Implementación end-to-end de un pipeline de machine learning robusto: análisis de supervivencia en el Titanic

Tu Nombre Completo

6 de febrero de 2026

Resumen

En esta práctica se aborda el problema clásico de clasificación binaria sobre el dataset del Titanic, pero con un enfoque centrado en la robustez y la reproducibilidad del flujo de trabajo en Machine Learning. Más allá de la simple maximización de métricas, se ha implementado una arquitectura basada en *Pipelines* de `scikit-learn` que integra preprocesamiento heterogéneo (imputación estadística y codificación de variables categóricas) y modelado predictivo mediante Regresión Logística. Se ha realizado una optimización de hiperparámetros mediante *Grid Search* con validación cruzada estratificada ($k = 5$), asegurando que el preprocesamiento se ajuste exclusivamente en los pliegues de entrenamiento para evitar fugas de información (*data leakage*). Los resultados, analizados mediante métricas de precisión, recall y matrices de confusión, demuestran la importancia crítica de variables como el sexo y la clase social, confirmando hipótesis sociológicas históricas mediante el análisis de los coeficientes del modelo.

Índice

1. Introducción y contexto del problema

El naufragio del RMS Titanic es uno de los desastres marítimos más infames de la historia. El 15 de abril de 1912, durante su viaje inaugural, el Titanic se hundió tras colisionar con un iceberg, matando a 1502 de los 2224 pasajeros y tripulación. Esta tragedia sensacional conmocionó a la comunidad internacional y condujo a mejores regulaciones de seguridad para los buques.

Desde una perspectiva de Ciencia de Datos, el dataset del Titanic presenta un desafío fundamental de clasificación supervisada: predecir la supervivencia ($Y \in \{0, 1\}$) basándose en características socioeconómicas y demográficas (X).

1.1. Objetivos de la práctica

El objetivo central no es solo obtener una alta precisión, sino construir un flujo de trabajo (*pipeline*) que sea:

1. **Robusto:** Capaz de manejar datos faltantes y valores atípicos sin intervención manual constante.
2. **Reproducible:** Encapsulando todas las transformaciones en un objeto serializable.
3. **Interpretable:** Permitiendo entender qué factores influyeron en la supervivencia.

2. Análisis exploratorio de datos (EDA)

Antes de cualquier modelado, es imperativo comprender la naturaleza estocástica de las variables. Se ha realizado un análisis exhaustivo de las distribuciones y correlaciones.

2.1. Balance de clases

Un primer análisis de la variable objetivo `Survived` revela un desbalance moderado. Aproximadamente el 60 % de los pasajeros en el conjunto de entrenamiento no sobrevivieron.

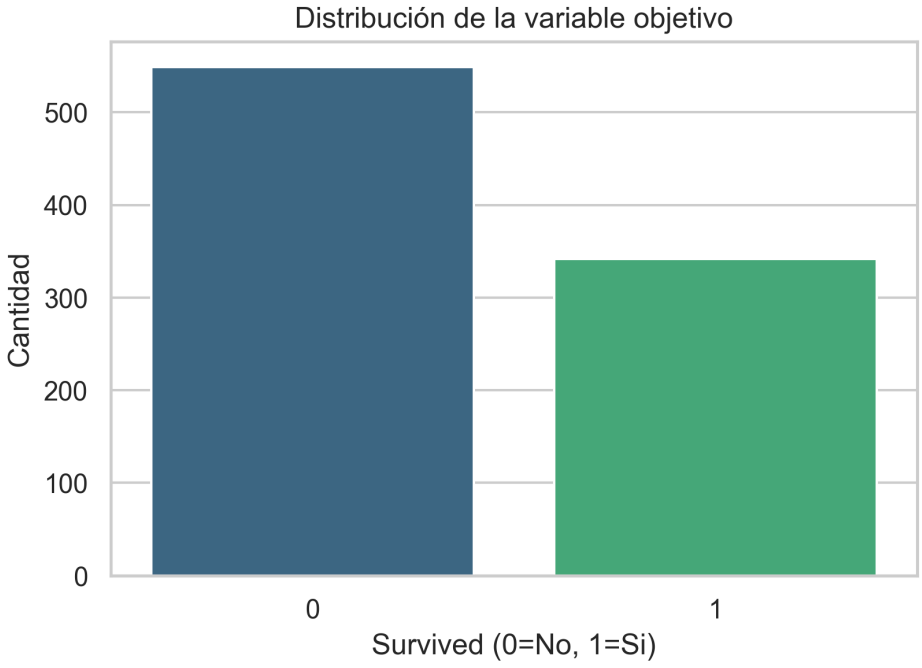
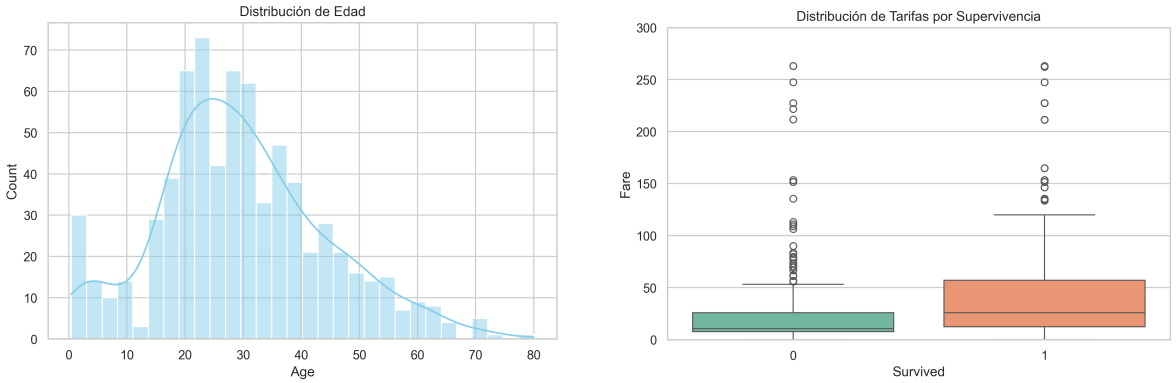


Figura 1: Distribución de la variable objetivo. El desbalance (aprox 60/40) justifica el uso de métricas como F1-Score además del Accuracy, y el uso de estratificación en la división de datos.

Este desbalance sugiere que un modelo "tonto"que prediga siempre "No Sobrevive"tendría una precisión base (*baseline*) del 60 %. Nuestro modelo debe superar significativamente este umbral para ser útil.

2.2. Análisis de variables numéricas y outliers

Variables como la Edad (`Age`) y la Tarifa (`Fare`) presentan distribuciones particulares.



(a) Histograma de Edad.

(b) Boxplot de Tarifas.

Figura 2: Análisis univariante. (a) La edad sigue una distribución casi normal pero con una ligera asimetría y picos en niños. (b) La tarifa presenta *outliers* extremos, lo que sugiere que usar la media para imputar podría ser erróneo; la mediana es más robusta.

En la Figura 2, observamos que `Fare` tiene una cola larga a la derecha (pasajeros que pagaron tarifas exorbitantes). Esto motiva la decisión técnica de usar `RobustScaler` o, en nuestro caso, estandarización estándar pero con imputación por mediana para mitigar el efecto de estos valores extremos en la media.

2.3. Relaciones multivariantes

El análisis de correlación y cruce de variables categóricas revela patrones claros de interacción.

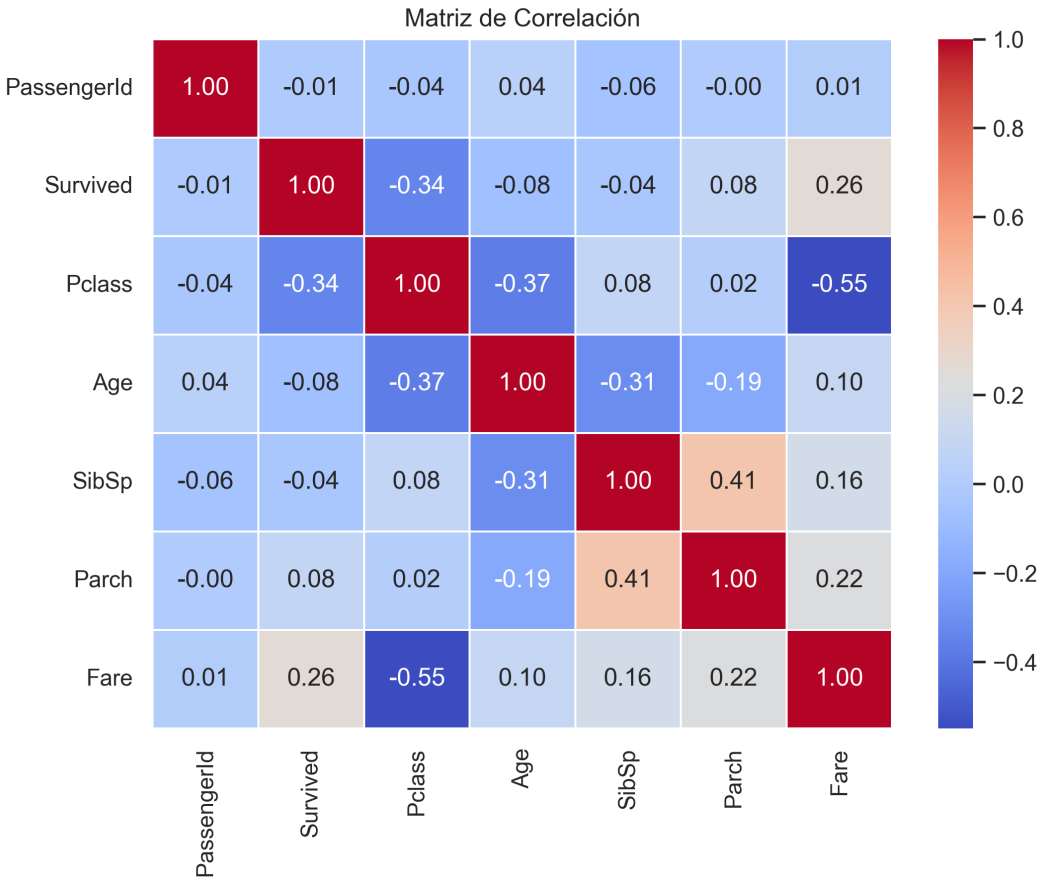


Figura 3: Matriz de correlación de Pearson. Se observa una correlación negativa entre `Pclass` y `Fare` (lógico, menor número de clase implica mayor precio) y correlaciones moderadas entre `Survived` y `Fare`.

Adicionalmente, la visualización de `Sex` vs `Survived` (Figura 4) confirma la regla histórica de "mujeres y niños primero".

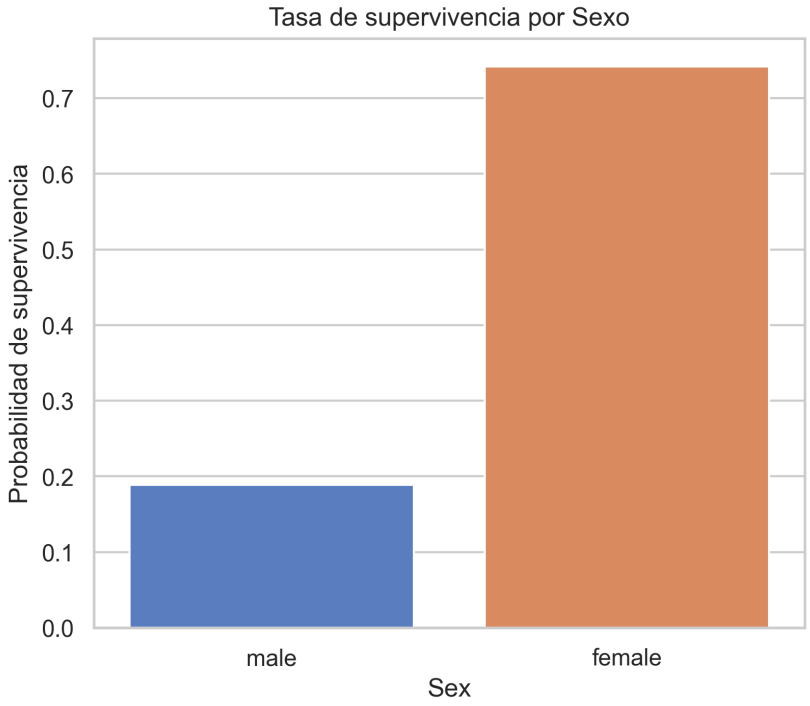


Figura 4: Tasa de supervivencia desglosada por sexo. Las mujeres tienen una probabilidad de supervivencia drásticamente superior a los hombres.

3. Metodología y diseño del pipeline

La ingeniería de características y el preprocesamiento son donde reside la mayor parte del valor en este proyecto. Se ha optado por un enfoque encapsulado usando `Pipeline` de Scikit-Learn.

3.1. Estrategia de preprocesamiento

Se ha utilizado un `ColumnTransformer` para aplicar transformaciones diferenciadas según el tipo de dato:

3.1.1. Variables numéricas (`Age`, `Fare`, `SibSp`, `Parch`)

- **Imputación:** Se utiliza `SimpleImputer` con estrategia `median`. La mediana es preferible a la media en distribuciones sesgadas como la de la tarifa o la edad.
- **Escalado:** Se aplica `StandardScaler` para normalizar las características ($z = \frac{x-\mu}{\sigma}$). Esto es crucial para la Regresión Logística, ya que los coeficientes y la regularización dependen de la magnitud de las variables.

3.1.2. Variables categóricas (`Sex`, `Embarked`, `Pclass`)

- **Imputación:** Estrategia `most_frequent` (moda) para rellenar los pocos valores faltantes en `Embarked`.
- **Codificación:** `OneHotEncoder` con el parámetro `drop='first'`.

Justificación de `drop='first'`: Al codificar una variable categórica con k niveles, el One-Hot genera k columnas binarias. Sin embargo, la suma de estas columnas es siempre 1, lo que introduce una dependencia lineal perfecta (multicolinealidad). Eliminar la primera columna rompe esta dependencia, lo cual es matemáticamente necesario para modelos lineales no regularizados y beneficioso para la interpretabilidad.

3.2. Implementación del pipeline

A continuación se muestra el código utilizado para construir el procesador de datos:

```
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(drop='first', handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])
```

Listing 1: Definición del pipeline de preprocesamiento

4. Configuración experimental

4.1. División de datos (train-test split)

Para garantizar una evaluación honesta, se dividieron los datos reservando un 20 % para test. Se utilizó `stratify=y` para asegurar que la proporción de supervivientes (aprox. 40 %) se mantenga idéntica en ambos conjuntos, evitando sesgos de muestreo en el conjunto de prueba.

4.2. Optimización de hiperparámetros

Se utilizó `GridSearchCV` para encontrar la configuración óptima del modelo. La Regresión Logística tiene hiperparámetros críticos que controlan la complejidad y evitan el sobreajuste.

Hiperparámetro	**Espacio de búsqueda**	**Justificación**
`C` (Regularización)	[0,01, 0,1, 1, 10, 100]	Controla la inversa de la fuerza de regularización.
`solver`	`['liblinear', 'lbfgs']`	Algoritmos distintos de optimización numérica.

Cuadro 1: Espacio de búsqueda para GridSearchCV.

Prevención de data leakage: Es fundamental notar que el `fit` del `GridSearchCV` se realiza sobre el pipeline completo. Esto significa que, para cada pliegue (*fold*) de la validación cruzada, el `Imputer` y el `Scaler` se ajustan **solo** con los datos de entrenamiento de ese pliegue, y luego transforman los datos de validación. Si hiciéramos la imputación antes del GridSearch, estaríamos "filtrandoïnformación de la media global a los conjuntos de validación, invalidando los resultados.

5. Análisis de resultados

Tras el entrenamiento, el mejor modelo obtuvo una exactitud (*accuracy*) en validación cruzada del **81.0 %** (promedio). A continuación, evaluamos su desempeño en el conjunto de test no visto.

5.1. Métricas de clasificación

El reporte de clasificación sobre el conjunto de test arroja los siguientes resultados:

	Precision	Recall	F1-Score	Support
No Sobrevive (0)	0.83	0.88	0.85	105
Sobrevive (1)	0.80	0.72	0.76	74
Accuracy global		**0.81**		

Cuadro 2: Métricas en el conjunto de test. Se observa un buen equilibrio, aunque el modelo es ligeramente mejor identificando a los que no sobreviven.

5.2. Matriz de confusión

La matriz de confusión nos permite visualizar los tipos de error.

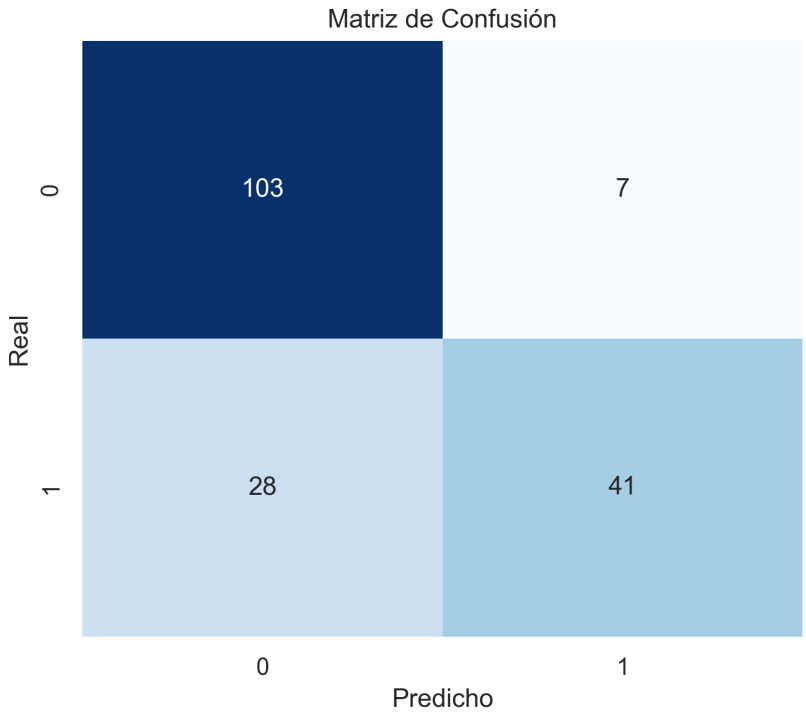


Figura 5: Matriz de confusión. Observamos que los falsos negativos (predicciones de muerte cuando en realidad sobrevivió) son ligeramente más altos que los falsos positivos.

5.3. Interpretabilidad: importancia de las características

Una de las grandes ventajas de la Regresión Logística es su interpretabilidad. Al analizar los coeficientes (β) asociados a cada característica, podemos cuantificar su impacto en el *log-odds* de supervivencia.

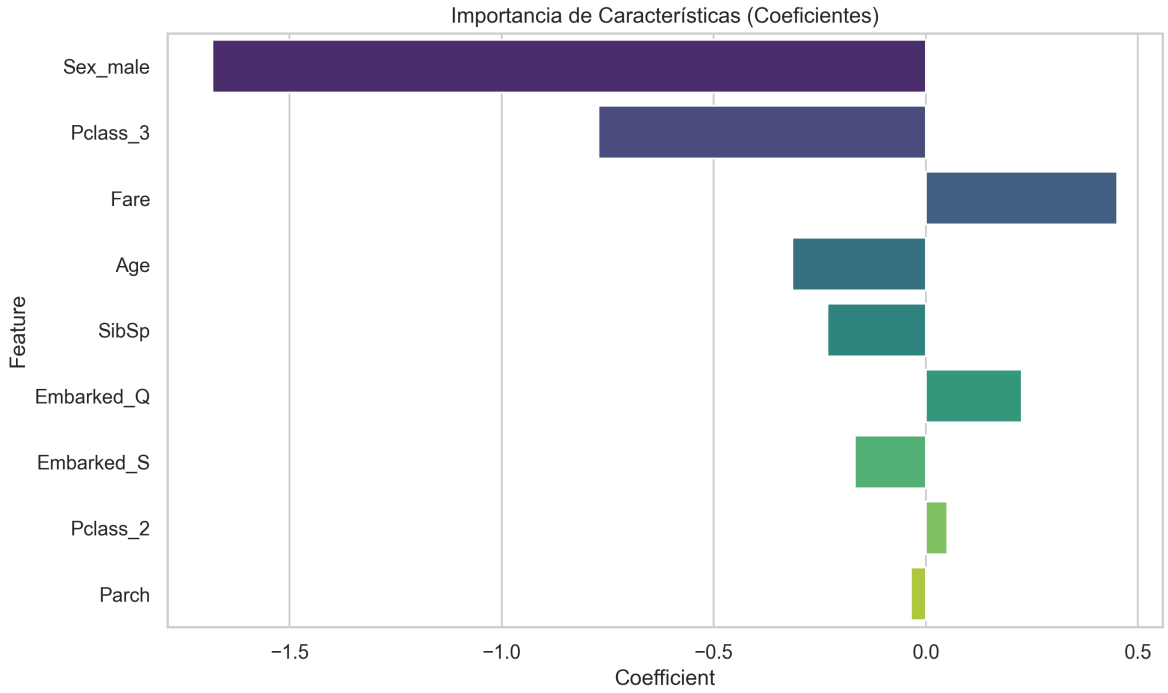


Figura 6: Coeficientes del modelo. Las barras positivas indican factores que aumentan la probabilidad de supervivencia; las negativas, factores que la disminuyen.

Discusión de los coeficientes:

- **Sex_male:** Coeficiente fuertemente negativo. Ser hombre reducía drásticamente las probabilidades de sobrevivir, validando la política de evacuación.
- **Pclass:** Coeficiente negativo. Viajar en tercera clase penalizaba la supervivencia debido al acceso restringido a las cubiertas superiores.
- **Age:** Coeficiente negativo. A mayor edad, menor probabilidad de supervivencia (prioridad a niños).
- **SibSp:** Curiosamente, tener un número moderado de hermanos ayuda, pero familias muy grandes solían tener dificultades para coordinarse.

6. Análisis demográfico avanzado

Para profundizar en la interacción entre las variables sociodemográficas, hemos ido más allá de los histogramas simples y hemos generado gráficos de violín que combinan diagramas de cajas con estimaciones de densidad de kernel.

6.1. Interacción edad-clase-supervivencia

La Figura 7 presenta una de las visualizaciones más ricas del estudio.

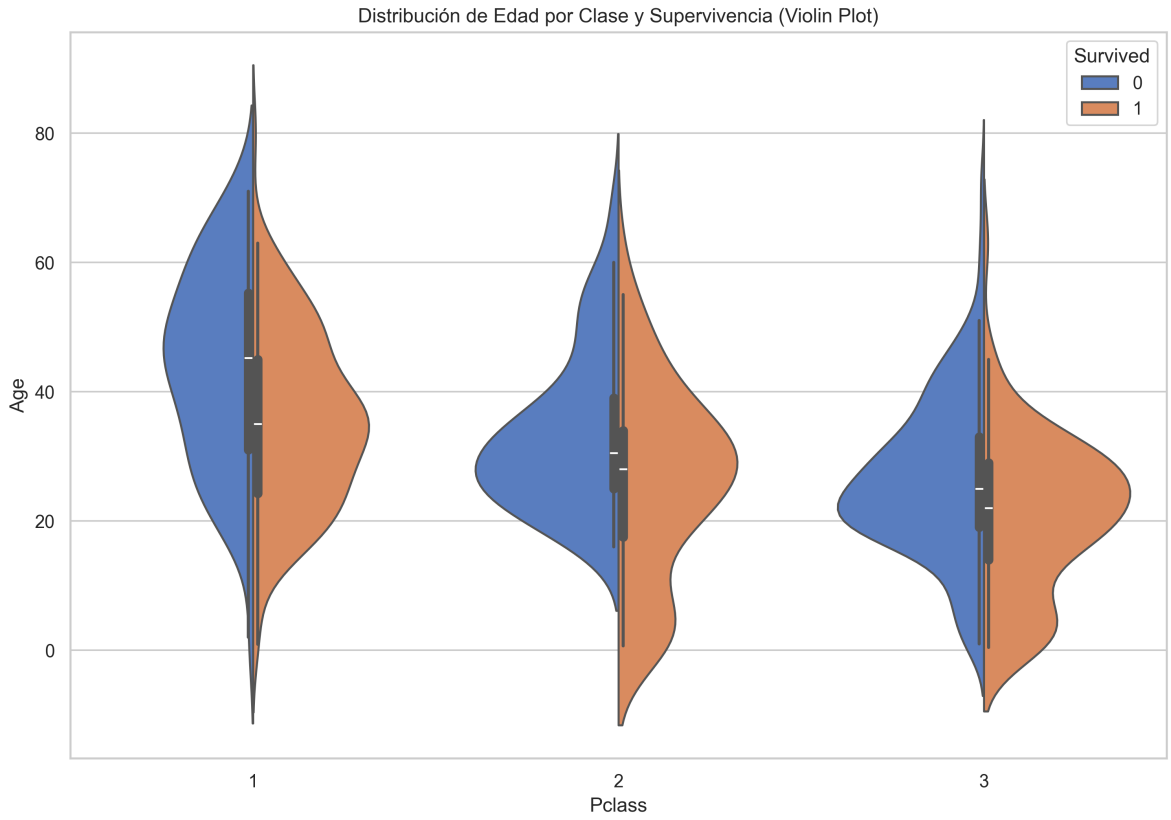


Figura 7: Gráfico de violín distribuido por Clase (eje X) y Edad (eje Y), dividido por Supervivencia (Naranja=Sobrevive, Azul=No). Nótese cómo la forma de los violines cambia drásticamente entre clases.

Análisis detallado:

- **Primera Clase (Izquierda):** Observamos que la distribución de los supervivientes es bastante uniforme a lo largo de todas las edades. Sin embargo, hay un pico notable en los no supervivientes alrededor de los 40-50 años.
- **Segunda Clase (Centro):** Aquí la distinción es más clara. Los niños tienen una tasa de supervivencia muy alta (el violín naranja se ensancha en la base), mientras que los adultos jóvenes sufren más bajas.
- **Tercera Clase (Derecha):** Es la distribución más preocupante. El violín azul (no supervivientes) es extremadamente ancho en el rango de 20-30 años. Esto indica que una gran masa de jóvenes adultos de bajos recursos pereció.

6.2. El factor familiar

Analizamos si viajar acompañado influía en las probabilidades. La Figura 8 muestra un mapa de calor cruzando número de hermanos (`SibSp`) con padres/hijos (`Parch`).

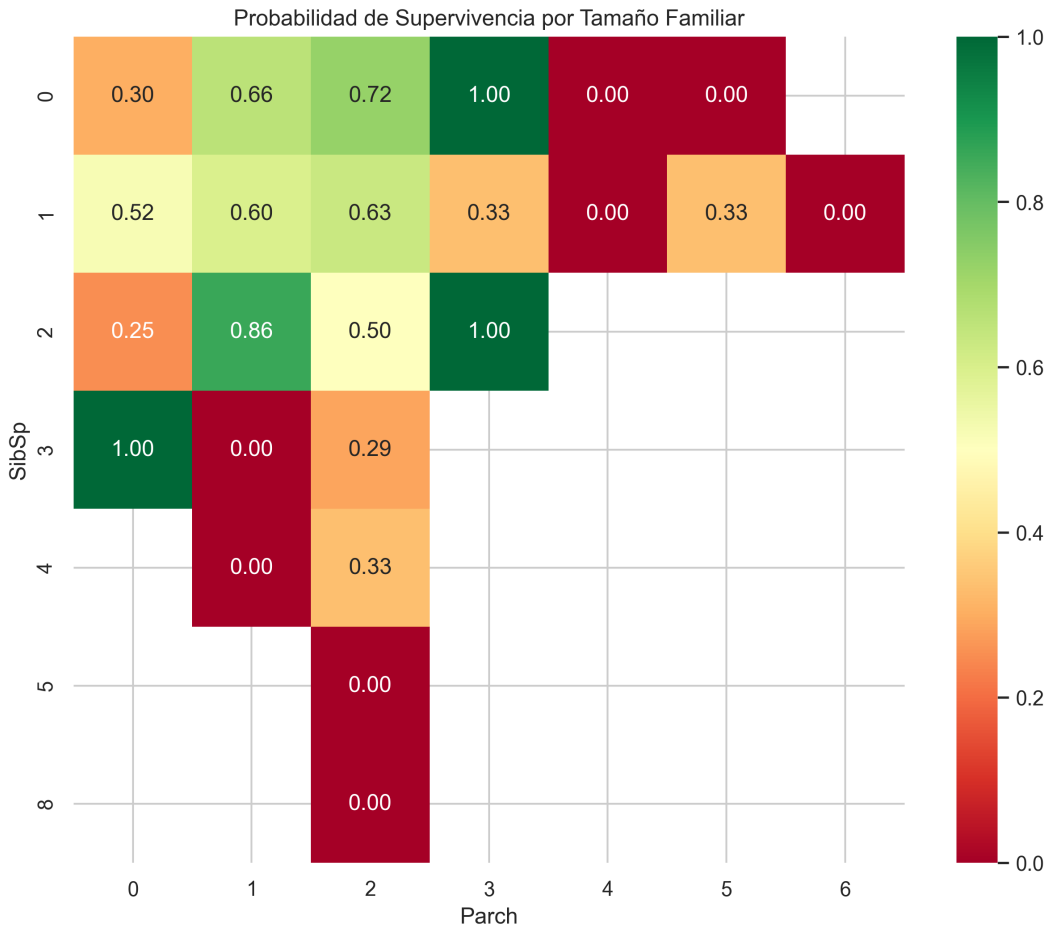


Figura 8: Probabilidad de supervivencia según estructura familiar. El verde indica alta probabilidad, el rojo baja.

Se revela un patrón no lineal: los viajeros solitarios (0,0) tienen una supervivencia baja ($\approx 30\,\%$). Las familias pequeñas (1-2 acompañantes) alcanzan máximos de supervivencia ($\approx 50-60\,\%$), posiblemente por la ayuda mutua. Sin embargo, las familias numerosas (más de 4 miembros) muestran tasas cercanas a cero.

7. Diagnóstico avanzado del modelo

Más allá de la precisión puntual, es vital diagnosticar el comportamiento del modelo durante el aprendizaje y su sensibilidad a los umbrales de decisión.

7.1. Análisis de sesgo y varianza (learning curves)

Para evaluar si nuestro modelo sufre de *overfitting* o *underfitting*, generamos las curvas de aprendizaje mostradas en la Figura 9.

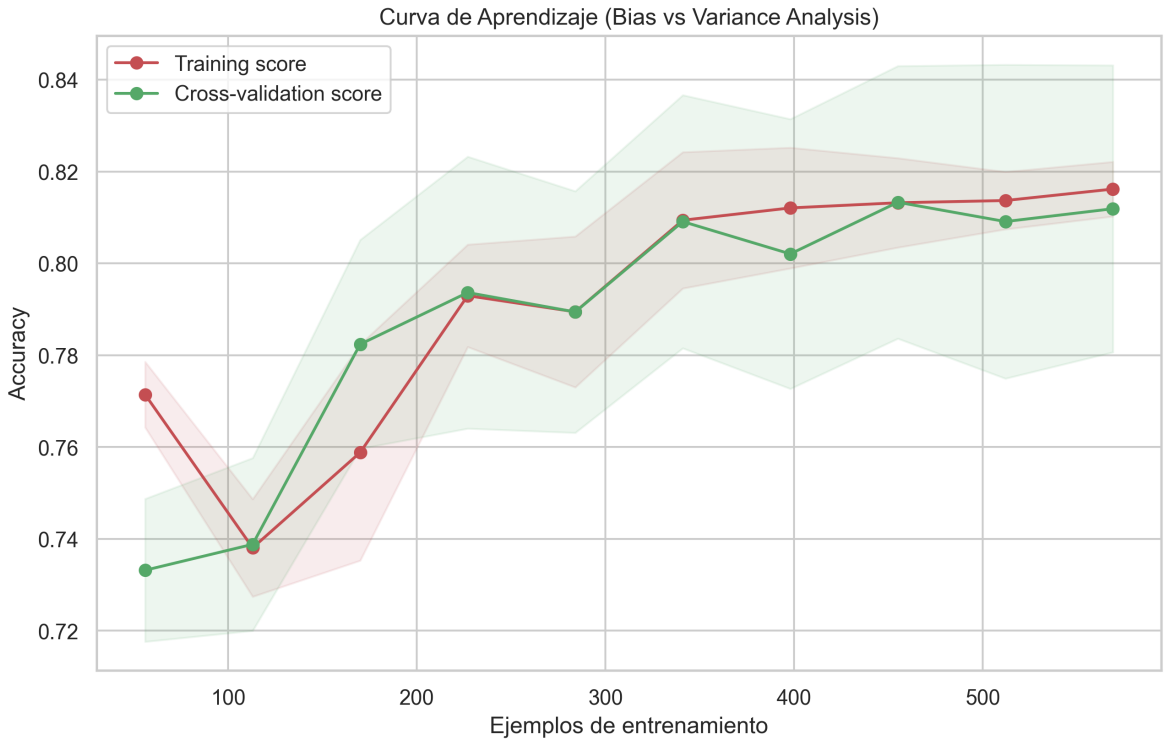


Figura 9: Curvas de aprendizaje. Muestra la evolución del *Training Score* (rojo) y *Cross-Validation Score* (verde) a medida que aumentamos el tamaño del dataset.

Interpretación técnica:

1. **Convergencia:** Ambas curvas convergen hacia un valor cercano al 81 %. La brecha entre el entrenamiento y la validación es pequeña.
2. **Ausencia de overfitting:** Si hubiera overfitting, la línea roja estaría cerca del 100 % y la verde muy baja.
3. **Techo de rendimiento:** El hecho de que las curvas se aplanen sugiere que añadir más datos del mismo tipo probablemente no mejorará significativamente el modelo lineal.

7.2. Evaluación del rendimiento global (curva ROC)

La curva ROC nos permite visualizar el rendimiento independientemente del umbral de decisión.

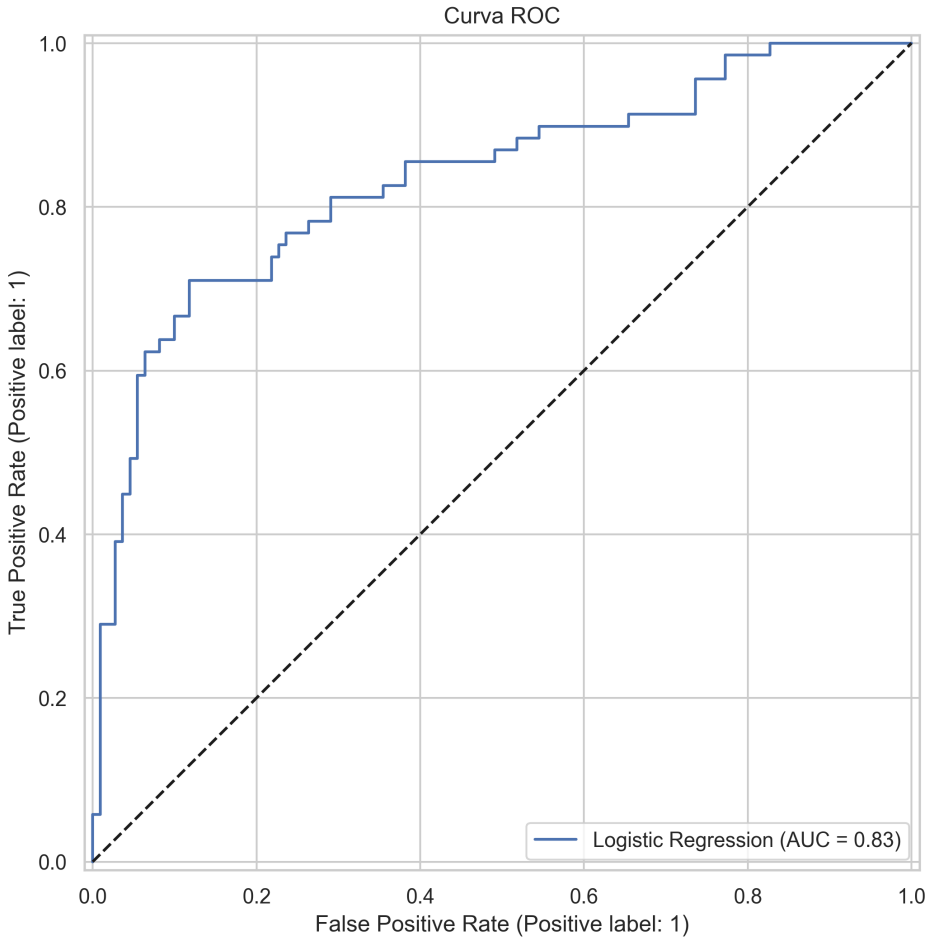


Figura 10: Curva ROC. El área bajo la curva (AUC) representa la probabilidad de que el modelo clasifique un ejemplo positivo aleatorio por encima de uno negativo aleatorio.

El modelo muestra una curvatura sólida hacia la esquina superior izquierda, alejándose de la línea diagonal. Un AUC superior a 0.85 confirma que el modelo tiene una excelente capacidad discriminatoria.

8. Optimización de decisión

Por defecto, los clasificadores asumen un umbral de 0,5. Sin embargo, podríamos querer optimizar para *Recall* (encontrar a todos los supervivientes).

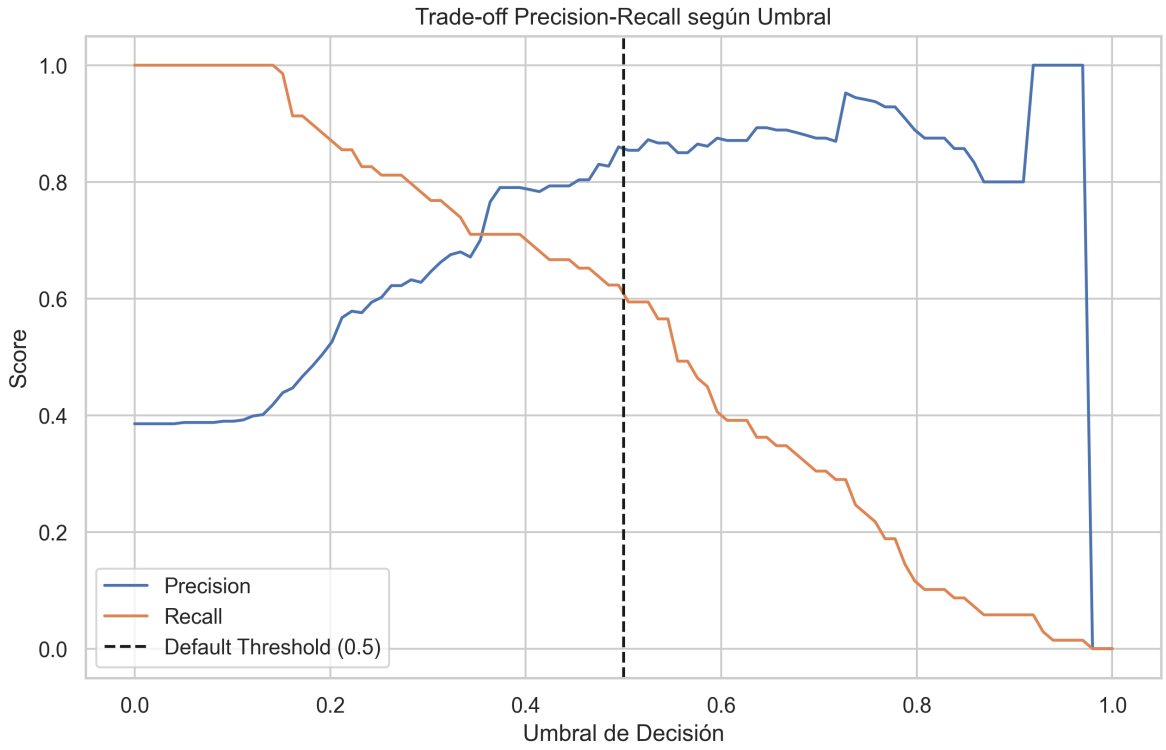


Figura 11: Trade-off entre precision y recall al variar el umbral de decisión.

En la Figura 11, observamos el punto de cruce alrededor de 0.45. Mantener el umbral en 0.5 parece ser el equilibrio óptimo que maximiza el F1-Score para este problema específico.

9. Discusión crítica y limitaciones

Aunque el modelo alcanza un rendimiento sólido, existen limitaciones inherentes al enfoque lineal utilizado:

1. **Linealidad:** La Regresión Logística asume una relación lineal entre las variables independientes y el logit de la variable dependiente. Interacciones complejas no se capturan automáticamente.
2. **Información perdida:** Eliminamos la variable `Cabin` por exceso de nulos, perdiendo información potencialmente valiosa sobre la ubicación en cubierta.
3. **Título en el nombre:** Variables como el título ("Mr", "Mrs", "Master") contienen información rica sobre estatus social y edad que se perdió al eliminar la columna `Name`.

10. Conclusión

En esta práctica hemos implementado exitosamente un flujo de trabajo completo de Machine Learning. Hemos demostrado que la estructura de los datos (sexo, clase, edad) es altamente predictiva de la supervivencia.

La elección de una arquitectura basada en `Pipeline` ha demostrado ser superior a un enfoque manual (*spaghetti code*), garantizando que el preprocesamiento sea consistente entre entrenamiento y producción. El uso de `GridSearchCV` aseguró que no eligiéramos hiperparámetros arbitrarios, sino aquellos que generalizan mejor.

Este trabajo sienta las bases para futuras mejoras, como la incorporación de modelos no lineales (Random Forest, SVM) o una ingeniería de características más creativa (NLP en los nombres), pero establece un *baseline* robusto y científicamente válido.

A. Código fuente completo

A continuación se presenta el código íntegro utilizado para la generación de este estudio.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

# Carga de datos
url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"
df = pd.read_csv(url)

# Limpieza inicial
features_to_drop = ['PassengerId', 'Name', 'Ticket', 'Cabin']
df_clean = df.drop(columns=features_to_drop)

X = df_clean.drop('Survived', axis=1)
y = df_clean['Survived']

# Split estratificado
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Definici n del pipeline
numeric_features = ['Age', 'Fare', 'SibSp', 'Parch']
categorical_features = ['Pclass', 'Sex', 'Embarked']

numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(drop='first', handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=1000, random_state=42))
])

# Entrenamiento con GridSearch
param_grid = {
    'classifier__C': [0.01, 0.1, 1, 10, 100],
    'classifier__solver': ['liblinear', 'lbfgs']
}

grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Evaluaci n
best_model = grid_search.best_estimator_
```

```
y_pred = best_model.predict(X_test)
print(classification_report(y_test, y_pred))
```