Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

Desarrollo de Modelos de Machine Learning para la Predicción de Vuelos en el Espacio Aéreo Español

Autor: Leonardo Aguilar Valarezo
Tutor: Alfonso Mateos Caballero
Co-Tutor: Sergi Mas Pujol

Madrid, Junio 2024

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Máster
Máster Universitario en Inteligencia Artificial

Título: Desarrollo de Modelos de Machine Learning para la Predicción de Vuelos en el Espacio Aéreo Español

Junio 2024

Autor:	Leonardo Aguilar Valarezo
Tutor:	Alfonso Mateos Caballero
	Departamento de Inteligencia Artificial
	ETSI Informáticos
	Universidad Politécnica de Madrid
Co-Tutor:	Sergi Mas Pujol
	Centro de Referencia I+D+i ATM (CRIDA)

Agradecimientos

Quisiera expresar mi más sincero agradecimiento a todas las personas e instituciones que han contribuido al desarrollo de este Trabajo de Fin de Máster.

En primer lugar, deseo agradecer a mi tutor académico, Ph.D. Alfonso Mateos Caballero, de la Universidad Politécnica de Madrid, por su inestimable guía, apoyo y paciencia durante todo el proceso. Su conocimiento y experiencia han sido fundamentales para la realización de este proyecto.

Asimismo, extiendo mi agradecimiento a la empresa de investigación en gestión de tráfico aéreo CRIDA, por brindarme la oportunidad de desarrollar este trabajo en un entorno tan profesional y enriquecedor. En especial, quiero agradecer al Ph.D. Sergi Mas Pujol y a todo el equipo de CRIDA, por su constante apoyo, asesoramiento técnico y por compartir su experiencia y conocimientos.

A mis compañeros y colegas, tanto de la universidad como de CRIDA, gracias por su colaboración y por las discusiones constructivas que contribuyeron al avance de esta investigación, sobre todo mis compañeros de prácticas en CRIDA que hicieron mucho más llevadero y divertido todo este proceso.

A Katty, quien fue mi más grande apoyo, siempre animándome y ayudándome a conseguir todo lo que me propongo con su constante motivación y respaldo incondicional.

Finalmente, quiero agradecer a mi familia y amigos por su paciencia, apoyo incondicional y por creer en mí durante todo este proceso.

Resumen

La Gestión del Flujo y la Capacidad del Tráfico Aéreo (ATFCM) constituye una parte fundamental de la Gestión del Tráfico Aéreo (ATM). Este estudio se ha centrado en el desarrollo y análisis de diversos modelos de inteligencia artificial para la predicción del tráfico aéreo en el espacio aéreo español. La información derivada de estos modelos es esencial para la planificación, la definición de sectores y la gestión del espacio aéreo de manera eficiente y equitativa.

En primer lugar, se desarrolló un modelo de clasificación para determinar si un vuelo realiza un sobrevuelo por el espacio aéreo español. Donde, a partir de un proceso metodológico, se desarrolló un modelo *Random Forest* que alcanzó una puntuación *F1* de 0.97983. Validado en un entorno real, el modelo mostró mejoras significativas en comparación con la metodología actual, además de aportar información adicional valiosa al diferenciar cuáles vuelos realizan el sobrevuelo, en lugar de únicamente determinar el número de vuelos como lo realiza actualmente DELFOS.

Posteriormente, se abordó la predicción de rutas a diferentes niveles de detalle: Núcleos, *Traffic Volume* y *WayPoints*. Para los cuales, se propuso un modelo de red recurrente capaz de predecir el siguiente elemento de la ruta a partir de n pasos anteriores.

Los modelos finales desarrollados lograron un rendimiento relativamente bueno en los casos de paso por Núcleos y *Traffic Volume*, utilizando un enfoque de clasificación. Estos modelos lograron eliminar las inconsistencias entre ambas predicciones de rutas, alcanzando aproximadamente un 86 % y un 60 % de rutas completas predichas correctamente a través de Núcleos y *Traffic Volume*, respectivamente.

Estos resultados proporcionan información intrínsecamente valiosa, ya que, aunque pueden ser inferiores en términos del número de pasos por los elementos del espacio aéreo (la predicción actualmente realizada por DELFOS), la capacidad de predecir la ruta completa de cada vuelo representa una mejora significativa. La información detallada y el alto porcentaje de acierto de las rutas completas van más allá del enfoque actual, ofreciendo un valor añadido considerable al análisis y gestión del tráfico aéreo.

Abstract

Air Traffic Flow and Capacity Management (ATFCM) is a fundamental component of Air Traffic Management (ATM). This study has focused on the development and analysis of various artificial intelligence models for predicting air traffic in Spanish airspace. The information derived from these models is essential for planning, sector definition, and the efficient and equitable management of airspace.

Firstly, a classification model was developed to determine if a flight overflies Spanish airspace. Through a methodological process, a Random Forest model was developed, achieving an F1 score of 0.97983. Validated in a real-world environment, the model demonstrated significant improvements compared to the current methodology, in addition to providing valuable additional information by distinguishing which flights overfly, rather than merely determining the number of flights as the current DELFOS system does.

Subsequently, the study addressed route prediction at different levels of detail: ATC Unit, Traffic Volume, and WayPoints. For these, a recurrent neural network model was proposed, capable of predicting the next element of the route based on the previous n steps.

The final models developed achieved relatively good performance in the cases of passing through ATC Unit and Traffic Volume, using a classification approach. These models successfully eliminated inconsistencies between the two route predictions, correctly predicting approximately 86 % and 60 % of complete routes through ATC Unit and Traffic Volume, respectively.

These results provide intrinsically valuable information, as although they may be inferior in terms of the number of steps through the airspace elements (the prediction currently performed by DELFOS), the ability to predict the complete route of each flight represents a significant improvement. The detailed information and high accuracy of the complete route predictions go beyond the current approach, offering considerable added value to the analysis and management of air traffic.

Tabla de contenidos

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

La Gestión del Flujo y la Capacidad del Tráfico Aéreo (ATFCM) es una parte fundamental de la Gestión del Tráfico Aéreo (ATM), cuyo objetivo es optimizar el uso del espacio aéreo. Según la Organización de Aviación Civil Internacional (OACI), el ATM incluye la Gestión del Espacio Aéreo (ASM), el ATFCM y el Control del Tráfico Aéreo (ATC) [1]. Dado que el espacio aéreo es un recurso limitado y valioso, especialmente en períodos de alta demanda, la ASM es esencial para planificar, definir sectores y gestionar el espacio aéreo de manera eficiente y equitativa. Además, para evitar sobrecargas y utilizar el espacio aéreo de manera más eficiente, es necesario un mecanismo dinámico de gestión del flujo [2]. ENAIRE, como gestor de la navegación aérea en España, tiene bajo su control más de 2 millones de kilómetros cuadrados de espacio aéreo.

Uno de los principales desafíos del ATFCM es la capacidad limitada del espacio aéreo y las infraestructuras de los aeropuertos frente al creciente número de vuelos que se registra cada año; lo cual ha incidido en la continua búsqueda y desarrollo de tecnologías que mejoren la eficiencia de esta gestión, tanto desde la optimización del uso del espacio aéreo y los recursos aeroportuarios, como la seguridad, reducción de retrasos y minimización del impacto ambiental. Uno de los factores clave del ATFCM es la estimación precisa de la demanda futura de tráfico a partir de información prevista que abarca un relativamente grande horizonte de tiempo (hasta un año). La información disponible (horarios, aerolíneas, origen, destino, etc.) y su nivel de incertidumbre asociado difieren entre las distintas fases de planificación del ATFCM, lo que genera diferencias cualitativas entre los tipos de pronósticos que son factibles en cada horizonte temporal. Las metodologías utilizadas para pronosticar el tráfico aéreo futuro difieren significativamente según los propósitos de la prognosis. En este contexto aparece la inteligencia artificial (AI), desde donde, si bien se han realizado abundantes investigaciones sobre la predicción táctica, es decir, durante el día de las operaciones, la predicción en la fase pretáctica, cuando hay pocos o ningún plan de vuelo disponible, ha recibido mucha menos atención.

En el contexto de la predicción pretáctica podemos mencionar varios trabajos con enfoques muy diferentes, pues se trata de un escenario ambiguo debido a la gran incertidumbre que presenta el tratar de predecir la saturación del espacio aéreo con un margen de tiempo muy grande. Para el análisis del flujo de tráfico aéreo en zonas bastante específicas el cuánto, con qué rapidez y dónde crecerá el transporte aéreo depende de una serie de factores, algunos relacionados con la economía y otros vin-

culados con la demografía y la evolución socioeconómica. El objetivo principal de la prognosis se centra en comprender cómo los diferentes factores contribuirán a explicar el futuro del transporte aéreo. En este contexto, estudios como [3] busca resaltar los factores más importantes que subyacen al crecimiento del Servicios de Tránsito Aéreo estadounidense (ATS), de tal forma que permita identificar los impulsores clave de la evolución del ATS de EE. UU. Para lo cual desarrolla un modelo que incluye el uso de técnicas de minería de datos para modelar la evolución de los ATS; la consideración de un conjunto amplio de variables explicativas; y el modelado explícito de la distribución de la demanda de pasajeros por pares de ciudades entre itinerarios. Otra forma de abordar este problema lo encontramos en [4], el cual busca predecir el número de vuelos que entraran en las diferentes rutas aéreas del espacio aéreo. En este trabajo utilizaron el concepto más básico de series temporales, basándose únicamente en las características de la fecha y hora para estimar la cantidad de tráfico. Propusieron dos modelos de predicción aplicando una máquina de vectores de soporte (SVR) y una red *Long short-term memory* (LSTM); cuyos resultados mostraron que el predictor basado en SVR puede predecir con precisión el flujo debido a su ventaja en el manejo de datos con tendencias reconocibles, mientras que el predictor basado en LSTM puede proporcionar un rendimiento mejorado, y se infiere que el predictor basado en LSTM puede abordar mejor los puntos anormales de la variación del flujo. Entre los trabajos más completos sobre esta problemática se encuentra [5], en el cual busca predecir la sobre carga en el espacio aéreo a nivel de rutas típicas utilizadas a partir de información esencial de un vuelo. En su trabajo realiza una segmentación según diferentes características de los vuelos (tipo de negocio, aerolínea, etc.) y diseña diferentes algoritmos para cada uno de los segmentos. Para la predicción de la ruta elegida por cada vuelo planteó 2 modelos: *Multinomial Logistic Regression* y *Decision Tree Regressor* para predecir la probabilidad de elección de una ruta. En el trabajo se muestran un gran número de experimentos destacando resultados con un error menor al 5 por ciento, y siendo el modelo de árbol de decisión el que obtuvo por lo general mejores resultados, sobre todo en aquellos experimentos que consideran un mayor número de rutas.

El presente trabajo nace de una de las aristas del proyecto DELFOS (criDa aEriaL FOrecaSt), el cual es un proyecto interno de CRIDA A.E.I. cuyo objetivo es el de proporcionar previsiones de tráfico aéreo instrumental a futuro y centradas en el espacio aéreo español. El proyecto DELFOS nació en el contexto de la pandemia del COVID-19. En esos momentos, la situación del tráfico global y nacional era inestable y debido a las diferentes políticas de confinamiento, dicho tráfico sufrió un declive considerable. Esto último provoca que el escenario de tráfico deje de ser un escenario estacional y, por ende, los métodos que se usaban para la predicción de tráfico se volvieron ineficientes.

Este escenario mostró la urgencia por metodologías más eficientes y adaptables a los cambios de escenario en la gestión de tráfico aéreo. Actualmente, DELFOS busca proveer predicciones estratégicas de la demanda hasta un año vista, incluyendo su distribución horaria y espacial. Dicha prognosis está basada en métodos estadísticos para calcular el número de vuelos en varios escenarios como:

- Una previsión vuelo a vuelo, incluyendo la aerolínea, aeropuerto de origen y de destino y los tiempos de despegue y aterrizaje.
- Una previsión por unidades de servicio de tránsito aéreo (ATS).

- Una previsión por aeropuertos.
- Una previsión a nivel de vuelo dando la sucesión de *Traffic Volume* más probable.

El presente estudio busca abordar las predicciones realizadas por DELFOS desde la inteligencia artificial, con el fin de analizar, por un lado, una comparativa de la precisión respecto a los resultados que actualmente se obtienen con su metodología estadística, y por otro, el valor de la información específica que puede brindar una predicción individual de cada uno de los vuelos, en lugar de únicamente el conteo de los vuelos en los diferentes escenarios.

En este estudio, se abordará desde dos enfoques principales:

- **Predicción de sobrevuelo en el espacio aéreo español:** Predicción que determine si un vuelo realizará un sobrevuelo por espacio aéreo español o no. Este enfoque consistirá en un análisis de clasificadores binarios para determinar el modelo con mejor rendimiento en la predicción de si un vuelo sobrevolará el espacio aéreo español.
- **Predicción de la trayectoria de vuelos en el espacio aéreo español:** este aspecto incluye una alta complejidad y se abordará a través de diferentes niveles de granularidad de la ruta de los vuelos que transitan, total o parcialmente, por el espacio aéreo español. Los niveles abordados en este trabajo fueron:
  - Plan de vuelo: Es una descripción detallada de la ruta que seguirá un vuelo. Este plan incluye una secuencia precisa de *waypoints*, radio ayudas (*radio-aids*) y aeródromos, los cuales constituyen puntos geográficos en el espacio aéreo.
  - *Traffic Volume*: Es una segmentación del espacio aéreo español basada en criterios como la densidad de tráfico en cada zona geográfica. Para el espacio aéreo español, en este trabajo se definieron 56 *Traffic Volume*.
  - Núcleo elemental: Representa la menor segmentación del espacio aéreo español. Cada núcleo está compuesto por uno o más *traffic volume* y, por lo general, estos núcleos no se combinan para formar zonas geográficas más grandes. Para el espacio aéreo español, en este trabajo se definieron 10 núcleos elementales.

  Si bien, el plan de vuelo es sumamente detallado y a patir de este se puede extrapolar a los demás niveles, para la predicción nos enfocaremos de forma inversa, ya que el espacio de búsqueda de los elementos del plan de vuelo es sumamente grande (miles). De esta forma, cada nivel brindará información y restringirá el espacio de búsqueda del siguiente.

  Debido a la complejidad de la predicción de secuencias, esta tarea no puede ser abordada por métodos convencionales de *Machine Learning*. En este trabajo planteamos el uso de Redes Neuronales Recurrentes (RNN) debido a su naturaleza y gran rendimiento en la predicción de secuenias [6].

1.1. Objetivos

1. **Objetivos Generales**

   - Analizar el comportamiento de herramientas de inteligencia artificial, en el contexto de la gestión de la capacidad y el flujo del tránsito aéreo (ATFCM) sobre la problemática de la predicción de la futura saturación de tráfico aéreo.
   - Aplicar modelos avanzados de *Machine/Deep Learning* como herramientas de apoyo en la ATFCM en el espacio aéreo español, con la finalidad de mejorar los resultados obtenidos con la metodología actual de DELFOS, basada en procesos estadísticos tradicionales, de tal forma que se logre optimizar la predicción de saturación de tráfico aéreo con el fin de reducir su impacto en el espacio aéreo gestionado por ENAIRE mediante la gestión temprana de los recursos disponibles.

2. **Objetivos Específicos**

   - Mejorar la precisión de la predicción realizada por DELFOS con la metodología estadística actual referente al número de vuelos por zona de espacio aéreo español en un determinado periodo de tiempo.
   - Desarrollar una arquitectura de Machine Learning para la detección de sobrevuelos en el espacio aéreo español, utilizando datos de vuelos globales proporcionados por la Guía Oficial de Aviación (OAG).
   - Desarrollo de tres arquitecturas de *Deep Learning*, una por cada nivel de granularidad (núcleos elementales, *Traffic Volume* y *waypoints*), para la predicción de la ruta seguida por un vuelo que realiza, completo o parcialmente, su recorrido por espacio aéreo español.

El presente trabajo se estructura en varios capítulos que abordan de manera sistemática y detallada los aspectos fundamentales del estudio realizado. En el Capítulo 2, titulado “Herramientas, Algoritmos y Métricas“, se describen las principales herramientas de software utilizadas, las bases teóricas de los algoritmos implementados y las métricas empleadas para evaluar el rendimiento de los modelos. El Capítulo 3, “Datos“, se centra en la descripción del conjunto de datos utilizado, incluyendo su origen, características y el procesamiento necesario aplicado. En el Capítulo 4, “Metodología“, se expone detalladamente el enfoque metodológico adoptado para el desarrollo del trabajo, incluyendo los procedimientos de construcción, validación, y optimización de los diferentes modelos desarrollados. El Capítulo 5, “Resultados“, presenta los hallazgos obtenidos a partir de la implementación de los modelos, acompañado de un análisis exhaustivo de los mismos. Finalmente, el Capítulo 6, “Conclusiones y Trabajos Futuros“, ofrece un resumen de las conclusiones derivadas del estudio y propone posibles direcciones para futuras investigaciones a partir de lo aquí desarrollado.

Capítulo 2

Herramientas, Algorítmos y Métricas

En el presente capítulo se detallarán las herramientas, algoritmos y métricas empleadas en el desarrollo del trabajo. La selección de estos elementos es fundamental para garantizar la robustez y eficiencia de los modelos predictivos desarrollados. En primer lugar, se describirán las herramientas de software y bibliotecas utilizadas, las cuales proporcionan la infraestructura necesaria para el procesamiento y análisis de datos. Posteriormente, se explicarán los algoritmos implementados abordando sus fundamentos teóricos. Finalmente, se presentarán las métricas de evaluación empleadas para medir el rendimiento de los modelos, destacando la importancia de cada métrica en la interpretación de los resultados y la toma de decisiones. Este capítulo proporciona una visión integral de los componentes técnicos que sustentan nuestro trabajo, permitiendo comprender la base metodológica sobre la cual se han construido y evaluado los modelos de inteligencia artificial.

2.1. Herramientas utilizadas

Todo el desarrollo del proyecto se llevó a cabo utilizando el lenguaje de programación *Python*, reconocido por su versatilidad y amplio ecosistema de bibliotecas especializadas en análisis de datos y aprendizaje automático. Las librerías utilizadas las podemos clasificar según se usen para el procesamiento de los datos y para la construcción de los modelos de IA.

1. **Herramientas para procesamiento de datos**

   Se utilizaron librerías básicas para el procesamiento de los datos disponibles, si bien estos datos son los básicos de un vuelo, mucha de su información debía ser tratada según su naturaleza en un contexto real, la cual suele ser errática debido a la captura de los datos por parte de los radares o situaciones de vuelo extraordinarias.

   - *Pyodbc:* es una biblioteca de Python que nos sirve para tener la integración de la comunicación con bases de datos de una manera sencilla [7]. Su uso en este proyecto se centró en el uso para llamadas de datos por medio de consultas SQL al SQL Server de CRIDA.

- *Pandas:* es una biblioteca de manipulación y análisis de datos de código abierto, rápida, potente, flexible y fácil de usar [8]. El hecho de que los datos provengan de un ambiente estructurado (tablas SQL) hace ideal el uso de Pandas con su estructura Dataframe.

- *Numpy:* esta librería da soporte para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas [9]. Esta flexibilidad permite la estructuración de los datos para buscar el máximo rendimiento en los modelos planteados.

2. **Herramientas para construcción de modelos**

   Se utilizaron librerías dedicadas a la inteligencia artificial para la construcción de los diferentes modelos. Estas librerías incluyen funciones que construyen la arquitectura de los diferentes algoritmos planteados y su gestión.

   - *Scikit learn:* es una biblioteca de aprendizaje automático gratuita y de código abierto [10] que incluye funciones que abarcan desde la preparación de los datos hasta el análisis de los resultados de los modelos de aprendizaje automático.

   - *Tensorflow:* es una biblioteca de código abierto para aprendizaje automático a través de un rango de tareas, y desarrollado por Google [11]. Es ampliamente utilizado para aprendizaje profundo, con la construcción de redes neuronales a partir de la concatenación de capas con diferentes arquitecturas.

   - *Mlflow:* es una plataforma de código abierto, diseñada específicamente para ayudar a manejar las complejidades del proceso de aprendizaje automático [12]. Se centra en el ciclo de vida completo de los proyectos de aprendizaje automático, garantizando que cada fase sea manejable, rastreable y reproducible.

2.2. Algoritmos

La inteligencia artificial está en auge y se continúa explorando nuevas formas de obtener predicciones cada vez más precisas. Aunque es posible diseñar algoritmos que produzcan mejores resultados que otros, estos suelen ser efectivos en contextos específicos más que desde una perspectiva objetiva general. Por ello, en el presente trabajo se evaluarán diferentes algoritmos con el fin de identificar aquel que proporcione el mejor rendimiento posible para nuestro contexto y datos específicos.

1. **Regresión Lineal**

   Es un método de aprendizaje automático supervisado que, como su nombre lo indica, predice valores numéricos (regresión) a partir de variables con las que tenga una relación lineal. Como se explica en [13], sirve para modelar la relación entre una variable dependiente (predicción) y una o más variables independientes (predictoras). Este modelo busca estimar una función de la forma:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

donde y es la variable dependiente o predicción del modelo; $[x_1, x_2, \cdots, x_n]$ son las n variables independientes; $[\beta_1, \beta_2, \cdots, \beta_n]$ son los coeficientes que definen la proporción de cambio producida en la predicción y β_0 es una constante que define el valor de la predicción cuando las variables predictoras se establecen en cero.

La estimación de dicha ecuación se realiza por lo general con el método de "mínimos cuadrados ordinarios" (MCO), el cual consiste en, dado un conjunto de datos (características y su predicción), minimizar las distancias al cuadrado entre los valores estimados y los valores reales de la variable dependiente.

2. **Regresión Logística**

   La regresión logística funciona de manera muy similar a la regresión lineal, pero con una variable de respuesta binomial, por tanto, predice la probabilidad de que una instancia pertenezca a una de las dos categorías. En [14], se formaliza esta definición, definiendo a la regresión logística como la predicción del valor esperado de Y, dado un conjunto de características $[x_1, x_2, \cdots, x_n]$, es decir $E(Y|x_1, x_2, \cdots, x_n)$, el cual, sea $E(Y|x_1, x_2, \cdots, x_n) = \pi(x)$ lo modela de la forma:

   $$\pi(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n)}}$$

   y dado que $\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$ es una ecuación lineal, la frontera de decisión en la regresión logística es un hiperplano lineal en el espacio de las características.

3. **K-Vecinos Más Cercanos (k-NN)**

   El algoritmo de k-vecinos más cercanos (k-NN) es un método de aprendizaje supervisado utilizado tanto para clasificación como para regresión. En [15], se explica como el algoritmo k-NN no requiere una fase de entrenamiento explícita, en cambio, almacena todos los ejemplos de entrenamiento, los cuales serán usados durante la fase de predicción.

   En tareas de clasificación, el algoritmo encuentra los k puntos de datos más cercanos en el conjunto de entrenamiento (usualmente usando una medida de distancia como la distancia euclidiana). La clase más común entre estos k vecinos se asigna a la nueva instancia. En tareas de regresión, en cambio, encuentra los k vecinos más cercanos y realiza una operación sobre sus valores (como la media), asignando este resultado a la nueva instancia.

   Este algoritmo, en comparación con los otros presentados en esta sección, destaca por su simplicidad en la construcción. No obstante, presenta una complejidad notable en la elección de la métrica de distancia a utilizar y, en el caso de la regresión, en la operación a realizar. Uno de sus principales inconvenientes radica en la ineficiencia computacional cuando se enfrenta a grandes volúmenes

de datos, debido a la necesidad de realizar búsquedas exhaustivas para encontrar las instancias más cercanas.

4. **Gaussian Naive Bayes**

   Es un algoritmo probabilístico específico del algoritmo Naive Bayes, propuesto en [16], el cual se basa en el teorema de Bayes y la asunción de independencia entre las características predictoras dada la variable clase, lo cual simplifica el cálculo de $P(X|C)$ como el producto de las probabilidades individuales de las características. El Gaussian Naive Bayes asume, además, que las características continuas siguen una distribución normal, facilitando la estimación de una distribución de probabilidad para las características. Sin embargo, esto suele presentar limitaciones debido a las múltiples asunciones que se realizan al construir este modelo.

5. **Árbol de decisión**

   El algoritmo de árbol de decisión es una técnica de aprendizaje supervisado utilizada tanto para clasificación como para regresión. En [17] se explica como un árbol de decisión utiliza un modelo en forma de árbol en el cual cada nodo representa una partición sobre una característica (por ejemplo $x_1 < 10$ o $x_1 = True$). El objetivo en la construcción del modelo consiste en encontrar las mejores particiones posibles de las características, siendo el algoritmo C4.5 para clasificación y CART para regresión, unos de los más usados y conocidos. De esta forma, el modelo de árbol, para clasificación, restringe el espacio de valores de cada una de las características asociadas a una clase; y para regresión se considera el valor medio de la variable objetivo de las muestras en cada una de las particiones y se aplica una transformación a estos valores.

6. **Adaptive Boosting (AdaBoost)**

   Es un algoritmo de la familia de boosting, los cuales buscan mejorar el rendimiento de los modelos débiles (modelo que tiene un rendimiento ligeramente mejor que el azar) combinándolos secuencialmente, donde cada nuevo modelo intenta corregir los errores de los modelos anteriores. AdaBoost fue propuesto en [18] cuyo concepto clave es, a partir de modelos débiles entrenados secuencialmente, asignar pesos a cada instancia del conjunto de datos. Inicialmente, todos los pesos son iguales, y en el proceso de entrenamiento de cada clasificador débil, los pesos se ajustan aumentándolos para las instancias mal clasificadas y reduciéndolos para las correctamente clasificadas. Esto permite que los clasificadores subsecuentes se enfoquen en las instancias más difíciles.

7. **Gradient boosting**

   Es otro algoritmo de la familia de boosting. Fue propuesto en [19], el cual combina la idea de boosting con la optimización de gradientes para crear un modelo predictivo fuerte a partir de una colección de modelos débiles, generalmente árboles de decisión, cuyo entrenamiento se realiza secuencialmente y utiliza un proceso de optimización de gradientes donde, en cada iteración, el modelo se

ajusta para corregir los errores residuales de los modelos anteriores en la dirección del gradiente descendente de la función de pérdida.

8. **Random Forest**

   El algoritmo Random Forest es un método de aprendizaje supervisado utilizado tanto para clasificación como para regresión. Fue propuesto en [20] como una extensión de las ideas de bagging (Bootstrap Aggregating), desarrollada como una alternativa al boosting. La idea principal consiste en crear múltiples muestras aleatorias del conjunto de datos original (con reemplazo) y entrenar un árbol de decisión en cada muestra, a diferencia del boosting donde cada modelo débil se entrenaba con todos los datos.

9. **Red Neuronal Recurrente**

   Una red neuronal es un modelo de IA inspirado en el funcionamiento del cerebro humano, compuesto por capas de nodos (neuronas) que procesan información a través de conexiones ponderadas, las cuales se utilizan para identificar patrones complejos y realizar tareas de clasificación y predicción a partir de ellos.

   Una red neuronal recurrente (RNN) es un tipo de red neuronal artificial que está diseñada para aprender patrones secuenciales o variables en el tiempo. Estas fueron introducidas en [21], donde se propuso un tipo de red neuronal recurrente conocida como red de Hopfield, una de las primeras arquitecturas de redes neuronales que introdujo la idea de los estados recurrentes y la capacidad de las redes para mantener información a lo largo del tiempo.

   A diferencia de las redes neuronales tradicionales, las RNNs tienen la capacidad de mantener información en "memoria" a lo largo de secuencias de datos, lo que las hace especialmente útiles para tareas donde el contexto temporal o secuencial es importante, donde en cada paso temporal, la misma red y los mismos parámetros se aplica a cada elemento de la secuencia. Sin embargo, esta naturaleza dificulta su entrenamiento, especialmente en problemas con dependencias temporales largas, debido a su naturaleza iterativa no lineal. Un pequeño cambio en un punto del proceso puede tener efectos muy grandes en iteraciones posteriores, lo que se conoce como "efecto mariposa". Esto significa que la derivada de la función de pérdida en un momento dado puede ser extremadamente sensible a las activaciones ocultas de momentos anteriores, haciendo que la función de pérdida sea muy sensible a pequeños cambios [22].

   Se han propuesto varias variantes de RNN para solucionar algunos problemas como los mencionados, entre estas variantes tenemos las Redes Long Short-Term Memory y Las Redes Bidireccionales.

   a) **Redes Long Short-Term Memory (LSTM)**

      Estas redes fueron diseñadas para abordar los problemas de desvanecimiento y explosión del gradiente, lo que permite a las RNN aprender dependencias a largo plazo de manera más efectiva. Fue propuesta en [23], donde lo define como un algoritmo eficiente basado en gradientes para una arquitectura que impone un flujo de errores constante a través de estados

internos de unidades especiales, pues siempre que el cálculo del gradiente se trunque en ciertos puntos específicos de la arquitectura esto no afecta al flujo de errores a largo plazo. Estas unidades especiales consisten en puertas multiplicativas que aprenden a abrir y cerrar el acceso al flujo constante de errores.

b) **Redes Neuronales Recurrentes Bidireccionales (BiRNN)**

Fueron propuestas en [24], bajo el enfoque de procesar secuencias de datos en ambas direcciones, hacia adelante y hacia atrás. La motivación detrás de este enfoque es aprovechar el contexto completo de la secuencia para mejorar la precisión y el rendimiento del modelo en tareas secuenciales, como el reconocimiento de voz y el procesamiento del lenguaje natural. La idea general es dividir las neuronas de estado de una RNN regular en una parte que es responsable de la dirección del tiempo positiva (estados directos) y una parte de las entradas del estado hacia atrás en el tiempo negativo que no se conocen.

Esta arquitectura de redes ha tenido un impacto significativo en diversas aplicaciones de aprendizaje automático, mejorando la precisión y el rendimiento de los modelos en tareas secuenciales complejas.

2.3. Métricas de evaluación

En el desarrollo y evaluación de modelos de inteligencia artificial, la precisión y fiabilidad de los modelos son de vital importancia. Las métricas de evaluación son herramientas fundamentales que permiten medir el desempeño de los modelos, identificar sus fortalezas y debilidades, y compararlos de manera objetiva. Esta sección se centrará en una variedad de métricas de evaluación ampliamente utilizadas en el campo de la IA y que serán utilizadas a lo largo de este proyecto, incluyendo precisión (accuracy), la puntuación F1, error absoluto medio (MAE) y error cuadrático medio (MSE). Además, se presentará una nueva métrica diseñada específicamente para comparar secuencias, la cual hemos desarrollado para abordar ciertas limitaciones observadas en las métricas tradicionales.

1. **Accurracy**

   Es una métrica básica y general que ha sido utilizada desde los inicios de la estadística y el aprendizaje automático. Es la proporción de predicciones correctas realizadas por el modelo en relación con el total de predicciones. Se utiliza comúnmente en clasificación binaria y multiclase [25]. El cálculo de esta métrica se realiza mediante la fórmula:

   $$Accurracy = \frac{TP + TN}{TP + TN + FP + FN},$$

   donde:

   - TP = Verdaderos Positivos
   - TN = Verdaderos Negativos

- FP = Falsos Positivos
- FN = Falsos Negativos

2. **Puntuación *F1***

   Es la media armónica de la precisión (precision) y la exhaustividad (recall), proporcionando un equilibrio entre ambas métricas, por lo cual es especialmente útil en casos de desequilibrio de clases. La precisión mide la exactitud de las predicciones positivas del modelo, es decir, la proporción de verdaderos positivos entre el total de predicciones positivas. La exhaustividad, o recall, evalúa la capacidad del modelo para identificar todas las instancias positivas, calculándose como la proporción de verdaderos positivos sobre el total de instancias realmente positivas. El cálculo de esta métrica se lo realiza mediante la fórmula:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall},$$

   donde:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

3. **Error Absoluto Medio (MAE)**

   Mide el promedio de los errores absolutos entre las predicciones del modelo y los valores reales, proporcionando una interpretación directa del error [25]. El cálculo de esta métrica se lo realiza mediante la fórmula:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

   Donde:

   - y_i = Valor real
   - $\hat{y}_i$ = Valor predicho
   - n = Número total de observaciones

4. **Error Cuadrático Medio (MSE)**

   Mide el promedio de los cuadrados de los errores. De esta forma, penaliza más severamente los errores grandes, destacando grandes discrepancias entre las predicciones y los valores reales [25]. EL cálculo de esta métrica se realiza mediante la fórmula:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

donde:

- y_i = Valor real
- $\hat{y}_i$ = Valor predicho
- n = Número total de observaciones

5. **Métrica: Comparación de rutas**

   Para evaluar el rendimiento de nuestros modelos de predicción de ruta se creó una métrica de penalización, que logre comparar las rutas predichas por el modelo con las reales según los criterios de interés en nuestro contexto de trayectorias aéreas. Esta métrica abordará 4 tipos diferentes de penalizaciones y poseerá un peso para cada penalización, de tal forma que, si se quiere dar más importancia a un tipo de penalización, se le pueda establecer un valor mayor:

   - *Penalización por omisión de elemento*: Se calcula como el producto del valor de la penalización por el número de elementos que omitió la predicción comparada con la ruta real.
   - *Penalización por elemento extra*: Se calcula como el producto del valor de la penalización por el número de elementos extra de la predicción comparado con la ruta real.
   - *Penalización por cambio de elemento*: Esta penalización se asigna cuando, una vez descartados los elementos omitidos y extra, se predijo un elemento diferente con el de la ruta real. Se calcula como el producto del valor de la penalización por la distancia relativa entre el elemento real y el elemento predicho.
   - *Penalización por cambio de posición de elemento*: Esta penalización se asigna cuando, una vez descartados los elementos omitidos y extra, el orden de la ruta predicha no coincide con la real. Se calcula como el producto del valor de la penalización por la diferencia de la posición del elemento predicha comparada con la posición que debió tener.

   Podemos ver ejemplos del funcionamiento de esta métrica si establecemos el valor de cada penalización en 1 en la Tabla 2.1.

Tabla 2.1: Ejemplos de penalización de métrica

Ruta Real	**Ruta Predicha**	**Penalizaciones**	**Penalización Total**
A, B, E, O, P	A, B, C, E, O, P	Penalización por elemento extra: la ruta predicha anadió el elemento C	1
A, B, E, O, P	A, B, O, P	Penalización por elemento omitido: la ruta predicha omitió el elemento E	1
A, B, E, O, P	A, B, C, O, P	Penalización por cambio de elemento: la ruta predicha cambio el elemento E por C	1
A, B, E, O, P	A, B, O, E, P	Penalización por cambio de posición de elemento: la ruta predicha ubicó al elemento O en la posición 3, cuando en la ruta real está en la posición 4 y viceversa.	2
A, C, D	B, A, D	Penalización por cambio de posición de elemento: la ruta predicha ubicó al elemento A en la posición 2, cuando en la ruta real está en la posición 1. Penalización por cambio de elemento: la ruta predijo el elemento B en lugar del elemento C.	2

Capítulo 3

Datos

Para la realización de este trabajo se contó con un volumen de datos sumamente grandecompuesto por varios millones de registros anuales. Estos datos fueron provistos por CRIDA y contenían la información básica de cada uno de los vuelos. Cabe destacar que la información disponible del detalle de los vuelos varía ligeramente entre el enfoque de sobrevuelos y el enfoque de rutas, a pesar de que en la práctica, los vuelos clasificados como sobrevuelos también serán considerados para la predicción de su ruta a través del espacio aéreo español.

Por razones de eficiencia, los conjuntos de datos utilizados en cada enfoque contienen registros de vuelos significativamente diferentes. El conjunto de datos para el enfoque de sobrevuelos se obtuvo de los datos de vuelos mundiales proporcionados por la Guía Oficial de Aviación (OAG), una organización que ofrece información y datos sobre el tráfico aéreo global. Posteriormente, se aplicaron procesos de filtrado definidos por CRIDA. Estos procesos incluyen la exclusión de vuelos nacionales e internacionales españoles, ya que su paso por el espacio aéreo español está prácticamente garantizado, salvo excepciones extraordinarias. Además, se aplicaron criterios para determinar la imposibilidad de que ciertos vuelos, como los nacionales de otros países, crucen el espacio aéreo español.

Por otro lado, el conjunto de datos para la predicción de rutas incluye los registros de vuelos nacionales e internacionales españoles, además de los sobrevuelos. Estos sobrevuelos consisten, en la práctica, en aquellos vuelos que nuestro modelo clasificaría como tales para la predicción de rutas.

Estas diferencias evidencian la variabilidad en el espacio de valores de cada una de las variables en los dos conjuntos de datos y la necesidad de un tratamiento específico para cada conjunto. A continuación, se detallará de forma separada el procesamiento aplicado en cada caso.

3.1. Datos del modelo de sobrevuelos

El conjunto de datos para determinar los sobrevuelos por espacio aéreo español era significativamente grande. CRIDA puso a nuestra disposición un conjunto de datos con 16 161 297 registros. Podemos ver la distribución de estos datos en la Tabla 3.1, donde observamos la ausencia de registros de los años 2020 y 2021, que debido a las irregularidades causadas por la pandemia de COVID-19 se descartó su inclusión.

Tabla 3.1: Datos disponibles para modelo de sobrevuelos

Año	Meses	Datos
2018	Diciembre	407,610
2019	Enero - Diciembre	6,559,787
2022	Enero - Diciembre	5,501,834
2023	Enero - Agosto	3,692,066
Total		16,161,297

Cada uno de los registros del conjunto de datos consta del detalle básico de los vuelos, que representan la información programada de las aerolíneas con un margen de tiempo de hasta 1 año. Estas variables disponibles son:

- Flight_ID (numérica): identificador único de cada vuelo.
- Origin (categórica): código ICAO del aeropuerto de origen del vuelo.
- Destination (categórica): código ICAO del aeropuerto de destino del vuelo.
- Airline (categórica): código ICAO de la aerolínea a la que pertenece el vuelo.
- Aircraft_Type (categórica): tipo de aeronave que realizó el vuelo.
- Date (Fecha): fecha en que se realizó el vuelo con formado yyyy-mm-dd.
- ATOT_Hour_UTC (numérica): tiempo real de salida del vuelo en estándar UTC.
- ATA_Hour_UTC (numérica): tiempo real de llegada del vuelo en estándar UTC (esta hora puede corresponder al día siguiente).
- Route_Length__NM (numérica): longitud del recorrido del vuelo en millas náuticas.
- crossSpain (booleana): variable objetivo, que indica si el vuelo cruzó o no espacio aéreo español.

Las variables categóricas de entrada (Origin, Destination, Airline y Aircraft_Type) requirieron un análisis detallado debido a la naturaleza de sus posibles valores. Una característica distintiva de estas variables es su elevada cardinalidad, como se observa en la Tabla 3.2. El espacio de posibles valores para cada una de estas variables comprende incluso miles de valores distintos. No obstante, esta gran cantidad no es restrictiva, ya que en un contexto real pueden surgir nuevos valores para cada una de estas variables debido a nuevas políticas, expansiones, o incluso situaciones extraordinarias que suelen ocurrir en el ámbito de la aviación.

Tabla 3.2: Espacio de valores de las variables categóricas

Variable	Cardinalidad
Origin	3,374
Destination	3,261
Airline	3,969
Aircraft_Type	634

A ello debemos sumarle la distribución que poseen cada una de ellas, pues tomando como ejemplo la variable “Origin“ (las demás variables presentan un comportamiento

similar), de la cual podemos ver su distribución en la Figura 3.1, donde observamos los 10 valores, tanto con mayor como con menor frecuencia, encontramos que la diferencia presente es sumamente grande, existiendo una gran cantidad de valores que aparecen únicamente una vez en los datos y, por el contrario, los 232 valores con mayor frecuencia constituyen el 90 % de los registros.

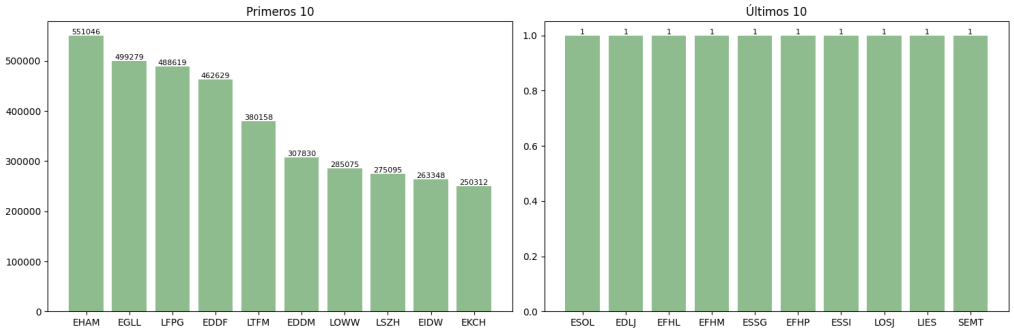

Figura 3.1: Frecuencia de los valores de la variable Origin

Al contar con tantos valores, resulta complicado tratar con este tipo de variables, la mejor solución fue mediante la codificación numérica de cada uno de sus valores, para ello fue necesaria la creación de una base de datos con estas codificaciones, con las funciones adecuadas para generar nuevas en caso de que el modelo, en un entorno real, encuentre valores no vistos, y por tanto, no codificados anteriormente. Es importante destacar que las variables “Origin“ y “Destination“ deben compartir una misma base de datos, pues los valores de ambos son aeropuertos y los mismos pueden aparecer en ambas variables.

Respecto al análisis de la variable objetivo, a pesar de que la cantidad de los datos era sumamente grande, la distribución de los que cruzan o no espacio aéreo español estaba sumamente desbalanceada. En la Figura 3.2 podemos observar esta situación, donde consta que únicamente un 7.62 % de los datos cruzan espacio aéreo español.

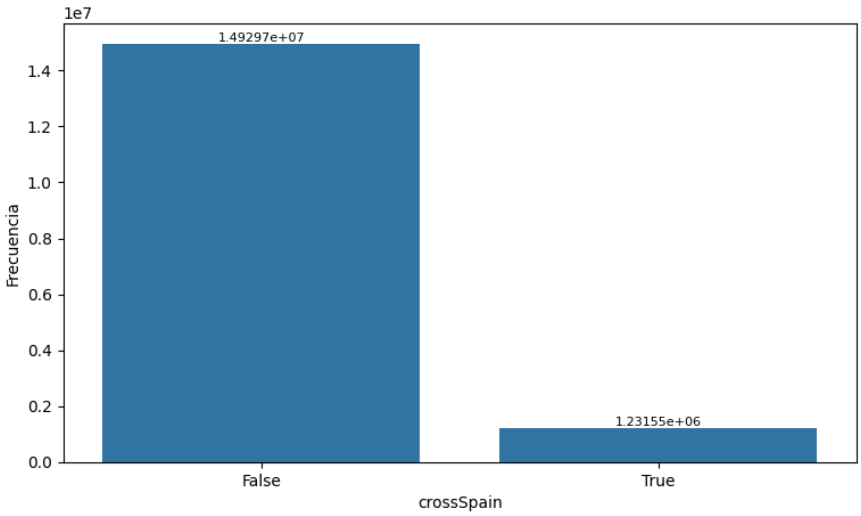

Figura 3.2: Frecuencia de vuelos que cruzan espacio aéreo español

Esta circunstancia, a primera vista, genera una problemática a la tarea de clasificación, sin embargo, en las siguientes secciones, veremos que no fue requerido realizar tratamientos especiales para solucionar este desbalance en nuestro escenario.

Una de las tareas en las que se puso especial énfasis en este trabajo fue la búsqueda de nueva información significativa para los modelos de inteligencia artificial. En el contexto de la construcción del modelo de sobrevuelos, se buscó información relacionada con los aeropuertos que pudiera aportar valor. Esta información fue proporcionada por CRIDA, quienes disponían de una base de datos de aeropuertos a nivel mundial. Esta contenía diversa información, sin embargo, contaba con campos vacíos en algunos aeropuertos y además, no constaban muchos aeropuertos (97) que estaban presentes el conjunto de datos de este modelo. Una de las alternativas era establecer valores por defecto a las características de estos aeropuertos; sin embargo, consideramos que asignar un valor por defecto igual para todos estos aeropuertos perjudicaría al modelo en la búsqueda de los patrones que caracterizan a aquellos vuelos que realizan un sobrevuelo que los contengan.

Debido a esta problemática, las variables que se decidieron añadir al conjunto de datos fueron:

- Region_dep (categórica): Región del aeropuerto de origen del vuelo. Está representada por la primera letra de su código ICAO
- Region_des (categórica): Región del aeropuerto de destino del vuelo. Está representada por la primera letra de su código ICAO
- Country_dep (categórica): País o subregión del aeropuerto de origen del vuelo. Está representada por las 2 primeras letras de su código ICAO, o en algunos casos con la primera.
- Country_des (categórica): País o subregión del aeropuerto de destino del vuelo. Está representada por las 2 primeras letras de su código ICAO, o en algunos casos con la primera.
- Latitude_dep (numérica): Coordenada de latitud del aeropuerto de origen
- Latitude_des (numérica): Coordenada de latitud del aeropuerto de destino
- Longitude_dep (numérica): Coordenada de longitud del aeropuerto de origen
- Longitude_des (numérica): Coordenada de longitud del aeropuerto de destino

Donde, las variables de región y país se pueden sacar directamente del código del aeropuerto, y las coordenadas de latitud y longitud son las únicas que se extraen a partir de la base de información de los aeropuertos con la que cuenta CRIDA. Además, para tratar adecuadamente a aquellos aeropuertos que no cuentan con esta información se diseñó un modelo independiente para estimar su localización (los detalles de este modelo se presentan en la Sección 4.2).

En lo concerniente a las demás variables, las numéricas no requieren de ningún tratamiento especial, sin embargo, la fecha del vuelo no corresponde a un formato que sea legible por un modelo de IA, por tanto, es necesario extraer la información que ésta provee en nuevas variables numéricas. Entonces, de la variable “Date“ se propuso separar su información en las siguientes nuevas variables:

- Day: número del día del mes presente en la fecha del vuelo

- Month: número del mes presente en la fecha del vuelo
- WeekDay: número del día de la semana correspondiente a la fecha del vuelo

Se descartó el uso del año debido a que, en términos de la predicción, siempre se realizará esta desde la fecha actual hasta un año en adelante, por tanto, el valor del año será uno que no se ha visto anteriormente en el entrenamiento.

El comportamiento de las variables que representan la fecha del vuelo, a grandes rasgos, siguen un comportamiento esperable, como vemos en la Figura 3.3, existe una mayor cantidad de vuelos en los meses de festividades, como lo son los de verano y diciembre.

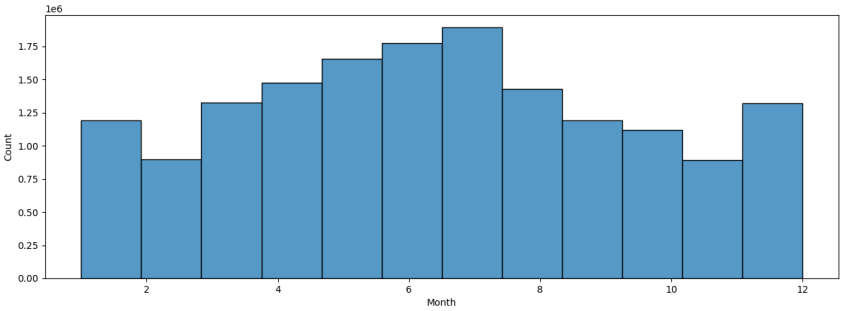

Figura 3.3: Frecuencia de vuelos en cada mes del año

La única variable relacionada a la fecha que presenta un comportamiento inesperado es “WeekDay“, pues se esperaría una mayor frecuencia en los días cercanos al fin de semana, sin embargo, en la Figura 3.4 podemos observar cómo no existe tal tendencia, y cada día de la semana presenta un número de vuelos parecido.

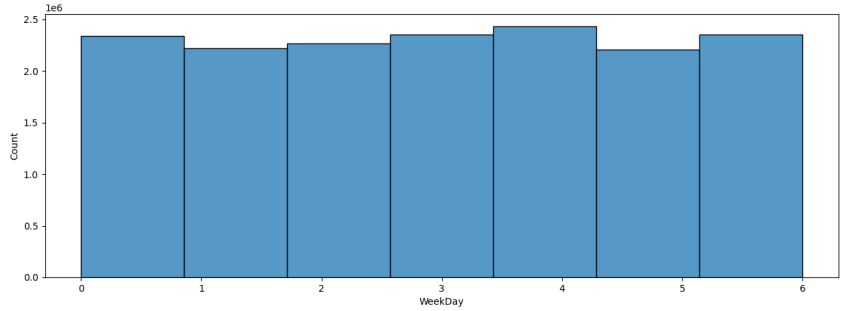

Figura 3.4: Frecuencia de vuelos en cada día de la semana

Finalmente, en la Figura 3.5, se presenta la matriz de correlación que ilustra las relaciones existentes entre las variables numéricas entre sí y con la variable objetivo. Es importante destacar que, entre las variables numéricas, no se observa una correlación significativa, con la excepción de la hora de despegue y aterrizaje del vuelo, cuya relación es esperada dada la naturaleza temporal de estas variables. Adicionalmente, resulta notable que ninguna de las variables numéricas muestra una correlación considerable con la variable objetivo, lo cual sugiere una gran independencia entre estas. Esta independencia indica que las variables numéricas consideradas no influyen de

manera directa en la variable objetivo, proporcionando una valiosa perspectiva sobre la estructura y comportamiento de los datos analizados.

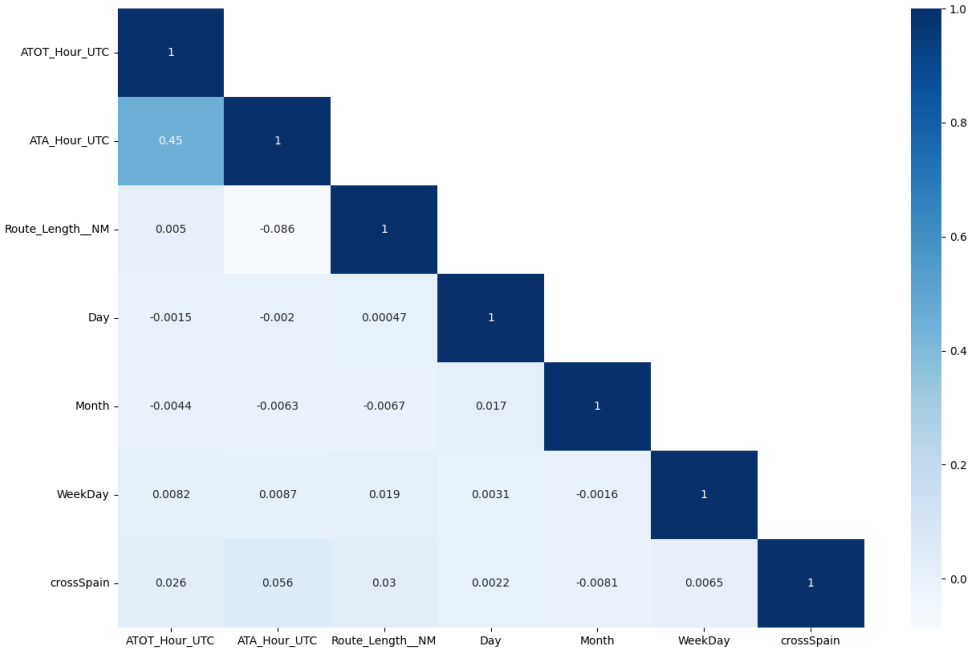

Figura 3.5: Correlación de variables numéricas y objetivo

3.2. Datos de los modelos de rutas

El escenario de los datos de rutas presenta un gran aumento del tamaño del conjunto de datos, esto es debido a que existe un registro por cada paso en la ruta, conservando el detalle del vuelo en cada uno de ellos. Es por esto que se seleccionó únicamente los datos a partir del año 2023 para que pueda ser plausible un entrenamiento con las capacidades computacionales disponibles.

En este escenario de paso por rutas, la información de entrada varía ligeramente con la vista para los sobrevuelos, pues no se disponen de algunas variables vistas anteriormente. Además, si bien el detalle del vuelo es el mismo para cada nivel de granularidad de la ruta, la información propia de la ruta, es decir, nuestras variables objetivo, mientras a más bajo nivel nos encontramos, presenta información adicional de detalle de la trayectoria.

Las variables del detalle de vuelo para cada nivel son:

- CFMUflightkey (numérica): identificador único de cada vuelo
- routeType (categórica): tipo de ruta que realiza el vuelo respecto a España. Puede ser Nacional, Internacional o Sobrevuelo
- adep (categórica): código ICAO del aeropuerto de origen del vuelo

- ades (categórica): código ICAO del aeropuerto de destino del vuelo
- airline (categórica): código ICAO de la aerolínea a la que pertenece el vuelo
- aircraftType (categórica): tipo de aeronave que realizó el vuelo
- processDateReference (Fecha): fecha en que se realizó el vuelo con formado yyyy-mm-dd

Observamos que en este escenario no están presentes varias características que sí se encontraban en el modelo de sobrevuelos, y además, ahora contamos con una característica que anteriormente no se consideraba. Las variables que ahora no están presentes son “ATOT_Hour_UTC“, “ATA_Hour_UTC“ y “Route_Length__NM“. Lo cual se puede entender debido a que, por ejemplo, la longitud de la ruta reflejaba una estimación de toda la distancia que recorrería el vuelo, pero sin darnos ninguna idea de cuanto de su recorrido abarcaría espacio aéreo español. En lo que respecta a las variables de los horarios del vuelo, en los niveles de TV y WP se considera información aún más precisa.

Las variables que se conservan del modelo anterior conservan su comportamiento en este escenario donde ya se incluyen los vuelos nacionales e internacionales españoles. Podemos ver este comportamiento reflejado en la variable “adep“, que como observamos en la Figura 3.6, el comportamiento es similar al observado en los datos de sobrevuelos y la codificación se realizará de la misma manera para todas las variables categóricas, creando una base para la codificación de cada una, siendo esta diferente a la del modelo de sobrevuelos.

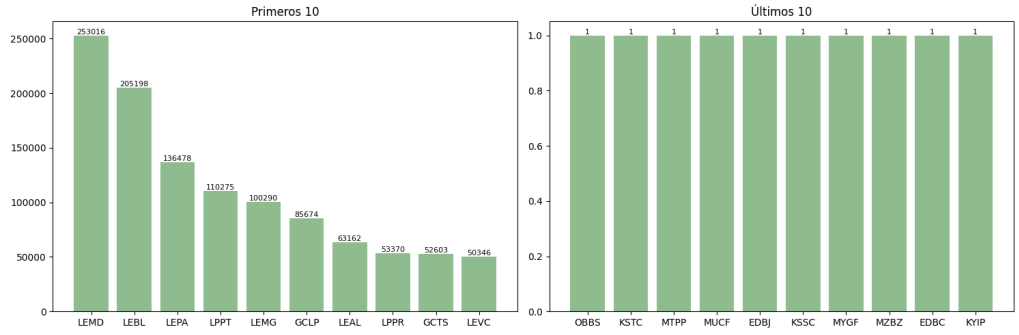

Figura 3.6: Frecuencia de valores de la variable adep

Siguiendo la línea de acción del modelo de sobrevuelos, se buscó para este escenario una nueva ampliación de la información de entrada. Sin embargo, no se utilizó las mismas características adicionales que en el caso anterior. Si bien conservamos la adición de las variables de latitud y longitud, la región y país no la consideramos pertinente; esto es debido a que, al encontrarnos en un espacio de valores más reducido que en el escenario de sobrevuelos (ver Tabla 3.3) son muy pocos los aeropuertos presentes en los datos que no constan en la base de información de los aeropuertos. Debido a esto se consideró añadir a nuestros datos la variable de subregión y para aquellos pocos aeropuertos que no disponen de información, ubicamos el valor por defecto “NULL“, y para la latitud y longitud un valor de 0 en ambas variables.

Por tanto, las variables extra consideradas para este escenario son:

- region_dep (categórica): Región en que se encuentra el aeropuerto de origen del vuelo. Según el país esta puede ser una provincia o municipio.
- region_des (categórica): Región en que se encuentra el aeropuerto de destino del vuelo. Según el país esta puede ser una provincia o municipio.
- latitude_dep (numérica): Coordenada de latitud del aeropuerto de origen.
- latitude_des (numérica): Coordenada de latitud del aeropuerto de destino.
- longitude_dep (numérica): Coordenada de longitud del aeropuerto de origen.
- longitude_des (numérica): Coordenada de longitud del aeropuerto de destino.

Tabla 3.3: Espacio de valores de las variables categóricas

Variable	Cardinalidad
adep	1 508
ades	1 513
airline	2 175
aircraftType	369

Con respecto a la información relacionada con la ruta, esta se presenta con mayor cantidad de información a medida que bajamos de nivel. Para el nivel de paso por núcleos, se cuenta con 2 variables:

- atcunit (categórica): Núcleo por el que registra paso el vuelo
- datefilterCriteria (DateTime): Fecha y hora registrada de su paso

Para el nivel de paso por *Traffic Volume*, se presenta información con una mayor precisión, tanto por constituir cada *Traffic Volume* un espacio geográfico menor que el de un núcleo, como por mayor detalle en los tiempos de paso. Las variables disponibles son:

- trafficVolumeCode (categórica): TV por el que registra paso el vuelo
- EntryTime (DateTime): fecha y hora del ingreso en el TV
- ExitTime (DateTime): fecha y hora de salida del TV

Finalmente, para el nivel de plan de vuelo encontramos variables que describen de forma precisa la ruta seguida por un vuelo, estas son:

- NameWP (categórica): WP, radio ayudas o aeródromo por el que registra paso el vuelo.
- ETO (DateTime): Fecha y hora en la que registra su paso
- FL (numérica): Nivel de vuelo al que se encuentra la aeronave en cada paso
- Latitude (numérica): Latitud en el punto de paso del vuelo
- Longitude (numérica): Longitud en el punto de paso del vuelo

Una de las principales problemáticas encontradas en los conjuntos de datos correspondientes a los diferentes niveles de las rutas es la existencia de inconsistencias entre estas. Por ejemplo, hay rutas por TV que, al escalar cada uno de sus elementos a núcleos, no corresponden exactamente con las rutas registradas en el conjunto de

datos de rutas por núcleos. Este tipo de inconsistencias está directamente relacionado con el nivel de detalle de la ruta en los distintos niveles, ya que, a niveles más altos, la ruta de interés sigue criterios más generales.

En el caso de la conversión de rutas de TV a núcleos, la causa más común de estas discrepancias radica en que, para la ruta por núcleos, se ignoran entradas en otros núcleos que hayan durado solo unos pocos minutos. Esto se debe principalmente a las acciones tomadas a partir de la comunicación entre el piloto y el controlador aéreo. En nuestros datos, el 81.36 % de las rutas por TV coinciden con las registradas en el conjunto de rutas por núcleos.

Las discrepancias entre la ruta de WP escalada a TV y la ruta real de TV son aún más significativas, ya que solo el 44.34 % de las rutas son consistentes entre sí. Esto se debe, además de a situaciones similares a las observadas en la conversión de TV a núcleos, a definiciones y reglas específicas en la gestión de tráfico aéreo español. Entre estos escenarios tenemos que algunos WP no están relacionados con ningún TV y, en las fronteras, aunque algunos WP pertenecen al espacio aéreo español, la gestión en estos puntos es llevada a cabo por otros países, como Francia.

3.2.1. Elementos constitutivos de las rutas

Como observamos anteriormente, cada uno de los niveles de ruta presenta una variable categórica, la cual representa los elementos de cada nivel por los que un vuelo puede pasar en su ruta. El primer nivel, correspondiente al de núcleos, cuenta con 10 diferentes núcleos elementales de interés:

- Palma TACC
- Barcelona Ruta Este
- Barcelona TMA
- Barcelona Ruta Oeste
- Levante TACC
- Madrid Ruta 2
- Madrid Ruta 1
- Madrid TMA
- Sevilla ACC
- Canarias ACC

A este nivel, es posible comprender de manera más precisa la distribución geográfica de cada uno de los núcleos. Por ejemplo, en la Figura 3.7, se observa el área geográfica correspondiente al espacio aéreo español en la península ibérica, destacándose en ella el núcleo denominado “Madrid Ruta 1“.

El segundo nivel corresponde a *Traffic Volume* (TV), para los cuales se definieron 56 según el enfoque adoptado por CRIDA. En la Tabla 3.4 observamos el número de TV presente en cada uno de los núcleos elementales.

El último nivel, correspondiente al plan de vuelo, cuyos elementos de forma general llamaremos a todos *WayPoints* (WPs). Los WPs tiene un espacio de posibles valores

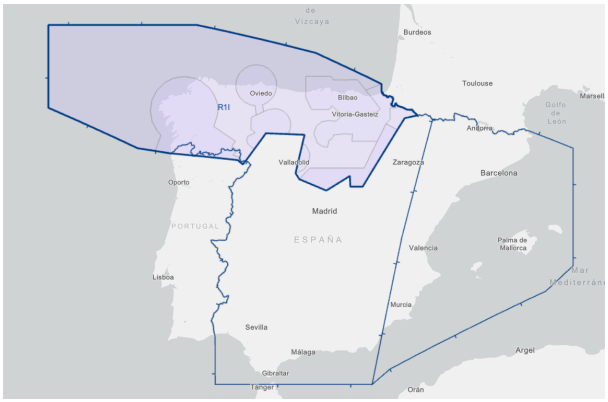

Figura 3.7: Núcleo: Madrid Ruta 1

Tabla 3.4: Número de TV por cada Núcleo

Núcleo	**Número de TV**
Barcelona Ruta Este	7
Barcelona Ruta Oeste	7
Barcelona TMA	1
Canarias ACC	9
Levante TACC	1
Madrid Ruta 1	10
Madrid Ruta 2	9
Madrid TMA	1
Palma TACC	1
Sevilla ACC	10

mucho más grande, pues representa plan de vuelo desde el máximo detalle posible. En la Tabla 3.5, observamos tanto, el número de WP que existen en los cuatro TV que contienen el mayor, como en los cuatro que contienen el menor número de WPs. Debemos considerar que un WP puede pertenecer a varios TV, debido a que varios TV pueden abarcar un punto específico, pero desde diferentes alturas.

Es importante destacar que, si bien los núcleos y TV corresponden a espacios geográficos claramente definidos, los WP son puntos específicos determinados por coordenadas de latitud y longitud.

Tabla 3.5: Número de WP por TV

TV	Número de WP
LEBLALB	1117
LEMDALL	947
GCCCTM3	577
LECSSEVL	458
...	...
LECBMNU	15
GCCCRCW	8
GCCCRCE	6
GCCCRWW	2

3.2.2. Transformación

Debido a la estructura original del conjunto de datos (un registro por cada paso) y la naturaleza de los datos de entrada necesaria para un modelo recurrente, se abordó una estructura de datos en que, por cada uno de los pasos de la ruta, le corresponda la información de los pasos anteriores y sus características del vuelo.

Definimos un grupo de n variables $f_1, \cdots, f_n$ que representan el detalle de un vuelo. Definimos además las m variables objetivo $q_1, \cdots, q_m$ correspondientes a la información de cada uno de los p pasos que contiene la ruta. Por conveniencia además, establecemos el vector $0_1, \cdots, 0_m$ para indicar cuando la ruta no contempla ningún elemento (lo cual sirve para representar los pasos anteriores al primer elemento de la ruta). Finalmente definimos r pasos previos a considerar como datos de entrada para el siguiente elemento de la ruta.

De esta forma, el primer registro de la ruta se transforma en una matriz de entrada como se ve en la Ecuación 3.1, donde las primeras n columnas corresponden a la información básica del vuelo, y las subsiguientes m a las variables objetivo constitutivas de la ruta (al ser el primer registro se utiliza el vector por defecto de elementos); y la salida como en la Ecuación 3.2, con un super índice que indica el primero de los p elementos de la ruta.

$$\begin{bmatrix} f_1 & f_2 & \cdots & f_n & 0_1 & 0_2 & \cdots & 0_m \\ f_1 & f_2 & \cdots & f_n & 0_1 & 0_2 & \cdots & 0_m \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ f_1 & f_2 & \cdots & f_n & 0_1 & 0_2 & \cdots & 0_m \end{bmatrix}_{n+m,r} \tag{3.1}$$

$$\begin{bmatrix} q_1^1 & q_2^1 & \cdots & q_m^1 \end{bmatrix} \tag{3.2}$$

El segundo registro se transforma en una matriz de entrada, con la misma estructura, añadiendo la información del paso anterior. Vemos esta información reflejada en la Ecuación 3.3, donde el vector de salida del paso anterior se encuentra en su última fila. En la Ecuación 3.4 vemos el vector que representa al siguiente de los p elementos.

$$\begin{bmatrix} f_1 & f_2 & \cdots & f_n & 0_1 & 0_2 & \cdots & 0_m \\ f_1 & f_2 & \cdots & f_n & 0_1 & 0_2 & \cdots & 0_m \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ f_1 & f_2 & \cdots & f_n & q_1^1 & q_2^1 & \cdots & q_m^1 \end{bmatrix}_{n+m,r} \tag{3.3}$$

$$\begin{bmatrix} q_1^2 & q_2^2 & \cdots & q_m^2 \end{bmatrix} \tag{3.4}$$

De la misma forma, se sigue este ciclo hasta que la salida corresponda al paso p. En este caso la matriz de entrada correspondería a la Ecuación 3.5 y el elemento final de la ruta quedaría representado como se ve en la Ecuación 3.6.

$$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ f_1 & f_2 & \cdots & f_n & q_1^{p-3} & q_2^{1p-3} & \cdots & q_m^{p-3} \\ f_1 & f_2 & \cdots & f_n & q_1^{p-2} & q_2^{p-2} & \cdots & q_m^{p-2} \\ f_1 & f_2 & \cdots & f_n & q_1^{p-1} & q_2^{p-1} & \cdots & q_m^{p-1} \end{bmatrix}_{n+m,r} \tag{3.5}$$

$$\begin{bmatrix} q_1^p & q_2^p & \cdots & q_m^p \end{bmatrix} \tag{3.6}$$

Capítulo 4

Metodología

En este capítulo se describe detalladamente el enfoque metodológico adoptado para el desarrollo de los modelos de predicción. La metodología abarca el desarrollo de cuatro modelos de predicción correspondientes a los objetivos del trabajo, además de un modelo adicional para un proceso complementario. En la Sección 4.1 se aborda la metodología del modelo de sobrevuelos, cuyo objetivo es predecir, entre un conjunto de datos de vuelos, cuáles realizan un sobrevuelo por el espacio aéreo español. La Sección 4.2, se dedica al modelo complementario, que estima la latitud y longitud de los aeropuertos con el fin de completar esta información faltante. La Sección 4.3 se centra en los modelos de predicción de rutas en los tres diferentes niveles definidos: Núcleos (Sección 4.3.1), *Traffic Volume* (Sección 4.3.2) y Plan de Vuelo (Sección 4.3.3).

Para cada proceso metodoógico descrito se expone el procedimiento de preprocesamiento de datos, que incluye la limpieza, normalización y transformación necesarias para asegurar la calidad de los mismos. A continuación, se presentan las herramientas y algoritmos de predicción seleccionados, justificando su elección en base a la literatura existente y a las características específicas del problema abordado. Finalmente, se detalla el proceso de entrenamiento y validación del modelo, incluyendo los parámetros y métricas utilizados para evaluar su desempeño, así como los métodos implementados para asegurar la reproducibilidad y robustez de los resultados obtenidos.

4.1. Modelo de Sobrevuelos

En esta sección se aborda la predicción de si un vuelo pasará por el espacio aéreo español, constituyendo así un problema de clasificación de los vuelos en dos categorías: aquellos que sobrevuelan el espacio aéreo español y aquellos que no. Por lo tanto, este escenario se abordó mediante un enfoque de clasificación binaria, una tarea ampliamente explorada y relativamente sencilla en el ámbito del *Machine Learning*. La metodología propuesta siguió un proceso iterativo, en el cual se analizaron diversos algoritmos y configuraciones. A partir de los resultados obtenidos en cada iteración, se tomaron acciones encaminadas a construir un modelo final que alcanzará los niveles de precisión deseados. Este enfoque permitió refinar continuamente los modelos mediante la evaluación y ajuste de sus componentes, garantizando así una mejora progresiva en su rendimiento y exactitud. En la Figura 4.1, se presenta una visión general de la metodología propuesta.

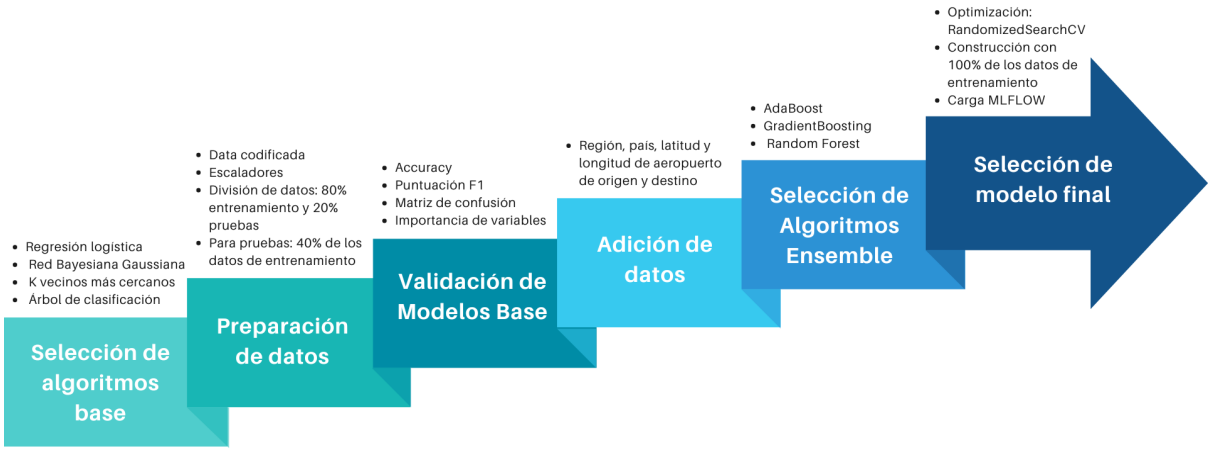

Figura 4.1: Metodología del modelo de sobrevuelos

4.1.1. Selección de algoritmos base

Se llevó a cabo la construcción de modelos de diversa naturaleza con el objetivo de evaluar y comparar el rendimiento de cada uno. Este enfoque permite identificar cuál de los modelos ofrece un mayor rendimiento y así determinar qué familias de modelos nos merecen una mayor atención y refinamiento. Los modelos considerados en este estudio fueron los siguientes:

- Regresión logística
- Red Bayesiana Gaussiana
- K vecinos más cercanos
- Árbol de clasificación

En su implementación en Python, Se utilizaron en general los hiperparámetros por defecto de dichos algoritmos en la biblioteca *Scikit-learn*, salvo en *LogisticRegression*, donde se asignó un peso de 3 a la clase positiva (que cuenta con relativamente muy pocos valores) y de 1 a la negativa con el fin de evitar los efectos del desbalanceo de las clases. Así mismo, tanto en *LogisticRegression* como en *DecisionTreeClassifier* definimos un *random_state* de 10, para asegurar una reproducibilidad en los resultados.

4.1.2. Preparación de datos

Uno de los puntos clave en la IA son los datos de entrada que se proporcionan al modelo. Las variables de entrada para la construcción de nuestro modelo, dado que deben ser numéricas, son las siguientes:

- Origin_l: codificación de la variable “Origin“.
- Destination_l : codificación de la variable “Destination“.
- Airline_l : codificación de la variable “Airline“.
- Aircraft_Type_l : codificación de la variable “Aircraft_Type“.
- Day: día extraído de la vaiable “Date“.

- Month: mes extraído de la vaiable “Date“.
- WeekDay: número del día de la semana extraído de la vaiable “Date“.
- ATOT_Hour_UTC: variable sin modificar.
- ATA_Hour_UTC: variable sin modificar.
- Route_Length__NM: variable sin modificar.

A partir de los modelos seleccionados, se realizaron pruebas utilizando 4 diferentes algoritmos para escalar los valores de cada una de las variables. El escalado se utiiliza para evitar errores numéricos o inestabilidad que pueden surgir al tratar variables que cuentan con valores muy grandes o pequeños. Las técnicas de escalado fueron:

- *MinMaxScaler*, que escala los datos en un rango de 0-1, tomando como referencia los valores mayores y menores vistos en cada característica
- *StandarScaler*, que ajusta las características de los datos para que tengan una media de 0 y una desviación estándar de 1.
- *Normalización L1*, que escala las características de los datos de tal manera que la suma absoluta de los valores de cada característica sea igual a 1.
- *Normalización L2*, que escala las características de los datos de tal manera que la suma de los cuadrados de los valores de cada característica sea igual a 1.

Finalmente, para la adecuada validación de los modelos predictivos, se procedió a la división de los datos totales en dos subconjuntos: un conjunto de entrenamiento al que se le asignó el 80 % del total de datos, y un conjunto de pruebas con el 20 % restante. Esta separación es crucial para asegurar que el modelo desarrollado pueda generalizarse efectivamente a datos no vistos y garantizar que no existan problemas de sobreajuste. Para esta tarea se utilizó el método *train_test_split* de la biblioteca *Scikit-learn*, el cuál garantiza una partición aleatoria y reproducible de los datos, facilitando así una evaluación imparcial del rendimiento del modelo.

Sin embargo, la vasta cantidad de datos disponibles representó una problemática considerable en términos de eficiencia del entrenamiento y su optimización. Por tanto, se utilizó una muestra aleatoria correspondiente al 40 % del total de datos del conjunto de entrenamiento para las pruebas subsiguientes de los modelos, dado que se observó una consistencia significativa en los resultados al emplear esta porción de datos, y únicamente los modelos finales seleccionados serán entrenados con la totalidad de los datos.

4.1.3. Validación de Modelos Base

A partir de las pruebas ejecutadas con diversos algoritmos base y los distintos métodos de escalado utilizados, se pudo observar, a través de la puntuación *F1*, el rendimiento de cada modelo en el contexto de nuestro problema. Esto permitió vislumbrar la capacidad de los diferentes modelos para adaptarse a la naturaleza de nuestros datos. Así, se pudo determinar tanto la familia de algoritmos a emplear como el método de escalado de datos más adecuado, con el fin de asegurar una mayor precisión en la predicción.

Asimismo, fue imperativo considerar toda la información derivada de la realización de estas primeras pruebas de rendimiento. Entre los aspectos más significativos se destaca el análisis de la relevancia de cada una de las variables de entrada. Dado que los resultados preliminares nos permiten enfocar nuestra atención en modelos que prometen un desempeño óptimo, el análisis exhaustivo de las características nos posibilita examinar en detalle las variables de mayor importancia. Esto, a su vez, nos orienta en la búsqueda de información adicional o en la aplicación de transformaciones que pueden ser aprovechadas de manera eficaz por el modelo.

4.1.4. Adición de datos

Dada la relevancia que presentan las variables que representan el origen y destino del vuelo, se consideró fundamental incluir variables adicionales que pudieran enriquecer el conjunto de datos y, por ende, mejorar la capacidad predictiva del modelo. Con este propósito, se añadieron como variables de entrada las siguientes:

- Region_dep_l: codificación de la variable “Region_dep“.
- Region_des_l: codificación de la variable “Region_des“.
- Country_de_l: codificación de la variable “Country_dep“.
- Country_des_l: codificación de la variable “Country_des“.
- Latitude_dep : variable sin modificar.
- Latitude_des : variable sin modificar.
- Longitude_dep : variable sin modificar.
- Longitude_des : variable sin modificar.

Asimismo, el método de escalado de datos seleccionado para este nuevo conjunto de datos corresponde al que demostró un rendimiento superior en la validación de los modelos base, el cual sirve como fundamento para las pruebas posteriores. Este enfoque asegura que se aprovechen al máximo las capacidades predictivas del modelo al utilizar el procedimiento de escalado que ha mostrado ser más eficaz en las evaluaciones preliminares.

4.1.5. Selección de Algoritmos *ensemble* y Optimización

Con la información adicional disponible, se propuso la implementación de modelos de mayor complejidad basados en el modelo base que obtuvo un rendimiento superior, además del mismo modelo base, con el objetivo de alcanzar la máxima eficiencia posible en la predicción. Los modelos considerados en este enfoque fueron los siguientes:

- AdaBoost
- GradientBoosting
- Random Forest

En esta etapa, el enfoque se dirigió directamente hacia la optimización de los algoritmos mediante la búsqueda de los hiperparámetros que maximicen la precisión en la clasificación. Para la búsqueda de los hiperparámetros óptimos, se empleó el método *RandomizedSearchCV*, el cual permite explorar una amplia gama de combinaciones

de hiperparámetros de manera eficiente. Este método selecciona aleatoriamente un número determinado de combinaciones posibles de los hiperparámetros, evaluando el rendimiento del modelo para cada combinación y permitiendo identificar los ajustes que proporcionan el mejor desempeño. La métrica elegida para la elección de la mejor configuración de hiperparámetros fue la puntuación *F1*.

4.1.6. Selección de modelo final

La selección del modelo final se basó en un análisis exhaustivo de varias métricas de rendimiento, incluyendo el *accuracy*, la puntuación *F1* y la matriz de confusión en cada uno de los modelos. Para este análisis se construyeron los modelos *ensemble* así como el algoritmo base elegido utilizando las siguientes consideraciones:

- El conjunto de datos de entrenamiento completo, incluyendo las variables adicionales.
- Los hiperparámetros que permitieron el mejor rendimiento en cada uno de los algoritmos determinado por el método *RandomizedSearchCV*.
- Carga en *MLFLOW* de los modelos para su gestión y su futuro despliegue en diferentes entornos de producción.

4.2. Modelo de predicción de latitud y longitud

Debido a la significativa cantidad de aeropuertos que no disponían de información sobre latitud y longitud (97 aeropuertos), y considerando la posibilidad de que este número aumente en predicciones futuras, se propuso la construcción de un modelo destinado a estimar la latitud y longitud de cada aeropuerto que carezca de esta información. Este modelo se basa en los datos extraídos del código ICAO de los aeropuertos.

La metodología para la construcción del modelo final siguió un enfoque similar al empleado en el modelo de sobrevuelos, aunque en una escala más reducida, pues no es imperativo obtener una predicción exhaustivamente precisa, ya que el objetivo de este modelo es simplemente proporcionar una estimación aproximada para evitar campos vacíos y valores por defecto en los datos de entrada del modelo de sobrevuelos. Además, la cantidad relativamente pequeña de datos (3374 aeropuertos diferentes) permitió un enfoque más sencillo, sin representar problemas de eficiencia con los recursos computacionales disponibles. Se implementaron los siguientes pasos:

1. **Preparación de datos**

   El conjunto de datos utilizado corresponde a todos los aeropuertos que estuvieron presentes en los datos del modelo de sobrevuelos. Los datos de entrada de este modelo fueron extraídos directamente del código ICAO de cada uno de los aeropuertos. Estas variables serían:

   - *id*: codificación del código ICAO del aeropuerto.
   - *id_region*: codificación de la región del aeropuerto (primer dígito de su código ICAO).

- *id_country*: codificación del país del aeropuerto (primer dígito de su código ICAO).

Dado que todas las variables de entrada son categóricas y se han codificado para obtener un valor numérico representativo, se seleccionó el algoritmo *MinMaxScaler* para llevar a cabo su escalado. Este método fue elegido porque preserva la distancia relativa entre las categorías de las variables, garantizando así la integridad de las relaciones originales en el espacio transformado.

Finalmente, se procedió a la división del conjunto de datos, empleando una proporción del 80 % para el entrenamiento de los modelos y el 20 % restante para la fase de pruebas. Esta división estratégica permite asegurar que el modelo se entrene con una cantidad significativa de datos, a la vez que se reserva una porción representativa para evaluar su desempeño y capacidad de generalización.

2. **Selección de algoritmos**

   Para obtener la mejor estimación posible, se construyeron varios modelos utilizando la biblioteca *scikit-learn*. Varios de los algoritmos utilizados son los equivalentes en regresión de los vistos en la predicción de sobrevuelos. Los modelos considerados fueron los siguientes:

   - Regresión lineal
   - K vecinos más cercanos para regresión
   - Árbol de regresión
   - Random Forest para regresión

3. **Optimización**

   Para cada uno de los modelos seleccionados, se llevó a cabo una búsqueda de los hiperparámetros óptimos mediante el método previamente mencionado, *RandomizedSearchCV* y utilizando todos los datos de entrenamiento. La métrica elegida para la comparación de los modelos fue el error cuadrático medio (MSE), lo cual permitió identificar las configuraciones que minimizaban este valor y, por ende, proporcionaban las mejores estimaciones posibles.

4. **Selección de modelo final**

   Se construyeron los modelos utilizando los hiperparámetros seleccionados en la etapa anterior y se emplearon para la predicción del conjunto de pruebas. Los valores predichos se compararon con los valores reales utilizando tanto el error absoluto medio (MAE) como el error cuadrático medio (MSE). Con base en estos resultados, se seleccionó el modelo que mejor desempeño mostró, el cual se utilizaría para estimar las latitudes y longitudes de aquellos aeropuertos cuya información es desconocida.

4.3. Modelo de rutas

Los modelos para la predicción de rutas, en sus diferentes niveles, fueron construidos mediante Redes Neuronales Recurrentes debido a la naturaleza de nuestra predicción secuencial. En este trabajo se propusieron diferentes arquitecturas en la búsqueda

de la mayor precisión posible en la predicción de la secuencia completa de las rutas en cada nivel de granularidad.

La idea básica de cada modelo consiste en, a partir de pasos previos de la ruta, predecir el elemento siguiente. La formación de la ruta consiste entonces en un algoritmo que retroalimenta la entrada para la predicción de un elemento, con la salida del anterior, y una vez aparece el elemento final, definido por defecto, concluir la predicción de nuevos elementos.

Es importante señalar que una de las principales características esperadas de estos modelos, además de alcanzar o superar el rendimiento obtenido con la metodología actual de DELFOS, es la consistencia de las predicciones entre los modelos construidos para cada nivel de granularidad. Por ejemplo, se debe evitar, en la medida de lo posible, que la ruta predicha por el modelo de TV de un vuelo contenga un TV que corresponda a un núcleo que no esté presente en la predicción de la ruta por núcleos del vuelo.

4.3.1. Modelo por Núcleos

Este constituye el primer nivel abordado en la predicción de rutas, dado que sus elementos abarcan el área geográfica más extensa. La predicción que se busca obtener en este primer nivel se centra exclusivamente en la ruta ordenada por núcleos que siguió cada vuelo de forma individual, sin considerar el tiempo específico en que cada vuelo atravesó cada núcleo. En la Figura 4.2, se presenta una visión general de la metodología propuesta para el desarrollo de este modelo.

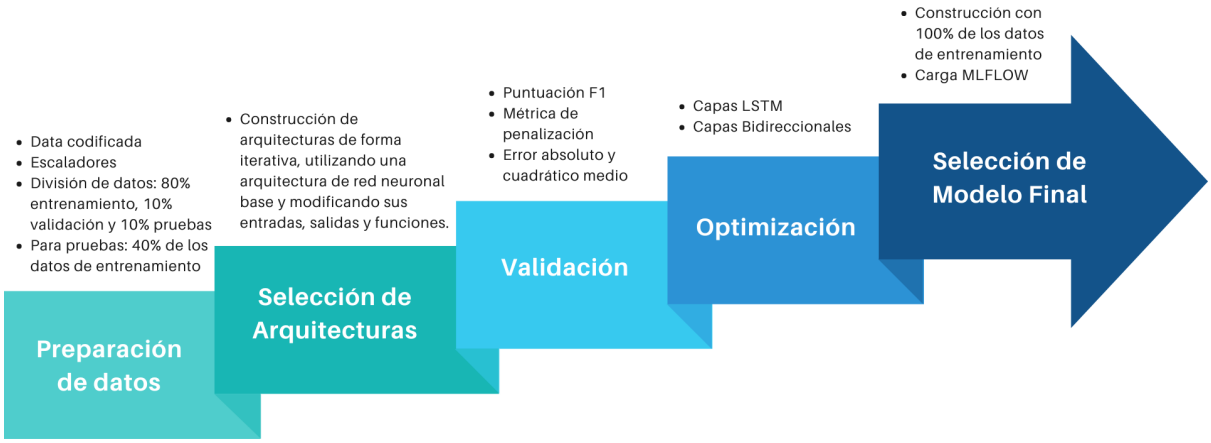

Figura 4.2: Metodología del modelo de paso por Núcleos

4.3.1.1. Preparación de Datos

Los datos de entrada para nuestros modelos constituyen, como en el caso del modelo de sobrevuelos, el detalle básico de un vuelo y, las variables extra que corresponden a información adicional de los aeropuertos. Las características consideradas para el entrenamiento modelo fueron:

- adep_l: codificación de la variable “adep“.
- region_dep_l: codificación de la variable “region_dep“.

- latitude_dep: variable sin modificar.
- longitude_dep: variable sin modificar.
- ades_l : codificación de la variable “Destination“.
- region_des_l: codificación de la variable “Origin“.
- latitude_des: variable sin modificar.
- longitude_des: variable sin modificar.
- routeType_l : codificación de la variable “routeType“.
- aircraftType_l : codificación de la variable “aircraftType“.
- Day: día extraído de la vaiable “processDateReference“.
- Month: mes extraído de la vaiable “processDateReference“.
- WeekDay: número del día de la semana extraído de la vaiable “processDateReference“.

A cada una de estas variables, basados en lo visto en el modelo de sobrevuelos, se utilizó un escalado con *MinMaxScaler* para asegurar que las características mantengan sus relaciones proporcionales y minimizar la influencia de las diferentes escalas de las variables originales.

Respecto a las variables objetivo, en este nivel se considerará únicamente la variable “atcunit“ a predecir, y la variable “datefilterCriteria“ servirá para determinar el orden que le corresponde a los núcleos en la ruta.

Para la adecuada validación de los modelos predictivos, se procedió a la división de los datos totales en tres subconjuntos: un conjunto de entrenamiento al que se le asignó el 80 % del total de datos, un conjunto de validación con otro 10 % de los datos, y un conjunto de prueba con el 10 % restante. Sin embargo, debido a la gran cantidad de registros, se extrajo muestras aleatorias de los conjuntos según nuestras capacidades computacionales para las pruebas y optimización de las arquitecturas y, como en el modelo de sobrevuelo, únicamente se utilizará el conjunto completo de datos para el modelo final. Por tanto, para las pruebas se utilizó el 40 % de los datos del entrenamiento (874 648 vuelos) y un 1 % de los datos de prueba (2,733 vuelos).

Para la entrada de los modelos en este escenario se decidió utilizar 3 pasos anteriores, basados en el promedio de núcleos recorridos en los vuelos disponibles en nuestro conjunto de datos, que es igual a 2.24 núcleos. Cada uno de estos pasos contará con las variables de entrada antes mencionadas, acompañadas de la codificación del núcleo previo correspondiente. Además, se considerará un núcleo final “END“ a cada una de las rutas, que servirá para indicar al modelo cuando la misma ha terminado y así dejar de predecir nuevos elementos.

4.3.1.2. Selección de Arquitecturas

Se propusieron diversas arquitecturas en función de los datos de entrada y la propia estructura de las redes recurrentes. El desarrollo de cada arquitectura propuesta siguió un enfoque iterativo, en el cual los resultados obtenidos de una arquitectura informaban las características que demostraban un mayor rendimiento, las cuales se

incorporaban en la construcción de la siguiente. Las arquitecturas planteadas fueron las siguientes:

1. **Arq1_3Steps_OHE_OriginalFeatures**

   Una primera arquitectura planteada utiliza únicamente las variables originales. Esta prueba de "concepto" sirve principalmente para identificar un nivel de referencia de la precisión del modelo, y permitirá comparar las otras arquitecturas propuestas para este modelo.

   Esta arquitectura fue diseñada utilizando como entrada en cada uno de los pasos anteriores las variables de entrada escaladas y la codificación en *one-hot* (donde cada categoría se convierte en una columna separada con valores de 0 o 1, indicando la presencia o ausencia de esa categoría) de cada uno de los núcleos previos. Como datos de salida, la codificación, en *one-hot* del núcleo siguiente a predecir en la ruta.

   Esta primera arquitectura del modelo fue propuesta enfocándonos en la simplicidad debido a las limitaciones computacionales. La arquitectura de la red propuesta fue la siguiente:

   - Entrada de dimensión (3, 19), correspondiendo el 3 a los pasos previos y el 19 a las variables de entrada y la codificación del núcleo.
   - 1 capa LSTM con 128 neuronas
   - 1 capa de salida con 11 neuronas y función de activación softmax para la predicción de probabilidad de cada núcleo
   - Optimizador: Estimación Adaptativa de Momentos (Adam)
   - Función de pérdida: Entropía cruzada categórica (categorical_crossentropy)

   En esta versión se analizará además el método de construcción de la ruta, pues en nuestro trabajo se han contemplado 2 opciones debido a que, al utilizar una capa de salida con activación *softmax* la predicción será un vector con las probabilidades de cada clase. La función *random.choise* de la librería *numpy* realiza la elección del elemento basándonos en la distribución del vector de salida del modelo, en cambio, *argmax* elige el elemento con mayor probabilidad.

2. **Arq2_3Steps_OHE_AlllFeatures**

   Esta segunda arquitectura fue diseñada utilizando la misma lógica que en el caso anterior, incluyendo en esta versión todas las características disponibles (incluidas las adicionadas de los aeropuertos). Este escenario modifica únicamente la capa de entrada del modelo. La matriz de entrada contará con las 6 nuevas variables en cada uno de los pasos previos de la ruta. La arquitectura de la red propuesta fue la siguiente:

   - Entrada de dimensión (3, 25), correspondiendo el 3 a los pasos previos y el 19 a las variables de entrada y la codificación del núcleo.
   - 1 capa LSTM con 128 neuronas

- 1 capa de salida con 11 neuronas y función de activación softmax para la predicción de probabilidad de cada núcleo
- Optimizador: Estimación Adaptativa de Momentos (Adam)
- Función de pérdida: Entropía cruzada categórica (categorical_crossentropy)

3. **Arq3_3Steps_LE_AlllFeatures**

   Para esta nueva arquitectura, a diferencia de la anterior, se propuso una codificación diferente para cada uno de los núcleos, representando cada núcleo con un valor numérico. Además, para considerar los pasos anteriores en los que aún no se ha predicho un núcleo, es necesario añadir un valor de "START" además de "END" para indicar, tanto como cuando no existe un núcleo en un paso anterior específico, como el final de la ruta. Este cambio transforma nuestro problema en uno de regresión en lugar de clasificación, modificando la arquitectura de esta nueva forma:

   - Entrada de dimensión (3, 15), correspondiendo el 3 a los pasos previos y el 15 a las variables de entrada y la codificación del núcleo.
   - 1 capa LSTM con 128 neuronas
   - 1 capa de salida con 1 neurona y función de activación lineal para la predicción del valor correspondiente al núcleo en la codificación.
   - Optimizador: Estimación Adaptativa de Momentos (Adam)
   - Función de pérdida: Error cuadrático medio (MSE)

   Este enfoque fue propuesto principalmente porque elimina las limitaciones de un modelo de clasificación cuando existe un gran número de clases, pues no es necesario contar con una neurona por cada elemento posible de la ruta (clase). Este enfoque realiza la selección del núcleo siguiente en función de la menor distancia entre la predicción y la codificación numérica del núcleo.

4. **Arq4_3Steps_OHE_Adjacent_AlllFeatures**

   En la búsqueda por aportar información de valor extra al modelo, se propuso incluir, como información de entrada los núcleos geográficamente adyacentes a cada uno de los pasos previos. Esta consideración se realizó para intentar evitar casos donde se construya una ruta que no sea posible de seguir en el contexto real. Además, para la codificación de los núcleos se utilizó aquella que brindó un mejor rendimiento respecto a las arquitecturas anteriores. Por tanto, la arquitectura del modelo sería:

   - Entrada de dimensión (3, 36), correspondiendo el 3 a los pasos previos y el 36 a las variables de entrada, la codificación de los núcleos adyacentes al núcleo y la codificación propia del núcleo.
   - 1 capa LSTM con 128 neuronas
   - 1 capa de salida con 11 neuronas y función de activación softmax para la predicción de probabilidad de cada núcleo.

- Optimizador: Estimación Adaptativa de Momentos (Adam)
- Función de pérdida: Entropía cruzada categórica (categorical_crossentropy)

5. **Arq5_3Steps_OHE_addTMA_AlllFeatures**

   Una de las constantes en las rutas, verificada como determinante por CRIDA, es que los vuelos que inician o finalizan su recorrido en núcleos correspondientes a Áreas de Control Terminal (TMA's) tienen garantizado su paso por el núcleo correspondiente (Madrid TMA y Barcelona TMA). Aunque esta información pueda parecer trivial, se desarrolló un algoritmo de construcción de rutas que contemple estos casos. De esta manera, se simplifica el modelo, eliminando la necesidad de predecir estos núcleos en situaciones donde el vuelo comienza o termina su recorrido en dichos sectores. Esto se lo realizó según 2 condiciones:

   - Si la región del aeropuerto de despegue coincide con la región de Barcelona o Madrid, el registro correspondiente no se utilizará durante el entrenamiento para la predicción de dicho núcleo. Además, en la construcción de la ruta se añadirá el núcleo correspondiente de acuerdo con esta misma condición. Este enfoque reduce la carga del modelo al eliminar un patrón a aprender y garantiza la inclusión del núcleo correcto.
   - Si la región del aeropuerto de destino coincide con la región de Barcelona o Madrid, no se realizarán cambios en los datos de entrada. El modelo deberá determinar en qué punto llegará al núcleo correspondiente. Sin embargo, el algoritmo de construcción de la ruta detendrá la predicción de nuevos elementos de la ruta al encontrar el núcleo correspondiente a la región de destino en caso de ser Barcelona o Madrid. Omitiendo la predicción del núcleo final, lo que mejora el rendimiento; y evitando posibles errores en los que el modelo pudo haber predicho un núcleo extra después del TMA.

   Respecto a la arquitectura de la red, en esta versión no se propuso cambio alguno, únicamente aplicar estas consideraciones a la red que, de todas las arquitecturas anteriores, brindaba mejores resultados.

6. **Arq6_3Steps_OHE_noTMA_AlllFeatures**

   Se considera una propuesta más audaz para esta arquitectura, basada en la premisa de que los pasos por núcleo TMA deberían estar reservados únicamente para los despegues y aterrizajes de las aeronaves en los aeropuertos de la región correspondiente. Sin embargo, esto no se refleja en los datos actuales, los cuales muestran una gran cantidad de vuelos con una ruta que incluye un núcleo TMA intermedio.

   En esta nueva arquitectura, se decide eliminar todos los pasos por núcleos TMA. Se construye una nueva codificación en la que únicamente se consideran ocho núcleos, además del núcleo de fin de ruta. Esta modificación reduce la dimensionalidad de la predicción y permite que el modelo se enfoque en la predicción de núcleos que no presentan patrones deterministas. De esta forma, la arquitectura quedaría:

   - Entrada de dimensión (3, 23), correspondiendo el 3 a los pasos previos y el

23 a las variables de entrada y la codificación de los núcleos excepto TMA.

- 1 capa LSTM con 128 neuronas
- 1 capa de salida con 9 neuronas y función de activación softmax para la predicción de probabilidad de cada núcleo.
- Optimizador: Estimación Adaptativa de Momentos (Adam)
- Función de pérdida: Entropía cruzada categórica (categorical_crossentropy)

Dado que es necesario que las rutas contengan los núcleos TMA, estos serán añadidos posterior a la predicción, basándose en las regiones de origen y destino de los vuelos.

4.3.1.3. Validación

A partir de las pruebas ejecutadas con las diferentes arquitecturas, se procedió a evaluar su rendimiento. Sin embargo, debido a la complejidad de nuestro modelo y nuestra tarea de predicción, esta puede ser evaluada desde diferentes perspectivas y utilizando múltiples métricas.

- Evaluar la precisión del modelo en la predicción del siguiente núcleo. Esta validación se realizó utilizando los datos destinados a la validación y se emplearon las métricas de *accuracy* y la puntuación *F1*.
- Evaluar el número de pasos por cada uno de los núcleos en el conjunto de datos de prueba. Para esta validación se utilizaron las métricas de error absoluto medio (MAE) y error cuadrático medio (MSE).
- Evaluar la precisión en la creación de la ruta completa generada por el modelo para cada vuelo. Para esta validación se utilizó una métrica desarrollada específicamente para este proyecto y a través de ella podemos obtener el valor de cada penalización, la penalización total y el número de rutas predichas correctamente.

De las métricas expuestas, la que representa de mejor forma el rendimiento del modelo es la métrica creada, sin embargo, debido al enfoque de la metodología actual de DELFOS (que predice únicamente el número de pasos por núcleo) las métricas de MAE y MSE deben ser consideradas con igual relevancia, pues estos valores serán los principalmente evaluados para la comparativa contra la metodología actual.

4.3.1.4. Selección de Modelo Final y Optimización

A partir de las validaciones realizadas en las distintas arquitecturas propuestas, se selecciona aquella que ofrece un rendimiento superior. En esta arquitectura del modelo, se efectúan ajustes adicionales para mejorar su rendimiento mientras se mantenga en los límites de nuestra capacidad computacional, integrando capas bidireccionales que permiten una mayor capacidad de aprendizaje y generalización. Además, para esta versión final del modelo, se utilizará la totalidad de los datos disponibles para el entrenamiento, en lugar de una muestra. Este enfoque exhaustivo tiene como objetivo maximizar la precisión y robustez del modelo, asegurando que pueda manejar la complejidad y variabilidad de los datos con mayor eficacia.

El modelo elegido se carga en *MLFLOW* para su gestión y futuro despliegue en diferentes entornos de producción.

4.3.2. Modelo por *Traffic Volume*

En este segundo nivel de rutas, en el cual cada elemento abarca un área geográfica considerablemente más pequeña que los núcleos, buscamos realizar la predicción tanto de los TV como del tiempo en que la aeronave permanece dentro de su porción de espacio aéreo correspondiente. En este escenario uno de los principales objetivos es introducir en el modelo información obtenida a partir del modelo anterior de paso por núcleos, para de esta forma obtener consistencia en las predicciones y así evitar que, por ejemplo, el modelo prediga el paso por algún TV que corresponde a un núcleo que no se encuentre en la predicción por núcleos. En la Figura 4.3, se presenta una visión general de la metodología propuesta para el desarrollo de este modelo.

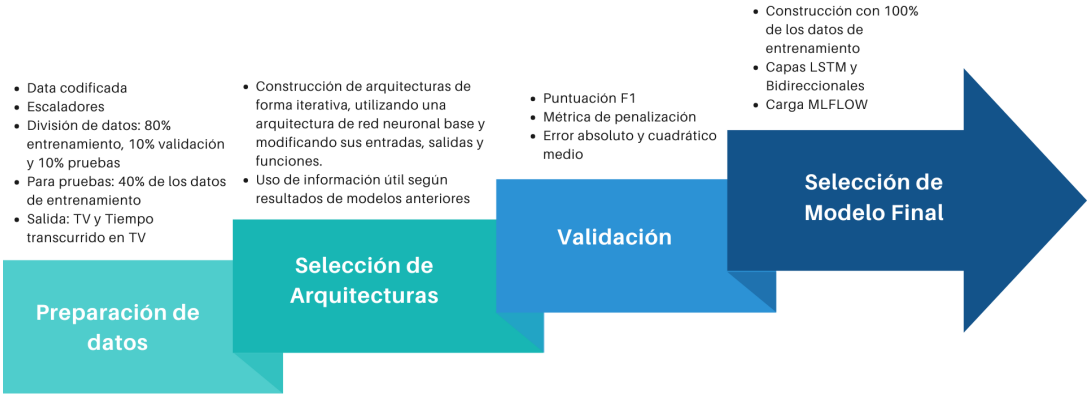

Figura 4.3: Metodología del modelo de paso por Traffic Volume

4.3.2.1. Preparación de Datos

Los datos de entrada para nuestros modelos son los mismos mencionados anteriormente en el modelo por núcleos, además del mismo escalador de datos. Sin embargo, al disponer de diferentes variables objetivo, es necesario realizar transformaciones sobre las mismas para ser utilizadas en nuestros modelos. De tal forma que, en este nivel, las variables objetivo son:

- trafficVolumeCode: variable sin modificar.
- TimeTv: Diferencia en segundos entre las variables EntryTime y ExitTime.

La división de los datos se hizo de la misma manera que en el escenario de núcleos, considerando un conjunto de entrenamiento al que se le asignó el 80% del total de datos, un conjunto de validación con otro 10% de los datos, y un conjunto de prueba con el 10% restante. Y considerando nuevamente un subconjunto generado aleatoriamente para las pruebas de los modelos que correspondan al 40% de los datos del entrenamiento (442,233 vuelos) y un 1% de los datos de prueba (1,384 vuelos).

Para la entrada de los modelos en este escenario se decidió utilizar diferentes números de pasos anteriores, basados en la arquitectura propuesta, así como la infor-

mación previa de entrada y las variables de salida. Se proponen modelos tanto para predecir de forma independiente las variables objetivo, como modelos que abarquen ambas.

4.3.2.2. Selección de Arquitecturas

De manera similar al modelo por núcleos, se adoptó un enfoque iterativo, aprovechando además la información proporcionada por las pruebas realizadas en los modelos anteriores basados en núcleos. De esta forma, se evitó la repetición de pruebas que no resultaron aplicables a estos modelos.

Las rutas por núcleo contemplan 10 elementos, mientras que las rutas por TV comprenden 56 elementos, lo cual incrementa la complejidad en la tarea de predicción. Por consiguiente, las arquitecturas que previamente mostraron malos resultados se esperaría que rindan aún peor en este escenario de mayor complejidad. Las arquitecturas planteadas fueron las siguientes:

1. **Arq1_3Steps_AllOHE-Concatenate-Times**

   Al requerir la predicción de 2 variables de diferente naturaleza (trafficVolumeCode categórica y TimeTv numérica) y, como se observó en el modelo de núcleos que la codificación *one-hot* ofrece mejores resultados en la predicción de los elementos de la ruta, se propuso la construcción de una arquitectura que cuente con 2 capas de salida, una de clasificación (TV) y otra de regresión (tiempo). Esta versión no contempla información del paso por núcleos del anterior modelo. Por tanto, la arquitectura del modelo sería:

   - Entrada de dimensión (3, 72), correspondiendo el 3 a los pasos previos y el 72 a las variables de entrada, la codificación del TV y el tiempo transcurrido en el mismo.
   - 1 capa LSTM con 128 neuronas
   - 1 capa de salida con 57 neuronas y función de activación softmax para la predicción de probabilidad de cada TV.
   - 1 capa de salida con una neurona y función de activación lineal para la predicción del tiempo transcurrido en el TV.
   - Optimizador: Estimación Adaptativa de Momentos (Adam)
   - Función de pérdida de capa softmax: Entropía cruzada categórica (categorical_crossentropy)
   - Función de pérdida de capa lineal: Error cuadrático medio (MSE)

2. **Arq2_3Steps_AllOHE_Times**

   Para esta segunda arquitectura se planteó la construcción de 2 modelos diferentes, uno para la predicción de los TV y otro para la del tiempo. El modelo de predicción de TV considerará al tiempo como una variable más de sus datos de entrada para la predicción del siguiente elemento contando con la siguiente arquitectura:

- Entrada de dimensión (3, 72), correspondiendo el 3 a los pasos previos y el 72 a las variables de entrada, la codificación del TV y el tiempo transcurrido en el mismo.
- 1 capa LSTM
- 1 capa de salida con 57 neuronas y función de activación softmax para la predicción de probabilidad de cada TV.
- Optimizador: Estimación Adaptativa de Momentos (Adam)
- Función de pérdida: Entropía cruzada categórica (categorical_crossentropy)

El modelo de predicción del tiempo, en cambio, considerará como entrada los pasos previos de la ruta omitiendo el tiempo de permanencia en cada uno de los pasos. Esta red buscará predecir el tiempo de permanencia en el último TV presente en los pasos previos. La arquitectura de este modelo entonces sería:

- Entrada de dimensión (3, 71), correspondiendo el 3 a los pasos previos y actual y el 71 a las variables de entrada y la codificación del TV.
- 1 capa LSTM
- 1 capa de salida con una neurona y función de activación lineal para la predicción del tiempo transcurrido en el TV.
- Optimizador: Estimación Adaptativa de Momentos (Adam)
- Función de pérdida: Error cuadrático medio (MSE)

De esta forma, la función de construcción de ruta contemplará, primero la predicción del modelo de TV, y con esa información predecir con la segunda red el tiempo que permaneció en dicho TV. Una vez predicho este tiempo, se añade esta información para que la red de TV prediga el siguiente elemento y así sucesivamente hasta encontrarnos con el elemento final "END" que detendrá la generación de nuevos elementos.

3. **Arq3_4Steps_AllOHE_Times**

   Dado que en este escenario las rutas constan de un mayor número de elementos, siendo estos más pequeños geográficamente, se decidió aumentar en 4 la cantidad de pasos previos a considerar. Esta decisión se fundamenta en el promedio de paso por TV en nuestros datos, que es de 3.46 TV por vuelo.

   La arquitectura general es similar a la presentada en la arquitectura 2, realizando la predicción mediante el uso de dos modelos que se retroalimentarán dentro del algoritmo de construcción de rutas.

4. **Arq4_4Steps_AllOHE_Times_NoShortCrossing**

   Una de las casuísticas observadas en una gran cantidad de datos es la presencia *short-crossings*, es decir, pasos por TV que se realizan durante un corto intervalo de tiempo. Este fenómeno ocurre principalmente debido a las acciones tomadas por los controladores aéreos en su labor, por ejemplo, de evitar aproximaciones entre aeronaves, lo cual provoca ligeros desvíos en sus rutas y, por ende, un

breve tránsito por las fronteras de un TV adyacente que no interesa en el análisis de rutas realizado por DELFOS. Para esta arquitectura, se conservaron las especificaciones de la arquitectura 3, eliminando de los datos aquellos pasos por TV cuya permanencia sea inferior a 2 minutos.

5. **Arq5_4Steps_AllOHE_Times_NoShortCrossing_noTMA**

   Una consideración adicional respecto a los datos se basó en el enfoque adoptado en el modelo anterior en relación con los núcleos TMA. Sin embargo, a nivel de TV, esta consideración resulta mucho más restrictiva. En general, un vuelo, salvo en casos excepcionales, transita por los TV correspondientes a los TMA únicamente al inicio o al final de la ruta, dependiendo de si el despegue o aterrizaje ocurre en el área correspondiente. Por lo tanto, se decidió eliminar los TV que corresponden a los TMA y asignarlos en el algoritmo de construcción de rutas, de acuerdo con la región del aeropuerto de despegue o aterrizaje.

   Las arquitecturas de los modelos requerirán, entonces, una ligera modificación en sus capas de entrada y salida debido a la disminución de elementos de ruta. El modelo de TV quedaría:

   - Entrada de dimensión (4, 70), correspondiendo el 4 a los pasos previos y el 70 a las variables de entrada, la codificación del TV y el tiempo transcurrido en el mismo.
   - 1 capa LSTM con 128 neuronas
   - 1 capa de salida con 55 neuronas y función de activación softmax para la predicción de probabilidad de cada TV.
   - Optimizador: Estimación Adaptativa de Momentos (Adam)
   - Función de pérdida: Entropía cruzada categórica (categorical_crossentropy)

   Y el modelo de tiempos sería:

   - Entrada de dimensión (4, 69), correspondiendo el 4 a los pasos previos y actual y el 69 a las variables de entrada y la codificación del TV.
   - 1 capa LSTM
   - 1 capa de salida con una neurona y función de activación lineal para la predicción del tiempo transcurrido en el TV.
   - Optimizador: Estimación Adaptativa de Momentos (Adam)
   - Función de pérdida: Error cuadrático medio (MSE)

6. **Arq6_OHE_Peninsula4Steps_Canarias3Steps_Times_NoShortCrossing_noTMA**

   En esta arquitectura, proponemos utilizar la información proporcionada por el modelo de núcleos con el objetivo de lograr, por un lado, una mayor precisión y, por otro, consistencia en las rutas de núcleos y TV predichas.

   Para ello, se propone la utilización de dos modelos: uno que abarque las predicciones de vuelos en la península y otro que cubra la región de Canarias. Esta separación permite disminuir la dimensionalidad de la predicción, ya que los TV

que anteriormente eran codificados para su clasificación en un solo modelo se repartirán en dos, reduciendo así la complejidad de la clasificación. Esta división se consideró adecuada debido a la situación geográfica de Canarias, que se encuentra a una distancia relativamente grande de la península y, a nivel de ruta, un vuelo generalmente transita por Canarias al inicio o al final de su trayecto, sin realizar recorridos en los que, pasando por algún punto de la península, llegue a Canarias y retorne a la península en un solo viaje. Esta característica permite una fácil integración en la construcción de la ruta.

Es importante recalcar que esta reducción se aplica únicamente a la salida de cada modelo, ya que la entrada incluirá la codificación de todos los TV previos, independientemente de si pertenecen a la península o a Canarias. Aunque con esta división el modelo de la península sigue considerando una gran cantidad de TV (45 TV) en comparación con el modelo de Canarias (9 TV), se mantendrá el uso de 4 pasos previos para el modelo de la península. Para el modelo de Canarias, sin embargo, se reducirá a 3 pasos previos, dado que no abarca rutas excesivamente largas. Esta versión también incluirá el modelo de predicción de tiempos, tal como se definió en la arquitectura 5. De esta manera, la arquitectura del modelo para la predicción de la ruta por TV en la península sería la siguiente:

- Entrada de dimensión (4, 70), correspondiendo el 4 a los pasos previos y el 70 a las variables de entrada, la codificación del TV y el tiempo transcurrido en el mismo.
- 1 capa LSTM con 128 neuronas
- 1 capa de salida con 46 neuronas y función de activación softmax para la predicción de probabilidad de cada TV.
- Optimizador: Estimación Adaptativa de Momentos (Adam)
- Función de pérdida: Entropía cruzada categórica (categorical_crossentropy)

Por su parte, el modelo para la predicción de la ruta en Canarias sería:

- Entrada de dimensión (3, 70), correspondiendo el 3 a los pasos previos y el 70 a las variables de entrada, la codificación del TV y el tiempo transcurrido en el mismo.
- 1 capa LSTM con 128 neuronas
- 1 capa de salida con 10 neuronas y función de activación softmax para la predicción de probabilidad de cada TV.
- Optimizador: Estimación Adaptativa de Momentos (Adam)
- Función de pérdida: Entropía cruzada categórica (categorical_crossentropy)

Es importante recordar que, si bien existen 45 y 9 TV en península y Canarias respectivamente, se debe añadir el elemento “END“ para enseñarle al modelo a detener la generación de elementos de la ruta. Finalmente mencionar, que para este escenario no se creó un nuevo modelo de predicción de tiempos, pues se consideró oportuno conservar el modelo de la arquitectura.

7. **Arq7_OHE_ModelByEachNucleo_Times_NoShortCrossing_noTMA**

   La última arquitectura propuesta se desarrolló utilizando la información completa proporcionada por el modelo de predicción de la ruta por núcleos, con el fin de evitar cualquier posible inconsistencia en los datos. Por tanto, se planteó la construcción de un modelo para predecir el paso de TV por cada núcleo de manera independiente, concatenando las rutas predichas según el orden determinado por el modelo de núcleos.

   Uno de los beneficios de esta arquitectura radica en la segmentación del problema. De esta manera, un único modelo no se encargará de predecir la totalidad de las rutas, sino únicamente una sección correspondiente. Este enfoque, además de eliminar las inconsistencias con la predicción por núcleos, reduce el problema a partes más manejables. Sin embargo, es importante considerar que cualquier error en la predicción por núcleos se trasladará al nivel de TV.

   Para esta arquitectura se construyeron seis modelos, correspondientes a aquellos núcleos que se dividen en varios TV. Los núcleos que se correspondan directamente con un único TV serán simplemente asignados a la ruta, eliminando la necesidad de predicción por parte del modelo. Los modelos construidos siguen la misma arquitectura base observada en casos anteriores. La entrada será consistente en todos los casos con la presentada en la arquitectura anterior, dado que, aunque cada modelo predecirá únicamente los TV del núcleo correspondiente, su entrada debe contener la codificación de todos los TV para indicar al modelo los pasos anteriores, incluso si pertenecen a otro núcleo. La salida, sin embargo, será diferente en cada caso, ya que está determinada en función del número de TV presentes en cada núcleo.

   - Salida del modelo Barcelona Ruta Este: 8 neuronas (TV constitutivos + “END“)
   - Salida del modelo Barcelona Ruta Oeste: 8 neuronas (TV constitutivos + “END“)
   - Salida del modelo Madrid Ruta 1: 11 neuronas (TV constitutivos + “END“)
   - Salida del modelo Madrid Ruta 2: 10 neuronas (TV constitutivos + “END“)
   - Salida del modelo Sevilla ACC: 11 neuronas (TV constitutivos + “END“)
   - Salida del modelo Canarias ACC: 10 neuronas (TV constitutivos + “END“)

   En esta arquitectura se mantiene el enfoque de la anterior, conservando un solo modelo para la predicción de los tiempos.

4.3.2.3. Validación

La validación de las arquitecturas propuestas se llevó a cabo utilizando las mismas métricas que en el modelo por núcleos. No obstante, en este nivel se buscó evaluar las inconsistencias presentes en la ruta predicha por núcleos, ya que la concordancia de las predicciones es un aspecto de gran relevancia para nuestro estudio. Por lo tanto, las validaciones a considerar serán las siguientes:

- Evaluar la precisión del modelo en la predicción del siguiente *Traffic Volume*. Esta validación se realizó utilizando los datos destinados a la validación y se

emplearon las métricas de *accuracy* y la puntuación *F1*.

- Evaluar el número de pasos por cada uno de los TV en el conjunto de datos de prueba. Para esta validación se utilizaron las métricas de error absoluto medio (MAE) y error cuadrático medio (MSE).
- Evaluar la precisión en la creación de la ruta completa generada por el modelo para cada vuelo, utilizando la métrica específica desarrollada para este propósito.
- Evaluar la concordancia entre la ruta por núcleos extrapolada a partir de la predicción de TV y la ruta de núcleos real registrada en el conjunto de datos de rutas de núcleos.

4.3.2.4. Selección del Modelo Final

Para la selección del modelo final en el contexto de la predicción de *Traffic Volume* (TV), se siguió un enfoque iterativo, considerando tanto las arquitecturas propuestas como los resultados de las validaciones realizadas.

Basado en los resultados de las validaciones realizadas, se seleccionó la arquitectura que mostró el mejor rendimiento general. Se integraron capas bidireccionales para mejorar la capacidad de aprendizaje del modelo y se utilizó la totalidad de los datos disponibles para el entrenamiento, en lugar de una muestra, con el objetivo de maximizar la precisión y robustez del modelo final. Esta metodología garantizó la elección de un modelo que no solo ofrece un alto rendimiento, sino que también mantiene la coherencia y precisión necesarias para las predicciones de *Traffic Volume*.

4.3.3. Modelo por Plan de Vuelo

Para este nivel fue necesario un enfoque diferente a los aplicados anteriormente, pues debido a la gran cardinalidad de los *WayPoints* (1,338 diferentes WP) es inabarcable como una tarea de clasificación. Es por ello que, bajo la misma lógica de una red recurrente de contar con pasos previos para predecir el siguiente elemento de la ruta, se propuso un modelo de regresión para la latitud, longitud y nivel de vuelo, donde, a partir de los valores de latitud y longitud, se determinará el WP correspondiente en la ruta. En la Figura 4.4, se presenta una visión general de la metodología propuesta para el desarrollo de este modelo.

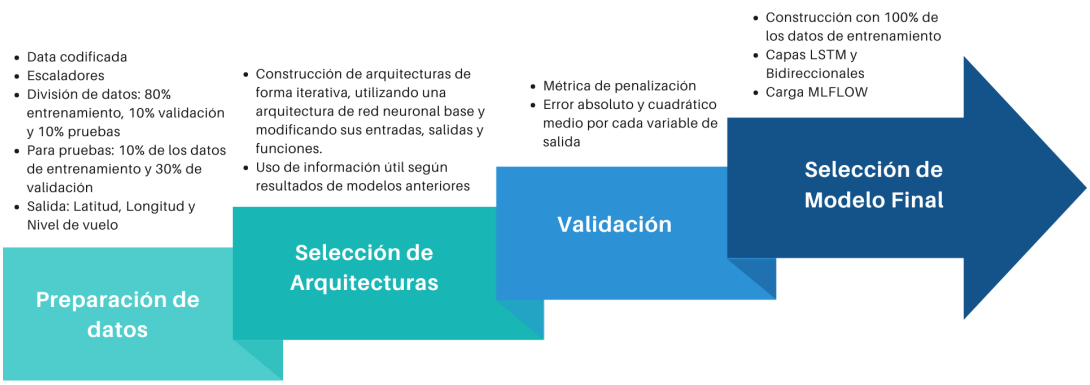

Figura 4.4: Metodología del modelo de plan de vuelo

4.3.3.1. Preparación de Datos

Los datos de entrada para nuestros modelos son los mismos que se abordaron en los modelos anteriores de rutas. En este nivel, igual que para el de *Traffic Volume*, las variantes estan dadas por las variables objetivo. Es este nivel las variables consideradas son:

- *Latitude*: coordenada de latitud del WP por el que registra su paso un vuelo.
- *Longitude*: coordenada de longitud del WP por el que registra su paso un vuelo.
- *FL*: nivel de vuelo por el que registra su paso un vuelo.

La variable "ETO" se utilizó unicamente para determinar el orden de los WP en la ruta. Respecto a la infromación de los pasos previos en este escenario, es necesaria una transformación, pues en modelos anteriores, cada paso previo utilizaba una codificación *one-hot* en este nuevo escenario todas las variables objetivo corresponden a valores numéricos con diferentes escalas. Debido a esto se consideró realizar un escalado *MinMax* a cada una de las variables objetivo cuado estas representen un paso previo en los datos de entrada del modelo.

Aunque en nuestros datos encontramos la variable correspondiente al *WayPoint*, esta será seleccionada a partir de los valores de latitud y longitud. Por tanto, se diseñó un conjunto de datos que incluye todos los WP junto con sus respectivas latitudes y longitudes. El WP correspondiente se seleccionará mediante un algoritmo que calcule el error cuadrático medio entre la predicción y los WP registrados, eligiendo aquel que tenga el menor valor de esta métrica de error. Una consideración importante fue la aplicación del escalado *MinMax* a las latitudes y longitudes antes de realizar la elección, ya que, de lo contrario, las diferentes escalas de estos valores podrían dar una falsa noción de su separación.

La división de los datos se hizo de la misma manera que en los escenarios anteriores, considerando un conjunto de entrenamiento al que se le asignó el 80 % del total de datos, un conjunto de validación con otro 10 % de los datos, y un conjunto de prueba con el 10 % restante. Sin embargo, los subconjuntos utilizados para las pruebas debieron reducirse debido a que el número de registros para este nivel de granularidad es sumamente grande, pues el número medio de WP recorridos en una ruta es de 9. Por tanto se consideró un subconjunto generado aleatoriamente para las pruebas de los modelos que correspondan al 20 % de los datos del entrenamiento (414,022 vuelos), 30 % de los datos de validación (77,964 vuelos) y un 1 % de los datos de prueba (2,599 vuelos).

4.3.3.2. Pruebas

Se diseñaron varias arquitecturas con el objetivo de conseguir la mayor precisión posible en la predicción de nuestras variables objetivo. Sin embargo, a diferencia de los niveles anteriores, en este nivel no se pudieron aplicar numerosas modificaciones o enfoques a las arquitecturas debido a la simplicidad de la ruta, cuyas variables, en términos generales, consisten en puntos en coordenadas tridimensionales.

1. **Arq1_4Steps_OneModel**

   Una primera arquitectura considerada consistió en un único modelo que prediga el paso siguiente en la ruta en base a 4 pasos previos ya recorridos. La

arquitectura del modelo sería:

- Entrada de dimensión (4, 17), correspondiendo el 4 a los pasos previos y el 17 a las variables de entrada y las 3 variables objetivo con un escalado *MinMax*.
- 1 capa LSTM con 128 neuronas
- 1 capa de salida con 3 neuronas y función de activación lineal para la predicción de las variables objetivo.
- Optimizador: Estimación Adaptativa de Momentos (Adam)
- Función de pérdida: Error cuadrático medio (MSE)

Es importante señalar que, tras realizar la predicción del paso siguiente, esta información es escalada utilizando el método *MinMax* antes de ser introducida al modelo como paso previo para la siguiente iteración.

2. **Arq2_4Steps_Lat_Long_FL_Independ**

   Una de las propuestas fue construir una red independiente para cada variable objetivo, dado que la evidencia sugiere que, en tareas de predicción, fragmentar el problema suele conducir a mejores resultados. Por lo tanto, para esta arquitectura se propuso el siguiente diseño para cada una de las redes:

   - Entrada de dimensión (4, 15), correspondiendo el 4 a los pasos previos y el 15 a las variables de entrada y la variable objetivo respectiva con un escalado *MinMax*.
   - 1 capa LSTM con 128 neuronas
   - 1 capa de salida con una neurona y función de activación lineal para la predicción de la variable objetivo respectiva.
   - Optimizador: Estimación Adaptativa de Momentos (Adam)
   - Función de pérdida: Error cuadrático medio (MSE)

   En esta arquitectura, es necesario disponer de tres escaladores *MinMax* diferentes, uno para cada una de las variables objetivo, para su transformación al ser introducidas como paso previo en cada uno de los modelos respectivos.

3. **Arq3_4Steps_LatLong_FL**

   Una tercera propuesta fue abordar la predicción de la latitud y longitud en un solo modelo, y el nivel de vuelo en otro modelo separado, de forma similar al modelo de tiempo en los *Traffic Volume*. La información obtenida de este modelo serviría para la predicción de la latitud y longitud, que en términos generales, representa el siguiente elemento de la ruta.

   La arquitectura del modelo de latitud y longitud sería la siguiente:

   - Entrada de dimensión (4, 17), correspondiendo el 4 a los pasos previos y el 17 a las variables de entrada, la latitus, longitud y nivel de vuelo de cada paso previo.

- 1 capa LSTM con 128 neuronas
- 1 capa de salida con 2 neuronas y función de activación lineal para la predicción de la latitud y longitud del siguiente elemento.
- Optimizador: Estimación Adaptativa de Momentos (Adam)
- Función de pérdida: Error cuadrático medio (MSE)

Por su parte, el modelo para la predicción del nivel de vuelo sería:

- Entrada de dimensión (4, 16), correspondiendo el 3 a los pasos previos y el 16 a las variables de entrada y la longitud y latitud de cada paso previo debidamente escalados.
- 1 capa LSTM con 128 neuronas
- 1 capa de salida con 1 neurona y función lineal para la predicción del nivel de vuelo en el paso actual al que se encuentra la ruta.
- Optimizador: Estimación Adaptativa de Momentos (Adam)
- Función de pérdida: Error cuadrático medio (MSE)

En esta arquitectura, se mantiene el proceso de escalado de las predicciones de la arquitectura 1, para transformar los valores predichos antes de ser introducidas como paso previo en el modelo de predicción de latitud y longitud. Además es necesario contar con otro escalador *MinMax* para unicamente la latitud y longitud el introducirse como entrada en el modelo de predicción de nivel de vuelo.

4. **Arq4_4Steps_Lat_Long_FL_Depend**

   Esta arquitectura sigue el mismo principio de la arquitectura 2, al proponer un modelo diferente para cada una de las variables objetivo. Sin embargo, en lugar de considerar únicamente la variable respectiva que predice el modelo, esta propuesta incluye todas las variables objetivo de los pasos previos como entradas. La arquitectura de cada red sería entonces la siguiente:

   - Entrada de dimensión (4, 17), correspondiendo el 4 a los pasos previos y el 17 a las variables de entrada y las variables objetivo con un escalado *MinMax*.
   - 1 capa LSTM con 128 neuronas
   - 1 capa de salida con una neurona y función de activación lineal para la predicción de la variable objetivo respectiva.
   - Optimizador: Estimación Adaptativa de Momentos (Adam)
   - Función de pérdida: Error cuadrático medio (MSE)

   La idea es realizar la predicción de cada modelo de forma individual, concatenar estos valores, escalarlos y ubicarlos como entrada para la predicción del siguiente paso en la ruta.

5. **Arq5_4Steps_Lat_Long_FL_Depend_Correction**

Para esta última arquitectura, se modificó únicamente la función de construcción de la ruta, conservando las redes de la arquitectura 4. En lugar de utilizar los valores predichos por las redes en los pasos previos, se emplearán los valores correspondientes a la latitud y longitud del *WayPoint* más cercano a la predicción. De esta manera, se busca corregir los pequeños desvíos realizados en las predicciones, garantizando una mayor precisión en la ruta generada.

4.3.3.3. Validación

La validación de las arquitecturas propuestas se llevó a cabo utilizando las mismas métricas a nivel de ruta que en los modelos de rutas anteriores. Sin embargo, al pasarnos de un enfoque de clasificación a uno puramente de regresión, la evaluación de los modelos se realzará con métricas correspondientes a su situación. Por tanto las metricas utilizadas en este nivel fueron:

- Evaluar la precisión de los modelos en la predicción de cada una de las variables objetivo. Esta validación se realizó utilizando los datos destinados a la validación y se emplearon las métricas de error absoluto medio (MAE) y error cuadrático medio (MSE).
- Evaluar la precisión en la creación de la ruta completa generada por el modelo para cada vuelo, utilizando la métrica específica desarrollada para este propósito.
- Evaluar la concordancia entre la ruta por *Traffic Volume* extrapolada a partir de la predicción de *WayPoints* y la ruta real de TV.

4.3.3.4. Selección del Modelo Final

Para la selección del modelo final en el contexto de la predicción de *Waypoints*, se siguió un enfoque iterativo, considerando tanto las arquitecturas propuestas como los resultados de las validaciones realizadas. Basado en dichas validaciones, se seleccionó la arquitectura que mostró el mejor rendimiento general. Además, se integraron capas bidireccionales para mejorar la capacidad de aprendizaje del modelo y se utilizó la totalidad de los datos disponibles para el entrenamiento, en lugar de una muestra, con el objetivo de maximizar la precisión y robustez del modelo final.

Capítulo 5

Resultados

En este capítulo se presentan los resultados obtenidos a partir de la implementación y validación de los modelos propuestos en los enfoques de sobrevuelos y rutas, abarcando sus distintos niveles de granularidad. Se detallan los rendimientos de los diversos enfoques metodológicos y arquitecturas desarrolladas, así como la comparación entre ellos. Los resultados se interpretan en el contexto de los objetivos planteados, y se discuten sus implicaciones prácticas y teóricas, proporcionando una visión integral del desempeño de los modelos y su aplicabilidad en escenarios reales.

5.1. Modelo de sobrevuelos

El desarrollo del modelo de sobrevuelos se centró en la utilización de múltiples algoritmos con el fin de analizar su rendimiento y determinar aquel que mejor se adapta a nuestro escenario, logrando la predicción más óptima según las métricas de evaluación consideradas. En esta sección podemos observar el modelo final desarrollado y su rendimiento, además de cada una de las pruebas y las validaciones que justificaron su selección.

Así mismo, se presentarán los resultados del uso del modelo final en un entorno real. A partir del cual se realizará una comparación del rendimiento del modelo con la metodología estadística actual utilizada por CRIDA y con los datos observados en el escenario real.

5.1.1. Pruebas

Un enfoque inicial consistió en realizar pruebas de los algoritmos utilizando muestras de datos representativas. Se procuró que los algoritmos empleados para la construcción de los modelos fueran lo más variados posible, con el objetivo de identificar la familia de algoritmos que proporcionara una mayor precisión en nuestro problema de clasificación. En la Tabla 5.1, se presenta una primera aproximación al rendimiento de cada uno de los algoritmos seleccionados, utilizando un escalado *MinMax* y construidos mediante la biblioteca *Scikit-learn*.

Los resultados expuestos revelan una precisión muy baja en la predicción por parte de los modelos, a excepción del modelo de árbol de decisión, que muestra un rendimiento significativamente superior al esperado en primera instancia. Sin embargo,

Tabla 5.1: Modelos Base de Clasifiación Binaria

Modelo	Hiperparámetros	F1	FP	FN
LogisticRegression	`class_weight= { True:3, False:1}, random_state=10`	0.00001	246,691	79
KNeighborsClassifier	`default`	0.5125	143,001	54,207
GaussianNB	`default`	0.0	246,693	0
DecisionTreeClassifier	`random_state = 10`	0.9686	7,619	7,836

debido a la naturaleza de los algoritmos, no es razonable descartarlos tras una única prueba y utilizando un solo escalador de datos, ya que muchos algoritmos son extremadamente sensibles a estas transformaciones. Por esta razón, se realizaron pruebas de rendimiento de cada algoritmo con los diferentes métodos de escalado propuestos.

En la Tabla 5.2 se observa el rendimiento de los modelos construidos con los mismos hiperparámetros que en la prueba anterior, y se puede apreciar cómo varía el rendimiento entre los diferentes escenarios de escalado. La mayoría de los algoritmos modifican su rendimiento en función de estas transformaciones, destacando especialmente el algoritmo de K vecinos más cercanos, que mejora notablemente su rendimiento con las normalizaciones L1 y L2, aunque sin alcanzar la precisión del árbol de decisión.

Tabla 5.2: Comparación de scalers para clasificación binaria

Modelo	Scaler	F1	FP	FN
LogisticRegression	Standar Scaler	0.00001	246,691	69
KNeighborsClassifier	Standar Scaler	0.6196	113,518	49,943
GaussianNB	Standar Scaler	0.0	246,693	0
DecisionTreeClassifier	Standar Scaler	0.9686	7,603	7,853
LogisticRegression	Normalización L1	0.0000	246,693	0
KNeighborsClassifier	Normalización L1	0.9518	14,078	9,453
GaussianNB	Normalización L1	0.1984	161,367	528,047
DecisionTreeClassifier	Normalización L1	0.9311	16,922	17,067
LogisticRegression	Normalización L2	0.0000	246,693	0
KNeighborsClassifier	Normalización L2	0.9521	14,002	9,369
GaussianNB	Normalización L2	0.1980	140,933	715,648
DecisionTreeClassifier	Normalización L2	0.9378	15,244	15,436

Respecto al árbol de decisión, este presenta un rendimiento similar tanto con *MinMaxScaler*, como con *StandarScaler*; sin embargo, se prefiere el *MinMaxScaler*, porque mantiene intacta la distancia relativa entre las clases de cada característica, lo que contribuye a una mejor partición de los nodos durante el entrenamiento. Además, debido a su rendimiento superior comparado con los demás modelos y su naturaleza inherentemente explicable, resulta factible su uso para el análisis de la importancia de las variables de entrada del modelo, permitiendo así determinar cuáles contribuyen significativamente a alcanzar los altos niveles de precisión observados.

En la Figura 5.1, se puede observar la importancia asignada por el modelo de árbol de decisión a cada una de las variables, destacándose que las variables correspondientes a los aeropuertos de origen y destino codificados poseen la mayor relevancia para el

modelo. Este resultado era, hasta cierto punto, esperable, ya que la gestión del tráfico aéreo busca que las rutas seguidas por las aeronaves sean lo más directas posible.

Por lo tanto, los aeropuertos de origen y destino que cuentan con un trayecto directo sin atravesar el espacio aéreo español tienden a evitar su paso por el mismo. Es importante señalar que estas rutas no son deterministas, ya que en la realidad se contemplan múltiples escenarios que requieren la toma de decisiones y correcciones por parte de las aeronaves debido a diversos factores, como condiciones naturales que obligan a seguir rutas alternativas. Estos escenarios pueden identificarse, en cierta medida, mediante la variable de longitud de la ruta (la tercera variable con mayor importancia). Dado un origen y un destino, si la longitud de la ruta resulta ser significativamente mayor que su valor estándar, esto indicaría un desvío notable de la trayectoria, lo cual aumentaría la probabilidad de realizar un sobrevuelo sobre el espacio aéreo español.

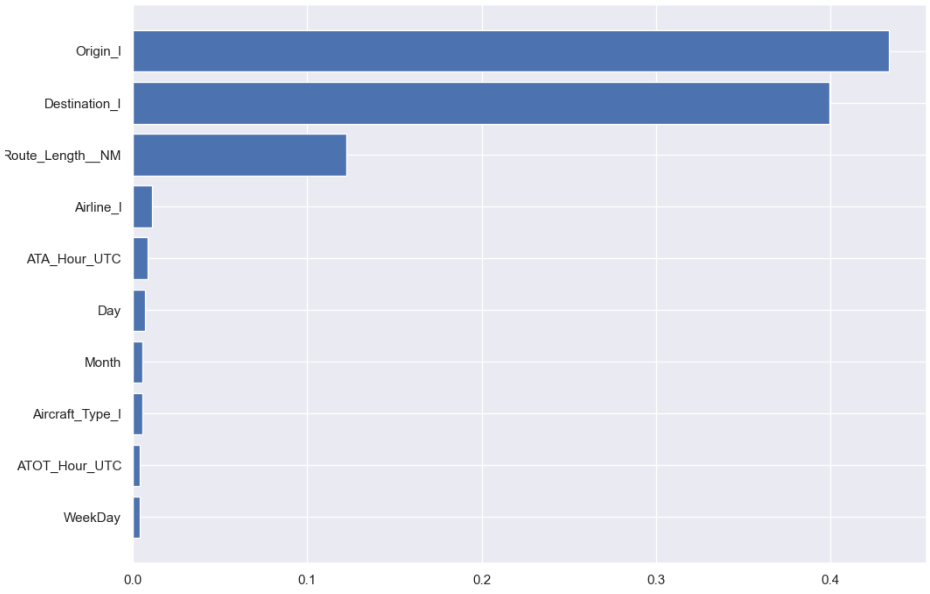

Figura 5.1: Importancia de las características para modelo de clasifiación binaria

Dada la relevancia de las variables correspondientes a los aeropuertos de origen y destino, se buscó enriquecer el modelo con información adicional relacionada con estos aeropuertos. Para ello, CRIDA nos proporcionó datos detallados de cada aeropuerto que tienen registrado. Sin embargo, es importante señalar que esta información no incluye varios aeropuertos que aparecen en los datos a predecir, ya sea por ser nuevos o, en algunos casos, por ser privados y no disponer de información sobre ellos. Debido a esta ausencia de datos, se consideró extraer únicamente las variables más representativas de cada aeropuerto: la latitud y longitud. Para abordar la ausencia de esta información en los aeropuertos, se desarrolló un modelo de predicción de latitud y longitud, cuya construcción se detalla en la Sección 5.2, además se añadió nuevas columnas extraídas a partir de la información brindada por el código ICAO del aeropuerto: región y país.

Una vez se contó con la información adicional completa de los aeropuertos para enriquecer los datos de origen y destino del vuelo, se procedió a re-analizar el rendimiento de los modelos, utilizando los hiperparámetros básicos definidos anteriormente y aplicando el escalado que permitió un mayor rendimiento a cada algoritmo. En la Tabla 5.3 se observa la mejora obtenida en cada uno de los modelos debido a la adición de las nuevas variables de entrada, siendo especialmente significativa en los modelos de regresión logística y *Gaussian Naive Bayes*. No obstante, el modelo de árbol de decisión continúa mostrando una precisión considerablemente superior a los demás.

Tabla 5.3: Rendimiento de Modelos Base con la adición de nuevas variables

Modelo	**Scaler**	**F1**	**FP**	**FN**
LogisticRegression	MinMaxScaler	0.6384	80,569	107,542
KNeighborsClassifier	Normalización L2	0.9582	12,548	7,844
GaussianNB	Normalización L1	0.3563	64,081	595,572
DecisionTreeClassifier	MinMaxScaler	0.9705	7,108	7,457

5.1.2. Optimización

Dado que el modelo de árbol de decisión fue el que presentó un mejor rendimiento, se relizó la búsqueda del mejor modelo en algoritmos *ensemble* basados en árboles, los cuales consisten en arquitecturas que unen varios árboles de decisión para mejorar la predicción. Los nuevos modelos elegidos basados en arboles son:

- AdaBoost
- GradientBoosting
- Random Forest

Aunque la construcción y optimización de estos nuevos modelos es relativamente sencilla utilizando *Scikit-learn*, el modelo *AdaBoost* presenta una gran problemática debido a la eficiencia de su entrenamiento y, por ende, su optimización. Este problema radica en la elección del modelo débil a utilizar, ya que dicho modelo requiere la configuración de sus propios hiperparámetros. Por esta razón, se buscaron las mejores configuraciones de hiperparámetros para los árboles de decisión en función de su profundidad máxima, utilizando el algoritmo *RandomizedSearchCV*y la puntuación *F1* como métrica de comparación. En la Figura 5.2 se puede observar el rendimiento de los árboles de decisión según su profundidad empleando las mejores configuraciones en cada caso.

El proceso de optimización para el modelo *AdaBoost* se llevó a cabo utilizando un árbol de decisión con una profundidad de 5. Esta elección se fundamenta en que en dicho punto se observa el último incremento significativo en la precisión, medido a través de la puntuación *F1*, pues a partir de esta profundidad, los aumentos en la precisión no son tan marcados. En relación con los demás hiperparámetros del modelo *AdaBoost*, así como de otros modelos *ensemble* y del árbol de decisión individual, se utilizó nuevamente el algoritmo *RandomizedSearchCV*. La puntuación *F1* se empleó como métrica para determinar la configuración óptima de los parámetros y evaluar su precisión.

En la Tabla 5.5 se presentan los resultados de la optimización. Se observa que los cuatro modelos propuestos exhiben un rendimiento muy similar, siendo este suma-

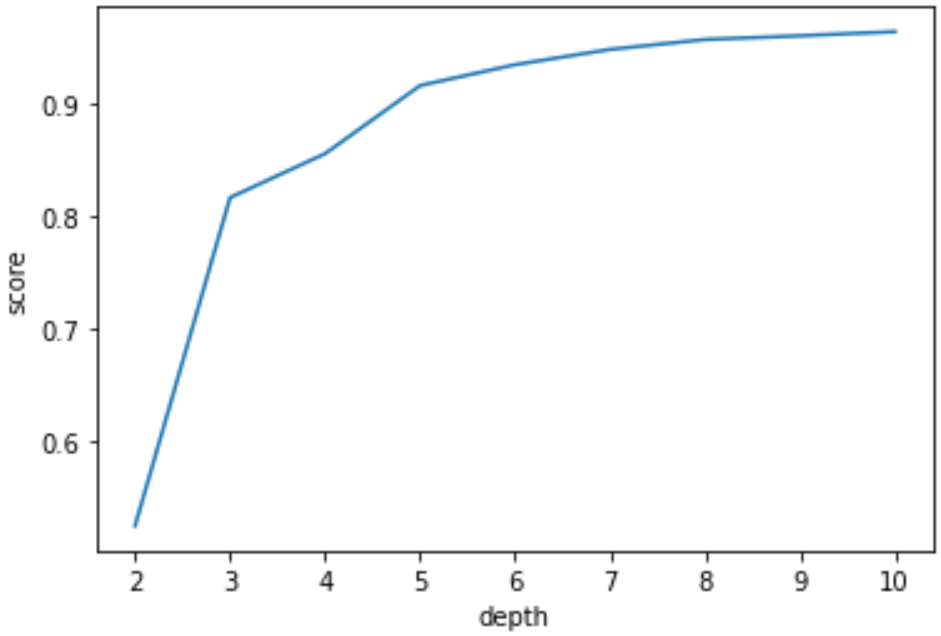

Figura 5.2: Rendimiento de clasificadores débiles

mente alto, con diferencias únicamente en los decimales de su valor en la puntuación *F1*. Por consiguiente, los cuatro modelos, configurados con sus hiperparámetros óptimos, serán sometidos a una evaluación final detallada. Esta evaluación exhaustiva permitirá determinar cuál de ellos proporciona los mayores beneficios para nuestro problema de predicción.

5.1.3. Selección del Modelo Final

Para realizar la selección del modelo final que será desplegado e integrado en el proyecto DELFOS, es necesario llevar a cabo una evaluación exhaustiva y detallada. Esta evaluación implica varios pasos críticos, comenzando con el entrenamiento de los modelos utilizando la totalidad de los datos disponibles. Este enfoque asegura que los modelos se beneficien de la máxima cantidad de información para optimizar su rendimiento y mejorar su capacidad de generalización. Los hiperparámetros utilizados para su construcción corresponden a los encontrados en las pruebas, realizadas con el 40 % de conjunto de entrenamiento y, en la construcción del modelo, se implementó su carga a MLFLOW, para su correcta gestión y futuro despliegue de uno o más modelos finales construidos.

La Tabla 5.5 presenta los resultados de los modelos finales de sobrevuelos evaluados en términos de las métricas de exactitud (accuracy), puntuación *F1*, y el número de falsos positivos y falsos negativos. El *acurracy* de los modelos es notablemente alto en todos los casos, siendo *RandomForestClassifier* el modelo con la mayor exactitud de 0.99694, mientras que el *GradientBoostingClassifier* tiene la menor valor de 0.99553. Sin embargo, estas diferencias son mínimas y sugieren que todos los modelos tienen un desempeño excelente.

En el caso de la puntuación *F1*, nuevamente el *RandomForestClassifier* lidera con un valor de 0.97983, seguida por otro *DecisionTreeClassifier* con 0.97637. El *Ada-*

Tabla 5.4: Optimización de Modelos de Clasifiación Binaria

Modelo	Hiperparámetros	F1	FP	FN
DecisionTreeClassifier	`min_samples_split: 5, min_samples_leaf: 7, max_depth: 21, criterion: gini`	0.97649	4,583	6,945
AdaBoostClassifier	`estimator: DecisionTreeClassifier (criterion: entropy, max_depth: 5, min_samples_split: 5, min_samples_leaf: 4, random_state: 10), n_estimators: 100, learning_rate: 0.7`	0.97174	5,541	8,284
GradientBoostingClassifier	`n_estimators: 110, min_samples_split: 4, min_samples_leaf: 5, max_depth: 5, loss: exponential, learning_rate: 0.9, criterion: friedman_mse`	0.97063	5,830	8,535
RandomForestClassifier	`n_estimators: 110, min_samples_split: 7, min_samples_leaf: 1, max_depth: 24, criterion: entropy`	0.97777	3,523	7,324

BoostClassifier y el *GradientBoostingClassifier* tienen puntuaciones *F1* ligeramente inferiores, con 0.97228 y 0.97057 respectivamente. Estos resultados reflejan que todos los modelos tienen un rendimiento muy similar y alto en términos de equilibrio entre precisión y recall.

La superioridad del modelo *RandomForestClassifier* se ve reflejada claramente en el número de falsos positivos y falsos positivos predichos en el conjunto de prueba, pues en ambos casos logro un significativamente menor número de predicciones erróneas comparado con los demás modelos.

Tabla 5.5: Modelos finales de sobrevuelos

Modelo	acurracy	F1	FP	FN
DecisionTreeClassifier	0.99641	0.97637	4661	6942
AdaBoostClassifier	0.99579	0.97228	5356	8240
GradientBoostingClassifier	0.99553	0.97057	5774	8657
RandomForestClassifier	0.99694	0.97983	3183	6695

5.1.4. Validación en Escenario Real

Para la evaluación del modelo final en un escenario real, el equipo encargado del proyecto DELFOS proporcionó el detalle de la planificación de vuelos registrada con fecha del 4 de marzo de 2024. Cabe recordar que estas planificaciones se realizan con hasta un año de anticipación. Sin embargo, para la validación se consideraron los vuelos planificados desde el 3 de marzo de 2024 hasta el 30 de abril de 2024, ya que, a la fecha, estos vuelos ya han sido realizados y se dispone de la información real sobre si fueron sobrevuelos o no.

Tras aplicar el procesamiento requerido a los datos provistos para la prueba en un escenario real, se realizó la predicción utilizando el modelo *RandomForestClassifier* final cargado a MLFLOW. En la Figura 5.3 observamos la frecuencia de las clases de la predicción, que corresponden a cruzar o no espacio aéreo español. Vemos que el desbalanceo existente es consistente con el visto anteriormente en los datos para la construcción del modelo, pues la predicción muestra que únicamente el 13.93 % de los vuelos fue clasificado como sobrevuelo.

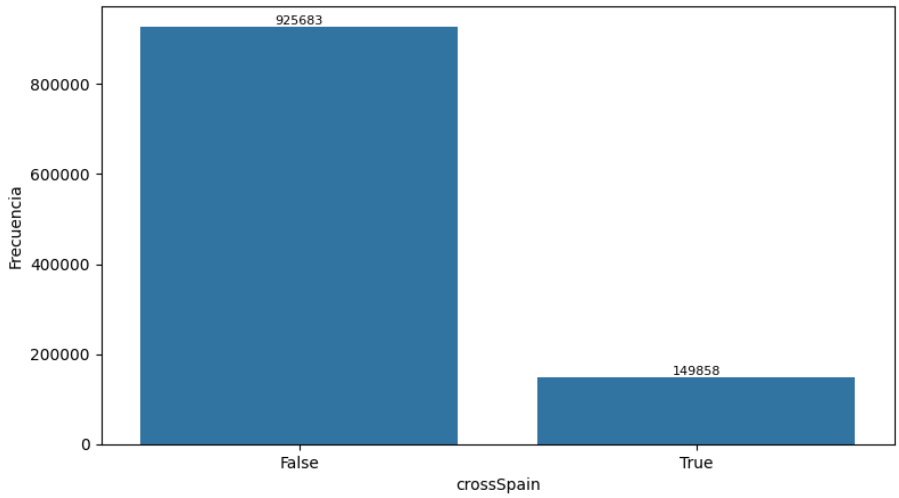

Figura 5.3: Frecuencia de sobrevuelos predicha

Enfocandonos en el rendimiento real del modelo, está determinada por la comparación entre la predicción del modelo, con la predicción realizada por DELFOS utilizando su metodología actual. De tal forma que se buscó analizar cual de las dos predicciones está más cerca del número real de sobrevuelos presentes en los datos. En la Figura 5.4 observamos una gráfica comparativa diaria del número de sobrevuelos registrado por nuestro modelo, la predicción con la metodología actual, y el número real de sobrevuelos registrado. Podemos observar cómo nuestra predicción (línea morada), se encuentra mucho más cerca del valor real (línea roja), oscilando entre predecir un número ligeramente menor o ligramente mayor de sobrevuelos. Por su parte, la predicicón realizada con la metodología actual de DELFOS (línea lila claro) se aleja mucho de la predicción real y siempre realizando muchas más predicciones de sobrevuelos que los realmente ocurridos. Este comportamiento lo podemos observar en las áreas mostradas en la gráfica, las cuales muestran el desvío de la predicción actual de DELFOS (área gris) y la propuesta por nuestro modelo (área

naranja), utilizando como referencia el número real de sobrevuelos.

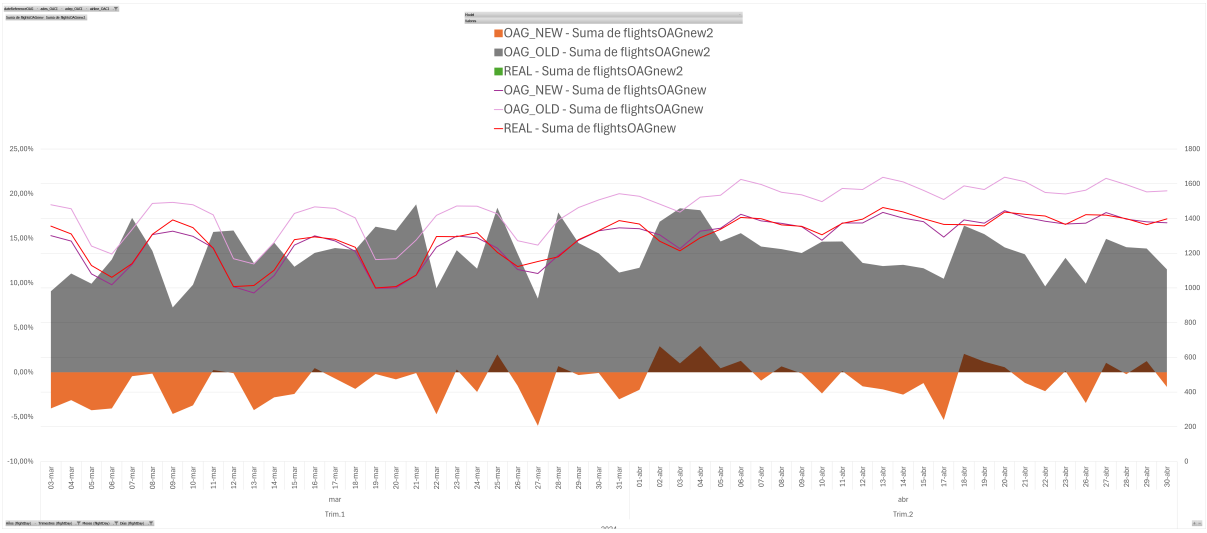

Figura 5.4: Validación de predicción de sobrevuelos

Es notable la mejora general que nuestro modelo propone en comparación con la metodología actual. Sin embargo, un análisis más detallado revela que existen escenarios en los cuales los resultados del modelo no ofrecen una predicción precisa. Desde DELFOS se realizaron validaciones considerando las diferentes aerolíneas y se encontró que el error del modelo se centra principalmente en predecir, casi de forma determinante, que los vuelos de ciertas aerolíneas nunca realizan un sobrevuelo por espacio aéreo español. No obstante, en la realidad, estas aerolíneas suelen registrar un número reducido de sobrevuelos.

Este comportamiento del modelo puede deberse a la cantidad tan pequeña de sobrevuelos de estas aerolíneas. Errar en estas predicciones durante el entrenamiento no representa una penalización considerable para ajustar los parámetros del modelo. Un ejemplo de ello lo encontramos en la aerolínea Royal Jordanian Airlines (RJA) de cuyos vuelos ninguno fue clasificado como sobrevuelo. De los datos utilizados para la creación del modelo que correspondían a esta aerolínea (30,727 vuelos), únicamente el 0.72 % (2,212 vuelos) eran sobrevuelos.

5.2. Modelo de Latitud y Longitud

Este modelo se diseñó específicamente para estimar la latitud y longitud de los aeropuertos que, aunque están presentes en los datos de entrada, no están incluidos en la base de datos de aeropuertos proporcionada por CRIDA. En los datos de entrada, se identificaron 97 aeropuertos sin la información correspondiente. No obstante, en un contexto real, donde pueden surgir nuevos valores no observados durante el entrenamiento, es probable que un nuevo aeropuerto se incorpore al grupo de aeropuertos sin información asociada. Por lo tanto, será necesario mantener este modelo operativo durante la vida útil del algoritmo de sobrevuelos.

El modelo se construyó utilizando la información de los aeropuertos presentes en los datos de entrada, los cuales se dividieron en subconjuntos para entrenamiento y pruebas. Los datos de entrada para este modelo son considerablemente limitados, ya

que únicamente corresponden a la información proporcionada por su código ICAO. Esta limitación complica la construcción de modelos complejos. Además, dado el objetivo específico de este modelo, no se consideró necesario emplear algoritmos o arquitecturas de gran complejidad para su desarrollo.

En la Tabla 5.6, se presenta el rendimiento de cada uno de los algoritmos propuestos, junto con los hiperparámetros seleccionados mediante el método *RandomizedSearchCV*. Para este proceso, se utilizó el error cuadrático medio (MSE) como métrica de selección, dado que penaliza de manera efectiva las desviaciones significativas en las estimaciones de latitud y longitud. Esta elección es particularmente importante para garantizar que las predicciones sean lo más precisas posible y minimizar los errores que podrían afectar negativamente el rendimiento del modelo en aplicaciones reales. Además, la tabla muestra el rendimiento de cada uno de los modelos construidos con los hiperparámetros óptimos seleccionados, proporcionando una comparación clara y detallada de su efectividad.

Tabla 5.6: Comparación de Modelos de Regresión

Modelo	**Hiperparámetros**	**MAE**	**MSE**
LinearRegression	`'fit_intercept': True, 'copy_X': True, 'positive': False`	19.89	1034.12
KNeighborsRegressor	`'weights': 'distance', 'n_neighbors': 28, 'metric': 'cityblock'`	3.33	41.25
DecisionTreeRegressor	`'criterion': 'squared_error', 'max_depth': 17, 'min_samples_split': 3, 'min_samples_leaf': 5, 'random_state': 10`	3.77	55.76
RandomForestRegressor	`'n_estimators': 110, 'criterion': 'absolute_error', 'max_depth': 12, 'min_samples_split': 6, 'min_samples_leaf': 1, 'random_state': 10`	3.50	44.72

A partir de los resultados obtenidos, se observó que el modelo de K vecinos más cercanos (KNN) mostró el mejor rendimiento en las métricas consideradas, específicamente en el Error Absoluto Medio (MAE) y el Error Cuadrático Medio (MSE). Debido a su superior desempeño, se decidió emplear este modelo para predecir las coordenadas de latitud y longitud de aquellos aeropuertos que carecen de esta información en los datos originales y así poder ser utilizados como entrada en el modelo de sobrevuelos.

5.3. Modelo de rutas

El enfoque de rutas aborda tres niveles de granularidad: núcleos, *Traffic Volume* y *Waypoint*, donde cada nivel proporciona un mayor detalle de la trayectoria seguida por un vuelo en comparación con el nivel anterior. En la presente sección se describe el proceso de construcción de un modelo recurrente para cada uno de estos niveles,

comenzando por el nivel de menor detalle hasta el de mayor detalle, cuyo número de iteraciones de su entrenamiento, en todos los casos, estará determinado por la condición de para el entrenamiento en caso de no mostrar una mejora considerable en 10 iteraciones, retornando siempre los pesos de la mejor iteración.

En esta sección se analizará el desarrollo realizado para la obtención del modelo final en cada nivel, con el objetivo de minimizar las inconsistencias entre las rutas predichas por cada nivel. Para lograr esto, se buscó implementar, en la medida de lo posible, un enfoque que permita que la predicción de un nivel de ruta sirva como base para el siguiente nivel.

5.3.1. Por Núcleos

En esta sección se describe la construcción del modelo de predicción que ofrece el mayor rendimiento en la construcción de rutas de núcleos. Inicialmente, nos enfocamos en todo el proceso de pruebas realizadas que justifican su elección, basándonos en la metodología propuesta. En el apartado final, se presenta la arquitectura del modelo seleccionada, así como los detalles de su construcción y rendimiento.

5.3.1.1. Pruebas

Las pruebas aqui descritas fueron realizadas utilizando una muestra aleatoria correspondiente al 40 % del conjunto de datos destinado al entrenamiento y un 1 % del conjunto de pruebas debido a las limitaciones computacionales disponibles. Además, a estos conjuntos de datos se les realizó las tranformaciones requeridas por una red recurrente definidas en la metodología.

El primer enfoque se abordó como un problema de clasificación, en el cual la red recurrente predice la probabilidad del siguiente elemento en la secuencia. La primera arquitectura emplea únicamente los datos originales del conjunto de datos con el fin de establecer un rendimiento de referencia para el modelo. Una consideración clave que influye en los resultados generales de la predicción de la ruta es el método de selección del siguiente elemento. Por esta razón, se analizaron las dos posibilidades propuestas en la metodología: *random.choice* y *argmax*.

En la Tabla 5.7 se presentan los resultados de las validaciones realizadas con cada uno de estos enfoques de selección de elementos. Se puede observar que la precisión, en cada una de nuestras métricas de evaluación, es superior al utilizar *argmax*.

Tabla 5.7: Comparación de métodos de selección de elemento con Arq1.

Método	**F1**	**Penalización**	**% correctas**	**MAE**	**MSE**
random.choise	0.86796	0.94584	66.85 %	14.2	321.4
argmax	0.90876	0.64068	76.44 %	13.1	296.7

La arquitectura 2 se construyó utilizando tanto las variables originales como las adicionales que corresponden al detalle de los aeropuertos de origen y destino. En todos los demás aspectos, es igual a la arquitectura 1, con el objetivo de verificar la mejora de los resultados al añadir estas características y determinar el valor que realmente aportan al modelo. En la Tabla 5.8 observamos el rendimiento de esta segunda arquitectura propuesta.

Tabla 5.8: Rendimiento de Arq2.

Arq3_3Steps_OHE_AlllFeatures	
Puntuación F1	0.94853
Penalización	0.26783
% correctas	86.13%
MAE	11.3
MSE	219.9

Se observa que la arquitectura 2, implementada con variables adicionales, presenta un rendimiento superior en comparación con la arquitectura que utiliza únicamente las variables originales. Las ligeras mejoras en la puntuación *F1* se traducen en una mejora global significativa en la predicción de la ruta completa. Por esta razón, todas las pruebas subsecuentes se llevarán a cabo utilizando tanto las variables originales como las variables adicionales correspondientes a los aeropuertos de origen y destino.

Para la siguiente arquitectura se adoptó un método alternativo de codificación de los núcleos. En lugar de utilizar una codificación *one-hot*, se asignó un valor numérico a cada uno de los diferentes núcleos. Esto permite utilizar una sola neurona para la salida, en lugar de una neurona para cada valor de la probabilidad de la clase (núcleo), transformando nuestro problema de una clasificación a una regresión.

En la Tabla 5.9 se presenta el rendimiento obtenido con esta arquitectura. Es relevante destacar que el rendimiento es considerablemente inferior al observado con las codificaciones mediante *one-hot*. Esto evidencia la necesidad de mantener dicha codificación en las arquitecturas futuras.

Tabla 5.9: Rendimiento de Arq3.

Arq3_3Steps_LE_AlllFeatures	
Puntuación F1	0.77532
Penalización	1.13318
% correctas	66.56%
MAE	129.0
MSE	24917.6

Para la arquitectura 4 se propuso la adición de la codificación *one-hot* de los núcleos adyacentes a cada uno de los pasos previos. De esta forma, se busca acotar el espacio de predicción de cada elemento siguiente a aquellos que sean adyacentes al último visitado. Esta consideración aumenta la dimensión de los datos de entrada, incrementando así la complejidad y los tiempos de entrenamiento del modelo. Sin embargo, como se observa en la Tabla 5.10, esto proporciona una mejora significativa en la predicción de rutas. Incluso logra un peor resultado en cuanto al error cuadrático medio del conteo de paso por núcleos, en comparación con lo obtenido en la arquitectura 2, la cual, hasta el momento, ha demostrado ser superior en rendimiento respecto a las demás.

Otra de las mejoras analizadas está relacionada con los pasos por núcleos TMA. Se realizaron las modificaciones detalladas en la metodología para la arquitectura 5, asignando los núcleos TMA en base a las variables region_dep_l y region_des_l. De esta manera, se busca evitar errores y mejorar el procesamiento basado en la región de destino. En la Tabla 5.11 observamos el rendimiento de esta arquitectura. Es

Tabla 5.10: Rendimiento de Arq4.

Arq4_3Steps_OHE_Adjacent_AlllFeatures	
Puntuación F1	0.94871
Penalización	0.25649
% correctas	86.24 %
MAE	11.2
MSE	251.8

evidente que tanto la penalización de la ruta predicha como el porcentaje de rutas predichas correctamente han disminuido. Este fenómeno se debe, en parte, a que, con la implementación realizada, hemos introducido un error sistemático asociado a las rutas reales que no incluyen el núcleo TMA correspondiente cuando el vuelo se origina en su región asignada.

Por lo tanto, se podría suponer que el rendimiento de esta arquitectura sería superior al de las otras si los datos reales no presentaran estos errores en sus rutas.

Tabla 5.11: Rendimiento de Arq5.

Arq5_3Steps_OHE_addTMA_AlllFeatures	
Puntuación F1	0.94327
Penalización	0.27515
% correctas	85.33 %
MAE	7.8
MSE	82.4

La última prueba propuesta se basa en una estructura ideal deseada para una ruta por núcleos, en la que no existan pasos por TMA intermedios en una ruta. En la arquitectura 6, se propuso ignorar los núcleos TMA en la construcción del modelo y añadirlos posteriormente, basándose en la región de inicio o destino del vuelo. Esto reduce la dimensionalidad de la predicción y la complejidad del modelo en general. En la Tabla 5.12, se observa cómo el rendimiento de esta arquitectura, especialmente en la predicción de la ruta completa, disminuye considerablemente. Esto se refleja en un aumento del valor de la penalización y una disminución significativa en el porcentaje de rutas predichas correctamente.

Tabla 5.12: Rendimiento de Arq6.

Arq6_3Steps_OHE_noTMA_AlllFeatures	
Puntuación F1	0.94718
Penalización	0.28686
% correctas	83.83 %
MAE	16.6
MSE	913.2

5.3.1.2. Optimización

En la Tabla 5.13 se presenta un resumen de las pruebas realizadas, donde se pueden apreciar con mayor claridad las diferencias entre cada uno de los modelos planteados y determinar cuál es la arquitectura que resulta conveniente optimizar.

Tabla 5.13: Resumen de pruebas de rutas por núcleos

Modelo	F1	Penalización	% correctas	MAE	MSE
Arq 1	0.90876	0.64068	76.44 %	13.1	296.7
Arq 2	0.94853	0.26783	86.13 %	11.3	219.9
Arq 3	0.77532	1.13318	66.56 %	129.0	24917.6
Arq 4	0.94871	0.25649	86.24 %	11.2	251.8
Arq 5	0.94327	0.27515	85.33 %	7.8	82.4
Arq 6	0.94718	0.28686	83.83 %	16.6	913.2

En base a estos resultados, se propuso el entrenamiento completo de la arquitectura 5, ya que presenta un mejor rendimiento en métricas de gran relevancia para nuestro trabajo. Estas métricas incluyen un menor error en el conteo de pasos por cada núcleo, además del valor intrínseco que aporta al asignar los pasos por los núcleos TMA correspondientes, incluso cuando estos no están presentes en los datos debido a algún error o situación extraordinaria.

Debido a las limitaciones computacionales, la optimización de la red neuronal se enfocó en la modificación y adición de capas para incrementar la complejidad dentro de los límites permitidos. La arquitectura base utilizada para las pruebas consistía en una sola capa LSTM. Las modificaciones propuestas incluyeron la adición de una segunda capa LSTM y la implementación de capas bidireccionales tanto en configuraciones de una como de dos capas LSTM. En la Tabla 5.14 se presenta el rendimiento de cada una de estas configuraciones. Es notable que la arquitectura con dos capas bidireccionales ofrece un buen rendimiento general en las métricas de predicción de ruta y el más alto en las de conteo de paso por núcleos.

Tabla 5.14: Optimización de modelo de ruta de núcleos

Modificaciones	F1	Penalización	% correctas	MAE	MSE
2 LSTM	0.94720	0.26930	85.84 %	9.8	166.0
1 Bidireccional LSTM	0.94472	0.28686	85.0 %	9.9	145.9
2 Bidireccional LSTM	0.94790	0.27881	85.47 %	7.1	88.5

Es importante destacar que, aunque las modificaciones aplicadas no afectaron significativamente la precisión en la construcción de las rutas, el tiempo de entrenamiento aumentó considerablemente. Esta restricción impuso una limitación en la posibilidad de desarrollar arquitecturas de mayor complejidad.

5.3.1.3. Selección del Modelo Final

Para la elaboración del modelo final, se optó por utilizar la arquitectura número 5, la cual está compuesta por dos capas de Long Short-Term Memory (LSTM) bidireccionales y emplea la totalidad de los datos de entrenamiento y pruebas. A pesar de que la precisión del modelo no experimentó un incremento significativo con la adición de capas, la implementación de una red de mayor tamaño permite un mejor aprovechamiento del conjunto total de datos de entrenamiento para este modelo final.

En la Figura 5.5, se ilustra el proceso de entrenamiento, el cual se realizó con un criterio de parada basado en la falta de mejora de la función de pérdida durante al menos 10 épocas consecutivas. Durante este proceso, se almacenaron los pesos

correspondientes a la época con el mejor valor de pérdida. La figura muestra que el valor de la función de pérdida es considerablemente bajo desde la primera época (aproximadamente 0.24), lo cual se atribuye al extenso conjunto de datos empleados en el entrenamiento. A pesar de que se observan fluctuaciones significativas en la pérdida de los datos de validación, esto se debe en gran medida a la escala de la gráfica. El entrenamiento se detiene en una etapa temprana, alcanzando su valor mínimo en la época 18, donde se observa la convergencia de la pérdida tanto para los datos de entrenamiento como para los de validación. Este entrenamiento tardó alrededor de 419 minutos.

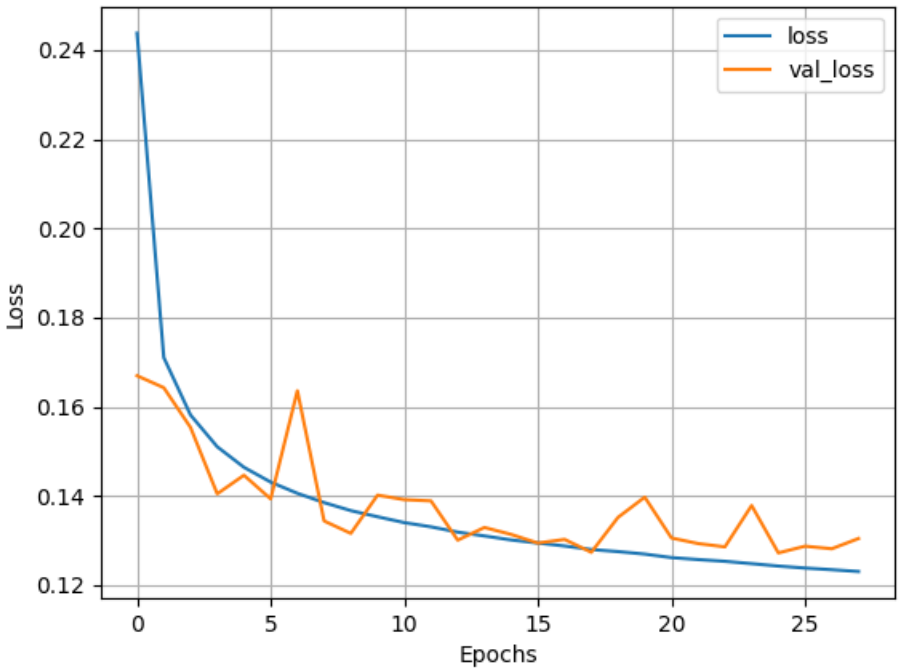

Figura 5.5: Entrenamiento del modelo final de ruta de núcleos

Este nuevo modelo construido fue evaluado utilizando las mismas métricas que en los casos anteriores, pero empleando la totalidad de los datos de prueba para obtener resultados más representativos de la predicción. En la Tabla 5.15 se observa que los valores de las métricas son prácticamente los más altos en lo que respecta a la construcción de las rutas, logrando un 86.37 % de rutas predichas a la perfección, siendo el mejor resultado obtenido hasta ahora. Esto puede deberse, en parte, a que la red presenta el mejor valor de la puntuación *F1* registrado.

Tabla 5.15: Rendimiento del modelo final de ruta de núcleos

Modelo Final	
Puntuación F1	0.94951
Penalización	0.26619
% correctas	86.37 %
MAE	824.6
MSE	997000.8

Respecto a las métricas del conteo de pasos por cada uno de los núcleos, obtenemos valores significativamente altos en comparación con los observados en las pruebas anteriores. Esto se debe a la escala del conteo de pasos, ya que en las pruebas se consideraron únicamente 2,732 vuelos, resultando en un número limitado de pasos por cada núcleo. En esta implementación final, los vuelos considerados para la predicción de ruta fueron 273,328. Para una mejor apreciación del rendimiento en el escenario de conteo de pasos, podemos observar la Figura 5.6, donde se evidencia una gran similitud entre el número de pasos predichos por cada núcleo y el número real.

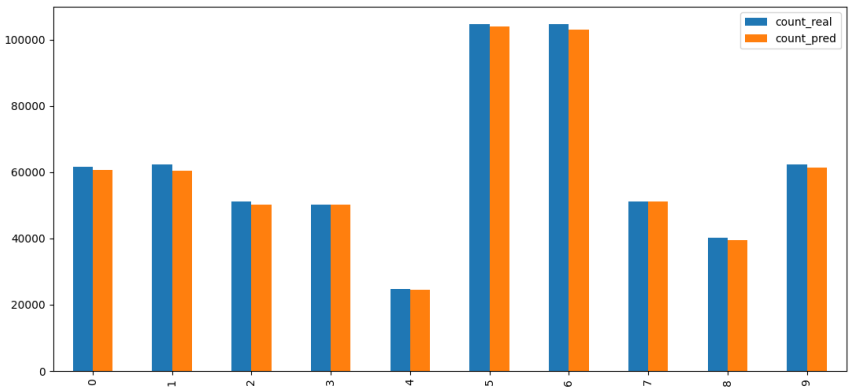

Figura 5.6: Comparativa del conteo de pasos por núcleos

5.3.2. Por Traffic Volume

Para este nivel de granularidad, la complejidad de la predicción aumentó debido al mayor número de elementos posibles en la ruta. Además, el incremento en la predicción del tiempo requiere un tratamiento distinto al de la ruta por núcleos.

Siguiendo el enfoque del modelo de ruta por núcleos, inicialmente se describen las pruebas implementadas en la búsqueda del mejor rendimiento en la predicción de rutas por TV. Posteriormente, se presenta la descripción del modelo final de TV, implementado en base a la validación de dichas pruebas.

Para la propuesta del análisis de los resultados de las pruebas de ruta por núcleos, se busca evitar la construcción de arquitecturas que se hayan demostrado significativamente ineficientes. Esto incluye el uso exclusivo de los datos originales como entrada, la selección del elemento mediante *random.choice*, y la aplicación de un modelo de regresión para la predicción del código numérico del elemento de la ruta.

5.3.2.1. Pruebas

Las pruebas realizadas para este escenario se llevaron a cabo considerando el mismo porcentaje de subconjuntos utilizado en el modelo de rutas: 40 % del conjunto de entrenamiento, 100 % del conjunto de validación y 1 % del conjunto de pruebas.

Las arquitecturas propuestas tienen como objetivo analizar la configuración que proporcione el máximo rendimiento posible del modelo de forma independiente. Poste-

riormente, esta información se utilizará en un modelo que incorpore datos provenientes de la predicción de la ruta por núcleos, con el fin de lograr consistencia en ambas rutas predichas.

Las primeras arquitecturas propuestas consistieron en la evaluación de la predicción mediante dos enfoques: en el primero, se construyó un único modelo que predice tanto el siguiente *Traffic Volume* como el tiempo transcurrido en el mismo; en el segundo enfoque, se desarrollaron dos modelos diferentes que se retroalimentan, prediciendo primero el siguiente *Traffic Volume* y posteriormente el tiempo transcurrido en el mismo.

En la Tabla 5.16, se observa que la arquitectura 1 presenta un rendimiento significativamente inferior en comparación con la arquitectura 2. La arquitectura 2 muestra un rendimiento muy superior en la predicción del *Traffic Volume*, aunque presenta una predicción ligeramente peor del tiempo transcurrido. No obstante, esta ligera disminución en la precisión de la predicción del tiempo no es comparable a la notable mejora en la predicción del *Traffic Volume*.

Tabla 5.16: Comparación Arq1. Vs. Arq2.

	Traffic Volume					**Tiempo**	
Modelo	**F1**	**Penalización**	**% correctas**	**MAE**	**MSE**	**MAE**	**MSE**
Arq 1	0.443	3.35598	1.75 %	68.89	7203.2	109.39	38971.7
Arq 2	0.860	1.58681	62.52 %	5.69	57.12	114.48	36944.7

Las pruebas anteriores se construyeron utilizando la misma mecánica que en el caso de núcleos, considerando tres pasos anteriores para predecir el siguiente. Para la arquitectura 3 se propuso aumentar el número de pasos previos. El rendimiento de esta arquitectura lo podemos observar en la Tabla 5.17, donde se observa un rendimiento muy similar al de la arquitectura 2, con una ligera mejora en los valores de penalización y en la predicción del tiempo.

Tabla 5.17: Rendimiento de Arq3.

Arq3_4Steps_AllOHE_Times		
Traffic Volume	Puntuación F1	0.86002
	Penalización	1.49259
	% correctas	62.85 %
	MAE	5.93
	MSE	60.62
Tiempo	MAE	114.44
	MSE	36927.71

En la arquitectura 4, se propuso un modelo eliminando los llamados *short-crossings*, descartando aquellos cuya permanencia no superara los 2 minutos en su paso por un *Traffic Volume*. Los resultados de esta arquitectura se pueden observar en la Tabla 5.18. Estos resultados, en varios aspectos como la predicción del tiempo y la penalización de las rutas, incluso superan a los de las arquitecturas anteriores. Reflejan que, aunque el modelo puede incurrir en errores, estos son menos pronunciados, lo que se traduce en una mayor precisión y consistencia en las predicciones. Además, se logra una mejora inherente en los resultados deseados del modelo al no considerar

escenarios con *short-crossings* que no serían relevantes en la validación requerida del modelo.

Tabla 5.18: Rendimiento de Arq4.

Arq4_4Steps_AllOHE_Times_NoShortCrossing		
Traffic Volume	Puntuación F1	0.86186
	Penalización	1.43270
	% correctas	61.1%
	MAE	100.53
	MSE	7.21
Tiempo	MAE	111.95
	MSE	36201.8

La arquitectura 5 propuso un enfoque del tratamiento de los TMA, un poco similar al abordado en el modelo de ruta de núcleos. En la Tabla 5.19 se presentan los resultados de esta arquitectura. Es importante destacar que estos resultados son ligeramente mejores que los de las versiones anteriores. Aunque las diferencias en precisión son mínimas, lo que resulta relevante en estas dos últimas arquitecturas son las mejoras intrínsecas que proveen, permitiendo una mejor adaptación a las futuras validaciones de interés de DELFOS.

Tabla 5.19: Rendimiento de Arq5.

Arq5_4Steps_AllOHE_Times_NoShortCrossing_noTMA		
Traffic Volume	Puntuación F1	0.86416
	Penalización	1.54913
	% correctas	61.05%
	MAE	71.74
	MSE	6.25
Tiempo	MAE	106.95
	MSE	32023.32

Hasta este punto, hemos propuesto arquitecturas que ofrecen el mejor rendimiento en la predicción de rutas por *Traffic Volume*. Sin embargo, estos modelos no consideran la información de las rutas por núcleos. Para analizar esta situación, podemos comparar las rutas presentes en los datos de paso por núcleos con las rutas predichas por nuestro modelo, transformadas de TV a núcleos.

En la Tabla 5.20 se presentan los resultados de nuestras métricas aplicadas a la ruta por núcleos, comparada con la ruta de TV escalada a núcleos y con la predicha por el modelo final de rutas de núcleos.

Tabla 5.20: Arq5. de TV vs Modelo Final de Núcleos

Modelo	**Penalización**	**% correctas**	**MAE**	**MSE**
Arq. 5	0.32080	81.36%	10.9	335.3
Modelo Núcleos	0.26619	86.37%	8.2	99.7

La menor precisión observada en la ruta de núcleos del modelo de TV se traduce directamente en inconsistencias con la predicción del modelo de núcleos. Por lo tanto,

se propusieron nuevos enfoques que consideren los resultados de la predicción de la ruta por núcleos para este modelo de TV.

La arquitectura 6 propuso construir dos modelos de predicción: uno para la ruta que abarca el núcleo de Canarias y otro para la parte peninsular. Estos modelos se utilizarían en base a la predicción por núcleos y sus resultados se concatenarían según dicha predicción.

La arquitectura 7, en cambio, buscó adoptar completamente la ruta de núcleos, creando un modelo para cada uno de los núcleos que contienen más de un *Traffic Volume* (TV). Esta última propuesta es sumamente arriesgada, ya que se debe considerar el error heredado del modelo de núcleos, incrementándolo con el error que producirán los propios modelos de TV.

En la Tabla 5.21 se presenta el rendimiento de las arquitecturas 6 y 7 en la predicción de la ruta de TV, utilizando el modelo creado en la arquitectura 5 para los tiempos. Según nuestras métricas, se puede observar que la arquitectura 6 supera a la arquitectura 7 en rendimiento. No obstante, ninguna de las dos arquitecturas alcanza el mejor rendimiento observado en las pruebas anteriores. Cabe destacar que, aunque la arquitectura 7 no ofrece el mejor rendimiento, su desempeño no se aleja significativamente del mejor registrado. Además, esta arquitectura elimina completamente las inconsistencias que podrían existir entre las rutas predichas por los modelos de núcleos y TV, lo cual es un aporte de gran relevancia para nuestro trabajo.

Tabla 5.21: Comparación Arq6. Vs. Arq7.

Modelo	**F1**	**Penalización**	**% correctas**	**MAE**	**MSE**
Arq 6	0.87887	1.48049	60.91 %	5.51	57.37
Arq 7	0.88554	1.53901	59.54 %	7.07	89.33

5.3.2.2. Selección del Modelo Final

La selección de la arquitectura final a desarrollar se basó en el análisis realizado a partir de todas las pruebas y sus respectivas validaciones con las métricas definidas para este fin. En la Tabla 5.22 observamos el valor obtenido por cada una de las arquitecturas en las métricas de validación.

Tabla 5.22: Resumen de pruebas de rutas por *Traffic Volume*

	Traffic Volume					**Tiempo**	
Modelo	**F1**	**Penalización**	**% correctas**	**MAE**	**MSE**	**MAE**	**MSE**
Arq 1	0.4438	3.35598	1.75 %	68.89	7203.2	109.3	38971.7
Arq 2	0.86	1.58681	62.52 %	5.69	57.12	114.4	36944.7
Arq 3	0.86	1.49259	62.85 %	5.93	60.62	114.4	36927.7
Arq 4	0.8618	1.43270	61.1 %	7.21	100.53	111.9	36201.8
Arq 5	0.8641	1.54913	61.05 %	6.25	71.74	106.9	32023.3
Arq 6	0.8788	1.48049	60.91 %	5.51	57.37	106.9	32023.3
Arq 7	0.8855	1.53901	59.54 %	7.07	89.33	106.9	32023.3

Debido al rendimiento demostrado y, en gran medida, para cumplir con los parámetros de validación de DELFOS, se seleccionó la arquitectura 7. Esta arquitectura

incluye automáticamente los pasos por TMA en las rutas basados en su región de origen o destino, aunque esto no se refleje en la totalidad de los datos de entrenamiento, lo cual explica cierto decaimiento en la precisión de las rutas. Además, al construir la ruta basada en la completa predicción de la ruta por núcleos, se eliminan por completo cualquier inconsistencia que otras rutas podrían haber causado, evitando la predicción de *Traffic Volumes* (TV) que correspondan a núcleos inexistentes en la predicción realizada de la ruta por núcleos.

Basados en la optimización realizada en el modelo final de núcleos, se implementa un tipo de arquitectura similar, utilizando dos capas bidireccionales LSTM, además de entrenar con la totalidad de los datos a los diferentes modelos. En la Figura 5.7 observamos la evolución del entrenamiento de cada uno de los modelos destinados a la predicción de *Traffic Volume* en cada uno de los núcleos y constatamos cómo varía su comportamiento en cada caso. Sin embargo, debido a las limitaciones disponibles, no se abordaron técnicas de mejora específicas para cada modelo de forma independiente.

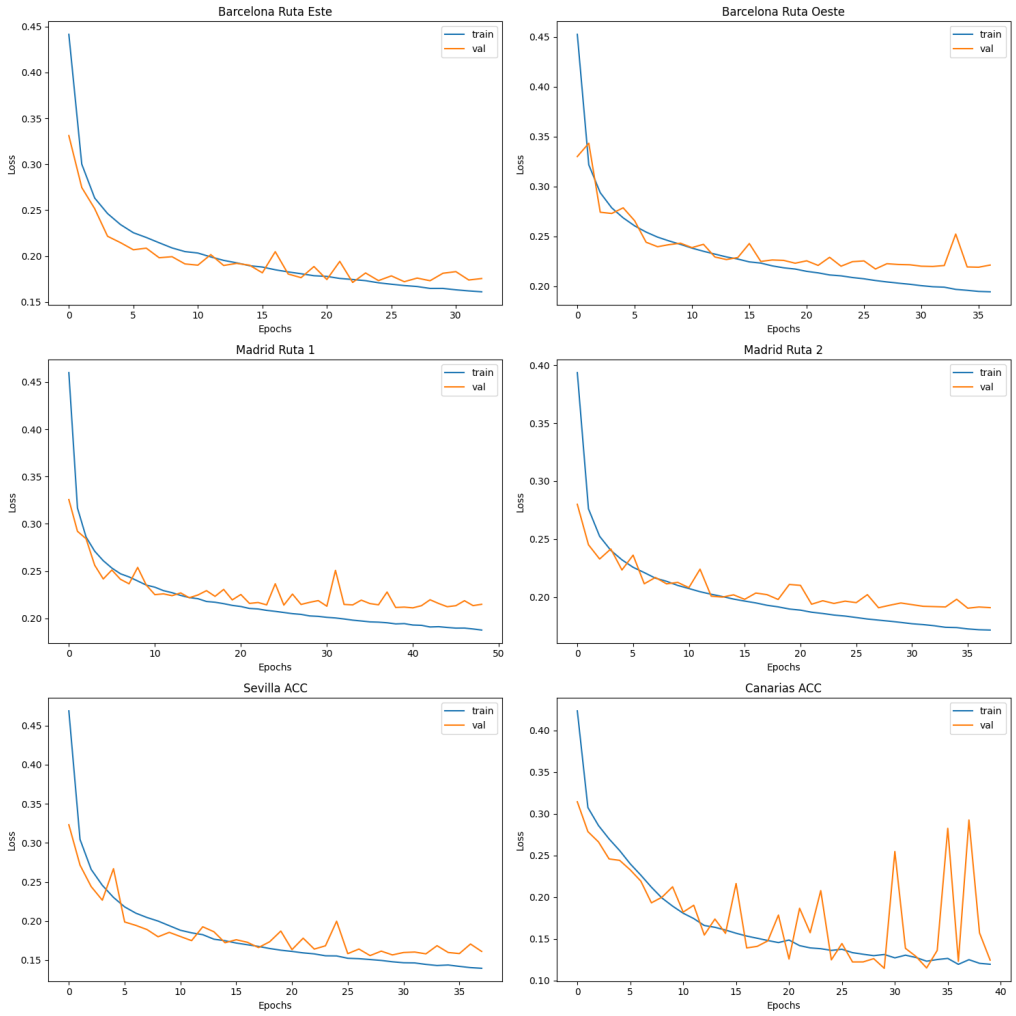

Figura 5.7: Entrenamiento de los modelos de predicción de TV

De la misma forma, en la Figura 5.8, se observa el entrenamiento del modelo de predicción de tiempos, que funciona como auxiliar en la predicción de los *Traffic*

Volume en la ruta. Este modelo presenta un entrenamiento mucho más estable, sin saltos notables en la evolución de su valor de pérdida.

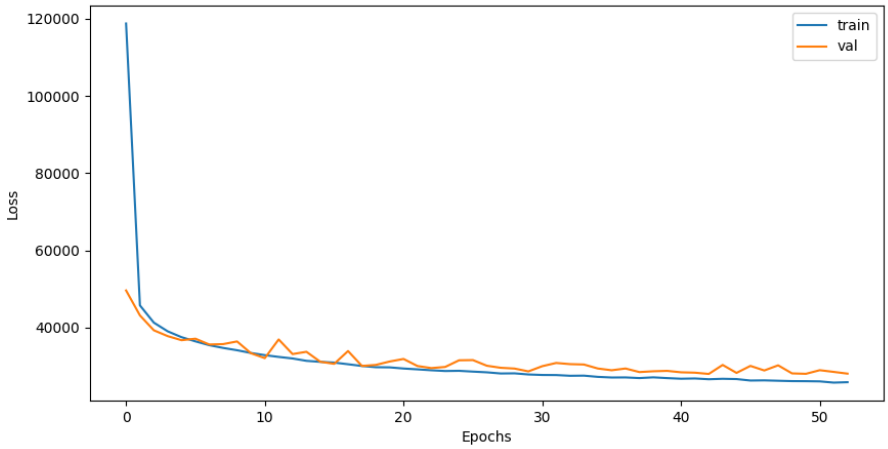

Figura 5.8: Entrenamiento de modelo de predicción de tiempo por TV

Nuestra arquitectura final fue evaluada utilizando las métricas anteriores, pero considerando la totalidad de los datos de prueba para obtener valores más representativos de cada métrica. En la Tabla 5.23 se observa cómo los valores de las métricas mejoraron ligeramente en comparación con pruebas anteriores que utilizaban una porción de los datos, alcanzando un rendimiento muy superior considerando las limitaciones computacionales disponibles y la complejidad de la tarea.

Tabla 5.23: Rendimiento del modelo final de ruta de *Traffic Volume*

Modelo Final		
Traffic Volume	Puntuación F1	0.90120
	Penalización	1.55924
	% correctas	60.04 %
	MAE	1235.18
	MSE	123148148.14
Tiempo	MAE	97.85
	MSE	27969.7

Respecto a las métricas del conteo de pasos por cada uno de los *Traffic Volume*, de igual forma que en el caso visto en el modelo de ruta de núcleos, observamos unos valores significativamente diferentes a los vistos en las pruebas debido al uso de la totalidad de los datos. Para una mejor apreciación del rendimiento en el escenario de conteo de pasos, podemos observar la Figura 5.9, donde se evidencia una gran similitud entre el número de pasos predichos por cada *Traffic Volume* y el número real.

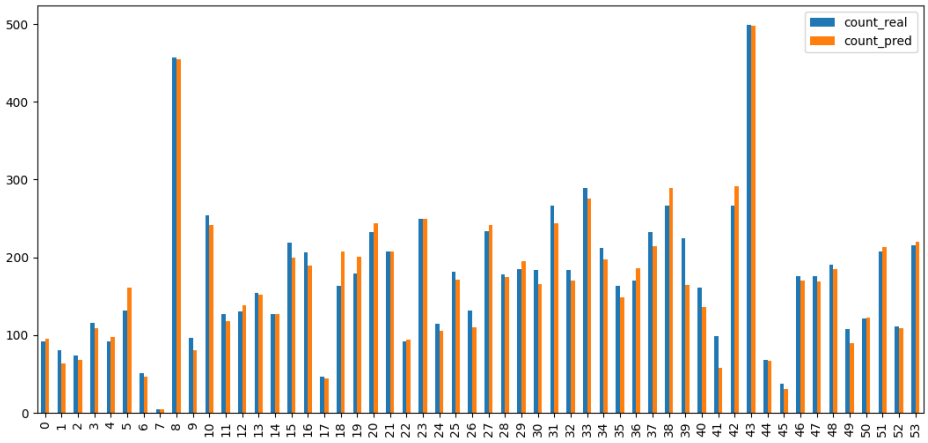

Figura 5.9: Comparativa del conteo de pasos por núcleos

5.3.3. Por Plan de Vuelo

El plan de vuelo constituye el nivel de mayor detalle disponible sobre la ruta seguida por un vuelo en el espacio aéreo. La precisión que este nivel describe representa el principal desafío en su predicción. En esta sección abordaremos todas las pruebas realizadas en la construcción de este modelo, considerando el análisis efectuado en los niveles anteriores para evitar pruebas innecesarias.

Una consideración general aplicada a la construcción de este modelo fue la definición del número de pasos previos, pues su elección no fue basada en el promedio de pasos por WP de una ruta, pues las limitaciones computacionales no permitieron aumentar el valor utilizado de 4.

5.3.3.1. Pruebas

La construcción de la ruta se abordó mediante un enfoque de regresión, centrándose en la predicción de la latitud, longitud y el nivel de vuelo en cada paso de la ruta. Los conjuntos definidos para la construcción de los modelos (entrenamiento, validación y pruebas) en este nivel fueron reducidos considerablemente debido al gran número de *WayPoints* que abarca cada una de las rutas. Las principales métricas están determinadas por las predicciones realizadas sobre el conjunto de pruebas, el cual, en este escenario, cuenta con 2,599 detalles de vuelos.

Las primeras cuatro arquitecturas definidas en la metodología fueron propuestas considerando la eficiencia en el entrenamiento de los modelos. Inicialmente, se propuso una arquitectura que abarque la predicción de los tres valores objetivo, así como varias opciones para la división de esta tarea, de manera que se logre una mayor precisión. En la Tabla 5.24 se presenta el rendimiento de cada una de estas arquitecturas, donde se observa una gran desviación en las predicciones y una precisión prácticamente nula en la predicción de la ruta completa.

Una de las consideraciones abordadas para mejorar los valores de predicción observados fue la definida en la arquitectura 5, la cual contempla una modificación

Tabla 5.24: Pruebas de rutas por *WayPoints*

Modelo	Latitud		Longitud		Nivel de Vuelo		Penaliz.	% correctas
	MAE	MSE	MAE	MSE	MAE	MSE		
Arq 1	2.6	26.7	1.4	4.5	9.8	796.5	12.766	0.0%
Arq 2	5.8	106.7	0.9	4.2	48.5	6860.4	12.240	0.0%
Arq 3	1.3	15.8	0.5	1.0	33.6	3249.7	11.091	0.0%
Arq 4	1.1	14.0	0.3	0.6	23.8	2118.6	10.064	0.12%

del algoritmo de construcción de la ruta. En lugar de utilizar los valores de latitud y longitud predichos para los pasos previos, se emplearon los valores del *WayPoint* elegido.

Para la implementación de esta arquitectura, se utilizaron los modelos correspondientes a la arquitectura que dio los mejores resultados en las pruebas anteriores, es decir, la arquitectura 4. Se llevó a cabo la predicción de las rutas considerando esta nueva modificación. En la Tabla 5.25, se observan las diferencias en las penalizaciones promedio, definidas para nuestra métrica, entre las rutas predichas de las arquitecturas 4 y 5 en referencia a las rutas reales.

Podemos notar que la predicción utilizando como pasos previos la latitud y longitud previstas logra al menos un mínimo porcentaje de precisión, al contrario de realizar las correcciones de estos valores en la selección del *WayPoint*. Esto, por sí mismo, descartaría la utilidad de realizar estas correcciones; sin embargo, el valor de la penalización es significativamente menor al realizar estas correcciones. La arquitectura 4 centra sus errores principalmente en la predicción de elementos que no existen en la ruta original y, en menor medida, en el aumento del número de elementos de la ruta. Por su parte, la arquitectura 5 concentra sus errores principalmente en la omisión de núcleos, presentando rutas mucho más cortas que las reales. El hecho de predecir muy pocos elementos explica que, a pesar de poseer una menor penalización por la predicción de un elemento que no existe en la ruta real, no consiga predecir ni una sola ruta completa correctamente.

Tabla 5.25: Métrica Arq4 y Arq5 Vs Real

Modelo	P. omisión	P. extra	P. cambio	P. posición	P. total	% correctas
Arq4.	1.9457	2.3967	6.6745	0.075	10.0646	0.12%
Arq5.	5.4948	0.676	3.3632	0.0088	9.54290	0.0%

Otra validación para comparar el rendimiento de las rutas predichas, dado que los resultados a nivel de *WayPoints* de nuestra predicción cuentan con una precisión sumamente baja, es realizar una comparativa con un nivel de más baja granularidad. En nuestro caso, realizamos una validación de la ruta predicha pero a nivel de *Traffic Volume* (TV), es decir, escalamos la ruta real y la predicha de *WayPoints* a TV y aplicamos nuestra métrica desarrollada.

Los resultados de esta validación muestran que, con la arquitectura 4 se consigue un 9.35% de rutas predichas correctamente a nivel de TV, y un 2.04% con la arquitectura 5. Por lo tanto, podemos determinar que la corrección de los valores de latitud y longitud en nuestro algoritmo de construcción de rutas no resulta efectiva.

5.3.3.2. Selección del Modelo Final

En base a la sección anterior de pruebas y al análisis desarrollado en ella, se determinó que la mejor arquitectura consistía en un modelo independiente para cada una de nuestras variables objetivo: Latitud, Longitud y Nivel de Vuelo. Cada modelo utiliza estas tres variables como entradas correspondientes a los pasos previos. El algoritmo de construcción de ruta emplea cada una de las predicciones de los modelos como nuevos pasos previos para la predicción del siguiente paso.

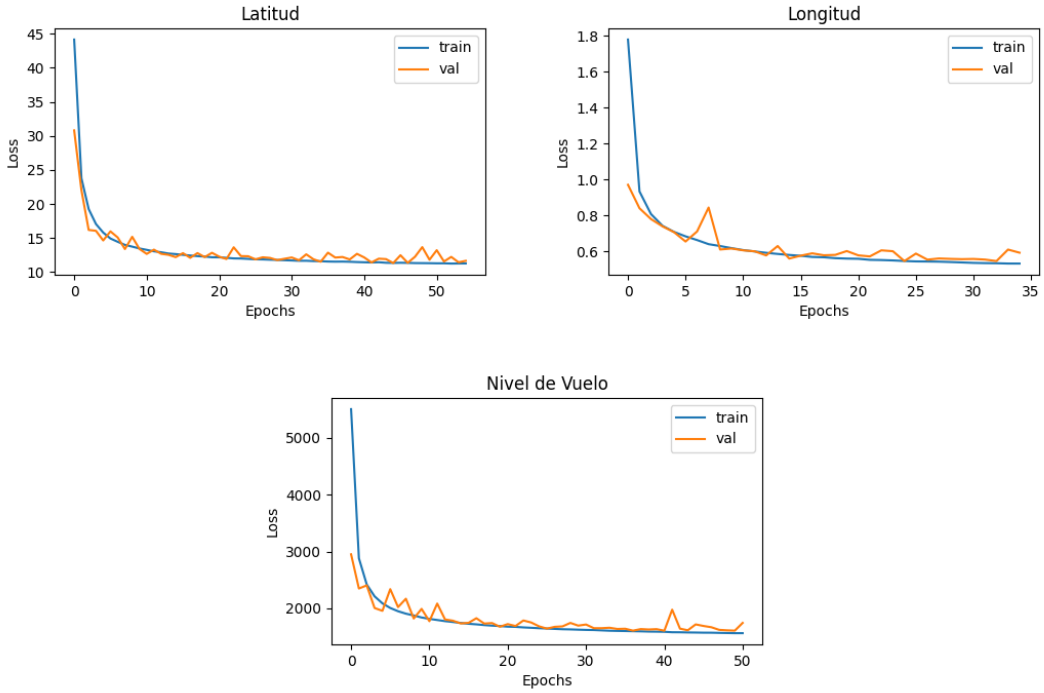

Figura 5.10: Entrenamiento de los modelos de predicción de WP

El entrenamiento de estos modelos, a diferencia de los casos anteriores, no pudo llevarse a cabo utilizando la totalidad de los datos disponibles debido a las limitaciones computacionales del proyecto. Con los recursos disponibles, que incluían 16 GB de RAM, una CPU 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz y una GPU Intel(R) Iris(R) Xe Graphics, se empleó el 30 % de los datos (2,070,110 registros) para el entrenamiento y el 50 % de los datos (259,881 registros) para la validación.

En la Figura 5.10, se presenta el entrenamiento de cada una de las redes recurrentes desarrolladas, utilizando dos capas bidireccionales en su arquitectura.

En la Tabla 5.26, se observa un rendimiento significativamente superior en comparación con las pruebas previas, en términos de la predicción de los valores de cada una de las variables, lo cual implica una mayor aproximación a la ruta real. No obstante, a pesar de esta mejora en la precisión de las predicciones, la traducción de estas a *WayPoints* no mostró un avance en comparación con el rendimiento observado en las pruebas.

El desarrollo de este modelo resultó ser de una gran complejidad, superando las capacidades computacionales disponibles. Esta limitación impidió la implementación

Tabla 5.26: Rendimiento del modelo final de ruta de *WwayPoints*

Modelo Final		
Latitud	MAE	0.79
	MSE	11.32
Longitud	MAE	0.28
	MSE	0.54
Nivel de Vuelo	MAE	17.78
	MSE	1605.64
Penalización		10.692
% Correctas		0.08 %

de arquitecturas más complejas y el uso de una mayor cantidad de datos disponibles, lo que podría haber conducido a un rendimiento superior, equiparable al de los modelos de los niveles anteriores. A pesar de estos desafíos, los resultados obtenidos demuestran un avance significativo en comparación con enfoques menos complejos, aunque se reconoce que un mayor rendimiento podría haberse logrado con recursos adicionales.

Capítulo 6

Conclusiones y Trabajos Futuros

6.1. Conclusiones

En este trabajo se analizaron diversos modelos de inteligencia artificial con el objetivo de estimar la afluencia de tráfico en el espacio aéreo español, la cual es crucial para mejorar la gestión y la planificación del tráfico aéreo, garantizando así la seguridad y eficiencia operativa. Además, permite una optimización de recursos y una mejor respuesta ante fluctuaciones en la demanda de vuelos.

La primera tarea en este trabajo fue el desarrollo de un modelo de clasificación para determinar si un vuelo realiza un sobrevuelo por el espacio aéreo español. Nuestra propuesta final fue un modelo de *Random Forest* que logró niveles de precisión sorprendentemente altos para cualquier tarea de clasificación en general, alcanzando, por ejemplo, una puntuación *F1* de 0.97983. Este modelo no solo ofrece una alta precisión, sino que también aporta información adicional valiosa, ya que la metodología estadística actual únicamente predice el número de vuelos sin diferenciar cuáles de ellos realizan el sobrevuelo. Dado que este modelo fue el primero en ser desarrollado, se pudo validar en un contexto real, observándose una mejora significativa en comparación con las predicciones obtenidas utilizando la metodología actual. Esta validación resalta la eficacia de nuestro modelo y su capacidad para mejorar el proceso de estimación de tráfico aéreo.

Durante la implementación de los modelos de rutas, se encontraron diversos problemas debido a la complejidad de la predicción, ya que se buscaba predecir la ruta seguida por un vuelo en el espacio aéreo español a diferentes niveles de detalle. A mayor nivel de detalle, mayor era la complejidad del problema. La ruta de más alto nivel está constituida por 10 núcleos elementales diferentes. Para abordar este desafío, se propuso un modelo de red recurrente para la clasificación, el cual, dados tres pasos anteriores, predice el siguiente elemento de la ruta. Esta propuesta brindó resultados muy satisfactorios, alcanzando aproximadamente un 86 % de rutas completas predichas correctamente.

Estos resultados, además de su gran precisión, cuentan con grandes ventajas y mejoras respecto a la metodología actual que se basa en el conteo de vuelos que pasarán por cada uno de los núcleos, pues aunque el modelo propuesto no alcanza el mismo nivel de precisión en este conteo, ofrece un detalle completo de la ruta de cada vuelo de forma individual. Esto proporciona una gran cantidad de información adicional

que podría complementar, y no necesariamente reemplazar, las predicciones actuales en DELFOS, enriqueciendo así el análisis y la toma de decisiones.

A medida que se profundiza en niveles de mayor detalle, la complejidad aumenta exponencialmente y la precisión se ve gravemente afectada. A nivel de *Traffic Volume*, se encuentran 56 elementos a seleccionar, lo que, junto con ciertas restricciones y limitaciones implementadas, otorgaba una precisión de aproximadamente 60 % de rutas predichas correctamente. Esto confirmó la viabilidad de la tarea de clasificación, a partir de la cual se implementaron diversas modificaciones con el objetivo de simplificar la tarea y utilizar la información de la predicción de la ruta de núcleos para mejorar la predicción en este nuevo nivel. Más importante aún, estas modificaciones eliminaron posibles incongruencias en las predicciones de los diferentes modelos, pues nuestros modelos debe evitar, en la medida de lo posible un escenario en el que se predice una ruta por *Traffic Volume* que, al escalar a núcleos, difiera de la ruta predicha por el modelo de núcleos. La solución propuesta fue segmentar el problema de clasificación, construyendo diferentes modelos para la predicción de la subruta de *Traffic Volume* seguida dentro de cada núcleo predicho por el modelo de núcleos para un vuelo. Los resultados de esta propuesta fueron sumamente alentadores. A pesar de heredar el error del modelo de núcleos, la precisión se mantuvo cerca del 60 % de rutas predichas correctamente, un valor muy similar al 62 % conseguido con el modelo independiente, pero eliminando por completo las posibles inconsistencias entre las predicciones.

El último nivel abordado, el de *WayPoints*, resultó ser de una gran complejidad debido a la cantidad de elementos, lo que hizo inviable una clasificación directa. Por esta razón, el enfoque se centró en la predicción de la latitud y longitud, y en base a esta información, seleccionar los elementos correspondientes utilizando el menor error cuadrático medio entre la predicción y la posición de los elementos. Este enfoque cambió de un modelo de clasificación a uno de regresión. Sin embargo, debido a la gran proximidad entre los valores de latitud y longitud de cada *WayPoint* (que suelen diferenciarse únicamente en su segundo decimal), esta tarea resultó en una precisión extremadamente baja. En conclusión, mientras que los niveles superiores lograron resultados prometedores, el nivel de *WayPoints* requiere enfoques adicionales y recursos más avanzados para lograr una precisión aceptable.

Finalmente, podemos concluir que los modelos de inteligencia artificial (IA) son de gran relevancia para abordar nuestra problemática. En los niveles de menor detalle, como el de sobrevuelos, los modelos desarrollados demostraron ser significativamente superiores a las metodologías estadísticas en casi todos los aspectos. No obstante, a medida que se profundiza en niveles de mayor detalle, la tarea se torna extremadamente compleja. En un nivel intermedio (núcleos y *Traffic Volume*), se obtuvieron resultados notablemente buenos. Aunque estos modelos podrían no superar el enfoque estadístico en términos del número de pasos por cada elemento, proporcionan una gran cantidad de información y detalle individual de cada vuelo. El nivel más bajo, el de *WayPoints*, resultó prácticamente inabarcable con las técnicas propuestas en este trabajo. No obstante, se debería considerar un cambio en el enfoque de validación de su predicción. Aunque estos modelos pueden no acertar de manera eficiente en los elementos específicos, podrían generar una ruta lo suficientemente próxima que funcione como una estimación útil para complementar otros niveles de granularidad. En resumen, los modelos de IA desarrollados ofrecen mejoras significativas en los niveles de menor y mediana granularidad, proporcionando información

detallada y valiosa. Para los niveles más detallados, es necesario explorar enfoques alternativos y métodos de validación que puedan aprovechar la proximidad de las predicciones para complementar y enriquecer las metodologías actuales.

6.2. Trabajos Futuros

Dada la extensión del trabajo realizado, existen muchas opciones de trabajos futuros. Si bien los modelos de rutas aún posen un amplio margen de mejora, primero nos centraremos en las debilidades del modelo de sobrevuelos.

Aunque el modelo de sobrevuelos ha obtenido resultados notables, aún existen ciertos errores que deben considerarse. Uno de los principales problemas es la clasificación de los casos atípicos, ya que hay aerolíneas o grupos de aeropuertos de origen y destino sumamente desbalanceados, los cuales el modelo tiende a clasificar casi en su totalidad como no correspondientes a sobrevuelos. Este es un escenario problemático, ya que provoca que el número de sobrevuelos predichos, aunque bastante cercano al valor real, generalmente se acerque a este valor desde abajo. No obstante, en nuestro contexto, sería ideal una ligera sobreestimación de sobrevuelos, ya que al destinar recursos para una determinada carga, es preferible que estos sean excedentes a que resulten insuficientes.

Respecto a los modelos de rutas, tanto los modelos por núcleos como *Traffic Volume* pueden presentar un mayor rendimiento contando con los recursos computacionales adecuados mediante la construcción de arquitecturas recurrentes de mayor complejidad, tarea en la que nos vimos sumamente limitados en este trabajo.

El nivel de *WayPoints* ofrece una gran posibilidad de análisis, pues sus datos son los más dificiles de abarcar y debe ser necesario una mayor especificación de las validaciones esperadas para dichas predicciones para poder determinar la utilidad del mismo que, de la misma forma que con los niveles anteriores, arquitecturas más complejas ayudarían a llegar a un nivel mucho mayor de aproximación.

Un poco independientemente de la inteligencia atificial, se considera de gran relevancia un futuro análisis y transformación de los datos de rutas, pues para el presente trabajo se proporcionó un conjunto de datos para cada nivel y la correspondencia entre los diferentes niveles de los elementos, estos poseían inconsistencias intrínsecas en sus rutas, pues analizando los datos a nivel de *Traffic Volume* se encontró gran cantidad de registros que, al pasar los TV a núcleos, esta ruta no correspondía a la encontrada en el conjunto de datos a nivel de núcleos.

Bibliografía

[1] International Civil Aviation Organization, *Air Traffic Management Documentation*, ICAO, 2024. [Online]. Available: https://www.icao.int

[2] EUROCONTROL, *ATFCM Operations Manual*, maint-2 ed., EUROCONTROL Network Manager, May 2024, available at: https://www.eurocontrol.int/publication/atfcm-operations-manual. [Online]. Available: https://www.eurocontrol.int/publication/atfcm-operations-manual

[3] J. G. Busquets, E. Alonso, and A. Evans, "Application of data mining in air traffic forecasting," in *15th AIAA Aviation Technology, Integration, and Operations Conference*. AIAA Press, 2015, copyright American Institute of Aeronautics and Astronautics 2015. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/6.2015-2732

[4] G. Gui, Z. Zhou, J. Wang, F. Liu, and J. Sun, "Machine learning aided air traffic flow analysis based on aviation big data," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 4817–4826, 2020.

[5] R. Marcos, O. G. Cantú, and R. Herranz, "A machine learning approach to air traffic route choice modelling," *CoRR*, vol. abs/1802.06588, 2018. [Online]. Available: http://arxiv.org/abs/1802.06588

[6] Z. C. Lipton, "A critical review of recurrent neural networks for sequence learning," *CoRR*, vol. abs/1506.00019, 2015. [Online]. Available: http://arxiv.org/abs/1506.00019

[7] M. Kleehammer, *pyodbc: Python SQL Driver*, 2023, retrieved June 12, 2024. [Online]. Available: https://github.com/mkleehammer/pyodbc/wiki

[8] T. P. D. Team, *pandas: Powerful data structures for data analysis, time series, and statistics*, 2023, retrieved June 12, 2024. [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/

[9] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, 2020, retrieved June 12, 2024. [Online]. Available: https://numpy.org/doc/stable/

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos,

D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, *Scikit-learn: Machine Learning in Python*, 2011, retrieved June 12, 2024. [Online]. Available: https://scikit-learn.org/stable/user_guide.html

[11] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*, 2015, retrieved June 12, 2024. [Online]. Available: https://www.tensorflow.org/guide

[12] T. M. Authors, *MLflow: A Machine Learning Lifecycle Platform*, 2018, retrieved June 12, 2024. [Online]. Available: https://mlflow.org/docs/latest/index.html

[13] T. M. Hope, "Chapter 4 - linear regression," in *Machine Learning*, A. Mechelli and S. Vieira, Eds. Academic Press, 2020, pp. 67–81. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128157398000043

[14] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013.

[15] D. T. Larose and C. D. Larose, "k-nearest neighbor algorithm," 2014.

[16] M. Minsky, "Steps toward artificial intelligence," *Proceedings of the IRE*, vol. 49, no. 1, pp. 8–30, 1961.

[17] S. B. Kotsiantis, I. Zaharakis, P. Pintelas *et al.*, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, no. 1, pp. 3–24, 2007.

[18] Y. Freund, R. E. Schapire *et al.*, "Experiments with a new boosting algorithm," in *icml*, vol. 96. Citeseer, 1996, pp. 148–156.

[19] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[20] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[21] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[22] I. Sutskever, *Training recurrent neural networks*. University of Toronto Toronto, ON, Canada, 2013.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[25] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2.