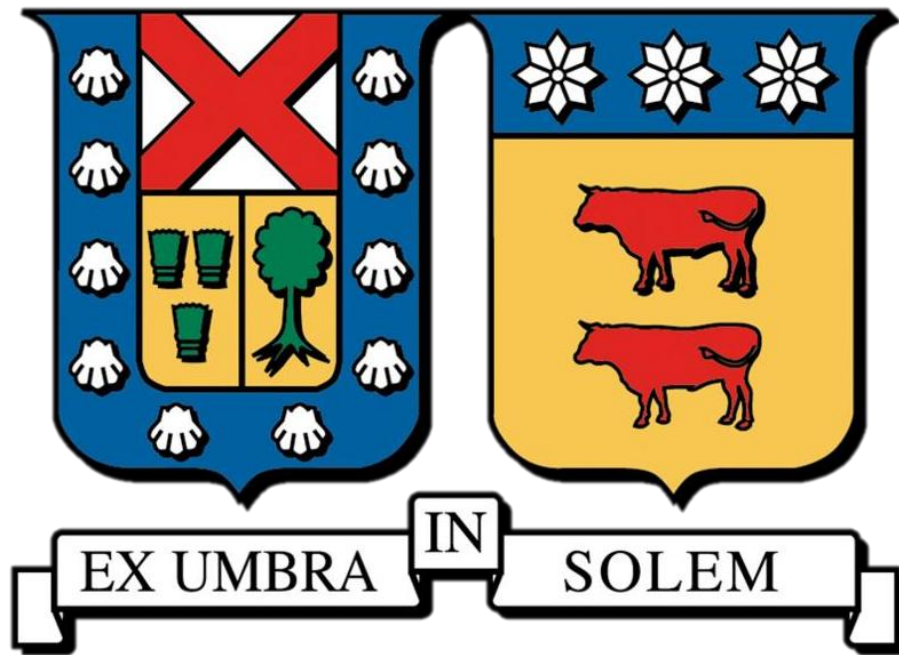


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE AERONÁUTICA VITACURA –
CHILE



PREDICCIÓN DE LA PUNTUALIDAD DEL AEROPUERTO
ARTURO MERINO BENITEZ UTILIZANDO ALGORITMOS DE
MACHINE LEARNING.

PABLO ALFONSO MOYANO ROZAS

MEMORIA PARA OPTAR AL TITULO DE
INGENIERO EN AVIACIÓN COMERCIAL

PROFESOR GUÍA	: Sr. VICTOR POBLETE GÓMEZ
PROFESOR CORREFERENTE	: Sr. ROBERTO SOLIS

ENERO 2026



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: PREDICCIÓN DE LA PUNTUALIDAD DEL AEROPUERTO ARTURO MERINO BENITEZ UTILIZANDO ALGORITMOS DE MACHINE LEARNING.

Nombre del candidato(a): Pablo Alfonso Moyano Rozas Carrera / Grado: Ingeniería en aviación comercial Campus: Santiago Vitacura; Departamento: Aeronáutica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Victor Poblete Gómez, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente DEJO CONSTANCIA que:

He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.

El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo NO contiene información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo CONTIENE información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 09/01/2026 ; Firma:

Estudiante o Candidato(a):

Fecha: 09/01/2026 ; Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.



Agradecimientos

“La humildad no consiste en ser pobre o ser humillado, es la capacidad lógica de seguir aprendiendo” (C. Rozas 2025). Estoy en deuda con la vida; mi pasión es mi deber y se debe cumplir. La disciplina es la piedra angular de cualquier mérito personal. Agradezco a mi familia por hacerme el hombre que soy hoy, desde los pequeños hábitos hasta los pilares fundamentales de mi ser. Queda un camino largo por recorrer y, sin duda, el frenesí rítmico resonará con los gigantes del cielo hasta el fin de los tiempos. Agradezco la fe incondicional que cada uno de ustedes puso en mí. Y por último, me agradezco a mí mismo por perseverar, creen en mí y, sobre todo, por nunca perder el hambre de conocimiento.



Resumen Ejecutivo

El crecimiento exponencial de la información y la interconexión global plantan desafíos significativos para la infraestructura computacional, especialmente en las áreas de almacenamiento, procesamiento y el entendimiento de conjuntos de datos multivariantes. La ausencia de metodologías robustas y eficientes en análisis de estos datos imposibilitan la extracción de conocimiento y, por ende, la toma de decisiones estratégicas.

La presente memoria aborda esta problemática centrándose en el sector aeroportuario, aplicando herramienta de aprendizaje automático (*Machine Learning*) con el objetivo de la extracción de conocimiento de los datos operacionales, mejorar la puntualidad y la asignación de puertas de embarque del aeropuerto Arturo Merino Benítez (SCL) con estos algoritmos.

El *Machine Learning* ofrece una solución efectiva para la identificación de patrones complejos y el modelado de comportamiento de corto y largo plazo, mediante el entrenamiento de algoritmos predictivos. Esta capacidad resulta crucial para disminuir los riesgos asociados a la incertidumbre operativa dentro de la gestión aeroportuaria.

Se evaluó un conjunto de algoritmos de *Machine Learning*, incluyendo Redes Neuronales Multicapa (MLP), Long Short-Term Memory (LSTM), K-Nearest Neighbors (KNN), Random Forest, Support Vector Regression (SVR) y la Red Neuronal Multicapa con función de pérdida de cuartil. El mejor desempeño predictivo se obtuvo con una Red Neuronal Multicapa con función de pérdida de cuartil.

Este modelo alcanza los siguientes indicadores de rendimiento:

- Error Cuadrático Medio (MSE): 103.8.
- Error Absoluto Medio (MAE): 5.58 minutos.
- Coeficiente de Determinación (R^2): 0.809.

además, en la prueba de validación de los itinerarios del mes de mayo, el modelo obtuvo un



sólido resultado con un error absoluto medio (MAE) de 6,24 minuto por vuelo, logrando capturar los patrones de los datos. Sin embargo, los residuales obtuvieron una media de 2.20 minutos, reflejando un ligero sesgo positivo. Esto confirma que el modelo no tiene errores sistemáticos. No obstante, la desviación estándar se sitúa en ± 30 minutos, lo que sugiere una alta imprecisión o que los errores individuales son ampliamente inconsistentes.



Abstract

The exponential growth of information and global interconnection presents significant challenges for computational infrastructure, especially in the areas of storage, processing, and the effective understanding of multivariate datasets. The lack of robust and efficient methodologies for analyzing this data hinders the extraction of knowledge and, consequently, strategic decision-making.

This thesis addresses this problem by focusing on the airport sector, applying Machine Learning tools with the objective of improving the punctuality and gate assignment at Arturo Merino Benítez International Airport (SCL).

Machine Learning offers an effective solution for identifying complex patterns and modeling short- and long-term behavior through the training of predictive algorithms. This capability is crucial for mitigating the risks associated with operational uncertainty within airport management.

A set of Machine Learning algorithms was evaluated, including Multilayer Perceptrons (MLP), Long Short-Term Memory (LSTM), K-Nearest Neighbors (KNN), Random Forest, Support Vector Regression (SVR), and the Multilayer Perceptron with a Quantile Loss function. The best predictive performance was achieved using the Multilayer Perceptron with a Quantile Loss function.

This model achieved the following performance indicators:

- Mean Squared Error (MSE): 103.8.
- Mean Absolute Error (MAE): 5.58 minutes.
- Coefficient of Determination (R^2): 0.8.

Furthermore, in the validation test using the May itineraries, the model obtained a solid result with a Mean Absolute Error (MAE) of 6.24 minutes per flight, successfully capturing the underlying data patterns. However, the standard deviation is situated at ± 30 minutes, which suggests high inaccuracy or that the individual errors are widely inconsistent.



Contenido

Agradecimientos	iii
Resumen Ejecutivo	iv
Abstract	vi
Índice de gráficos	ix
Índice de tabla	ix
Índice de figuras	x
Lista de abreviación	xi
Introducción	1
1 Capítulo I: Antecedentes Generales	2
1.1 Justificación del tema	2
1.2 Objetivo General	5
1.3 Objetivo Especifico	5
1.4 Metodología	6
1.5 Alcance.....	8
2 Capitulo II: Estado del Arte.....	9
2.1 Antecedentes	9
2.1.1 Rol de los administradores de aeropuerto y contexto de la puntualidad.....	9
2.1.2 Contexto actual del Arturo Merino Benítez.....	13
2.1.3 Historia y desarrollo del <i>Machine Learning</i>	16
2.1.4 Aplicaciones del <i>Machine Learning</i> en la historia aeronáutica	18
2.2 Marco teórico	24
2.2.1 Puntualidad	24
2.2.2 Aprendizaje de maquina	40
2.2.3 Redes neuronales	43
2.2.4 Modelos de <i>Machine Learning</i>	60
2.2.5 Distribuciones	73
2.2.6 Indicadores y transformaciones	77
2.2.7 Metodología ETL.....	80
2.2.8 Propuesta Metodológica.....	81
3 Capitulo III: Desarrollo	83
3.1 Metodología ETL	83
3.1.1 Acumulación de datos (Extract).....	83
3.1.2 Procesamiento de datos (Transform)	86
3.2 Definición de variables.....	89
3.2.1 Base de datos	89



3.2.2	Variables de entrada.....	93
3.2.3	Variables de salida	95
3.3	Estadística	95
3.4	Selección de herramientas IA.....	98
3.5	Entrenamiento	100
3.6	Resultados	104
3.6.1	Distribución Normal	108
3.6.2	Distribución Log-Normal.....	109
3.6.3	Distribución Gamma.....	110
3.6.4	Distribución Weibull.....	111
3.6.5	Comparación.....	112
3.7	Validación del modelo	113
Conclusiones		117
Bibliografía		120
Anexos		124



Índice de gráficos

Gráfica 2-1: Puntualidad desde 2022 hasta 2025 separado por trimestre de Ámsterdam	10
Gráfica 2-2: Puntualidad desde 2022 hasta 2025 separado por trimestre de Barcelona.	10
Gráfica 2-3: Puntualidad desde 2022 hasta 2025 separado por trimestre de Paris Charles de Gaulle	11
Gráfica 2-4: Puntualidad desde 2022 hasta 2025 separado por trimestre de Frankfurt	12
Gráfica 2-5: Puntualidad desde 2023 hasta 2025 separado por trimestre de Aeropuerto Arturo Merino Benítez	15
Gráfica 2-6: Función Sigmoide	47
Gráfica 2-7: Función tangente hiperbólica.	48
Gráfica 2-8: <i>Gráfico de regresión lineal</i> , ...	68
Gráfica 2-9: Distribución Normal.....	74
Gráfica 2-10: Distribución log-normal	75
Gráfica 2-11: Distribución <i>gamma</i>	76
Gráfica 2-12: Distribución Weibull	77
Gráfica 3-1: histograma de On-time performace.	96
Gráfica 3-2: Resultado de predicción de OTP para el modelo de Random Forest	101
Gráfica 3-3: Resultados de predicción de OTP para el modelo red neuronal multicapa con la función de pérdida de cuantiles	102
Gráfica 3-4: Importancia de las variables para el modelo de red neuronal multicapa con la función de pérdida de cuantiles.....	105
Gráfica 3-5: Importancia de las variables para el modelo de Random Forest	107
Gráfica 3-6: Distribución de OTP predicho vs testeo por Red Neuronal Multicapa con función de pérdida de cuantiles.....	108
Gráfica 3-7: Densidad de OTP respecto a una distribución Normal.....	109
Gráfica 3-8: Densidad de OTP respecto a una distribución Log-Normal	110
Gráfica 3-9: Densidad de OTP respecto a una distribución Gamma	111
Gráfica 3-10: Densidad de OTP respecto a una distribución Weibull	112
Gráfica 3-11: histograma de los datos residuales	116

Índice de tabla

Tabla 1-1: Comparación por trimestre de PAX contra la puntualidad del aeropuerto AMB.....	3
Tabla 2-1: Rendimiento de puntualidad del aeropuerto Arturo Merino Benítez.	14
Tabla 2-2: Comparación de papers relacionados a metodologías de Machine Learning	22
Tabla 2-3: Tabla de P.S.V y PD para tripulantes de vuelo	39
Tabla 2-4: códigos IATA Delay	40
Tabla 3-1: Descripción de las variables usadas para entrenar los modelos de machine learning	90
Tabla 3-2: Variables temporales de la base de datos	90
Tabla 3-3: Variables categóricas de las bases de datos.....	91
Tabla 3-4: Variables climatológicas de la base de datos	92
Tabla 3-5: Variable operacionales de la base de datos	92
Tabla 3-6: Descripción de las variables de entrada para entrenar los modelos de Machine Learning	94
Tabla 3-7: Descripción de la variable de salida	95
Tabla 3-8: Comparación de parámetros entre distribuciones.....	113
Tabla 3-9: Extracto de vuelos del mes de Mayo.....	115



Índice de figuras

Figura 1-1: Diagrama de flujo, representando el flujo de trabajo para esta memoria	6
Figura 2-1: Clasificación del On Time Performance	24
Figura 2-2: Mapa representacional de un aeródromo	25
Figura 2-3: Representación de zonas de un aeródromo	26
Figura 2-4: Procesamiento del pasajero en un aeropuerto	27
Figura 2-5: Patio de clasificación de equipajes.....	28
Figura 2-6: Procedimiento de pushback	29
Figura 2-7: Servicio de reabastecimiento de suministros de <i>catering</i>	29
Figura 2-8: Reabastecimiento de combustible.....	30
Figura 2-9: Procedimiento de carga de ULD a la bodega delantera con un <i>cargolader</i>	30
Figura 2-10: Procedimiento de De-icing	31
Figura 2-11: Procedimiento de <i>towing</i> Teltonika GPS	32
Figura 2-12: conexión del puente de embarque con la aeronave	32
Figura 2-13: Servicio de limpieza de una aeronave.....	33
Figura 2-14: Procedimiento de drenaje de desperdicios	33
Figura 2-15: Carta Gantt de los servicios de ground handling.....	34
Figura 2-16: Factores que determinan la capacidad del aeródromo	36
Figura 2-17: Representación gráfica de subajuste y sobreajuste	42
Figura 2-18: Estructura de una neurona biológica.....	43
Figura 2-19: Esquema de un perceptrón con entradas, pesos y función de activación	45
Figura 2-20: Conjunto de perceptrones formando una red neuronal artificial	46
Figura 2-21: derivada direccional	50
Figura 2-22: Representación gráfica en 3 dimensiones de la función de costo en función de sus pesos	52
Figura 2-23: Representación de red neuronal con 1 capa oculta mostrando backpropagation.	52
Figura 2-24: cambio del error mediante descenso del gradiente.....	54
Figura 2-25: cambio de error mediante descenso del gradiente en 2 dimensiones	55
Figura 2-26: Comparación gráfica de descenso de gradiente estocástico vs descenso de gradiente.....	57
Figura 2-27: Célula LSTM	61
Figura 2-28: Modelo de aprendizaje automático supervisado <i>Random Forest</i>	66
Figura 2-29: Modelo de <i>support vector regression</i>	70
Figura 2-30: Demostración grafica de la transformación de datos a K-dimensionalidad	71
Figura 2-31: <i>Grafico de K Nearest Neighbors de K = 3</i>	73
Figura 3-1: Mapa de proceso del entrenamiento del modelo de machine learning.....	93
Figura 3-2: red neuronal multicapa construida en el modelo para predecir OTP	98
Figura 3-3: Resultados de los modelos machine learning.....	102
Figura 3-4: Resultado de las redes neuronales multicapa con la función de perdida de cuantiles.	102



Lista de abreviación

ACARS: *Aircraft Communications Addressing and Reporting System*

AMB: Arturo Merino Benítez

ATFM: Oficina de Gestión de Afluencia de Tránsito Aéreo

CCO: Centro de control de operaciones

ETA: *Estimated Time of Departure*

ETD: *Estimated Time of Arrival*

ETL: *Extract, Transform and Load*

GPU: *Ground Power Unit*

HUB: *Hub and Spoke*

IATA: *International Air Transport Association*

ICAO: *International Civil aviation organization*

JAC: Junta de Aviación Civil

KNN: *K-Nearest Neighbors*

LSTM: *Long Short Term Memory*

MAE: *Mean absolute error*

METAR: *Metorological aerodrome report*

ML: *Machine Learning*

MLP: *Multiplayer Perceptrón*

MSE: *Mean square error*

OTP: *On Time Performace*

PDI: Policia de Investigaciones



RF: *Random Forest*

RNN: Red Neuronal Recurrente

RWY: *Runway*

SCL: Aeropuerto de Santiago de Chile

STA: *Scheduled Time of Arrival*

STD: *Scheduled Time of Departure*

SVR: *Support Vector Regression*

TAT: *Turn Around Time*



Introducción

Las operaciones aéreas a nivel global imponen una necesidad de optimizar la eficiencia de la infraestructura aeroportuaria. Uno de los desafíos operativos más relevantes es la gestión dinámica de las puertas de embarque (*gates*), la cual se ve constantemente comprometida por el solapamiento de vuelos producto de retrasos o adelantos en las llegadas y salida de las aeronaves.

Esta problemática se acentúa en contextos de alta demanda, como el del aeropuerto Arturo Merino Benítez (SCL), haciendo indispensable la exploración de nuevas herramientas tecnológicas que permiten mitigar estos conflictos operativos y mejorar la utilización de los recursos.

Antes de la crisis sanitaria del COVID-19, las previsiones de la Asociación Internacional de Transporte Aéreo (IATA) indicaba que el número de pasajeros se duplicarían para 2036, alcanzando los 7.800 millones. (Falque, Mazure, Tabia, 2024, p.1). Este incremento implica una mayor complejidad logística y una sensibilización en los errores operativos, como la demora en los vuelos y el efecto dominó.

En la presente memoria, se busca la mejor herramienta entre seis de ellas que logre capturar los patrones de los itinerarios y meteorológicos, precisamente el *On Time Performance*.

Se realizará una prueba mediante los indicadores Median Square Error (MSE), Median Absolute Error (MAE) y el coeficiente de determinación (R^2). Además, se analizará visualmente el desempeño y como el subajuste o sobreajuste afecta en las predicciones.

El estudio se realiza en el aeropuerto Arturo Merino Benítez para vuelos de llegada y salida entre -200 y 180 minutos de *On Time Performance* desde 2023 hasta 2025.



1 Capítulo I: Antecedentes Generales

1.1 Justificación del tema

El crecimiento exponencial de la información y la interconexión global plantean desafíos significativos en la infraestructura computacional, principalmente en almacenamiento, procesamiento y el entendimiento de conjuntos de datos con múltiples variables. Las ausencias de algoritmos robustos dentro de los procesos internos de una organización imposibilitan la extracción de conocimiento y, por ende, la toma de decisiones estratégicas. Según Statista, (2024) el crecimiento de datos desde 2022 hasta 2025 se ha doblado, pasando de 97 zetabytes a 181 zetabytes.

La presente memoria aborda esta problemática aplicando herramienta de aprendizaje automático (*Machine Learning*) con el objetivo de predecir la puntualidad del aeropuerto Arturo Merino Benítez (SCL).

El *Machine Learning* ofrece una solución para la identificación de patrones complejos y el modelado de comportamiento de corto y largo plazo, mediante el entrenamiento de algoritmos predictivos. Esta capacidad logra disminuir los riesgos asociados a la incertidumbre operativa.

Maarten Beltman y sus colaboradores (2025), en artículo “*Dynamically forecasting airline departure delay probability distributions for individual flight using supervised learning*” presenta un enfoque innovador para predecir la probabilidad de retraso en la salida de un vuelo, Utilizando datos históricos y en tiempo real (como clima, congestión aeroportuaria y conexiones de pasajeros) logra modelos con error de hasta 7.37 minutos por vuelo.

En el contexto aeroportuario, la variabilidad de los itinerarios y las condiciones operativas complejiza la gestión de recursos críticos. Un ejemplo de esto es la asignación de puertas de embarque de un aeropuerto, siendo el principal problema el



solapamiento de vuelos en una misma puerta de embarque. La capacidad de predecir la puntualidad de los vuelos es de alto interés para los Centro de Control de Operaciones (CCO), ya que puede anticipar la gestión de la asignación de puertas de embarque y maximizar la utilización estos mismos.

Esta necesidad se vuelve más apremiante al considerar el crecimiento proyectado del tráfico aéreo. Antes de la crisis sanitaria del COVID-19, las previsiones de la Asociación Internacional de Transporte Aéreo (IATA) indicaba que el número de pasajeros se duplicarían para 2036, alcanzando los 7.800 millones. (Falque, Mazure, Tabia, 2024, p.1). Este incremento implica una mayor complejidad logística y una sensibilización en los errores operativos, como la demora en los vuelos y el efecto domino.

El Aeropuerto Arturo Merino Benítez ha experimentado una disminución observable en su índice de puntualidad en el periodo comprendido entre el primer trimestre de 2023 y el primer trimestre de 2024.

La Tabla 1.1 muestra para el trimestre 1 del 2023, la terminal registró un índice de puntualidad de 90,0% (JAC, 2023). Es fundamental contextualizar este desempeño con el volumen de tráfico aéreo de la época. Los datos del ministerio de obras públicas, dirección general de concesiones (2023) indican la cifra total de pasajeros transportados.

Año	Puntualidad	Pasajeros
2023 Q1	90,0%	4.134.639
2024 Q1	84,70%	4.987.768
2025 (MOP) Q1	-	5.486.545
2026 (MOP) Q1	-	6.035.199

Tabla 1-1: Comparación por trimestre de PAX contra la puntualidad del aeropuerto AMB.



Durante el primer trimestre de 2023, los pasajeros transportados a través de la terminal nacional T1 e internacional T2 ascendió a 4.134.639 pasajeros (Ministerio de Obras Públicas, Dirección General de Concesiones, 2023).

En contraste, el primer trimestre del año 2024 presentó un marcado descenso en la puntualidad que se situó en 84,4%. Estos datos se correlacionan con un incremento sustancial en pasajeros transportados a través de las terminales. Para ese periodo los pasajeros transportados alcanzaron 4.987.768 pasajeros movilizados (Ministerio de Obras Públicas, Dirección General de Concesiones, 2024).

El ministerio de obras públicas indica que La planificación de esta expansión de AMB responde al progresivo aumento de la demanda en el recinto, que desde 2010 ha crecido en torno al 10% anual con excepción del período de pandemia (MOP, 2025). Esto después de anunciar la triplicación de capacidad aeroportuaria a construir.



1.2 Objetivo General

- Predecir el *On-Time Performace* (OTP) en el aeropuerto Arturo Merino Benítez mediante algoritmos de *Machine Learning*, integrando datos operacionales y climatológicos.

1.3 Objetivo Especifico

- Estudiar algoritmos de Machine Learning que sean capaces de desarrollar mecanismos de predicción de múltiples variables.
- Definir variables operacionales a usar para predecir el indicador On Time Performance para el aeropuerto Arturo Merino Benítez.
- Comparar modelos desarrollados mediante el error que entregan, seleccionando el de mejor desempeño.
- Validar el modelo seleccionado, comparando su operación contra datos reales de la operación.

1.4 Metodología

El diagrama de flujo presenta la metodología que se usa en la presente memoria, donde se detalla el flujo del método usado para el desarrollo de recopilaciones de información, definiciones generales, proceso de entrenamiento y resultados. A continuación, se presenta la metodología en la figura 1.1.

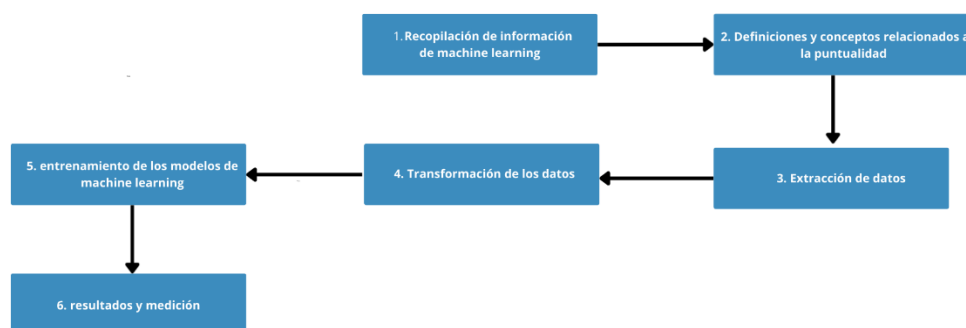


Figura 1-1: Diagrama de flujo, representando el flujo de trabajo para esta memoria (Open AI, 2024).

1. Recopilación de información de *Machine Learning*:

Investigar y documentar principales algoritmos predictivos, métricas de evaluación pertinentes al sector aeronáutico.

2. Definiciones y conceptos relacionados a la puntualidad:

Establecer datos operacionales a usar según su disponibilidad y calidad del dato. Además, establecer métricas de evaluación relacionado a la puntualidad, estudiado en el marco teórico.

3. Extracción de datos:

Obtener fuentes de datos de itinerarios y meteorología necesarias para construir el conjunto de entrenamiento, testeo y validación.



4. Transformación de los datos:

Implementar técnicas de limpieza de datos, normalización, ciclicidad y *Features Engineering* necesarias para transformarlos en conjuntos de datos útiles para el uso, asegurando su calidad y formato para el entrenamiento de los algoritmos de *Machine Learning*.

5. Entrenamiento de los modelos de *Machine learning*:

Realizar el entrenamiento de los algoritmos y optimización de estos sobre el conjunto de datos preprocesados, buscando la configuración optima (Hiperparámetros) que maximice el rendimiento del modelo.

6. Resultados y medición:

Determinar desempeño del algoritmo de *Machine Learning* utilizando métricas estandarizadas (RMSE, MSE y R^2).



1.5 Alcance

Se realiza un análisis correlacional, debido a la necesidad de identificar el grado de influencia y dependencia entre las variables operacionales, climatológicas y temporales respecto a la demora de los vuelos. Asimismo, se lleva a cabo un análisis exploratorio con el fin de encontrar modelos de *machine learning*, casos aplicados, variables usadas y resultados previstos en investigaciones.

Para ello, se utiliza variables independientes tales como *Scheduled Time*, *Estimated Time*, condiciones meteorológicas del aeropuerto (viento, presión y temperatura), las cuales permiten explicar la variable dependiente de *On Time Performance* (OTP).

Este estudio se ejecuta mediante distintos modelos de *Machine Learning*, comparando el rendimiento de *Random Forest*, *K-Nearest Neighbors (KNN)*, *Support Vector Regresión* (SVR) y redes neuronales *Long-Short Term Memory* (LSTM) y Red neuronal multicapa con función de pérdida de cuartiles. El análisis utiliza datos de dos años, desde abril de 2023 hasta abril de 2025, para predecir la puntualidad de arribos y salidas del Aeropuerto Arturo Merino Benítez (Nueva Pudahuel), en un rango de entre -200 y 180 minutos.



2 Capítulo II: Estado del Arte

2.1 Antecedentes

2.1.1 Rol de los administradores de aeropuerto y contexto de la puntualidad

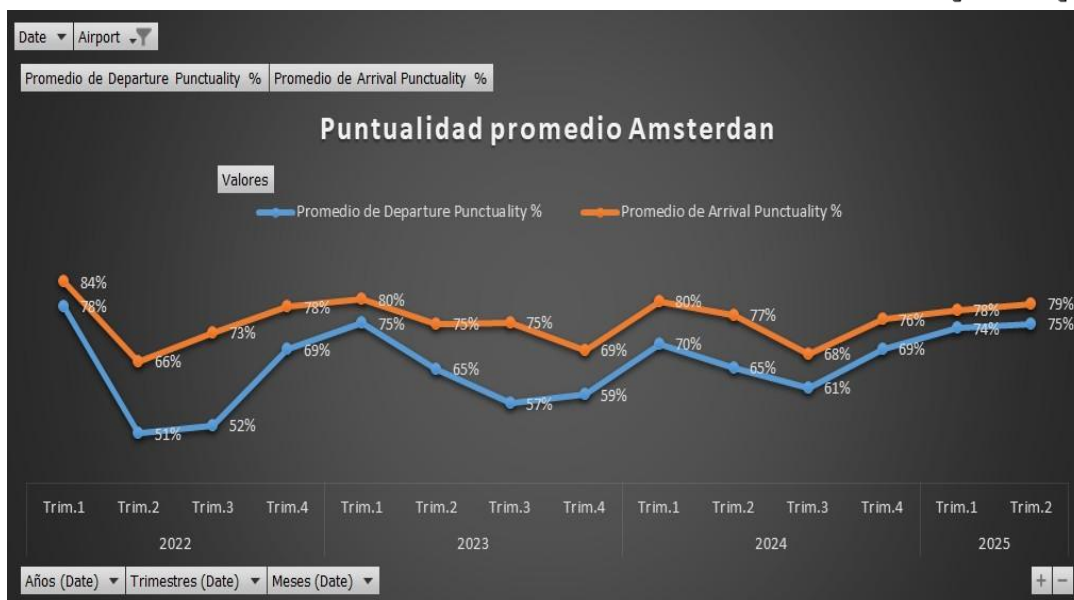
La puntualidad (On Time Performance) es un indicador de desempeño crítico dentro de la industria aeronáutica. En el marco de la operación aeroportuaria, un error o demora de una aerolínea genera un efecto dominó con importantes consecuencias financieras y logísticas.

El desafío de la puntualidad es global. Informes recientes muestran las métricas de On Time Performance con alta volatilidad, lo que refuerza la necesidad de herramientas predictivas.

A nivel regional, Según el informe *Transportes 2016 versão final web* (Ministerio dos Transportes, Portos e Aviação Civil, 2016) se registró un índice de puntualidad del 89% en los aeropuertos brasileños durante diciembre de 2016.

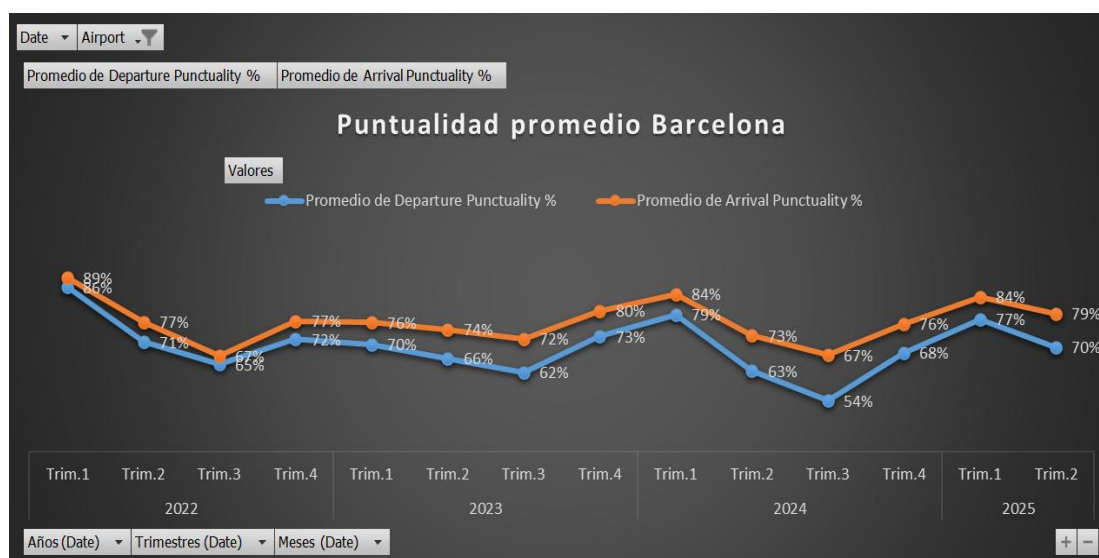
En el contexto de grandes centros aéreos europeos, con datos desde el año 2022 hasta el año 2025, según Eurocontrol (2025), revela la complejidad de mantener la puntualidad en entornos de alta densidad.

En el gráfico 2.1, Ámsterdam muestra un declive en la puntualidad para el segundo trimestre del año 2022, tanto en salidas como arribos, manteniendo un promedio de 57% de vuelos puntuales en salidas, lo que sugiere desafíos persistentes en la gestión operacional y 72% en arribos en los siguientes años hasta el cuarto trimestre del año 2023, manteniendo números similares hasta la fecha.



Gráfica 2-1: Puntualidad desde 2022 hasta 2025 separado por trimestre de Ámsterdam, Eurocontrol (2025).

Barcelona muestra una performance parecida al aeropuerto de Ámsterdam en cuanto a puntualidad de salidas y arribos, según el gráfico 2.2.



Gráfica 2-2: Puntualidad desde 2022 hasta 2025 separado por trimestre de Barcelona, Eurocontrol (2025).

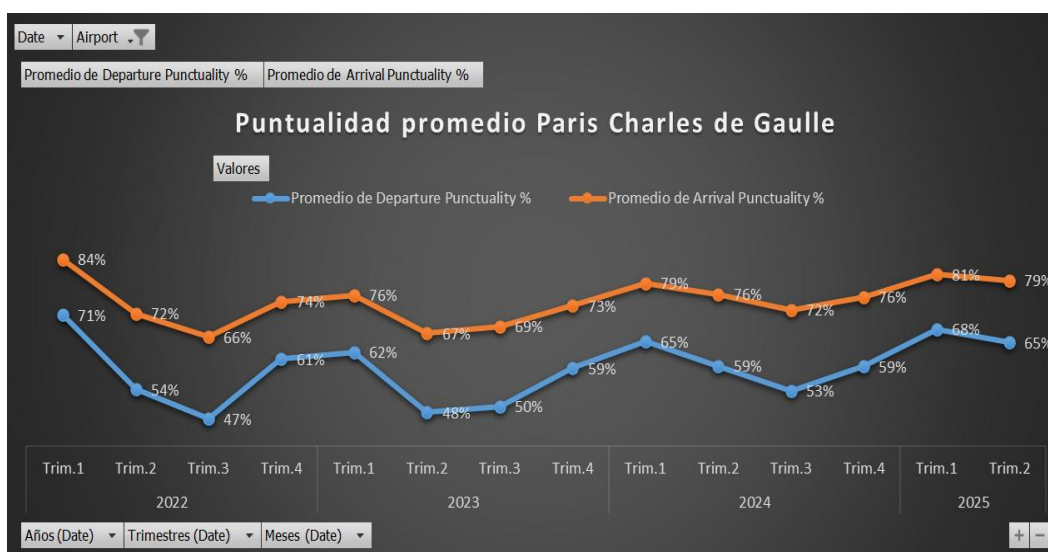
En el año 2022, tras un sostenido descenso en el segundo trimestre, la puntualidad de salida cae hasta un 65%, mientras que la llegada desciende hasta un 71%. A pesar de esto, ambos indicadores se recuperan gradualmente durante el resto del año.



Durante 2023, se mantiene una estabilidad relativa con valores entre el 62% y 74% en salidas, y entre el 72% y 84% en llegadas. El mejor rendimiento se observa en el cuarto trimestre de 2023.

En el año 2024, la puntualidad se ve nuevamente afectada, tocando su punto más bajo en el segundo trimestre (salidas 54%, llegadas 73%). Sin embargo, repunta al finalizar el año.

Para el año 2025, se registra una mejora evidente, con salidas alcanzando el 77% y arribos el 79% en el segundo trimestre.



Gráfica 2-3: Puntualidad desde 2022 hasta 2025 separado por trimestre de Paris Charles de Gaulle, Eurocontrol (2025).

El aeropuerto parisino presenta una marcada disparidad entre la puntualidad de salida y de llegada, con salidas frecuentemente más afectadas (Véase el gráfico 2.3).

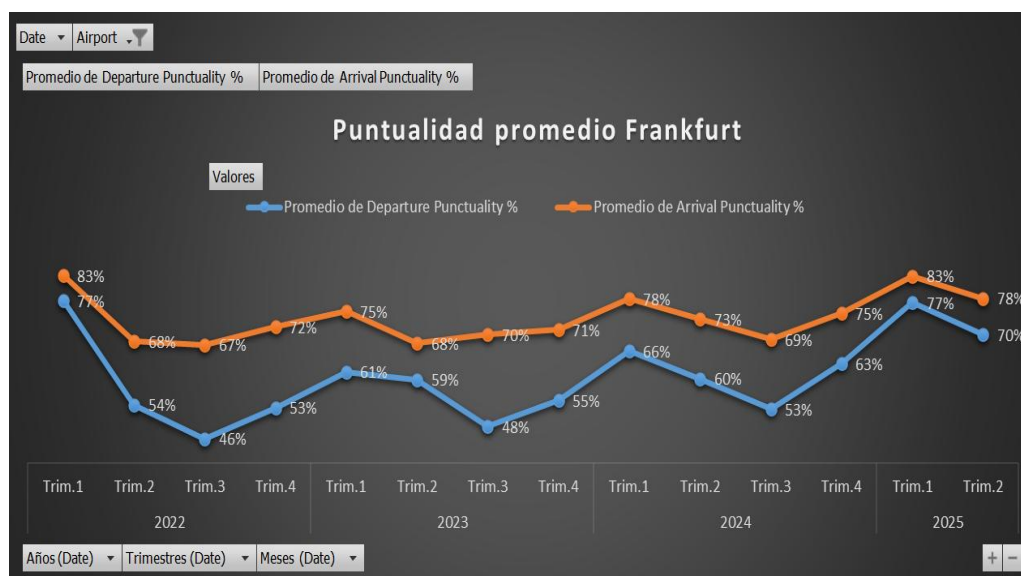
En el año 2022, la puntualidad de salidas cae abruptamente desde 71% en el primer trimestre hasta 47% en el tercero, mientras que los arribos presentan mejoras, estabilizándose entre 71% y 76%.



Durante el año 2023, los niveles de puntualidad siguen siendo modestos en salidas (entre 48% y 62%), mientras que las llegadas muestran una leve recuperación hacia el final del año (73% en el cuarto trimestre).

En el año 2024, la diferencia entre salida y llegada se mantiene. El punto más bajo de puntualidad en salidas es de 53% en el tercer trimestre, mientras que los arribos no bajan de 72%.

Finalmente, en el año 2025, se observa una mejora general, con salidas en 68% y llegadas en 79%.



Gráfica 2-4: Puntualidad desde 2022 hasta 2025 separado por trimestre de Frankfurt, Eurocontrol (2025).

Frankfurt presenta los valores más volátiles entre los cuatro aeropuertos analizados, con una fuerte caída inicial y una recuperación más lenta. En el año 2022, la puntualidad cae drásticamente de 79% a 46% en salidas entre el primer y tercer trimestre. Las llegadas también sufren una caída importante hasta un 67% (Véase gráfico 2.4).

En el año 2023, la puntualidad mejora gradualmente en salidas, pasando de 53% a



66%, mientras las llegadas se mantienen estables cerca del 70%.

Durante el año 2024, ambas métricas fluctúan, con puntos bajos en el tercer trimestre (salidas 53%, llegadas 69%). Esto puede reflejar congestión estacional o factores climáticos.

Para el año 2025, se alcanza el mejor rendimiento desde 2022: 77% en salidas y 78% en llegadas en el segundo trimestre. Frankfurt muestra la recuperación más significativa en comparación con su bajo rendimiento inicial.

2.1.2 Contexto actual del Arturo Merino Benítez

El aeropuerto Arturo Merino Benítez opera como la principal puerta de entrada al país y funciona bajo la configuración de terminal tipo Hub, implicando gestionar elevados volúmenes de tráfico.

En materia de puntualidad para el año 2023 y 2024, según el estudio “*Informe de regularidad y puntualidad*” (Junta de Aeronáutica Civil - JAC, 2024) y datos facilitados por la concesionaria Nueva Pudahuel (Véase Tabla 2.1), se ha observado las siguientes tendencias:

- Primer Trimestre (T1) de 2023: se registró una puntualidad del 90,0%
- Primer Trimestre (T1) de 2024: se registró una puntualidad del 83,2%



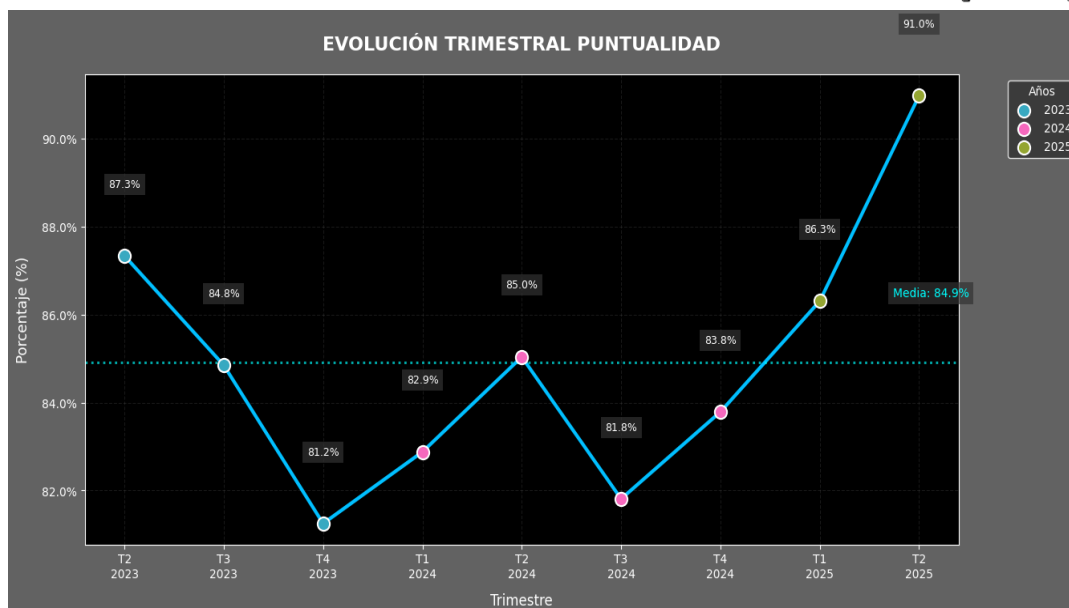
Año	Trimestre	Puntualidad	Atrasos		
			16 a 30 min.	31 a 60 min.	>60 min.
2023	T1	90,0%	5,2%	2,3%	2,5%
	T2	87,8%	6,7%	3,2%	2,4%
	T3	85,3%	7,2%	4,1%	3,5%
	T4	81,8%	9,1%	5,2%	3,9%
2024	T1	83,2%	8,9%	4,8%	3,1%
	T2	84,9%	7,7%	4,1%	3,3%
	T3	82,3%	8,2%	4,9%	4,6%
	T4	84,4%	8,2%	4,2%	3,2%

Tabla 2-1: Rendimiento de puntualidad del aeropuerto Arturo Merino Benítez.

Nota. Adaptado de “Informe de regularidad y puntualidad” (p. 3), por Junta de Aeronáutica Civil, 2024.

Esta variación subraya la alta volatilidad de Arturo Merino Benítez y justifica el desarrollo de un modelo predictivo capaz de anticiparse a estas caídas de rendimiento. Esta información toma solamente en cuenta los vuelos regulares con al menos 1 vuelo semanal y solo vuelos de pasajeros. No toma en consideración otros vuelos.

En el gráfico 2.5 se ve la puntualidad del aeropuerto con un horizonte de estudio de 2023 hasta el 2025. La puntualidad de las llegadas y salidas en el aeropuerto fue en promedio de 84,9%. Este desempeño es comparable con la región y Europa, mostrando una puntualidad igual o superior.



Gráfica 2-5: Puntualidad desde 2023 hasta 2025 separado por trimestre de Aeropuerto Arturo Merino Benítez
(Elaboración propia).

Arturo Merino Benítez cuenta con una capacidad de 55 operaciones/hora para salidas y arribos (Departamento de afluencia de tránsito aéreo, 2025). Este número por parte de Tránsito aéreo busca aumentar, realizando mejoras en el tiempo de espera entre operaciones en la pista, aumentando la carga de operaciones del aeropuerto. Por ello, técnicas de inteligencia artificial podrían ayudar conservar estos números o mejorarlos.



2.1.3 Historia y desarrollo del *Machine Learning*

La base conceptual del aprendizaje automático se encuentra ligada al desarrollo de las redes neuronales artificiales, remontado al siglo XX.

El primer hito conceptual se sitúa 1943, en el contexto de la segunda guerra mundial. El neurocientífico Warren McCulloch y el matemático Walter Pitts mencionan por primera vez el término red neuronal. Este publica el que es hoy en día el primer artículo referido a esta tecnología “*un cálculo lógico de las ideas inmanentes en la actividad nerviosa*”.

En esta publicación, los autores presentaron el primer modelo computacional simplificado de cómo las neuronas biológicas podrían trabajar conjuntamente en los cerebros de los animales para realizar cálculos complejos usando lógica proposicional. Ésta fue la primera arquitectura de red neuronal artificial.” (Géron, A., 2019, p. 280).

En laboratorios de IBM en la década de 1950, el campo experimentó por primera vez simular una hipotética red neuronal, realizado por Nathaniel Rochester, un destacado informático y arquitecto jefe de la primera computadora científica de IBM, sin embargo, los ensayos iniciales fallaron. Se podría decir que fue participe de la fundación del campo de la inteligencia artificial.

Un avance crucial para la metodología del aprendizaje automático se introduce en 1962 con el desarrollo del algoritmo ADALINE. Este trabajo fue realizado en la universidad de Stanford por el ingeniero Bernard Widrow y su estudiante de doctorado Ted Hoff. Consiste en combinar el error cuadrático medio de las predicciones de una función lineal y con el algoritmo de descenso de gradiente, buscaban lograr minimizar la variación del valor real y el predicho.

No fue hasta el año 1982, se resolvió uno de los mayores desafíos de la época. Rumelhart, Hinton y Williams demostraron la viabilidad de aplicar el algoritmo de



descenso de gradiente, para entrenar redes neuronales con multicapa (*Multilayers*). Antes de su trabajo, las aplicaciones de estas arquitecturas eran consideradas imposibles. La publicación de estos resultados estableció las bases metodológicas que hasta el día de hoy siguen siendo predominantes.



2.1.4 Aplicaciones del *Machine Learning* en la historia aeronáutica

La aplicación del aprendizaje profundo (*Deep Learning*) para la predicción de retrasos en el campo de la aviación es un tema activo de investigación, donde los autores presentan distintos desafíos relacionados a las predicciones de retrasos de llegadas y salidas de vuelos programados, distribuciones de probabilidad de llegada de aeronaves y predicciones de retrasos en calles de rodaje por mantenimiento de estas en ambientes simulados.

"La metodología propuesta por Thiruthuvanathan y sus colaboradores (2025), en el Paper "*Forescasting Flight Delays with a multilayered memory fusion network*" (2025) aborda la mejora del pronóstico de retrasos mediante un sofisticado método de *Deep Learning*, La metodología combina algoritmos de redes neuronales *Long-Short Term Memory bidireccional* (LSTM) y una red LSTM estándar, utilizando datos meteorológicos e itinerario de vuelos de Estados Unidos e India, capturando patrones complejos.

Los resultados demuestran la capacidad de los métodos modernos para abordar patrones de retrasos.

- Estados unidos: Se obtuvo un Error Absoluto Medio (MAE) de 72,41 y un Error Cuadrático Medio (RMSE) de 118,87.
- India: Se obtuvo un Error Absoluto Medio (MAE) de 11,83 y un Error Cuadrático Medio (RMSE) de 21,82.

Esto implica que el algoritmo de *Machine Learning* desarrollado por el estudio puede predecir los vuelos con un error de 11,83 minutos, aun siendo alto para ocuparlo en planificación de asignación de puentes de embarque.

El impacto operativo y financiero es alto. La legislación india estipula que, en caso de



un retraso de 6 horas en un vuelo, la aerolínea debe reprogramar al pasajero o emitir un reembolso total según lo indicado en el paper. Mejorar las metodologías para pronosticar retrasos en los vuelos generar ahorros anuales de millones de dólares.

No obstante, los autores identifican una limitación crucial. Las magnitudes predichas subestiman los valores, lo cual sigue siendo un desafío para la precisión final.

Un enfoque innovador para predecir la probabilidad de retraso en la salida de vuelos fue presentado recientemente (Beltman, Ribeiro, de Wilde y Sun, 2025), en el artículo “*Dynamically forecasting airline departure delay probability distributions for individual flight using supervised learning*” presenta un modelo para predecir la probabilidad de retraso en la salida de un vuelo, operados bajo un sistema *Hub And Spoke*, combinando técnicas como *Random Forest*, *CatBoost* y redes neuronales para generar distribuciones de probabilidad dinámicas.

La propuesta utiliza un modelo de aprendizaje supervisado para mejorar la puntualidad y optimizar el consumo de combustible, permitiendo a las aerolíneas ajustar la velocidad en ruta con base en pronósticos precisos. Utilizando datos históricos y en tiempo real (como clima, congestión aeroportuaria y conexiones de pasajeros) de una aerolínea anónima, datos Metar y datos de tráfico aéreo de Dallas-fort Worth.

El modelo alcanzó un MAE de 8.46 minutos a 90 minutos antes del despegue, reducido ese error a 7.37 minutos a 15 minutos previos al despegue, superando a modelos estadísticos tradicionales. Además, ofrece explicaciones claras a los despachadores sobre los factores que influyen en los retrasos (demoras en vuelos anteriores o condiciones meteorológicas). No obstante, enfrenta desafíos como la sobrestimación de retrasos pequeños y la dependencia de datos operativos detallados. Aun así, su implementación reduce costos operativos y mejorar la eficiencia en aeropuertos con alta demanda de conexiones en lo que respecta a asignación de puentes de embarque.

Aunque el estudio de Al-Ghzawi y El-Rayes (2024), titulado “*Machine learning for*



predicting the impact of construction activities on air traffic operations during airport expansion projects” (2024) no se centra en la predicción de la puntualidad (*On time Performance*), utiliza metodologías parecidas de la presente memoria.

El estudio se realizó en el aeropuerto de San Diego con el fin de optimizar la planificación de mantenimiento de calles de rodaje con un escenario simulado en conjunto de la aplicación de aprendizaje supervisado.

Se logra construir un modelo de *Machine Learning* capaz de predecir el tiempo del rodaje de la aeronave en función del cierre de ciertas combinaciones de calles de rodajes, logrando predecir cuáles son las combinaciones que afectan más al tiempo de (rodaje) *tax-in* y *tax-out*.

- El modelo con mejor desempeño fue una red neuronal.
- Se utilizó el *software Simmod Pro* para simular escenarios de cierre de calles de rodaje y el algoritmo de Dijkstra para calcular los tiempos mínimos de rodaje bajo diferentes combinaciones.
- Se obtuvo una precisión del **93,77%** en los tiempos de rodaje.
- Se obtuvo un MAE de **0.128** minutos.

Los resultados demuestran la efectividad de este algoritmo para la gestión operativa aeroportuaria, con uno de los errores más bajo. Esto infiere el uso de la tecnología de red neuronal como posible solucionador del problema de predicción de las variables operaciones críticas en el aeropuerto Arturo Merino Benítez.

La tabla 2.2 entrega la comparación entre los distintos estudios vistos anteriormente, comparando objetivo principal, metodología / modelo principal, datos utilizados, resultados, beneficios / implicaciones, limitaciones / desafíos y enfoque de predicción.



Paper	"Forecasting Flight Delays with a multilayered memory fusion network" (2025)	"Dynamically forecasting airline departure delay probability distributions for individual flight using supervised learning" (2025)	"Machine learning for predicting the impact of construction activities on air traffic operations during airport expansion projects" (2024)
Objetivo Principal	Mejorar el pronóstico de retrasos de aeronaves utilizando un método de aprendizaje profundo avanzado.	Predecir distribuciones de probabilidad dinámicas de retrasos en la salida de vuelos individuales para aerolíneas con modelo <i>hub and spoke</i> .	Predecir el impacto de las actividades de construcción (cierre de calles de rodaje) en las operaciones de tráfico aéreo (tiempo de rodaje) durante la expansión aeroportuaria.
Metodología / Modelo Principal	Aprendizaje Profundo (<i>Deep Learning</i>): Fusión de LSTM bidireccional y red LSTM (<i>Multilayered Memory Fusion Network</i>).	Aprendizaje Supervisado: Combinación de <i>Random Forest</i> , <i>CatBoost</i> y redes neuronales.	<i>Machine Learning</i> (Modelo de Red Neuronal con mejor éxito). Apoyado por <i>Simmod Pro</i> (simulación) y Algoritmo de <i>Dijkstra</i> (cálculo de tiempo mínimo).
Datos Utilizados	Datos meteorológicos y de itinerario de Estados Unidos e India.	Datos históricos y en tiempo real (clima, congestión, conexiones de pasajeros) de una aerolínea anónima, datos Metar y datos de tráfico aéreo de Dallas-Fort Worth.	Datos del aeropuerto de San Diego (relacionados con cierres de calles de rodaje y operaciones).



Paper	"Forecasting Flight Delays with a multilayered memory fusion network" (2025)	"Dynamically forecasting airline departure delay probability distributions for individual flight using supervised learning" (2025)	"Machine learning for predicting the impact of construction activities on air traffic operations during airport expansion projects" (2024)
Resultados (Métricas de Error / Precisión)	EE. UU.: MAE de 72.41 y RMSE de 118.87. India: MAE de 11.83 y RMSE de 21.82.	MAE de 8.46 minutos (90 min. antes del despegue) y 7.37 minutos (15 min. antes del despegue).	Precisión del 97.77% en los tiempos de rodaje y MAE de 0.128 minutos (tiempo de rodaje).
Beneficios/Implicaciones	Generar mejores pronósticos para millones de dólares en ahorros anuales, especialmente relevante ante normativas como la de India. Aborda patrones complejos impulsados por el clima.	Mejorar la puntualidad y optimizar el consumo de combustible (ajuste de velocidad en ruta). Reducir costos operativos y mejorar la asignación de puentes de embarque. Ofrece explicaciones claras sobre los factores del retraso.	Optimizar la planificación de expansión aeroportuaria (reconstrucción de calles de rodajes). Identificar las combinaciones de cierres que minimizan el impacto operativo.
Limitaciones/Desafíos	Las magnitudes de la predicción son menores a las reales.	Sobrestimación de retrasos pequeños y dependencia de datos operativos detallados.	(El texto no menciona una limitación directa del modelo de ML, sino que su enfoque es en el tiempo de rodaje por construcción y no en la puntualidad (OTP)).
Enfoque de Predicción	Retraso de vuelos (duración/magnitud).	Probabilidad de retraso en la salida de vuelos individuales.	Tiempo de rodaje (<i>tax-in</i> y <i>tax-out</i>) debido a cierres de calles de rodaje.

Tabla 2-2: Comparación de papers relacionados a metodologías de Machine Learning (Elaboración propia).



Se destaca el desempeño del estudio "*Dynamically forecasting airline departure delay probability distributions for individual flight using supervised learning*" con un error absoluto medio de 8.46 minutos (90 minutos antes del despegue) y 7.37 minutos (15 minutos antes del despegue). Realizar distintas predicciones en distintos tiempos logra una flexibilidad de planificación, cambiando las decisiones en tiempo real. Se utiliza una tecnología que no se toca en la presente memoria denominada "Catboost" que logra estos desempeños.

Este estudio utiliza fuente de datos similares a las que se busca obtener para el proceso de obtención de datos de entrenamiento. Por lo que, es un estudio valido para comparar desempeño frente al que se obtiene en esta memoria.

"Machine learning for predicting the impact of construction activities on air traffic operations during airport expansion projects" si bien no es un modelo de predicción para incumplimiento de itinerario, este no solo utiliza algoritmos de ML, sino que combina otros algoritmos como el djakara para predecir tiempo de rodajes según los cierres de calles de rodaje por mantenimiento, con un resultado sorprendente de 0.128 minutos de error absoluto medio.

"Forescasting Flight Delays with a multilayered memory fusion network" prueba una combinación de algoritmos de ML, siendo dos redes LSTM en su arquitectura. Muestra que existen soluciones donde se puedan probar distintos algoritmos a la vez para lograr predecir valores y cumplir con el objetivo de un error lo suficientemente pequeño, según el contexto, para ser utilizado como herramienta predictiva.

2.2 Marco teórico

2.2.1 Puntualidad

Según la JAC (2024), se define puntualidad como “el porcentaje de vuelos que se inician antes o hasta 15 minutos después del horario de salida programado por itinerario”. Además, se define los vuelos atrasados, estos se dividen en cuatro categorías: menos o igual a 15 minutos, entre 16 y 30 minutos de atraso; entre 31 y 60 minutos de atraso; y más de 60 minutos de atraso.

En la figura 2.1 se muestra la clasificación de tiempos de arribos o salidas del *On Time Performance (OTP)* de un vuelo.



Figura 2-1: Clasificación del *On Time Performance* (Elaboración propia).

Su clasificación se define de la siguiente forma:

$$\text{On Time Performance} = \begin{cases} \text{ON Block} - \text{Scheduled}, & \text{si la operación es llegada} \\ \text{OFF Block} - \text{Scheduled}, & \text{si la operación es salida} \end{cases}$$

Donde,

- *ON Block*: Hora real de llegada al estacionamiento.
- *OFF Block*: Hora real de *pushback*.
- *Scheduled*: Hora programada por itinerario

A. Actores involucrados

Para el entendimiento del fenómeno de atraso y puntualidad de los itinerarios, se requiere saber quiénes son los grupos de interés que colaboran directa e indirectamente a que un vuelo salga a la hora. Se definen 5 partes, cuya sinergia contribuye con la puntualidad.

1. Aeropuerto

La *International Civil Aviation Organization* (ICAO) en el anexo 14 define aeródromo como “área definida de tierra o de agua (que incluye todas sus edificaciones, instalaciones y equipos) destinada total o parcialmente a la llegada, salida y movimiento en superficie de aeronaves” (2016). Esto se ilustra en la figura 2.2, donde se muestran las distintas partes de un aeropuerto.

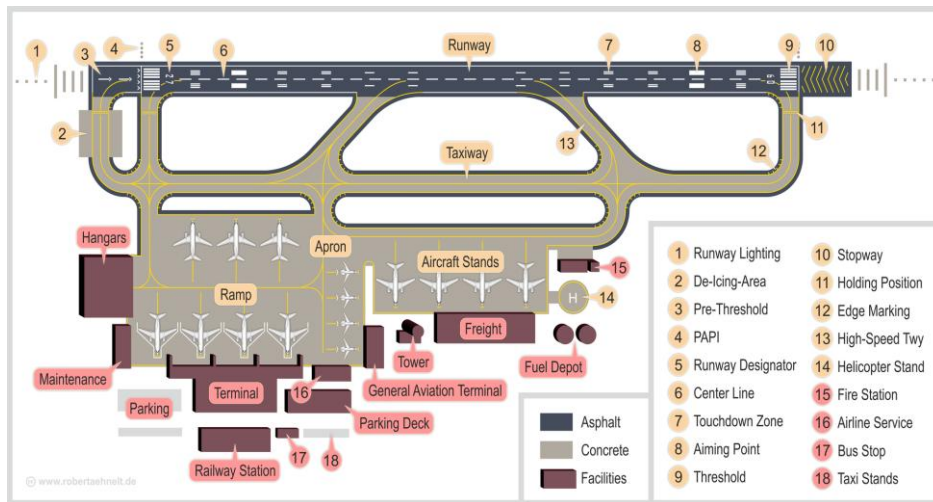


Figura 2-2: Mapa representacional de un aeródromo. (Sagar A. (2022). *Layout of Airport Planning* [Diapositiva]. SlideShare.

<https://es.slideshare.net/slideshow/layout-of-airport-planning/250705509>

Según el "reglamento de aeródromos del decreto 173 un aeropuerto es un “aeródromo público que se encuentra habilitado para la salida y llegada de aeronaves en vuelos internacionales.” (2005).

El aeropuerto se separa en dos macro partes, el lado landside y el airside. El landside componente todos los lugares no aeronáuticos cuyos accesos no tienen restricción, estos son; Estacionamiento, terminal de pasajero, check in, locomoción como bus, metro, auto y el uso de vías de acceso al aeropuerto, mientras que el lado airside se compone de todos los lugares aeronáuticos. Tales como; calle de rodaje, estacionamiento de aeronaves, apron, pista, torre de control, hangares y terminal de carga. Tal como se muestra en la figura 2.3.

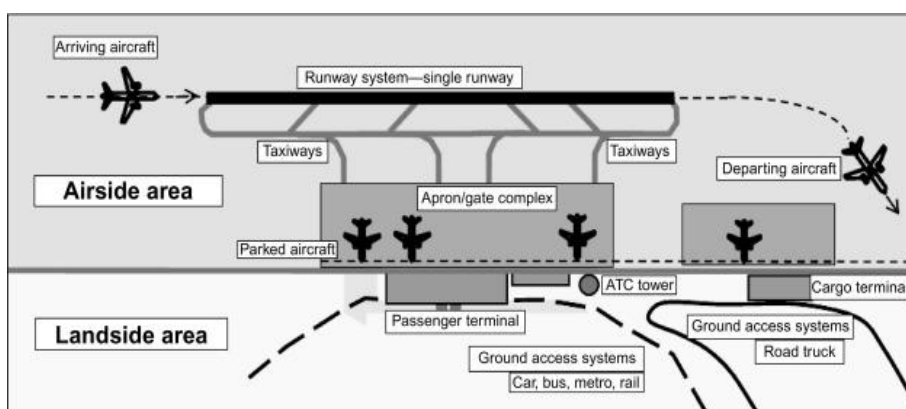


Figura 2-3: Representación de zonas de un aeródromo (ScienceDirect. (s.f.). Airfield. ScienceDirect Topics. Recuperado de <https://www.sciencedirect.com/topics/engineering/airfield>).

El procesamiento de pasajeros es un factor clave en la puntualidad del vuelo. Los cuellos de botellas son comunes en aeropuertos concurridos, en especial en horas pico. El aeropuerto Arturo Merino Benítez hoy en día refleja este problema dentro del proceso de Policía de Investigaciones (PDI), donde la falta de personal afecta el procesamiento de identificación de identidad. En la figura 2.4 se muestra el procesamiento.



Figura 2-4: Procesamiento del pasajero en un aeropuerto (All Nippon Airways. (s.f.). Procedimientos de Embarque. Sitio web de All Nippon Airways. Recuperado de <https://www.ana.co.jp/es/mx/travel-information/boarding-procedures/>).

La gestión de recursos en el aeropuerto Arturo Merino Benítez, a cargo de su centro de control de operaciones (CCO), incluye la asignación de puentes de embarque para el desembarque de pasajeros de vuelos entrantes y el embarque de los que salen. Generalmente, esta tarea se realiza entre 2 a 3 horas antes del vuelo, tomando la decisión de donde asignarle.

Sumado a eso, este gestiona el estacionamiento al que se dirigirá la aeronave y la asignación de cintas de equipajes para la llegada de maletas. Esto se realiza en el día a día tanto la asignación de puentes, estacionamiento y asignación de cintas de equipaje.

2. Ground Handling

“Servicios necesarios para la llegada y salida de una aeronave a un aeropuerto, distintos de los servicios de tránsito aéreo” (Flyinggroup, 2022).

“La prestación de un conjunto determinado de servicios aeroportuarios a las aeronaves, pasajeros, equipajes y mercancías en un aeropuerto y que son necesarios para el intercambio del modo de transporte aéreo al terrestre y viceversa, así como del aéreo al aéreo” (Carvallo, 2024).

En función de estas definiciones se procede a definir los servicios que prestan las

empresas de ground handling. Los servicios entregados se separan en dos; sobre el ala y bajo el ala.

Sobre el ala:

Asistencia al pasajero: “Comprende toda forma de asistencia a los pasajeros a la salida, la llegada, en tránsito o en correspondencia, en particular en control de billetes y documentos de viaje, la facturación de los equipajes y el transporte de equipajes hasta las instalaciones de clasificación” (carvallo. C, 2024).

Bajo el ala:

Tratamiento del equipaje: se realiza la clasificación del equipaje, el traslado del equipaje con tractores desde instalaciones hasta el avión correspondiente.



Figura 2-5: Patio de clasificación de equipajes. (Clarín. (2023, 22 de marzo). *Cuántas valijas se pierden en los vuelos, cuáles son las causas más frecuentes y el dato de robo y daño de equipaje*. Clarín Viajes. https://www.clarin.com/viajes/cuantas-valijas-pierden-vuelos-causas-frecuentes-dato-robo-dano-equipaje_0_tTQsZvy53Q.html)

Pushback: servicio de empuje del avión una vez terminado los servicios de handling para posicionar el avión en la calle de rodaje.



Figura 2-6: Procedimiento de pushback (Transpoco. (2022, 10 de marzo). Why Tracking Pushback Tractors Brings Huge Value to Your Ramp Operations. Blog de Transpoco. <https://www.transpoco.com/blog/why-tracking-pushback-tractors-brings-huge-value-to-your-ramp-operations>).

Catering: servicio de entrega de suministros alimenticios a la aeronave.



Figura 2-7: Servicio de reabastecimiento de suministros de catering (La Opinión. (2021, 5 de octubre). Gate Gourmet, empresa de catering aéreo, despedirá a más de 240 trabajadores del Aeropuerto de Los Ángeles. La Opinión. <https://laopinion.com/2021/10/05/gate-gourmet-empresa-de-catering-aereo-despedira-mas-de-240-trabajadores-del-aeropuerto-de-los-angeles/>).

Re-fueling: servicio de reabastecimiento en tierra de combustible.



Figura 2-8: Reabastecimiento de combustible (ARC Refuellers. (s.f.). Aircraft Ground Refuelling. Sitio web de ARC Refuellers. Recuperado de <https://arc-refuellers.be/aircraft-ground-refuelling/>).

Tratamiento del cargamento: servicio de traslado de cargamento en conjunto de equipos como tractores y montacargas para cargar el avión con carga útil.



Figura 2-9: Procedimiento de carga de ULD a la bodega delantera con un cargolader. (Joloda. (s.f.). Almacenamiento de ULD. Sitio web de Joloda. Recuperado de <https://www.joloda.com/es/equipos-de-carga-aerea/soluciones/almacenamiento-de-uld/>).

De-icing: para temporadas de invierno, donde aparecen fenómenos de engelamiento sobre las superficies aerodinámicas, es necesario una medida de protección contra este fenómeno. Para ello, se rocía con un líquido a base de etileno glicol y propilenglicol (depresores del punto de congelación), agentes espesantes, tensioactivos e inhibidores de corrosión.



Figura 2-10: Procedimiento de De-icing (Aviación Digital. (s.f.). El backstage del proceso de deshielo en los aviones de Delta. Aviación Digital. <https://aviaciondigital.com/el-backstage-del-proceso-de-deshielo-en-los-aviones-de-delta%ef%bb%bf/>).

Servicio *Towing*: consiste en el remolcado de la aeronave, con el fin de trasladar el avión desde los centros de mantenimientos o estacionamiento de pernocte hacia el espigón correspondiente.



Figura 2-11: Procedimiento de *towing* Teltonika GPS. (s.f.). *Seguimiento de activos y vehículos terrestres en aeropuertos*. Teltonika GPS. Recuperado de <https://www.teltonika-gps.com/es/use-cases/telematics/airport-ground-vehicles-and-assets-tracking>.

Puente de embarque: consiste en un puente, cuyo movimiento es motorizado para poder mover su brazo, extendiendo, rotando y ajustando su altura para cada tipo de aeronave, además, provee servicio de unidad de potencia auxiliar (GPU) para la energización de la aeronave. Vital servicio para el ingreso de pasajeros.



Figura 2-12: conexión del puente de embarque con la aeronave (TK Elevator (Airport Solutions)). (s.f.). *Passenger Boarding Bridges*. Sitio web de TK Elevator (Airport Solutions). Recuperado de <https://airportsolutions.tkelevator.com/global-en/products/passenger-boarding-bridges/>.

Servicio de limpieza: servicio de limpieza de baños, asientos, y suelo de la aeronave en un tiempo acotado con herramientas de limpieza como bolsas, guantes y aspiradora.



Figura 2-13: Servicio de limpieza de una aeronave

(<https://www.periodicodeibiza.es/noticias/baleares/2020/05/18/1166013/hora-vuelo-mas-dos-para-desinfeccion.html>).

Agua y baño: suministro de agua a los depósitos del avión y el drenaje de desperdicios del baño.



Figura 2-14: Procedimiento de drenaje de desperdicios (El Confidencial. (2023, 1 de marzo).

Todos estos servicios tienen un tiempo acotado para su realización, siendo vitales para que el avión pueda retomar sus servicios. Por lo general, estos servicios son tercerizados y varias empresas en conjunto lo realizan. Por lo que, la eficiencia de cada uno por separado es fundamental para el cumplimiento del *Scheduled Time of Arrival* (STA) y el *Scheduled Time of Departure* (STD).

A continuación, se presenta una carta *Gantt* de los servicios de un *Turn Around Time* (TAT).

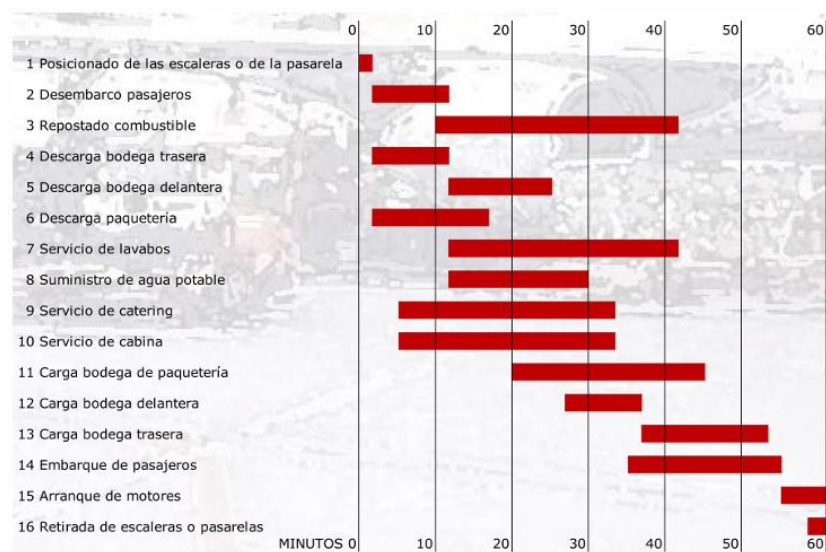


Figura 2-15: Carta Gantt de los servicios de ground handling (Carvallo. C, 2024).

Servicio de tránsito aéreo

La oficina de gestión de afluencia de tránsito aéreo (ATFM) se encarga de la gestión de capacidad del sistema nacional de aeródromos de Chile de la red primaria. Gestiona los itinerarios entregados por las aerolíneas con interés de operar rutas regulares, asegurando no superar la capacidad máxima por hora de llegadas y salidas de un aeródromo.

Este trabajo se realiza con el fin de optimizar los itinerarios, alineándolo con la capacidad operativa del aeródromo y reducir los tiempos de espera para despegar o los holdings antes de aterrizar, reduciendo problemas de On Time Performance de los aeropuertos.

Las capacidades operacionales no son valores estáticos, puesto que varían en función de la complejidad del tránsito, hora del día, mantenimiento, entre otros factores. Por lo general, los niveles de demanda superior a la capacidad de pista dan lugar a algún tipo de intervención de la oficina de ATFM, al tiempo que valores máximos de demanda durante un breve período de tiempo, moderadamente superiores a la capacidad, pueden gestionarse mediante un análisis detallado, sin intervención. Los umbrales de tolerancia pueden definirse para fijar esas posibles variaciones de la capacidad y velar porque dichas variaciones se ajusten a un rango determinado. (Dirección General de Aeronáutica Civil. (2022). Gestión de afluencia del tránsito aéreo (DAP 11 136, Edición 1). DGAC.)

En la figura 2.16, se muestran los principales factores que determinan la capacidad del aeródromo.

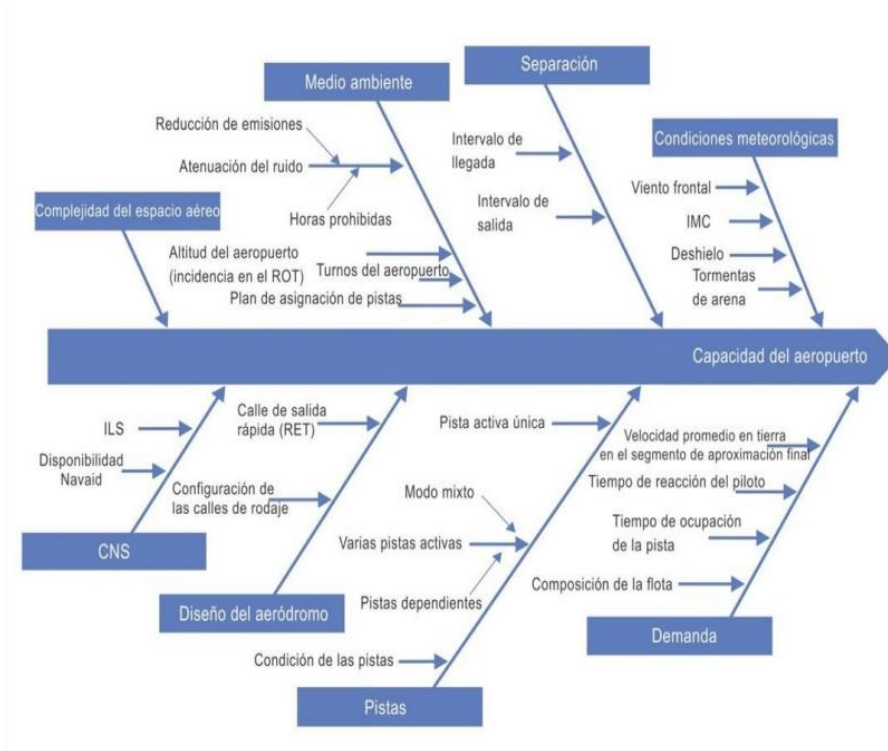


Figura 2-16: Factores que determinan la capacidad del aeródromo (Dirección General de Aeronáutica Civil (DGAC). (2022, 12 de mayo). *DAP-11/136 OPINION* [PDF]. Recuperado de https://www.dgac.gob.cl/wp-content/uploads/2022/05/DAP-11_136_OPINION_12-MAY-2022.pdf).

Aerolínea

Las aerolíneas gestionan sus recursos de manera meticulosa para lograr la rentabilidad. El contexto latinoamericano para el año 2025 sitúa a la región como aerolíneas en crecimientos y otras con fuertes dificultades financieras.

Según IATA (2025), el margen de beneficio neto de las aerolíneas latinoamericanas en 2025 será del 2.4%, inferior al 2.8% registrado en 2024, debido a la debilidad cambiaria y presiones regulatorias" (p. 6). En este contexto, una mala gestión de los recursos fácilmente pondría en jaque los márgenes de los vuelos. Las ineficiencias en tiempo son castigadas por multas por parte de los concesionarios de aeropuertos, incentivando la buena gestión de la aerolínea y de los servicios satélites.

Los distintos factores que pueden influenciar en el On Time Performance donde la aerolínea tiene control son:

- Programación de vuelos: La planificación de los horarios de vuelo es una de las decisiones estratégicas más importantes que toman las aerolíneas, ya que tiene un impacto directo en la eficiencia operativa y la rentabilidad (Birolini et al., 2021). Normalmente, consta de cinco pasos: programación de vuelos, asignación de flota, rotación de aeronaves, gestión de ingresos y programación de la tripulación (Belobaba et al., 2015).

La programación de vuelos según Belobaba comenta” Los tramos de vuelo a realizar y la hora de salida de cada tramo definen en gran medida la posición competitiva de una aerolínea y, por lo tanto, son un determinante clave de la rentabilidad de las aerolíneas” (2015).

La programación de estos puede evitar itinerarios ya congestionados. Por lo tanto, la eficiencia de estos pasos marca una diferencia estratégica para la aerolínea, la equivocación en uno de estos no solo afectaría en la puntualidad sino en la

rentabilidad.

- Turno de tripulación: La asignación de tripulación para un vuelo no es ambiguo y requiere de cumplimientos legales estrictos que, de no cumplirlos, degradarían la puntualidad del vuelo. además, mencionar la disponibilidad de una tripulación según sus horas de vuelo determinarán a que vuelo serán asignados y, por ende, la construcción de los itinerarios.

Según la Dirección General de Aeronáutica Civil (DGAC, 2009, Resolución N° 01344) define los siguientes conceptos.

- Periodo de servicio de vuelo: Incluye desde la presentación del tripulante en el aeropuerto hasta su liberación post-vuelo. Cubre también entrenamientos y traslados operativos.
- Tiempo de vuelo: Calculado desde el despegue hasta el aterrizaje
- Periodo de descanso: periodo de descanso legal que debe tener la tripulación en función del periodo de servicio en vuelo.
- Reposo en vuelo: corresponde a el lugar de descanso en vuelo.

Con esto en mente, el periodo de servicio de vuelo y el periodo de descanso se ve reflejado en la tabla 2.3

Periodo de Servicio de Vuelo (P.S.V)	Periodo de Descanso (PD)
7 horas o menos	10 horas
8 horas	12 horas
9 horas	13 horas
10 horas	14 horas
11 horas	15 horas
12 horas	15 horas
13 horas	15 horas
14 horas	17 horas
15 horas	17 horas
16 horas	18 horas
17 horas	19 horas
18 horas	20 horas
19 horas	22 horas
20 horas	24 horas

Tabla 2-3: Tabla de P.S.V y PD para tripulantes de vuelo (Dirección General de Aeronáutica Civil (DGAC). (2023). DAN 92, Volumen II, Edición 1 [PDF]. Recuperado de <https://www.dgac.gob.cl/wp-content/uploads/2023/08/DAN-92-Vol-II-ED.-1-.pdf>.

Un error en la gestión de estos tiempos traería restricciones legales en el uso de la tripulación y, por ende, un atraso por falta de tripulación.

B. Variables que afectan a la puntualidad

La International Air Transport Association (IATA), una entidad privada, cuyo objetivo es la estandarización de procesos en aerolíneas, productos y servicios satélite, donde estas se suscriben anualmente, ha definido todas las variables que afectan a los atrasos de los vuelos.

Este contiene 99 códigos distintos (Véase anexo A.1), donde se describe el motivo de un atraso, documentando problemas como documentación de aeronave tardía/incorrecta (peso, balance, manifestó, etc.) o daño en tierra (colisiones, contaminación, mal tiempo).

En la Tabla 2.4 se presentan aquellas que se usarán en esta memoria.

71 (WO)	Clima en estación de salida
72 (WT)	Clima en estación de destino
99 (MX)	Otros motivos no especificados

Tabla 2-4: códigos IATA Delay (ANS Performance. (s.f.). IATA Delay Codes. ANS Performance.

Recuperado de <https://ansperformance.eu/definition/iata-delay-codes/>.

La base de datos se compone principalmente de operaciones atrasadas, puntuales, cantidad de operaciones y datos climatológicos como dirección del viento, magnitud del viento y temperatura en el sector del aeropuerto. Por lo que, se presume que estos códigos son aquellos que más afectan al OTP de la base de datos a usar.

2.2.2 Aprendizaje de maquina

A. Aprendizaje supervisado

“El aprendizaje supervisado es una técnica de Machine Learning que utiliza conjuntos de datos de entrada y salida etiquetados por humanos para entrenar modelos de inteligencia artificial” (IBM, 2024).

Los modelos que utilizan esta técnica se caracterizan por utilizar conjuntos de datos en los que ya se conoce la variable objetivo. Los modelos de aprendizaje supervisado se dividen según cuál es su variable para predecir, estos entregan un dato tipo regresión (continua) o tipo clasificación (binaria).

“El aprendizaje supervisado utiliza un conjunto de datos de entrenamiento etiquetado para comprender las relaciones entre los datos de entrada y salida.” (IBM, 2024)

Ejemplos de estos modelos son: Regresión Lineal, Regresión Logística, K-Nearest-

Neighbor, Support Vector Machine, Random Forest, redes neuronales como Long-Short Term Memory (LSTM) o Red Neuronal Multicapa (MLP) y Clasificación Bayesiana.

B. Aprendizaje no supervisado

La contraparte del aprendizaje supervisado es el aprendizaje no supervisado, cuya principal particularidad radica en que el conjunto de datos no viene con una variable objetivo o etiqueta predefinida.

IBM (2024) define esta técnica como:

“El aprendizaje no supervisado, también conocido como Machine Learning no supervisado, utiliza algoritmos de Machine Learning (ML) para analizar y agrupar conjuntos de datos no etiquetados. Estos algoritmos descubren patrones ocultos o agrupaciones de datos sin necesidad de intervención humana”.

En esencia, el algoritmo opera de manera autónoma, logrando encontrar patrones y etiquetarlos según el mismo criterio del algoritmo. Una aplicación común de estos modelos es el etiquetado de datos no clasificados previamente, usados por ejemplo para la segmentación de pasajeros, identificando si su comportamiento es business passenger o leisure passenger.

Los modelos de aprendizaje no supervisados se separan en dos categorías.

- *Clustering* (agrupación): busca patrones ocultos para crear grupos (clusters), identificando características similares en un conjunto de datos no etiquetados.
- Reducción de dimensionalidad: Se enfoca en identificar variables no significativas o redundantes para el modelo, eliminándolos automáticamente.

C. Subajuste

“El subajuste es un escenario en la ciencia de datos en el que un modelo de datos no puede capturar con precisión la relación entre las variables de entrada y salida, lo que genera una alta tasa de error tanto en el conjunto de entrenamiento como en los datos no vistos” (IBM, 2024).

“El subajuste ocurre cuando un modelo es demasiado simple, lo que puede ser el resultado de que un modelo necesite más tiempo de entrenamiento, más características de entrada o menos regularización” (IBM, 2024).

D. Sobreajuste

“En el Machine Learning, el sobreajuste ocurre cuando un algoritmo se ajusta demasiado o incluso exactamente a sus datos de entrenamiento, lo que da como resultado un modelo que no puede hacer predicciones o conclusiones precisas a partir de ningún otro dato que no sea el de entrenamiento” (IBM, 2024).

En la Figura 2.17, se muestra la representación gráfica de subajuste y sobreajuste de un modelo.

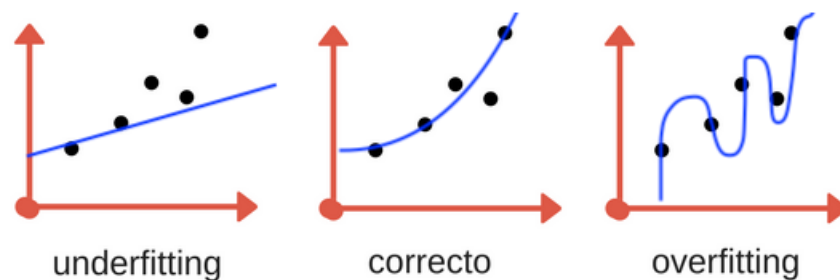


Figura 2-17: Representación gráfica de subajuste y sobreajuste (Aprende Machine Learning. (s.f.). *Qué es Overfitting y Underfitting y cómo solucionarlo. Aprende Machine Learning. Recuperado de.*

2.2.3 Redes neuronales

A. Neuronas

“Las neuronas son las células que forman la corteza cerebral de los seres vivos, cada una está formada por elementos llamados cuerpo, axón y dendritas” (Rosenblatt, 1957, p. 73). En la Figura 2.18 se muestra una representación de una neurona biológica.

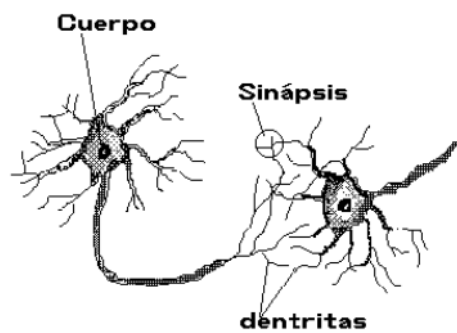


Figura 2-18: Estructura de una neurona biológica. Adaptada de *Red Neuronal Artificial. El Perceptrón [Capítulo 5]*, por F. Rosenblatt, 1957, p. 73.

En la sinapsis, la neurona vecina envía información en forma de un impulso eléctrico. Si la neurona vecina envía suficiente impulso eléctrico, la neurona receptora admitirá la información. Esto se denomina señal excitatoria, de lo contrario la rechaza y se denomina señal inhibidora. La información una vez aceptada, viaja por el axón, es procesada y está lista para ser liberada a través de las dendritas. Esta misma lógica está detrás de una neurona artificial, donde las señales son los outputs del sistema, y la activación de esta son las funciones de activación. Estos términos serán tomados en algunas páginas más adelante.

B. Perceptrón

El perceptrón constituye la representación artificial básica de una neurona biológica, siendo la unidad fundamental de una red neuronal. Su estructura se compone de cinco elementos: entradas, pesos, suma ponderada, función de activación y salida.

La Figura 2.19 ilustra la arquitectura de un perceptrón. Las entradas (Inputs), denotadas por una x_m , representan la información de interés del sistema, siendo datos de interés. Cada entrada x_m está asociada a un peso w_m , funcionando como un grado de importancia que se le da a cada entrada x_m .

El primer paso dentro del perceptrón es calcular el pre-output, que es la suma ponderada de cada entrada por su respectivo peso, tal como se muestra en la ecuación (2).

$$\sum_{m=1}^m w_m \cdot x_m + w_0 \quad (2)$$

Donde w_0 es el sesgo (bias). En similitud a la neurona biológica, se tiene una activación (señal excitatoria) o una desactivación (señal inhibidora) de la neurona, generalmente representado por alguna función de activación como la función sigmoide, tangente hiperbólico o incluso función lineal.

El resultado de la suma ponderada hecha anteriormente alimenta la función de activación, entregando el valor final del perceptrón.

El condominio de salida de la función de activación se encuentra entre $[0,1]$ (para funciones como sigmoide) o $[-1,1]$ (para funciones como tangente hiperbólica), dependiendo del tipo de función. Esta se presenta en la ecuación (3).

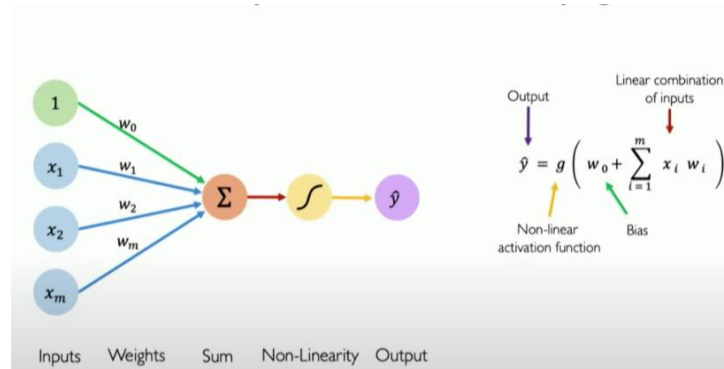


Figura 2-19: "Esquema de un perceptrón con entradas, pesos y función de activación". ¿Adaptada de *What is a Perceptron?*, por AIML.com, [s.f.]. Recuperado de <https://aiml.com/what-is-a-perceptron/>

$$y = f\left(\sum_{m=1}^m w_m \cdot x_m + w_0\right) \quad (3)$$

Una vez aplicado la ecuación (3), el flujo del perceptrón termina. Este puede continuar como la entrada de otro perceptrón, convirtiéndose en un conjunto de perceptrones (red neuronal) o ser el valor final del flujo.

Las redes neuronales multicapa (MLP) es un conjunto de perceptrones unidos de n capas, separadas por capas de entrada, capas oculta y capa de salida.

Como se ve, esto se asemeja a la actividad cerebral en el procesamiento de la información. “El modelo de una neurona artificial es una imitación del proceso de una neurona biológica” (Rosenblatt, 1957, p. 76).

El objetivo de una red neuronal multicapa es modelar la realidad, ajustando sus parámetros para acercarse a la realidad tal como se ha visto. En este modelo, las entradas x_m de las capas ocultas y capas de salida de la Figura 2.20 son las salidas de perceptrones de la capa anterior.

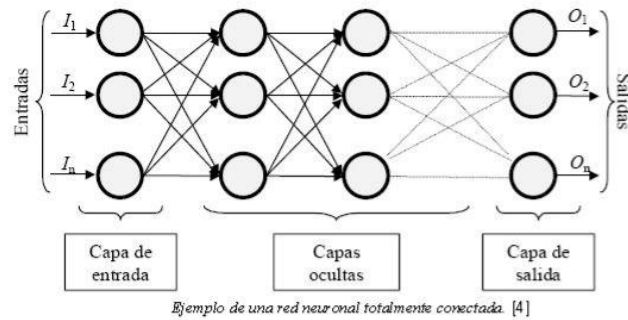


Figura 2-20: Conjunto de perceptrones formando una red neuronal artificial. Adaptada de Redes Neuronales (RNA), por AdvancedTech, 2008. Recuperado de <https://advancedtech.wordpress.com/2008/08/07/redes-neuronales-rna/>

- Componentes de una red neuronal

Esta sección se dedicará a la profundización en los elementos constitutivos de la neurona artificial y la teoría matemática que sostiene el entrenamiento y el aprendizaje automático de este algoritmo.

- Función de activación

Según la Pontificia Universidad Católica de Valparaíso (2010) “Las funciones de activación se encargan de calcular el estado de actividad de una neurona transformando la entrada total en un valor de activación cuyo rango generalmente se encuentra entre $[0; 1]$ o de $[-1; 1]$, estos valores indican que la neurona puede estar activa (1) o inactiva (-1 o 0 según la función)”.

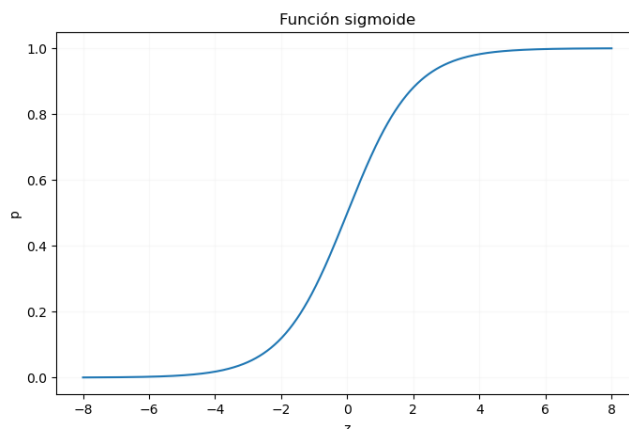
A continuación, se presenta las funciones de activación comunes en redes neuronales:

- Función sigmoide

La función Sigmoidal de la ecuación (4) es una de las más utilizadas en las redes neuronales de tipo perceptrón multicapa, debido a la flexibilidad y rango de resultados que esta ofrece. Esta función permite a las redes aprender variaciones no lineales.

$$f(x) = \frac{1}{1+e^{-x}} \quad (4)$$

A continuación, se presenta la gráfica de la función:



Gráfica 2-6: Función Sigmoide (Interactive Chaos. (s.f.). La función Sigmoide. Interactive Chaos. Recuperado de <https://interactivechaos.com/es/manual/tutorial-de-machine-learning/la-funcion-sigmoide>).

- Función tangente hiperbólica

Según Pérez (2011) menciona: “Hay algunas combinaciones de las funciones $\exp(x)$ y $\exp(-x)$ que aparecen con tanta frecuencia que se les da nombre propio. Ellas son las funciones seno hiperbólico, representada por $\sinh$, y coseno hiperbólico, representada por $\cosh$, y están definidas para todo $x \in R$ representada por: $\sinh(x)$ y coseno hiperbólico, representada por $\cosh(x)$ ”. Se definen de acuerdo con las ecuaciones (5) y (6).

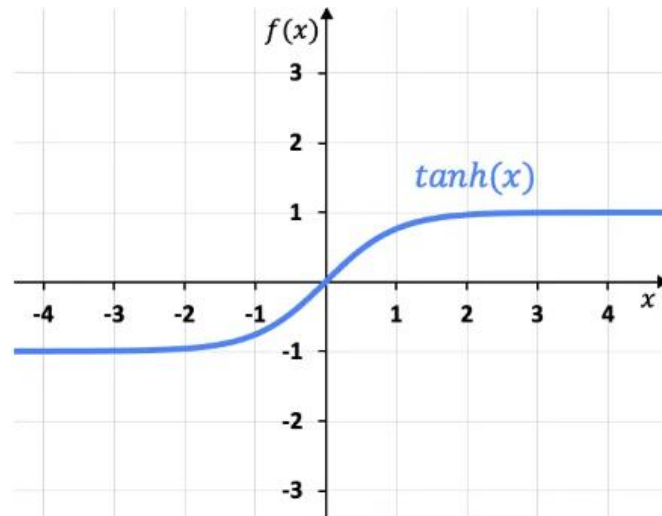
$$\sinh(x) = \frac{e^x - e^{-x}}{2} \quad (5)$$

$$\cosh(x) = \frac{e^x + e^{-x}}{2} \quad (6)$$

En combinación de ambas funciones se tiene la función tangente hiperbólica, ampliamente usado en el contexto de redes neuronales. La función tangente hiperbólica que se representa por $\tanh$ es la función definida para todo $x \in R$ por la ecuación (7):

$$tgh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (7)$$

Esta es representada por la siguiente la Gráfica 2.7:



Gráfica 2-7: Función tangente hiperbólica (Funciones.xyz. (s.f.). *Función Tangente Hiperbólica*. Funciones.xyz.

Recuperado de <https://www.funciones.xyz/funcion-tangente-hiperbolica/>.

- Función de pérdida

Según IBM (2023), “se utiliza una función de pérdida para medir el rendimiento del modelo calculando la desviación de las predicciones de un modelo con respecto a las predicciones correctas de verdad fundamental”. La optimización de un modelo implica ajustar sus parámetros para minimizar la obtención de alguna función de pérdida.”

El error cuadrático medio (MSE) es una función de pérdida típica en modelos de red neuronal. Se calcula como la media de las diferencias al cuadrado entre el valor previsto y el valor real en todos los ejemplos de entrenamiento. La ecuación (8) sirve para calcular el MSE a través de n puntos de datos se escribe como:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (8)$$

en la que y_i es el valor verdadero e $\hat{y}_i$ es el valor predicho. El MSE siempre es diferenciable, lo que lo hace práctico para optimizar los modelos de regresión mediante el descenso de gradiente (IBM, 2023).

Esto será útil para ajustar el modelo de red neuronal, buscando minimizar el MSE. Esto se verá en la sección 2.2.3 H.

- Derivada direccional

En el contexto de las redes neuronales, buscamos minimizar el error cuadrático medio (MSE), que actúa como función de coste (E). Es imprescindible determinar la dirección en el espacio de pesos (w_m) donde el error se reduce rápidamente (Véase figura 2.24). Por ello, la necesidad de la herramienta matemática de cálculo multivariable como la derivada direccional y su relación con el vector gradiente es fundamental como construcción de bases que sostiene el entrenamiento de este algoritmo.

Según Steward (2010), si $z = f(x, y)$, entonces las derivadas parciales f_x y f_y definen de acuerdo con las ecuaciones (9) y (10):

$$f_x(x_0, y_0) = \lim_{h \rightarrow 0} \frac{f(x_0 + h, y_0) - f(x_0, y_0)}{h} \quad (9)$$

$$f_y(x_0, y_0) = \lim_{h \rightarrow 0} \frac{f(x_0, y_0 + h) - f(x_0, y_0)}{h} \quad (10)$$

y representan las razones de cambio de z en las direcciones x e y ; es decir, en las direcciones de los vectores unitarios i y j .

Supongamos que ahora queremos encontrar la razón de cambio de z en x_0, y_0 en la

dirección de un vector unitario arbitrario (buscando posibles direcciones donde el error pueda minimizarse). Para hacer esto consideremos la superficie S de la Figura 2.21, cuya ecuación es $z = f(x, y)$, y sea $z_0 = f(x_0, y_0)$. Entonces el punto $P(x_0, y_0, z_0)$ queda sobre S . El plano vertical que pasa por P en la dirección de u interseca a S en una curva C . La pendiente de la recta tangente T a C en el punto P es la razón de cambio de z en la dirección de u .

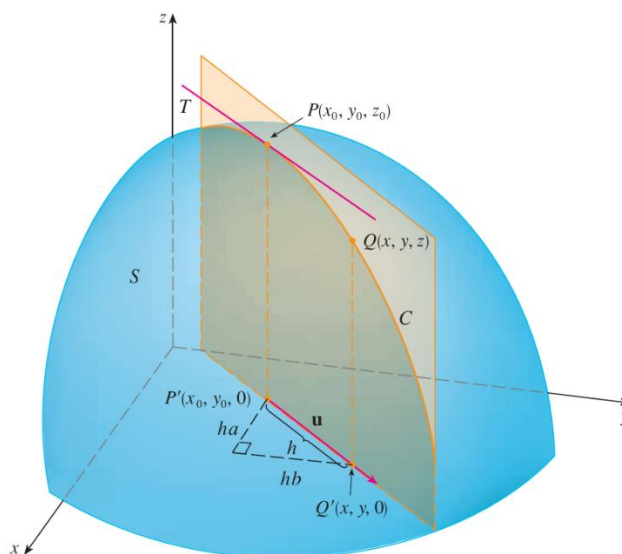


Figura 2-21: derivada direccional (Universidad Tecnológica Nacional (UTN), Facultad Regional La Plata (FRLP). (s.f.). TP8 [Material de curso]. Cátedra de Análisis 2. Recuperado de <http://analisis2.frlp.utn.edu.ar/tps/TP8.pdf>).

Si $Q(x, y, z)$ es otro punto sobre C y P', Q' , son las proyecciones de P, Q sobre el plano xy , entonces el vector es paralelo a u y entonces:

$$\vec{P'Q'} = h\vec{u} = \{ha, hb\}$$

En la ecuación (11) para algún escalar h . Por tanto, $x - x_0 = ha$, $y - y_0 = hb$, por lo que $x = x_0 + ha$, $y = y_0 + hb$, y Si tomamos el límite cuando $h \rightarrow 0$, obtenemos la razón de cambio de z con respecto a la distancia en la dirección de u , la cual se denomina derivada direccional de f en la dirección de u .

$$\lim_{h \rightarrow 0} \frac{\Delta z}{h} = \lim_{h \rightarrow 0} \frac{z - z_0}{h} = D_u f(x_0, y_0) = \lim_{h \rightarrow 0} \frac{f(x_0 + ha, y_0 + hb) - f(x_0, y_0)}{h} \quad (11)$$

Esto será útil para la minimización del error del modelo de red neuronal LSTM, red neuronal multicapa (MLP) y con función de pérdida de cuartil que se verá más adelante.

- Backpropagation

Según (Werbos, P.) “backpropagation a través del tiempo es una herramienta muy poderosa, con aplicaciones de reconocimiento de patrones, modelaje dinámico, análisis de sensibilidad, y el control de los sistemas a través del tiempo, entre otros. Puede ser aplicado a redes neuronales, modelos econométricos, estructuras lógicas difusas hasta modelo de fluidos dinámicos”.

Además, se menciona; en el aprendizaje supervisado, se trata de adaptar una red de inteligencia neuronal, para así, sus valores se acerquen a algún valor objetivo. El objetivo es adaptar los parámetros de la red, para así, su desempeño sea el correcto (Werbos, 1990).

Backpropagation es el motor del entrenamiento de una red neuronal. Utiliza derivadas parciales para poder reducir la función de pérdida, variando los pesos w de la red de manera iterativa. Busca los valores óptimos y direcciones (derivada direccional) que minimicen la función con el objetivo de encontrar el mínimo global o local de esta, lo que permite mejorar las predicciones. Esto se representa en la figura 2.22, donde se muestra a que valores de peso el valor de error (loss) se minimiza (mínimum point).

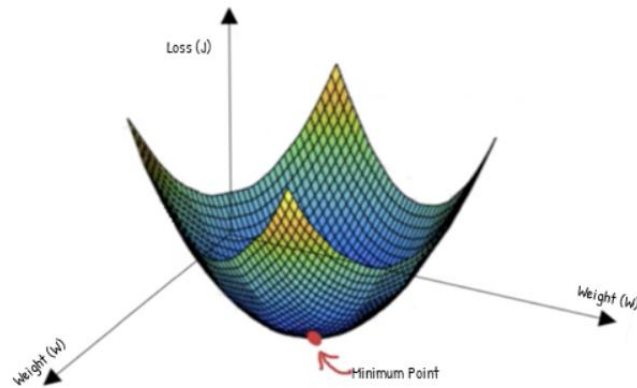


Figura 2-22: Representación gráfica en 3 dimensiones de la función de costo en función de sus pesos.

(Intellipaat. (s.f.). Back Propagation Algorithm. Intellipaat Blog. Recuperado de <https://intellipaat.com/blog/tutorial/artificial-intelligence-tutorial/back-propagation-algorithm/>)

El proceso de entrenamiento se ilustra mediante la Red Neuronal Multicapa (MLP) simplificada con una capa oculta, presentada en la Figura 2.23 presenta un output de valor γ cuyo valor contiene una desviación de la realidad. La manera de reducir el error es variando los pesos w como se mencionó antes.

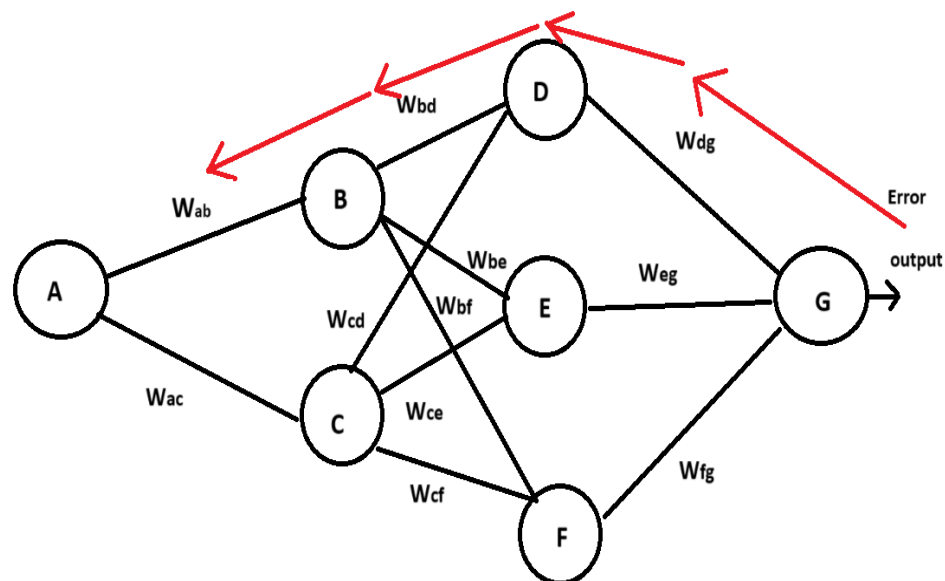


Figura 2-23: Representación de red neuronal con 1 capa oculta mostrando backpropagation. (Elaboración propia).

Se usan las derivadas parciales y la regla de la cadena para resolver esto. Se ve como varia el error “E” respecto al peso W_{ab} . El resultado es la tasa de cambio del error respecto a cada peso de la ecuación (12).

$\frac{\partial E}{\partial W_{ab}}$ es un valor numérico que representa la gradiente, el cual apunta a la dirección en el espacio de pesos donde la tasa de cambio del error se maximiza. Dado que el objetivo es reducir el error, se busca la dirección que lo minimice. Esto se logra al movernos dirección opuesta a la gradiente, agregando un signo negativo.

$$\frac{\partial E}{\partial W_{ab}} = - \frac{\partial \phi}{\partial W_{dg}} \cdot \frac{\partial W_{dg}}{\partial W_{bd}} \cdot \frac{\partial W_{bd}}{\partial W_{ab}} \quad (12)$$

Esto se repite para cada peso de cada capa de la red neuronal, buscando la mejor combinación que logré minimizar el error global del modelo. Este valor es únicamente numérico y solo indica la dirección que minimiza el error del modelo, aún hay que actualizar los valores de los pesos W_m .

- Descenso de gradiente

El descenso de gradiente es un algoritmo que, utilizando la tasa de cambio del error respecto a los pesos (BackPropagation), se ajusta estos mismos, mediante iteración.

“El descenso de gradiente es un método de optimización iterativo utilizado para minimizar funciones objetivo ajustando sus parámetros en la dirección opuesta al gradiente (o derivada) de la función. En el contexto del aprendizaje automático, este enfoque es fundamental para entrenar modelos, como redes neuronales, al reducir progresivamente una función de pérdida mediante actualizaciones basadas en la tasa de aprendizaje y la información local proporcionada por el gradiente” (Abdulkadirov & Lyakhov, 2023).

Una vez conocida esa tasa de cambio (backpropagation) se procede a actualizar el valor

de W_{ab} , siendo η conocido como tasa de aprendizaje que determinará la magnitud del paso con la que nos movemos en contra del gradiente, comúnmente con valor de 0,01 o 0,001. El gradiente indica a que dirección la función más crece, dado la lógica de ir en dirección opuesta al crecimiento, se agrega un signo negativo para ir en contra del gradiente como ya se mencionó.

La ecuación (13) representa la actualización de los pesos en cada iteración. Esta iteración es determinada por el experimentador o bien es programada a detenerse cuando la gradiente es lo suficientemente pequeño para no cambiar significativamente los valores de W_{ab} (explosión de gradiente).

$$W_{ab} = W_{ab} - \eta \cdot \frac{\partial E}{\partial W_{ab}} \quad (13)$$

De manera iterativa como se mencionó, se van ajustando el peso W_{ab} hasta converger en un mínimo global o local, tal como se muestra en la figura 2.24 y 2.25.

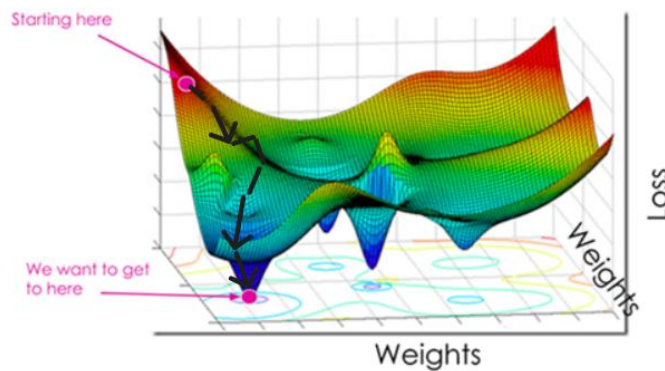


Figura 2-24: cambio del error mediante descenso del gradiente (Logongas.es. (s.f.). *Backpropagation Descenso de Gradiente*. Logongas.es. Recuperado de https://logongas.es/doku.php?id=clase:iabd:pia:2eval:tema07.backpropagation_descenso_gradiente).

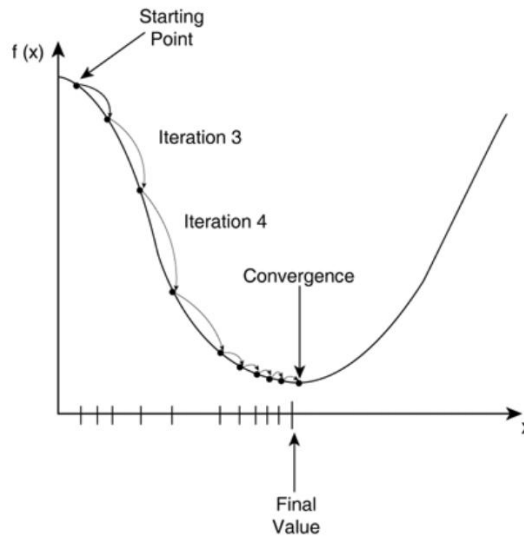


Figura 2-25: cambio de error mediante descenso del gradiente en 2 dimensiones (Sancho, F. (s.f.). Mejorando Gradiente Descendiente. Blog de F. Sancho, Universidad de Sevilla. Recuperado de https://www.cs.us.es/~fsancho/Blog/posts/Mejorando_Gradiente_Descendiente.md).

De manera generalizada, en redes de n capas. La ecuación pasa a notación matricial, modificando sus valores para todos los puntos a la vez.

A continuación, se presenta un problema relacionado con backpropagation y el descenso de gradiente: “Gradientes que desaparecen” y “explosión de gradiente”.

“Esto ocurre cuando el gradiente es demasiado pequeño. A medida que retrocedemos durante la retro propagación, el gradiente continúa haciéndose más pequeño, lo que hace que las primeras capas en la red aprendan más lentamente que las capas posteriores. Cuando esto sucede, los parámetros de ponderación se actualizan hasta que se vuelven insignificantes, es decir, 0, lo que da como resultado un algoritmo que ya no aprende” (IBM, 2023).

“Gradientes que explotan. Esto sucede cuando el gradiente es demasiado grande, creando así un modelo inestable. En este caso, las ponderaciones del modelo crecerán demasiado y, finalmente, se representarán como NaN” (IBM, 2023).

Un problema de esta técnica es el costo computacional que tiene. El descenso de gradiente toma todos los puntos de la base de datos, buscando mínimos globales. Esto lo hace deficiente para base de datos grandes.

- Modelo de descenso de gradiente

Cada modelo de descenso de gradiente soluciona un problema del planteado anteriormente. Descenso de gradiente estocástico con momento soluciona el problema de gradiente que desaparecen y RMSprop soluciona la explosión de gradiente, juntos conforman el modelo de descenso de gradiente ADAM que será usado para la red Long-Short Term Memory (LSTM) y Red Neuronal Multicapa (MLP). Además, estas técnicas reducen el tiempo de cálculo de los modelos, ahorrando estrés computacional.

- Descenso de gradiente estocástico (SGD)

En el descenso de gradiente tradicional, tal como se vio en la sección 2.2.3 G. Los gradientes se calculan basados en todo el conjunto de datos. Este proceso requiere una gran cantidad cálculos realizados por la computadora, lo que lo hace costoso en términos de tiempo.

El objetivo de este método de ajuste de los pesos es reducir este tiempo, encontrar los pesos óptimos y reducir la iteración. Gráficamente en la figura 2.26 se representa como una curva de nivel.

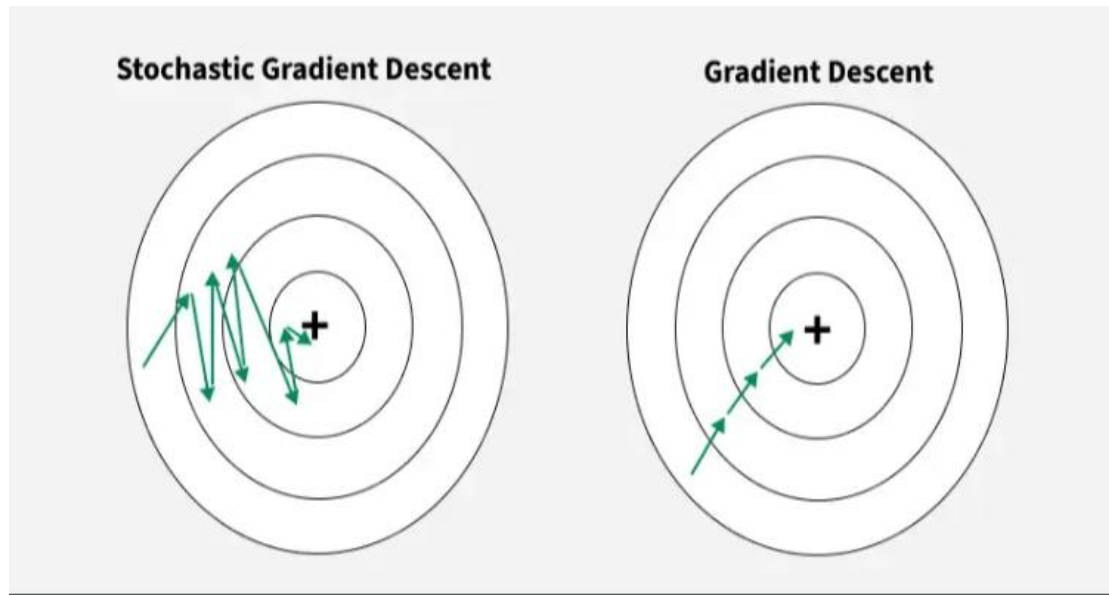


Figura 2-26: Comparación gráfica de descenso de gradiente estocástico vs descenso de gradiente (GeeksforGeeks. (s.f.). ML | Stochastic Gradient Descent (SGD). GeeksforGeeks. Recuperado de <https://www.geeksforgeeks.org/machine-learning/ml-stochastic-gradient-descent-sgd/>).

El descenso de gradiente toma cada dato e itera hasta llegar al resultado, mientras que el descenso de gradiente estocástico toma un mini-lote aleatorio (estocástico) del conjunto de datos de la forma (x_i, y_i) para realizar su iteración. Por cada paso toma un nuevo dato aleatorio de este mini-lote. Este proceso se le conoce como “zigzagado”.

Existe la probabilidad de encontrar un dato que nos acerque más rápido a la convergencia de la iteración que realizarlo para el conjunto de todos los datos. Esto se logra con la ecuación (14), utilizando no el conjunto de datos completos, sino que uno aleatorio.

$$w_{t+1} = w_t - \eta \cdot \frac{\partial E}{\partial w_{ab}}(w_t; x_i, y_i) \quad (14)$$

- SGD con momento

El SGD con momento es una variación del SGD, agregando un concepto de memoria en la dirección anterior del optimizado, ayudando a no caer en mínimos locales y llegar a converger en mínimos lo más globales posibles.

Este método, reflejado en la ecuación (15), se aprovecha del dato estocástico de cada iteración. Al ingresarlo a la gradiente y promediarlos, se logra tener este concepto de momento y una memoria de las direcciones anteriores. Este será controlado mediante un parámetro denominado Beta β .

$$v_{t+1} = \beta \cdot v_t + (1 - \beta) \cdot \frac{\partial E}{\partial W}(w_t; x_i, y_i) \quad (15)$$

Con v_{t+1} : velocidad acumulada en el paso t y β el Factor del momento. La actualización del valor w_{t+1} se ve reflejado en la ecuación (16).

$$w_{t+1} = w_t - v_{t+1} \quad (16)$$

- RMpSprop

RMSprop utiliza la base de SGD Momento para realizar un ajuste en su formulación. Su función es ajustar automáticamente la tasa de aprendizaje para cada parámetro, así según este, los saltos que ira dando en la función de perdida variará, reduciendo las oscilaciones en direcciones de alto gradiente.

$$E\left[\frac{\partial E^2}{\partial W}\right]_t = \gamma \cdot E\left[\frac{\partial E^2}{\partial W}\right]_{t-1} + (1 - \gamma) \cdot \left(\frac{\partial E}{\partial W}(w_t; x_i, y_i)\right)^2 \quad (17)$$

Con $E[\frac{\partial E^2}{\partial W}]_t$: promedio móvil del cuadrado de los gradientes.

γ : Factor de decaimiento.

Notar que, la formula (17) representa el momento de segundo orden de la formula (15).

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{E[\frac{\partial E^2}{\partial W}]_t + \epsilon}} \cdot \frac{\partial E}{\partial w}(w_t; x_i, y_i) \quad (18)$$

- Algoritmo ADAM

El algoritmo ADAM mezcla las ventajas de SGD Momentum para acelerar la convergencia y RMSprop para adaptar la tasa de aprendizaje por parámetros.

Se toma la formula (15) y la formula (17) para construir este algoritmo. Primero para SGD con momento, se busca corregir el sesgo.

Para esto se toma el momento de primer orden o formula (15), se cambiará los v por m para no confundir entre formulas;

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \frac{\partial E}{\partial w}(w_t; x_i, y_i) \quad \text{Con } \widehat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (19)$$

Y el momento de segundo orden o formula (17):

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot (\frac{\partial E}{\partial w}(w_t; x_i, y_i))^2 \quad \text{Con } \widehat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (20)$$

Luego la actualización de los pesos:

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t + \varepsilon}} \cdot \hat{m}_t \quad (21)$$

Con esto se tiene el modelo descenso de gradiente que se usará para la Red Neuronal Multicapa (MLP) y la red *Long-Short Term Memory* (LSTM) para la actualización de sus pesos.

2.2.4 Modelos de *Machine Learning*

A. Red LSTM

“LSTM es una excelente variación de RNN que puede evitar problemas de desaparición y explosión de gradientes” (Hochreiter y Schmidhuber, 1997).

Long Short Term Memory es un modelo de red neuronal profunda, usada para detección de patrones complejos de alta temporalidad, pudiendo procesar información de años. La ventaja de este modelo es el almacenamiento o eliminación de datos temporales según el algoritmo lo estime, aprovechando sus celdas de perceptrones de candidato para actualizar información o la eliminación de información con su celda de olvido.

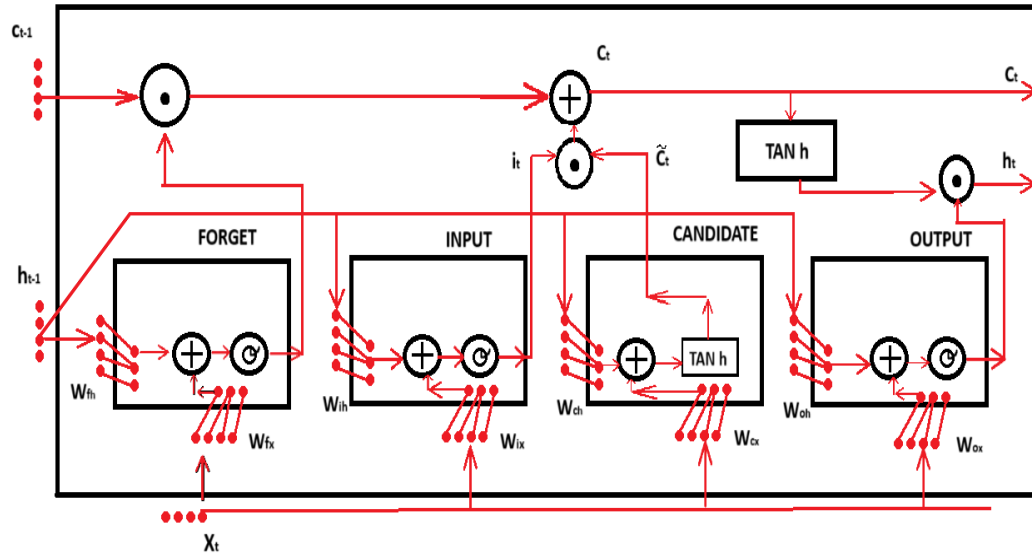


Figura 2-27: Célula LSTM (Elaboración propia).

El modelo LSTM se compone de cinco elementos:

- Puerta de entrada (*Input*)
- Puerta de salida (output)
- Puerta de olvido (Forget)
- Puerta de candidato (Candidate)
- Vectores de activación celular (Tangente Hiperbólica, función Sigmoide)

El modelo requiere que los datos estén en un cierto formato donde se mencionen algunos hiperparámetros que son a elección del desarrollador. Se define los siguientes hiperparámetros:

- *Batch Size*: Número de muestra de entrenamiento que el modelo procesa a la vez. Por ejemplo, si este es cinco, se entrenará los pesos con cinco datos a la vez de manera secuencial hasta terminar toda la muestra.
- *Timesteps*: Determina la cantidad de pasos de tiempo que se proporcionará a la red como secuencia de entrada, medido en la unidad de medida definida o del formato de

los datos (15 min, 1 hora, 24 horas, etc...).

- *Features*: Numero de variables que se usarán para entrenar el modelo.
- *Hidden size*: Cantidad de neuronas internas que tiene la capa LSTM, haciendo la función de memoria interna del modelo.

El modelo requiere un preprocesamiento de los datos para ser utilizado;

- X_t : Es la secuencia de entrada que la red procesa en el paso de tiempo t , el formato requerido que debe tener la entrada de este es (*Batch Size*, *Timesteps*, *Features*).
- h_{t-1} : El estado oculto *anterior*, es la memoria inmediatamente anterior que recuerda patrones hasta lo que se define en el *Batch Size*, su formato requerido como input es (*Batch Size*, *Hidden Size*).
- C_{t-1} : Memoria a *largo* plazo, siendo la memoria acumulada de la red, donde las puertas de olvido o entrada deciden cambiar su información. Su formato es (*Batch Size*, *Hidden Size*).

A continuación, se procede a definir las células internas de la red LSTM.

- Puerta de olvido
- Esta primera entrada se mide que tanta información del estado previo C_{t-1} recordar, multiplicando los pesos respectivos, las entradas del x y el estado previo más un *bias*. Luego, se le aplica la función sigmoide tal como se ve en la ecuación (22).

$$f_t = \sigma(w_{fx} \cdot x_t + w_{fh} \cdot h_{t-1} + b_f) \quad (22)$$

- Puerta de entrada

Esta celda determina que tanta nueva información de la puerta candidato se almacenará en el nuevo estado de memoria en la ecuación (23).

$$i_t = \sigma(w_{ix} \cdot x_t + w_{ih} \cdot h_{t-1} + b_f) \quad (23)$$

- Puerta de candidato

Esta celda determina la nueva información potencial a ser almacenada, su estructura es la de una red RNN. Se le aplica una función tangente hiperbólica en la ecuación (24).

$$\tilde{C}_t = \tanh(w_{ix} \cdot x_t + w_{ih} \cdot h_{t-1} + b_f) \quad (24)$$

Luego, este se actualiza mediante la ecuación (25):

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (25)$$

- Puerto de salida

Esta puerta decide que parte del estado de memoria se enviará como salida para la siguiente iteración (t+1) de la celda LSTM, representado en la ecuación (26).

$$O_t = \sigma(w_{ox} \cdot x_t + w_{oh} \cdot h_{t-1} + b_f) \quad (26)$$

Usando el estado de la celda C_t se determina el siguiente valor h_t con la ecuación (27).

$$h_t = O_t \odot \tanh C_t \quad (27)$$

A continuación, se explica el flujo de la celda LSTM de la figura 2.27:

Se presenta tres entradas de información:

- la información actual x_t .
- la información de las capas ocultas h_{t-1} (información obtenida de la célula anterior)
- estado de la celda C_{t-1} (aquella que se encarga de recordar información).

En el tiempo t entran como información de entrada el vector (x_t, h_{t-1}, C_{t-1}) , siendo procesada.

x_t y h_{t-1} entra en cada puerta (Forget, Input, Candidate, Output) para luego ser multiplicado por su correspondiente peso “W”.

Se procede a sumar y aplicar la función correspondiente (sigmoide o tangente hiperbólica) según fórmulas 22, 23, 24 y 26, concluyendo el procedimiento de cada celda. Notar que se ha aplicado la misma lógica de la sección 2.2.3 B.

El estado de celda C_{t-1} es reajustado mediante el ajuste por multiplicación punto a punto con la celda Forget.

La puerta Candidate determina el posible candidato (información) a ser usada para actualizar el estado de celda, antes de esto la puerta de entrada regula que tanto de la puerta candidate usar para actualizar mediante un producto punto a punto. Luego, se suma al estado de celda C_{t-1} actualizándose a C_t con la fórmula (25).

La puerta Output utiliza la tangente hiperbólica del estado de celda ya actualizado a tiempo t para hacer un producto punto a punto como se ve en la fórmula (27).

Finalmente se tiene el valor para la siguiente iteración.

Este flujo se repite hasta terminar de procesar toda la muestra. Una vez ya entrenado los pesos se puede realizar las predicciones.

B. *Multilayer Perceptron* (MLP) con la función de pérdida de cuantiles

La base de un perceptrón multicapa es la misma vista en la figura 2.20. Se agrega la función de pérdida de cuantiles.

En el artículo científico "Distribution Prediction of Strategic Flight Delays via Machine Learning Methods" menciona “el foco está en la predicción de intervalos (probabilística). La atención se centra en determinar si la distribución de la demora predicha está lo suficientemente cerca de la distribución de la demora real.” (Wang et al., 2022, p. 5).

En este modelo se busca predecir intervalos más que la predicción de puntos. cuantiles, siendo los de interés el Q_1 , Q_2 , Q_3 y Q_4 .

A diferencia de la función de costo MSE explicada en la sección 2.2.3 D. La función de costo que utiliza este modelo es la denominada Pérdida de Cuantiles, representado en la ecuación (28), que contiene dos partes, esta tomará la primera parte del sumando en el caso que el error encontrado entre el valor real y predicho sea del tipo “Error por sobrepredicción” ósea que el valor predicho sea mayor al valor real.

Se toma el segundo sumando en el caso que la comparación del valor real y predicho se subestime, una “error por subpredicción” o que el valor predicho sea menor al valor real.

$$Loss_{Quantile} = \frac{1}{m} \cdot \sum_{i=1}^m (\sum_{i: y_i < \hat{y}_i} (1 - \gamma) |y_i - \hat{y}_i| + \sum_{i: y_i > \hat{y}_i} \gamma |y_i - \hat{y}_i|) \quad (28)$$

Donde:

y : es el valor de ajuste del cuartil.

$\gamma \in [0,1]$.

m : número de muestras.

C. Random Forest

“Un bosque aleatorio es un clasificador de aprendizaje automático que comprende una estructura arborizada $\{h(x, k) \text{ } k= 1, 2 \dots\}$ con un vector independiente único $\{0(k) \text{ y una entrada para la clase más conocida de } x\}$ (D. Pijush., P. Shobhandeb., K. Asok. 2019).

La idea base de Random Forest es la de un árbol de decisión, donde para una entrada x_i de datos, pasa por un flujo de nodos interconectados con ramificaciones, realizándose preguntas para decidir si tomar el camino derecho o izquierdo, tal como se ve en la Figura 2.28.

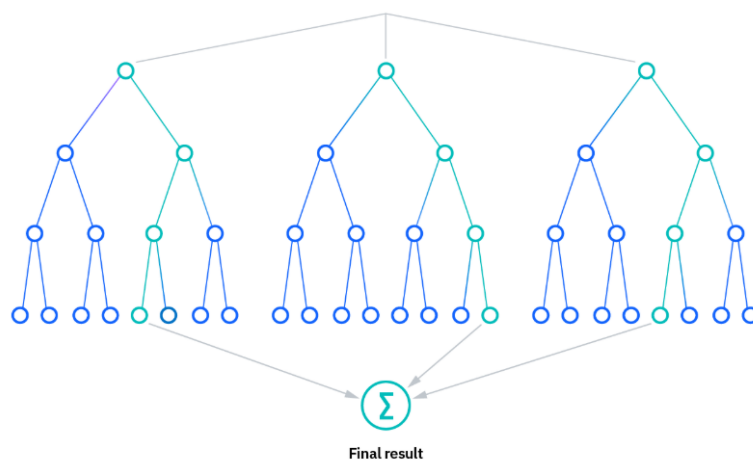


Figura 2-28: Modelo de aprendizaje automático supervisado *Random Forest* (IBM. (s.f.).

Classification Machine Learning. IBM. Recuperado de

<https://www.ibm.com/think/topics/classification-machine-learning>).

El árbol de decisión tiene problemas con el sobreajuste de los datos, esto quiere decir

que el modelo memoriza el comportamiento de una base de datos, en vez de aprender sus patrones. Por lo que, al mostrarle nuevos datos, este no logrará predecir y arrojará un valor erróneo.

Este problema encuentra solución con Random Forest, donde se toman N cantidad de árboles de decisión, con M profundidad, donde una vez el flujo del dato ha pasado por los N arboles de decisión y todos los pisos de profundidad, se realiza el promedio de estas obteniendo el resultado final. Este promedio es la característica clave que ayuda a evitar el sobreajuste.

Otra gran ventaja de este modelo de aprendizaje automático es su fácil entendimiento y la facilidad de identificar el flujo de datos, no así para los modelos de redes neuronales, donde el flujo se pierde en las capas ocultas.

Para la toma de datos se usa el denominado método Bagging o agregación de Bootstrap:

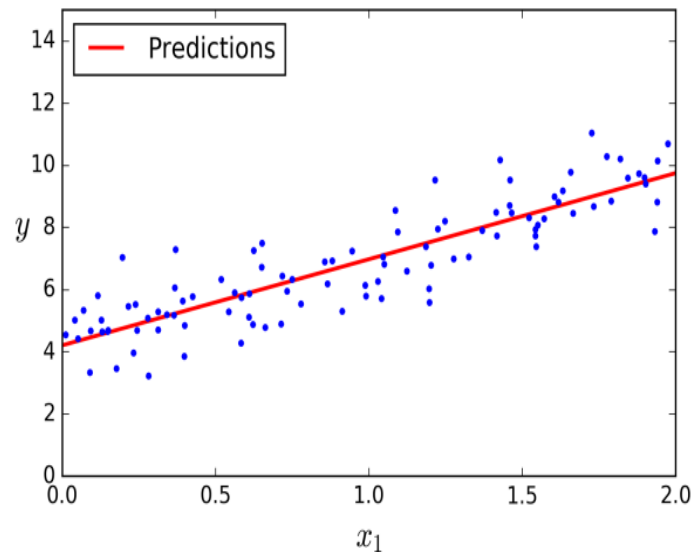
- 1- Selección de subconjunto: se forman subconjuntos aleatorios con remplazo de la base de datos.
- 2- Muestreo Bootstrap: cada árbol de decisión toma uno de estos subconjuntos de datos.
- 3- Formación de modelos independientes: en cada división de los nodos, solo se evalúa el subconjunto de datos con m columnas, ósea no toma todas las columnas para evaluar. Así, evitando que los árboles sean iguales y reducir la correlación entre ellos.
- 4- Votación mayoritaria: se promedia la predicción de cada árbol.

Las características de un árbol de decisión son:

- a. Número de estimadores: Número de árboles que construye el modelo antes de promediar.
- b. características máximas: Número máximo de características que un bosque aleatorio considera dividir un nodo, estimándolo con la función de costo MSE.

D. Regresión Lineal

La regresión lineal es una configuración de sumas ponderadas donde se busca en base a una o varias variables independientes explicar una variable dependiente, este se representa en la Figura 2.8.



Gráfica 2-8: Gráfico de regresión lineal,

Nota. Adaptado de *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, por A. Géron, 2017, O'Reilly Media (https://www.clc.hcmus.edu.vn/wp-content/uploads/2017/11/Hands_On_Machine_Learning_with_Scikit_Learn_and_TensorFlow.pdf).

La regresión lineal es una función compuesta por la variable predictora x , el valor predicho y , coeficientes β_0 y β_1 y, por último, un valor residual ε , tal como se ve en la ecuación (29)

$$y = \beta_0 + \beta_1 \cdot x + \varepsilon \quad (29)$$

En base a los datos introducidos a la ecuación (29) se nos presenta un valor predicho y_0 como resultado del ingreso de datos. Se usa el método de estimación de mínimos

cuadrados ordinarios para obtener los valores estimados de β_0 y β_1 .

Para ello se requiere derivar ese error respecto a los coeficientes β_0 y β_1 .

$$\text{Min } W_{\beta_0, \beta_1} = \sum_{i=1}^N (y_i - \beta_0 - \beta_1 \cdot x_i)^2 \quad (30)$$

Con la ecuación (30) para el parámetro de β_0 .

$$\frac{\delta W}{\delta \beta_0} = \sum_{i=1}^N -2(y_i - \beta_0 - \beta_1 \cdot x_i) = 0 \quad (31)$$

Con la ecuación (32) para el parámetro de β_1 .

$$\frac{\delta W}{\delta \beta_1} = \sum_{i=1}^N -2(y_i - \beta_0 - \beta_1 \cdot x_i) = 0 \quad (32)$$

Con esto se garantiza encontrar la combinación de β_0 y β_1 que minimice el valor residual (error).

E. Vector Support Regression

Vector support regression (SVR) utiliza los principios del perceptrón. A diferencia de métodos tradicionales como es la regresión lineal, este en vez de minimizar los valores del error, este busca minimizar el error dentro de un margen de tolerancia ϵ para evitar el sobreajuste de los datos.

La ecuación (28) es tanto la de la regresión lineal como el del perceptrón.

$$f(x) = \langle w, x \rangle + b \quad (33)$$

Se asigna un margen de tolerancia ϵ , evitando penalizar errores pequeños, considera lo siguiente:

Si el error $|y_i - f(x_i)| \leq \epsilon$, entonces no se considerará como una pérdida. Esto se ve en la Figura 2.29.

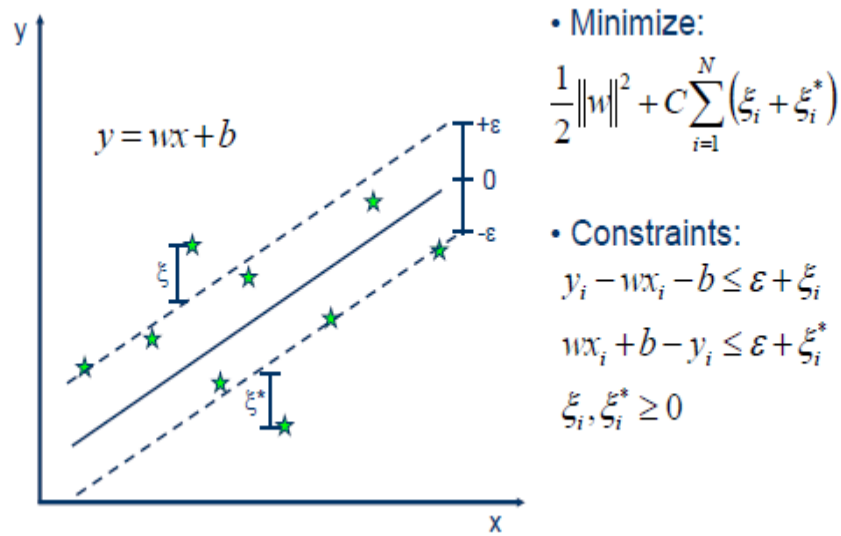


Figura 2-29: Modelo de *support vector regression* (Sayad, S. (s.f.). *Support Vector Machine Regression*. Sitio web de Saed Sayad. Recuperado de http://www.saedsayad.com/support_vector_machine_reg.htm).

Para permitir que algunos puntos violen el margen ϵ , evitando sobreajuste, se introducen variables de holgura:

$$\begin{cases} y_i - f(x_i) \leq \epsilon + \xi_i \\ f(x_i) - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases}$$

Para optimizar el SVR se realiza la minimización de la ecuación (34):

$$\min_{w,b,\xi_i,\xi_i^*} \frac{1}{2} \cdot \|w\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) \quad (34)$$

La particularidad de este método es la adopción de la función Kernel en el segundo multiplicando de la formula (35), esto para aquellos datos que no sean linealmente separable se le aplica una transformación no lineal. Es ayudar a darle mejor particularidad al dato. Al subir la dimensionalidad, tal como se ve en la Figura 2.30, el algoritmo puede entender que existen datos con otras características o grupo que tienen su propio comportamiento y los separa en una representación tridimensional.

$$f(x) = \sum_{i=1}^N (\alpha_i + \alpha_i^*) \cdot K(x_i, x) + b \quad (35)$$

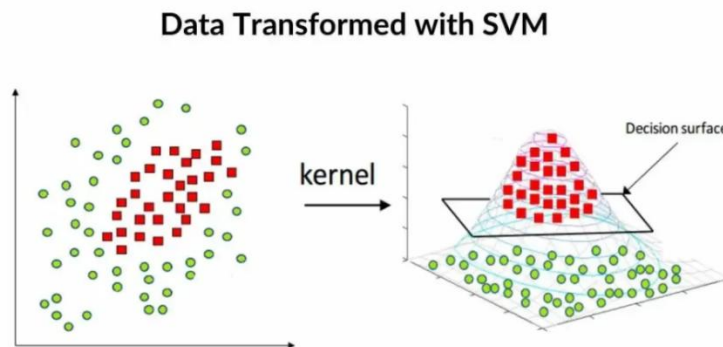


Figura 2-30: Demostración grafica de la transformación de datos a K-dimensionalidad (Spot Intelligence. (2024, 8 de mayo). *Support Vector Regression (SVR)*. Spot Intelligen).

F. Ridge Regression

El modelo de *Ridge Regression* se basa en la regresión lineal, donde se busca resolver el problema del sobre ajuste.

“La regresión de Ridge es una técnica de regularización estadística. Corrige el sobreajuste de los datos de entrenamiento en los modelos de Machine Learning” (IBM,

2024).

La primera sumatoria de la ecuación (36) corresponde a la suma de los residuos ya vistos en la formula (30) de regresión lineal y el segundo se le llama la penalización L2 sobre los coeficientes β_j , logrando hacer que el modelo no se apegue tanto a los datos de entrenamiento, evitar el sobreajuste y, cuando se enfrente a nuevos datos, logré entregar resultados más fidedignos.

A medida que λ crece, la regularización aumenta y los coeficientes tienden a reducirse hacia cero con el fin de evitar sobreajuste. Su potencial viene dado cuando es usado para muestras de datos y variables con colinealidad.

$$\text{Min } W_{\beta_0, \beta_1} = \sum_{i=1}^N (y_i - \beta_0 - \sum_{j=1}^p \beta_j \cdot x_{ij})^2 + \lambda \sum_{j=1}^p \beta_j^2 \quad (36)$$

G. K-Nearest *Neighbors* Regression (KNN)

Es un algoritmo de regresión de K vecinos más cercanos. Es un método de aprendizaje supervisado de tipo no paramétrico. Se basa en la distancia euclidiana para predecir el valor continuo de una variable objetivo a partir de la media de los valores de los k vecinos más próximos en el espacio de características, tal como se aprecia en la figura 2.31.

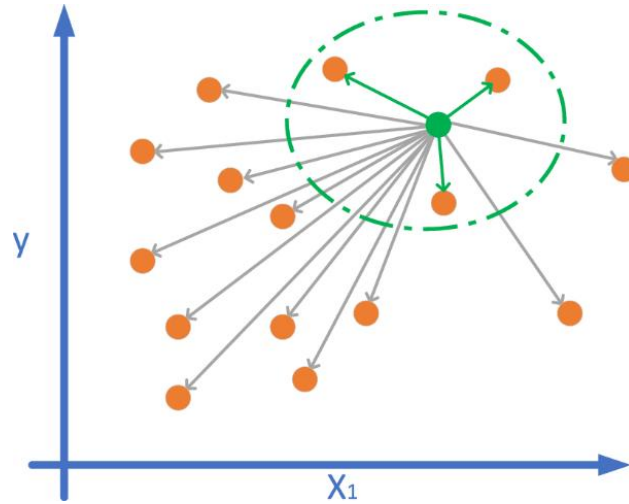


Figura 2-31: Grafico de K Nearest Neighbors de $K = 3$.

Nota. Adaptado de "Temporal prediction of traffic characteristics on real road scenarios in Amman por Qaddoura. R, 2022.

Se define un conjunto de observaciones $\{x, y\}$ donde se selecciona el número de K vecinos a tomar en consideración para hacer la predicción. Luego, se evalúa la distancia euclidiana entre ellos con la fórmula (37).

Una vez encontrado las distancias más próximas a ese valor, se realiza una media de ellos, siendo este el resultado predicho y_i .

$$d(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2} \quad (37)$$

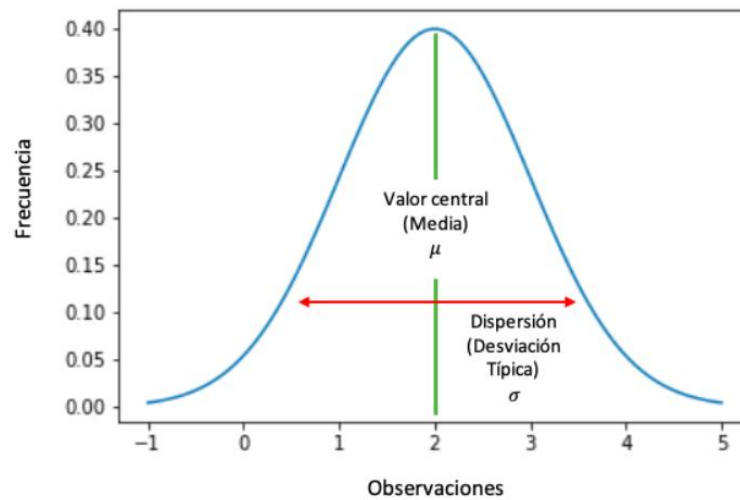
2.2.5 Distribuciones

A. Distribución Normal

Se dice que una variable aleatoria X se encuentra normalmente distribuida si su función de densidad de probabilidad está dada por:

$$f(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad (38)$$

Con parámetros el dato x , la media y la varianza μ, σ

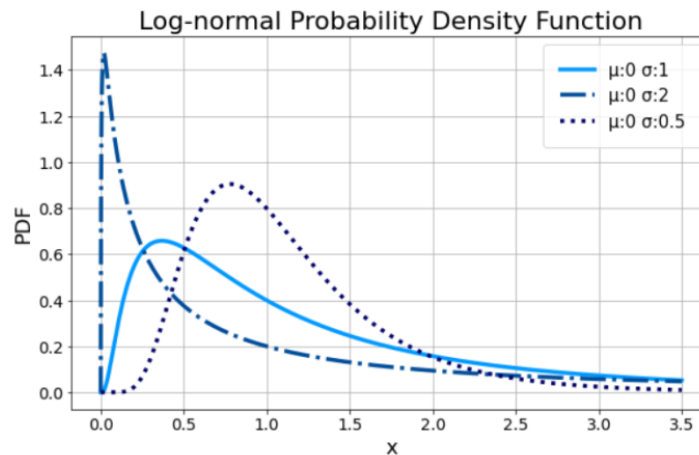


Gráfica 2-9: Distribución Normal Economipedia. (s.f.). *Distribución normal*. Economipedia. Recuperado de <https://economipedia.com/definiciones/distribucion-normal.html>.

B. Distribución Log-Normal

Se dice que una variable aleatoria X se encuentra log-normalmente distribuida si su función de densidad de probabilidad es sesgada hacia la derecha y está dada por:

$$(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{\ln x - \mu}{\sigma}\right)^2} \quad (39)$$

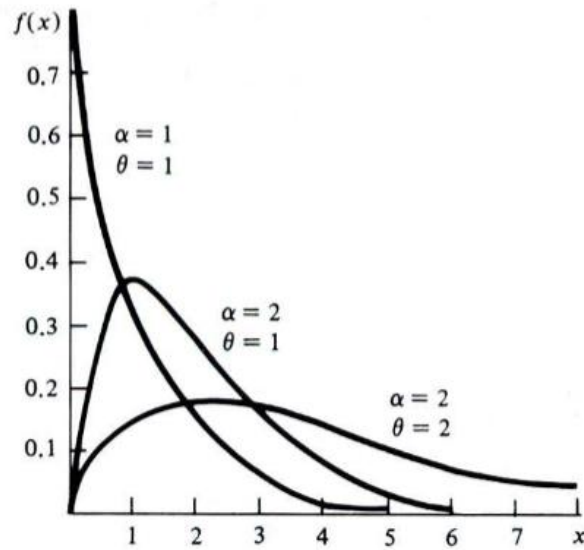


Gráfica 2-10: Distribución log-normal (Towards Data Science. (2021, 11 de mayo). Log-Normal Distribution: A simple explanation. Towards Data Science (Medium). <https://towardsdatascience.com/log-normal-distribution-a-simple-explanation-7605864fb67c/>).

C. Distribución Gamma

Se dice que la variable aleatoria X tiene distribución *gamma* si su función de densidad de probabilidad está dada por:

$$f(x; \alpha, \theta) = \begin{cases} \frac{1}{\Gamma(\alpha)\theta^\alpha} \cdot x^{\alpha-1} \cdot e^{-\frac{x}{\theta}} & x > 0, \quad \alpha, \theta > 0 \\ 0 & \text{Para cualquier otro valor} \end{cases} \quad (40)$$

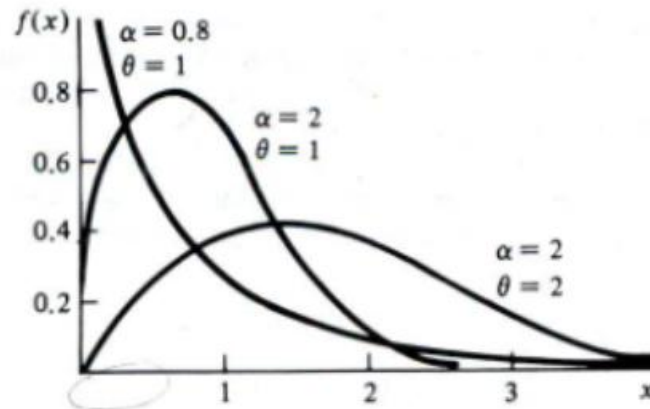


Gráfica 2-11: Distribución *gama* (Canavos, G. (1988). *Probabilidad y Estadística: Aplicaciones y Métodos* [Libro en PDF]. Recuperado de <https://gsosa61.wordpress.com/wp-content/uploads/2008/03/10-canavos-g-probabilidad-y-estadistica-aplicaciones-y-metodos.pdf>).

D. Distribución Weibull

Típicamente usada en contextos de esfuerzos de materiales. Se dice que una variable aleatoria X tiene una distribución de *Weibull* si su función de densidad de probabilidad está dada por:

$$f(x; \alpha, \theta) = \begin{cases} \frac{\alpha}{\theta^\alpha} \cdot x^{\alpha-1} \cdot e^{-\left(\frac{x}{\theta}\right)^\alpha} & x > 0, \quad \alpha, \theta > 0 \\ 0 & \text{Para cualquier otro valor} \end{cases} \quad (41)$$



Gráfica 2-12: Distribución Weibull (Canavos, G. (1988). *Probabilidad y Estadística: Aplicaciones y Métodos* [Libro en PDF]. Recuperado de <https://gsosa61.wordpress.com/wp-content/uploads/2008/03/10-canavos-g-probabilidad-y-estadistica-aplicaciones-y-metodos.pdf>).

2.2.6 Indicadores y transformaciones

A. Coeficiente de determinación

Denotado como R^2 , es una medida estadística que indica la proporción de la varianza de la variable dependiente (Y) que es explicada por el modelo de regresión en relación con la variable independiente (X).

Se detalla su fórmula:

$$R^2 = \frac{\text{Variabilidad explicada por el modelo}}{\text{variabilidad total}} = \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y}_i)^2} \quad (42)$$

Donde,

y_i : Valor observado

$\hat{y}_i$: Valor predicho

$\bar{y}_i$: Media de valores observados

B. Error Absoluto medio

Una métrica utilizada para evaluar la precisión de un modelo de predicción, especialmente en problemas de regresión.

Se calcula como el promedio de las diferencias absolutas entre los valores reales y los valores predichos:

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^n |y_i - \hat{y}_i| \quad (43)$$

donde:

y_i : *Valor observado.*

$\hat{y}_i$: *Valor predicho.*

n : *Número de observaciones.*

C. Coeficiente de correlación de Pearson

Mide estadísticamente la fuerza y dirección de la relación lineal entre dos variables cuantitativas continuas entre -1 y +1.

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \cdot \sum_{i=1}^n (y_i - \bar{y})^2}} \quad (44)$$

D. Ciclicidad

Las variables temporales de una base de datos tales como “Día del mes” o “mes del año” cuentan con problema en el entrenamiento de los modelos predictivos, en especial las redes neuronales y K vecinos más cercanos.

Estos modelos no logran comprender la distancia temporal entre un mes y otro. Por

ejemplo, diciembre siendo el mes 12 y enero el mes 1 se encuentran a una distancia euclidiana de 11 meses, siendo erróneo. Por ello, se requiere de herramientas matemática para capturar la ciclicidad de los datos. Se realiza transformaciones con funciones sinusoidales, llevando los datos temporales a un campo vectorial donde serán representados en ese espacio bidimensional.

Según el paper “*An experimental assessment of treatments for cyclical data*” menciona: “una alternativa recomendada de codificación es el sin/cos, el cual integra valores ya sea de hora del día, semana del año, mes del año y trimestre es reposicionada en dos variables” (2021).

$$\left[\sin \left(\frac{2\pi h}{\text{periodo}} \right), \cos \left(\frac{2\pi h}{\text{periodo}} \right) \right] \quad (45)$$

E. Normalización

La normalización de datos se define como el proceso de transformar los valores de un conjunto de datos en la misma escala, a menudo aplicado como un paso de preprocesamiento estándar en modelos para mejorar el rendimiento de la clasificación, particularmente en casos de alta variabilidad, como las señales EEG. (Elsevier, 2024).

Tecnologías como las redes neuronales requieren una transformación de las variables continuas, estas identifican aquellas variables de mayores escalas como de mayor importancia, dándoles mayores pesos w_i a estas, aun si éstas no lo son. Por ello, la importancia de escalar todas las variables.

Para esto, se realiza la siguiente operación, siendo su codominio entre 0 y 1.

$$X_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (46)$$

2.2.7 Metodología ETL

La metodología ETL es usado para el procesamiento de la información en un flujo continuo.

“Extracción, transformación y carga (ETL) es una canalización de datos que se usa para recopilar datos de varios orígenes, transforma los datos según las reglas de negocio y los carga en un almacén de datos de destino. El trabajo de transformación ETL tiene lugar en un motor especializado y, a menudo, implica el uso de tablas de almacenamiento provisional para conservar los datos temporalmente a medida que estos se transforman y, finalmente, se cargan en su destino” (Azure, s.f.).

2.2.8 Propuesta Metodológica

A continuación, se presenta la propuesta metodológica:

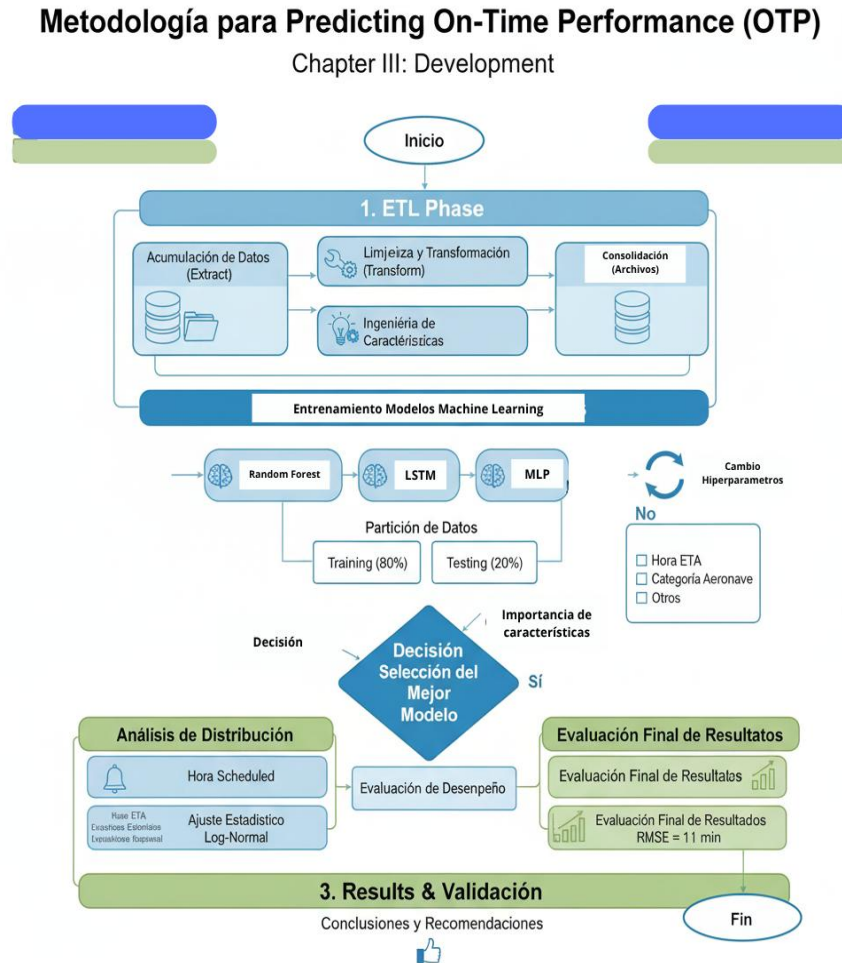


Figura 2.32: Diagrama de flujo (OpenAI, 2025).

1. Fase ETL (*Extract, Transform, Load*)

- a. Corresponde a la obtención de datos operacionales de vuelos (itinerarios) y la información externa (meteorología) que constituye la base de datos.
- b. Los datos son procesados, realizando ingeniería de características, eliminando columnas innecesarias como: filas nulas, valores atípicos y limitación de los datos de OTP según lo establecido en -200 a 180 minutos.

- c. Se consolida la base de datos, dejando un archivo limpio y listo para su uso.

2. Entrenamiento *de* modelos *Machine Learning*.

- Se entrenan diversos algoritmos de *Machine Learning* como *Random Forest*, *LSTM*, regresión lineal, *MLP*, *K Nearest Neighbors Regression*, *SVR* y *ridge regression*.
- Se realiza la partición de datos de entrenamiento (80%) y datos de testeo (20%).
- Análisis de importancia de variables del modelo para el entendimiento en la explicación del modelo.
- Si la evaluación no cumple con expectativas, se procede a cambiar hiperparámetros y volver a entrenar los modelos.

3. Decisión y evaluación final de resultados

- a. Evaluación del desempeño con los indicadores RMSE, MSE y R^2 .
- b. Análisis de distribución OTP.
- c. Ajuste a la distribución que mejor se adapte.

3 Capítulo III: Desarrollo

En esta parte del capítulo se entrega los métodos usados para realizar tanto la limpieza de datos, su transformación, estadística descriptiva, definiciones de variables, selección de herramienta a utilizar, presentar los resultados obtenidos y validación.

3.1 Metodología ETL

3.1.1 Acumulación de datos (Extract)

En esta parte de la metodología, se dio un periodo de 3 meses para la búsqueda de información. Diversos actores participaron y/o colaboraron con información de procesos, base de datos y de experiencias para determinar aquellas variables a tomar en cuenta y, que podrían explicar el desempeño del indicador On-Time Performance.

La empresa privada Nueva Pudahuel junto con el Departamento de Ingeniería de Operaciones, el Departamento de Aeródromos y Servicios Aeronáuticos, en la Oficina de Gestión de Afluencia de Tránsito Aéreo de la DGAC y datos meteorológicos del mismo, alimentaron la base de datos con la que el modelo de Machine Learning fue entrenado.

A continuación, se presenta la estructura de las bases de datos:

- La base de datos de Pudahuel se estructura con cuatro archivos .xlsx denominado “Datos NPU Parte 1”, habiendo 4 partes. Estos archivos contienen datos operacionales para cada vuelo desde 01-04-2023 hasta el 30-04-2025 estructurando su base de datos de la siguiente manera:
 - Fecha de operación
 - Año

- Mes de operación
- Día de operación
- Hora fija
- Aerolínea
- Aeropuerto
- Descripción tipo aeronave
- Categoría Aeronave
- Svc type
- Numero de vuelo
- Matricula aeronave
- Asientos2
- Tipo Stand
- Stand
- Puerta de Embarque
- Operación
- Negoció
- Scheduled
- Estimado
- Actual
- Delta
- ON Block
- OFF Block
- Cantidad de operaciones
- Cantidad de pasajeros totales
- Distancia
- Incumplimiento

Se realizó una consulta en el mes de marzo en a base de datos de la DGAC. Este contiene dos partes, la primera contiene datos meteorológicos obtenidos de la página oficial <https://climatologia.meteochile.gob.cl/>.

La base de datos parte desde abril del 2023 hasta abril del 2025, conteniendo 6 carpetas separadas en agua caída, nubosidad, presión y humedad, temperatura, viento y visibilidad, cada carpeta con 29 archivos .csv separado por mes.

Características de la estación meteorológica:

- Ubicación: Pudahuel, Santiago.
- Altura: 474 mts.
- Coordenadas: -33.37833°, -70.78778°.

La base de datos del departamento de aeródromos y servicios aeronáuticos, en la oficina de gestión de afluencia de tránsito aéreo de la DGAC otorgó base de datos desde abril del 2023 hasta abril del 2025 con datos operacionales complementarios a los entregados por la concesionaria Nueva Pudahuel.

- Date
- ARR/DEP
- CALLSIGN
- Type of ACFT
- FROM/TO
- RWY (S o N)
- RWY
- ETD
- ETE
- ETA
- EO
- TIME
- TAXI IN
- TAXI OUT
- PARK

Estos datos sirvieron para comprobar la calidad de los datos del aeropuerto, siendo estos, alineados a los datos del departamento de aeródromos y servicios aeronáuticos.

3.1.2 Procesamiento de datos (Transform)

En este paso se realiza el procesamiento de la información entregada por los distintos involucrados para consolidar la información en una base de datos única. además, se realiza procedimientos matemáticos para ajustar los datos, eliminar nulos y outliers, mediante el software Python.

Los datos climatológicos separados por tipo de variable contienen 29 archivos .csv, lo primero es consolidar los 29 archivos en un único archivo. Esto se realiza para agua caída, nubosidad, presión y humedad, temperatura, viento y visibilidad. Por último, se realiza una fusión de datos para unificarlos en un único archivo llamado “datos_meteorologicos_completos”.

Se procesa los distintos archivos con formato .csv y junto a la herramienta de programación Python, se logra consolidar 175 archivos de agua caída, nubosidad, presión, humedad, temperatura, viento, visibilidad e itinerario.

Se realiza una limpieza de columnas que no se usarán para los datos operacionales. además, se configura los formatos de columnas tipo fecha, string, booleano y numérico para finalmente con la función “Merge”, se realiza la fusión de los datos entre los climatológicos y operacionales.

Se crean columnas de temporalidad de los datos para que los modelos de Machine Learning reconozcan patrones de tiempo, creando las variables; mes, día del mes, día de la semana, hora scheduled, hora actual, hora real y la variable a predecir OTP. Aún estas variables deben ser transformadas con funciones sinusoidales para capturar su

temporalidad.

Para el cálculo del OTP, este se calcula según la ecuación (1).

Mediante “.dropna”, se eliminan todos los datos que se encuentren nulos. Se identifica aquellas variables que son categóricas y se le aplica una transformación con la librería de sklearn con la función LabelEncoder para transformarlos en variables categóricas o también denominadas dummy.

Se aplica a toda variable temporal creada las transformaciones con funciones sinusoidales creando los valores temporales:

- Trimestre_sin
- Trimestre_cos
- Mes_sin
- Mes_cos
- Día_semana_sin
- Día_semana_cos
- Día_mes_sin
- Día_mes_cos
- Hora_scheduled_sin
- Hora_scheduled_cos
- Hora_ETA_sin
- Hora_ETA_cos

Además, a todas las variables numéricas se aplica una normalización entre 0 y 1, siendo:

- Cantidad_Pasajeros_Totales
- p0
- tMax12Horas

- dd02Minutos
- ff02Minutos
- vis2Minutos
- Distancia Recorrida

Se limita la base de datos para OTP según los datos *outliers*. Se busca eliminar todo ruido que los datos pueda tener, errores de tiempos informado por la torre de control o marcado por el agente de CCO:

- $OTP > -200$ minutos
- $OTP \leq 180$ minutos

3.2 Definición de variables

3.2.1 Base de datos

En la Tabla 3.1 se presentan las variables de la base de datos después de la metodología ETL y una descripción.

Variables	Tipo de variable	descripción
Mes Sin	Numérica	mes del año en función Sin
Mes Cos	Numérica	mes del año en función Cos
Día del mes Sin	Numérica	día del mes del año en función Sin
Día del mes Cos	Numérica	día del mes del año en función Cos
Día de la semana Sin	Numérica	día de la semana
Día de la semana Cos	Numérica	día de la semana
Trimestre Sin	Numérica	trimestre del año en función sin
Trimestre Cos	Numérica	trimestre del año en función cos
Año	Numérica	Año de operación
País	Categórica	País de inicio del vuelo
Aerolínea	Categórica	aerolínea operando
Aeropuerto	Categórica	aeropuerto que viene o va
Matricula Aeronave	Categórica	matricula de la aeronave
Tipo de stand	Categórica	stand remoto o contacto (puente de embarque)
Stand	Categórica	número del stand
Negocio	Categórica	vuelo nacional o internacional
Operación	Categórica	operación de salida o llegada
Numero de vuelo	Categórica	número de vuelo
Cantidad de pasajeros totales	Numérica	cantidad de pasajeros en el vuelo
Distancia Recorrida	Numérica	Distancia en Km del vuelo
Temperatura máxima de hace 12 horas	Numérica	temperatura máxima hace 12 horas
Dirección del viento	Numérica	dirección de viento
Magnitud del viento	Numérica	Magnitud del viento
Presión	Numérica	Presión
Visibilidad hace 2 minutos	Numérica	Visibilidad hace 2 minutos
Hora Scheduled Sin	Numérica	hora programada en sistema militar o formato 24 horas en función Sin (para llegada y salida)
Hora Scheduled Cos	Numérica	hora programada en sistema militar o formato 24 horas en función Cos (para llegada y salida)
Hora ETA Sin	Numérica	hora estimada en sistema militar o formato 24 horas en función Sin (para llegada y salida)
Hora ETA Cos	Numérica	hora estimada en sistema militar o formato 24 horas en función Cos (para llegada y salida)
Hora Scheduled	Numérica	hora programada en sistema militar o formato 24 horas
Hora ETA	Numérica	hora estimada en sistema militar o formato 24 horas

OTP	Numérica	on time performance
-----	----------	---------------------

Tabla 3-1: Descripción de las variables usadas para entrenar los modelos de machine learning (Elaboración propia).

El conjunto cuenta con 31 variables y 138.600 filas, representando los vuelos programados desde abril del 2023 hasta abril del 2025. Cada columna tiene datos ya sean continuos o categóricos (escritos, por ejemplo: Aeropuerto Mataverí). A los categóricos se reemplaza el escrito por un número que lo identifica, categorizándolo.

La base de datos se construye por distintos tipos de variables separados por:

- Variables temporales.
- Variables categóricas.
- Variables climatológicas.
- Variables operacionales.

En la tabla 3.2 se muestra aquellas variables que son del tipo temporales. Ósea, variables que capturar fechas como meses, días, trimestres o el año, siendo transformadas con funciones seno y coseno para capturar ciclicidad de los datos.

Variables	Tipo de variable	de descripción
Mes Sin	Numérica	mes del año en función Sin
Mes Cos	Numérica	mes del año en función Cos
Día del mes Sin	Numérica	día del mes del año en función Sin
Día del mes Cos	Numérica	día del mes del año en función Cos
Día de la semana Sin	Numérica	día de la semana
Día de la semana Cos	Numérica	día de la semana
Trimestre Sin	Numérica	trimestre del año en función sin
Trimestre Cos	Numérica	trimestre del año en función cos
Año	Numérica	Año de operación

Tabla 3-2: Variables temporales de la base de datos (Elaboración Propia).

En la tabla 3.3 se muestra aquellas variables que son del tipo categóricas. Estas capturan variables que caracterizan a un vuelo singular. Estas variables se encuentran en formato escrito, esto quiere decir que son cualitativa. Por ejemplo, un vuelo que viene del país Argentina, de la aerolínea Sky Airline, del aeropuerto de Ezeiza. Estas características son únicas del vuelo y deben ser capturadas, para eso para cada tipo de cualidad se reemplaza por un término numérico, siendo único e identificador de la cualidad.

Variables	Tipo de variable	descripción
País	Categórica	País de inicio del vuelo
Aerolínea	Categórica	aerolínea operando
Aeropuerto	Categórica	aeropuerto que viene o va
Matricula Aeronave	Categórica	matricula de la aeronave
Tipo de stand	Categórica	stand remoto o contacto (puente de embarque)
Stand	Categórica	número del stand
Negocio	Categórica	vuelo nacional o internacional
Operación	Categórica	operación de salida o llegada
Numero de vuelo	Categórica	número de vuelo

Tabla 3-3: Variables categóricas de las bases de datos (Elaboración Propia).

En la tabla 3.4 se muestra aquellas variables que son del tipo climatológicas. Describe características meteorológicas del aeropuerto Arturo merino Benítez, según la disponibilidad y calidad de datos que se obtuvo en la base de datos de la DGAC.

Variables	Tipo de variable	descripción
Temperatura máxima de hace 12 horas	Numérica	temperatura máxima hace 12 horas
Dirección del viento	Numérica	dirección de viento
Magnitud del viento	Numérica	Magnitud del viento

Presión	Numérica	Presión
Visibilidad hace 2 minutos	Numérica	Visibilidad hace 2 minutos

Tabla 3-4: Variables climatológicas de la base de datos (Elaboración Propia).

En la tabla 3.5 se muestra aquellas variables que son del tipo operacionales. Caracterizado por datos de llegadas o salidas programadas, estimadas, distancias y cantidad de pasajeros.

Variables	Tipo de variable	descripción
Cantidad de pasajeros totales	Numérica	cantidad de pasajeros en el vuelo
Distancia Recorrida	Numérica	Distancia en Km del vuelo
Hora Scheduled Sin	Numérica	hora programada en sistema militar o formato 24 horas en función Sin (para llegada y salida)
Hora Scheduled Cos	Numérica	hora programada en sistema militar o formato 24 horas en función Cos (para llegada y salida)
Hora ETA Sin	Numérica	hora estimada en sistema militar o formato 24 horas en función Sin (para llegada y salida)
Hora ETA Cos	Numérica	hora estimada en sistema militar o formato 24 horas en función Cos (para llegada y salida)
Hora Scheduled	Numérica	hora programada en sistema militar o formato 24 horas
Hora ETA	Numérica	hora estimada en sistema militar o formato 24 horas
OTP	Numérica	on time performance

Tabla 3-5: Variable operacionales de la base de datos (Elaboración propia).

Se realiza una partición de los datos, uno denominado datos de entrenamiento (x_{train} , y_{train}) y los datos de testeo de precisión de los modelos de *Machine Learning* (x_{test} , y_{test}).

- Los datos de entrenamiento cuentan con 31 variables con 110.880 filas (x_{train}) y 1 variable con 110.800 filas. (y_{train}).
- Los datos de testeo cuentan con 31 variables con 27.720 filas. (x_{test}) y 1 variable con 27.720 filas. (y_{test}).

Utilizando los datos de entrenamiento (X_{train}, Y_{train}) se procede a aplicar los modelos de Machine Learning uno a uno en esta fase, aprendiendo los patrones de la base de datos. Una vez terminado el entrenamiento, se les entrega a los modelos los valores de testeo (X_{test}). Estos generan valores de predicción (Y_{pred}) para el testeo.

Finalmente, se realiza una comparación entre el valor predicho (Y_{pred}) y el valor real (Y_{test}). Esta comparación ayuda a entender el rendimiento del modelo. Este flujo se representa en la Figura 3.1.

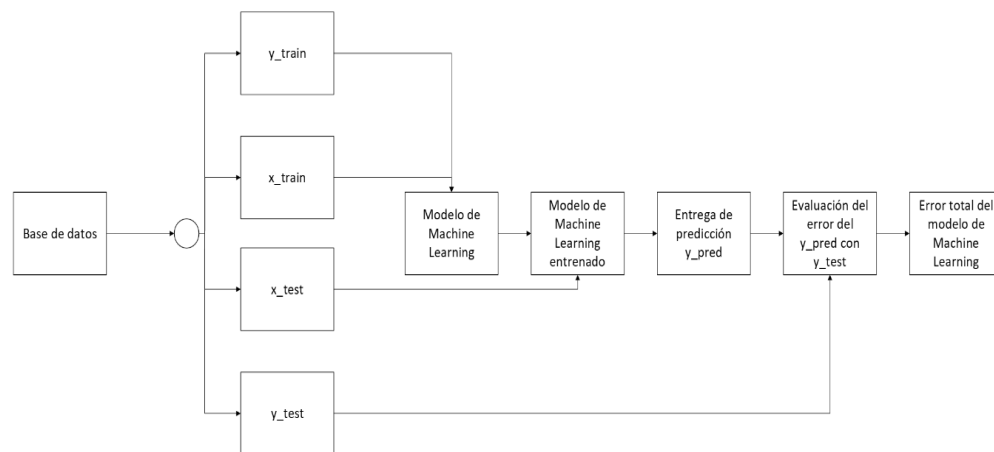


Figura 3-1: Mapa de proceso del entrenamiento del modelo de Machine Learning (Elaboración propia).

3.2.2 Variables de entrada

La tabla 3.6 muestra todas las variables de entrada que cada modelo de ML tendrá. Se realizó las transformaciones correspondientes al formato que requiere cada modelo. La tabla se compone de 31 variables de entrada, elegidas según disponibilidad de datos

y nulos que contenían, para el modelo de red neuronal multicapa con función de pérdida de cuartil y, para los otros modelos se usan solamente 29 variables, omitiendo hora_scheduled y hora_ETA.

Variables	Tipo de variable	descripción
Mes Sin	Numérica	mes del año en función Sin
Mes Cos	Numérica	mes del año en función Cos
Día del mes Sin	Numérica	día del mes del año en función Sin
Día del mes Cos	Numérica	día del mes del año en función Cos
Día de la semana Sin	Numérica	día de la semana
Día de la semana Cos	Numérica	día de la semana
Trimestre Sin	Numérica	trimestre del año en función sin
Trimestre Cos	Numérica	trimestre del año en función cos
Año	Numérica	Año de operación
País	Categórica	País de inicio del vuelo
Aerolínea	Categórica	aerolínea operando
Aeropuerto	Categórica	aeropuerto que viene o va
Matricula Aeronave	Categórica	matricula de la aeronave
Tipo de stand	Categórica	stand remoto o contacto (puente de embarque)
Stand	Categórica	número del stand
Negocio	Categórica	vuelo nacional o internacional
Operación	Categórica	operación de salida o llegada
Numero de vuelo	Categórica	número de vuelo
Cantidad de pasajeros totales	Numérica	cantidad de pasajeros en el vuelo
Distancia Recorrida	Numérica	Distancia en Km del vuelo
Temperatura máxima de hace 12 horas	Numérica	temperatura maxima hace 12 horas
Dirección del viento	Numérica	dirección de viento
Magnitud del viento	Numérica	Magnitud del viento
Presión	Numérica	Presión
Visibilidad hace 2 minutos	Numérica	Visibilidad hace 2 minutos
Hora Scheduled Sin	Numérica	hora programada en sistema militar o formato 24 horas en función Sin (para llegada y salida)
Hora Scheduled Cos	Numérica	hora programada en sistema militar o formato 24 horas en función Cos (para llegada y salida)
Hora ETA Sin	Numérica	hora estimada en sistema militar o formato 24 horas en función Sin (para llegada y salida)
Hora ETA Cos	Numérica	hora estimada en sistema militar o formato 24 horas en función Cos (para llegada y salida)
Hora Scheduled	Numérica	hora programada en sistema militar o formato 24 horas
Hora ETA	Numérica	hora estimada en sistema militar o formato 24 horas

Tabla 3-6: Descripción de las variables de entrada para entrenar los modelos de Machine Learning (Elaboración propia).

3.2.3 Variables de salida

La variable de salida es el resultado del modelo de *Machine Learning*, el *On-Time Performance* saliente será en formato numérico y en unidad de minutos, tal como lo indica la tabla 3.7.

Variable	Tipo de variable	descripción
OTP	Númerica	on time performance

Tabla 3-7: Descripción de la variable de salida (Elaboración propia).

3.3 Estadística

En esta sección se presentan características estadísticas de la base de datos, enfocándose en el comportamiento individual de las variables, la correlación lineal entre ellas y, el comportamiento de la gráfica de densidad del On-Time Performace.

La matriz de correlación (véase en anexo B.1) se construyó utilizando las siguientes variables claves son: mes, día_mes, día_semana, cantidad_pasajeros_totales, tMax12horas, dd02Minutos, ff02Minutos, p0, vis2Minutos, Hora_Scheduled, Hora_ETA, distancia_recorrida y OTP.

Se observa una correlación de 0,44 entre la distancia recorrida y la cantidad de pasajeros. Este resultado puede explicarse porque las aeronaves de mayor envergadura tienen mayor capacidad de asientos para más pasajeros, mayores tanques de combustible, permitiendo realizar vuelos de mayor distancia.

La hora_Scheduled y la hora_ETA tienen una correlación de un 0.88, lo que implica un problema de dependencia entre variables, lo que puede ocasionar problemas de rendimiento en los modelos.

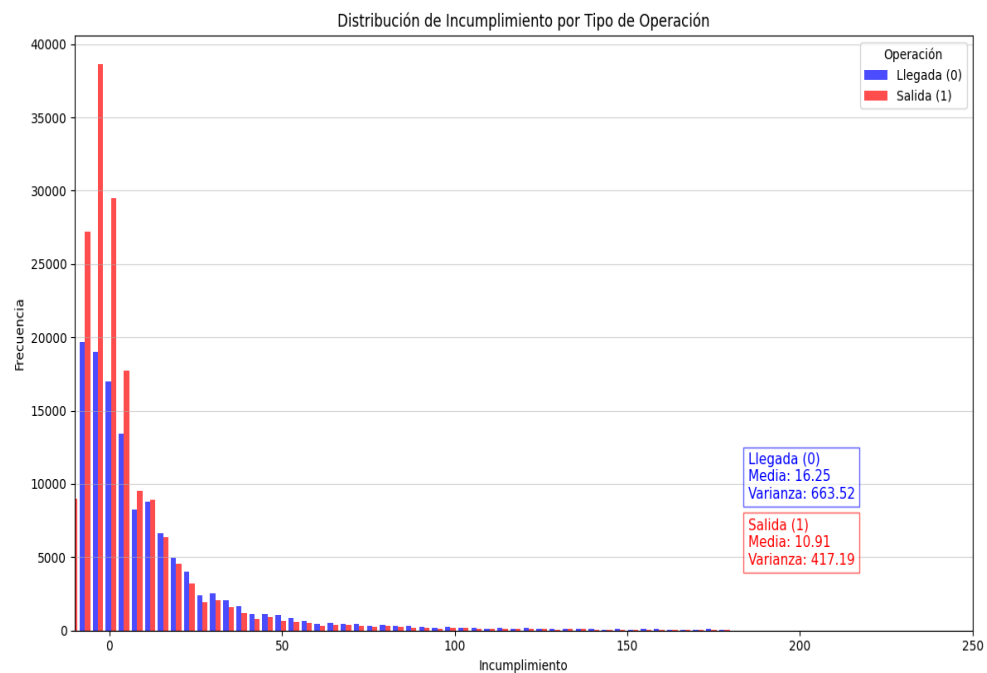
Esta correlación se explica debido a la lógica operacional de esa información. Las aerolíneas envían sus itinerarios con la intención de vuelo 30 días antes con su

respectiva hora_Scheduled. Durante el transcurso del vuelo, se envían con el sistema ACARS información actualizada de la hora estimada de llegada (hora_ETA), según como se va desarrollando el vuelo siempre en base a lo planificado y las circunstancias que atrasen o adelanten un vuelo. Así, se logra planificar y estimar si algún vuelo solapase otro ya planificado en un puente de embarque.

La correlación es una métrica que busca relaciones lineales entre variables. La gran mayoría de variables no encuentran una relación dado que estos pueden no tener una relación lineal, no así una relación no lineal.

En el Grafico 3.1, se logra apreciar que el comportamiento de la llegada de aeronaves torna en una media de 16.25 minutos, 1 minuto más de lo establecido como “a tiempo” mientras que la media de salidas se encuentra en 10.91 minutos.

Estos distintos desempeños estadísticos marcan el entrenamiento de los modelos de Machine Learning, pudiendo afectar el desempeño a la hora de predecir vuelos de salida o de llegada.



Gráfica 3-1: histograma de On-time performace (Elaboración propia).

Los patrones de los vuelos de llegada como los vuelos de salida son en su esencia parecidos. Esto se refleja en la figura 3.1, donde se aprecia una distribución parecida, pero con medias distintas. En diversas publicaciones científicas se realiza una separación de modelos, aquellos encargados de la predicción de operaciones de salida y otra de llegada. Incluso, modelos separados según terminal. La separación de estos modelos es un punto de estudio, donde se pone a prueba los indicadores de desempeños frente a un modelo unificado como el de la presente memoria.

3.4 Selección de herramientas IA

A continuación, se presentan las seis técnicas de Machine Learning con su configuración de hiperparámetros y resultados de desempeño de cada algoritmo.

En el caso de la figura 3.2, se muestra el modelo de red neuronal multicapa. A elección del desarrollador del modelo, se eligió 10 neuronas de entrada, 2 capas ocultas de 10 neuronas cada una y finalmente una capa de salida de 1 neurona que arroja un valor de entrada para una función lineal, siendo la salida el valor de On Time Performance.

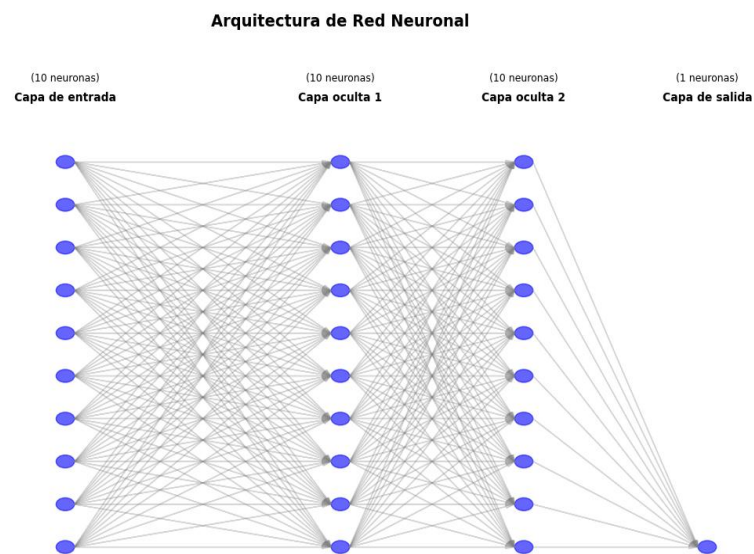


Figura 3-2: red neuronal multicapa construida en el modelo para predecir OTP (Elaboración propia).

Aparte de este modelo de red neuronal multicapa, se entrenó una red neuronal multicapa con función de pérdida de cuartiles con la siguiente configuración.

- Capa de entrada: 5 neuronas.
- Capas ocultas: 3 de 5 neuronas.
- Capa de salida: 1 neurona.

El algoritmo de *Random Forest* presenta un problema de sobreajuste, cuyo número de

profundidad debe ser controlada para evitar este fenómeno. La elección de hiperparámetros fue la siguiente:

- Número de estimadores: 500.
- Profundidad máxima: 20.
- Estado de aleatoriedad: 42.
- Criterio: Error Cuadrático.

Se presenta los hiperparámetros del algoritmo KNN.

- Número de vecinos: 2.

Se presenta el modelo *Long-Short Term Memory*.

- Neurona de entrada: 64, activación tanh.
- Neurona de capa oculta: 32, activación relu.
- Neuronas de salida: 1, activación lineal.
- Time Steps: 5.
- Learning Rate: 0.001.
- Función de costo: MSE.
- Épocas: 50.
- Batch size: 8.

Modelo *Ridge Regression* tiene un solo hiperparámetro.

- Alpha: 1.

Por último, los hiperparámetros del modelo SVR:

- Kernel: rbf.
- C: 1.

3.5 Entrenamiento

Los datos siguen el flujo de la figura 3.1, donde la evaluación del error del (Y_{test}) con (Y_{test}) es el punto clave de evaluación de los modelos.

Entre estos algoritmos, resulta el modelo de Random Forest aquella de mejor desempeño visual, infiriendo que no está sobreajustado dado que su gráfica tiene subestimación. Otros modelos de Machine Learning lograron buen rendimiento, siendo entre estos las redes neuronales multicapa con la función de pérdida de cuantiles, las redes neuronales multicapa (MLP) y KNN. Los modelos LSTM, Ridge Regression y SVR tuvieron comportamiento malos o casi nulos, donde sus gráficas no solo obtuvieron de manera visual un subajuste, sino que no logra capturar los patrones (véase anexo 4).

El Gráfico 3.2 presenta los resultados del On-Time Performace (OTP).

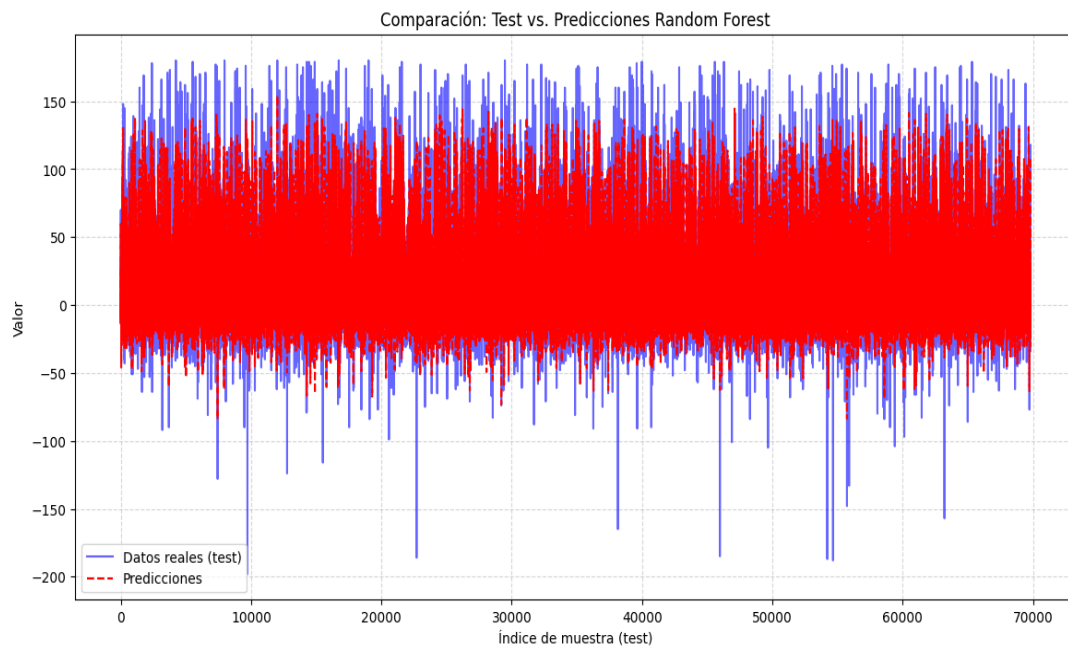
Los datos se dividen de la siguiente manera:

- La serie en azul representa los datos reales (Y_{test}), ósea los datos del conjunto conocido.
- La serie en rojo muestra los datos predichos (Y_{pred}) por el modelo de *Machine Learning*.

Visualmente, el Gráfico 3.2 permite identificar la similitud de las predicciones (Y_{pred}) en comparación con los datos de testeo (Y_{test}), sospechando la capacidad predictiva del modelo Random Forest. La complejidad de este modelo se ve reflejado en su estructura arborizada automáticamente seleccionada por el algoritmo (Véase anexo D.1). Cabe señalar, los algoritmos de Machine Learning subestiman los valores predichos como se visualiza en el Gráfico 3.2.

Para la evaluación cualitativa, se empleará las siguientes métricas vistas en el capítulo II.

- Error cuadrático medio (MSE)
- Error absoluto medio (MAE)
- Coeficiente de determinación (R^2)



Gráfica 3-2: Resultado de predicción de OTP para el modelo de Random Forest (Elaboración propia).

A continuación, en la Figura 3.3 se presenta los resultados de desempeño de los modelos de *Machine Learning*.

=== Comparación de Modelos ===

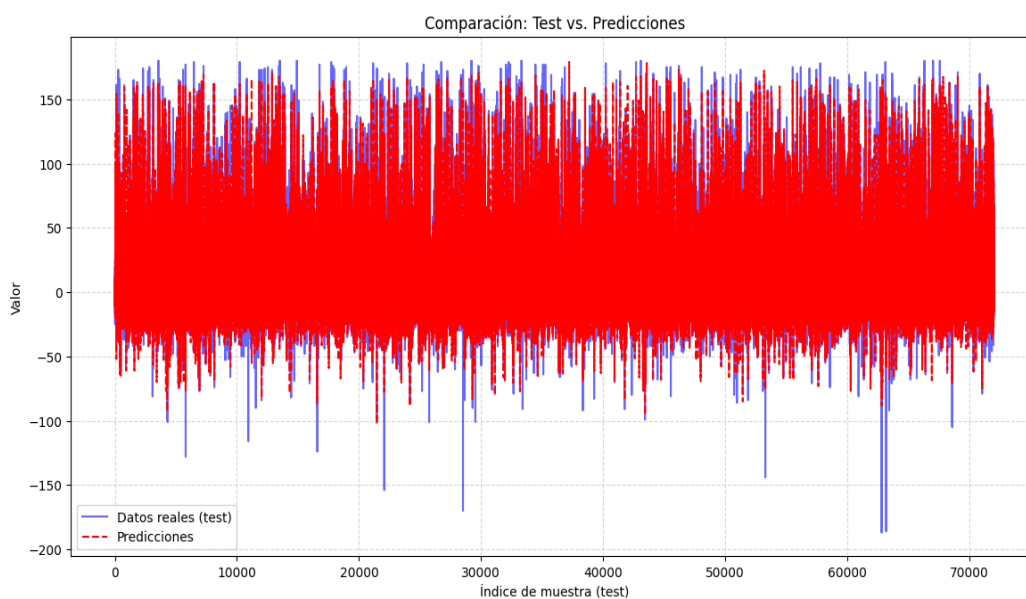
	Model	MSE	MAE	R ²
4	Random Forest	122.4871	6.7891	0.7746
0	MLP	135.0660	7.8505	0.7515
3	KNN	289.8390	11.2079	0.4667
5	SVR	544.9518	13.2241	-0.0026
2	Ridge Regression	511.3267	13.7669	0.0592
1	LSTM	549.1927	14.7028	-0.0008

Figura 3-3: Desempeño de los modelos Machine Learning (Elaboración propia).

Además, la Figura 3.4 se muestra el desempeño de las redes neuronales multicapa con la función de pérdida de cuantiles, siendo esta la que obtuvo mejor desempeño en MSE y MAE.

✓	Raíz del Error Cuadrático Medio (MSE): 103.8540
✓	Error Absoluto Medio (MAE): 5.5872
✓	Coefficiente de Determinación (R ²): 0.8094

Figura 3-4: Resultado de las redes neuronales multicapa con la función de pérdida de cuantiles.



Gráfica 3-3: Resultados de predicción de OTP para el modelo red neuronal multicapa con la función de pérdida de cuantiles (Elaboración propia).

En la Figura 3.3 y 3,4 donde aparecen desempeños de los modelos, se destaca el R^2 de 0.77 de Random Forest, siendo capaz de explicar la variable On-Time Performance de los vuelos. Un MSE de 122.4 y un MAE de 6.78 minutos. Estos desempeños son concluyentes de un buen resultado con baja probabilidad de sobreajuste. Sin embargo, debido a la sensibilidad del dato y en búsqueda del error absoluto medio más bajo, se evidencia un mejor desempeño para las redes neuronales multicapa con la función de pérdida de cuantiles en la figura 3.3, con un claro seguimiento de las tendencias y patrones de la variable a predecir con el riesgo de posible sobreajuste en el entrenamiento. Esto se podrá validar o refutar con la capacidad predictiva de la prueba de validación en la sección 3.8.

En el paper de Zhang et al. (2025) titulado "Predictive and prescriptive analytics for robust airport gate assignment planning" se obtuvo un resultado de MSE y MAE de 882.65 y 14.81 respectivamente. Si bien el MAE obtenido en esta memoria es menor al obtenido en el paper, esto se puede deber a un leve sobreajuste en el modelo tal como ya se mencionó.

Las redes neuronales multicapa (MLP) obtuvo un buen desempeño, situándose en segundo lugar. Con un R^2 de 0.75, siendo capaz de explicar la variable On-Time Performance, con resultados contundentes para aquellos vuelos con adelanto. Además, EL MSE y MAE no suben drásticamente, situándose a unos 135,0 y 7,8 tal como se ve en la figura 3.3. El desempeño de este puede deberse a una buena elección de hiperparámetros, siendo la configuración de nodos 10x10x10x1 cercano a la más óptima.

SVR, LSTM y Ridge Regression no logran capturar los patrones del itinerario. El error MAE promedio de los 4 modelos es de 13.3 minutos y en la prueba visual reflejan tendencias nulas (Véase anexo 3), reflejando una alta variación entre valores reales y el predichos.

Finalmente, se decide usar las redes neuronales multicapa con la función de pérdida de cuantiles como herramienta de predicción.

3.6 Resultados

Los datos de testeo de las predicciones de OTP del modelo de red neuronal multicapa con la función de pérdida de cuantiles, son aproximadamente 27.000 datos como ya se mencionó, siendo datos de la semana 13 de 2023 hasta la semana 13 de 2025. Se somete a una prueba de comportamiento, donde se evaluará la importancia de las variables de los dos mejores modelos y a que distribución de densidad mejor se asemeja los datos predichos. Luego, se tomará para el mes de mayo del 2025 para realizar una prueba de validación en contextos reales con datos jamás visto por el modelo de inteligencia artificial, donde en base al itinerario de mayo, se realizan las predicciones del OTP de los vuelos.

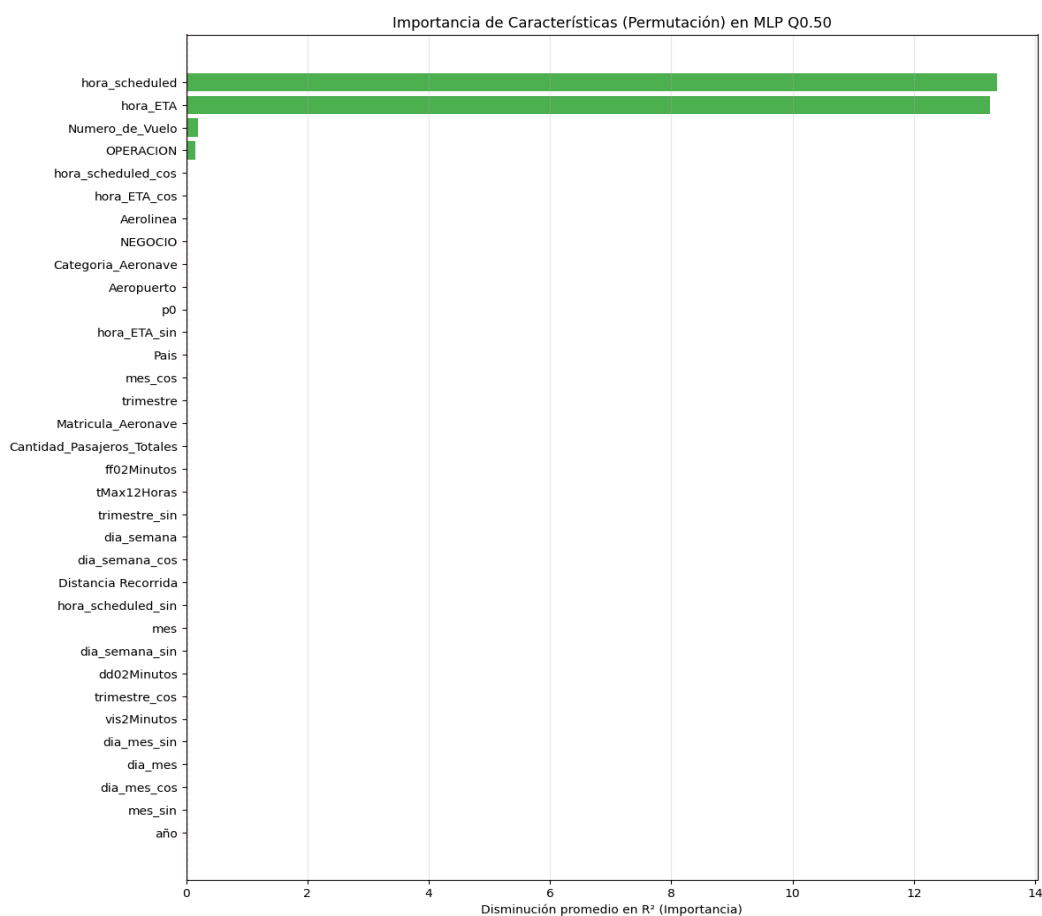
El modelo de redes neuronales multicapa con la función de pérdida de cuantiles tiene la opción de solicitar el peso de la conexión de los nodos (véase figura 3.2). La suma de los pesos de la primera capa de entrada muestra la importancia que tiene cada variable para el modelo, siendo aquel más importante las variables temporales de tipo operacional como hora programada y hora estimada (Véase gráfica 3.4). Cabe señalar que los modelos de red neuronal estiman las variables con mayores escalares como las más importantes. Estas dos variables no fueron escaladas y esto fue determinado en tres pruebas de rendimiento:

- Ninguna variable escalada.
- Todas las variables escaladas.
- Todas las variables escalas + scheduled y ETA no escaladas.

Los resultados fueron determinantes. La configuración “Todas las variables escalas + scheduled y ETA no escaladas” resulto tener mejor desempeño, incluso cuando estas

variables agregadas no se normalizaron.

Según la figura 3.4, donde se ven la importancia de estas variables en la construcción del modelo de ML. El modelo se explica entr dos variables; hora_scheduled y hora_ETA, capturando el comportamiento del On Time Performance. Esto da la posibilidad de eliminación de otras variables, buscando la combinación de variables que mejore el desempeño para trabajos futuros.



Gráfica 3-4: Importancia de las variables para el modelo de red neuronal multicapa con la función de pérdida de cuantiles (Elaboración propia).

Random Forest también se rescata un grado de importancia de cada variable en la explicación del modelo de predicción, siendo porcentajes y unidades adimensionales.

El gráfico 3.5, el modelo de Random Forest presenta la importancia de los datos en las

decisiones del árbol. En Random Forest, una variable es más importante si, al usarse para dividir nodos, produce grandes reducciones en la impureza, ósea, reducción en el error MSE. Esto se explicó en sección 2.2.11 Random Forest.

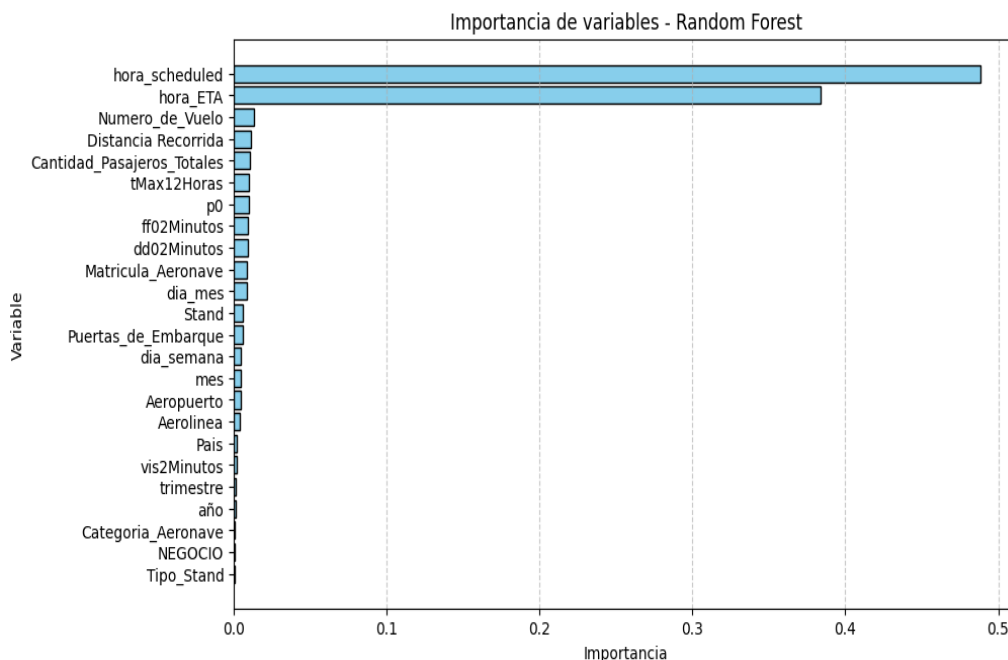
Las variables hora_scheduled y hora_ETA (Estimated Time of Arrival) emergen como los predictores de mayor importancia, logrando la reducción en el error cuadrático medio (MSE).

El modelo ha identificado que estas dos variables, de manera conjunta, explican casi el 90% del On Time Performance (OTP). La alta relevancia de la variable Estimated Time of Arrival (hora_ETA) resulta fundamentalmente razonable. Este dato es transmitido por el sistema Aircraft Communications Addressing and Reporting System (ACARS) de la aeronave en tiempo real para poder realizar predicciones de llegada gestionando la asignación de puente de embarque. Ósea, tratar de predecir el OTP. Por su naturaleza predictiva y en tiempo real, su peso en la determinación de la variable objetivo es esperado.

La hora programada (hora_Scheduled) por la aerolínea es reconocida como la variable más importante. Este hallazgo está en concordancia con las estrategias operacionales de las aerolíneas. El objetivo es maximizar el factor de utilización de la aeronave, cubrir una frecuencia de vuelo parecida a la competencia y priorizar horarios de noche para vuelos internacionales y horarios tempranos para vuelos nacionales.

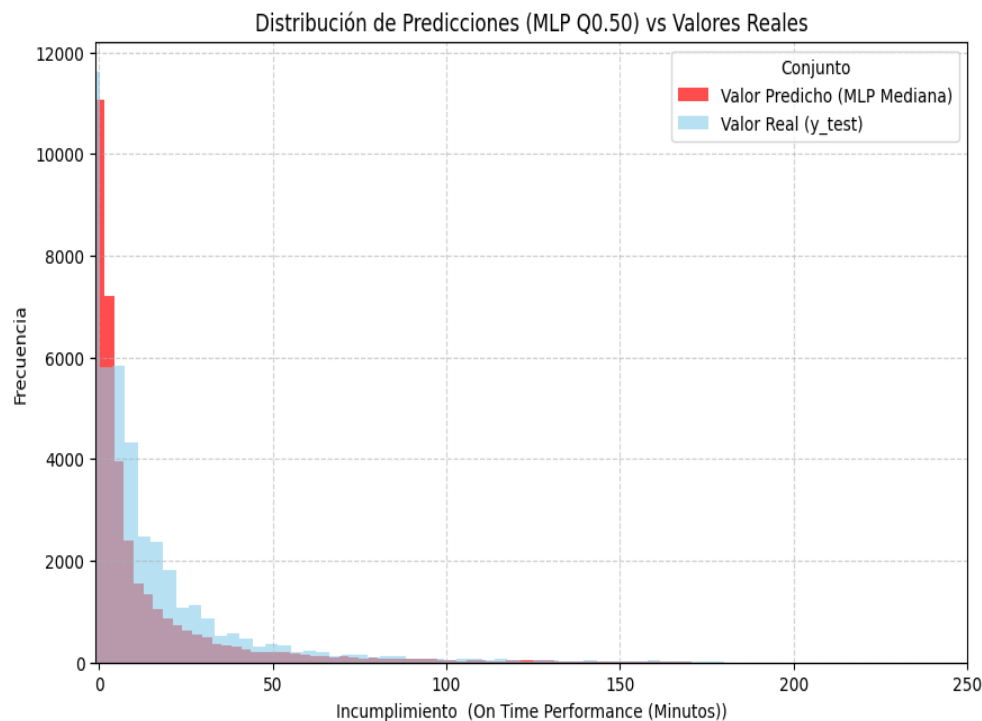
Además, se menciona que el modelo de Random Forest no requiere de ningún ajuste de ciclicidad ni de normalización. Este modelo no le entrega más importancia a la escala de la variable. Por lo que, en este modelo se puede concluir que las variables hora_scheduled y hora_ETA reflejan la explicación del 90% del OTP al igual que el modelo de MLP con función de pérdida de cuantil.

Las demás variables realizar poco aporte al modelo, pero no es más del 10% del modelo completo.



Gráfica 3-5: Importancia de las variables para el modelo de Random Forest (Elaboración propia).

El desempeño visto por el modelo de red neuronal multicapa con la función de pérdida de cuantiles en términos de indicadores de error son prometedores. El desempeño visual de la gráfica 3.6 muestra un histograma de los valores de testeo (Y_{test}) y los valores predichos (Y_{pred}). Se ve que entre los valores 0 y 10 minutos el modelo sobrestima los valores predichos, llegando a superar las 6000 observaciones predichas entre esos valores. Entre los valores 10 y 50, si bien la tendencia de la distribución se acerca a los valores reales. Esto indica que el modelo subestima las predicciones, estando por debajo de los valores celestes.



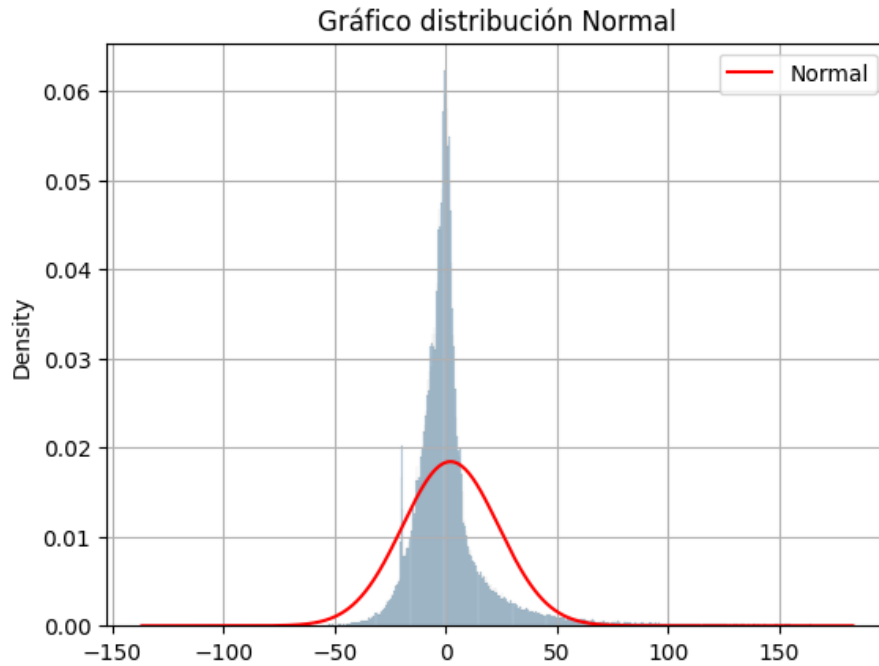
Gráfica 3-6: Distribución de OTP predicho vs testeo por Red Neuronal Multicapa con función de pérdida de cuartiles (Elaboración propia).

3.6.1 Distribución Normal

En las próximas secciones se analiza el comportamiento de los datos de la predicción, buscando alguna distribución que se asemeje a los datos de On Time Performance.

La densidad del OTP del gráfico 3.7 se comporta con un leve sesgo a la izquierda. Los parámetros de la distribución son:

- Media: 2.07.
- Desviación Estándar: 21.6.



Gráfica 3-7: Densidad de OTP respecto a una distribución Normal (Elaboración propia).

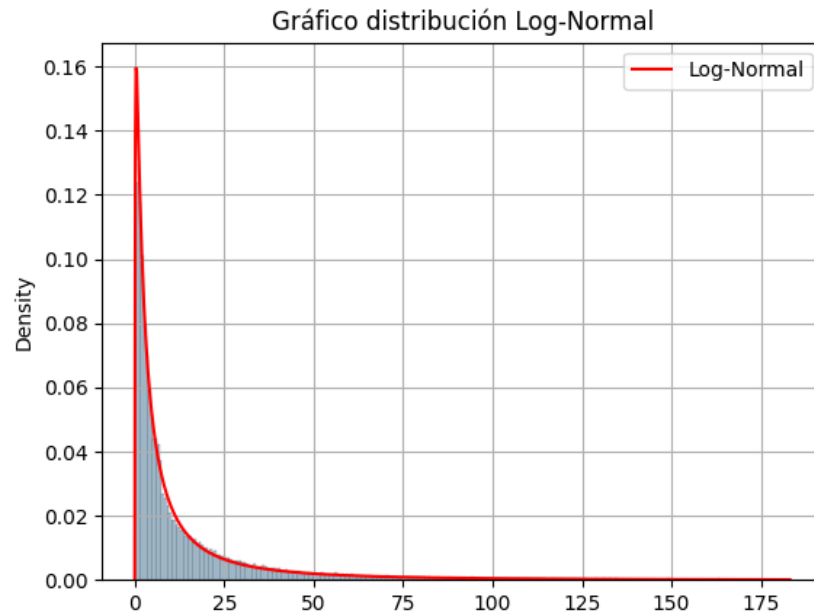
Su función según la ecuación (38) se ve expresada como:

$$f(x; \mu, \sigma) = \frac{1}{21,6\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-2,07}{21,6}\right)^2}$$

3.6.2 Distribución Log-Normal

La función de distribución Logarítmica-Normal del gráfico 3.8 presenta un comportamiento parecido a la densidad del OTP, siendo capaz de capturar los picos de frecuencia cerca del 0. Se debe aclarar que el comportamiento de una función log-normal debe ser mayor a 0 estricto. Por lo que, se realizó el corte de datos mayor a 0 hasta 180 minutos para la construcción del gráfico.

Esta distribución podría explicar el OTP, pero solo para los valores de atraso positivos.



Gráfica 3-8: Densidad de OTP respecto a una distribución Log-Normal (Elaboración propia).

Los parámetros de la distribución son:

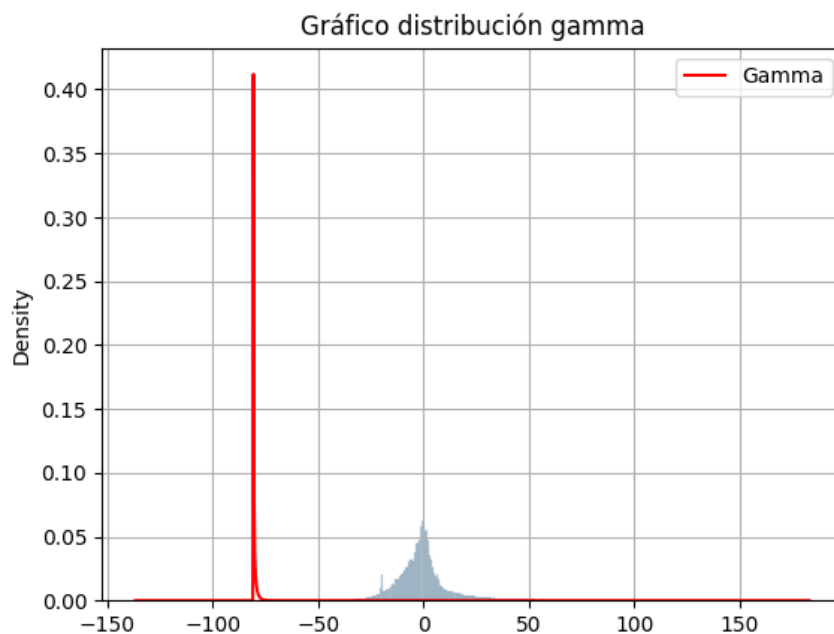
- Media: 5.59.
- Desviación Estándar: 1.59.

Su función según la ecuación (39) se ve expresada como:

$$(x; \mu, \sigma) = \frac{1}{1,59\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{\ln x - 5,59}{1,59}\right)^2}$$

3.6.3 Distribución Gamma

La distribución Gamma del gráfico 3.9 no logra explicar el OTP, llegando a situarse en los valores negativos. La distribución gamma también requiere que los valores sean positivos. Por lo que, se usó un desplazamiento de -80 para poder ser usados.



Gráfica 3-9: Densidad de OTP respecto a una distribución Gamma (Elaboración propia).

Los parámetros de la distribución son:

- Alpha: 0.03.
- Beta: 1.89.
- Desplazamiento: -80

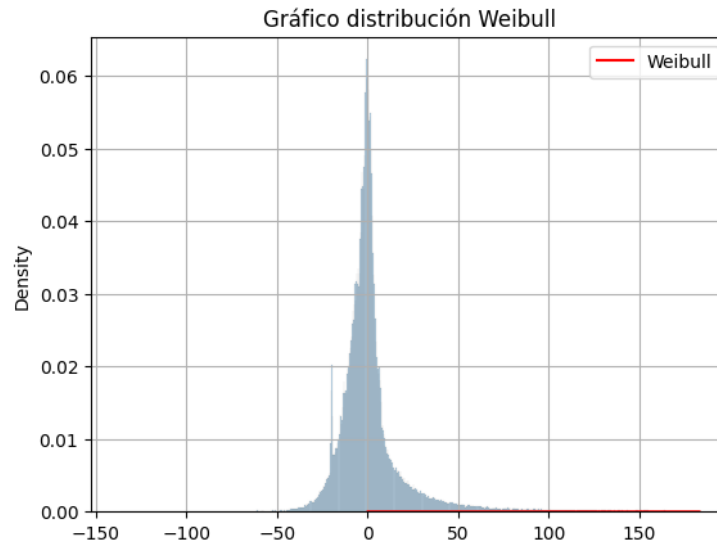
Su función según la ecuación (40) se ve expresada como:

$$f(x; \alpha, \theta) = \begin{cases} \frac{1}{\Gamma(0,03)1,89^{0,03}} \cdot (x - (-80))^{0,03-1} \cdot e^{\frac{x-(-80)}{1,89}} & x > 0, \quad \alpha, \theta > 0 \\ 0 & \text{Para cualquier otro valor} \end{cases}$$

3.6.4 Distribución Weibull

La distribución Weibull del gráfico 3.10, generalmente usada para análisis de confiabilidad, análisis de vida en el área de mecánica de la ingeniería, se aleja del uso que se está buscando. Se descarta su utilización, su gráfica no logra tomar los valores

de OTP.



Gráfica 3-10: Densidad de OTP respecto a una distribución Weibull (Elaboración propia).

Los parámetros de la distribución son:

- Alpha: 0.07.
- Beta: $4.31 \cdot 10^{-16}$

Su función según la ecuación (41) se ve expresada como:

$$f(x; \alpha, \theta) = \begin{cases} \frac{0,07}{(4.31 \cdot 10^{-16})^{0,07}} \cdot x^{0,07-1} \cdot e^{-\left(\frac{x}{4.31 \cdot 10^{-16}}\right)^{0,07}} & x > 0, \quad \alpha, \theta > 0 \\ 0 & \text{Para cualquier otro valor} \end{cases}$$

3.6.5 Comparación

La tabla 3.8 muestra parámetros de cada una de las distribuciones, siendo estas obtenidas según los ajustes a los datos de entrenamiento.

Distribución	Parámetro 1	Valor 1	Parámetro 2	Valor 2
Normal	Media	2.07	Desviación Estándar	21.6
Log-Normal	Media	5.59	Desviación Estándar	1.59
Gamma	Alpha	0.03	Beta	1.89
Weibull	Alpha	0.07	Beta	$4.31 \cdot 10^{-16}$

Tabla 3-8: Comparación de parámetros entre distribuciones (Elaboración propia).

Se identificó que la distribución Log-Normal solo logra explicar el comportamiento del *On Time Performance* predichos para el conjunto de valores positivos. Sin embargo, la necesidad de buscar una función que modele tanto los atrasos (valores positivos) como los adelantos (valores negativos) en los vuelos obliga a utilizar una distribución con un dominio irrestricto. Por lo tanto, la distribución normal se selecciona como la distribución más eficaz en capturar el fenómeno de adelanto o atraso operacional.

3.7 Validación del modelo

Se procede a poner a prueba el modelo de red neuronal multicapa con la función de perdida de cuantiles para predecir los *On Time Performance* del mes de mayo de 2025 para las operaciones del aeropuerto. Esto se realiza en búsqueda de la validación del modelo en un caso aplicado.

El itinerario del mes de mayo presenta tantos vuelos de arribo y salidas, siendo 11.640 vuelos programado. En la tabla 3.9 se presenta un extracto del itinerario de 17 vuelos, apreciando los datos oficiales reales de *On Time Performance* en la columna incumplimiento (OTP) y la predicción de la inteligencia artificial en la columna MLP_Media_Predicha_min, variando en su desempeño, siendo algunos datos más cercanos que otros.

Scheduled	OPERACION	Aerolinea	Pais	Aeropuerto	Incumplimiento (OTP)	MLP_Media_Predicha_min	Diferencia Porcentual	residual
2025-04-30 07:24:00	Llegada	Aerolinea 11	Spain	Madrid	24	12,3	-95,7%	11,7
2025-04-30 22:10:00	Llegada	Aerolinea 12	Brazil	Rio De Janeiro, GIG	336	-6,2	5532,6%	342,2
2025-04-30 23:50:00	Salida	Aerolinea 23	Ecuador	Guayaquil	-3	-17,8	83,2%	14,8
2025-04-30 23:55:00	Llegada	Aerolinea 16	Chile	Antofagasta	38	42,2	9,9%	-4,2
2025-04-30 23:55:00	Salida	Aerolinea 5	Colombia	Bogota	-4	0,6	726,7%	-4,6
2025-05-01 00:13:00	Salida	Aerolinea 7	Panama	Ciudad de Panama	-3	-3,3	9,3%	0,3
2025-05-01 00:15:00	Llegada	Aerolinea 18	Chile	Puerto Montt	0	-5,1	100,0%	5,1
2025-05-01 00:24:00	Salida	Aerolinea 7	Panama	Ciudad de Panama	-2	-3,9	49,2%	1,9
2025-05-01 00:26:00	Llegada	Aerolinea 9	Chile	Calama	-3	-4,2	28,3%	1,2
2025-05-01 00:30:00	Salida	Aerolinea 16	Colombia	Bogota	-8	-7,3	-9,4%	-0,7
2025-05-01 00:34:00	Llegada	Aerolinea 18	Chile	Calama	-1	2,1	146,8%	-3,1
2025-05-01 00:40:00	Salida	Aerolinea 10	Argentina	Buenos Aires, EZE	8	-2,6	408,0%	10,6
2025-05-01 00:40:00	Salida	Aerolinea 16	Brazil	Sao Paulo, GRU	-3	-3,2	7,4%	0,2
2025-05-01 00:45:00	Llegada	Aerolinea 16	Brazil	Sao Paulo, GRU	22	24,3	9,5%	-2,3
2025-05-01 01:24:00	Llegada	Aerolinea 9	Chile	Concepcion	-6	3,3	283,0%	-9,3
2025-05-01	Llegada	Aerolinea 16	Chile	Iquique	-2	-1,1	-88,5%	-0,9

Tabla 3-9: Extracto de vuelos del mes de Mayo (Elaboración propia).

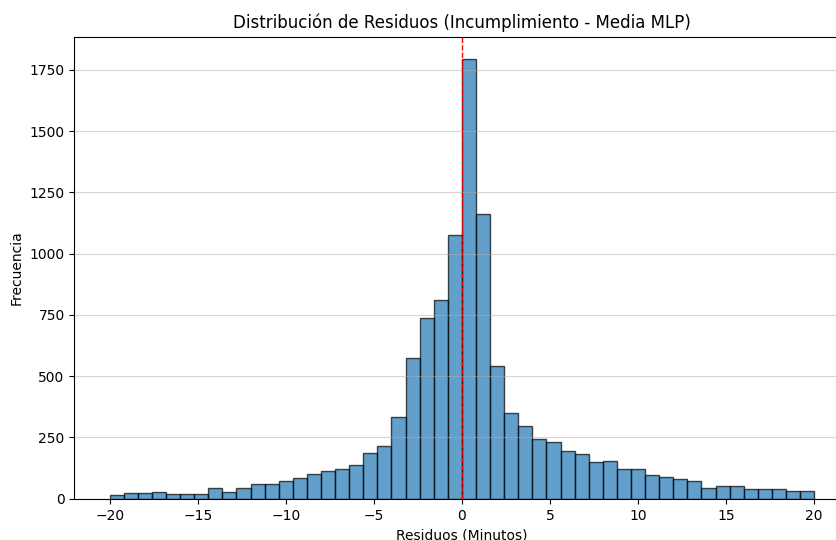
Se observa un desempeño variado en la diferencia porcentual de cada vuelo unitario. Si bien, cada vuelo puede llegar a diferir en gran medida, numéricamente las diferencias no son tan extremas. Por ejemplo, el vuelo a Iquique presenta un OTP de -2 minutos, el modelo predice -1.1 minutos con una diferencia porcentual de -88.5%. En términos operacionales una diferencia de 1 minutos de operación no impacta en la planificación como si lo fuese el caso del vuelo a Brasil con una OTP de 336 minutos, prediciendo -6.2 minutos. Sin embargo, es esperado este resultado debido a las limitaciones del modelo entre vuelos de -200 y 180 minutos.

Varios vuelos presentan cercanías a la hora de la predicción. Los primeros 7 vuelos de la tabla 3.9 presentan grandes similitudes en el OTP omitiendo el vuelo de Brasil, siendo la cercanía incluso llegando a 1 minuto de variación.

El algoritmo logra realizar predicciones para vuelos adelantados como los atrasados, demostrando su efectividad. A continuación, se realiza una prueba de desempeño con el indicador MAE, en busca de conocer el error absoluto de las predicciones del mes de mayo.

El mes de mayo presenta un error absoluto promedio de 6.24 minutos por vuelo, dejando en claro la efectividad del modelo de red neuronal multicapa con función de pérdida de cuartiles.

La distribución normal de los residuales es un punto clave del diagnóstico de un modelo de Machine Learning. Los residuos describen la brecha entre los valores de resultados observados y previstos. Este determinará si el modelo cuenta con errores sistemáticos o solo con errores aleatorios. El gráfico 3.11 muestra la distribución de los valores $y_{real} - y_{pred}$, representando el valor residual. Notar que la distribución esta centrada casi en el 0 (0 se denota con una línea roja discontinua).



Gráfica 3-11: histograma de los datos residuales (Elaboración propia).

La media de los residuales es de 2.20 minutos. El modelo tiene un ligero sesgo positivo, subestimando el valor real de incumplimiento (OTP) por 2.20 minutos. Si bien este sesgo es pequeño en términos absolutos, idealmente la media de los residuales debe ser lo más cercano a 0 para confirmar que el modelo no tiene un sesgo sistemático.

La desviación estándar de los residuales es de alta imprecisión, con una desviación de 30.88 minutos. Esto indica que si bien el error promedio es de 2.20 minutos, los errores individuales son ampliamente inconsistentes.

Esto refleja una baja precisión. Una gran dispersión sugiere que para algunos datos el error es bajo, pero para otros el error puede ser de ± 30 minutos.

Conclusiones

El modelo de red neuronal multicapa con función de pérdida de cuartiles obtuvo un desempeño contundente, siendo capaz de realizar predicciones de los vuelos del mes de mayo. El desempeño del modelo obtenido en términos absolutos fue de 5,8 minutos por vuelo y un R^2 fue de 0.809 para el proceso de entrenamiento y para la validación de los datos obtuvo un error absoluto de 6,24 minutos.

En la validación del modelo, los residuales obtuvieron una media de 2.20 minutos, reflejando un ligero sesgo positivo. Esto confirma que el modelo no tiene errores sistemáticos. Sin embargo, la desviación estándar se sitúa en ± 30 minutos, lo que sugiere una alta imprecisión o que los errores individuales son ampliamente inconsistentes.

Se logró realizar la aplicación real del marco teórico, desarrollando y ajustando algoritmos de Machine Learning a problemas reales del sector aeronáutico, identificando problemáticas relacionadas a la extracción, transformación y carga de los datos. El refuerzo de conocimientos matemáticos adquiridos en el desarrollo de la carrera fue adquirido para el entendimiento del modelo y las razones de fallas.

Con los resultados contundentes, se identificó el algoritmo de Machine Learning que mejor se desempeñó y que en futuras aplicaciones podrá ser utilizados para el ahorro de tiempo en la investigación de herramientas predictivas.

En base a los objetivos planteados en la sección 1.3 y 1.4, se logró predecir el On Time Performance del aeropuerto Arturo Merino Benítez mediante modelos de Machine Learning (Véase la sección 3.7), donde se integró datos operacionales y climatológicos (Véase la sección 3.2.2). Además, se logró realizar el estudio de seis herramientas de Machine Learning, se definió la variable a predecir y en la sección 3.5 se comparó los

errores de los distintos algoritmos.

Los modelos de red neuronal multicapa logran capturar complejos patrones debidos a sus amplias configuraciones lineales, posibles con sus conexiones entre los nodos definidos y su capacidad de aprendizaje de sus errores pasados con la minimización de la función de costo. Sin embargo, el problema de estas herramientas es las altas exigencias computacionales, siendo dependiente de la capacidad del hardware del dispositivo de entrenamiento.

Para trabajos posteriores, el análisis del impacto de las variables de la figura 3.3 en la explicación del modelo debe ser estudiado. El grado de impacto que un modelo sobreajustado puede hacer que los indicadores de desempeño sean engañosos, confundiendo los resultados con “buen desempeño”. Para esto se puede realizar una sensibilización del modelo, entrenando los distintos modelos con diferentes combinaciones de variables, buscando aquella que mejor desempeño arroja.

En la presente memoria se configuró los hiperparámetros de las redes neuronales multicapa con función de perdida de cuartiles de manera aleatoria. Esta práctica debe ser mejorada mediante la utilización de algoritmos de búsqueda de arquitecturas de red neuronal más eficientes en términos de desempeño y capacidad computacional, en búsqueda de la reducción del tiempo de entrenamiento y mejor desempeño de MSE, MAE y R^2 .

Los patrones de los vuelos de llegada como los vuelos de salida son en su esencia parecidos. Esto se refleja en la figura 3.1, donde se aprecia una distribución parecida, pero con medias distintas. En diversas publicaciones científicas se realiza una separación de modelos, aquellos encargados de la predicción de operaciones de salida y otra de llegada. Incluso, modelos separados según terminal. Para futuros trabajos, la separación de estos modelos es un punto de estudio, donde se pone a prueba los indicadores de desempeños frente a un modelo unificado como el de la presente memoria.

La fiabilidad de los datos de incumplimiento (On Time Performance) es un punto clave dentro del estudio. Los agentes del centro de control de operaciones (CCO) son los encargados de asignar las puertas de embarques a los vuelos de llegada o salida, siendo informados por la torre de control. Errores en las marcas de tiempos de llegada y salida real de la aeronave afecta a la fiabilidad de los datos y por tanto al entrenamiento del modelo, reduciendo su desempeño en las pruebas de desempeño y validación. Se requiere en futuros trabajos realizar validación de los datos en conjunto con la oficina de TI para asegurar la calidad de los datos.

Bibliografía

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5), 359–366.

Pebrianti, D., Khilani, Z., Rusdah., Bayuaji, L., (2024). Predictive maintenance in aerospace industry using convolutional neural network.

Géron, A., (2019). *Hands-on Machine Learning with Scikit – Learn, Keras & Tensorflow*.

Rochester, N., (1983). "El Proyecto 701 visto por Su Arquitecto Jefe," , p. 202-204.

Widrow, B., Hoff, M., (1960) “Adaptive Switching Circuit”.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, pág 533–536. **(está bien referenciado seguir este tema)**

Priyanka, G. (2018). Prediction of airline delays using K nearest neighbor algorithm. *International Journal of Emerging Technology and Innovative Engineering*, 4(5), 87.

Li, Q., & Jing, R. (2022). Flight delay prediction from spatial and temporal perspective. *Expert Systems With Applications*, 205, 117662

International Air Transport Association. (2025). Air passenger market analysis: December 2024. <https://www.iata.org/monthly-traffic-statistics>

International Air Transport Association. (2024). La demanda global del transporte aéreo de pasajeros continuó recuperándose en 2023 (Comunicado N.º 06). <https://www.iata.org>

International Air Transport Association. (2025). La demanda mundial de pasajeros aéreos alcanza su máximo histórico en 2024 (Comunicado N.º 05). <https://www.iata.org>

Airbus. (2024). Global Market Forecast 2024-2043 [PDF]. <https://www.airbus.com>

Ministerio dos Transportes, Portos e Aviação Civil. (2016). *Transportes 2016: Versão final web* [PDF]. Governo de Brasil. <https://www.gov.br/infraestrutura/pt-br/assuntos/transportes>

De Arce, R., Mahía, R. (2001). Modelos Arima. U.D.I Econometría e informática



<https://gc.scalahed.com/recursos/files/r161r/w24503w/Box-Jenkins.PDF>

Eurocontrol. (2024). *Daily punctuality* –
Airports. <https://www.eurocontrol.int/Economics/DailyPunctuality-Airports.html>

Universidad de Sevilla. (2012). *Memoria por capítulos: Capítulo 4. Red Neuronal Artificial. El Perceptrón*. Proyecto de investigación.

Pontificia Universidad Católica de Valparaíso. (2010). *Redes neuronales artificiales: El modelo del perceptrón* [PDF]. Recuperado de http://opac.pucv.cl/pucv_txt/txt-8500/UCC8513_01.pdf

Pérez, J. (2011). *Cálculo en una variable: Capítulo 2* [PDF]. Universidad Autónoma de Madrid. Recuperado de https://matematicas.uam.es/~fernando.chamizo/asignaturas/calclinf1011/apjperez/calculo_cap02.pdf

Stewart, J. (2012). *Cálculo de varias variables transcendentales tempranas: Capítulo x* [PDF]. Cengage Learning, pág 993-994.

IBM. (2023). *Función de pérdida (loss function): Qué es y cómo se usa en machine learning*. IBM Think. Recuperado de <https://www.ibm.com/es-es/think/topics/loss-function>

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536. <https://doi.org/10.1038/323533a0>

Abdulkadirov, R. I., & Lyakhov, P. A. (2023). Un nuevo enfoque para el entrenamiento de redes neuronales utilizando el descenso de gradiente natural con momento basado en distribuciones de Dirichlet. *Óptica Computacional*, *47*(1), 160–169. <https://doi.org/10.18287/2412-6179-CO-1147>

Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory*. *Neural Computation*, 9(8), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>

International Air Transport Association [IATA]. (2025, June 2). *Standard Practices for Aircraft Ground Handling* (Report No. SP-2025-06-02-01). Recuperado de <https://www.iata.org/contentassets/c6c0d5cd9030412b85da743c94f45f98/2025-06-02-01-sp.pdf>

Organización de Aviación Civil Internacional [OACI]. (2020). *Anexo 14 al Convenio sobre Aviación Civil Internacional: Aeródromos, Volumen I – Diseño y*



operaciones de aeródromos (8.^a ed.).
Recuperado de <https://www.anac.gov.ar/anac/web/uploads/normativa/anexos-oaci/anexo-14-vol-i.pdf>

International Civil Aviation Organization [ICAO]. (2021). *Glossary of Aviation and Public Health Terms*.
Recuperado de <https://www.icao.int/safety/CAPSCA/PublishingImages/Pages/ICAO-References/Glossary%20of%20Aviation%20and%20Public%20Health%20Terms.pdf>

Dirección General de Aeronáutica Civil [DGAC]. (2009). DAN 121 Requisito de operación: operaciones nacionales, regulares y no regulares, pág. 99-102.

Dirección General de Aeronáutica Civil [DGAC]. (2024). DAN 43 *Norma Aeronáutica "Mantenimiento"*.

Barnhart, C., Belobaba, P., Odoni, A. (2003). Application of Operations Research in the air transport industry, pag 369

Thiruthuvanathan, M., Prathap, B., Kumar, K., Murthy, H., Pillai, V. (2025). Forecasting Flight Delays with a multilayered memory fusion network, pag 1302-1315

International Air Transport Association [IATA]. (s.f.). *Airport Handling Manual (AHM 730 & 731)*. IATA.

Beltman, M., Ribeiro, M., de Wilde, J., & Sun, J. (2025). *Dynamically forecasting airline departure delay probability distributions for individual flights using supervised learning*. Journal of Air Transport Management, *126*, 102788.

Canavos, G. C. (1988). *Probabilidad y estadística: aplicaciones y métodos* (1a ed.). McGraw-Hill. (Trabajo original pp. 130-180).

Microsoft. (s.f.). *Guía de datos relacionales: ETL (Extracción, transformación y carga).* Azure Architecture Center. <https://learn.microsoft.com/es-es/azure/architecture/data-guide/relational-data/etl>

Al-Ghzawi, M., & El-Rayes, K. (2024). *Machine learning for predicting the impact of construction activities on air traffic operations during airport expansion projects.*

Dirección General de Aeronáutica Civil. (2022). *Gestión de afluencia del tránsito aéreo (DAP 11 136, Edición 1)*. DGAC.



Mahajan, T., Singh, G., & Bruns, G. (2021, 3 de marzo). *An experimental assessment of treatments for cyclical data* [Sesión de conferencia]. 2021 Computer Science Conference for CSU Undergraduates, Virtual.
<https://scholarworks.calstate.edu/downloads/pv63g5147>

Ministerio de Obras Públicas, Dirección General de Concesiones. (2024). *Informe Final T1-2024*. Gobierno de Chile.
<https://concesiones.mop.gob.cl/uploads/sites/4/2024/12/INFORME-FINAL-T12024.pdf>

Anexos

Anexo A

IATA Delay Code

Código	Descripción
OTHERS	
00-05	Códigos internos de la aerolínea
06 (OA)	No hay puerta/stand disponible por actividad de la propia aerolínea
09 (SG)	Tiempo en tierra programado menor al mínimo declarado

Tabla A.1: Codigos IATA para atrasos (Tabla 2.3: códigos IATA *Delay* (ANS Performance. (s.f.). *IATA Delay Codes*. ANS Performance. Recuperado de <https://ansperformance.eu/definicion/iata-delay-codes/>).

Passenger and Baggage	
11 (PD)	Check-in tardío, aceptación después del plazo
12 (PL)	Check-in tardío, congestión en área de check-in
13 (PE)	Error en check-in (pasajeros y equipaje)
14 (PO)	Sobre vendido, errores de reserva
15 (PH)	Embarque: discrepancias, paginación o pasajero check-in perdido
16 (PS)	Publicidad comercial/conveniencia del pasajero (VIP, prensa, comidas, objetos perdidos)
17 (PC)	Catering: pedido tardío o incorrecto
18 (PB)	Procesamiento de equipaje (clasificación, etc.)
19 (PW)	Pasajeros con movilidad reducida (embarque/desembarque)

Tabla A.2: Codigos IATA para atrasos (Tabla 2.3: códigos IATA *Delay* (ANS Performance. (s.f.). *IATA Delay Codes*. ANS



Performance. Recuperado de <https://ansperformance.eu/definition/iata-delay-codes/>).

Cargo and Mail	
21 (CD)	Documentación de carga: errores
22 (CP)	Posicionamiento tardío de carga
23 (CO)	Aceptación tardía de carga
24 (CI)	Embalaje inadecuado
25 (CO)	Sobrevendido (carga)
26 (CI)	Preparación tardía en almacén
27 (CE)	Documentación/embalaje (solo correo)
28 (CL)	Posicionamiento tardío (solo correo)
29 (CA)	Aceptación tardía (solo correo)

Tabla A.3: Codigos IATA para atrasos (Tabla 2.3: códigos IATA *Delay* (ANS Performance. (s.f.). *IATA Delay Codes*. ANS Performance. Recuperado de <https://ansperformance.eu/definition/iata-delay-codes/>).

Aircraft and Ramp Handling	
31 (GD)	Documentación de aeronave tardía/incorrecta (peso, balance, manifiesto, etc.)
32 (GL)	Carga/descarga: carga especial, falta de personal
33 (GE)	Equipo de carga: falta o falla (ej. cargador de contenedores)
34 (GS)	Equipo de servicio: falta o falla (ej. escaleras)
35 (GC)	Limpieza de aeronave
36 (GT)	Repostaje/desfuele: proveedor de combustible
37 (GB)	Catering: entrega o carga tardía
38 (GU)	ULD: falta o falta de servicio
39 (GT)	Equipo técnico: falta o falla (ej. remolque)

Tabla A.4: Codigos IATA para atrasos (Tabla 2.3: códigos IATA *Delay* (ANS Performance. (s.f.). *IATA Delay Codes*. ANS Performance. Recuperado de <https://ansperformance.eu/definition/iata-delay-codes/>).

Technical and Aircraft Equipment	
41 (TD)	Defectos en la aeronave
42 (TM)	Mantenimiento programado: liberación tardía
43 (TN)	Mantenimiento no programado (revisiones adicionales)
44 (TS)	Repuestos/equipo de mantenimiento: falta o falla
45 (TA)	Repuestos AOG (para enviar a otra estación)
46 (TC)	Cambio de aeronave por razones técnicas
47 (TL)	Falta de aeronave de reserva planificada
48 (TV)	Ajustes programados de configuración/versión de cabina

Tabla A.5: Codigos IATA para atrasos (Tabla 2.3: códigos IATA *Delay* (ANS Performance. (s.f.). *IATA Delay Codes*. ANS Performance. Recuperado de <https://ansperformance.eu/definition/iata-delay-codes/>).

Damage to Aircraft & EDP/Automated Equipment Failure	
51 (DF)	Daño durante operaciones de vuelo (aves, rayos, turbulencia)
52 (DG)	Daño en tierra (colisiones, contaminación, mal tiempo)
55 (ED)	Control de salida
56 (EC)	Preparación/documentación de carga
57 (EF)	Planes de vuelo
58 (EO)	Otros sistemas automatizados

Tabla A.6: Codigos IATA para atrasos (Tabla 2.3: códigos IATA *Delay* (ANS Performance. (s.f.). *IATA Delay Codes*. ANS Performance. Recuperado de <https://ansperformance.eu/definition/iata-delay-codes/>).

Flight Operations and Crewing	
61 (FP)	Plan de vuelo: completado tardíamente o modificado
62 (FF)	Requerimientos operativos (combustible, cambio de carga)
63 (FT)	Tripulación tardía (embarque o procedimientos)
64 (FS)	Escasez de tripulación de vuelo (enfermedad, espera de reserva)
65 (FR)	Solicitud especial de tripulación de vuelo
66 (FL)	Tripulación de cabina tardía (embarque o procedimientos)
67 (FC)	Escasez de tripulación de cabina
68 (FA)	Error o solicitud especial de tripulación de cabina
69 (FB)	Solicitud del capitán para verificación de seguridad

Tabla A.7: Codigos IATA para atrasos (Tabla 2.3: códigos IATA *Delay* (ANS Performance. (s.f.). *IATA Delay Codes*. ANS Performance. Recuperado de <https://ansperformance.eu/definition/iata-delay-codes/>).

Weather	
71 (WO)	Clima en estación de salida
72 (WT)	Clima en estación de destino
73 (WR)	Clima en ruta o alterno
75 (WI)	Deshielo de aeronave
76 (WS)	Remoción de nieve/hielo/agua/arena en aeropuerto
77 (WG)	Operaciones en tierra afectadas por clima adverso

Tabla A.8: Codigos IATA para atrasos (Tabla 2.3: códigos IATA *Delay* (ANS Performance. (s.f.). *IATA Delay Codes*. ANS Performance. Recuperado de <https://ansperformance.eu/definition/iata-delay-codes/>).

AFTM + Airport + Governmental Authorities Aircraft Traffic Low Management Restrictions	
81 (AT)	Restricciones ATFM por capacidad ATC en ruta
82 (AX)	Restricciones ATFM por falta de personal/equipo ATC
83 (AE)	Restricciones ATFM por cierre en aeropuerto de destino
84 (AW)	Restricciones ATFM por clima en destino

Tabla A.9: Codigos IATA para atrasos (Tabla 2.3: códigos IATA *Delay* (ANS Performance. (s.f.). *IATA Delay Codes*. ANS Performance. Recuperado de <https://ansperformance.eu/definition/iata-delay-codes/>).

Aiport and Governmental Authorities	
85 (AS)	Seguridad obligatoria
86 (AG)	Inmigración, aduanas, salud
87 (AF)	Instalaciones del aeropuerto (stands, iluminación, congestión)
88 (AD)	Restricciones en aeropuerto de destino (cierre, protestas, toque de queda)
89 (AM)	Restricciones en aeropuerto de salida (con/sin ATFM)

Tabla A.10: Codigos IATA para atrasos (Tabla 2.3: códigos IATA *Delay* (ANS Performance. (s.f.). *IATA Delay Codes*. ANS Performance. Recuperado de <https://ansperformance.eu/definition/iata-delay-codes/>).

Reactionary	
91 (RL)	Conexión de carga
92 (RT)	Error en check-in combinado (pasajeros/equipaje)
93 (RA)	Rotación de aeronave (llegada tardía de otro vuelo)
94 (RS)	Rotación de tripulación de cabina
95 (RC)	Rotación de tripulación (espera de otro vuelo)



96 (RO)	Control de operaciones (redireccionamiento, consolidación)
----------------	--

Tabla A.11: Códigos IATA para atrasos (Tabla 2.3: códigos IATA *Delay* (ANS Performance. (s.f.). *IATA Delay Codes*. ANS Performance. Recuperado de <https://ansperformance.eu/definition/iata-delay-codes/>).

Miscellaneous	
97 (MI)	Acción industrial dentro de la aerolínea
98 (MO)	Acción industrial externa (excepto ATC)
99 (MX)	Otros motivos no especificados

Tabla A.12: Códigos IATA para atrasos (Tabla 2.3: códigos IATA *Delay* (ANS Performance. (s.f.). *IATA Delay Codes*. ANS Performance. Recuperado de <https://ansperformance.eu/definition/iata-delay-codes/>).

Anexo B

Gráfico de correlación de Pearson de las variables numéricas estudiadas

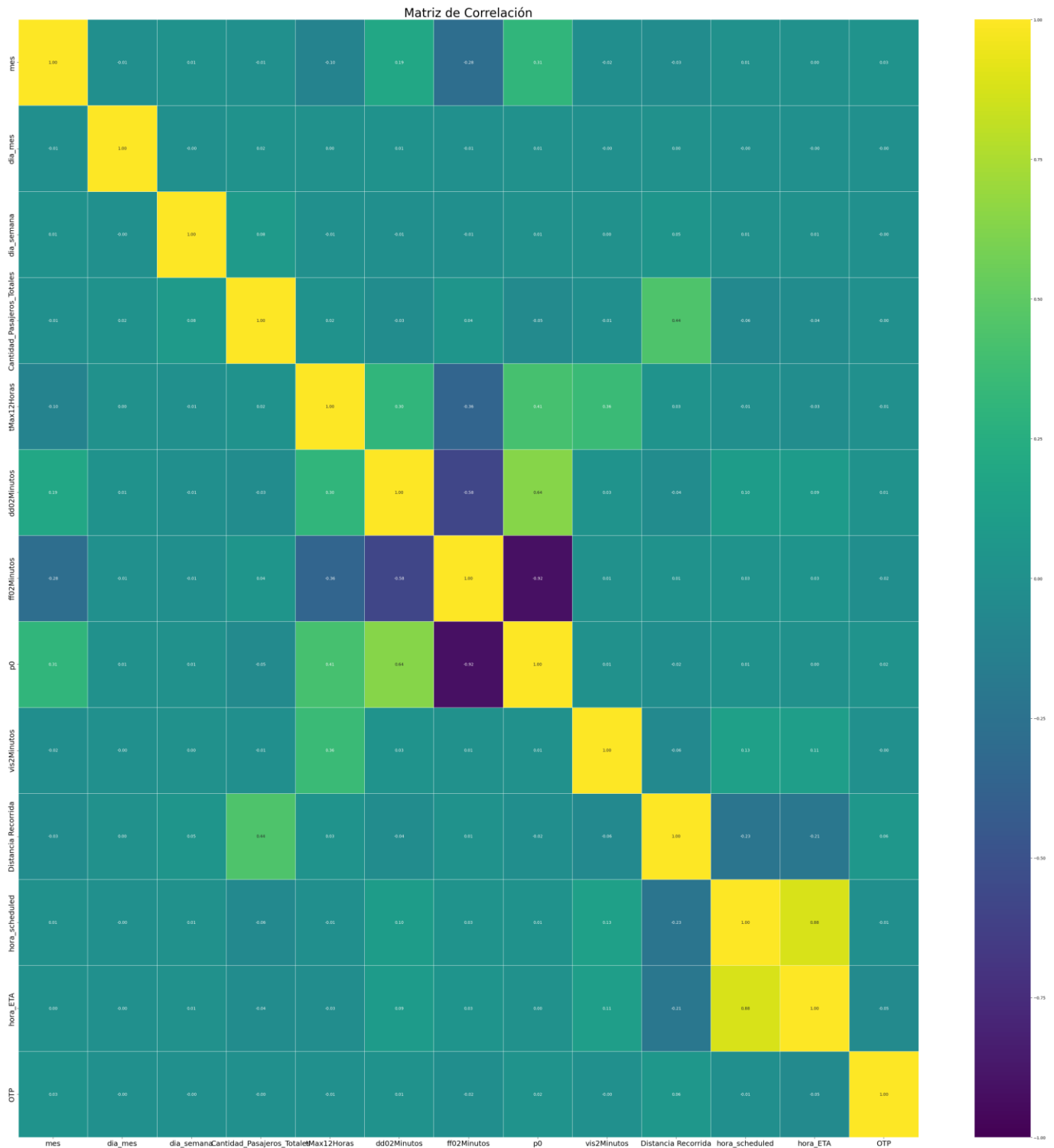


Figura B.1: Matriz de correlación de las variables (Elaboración propia).

Anexo C

Graficas de los modelos de machine learning comparando datos reales (azules) con datos de predicción (rojos)

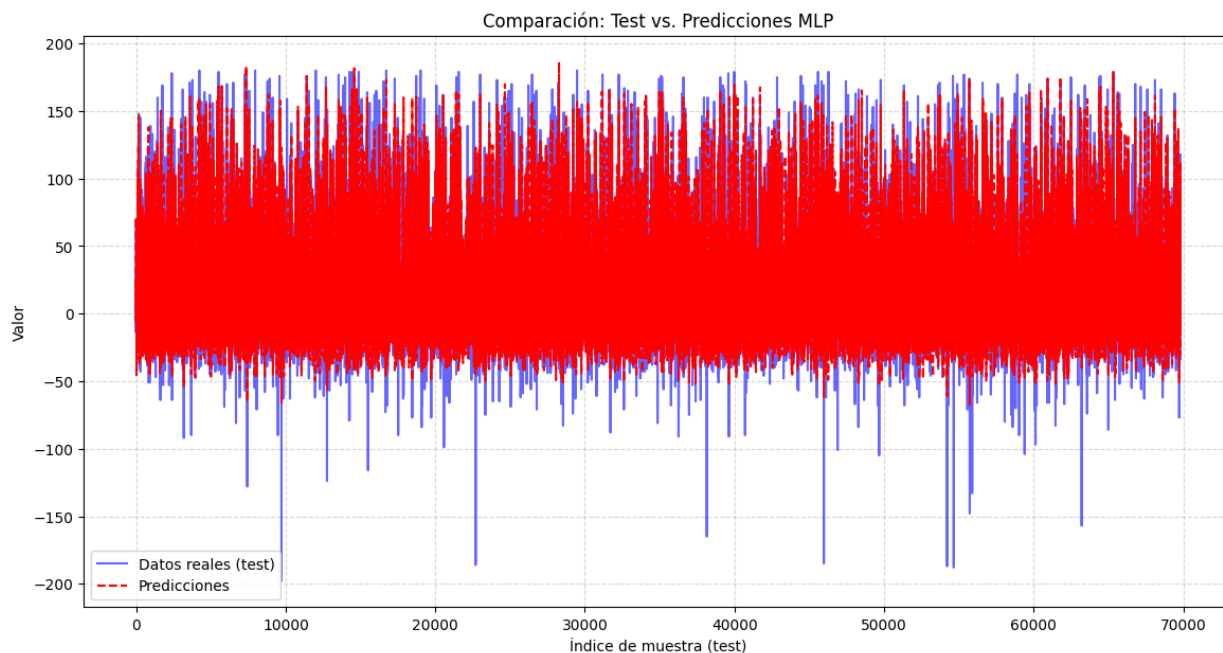


Figura C.1: Resultado de predicción de OTP para el modelo de red neuronal multicapa (Elaboración propia).

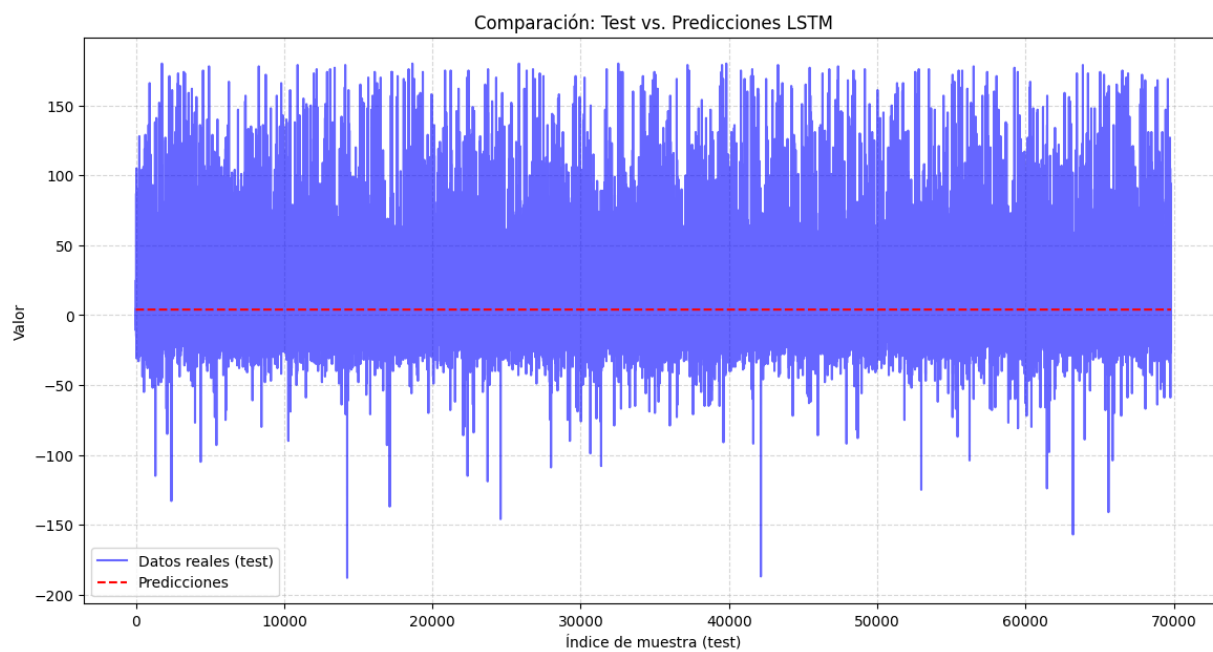


Figura C.2: Resultado de predicción de OTP para el modelo de Long-Short Term Memory LSTM (Elaboración propia).

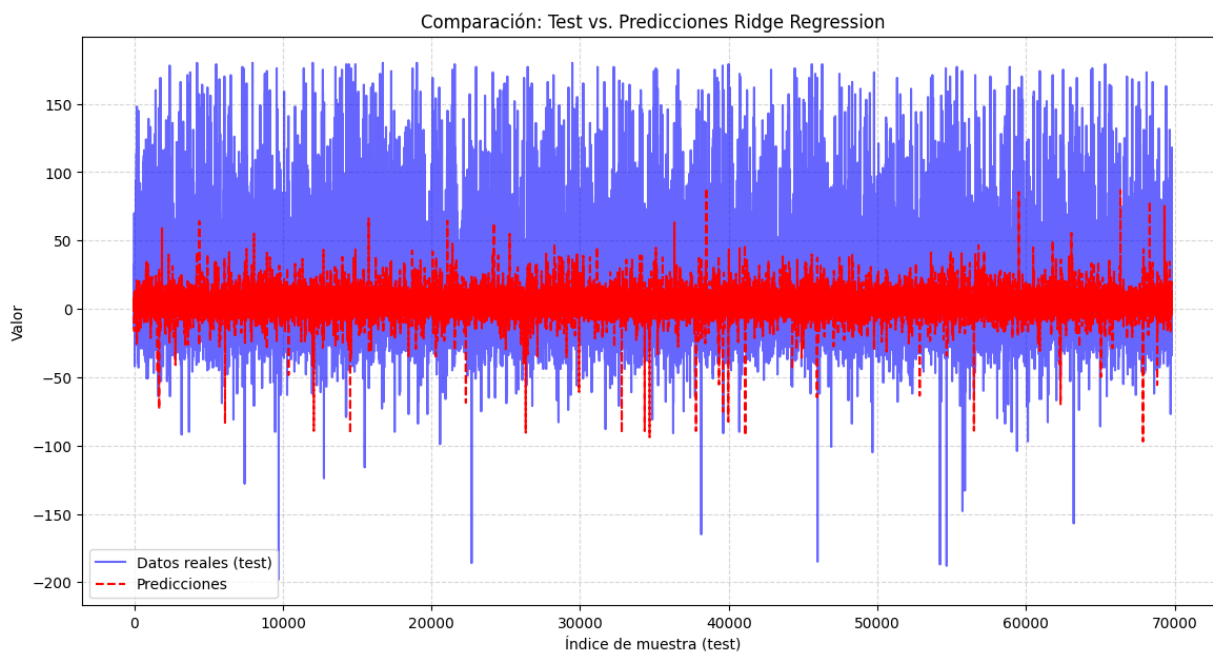


Figura C.3: Resultado de predicción de OTP para el modelo de ridge regression (Elaboración propia).

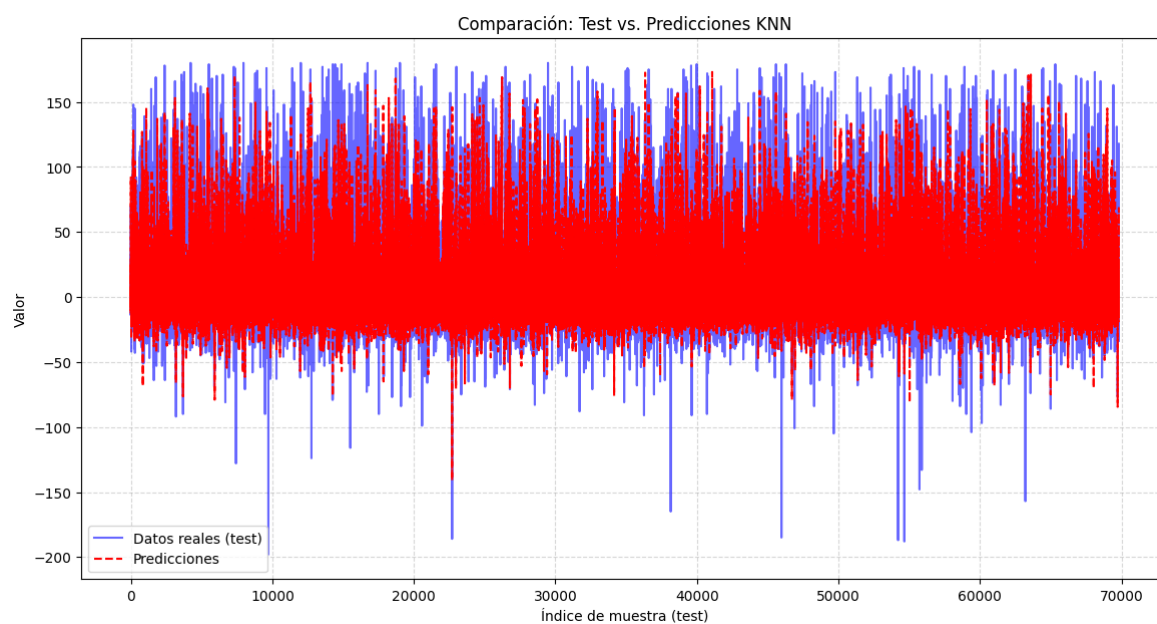


Figura C.4: Resultado de predicción de OTP para el modelo de KNN (Elaboración propia).

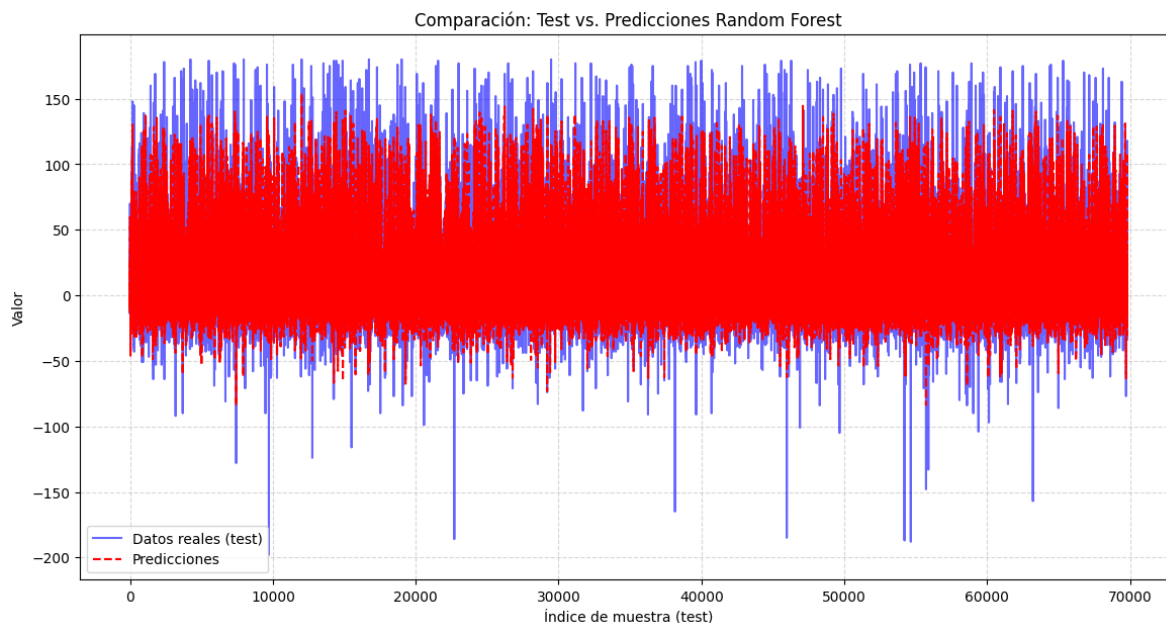


Figura C.5: Resultado de predicción de OTP para el modelo de Random Forest (Elaboración propia).

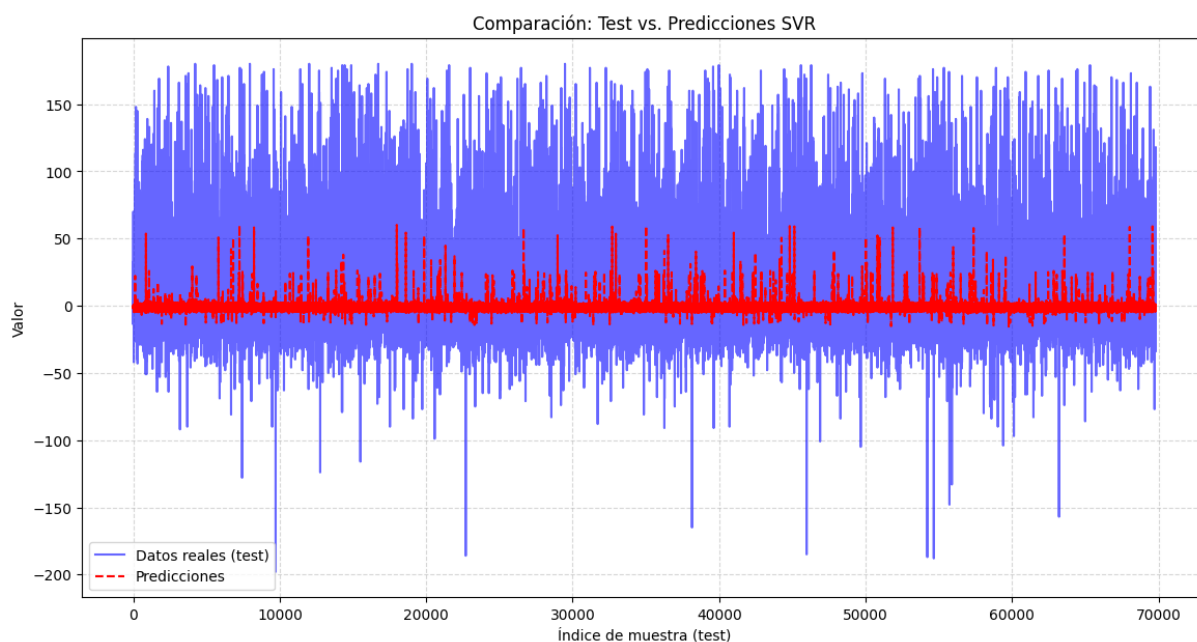


Figura C.6: Resultado de predicción de OTP para el modelo de SVR (Elaboración propia).

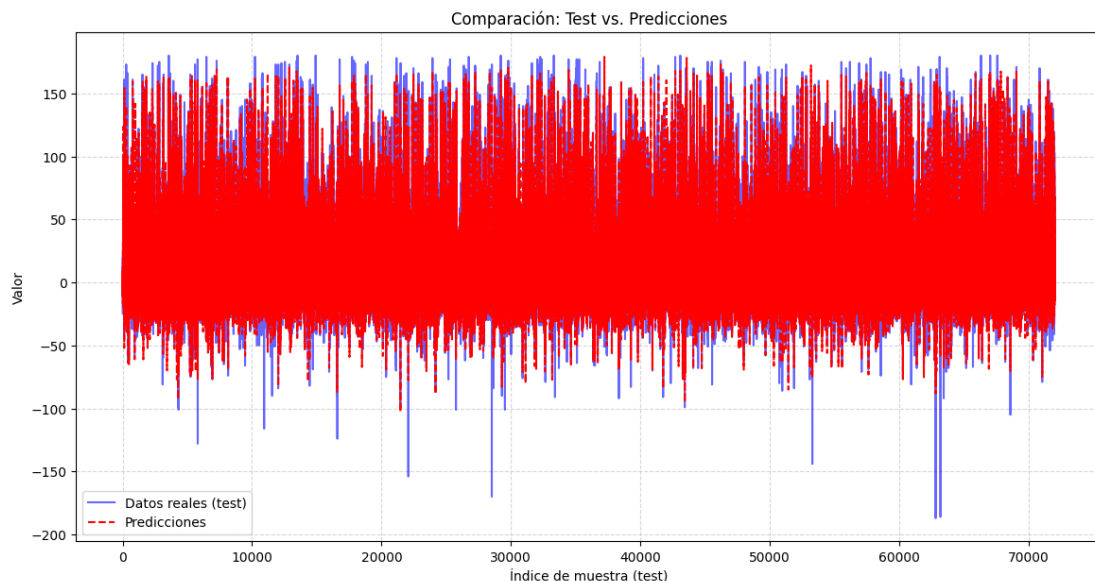


Figura C.7: Resultados de predicción de OTP para el modelo de Red Neuronal Multicapa con función de pérdida de cuartiles (Elaboración propia).

Anexo D

Diagrama de los árboles de decisiones (Random Forest) generados por la IA para la predicción de valores OTP.

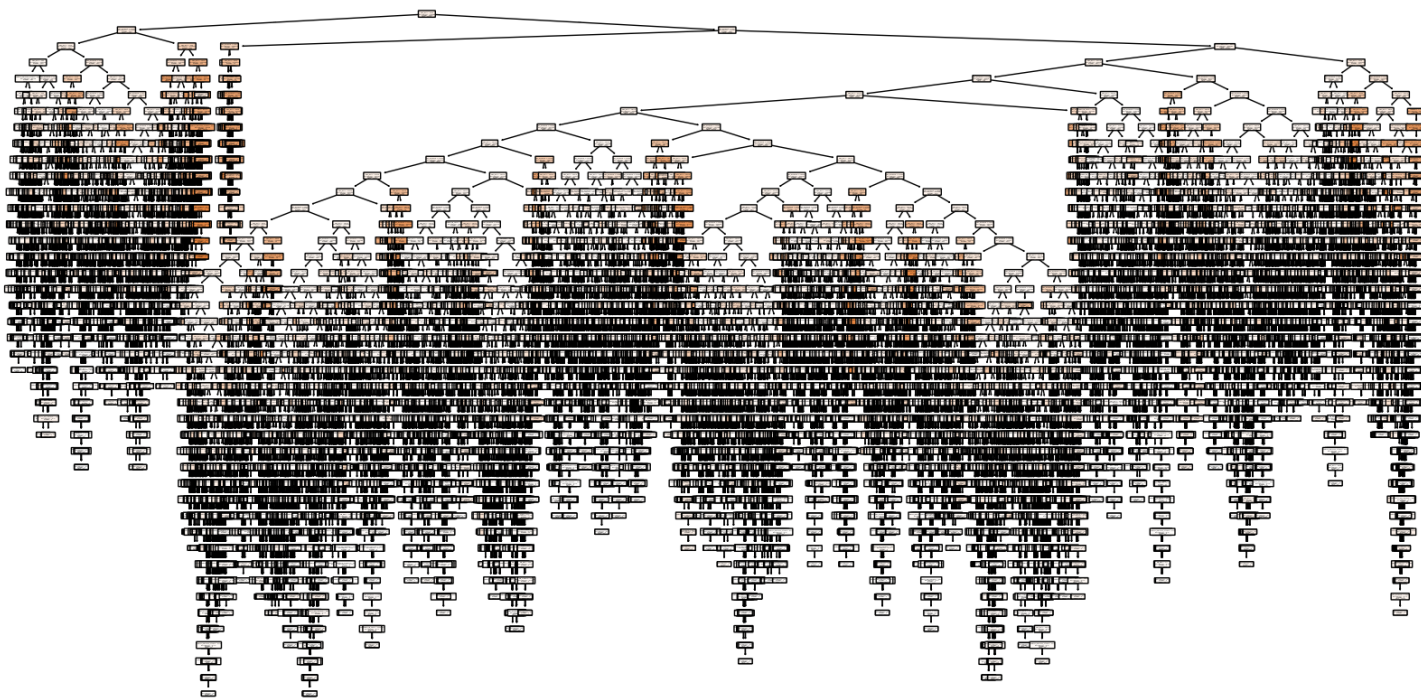


Figura D.1: árbol aleatorio entrenado para predecir OTP (Elaboración propia).



Anexo E

Código en Python del modelo de Machine Learning LSTM, MLP, SVR, KNN, Ridge Bridge y Random Forest

```
import pandas as pd

# Configuración inicial
folder_path = "C:\\Users\\conta\\Desktop\\clima\\"
output_file = folder_path + "datos_meteorologicos_completos.csv"

# 1. Leer todos los archivos CSV con los parámetros correctos
def leer_csv(nombre_archivo):
    """Función para leer los CSV con configuración uniforme"""
    return pd.read_csv(
        folder_path + nombre_archivo,
        sep=";",
        parse_dates=["momento"],
        encoding="latin1",
        on_bad_lines="skip"
    )

# Cargar todos los datasets
df_agua = leer_csv("agua_caida_combinado.csv")
df_nubosidad = leer_csv("Nubosidad_combinado.csv")
df_presion_humedad = leer_csv("presion y humedad_combinado.csv")
df_temperatura = leer_csv("Temperatura.csv")
df_viento = leer_csv("viento.csv")
df_visibilidad = leer_csv("visibilidad.csv")

# 2. Limpieza de columnas redundantes en cada DataFrame
# Agua Caída
df_agua = df_agua.drop(columns=[
    'codigoNacional', 'idEquipo', 'idPista', 'momentoRegistro',
    'rrTotalAnual', 'rr15m', 'nieveAcumulada'
], errors="ignore")

# Nubosidad
df_nubosidad = df_nubosidad.drop(columns=[
    'codigoNacional', 'momentoRegistro', 'isSkyClear', 'isVertVis',
    'visVertical', 'alturaCapa1Inst', 'alturaCapa2Inst', 'alturaCapa3Inst'
], errors="ignore")

# Presión y Humedad
df_presion_humedad = df_presion_humedad.drop(columns=[
    'codigoNacional'
], errors="ignore")

# Temperatura (no eliminamos columnas aquí ya que parecen todas necesarias)
# Viento (no eliminamos columnas aquí)
# Visibilidad (no eliminamos columnas aquí)
```



3. Merge secuencial con sufijos controlados

```
df_final = df_agua.merge(  
    df_nubosidad, on="momento", how="outer", suffixes=("", "_nub")  
)  
.merge(  
    df_presion_humedad, on="momento", how="outer", suffixes=("", "_ph")  
)  
.merge(  
    df_temperatura, on="momento", how="outer", suffixes=("", "_temp")  
)  
.merge(  
    df_viento, on="momento", how="outer", suffixes=("", "_viento")  
)  
.merge(  
    df_visibilidad, on="momento", how="outer", suffixes=("", "_vis")  
)  
)
```

4. Limpieza final de columnas duplicadas

Eliminamos columnas que terminan con _x o _y (por si quedan)

```
df_final = df_final.loc[:, ~df_final.columns.str.endswith(('_x', '_y'))]
```

5. Ordenar por fecha y guardar

```
df_final = df_final.sort_values("momento")  
df_final.to_csv(output_file, index=False, encoding="utf-8")
```

```
print(f"¡Proceso completado! Datos guardados en: {output_file}")
```

```
print(f"Total de registros combinados: {len(df_final):,}")
```

```
print(f"Columnas finales: {list(df_final.columns)}")
```

```
##df = pd.read_excel("C:\\Users\\conta\\Desktop\\base de datos consolidada.xlsx")##  
import pandas as pd
```

```
df = pd.read_excel("C:\\Users\\conta\\Desktop\\DATOS 2023 - 2025.xlsx")
```

```
df1 = df.drop(["Año", "Mes de Operacion", "Dia Operacion", "HORA  
FIJA", "Descripcion Tipo Aeronave", "Columna1", "Puntualidad", "hora", "minuto",  
    "fecha actual", "fecha actual", "delta dia", "delta hora", "año actual", "mes  
actual", "dia actual", "fecha scheduled",  
    "HORA SCHEDULED", "hora scheduled2", 'fecha y hora', 'hora  
Actual', 'Cantidad de Operacion', 'año scheduled', 'mes scheduled', 'dia scheduled'
```

```
    ], axis=1)
```

```
df1 = df1.drop(["Cantidad de Operacion", "Cinta de Retiro de Equipajes", "Tipo  
Aeronave"], axis=1)
```

```
df1 = df1[df1["Svc Type"].isin(["J"])]
```

```
df1 = df1[df1["Categoria Aeronave"].isin(["C", "D", "E"])]
```

```
df1.head(2)
```



```
df1['Scheduled'] = pd.to_datetime(df1['Scheduled'])
df1['ON block'] = pd.to_datetime(df1['ON block'])
df1['Actual'] = pd.to_datetime(df1['Actual'])
df1['ETA'] = pd.to_datetime(df1['Estimadas'])

df_meteorologico =
pd.read_csv("C:\\Users\\conta\\Desktop\\clima\\datos_meteorologicos_completos.csv")

df_meteorologico.head(20)

df_meteorologico = df_meteorologico.rename(columns={"momento": "Scheduled"})

1. Verificación inicial
print("Ejemplos antes de conversión:")
print(df_meteorologico['Scheduled'].head().to_list())

# 2. Conversión directa especificando el formato exacto
df_meteorologico['Scheduled'] = pd.to_datetime(
    df_meteorologico['Scheduled'],
    format='%Y-%m-%d %H:%M:%S', # Formato exacto de tus datos
    errors='raise' # Cambia a 'coerce' si quieres manejar errores silenciosamente
)

# 4. Verificación
print("\nDespués de conversión:")
print(df_meteorologico['Scheduled'].head())
print("\nTipo de dato:", df_meteorologico['Scheduled'].dtype)

"""df_meteorologico['Scheduled'] =
pd.to_datetime(df_meteorologico['Scheduled'].str.split('.').str[0], format='%Y-%m-%d
%H:%M:%S')"""

df_meteorologico['Scheduled'] = pd.to_datetime(
    df_meteorologico['Scheduled'],
    format='%y-%m-%d %H:%M:%S', # Formato personalizado
    errors='coerce' # Convierte valores problemáticos a NaT
)

print(df_meteorologico['Scheduled'].head().to_list())
print(df1['Scheduled'].head().to_list())

# Realizar el merge derecho (right join)
df_combinado = df1.merge(
    df_meteorologico,
    on='Scheduled',
    how='right' # Right join: conserva solo las fechas de df_meteorologico que coincidan
    con df1
)

# Opcional: Eliminar filas con valores NaN (si no hubo coincidencia exacta)
```



```
df_combinado = df_combinado.dropna(subset=['Scheduled'])

# Verificar el resultado
print(f'Registros en df1: {len(df1):,}')
print(f'Registros en df_meteorologico: {len(df_meteorologico):,}')
print(f'Registros combinados: {len(df_combinado):,}')
print(f'\nColumnas resultantes:', df_combinado.columns.tolist())

print("Columnas df1:", df1.columns.tolist())
print("Columnas df_meteorologico:", df_meteorologico.columns.tolist())

# Convertir a datetime y ordenar
df1['Scheduled'] = pd.to_datetime(df1['Scheduled'])
df_meteorologico['Scheduled'] = pd.to_datetime(df_meteorologico['Scheduled'])

df1 = df1.sort_values('Scheduled')
df_meteorologico = df_meteorologico.sort_values('Scheduled')

## 2. Merge aproximado ##

df_combinado = pd.merge_asof(
    df1,                # DataFrame principal
    df_meteorologico,    # Datos meteorológicos
    on='Scheduled',     # Columna común (antes 'momento' en df_meteorologico)
    direction='nearest', # Coincidencia más cercana
    tolerance=pd.Timedelta('15min'), # Máxima diferencia permitida
    suffixes=(',', '_meteo') # Sufijos para columnas duplicadas
)

## 4. Verificación de resultados ##

print("\nResumen del merge:")
print(f'- Registros originales: {len(df1)}')
print(f'- Registros combinados: {len(df_combinado)}')
print(f'- Coincidencias exactas: {(df_combinado['Scheduled'] ==
df_combinado.get('Scheduled_meteo', pd.NaT)).sum()}')

##df_combinado.to_csv("Base de datos consolidada.csv", index=False)##

df_combinado.to_csv("BASE DE DATOS AGAP.csv", index=False)

Se procede a evaluar variable a variable aquellas que no tengan más allá del 10% en
nulos, se eliminarán del estudio.

print(df_combinado.shape)
a = 0
lista = []
for i in df_combinado.columns:
    null_counts = df_combinado[i].isnull().sum()
```



```
if (null_counts/df_combinado.shape[0]) > 0.1:
    print(null_counts)
    print(i)
    print("_____")
    lista.append(i)
    a = a+1
else:
    pass

print(a)

df_combinado = df_combinado.drop(lista,axis=1)
```

EXPLICACION DE VARIABLES

AGUA CAIDA

- rrInst mm Agua caída del instante de la medición
- rr1Hora mm Agua caída acumulada de la última hora
- rr3Horas mm Agua caída acumulada de las últimas 3 hora
- rr6Horas mm Agua caída acumulada de las últimas 6 hora
- rr12Horas mm Agua caída acumulada de las últimas 12 hora
- rr24Horas mm Agua caída acumulada de las últimas 24 hora
- momentoRegistro Fecha y hora en que los datos son persitados en la base de datos
- rrTotalAnual mm Total Anual a partir del 1 de enero a las 00:00
- rr15m mm Agua caída en 15 Minutos

PRESION HUMEDAD

- hr % Humedad relativa del aire
- p0 hPa Presión leída del sensor
- qfe1 hPa Presión A nivel de la estación
- qfe2 hPa Presión A nivel de la estación segunda
- qff hPa Presión A nivel del Mar
- qnh hPa Presión A nivel estándar
- momento Registro Fecha y hora en que los datos son persitados en la base de datos
- hrMax15m % Humedad Relativa Máxima en 15 Minutos
- hrMed15m % Humedad Relativa Media en 15 Minutos
- hrMin15m % Humedad Relativa Mínima en 15 Minutos

TEMPERAURA

- ts °C Temperatura del Aire Seco
- td °C Temperatura del Punto de Rocío
- tMin12Horas °C Temperatura Mínima de las últimas 12 horas
- tMax12Horas °C Temperatura Máxima de las últimas 12 horas
- tMin24Horas °C Temperatura Mínima de las últimas 24 horas
- tMax24Horas °C Temperatura Máxima de las últimas 24 horas
- horaTMin12Horas Hora de la Temperatura Mínima de las últimas 12 horas
- horaTMax12Horas Hora de la Temperatura Máxima de las últimas 12 horas



- horaTMin24Horas Hora de la Temperatura Mínima de las últimas 24 horas
- horaTMax24Horas Hora de la Temperatura Máxima de las últimas 24 horas
- ts02 °C Temperatura a 2 metros
- ts10 °C Temperatura a 10 metros
- ts30 °C Temperatura a 30 metros
- momentoRegistro Fecha y hora en que se persisten los datos en la DB
- tsMed15m °C Temperatura del Aire Media en 15 Minutos
- tsMax15m °C Temperatura del Aire Máxima en 15 Minutos
- tsMin15m °C Temperatura del Aire en 15 Minutos
- tsSupMax15m °C Temperatura de Superficie Media en 15 Minutos
- tsSupMed15m °C Temperatura de Superficie Máxima en 15 Minutos
- tsSupMin15m °C Temperatura de Superficie Mínima en 15 Minutos
- tMin1M °C Temperatura Mínima en 1 minuto
- tMax1M °C Temperatura Máxima en 1 minuto

VIENTO

- ddInst 0° a 359° Temperatura Máxima en 1 minuto
- ffInst Intensidad del Viento del instante de la medición
- dd02Minutos 0° a 359° Dirección promedio de los últimos 2 minutos
- ff02Minutos kt Intensidad promedio de los últimos 2 minutos
- dd10Minutos 0° a 359° Dirección promedio de los últimos 10 minutos
- ff10Minutos kt Intensidad promedio de los últimos 10 minutos
- dd02MinutosMin 0° a 359° Dirección Mínima de los últimos 2 minutos
- ff02MinutosMin kt Intensidad Mínima de los últimos 2 minutos
- dd02MinutosMax 0° a 359° Dirección Máxima de los últimos 2 minutos
- ff02MinutosMax kt Intensidad Máxima de los últimos 2 minutos
- dd10MinutosMin 0° a 359° Dirección Mínima de los últimos 10 minutos
- ff10MinutosMin kt Intensidad Mínima de los últimos 10 minutos
- dd10MinutosMax 0° a 359° Dirección Máxima de los últimos 10 minutos
- ff10MinutosMax kt Intensidad Máxima de los últimos 10 minutos
- dd01Minutos 0° a 359° Dirección del Viento promedio de 1 minuto
- ff01Minutos kt Intensidad del Viento promedio de 1 minuto
- dd01MinutosMin 0° a 359° Dirección Mínima del Viento en 1 minuto
- ff01MinutosMin kt Intensidad Mínima del Viento en 1 minuto
- dd01MinutosMax 0° a 359° Dirección Máxima del Viento en 1 minuto
- ff01MinutosMax kt Intensidad Máxima del Viento en 1 minuto
- momentoRegistro Fecha y hora en que los datos son persistidos en la base de datos
- dd15m 0° a 359° Dirección del Viento en 15 Minutos
- ffMax15m kt Velocidad Viento Máxima en 15 Minutos
- ffMed15m kt Velocidad Viento Media en 15 Minutos
- dn02Min 0° a 359° Dirección del Viento Mínima en 2 minutos
- dx02Min 0° a 359° Dirección del Viento Máxima en 2 minutos
- dn10Min 0° a 359° dirección del Viento Mínima en 10 minutos
- dx10Min 0° a 359° Dirección del Viento Máxima en 10 minutos
- dd02Mts 0° a 359° Dirección del Viento a 2 metros de altura
- ff02Mts Intensidad del Viento a 2 metros de altura

NUBOSIDAD

- isSkyClear Condición de cielo visible
- isVertVis Condición de visibilidad vertical



- visVertical m Visibilidad vertical
- alturaBaseNube m Altura de la base nubosa
- alturaCapa1Inst m Altura instantánea de la Primera Capa en metros
- alturaCapa2Inst m Altura instantánea de la Segunda Capa en metros
- alturaCapa3Inst m Altura instantánea de la Tercera Capa en metros
- cantidadOctasCapa1 0 - 8 octas Cantidad de octas de la Primera Capa
- alturaCapa1 m Altura de la Primera Capa de nubes
- cantidadOctasCapa2 0 - 8 octas Cantidad de octas de la Segunda Capa
- alturaCapa2 m Altura de la Segunda Capa de nubes
- cantidadOctasCapa3 0 - 8 octas Cantidad de octas de la Tercera Capa
- alturaCapa3 m Altura de la Tercera Capa de nubes
- momentoRegistro Fecha y hora en que los datos son persistidos en la base de datos

```
import pandas as pd
```

```
df_combinado = pd.read_csv("C:\\Users\\conta\\Desktop\\Base de datos  
consolidada.csv")
```

```
import pandas as pd
```

```
# Primero asegurar que la fecha es datetime
```

```
df_combinado['Fecha Operacion'] = pd.to_datetime(df_combinado['Fecha Operacion'])
```

```
# Extraer año y trimestre
```

```
df_combinado['Trimestre'] = df_combinado['Fecha Operacion'].dt.year.astype(str) + '-'  
T' + df_combinado['Fecha Operacion'].dt.quarter.astype(str)
```

```
# Calcular el porcentaje de categoría 1 por trimestre
```

```
resultado = df_combinado.groupby('Trimestre').apply(  
    lambda x: pd.Series(  
        'Total_Registros': len(x),  
        'Categoria_1': x['Categoria'].eq(1).sum(),  
        'Porcentaje_Categoria_1': (x['Categoria'].eq(1).mean() * 100).round(2)  
    })  
)  
.reset_index()
```

```
# Ordenar por trimestre cronológicamente
```

```
resultado['Año'] = resultado['Trimestre'].str.split('-T').str[0].astype(int)  
resultado['Trimestre_Num'] = resultado['Trimestre'].str.split('-T').str[1].astype(int)  
resultado = resultado.sort_values(['Año', 'Trimestre_Num']).drop(['Año',  
'Trimestre_Num'], axis=1)
```

```
print(resultado)
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
from matplotlib.ticker import PercentFormatter
```

```
import numpy as np
```



```
# Configurar estilo
plt.style.use('dark_background')
sns.set_palette("husl")

# Preparar datos
resultado['Año'] = resultado['Trimestre'].str.split('-T').str[0]
resultado['Trim'] = resultado['Trimestre'].str.split('-T').str[1]
resultado['Año-Trim'] = resultado['Año'] + '-T' + resultado['Trim']

# Ordenar cronológicamente
resultado = resultado.sort_values(['Año', 'Trim'])

# Crear figura
fig, ax = plt.subplots(figsize=(14, 7), facecolor='#222222')
fig.patch.set_alpha(0.7)

# Gráfico de línea CONTINUO (una sola línea)
main_line = sns.lineplot(
    data=resultado,
    x='Año-Trim',
    y='Porcentaje_Categoria_1',
    color='deepskyblue', # Color principal
    linewidth=3,
    marker='o',
    markersize=10,
    markeredgewidth=1,
    ax=ax
)

# Resaltar puntos por año
unique_years = resultado['Año'].unique()
colors = sns.husl_palette(len(unique_years), h=.6)

for i, year in enumerate(unique_years):
    year_data = resultado[resultado['Año'] == year]
    ax.scatter(
        x=year_data['Año-Trim'],
        y=year_data['Porcentaje_Categoria_1'],
        color=colors[i],
        s=120,
        edgecolor='white',
        linewidth=1.5,
        label=year,
        zorder=5 # Para que queden sobre la línea
    )

# Personalización
ax.set_title('EVOLUCIÓN TRIMESTRAL PUNTUALIDAD',
             fontsize=16, pad=20, color='white', fontweight='bold')
ax.set_xlabel('Trimestre', fontsize=12, color='white', labelpad=10)
```



```
ax.set_ylabel('Porcentaje (%)', fontsize=12, color='white', labelpad=10)
ax.yaxis.set_major_formatter(PercentFormatter())

# Rotar etiquetas
ax.set_xticks(range(len(resultado)))
ax.set_xticklabels([f'T{row['Trim']}\n{row['Año']}' for _, row in resultado.iterrows()],
                    rotation=0, ha='center', color='white')

# Cuadrícula
ax.grid(color='#555555', linestyle='--', alpha=0.3)

# Añadir etiquetas
for x, y in enumerate(resultado['Porcentaje_Categoria_1']):
    ax.text(x, y+1.5, f'{y:.1f}%',
            ha='center', va='bottom',
            fontsize=9, color='white',
            bbox=dict(facecolor='#333333', alpha=0.7, edgecolor='none'))

# Línea de media
mean_val = resultado['Porcentaje_Categoria_1'].mean()
ax.axhline(mean_val, color='cyan', linestyle=':', alpha=0.7, linewidth=2)
ax.text(len(resultado)-0.5, mean_val+1.5, f'Media: {mean_val:.1f}%',
        ha='right', color='cyan', fontsize=10,
        bbox=dict(facecolor='#333333', alpha=0.7, edgecolor='none'))

# Leyenda de años
legend = ax.legend(title='Años', frameon=True,
                  facecolor='#333333', edgecolor='white',
                  bbox_to_anchor=(1.05, 1), loc='upper left')
plt.setp(legend.get_title(), color='white')

# Ajustar márgenes
plt.tight_layout()
plt.subplots_adjust(right=0.85)

plt.show()
```

MODELOS DE MACHINE LEARNING A USAR

```
import pandas as pd
```

```
###raw = pd.read_csv("C:\\Users\\conta\\Desktop\\Base de datos consolidada.csv")###
```

```
raw = pd.read_excel("C:\\Users\\conta\\Desktop\\Proyecto Predict and Optimize\\BASE DE DATOS AEROPUERTO.xlsx")
```

```
raw.rename(columns={'Numero de Vuelo': 'Numero_de_Vuelo'}, inplace=True)
raw.rename(columns={'Tipo Stand': 'Tipo_Stand'}, inplace=True)
raw.rename(columns={'Puertas de Embarque': 'Puertas_de_Embarque'}, inplace=True)
raw.rename(columns={'Cantidad Pasajeros Totales': 'Cantidad_Pasajeros_Totales'},
            inplace=True)
raw.rename(columns={'Categoria Aeronave': 'Categoria_Aeronave'}, inplace=True)
```



```
raw.rename(columns={'Matricula Aeronave': 'Matricula_Aeronave'}, inplace=True)
```

```
eliminar = [  
    'rr1Hora', 'rr3Horas', 'rr6Horas',  
    'rr12Horas', 'rr24Horas', 'qfe1',  
    'qfe2', 'qff', 'qnh',  
    'td', 'tMin12Horas',  
    'tMin24Horas', 'tMax24Horas',  
    'ffInst',  
    'dd10Minutos', 'ff10Minutos', 'dd02MinutosMin', 'ff02MinutosMin',  
    'dd02MinutosMax', 'ff02MinutosMax', 'dd10MinutosMin', 'ff10MinutosMin',  
    'dd10MinutosMax', 'ff10MinutosMax',  
    'vis10Minutos', 'vis10MinutosMin',  
    'vis10MinutosMax']
```

```
usar = ['Fecha Operacion', 'OPERACION', 'Aerolinea', 'Pais',  
    'Aeropuerto', 'Categoria_Aeronave',  
    'Numero_de_Vuelo', 'Matricula_Aeronave', 'Tipo_Stand',  
    'Stand', 'Puertas_de_Embarque', 'NEGOCIO', 'Cantidad_Pasajeros_Totales',  
    'p0', 'tMax12Horas', 'dd02Minutos', 'ff02Minutos',  
    'vis2Minutos', 'Scheduled', 'ETA', 'Actual', 'ON block', 'OFF  
block', 'Incumplimiento', 'Distancia Recorrida']
```

```
raw = raw[usar]  
raw.head(2)
```

```
raw = raw.copy()
```

```
# Convertir fechas (manejar errores)
```

```
raw["Scheduled"] = pd.to_datetime(raw["Scheduled"], errors='coerce')  
raw["Actual"] = pd.to_datetime(raw["Actual"], errors='coerce')  
raw["ETA"] = pd.to_datetime(raw["ETA"], errors='coerce')  
raw["ON block"] = pd.to_datetime(raw["ON block"], errors='coerce')  
raw["OFF block"] = pd.to_datetime(raw["OFF block"], errors='coerce')
```

```
# Crear características temporales
```

```
raw["año"] = raw["Scheduled"].dt.year  
raw["trimestre"] = raw["Scheduled"].dt.quarter  
raw["mes"] = raw["Scheduled"].dt.month  
raw["dia_mes"] = raw["Scheduled"].dt.day  
raw["dia_semana"] = raw["Scheduled"].dt.dayofweek
```

```
# Para las horas, manejar valores nulos
```

```
raw['hora_scheduled'] = (raw['Scheduled'].dt.hour * 100 + raw['Scheduled'].dt.minute)  
raw['hora_actual'] = (raw['Actual'].dt.hour * 100 + raw['Actual'].dt.minute)  
raw['hora_ETA'] = (raw['ETA'].dt.hour * 100 + raw['ETA'].dt.minute)
```

```
# Convertir a Int64 (que soporta nulos) en lugar de int
```



```
raw['hora_scheduled'] = raw['hora_scheduled'].astype('Int64')
```

```
raw['hora_actual'] = raw['hora_actual'].astype('Int64')
```

```
raw['hora_ETA'] = raw['hora_ETA'].astype('Int64')
```

```
raw.columns
```

```
raw.rename(columns={'Numero de Vuelo': 'Numero_de_Vuelo'}, inplace=True)
```

```
raw.rename(columns={'Tipo Stand': 'Tipo_Stand'}, inplace=True)
```

```
raw.rename(columns={'Puertas de Embarque': 'Puertas_de_Embarque'}, inplace=True)
```

```
raw.rename(columns={'Cantidad Pasajeros Totales': 'Cantidad_Pasajeros_Totales'},  
inplace=True)
```

```
A = raw.isnull().sum()
```

```
print(A)
```

```
from sklearn.preprocessing import LabelEncoder as LE
```

```
raw['Numero_de_Vuelo'] = raw['Numero_de_Vuelo'].astype(str)
```

```
columnas_para_dropna = ['p0', 'tMax12Horas', 'dd02Minutos', 'ff02Minutos',  
'vis2Minutos', 'hora_ETA']
```

```
# Solo eliminar filas donde TODAS las columnas especificadas son nulas
```

```
raw = raw.dropna(subset=columnas_para_dropna)
```

```
#FILTRO DE DATOS DE AGOSTO PARA LUEGO VALIDAR EL MODELO CON  
EL MES DE AGOSTO#
```

```
filtro_agosto_2025 = (raw['Fecha Operacion'].str.startswith('2025-05')) | \  
                    (raw['Fecha Operacion'].str.startswith('2025-06'))
```

```
# 3. Crear el nuevo DataFrame con las operaciones de agosto de 2025.
```

```
df_agosto_2025 = raw[filtro_agosto_2025].copy()
```

```
# 4. Eliminar las filas filtradas del DataFrame original (df).
```

```
# El operador '~' invierte la máscara, seleccionando todas las filas *excepto* las de  
agosto de 2025.
```

```
raw = raw[~filtro_agosto_2025].copy()
```

```
# 5. Guardar el nuevo DataFrame en un archivo Excel.
```

```
output_excel_file = 'operaciones_agosto_2025:2.xlsx'
```

```
df_agosto_2025.to_excel(output_excel_file, index=False)
```

```
#
```

```
#
```

```
#ESTRUCTURA DE CREACIÓN DE ARCHIVOS QUE GUARDAN LAS  
LABELS EN .PKL PARA QUE EN OTRO CODIGO SE LLAMEN CON  
JOBLIB.LOAD()
```



#ESTO SE HACE SI LA BASE DE DATOS ES NUEVA, EN CASO DE QUE YA SE REALIZÓ ENTRENAMIENTOS PREVIOS, OMITIR Y SOLO BUSCAR LOS ARCHIVOS .PKL PARA CODIFICAR LABELS DE SU BASE DE DATOS

```
from sklearn.preprocessing import LabelEncoder as LE
raw['Numero_de_Vuelo'] = raw['Numero_de_Vuelo'].astype(str)

columnas_para_dropna = ['p0', 'tMax12Horas', 'dd02Minutos', 'ff02Minutos',
'vis2Minutos', 'hora_ETA']

# Solo eliminar filas donde TODAS las columnas especificadas son nulas
raw = raw.dropna(subset=columnas_para_dropna)

import pandas as pd
from sklearn.preprocessing import LabelEncoder as LE
import joblib
import os
# Asumimos que 'raw' ya está cargado aquí

# Directorio donde se guardarán los archivos
CODERS_DIR = r'C:\Users\conta\Desktop\CODERS'
os.makedirs(CODERS_DIR, exist_ok=True)
print(f'Directorio de codificadores asegurado: {CODERS_DIR}')

# Lista de columnas categóricas a codificar
columnas_categoricas = [
    'Aerolinea', 'Pais', 'Aeropuerto', 'Matricula_Aeronave', 'Tipo_Stand',
    'Puertas_de_Embarque', 'NEGOCIO', 'Stand', 'OPERACION',
    'Numero_de_Vuelo', 'Categoria_Aeronave'
]

# --- 1. PREPROCESAMIENTO Y FILTROS INICIALES ---

raw['Numero_de_Vuelo'] = raw['Numero_de_Vuelo'].astype(str)
columnas_para_dropna = ['p0', 'tMax12Horas', 'dd02Minutos', 'ff02Minutos',
'vis2Minutos', 'hora_ETA']
raw = raw.dropna(subset=columnas_para_dropna)

# Diccionario para almacenar las instancias de LabelEncoder
encoders_entrenados = {}

# 2. AJUSTAR Y GUARDAR CADA INSTANCIA SEPARADAMENTE
print("Iniciando ajuste y guardado de LabelEncoders individuales...")

for col in columnas_categoricas:

    # Crea una instancia para esta columna
    le_instance = LE()
```



```
# CRÍTICO: Usar .astype(str) para replicar tu entrenamiento y asegurar consistencia
data_to_fit = raw[col].astype(str)
```

```
# 3. Ajustar y transformar (fit_transform)
raw[col] = le_instance.fit_transform(data_to_fit)
```

```
# 4. Almacenar y Guardar la instancia entrenada
encoders_entrenados[col] = le_instance
```

```
filename = f'le_{col}.pkl'
joblib.dump(le_instance, os.path.join(CODERS_DIR, filename))
```

```
print(f'✅ Codificador '{col}' ajustado y guardado.")
```

```
# Ahora, la variable 'raw' está codificada y todos los codificadores están guardados.
# Puedes continuar con el entrenamiento de tu MLP.
```

```
#                                LLAMADO A LA CARPETA CON CODIGOS PARA
CLASIFICAR DATOS                                #
```

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder as LE
import joblib
import os
```

```
# Directorio de codigos de los datos
```

```
CODERS_DIR = r'C:\\Users\\conta\\Desktop\\CODERS'
```

```
# Lista de columnas categóricas a codificar
columnas_categoricas = [
    'Aerolinea', 'Pais', 'Aeropuerto', 'Matricula_Aeronave', 'Tipo_Stand',
    'Puertas_de_Embarque', 'NEGOCIO', 'Stand', 'OPERACION',
    'Numero_de_Vuelo', 'Categoria_Aeronave']
```

```
encoders_entrenados = {}
```

```
for col in columnas_categoricas:
```

```
    # Crea una instancia para esta columna
    le_instance = LE()
```

```
    # CRÍTICO: Usar .astype(str) para replicar tu entrenamiento y asegurar consistencia
    data_to_fit = raw[col].astype(str)
```

```
    # 3. Ajustar y transformar (fit_transform)
    raw[col] = le_instance.fit_transform(data_to_fit)
```

```
    # 4. Almacenar y Guardar la instancia entrenada
    encoders_entrenados[col] = le_instance
```



```
filename = f'le_{col}.pkl'
joblib.dump(le_instance, os.path.join(CODERS_DIR, filename))

print(f"✅ Codificador '{col}' ajustado y guardado.")

# Ahora, la variable 'raw' está codificada y todos los codificadores están guardados.
# Puedes continuar con el entrenamiento de tu MLP.

columnas = ['hora_ETA']

raw = raw.dropna(subset=columnas)
raw

categoricas =
['OPERACION','Aerolinea','Pais','Aeropuerto','Categoria_Aeronave','Numero_de_vuel
o','Matricula_Aeronave','NEGOCIO']

ciclicos = ['año','trimestre','mes','dia_mes','dia_semana','hora_scheduled','hora_ETA']

continuas =
['Cantidad_Pasajeros_Totales','p0','tMax12Horas','dd02Minutos','ff02Minutos','vis2Mi
nutos','Distancia Recorrida']

# _____#
#proceso de ciclicidad de los datos temporales

import numpy as np
import pandas as pd
# Asume que 'df' es tu DataFrame con todas las columnas.

#
=====
=====
# FUNCIÓN DE TRANSFORMACIÓN CÍCLICA
#
=====
=====

def apply_cyclic_encoding(df, col_name, period):
    """Aplica la transformación Seno y Coseno para una columna cíclica."""
    # Los nuevos nombres de columna incluyen la base y la transformación.
    df[f'{col_name}_sin'] = np.sin(2 * np.pi * df[col_name] / period)
    df[f'{col_name}_cos'] = np.cos(2 * np.pi * df[col_name] / period)
    return df

#
=====
=====
# 1. CONVERSIÓN DE HHMM A MINUTOS (para Hora)
```



```
#
=====

# El formato HHMM (2350, 953) debe convertirse a Minutos del Día (0-1439)
# para que la transformación cíclica funcione linealmente con respecto al tiempo.

# Función para convertir HHMM a Minutos del Día (e.g., 2350 -> 1430 minutos)
def convert_hhmm_to_minutes(hhmm_series):
    # Asegurar que sean enteros (Int64 soporta nulos)
    hhmm_series = hhmm_series.astype('Int64').dropna()

    # Horas: 2350 // 100 = 23
    hours = hhmm_series // 100
    # Minutos: 2350 % 100 = 50
    minutes = hhmm_series % 100

    return hours * 60 + minutes

# Aplicar la conversión
raw['hora_scheduled'] = convert_hhmm_to_minutes(raw['hora_scheduled'])
raw['hora_ETA'] = convert_hhmm_to_minutes(raw['hora_ETA'])

# La hora tiene un periodo de 1440 minutos (24 * 60)
PERIOD_MINUTES = 1440

#
=====

# 2. APLICACIÓN DE LA CODIFICACIÓN CÍCLICA
#
=====

# -----
# A. Variables con Ciclo Fijo
# -----

# Trimestre: Periodo = 4 (1, 2, 3, 4)
raw = apply_cyclic_encoding(raw, 'trimestre', 4)

# Mes: Periodo = 12 (1 a 12)
raw = apply_cyclic_encoding(raw, 'mes', 12)

# Día de la Semana: Periodo = 7 (0 a 6)
raw = apply_cyclic_encoding(raw, 'dia_semana', 7)

# -----
# B. Variables con Ciclo Variable/Convertido
# -----

# Día del Mes: Periodo = 31 (Máximo posible)
```



```
# Nota: Usamos 31 y el modelo aprenderá las variaciones a través de la característica
'mes'.
raw = apply_cyclic_encoding(raw, 'dia_mes', 31)

# Hora Programada (Scheduled): Periodo = 1440 minutos
raw = apply_cyclic_encoding(raw, 'hora_scheduled', PERIOD_MINUTES)

# Hora Estimada (ETA): Periodo = 1440 minutos
raw = apply_cyclic_encoding(raw, 'hora_ETA', PERIOD_MINUTES)

print("✅ Transformación cíclica completada.")
print("Ahora tienes nuevas columnas (ej. 'mes_sin', 'trimestre_cos', etc.) que
representan la ciclicidad.")

# _____#
#proceso de normalización de variables continuas

from sklearn.preprocessing import MinMaxScaler
import numpy as np

# Excluir 'Incumplimiento' si es la variable objetivo (y) que no debe ser escalada.
# Si 'Incumplimiento' se usa como característica (X) en algún momento, debe escalarse.
# Pero generalmente, la variable 'y' se mantiene en su escala original.
# Para este ejemplo, lo excluirémos asumiendo que es tu target.

# Crear el objeto MinMaxScaler
scaler = MinMaxScaler()

# =====
# APLICAR NORMALIZACIÓN
# =====

# NOTA IMPORTANTE: Debes asegurarte de manejar los NaNs (valores faltantes)
# antes de esta etapa. MinMaxScaler no maneja NaNs.
# En tu código anterior ya hacías un 'dropna', pero si hay nulos remanentes
# en estas columnas, el escalado fallará.

# Aplicar el escalador a las columnas continuas
# El resultado son arrays de numpy, los reasignamos al DataFrame (df)
raw[continuas] = scaler.fit_transform(raw[continuas])

# Opcional: Si necesitas guardar el objeto scaler para aplicarlo a nuevos datos
(predicción)
# (ej. a un conjunto de validación o a los datos futuros)
# import joblib
# joblib.dump(scaler, 'minmax_scaler.pkl')

print("✅ Normalización de características continuas completada.")
```



```
print(f"Columnas normalizadas: {continuas}")

raw = raw[raw['Incumplimiento']>-200]
raw = raw[raw["Incumplimiento"]<=180]

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# --- PASO 1: Filtrar los datos por 'OPERACION' ---
# Suponiendo que 'raw' ya está cargado y disponible
incumplimiento_llegada = raw[raw['OPERACION'] == 0]['Incumplimiento']
incumplimiento_salida = raw[raw['OPERACION'] == 1]['Incumplimiento']

# --- PASO 2: Calcular estadísticas por grupo ---
# Llegada (OPERACION = 0)
media_0 = np.mean(np.abs(incumplimiento_llegada))
varianza_0 = incumplimiento_llegada.var()
texto_0 = f"Llegada (0)\nMedia: {media_0:.2f}\nVarianza: {varianza_0:.2f}"

# Salida (OPERACION = 1)
media_1 = np.mean(np.abs(incumplimiento_salida))
varianza_1 = incumplimiento_salida.var()
texto_1 = f"Salida (1)\nMedia: {media_1:.2f}\nVarianza: {varianza_1:.2f}"

# --- PASO 3: Generar el gráfico ---
plt.style.use('default')
plt.figure(figsize=(12, 7))

# Se usa la función hist con una lista de datos
# Esto superpone los histogramas, les da colores automáticos y genera la leyenda
plt.hist(
    [incumplimiento_llegada, incumplimiento_salida],
    bins=100,
    label=['Llegada (0)', 'Salida (1)'], # Leyendas para cada grupo
    color=['blue', 'red'],             # Colores Azul y Rojo
    alpha=0.7                           # Transparencia para ver ambos gráficos
)

plt.title('Distribución de Incumplimiento por Tipo de Operación')
plt.xlabel('Incumplimiento')
plt.ylabel('Frecuencia')

# Límites de visualización (tomados de tu código original)
plt.xlim(right=250)
plt.xlim(left=-10)

# Anotaciones de las estadísticas para Llegada (Azul)
plt.text(
    185, 9000,
    texto_0,
```



```
    fontsize=11,
    color="blue",
    bbox=dict(facecolor='white', alpha=0.6, edgecolor='blue') # Borde azul
)

# Anotaciones de las estadísticas para Salida (Roja)
plt.text(
    185, 4500, # Ajusta la posición vertical para que no se superpongan
    texto_1,
    fontsize=11,
    color="red",
    bbox=dict(facecolor='white', alpha=0.6, edgecolor='red') # Borde rojo
)

plt.legend(title='Operación') # Muestra la leyenda de los colores
plt.grid(axis='y', alpha=0.5) # Deja solo las líneas de la grilla verticales
plt.tight_layout() # Ajusta el layout para que no se corten etiquetas
plt.show() # O plt.savefig('histograma_dual.png')

import seaborn as sns

columnas_interes = ['mes', 'dia_mes', 'dia_semana', 'Cantidad_Pasajeros_Totales',
'tMax12Horas',
    'dd02Minutos', 'ff02Minutos', 'p0', 'vis2Minutos', 'Distancia Recorrida',
'hora_scheduled', 'hora_ETA',
    'OTP'] #se quita todas las variables que sean categoricas y se dejan solo las
numericas

matriz_correlacion = raw[columnas_interes].corr()

plt.figure(figsize=(50, 50))
sns.heatmap(matriz_correlacion,
    annot=True,
    cmap='viridis',
    fmt=".2f",
    vmin=-1,
    vmax=1,
    linewidths=0.5)
plt.title('Matriz de Correlación', fontsize=30)

plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.show()

from sklearn.model_selection import train_test_split

# la parte de grafica se usará para sacar información de fechas para construir las
graficas de distribución más abajo.
```



```
y_grafica=raw.Incumplimiento.values
x_grafica=raw.drop(["ETA","Actual","ON block","OFF
block","OPERACION","hora_actual","Incumplimiento",'Tipo_Stand', 'Stand',
'Puertas_de_Embarque'],axis=1).values #se sacara negocio y tipo de stand dado que no
representa una buena variable para el modelo
```

```
['mes', 'dia_mes', 'dia_semana', 'Aerolinea', 'Aeropuerto',
'matricula', 'Tipo_Stand', 'Stand', 'NEGOCIO', 'OPERACION',
'Numero_de_Vuelo', 'Cantidad_Pasajeros_Totales', 'tMax12Horas',
'dd02Minutos', 'ff02Minutos', 'p0', 'vis2Minutos', 'hora_scheduled',
'hora_real']
```

```
x_train_grafica, x_test_grafica, y_train_grafica, y_test_grafica = train_test_split(
    x_grafica, y_grafica, test_size=0.2, random_state=42
)
```

```
"
_____
"
```

```
y=raw.Incumplimiento.values
x=raw.drop(["ETA","Scheduled","Actual","ON block","OFF block","Fecha
Operacion","hora_actual","Incumplimiento",'Tipo_Stand', 'Stand',
'Puertas_de_Embarque'],axis=1).values #se sacara negocio y tipo de stand dado que no
representa una buena variable para el modelo
```

```
['mes', 'dia_mes', 'dia_semana', 'Aerolinea', 'Aeropuerto',
'matricula', 'Tipo_Stand', 'Stand', 'NEGOCIO', 'OPERACION',
'Numero_de_Vuelo', 'Cantidad_Pasajeros_Totales', 'tMax12Horas',
'dd02Minutos', 'ff02Minutos', 'p0', 'vis2Minutos', 'hora_scheduled',
'hora_real']
```

```
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, random_state=42
)
```

```
print(f"Train: {x_train.shape}, Test: {x_test.shape}")
print(f"Train_grafica: {x_train_grafica.shape}, Test_grafica: {x_test_grafica.shape}")
```

```
import matplotlib.pyplot as plt
```



```
import numpy as np

fig, ax = plt.subplots(1, 1, figsize=(12, 8))
ax.set_xlim(0, 10)
ax.set_ylim(0, 10)
ax.axis('off')

# Configuración de las capas
capas = [
    {"nombre": "Capa de entrada", "neuronas": 10, "x": 2},
    {"nombre": "Capa oculta 1", "neuronas": 10, "x": 5},
    {"nombre": "Capa oculta 2", "neuronas": 10, "x": 7},
    {"nombre": "Capa de salida", "neuronas": 1, "x": 9}
]

# Dibujar neuronas como círculos
for capa in capas:
    y_positions = np.linspace(2, 8, capa["neuronas"])
    for y in y_positions:
        circle = plt.Circle((capa["x"], y), 0.1, color='blue', alpha=0.6)
        ax.add_patch(circle)

    # Etiqueta de la capa
    ax.text(capa["x"], 9, capa["nombre"], ha='center', va='center', fontsize=10,
            weight='bold')
    ax.text(capa["x"], 9.3, f'({capa["neuronas"]} neuronas)', ha='center', va='center',
            fontsize=9)

# Dibujar conexiones con flechas
for i in range(len(capas) - 1):
    capa_actual = capas[i]
    capa_siguiente = capas[i + 1]

    # Muestra solo algunas conexiones para que no se vea sobrecargado
    neuronas_origen = np.linspace(2, 8, min(10, capa_actual["neuronas"]))
    neuronas_destino = np.linspace(2, 8, min(10, capa_siguiente["neuronas"]))

    for y_origen in neuronas_origen:
        for y_destino in neuronas_destino:
            ax.arrow(capa_actual["x"] + 0.1, y_origen,
                    capa_siguiente["x"] - capa_actual["x"] - 0.2,
                    y_destino - y_origen,
                    head_width=0.05, head_length=0.1,
                    fc='gray', ec='gray', alpha=0.3,
                    length_includes_head=True)

plt.title('Arquitectura de Red Neuronal', fontsize=14, weight='bold')
plt.tight_layout()
plt.savefig('red_neuronal_circulos.png', dpi=300, bbox_inches='tight')
plt.show()
```



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.optimizers import Adam
```

```
# =====
# Preparación de datos
# =====
# Asegúrate de tener raw cargado
y = raw.Incumplimiento.values.astype(np.float32)
X = raw.drop(["ETA", "Scheduled", "Actual", "ON block", "OFF
block", "OPERACION", "Fecha Operacion", "hora_actual", "Incumplimiento"],
axis=1).values.astype(np.float32)
```

```
# =====
# Secuencias para LSTM
# =====
def create_sequences(X, y, timesteps=5):
    Xs, ys = [], []
    for i in range(len(X) - timesteps):
        Xs.append(X[i:i+timesteps])
        ys.append(y[i+timesteps])
    return np.array(Xs, dtype=np.float32), np.array(ys, dtype=np.float32)
```

```
timesteps = 5
X_seq, y_seq = create_sequences(X, y, timesteps)
```

```
# Split para modelos clásicos
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```
# Split para LSTM
X_train_seq, X_test_seq, y_train_seq, y_test_seq = train_test_split(
    X_seq, y_seq, test_size=0.2, random_state=42
)
```

```
# =====
# Modelos
# =====
models = {
```



```
"MLP": Sequential([
    Dense(10, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(10, activation='relu'),
    Dense(10, activation='relu'),
    Dense(1, activation='linear')

]),
"LSTM": Sequential([
    LSTM(64, activation='tanh', input_shape=(timesteps, X.shape[1])),
    Dense(32, activation='relu'),
    Dense(1, activation='linear')
]),
"Ridge Regression": Ridge(alpha=1.0),

"KNN": KNeighborsRegressor(n_neighbors=2),

"Random Forest": RandomForestRegressor(
    n_estimators=500,
    min_samples_split=8,
    min_samples_leaf=9,
    max_depth=50,
    max_features=0.75,
    random_state=42),

"SVR": SVR(kernel='rbf', C=1.0)
}

# =====
# Entrenamiento y evaluación
# =====
results = []
predictions = {} # <- Aquí se guardarán todas las predicciones

for name, model in models.items():
    print(f"\n==== Entrenando {name} ====")

    if name == "MLP":
        model.compile(optimizer=Adam(learning_rate=0.001), loss='mse')
        history = model.fit(X_train, y_train, epochs=50, batch_size=8, verbose=0)
        y_pred = model.predict(X_test).reshape(-1)
        y_test_eval = y_test

    elif name == "LSTM":
        model.compile(optimizer=Adam(learning_rate=0.001), loss='mse')
        history = model.fit(X_train_seq, y_train_seq, epochs=1, batch_size=8, verbose=0,
                            validation_split=0.1)
        y_pred = model.predict(X_test_seq).reshape(-1)
        y_test_eval = y_test_seq

    else: # Modelos scikit-learn
```



```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
y_test_eval = y_test

# Guardar métricas
mse = mean_squared_error(y_test_eval, y_pred)
mae = mean_absolute_error(y_test_eval, y_pred)
r2 = r2_score(y_test_eval, y_pred)

results.append({
    'Model': name,
    'MSE': round(mse, 4),
    'MAE': round(mae, 4),
    'R²': round(r2, 4)
})

# Guardar predicciones y reales
predictions[name] = {
    "y_true": y_test_eval,
    "y_pred": y_pred
}

# =====
# Gráfico comparativo
# =====
plt.figure(figsize=(14, 7))
plt.plot(range(len(y_test_eval)), y_test_eval, label='Datos reales (test)', color='blue',
alpha=0.6)
plt.plot(range(len(y_pred)), y_pred, label='Predicciones', color='red', linestyle='--')
plt.legend()
plt.xlabel('Índice de muestra (test)')
plt.ylabel('Valor')
plt.title(f'Comparación: Test vs. Predicciones {name}')
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()

# =====
# Resultados finales
# =====
results_df = pd.DataFrame(results)
print("\n==== Comparación de Modelos ====")
print(results_df.sort_values(by='MAE'))

# =====
# Ejemplo de acceso a predicciones
# =====
print("\nModelos disponibles en predictions:", list(predictions.keys()))
print("Primeras 5 predicciones del LSTM:")
print(predictions["LSTM"]["y_pred"][:5])
```

#ENTRENAMIENTO DE MLP CON QUARTILES#



```
from sklearn.model_selection import train_test_split

# _____#
# _____#

# Directorio donde se guardarán los modelos
MODELS_DIR_2 = r'C:\\Users\\conta\\Desktop\\Proyecto Predict and
Optimize\\MODELS_MLP_2'
import os
os.makedirs(MODELS_DIR_2, exist_ok=True)
print(f"Directorio de modelos asegurado: {MODELS_DIR_2}")

# _____#
# _____#

y = raw.Incumplimiento.values.astype(np.float32)
X = raw.drop(["ETA", "Scheduled", "Actual", "ON block", "OFF block", "Fecha
Operacion", "hora_actual", "Incumplimiento", 'Tipo_Stand', 'Stand',
'Puertas_de_Embarque'], axis=1).values.astype(np.float32)

# Aplicar el SPLIT y definir las variables globales X_train, y_train, etc.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
import numpy as np

# --- 1. Definición de la Pérdida de Cuantil ---
def quantile_loss(quantile):
    """Retorna la función de pérdida de cuantil para el cuantil (gamma) dado."""
    def loss(y_true, y_pred):
        error = y_true - y_pred
        return tf.reduce_mean(tf.maximum(quantile * error, (quantile - 1) * error))
    return loss

# --- 2. Definición de la Arquitectura de la MLP ---
def create_mlp_model(input_dim, quantile):
    """Crea un modelo MLP con la arquitectura del paper y la pérdida de cuantil."""
    # Los hiperparámetros del paper (ej. Dep.delay): 4 capas, 10 neuronas, 0.0001 LR
    [cite: 331]
    model = Sequential([
        Dense(5, activation='relu', input_shape=(input_dim,)),
        Dense(5, activation='relu'),
        Dense(5, activation='relu'),
        Dense(1, activation='linear')
    ])

    # Usamos la pérdida de cuantil específica para el cuantil
    model.compile(
```



```
optimizer=Adam(learning_rate=0.0001),
loss=quantile_loss(quantile)
)
return model

# --- 3. Entrenamiento (Ejemplo) ---

# Define los cuantiles que necesitas para estimar la distribución.
# Por ejemplo: el cuantil 5%, 50% (mediana) y 95%.
QUANTILES = [0.05, 0.50, 0.95]

# Asegúrate de que X_train y y_train estén disponibles
# X_train = ... (tus características)
# y_train = ... (tu variable Incumplimiento/retraso)

# Simulación de variables de entrada para demostración (REEMPLAZA con tus datos reales)
# Asume que X_train.shape[1] = el número de tus características cíclicas y continuas.
input_dim = X_train.shape[1]

quantile_models = {}
quantile_predictions = {}

print(f"Iniciando entrenamiento de {len(QUANTILES)} modelos MLP...")

for q in QUANTILES:
    print(f"Entrenando modelo para Cuantil  $\gamma = \{q:.2f\}$ ...")

    # Crear y entrenar el modelo específico del cuantil
    model = create_mlp_model(input_dim, q)

    # Usamos más épocas (ej. 100) y un tamaño de lote (batch_size) ajustado
    history = model.fit(
        X_train,
        y_train,
        epochs=100,      # Número de épocas del paper [cite: 279]
        batch_size=32,    # Un batch size más grande puede ser más estable
        verbose=0         # No imprimir la salida de cada época
    )

    # 🚨 PASO CRÍTICO: Guardar el modelo en el disco
    model_filename = os.path.join(MODELS_DIR_2, f'mlp_quantile_{q:.2f}.keras')
    model.save(model_filename)

    # Guardar el modelo
    quantile_models[q] = model

    # Hacer predicciones en el conjunto de prueba
    predictions = model.predict(X_test).reshape(-1)
    quantile_predictions[q] = predictions
```



```
print(f" -> Predicción para Cuantil {q:.2f} completada.")

from scipy.stats import norm

# --- 4. Extracción de la Distribución ( $\mu$  y  $\sigma$ ) ---

# Para obtener la MEDIA ( $\mu$ ) y la DESVIACIÓN ESTÁNDAR ( $\sigma$ ) predichas:
# Si asumes una distribución Normal (como hace el paper [cite: 201]),
# puedes estimar  $\mu$  y  $\sigma$  de los cuantiles predichos.
# Por ejemplo, la mediana (cuantil 0.50) es una buena aproximación de  $\mu$ .
# El rango intercuartílico (Q95 - Q05) puede usarse para aproximar  $\sigma$ .

# 1. Media ( $\mu$ ) predicha: Usa el cuantil 50% (mediana)
media_predicha_mlp = quantile_predictions[0.50]

# 2. Desviación Estándar ( $\sigma$ ) predicha (Aproximación empírica):
# Para una distribución normal, ~90% de los datos están entre Q05 y Q95.
# Esto corresponde a  $\sim 3.29 * \sigma$ .
Q95 = quantile_predictions[0.95]
Q05 = quantile_predictions[0.05]
#  $\sigma \approx (Q95 - Q05) / 3.29$ 
std_dev_predicha_mlp = (Q95 - Q05) / (norm.ppf(0.95) - norm.ppf(0.05)) # Usando
valores exactos

print("\n--- Resultado Final de Predicción de Distribución (MLP) ---")
print(f"Media ( $\mu$ ) Predicha (basada en Q50): {media_predicha_mlp[:5]}")
print(f"Std. Dev. ( $\sigma$ ) Predicha: {std_dev_predicha_mlp[:5]}")

import seaborn as sns
import scipy.stats as stats
import matplotlib.pyplot as plt

# Ajustar parámetros para distribución normal
mean, std = stats.norm.fit(media_predicha_mlp) # Ajusta media y desviación estándar

# Función de densidad de probabilidad (PDF) para normal
def pdf_normal(x):
    return stats.norm.pdf(x, mean, std)

# Gráfico
x = np.linspace(np.min(media_predicha_mlp), np.max(media_predicha_mlp), 1000)
plt.plot(x, pdf_normal(x), 'r-', label='Normal')
sns.histplot(media_predicha_mlp, stat='density', alpha=0.3)
plt.legend()
plt.grid()
plt.title("Gráfico distribución Normal")
```



```
plt.show()

print(f'media: {mean} desviación estandar {std}' )

# Ajustar parámetros

datos_positivos = media_predicha_mlp[media_predicha_mlp > 0]

shape, loc, scale = stats.lognorm.fit(datos_positivos, floc=0) # floc=0 fuerza loc=0

# Función de densidad de probabilidad (PDF)
def pdf_lognormal(x):
    return stats.lognorm.pdf(x, shape, loc, scale)

# Gráfico
x = np.linspace(0, np.max(datos_positivos), 1000)
plt.plot(x, pdf_lognormal(x), 'r-', label='Log-Normal')
sns.histplot(datos_positivos, stat='density', alpha=0.3)
plt.legend()
plt.grid()
plt.title("Gráfico distribución Log-Normal")
plt.show()

print(f'shape: {shape} loc: {loc} scale: {scale}' )

# Ajustar parámetros
alpha, loc, beta = stats.gamma.fit(media_predicha_mlp)
x = np.linspace(np.min(media_predicha_mlp), np.max(media_predicha_mlp), 1000)
# Función de densidad
def pdf_gamma(x):
    return stats.gamma.pdf(x, alpha, loc, beta)

# Gráfico
plt.plot(x, pdf_gamma(x), 'r-', label='Gamma')
sns.histplot(media_predicha_mlp, stat='density', alpha=0.3)
plt.legend()
plt.grid()
plt.title("Gráfico distribución gamma")
plt.show()

print(f'alpha: {alpha} loc: {loc} beta: {beta}' )

# Ajustar parámetros para distribución Weibull
c, loc, scale = stats.weibull_min.fit(media_predicha_mlp, floc=0) # c = parámetro de
forma, floc=0 fija loc=0

# Función de densidad de probabilidad (PDF) para Weibull
def pdf_weibull(x):
    return stats.weibull_min.pdf(x, c, loc, scale)

# Gráfico
```



```
x = np.linspace(0, np.max(media_predicha_mlp), 1000)
plt.plot(x, pdf_weibull(x), 'r-', label='Weibull')
sns.histplot(media_predicha_mlp, stat='density', alpha=0.3)
plt.legend()
plt.title("Gráfico distribución Weibull")
plt.grid()
plt.show()

print(f'c: {c} loc: {loc} scale: {scale}')
```

PREDICCION DATOS DE VALIDACION Mayo 2025

A continuación el modelo MLP_2 será sometido a un mes de prueba de Mayo con datos nunca antes vistos (codigo dice agosto pero es mayo)

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.preprocessing import LabelEncoder as LE
from sklearn.preprocessing import MinMaxScaler
from scipy.stats import norm
import joblib
import os
# Se asume que CODERS_DIR ya está definido (donde se guarda los .pkl)

# =====
# 0. CARGA DE HERRAMIENTAS Y DATOS (Incluyendo Codificadores)
# =====

# Cargar el nuevo itinerario (se mantiene tu código original)
file_path = "C:\\Users\\conta\\Desktop\\operaciones_agosto_2025.xlsx"
datos_25_05_2025 = pd.read_excel(file_path)

# Crear una copia de las columnas de texto originales antes del encoding
datos_25_05_2025_ORIGINAL_STRINGS = datos_25_05_2025.copy()
# Guardar el índice original para la exportación final
index_original_data = datos_25_05_2025.index.copy()

# --- LISTA DE COLUMNAS CÓDIFICADAS ---
columnas_categoricas = ['Aerolinea', 'Pais', 'Aeropuerto', 'Matricula_Aeronave',
                        'Tipo_Stand',
                        'Puertas_de_Embarque', 'NEGOCIO', 'Stand', 'OPERACION',
                        'Numero_de_Vuelo', 'Categoria_Aeronave']

# --- Cargar todos los codificadores entrenados ---
CODERS_DIR = r'C:\\Users\\conta\\Desktop\\Proyecto Predict and
Optimize\\CODERS' # Debe coincidir con el path de guardado
codificadores = {}

print(f'Cargando {len(columnas_categoricas)} LabelEncoders ajustados...")
```



```
for col in columnas_categoricas:
    try:
        filename = f'le_{col}.pkl'
        codificadores[col] = joblib.load(os.path.join(CODERS_DIR, filename))
    except FileNotFoundError:
        print(f'FATAL: No se encontró el codificador ajustado para '{col}'. ¡Deteniendo!')
        raise # Detener el proceso si falta un codificador.

# =====
# 1. PREPROCESAMIENTO IDÉNTICO AL ENTRENAMIENTO
# =====

# --- A. Conversión de Fechas y Nombres de Columnas (se mantiene tu código) ---
# ... (Tu código de renombramiento y manejo de fechas) ...

for col in ["Scheduled", "Actual", "ETA", "ON block", "OFF block"]:
    datos_25_05_2025[col] = pd.to_datetime(datos_25_05_2025[col], errors='coerce')

datos_25_05_2025.rename(columns={'Numero de Vuelo': 'Numero_de_Vuelo',
                                'Tipo Stand': 'Tipo_Stand',
                                'Puertas de Embarque': 'Puertas_de_Embarque',
                                'Cantidad Pasajeros Totales': 'Cantidad_Pasajeros_Totales'},
                        inplace=True)
datos_25_05_2025['Numero_de_Vuelo'] =
datos_25_05_2025['Numero_de_Vuelo'].astype(str)

# --- B. Ingeniería de Características Cíclicas (se mantiene tu código) ---
# ... (Tu código de funciones apply_cyclic_encoding, conversiones a minutos y aplicación
# de codificación cíclica) ...
def convert_hhmm_to_minutes(hhmm_series):
    hhmm_series = hhmm_series.astype('Int64').fillna(0)
    hours = hhmm_series // 100
    minutes = hhmm_series % 100
    return hours * 60 + minutes

def apply_cyclic_encoding(df, col_name, period):
    df[f'{col_name}_sin'] = np.sin(2 * np.pi * df[col_name] / period)
    df[f'{col_name}_cos'] = np.cos(2 * np.pi * df[col_name] / period)
    return df

datos_25_05_2025["año"] = datos_25_05_2025["Scheduled"].dt.year
datos_25_05_2025["trimestre"] = datos_25_05_2025["Scheduled"].dt.quarter
datos_25_05_2025["mes"] = datos_25_05_2025["Scheduled"].dt.month
datos_25_05_2025["dia_mes"] = datos_25_05_2025["Scheduled"].dt.day
datos_25_05_2025["dia_semana"] = datos_25_05_2025["Scheduled"].dt.dayofweek
datos_25_05_2025['hora_scheduled'] = (datos_25_05_2025['Scheduled'].dt.hour * 100 +
datos_25_05_2025['Scheduled'].dt.minute)
datos_25_05_2025['hora_ETA'] = (datos_25_05_2025['ETA'].dt.hour * 100 +
datos_25_05_2025['ETA'].dt.minute)

datos_25_05_2025['minutos_scheduled'] =
convert_hhmm_to_minutes(datos_25_05_2025['hora_scheduled'])
```



```
datos_25_05_2025['minutos_ETA'] =
convert_hhmm_to_minutes(datos_25_05_2025['hora_ETA'])

datos_25_05_2025 = apply_cyclic_encoding(datos_25_05_2025, 'trimestre', 4)
datos_25_05_2025 = apply_cyclic_encoding(datos_25_05_2025, 'mes', 12)
datos_25_05_2025 = apply_cyclic_encoding(datos_25_05_2025, 'dia_semana', 7)
datos_25_05_2025 = apply_cyclic_encoding(datos_25_05_2025, 'dia_mes', 31)
datos_25_05_2025 = apply_cyclic_encoding(datos_25_05_2025, 'minutos_scheduled',
1440)
datos_25_05_2025 = apply_cyclic_encoding(datos_25_05_2025, 'minutos_ETA', 1440)

# --- C. Filtro de Datos y Encoding (USO DE INSTANCIAS CARGADAS) ---

# Aplicar el mismo filtro y la misma limpieza de NaNs que en el entrenamiento
columnas_para_dropna = ['p0', 'tMax12Horas', 'dd02Minutos', 'ff02Minutos',
'vis2Minutos', 'minutos_ETA']
datos_25_05_2025 = datos_25_05_2025.dropna(subset=columnas_para_dropna)
datos_25_05_2025.reset_index(drop=True, inplace=True)

# 🚨 TRANSFORMACIÓN ROBUSTA DE CADA COLUMNA CATEGÓRICA
for col in columnas_categoricas:

    # 1. Limpieza y Consistencia de Tipo (IDÉNTICA AL ENTRENAMIENTO)
    col_data_cleaned = datos_25_05_2025[col].astype(str)

    # Obtener la instancia de codificador correcta para esta columna
    le_instance = codificadores[col]

    try:
        # 2. Transformar usando la INSTANCIA ESPECÍFICA
        datos_25_05_2025[col] = le_instance.transform(col_data_cleaned)

    except ValueError as e:
        # 3. Manejo de Fallback si la etiqueta es desconocida
        print(f"\n🔴 Error de transformación en columna: {col}. Aplicando fallback.")

        etiquetas_desconocidas = set(col_data_cleaned.unique()) - set(le_instance.classes_)

        # Estrategia: Reemplazar las desconocidas con la primera clase conocida (código 0)
        temp_series = col_data_cleaned.copy()
        temp_series[temp_series.isin(etiquetas_desconocidas)] = le_instance.classes_[0]

        # 4. Transformar la serie corregida
        datos_25_05_2025[col] = le_instance.transform(temp_series)

# --- D. Normalización (Scaling) ---
# ... (Tu código de Normalización) ...
features_a_normalizar = [
    'Cantidad_Pasajeros_Totales', 'p0', 'tMax12Horas', 'dd02Minutos',
    'ff02Minutos', 'vis2Minutos', 'Distancia Recorrida'
]
```



```
features_a_normalizar = [col for col in features_a_normalizar if col in
datos_25_05_2025.columns]

# Asumiendo que 'scaler' fue ajustado en el entrenamiento:
datos_25_05_2025[features_a_normalizar] =
scaler.transform(datos_25_05_2025[features_a_normalizar])

# --- E. Preparar X para la Predicción ---
columnas_a_excluir = ["ETA", "Scheduled", "Actual", "ON block", "OFF block", "Fecha
Operacion", "hora_actual", "Incumplimiento", "Tipo_Stand", "Stand",
'Puertas_de_Embarque', 'hora_scheduled',
' hora_ETA']
columnas_a_excluir = [col for col in columnas_a_excluir if col in
datos_25_05_2025.columns]

X_25_05_2025 = datos_25_05_2025.drop(columnas_a_excluir,
axis=1).values.astype(np.float32)

#Condiciones de Incumplimiento

print(f'\n✅ Preprocesamiento del nuevo itinerario completado. Dimensiones de X:
{X_25_05_2025.shape}')

import tensorflow as tf
from tensorflow.keras.models import load_model
import os
from typing import Dict, Any

# --- FUNCIÓN DE PÉRDIDA PERSONALIZADA (CRÍTICA) ---
# Debe coincidir exactamente con la definición usada en el entrenamiento.
def quantile_loss(quantile: float):
    # Asegúrate de usar 'tf.constant' o 'tf.cast' si es necesario
    q = tf.constant(quantile, dtype=tf.float32)
    def loss(y_true, y_pred):
        e = y_true - y_pred
        return tf.keras.backend.mean(tf.maximum(q * e, (q - 1) * e), axis=-1)
    return loss

# LLAMAR AL ARCHIVO DE MODELS_MLP_2#

# --- Configuración de la Ruta ---
MODELS_DIR = r'C:\\Users\\conta\\Desktop\\Proyecto Predict and
Optimize\\MODELS_MLP_2'
Q_TARGET = 0.50 # El cuantil que deseas cargar (ej. la mediana)
QUANTILES = [0.05, 0.50, 0.95]
quantile_models: Dict[float, tf.keras.Model] = {}
quantile_predictions = {}

# 1. Construir el nombre del archivo
model_filename = os.path.join(MODELS_DIR,
f'mlp_quantile_{Q_TARGET:.2f}.keras')
```



2. Cargar el modelo con la función de pérdida

for q in QUANTILES:

```
    model_filename = os.path.join(MODELS_DIR, f'mlp_quantile_{q:.2f}.keras')
```

```
    try:
```

```
        model = load_model(
            model_filename,
            custom_objects={'loss': quantile_loss(q)}
        )
```

```
        quantile_models[q] = model
```

```
        print(f"✅ Modelo Cuantil  $\gamma$ ={q:.2f} cargado exitosamente.")
```

```
    except Exception as e:
```

```
        # 🚨 ¡IMPORTANTE! Hemos quitado el 'raise'.
```

```
        print(f"❌ ERROR CRÍTICO al cargar el modelo {model_filename}. Causa: {e}")
```

```
        # El modelo no se añade a quantile_models[q] y el programa continúa.
```

2. PREDICCIÓN CON MLP QUANTILE ENSEMBLE

=====

Se asume que 'quantile_models' (diccionario de modelos {0.05: model, 0.50: model, ...}) existe.

```
print(f"Iniciando predicción con {len(QUANTILES)} modelos MLP...")
```

for q in QUANTILES:

```
    modelo_cuantil_cargado = quantile_models[q]
```

```
    predictions = modelo_cuantil_cargado.predict(X_25_05_2025).reshape(-1)
```

```
    quantile_predictions[q] = predictions
```

```
    print(f" -> Predicción para Cuantil  $\gamma$ ={q:.2f} completada.")
```

=====

3. EXTRACCIÓN DE LA DISTRIBUCIÓN FINAL (μ y σ)

=====

```
Q95 = quantile_predictions[0.95]
```

```
Q50 = quantile_predictions[0.50]
```

```
Q05 = quantile_predictions[0.05]
```

Media (μ) predicha: Usa el cuantil 50%

```
media_predicha_mlp = Q50
```

Desviación Estándar (σ) predicha: Estimación del rango 90%

```
Z_FACTOR_90_PERCENT = norm.ppf(0.95) - norm.ppf(0.05)
```

```
std_dev_predicha_mlp = (Q95 - Q05) / Z_FACTOR_90_PERCENT
```

```
varianza_predicha_mlp = std_dev_predicha_mlp ** 2
```

```
print("\n✅ Parámetros de distribución predichos para el nuevo itinerario.")
```



```
print(f" Media predicha ( $\mu$ ) (primeras 5 muestras): {media_predicha_mlp[:5]}")
```

```
import pandas as pd
import numpy as np
import os # Importar os para manejo de rutas
```

```
# --- ASUNCIONES CLAVE ---
```

```
# 1. Las variables 'media_predicha_mlp', 'varianza_predicha_mlp',
# 'std_dev_predicha_mlp' ya existen y contienen las predicciones.
# 2. El DataFrame 'datos_25_05_2025_ORIGINAL_STRINGS' es el clon legible
# del itinerario cargado originalmente.
# 3. El DataFrame 'datos_25_05_2025' es el DataFrame FILTRADO Y PROCESADO.
# Su índice (datos_25_05_2025.index) contiene las filas que fueron modeladas.
# -----
```

```
# =====
# 3. CONSOLIDACIÓN Y EXPORTACIÓN CON MEDIA Y VARIANZA DE MLP
# =====
```

```
# --- A. Preparar la Matriz de Salida (Media y Varianza MLP) ---
```

```
# Crear un DataFrame con los parámetros predichos usando los arrays del MLP
df_parametros_predichos = pd.DataFrame({
    'MLP_Media_Predicha_min': media_predicha_mlp,      #  $\mu_p$  de la MLP (basada
en Q50)
    'MLP_Varianza_Predicha_min2': varianza_predicha_mlp,  #  $\sigma^2_p$  de la MLP
    'MLP_StdDev_Predicha_min': std_dev_predicha_mlp      #  $\sigma_p$  de la MLP
})
```

```
# --- B. Crear el DataFrame Base Legible (Usando el CLON ORIGINAL) ---
```

```
# 1. Definir Columnas Categóricas y de Valor Original (Strings)
```

```
COLS_BASE_LEGIBLE = [
    'Scheduled', 'ON block', 'OFF block', 'OPERACION', 'Aerolinea', 'Pais', 'Aeropuerto',
    'Matricula_Aeronave',
    'Tipo_Stand', 'Puertas_de_Embarque', 'NEGOCIO', 'Stand',
    'Numero_de_Vuelo', 'Categoria_Aeronave', 'Cantidad_Pasajeros_Totales',
    'Distancia_Recorrida', 'Scheduled', 'Incumplimiento'
]
```

```
# 2. OBTENER DATOS LEGIBLES Y ASEGURAR ÍNDICE SECUENCIAL:
```

```
# El uso de .loc[datos_25_05_2025.index] asegura que solo se incluyan las filas
# que pasaron el filtro de NaNs y el filtro de Incumplimiento.
df_salida_base =
datos_25_05_2025_ORIGINAL_STRINGS.loc[datos_25_05_2025.index,
COLS_BASE_LEGIBLE].copy()
```

```
# 3. Forzamos el índice a ser secuencial (0, 1, 2...) para que la concatenación funcione
correctamente.
```

```
df_salida_base.reset_index(drop=True, inplace=True)
df_parametros_predichos.reset_index(drop=True, inplace=True)
```



```
# --- C. Fusionar y Exportar ---

# 4. Fusionar por posición (concatenar en el eje 1)
df_salida_final = pd.concat([
    df_salida_base,
    df_parametros_predichos
], axis=1)

# 1. Definir el directorio de destino
directorio_destino = r'C:\\Users\\conta\\Desktop\\Proyecto Predict and Optimize'

# 2. Definir el nombre del archivo (Añadimos 'MLP' para distinguirlo)
nombre_archivo_salida = 'itinerario_prediccion_distribucion_agosto.xlsx'

# 3. Combinar el directorio y el nombre del archivo para obtener la ruta completa
ruta_completa = os.path.join(directorio_destino, nombre_archivo_salida)

# 4. Exportar usando la ruta completa
df_salida_final.to_excel(ruta_completa, index=False)

print(f'\n✅ PROCESO COMPLETADO. Itinerario con Distribución (MLP)
exportado a: {ruta_completa}')
print(f'El archivo contiene la media ({df_parametros_predichos.columns[0]}) y la
varianza ({df_parametros_predichos.columns[1]}) predichas por el modelo MLP.')

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np # Necesario para cálculos si los haces después

# Cargar el archivo
file_path = 'C:\\Users\\conta\\Desktop\\Proyecto Predict and
Optimize\\itinerario_prediccion_distribucion_agosto.xlsx'
residual = pd.read_excel(file_path)

# 1. Calcular los residuos
residuos = residual["Incumplimiento"] - residual["MLP_Media_Predicha_min"]

# 2. Definir los límites y el número de barras (bins)
# Límite: [mínimo, máximo] -> [-20, 20]
# Más columnas (bins): 50 barras en lugar de las 10 predeterminadas
LIMITE_MIN = -20
LIMITE_MAX = 20
NUM_BINS = 50

# 3. Generar el Histograma
plt.figure(figsize=(10, 6)) # Define el tamaño del gráfico
plt.hist(
    residuos,
    bins=NUM_BINS,
```



```
range=(LIMITE_MIN, LIMITE_MAX), # Limita el eje X entre -20 y 20
edgecolor='black', # Dibuja bordes negros para separar las barras
alpha=0.7 # Transparencia
)

# 4. Añadir Títulos y Etiquetas (para una mejor visualización)
plt.title('Distribución de Residuos (Incumplimiento - Media MLP)')
plt.xlabel('Residuos (Minutos)')
plt.ylabel('Frecuencia')
plt.grid(axis='y', alpha=0.5) # Añade una cuadrícula suave
plt.axvline(0, color='red', linestyle='dashed', linewidth=1) # Línea en el cero

plt.show()

media_residuos = residuos.mean()
varianza_residuos = residuos.var()

print(f"Media de los residuos: {media_residuos:.4f}")
print(f"Varianza de los residuos: {varianza_residuos:.4f}")
```



Anexo F

Código en Python del modelo de Machine Learning MLP con pérdida de cuantiles

MODELO DE RED NEURONAL PARA LA PREDICCIÓN DE INCUMPLIMIENTO

```
from sklearn.model_selection import train_test_split

# _____#
# _____#

# Directorio donde se guardarán los modelos
MODELS_DIR = r'C:\\Users\\conta\\Desktop\\Proyecto Predict and
Optimize\\MODELS_MLP_2'
import os
os.makedirs(MODELS_DIR, exist_ok=True)
print(f"Directorio de modelos asegurado: {MODELS_DIR}")

# _____#
# _____#

y = raw.Incumplimiento.values.astype(np.float32)
X = raw.drop(["ETA", "Scheduled", "Actual", "ON block", "OFF block", "Fecha
Operacion", "hora_scheduled", "hora_actual", "Incumplimiento", "Tipo_Stand", 'Stand',
'Puertas_de_Embarque'], axis=1).values.astype(np.float32)

# Aplicar el SPLIT y definir las variables globales X_train, y_train, etc.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
import numpy as np

# --- 1. Definición de la Pérdida de Cuantil ---
def quantile_loss(quantile):
    """Retorna la función de pérdida de cuantil para el cuantil (gamma) dado."""
    def loss(y_true, y_pred):
        error = y_true - y_pred
        return tf.reduce_mean(tf.maximum(quantile * error, (quantile - 1) * error))
    return loss

# --- 2. Definición de la Arquitectura de la MLP ---
def create_mlp_model(input_dim, quantile):
    """Crea un modelo MLP con la arquitectura del paper y la pérdida de cuantil."""
    # Los hiperparámetros del paper (ej. Dep.delay): 4 capas, 10 neuronas, 0.0001 LR [cite:
331]
    model = Sequential([
```



```
Dense(10, activation='relu', input_shape=(input_dim,)),
Dense(10, activation='relu'),
Dense(10, activation='relu'),
Dense(1, activation='linear')
])

# Usamos la pérdida de cuantil específica para el cuantil
model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss=quantile_loss(quantile)
)
return model

# --- 3. Entrenamiento (Ejemplo) ---

# Define los cuantiles que necesitas para estimar la distribución.
# Por ejemplo: el cuantil 5%, 50% (mediana) y 95%.
QUANTILES = [0.05, 0.50, 0.95]

# Asegúrate de que X_train y y_train estén disponibles
# X_train = ... (tus características)
# y_train = ... (tu variable Incumplimiento/retraso)

# Simulación de variables de entrada para demostración (REEMPLAZA con tus datos reales)
# Asume que X_train.shape[1] = el número de tus características cíclicas y continuas.
input_dim = X_train.shape[1]

quantile_models = {}
quantile_predictions = {}

print(f"Iniciando entrenamiento de {len(QUANTILES)} modelos MLP...")

for q in QUANTILES:
    print(f"Entrenando modelo para Cuantil  $\gamma = \{q:.2f\}$ ...")

    # Crear y entrenar el modelo específico del cuantil
    model = create_mlp_model(input_dim, q)

    # Usamos más épocas (ej. 100) y un tamaño de lote (batch_size) ajustado
    history = model.fit(
        X_train,
        y_train,
        epochs=100,      # Número de épocas del paper [cite: 279]
        batch_size=32,    # Un batch size más grande puede ser más estable
        verbose=0         # No imprimir la salida de cada época
    )

    # 🚨 PASO CRÍTICO: Guardar el modelo en el disco
    model_filename = os.path.join(MODELS_DIR, f'mlp_quantile_{q:.2f}.keras')
    model.save(model_filename)
```



```
# Guardar el modelo
quantile_models[q] = model

# Hacer predicciones en el conjunto de prueba
predictions = model.predict(X_test).reshape(-1)
quantile_predictions[q] = predictions

print(f" -> Predicción para Cuantil {q:.2f} completada.")

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from math import sqrt

# =====
# 4. EVALUACIÓN DE DESEMPEÑO (Usando Q0.50 como Predicción Central)
# =====

# El cuantil 0.50 (Mediana) se usa como la mejor estimación de la tendencia central
# (promedio).
predictions_q50 = quantile_predictions[0.50]
y_reales = y_test

# 1. Calcular el Error Cuadrático Medio (MSE)
mse = mean_squared_error(y_reales, predictions_q50)

# 2. Calcular la Raíz del Error Cuadrático Medio (RMSE)
# Mide el error promedio, en las mismas unidades que 'Incumplimiento'.
rmse = sqrt(mse)

# 3. Calcular el Error Absoluto Medio (MAE)
# Es menos sensible a outliers que el RMSE.
mae = mean_absolute_error(y_reales, predictions_q50)

# 4. Calcular el Coeficiente de Determinación (R2)
# Mide qué tan bien se ajustan las predicciones a los valores reales.
r2 = r2_score(y_reales, predictions_q50)

print("\n" + "=" * 65)
print("--- Resultados de Evaluación (Basados en Q0.50 - Mediana) ---")
print(f"Número de muestras de prueba (X_test): {len(y_reales)}")
print("-" * 65)
print(f"✅ Raíz del Error Cuadrático Medio (RMSE): {rmse:.4f}")
print(f"✅ Error Absoluto Medio (MAE): {mae:.4f}")
print(f"✅ Coeficiente de Determinación (R2): {r2:.4f}")
print("=" * 65)
```