Predicción de satisfacción de clientes
Informe Técnico — Proyecto de Clasificación de Satisfacción de Clientes Aéreos

You & Airvryone

Yeder Pimentel
Alfonso Bermúdez
Teo Ramos Ruano
Maribel Gutiérrez Ramírez

Introducción

Dentro del Bootcamp de Inteligencia Artificial, se nos planteó el desafío de desarrollar una solución capaz de realizar predicciones a partir de un conjunto de datos reales. La consigna principal consistía en abordar un problema de clasificación supervisada, aplicando técnicas de análisis de datos, modelado y despliegue de modelos de *machine learning*.

Nuestro equipo decidió trabajar con un dataset que refleja la satisfacción de los clientes de una aerolínea, el cual reúne múltiples factores vinculados al servicio y a la experiencia del pasajero. Entre las variables disponibles se encuentran características como el tipo de viaje, la clase del boleto, el confort del asiento, la limpieza, la atención del personal y el proceso de embarque en línea, entre otras.

A partir de este conjunto de datos, buscamos entrenar un modelo de clasificación que permita anticipar si un cliente se encontrará satisfecho o insatisfecho con su experiencia. Este tipo de predicción puede resultar de gran utilidad para las empresas del sector aeronáutico, ya que ofrece la posibilidad de detectar puntos críticos en el servicio y orientar estrategias de mejora basadas en evidencia.

El desarrollo del proyecto se estructuró en distintas fases: en primer lugar, se realizó un análisis exploratorio de los datos (EDA) para comprender la distribución, correlación y relevancia de las variables. Posteriormente, se llevó a cabo el procesamiento y limpieza de los datos, seguido de la selección y entrenamiento de modelos de clasificación, evaluando su rendimiento mediante métricas como *accuracy*, *precision*, *recall* y *F1-score*.

Finalmente, se integró el modelo en una aplicación web interactiva desarrollada con FastAPI en el backend y React con TailwindCSS en el frontend, tomando como referencia un diseño previo elaborado en Figma. Esta interfaz permite a los usuarios ingresar los datos de un cliente y obtener en tiempo real una predicción sobre su nivel de satisfacción.

En conjunto, el proyecto no solo persigue el objetivo técnico de construir un modelo de clasificación preciso, sino también el propósito práctico de productivizar una solución de inteligencia artificial que pueda ser utilizada por usuarios no técnicos en un entorno web intuitivo y accesible.

2. Análisis Exploratorio del Conjunto de Datos (EDA)

2.1 Descripción general

El conjunto de datos utilizado corresponde al Airline Passenger Satisfaction Dataset, disponible públicamente en Kaggle. Este dataset contiene **103.903** registros y **23** variables (inicialmente 25, 2 eliminadas porque eran irrelevantes), entre numéricas y categóricas, que describen distintos aspectos del servicio brindado por una aerolínea y las características de sus pasajeros.

La variable objetivo es "***satisfaction***", la cual indica si un cliente se encuentra "***Satisfied***" o "***Neutral or Dissatisfied***".

2.2 Variables principales

Algunas de las variables más relevantes del dataset incluyen:

- ***Gender*** (género del pasajero)
- ***Customer Type*** (cliente frecuente o no -Loyal/Disloyal-)
- ***Age*** (edad del pasajero)
- ***Type of Travel*** (viaje personal o de negocios)
- ***Class*** (clase del boleto: Economy, Business, First)
- ***Seat Comfort***, ***Inflight Service***, ***Cleanliness***, ***Online Boarding***, entre otras variables de servicio calificadas en escala de 1 a 5.

2.3 Limpieza de datos

Durante el proceso de exploración se identificaron:

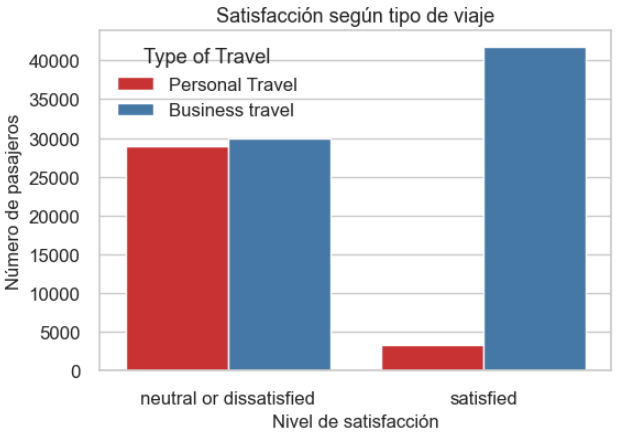
- Columnas irrelevantes ("Unnamed: 0", "id")
- Valores nulos en un 3,3% de las observaciones (imputados con la mediana o la moda según el tipo de variable). Mediana en la variable "***Departure/Arrival time convenient***".
- Valores atípicos.
- Ausencia de valores duplicados.
- Variables categóricas codificadas con OneHotEncoder para su posterior uso en el modelo.
- Transformación de la variable objetivo a valores binarios (1 = Satisfied, 0 = Unsatisfied).

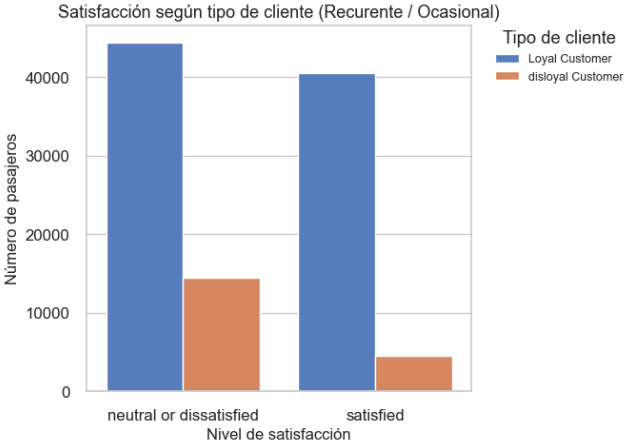
2.4 Visualizaciones

Se realizaron distintas visualizaciones exploratorias:

- Histogramas para analizar la distribución de la edad y las valoraciones.
- Boxplots que mostraron una clara diferencia en las puntuaciones de "**Inflight Service**" entre clientes satisfechos e insatisfechos.
- Matriz de correlación, donde se evidenció que las variables de servicio están fuertemente relacionadas con la satisfacción.

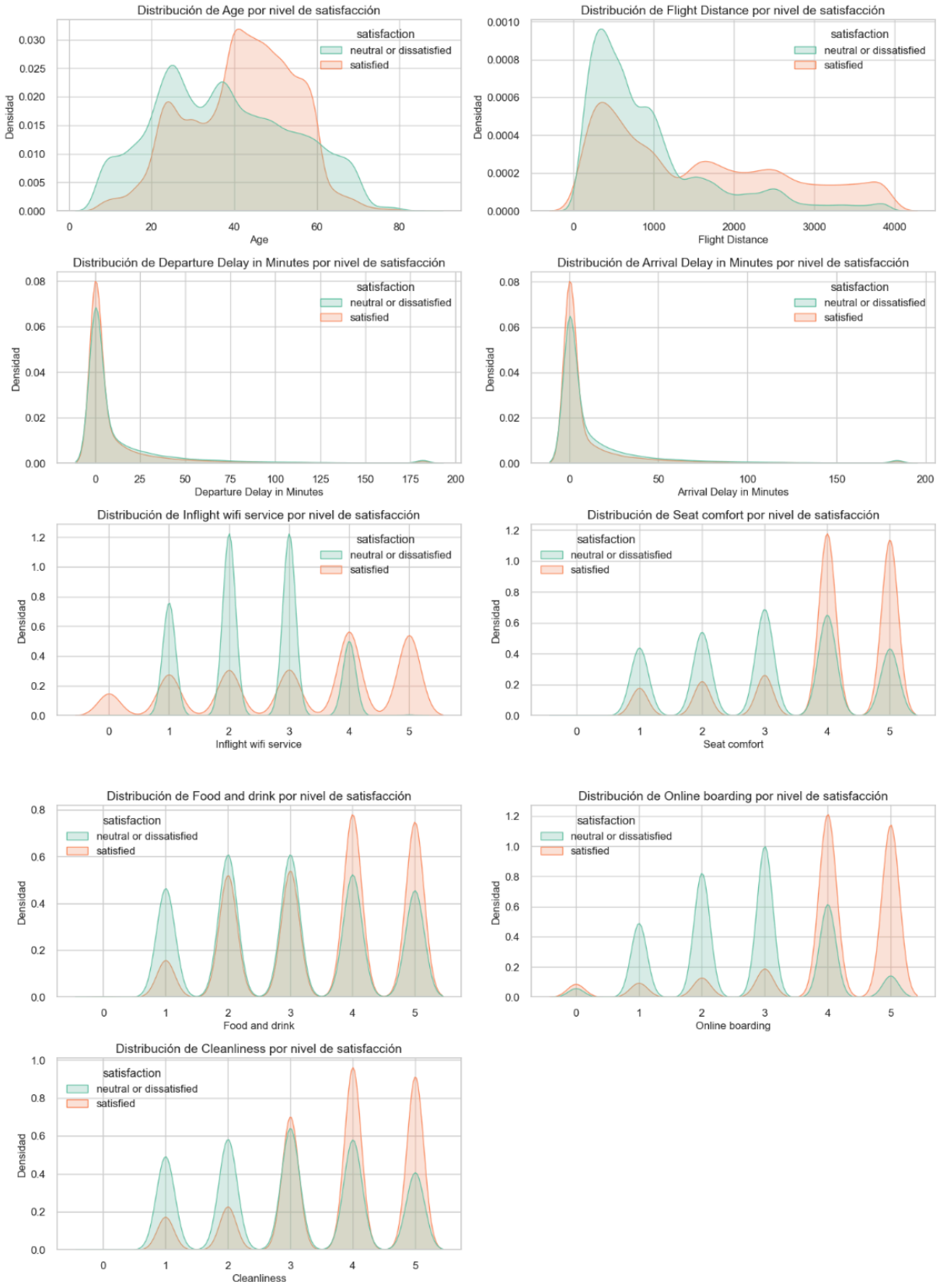
El análisis permitió confirmar que las características relacionadas al confort y la calidad del servicio influyen significativamente en el nivel de satisfacción del pasajero.





Interpretación visual:

- En el primer gráfico (**Tipo de viaje**), se observa que los pasajeros de viajes de negocios (**Business Travel**) presentan una mayor proporción de satisfacción que los de viajes personales (**Personal Travel**).
  Esto podría indicar que la aerolínea ofrece un mejor servicio o más beneficios en viajes de negocios.
- En el segundo gráfico (**Tipo** de cliente), los clientes recurrentes (**Loyal Customers**) muestran una satisfacción significativamente mayor frente a los nuevos clientes.
  Este patrón sugiere que la fidelización y la experiencia previa influyen positivamente en la percepción del servicio.
- Estos hallazgos son importantes porque permiten segmentar estrategias de mejora y entender qué tipo de pasajero contribuye más al nivel global de satisfacción.

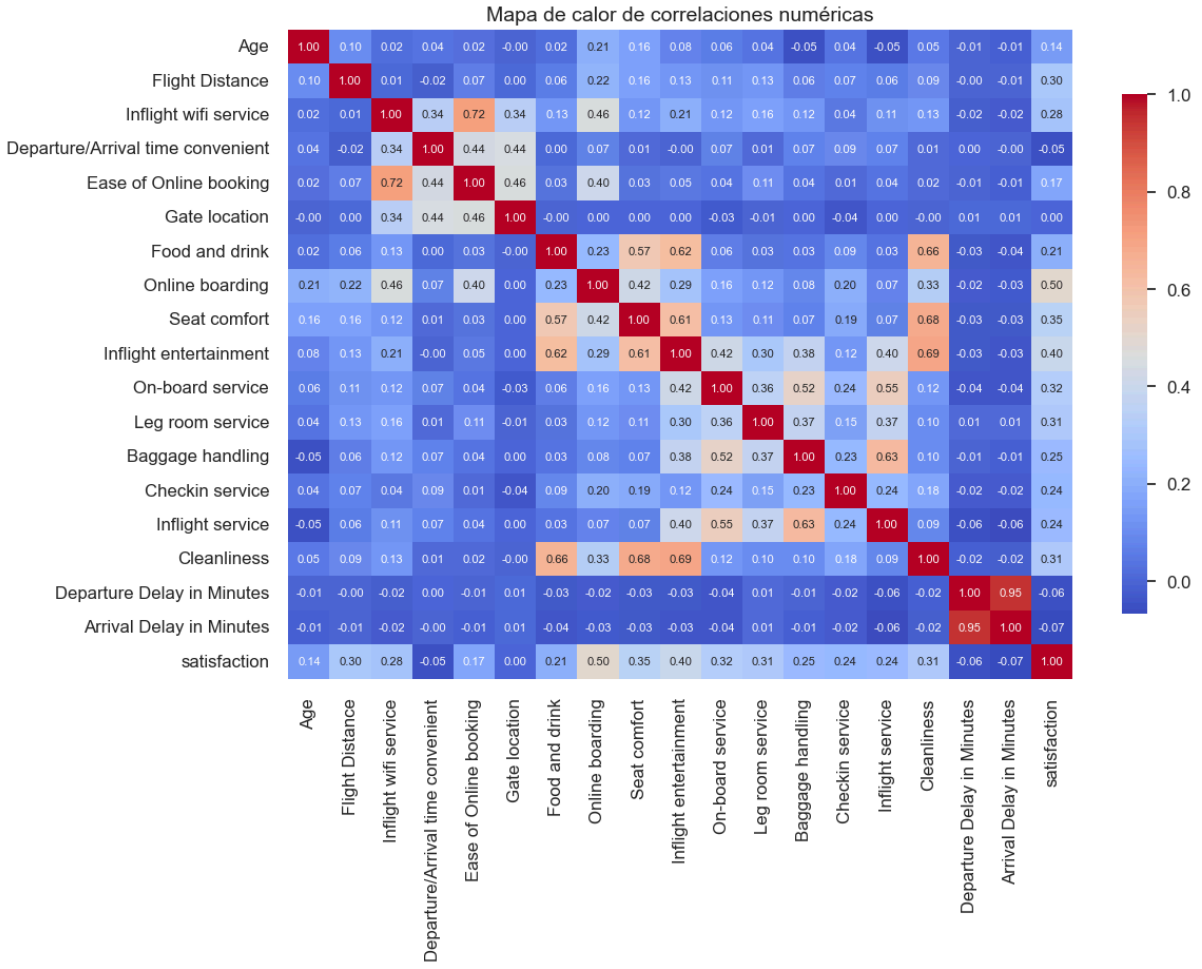


Interpretación visual:

- Las curvas muestran diferencias notables entre las clases en varias variables.
  Por ejemplo:

- En **Online boarding**, los pasajeros satisfechos tienden a concentrarse en valores altos, lo que sugiere que la eficiencia en el embarque influye positivamente en la satisfacción.
- En **Flight Distance**, ambas clases presentan distribuciones similares,
  por lo que esta variable probablemente no sea muy discriminante.
- En **Departure y Arrival Delay**, los pasajeros insatisfechos se concentran más en retrasos altos.

- En general, estas observaciones permiten identificar las variables más relevantes que podrían tener un mayor impacto en los modelos de clasificación.



Interpretación visual:

- Variables como ***Online boarding***, ***Inflight entertainment***, ***Seat comfort*** y ***Cleanliness*** muestran correlaciones positivas altas con ***satisfaction***, lo que indica que influyen fuertemente en la percepción del pasajero.

- Por el contrario, **Departure Delay** y **Arrival Delay** presentan correlaciones negativas, sugiriendo que los retrasos reducen significativamente la satisfacción.
- Este patrón es coherente con la lógica de negocio:
  una mejor experiencia en vuelo y puntualidad se traducen en clientes más satisfechos.

Estas correlaciones nos han guiado para la selección de características en los modelos predictivos, permitiendo centrarnos en las variables más relevantes y evitar redundancias.

3. Preprocesamiento de los Datos

Para garantizar un entrenamiento eficiente y un modelo robusto, se aplicaron las siguientes técnicas de preprocesamiento:

- **Separación de variables predictoras y objetivo**:

  La variable objetivo es **`***satisfaction***`**, con dos clases:

  - `"***satisfied***"`
  - `"***neutral or dissatisfied***"`

  En este paso se separa el dataset en:
  - **`***X***`** → ***Variables predictoras*** (todas las características usadas para predecir).
  - **`***y***`** → ***Variable objetivo*** (*target*), que en este caso es la columna `satisfaction`.

  Esta separación es esencial para el entrenamiento supervisado de los modelos.

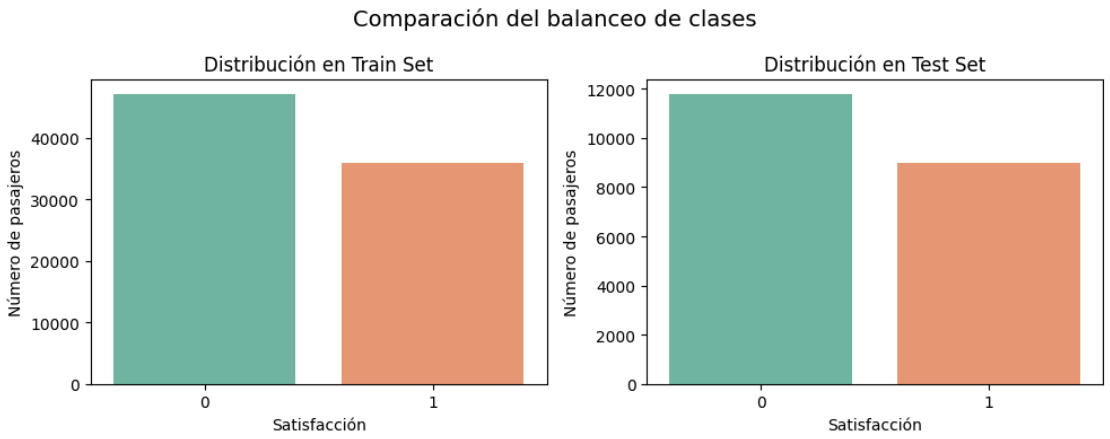
- ***Winsorización*** **de valores atípicos o extremos** (***outliers***) en las variables ***Departure Delay in Minutes***, ***Arrival Delay in Minutes*** y ***Flight Distance*** , en lugar de usar el método IQR de eliminación.

- **Normalización de la variable objetivo**:
  - La variable ***satisfaction*** es convertida a formato binario (0 / 1).
  - Esto permite que todos los modelos de clasificación trabajen de forma coherente.

    - Facilita el cálculo de métricas como ***ROC-AU***C, ***F1-score*** y ***Precision/Recall***.
    - Los valores ahora representan:
        - 0: Clientes neutros o insatisfechos.
        - 1: Clientes satisfechos.

- **Codificación de variables categóricas**: mediante ***OneHotEncoder***:
    - Para utilizar el dataset en modelos supervisados de Machine Learning, las variables categóricas deben transformarse en formato numérico.
    - Por ello, convertimos las variables categóricas (`Gender`, `Customer Type`, `Type of Travel`, `Class`) en variables binarias mediante *One-Hot Encoding mediante pd.get_dummies(), creando una nueva columna (0/1) por cada categoría posible, pero eliminando una por variable (drop_first=True) para evitar multicolinealidad, reduciendo la redundancia (evita el problema de la "trampa de las variables ficticias").
    - El aumento en el número de columnas indica cuántas nuevas categorías se generaron tras la codificación.
    - Este paso permite que modelos basados en valores numéricos (como Regresión Logística, SVM, o KNN) puedan interpretar correctamente estas variables.

- **División del dataset en entrenamiento y prueba**:
    - Dividimos el dataset en subconjuntos:
      - 80% de datos para entrenamiento (train) → para ajustar los modelos.
      - 20% prueba (test) → para evaluar el rendimiento.
    - La división mantiene la proporción de clases mediante `stratify=y` para evitar que el modelo aprenda con un conjunto desbalanceado.



- **Escalado de variables numéricas**: utilizando ***StandardScaler*** para mantener una escala homogénea.

  El escalado homogeniza las magnitudes de las variables numéricas continuas, **garantiza** que **todas las características tengan la misma escala** y ninguna domine el proceso de aprendizaje del modelo.

  Se aplica ***StandardScaler*** de `***scikit-learn***`, que transforma los valores para que tengan o se centren a la media en 0 y desviación estándar a 1.

  - Los **algoritmos basados en distancia o gradiente** (***SVM***, ***KNN***, ***Regresión Logística***) **necesitan** este paso de **escalado**.

  - Los **modelos basados en árboles** (***Random Forest***, ***XGBoos***t, ***LightGBM***) **no requieren escalado**.

  Por ello, se han generado dos versiones del dataset: escalada y no escalada que usaremos dependiendo del algoritmo analizado indistintamente.

  Solo se escalan las **variables numéricas continuas**, mientras que las variables categóricas codificadas (dummies) permanecen sin escalar.

  Consideraciones:

  - El escalado se ajusta solo con los datos de entrenamiento (***fit_transform***) y se aplica al conjunto de prueba (transform) para evitar fugas de información (data leakage).

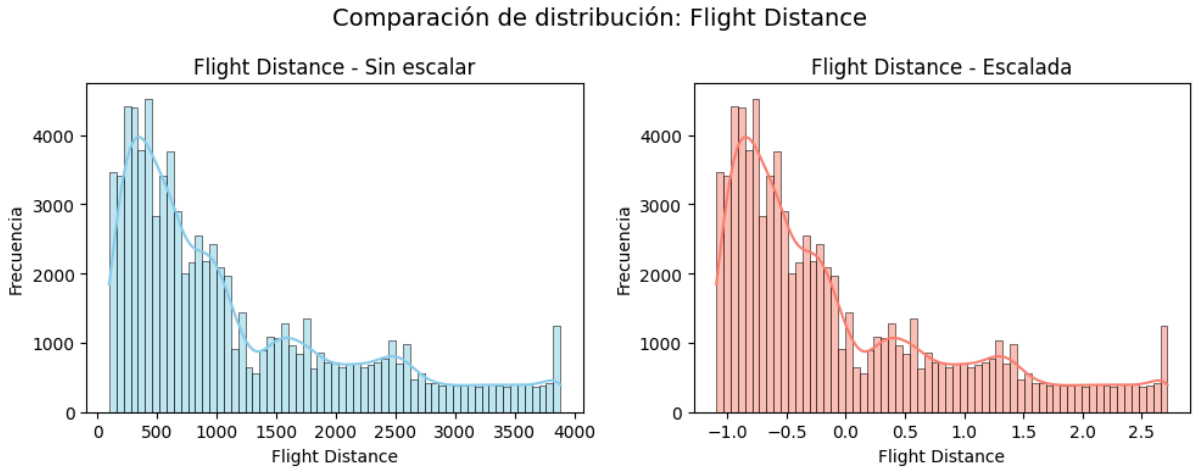
- Las variables numéricas quedan ahora en una escala comparable.

Comprobación visual del escalado:

Visualizamos la distribución de una variable continua, antes y después del escalado, para confirmar que el `StandardScaler` ha centrado los valores en torno a 0.

El escalado de variables numéricas es fundamental para garantizar que todos los atributos tengan el mismo rango de valores y **no dominen el proceso de aprendizaje** de los modelos.

Aquí se muestra la diferencia en la distribución de una variable continua (*Flight Distance*) antes y después del escalado.



Interpretación visual:

- El gráfico izquierdo muestra la distribución original de la variable Flight Distance, donde los valores pueden tener una gran dispersión (por ejemplo, vuelos cortos y largos).
- El gráfico derecho representa la misma variable tras aplicar escalado (StandardScaler), centrando los valores en torno a la media (0) y ajustando la desviación estándar (1).
- Aunque la forma de la distribución (KDE) se mantiene, el rango cambia: ahora todos los valores están normalizados, lo que permite que los modelos basados en distancia (como KNN o SVM) funcionen correctamente sin que una variable domine sobre las demás.

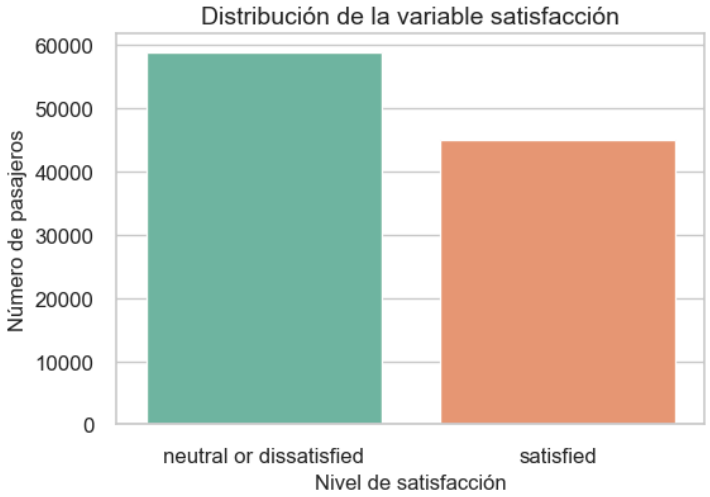
- **Exportación de datasets para modelado:** Se generan dos versiones del conjunto de datos:

- **Escalado**: para **algoritmos sensibles a la magnitud de las variables** (**SVM**, **KNN**, **Regresión Logística**).
- **No escalado**: para **modelos basados en árboles** (**Decision Tree**, **Random Forest**, **XGBoost**, **LightGBM**).

Estos datasets se encuentran exportados en la carpeta `***../data/processed***/` y listos para el modelado en los siguientes notebooks:

`***03_<modelo>.ipynb***` — Entrenamiento, validación cruzada y tuning de hiperparámetros.

- **Manejo del desbalanceo**: se aplicó ***class_weight='balanced'*** para compensar el desbalanceo del dataset, ya que la clase de clientes insatisfechos era ligeramente menor.



Interpretación visual:

- Se observa que la clase "***neutral or dissatisfied***" tiene una proporción ligeramente mayor que la clase "**satisfied**", lo que indica un **desbalanceo moderado en el dataset**.
- Este desequilibrio debe considerarse durante el entrenamiento para evitar que el modelo favorezca la clase mayoritaria.
- En fases posteriores, se puede mitigar este efecto mediante técnicas como:
  - Ajuste de pesos (***class_weight='balanced'*** en modelos lineales). **Este el método que hemos usado**.
  - Submuestreo o sobremuestreo de clases.

- **Evaluación con métricas adecuadas** (**F1-score**, **ROC-AU**C) **en lugar de solo accuracy**.

4. Modelado y Entrenamiento

Durante la fase de modelado, se probaron diferentes algoritmos de clasificación, cada uno en un notebook independiente, con el fin de encontrar el que mejor balanceara rendimiento y generalización.

Cada modelo se entrena en tres fases: ***baseline***, ***GridSearchCV*** y ***Optuna tuning***.

- **Modelos analizados (notebooks 03_<algotirmo>.ipynb):**
  - **Regresión Logística**
  - **K-Nearest Neighbors (KNN)**
  - **Árbol de Decisión**
  - **Random Forest**
  - **XGBoost**

- **Validación cruzada (Cross-Validation):** Aplicamos validación cruzada (K-folds, en nuestro caso **5-folds**) sobre el modelo baseline para **estimar su capacidad de generalizació**n **y verificar la estabilidad del rendimiento antes del ajuste** de **hiperparámetros**.

  La validación cruzada permite evaluar el modelo en diferentes subconjuntos del conjunto de entrenamiento, detectando posibles problemas de sobreajuste (overfitting) o inestabilidad del modelo.

- **Baseline**: Este modelo baseline nos servirá como referencia inicial antes de aplicar técnicas de optimización de hiperparámetros (GridSearchCV y Optuna), permitiendo comparar si las mejoras obtenidas justifican el coste computacional adicional. Aquí se fijan los valores por defecto sin ajustar de cada modelo.

5. Comparación de modelos

Comparación de modelos (notebook 04_Comparison_Modelo.ipynb), con métricas estandarizadas.

- **Métricas de Evaluación:** Dado que se trata de un problema de clasificación binaria, se emplean las siguientes métricas:
    - Accuracy: proporción de predicciones correctas.
    - ***Precision***: proporción de verdaderos positivos entre las predicciones positivas (capacidad de evitar falsos positivos).
    - ***Recall***: proporción de verdaderos positivos entre los casos reales positivos (capacidad de detectar verdaderos positivos).
    - ***F1-score***: media armónica, equilibrio entre precisión y recall.
    - ***ROC-AUC***: capacidad del modelo para distinguir entre clases (área bajo la curva ROC, mide la discriminación global del modelo).

    Utilizaremos el ***F1-score*** y ***mean-score*** como métrica principal, dado el leve desbalanceo de clases en la variable objetivo (***satisfaction***).

5.1 Elección del mejor modelo

En la siguiente tabla se muestra los resultados de las métricas obtenidas con cada uno de los algoritmos, donde el mejor modelo ha sido:

Ranking de orden los mejores modelos por métrica

Consolidación de métricas y rendimiento promedio por modelo

	Model	Accuracy	Precision	Recall	F1-score	ROC-AUC	mean_score	Mean Score
#1	xgboost_optuna_metrics	0.965	0.971	0.947	0.959	0.995	0.967247	0.967
#2	xgboost_gridsearch_metrics	0.963	0.961	0.954	0.957	0.995	0.966054	0.966
#3	xgboost_baseline_metrics	0.961	0.958	0.952	0.955	0.995	0.964315	0.964
#4	random_forest_baseline_metrics	0.961	0.967	0.943	0.955	0.994	0.963937	0.964
#5	random_forest_optuna_metrics	0.962	0.967	0.944	0.955	0.960	0.957529	0.958
#6	random_forest_randomized_metrics	0.961	0.968	0.942	0.955	0.959	0.957164	0.957
#7	random_forest_gridsearch_metrics	0.961	0.962	0.946	0.954	0.959	0.956422	0.956
#8	Decision Tree (optuna)	0.950	0.950	0.934	0.942	0.986	0.952749	0.953
#9	Decision Tree (gridsearchcv)	0.950	0.944	0.942	0.943	0.980	0.951821	0.952
#10	Decision Tree (baseline)	0.945	0.933	0.941	0.937	0.945	0.940187	0.940
#11	knn_gridsearch_metrics	0.938	0.943	0.911	0.927	0.981	0.939930	0.940
#12	knn_optuna_metrics	0.936	0.943	0.908	0.925	0.983	0.939157	0.939
#13	knn_baseline_metrics	0.926	0.935	0.893	0.913	0.970	0.927486	0.927
#14	Logistic Regression (Gridsearchcv)	0.870	0.842	0.861	0.852	0.928	0.870551	0.871
#15	Logistic Regression (Optuna)	0.869	0.841	0.862	0.851	0.928	0.870420	0.870
#16	Logistic Regression (Baseline)	0.869	0.841	0.862	0.851	0.928	0.870377	0.870

Modelos aparentemente mejores

Mejor modelo

- **Resultados Principales:**

    Tras comparar el rendimiento de todos los algoritmos:

- El ***Random Forest*** optimizado con ***GridSearchCV*** obtiene **el mejor rendimiento global**, con una media de métricas (***Mean Score***) de **0.956**, superando a los demás modelos.
- El ***XGBoost*** con ***Optuna*** logra valores muy competitivos, especialmente en ***F1-score*** y ***ROC-AUC***, pero con una **ligera menor estabilidad**.
- Las ***10 variables más relevantes*** fueron determinadas mediante ***feature importance***, y con ellas se reentrenó el modelo final para producción.

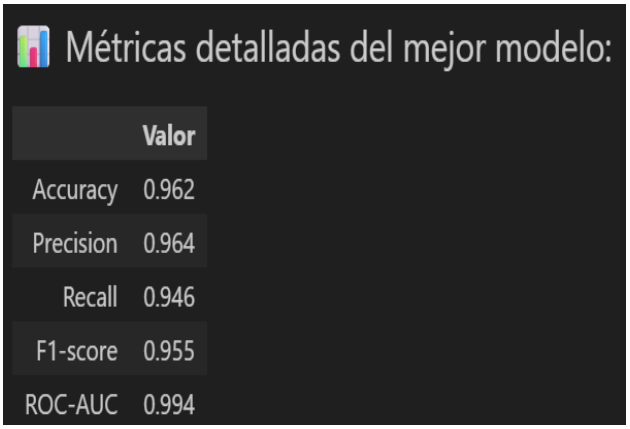
5.2 Justificación del mejor modelo escogido

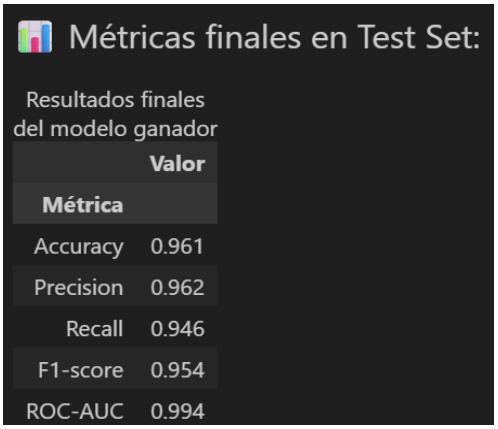
Aunque *XGBoost* (*Optuna*) obtuvo el *Mean Score* más alto, el **modelo** seleccionado como **ganador** fue **Random Forest (GridSearchCV)**, debido a su:

- **Mayor estabilidad** entre validaciones cruzadas,
- **Mejor rendimiento** en el conjunto de test (generalización),
- Y una **interpretabilidad más clara** para el análisis de las variables.

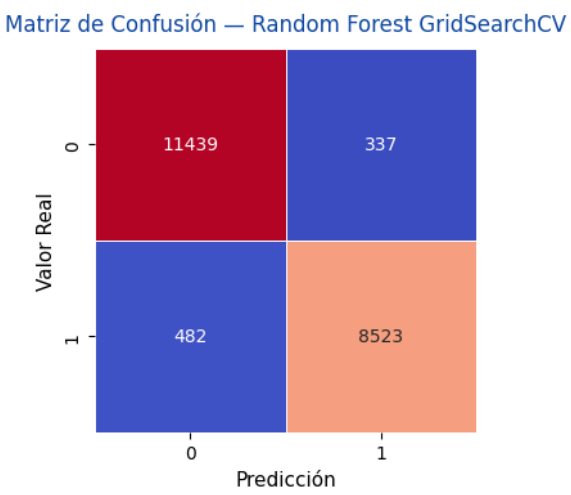
6. Evaluación del modelo ganador

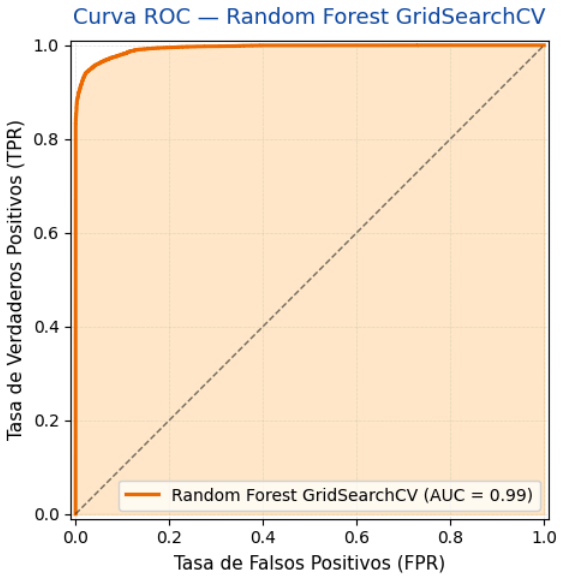
Los resultados obtenidos con el conjunto de prueba fueron los siguientes:





El modelo ***Random Forest GridSearchCV*** ha demostrado el **mejor equilibrio global entre *precisión* y *recall***, obteniendo un ***F1-score*** de 0.954 y una ***ROC-AUC*** de **0.994** y un ***mean-score*** de **0.956**, lo que indica una excelente capacidad para distinguir entre clases.





Interpretación de resultados:

- Las métricas cuantifican el rendimiento final del modelo sobre datos nunca vistos.
- La **matriz de confusión** muestra el balance entre verdaderos y falsos positivos/negativos, con una baja tasa de falsos negativos (6%) y falsos positivos (5%).
- La **curva ROC** ilustra la capacidad del modelo para distinguir entre clases.
- El área sombreada naranja representa el **AUC (Área Bajo la Curva)**: cuanto mayor, mejor rendimiento.

- **Identificación de la importancia de variables:** Identificamos las variables **que más influyen en las predicciones del modelo ganador**.

  Según el tipo de algoritmo, la importancia de variables se calcula de manera diferente:

  - **Modelos basados en árboles** (***Decision Tree***, ***Random Forest***, ***XGBoost***, etc.) → usan el atributo `***feature_importances_***`, que mide la reducción de impureza.
  - **Modelos lineales** (***Logistic Regression***, ***Linear SVM***, etc.) → usan `***coef_***`, que indica el peso absoluto de cada variable.
  - **Modelos sin atributos internos** (***KNN***, ***SVM*** con kernel no lineal, etc.) → se calcula la importancia mediante ***Permutation***

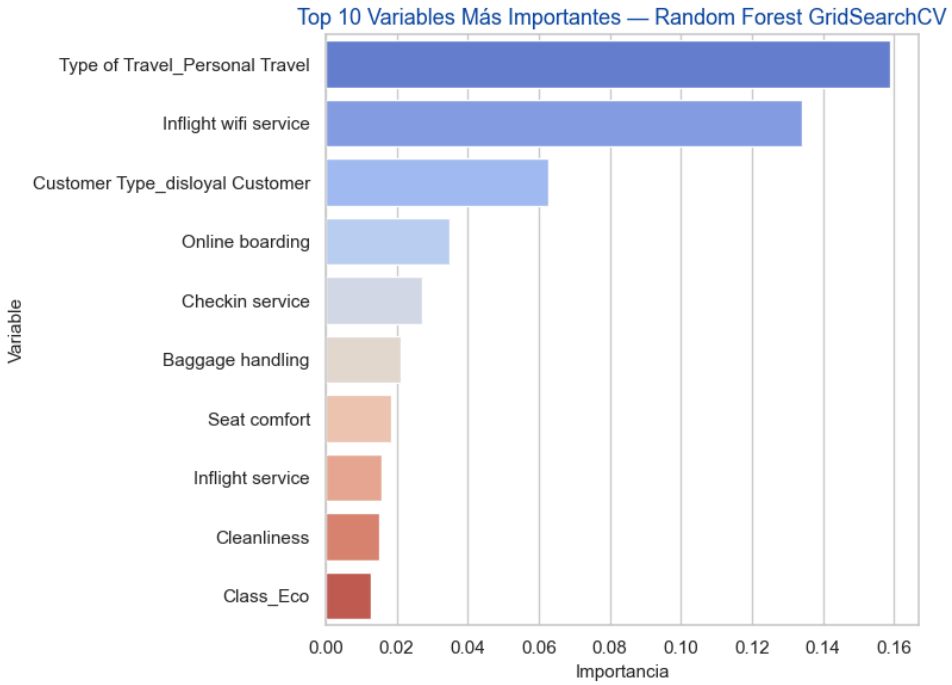
Importance, una técnica que mide cuánto se degrada el rendimiento al permutar una variable.

Esta estrategia garantiza una **evaluación universal y coherente**, sin importar el tipo de modelo que haya resultado ganador.

El análisis de **importancia de variables** reveló que los factores más determinantes fueron:

1. **Inflight Service**
2. **Customer Type**
3. **Seat Comfort**
4. **Online Boarding**
5. **Checking Service**
6. **Inflight Wifi Service**
7. **Cleanliness**
8. **Baggage Handling**
9. **OnBoard Service**
10. **Class_Eco**

Importancia de variables mediante Permutation Importance



6.1 Reentrenamiento del modelo con las 10 variables más importantes

En este paso se **entrena nuevamente** el **modelo ganador utilizando únicamente las 10 variables más relevantes** identificadas en el análisis anterior.

El objetivo es evaluar si el modelo mantiene un rendimiento similar —o incluso mejor— cuando se simplifica el conjunto de variables, lo cual aporta **ventajas** como:

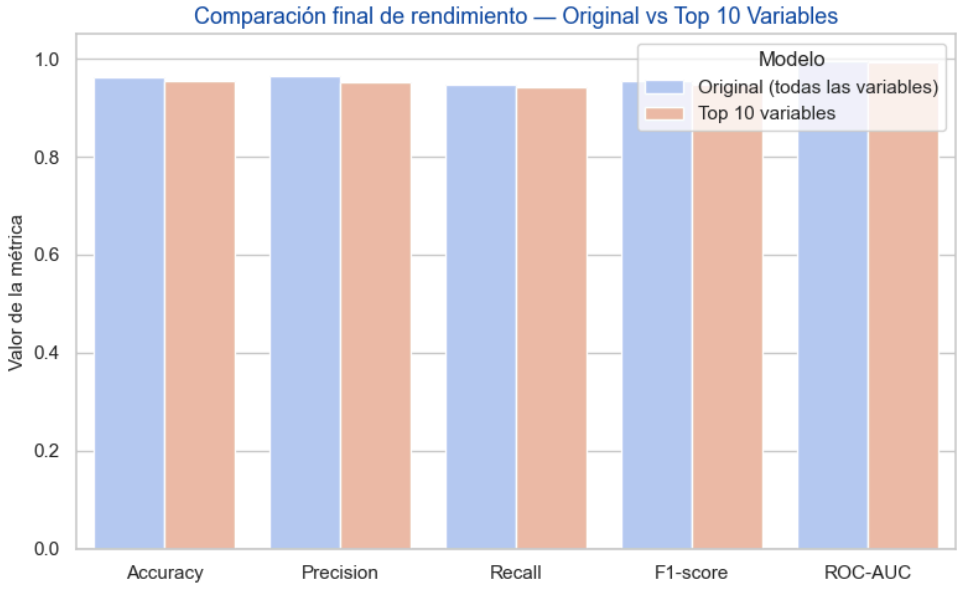
- **Mayor interpretabilidad** del modelo.
- **Menor complejidad computacional**.
- **Reducción del riesgo de *overfitting***.

Entrenamos una nueva instancia del mismo algoritmo ganador y se evalúa su rendimiento sobre el **conjunto de test** utilizando solo esas 10 variables.

📋 **Comparativa de métricas — Modelo original vs reducido**

	Métrica	Original (todas las variables)	Top 10 variables	Diferencia (%)
0	Accuracy	0.962	0.954	-0.82%
1	Precision	0.964	0.952	-1.29%
2	Recall	0.946	0.941	-0.60%
3	F1-score	0.955	0.946	-0.94%
4	ROC-AUC	0.994	0.992	-0.22%

Conclusión: El modelo reducido mantiene un rendimiento prácticamente idéntico al modelo original (***ΔF1* = -0.94%**, ***ΔROC-AUC* = -0.22%**).



7. Productivización del Modelo

El **modelo desplegado para su uso en producción** es el que hemos **entrenado con las 10 variables más importantes**.

El **modelo listo para producción** se encuentra en:
`***../models/deployed/model_random_forest_gridsearchcv_deployed.pkl***
`

El modelo seleccionado (reducido o completo) es registrado formalmente como el modelo de referencia para producción y cuyas **métricas**, comparación y trazabilidad quedan **documentadas** en los archivo**s** generados **para auditoría o integración futura en archivos JSON**, junto con el análisis del resto de modelos en los pasos anteriores a la comparación de métricas.

7.1 Arquitectura del sistema

Frontend (React + Tailwind)
↓
Backend (FastAPI + Pydantic)
↓
Modelo (Random Forest GridSearchCV .pkl)

7.2 Backend — *FastAPI*

El backend fue implementado con ***FastAPI***, un framework ligero y de alto rendimiento para construir ***APIs REST***.

Sus **principales componentes** incluyen:

- **Endpoint** ***/predict*** que recibe los datos de un cliente en formato JSON.
- **Validación de entrada** mediante modelos de ***Pydantic.***
- **Carga del modelo entrenado** desde un archivo ***.pkl.***
- **Respuesta JSON** con la **predicción** (***Satisfied*** / ***Unsatisfied***).

Ejemplo de request:

```json
{
  "Gender": "Female",
  "Age": 34,
  "Seat_Comfort": 4,
  "Inflight_Service": 5,
  "Cleanliness": 4,
  "Online_Boarding": 5
}
```

7.3 Frontend — React + Tailwind

El frontend fue construido en ***React***, con ***TailwindCSS*** para el estilo.
Cuenta con un **formulario intuitivo** donde el usuario ingresa las variables relevantes.

Una vez enviados los datos, la aplicación consume el endpoint del backend y muestra el resultado en tiempo real.

El diseño fue planificado en ***Figma***, asegurando una experiencia visual clara, moderna y coherente con la temática del proyecto.

8. Control de Versiones y Colaboración

- **Repositorio GitHub**: estructura organizada por ramas (main, dev, feature/*).
- **Commits**: redactados bajo la convención *Conventional Commits* para mayor trazabilidad.
- **Herramienta de gestión**: Git Projects, con tableros que reflejan el flujo de trabajo (*To Do*, *In Progress*, *Done*).
- **Documentación técnica**: generada automáticamente en /docs mediante la interfaz interactiva de FastAPI.

9. Conclusiones y Trabajo Futuro

El modelo desarrollado, como mejor modelo de los analizados, ***Random Forest GridSearchCV***, demostró una excelente capacidad predictiva, alcanzando un ***F1-score*** de ***0.95***, un ***mean-score*** de ***0,96*** y un ***overfitting inferior al 5%***, cumpliendo con los criterios de rendimiento exigidos.

Conclusiones principales:

- Las variables relacionadas con la experiencia de vuelo son los factores más influyentes en la satisfacción del cliente.
- El modelo ***Random Forest GridSearchCV*** ofreció el mejor equilibrio entre precisión y generalización de entre todos los modelos, incluso por encima de XGBoost con Optuna para este dataset.
- La aplicación web logra integrar de manera efectiva un modelo de IA con una interfaz accesible y funcional.

Trabajo futuro:

- Desplegar la aplicación en Render o AWS con CI/CD automatizado.
- Implementar un sistema de feedback en producción que permita monitorear el rendimiento del modelo y detectar *data drift*.
- Desarrollar una API de reentrenamiento automático, integrando conceptos básicos de MLOps.

10. Referencias

- Kaggle — *Airline Passenger Satisfaction Dataset*
- Scikit-learn Documentation
- FastAPI Documentation
- React & TailwindCSS Official Docs
- XGBoost Documentation
- Figma Community — *UI Design Systems*

11. Anexos

- Código fuente del modelo (model_training.ipynb)
- API Backend (main.py)
- Frontend React (App.jsx, PredictionForm.jsx)
- Capturas del diseño en Figma
- Requerimientos (requirements.txt)