Número Grupo:

INTEGRANTES DEL GRUPO:

Nombre 1:

Nombre 2:

etc.

Indicaciones

A. Objetivos de la tarea

El objetivo de esta tarea es desarrollar y aplicar un modelo de machine learning para predecir el número de personas que no asisten a abordar los vuelos de la aerolínea AeroML, basándose en distintos parámetros como ruta, día de la semana, hora del día, entre otros.

B. Prerrequisitos para desarrollar la tarea

Antes de trabajar en esta tarea deben haber comprendido los contenidos de la unidad de sobre algoritmos de aprendizaje supervisado, especialmente la temática sobre regresión.

C. Instrucciones para la elaboración de la tarea

REQUISITOS GENERALES

1. **Análisis exploratorio de los datos**
2. **Preparación de los datos**
3. **Extracción de atributos de la fecha**
4. **Selección de variables**
5. **Desarrollo del modelo**
6. **Reflexione sobre el modelo**
7. **Evaluación del modelo**
8. **Reflexión final**

Descripción de Variables en el Conjunto de Datos de Vuelos

Variables Principales

1. `id`:
    - Identificador único para cada registro.
    - Útil para indexación y combinación de datos.
2. `fecha`:
    - Fecha del vuelo.
    - Importante para el análisis temporal y la identificación de patrones estacionales o tendencias.
3. `numero_vuelo`:
    - Identificador único del vuelo.
    - Sirve para rastrear la eficiencia y popularidad de rutas específicas.
4. `origen` y `destino`:
    - Aeropuertos de origen y destino.
    - Pueden usarse para analizar la demanda entre diferentes ubicaciones.
5. `distancia`:
    - Distancia entre el origen y el destino.
    - Puede influir en la tarifa y el consumo de combustible.
6. `capacidad`:
    - Número total de asientos en el avión.
    - Útil para calcular la tasa de ocupación y eficiencia del vuelo.
7. `venta_usd`:
    - Ingresos totales del vuelo en dólares estadounidenses.
    - Un KPI clave para la rentabilidad.
8. `agendados`:
    - Número total de reservas para el vuelo.
    - Indica la demanda y permite calcular la tasa de ocupación.

Variables de Pasajeros

9. `inasistencia`:
    - Número de pasajeros que no se presentaron.
    - Esto puede afectar la rentabilidad y requiere estrategias de overbooking cuidadosas.
10. `vuelo_denegado`:
    - Número de pasajeros que no pudieron abordar debido al exceso de reservas.
    - Importante para evaluar la eficiencia del algoritmo de overbooking.

Variables de Tarifas

11. `tarifa_mediabaja`, `tarifa_alta`, `tarifa_mediaalta`, `tarifa_baja`:
    - Distribución de los tipos de tarifas adquiridas.
    - Esto puede ayudar a segmentar a los clientes y ajustar las estrategias de precios.
12. `pax_freqflyer`:

- Número de pasajeros que redimieron millas.
- Importante para medir el compromiso del cliente.

13. `agendado_grupal`:
    - Número de reservas grupales.
    - Estos suelen tener tarifas más bajas y podrían afectar la rentabilidad.

Variables de Conexión

14. `conexion_nacional`, `conexion_internacional`, `sin_conexion`:
    - Indican si el vuelo es parte de una conexión nacional, internacional o si los pasajeros no están en una conexión.
    - Esto puede afectar la logística y la planificación.

Variables Adicionales

15. `sin_stock`:
    - Variable binaria que muestra los días sin capacidad para vender más boletos.
    - Un indicador de alta demanda que podría usarse para ajustar tarifas o frecuencias de vuelo.
16. `year`, `month`, `day`, `day_of_week`, `hour`:
    - Variables temporales que pueden ser útiles para modelar efectos estacionales o patrones diurnos.

Lo primero es leer el dataframe y comprender sus características y estadística descriptiva básica.

In [1]:
```
import pandas as pd
import numpy as np

# Ruta al archivo CSV
ruta_archivo = 'datos_vuelos_AeroML.csv'

# Lectura del archivo CSV en un DataFrame de pandas
df = pd.read_csv(ruta_archivo)

print(df.info())
print(df.head())
print(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18322 entries, 0 to 18321
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      18322 non-null  int64
 1   fecha                   18322 non-null  object
 2   numero_vuelo            18322 non-null  int64
 3   origen                  18322 non-null  object
 4   destino                 18322 non-null  object
 5   distancia               18322 non-null  int64
 6   inasistencia            18322 non-null  int64
 7   vuelo_denegado          18322 non-null  int64
 8   tarifa_mediabaja        17315 non-null  float64
 9   tarifa_alta             18322 non-null  int64
 10  tarifa_mediaalta        18322 non-null  int64
 11  tarifa_baja             18322 non-null  int64
 12  pax_freqflyer           18322 non-null  int64
 13  agendado_grupal         18322 non-null  int64
 14  sin_stock               18322 non-null  int64
 15  conexion_nacional       18322 non-null  int64
 16  conexion_internacional  18322 non-null  int64
 17  sin_conexion            18048 non-null  float64
 18  hora_salida             18317 non-null  object
 19  capacidad               18322 non-null  int64
 20  venta_usd               18322 non-null  float64
 21  agendados               17681 non-null  float64
dtypes: float64(4), int64(14), object(4)
memory usage: 3.1+ MB
None
        id       fecha  numero_vuelo origen destino  distancia  inasistencia  \
0    69922  2009-02-23          8942    ANF     SCL       1106             7
1   469723  2010-01-13          8941    SCL     ANF       1106            18
2   779308  2010-10-04          9128    ANF     SCL       1106             6
3   429392  2009-12-11          7941    SCL     ANF       1106            10
4  1286557  2011-11-20          9139    SCL     ANF       1106             8

   vuelo_denegado  tarifa_mediabaja  tarifa_alta  ...  pax_freqflyer  \
0               0             124.0            5  ...             20
1               0              56.0            0  ...              4
2               0               1.0            0  ...              2
3               0             122.0            2  ...             14
4               0               1.0            0  ...             14

   agendado_grupal  sin_stock  conexion_nacional  conexion_internacional  \
0                0          0                  0                       0
1                0          0                  7                       6
2                0          0                  0                       1
3                0          0                 11                       1
4                0          1                  0                       1

   sin_conexion          hora_salida  capacidad venta_usd  agendados
0         259.0  2024-12-28 21:15:00        168    8399.7      259.0
1          96.0  2024-12-28 18:15:00        174    8535.7      109.0
2          78.0  2024-12-28 17:26:00        218    3525.4       79.0
3         223.0  2024-12-28 20:20:00        174   10578.4      235.0
4         199.0  2024-12-28 07:47:00        220    5769.4      200.0

[5 rows x 22 columns]
                id  numero_vuelo  distancia  inasistencia  vuelo_denegado  \
count  1.832200e+04  18322.000000    18322.0  18322.000000    18322.000000
mean   6.129082e+05   8960.270276     1106.0      8.327148        0.055180
```

```
std    3.838625e+05    325.001168        0.0      5.443675        0.578305
min    1.580000e+02   7926.000000     1106.0      0.000000        0.000000
25%    2.771338e+05   8933.000000     1106.0      5.000000        0.000000
50%    5.752390e+05   9126.000000     1106.0      8.000000        0.000000
75%    9.329845e+05   9138.000000     1106.0     11.000000        0.000000
max    1.350202e+06   9251.000000     1106.0     79.000000       20.000000

       tarifa_mediabaja   tarifa_alta  tarifa_mediaalta   tarifa_baja  \
count      17315.000000  18322.000000      18322.000000  18322.000000
mean          50.279873      2.717607          4.352254     83.468781
std           49.124795      5.704688          9.669340     40.563930
min            0.000000      0.000000          0.000000      0.000000
25%            2.000000      0.000000          0.000000     57.000000
50%           36.000000      1.000000          1.000000     87.000000
75%           97.000000      3.000000          4.000000    111.000000
max          185.000000    109.000000        138.000000    276.000000

       pax_freqflyer  agendado_grupal     sin_stock  conexion_nacional  \
count   18322.000000     18322.000000  18322.000000       18322.000000
mean       11.110523         3.858804      0.072863          10.299858
std         7.556938        12.973262      0.259919          18.130702
min         0.000000         0.000000      0.000000           0.000000
25%         6.000000         0.000000      0.000000           1.000000
50%        10.000000         0.000000      0.000000           5.000000
75%        15.000000         0.000000      0.000000          13.000000
max        84.000000       158.000000      1.000000         181.000000

       conexion_internacional  sin_conexion     capacidad     venta_usd  \
count            18322.000000  18048.000000  18322.000000  18322.000000
mean                 4.090929    137.524601    177.890460  10767.573158
std                  6.044964     64.266551     43.336907   5968.978632
min                  0.000000      0.000000      0.000000    193.700000
25%                  1.000000     89.000000    174.000000   6517.850000
50%                  2.000000    127.000000    174.000000   9595.400000
75%                  5.000000    184.000000    218.000000  13753.200000
max                 89.000000    366.000000    274.000000  52995.600000

         agendados
count  17681.000000
mean     152.004864
std       64.448983
min        3.000000
25%      102.000000
50%      144.000000
75%      201.000000
max      384.000000
```

Según lo observado se corregirán las columnas 'Fecha' y 'hora_salida'. LA primera se transformará a tipo de dato datetime64[ns], mientras a la segunda se extraerá únicamenta la hora y se mantendrá su tipo de dato object.

In [2]:
```
# Convertir 'fecha' a tipo datetime
df['fecha'] = pd.to_datetime(df['fecha'], errors='coerce')

# Convertir 'hora_salida' a tipo datetime y extraer solo la hora
df['hora_salida'] = pd.to_datetime(df['hora_salida'], errors='coerce').dt.time

# Mostrar las primeras filas para ver el resultado
print(df[['fecha', 'hora_salida']].head())
print(df.info())
```

```
       fecha hora_salida
0 2009-02-23    21:15:00
1 2010-01-13    18:15:00
2 2010-10-04    17:26:00
3 2009-12-11    20:20:00
4 2011-11-20    07:47:00
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18322 entries, 0 to 18321
Data columns (total 22 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   id                     18322 non-null  int64
 1   fecha                  18322 non-null  datetime64[ns]
 2   numero_vuelo           18322 non-null  int64
 3   origen                 18322 non-null  object
 4   destino                18322 non-null  object
 5   distancia              18322 non-null  int64
 6   inasistencia           18322 non-null  int64
 7   vuelo_denegado         18322 non-null  int64
 8   tarifa_mediabaja       17315 non-null  float64
 9   tarifa_alta            18322 non-null  int64
 10  tarifa_mediaalta       18322 non-null  int64
 11  tarifa_baja            18322 non-null  int64
 12  pax_freqflyer          18322 non-null  int64
 13  agendado_grupal        18322 non-null  int64
 14  sin_stock              18322 non-null  int64
 15  conexion_nacional      18322 non-null  int64
 16  conexion_internacional 18322 non-null  int64
 17  sin_conexion           18048 non-null  float64
 18  hora_salida            18317 non-null  object
 19  capacidad              18322 non-null  int64
 20  venta_usd              18322 non-null  float64
 21  agendados              17681 non-null  float64
dtypes: datetime64[ns](1), float64(4), int64(14), object(3)
memory usage: 3.1+ MB
None
```

Ahora, para facilidad de visualización se despliega una tabla estilizada.

In [3]:
```
df.head().style.set_properties(**{'background-color': '#f0f0f0', 'color': 'black'})
```

Out[3]:

	id	fecha	numero_vuelo	origen	destino	distancia	inasistencia	vuelo_denegado	tarifa_me
0	69922	2009-02-23 00:00:00	8942	ANF	SCL	1106	7	0	124
1	469723	2010-01-13 00:00:00	8941	SCL	ANF	1106	18	0	56
2	779308	2010-10-04 00:00:00	9128	ANF	SCL	1106	6	0	1
3	429392	2009-12-11 00:00:00	7941	SCL	ANF	1106	10	0	122
4	1286557	2011-11-20 00:00:00	9139	SCL	ANF	1106	8	0	1

Desarrollo

1. Análisis Exploratorio y Preparación de Datos

Realice un análisis exploratorio y una preparación de datos que incluya:

Comentarios:

- Comente si existen datos faltantes, si hay valores que no tienen sentido, si hay outliers, etc.
- Describa las decisiones que tomó para la limpieza, imputación y/o transformación de los datos.

a) La gestión de valores faltantes o nulos (si existieran).

La única columna con valores faltantes es tarifa_mediabaja tiene 17,315 valores no nulos de un total de 18,322. Eso implica 1,007 valores faltantes (~5.5% del total).

Rellenaremos los valores nulos usando la mediana (median) esta estrategia es más simple considerando porcentaje de nulos no es demasiado alto.

In [4]:
```
df['tarifa_mediabaja'] = df['tarifa_mediabaja'].fillna(df['tarifa_mediabaja'].median())
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18322 entries, 0 to 18321
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      18322 non-null  int64
 1   fecha                   18322 non-null  datetime64[ns]
 2   numero_vuelo            18322 non-null  int64
 3   origen                  18322 non-null  object
 4   destino                 18322 non-null  object
 5   distancia               18322 non-null  int64
 6   inasistencia            18322 non-null  int64
 7   vuelo_denegado          18322 non-null  int64
 8   tarifa_mediabaja        18322 non-null  float64
 9   tarifa_alta             18322 non-null  int64
 10  tarifa_mediaalta        18322 non-null  int64
 11  tarifa_baja             18322 non-null  int64
 12  pax_freqflyer           18322 non-null  int64
 13  agendado_grupal         18322 non-null  int64
 14  sin_stock               18322 non-null  int64
 15  conexion_nacional       18322 non-null  int64
 16  conexion_internacional  18322 non-null  int64
 17  sin_conexion            18048 non-null  float64
 18  hora_salida             18317 non-null  object
 19  capacidad               18322 non-null  int64
 20  venta_usd               18322 non-null  float64
 21  agendados               17681 non-null  float64
dtypes: datetime64[ns](1), float64(4), int64(14), object(3)
memory usage: 3.1+ MB
None
```

b) La codificación de variables categóricas (si existieran).

Variables categóricas: Según .info(), las variables que requieren codificación son:

origen destino

Estas son variables de tipo object (texto) y representan aeropuertos.

In [5]:
```python
df = pd.get_dummies(df, columns=['origen', 'destino'], drop_first=False)
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18322 entries, 0 to 18321
Data columns (total 24 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      18322 non-null  int64
 1   fecha                   18322 non-null  datetime64[ns]
 2   numero_vuelo            18322 non-null  int64
 3   distancia               18322 non-null  int64
 4   inasistencia            18322 non-null  int64
 5   vuelo_denegado          18322 non-null  int64
 6   tarifa_mediabaja        18322 non-null  float64
 7   tarifa_alta             18322 non-null  int64
 8   tarifa_mediaalta        18322 non-null  int64
 9   tarifa_baja             18322 non-null  int64
 10  pax_freqflyer           18322 non-null  int64
 11  agendado_grupal         18322 non-null  int64
 12  sin_stock               18322 non-null  int64
 13  conexion_nacional       18322 non-null  int64
 14  conexion_internacional  18322 non-null  int64
 15  sin_conexion            18048 non-null  float64
 16  hora_salida             18317 non-null  object
 17  capacidad               18322 non-null  int64
 18  venta_usd               18322 non-null  float64
 19  agendados               17681 non-null  float64
 20  origen_ANF              18322 non-null  bool
 21  origen_SCL              18322 non-null  bool
 22  destino_ANF             18322 non-null  bool
 23  destino_SCL             18322 non-null  bool
dtypes: bool(4), datetime64[ns](1), float64(4), int64(14), object(1)
memory usage: 2.9+ MB
None
```

In [6]:
```
df.head().style.set_properties(**{'background-color': '#f0f0f0', 'color': 'black'})
```

Out[6]:

	id	fecha	numero_vuelo	distancia	inasistencia	vuelo_denegado	tarifa_mediabaja	tarifa_al
0	69922	2009-02-23 00:00:00	8942	1106	7	0	124.000000	
1	469723	2010-01-13 00:00:00	8941	1106	18	0	56.000000	
2	779308	2010-10-04 00:00:00	9128	1106	6	0	1.000000	
3	429392	2009-12-11 00:00:00	7941	1106	10	0	122.000000	
4	1286557	2011-11-20 00:00:00	9139	1106	8	0	1.000000	

c) La normalización o estandarización de los datos.

ESTANDARIZACIÓN usando StandardScaler

La estandarización deja todas las variables numéricas con media 0 y desviación estándar 1. También evita que alguna variable con un rango alto (como venta_usd) domine sobre las demás.

Variables a estandarizar: Todas las variables numéricas que usará el modelo como predictoras (EXCEPTO la variable objetivo inasistencia y las identificadoras como id o numero_vuelo).

Variables a estandarizar: distancia vuelo_denegado tarifa_mediabaja tarifa_alta tarifa_mediaalta tarifa_baja pax_freqflyer agendado_grupal sin_stock conexion_nacional conexion_internacional sin_conexion capacidad venta_usd agendados

In [7]:
```
from sklearn.preprocessing import StandardScaler

# Selección de columnas numéricas a estandarizar
cols_a_estandarizar = [
    'distancia', 'vuelo_denegado', 'tarifa_mediabaja', 'tarifa_alta',
    'tarifa_mediaalta', 'tarifa_baja', 'pax_freqflyer', 'agendado_grupal',
    'sin_stock', 'conexion_nacional', 'conexion_internacional',
    'sin_conexion', 'capacidad', 'venta_usd', 'agendados'
]

scaler = StandardScaler()
df[cols_a_estandarizar] = scaler.fit_transform(df[cols_a_estandarizar])
df.head().style.set_properties(**{'background-color': '#f0f0f0', 'color': 'black'})
```

Out[7]:

	id	fecha	numero_vuelo	distancia	inasistencia	vuelo_denegado	tarifa_mediabaja	tarifa_al
0	69922	2009-02-23 00:00:00	8942	0.000000	7	-0.095419	1.556561	0.40010
1	469723	2010-01-13 00:00:00	8941	0.000000	18	-0.095419	0.135902	-0.47639
2	779308	2010-10-04 00:00:00	9128	0.000000	6	-0.095419	-1.013160	-0.47639
3	429392	2009-12-11 00:00:00	7941	0.000000	10	-0.095419	1.514777	-0.12579
4	1286557	2011-11-20 00:00:00	9139	0.000000	8	-0.095419	-1.013160	-0.47639

2. Selección de Variables y Extracción de Atributos

Elija las variables más relevantes para predecir la inasistencia y realice una extracción de atributos de la fecha. Explique las razones para dejar algunas características fuera.

Comentarios:

- Describa las variables que eligió y por qué.
- Explique el proceso de extracción de atributos de la fecha y su relevancia para el modelo.

Selección de variables relevantes para predecir inasistencia

Variables incluidas:

distancia: Puede influir en la decisión de asistir; los vuelos más cortos pueden tener más inasistencias.

vuelo_denegado: El exceso de reservas puede estar relacionado con patrones de inasistencia.

tarifa_mediabaja, tarifa_alta, tarifa_mediaalta, tarifa_baja: El tipo de tarifa comprada puede correlacionar con la propensión a no asistir; tarifas bajas suelen ser menos flexibles y tal vez más propensas a inasistencia.

pax_freqflyer: Los pasajeros frecuentes pueden tener otro comportamiento respecto a la asistencia.

agendado_grupal: Las reservas grupales pueden presentar patrones distintos de inasistencia.

sin_stock: Indica vuelos con muy alta demanda, lo que puede afectar la proporción de no presentados.

conexion_nacional, conexion_internacional, sin_conexion: La conexión puede impactar la probabilidad de no presentarse al vuelo.

capacidad: Puede influir, ya que en vuelos con más capacidad puede haber diferentes políticas de reserva y asistencia.

venta_usd: Representa el ingreso del vuelo; puede correlacionarse indirectamente con la demanda y la inasistencia.

agendados: Número de reservas realizadas, directamente vinculado a la gestión de asientos e inasistencia.

origen_XXX, destino_XXX: Variables dummy del one-hot encoding de aeropuerto, porque la ruta origendestino puede ser relevante para la inasistencia.

Atributos extraídos de la fecha.

Variables excluidas y razones:

id, numero_vuelo: Son identificadores únicos y no contienen información relevante para la predicción.

fecha: No se usa directamente como variable continua, pero de ella se pueden extraer variables temporales de alto valor predictivo.

hora_salida: Es una variable temporal que debe ser procesada a un valor numérico (por ejemplo, minutos desde medianoche o extraer la hora solamente).<-- REVISAR

inasistencia: Es la variable objetivo, no se incluye como predictor.

Extracción de atributos de fecha

Proceso: Para aprovechar la información temporal, se extraen los siguientes atributos de la columna fecha:

Por qué es relevante:

Year: Permite capturar tendencias temporales a largo plazo o cambios de política o estacionalidad año a año.

Month: Permite capturar estacionalidad (meses de alta/baja) y patrones de vacaciones.

Day_of_week: Importante porque el comportamiento de reserva y asistencia varía mucho entre días laborales y fines de semana.

In [8]:
```
df['year'] = df['fecha'].dt.year
df['month'] = df['fecha'].dt.month
df['day'] = df['fecha'].dt.day
df['day_of_week'] = df['fecha'].dt.weekday  # 0 = lunes
df.head().style.set_properties(**{'background-color': '#f0f0f0', 'color': 'black'})
```

Out[8]:

	id	fecha	numero_vuelo	distancia	inasistencia	vuelo_denegado	tarifa_mediabaja	tarifa_al
0	69922	2009-02-23 00:00:00	8942	0.000000	7	-0.095419	1.556561	0.4001(
1	469723	2010-01-13 00:00:00	8941	0.000000	18	-0.095419	0.135902	-0.4763
2	779308	2010-10-04 00:00:00	9128	0.000000	6	-0.095419	-1.013160	-0.4763
3	429392	2009-12-11 00:00:00	7941	0.000000	10	-0.095419	1.514777	-0.1257
4	1286557	2011-11-20 00:00:00	9139	0.000000	8	-0.095419	-1.013160	-0.4763

Extraer la hora en formato numérico:

Esto permite observar si hay más inasistencias en ciertas franjas horarias.

In [9]:
```
df['hour'] = pd.to_datetime(df['hora_salida'], format='%H:%M:%S').dt.hour
df.head().style.set_properties(**{'background-color': '#f0f0f0', 'color': 'black'})
```

Out[9]:

	id	fecha	numero_vuelo	distancia	inasistencia	vuelo_denegado	tarifa_mediabaja	tarifa_al
0	69922	2009-02-23 00:00:00	8942	0.000000	7	-0.095419	1.556561	0.4001(
1	469723	2010-01-13 00:00:00	8941	0.000000	18	-0.095419	0.135902	-0.4763
2	779308	2010-10-04 00:00:00	9128	0.000000	6	-0.095419	-1.013160	-0.4763
3	429392	2009-12-11 00:00:00	7941	0.000000	10	-0.095419	1.514777	-0.1257
4	1286557	2011-11-20 00:00:00	9139	0.000000	8	-0.095419	-1.013160	-0.4763

Para evitar sesgo y asegurar que todas las variables numéricas estén en la misma escala, estandarizaremos las columnas year, month, day, day_of_week y hour..

In [10]:
```
from sklearn.preprocessing import StandardScaler

cols_nuevas = ['year', 'month', 'day', 'day_of_week', 'hour']
df[cols_nuevas] = StandardScaler().fit_transform(df[cols_nuevas])
df.head().style.set_properties(**{'background-color': '#f0f0f0', 'color': 'black'})
```

Out[10]:

	id	fecha	numero_vuelo	distancia	inasistencia	vuelo_denegado	tarifa_mediabaja	tarifa_al
0	69922	2009-02-23 00:00:00	8942	0.000000	7	-0.095419	1.556561	0.4001(
1	469723	2010-01-13 00:00:00	8941	0.000000	18	-0.095419	0.135902	-0.4763
2	779308	2010-10-04 00:00:00	9128	0.000000	6	-0.095419	-1.013160	-0.4763
3	429392	2009-12-11 00:00:00	7941	0.000000	10	-0.095419	1.514777	-0.1257
4	1286557	2011-11-20 00:00:00	9139	0.000000	8	-0.095419	-1.013160	-0.4763

3. Desarrollo y Evaluación del Modelo

Desarrolle uno o más modelos de regresión y evalúe el desempeño de los modelos mediante el error absoluto medio (MAE). Interprete correctamente los resultados y su impacto en la precisión del modelo.

Comentarios:

- Describa el modelo que eligió y por qué.
- Comente sobre los resultados obtenidos del MAE y su interpretación.
- Elija otra métrica de evaluación, interprete los resultados y compare con el MAE.

3.1 Selección de modelos a evaluar

Evaluaremos los siguientes tres modelos:

1. Regresión Lineal

Por qué: Proporciona una línea base sencilla y fácil de interpretar. Permite identificar rápidamente relaciones lineales en los datos y establecer un mínimo de referencia.

Modelo: LinearRegression de scikit-learn.

2. Árbol de Decisión para Regresión

Por qué: Permite capturar relaciones no lineales y efectos de interacción entre variables, que son habituales en fenómenos complejos como la inasistencia.

Modelo: DecisionTreeRegressor de scikit-learn.

3. Random Forest para Regresión

Por qué: Modelo robusto, reduce sobreajuste y ofrece excelente desempeño en problemas tabulares.

Modelo: RandomForestRegressor de scikit-learn.

Paso 1: Definición de variables predictoras (X) y variable objetivo (y):

y = columna inasistencia

X = todas las variables seleccionadas para predicción.

Paso 2: Separamos los datos en entrenamiento y test (80%/20%):

In [11]:
```
from sklearn.model_selection import train_test_split

X = df.drop(['inasistencia', 'id', 'numero_vuelo', 'fecha', 'hora_salida'], axis=1)
y = df['inasistencia']

for col in X.columns:
    if X[col].isnull().any():
        if X[col].dtype in ['float64', 'int64']:
            X[col] = X[col].fillna(X[col].median())
        elif X[col].dtype == 'bool':
            X[col] = X[col].fillna(False)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Paso 3: Entrenamiento y evaluación de cada modelo:

Regresión Lineal

In [12]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error

linreg = LinearRegression()
linreg.fit(X_train, y_train)
y_pred_linreg = linreg.predict(X_test)
mae_linreg = mean_absolute_error(y_test, y_pred_linreg)
```

Árbol de Decisión

In [13]:
```
from sklearn.tree import DecisionTreeRegressor

dtr = DecisionTreeRegressor(random_state=42)
dtr.fit(X_train, y_train)
y_pred_dtr = dtr.predict(X_test)
mae_dtr = mean_absolute_error(y_test, y_pred_dtr)
```

Random Forest

In [14]:
```
from sklearn.ensemble import RandomForestRegressor

rfr = RandomForestRegressor(n_estimators=100, random_state=42)
rfr.fit(X_train, y_train)
y_pred_rfr = rfr.predict(X_test)
mae_rfr = mean_absolute_error(y_test, y_pred_rfr)
```

Paso 4: Calcular Error cuadrático medio (MSE o RMSE). Esto porque penaliza más los errores grandes y da otra perspectiva sobre la dispersión de los errores.

In [15]:
```
rmse_linreg = np.sqrt(mean_squared_error(y_test, y_pred_linreg))
rmse_dtr = np.sqrt(mean_squared_error(y_test, y_pred_dtr))
rmse_rfr = np.sqrt(mean_squared_error(y_test, y_pred_rfr))
```

Resultados:

In [16]:
```
resultados = {
    'Modelo': ['Regresión Lineal', 'Árbol de Decisión', 'Random Forest'],
    'MAE': [mae_linreg, mae_dtr, mae_rfr],
    'RMSE': [rmse_linreg, rmse_dtr, rmse_rfr]
}

tabla_resultados = pd.DataFrame(resultados)

tabla_resultados.style.set_properties(**{'background-color': '#f0f0f0', 'color': 'black'})
```

Out[16]:

	Modelo	MAE	RMSE
0	Regresión Lineal	3.701673	5.197735
1	Árbol de Decisión	4.967531	7.020531
2	Random Forest	3.507539	4.849931

Interpretación de las métricas

MAE (Mean Absolute Error): Indica en promedio cuánto se equivoca el modelo en sus predicciones, en las mismas unidades que la variable objetivo (en este caso, el número promedio de no-shows). Un valor más bajo significa mayor precisión.

RMSE (Root Mean Squared Error): Penaliza errores grandes más que el MAE, ya que eleva al cuadrado las diferencias antes de promediarlas. Es especialmente útil cuando los errores grandes son costosos o indeseables para el negocio.

Interpretación:

Random Forest obtiene los valores más bajos tanto en MAE como en RMSE, lo que indica que es el modelo que comete menos errores en promedio y también menos errores grandes. Regresión Lineal está cerca en desempeño, pero se queda ligeramente por detrás del Random Forest. Árbol de Decisión es el que obtiene el peor desempeño, con errores significativamente mayores en promedio y también mayores errores extremos.

Recomendación:

Mejor modelo:

El Random Forest es la mejor opción, ya que obtiene los mejores resultados tanto en MAE como en RMSE. Esto significa que es el modelo más preciso y el que mejor mitiga los errores grandes, lo cual es especialmente valioso cuando estos pueden afectar de manera considerable la experiencia del cliente o la eficiencia operacional.

Equilibrio entre MAE y RMSE:

Si el principal objetivo de negocio es minimizar el error promedio en la predicción de no-shows (por ejemplo, para mejorar la planificación operativa del vuelo), el MAE es suficiente y Random Forest continúa siendo superior. Si los errores grandes son muy costosos (por ejemplo, si una sobreestimación grande puede causar pérdidas de ingresos o problemas logísticos graves), entonces el RMSE cobra aún más importancia, y nuevamente Random Forest resulta ser el más adecuado.

Conclusión

Se recomienda implementar el modelo Random Forest para la predicción de no-shows, ya que logra el mejor equilibrio entre precisión promedio y control de errores extremos. Es un modelo robusto que generalmente maneja mejor la heterogeneidad y las relaciones no lineales presentes en los datos de vuelos y pasajeros.

4. Reflexión Final

El desarrollo del modelo para predecir no-shows de pasajeros en AeroML ha sido un proceso de mucho aprendizaje y desafiante, permitiéndonos profundizar en las distintas etapas del ciclo de vida de un proyecto de machine learning y hemos experimentado de primera mano la importancia de un enfoque metódico y crítico en cada paso de la construcción de modelos predictivos.

Durante la etapa de preprocesamiento de datos, nos enfrentamos al reto de manejar datos incompletos y heterogéneos, especialmente en lo relativo a valores faltantes y diferentes tipos de variables (numéricas, categóricas, temporales). Decidir cómo imputar los datos faltantes, normalizar correctamente las variables y convertir las categorías en representaciones numéricas apropiadas (one-hot encoding) fue clave para asegurar que los modelos pudieran aprender patrones significativos sin verse afectados por problemas en la calidad de los datos originales.

En la selección y extracción de características, optamos por priorizar aquellas variables que, desde un punto de vista tanto lógico como exploratorio, tenían mayor potencial predictivo. Decidir excluir identificadores no predictivos o información redundante fue fundamental para evitar ruido y complejidad innecesaria en el modelo.

Al llegar al desarrollo y evaluación de modelos nos enfrentamos con varias dificultades técnicas, como las diferencias en el manejo de datos entre modelos y librerías, y la gestión de errores derivados de datos atípicos o incompatibilidades. Resolver estos problemas requirió investigar buenas prácticas, experimentar con diferentes enfoques y "debuggear" de manera paciente cada error hasta llegar a una solución adecuada.

La evaluación y selección de modelos resultó ser una etapa crítica. Utilizamos métricas complementarias (MAE y RMSE) para asegurarnos de que el modelo seleccionado no solo minimizara el error promedio, sino que también fuera robusto ante errores grandes que pudieran afectar el negocio. Comparando regresión lineal, árboles de decisión y random forest, el aprendizaje es claro: modelos más complejos y robustos como random forest pueden obtener ventajas considerables cuando los datos presentan relaciones no lineales y múltiples interacciones.

En conclusión este proceso nos ha reafirmado que la construcción de modelos de machine learning precisos y confiables depende tanto del conocimiento técnico como de una actitud reflexiva y crítica ante cada decisión. La evaluación rigurosa y la selección adecuada del modelo final son esenciales para asegurar no solo el rendimiento, sino también la utilidad real de la solución en un entorno de negocio.

Referencias

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd ed.). O'Reilly Media.

Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer. https://doi.org/10.1007/978-1-4614-6849-3