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

ANÁLISIS EXPLORATORIO DE MODELOS DE APRENDIZAJE DE MÁQUINAS PARA LA ESTIMACIÓN DEL TIEMPO DE VIDA REMANENTE DE UNA UNIDAD PROPULSORA DE AERONAVE

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MECÁNICO

AGUSTÍN ELISEO BELTRÁN CARVAJAL

PROFESOR GUÍA:
ENRIQUE ANDRÉS LÓPEZ DROGUETT

MIEMBROS DE LA COMISIÓN:
VIVIANA MERUANE NARANJO
JUAN TAPIA FARIAS

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL MECÁNICO
POR: AGUSTÍN ELISEO BELTRÁN CARVAJAL
FECHA: ABRIL 2020
PROF. GUÍA: ENRIQUE ANDRÉS LÓPEZ DROGUETT

ANÁLISIS EXPLORATORIO DE MODELOS DE APRENDIZAJE DE MÁQUINAS PARA LA ESTIMACIÓN DEL TIEMPO DE VIDA REMANENTE DE UNA UNIDAD PROPULSORA DE AERONAVE

En el campo de la industria y el mantenimiento, nos podemos encontrar con fallas cuyo origen o desarrollo es desconocido, o bien de las que no se cuenta con la información suficiente para diagnosticar o tratar dicha falla. En este campo, el aprendizaje de máquinas se presenta como una herramienta capaz de aprender relaciones en donde los métodos convencionales son incapaces, utilizando la información disponible.

En este contexto, el área de mantenimiento de una empresa de aeronáutica presenta en algunas unidades específicas una falla, de la cual no saben cómo se desarrolla o cómo evitar que se manifieste de manera crítica. La presente tesis consiste en verificar preliminarmente si utilizando datos que se relacionan de manera indirecta con la unidad en falla se puede predecir en cuánto tiempo se producirá la falla.

Para ello, se decide realizar cinco tipos de modelos distintos (tres de ellos redes neuronales), con el fin de verificar la existencia de tendencias que permitan generar una regresión que se aproxime a la falla, para posteriormente determinar si es factible o no realizar dicha regresión con la información disponible.

Finalmente, se obtiene que los modelos reconocen una tendencia en los datos, pero su desempeño no es suficiente para representar resultados aceptables para lo requerido. Es por esto por lo que se propone continuar su desarrollo utilizando de base lo confirmado en esta tesis para generar un modelo más robusto empleando modelos más complejos y bases de datos más extensas. Además, de la falla se consiguió detectar la formación de dos clusters que podría relacionarse al origen de la falla estudiada, pero que con la información disponible no puede confirmarse.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Introducción

La industria aeronáutica comenzó su desarrollo en torno al año 1919, época en que la tecnología era predominantemente la impulsión con viento. Pero fue posterior a la Segunda Guerra Mundial que experimentó una fase de crecimiento importante con la entrada de sistemas de generación de energía, permitiendo el transporte de pasajeros en grandes cantidades y de manera económicamente factible. En los años 1980 se incorpora la tecnología IDG (*Integrated Drive Generator*)[8], que pese a no ser la tecnología más moderna en propulsión de aeronoaves, es la que se trata en el presente documento.

En este rubro, uno de los aspectos más relevantes son de optimización y mantenimiento, dado que es fundamental garantizar la operación de las aeronaves al mínimo costo posible y cumpliendo exigentes normas de seguridad. Es por lo mismo que en empresas de fabricación de aviones (como Boeing y Airbus) ponen gran énfasis en el desarrollo de tecnología e innovación para lograr alcanzar mejores niveles de eficiencia, mientras que las aerolíneas preparan grandes equipos de mantención para garantizar la operación, implicando un mínimo riesgo para la gestión y sus pasajeros a la vez que disminuyen sus costos operacionales.

Dadas las condiciones de operación y problemas de diseño o fabricación, estos equipos presenten irregularidades o fallas en su funcionamiento, las cuales son atendidas por los equipos de mantenimiento de cada empresa. Por lo mismo, existe un amplio desarrollo académico en lo que respecta a la prevención, detección y corrección de las múltiples fallas que pueden llegar a presentarse, lo cual no aplica sólo al sector aeronáutico, sino que al sector industrial en su totalidad.

Las fallas pueden presentar distintos grados de complejidad dependiendo de dónde se manifiesten. Así, por ejemplo, para las unidades de propulsión y generación se ha buscado exhaustivamente mejorar su nivel de confiabilidad y eficiencia, disminuir su peso y dimensiones y mantener su costo en un nivel razonable. Sin embargo, son sistemas difíciles de monitorizar con mecanismos de falla altamente variables y diversos. Son precisamente éstos casos, donde la causa del problema no es evidente o no se puede identificar con claridad, donde se presenta un gran problema para el equipo de mantenimiento, dado que sin saber qué impulsa la falla, no se puede adoptar una reacción adecuada. Esto, en el peor de los casos, puede vulnerar la seguridad de la nave en vuelo o tener impactos económicos muy negativos para la aerolínea.

Lo anterior representa un desafío bastante complejo para manejarlo con las herramientas existentes actualmente, por lo que es necesario aplicar nuevos métodos para afrontar este desafío. Es en este contexto donde el *aprendizaje de máquinas* se presenta como una herra-

mienta interesante, dado que puede ser utilizado para encontrar relaciones entre fallas y las variables del sistema sin requerir un conocimiento completo y detallado. Además, con los nuevos equipos y plataformas informáticas, estos análisis pueden realizarse en tiempos razonables, permitiendo abarcar más alternativas que con las metodologías más tradicionales de mantenimiento.

Lo anterior no ha pasado inadvertido en el área del mantenimiento, donde la flexibilidad y velocidad de los análisis que pueden hacer estos algoritmos ha permitido monitorizar sistemas de manera completa y continua, permitiendo llegar de manera más oportuna a la falla y actuar con mayores márgenes de tiempo y seguridad. En este contexto se desarrolla la presente tesis, donde se manifiesta una falla crítica importunando la operación de aeronaves de una línea aérea.

Capítulo 1

Motivación y objetivos

1.1. Motivación

La identificación y comprensión de los mecanismos de fallas es uno de los actuales desafíos para mejorar el desempeño y confiabilidad de las unidades generadoras, como se indica en el paper *Electrical Power Generation in Aircraft: Review, Challenges and Opportunities*[8]. Para ello, la industria aeronáutica busca asimilar las nuevas herramientas de aprendizaje profundo, que han demostrado obtener mejores resultados y de manera instantánea, a diferencia de los métodos clásicos que son más lentos y menos confiables.

La tarea anterior es altamente compleja, ya que como indican Klein y Bergmann [6], existe múltiples fuentes de falla por la cantidad de mecanismos y componentes que emplea en su funcionamiento. Esto implica que no siempre se cuenta con sensores monitorizando directamente los componentes que fallan, por lo que los datos con que se cuenta suelen ser de equipos simpáticos.

Con lo anterior, preparar una base de datos adecuada representa un desafío en sí, ya que las fallas suelen ser eventos aislados raramente monitorizados de manera directa, por lo que la base de datos termina siendo forzosamente reducida y desbalanceada, además de requerir una gran cantidad de preprocesamiento para extraer la información más relevante al objetivo que se busca cumplir.

Para el desarrollo de la presente tesis, como se cuenta con una base de datos limitada y una falla incierta, se busca realizar una exploración preliminar a datos operacionales para modelar una falla crítica en una aeronave de la cual no se conoce su origen o como manejarla. El costo que implica esta falla para la aerolínea es muy elevado y cuentan con datos limitados, como temperatura de la unidad en falla y vibraciones de equipos solidarios. Es por ello por lo que será especialmente relevante identificar patrones y tendencias que representen una primera aproximación para definir el problema, de modo que se avance hacia la detección, comprensión y manejo de la falla presentada.

1.2. Objetivos

Objetivo general

Al finalizar este trabajo, se espera presentar resultados exploratorios para la predicción del tiempo remanente en la manifestación de una falla crítica en unidades de aeronaves, empleando una serie de modelos de aprendizaje de máquinas y utilizando datos operacionales de la unidad y de equipos simpáticos.

Objetivos específicos

- Definir diversos criterios para determinar las etiquetas de los datos.
- Identificar en los datos patrones que puedan dar indicios de la falla.
- Plantear el problema para abordarlo con objetivos de clasificación y regresión.
- Probar distintos modelos para verificar su comportamiento. Se debe programar modelos para usar de referencia y se debe utilizar modelos de redes neuronales.
- Estudiar factibilidad de modelos que se aplican.
- Validar los modelos con un set de datos adicional, correspondiente a una unidad sana.

1.3. Alcances

Para el desarrollo de la tesis, se recibieron tres bases de datos de tres unidades distintas, los cuales se componen de las variables de altura, temperatura de aceite y valor RMS de vibración de un equipo solidario. Dos de las bases de datos manifiestan la falla, mientras que la tercera corresponde a una unidad sana y es utilizada para validar la respuesta de los modelos, pero no para el proceso de entrenamiento.

- Los datos de los equipos son provistos por la empresa.
- Los datos utilizados corresponden a temperatura, presión y altura con una tasa de muestreo de 1[muestra/s].
- De las variables recibidas, la altura no se utilizará en los modelos, sino sólo para determinar el estado de vuelo del avión.
- Ya se confirmó la existencia de la falla y su detección en los datos mediante un detector de anomalías.
- De momento, se utilizará los datos medidos durante el vuelo estable de la aeronave, omitiendo el despegue y aterrizaje.
- Se da por supuesto que la falla se desarrolla en un período de tiempo extenso y no que ésta se manifiesta de manera repentina.
- No se impone un nivel mínimo a superar con las métricas, dado que no se puede asegurar que la información existente sea suficiente para modelar el comportamiento de la falla.
- De la falla se tiene sólo el momento en que se manifiesta (última medición). El origen, sus características y el momento en que empieza a desarrollarse son desconocidos.
- Se dará por supuesto que la degradación es lineal con valor máximo en la primera muestra y mínimo en la última.

- Al ser un análisis exploratorio, no se contempla la integración del modelo resultante al sistema actual de la empresa.

Capítulo 2

Antecedentes

2.1. Aprendizaje de máquinas

Una de las ramas más interesantes y de mayor potencial en el campo de la inteligencia artificial es el aprendizaje de máquinas, el cual se define como un *estudio sistemático de algoritmos y sistemas que mejoran su conocimiento o desempeño con experiencia* [2]

En otras palabras, el aprendizaje de máquinas es un sistema o **modelo** que aprende a entregar o **predecir** la respuesta adecuada a una tarea específica utilizando la información disponible, en un proceso llamado **entrenamiento**. Para poder abordar las tareas, la situación se interpreta como un problema matemático, donde se optimiza para determinar la probabilidad de que cada respuesta sea la correcta.

Como se puede inferir del párrafo anterior, para poder resolver una tarea con aprendizaje de máquinas se requiere de tres elementos esenciales: una base de datos etiquetada, un modelo y un objetivo a cumplir. Las etiquetas de la base de datos corresponden a la respuesta esperada de los datos de entrada y el modelo un algoritmo matemático que resuelve el problema probabilístico. Un buen modelo es capaz de entregar una respuesta adecuada para cada *input*, a pesar de que éste no se encuentre en la base de datos utilizada para entrenar.

2.1.1. Tipos de aprendizajes según las etiquetas de la base de datos

Dependiendo de si la respuesta que se busca obtener de los datos de entrenamiento es conocida o no, podemos tener básicamente los siguientes tipos de aprendizajes para la red neuronal [3]:

1. Entrenamiento supervisado: Para los datos de entrenamiento, las respuestas deseadas son conocidas y llamadas *etiquetas*.
2. Entrenamiento no supervisado: Los datos de entrenamiento no están etiquetados.
3. Entrenamiento semi supervisado: Sólo una porción de los datos de entrenamiento se encuentran etiquetados.
4. Entrenamiento reforzado: Los datos de entrenamiento no se encuentran etiquetados,

pero se implementa con un sistema de recompensas y penalizaciones. En este caso, se busca maximizar los beneficios obtenidos.

Los problemas más comunes corresponden a entrenamiento supervisado, dado que son más simples de plantear y resolver, además de que la respuesta suele ser adquirible directamente.

Por otro lado, los problemas de aprendizaje reforzado son más complejos, dado que se debe definir el sistema de recompensas adecuadamente para que el modelo pueda aprender. Estos últimos son ampliamente utilizados en tareas que buscan replicar la respuesta humana o que se requiere sean adaptables a la situación que se enfrenta.

2.1.2. Tipos de tareas que pueden realizarse con aprendizaje de máquinas

Al desarrollar un modelo de aprendizaje de máquinas, es necesario definir el tipo de respuesta que esperamos de dicho modelo. De este modo, a continuación se enumeran tres de los tipos de tareas básicos según las características de la respuesta esperada, que además son los más utilizados en el aprendizaje de máquinas:

1. Clasificador binario: El modelo entrega una respuesta binaria.
2. Clasificador multiclase: El modelo entrega un valor natural como respuesta.
3. Regresión: El modelo entrega un valor real como respuesta.

Un ejemplo más concreto se presenta con una de sus aplicaciones más sencillas es la detección de spam en correos, donde la respuesta del modelo es la probabilidad de que el mail corresponda a uno no deseado. En este caso, la tarea es de **clasificación binaria** (es o no spam), pero se puede ir un poco más allá y separar los correos en grupos de similares características en una **clasificación multiclase**. Una tarea de **regresión** para este caso podría corresponder a un ordenamiento de los correos según, por ejemplo, su importancia.

2.2. Redes Neuronales Artificiales

Los modelos de neuronas artificiales son un tipo de modelo de aprendizaje de máquinas que se inspira en la neurona biológica, donde cada unidad se conecta a muchas otras, generando una red. Los últimos años ha tomado gran relevancia, en especial por el desarrollo tecnológico en el área de la computación. Este tipo de modelos ha conseguido obtener resultados sorprendentes, caracterizándose por ser altamente flexibles y eficientes.

Como se define en la publicación *"Deep Learning in Neural Networks: An Overview"*[10], en este tipo de modelos existe una unidad básica llamada **neurona**, la cual realiza una operación matemática o **función de activación** sobre un valor. Cada neurona se conecta a otras formando una red, teniendo de manera simultánea múltiples entradas y salidas. Las conexiones tienen un valor (**weight**) que pondera la salida de la neurona anterior y se suma al resto de las conexiones que entran a la neurona siguiente. El proceso de entrenamiento en este caso, busca valores para los pesos de tal manera que la red entrega la respuesta que se desea, empleando un algoritmo que se conoce como **back propagation**.

Este tipo de redes han sido modificadas y complementadas para dar origen a todo tipo de modelos y estructuras nuevas, permitiendo ampliar el potencial de esta área. A continuación, se describe las capas que se utilizan en el desarrollo de la presente tesis.

2.2.1. Capa *fully connected*

La red neuronal fully connected es la red neuronal más básica que surge directamente de la definición de neurona. Consiste en una serie de neuronas ordenadas en **capas**, donde cada una de ellas se conecta a la totalidad de las neuronas de la capa anterior y de la capa siguiente. Este tipo de ordenamiento es utilizado en prácticamente la totalidad de los modelos que se han desarrollado, dado que permite estructurar de manera clara y sencilla la arquitectura de la red neuronal.

La capa de entrada se conecta directamente con la fuente de los datos, mientras que la capa de salida entrega un valor que representa la respuesta de la red. A las capas intermedias se les conoce como **capas ocultas**.

2.2.2. Capa de *Batch Normalization*

Uno de los problemas al entrenar redes, es que la distribución y los parámetros de cada capa del modelo neuronal cambian durante el entrenamiento. Esto relentiza el entrenamiento y complejiza el modelo para casos no lineales, dado que la red debe adaptarse a estas variaciones.

Para solucionar esto, se propuso normalizar cada mini-batch de datos que entra a la red, permitiendo darle más estabilidad al proceso de entrenamiento, como lo proponen Sergey Ioffe y Christian Szegedy[5]. Para ello, cada batch de datos se normaliza para que su promedio sea 0 y su desviación estándar 1. De este modo, sea μ el promedio del batch y σ^2 su varianza, la ecuación 2.1 muestra la fórmula de normalización aplicada a cada dato, sobre el que posteriormente se realizar un proceso de escalado y desplazamiento.

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mu_B}{\sqrt{\sigma_B^2}} \tag{2.1}$$

2.2.3. Capa de *Dropout*

Los modelos de redes neuronales poseen una gran cantidad de parámetros que permiten representar relaciones complejas entre la entrada y la salida. Esto es un gran beneficio, pero también da lugar a que estas relaciones aprendidas sean el ruido de los datos o estados memorizados, permitiendo el problema del sobre entrenamiento u ***overfitting***.

Una de las propuestas para reducir el *overfitting* es generar muchos modelos de igual arquitectura y ponderar sus hiperparámetros, de modo que el modelo final sea incapaz de identificar el ruido. El problema con esto es que es altamente costoso en recursos computacionales, por lo que es factible para pequeños modelos. Otra posibilidad es entrenar varias redes de arquitectura distinta y combinarlos en el modelo final, pero con la cantidad acotada de

datos para entrenamiento y lo complejo que es optimizar un modelo vuelve a esta alternativa poco factible.

Una solución que reduce el *overfitting* y aplica los dos puntos anteriores evitando los inconvenientes que conlleva es el *dropout*, como se indica en la publicación *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*[11] consiste en remover neuronas del modelo de manera aleatoria, entrenar una etapa del modelo y repetir el proceso. Esto da a lugar a sub modelos del principal, permitiendo las ventajas de entrenar "varios modelos distintos" y combinarlos a la vez, con un menor costo computacional y con el mismo set de datos de entrenamiento.

Para realizarlo, la red recibe como input la probabilidad de que cada neurona sea descartada de cada etapa de entrenamiento y en cada iteración de entrenamiento, se verifica antes qué neuronas se mantienen en el modelo y cuáles son temporalmente descartadas.

2.3. Red neuronal convolucional

Las redes neuronales de tipo convolucional fueron propuestas en el paper *Gradient Based Learning Applied to Document Recognition* en 1998[7] y se inspiran en el modo en que las neuronas biológicas permiten el análisis de imágenes, las cuales procesan pequeños campos de la imagen con la posibilidad de superponerse entre ellos[3]. Al aplicarlo en el campo del aprendizaje de máquinas, se puede reducir la dimensión de los datos de entrada (reduciendo la cantidad de parámetros del modelo) al extraer las relaciones locales de cada campo. En este caso, cada campo es tratado como una matriz de píxeles contenida en la imagen.

Para realizar esta tarea computacionalmente, se emplea *filtros* o matrices de números definidas, que son operados por convolución con los campos de la capa anterior y entrega un valor representativo de dicho campo. En una capa convolucional, se aplica varios filtros a la matriz de datos de la capa anterior, entregando una representación simplificada y más relevante de los datos, como se muestra en la 2.1.

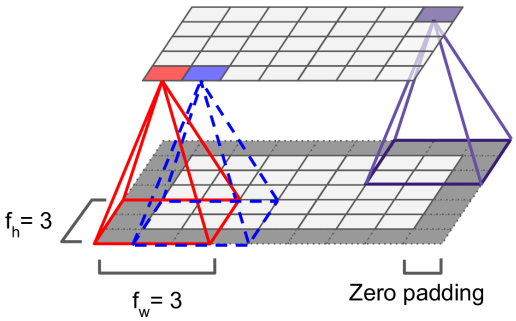

Figura 2.1: Representación gráfica del proceso de convolución aplicado a una matriz de entrada, donde de campos de 3x3 píxeles se obtiene un valor. Fuente: *Hands On Machine Learning with Scikit-Learn & Tensorflow*[3].

En la presente tesis, se utiliza la abreviación *CNN* para referirse a las redes neuronales convolucionales y se trabaja con un caso particular, donde la entrada es un vector de datos y no una matriz, realizando convoluciones en una dimensión (*Conv1D*).

2.4. Redes Neuronales Recurrentes

Las redes neuronales recurrentes son un tipo de arquitectura especializada, con la principal ventaja de que permite realizar un análisis considerando la secuencia temporal de los datos. Para conseguir esto, se agregan conexiones a las salidas de la misma neurona en iteraciones anteriores.

2.4.1. *Long Short Term Memory*

Estos modelos abreviados *LSTM* tiene su origen en la propuesta de Sepp Hochreiter y Jürgen Schmidhuber de 1997[4] y son un tipo de redes recurrentes que aplican dos tipos de conceptos: la *memoria a corto plazo* y la *memoria a largo plazo*. La unidad neuronal de las redes LSTM se componen de cuatro elementos: puerta de entrada, puerta de salida, una puerta de olvido y una unidad de memoria. La unidad de memoria es capaz de almacenar y entregar información de iteraciones anteriores, mientras que el resto de los elementos regulan el flujo de información a través de dicha unidad.

La figura 2.2 es una representación de una unidad neuronal utilizada en una red *LSTM*, donde se puede observar que:

- La señal $c_{(t)}$ representa la memoria a largo plazo luego de realizar la iteración t.
- La señal $h_{(t)}$, que es igual a la respuesta de la unidad $y_{(t)}$, corresponde a la memoria a corto plazo para la iteración t.
- La señal $g_{(t)}$ corresponde al primer procesamiento de información entre la memoria a corto plazo $h_{(t-1)}$ y la señal de entrada $x_{(t)}$, para posteriormente ser agregada a la memoria a largo plazo y procesada en la respuesta final $y_{(t)}$.
- Las señales $f_{(t)}$, $\mathrm{i}_{(t)}$ y $o_{(t)}$ son los activadores de las puertas de olvido, entrada y salida respectivamente.

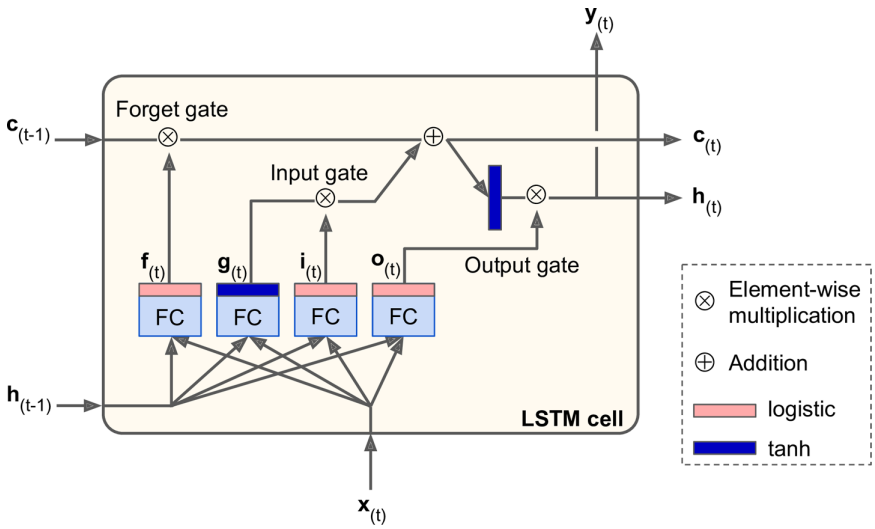

Figura 2.2: Representación gráfica de la interacción los agentes que participan en una unidad neuronal de una capa *LSTM*. Fuente: *Hands On Machine Learning with Scikit-Learn & Tensorflow*[3].

2.4.2. *Gated Recurrent Unit*

Las *GRU* fueron propuestas el 2014 por Kyunghyun Cho et al.[1] y consisten en una versión simplificada de la red *LSTM*, que si bien no la supera para tareas complejas como procesamiento de texto, es más simple y manejable por presentar menos parámetros.

Su arquitectura cuenta con los mismos elementos que la red *LSTM*, pero carece de una puerta de salida. Esto otorga la misma mecánica de análisis con una red más sencilla, pero sacrificando un poco de potencial de procesamiento. Su representación gráfica se presenta en la figura 2.3, donde las principales diferencias con la red *LSTM* son:

- Las memorias a corto y largo plazo son unificadas en la señal $h_{(t)}$, que es igual a la respuesta de la unidad $y_{(t)}$.
- Tanto la puerta de olvido como la puerta de entrada son controladas por la señal $z_{(t)}$ y son de comportamiento opuesto.
- No existe puerta de salida, pero existe una puerta de entrada adicional sobre la señal $h_{(t)}$, controlada por $r_{(t)}$.

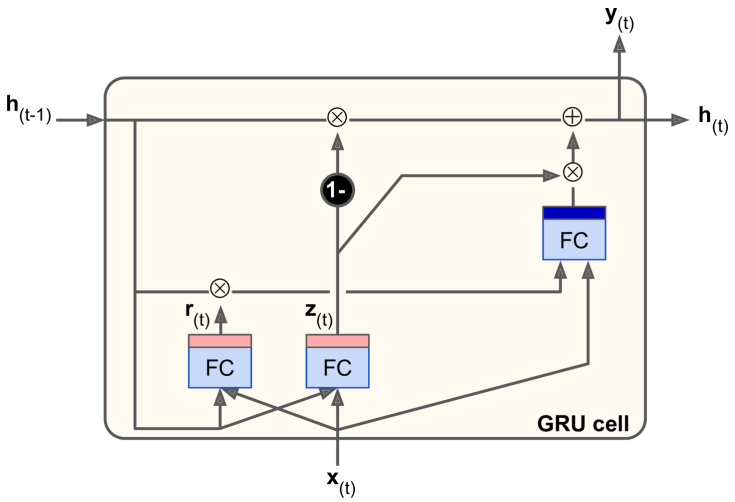

Figura 2.3: Representación gráfica de la interacción los agentes que participan en una unidad neuronal de una capa *GRU*. Fuente: *Hands On Machine Learning with Scikit-Learn & Tensorflow*[3].

2.5. *Random Forest*

El algoritmo de árboles consiste en en una estructura ramificada en la que se toman decisiones. Al final, cada extremo de cada rama (llamada hoja) vota por una decisión y la alternativa más votada corresponde a la predicción del modelo. Este modelo es bastante sencillo, pero permite generar modelos de buen desempeño.

El modelo anterior fue utilizado de base para el algoritmo de *Random Forest* o árboles aleatorios, que consiste en varios modelos de árboles con un largo aleatorio de ramas[12]. Al final, los árboles votan por alguna clase y la más elegida pasa a ser la respuesta del modelo.

2.6. *Gradient Boosting*

El algoritmo de *Gradient Boosting*, al igual que el de *random forest*, aprovecha el concepto de múltiples predicciones para seleccionar una respuesta final[9]. La diferencia radica en que los algoritmos de tipo *boosting* buscan favorecer la correcta predicción de cada unidad, robusteciendo la respuesta que entregan en conjunto.

Para conseguirlo, el algoritmo asigna una relevancia o peso mayor a aquellas unidades más exactas en sus predicciones. Estos pesos son optimizados en la medida que el modelo es entrenado, permitiendo desarrollar modelos con un alto desempeño y a un costo computacional bajo.

2.7. Métricas de evaluación

Al desarrollar un modelo, se necesita comprobar si sus resultados predichos son consistentes con los reales. Para ello, se utiliza las métricas de evaluación para cuantificar el desempeño del modelo. Dependiendo de la tarea que se realice, existen múltiples métricas que pueden ser utilizadas y siempre hay que seleccionar la que se ajuste mejor a nuestro proyecto.

A continuación, se presentan las métricas que serán utilizadas para las tareas de clasificación y regresión.

2.7.1. Métricas de clasificación

Al tratar con un problema de clasificación, es necesario definir la matriz de confusión, que corresponde a una representación gráfica del desempeño de un algoritmo. Para elaborarla, se ubican los casos en una tabla como la 2.1 y se suma la cantidad de cada celda. Para multiclase, simplemente se aumenta la cantidad de filas y columnas.

Tabla 2.1: Matriz de confusión con el valor de cada posición.

		Valor Predicho	
		Positivo	Negativo
Valor Real	Positivo	Verdadero positivo (TP)	Falso negativo (FN)
	Negativo	Falso positivo (FP)	Verdadero negativo (FN)

La matriz de confusión permite visualizar las posibles combinaciones de acierto o falla con la cantidad de etiquetas que presenta el modelo. Esto es base para definir las métricas de evaluación que se suelen utilizar en tareas de clasificación, además de que permiten visualizar los resultados.

Accuracy

El *accuracy* o exactitud es una métrica que se utiliza para la evaluación de modelos de clasificación y consiste en determinar qué porción de las predicciones fueron acertadas. Su ecuación es la 2.2.

$$acc = \frac{\sum TP + \sum TN}{TP + TN + FP + FN} \tag{2.2}$$

Precisión, exhaustividad y *F-score*

La precisión corresponde a la exactitud de predicciones positivas, mientras que la exhaustividad es la proporción de positivos correctamente detectados. A continuación, se presentan las ecuaciones para cada métrica:

$$Precisión = \frac{TP}{TP + FP} \tag{2.3}$$

$$Exhaustividad = \frac{TP}{TP + FN} \tag{2.4}$$

Tanto la precisión como la exhaustividad son métricas que se busca maximizar para asegurar un buen modelo, pero su comportamiento es inverso entre ellas, de modo que al aumentar una, la otra tiende a decrecer. Para poder optimizar ambas de manera simultánea, se utiliza la métrica *F-score*, que se determina como la media armónica entre la precisión y la exhaustividad y cuya ecuación es la siguiente:

$$F\ score = 2 \cdot \frac{Precisión \cdot Exhaustividad}{Precisión + Exhaustividad} \tag{2.5}$$

2.8. Métricas de regresión

Error Cuadrático Medio

El error cuadrático medio, o *mean squared error* corresponde al promedio del error cuadrático que presenta la predicción con respecto a los valores reales. Para calcularlo, se utiliza la ecuación 2.6, donde $\hat{y}$ es el valor real, y_i el valor propuesto y n la cantidad de muestras.

$$mse = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \tag{2.6}$$

Coeficiente de determinación

El coeficiente de determinación, también llamado R^2 *score*, es un estadístico que evalúa el modelo con un valor de 0 a 1, donde 1 es el mejor modelo posible. Para determinarlo, se

calcula el cuadrado del coeficiente de Pearson, denominado con el símbolo ρ. De este modo, sea σ^2_{XY} la covarianza entre X e Y, σ^2_X la desviación estándar de X y σ^2_Y la desviación estándar de Y, la ecuación 2.7 permite determinar el coeficiente R^2.

$$R^2 = \rho^2 = \left(\frac{\sigma_{XY}}{\sigma_X \sigma_Y}\right)^2 \tag{2.7}$$

Capítulo 3

Metodología

Para el desarrollo de este proyecto de aprendizaje de máquinas, se decide utilizar la metodología recomendada en el libro *Hands On Machine Learning with Scikit-Learn & Tensorflow* [3], que permite abarcar proyectos desde su planteamiento hasta su aplicación e integración. La figura 3.1 presenta el diagrama de las tareas realizadas con la relación entre ellas.

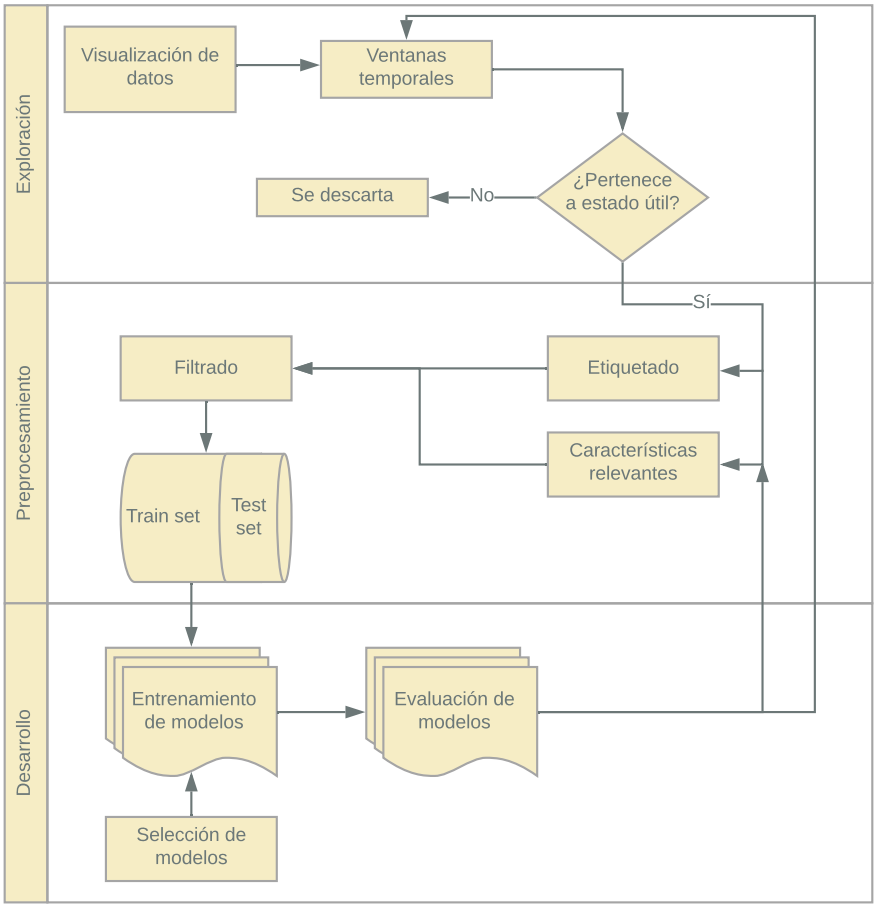

Figura 3.1: Secuencia de tareas realizadas, separadas en las etapas de ejecución, preprocesamiento y desarrollo.

3.1. Exploración y preparación de datos

En la primera etapa del desarrollo de la tesis, se prepara los datos correspondientes a la unidad de la aeronave, con la finalidad de implementarlos de manera adecuada en los modelos de estimación. Para ello, se realizará lo siguiente:

- Se visualiza los datos de una manera que permita comprender su naturaleza.
- Se categoriza cada medición según el estado de vuelo en que se encuentre la aeronave. Éstos son: despegue, en vuelo, aterrizaje y fuera de operación.
- Se define bajo un criterio pertinente qué datos no contribuyen a la tarea objetivo y se separa de la base de datos.
- Se selecciona y programa las características relevantes en base a ventanas de tiempo.
- Se define criterios y se procede a etiquetar la base de datos.
- Se optimiza los algoritmos de preprocesamiento para poder ajustarlos de manera eficiente.
- Se genera un archivo con las modificaciones ya realizadas.

3.2. Desarrollo y ajuste de modelos de predicción

En esta etapa, se selecciona el o los modelos que se desarrollan y se ajustan para su posterior entrenamiento. Como en esta etapa no es seguro que exista resultados que respondan al objetivo principal, al finalizarlas se evalúa si se vuelve a la etapa anterior o si se procede a la etapa siguiente. Los hitos clave son los siguientes:

- Se selecciona el o los modelos por aplicar, tanto para clasificación como para regresión.
- A partir de la base de datos con expansión de características relevantes, se filtra y descarta las características que entreguen información redundante o irrelevante.
- Se separa los datos ya estandarizados en un set de entrenamiento y pruebas.
- Se entrena modelos de clasificación y regresión que serán utilizados como *baseline*.
- Se entrena modelos de redes neuronales y optimiza los parámetros de entrada. Para esta etapa, se utiliza siempre la misma base de datos para los modelos *baseline* y los de redes neuronales.
- Se itera sobre los tamaños de ventanas temporales del preprocesamiento, características relevantes y parámetros de los modelos para obtener el máximo *performance*.
- Se programa y revisa los algoritmos que permitan iterar el preprocesamiento y entrenamiento de los modelos de manera eficiente.
- Se registra los resultados obtenidos.

3.3. Análisis de resultados

Con los modelos ya entrenados y ajustados, se evalúa si los resultados permiten cumplir con el objetivo general. Para ello, se realizará lo siguiente:

- Se comenta resultados y desempeño alcanzados.
- Se identifica, enlista y explica posibles fuentes de errores que puedan implicar una disminución de desempeño.
- Se valida el modelo con datos nuevos de comparación, que permitan comprobar si los resultados son acorde a lo que se espera.

Capítulo 4

Preprocesamiento de datos

En la presente tesis, se trabaja en dos sets de datos correspondientes a dos unidades distintas, de las cuales la última medición corresponde a un mes antes de la manifestación de una misma falla. El origen o características de la falla no son conocidos, por lo que se procede a determinar si a partir de los datos se puede obtener el tiempo que queda para la falla o si se logra identificar algún patrón relevante.

Sobre las variables, se poseen datos de altitud del vuelo, temperatura de entrada y salida de aceite del equipo y valores de vibración RMS de un equipo que opera solidario a la unidad que manifiesta la falla.

El enfoque de manejo de la base de datos es en ventanas temporales, extrayendo la mayor cantidad de características posibles para posteriormente seleccionar las que aportan más información relevante para el problema en cuestión.

4.1. Metodología de trabajo de la base de datos

De parte de la empresa, se recibió dos set de datos de dos unidades IDG distintas para responder al problema propuesto. Como sólo se cuenta con los datos de una falla para cada una de las dos unidades y la cantidad de muestras es baja, se decide unir ambas bases de datos en una, lo que permite ampliar el set de entrenamiento y otorga más capacidad de generalizar a los modelos resultantes.

Como se indica anteriormente, se opta por utilizar ventanas temporales para el preprocesamiento de la base de datos, dado que se requiere de vectores de información para determinar características relevantes. Al desarrollar los modelos, se comprueba que aquellos vuelos que presentan únicamente una o dos ventanas temporales disminuyen el desempeño en general, por lo que se descartan al generar el set de datos que se utilizará en los modelos.

Posteriormente, sobre cada ventana se determina una serie de propiedades y características matemáticas, que son las que se utilizan en la entrada de los modelos. Más adelante se profundiza sobre estas operaciones.

Además, se busca identificar el estado de vuelo de la nave, dado que existe diferencias importantes en el comportamiento de las variables al comparar estos estados, como se verifica más adelante. De este modo, se simplifica la entrada de datos al modelo, favoreciendo el proceso de predicción.

4.2. Características generales de la base de datos

4.2.1. Bases de datos y variables medidas

Como se indica en el principio de este capítulo, los datos medidos corresponden a dos unidades que presentaron la falla que se busca detectar. Como la condición de parte de la empresa sobre los resultados es detectar la falla un mes antes de que ésta se manifieste, la última medición corresponde a aquella que marca un mes previo a la manifestación de la falla.

En la tabla 4.1 se indica las características de cada base de datos, de las cuales cada una presenta más de 10 millones de puntos a una tasa de una muestra por segundo. Cada una presenta cinco variables que se resumen en la tabla 4.2: altitud del vuelo, temperaturas de entrada y salida de aceite y dos mediciones de vibración mecánica de un equipo solidario.

Las unidades de medida de temperatura y vibración no se indican, por lo que no se realiza análisis de las unidades físicas o su comportamiento específico.

Tabla 4.1: Información sobre cada una de las bases de datos recibidas.

Set de datos	1	2
Tasa de muestreo	1 [muestra/s]	1 [muestra/s]
Cantidad de mediciones	13.388.564	19.083.843
Fecha primera medición	26-12-2018	26-12-2018
Fecha última medición	30-05-2019	04-08-2019
Cantidad de vuelos	746	1107
Duración promedio por vuelo en estado crucero	0.71 [h]	0.70 [h]

Tabla 4.2: Variables medidas en el sistema y breve descripción. No se conoce las unidades de medida de todas las variables.

Variable de medición	Unidad	Descripción
Altitud	Pies	Altura del avión al momento de tomar la medición.
Temperatura entrada	-	Temperatura de entrada de aceite al equipo.
Temperatura salida	-	Temperatura de salida de aceite del equipo.
Vibración N1	-	Valor N1 de vibración de equipo solidario.
Vibración N2	-	Valor N2 de vibración de equipo solidario.

4.2.2. Estados de operación de la aeronave

Para la aeronave, podemos identificar tres estados de operación distintos:

1. Fuera de operación: La aeronave se encuentra en tierra.
2. Despegue/aterrizaje: La aeronave se encuentra ganando o perdiendo altitud, al despegar o aterrizar.
3. Vuelo crucero: La aeronave se encuentra en vuelo estable sin variar considerablemente su altitud.

En la figura 4.1 se presenta la altitud de la aeronave correspondiente al set de datos 2 para el mes de enero del 2019. En primer lugar, se aprecia una altitud muy cercano a los 0 pies, que corresponde al estado de fuera de operación. Los peaks que superan los 25.000 pies son los vuelos crucero de la aeronave. Además, se aprecia estados de reposo de la nave entre días, e incluso períodos superiores a un día fuera de operación.

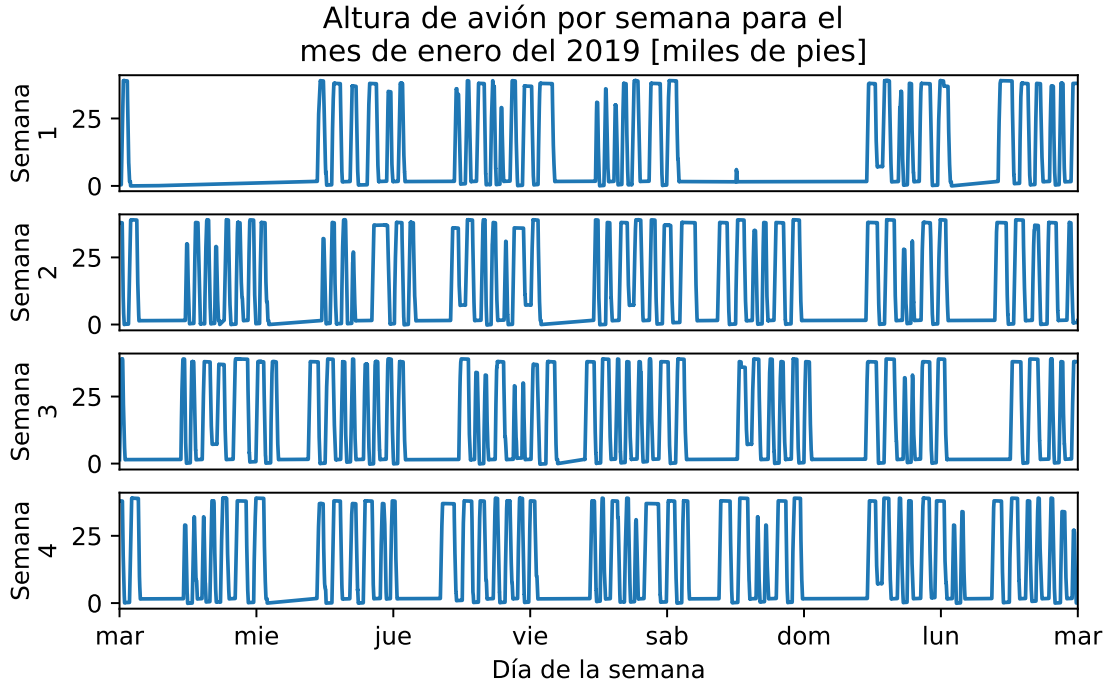

Figura 4.1: Medición de altura del avión por semana para las primeras cuatro semanas del año 2019.

Dado lo anterior, se define dos criterios para determinar el estado de vuelo, los cuales consisten en dos validaciones basadas en la altura de la nave que se verifican para cada ventana temporal a trabajar y cuyos parámetros utilizados dependen del largo de la ventana temporal empleada. Los criterios se describen a continuación:

1. Si el valor *peak to peak* de la altitud de la nave supera un valor crítico, entonces se dirá que la ventana corresponde a un estado de despegue o aterrizaje de la nave. De no superarlo, entonces se dirá que la nave se encuentra en estado estacionario, que puede corresponder a vuelo crucero o fuera de operación.

2. Si existe estado estacionario, se verificará si el promedio de altitud de la ventana temporal supera cierto valor crítico. De ser así, se dirá que la nave se encuentra en vuelo crucero o constante y en caso contrario, su estado será fuera de operación.

4.2.3. Análisis de las bases de datos

Si ahora se observa el gráfico de altitud en un día y se grafica las variables de temperatura y vibración, se tiene lo presentado en la figura 4.2. En el segundo gráfico de temperatura, se puede notar que éste presenta un comportamiento lineal anómalo cuando la altitud es cercana a cero. Esto nos permite inferir que las mediciones realizadas con la aeronave fuera de operación no son fiables o existentes, por lo que en dichos rangos de datos serán descartados.

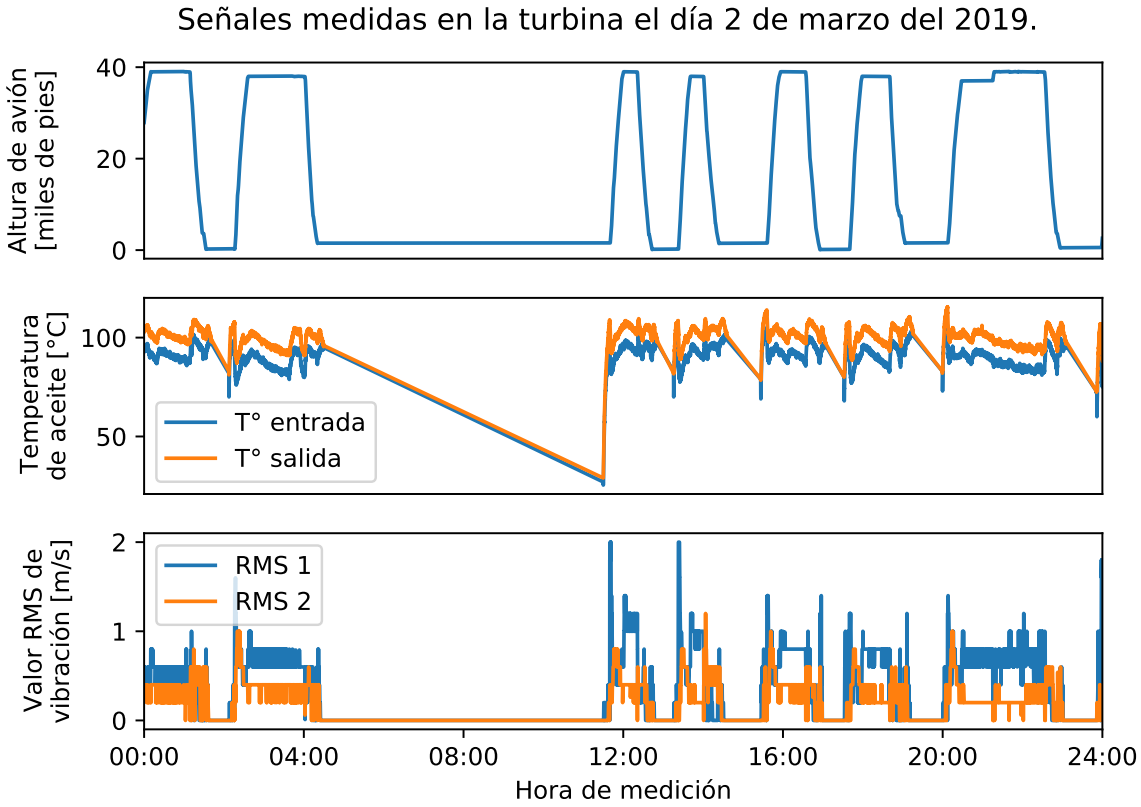

Figura 4.2: Señales de altura, temperatura y vibración para el 2 de marzo del 2019.

Al expandir el gráfico al comportamiento de las variables en un vuelo, se obtiene lo mostrado en la figura 4.3. Las señales de temperatura de entrada y salida en el segundo gráfico muestran un comportamiento similar con tres irregularidades notorias:

- Un *peak* importante durante el despegue que es natural en este estado, dado que la unidad alcanza su temperatura máxima en ese instante.
- Un comportamiento cíclico durante el vuelo crucero, con pequeñas alzas en las temperaturas descritas.

- Una inestabilidad notoria próximo al aterrizaje de la nave.

Por otro lado, las variables de vibración del tercer gráfico presentan las siguientes características:

- Un período de inestabilidad durante el despegue.
- Una fase constante durante el vuelo crucero.
- Una caída durante el aterrizaje relacionada al apagado de las unidades.

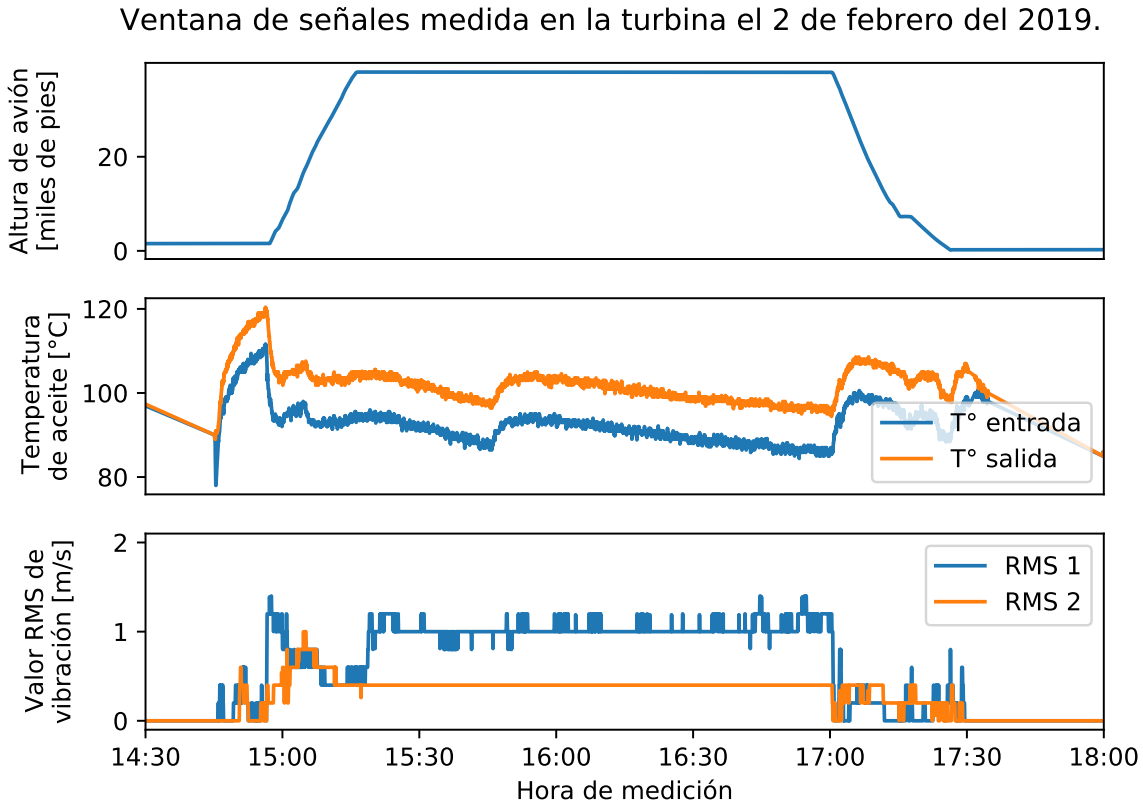

Figura 4.3: Señales de altura, temperatura y vibración para una ventana de la señal el 2 de febrero del 2019. Fuente: elaboración propia

4.3. Expansión de base de datos

Con el fin de extraer la mayor cantidad posible de información de la base de datos, se aplica distintas técnicas para ampliar la cantidad de variables que se entregan a los modelos y que requieren de una estructura de vectores de datos. En específico, se trabaja con operaciones matemáticas entre variables, con características relevantes, derivada e integral de la señal y con la transformada de Fourier por bandas.

4.3.1. Operaciones matemáticas entre variables

En primer lugar, se realiza operaciones matemáticas entre las variables para expandir la cantidad de variables utilizadas para realizar la predicción. Dado que no se conoce la naturaleza física de estas variables, se busca relacionarlas de la manera más directa posible, por lo que se utiliza las diferencias presentadas en las ecuaciones 4.1 y 4.2.

$$f(T_{out}, T_{\mathrm{in}}) = T_{out} - T_{in} \tag{4.1}$$

$$f(N_1, N_2) = N_1 - N_2 \tag{4.2}$$

4.3.2. Derivada e integral de la señal

El comportamiento diferencial e integral se ha utilizado en los sistemas de automatización mediante los controladores *PID*, que permiten utilizar la tendencia y información previa en señales temporales para la toma de decisiones. Es por ello que sobre las señales trabajadas, se decide utilizar la derivada e integral de las señales para expandir la base de datos. Las ecuaciones utilizadas para calcular el diferencial y el integrativo de la señal son las 4.3 y 4.4 respectivamente.

$$x_{\mathrm{i}+1} = x_{\mathrm{i}} - x_{\mathrm{i}-1} \tag{4.3}$$

$$x_{\mathrm{i}+1} = x_{\mathrm{i}} + x_{\mathrm{i}-1} \tag{4.4}$$

4.3.3. Características relevantes

Se selecciona una serie de propiedades estadísticas que se aplican a cada ventana de datos para obtener un dato más específico y concreto de éstas. Las propiedades son las mostradas en la tabla 4.3, donde además se presenta la ecuación utilizada para calcularlas.

Tabla 4.3: Características determinadas para cada variable a partir de las ventanas de tiempo.

Característica	Fórmula
Raíz Cuadrática Media	$F_{rms} = \sqrt{\sum_1^N x(n)/N}$
Valor máximo	$F_p = max(\|x(n)\|)$
Amplitud	$F_A = max(\|x(n)\|) - min(\|x(n)\|)$
Factor cresta	$F_{cf} = F_p/F_{rms}$
Promedio	$F_m = \sum_1^N x(n)/N$
Varianza	$F_{std} = \sqrt{\sum_1^N (x(n) - F_m)))}$
Asimetría	$F_S = \frac{\sum_1^N x(n)^3/N}{F_{rms}^3}$
Kurtosis	$F_K = \frac{\sum_1^N x(n)^4/N}{F_{rms}^4}$
5° momento de inercia	$F_{5M} = \frac{\sum_1^N x(n)^5/N}{F_{rms}^5}$

4.3.4. Transformada de Fourier por bandas

Otra herramienta de expansión de variables utilizada corresponde a la transformada de Fourier aplicada sobre cada variable. Para ello, se utiliza la librería de Python *numpy*. Para su cálculo, se sigue los siguientes pasos:

1. Se determina la transformada de Fourier para cada señal, su integral y su derivada, tal como se señaló en la sección anterior.
2. Se normaliza la transformada de Fourier.
3. Se separa las primeras 10 frecuencias de cada transformada y agregarlas como características relevantes de cada variable.

4.4. Selección de características relevantes

Una vez calculadas las características relevantes de cada variable, se obtuvo una matriz de datos con 494 columnas. Sin embargo, muchas de éstas correspondían a información redundante o irrelevante para la falla, lo que desfavorece la implementación de los modelos de predicción. Es por ello que se decide descartar columnas acorde a dos criterios de selección:

1. **Criterio de correlación**: Se determina cuáles características presentan una mayor correlación con la variable objetivo en seleccionada y se descarta aquellas cuyo valor es superior a 0.1. Con esto, se descarta características que entregan información irrelevante para la predicción de la falla.
2. **Criterio de error medio**: Se determina el error medio entre cada par de vectores, descartando uno de éstos cuando el valor obtenido es inferior a 0.01. Esto permite descartar características altamente similares entre ellas.

4.5. Separación de set de entrenamiento y de pruebas

Para poder realizar el entrenamiento del modelo y comprobar su desempeño, es necesario separar la base de datos en un set de entrenamiento y otro de pruebas. Se optó por no utilizar un set de validación para supervisar el entrenamiento de los modelos, dado que se dio prioridad a realizar mayor cantidad de experimentos y el volumen de la base de datos es bajo para el entrenamiento de modelos de redes neuronales.

Dado lo anterior, se optó por apartar 20 % de la base de datos para el proceso de pruebas y se dejó 80 % para el entrenamiento de los modelos, separación que fue realizada de manera aleatoria. Durante el proceso de entrenamiento, siempre se utilizó el mismo set de entrenamiento para que la comparación de los resultados entre modelos sea bajo condiciones de entrada similares.

Capítulo 5

Experimentos realizados y manejo de etiquetas

En primer lugar, se define lo que se espera obtener a partir de los datos considerando los siguientes elementos:

1. El contexto que representa el desarrollo de la tesis, que corresponde al departamento de mantenimiento de una aerolínea.
2. La aplicación final de la predicción realizada.
3. La validación de la existencia de una tendencia en los datos presentados.

El tercer punto es fundamental, ya que como se indica en la sección 1.3 de alcances, no existe conocimiento de la naturaleza de la falla ni de cómo ésta se desarrolla. Es por ello que es necesario comprobar que su desarrollo es representable a partir de las bases de datos recibidas. La posibilidad de detectar la falla a partir de los datos se asume como acertada, ya que fue comprobada por trabajos previos con un modelo detector de anomalías.

Con el fin de comprobar que en los datos se puede realizar una tarea de predicción de la falla modelada, se decide realizar una cantidad variada de experimentos, para lo cual se consideran las tareas de clasificación y regresión.

Dentro del universo de experimentos realizados, los modelos de *Random Forest*, *Gradient Boosting*, *Convolucional de 1 dimensión*, *Gated Recurrent Unit* y *Long Short Term Memory* entregaron las predicciones más consistentes y con mejor desempeño, por lo cual se seleccionan para el desarrollo de la tesis. Estos modelos son presentados más adelante.

5.1. Tipos de predicción modelada

Para verificar la existencia de tendencia en los datos, se procede a desarrollar dos tipos de predicciones distintas: Clasificación y Regresión. Estos tipos de tareas, si bien pueden utilizar los mismos modelos, entregan resultados distintos.

En primer lugar, se realiza una tarea de clasificación multiclase, las que no son idóneas para analizar problemas donde la variable objetiva es continua. Sin embargo, sus resultados pueden ser utilizados de base para verificar si existen patrones al comparar distintas ventanas de tiempo, de modo que una clase bien definida puede ser indicio de un patrón distinto al resto en dicha ventana temporal. De estos modelos, se espera que éste asigne para cada muestra una ventana temporal que corresponde a un rango de tiempo para la manifestación de la falla.

Por otra parte, los modelos de regresión desarrollados entregan un tiempo estimado remanente para la manifestación de la falla. Las regresiones pueden retornar un valor real en lugar de discreto, por lo que se moldean mejor a la tarea que se enfrenta. Sin embargo, este tipo de modelos tiende a ser más complejo, por lo que requiere de mayor volumen de datos para poder ser entrenado adecuadamente. De estos modelos, se espera obtener un vector de predicciones, que al ordenarlo temporalmente debe aproximarse a la curva de degradación del equipo para la falla modelada.

5.2. Modelos de aprendizaje de máquinas utilizados

La variedad de modelos implementados entrega robustez y coherencia a los resultados obtenidos, siempre y cuando éstos sean similares. Es por ello que se decide utilizar más de un tipo de modelo para explorar el problema dado.

Para la presente tesis, se experimenta con cinco modelos distintos: *RF*, *XGB*, *Conv1D*, *GRU* y *LSTM*. La tabla 5.1 se resume las nomenclaturas que se emplean en adelante para referirse a cada uno de estos modelos.

Tabla 5.1: Nomenclatura utilizada para referirse a los dos tipos de vectores utilizados para definir las etiquetas.

Referencia	Descrición
RF	Modelo de *Random Forest*.
XGB	Modelo de *Gradient Boosting*.
Conv1D	Modelo *Neural Network* con capas convolucionales.
GRU	Modelo *Gated Recurrent Unit*, tipo de *Recurrent Neural Network*.
LSTM	Modelo *Long Short Term Memory*, tipo de *Recurrent Neural Network*.

Los modelos *RF* y *XGB* utilizan algoritmos que presentan resultados muy estables y acertados, además de ser más fáciles de implementar. Es por esto por lo que se eligieron para utilizarlos como modelos base. Por otro lado, los modelos de redes neuronales utilizados permiten relacionar los parámetros de entrada espacialmente (*Conv1D*) y temporalmente (*GRU* y *LSTM*). Esto se adapta bien al problema.

Los modelos *RF* y *XGB* se implementan mediante las librerías de *python*, *scykit − learn* y *XGBoost*. Como parámetros, utilizan la cantidad de unidades que creará el modelo (*n_estimators*) y la profundidad máxima (*max_depth*). Por otro lado, los modelos neuronales se estructuraron de manera similar, diferenciándose en la capa especializada que utiliza cada red (*Conv1D*, *GRU* o *LSTM*). La primera parte de cada red neuronal consiste en un conjunto de tres capas que pueden repetirse hasta tres veces. Esta secuencia se describe a continuación:

1. Una capa opcional de *batch normalization*.
2. Una capa especializada dependiendo del modelo a la que se le define la cantidad de neuronas que presenta.
3. Una capa opcional de *dropout* con probabilidad fija en 0.2.

Posterior a las capas anteriores, se sigue una estructura neuronal más clásica, con la siguiente arquitectura:

1. Una capa *flatten*, que convierte la entrada multidimensional en un vector unidimensional.
2. Una serie de capas *fully connected* con función de activación *relu*. Para cada capa, se define la cantidad de neuronas que presenta.
3. La capa de salida depende de si la tarea es de clasificación (diez neuronas con función de activación *softmax*) o de regresión (una neurona de salida sin función de activación).

5.3. Etiquetado de la base de datos

En primer lugar, hay que notar que no se posee los estados de falla de la aeronave, ya que no se cuenta con las etiquetas de estado para cada medición. Para este caso, se opta por determinar una etiqueta para cada medición, la cual debe ser continua para la regresión y categórica para la clasificación. Es por ello que se debe definir los criterios que entregarán la etiqueta para cada medición basándose en la siguiente información conocida:

1. La última medición corresponde a un mes antes de que la falla que se busca modelar se exprese.
2. La tasa de muestreo es de una muestra por segundo, por lo cual se puede etiquetar por cantidad de tiempo remanente para la manifestación de la falla.
3. Se puede separar las muestras por cantidad de vuelos realizados por la aeronave y etiquetar según la cantidad de vuelos restantes para la manifestación de la falla.
4. El tipo de degradación que experimenta la falla sobre la unidad es desconocida, pero para efectos de exploración, se define como una degradación lineal.

Bajo condiciones regulares de trabajo, la elección de contar por tiempo remanente o cantidad de vuelos remanentes se determinaría a partir de la naturaleza de la falla, según ésta se degrade por ciclo de despegue/aterrizaje o por operación regular de la nave. Sin embargo, en este caso es imposible determinar cualitativamente la naturaleza de la falla (dado que ésta es desconocida), por lo que se decide se entrenar los modelos utilizando ambos criterios, fijando el valor nulo en la última medición. Para definir el tiempo remanente, se concatena las

ventanas de datos temporalmente y se les asigna valores naturales descendientes de tal modo que la última medición sea el valor nulo. Para el caso de los vuelos remanentes, se procede de la misma manera, asegurando que las ventanas de cada vuelo compartan la misma etiqueta.

Lo anterior se resume en la tabla 5.2, donde se indica la nomenclatura que se utiliza para referenciar cada etiqueta.

Tabla 5.2: Nomenclatura utilizada para referirse a los dos tipos de vectores utilizados para definir las etiquetas.

Referencia	Descrición
E_T	Variable que cuenta la cantidad de tiempo remanente para que se presente la falla.
E_V	Variable que cuenta la cantidad de vuelos remanentes para que se presente la falla.

Para la clasificación, la variable objetivo se separa en diez intervalos, cada uno representando una fracción del valor máximo de la etiqueta. Es por ello que para cada una de las etiquetas E_T y E_V resulta un vector de etiquetas para clasificación distinto. En la tabla 5.3 se indica la cantidad de muestras resultantes para cada intervalo.

Tabla 5.3: Etiquetas para clasificación asignadas y cantidad de muestras en cada intervalo.

Intervalo	Cantidad de muestras	
	Etiqueta E_T	$EtiquetaE_V$
9	798	940
8	789	820
7	791	857
6	798	789
5	792	715
4	794	782
3	783	750
2	496	418
1	581	600
0	332	283

5.4. Post procesamiento: Análisis por vuelo

Al comienzo de este capítulo, se define los criterios que guían el desarrollo de los modelos y uso de resultados. En esta línea, las predicciones realizadas con los modelos anteriores omiten un punto fundamental, en el cual las ventanas temporales corresponden a distintos vuelos de las aeronaves.

En esta tesis, se decide realizar un post procesamiento de los datos para determinar una predicción global por vuelo, en lugar de una individual por ventana temporal. Esto permite simplificar el resultado, ya que permite resumir el resultados de varias ventanas en uno. A su vez, mitiga el efecto de predicciones fuera de la distribución. Esto además facilita el manejo de la base de datos y programación de los modelos, dado que todos los vuelos presentan cantidades dispares de ventanas, alcanzando algunas más de 20.

Para la clasificación, Se opta por utilizar la moda de las predicciones. En el caso de la regresión, se utiliza el promedio de las predicciones, que entrega resultados muy superiores al resto de las alternativas.

5.5. Validación de modelos con datos sanos

Para validar que los modelos no están aprendiendo de la degradación natural del equipo y, en su lugar, modelan la falla que se busca detectar, se decide probar los modelos con datos correspondientes a una unidad sin fallas. Esto permite verificar cuánto los modelos confunden la degradación natural del equipo con la falla que se busca representar.

De esto, lo esperado es que entregue una predicción en torno al estado más alejado de la falla, sin embargo, existen algunos problemas con esta validación:

1. No se sabe exactamente cuándo se comenzó a desarrollar la falla, por lo que los datos utilizados para entrenar podrían poseer una sección de información sin falla, lo que implicaría que éstos podrían haberse considerado como estado fallido.
2. Sólo se utilizó la falla de dos aeronaves para el entrenamiento, por lo que los modelos podrían no generalizar lo suficiente como para modelar nuevos datos.

5.6. Resumen de experimentos

Para resumir los experimentos a realizar, en la tabla 5.4 se presenta la lista con las variaciones entre ellos en cuanto a la tarea realizada, el modelo entrenado y la etiqueta utilizada.

Para la presentación de resultados y el análisis posterior, se analiza de manera separada los experimentos de clasificación y regresión, dado que las métricas utilizadas no son comparables entre estos dos grupos.

Tabla 5.4: Lista de experimentos realizados especificando la tarea, el modelo y la etiqueta utilizados.

nº	Tarea	Modelo	Etiqueta
1	Clasificación	*RF*	E_T
2			E_V
3		*XGB*	E_T
4			E_V
5		*Conv1D*	E_T
6			E_V
7		*GRU*	E_T
8			E_V
9		*LSTM*	E_T
10			E_V
11	Regresión	*RF*	E_T
12			E_V
13		*XGB*	E_T
14			E_V
15		*Conv1D*	E_T
16			E_V
17		*GRU*	E_T
18			E_V
19		*LSTM*	E_T
20			E_V

Capítulo 6

Presentación resultados obtenidos

En el presente capítulo, se presenta los resultados obtenidos a parir de los modelos utilizados para la predicción de la falla modelada, los que se dividen en dos secciones: clasificación en 10 etiquetas y regresión.

Para cada sección, se comienza por presentar los parámetros de los modelos finales de *RF* y *XGB*, junto con el modelo de red neuronal de mejor desempeño. Posteriormente se encuentran los resultados, los valores de métricas alcanzados y los gráficos correspondientes a algunos de los modelos. Posteriormente, se presenta la comparación de los resultados al entregar la respuesta por vuelo.

La arquitectura de los modelos neuronales y los resultados que no se encuentran en esta sección, se presentan en los anexos, sección A.

6.1. Modelos de Clasificación

En la tabla 6.1 se encuentran los parámetros utilizados para los modelos *XGB* y *RF*.

De los modelos de redes neuronales, el que obtuvo el mejor desempeño es el *LSTM* para la etiqueta E_T, cuya arquitectura se muestra en la tabla 6.2

Tabla 6.1: Parámetros utilizados para los modelos de clasificación utilizando las variables objetivo E_T y E_V.

Etiqueta	E_T		E_V	
Modelo	RF	XGB	RF	XGB
n_estimators	376	283	62	194
max_depth	380	381	368	220

Tabla 6.2: Estructura de la red neuronal *LSTM* utilizada para la etiqueta E_T

Capa	Parámetro	Valor
Batch Normalization	-	-
Capa LSTM	Neuronas	185
Batch Normalization	-	-
Capa LSTM	Neuronas	115
Batch Normalization	-	-
Capa Flatten	-	-
Fully connected	Neuronas	114
	Función de activación	Softmax
Fully connected	Neuronas	50
	Función de activación	Softmax
Fully connected	Neuronas	10
	Función de activación	Softmax

Las performance alcanzadas para las métricas *accuracy* y *F-score* se presentan en la tabla 6.3. Posteriormente, en las figuras 6.1 a 6.4 se encuentra las matrices de confusión obtenidas para los modelos *XGB* y *LSTM*.

Tabla 6.3: Resultados de *accuracy* y *F-score* obtenidos para los modelos de clasificación utilizando las variables objetivo E_T y E_V.

Modelo	E_T		E_V	
	Accuracy [%]	F-score [%]	Accuracy [%]	F-score [%]
Random Forest	57.73	57.81	56.58	56.34
Gradient Boosting	59.88	60.02	58.02	57.45
Capas Convolucionales	52.98	51.41	51.55	50.48
Capas GRU	50.68	50.69	50.40	49.43
Capas LSTM	53.70	53.38	52.77	52.65

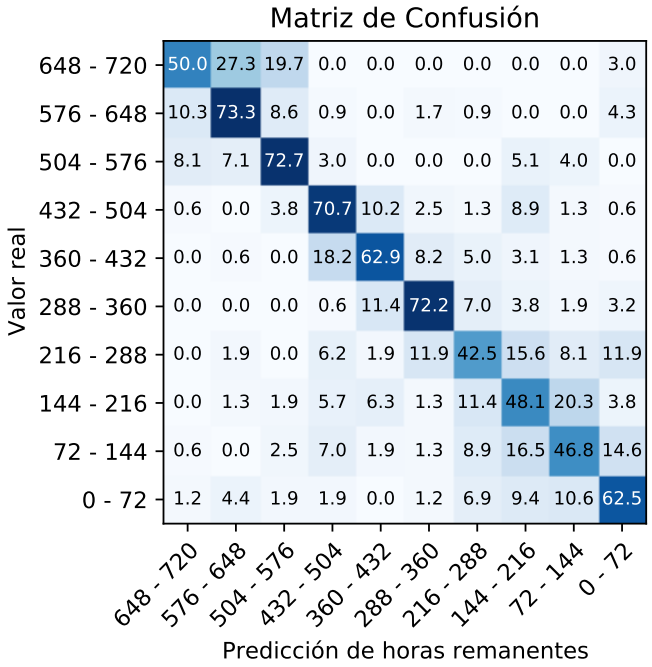

Figura 6.1: Matriz de confusión obtenida para la clasificación por *XGB* utilizando la variable objetivo E_T.

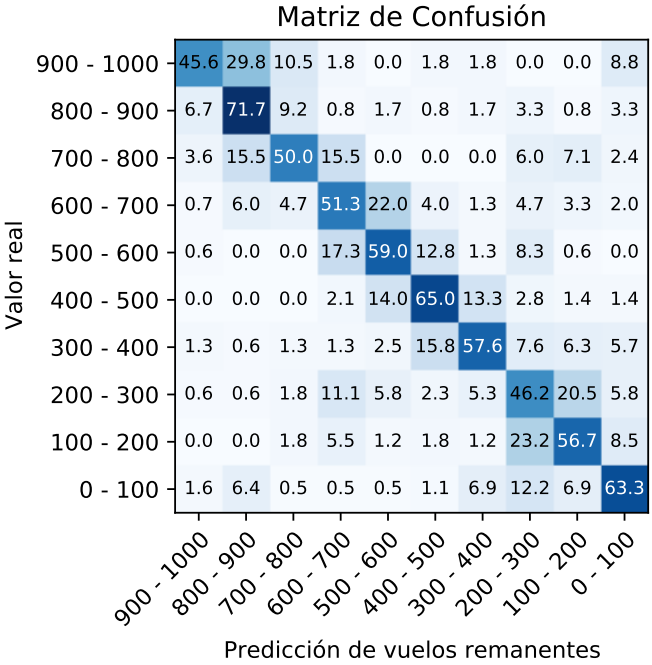

Figura 6.2: Matriz de confusión obtenida para la clasificación por *XGB* utilizando la variable objetivo E_V.

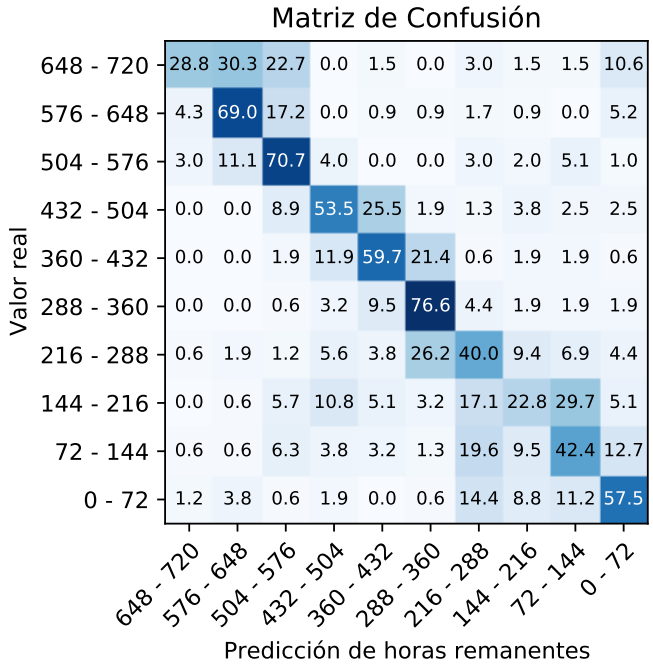

Figura 6.3: Matriz de confusión obtenida para la clasificación por un modelo de red neuronal *LSTM* utilizando la variable objetivo E_T.

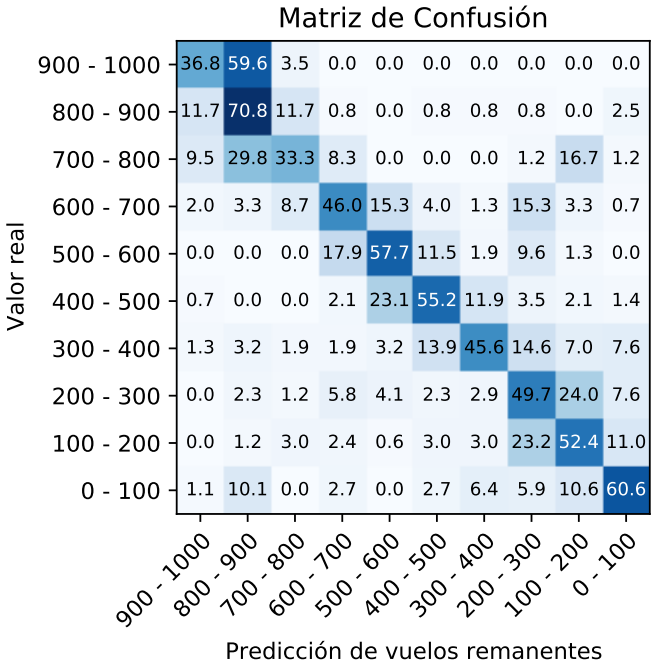

Figura 6.4: Matriz de confusión obtenida para la clasificación por un modelo de red neuronal *LSTM* utilizando la variable objetivo E_V.

En la tabla 6.4 se resume la comparación entre el antes y después de aplicar el post procesamiento, entregando una respuesta por vuelo. Las figuras 6.5 y 6.6 presentan las matrices de confusión para los modelos de *XGB* y *Conv1D*.

Tabla 6.4: Comparación entre resultados entregados por ventana de tiempo (columnas venta) y determinando un resultado por vuelo (columna vuelo) para los modelos de clasificación utilizando la variable objetivo E_T.

Modelo	Accuracy [%]			F-score [%]		
	Ventana	Vuelo	Diferencia	Ventana	Vuelo	Diferencia
Random Forest	57.73	61.53	3.80	57.81	60.83	3.02
Gradient Boosting	59.88	63.87	3.99	60.02	63.03	3.01
Capas Convolucionales	52.98	56.96	3.98	51.41	54.52	3.11
Capas GRU	50.68	51.83	1.15	50.69	51.27	0.58
Capas LSTM	53.70	58.11	4.41	53.38	57.86	4.48

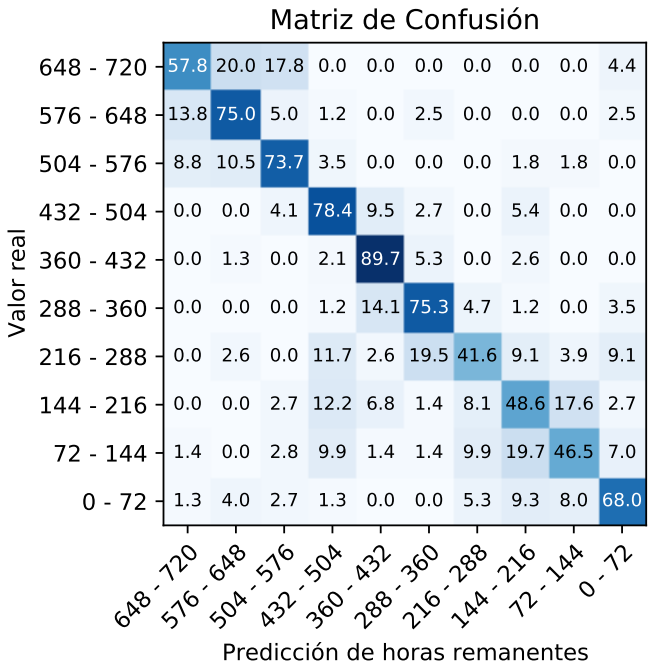

Figura 6.5: Matriz de confusión obtenida para la clasificación por *XGB* utilizando la variable objetivo E_T entregando una respuesta por vuelo.

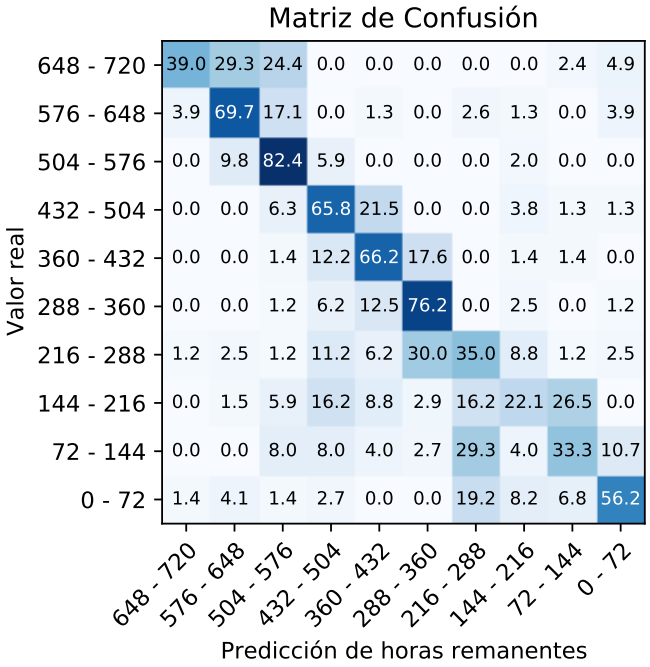

Figura 6.6: Matriz de confusión obtenida para la clasificación por un modelo de red neuronal *LSTM* utilizando la variable objetivo E_T entregando una respuesta por vuelo.

A continuación, en la tabla 6.5 se presenta el promedio y varianza para la predicción realizada por los modelos utilizando datos sanos, distintos a los de entrenamiento. Las figuras 6.7 y 6.8 presentan dos tipos de regresiones obtenidas.

Tabla 6.5: Promedio y varianza de las predicciones realizadas por los modelos de clasificación utilizando datos sanos.

Modelo	E_T		E_V	
	Promedio	Varianza	Promedio	Varianza
Random Forest	4.69	1.07	2.29	1.07
Gradient Boosting	4.23	1.97	4.61	2.15
Capas Convolucionales	2.08	0.68	2.12	1.62
Capas GRU	4.16	1.88	6.76	2.19
Capas LSTM	6.23	2.27	2.86	1.31

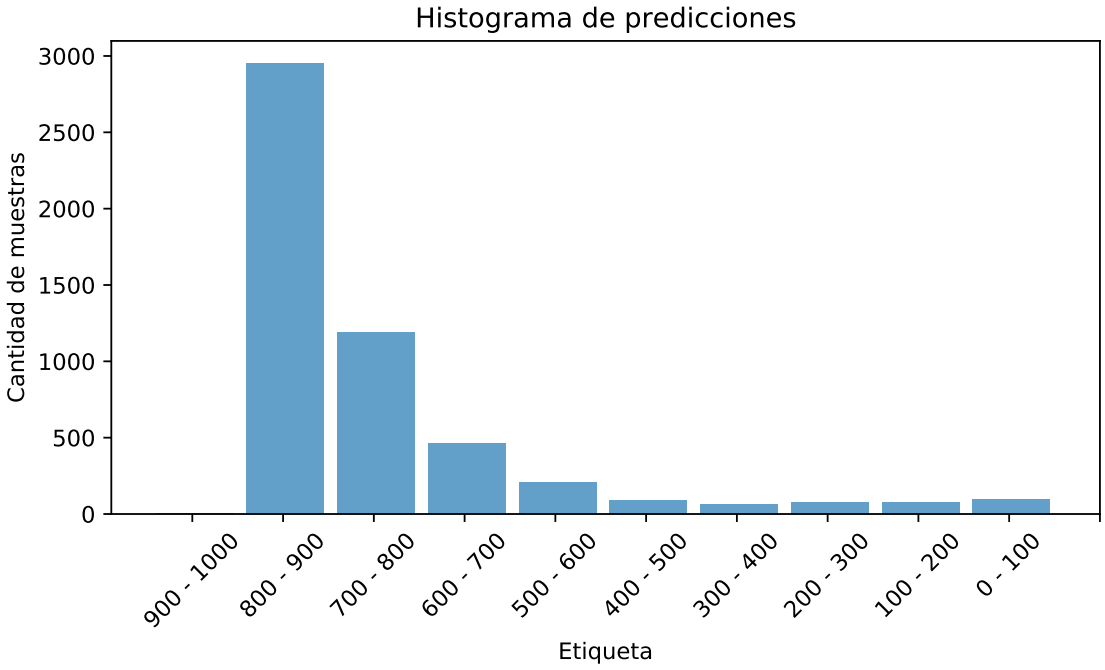

Figura 6.7: Predicciones para datos sanos del modelo *Conv1D* entrenado con la etiqueta E_T.

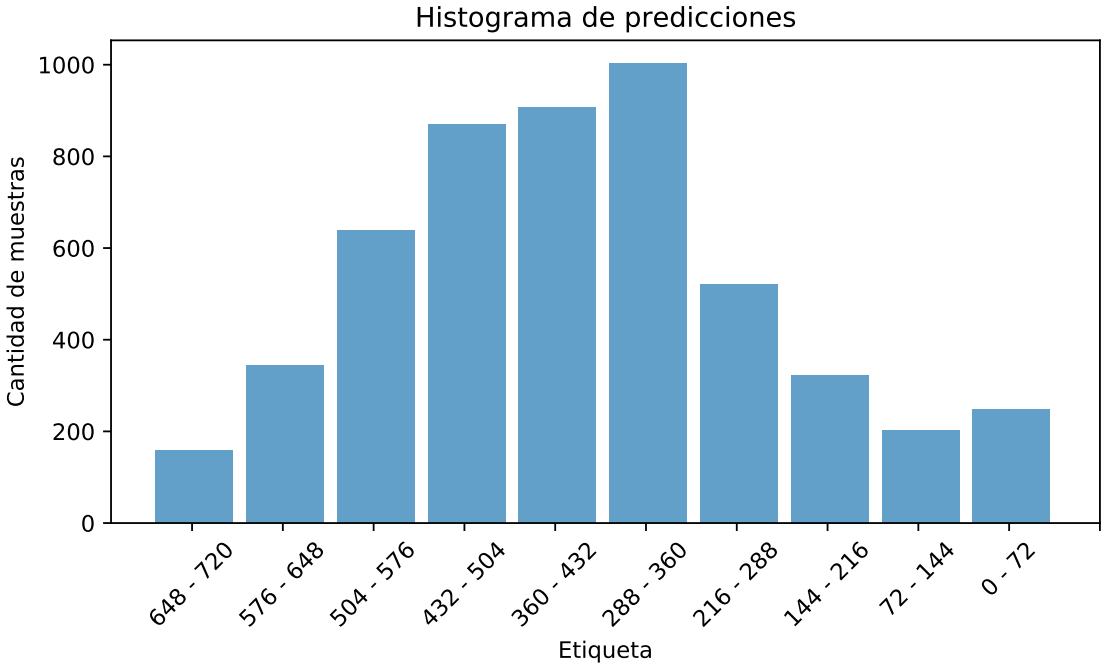

Figura 6.8: Predicciones para datos sanos del modelo *XGB* entrenado con la etiqueta E_T.

6.2. Modelos de regresión

En la tabla 6.6 se encuentran los parámetros utilizados para los modelos *XGB* y *RF*.

De los modelos de redes neuronales, el que obtuvo el mejor desempeño es el *GRU* para la etiqueta E_T, cuya arquitectura se muestra en la tabla 6.7

Tabla 6.6: Parámetros utilizados para los modelos de regresión utilizando las variables objetivo E_T y E_V.

Etiqueta	E_T		E_V	
Modelo	RF	XGB	RF	XGB
Número de estimadores	185	372	95	354
Profundidad máxima	402	89	418	460

Tabla 6.7: Estructura de la red neuronal *GRU* utilizada para la etiqueta E_T

Capa	Parámetro	Valor
Batch Normalization	-	-
Capa GRU	Neuronas	230
Batch Normalization	-	-
Capa GRU	Neuronas	155
Batch Normalization	-	-
Capa GRU	Neuronas	83
Capa Flatten	-	-
Fully connected	Neuronas	67
	Función de activación	Softmax
Fully connected	Neuronas	11
	Función de activación	Softmax
Fully connected	Neuronas	1

El desempeño alcanzado para las métricas *mse* y R^2 se presentan en la tabla 6.8. Posteriormente, en las figuras 6.9 a 6.12 se encuentra las matrices de confusión obtenidas para los modelos *RF* y *GRU*.

Tabla 6.8: Valores de las métricas *mse* y R^2 obtenidos para los modelos de regresión utilizando las variables objetivo E_T y E_V.

Modelo	E_T		E_V	
	mse	R^2	mse	R^2
Random Forest	0.30	0.70	0.32	0.68
Gradient Boosting	0.33	0.68	0.37	0.64
Capas Convolucionales	0.38	0.64	0.37	0.64
Capas GRU	0.32	0.68	0.37	0.63
Capas LSTM	0.34	0.66	0.39	0.62

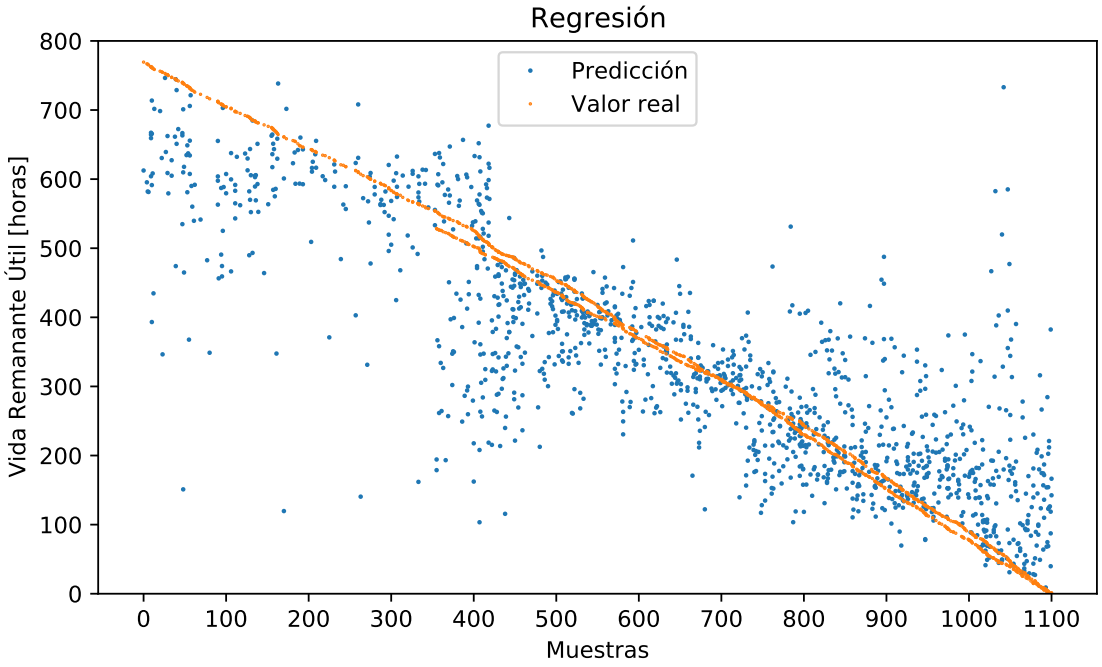

Figura 6.9: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo RF utilizando la variable objetivo E_T.

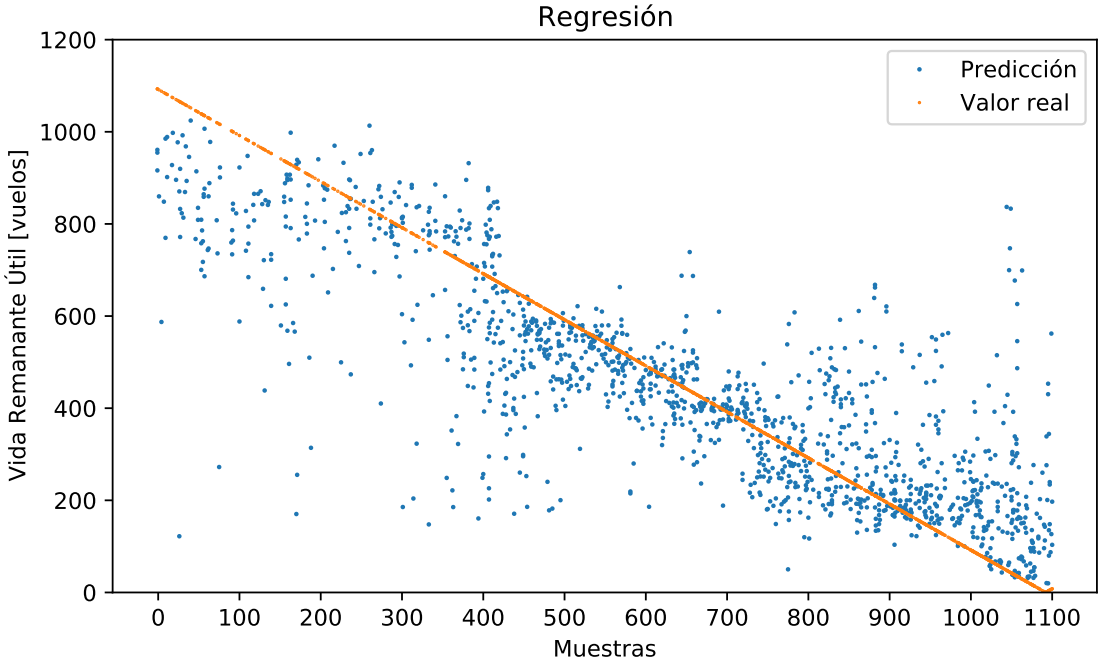

Figura 6.10: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo RF utilizando la variable objetivo E_V.

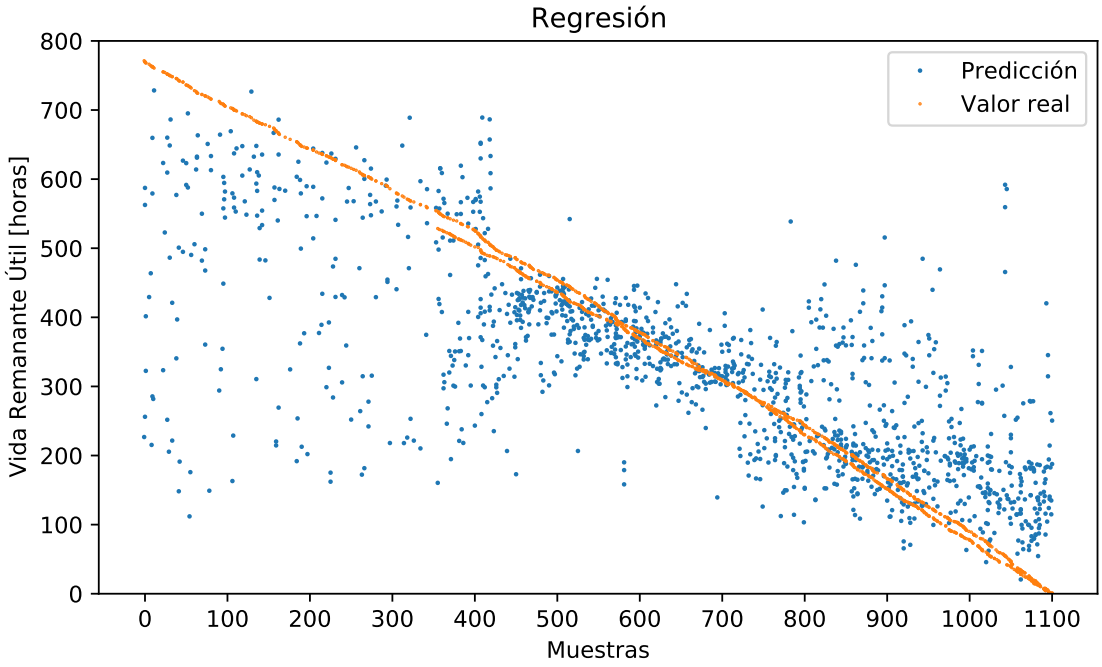

Figura 6.11: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo de red neuronal *GRU* utilizando la variable objetivo E_T.

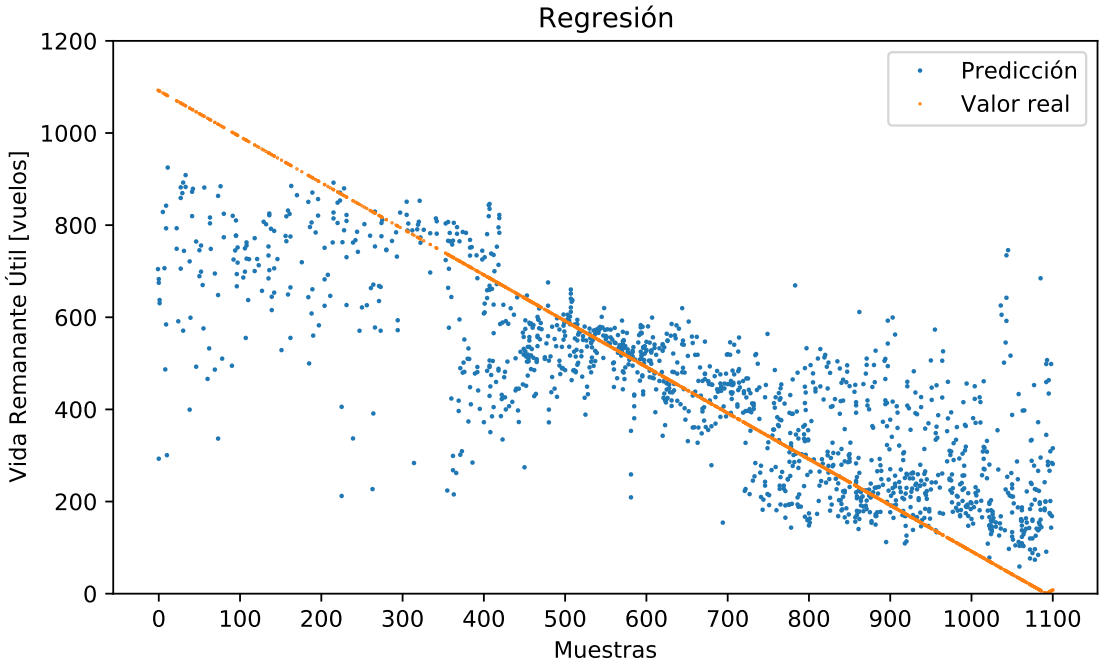

Figura 6.12: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo de red neuronal *GRU* utilizando la variable objetivo E_V.

En la tabla 6.9 se resume la comparación entre el antes y después de aplicar el post procesamiento, entregando una respuesta por vuelo. Las figuras 6.13 y 6.14 presentan las matrices de confusión para los modelos de *RF* y neuronal *LSTM*.

Tabla 6.9: Comparación entre resultados entregados por ventana de tiempo (columnas venta) y determinando un resultado por vuelo (columna vuelo) para los modelos de regresión utilizando la variable objetivo E_T.

Modelo	mse			R^2		
	Ventana	Vuelo	Diferencia	Ventana	Vuelo	Diferencia
Random Forest	0.30	0.24	-0.06	0.70	0.77	0.07
Gradient Boosting	0.33	0.26	-0.07	0.68	0.76	0.08
Capas Convolucionales	0.38	0.32	-0.06	0.64	0.71	0.07
Capas GRU	0.32	0.29	-0.03	0.68	0.71	0.03
Capas LSTM	0.34	0.28	-0.06	0.66	0.74	0.08

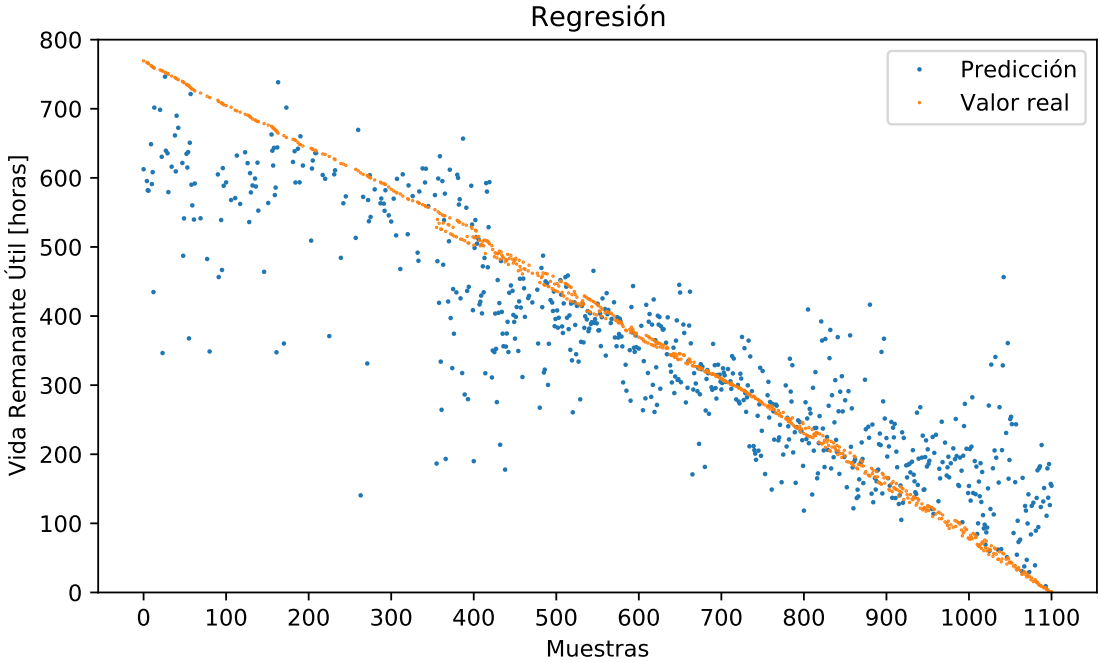

Figura 6.13: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo *RF* utilizando la variable objetivo E_T entregando una respuesta por vuelo.

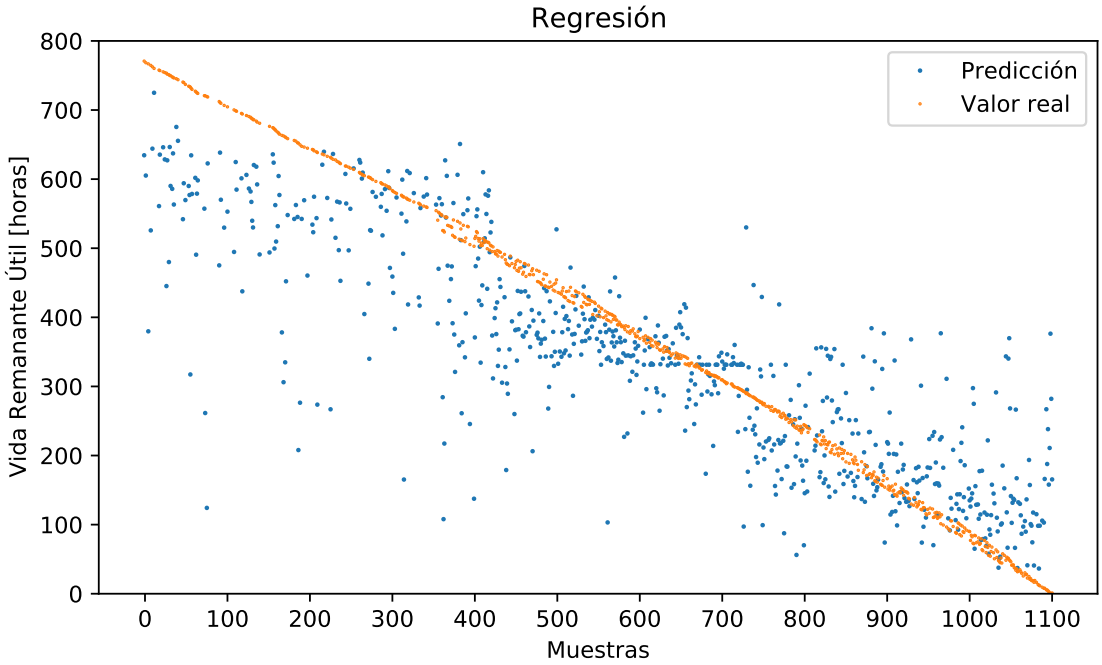

Figura 6.14: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo *LSTM* utilizando la variable objetivo E_T entregando una respuesta por vuelo.

A continuación, en la tabla 6.10 se presenta el promedio y varianza para la predicción realizada por los modelos utilizando datos sanos, distintos a los de entrenamiento. Las figuras 6.15 y 6.16 presentan dos tipos de regresiones obtenidas.

Tabla 6.10: Promedio y varianza de las predicciones realizadas por los modelos de regresión utilizando datos sanos.

Modelo	E_T		E_V	
	Promedio	Varianza	Promedio	Varianza
Random Forest	387	153.4	435	153.4
Gradient Boosting	453	87.8	470	141.3
Capas Convolucionales	785	507.3	955	170.1
Capas GRU	343	55.6	548	119.8
Capas LSTM	428	81.0	411	94.9

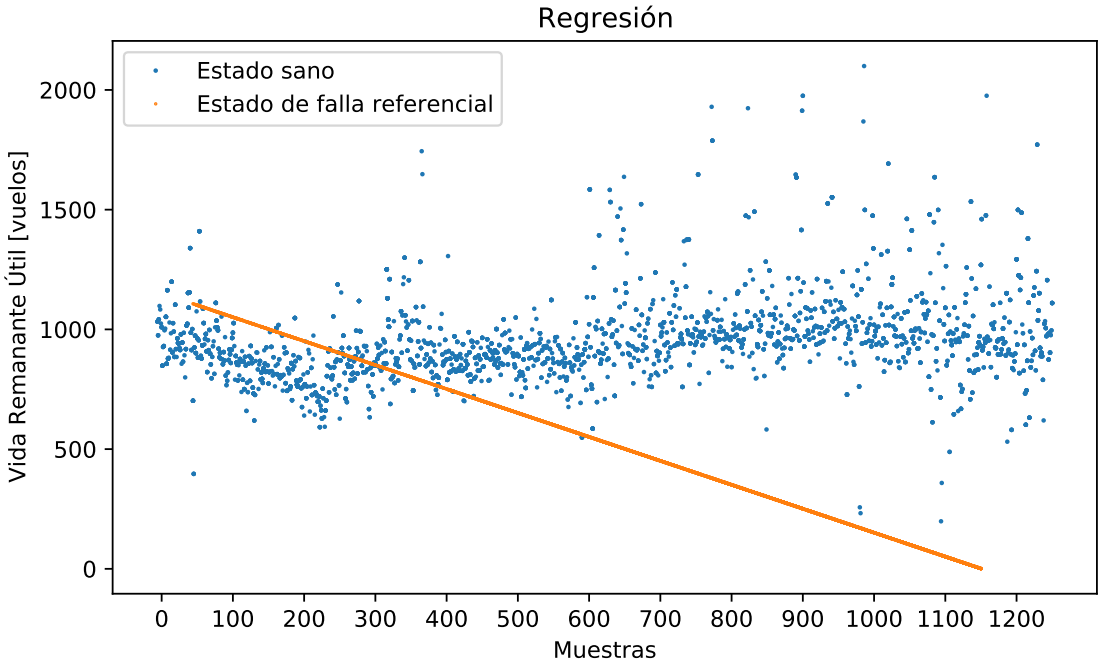

Figura 6.15: Predicciones para datos sanos del modelo *Conv1D* entrenado con la etiqueta E_V.

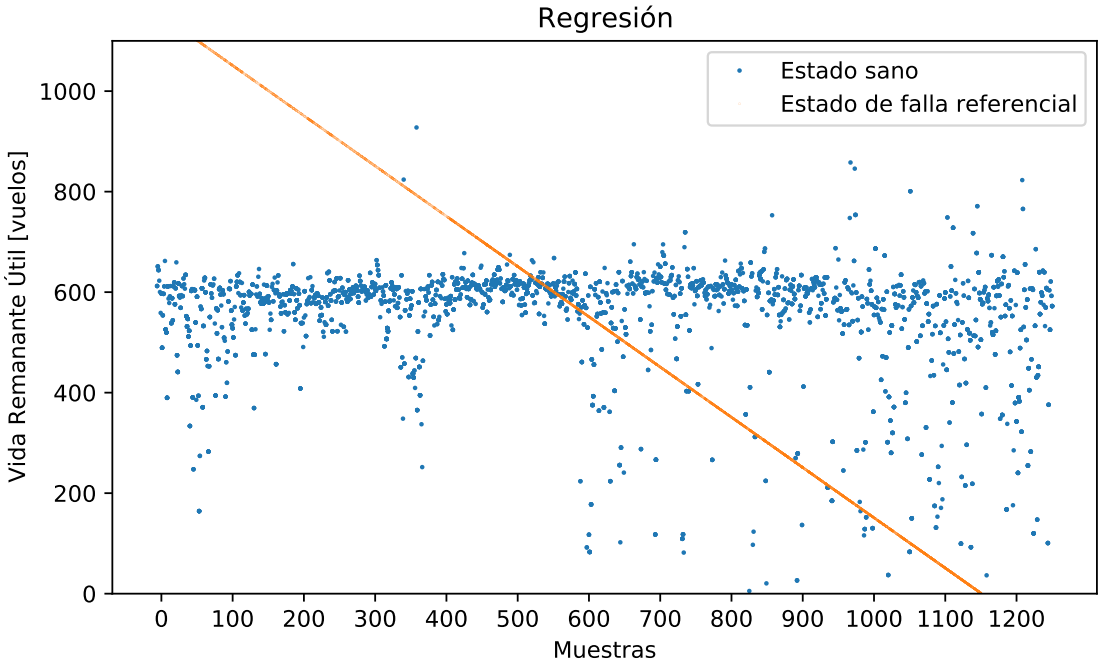

Figura 6.16: Predicciones para datos sanos del modelo *GRU* entrenado con la etiqueta E_V.

Capítulo 7

Análisis de resultados

En una primera revisión general de los resultados, se tiene que para todos los casos, los modelos entregaron resultados muy similares tanto en distribución como en desempeño, como se puede verificar en las tablas como en las figuras del capítulo 6. Esto indica que existe una tendencia en los datos identificable por herramientas de aprendizaje de máquinas y no son representaciones aleatorias de los modelos.

Por otro lado, los desempeños de las distintas métricas son regulares, bordeando el valor 0.6 en *F1 score* para los modelos de clasificación y 0.7 en métrica R^2 para los modelos de regresión, lo cual si bien es bajo para representar resultados definitivos, son aceptables para el proceso de exploración realizado. Esto implica que deben ser mejorado con alternativas como mayor volumen de datos y modelos más complejos.

A continuación, se analiza distintos puntos de relevancia con respecto a los resultados obtenidos.

7.1. Resultados entre etiquetas E_T y E_V

Al momento de definir ambas métricas, el objetivo era utilizar un método de conteo altamente representativo de los datos (conteo de tiempo remanente a la falla) y otro que, si bien sacrifica un poco de información, es más manejable para aplicar mantenimiento (conteo de vuelos remanentes a la falla). En ambos casos, el comportamiento de la variable objetivo es similar, siendo su diferencia que E_T utiliza valores reales y con pocas repeticiones de etiqueta, mientras que E_V utiliza valores naturales con múltiples repeticiones entre etiquetas.

Si se observa la tabla 6.3 que presenta los resultados en clasificación, se puede notar que el mejor modelo corresponde al *XGB* con 60.02 en *F-score*, seguido por el modelo *RF* con 57.81, ambos para la etiqueta E_T. En esta tarea, los resultados para los modelos de redes neuronales fueron mejores para dicha etiqueta, pero similares a los obtenidos con la etiqueta E_V, indicando que ambas presentan una respuesta similar a los modelos desarrollados.

Al revisar la tarea de regresión, en particular observando la tabla 6.8, se puede indicar que los comentarios son comunes con los de la tarea de clasificación, presentando un comporta-

miento similar para ambas etiquetas. Además, los gráficos de las figuras 6.9 a 6.12 presentan un comportamiento muy similar, lo que favorece la idea de que independiente de la etiqueta utilizada, se obtiene resultados similares.

Con esto, en general se puede indicar que los resultados no varían de manera importante al utilizar la etiqueta E_T y E_V, sin poder concluir si una de éstas es mejor para efectos de entrenamiento de los modelos. Sin embargo, como se ha indicado anteriormente, la etiqueta E_V puede presentar algunos beneficios para el proceso de implementación e interpretación, lo que vuelve de ella una alternativa interesante a desarrollar.

7.2. Resultados entre modelos

Para ambas tareas realizadas, los resultados alcanzados por los modelos *XGB* y *RF* superaron a los de redes neuronales tanto por etiqueta como por tipo de tarea, siendo menor la diferencia de su desempeño para la regresión que para la clasificación. Esto no necesariamente implica que estos dos modelos sean más adecuados para la tarea, dado que los modelos de redes neuronales requieren de un mayor volumen de datos de los que se poseen actualmente para ser entrenados adecuadamente.

Por otro lado, el que todos los modelos hayan alcanzado resultados similares en orden permite afirmar que existe una clara tendencia en los datos presentados y que ésta fue parcialmente definida para todos los casos. La ventaja es que confirma la robustez de los modelos, dado que varios algoritmos distintos fueron capaces de reconocer las mismas tendencias, sin embargo, esto también significa que ningún modelo sobresale del resto, por lo que acoplar más de un algoritmo en un modelo no necesariamente mejorará los resultados de manera apreciable.

De este modo, hasta el momento no existe una diferencia notoria al emplear modelos de redes neuronales, que tienden a ser más complejos que otras alternativas. Sin embargo, éstos alcanzaron resultados ligeramente inferiores a los modelos *RF* y *XGB* que no emplean unidades neuronales, por lo que al avanzar de una fase de exploración a una de desarrollo y ampliando la base de datos, los modelos de aprendizaje profundo pueden ser entrenados de manera más eficiente y precisa.

7.3. Análisis de resultados por vuelo

El análisis por vuelo es una medida de post procesamiento que permite entregar una respuesta por cada vuelo de la aeronave en lugar de cada ventana que compone dicho vuelo. Esta medida demostró ser efectiva, ya que la totalidad de los modelos mejoró su desempeño de manera importante.

Así, por ejemplo, si se observa la tabla 6.4 de resultados para clasificación, se puede observar que los modelos mejoraron en torno a tres puntos su *accuracy* y *F-score*, siendo el mejor caso la red *LSTM* con una mejora de 4.41 y 4.48 puntos respectivamente.

Para el caso de la regresión el comportamiento es similar, como se ve en la tabla 6.9 disminuyendo en 0.6 a 0.7 el error cuadrático medio y mejorando de 0.7 a 0.8 su R^2. Un

detalle que sobresale es que el modelo *GRU*, que presentó los mejores resultados para las redes neuronales, mejoró la mitad que el resto. De esto, se puede decir que este modelo presentó resultados más precisos que el resto previo al análisis por vuelo, ya que el efecto de mitigación que provocó esta conversión de resultados fue menor.

Si se observa los gráficos de resultados por vuelo para regresión (figuras 6.13 y 6.14) y se compara con sus respectivos gráficos por ventana 6.9 y 6.11), se puede notar que la cantidad de predicciones fuera de la distribución disminuyen, reafirmando la idea de que se mitiga las predicciones imprecisas de cada vuelo. Similar ocurre con la tarea de clasificación (figuras 6.5, 6.6 para predicción por vuelo y 6.1,6.3 para predicción por ventana), donde se disminuye la clasificación errónea con etiquetas vecinas y las predicciones fuera de la distribución.

7.4. Revisión de potenciales cluster en la regresión

Un hallazgo interesante es que si se compara en general los gráficos de regresiones (figuras 6.9 a 6.12), se puede apreciar que presentan la noción de tres clusters, con separación en torno a las 400 y 700 muestras. En primer lugar, la división en torno a las 500 horas o 700 vuelos remanentes coincide con el punto donde los datos de la segunda unidad comienzan a aparecer, ya que recordemos que los datos corresponden a dos unidades distintas que se juntaron en una base de datos y una de ellas comprende un período más extenso. Por ello, preliminarmente correspondería a una diferencia por la unión de dos bases de datos dispares, por lo que para poder asegurar que dicha diferencia puede significar un cluster, sería necesario validarlo con más muestras, a la vez que se amplía y homogeneiza la base de datos.

Por otro lado, el cluster que se manifiesta en torno a las 400 horas o 300 vuelos podría corresponder a una diferencia en el funcionamiento de las unidades. Mediante la validación de los modelos empleando los datos sanos analizada en la próxima sección, se consiguió relacionar la respuesta de algunos modelos con esta anomalía, lo que podría representar el inicio de la manifestación de la falla. Con la información disponible, no es posible concluir acerca de la naturaleza de dicha anomalía, pero sí se tiene evidencia que la relaciona a una diferencia con un estado sano.

7.5. Validación de datos sanos

Realizar la validación de los modelos utilizando datos sanos es una etapa necesaria para confirmar que los modelos no estuviesen modelando la degradación natural del equipo y, que en su lugar, representan la falla estudiada. En esta línea, al realizar la predicción mediante el set de datos sano, se puede identificar dos tipos de respuestas a partir de los modelos.

Por un lado, como se muestra en la figura 6.7 para clasificación y 6.15 para regresión, algunos modelos presentaron resultados alineados con lo esperado, alejándose notoriamente de una respuesta de falla. Además, el modelo *Conv1D* presentó un R^2 de 0.64 para la etiqueta E_V, lo que confirma la capacidad de diferenciar el estado sano y en falla.

Por otra parte, las figuras 6.8 y 6.16 presentan una predicción menor, en torno a las 400 horas para el modelo *XGB* de clasificación. Esto, si bien se aleja un poco de lo esperado para

datos sanos, sí se relaciona a la anomalía detectada en la sección anterior, que se ubicaba en el mismo rango. La regresión de la figura 6.16 modela una distribución en torno a los 600 vuelos, que vendría a corresponder al cluster central que puede apreciarse en cualquier regresión de las obtenidas.

Capítulo 8

Conclusión

En primer lugar, los resultados alcanzados son insuficientes para entrar en una etapa de implementación firme, dado que el error alcanzado para todos los resultados supera el 40 %. Sin embargo, desde un punto de vista exploratorio, se consiguió identificar una tendencia existente en los datos, lo que permite indicar que un desarrollo más profundo es prometedor para obtener resultados aceptables.

Se consigue desarrollar tres modelos de redes neuronales con resultados consistentes entre ellos y similares a los modelos *RF* y *XGB*, algoritmos más clásicos de aprendizaje de máquinas, lo que permite asegurar la robustez de los modelos entrenados. Por otro lado, si bien no presentan una capacidad de predicción sobresaliente, existen múltiples aristas que pueden ser desarrolladas en fases posteriores del proyecto para mejorar su desempeño en general. De los dos tipos de etiquetas definidas, ambas presentaron resultados muy similares tanto gráfica como numéricamente. Esto implica que no conlleva algún beneficio en el entrenamiento al elegir una por sobre otra.

En las regresiones se detectó la formación de dos clusters que si se analizan con los resultados obtenidos utilizando los datos sanos, aparece una posible relación a la falla crítica que se busca identificar. En particular, podría ayudar a definir el momento en que la falla comienza a desarrollarse de manera apreciable. No obstante lo anterior, la información existente impide llegar a un diagnóstico resolutivo al respecto, por lo que se propone profundizar su análisis a futuro.

Por otra parte, el post procesamiento por vuelo permite mejorar los resultados obtenidos y eliminar predicciones fuera de la distribución, por lo que es una buena medida para mejorar el desempeño final. Sin embargo, al no ser utilizada esta información en la fase de entrenamiento, no se extrae relaciones relevantes con el resto de las variables, limitando la capacidad de los modelos.

Finalmente, se estima que con la información disponible es posible realizar la predicción de la falla de manera aceptable, además de obtener información relevante sobre la falla, pero aún es necesario desarrollar más el proyecto para que ésta pueda ser utilizada de manera confiable.

8.1. Trabajo futuro

Los resultados presentados confirman la existencia de tendencias en la base de datos presente, pero aún son insuficientes para ser utilizados por el área de mantenimiento. En esta línea, se propone las siguientes acciones para obtener un modelo más robusto y preciso:

1. Ampliar la base de datos agregando más unidades en falla, ya que muchos de los filtros que son necesarios reducen la cantidad de datos de manera considerable. Se puede explorar las aristas de generación de datos o uso de bases de datos estándar con aprendizaje reforzado.
2. Revisar la naturaleza de la anomalía detectada en la sección de análisis y verificar si ésta se relaciona con la falla.
3. De ser afirmativo el punto anterior, separar la parte sana de la parte enferma y utilizar cada set de datos acorde a su estado.
4. Obtener y utilizar datos sin falla en el proceso de entrenamiento.
5. Agregar características relevantes para encontrar algunas que respondan de manera más eficiente.
6. Involucrar en el desarrollo del modelo el estado de la aeronave en despegue.

Apéndice A

Resultados adicionales

A.1. Arquitecturas de redes neuronales de clasificación

Tabla A.1: Estructura de la red neuronal *Conv1D* utilizada para la etiqueta E_T

Capa	Parámetro	Valor
Capa Conv1D	Neuronas	179
Capa Conv1D	Neuronas	101
Capa Flatten	-	-
Fully connected	Neuronas	100
	Función de activación	Relu
Fully connected	Neuronas	10
	Función de activación	Softmax

Tabla A.2: Estructura de la red neuronal *GRU* utilizada para la etiqueta E_T

Capa	Parámetro	Valor
Batch Normalization	-	-
Capa GRU	Neuronas	243
Batch Normalization	-	-
Capa GRU	Neuronas	139
Batch Normalization	-	-
Capa GRU	Neuronas	81
Capa Flatten	-	-
Fully connected	Neuronas	77
	Función de activación	Relu
Fully connected	Neuronas	10
	Función de activación	Softmax

Tabla A.3: Estructura de la red neuronal *Conv1D* utilizada para la etiqueta E_V

Capa	Parámetro	Valor
Capa Conv1D	Neuronas	249
Dropout	Drop	0.2
Capa Conv1D	Neuronas	221
Dropout	Drop	0.2
Capa Conv1D	Neuronas	206
Dropout	Drop	0.2
Capa Flatten	-	-
Fully connected	Neuronas	55
	Función de activación	Relu
Fully connected	Neuronas	10
	Función de activación	Softmax

Tabla A.4: Estructura de la red neuronal *GRU* utilizada para la etiqueta E_V

Capa	Parámetro	Valor
Capa GRU	Neuronas	166
Dropout	Drop	0.2
Capa GRU	Neuronas	138
Dropout	Drop	0.2
Capa GRU	Neuronas	121
Dropout	Drop	0.2
Capa Flatten	-	-
Fully connected	Neuronas	127
	Función de activación	Relu
Fully connected	Neuronas	56
	Función de activación	Relu
Fully connected	Neuronas	10
	Función de activación	Softmax

Tabla A.5: Estructura de la red neuronal *LSTM* utilizada para la etiqueta E_T

Capa	Parámetro	Valor
Batch Normalization	-	-
Capa LSTM	Neuronas	248
Dropout	Drop	0.2
Batch Normalization	-	-
Capa LSTM	Neuronas	148
Dropout	Drop	0.2
Batch Normalization	-	-
Capa LSTM	Neuronas	109
Dropout	Drop	0.2
Capa Flatten	-	-
Fully connected	Neuronas	57
	Función de activación	Relu
Fully connected	Neuronas	50
	Función de activación	Softmax

A.2. Resultados de clasificación por modelo *RF*

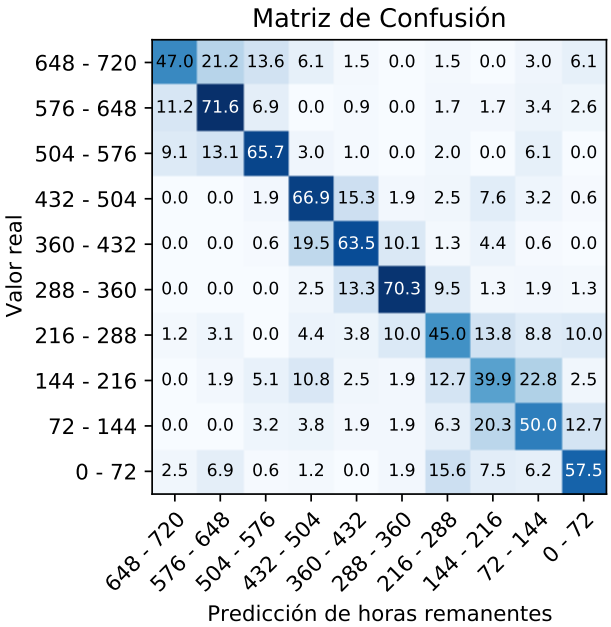

Figura A.1: Matriz de confusión obtenida para la clasificación por *RF* utilizando la variable objetivo E_T.

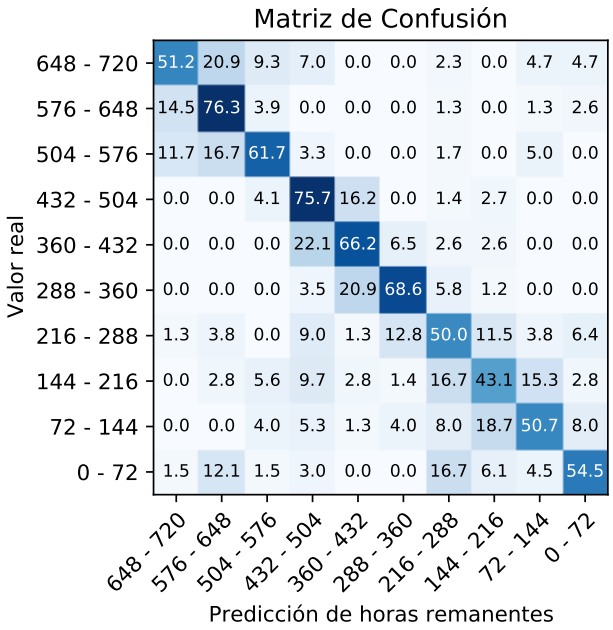

Figura A.2: Matriz de confusión obtenida para la clasificación por RF utilizando la variable objetivo E_T entregando una respuesta por vuelo.

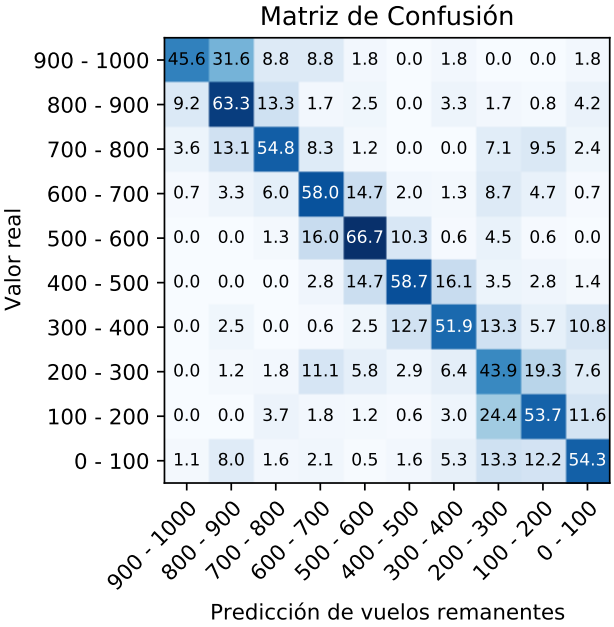

Figura A.3: Matriz de confusión obtenida para la clasificación por RF utilizando la variable objetivo E_V.

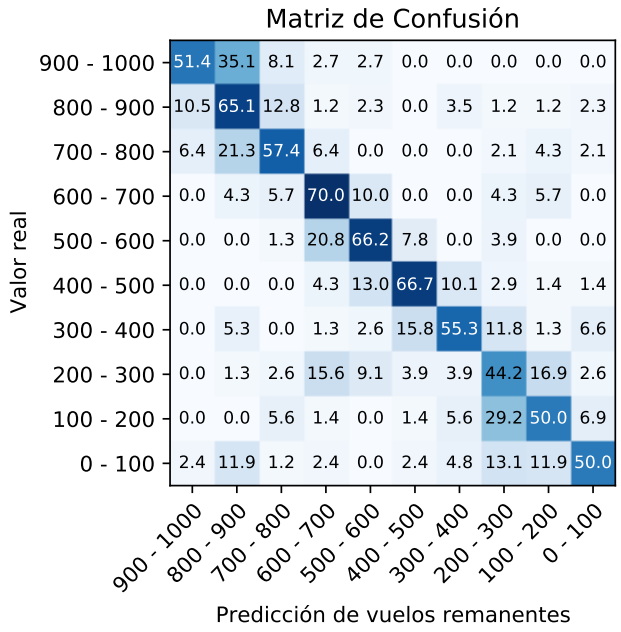

Figura A.4: Matriz de confusión obtenida para la clasificación por RF utilizando la variable objetivo E_V entregando una respuesta por vuelo.

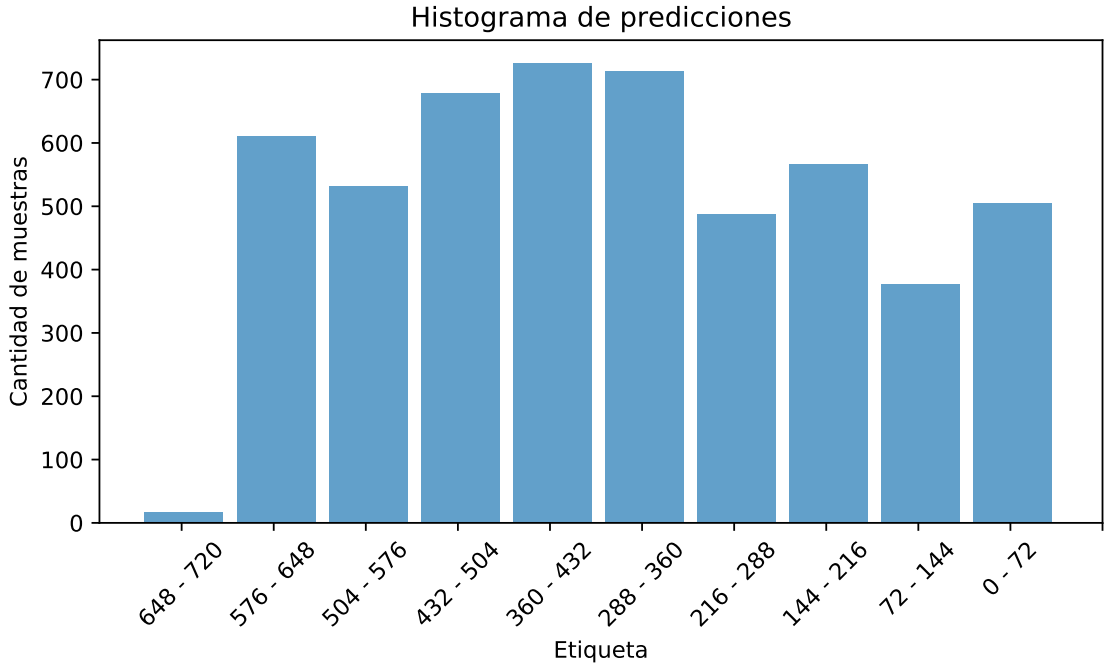

Figura A.5: Predicciones para datos sanos del modelo RF entrenado con la etiqueta E_T.

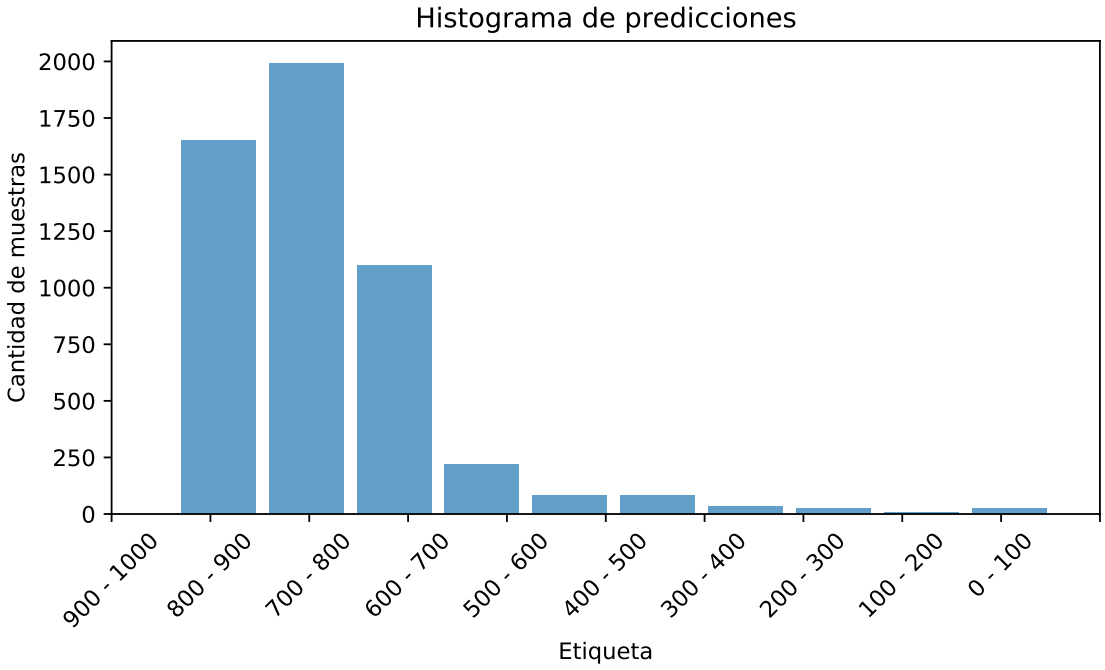

Figura A.6: Predicciones para datos sanos del modelo RF entrenado con la etiqueta E_V.

A.3. Resultados de clasificación por modelo *XGB*

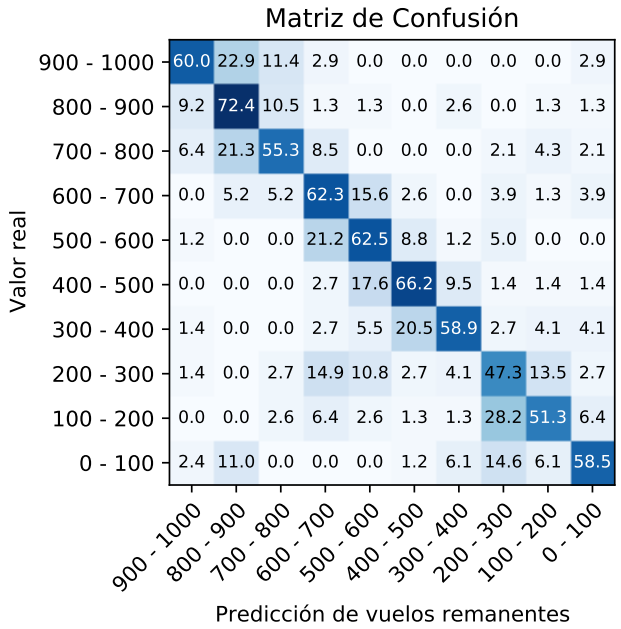

Figura A.7: Matriz de confusión obtenida para la clasificación por XGB utilizando la variable objetivo E_V entregando una respuesta por vuelo.

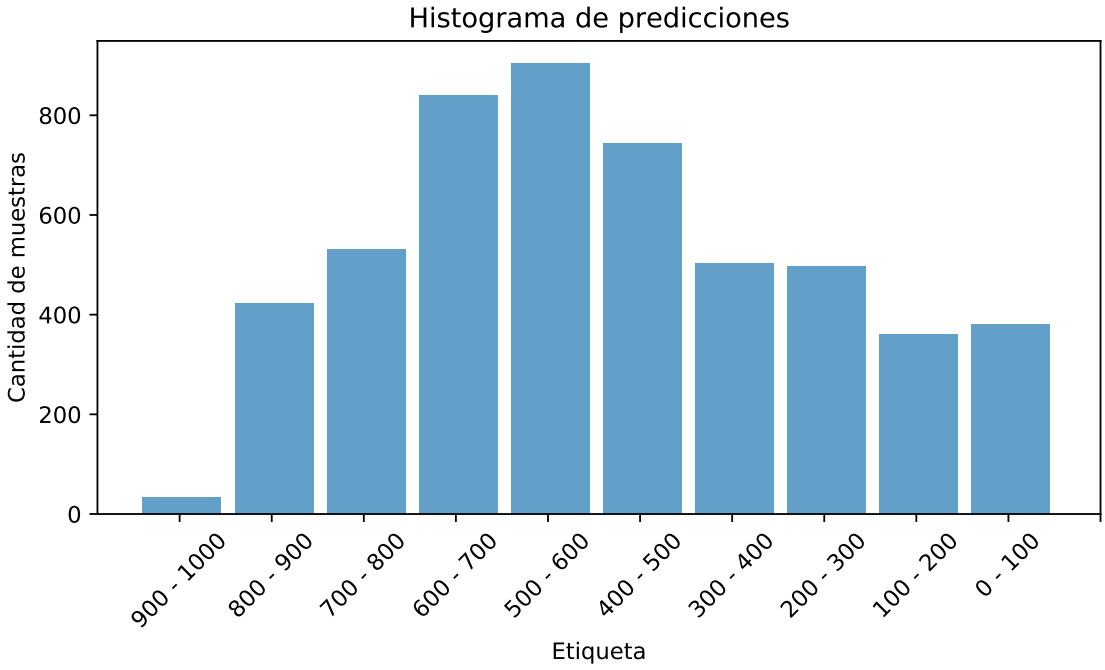

Figura A.8: Predicciones para datos sanos del modelo *XGB* entrenado con la etiqueta E_V.

A.4. Resultados de clasificación por modelo *Conv1D*

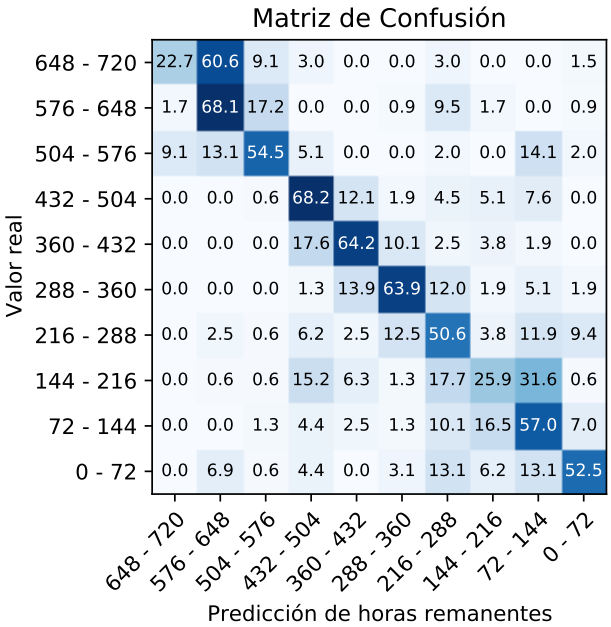

Figura A.9: Matriz de confusión obtenida para la clasificación por *Conv1D* utilizando la variable objetivo E_T.

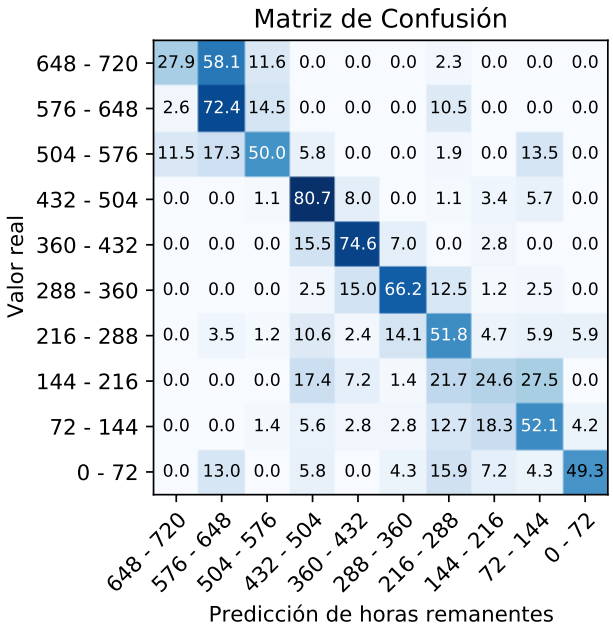

Figura A.10: Matriz de confusión obtenida para la clasificación por *Conv1D* utilizando la variable objetivo E_T entregando una respuesta por vuelo.

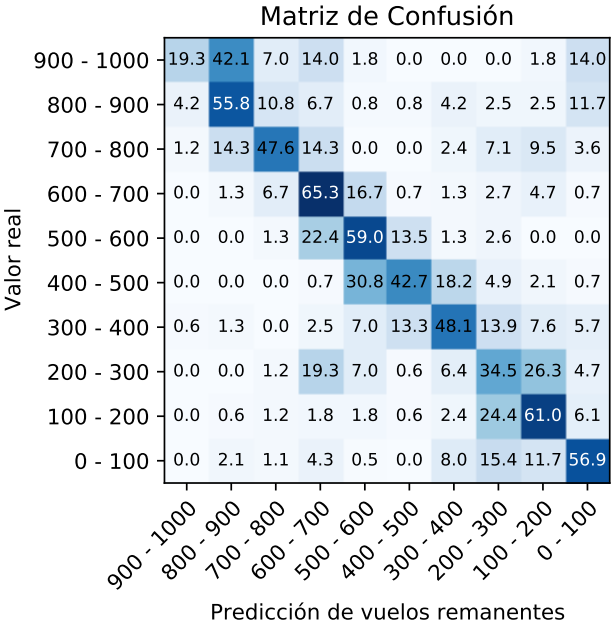

Figura A.11: Matriz de confusión obtenida para la clasificación por *Conv1D* utilizando la variable objetivo E_V.

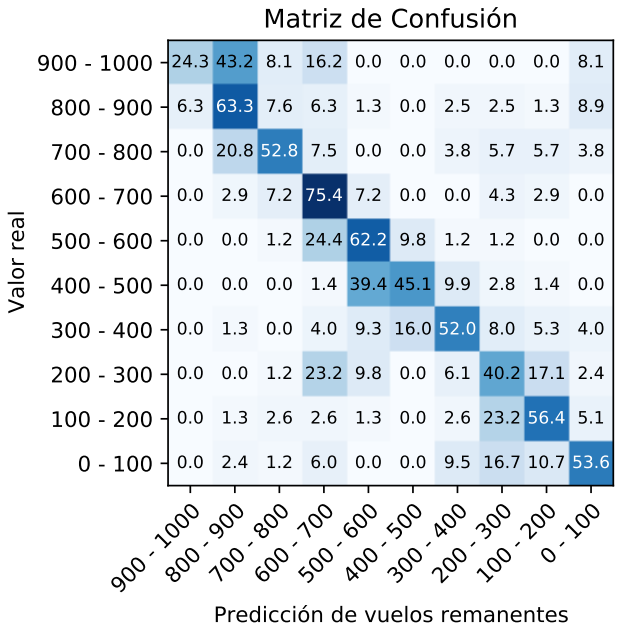

Figura A.12: Matriz de confusión obtenida para la clasificación por *Conv1D* utilizando la variable objetivo E_V entregando una respuesta por vuelo.

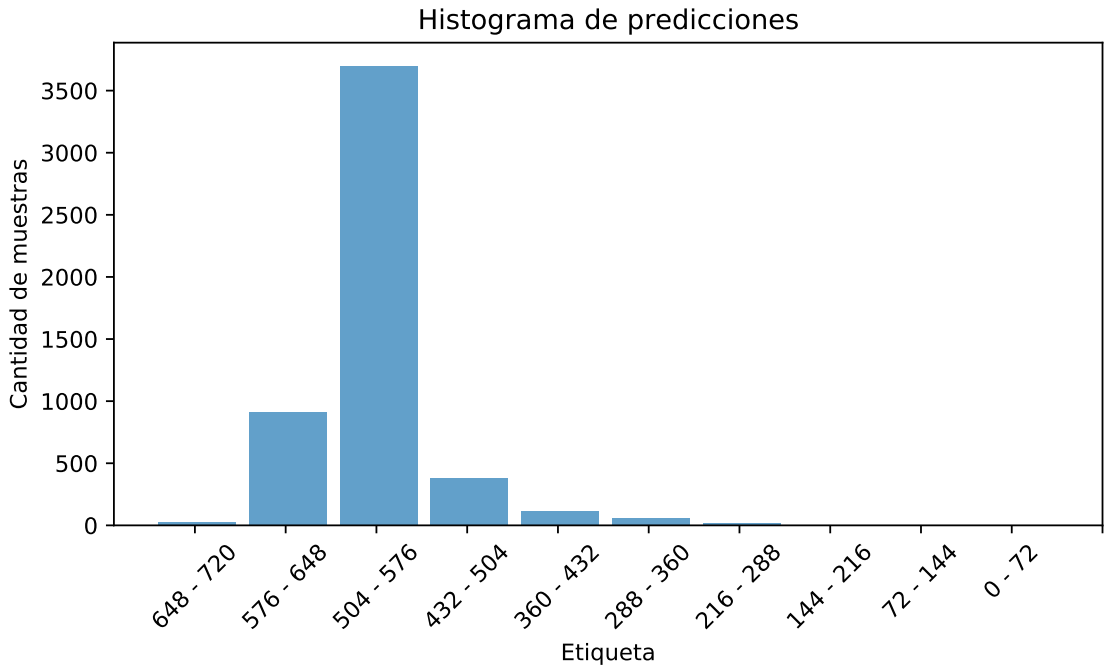

Figura A.13: Predicciones para datos sanos del modelo *Conv1D* entrenado con la etiqueta E_T.

A.5. Resultados de clasificación por modelo neuronal *GRU*

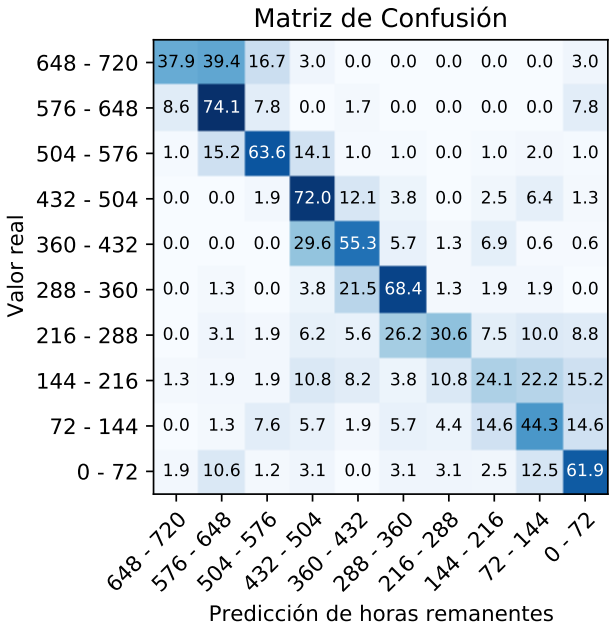

Figura A.14: Matriz de confusión obtenida para la clasificación por un modelo de red neuronal con capas *GRU* utilizando la variable objetivo E_T.

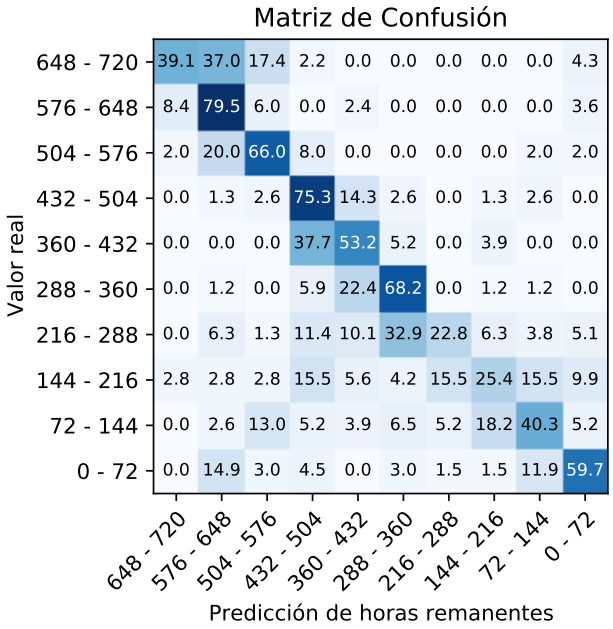

Figura A.15: Matriz de confusión obtenida para la clasificación por un modelo de red neuronal con capas *GRU* utilizando la variable objetivo E_T entregando una respuesta por vuelo.

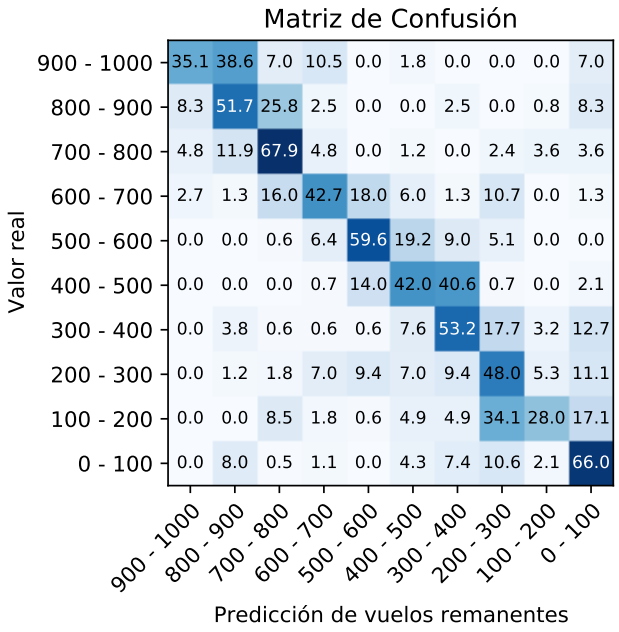

Figura A.16: Matriz de confusión obtenida para la clasificación por un modelo de red neuronal con capas *GRU* utilizando la variable objetivo E_V.

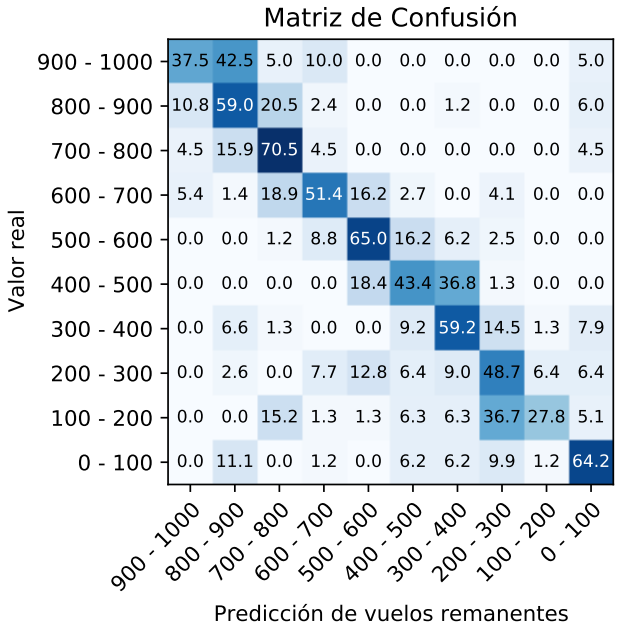

Figura A.17: Matriz de confusión obtenida para la clasificación por un modelo de red neuronal con capas *GRU* utilizando la variable objetivo E_V entregando una respuesta por vuelo.

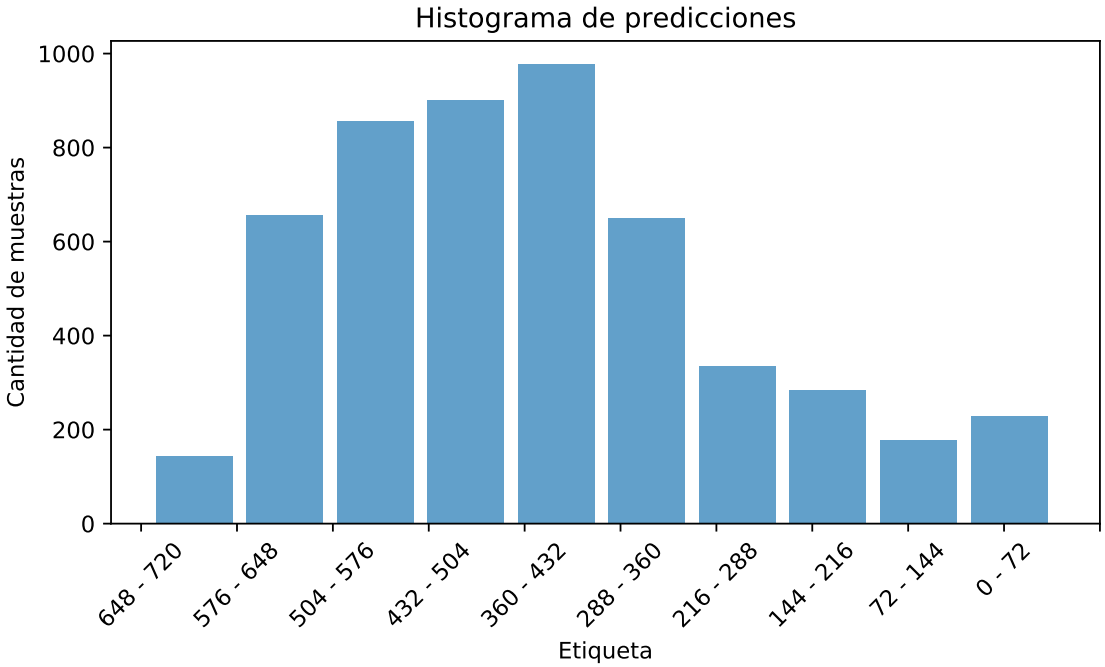

Figura A.18: Predicciones para datos sanos del modelo *GRU* entrenado con la etiqueta E_T.

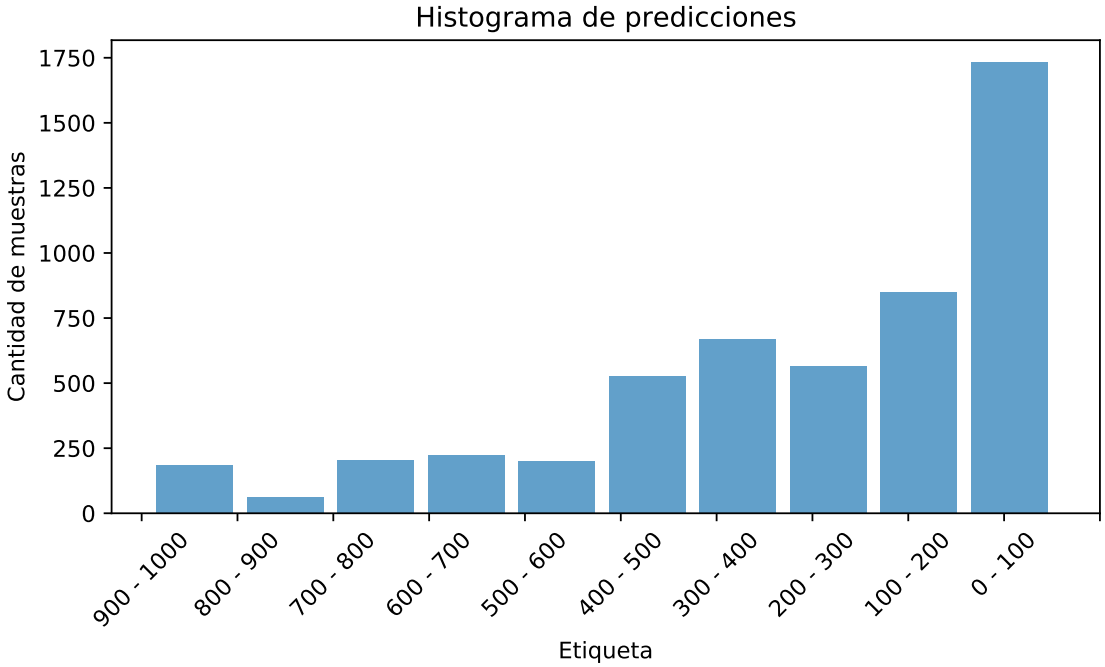

Figura A.19: Predicciones para datos sanos del modelo *GRU* entrenado con la etiqueta E_V.

A.6. Resultados de clasificación por modelo *LSTM*

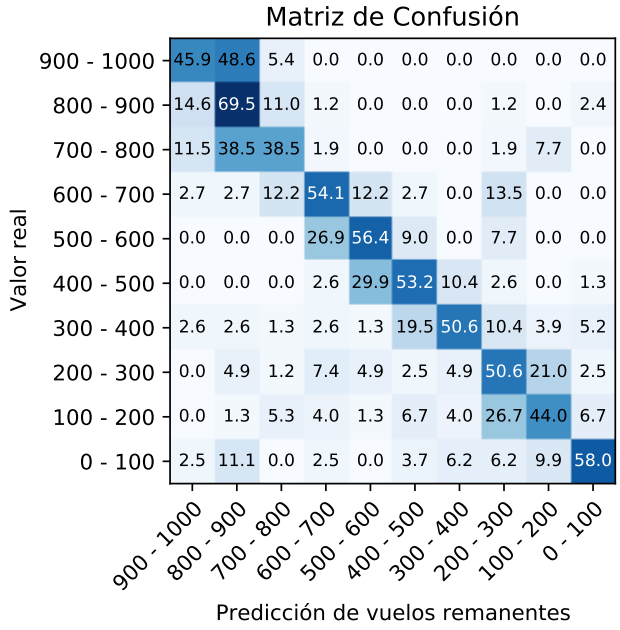

Figura A.20: Matriz de confusión obtenida para la clasificación por un modelo de red neuronal con capas *LSTM* utilizando la variable objetivo E_V entregando una respuesta por vuelo.

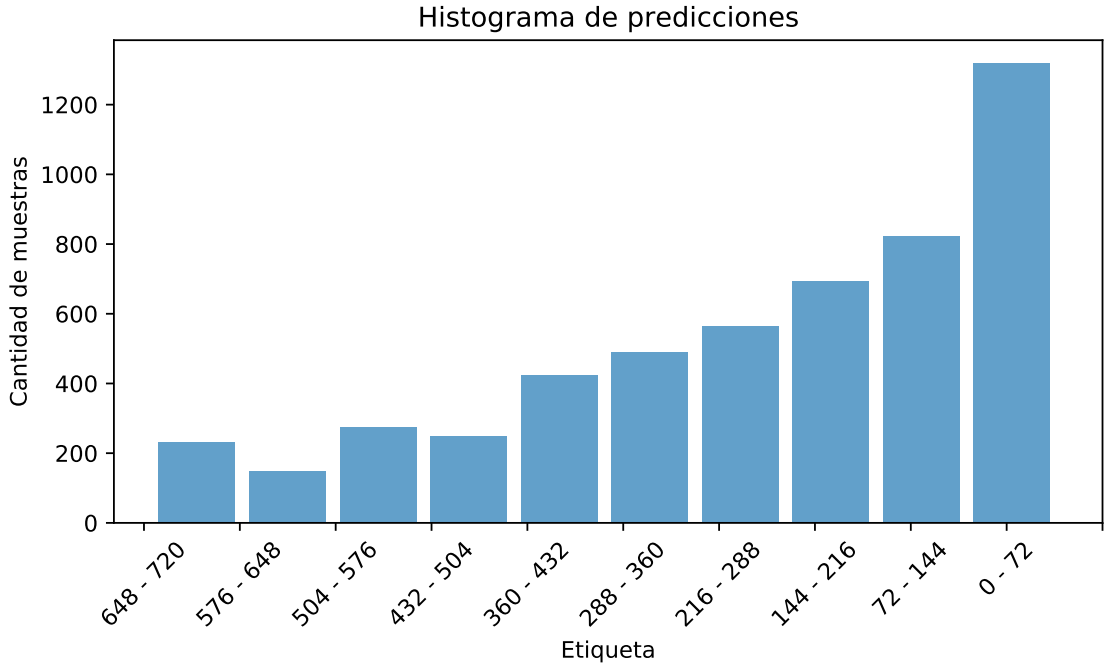

Figura A.21: Predicciones para datos sanos del modelo *LSTM* entrenado con la etiqueta E_T.

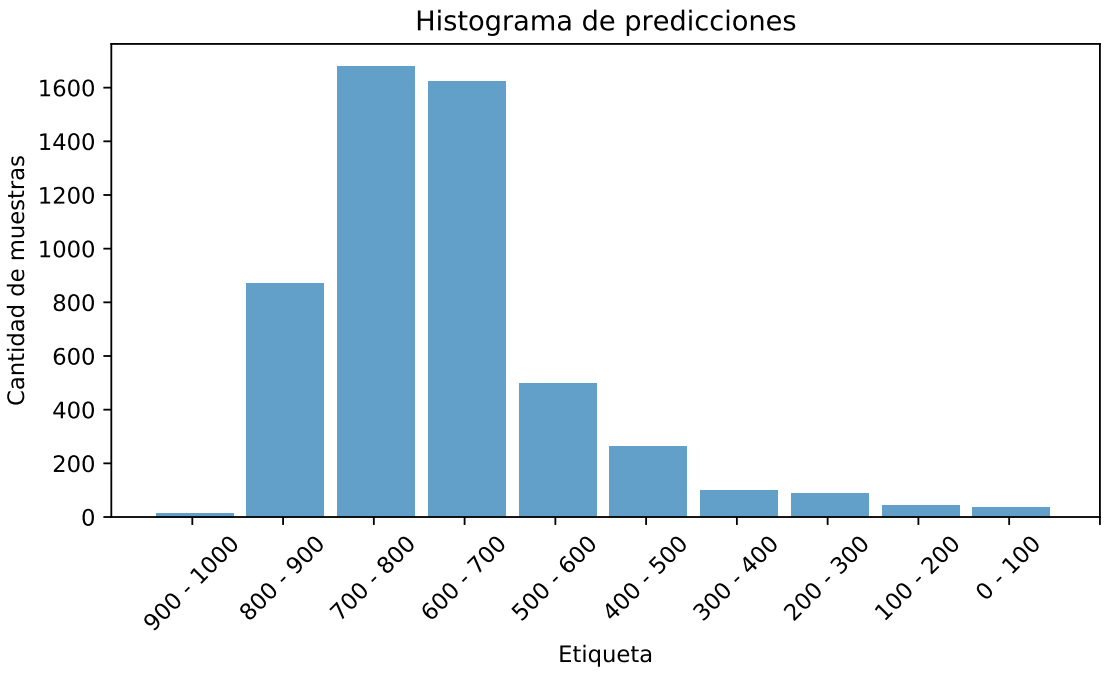

Figura A.22: Predicciones para datos sanos del modelo *LSTM* entrenado con la etiqueta E_V.

A.7. Arquitecturas de redes neuronales de regresión

Tabla A.6: Estructura de la red neuronal *Conv1D* utilizada para la etiqueta E_T

Capa	Parámetro	Valor
Capa Conv1D	Neuronas	253
Capa Conv1D	Neuronas	141
Capa Conv1D	Neuronas	80
Capa Flatten	-	-
Fully connected	Neuronas	105
	Función de activación	Relu
Fully connected	Neuronas	1

Tabla A.7: Estructura de la red neuronal *LSTM* utilizada para la etiqueta E_T

Capa	Parámetro	Valor
Batch Normalization	-	-
Capa LSTM	Neuronas	203
Batch Normalization	-	-
Capa LSTM	Neuronas	142
Batch Normalization	-	-
Capa LSTM	Neuronas	103
Capa Flatten	-	-
Fully connected	Neuronas	6
	Función de activación	Relu
Fully connected	Neuronas	1

Tabla A.8: Estructura de la red neuronal *Conv1D* utilizada para la etiqueta E_V

Capa	Parámetro	Valor
Batch Normalization	-	-
Capa Conv1D	Neuronas	157
Batch Normalization	-	-
Capa Conv1D	Neuronas	149
Batch Normalization	-	-
Capa Conv1D	Neuronas	76
Capa Flatten	-	-
Fully connected	Neuronas	46
	Función de activación	Relu
Fully connected	Neuronas	8
	Función de activación	Relu
Fully connected	Neuronas	1

Tabla A.9: Estructura de la red neuronal *GRU* utilizada para la etiqueta E_V

Capa	Parámetro	Valor
Capa LSTM	Neuronas	237
Capa LSTM	Neuronas	189
Capa LSTM	Neuronas	122
Capa Flatten	-	-
Fully connected	Neuronas	104
	Función de activación	Relu
Fully connected	Neuronas	1

Tabla A.10: Estructura de la red neuronal *LSTM* utilizada para la etiqueta E_V

Capa	Parámetro	Valor
Capa LSTM	Neuronas	204
Dropout	Drop	0.2
Capa LSTM	Neuronas	123
Dropout	Drop	0.2
Capa LSTM	Neuronas	122
Dropout	Drop	0.2
Capa Flatten	-	-
Fully connected	Neuronas	75
	Función de activación	Relu
Fully connected	Neuronas	1

A.8. Resultados de regresión por modelo *RF*

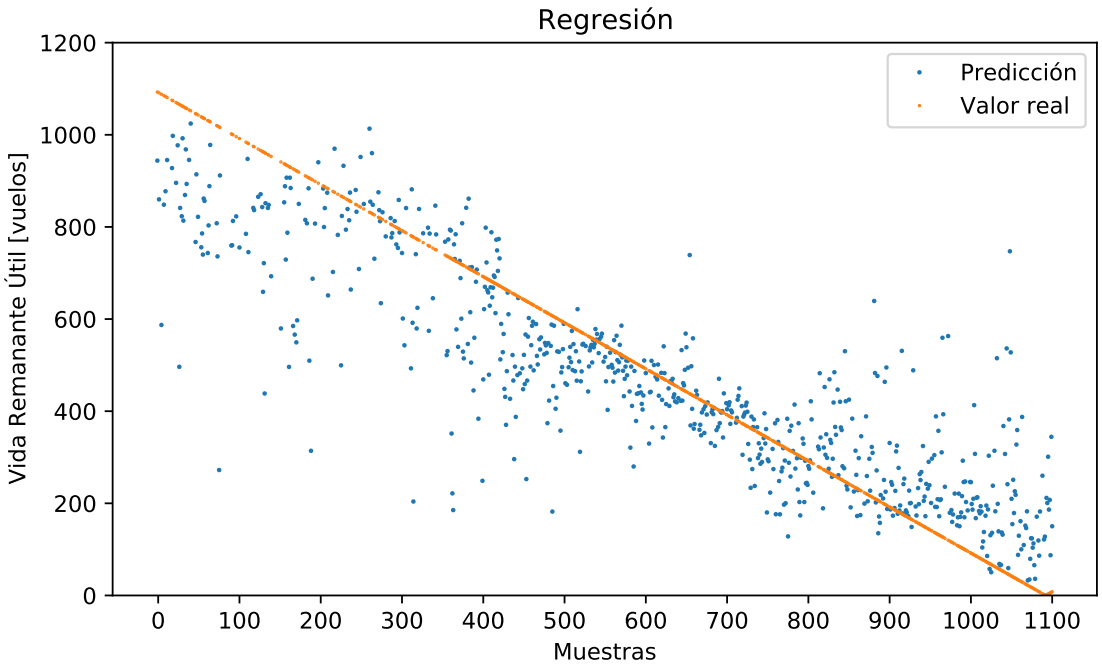

Figura A.23: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo *RF* utilizando la variable objetivo E_V entregando una respuesta por vuelo.

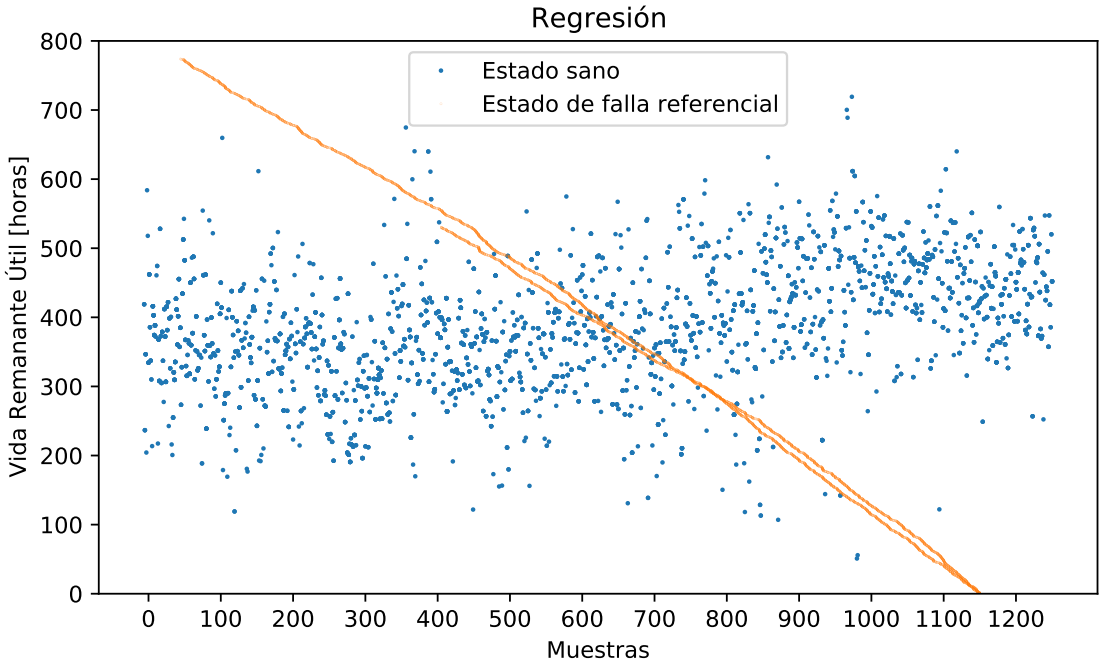

Figura A.24: Predicciones para datos sanos del modelo RF entrenado con la etiqueta E_T.

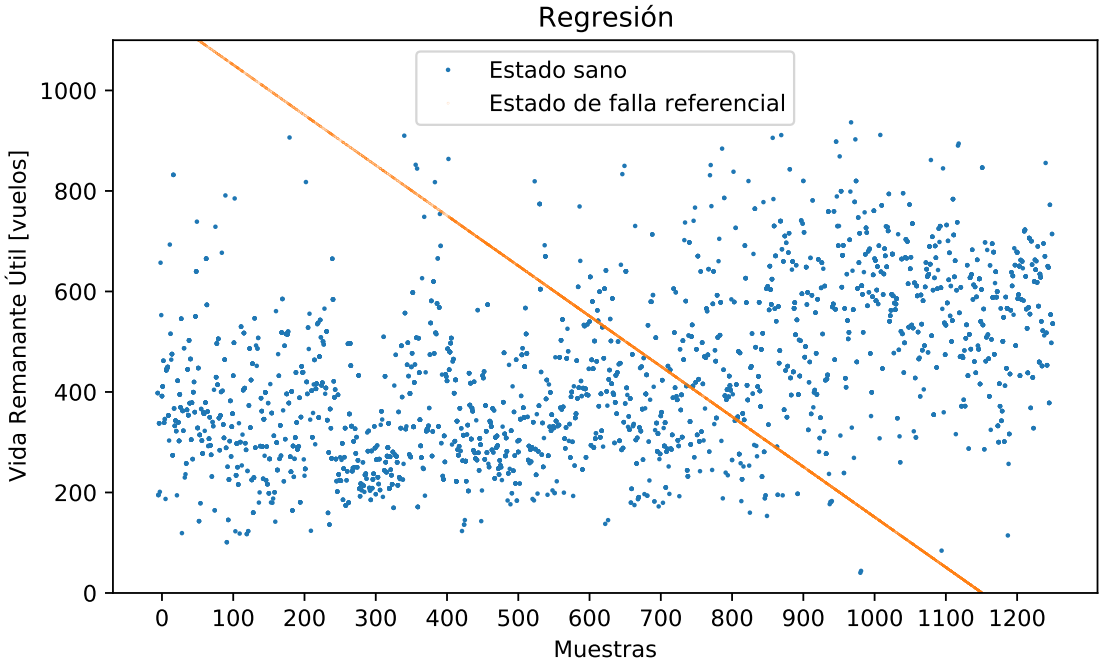

Figura A.25: Predicciones para datos sanos del modelo RF entrenado con la etiqueta E_V.

A.9. Resultados de regresión por modelo neuronal *XGB*

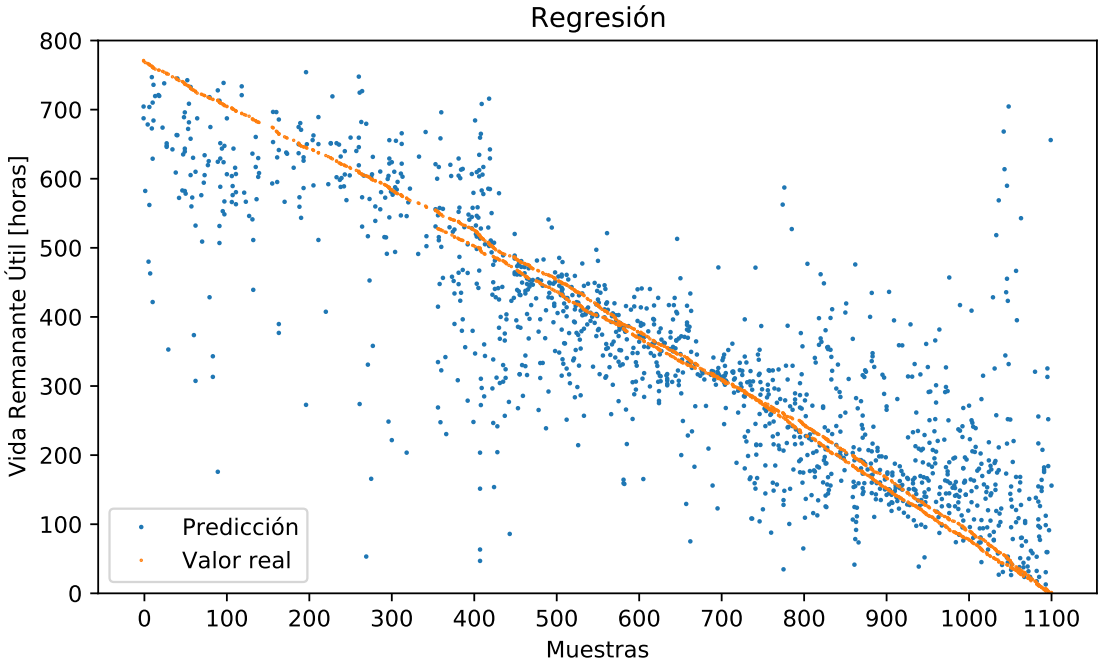

Figura A.26: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo de red neuronal con capas *XGB* utilizando la variable objetivo E_T.

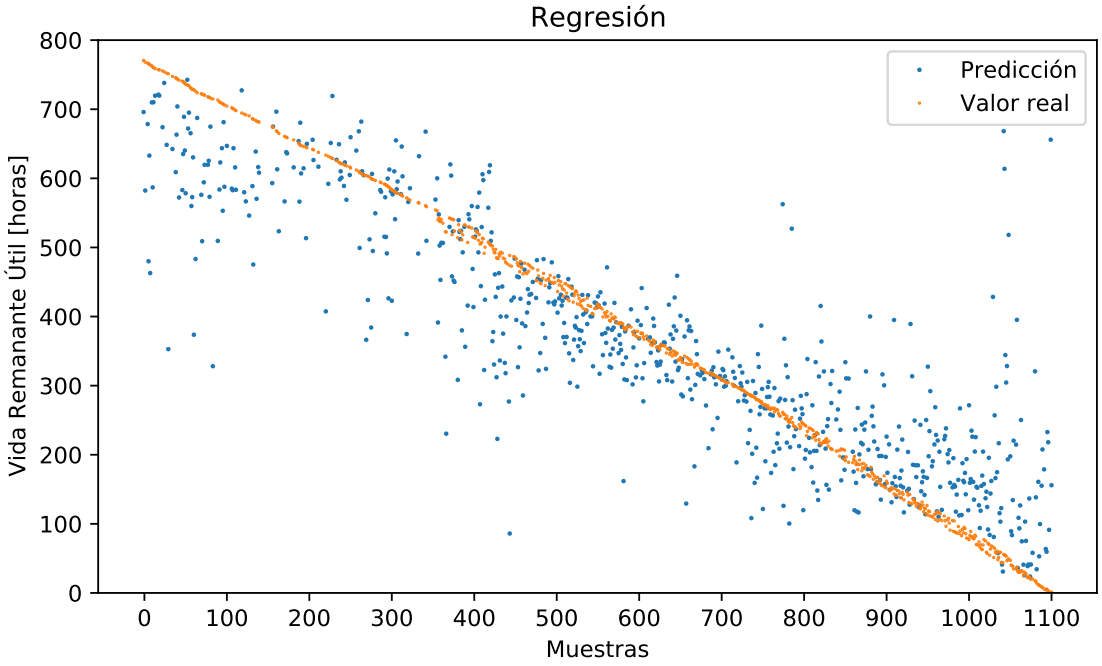

Figura A.27: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo de red neuronal con capas *XGB* utilizando la variable objetivo E_T entregando una respuesta por vuelo.

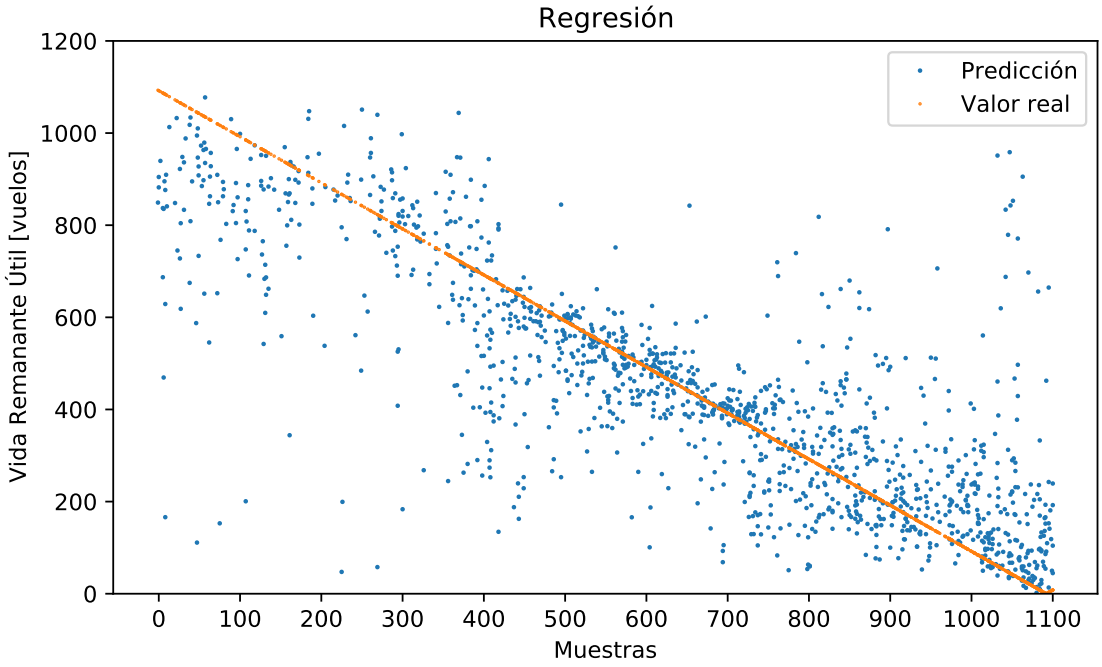

Figura A.28: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo de red neuronal con capas XGB utilizando la variable objetivo E_V.

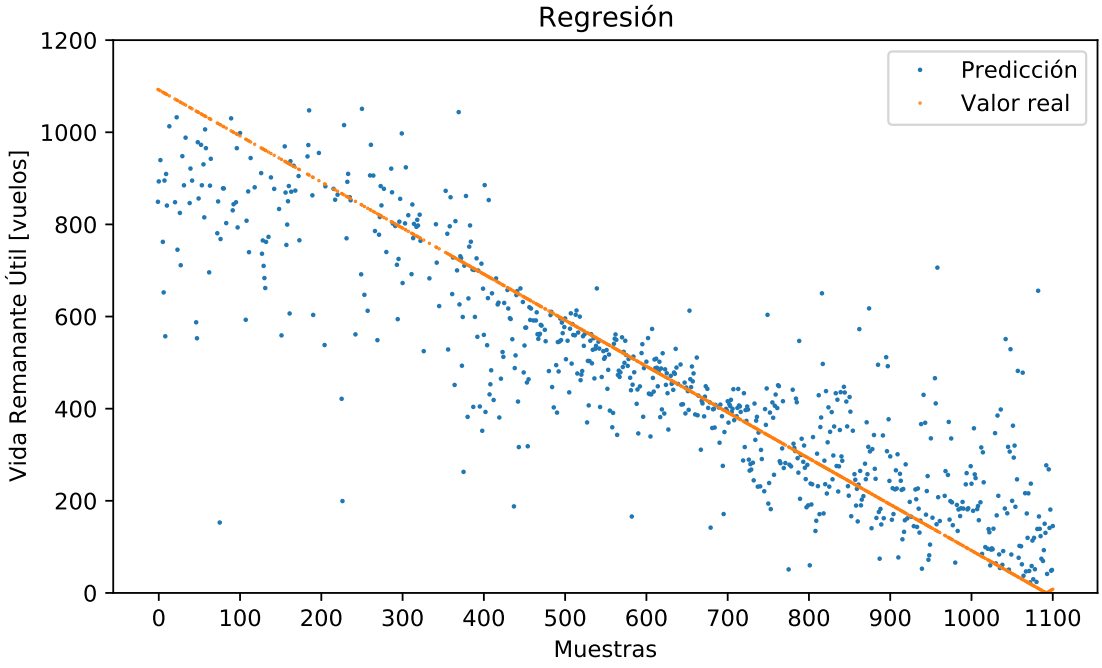

Figura A.29: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo de red neuronal con capas XGB utilizando la variable objetivo E_V entregando una respuesta por vuelo.

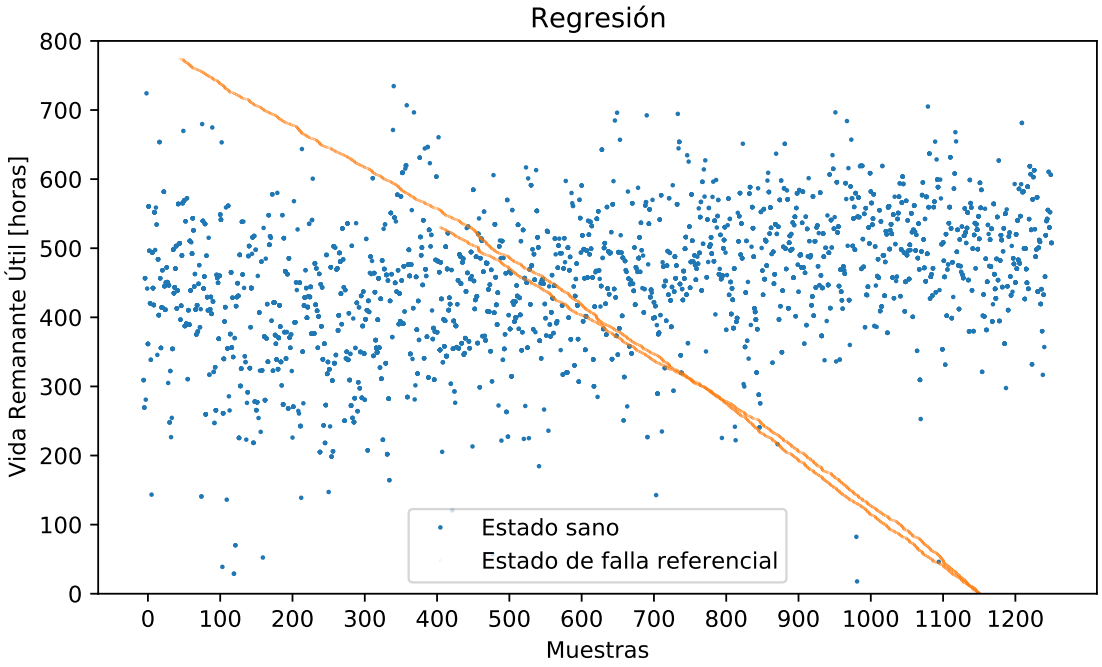

Figura A.30: Predicciones para datos sanos del modelo *XGB* entrenado con la etiqueta E_T.

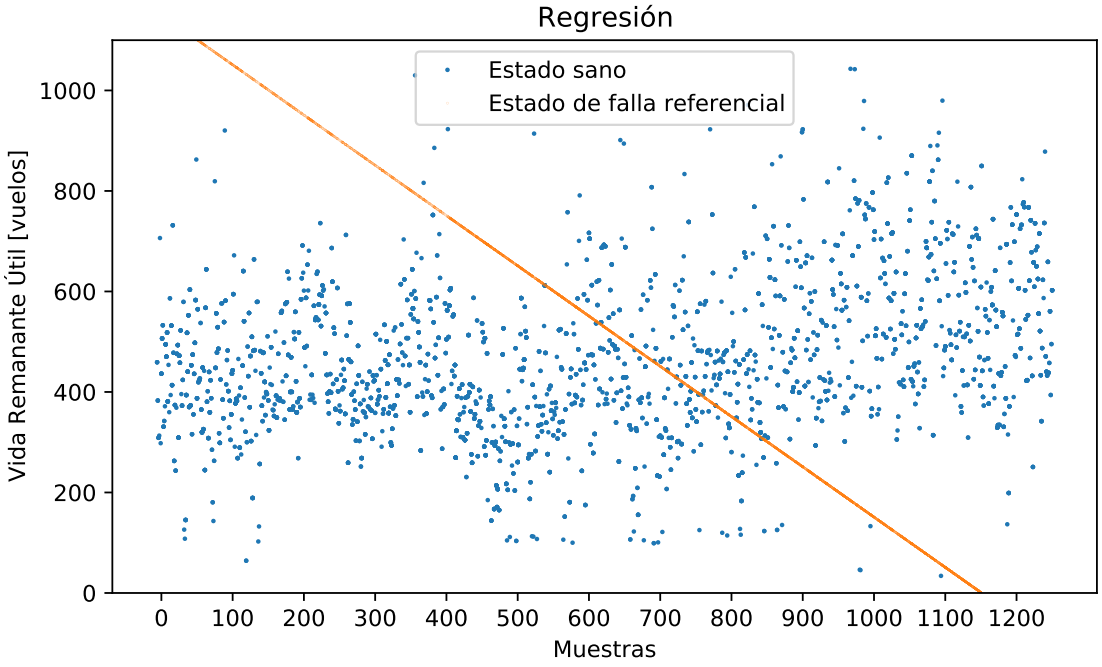

Figura A.31: Predicciones para datos sanos del modelo *XGB* entrenado con la etiqueta E_V.

A.10. Resultados de regresión por modelo neuronal *Conv1D*

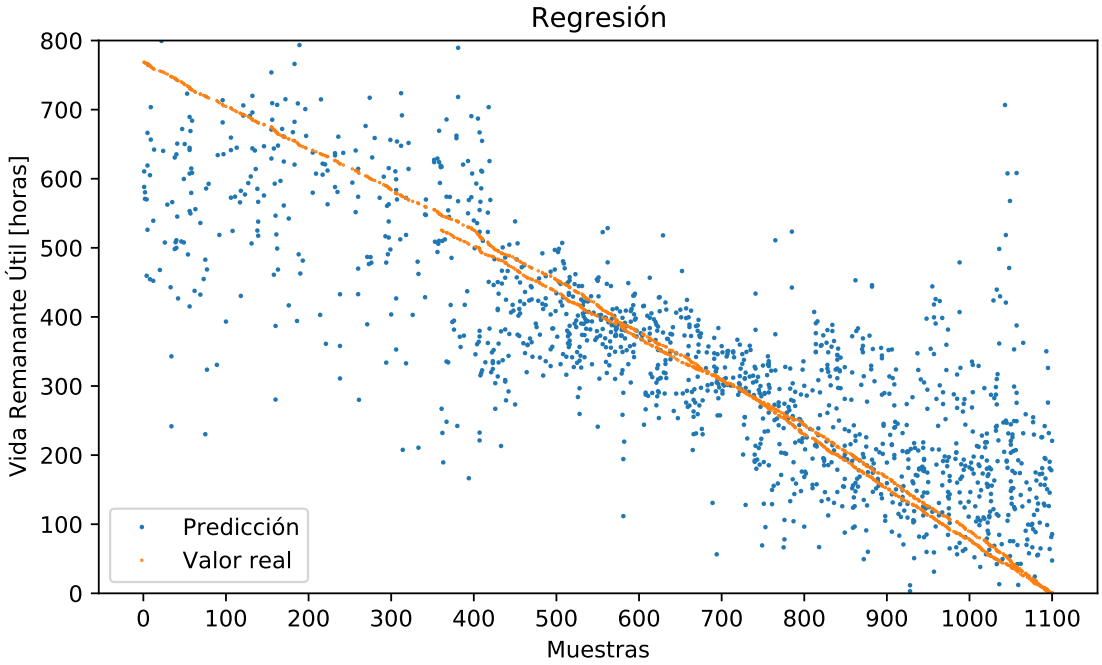

Figura A.32: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo de red neuronal con capas *Conv1D* utilizando la variable objetivo E_T.

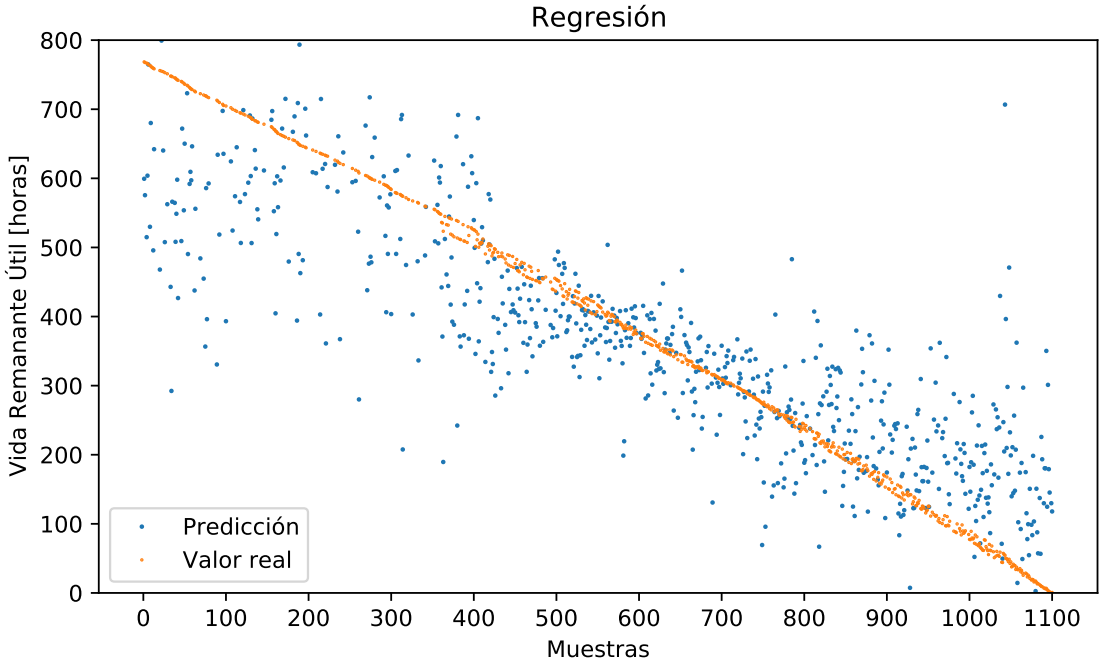

Figura A.33: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo de red neuronal con capas *Conv1D* utilizando la variable objetivo E_T entregando una respuesta por vuelo.

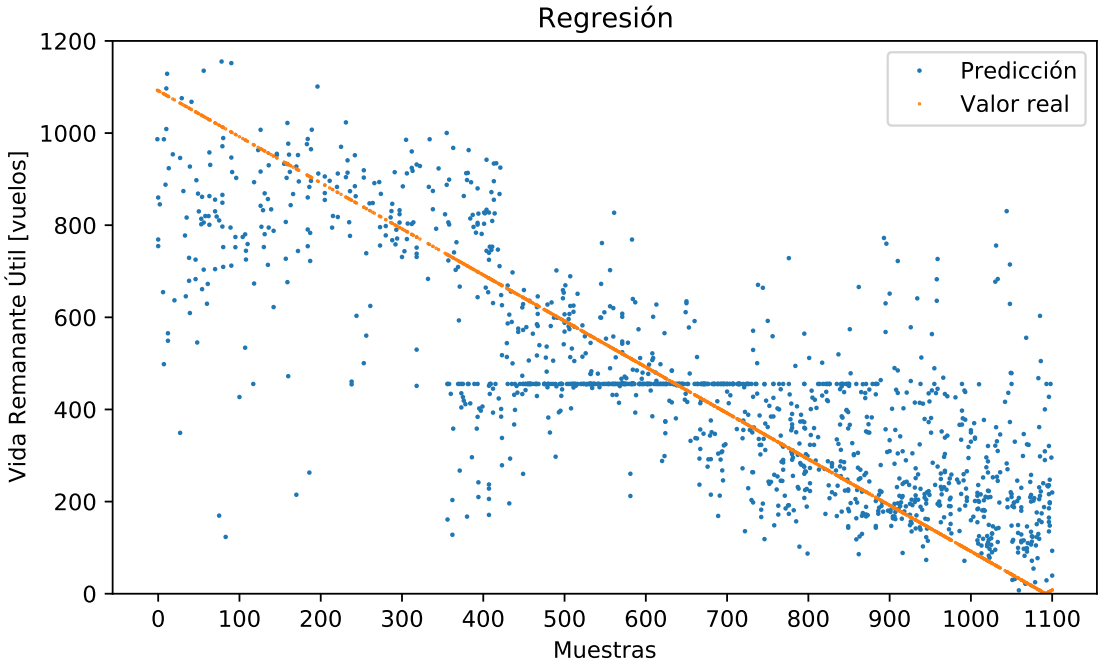

Figura A.34: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo de red neuronal con capas *Conv1D* utilizando la variable objetivo E_V.

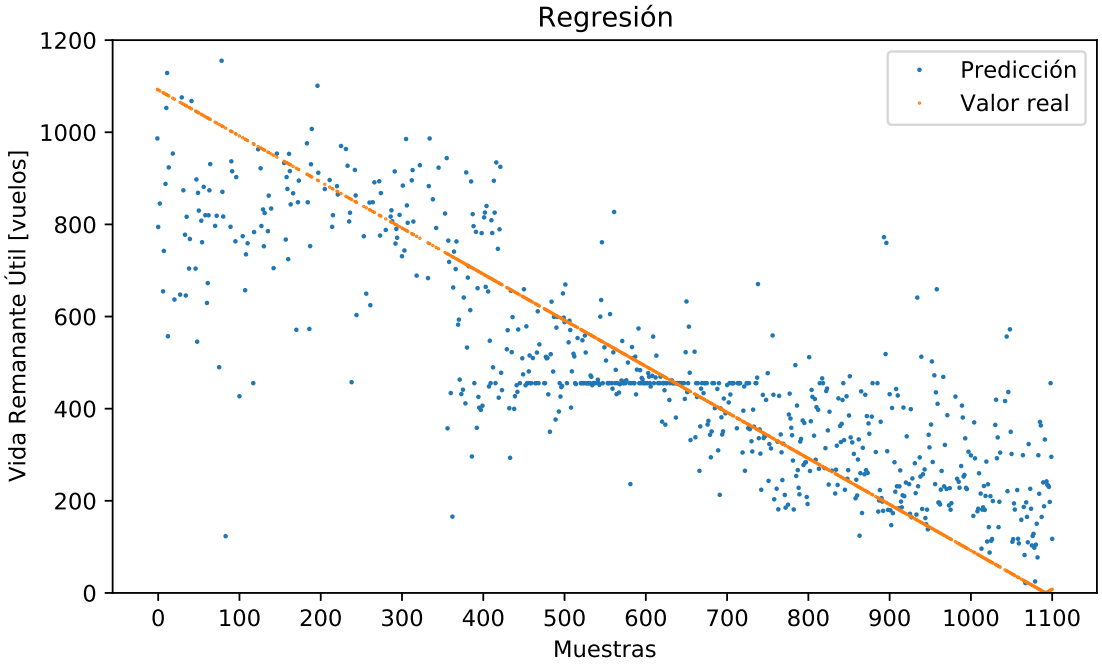

Figura A.35: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo de red neuronal con capas *Conv1D* utilizando la variable objetivo E_V entregando una respuesta por vuelo.

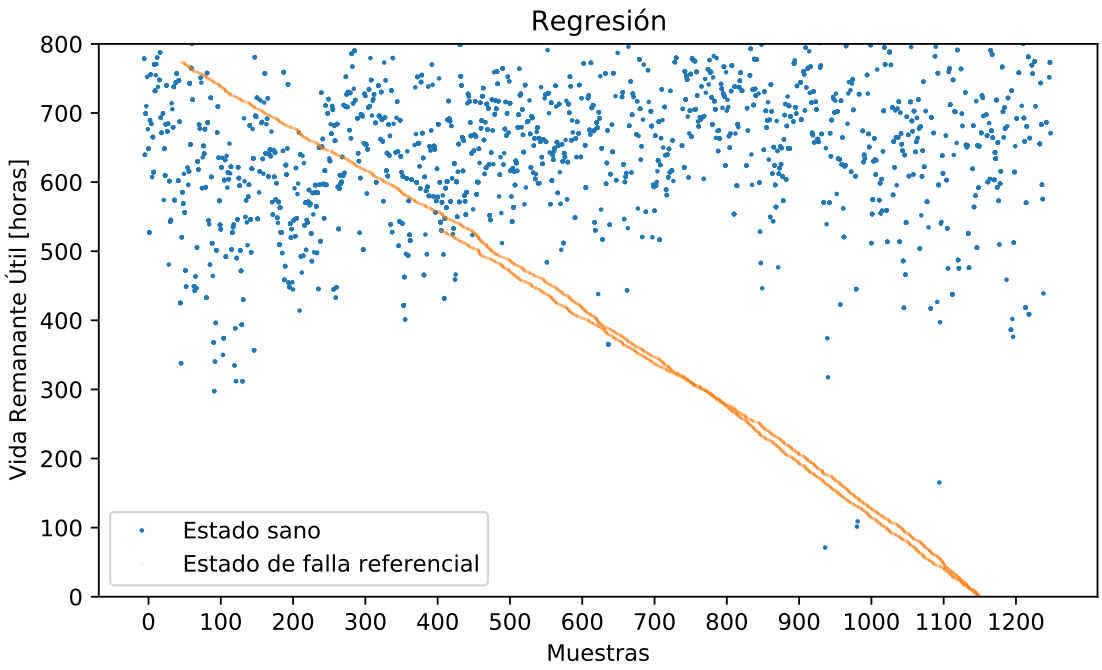

Figura A.36: Predicciones para datos sanos del modelo *Conv1D* entrenado con la etiqueta E_T.

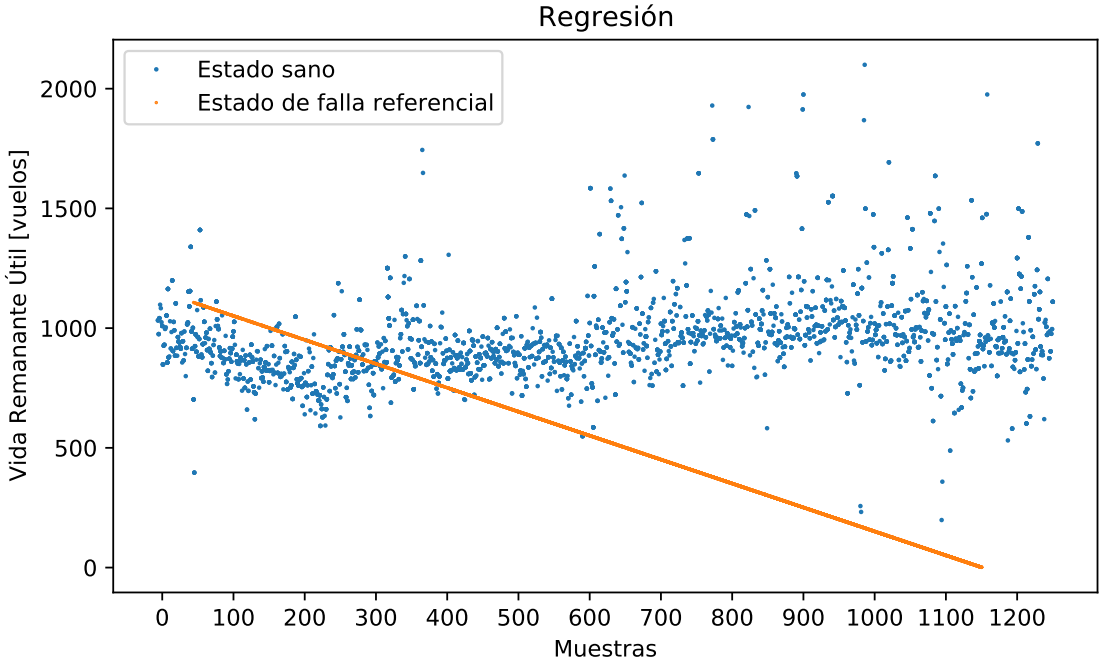

Figura A.37: Predicciones para datos sanos del modelo *Conv1D* entrenado con la etiqueta E_V.

A.11. Resultados de regresión por modelo *GRU*

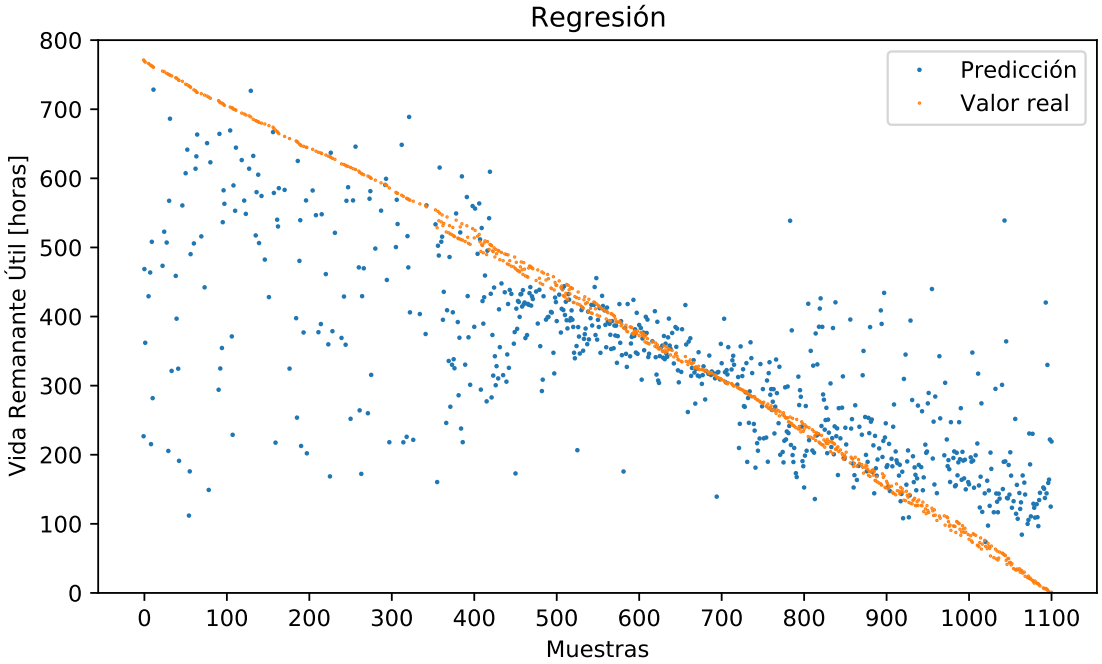

Figura A.38: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo de red neuronal con capas *GRU* utilizando la variable objetivo E_T entregando una respuesta por vuelo.

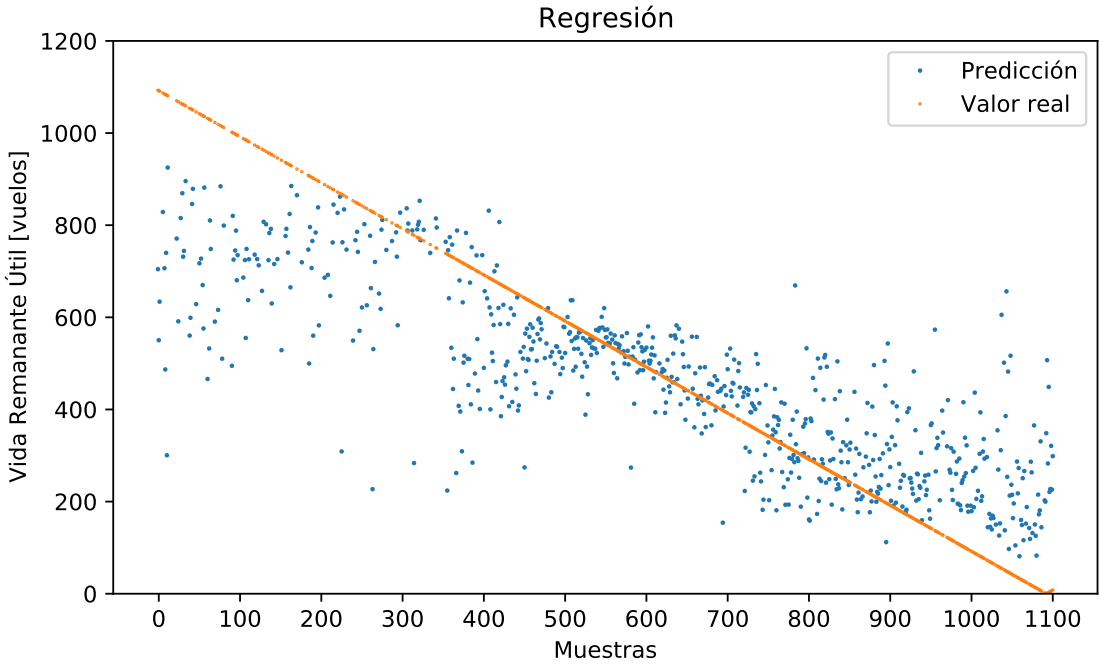

Figura A.39: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo de red neuronal con capas GRU utilizando la variable objetivo E_V entregando una respuesta por vuelo.

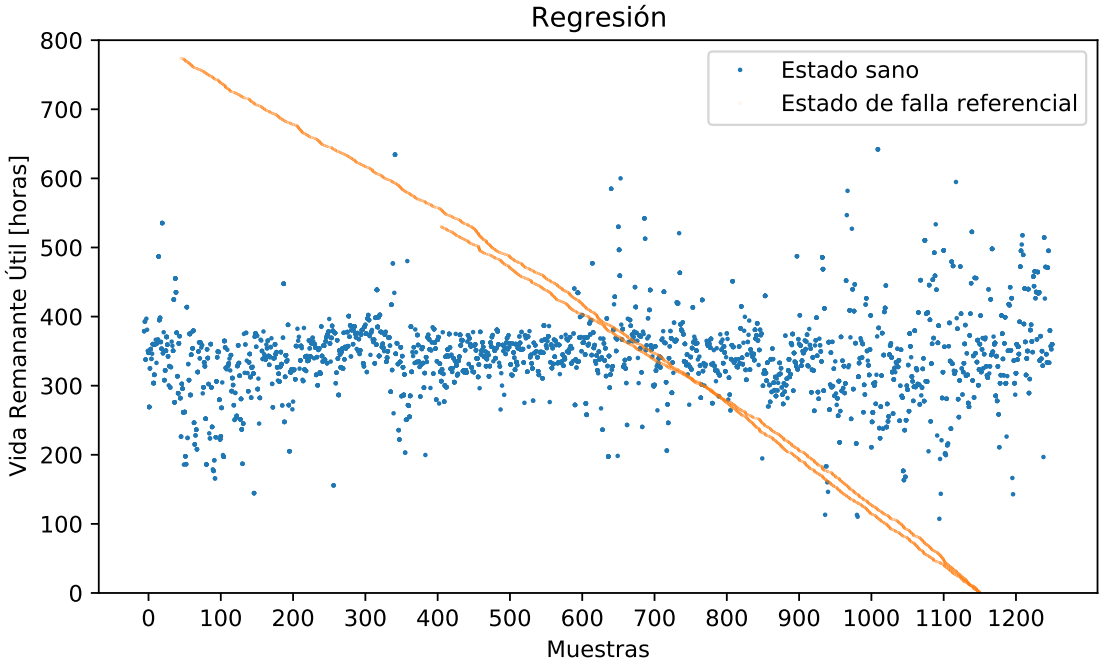

Figura A.40: Predicciones para datos sanos del modelo GRU entrenado con la etiqueta E_T.

A.12. Resultados de regresión por modelo neuronal *LSTM*

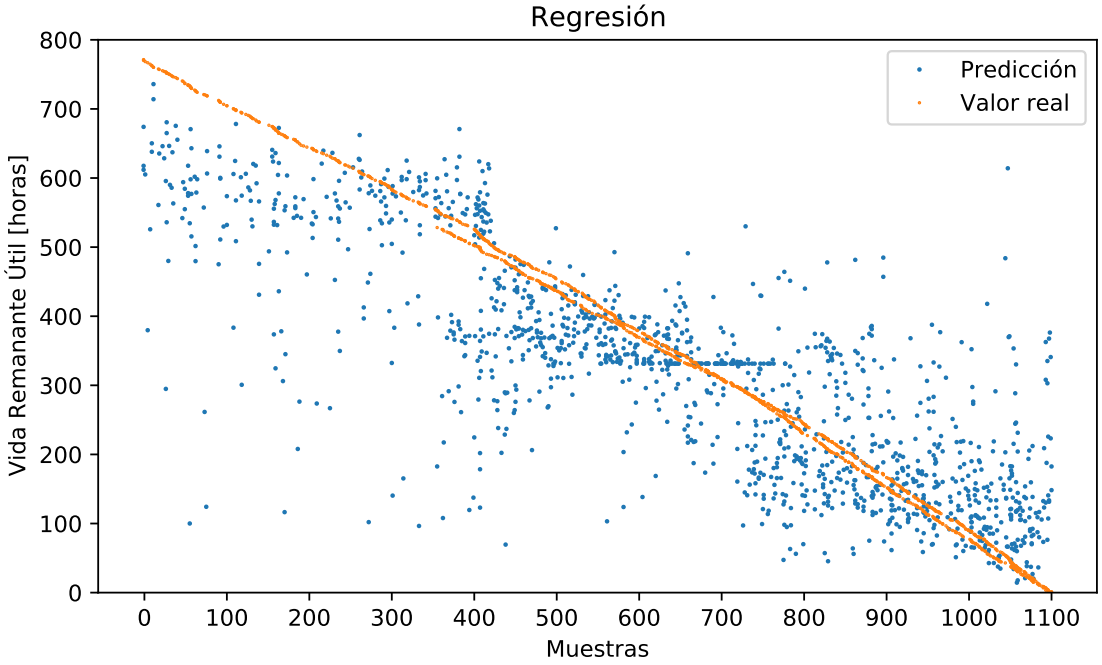

Figura A.41: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo de red neuronal con capas *LSTM* utilizando la variable objetivo E_T.

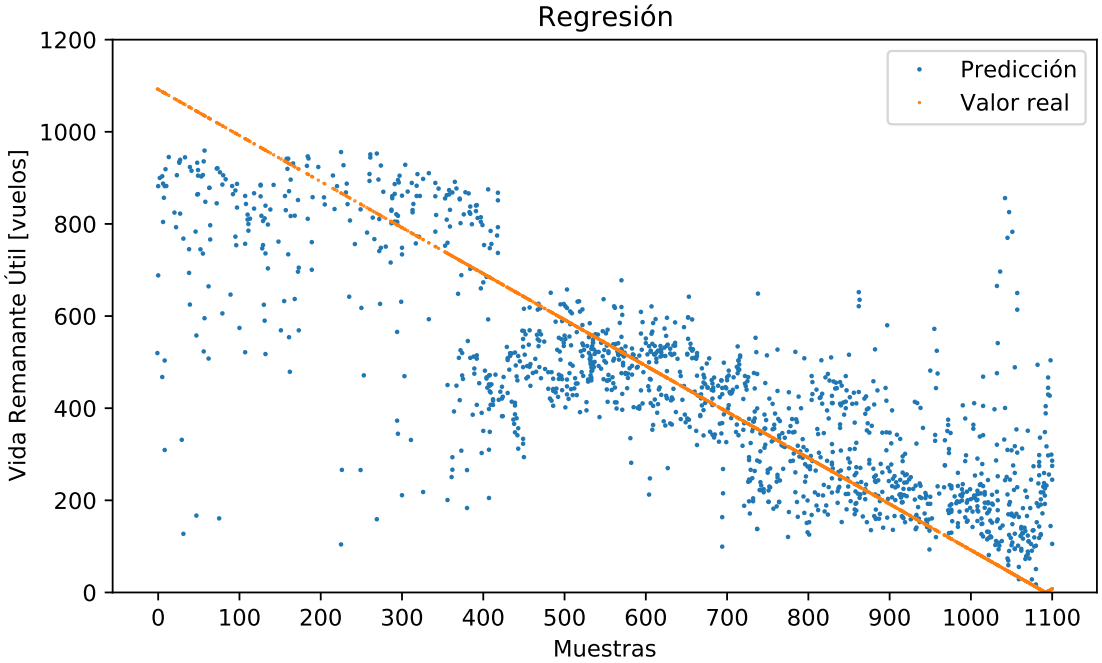

Figura A.42: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo de red neuronal con capas *LSTM* utilizando la variable objetivo E_V.

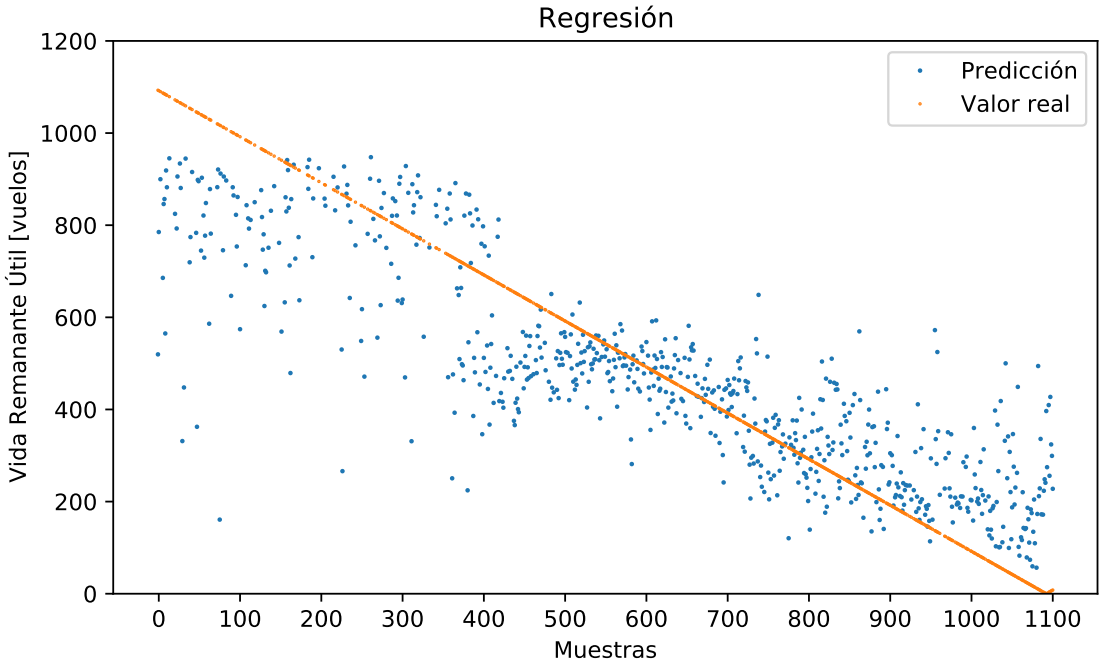

Figura A.43: Gráfico de valores reales y predicciones de la regresión obtenida para el modelo de red neuronal con capas *LSTM* utilizando la variable objetivo E_V entregando una respuesta por vuelo.

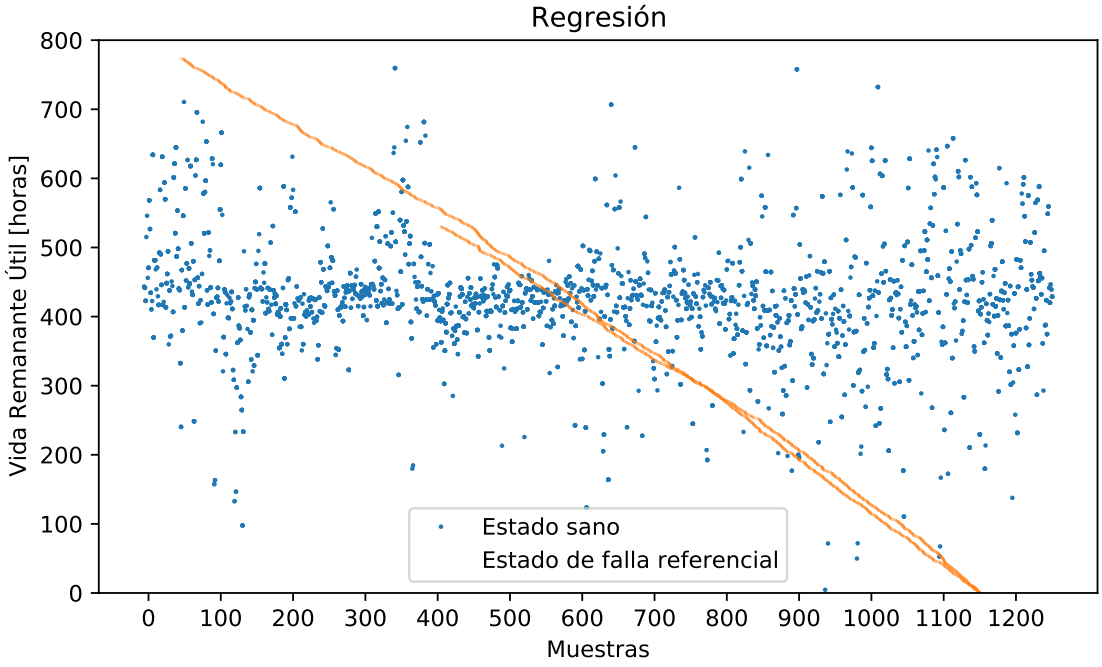

Figura A.44: Predicciones para datos sanos del modelo *LSTM* entrenado con la etiqueta E_T.

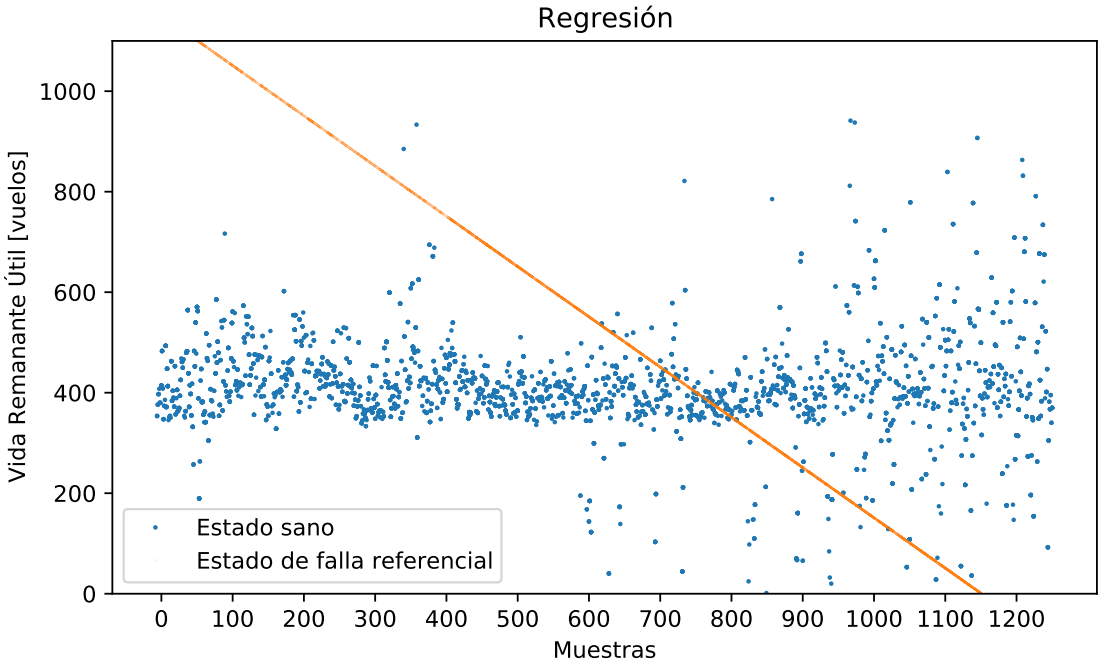

Figura A.45: Predicciones para datos sanos del modelo *LSTM* entrenado con la etiqueta E_V.

Apéndice B

Bibliografía

[1] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[2] Peter Flach. *Machine Learning - The Art and Science of Algorithms that Make Sense of Data*. Cambridge, 2012.

[3] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn & Tensorflow*. O'Reilly, 2017.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[6] Patrick Klein. and Ralph Bergmann. Generation of complex data for ai-based predictive maintenance research with a physical factory model. In *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO,*, pages 40–50. INSTICC, SciTePress, 2019.

[7] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[8] V. Madonna, P. Giangrande, and M. Galea. Electrical power generation in aircraft: Review, challenges, and opportunities. *IEEE Transactions on Transportation Electrification*, 4(3):646–659, 2018.

[9] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent in function space, 1999.

[10] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *CoRR*, ab-

s/1404.7828, 2014.

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, and Yoshua Bengio. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 1929.

[12] Leo Breiman Statistics and Leo Breiman. Random forests. In *Machine Learning*, pages 5–32, 2001.