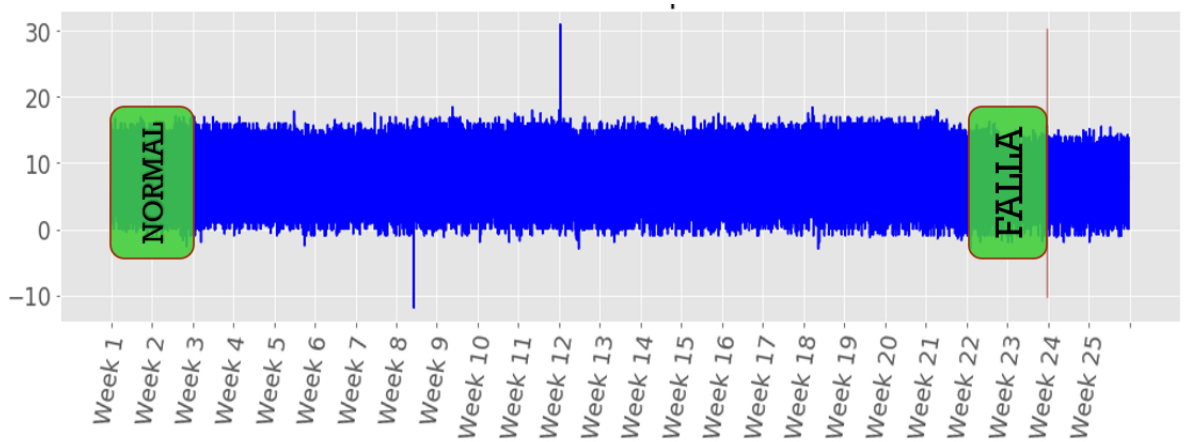

Figura 4.17: Ejemplo de entrenamiento con 2 semanas, 22-23.

De esta forma, se entrenan modelos con 1, 2 y 3 semanas de datos para Normal y para Falla, tomando las semanas relevantes de la sección anterior. Notar que como el código fue modificado para separar cada vuelo, los resultados pueden variar respecto a la sección anterior. Los resultados individuales se encuentran en Anexo A.

Tabla 4.7: Entrenamiento de múltiples semanas para G1.

	G1 - F1 score - Mean							
Train\Test	wk 17	wk 19	wk 20	wk 21	wk 22	wk 23	wk 24	wk 25
week 23	26,5	28,8	35,1	44,4	44,7	96,1	96,1	96,2
weeks 22-23	27,1	28,2	33,9	73,0	75,0	96,9	97,4	97,6
weeks 21-23	38,5	51,7	81,3	88,5	76,1	93,7	93,4	93,3
week 22	38,2	35,2	41,3	75,4	84,4	92,6	92,1	91,1
weeks 21-22	39,9	50,6	81,9	88,9	75,7	94,2	94,0	92,1
weeks 20-22	46,5	59,1	82,4	81,6	65,6	88,5	88,2	86,6
week 21	46,0	54,6	75,7	75,3	63,5	79,2	76,5	77,4
weeks 20-21	43,0	62,6	85,6	75,7	58,5	82,1	79,9	81,6
weeks 19-21	48,4	65,1	82,1	76,8	57,5	81,7	78,9	80,1

Es relevante destacar que lo más importante es la clasificación de las semanas futuras a las usadas de entrenamiento, ya que estas dan una idea real de cuanto se puede anticipar la falla, ya que la clasificación no puede ser una interpolación de datos aprendidos; por lo que se busca la mayor anticipación de falla, junto con los mejores resultados en semanas futuras; teniendo en cuenta que esto último debiese mejorar al utilizar la clasificación de vuelos completos.

Teniendo lo anterior en mente, se puede ver en la Tabla 4.7 que, en general, entrenar con más semanas no aporta beneficios considerables a la clasificación de las últimas semanas, solo a semanas anteriores; lo cual es lógico ya que se está entrenando con datos más cercanos a estos, y por lo tanto, probablemente más similares. Por lo tanto, los mejores resultados son los entregados por los entrenamientos 'week 22' y 'weeks 21-22', ya que se tiene una mayor anticipación de falla que al incluir la semana 23 en el entrenamiento; de estas dos puede elegirse 'week 22' como el mejor candidato, ya que los resultados de Test en semana 22 son considerablemente mejores. Finalmente, también puede incluirse 'week 21' como un candidato, ya que entrega una semana más de anticipación, y los resultados podrían ser suficientes una vez realizada la clasificación de vuelos completos. Resultados individuales en Anexo A.

Tabla 4.8: Entrenamiento de múltiples semanas para G2.

	G2 - F1 score - Mean							
Train\Test	wk 16	wk 17	wk 18	wk 19	wk 20	wk 21	wk 22	wk 23
week 22	42,1	32,8	36,8	39,5	55,5	97,1	95,9	96,5
weeks 21-22	41,6	33,0	37,0	37,5	55,2	97,9	95,8	97,1
weeks 20-22	44,2	63,4	59,7	60,2	71,4	89,6	86,2	90,0
week 21	42,2	32,7	36,8	38,7	55,3	97,5	94,7	95,7
weeks 20-21	45,5	53,2	52,6	56,1	69,8	89,1	80,6	85,2
weeks 19-21	49,4	63,3	64,8	63,1	75,1	88,1	78,9	83,6

En este caso se decidió no entrenar con semanas anteriores a las mostradas en la Tabla 4.8, ya que se estaría entrenando con los datos donde el generador se encontraba aparentemente apagado. De esta tabla se desprende que el entrenamiento en 'week 21' es el mejor, ya que entrega la mayor anticipación a la falla, y buenos resultados.

Tomando los candidatos anteriores, se procede a clasificar vuelos de todas las semanas con estos modelos. En esta ocasión se muestran los resultados de TD-MLP, RNN y CNN por separado, ya que busca el modelo con los mejores resultados; estos fueron levemente mejorados, las nuevas arquitecturas pueden encontrarse en Anexo.

Nota: Como el objetivo de obtener un clasificador es utilizarlo en datos de un generador distinto, y como solo estandarizar con datos del mismo generador entrega bueno resultados, se deben utilizar datos del inicio de la grabación para la estandarización, y no los datos de entrenamiento (que contienen datos de Normal y de Falla), ya que no se tendrán datos de Falla para un generador que aun no ha fallado; por lo que se utilizan las primeras 2 semanas de datos del generador respectivo para esto.

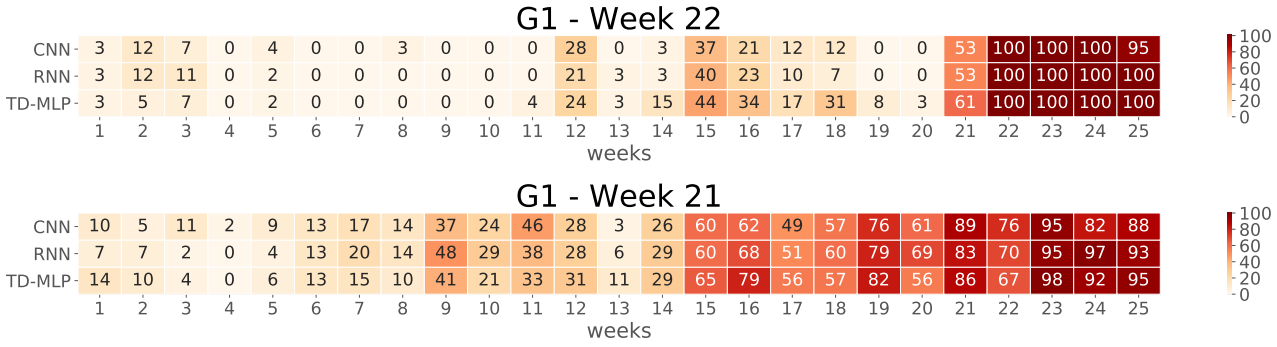

Figura 4.18: Porcentaje de vuelos clasificados como Falla en cada semana, para cada modelo, en G1.

Observando la Figura 4.18, puede notarse que los resultados de cada modelo son similares. Además, para los modelos entrenados en semana 22, CNN puede considerarse el de los mejores resultados, ya que se equivoca menos en la semana 15, que es la semana con más errores; para los entrenados en semana 21 es más complicado de definir, pero nuevamente CNN puede declararse como mejor, debido al más bajo error en la clasificación de la semana 9, la cual es la primera semana de altos errores. Si se comparan los entrenamientos de distintas semanas, se nota que los de la semana 22 clasifican confiablemente como Falla solo las últimas 5 semanas, teniendo algunos errores; en cambio, los entrenados en la semana 21 clasifican más de 5 semanas como falla, y con menos confianza, ya que existe un porcentaje alto de vuelos clasificados como Normal en las últimas 4 semanas, especialmente en la semana 22. De esta forma, se escoge el modelo CNN entrenado en la semana 22 como el mejor modelo, ya que clasifica con mayor confianza, y las semanas clasificadas como Falla coinciden con las esperadas de los resultados anteriores.

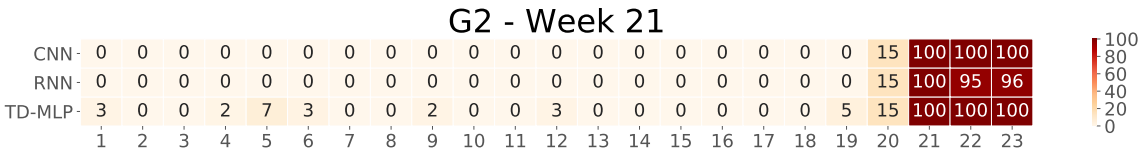

Figura 4.19: Porcentaje de vuelos clasificados como Falla en cada semana, para cada modelo, en G2.

En este caso, para G2 se escoge CNN como el mejor modelo, principalmente por la confianza en la clasificación de las últimas 2 semanas y los bajos errores en las demás; aunque los resultados de todos los modelos son similares.

Teniendo estos 2 modelos entrenados, se pasa a probar la capacidad de generalización de estos, a través de la clasificación de datos del otro generador; lo cual se muestra a continuación.

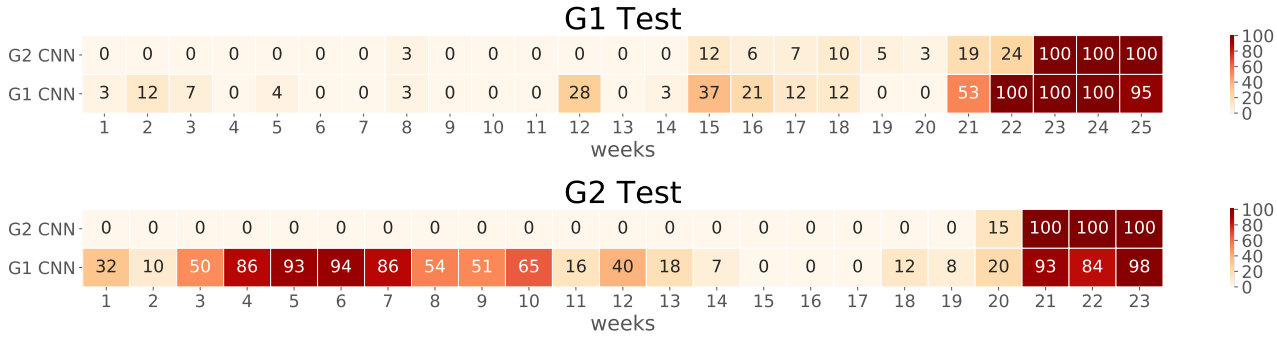

Figura 4.20: Porcentaje de vuelos clasificados como Falla para los modelos seleccionados, entre G1 y G2.

Como se puede ver en la figura anterior, ambos modelos entregan resultados razonables para el Test en G1, pero esto no es así para G2, donde la CNN entrenada en G1 tiene muchos errores. Esto muestra que la CNN entrenada en G2 generaliza de mejor manera, por lo tanto, se considera como el modelo final de clasificación; el cual puede detectar la falla incipiente con 3 semanas de anticipación en G1, y 2 a 3 en G2 (ya que se entrenó con la semana 21, no se puede decir con total confianza que son 3 semanas).

4.4. Regresión

4.4.1. Objetivo de Regresión

El objetivo de realizar regresión es entrenar un modelo que intente ser capaz de predecir la RUL (vida útil remanente) del generador, resultando en la predicción temprana de la falla y estimación de tiempo de vida restante.

4.4.2. Construcción de RUL

Para realizar regresión, es necesario tener una etiqueta numérica que represente la RUL, la cual debe construirse a partir de los datos. En este caso se intenta recrear una medida similar al tiempo real restante de funcionamiento.

A cada ventana de datos se le asigna un numero de etiqueta que representa la posición cronológica de la ventana (ejemplo: primera ventana tiene n°1, ventana 200, tiene n°200, etc.); esto puede entenderse inversamente al considerar que si te tiene el orden ascendente de las ventanas, también se tiene el orden descendente, es decir, se tiene la cantidad de ventanas restantes antes de la falla (ejemplo: si la última ventana es n°31000, el ventana n°30000 representa 1000 ventanas antes de la falla). Además, estas etiquetas son transformadas para que tomen valores desde 0 a 1000, por simplicidad.

4.4.3. Predicción de RUL

Utilizando modelos con las mismas arquitecturas que los de clasificación, salvo un cambio en la capa de salida (Arquitecturas en Anexo B), se intenta predecir la RUL construida. Los modelos se entrenan aproximadamente en las semanas desde donde se puede detectar la falla incipiente de acuerdo a los resultados de clasificación.

Es importante destacar que los gráficos muestran la división de semanas en las líneas verticales, pero como cada vuelo tiene distinta cantidad de ventanas, esta no es perfecta, solo es referencial. Además, los gráficos muestran el mejor resultado de los 3 modelos entrenados, pero estos son relativamente similares.

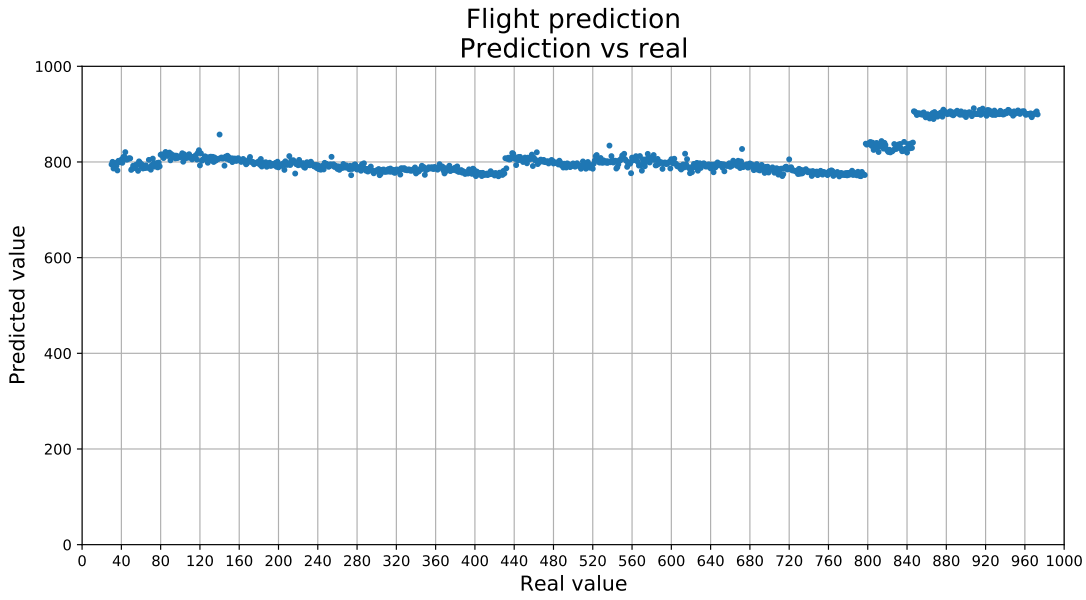

Figura 4.21: Predicción de RUL de mejor modelo (RNN) en G1.

Nota: En G1, los modelos fueron entrenados con las semanas 19-25. Resultados de cada modelo en Anexo A.

Como puede verse en la Figura 4.21, se distinguen 3 distintas zonas a modo de escalones, las cuales concuerdan con los cambios de DtTemp en la Figura 4.11; pero la predicción no tiene una forma diagonal para ninguna zona, es decir, el valor predicho no aumenta en conjunto con el valor real, lo que muestra que el modelo no aprende ningún concepto de 'tiempo hasta la falla', si no que funciona más como un clasificador.

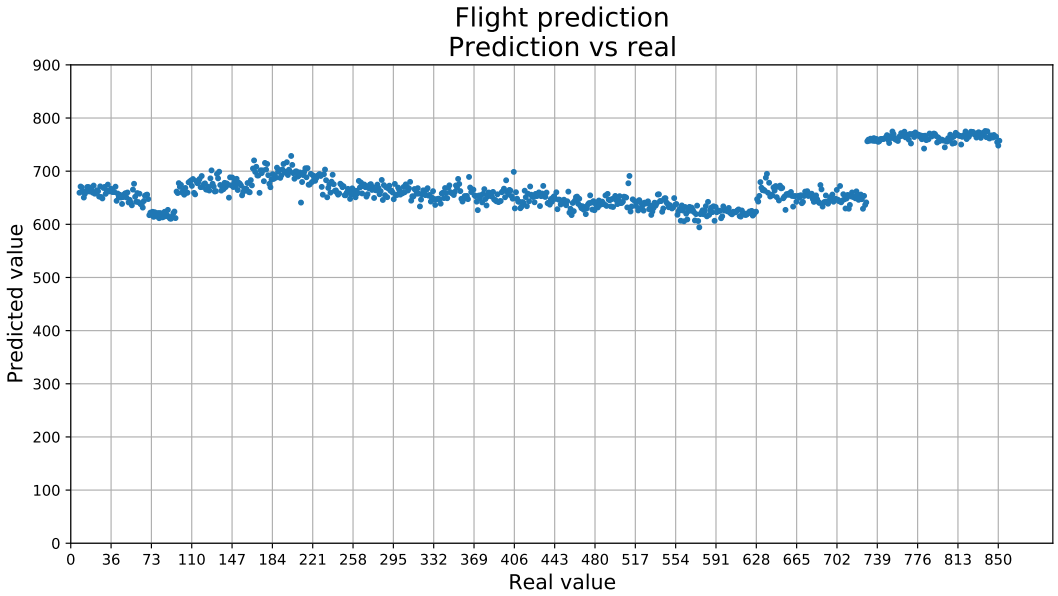

Figura 4.22: Predicción de RUL de mejor modelo (CNN) en G2.

Nota: En G2, los modelos fueron entrenados con las semanas 17-23. Resultados de cada modelo en Anexo A.

El caso de la Figura 4.22 para G2 es similar a G1, solo que no puede distinguirse la baja repentina de DtTemp visible en la Figura 4.12; pero nuevamente los modelos no aprenden un concepto temporal.

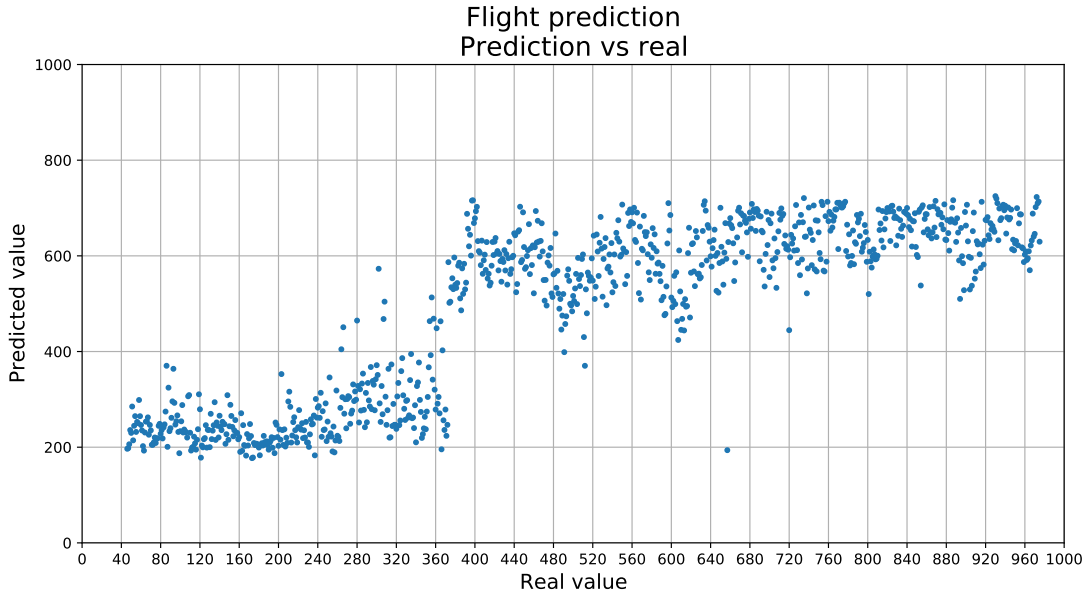

Figura 4.23: Predicción de RUL de mejor modelo (RNN) en G3.

Nota: En G3, los modelos fueron entrenados con todas las semanas, ya que el cambio de estado en los datos puede ser detectado con una anticipación de más de la mitad del total de semanas. Resultados de cada modelo en Anexo A.

Si bien ya se había descartado la utilización de G3 para un clasificador, la Figura 4.23 muestra como existe solo dos secciones donde las predicciones cambian, lo que concuerda con el análisis de clasificación; y también muestra como nuevamente no se aprende un concepto de tiempo hasta la falla. Estos resultados de regresión reafirman la decisión tomada de no utilizar G3 para la predicción de falla, ya que, como fue mencionado antes, los modelos entrenados no generalizan para G1 y G2, DtTemp no muestra la misma tendencia que G1 y G2, y el cambio de estado en los datos puede ser realizada muchas semanas antes que en G1 y G2.

Como fue visto en las figuras para G1 y G2, los modelos no logran aprender a diferenciar temporalmente los datos, o en otras palabras, a predecir el RUL construido; solo logran diferenciar ciertas secciones del total de datos, similar a un clasificador; por esto, puede decirse que los resultados de regresión no son útiles, no se puede predecir el RUL con estos.

Capítulo 5

Discusiones

En el análisis de datos se encontró que DtTemp y VibrationN2 tienen los cambios más notorios a lo largo de las grabaciones, en particular, G1 muestra cambios de ambos, G2 solo de DtTemp, y G3 solo de VibrationN2; pero todos los resultados parecieran seguir solo los cambios de DtTemp, los clasificadores y regresores diferencian los datos cuando DtTemp cambia, y no hay cambios notorios cuando solo VibrationN2 cambia; es más, en la regresión de G3 esto es muy evidente, el cambio en la predicción (Figura 4.23) cambia drásticamente en la semana donde DtTemp presenta un leve cambio (Figura 4.13), pero en el resto de semanas el cambio es mínimo. VibratioN2 no pareciera entregar información útil, y esto puede deberse a que los valores de este sensor cambian bastante entre vuelos adyacentes (Figura 4.6); esto hace que los modelos tengan problemas en aprender una relación dato-etiqueta robusta.

La clasificación entrega un modelo entrenado en G2 capaz de reconocer la falla hasta 3 semanas antes de que ésta ocurra, lo cual se comprueba con los datos de G1 y G2; pero el modelo entrenado en G1 podría detectar la falla con 4 o 5 semanas de anticipación en G1 (Figura 4.20). Esto probablemente se debe a que estas semanas extra tienen valores de DtTemp intermedios entre el inicio y final de los datos (Figura 4.11), lo que podría significar que el modelo entrenado en G1 requiere una desviación más pequeña que el entrenado en G2 para clasificar como falla, lo que funciona bien en G1, pero no en G2; esto querría decir que el modelo entrenado en G2 es más confiable, pero a la vez requiere un mayor cambio en los datos para clasificar como falla.

Los resultados de regresión no son útiles, ya que los modelos no logran aprender un concepto de 'tiempo hasta la falla'; una explicación es que se deba a que los resultados parecen seguir una forma similar al DtTemp del respectivo generador, y DtTemp no cambia paulatinamente; esto podría significar que los datos no tienen información suficiente para predecir RUL. Otra explicación podría ser que el utilizar la cantidad de ventanas hasta la falla no es un sustituto equivalente a usar el tiempo hasta la falla, ya que no todas las ventanas ocurren inmediatamente después de otra, existen saltos de tiempo entre ventanas; esto podría generar problemas en el aprendizaje de los modelos, hasta el punto de no ser capaz de aprender el RUL.

Capítulo 6

Conclusiones

Al realizar un análisis de los datos recibidos, y calcular el delta de temperatura entre la entrada y salida de aceite, se nota que este delta (nombrado DtTemp) tiene cambios significativos hacia el final de la grabación, al menos para G1 y G2, los generadores confirmados con falla por la empresa; por lo que es probable que la diferencia de temperatura entre la entrada y salida de aceite sea un indicador importante en este tipo de falla de generador.

Los resultados indican que G1 y G2 probablemente presentaron una falla, lo cual concuerda con lo informado por la empresa; G3 presenta cambios en los datos del inicio y final de la grabación, pero debido a la forma de estos, se considera una anomalía, no una falla; G4 y G5, no presentan cambios considerables en los datos de funcionamiento, aunque G5 tiene una variación de DtTemp anómala, casi cíclica durante toda la grabación.

Se obtiene un modelo de Aprendizaje Profundo capaz de clasificar la existencia de una falla incipiente en G1 y G2, el cual fue entrenado en G2; y puede detectar la falla incipiente con 2 a 3 semanas de anticipación en ambos generadores. Este clasificador puede ser utilizado como detector de anomalía para generadores de este tipo, entregando un aviso cuando se detecte una falla incipiente para la que fue entrenado.

Se entrenan modelos de regresión para predecir la vida útil remanente (RUL) del generador, pero estos no entregan resultados satisfactorios. Las razones que se barajan como explicación de esto son las siguientes:

- Los resultados parecen seguir una forma similar al DtTemp del respectivo generador, y DtTemp no cambia paulatinamente.
- Se utiliza la cantidad de vuelos hasta la falla, y no el tiempo hasta la falla, lo que puede producir problemas, ya que no todas las ventanas ocurren inmediatamente después de otra, existen saltos de tiempo entre ventanas.

Bibliografía

[1] Carvalhoa T., SoaresaF., Vitac R., Francisco R., Bastoc J., Alcaláb S.(2019). A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, *137*, 106024.

[2] Tom M Mitchell. (1997). *Machine learning*. McGraw-Hill Science/Engineering/Math.

[3] Andriy Burkov. (2019). *The Hundred-Page Machine Learning Book*. Andriy Burkov.

[4] Ian Goodfellow and Yoshua Bengio and Aaron Courville. (2016). *Deep Learning*. MIT Press
http://www.deeplearningbook.org

[5] Christopher M. Bishop. (2006). *Pattern Recognition and Machine Learning*. Springer.

[6] Felipe Tobar. (2019). *Aprendizaje de Maquinas*.
https://github.com/GAMES-UChile/Curso-Aprendizaje-de-Maquinas

[7] Aurélien Géron. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. O'Reilly Media, Inc.

[8] Aviation Rule Making Committee (ARC). (2013). *A Report from the Portable Electronic Devices (PED) Aviation Rule Making Committee (ARC) to the Federal Aviation Administration (FAA).*
http://code7700.com/pdfs/ped_arc_final_report.pdf

Anexo A

Resultados Completos

En esta sección se muestran los resultados que no se encuentran en desarrollo, ya que se prefirió incluir solo lo esencial para mantener el flujo del análisis.

Las Tablas A.1 y A.2 corresponden al procedimiento común de estandarización, donde todos los Test se estandarizan con el mismo Train que se está utilizando, lo cual no entrega generalización entre G1 y G2.

Tabla A.1: Resultados promedio de clasificación entre generadores, estandarizando Test con Train del modelo, set Mid.

	F1 score - Mean - Mid				
Train\Test	G1	G2	G3	G4	G5
G1	97,4	64,3	26,3	58,4	50,1
G2	18,1	95,0	6,2	41,1	7,2
G3	61,5	64,0	90,0	53,4	68,0
G4	66,0	65,9	41,6	54,2	42,3
G5	69,8	70,4	49,8	66,1	31,9

Tabla A.2: Resultados promedio de clasificación entre generadores, estandarizando Test con Train del modelo, set Full.

	F1 score - Mean - Full				
Train\Test	G1	G2	G3	G4	G5
G1	95,6	64,4	45,7	62,2	55,2
G2	34,1	95,4	8,5	49,0	14,4
G3	61,2	62,1	89,6	52,2	67,9
G4	73,6	66,8	44,9	65,5	43,5
G5	73,0	67,8	47,4	64,0	43,2

Tabla A.3: Resultados de variación de semana Falla, para MLP, en G1.

	G1 - F1 score - MLP							
TrainTest	wk 17	wk 19	wk 20	wk 21	wk 22	wk 23	wk 24	wk 25
week 25	26,1	28,3	33,5	40,1	39	97,8	97,8	98,5
week 24	26,7	28,1	36	44,8	42,2	96,2	96,7	97
week 23	25,9	28,1	35,5	46,3	45,5	93,8	93,1	94,1
week 22	37,1	31,9	34,9	75,5	84,3	95,9	96,4	95,7
week 21	48,9	55,9	74,3	71,8	63,3	80,4	79,9	79
week 20	41,5	59,7	85,1	54,6	29,9	25,8	25,4	28,2
week 19	50,6	62,7	75,1	52,2	37,7	43	45	45,3
week 17	48,8	59,6	61,4	39,9	31,8	14,1	14,1	16,1

Tabla A.4: Resultados de variación de semana Falla, para RNN, en G1.

	G1 - F1 score - RNN							
TrainTest	wk 17	wk 19	wk 20	wk 21	wk 22	wk 23	wk 24	wk 25
week 25	24,9	28,9	32,6	41,2	42,9	96,7	94,6	98,5
week 24	25,9	28,5	33,6	43,8	45,6	99,1	98,2	99,3
week 23	26,2	28,5	34,2	42,3	44	96,2	96,1	94,8
week 22	37,9	34,3	38,7	71,9	79,5	90,7	91,9	90,1
week 21	42,7	54,7	74,6	73,2	64,1	80,7	79,8	78,1
week 20	38,6	59,8	75,1	47,4	30,6	21,6	22,4	22,6
week 19	45,8	58,5	69	45,6	32,3	20,9	20,5	24,3
week 17	46,7	51,6	56,9	43,5	37,8	15,9	16,1	17,5

Tabla A.5: Resultados de variación de semana Falla, para CNN, en G1.

	G1 - F1 score - CNN							
TrainTest	wk 17	wk 19	wk 20	wk 21	wk 22	wk 23	wk 24	wk 25
week 25	26,2	27,4	33,6	41,5	40,1	94,9	90,4	93,9
week 24	25,9	27,4	33,1	43	44,4	97,5	97,6	98,5
week 23	25,4	27,5	33,4	43,9	44,9	94,7	93,5	89,7
week 22	27,9	27,1	31,2	71,4	82,5	87,1	87,1	86,6
week 21	42	53,2	81,9	82,2	71,4	77,9	69,7	73,2
week 20	48,2	66,3	82,9	53,3	36,8	26,4	26,2	30
week 19	48,2	59,5	69,9	46,4	34,3	32,3	33,1	40,4
week 17	49,5	57,7	60,9	42,4	38,4	21,4	18,5	18,8

Tabla A.6: Resultados de variación de semana Falla, para MLP, en G2.

	G2 - F1 score - MLP							
TrainTest	wk 16	wk 17	wk 18	wk 19	wk 20	wk 21	wk 22	wk 23
week 23	41,6	32,8	36,4	38	53,7	97,8	98,4	98
week 22	41,8	32,6	36,6	37,3	53,5	97,2	95,2	95,5
week 21	42,6	32,9	36,6	37,7	54,4	97,2	90	93,8
week 20	47,4	64,9	65,3	60,3	65	71,9	68,3	52,6
week 19	56,4	41,5	59,7	63,7	59,3	45,4	59,7	52,3
week 18	40,6	69,1	69,3	60,1	67	79,8	69,5	77,2
week 17	48,1	72,5	46,4	36,7	47,6	26,7	27,6	24,5
week 16	56,8	64,5	53,3	44,4	52,9	32,2	33,7	30,4

Tabla A.7: Resultados de variación de semana Falla, para RNN, en G2.

	G2 - F1 score - RNN							
TrainTest	wk 16	wk 17	wk 18	wk 19	wk 20	wk 21	wk 22	wk 23
week 23	41,5	32,3	37,7	39	53	98,4	97,3	98
week 22	42,5	32,4	35,3	37,5	54,1	97	95,4	97
week 21	43,3	32,8	36,1	38	55	96,6	94	92,5
week 20	48,9	61,3	64,3	65,9	72,4	73,4	63,5	47,5
week 19	52,9	59,1	63	69,7	74,7	56,7	64,4	55,1
week 18	41,7	63,6	73,2	65	76	80,5	70,8	77,1
week 17	54,1	76,7	52,6	42,8	47,2	27,3	27,2	24,8
week 16	65,5	65,6	47,2	50,4	51	32,2	33,3	31,9

Tabla A.8: Resultados de variación de semana Falla, para CNN, en G2.

	G2 - F1 score - CNN							
TrainTest	wk 16	wk 17	wk 18	wk 19	wk 20	wk 21	wk 22	wk 23
week 23	43,9	32,6	37	39	55,5	97,6	96,9	97,4
week 22	41,6	32,8	36,1	38,4	56,2	96,8	96	97
week 21	41,6	33,1	37,1	38,4	54,6	97	92,8	92,8
week 20	47,8	67,1	62	63,3	72,4	79	66,1	55,6
week 19	56,4	54,8	54	64,9	61,5	41,2	55,8	52,7
week 18	42,9	67,2	72,4	64,5	74,2	79,5	72	80,5
week 17	57,9	75	51,3	42,7	46,9	31,1	30,6	26,5
week 16	65,4	70,7	50,5	54,5	60,7	38,2	39,9	40,8

Tabla A.9: Resultados de variación de semana Falla, para MLP, en G3.

	G3 - F1 score - MLP							
TrainTest	wk 6	wk 7	wk 8	wk 9	wk 10	wk 11	wk 12	wk 13
wk 13	48	47,7	50,1	65,6	63,6	82,5	86,9	91,4
wk 12	47,1	48,1	45,2	60	59,4	88,1	88,8	92,4
wk 11	48,1	58,2	47,9	62,7	56,4	89	80,4	81
wk 10	52,3	34,1	53,1	52,4	60,4	52	58	62,8
wk 9	50,6	37,4	57,8	65,1	65,6	53,3	63,5	61,7
wk 8	44,4	41,5	59,7	52,8	55,9	63	59,2	59,3
wk 7	44,4	44,7	38,7	39,2	35,5	41,6	43,1	39,3
wk 6	49,3	48,9	49,3	46,3	40	49,3	43	41,6

Tabla A.10: Resultados de variación de semana Falla, para RNN, en G3.

	G3 - F1 score - RNN							
TrainTest	wk 6	wk 7	wk 8	wk 9	wk 10	wk 11	wk 12	wk 13
wk 13	53,1	48,3	47,8	66	63,2	91,5	89,2	94,3
wk 12	50,5	49,1	47,1	60,8	54,6	88,2	92,3	93,2
wk 11	49,5	57,1	49	59,5	57,3	94,6	85,9	79,4
wk 10	53,9	38,6	58,1	62,3	66,3	56,6	69,7	69,4
wk 9	53,9	38,5	59	67,2	66,3	56,2	68,6	70,8
wk 8	54,7	54,6	61,1	62,7	63,7	69,8	62,4	64,7
wk 7	43,3	47	40,2	39,3	35,1	41,2	43,4	38,9
wk 6	55	65,3	56,3	51,2	39,9	72,1	49,5	43,1

Tabla A.11: Resultados de variación de semana Falla, para CNN, en G3.

	G3 - F1 score - CNN							
TrainTest	wk 6	wk 7	wk 8	wk 9	wk 10	wk 11	wk 12	wk 13
wk 13	49,9	48,9	47	62,2	60,3	91,7	91	95,1
wk 12	48,4	50,5	47,3	60,3	54,6	84,1	89,6	92,7
wk 11	49,2	54,4	48,7	59,6	54,7	93,8	87,9	88,7
wk 10	59,8	39,7	58,4	68,2	72,2	64,3	73,1	79
wk 9	51,7	40,4	60,3	68	69,1	62,5	71,7	76,3
wk 8	48,8	51	60	64	63,1	64,6	63,3	70
wk 7	44,4	50,2	41,8	39,2	35,7	43,8	43	39,2
wk 6	49,9	66,7	56,8	48,7	38	52,9	41,7	38,2

Tabla A.12: Entrenamiento de múltiples semanas para G1, MLP.

	G1 - F1 score - MLP							
TrainTest	wk 17	wk 19	wk 20	wk 21	wk 22	wk 23	wk 24	wk 25
week 23	26,2	28,4	36	42,6	40,1	97,1	95,4	96,5
weeks 22-23	28,6	28,4	34,5	72,3	74,6	98,2	98	98
weeks 21-23	43,7	56,6	77,2	86,8	71,3	91,8	91,9	91,3
week 22	45,9	41,7	53,8	77	82,1	94	94,5	94
weeks 21-22	48,3	54,2	82,3	89,3	72,4	95	94,9	93,3
weeks 20-22	51	61,5	80,4	76,8	58,3	86,5	87,1	86,1
week 21	53,9	57,7	77,9	76,4	62,1	80,2	83,9	83,8
weeks 20-21	52,7	67,4	84,8	75,3	57,1	85,2	81,8	85,8
weeks 19-21	49,5	63,8	83,3	73,9	57,6	86	85,1	84,5

Tabla A.13: Entrenamiento de múltiples semanas para G1, RNN.

	G1 - F1 score - RNN							
TrainTest	wk 17	wk 19	wk 20	wk 21	wk 22	wk 23	wk 24	wk 25
week 23	26,1	29,7	34,1	42,9	44,6	95,1	96,1	95,1
weeks 22-23	26	28,2	33,2	72,4	74,9	98,2	99,1	99,3
weeks 21-23	33,8	49,2	84,5	91,2	80	96,4	96	96
week 22	37,3	34,9	38	73,4	82,5	92,2	91	88,8
weeks 21-22	40,2	55,6	82,6	87,4	74	94,8	94,7	92,3
weeks 20-22	45,6	62,1	82,6	82	67,9	89,5	89,4	88,1
week 21	44,3	54,1	74,5	77,4	65,7	83	74,6	77,5
weeks 20-21	40,1	60,7	86,7	80,6	63,9	86	87,1	85,2
weeks 19-21	48,7	68,6	80,7	76,7	55,3	81,4	80,1	78,2

Tabla A.14: Entrenamiento de múltiples semanas para G1, CNN.

	G1 - F1 score - CNN							
TrainTest	wk 17	wk 19	wk 20	wk 21	wk 22	wk 23	wk 24	wk 25
week 23	27,1	28,2	35,2	47,8	49,4	96,2	96,9	97
weeks 22-23	26,7	28	34,1	74,3	75,6	94,3	95	95,6
weeks 21-23	38	49,4	82,2	87,4	77,1	92,9	92,2	92,6
week 22	31,4	28,9	32,2	75,8	88,6	91,7	90,8	90,6
weeks 21-22	31,1	42,1	80,8	90	80,7	92,7	92,4	90,7
weeks 20-22	43	53,8	84,3	85,9	70,6	89,6	88,2	85,5
week 21	39,9	52	74,8	72,2	62,6	74,3	71,1	70,9
weeks 20-21	36,3	59,6	85,4	71,2	54,5	75	70,8	73,7
weeks 19-21	47,1	62,9	82,4	79,8	59,5	77,6	71,4	77,7

Tabla A.15: Entrenamiento de múltiples semanas para G2, MLP.

	G2 - F1 score - MLP							
TrainTest	wk 16	wk 17	wk 18	wk 19	wk 20	wk 21	wk 22	wk 23
week 22	41,4	32,9	36,5	40,2	55,8	96,8	96,3	97
weeks 21-22	41,5	32,6	36,3	37,4	54,3	97,8	95,3	97
weeks 20-22	45	71,8	59,5	55,5	67,9	85,4	81,9	86,3
week 21	42,4	32,7	36,7	40,8	56,5	98	95	96,2
weeks 20-21	46,6	47,2	46,7	51,6	66,1	90,8	85,3	87,4
weeks 19-21	43,7	64,9	66,8	61,4	69,5	87,6	77	79,7

Tabla A.16: Entrenamiento de múltiples semanas para G2, RNN.

	G2 - F1 score - RNN							
TrainTest	wk 16	wk 17	wk 18	wk 19	wk 20	wk 21	wk 22	wk 23
week 22	42,5	32,8	36,8	39,6	55	97,2	95,8	95,7
weeks 21-22	41,7	32,5	36,1	37,2	54,7	98	95,4	96,2
weeks 20-22	45,3	62,7	60,1	60,7	73,9	91,7	89,5	92,1
week 21	41,8	32,6	35,8	36,8	53,1	96,2	93,4	93,8
weeks 20-21	43,6	54,5	55,9	59,1	74,5	88,5	77,2	83,2
weeks 19-21	50,4	63,1	63,9	64,5	80,9	89,8	83	86,5

Tabla A.17: Entrenamiento de múltiples semanas para G2, CNN.

	G2 - F1 score - CNN							
TrainTest	wk 16	wk 17	wk 18	wk 19	wk 20	wk 21	wk 22	wk 23
week 22	42,5	32,8	37,1	38,8	55,8	97,4	95,6	96,9
weeks 21-22	41,6	34	38,6	38	56,7	98	96,7	98,2
weeks 20-22	42,4	55,6	59,4	64,4	72,5	91,8	87,2	91,6
week 21	42,5	32,8	37,9	38,4	56,2	98,4	95,8	97
weeks 20-21	46,3	58	55,2	57,6	68,8	88	79,2	84,9
weeks 19-21	54,2	62	63,6	63,4	74,8	86,8	76,6	84,6

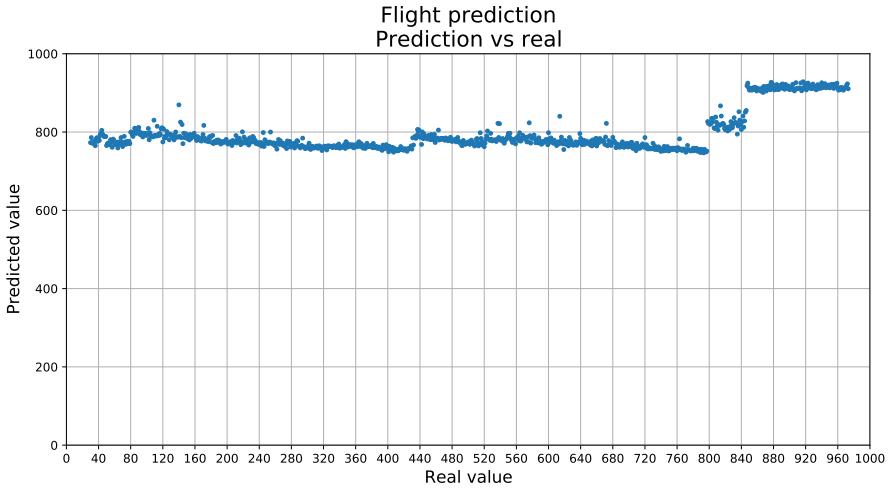

Figura A.1: Predicción de RUL de modelo MLP en G1.

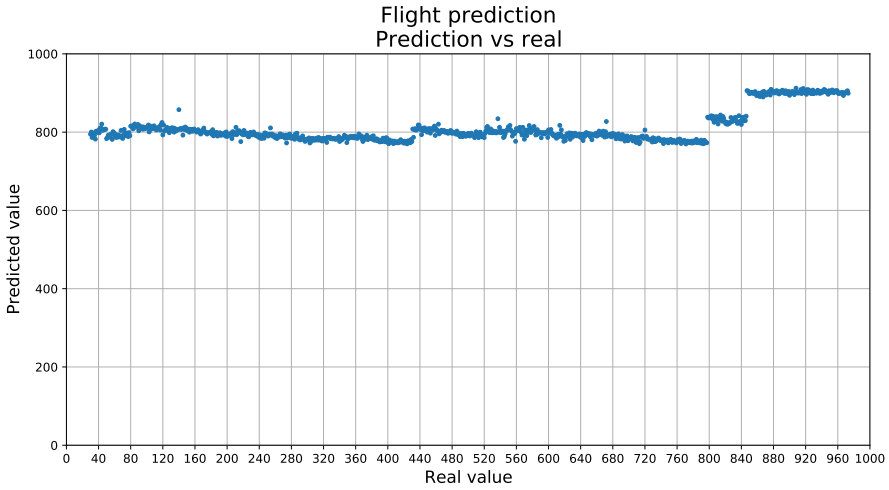

Figura A.2: Predicción de RUL de modelo RNN en G1.

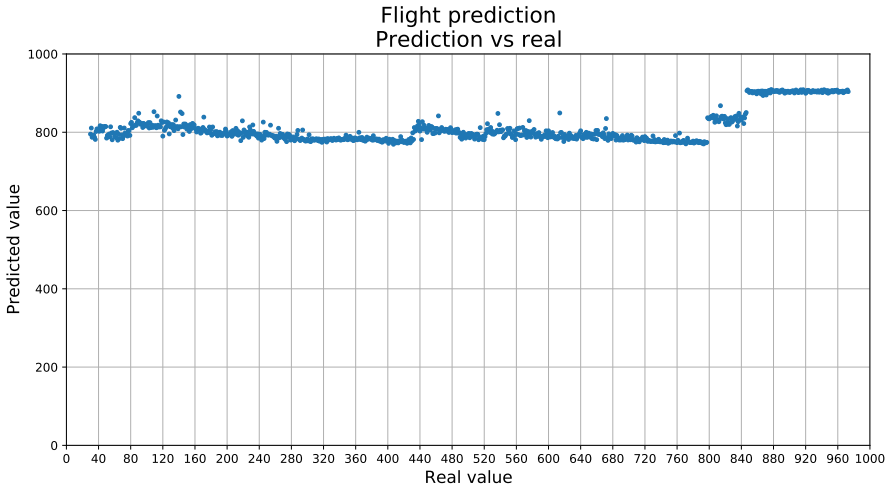

Figura A.3: Predicción de RUL de modelo CNN en G1.

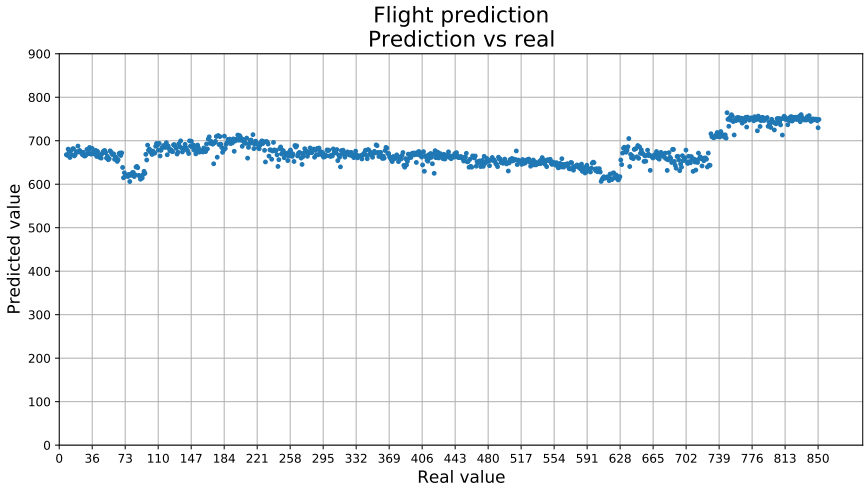

Figura A.4: Predicción de RUL de modelo MLP en G2.

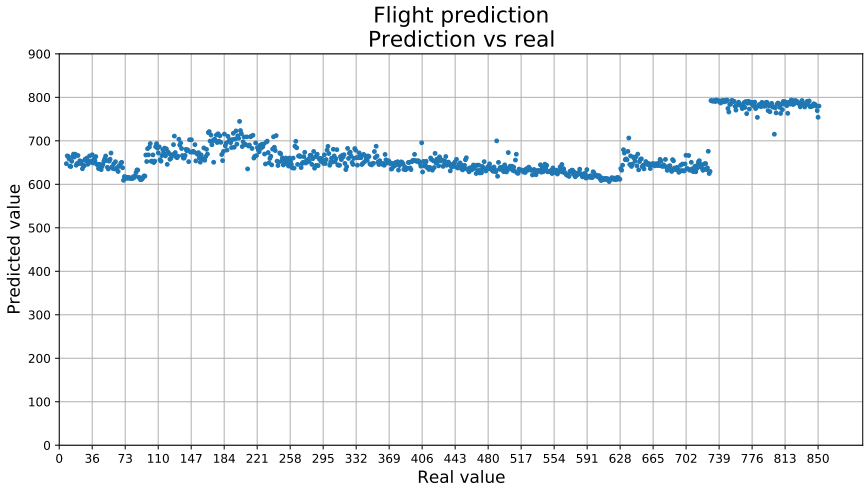

Figura A.5: Predicción de RUL de modelo RNN en G2.

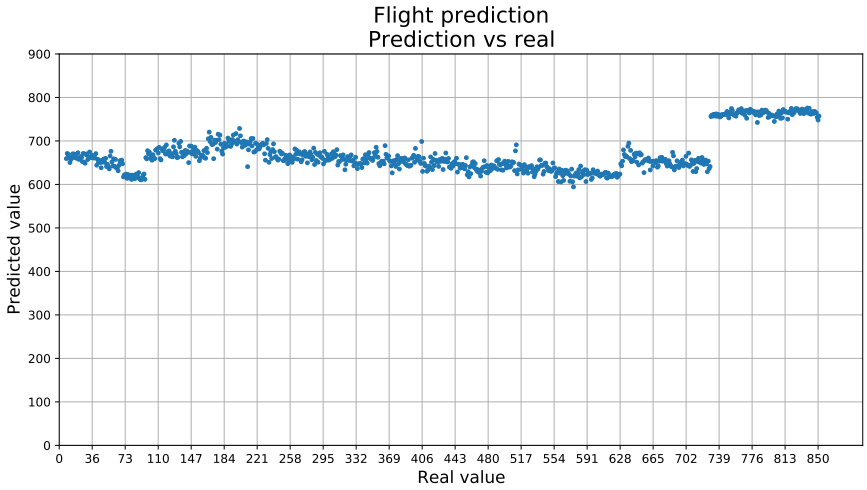

Figura A.6: Predicción de RUL de modelo CNN en G2.

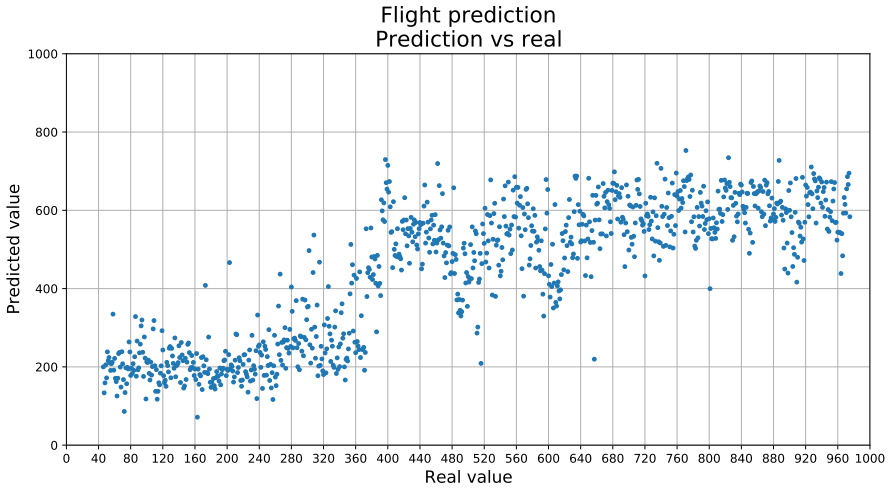

Figura A.7: Predicción de RUL de modelo MLP en G3.

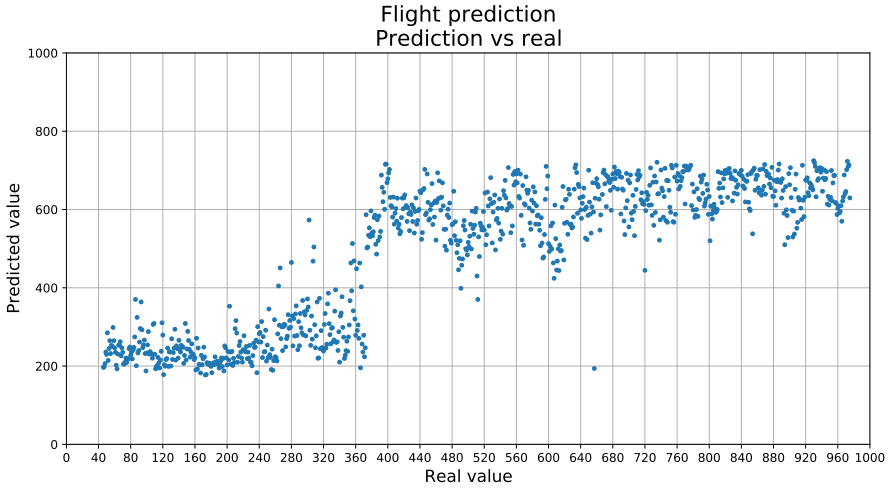

Figura A.8: Predicción de RUL de modelo RNN en G3.

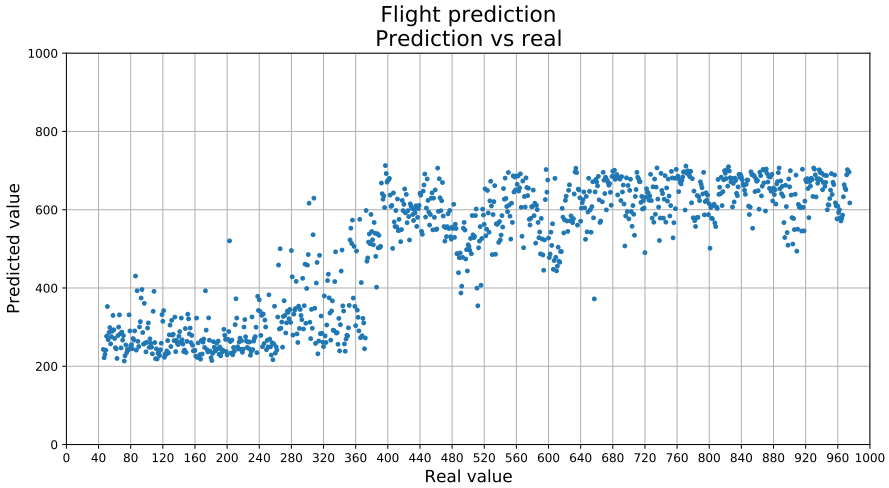

Figura A.9: Predicción de RUL de modelo CNN en G3.

Anexo B

Arquitecturas de red

Esta sección incluye las arquitecturas de las redes de Aprendizaje Profundo utilizadas.

Código B.1: Arquitectura TD-MLP Clasificación

```
model = tf.keras.Sequential()
drop = 0.3
model.add(layers.Dense(6, activation = 'relu', input_shape=input_size))
model.add(layers.BatchNormalization())
model.add(layers.Dense(6, activation = 'relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(drop))
model.add(layers.Dense(6, activation = 'relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(drop))
model.add(layers.Dense(6, activation = 'relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(drop))
model.add(layers.Dense(6, activation = 'relu'))
model.add(layers.BatchNormalization())
model.add(layers.Flatten())

model.add(layers.Dropout(drop))
model.add(layers.Dense(512, activation = 'relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(drop))
model.add(layers.Dense(128, activation = 'relu'))
model.add(layers.BatchNormalization())

model.add(layers.Dense(2, activation='softmax'))
```

Código B.2: Arquitectura CNN Clasificación

```
model = tf.keras.Sequential()
drop=0.3
model.add(layers.Conv2D(4, (5, 1), padding='same',input_shape=input_size))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 1)))

model.add(layers.Conv2D(8, (5, 1),padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(8, (5, 1),padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 1)))

model.add(layers.Flatten())
model.add(layers.Dropout(drop))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.BatchNormalization())

model.add(layers.Dense(2, activation='softmax'))
```

Código B.3: Arquitertura RNN Clasificación

```
model = tf.keras.Sequential()
drop=0.0
model.add(layers.InputLayer(input_shape=input_size))
model.add(layers.GRU(16, activation='relu'))
model.add(layers.BatchNormalization())

model.add(layers.Dropout(drop))
model.add(layers.Dense(32, activation = 'relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(drop))
model.add(layers.Dense(16, activation = 'relu'))
model.add(layers.BatchNormalization())

model.add(layers.Dense(2, activation='softmax'))
```

Las siguientes arquitecturas mejoradas fueron utilizadas tanto para clasificación como para regresión, cambiando solo la capa de salida en cada una.

Código B.4: Arquitectura TD-MLP mejorada

```
model = tf.keras.Sequential()
drop=0.3
model.add(layers.InputLayer(input_shape=input_size))

model.add(layers.Dense(6, activation = 'relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dense(6, activation = 'relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(drop))
model.add(layers.Dense(6, activation = 'relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(drop))
model.add(layers.Dense(6, activation = 'relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dense(6, activation = 'relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(drop))
model.add(layers.Dense(6, activation = 'relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(drop))
model.add(layers.Dense(6, activation = 'relu'))
model.add(layers.BatchNormalization())
model.add(layers.Flatten())

model.add(layers.Dropout(drop))
model.add(layers.Dense(16, activation = 'relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(drop))
model.add(layers.Dense(8, activation = 'relu'))
model.add(layers.BatchNormalization())

#Clasificacion
model.add(layers.Dense(2, activation='softmax'))

#Regresion
model.add(layers.Dense(1, activation='relu'))
```

Código B.5: Arquitectura RNN mejorada

```
model = tf.keras.Sequential()
drop=0.2
model.add(layers.InputLayer(input_shape=input_size))
model.add(layers.GRU(16))
model.add(layers.LayerNormalization())

model.add(layers.Dropout(drop))
model.add(layers.Dense(32, activation = 'relu'))
```

```
model.add(layers.BatchNormalization())
model.add(layers.Dropout(drop))
model.add(layers.Dense(16, activation = 'relu'))
model.add(layers.BatchNormalization())

#Clasificacion
model.add(layers.Dense(2, activation='softmax'))

#Regresion
model.add(layers.Dense(1, activation='relu'))
```

Código B.6: Arquitectura CNN mejorada

```
model = tf.keras.Sequential()
drop=0.3
model.add(layers.InputLayer(input_shape=input_size))
model.add(layers.Reshape((input_size[0], input_size[1], 1)))

model.add(layers.Conv2D(4, (5, 2), padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 1)))
model.add(layers.Conv2D(8, (5, 2),padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(8, (5, 2),padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 1)))
model.add(layers.SeparableConv2D(16, (5, 2),padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.SeparableConv2D(16, (5, 2),padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.GlobalAveragePooling2D())

model.add(layers.Flatten())
model.add(layers.Dropout(drop))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(drop))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.BatchNormalization())

#Clasificacion
model.add(layers.Dense(2, activation='softmax'))

#Regresion
model.add(layers.Dense(1, activation='relu'))
```