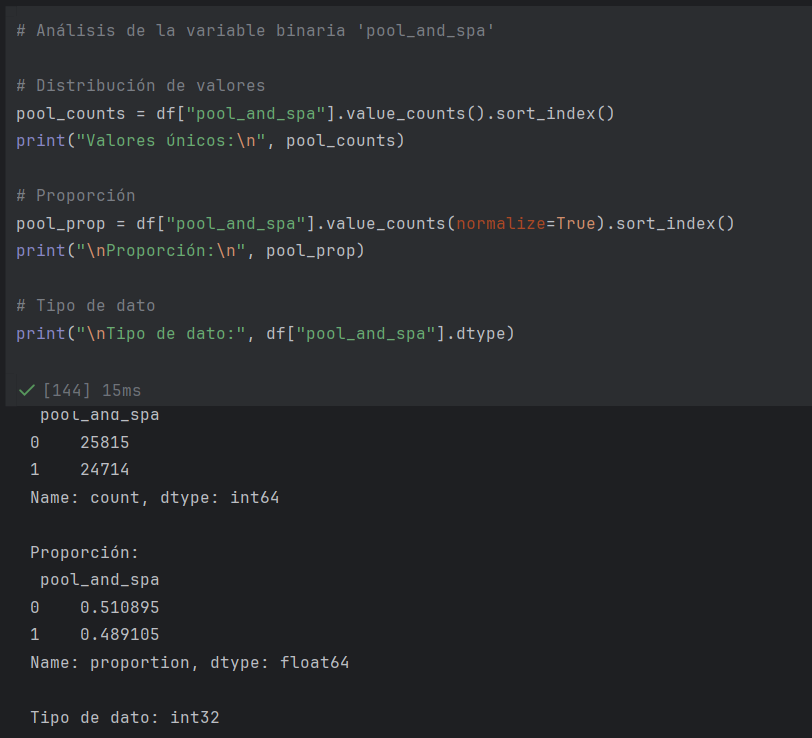
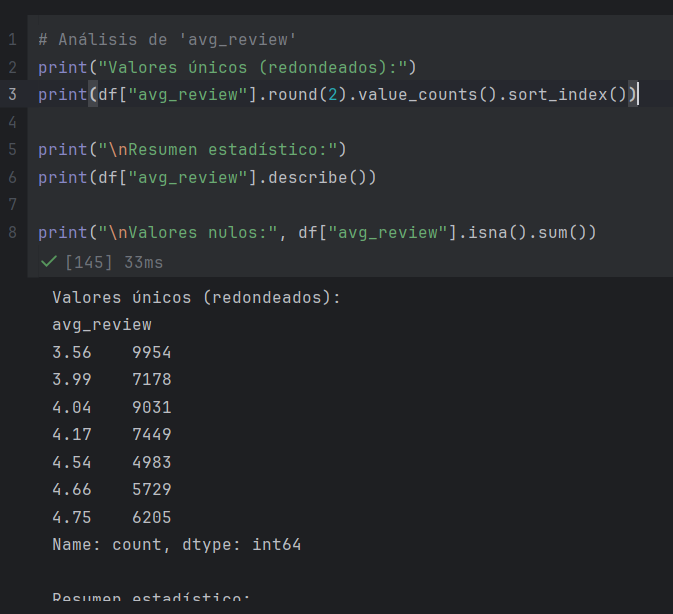
MEMORIA TRABAJO 2

Transformaciones iniciales:

Creamos la función base\_transform esta función aplica las transformaciones básicas necesarias para obtener un dataset coherente, limpio y preparado para el preprocesamiento y entrenamiento del modelo. Las decisiones tomadas en esta fase se basan en un análisis exploratorio manual de todas las variables, identificando outliers, errores, valores atípicos, categorías raras y problemas semánticos:

* Filtro de registro: eliminación de la variable booked. Se eliminan las reservas cuyo estado final es "Booked", ya que no se sabe aún si serán canceladas o no. Mantenerlas introduciría ruido en la variable objetivo, ya que la reserva esta activa, pero no completada.
* Conversión de fechas y creación de nueva variable, en este punto convertimos a datetime y creamos lead\_time, mide la antelación con la que se hace la reserva es importante para intentar predecir si se cancelara o no la reserva.
* Reducción de cardinalidad, que se agrupan categorías poco frecuentes para evitar una explosión de columnas tras OneHotEncoding y reducir el riesgo de **overfitting**
* Conversión de booleanos, se convierten a enteros para facilitar el uso en modelos lineales y evitar problemas de codificación.
* Para ver y valorar outliers analizamos una a una las columnas del df y vemos que hacer con cada una, ejemplos de algunos de los campos que analice, pero fueron las 23 columnas:





Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Algunos de los cambios que hice fueron:



La mayoría de los clientes solicitaban 0 o 1 plaza de parking,

En el análisis se detectaron valores como 6 o más, sin justificación clara y se decidió recortar a un máximo de 3 para evitar que estos valores extremos distorsionen el modelo.

Los valores nulos se imputaron con 0, ya que se interpreta que si no se indicó nada, lo más probable es que no se haya solicitado plaza



Se eliminaron las reservas con 0 noches de estancia, ya que representan inconsistencias o errores en la base de datos, analice que la mayoría eran reservas entre 1 y 7 noches asique quite las de mas de 21 para evitar casos extremos de 50-100 noches.



Decidí eliminar valores más allá de 210 días de antelación a la reserva ya que la mayoría de las reservas se hacían mucho menos de 6 meses de antelación.

Texto

El contenido generado por IA puede ser incorrecto.

También creamos estas dos nuevas variables para capturar patrones estacionales en las reservas y cancelaciones.

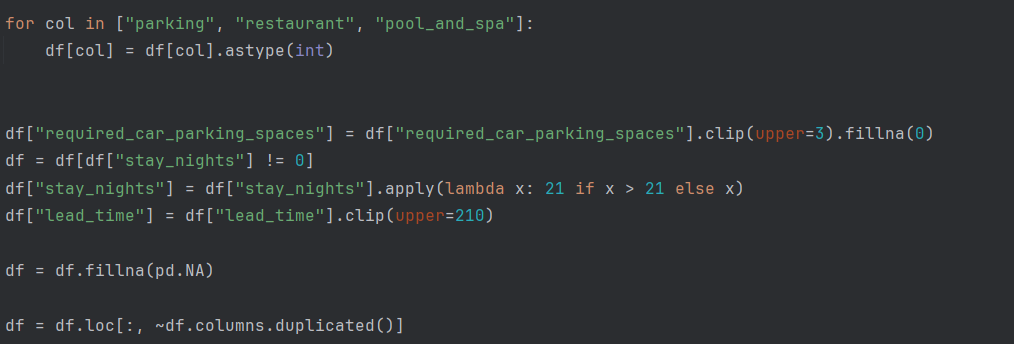
Algunos meses del año, como julio, agosto o diciembre, suelen tener más cancelaciones debido a la estacionalidad turística.

Además, el modelo puede aprender diferencias de comportamiento entre temporadas altas y bajas.

Texto

El contenido generado por IA puede ser incorrecto.

Muchas variables categóricas presentes en el dataset original contenían un número elevado de categorías distintas, e incluirlas supondría un numero altisimo de columnas tras el OneHotEncoding, mayor riesgo de overfiting y posibles problemas en la inferencia



Para terminar, hice una serie de transformaciones para dejar los datos limpios y coherentes.

Como reducir el numero de plazas de parking a 3, porq había outliers de reservas de muchas plazas una persona, también eliminar las reservas con 0 noches porque sería un fallo, recortar valores excesivos a 21 noches y las reservas con más de 210 días de antelación son inusuales y se consideran outliers.

Una vez definida la base\_transform pasamos a la creación de

**PREPROCESPIPE:**

-**Con eliminación de columnas:**

Identificadores (hotel\_id)

Fechas que ya se han utilizado para generar variables (arrival\_date, booking\_date, etc.)

La propia variable objetivo (reservation\_status

**-Agrupación de países (TopCountryGrouper)**

Se creó una clase personalizada que agrupa todos los países menos frecuentes bajo la categoría "Other", manteniendo solo los 5 países más frecuentes. Esto permite:

* Reducir la dimensionalidad del dataset tras el OneHotEncoding
* Mejorar la generalización del modelo
* Evitar ruido derivado de países con muy pocas reservas

**-Separación de transformaciones numéricas**

Se definieron dos grupos de columnas numéricas:

* **Variables donde el 0 tiene un significado válido** (required\_car\_parking\_spaces, special\_requests):
  + Se imputan con 0
  + Se escalan con StandardScaler
* **Otras numéricas**:
  + Se imputan con la mediana
  + También se escalan con StandardScaler

**-Transformaciones categóricas**

Las variables categóricas se imputan con "Missing" y se codifican con OneHotEncoder, eliminando la primera categoría para evitar redundancias (drop='first'). Además, handle\_unknown='ignore' garantiza que el sistema funcione correctamente en producción ante valores no vistos durante el entrenamiento.

**-Eliminación de columnas de baja varianza**

Se aplica un VarianceThreshold(threshold=0.01) para eliminar columnas que no aportan información relevante al modelo (por ejemplo, columnas constantes o casi constantes).

Y esta es la pipe final:

Texto

El contenido generado por IA puede ser incorrecto.

Después de definir todo vamos con la creación de la variable objetivo, de la **TARGET:**

El objetivo es predecir si una reserva será cancelada en los 30 días previos a la fecha de llegada del cliente.

Para ello, cree la función get\_target(df), que transforma el dataset original en una serie binaria (1 = cancelación, 0 = no cancelación reciente).

Con todo ya definido pasamos a:

**PRIMERA PRUEBA DEL MODELO: BASELINE CON REGRESIÓN LOGÍSTICA**

Tras preparar completamente los datos (X) y definir la variable objetivo (y), se procedió a realizar la primera prueba de modelado, usando una regresión logística básica integrada en un pipeline. Esta prueba sirve como modelo base (baseline) para comparar mejoras posteriores.

Calendario

El contenido generado por IA puede ser incorrecto.

Esta versión servirá como baseline: a partir de aquí se evaluarán mejoras con nuevas features, ajustes de hiperparámetros, técnicas de rebalanceo o calibración de thresholds.

**VERIFICACIÓN DE OVERFITTING: EVALUACIÓN SIMPLE CON REGRESIÓN LOGÍSTICA SIN AJUSTES**

Esta prueba tiene como objetivo analizar si el modelo tiende a sobreajustarse a los datos de entrenamiento.

Texto

El contenido generado por IA puede ser incorrecto.

**VALIDACIÓN CRUZADA CON STRATIFIEDGROUPKFOLD**

Para evaluar correctamente el rendimiento del modelo y evitar que este aprenda patrones específicos de un mismo hotel, se implementó una validación cruzada agrupada por hotel. Esto asegura que los datos de un mismo hotel no aparezcan tanto en entrenamiento como en test dentro de una misma fold, simulando mejor el escenario real en el que se quiere predecir sobre hoteles nuevos.

**Stratified**: mantiene la proporción de cancelaciones (clase 1) en cada fold.

**GroupKFold**: asegura que no se mezclen reservas del mismo hotel entre train y test.

Imagen de la pantalla de un celular con letras

El contenido generado por IA puede ser incorrecto.

Vemos la mejora del modelo, lo que quiere decir que vamos bien encaminados.

**INTEGRACIÓN DEL FEATUREENGINEER AL PIPELINE**

Tras validar el comportamiento del modelo base con StratifiedGroupKFold, se buscó mejorar la capacidad predictiva del modelo a través de la creación de nuevas variables derivadas. Para ello, se diseñó e integró una clase llamada FeatureEngineer.

Tabla

El contenido generado por IA puede ser incorrecto.

Esta etapa marca un punto de inflexión en el modelo, al enriquecer los datos con variables diseñadas específicamente para el contexto del negocio hotelero. La implementación en forma de clase dentro del pipeline permite mantener la modularidad, reproducibilidad y escalabilidad del proyecto.

Una vez definida, se integró FeatureEngineer como primer paso en el pipeline completo, justo antes del preprocesamiento tradicional:

Texto

El contenido generado por IA puede ser incorrecto.

**EVALUACIÓN DEL PIPELINE EXTENDIDO CON FEATUREENGINEER**

Una vez construida e integrada la clase FeatureEngineer con nuevas variables derivadas, se procedió a evaluar su impacto real en el rendimiento del modelo, entonces volvimos a usar stratifiedgroupkfold para ver como trataba la métrica y :

Imagen de la pantalla de un celular con letras

El contenido generado por IA puede ser incorrecto.

Vemos que le modelo mejoro, con que lo estamos haciendo bien

**EVALUACIÓN DE TÉCNICAS DE REBALANCEO PARA MEJORAR EL RENDIMIENTO EN LA CLASE MINORITARIA**

Tras incorporar variables derivadas mediante featureengeniere, se evaluó el impacto de distintas técnicas de rebalanceo sobre el rendimiento del modelo. El objetivo era comprobar si mejorar la representación de la clase minoritaria (reservas canceladas) podía ayudar al modelo a detectar mejores cancelaciones sin perder capacidad de generalización.

SMOTE

Accuracy Train: 0.6163 // Test: 0.6480

F1 Score Train: 0.4067 // Test: 0.3947

RandomOverSampler

Accuracy Train: 0.6138 // Test: 0.6302

F1 Score Train: 0.4070 // Test: 0.3968

ADASYN

Accuracy Train: 0.5905 // Test: 0.5880

F1 Score Train: 0.3786 // Test: 0.3162

No Rebalance

Accuracy Train: 0.8240 // Test: 0.8197

F1 Score Train: 0.0209 // Test: 0.0066

Al aplicar técnicas de sobremuestreo, como SMOTE o RandomOverSampler, el modelo sacrifica parte de esa accuracy, pero aumenta significativamente su capacidad para detectar cancelaciones, elevando el F1-score desde 0.0066 hasta alrededor de 0.39-0.40.

**TUNING DE HIPERPARÁMETROS Y OPTIMIZACIÓN DEL THRESHOLD**

Con el pipeline final ya integrado con ingeniería de variables, rebalanceo y validación robusta, se procedió a ajustar los hiperparámetros del modelo con el fin de maximizar su capacidad predictiva.

Además, se realizó una búsqueda del umbral de decisión (threshold) que maximiza el F1-score, dado que por defecto Scikit-learn usa un threshold de 0.5, lo cual no siempre es óptimo, especialmente en contextos desbalanceados.

**Búsqueda de hiperparámetros con RandomizedSearchCV**

Se exploraron tres tipos de configuraciones para la regresión logística:

1. **Sin penalización (penalty=None)**
2. **Con regularización L2** (ridge)
3. **Con regularización L1** (lasso)

Mejores hiperparámetros obtenidos:



**Optimización del threshold de clasificación**

Aunque los modelos clasifican por defecto con un threshold de 0.5, se evaluó si otro umbral podía **mejorar el F1-score**, usando las probabilidades de predicción (predict\_proba) del modelo optimizado.

Se probó cada threshold entre 0 y 1 con paso de 0.01 y se midió el F1-score correspondiente.

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.



Finalmente, se comparó el rendimiento del modelo usando:

* **Threshold por defecto (0.5)**
* **Threshold ajustado (óptimo encontrado)**

Calendario

El contenido generado por IA puede ser incorrecto.

Calendario

El contenido generado por IA puede ser incorrecto.

**ANÁLISIS DE LOS MEJORES MODELOS OBTENIDOS POR RANDOMIZEDSEARCHCV**

Tras ejecutar RandomizedSearchCV con múltiples configuraciones de hiperparámetros, se extrajeron los 10 modelos con mejor rendimiento medio en validación cruzada, ordenados por su mean\_test\_score (F1-score promedio)

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

* La **desviación estándar (std\_test\_score)** de los modelos top se mantiene en rangos moderados (≈0.04–0.05), lo que indica una **buena estabilidad** del rendimiento entre los distintos folds de validación cruzada.
* Todos los modelos top tienen valores de C entre 2.2 y 8.3, lo cual confirma que una penalización ligera (valores más altos de C) fue beneficiosa.
* La presencia recurrente de solvers como saga, lbfgs o newton-cg también valida que estas opciones fueron compatibles con el problema y los datos tratados.

**COMPARATIVA CON REGRESIÓN LOGÍSTICA ENTRENADA CON STOCHASTIC GRADIENT DESCENT**

Como último experimento, se entrenó una regresión logística mediante el clasificador SGDClassifier, que optimiza el modelo usando descenso de gradiente estocástico (Stochastic Gradient Descent, SGD).

El modelo se entrenó con el mismo pipeline de ingeniería de variables y preprocesamiento usado anteriormente, y se evaluó con las mismas métricas sobre el conjunto de test.

Tabla

El contenido generado por IA puede ser incorrecto.

El modelo entrenado con SGD obtuvo un rendimiento **ligeramente inferior** al modelo base optimizado con LogisticRegression, especialmente en precisión, aunque logró mantener un **recall alto**. Esta comparación confirma que el modelo con SGD podría ser una alternativa **válida y más eficiente computacionalmente**, pero en este caso, **no superó al modelo final optimizado** en términos de equilibrio general entre precisión y recall.

Por tanto, se mantuvo el modelo LogisticRegression como **opción definitiva para producción**, al ofrecer un mejor rendimiento tras tuning e incorporación del threshold óptimo.