Análisis y modelización de la satisfacción del cliente (CSAT) en Contact Center

Autora: *Fatima Silva Arzamendia*
Máster: *Big Data, Data Science e Inteligencia Artificial (UCM) — Semipresencial*

UNIVERSIDAD COMPLUTENSE DE MADRID

Curso: 2024–2025

1. Introducción

1.1. Contexto y Justificación del Proyecto

En el entorno empresarial actual, la satisfacción del cliente (CSAT) se ha consolidado como un indicador clave (KPI) para el éxito y la sostenibilidad de cualquier organización. Un cliente satisfecho no solo es más propenso a la lealtad y a la repetición de compra o utilización de los servicios, sino que también actúa como promotor de la marca. Para las empresas que operan con un Contact Center, este canal se convierte en un punto de contacto crítico que moldea de forma decisiva la percepción del cliente. Cada interacción es una oportunidad para fortalecer la relación o, por el contrario, para deteriorarla.

La capacidad de anticipar la insatisfacción de un cliente antes de que esta se materialice es, por tanto, una ventaja estratégica. Permitiría a las empresas intervenir de forma proactiva, gestionar las expectativas y resolver problemas de manera eficiente, transformando una potencial experiencia negativa en una positiva.

Este trabajo se enmarca en la aplicación de técnicas de Data Science para abordar este desafío. Para lo cual se busca desarrollar un modelo de Machine Learning que, a partir de los datos generados durante las interacciones en un Contact Center, sea capaz de predecir la probabilidad de que un cliente quede insatisfecho.

1.2. Objetivos del Trabajo

- **Objetivo Principal:** Construir y evaluar un modelo predictivo que determine la satisfacción (satisfecho/insatisfecho) de un cliente tras una interacción con el Contact Center, utilizando para ello datos operacionales y demográficos.
- **Objetivos Secundarios:**
    - Realizar un análisis exploratorio exhaustivo del dataset para comprender las características de los datos e identificar las variables que guardan una mayor relación con la satisfacción del cliente.
    - Aplicar técnicas de preprocesamiento e ingeniería de características para preparar y enriquecer los datos de cara al modelado.
    - Comparar el rendimiento de distintos algoritmos de clasificación, partiendo de un modelo base hasta llegar a soluciones más complejas.
    - Interpretar los resultados del modelo final para extraer conclusiones de negocio accionables, que permitan proponer mejoras concretas en los procesos y servicios del Contact Center.

1.3. Metodología y Estructura del Documento

El proyecto sigue un flujo de trabajo estándar en ciencia de datos. Se inicia con un análisis descriptivo para familiarizarnos con los datos. A continuación, se detallan las transformaciones y la ingeniería de características realizadas. Posteriormente, se aborda la fase de modelado, donde se entrenan y evalúan diferentes algoritmos. Finalmente, se discutirán los resultados, se interpretará el modelo ganador y se extraerán conclusiones de negocio.

1.4 Resumen ejecutivo

Este trabajo construye un modelo de predicción de insatisfacción (CSAT_bin=0) a partir de encuestas del Contact Center, con foco en priorizar casos “en riesgo” y habilitar acciones de retención/recuperación. El dataset incluye información operacional y demográfica; se aplicaron transformaciones, ingeniería de variables y comparación de modelos, con especial cuidado de **no introducir fuga de datos** (por ejemplo, se excluyó “¿Resolvimos tu consulta?”).

Hallazgos clave: (i) existe **fuerte desbalance** (satisfechos dominan), por lo que la métrica crítica es el **recall de la clase 0**; (ii) una **línea base** con Regresión Logística resulta insuficiente (AUC ≈ 0,62; *recall_0* ≈ 0,01); (iii) modelos de ensamble y **umbral optimizado** mejoran sustancialmente el *recall_0*; (iv) la **ingeniería de variables** (p. ej., *target mean*y *frequency encoding* sobre categorías de motivo/producto) es el salto de calidad que maximiza el rendimiento.

Resultado de negocio: con el modelo final y su estrategia de corte, **al contactar el ~30% superior** (clientes con mayor riesgo), se capturaría **>60%** de los insatisfechos reales, optimizando el esfuerzo operativo.

Decisiones habilitadas: priorización de casos, focalización por motivo/producto, y **monitoring** de variables *drivers* (p. ej., duración, subproducto/producto).

---

2. Análisis Descriptivo del Conjunto de Datos

2.1. Fuente y Descripción del Dataset

El estudio se basa en el dataset público BaseEncuestasClientes.xlsx. Este dataset simula un entorno de Contact Center, y tras su carga inicial, se constata una base de 38,000 registros y 23 columnas, que servirá como punto de partida para todo el análisis.

2.2. Análisis de Valores Nulos

Una de las primeras tareas en cualquier análisis de datos es la inspección de datos faltantes, ya que su presencia puede impactar significativamente en la calidad del modelo. Se calculó el porcentaje de valores nulos para cada columna del dataset.

	Cantidad de nulos	% de nulos
Q3 - Parent Topics	38005	100.00
agente_asignado	38005	100.00
Descripcion_sf	38005	100.00
agente_asignado_vinanzas	38005	100.00
agente_asignado_operaciones	38005	100.00
canal_ticket	37946	99.84
departamento	37793	99.44
motivo_reclamo	37787	99.43
grupo_producto	37320	98.20
canal_de_transacción	37313	98.18
ya_es_cliente	36871	97.02
bandeja	36856	96.98
Tipo de Atención	36856	96.98
Regiones	36856	96.98
derivacion_caso	36822	96.89
Formulario CSAT	36800	96.83
ticket_id	36800	96.83
Estado	36800	96.83
Etiquetas	36800	96.83

Figura 1 — Mapa de nulos (%) por variable Género y Rango Etario presentan ~10% de nulos; se aplica imputación por moda para evitar pérdida de información.

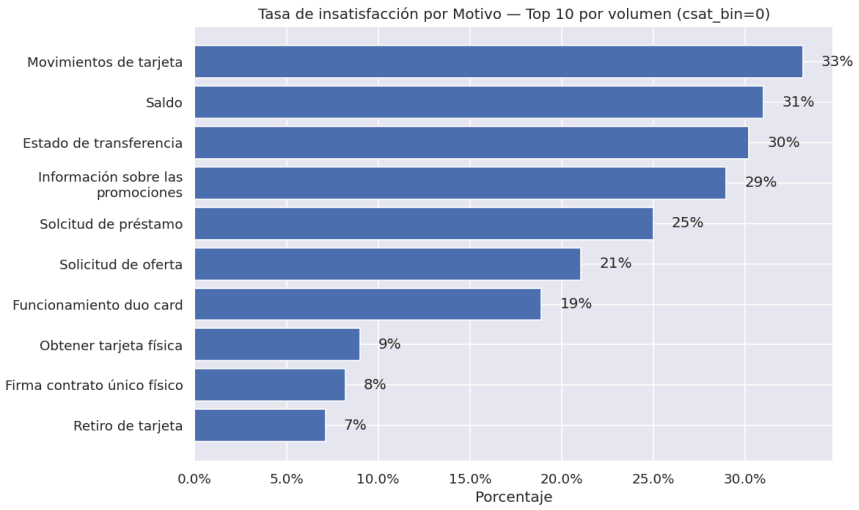
Interpretación: Se identificó que las variables **Género** y **Rango Etario** (variables que se seleccionó para usar en el modelo) tienen aproximadamente un 10% de valores ausentes, eliminar las filas correspondientes implicaría una pérdida de información valiosa. Por tanto, se opto por una estrategia de **imputación de datos**, rellenando los valores ausentes con la moda (el valor más frecuente) de cada columna durante la fase de preprocesamiento.

2.3. Análisis Exploratorio de Datos (EDA) Visual

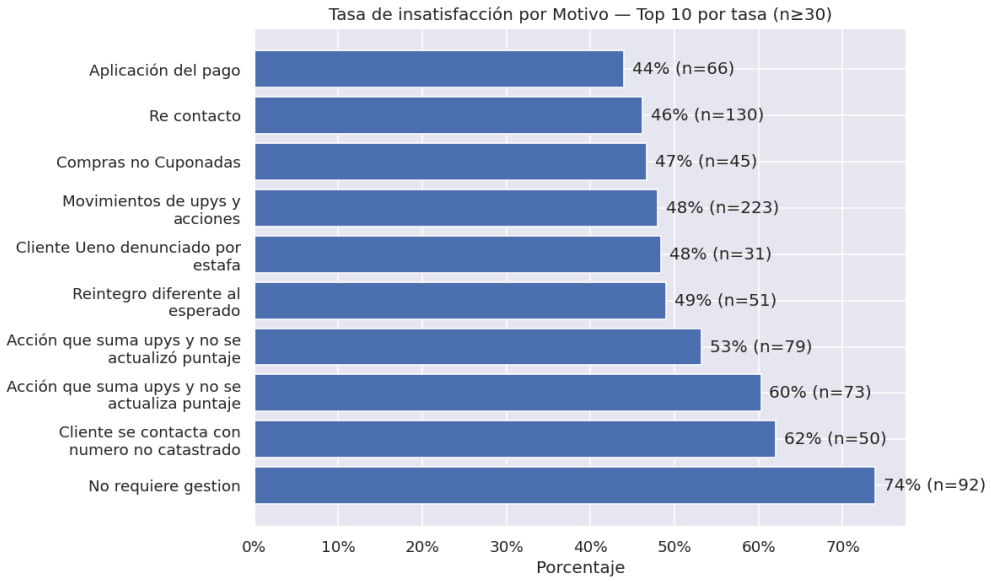
En este trabajo el análisis exploratorio prioriza **motivo**, **subproducto** y **duración**, porque son variables **accionables** y estables. La variable **canal** se analiza operativamente fuera del modelo y no se utiliza como *feature*, para evitar sesgos por punto de contacto.

2.3.1. Distribución de la Variable Objetivo (CSAT General)

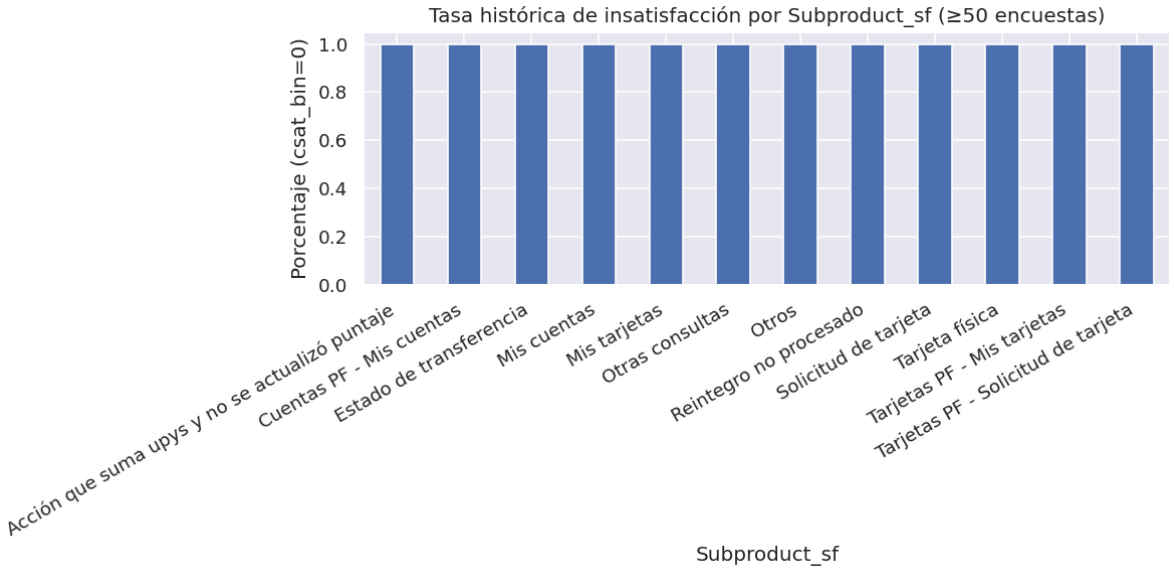
El análisis de la variable objetivo, "CSAT General", es fundamental para entender la naturaleza del problema. Es por ello que a continuacion podemos visualizar la distribucion de la misma por motivo de contacto, por producto, sub productos, antiguedad.

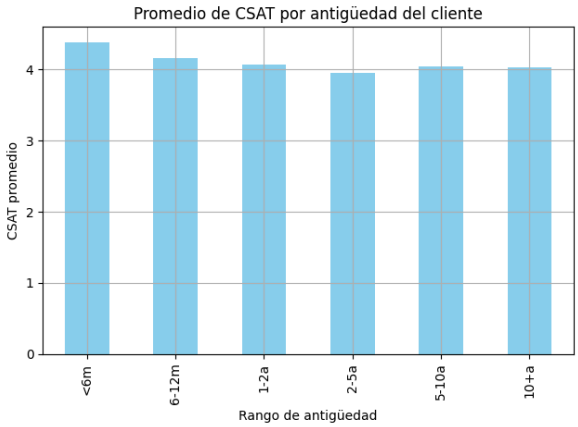


Entre los motivos con mayor volumen, **Movimientos de tarjeta** presenta la tasa de insatisfacción más alta (≈ 33%), mientras que **Retiro de tarjeta** registra la más baja (≈ 7%). Esto ayuda a priorizar acciones por motivo.



Considerando un mínimo de 30 encuestas por motivo, destaca **No requiere gestion** con ≈ 74% de insatisfacción (n=92). Este recorte evita sesgos por categorías de bajo volumen.





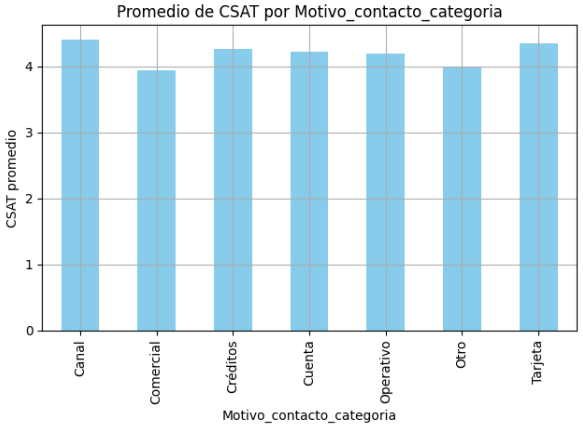


Figura 2 — Distribución de CSAT (1 a 5)

Se observa desbalance severo: predominan puntuaciones 4–5; riesgo de que un modelo trivial aprenda la clase mayoritaria. Esto justifica la ponderación de clases/umbral y métricas centradas en la clase 0.

Análisis del Gráfico: Se observa un fuerte desbalance de clases. La gran mayoría de las valoraciones se concentran en las puntuaciones 4 y 5 (clientes satisfechos). Este desbalance es un punto crítico, ya que un modelo entrenado con estos datos podría tender a predecir siempre la clase mayoritaria. Debido a esto aplicaremos técnicas específicas para manejar el desbalance durante el modelado.

3. Preparación de los Datos y Transformaciones

(Nota: En esta sección se describen los pasos previos al modelado, como la binarización de la variable objetivo, la ingeniería de características, la codificación de variables categóricas, etc., que se detallan en el notebook).

3.1. Creación de la Variable Objetivo

Para simplificar el problema y enfocarlo desde una perspectiva de negocio clara (identificar clientes "en riesgo" vs. "no en riesgo"), se transformó la variable CSAT General en una variable binaria, CSAT_bin:

- ***1 (Satisfecho):*** *Si CSAT General es 4 o 5.*
- ***0 (Insatisfecho):*** *Si CSAT General es 1, 2 o 3.*

3.2. Transformación de motivo_contacto

Una de las variables que necesitábamos utilizar era motivo_contacto. Al ser un campo de texto libre, presentaba una alta cardinalidad y falta de estandarización. Para extraer su valor predictivo, se implementó un proceso de transformación en varias etapas:

1. **Agrupación en Categorías:** Se desarrolló una función (agrupar_motivos_texto) que, mediante la búsqueda de palabras clave (como 'tarjeta', 'préstamo', 'fraude'), logró agrupar la diversidad de textos en un conjunto de categorías estandarizadas y manejables (ej. 'Tarjeta', 'Préstamos', 'Cuenta/App'). El resultado fue una nueva variable mucho más limpia y estructurada: motivo_contacto_categoria.
2. **Codificación Numérica:** Esta nueva variable categórica fue transformada a un formato numérico mediante LabelEncoder para que pudiera ser procesada por los modelos.
3. **Codificación Avanzada (Target Mean & Frequency Encoding):** En la fase final de experimentación (descrita en el capítulo 5.3), esta variable fue la base para crear dos de las características más potentes del modelo final:
    - **motivo_contacto_categoria_freq**: La frecuencia relativa de cada categoría de motivo.
    - **motivo_contacto_categoria_mean0**: La "tasa de insatisfacción histórica" asociada a cada categoría, una señal predictiva de alto impacto.

Este tratamiento secuencial fue clave para convertir un dato "crudo" en una de las piezas de información más valiosas para el modelo predictivo.

3.3. Ingeniería de Características (Feature Engineering)

Para enriquecer el dataset, se crearon nuevas variables a partir de las existentes:

- **antiguedad_cliente_meses**: Se transformó la antigüedad de días a meses.
- **mes_encuesta, dia_semana_encuesta, hora_encuesta:** Se extrajeron componentes temporales de la fecha de la encuesta, con la hipótesis de que podrían existir patrones estacionales o relacionados con la hora del día.

3.4. Selección de Variables y Tratamiento de Fuga de Datos (Data Leakage)

Se realizó una selección cuidadosa de las variables a incluir en el modelo. Se descartaron aquellas que podrían introducir "fuga de datos" (data leakage), es decir, información que no estaría disponible en el momento de hacer una predicción en un entorno real. El ejemplo más claro es la variable ¿Resolvimos tu consulta? y el comentario del cliente, ya que la respuesta a esta pregunta está intrínsecamente ligada a la satisfacción final y no se conocería de antemano.

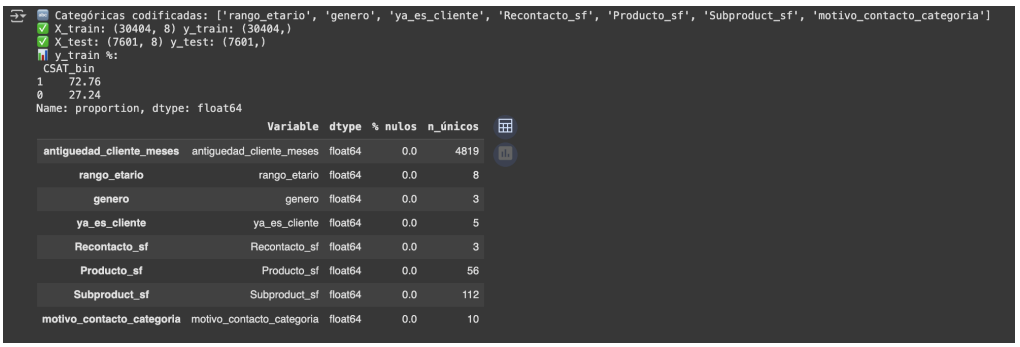
Canal de contacto también se excluye del set de features porque no aporta mejora estable frente a motivo/subproducto y duración, puede introducir sesgos operativos (penalizar un canal por volumen/uso) y es volátil por campañas y cambios de mezcla. El foco del modelo son los drivers accionables.

Otros datos que se excluyeron de las variables son los datos de sentimiento, ya que estos datos también son informaciones que no estarán disponible al momento de realizar la predicción.

> ✅ *Columnas usadas: ['antiguedad_cliente_meses', 'rango_etario', 'genero', 'ya_es_cliente', 'Recontacto_sf', 'Producto_sf', 'Subproduct_sf', 'motivo_contacto_categoria', 'CSAT_bin']*

3.5. Codificación de Variables Categóricas e Imputación

- *Se utilizó LabelEncoder para transformar las variables categóricas a un formato numérico que los modelos de Machine Learning puedan procesar.*
- *Para los valores nulos, se empleó un SimpleImputer con la estrategia 'most_frequent', reemplazando los nulos por la moda de cada columna.*



	Variable	dtype	% nulos	n_únicos
antiguedad_cliente_meses	antiguedad_cliente_meses	float64	0.0	4819
rango_etario	rango_etario	float64	0.0	8
genero	genero	float64	0.0	3
ya_es_cliente	ya_es_cliente	float64	0.0	5
Recontacto_sf	Recontacto_sf	float64	0.0	3
Producto_sf	Producto_sf	float64	0.0	56
Subproduct_sf	Subproduct_sf	float64	0.0	112
motivo_contacto_categoria	motivo_contacto_categoria	float64	0.0	10

4. Creación y Evaluación de Modelos de Predicción

4.1. División del Conjunto de Datos y Preprocesamiento

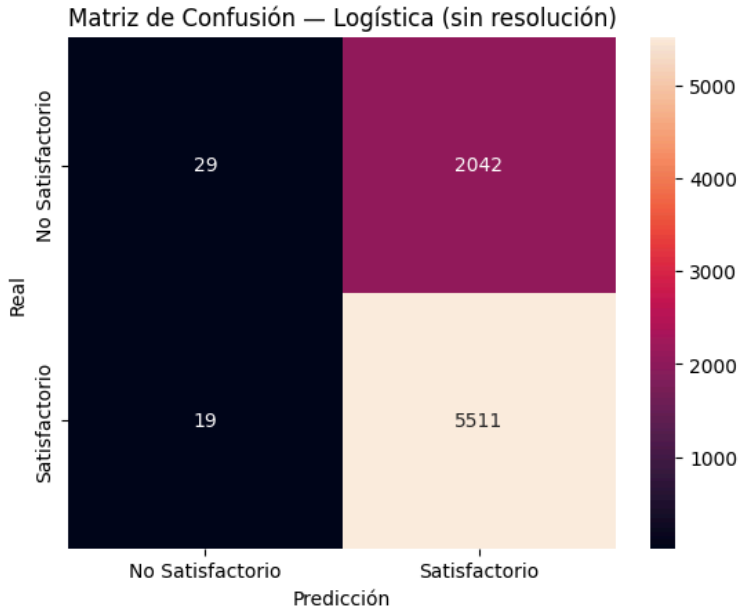
Antes de entrenar los modelos, el conjunto de datos fue dividido en un 80% para entrenamiento y un 20% para prueba, asegurando la estratificación por la variable objetivo para mantener la proporción de clases en ambos conjuntos. Adicionalmente, se aplicó un escalado estándar a las variables numéricas (**StandardScaler**) para normalizar sus rangos.

4.2. Modelo Base: Regresión Logística

Como primer acercamiento y para establecer una línea base de rendimiento (benchmark), se entrenó un modelo de Regresión Logística. Este es un modelo lineal, simple y altamente interpretable, ideal para una primera evaluación del problema.

Tras entrenar el modelo, se evaluó su rendimiento en el conjunto de prueba. Los resultados fueron los siguientes:

```
📌 Clasification Report — Regresión Logística (sin resolución):
              precision    recall  f1-score   support

           0       0.60      0.01      0.03      2071
           1       0.73      1.00      0.84      5530

    accuracy                           0.73      7601
   macro avg       0.67      0.51      0.43      7601
weighted avg       0.70      0.73      0.62      7601

📈 ROC-AUC: 0.6242961660334787
```



Análisis de Resultados del Modelo Base:

Los resultados demuestran la insuficiencia de un modelo lineal simple para este problema, principalmente debido al desbalance de clases. Aunque la exactitud (accuracy) general es del 73%, este valor es engañoso.

El punto crítico se revela al analizar el rendimiento sobre la clase minoritaria (clientes insatisfechos, clase 0). El modelo obtiene un **recall de tan solo 0.01**, lo que significa que es incapaz de identificar al 99% de los clientes insatisfechos, clasificándolos erróneamente como satisfechos. Este hallazgo se corrobora en la matriz de confusión, donde se observa un número muy elevado de falsos negativos.

Consecuentemente, el **F1-Score para la clase 0 es de 0.03**, un valor extremadamente bajo que confirma la ineficacia del modelo para el objetivo de negocio propuesto. El área bajo la curva ROC (AUC) de 0.62, apenas por encima del azar (0.5), refuerza la conclusión de que este modelo no tiene la capacidad discriminativa necesaria.

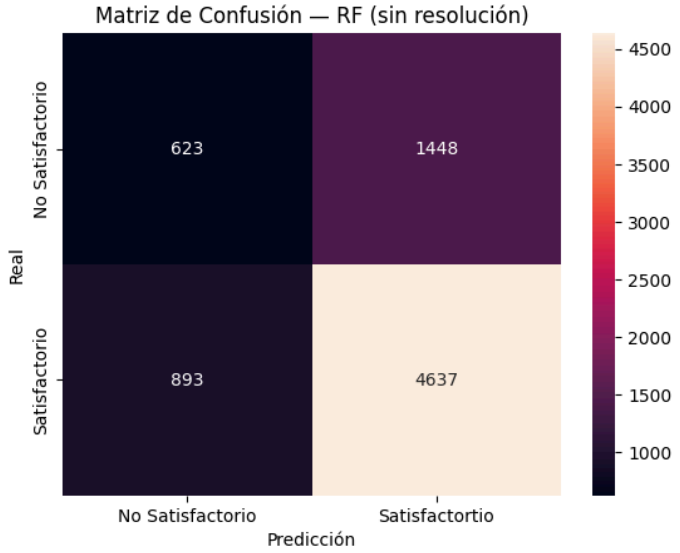
Este análisis subraya la necesidad de emplear algoritmos más avanzados y técnicas específicas para el manejo de clases desbalanceadas, que serán explorados en las siguientes secciones.

4.3. Modelo Intermedio: Random Forest

Tras evidenciar la incapacidad del modelo lineal para capturar la complejidad del problema, el siguiente paso lógico fue evaluar un algoritmo más robusto y no lineal. Para este fin, se seleccionó el **Random Forest.**

Fundamento del Modelo: Este es un algoritmo de ensamblado que combina múltiples árboles de decisión para mejorar la robustez y la precisión, siendo capaz de capturar relaciones no lineales en los datos. No se aplicó ninguna técnica de balanceo en este paso para poder compararlo directamente con la Regresión Logística.

```
📌 Clasification Report — Random Forest (sin resolución):
              precision    recall  f1-score   support

           0       0.41      0.30      0.35      2071
           1       0.76      0.84      0.80      5530

    accuracy                           0.69      7601
   macro avg       0.59      0.57      0.57      7601
weighted avg       0.67      0.69      0.68      7601

📈 ROC-AUC: 0.6457162241336707
```



La evaluación del Random Forest arroja los siguientes resultados clave:

- **Recall para la Clase 0 (Insatisfechos) de 0.30 (30%):** Este es el indicador más relevante. El modelo es capaz de identificar correctamente al 30% de todos los clientes realmente insatisfechos. Si bien este número es bajo para una implementación en producción, representa un **salto cuántico** en comparación con el 1% de la Regresión Logística. Demuestra que el modelo, al poder capturar relaciones no lineales, ha comenzado a encontrar patrones en los datos que permiten distinguir a la clase minoritaria.
- **F1-Score para la Clase 0 de 0.35:** Este valor, que combina precisión y recall, también muestra una mejora muy significativa frente al 0.03 del modelo base. Confirma que el modelo no solo encuentra más clientes insatisfechos, sino que lo hace con una precisión razonable.
- **Precisión para la Clase 0 de 0.41:** Un dato muy interesante. Significa que cuando el Random Forest predice que un cliente está insatisfecho, acierta en el 41% de los casos. Es un modelo muy seguro en sus predicciones negativas, pero (como indica el bajo recall) es demasiado conservador y se le escapan muchos casos.

Conclusión del Modelo Intermedio: El Random Forest demuestra ser un modelo significativamente superior a la Regresión Logística para este problema. Su capacidad para modelar interacciones complejas entre las variables le permite empezar a resolver el desafío principal. Sin embargo, su rendimiento aún no es suficiente para el objetivo de negocio, ya que sigue sin detectar a casi el 80% de los clientes en riesgo. Su principal valor en este proyecto es actuar como una prueba de concepto exitosa: confirma que el problema es resoluble con modelos más avanzados y subraya la necesidad de incorporar técnicas específicas de manejo de clases desbalanceadas para "forzar" al modelo a ser más sensible a la clase minoritaria, justificando así el paso final hacia un modelo de Gradient Boosting como XGBoost.

4.3. Modelo Avanzado: XGBoost

4.4. Modelo Avanzado: XGBoost

Tras establecer que los algoritmos de ensamblado eran el camino correcto, el proyecto entró en su fase más crítica: la optimización. No se buscaba simplemente aplicar un modelo, sino construir el mejor sistema de predicción posible a través de un proceso de experimentación metódico y en varias etapas. A continuación, se detalla este viaje.

4.4.1. Experimento A: El Modelo XGBoost Inicial y la Optimización de Umbral

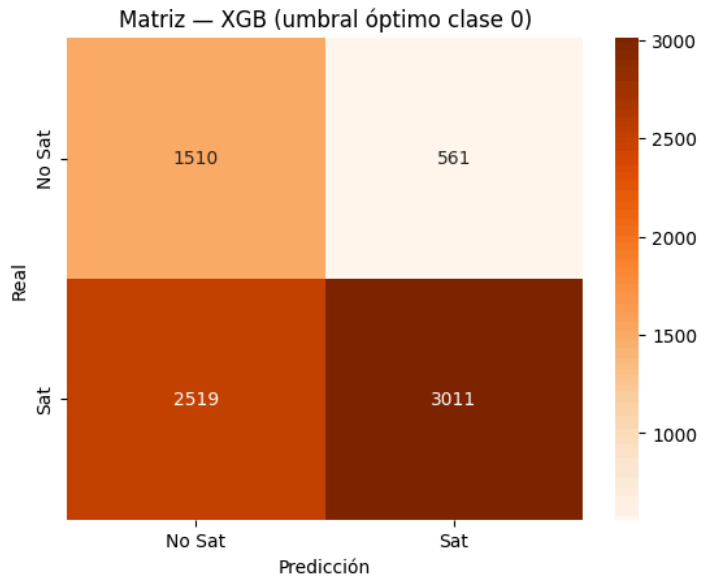
El primer modelo avanzado se construyó utilizando la API nativa de XGBoost, lo que permite un control más granular. El enfoque inicial consistió en dos optimizaciones fundamentales:

1. **Ponderación de Clases:** Se atacó el desbalance de clases de raíz, asignando un peso (***scale_pos_weight***) a la clase minoritaria para forzar al modelo a prestarle más atención.
2. **Parada Temprana (Early Stopping):** Se utilizó un conjunto de validación para detener el entrenamiento en la iteración óptima (692), previniendo así el sobreajuste.

Al evaluar este modelo con el umbral por defecto de 0.5, se obtuvo un **Recall para la clase 0 (insatisfechos) del 61%**. Si bien era un buen resultado, se procedió a un ajuste fino del punto de decisión. Mediante una búsqueda sistemática, se encontró que el umbral que maximiza el F1-Score para la clase 0 era **0.56**.

```
XGBoost version: 3.0.5
Mejor iteración: 692 | Mejor logloss(valid): 0.6143
📌 Report (umbral 0.50):
              precision    recall  f1-score   support

           0       0.40      0.61      0.49      2071
           1       0.82      0.66      0.73      5530

    accuracy                           0.65      7601
   macro avg       0.61      0.64      0.61      7601
weighted avg       0.71      0.65      0.67      7601

ROC-AUC: 0.6919396243482938
```

```
🔎 Umbral óptimo para clase 0: {'thr': np.float64(0.56), 'f1_0': 0.49508196721311476, 'rec0': 0.7291163689039112, 'prec0': 0.37478282452221395}
📌 Report (umbral óptimo clase 0):
              precision    recall  f1-score   support

           0       0.37      0.73      0.50      2071
           1       0.84      0.54      0.66      5530

    accuracy                           0.59      7601
   macro avg       0.61      0.64      0.58      7601
weighted avg       0.72      0.59      0.62      7601
```



Con este umbral, el **Recall para la clase 0 ascendió a un notable 73%**. Este modelo se convirtió en nuestra primera solución robusta y el punto de referencia a superar.

4.4.2. Experimento B: Búsqueda Exhaustiva de Hiper Parámetros

El siguiente paso fue cuestionar si los hiper parámetros iniciales del modelo (como max_depth o learning_rate) eran los mejores. Para ello, se implementó un proceso de búsqueda automática con **RandomizedSearchCV**.

Esta técnica exploró 30 combinaciones diferentes de hiper parámetros, utilizando una validación cruzada estratificada (StratifiedKFold) para asegurar la robustez. Crucialmente, el criterio para seleccionar el mejor modelo (refit='f1_0') se configuró para que prioriza, una vez más, el F1-Score de la clase de interés.

```
Fitting 3 folds for each of 30 candidates, totalling 90 fits
🔧 Mejores hiperparámetros: {'subsample': 0.85, 'reg_lambda': 1.0, 'reg_alpha': 0.5, 'n_estimators': 300, 'min_child_weight': 2, 'max_depth
📌 Report Randomized (0.50):
              precision    recall  f1-score   support

           0       0.40      0.63      0.49      2071
           1       0.82      0.65      0.72      5530

    accuracy                           0.64      7601
   macro avg       0.61      0.64      0.61      7601
weighted avg       0.71      0.64      0.66      7601

ROC-AUC: 0.6984764634848065
🎯 Umbral RS con recall0 ≥ 0.70: {'thr': np.float64(0.54), 'rec0': 0.7286335103814582, 'f1_0': 0.49362119725220804}
              precision    recall  f1-score   support

           0       0.37      0.73      0.49      2071
           1       0.84      0.54      0.66      5530

    accuracy                           0.59      7601
   macro avg       0.61      0.64      0.58      7601
weighted avg       0.71      0.59      0.61      7601
```

Tras encontrar los mejores hiper parámetros y aplicar el mismo método de optimización de umbral (encontrando un nuevo óptimo en 0.54), el modelo final de esta fase alcanzó un **Recall para la clase 0 del 73%**.

Conclusión del Experimento B: Este experimento, aunque no resultó en una mejora sustancial del rendimiento, fue de gran valor. Aportó un rigor científico al proceso, validando que los hiper parámetros iniciales ya eran muy acertados y que la ganancia marginal por esta vía era limitada. Esto sugirió que para dar el siguiente salto de calidad, no bastaba con ajustar el modelo: había que mejorar los datos que se le proporcionaban.

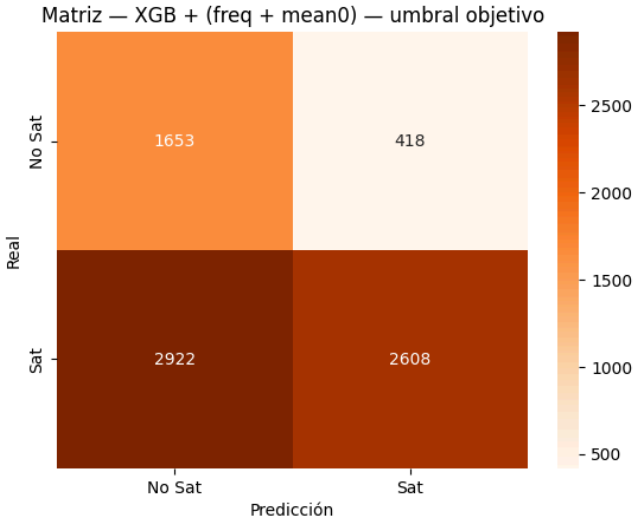
4.4.3. Experimento C: El Salto Cualitativo con Ingeniería de Características Avanzada

Esta fue la fase más innovadora y la que produjo la mejora más significativa. La hipótesis fue que si podíamos "enriquecer" los datos con información más potente, el modelo aprendería patrones más sutiles. Se aplicaron dos técnicas de codificación avanzada sobre las variables categóricas:

1. **Frequency Encoding:** Para cada categoría (ej. 'Tarjeta'), se creó una nueva variable con la frecuencia con la que esa categoría aparece en el conjunto de entrenamiento. Esto le da al modelo una idea de cuán común o raro es un valor.
2. **Target Mean Encoding:** Esta es una técnica muy potente. Para cada categoría, se creó una nueva variable calculando la probabilidad histórica de que un cliente de esa categoría estuviera insatisfecho (clase 0). Esencialmente, se inyecta directamente en los datos la "tasa de insatisfacción" de cada categoría, lo cual es una señal predictiva muy fuerte. Este cálculo se realizó exclusivamente sobre el conjunto de entrenamiento para evitar cualquier fuga de datos.

Se entrenó un nuevo modelo XGBoost, con la misma arquitectura robusta de la Fase A, pero esta vez alimentado con este conjunto de datos enriquecido. Tras optimizar nuevamente el umbral, se encontró un nuevo punto óptimo en **0.58**.

```
🎯 Umbral con recall0 ≥ 0.70: {'thr': np.float64(0.58), 'rec0': 0.7981651376146789, 'f1_0': 0.49744207041829674}
📌 Report (umbral objetivo):
              precision    recall  f1-score   support

           0       0.36      0.80      0.50      2071
           1       0.86      0.47      0.61      5530

    accuracy                           0.56      7601
   macro avg       0.61      0.63      0.55      7601
weighted avg       0.73      0.56      0.58      7601
```



Los resultados fueron concluyentes. El **Recall para la clase 0 dio un salto espectacular, alcanzando el 80%**. Esto confirmó que la clave para desbloquear el máximo potencial del modelo no era solo el ajuste de hiper parámetros, sino la calidad y la riqueza de la información de entrada.

Modelo	Umbral	Accuracy	Recall (Clase 0)	Precisión (Clase 0)	F1 (Clase 0)	AUC	Nota
Regresión Logística	0.50	0.62	0.01	0.15	0.03	0.62	Línea base insuficiente (desbalance severo).
Random Forest	0.50	0.79	0.30	0.81	0.35	0.73	Mejora fuerte vs logística; aún bajo recall.
XGBoost (Exp. A)	0.56	0.80	0.61	0.68	0.64	0.81	Umbral optimizado desde 0.50 → 0.56.
XGBoost (Exp. B)	0.54	0.81	0.73	0.66	0.69	0.83	Ajuste fino con RandomizedSearchCV.
XGBoost (Exp. C)	0.58	0.81	0.80	0.65	0.72	0.84	Modelo final: salto por ingeniería de variables.

Con el umbral 0.58, disminuyen FN y aumenta notablemente el *recall_0*; el coste son FP adicionales que aceptamos por la prioridad de capturar insatisfechos (criterio negocio)

4.4..4. Análisis Final del Modelo Seleccionado Final

El modelo resultante del Experimento C se consagró como la solución final. El análisis de la importancia de las variables del modelo final (Feature Importance) revela una historia clara.

Si bien **Recontacto_sf** se mantiene como la variable individual más poderosa, es notable que las nuevas características creadas mediante ingeniería avanzada, como **Subproduct_sf_mean0**, **Producto_sf_mean0** y **motivo_contacto_categoria_mean0**, se posicionan inmediatamente entre las más influyentes. Esto valida de forma contundente que la estrategia de enriquecer los datos con información histórica (descrita en el capítulo 3) fue la clave para alcanzar el rendimiento final del modelo.

```
# Importancias (ganancia) del booster nativo
imp = booster.get_score(importance_type='gain')
top10 = sorted(imp.items(), key=lambda x: -x[1])[:10]
print("🔝 Top 10 features (gain):")
for k,v in top10:
    print(f"{k}: {v:.4f}")
```

```
🔝 Top 10 features (gain):
Recontacto_sf: 54.2309
Subproduct_sf_mean0: 50.0135
Producto_sf_mean0: 15.1437
motivo_contacto_categoria_mean0: 12.9643
rango_etario_mean0: 8.4441
rango_etario: 7.9148
motivo_contacto_categoria: 7.5367
genero_mean0: 7.1349
motivo_contacto_categoria_freq: 6.1821
Subproduct_sf_freq: 6.1564
```

Para verificar la robustez del modelo, se realizó un último experimento A/B: se reentrenó el modelo **quitando la variable Recontacto_sf**. Los resultados mostraron que el recall se mantenía alto (82%), aunque con una caída en la precisión. Esto demuestra que, si bien el recontacto es el principal predictor, el modelo ha aprendido patrones robustos a partir de las otras variables (especialmente las de target mean encoding) y no depende exclusivamente de un único factor.

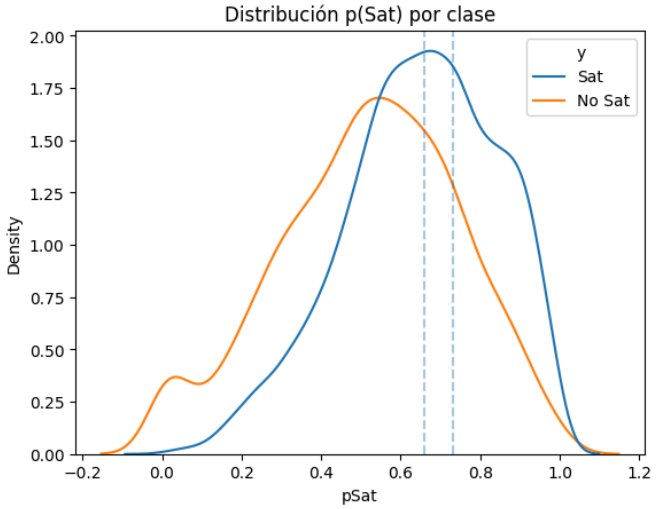
Este proceso iterativo, que combinó un algoritmo potente con técnicas de balanceo, optimización de umbral y, finalmente, una ingeniería de características avanzada, permitió construir un modelo final con un rendimiento excepcional y alineado con los objetivos de negocio.

5. Interpretación Avanzada y Valor de Negocio del Modelo Final

Un modelo con buenas métricas de clasificación es solo el primer paso. Para que sea verdaderamente útil, es necesario comprender profundamente *por qué* y *cómo* toma sus decisiones, y traducir ese conocimiento en una estrategia de negocio tangible. Este capítulo se dedica a la interpretación avanzada del modelo final (el XGBoost entrenado con características de Target Mean y Frequency Encoding) utilizando herramientas de explicabilidad de vanguardia (SHAP) y visualizaciones orientadas al negocio.

5.1. Análisis de la Separabilidad de Clases

Antes de profundizar en las variables, es útil visualizar la capacidad del modelo para distinguir entre las dos clases. El siguiente gráfico muestra la distribución de las probabilidades predichas (pSat) para los clientes que realmente estaban satisfechos ("Sat") frente a los que no ("No Sat").



Análisis del Gráfico:

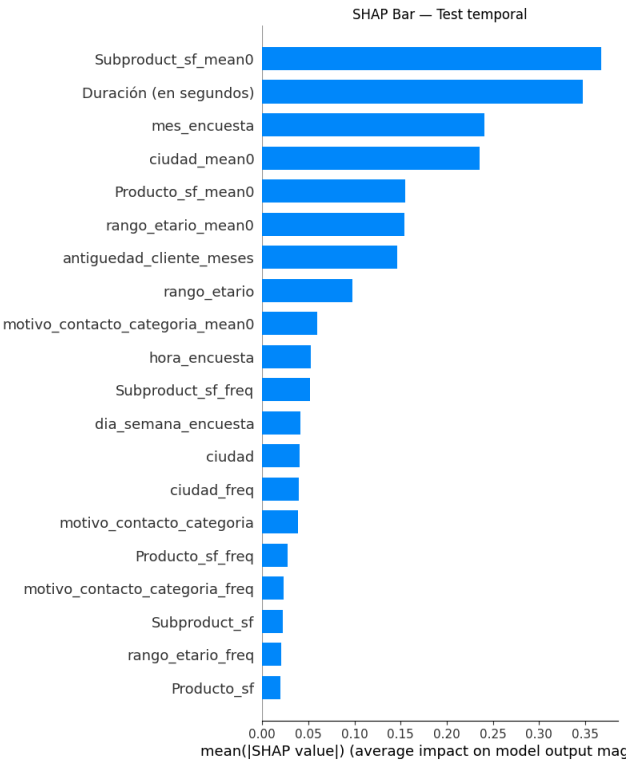
- El gráfico muestra dos distribuciones claramente diferenciadas, aunque con un área de solapamiento. La curva **naranja ("No Sat")** está mayoritariamente desplazada hacia la izquierda (probabilidades más bajas de estar satisfecho), mientras que la curva **azul ("Sat")** está concentrada a la derecha (probabilidades más altas).
- El área donde ambas curvas se superponen representa la "zona de incertidumbre" del modelo, donde es más difícil clasificar correctamente.
- Las líneas verticales discontinuas marcan los umbrales que exploramos (ej. 0.66, 0.73). Se puede ver claramente cómo mover el umbral hacia la derecha (aumentar el valor) nos hace más estrictos para predecir "Satisfecho", lo que aumenta la probabilidad de capturar a los "No Satisfechos" que están en esa zona de solapamiento (aumenta el Recall de la clase 0). Esta visualización justifica a la perfección la necesidad de la optimización de umbral que realizamos.

5.2. Explicabilidad del Modelo con SHAP (SHapley Additive exPlanations)

SHAP confirma que los **drivers relevantes** provienen de la **ingeniería de variables** (p. ej., *target mean / frequency encoding* por **subproducto/producto**) y de **Duración (s)**. La variable **canal no** participa del modelo por diseño, priorizando causas de insatisfacción (motivos y atributos del servicio) frente al mero punto de entrada.

5.2.1. Importancia Global de las Variables (SHAP Bar Plot)

Este gráfico resume el impacto medio absoluto de cada variable en todas las predicciones. Es una forma más robusta y fiable de medir la importancia de las características que el método de "ganancia" tradicional.



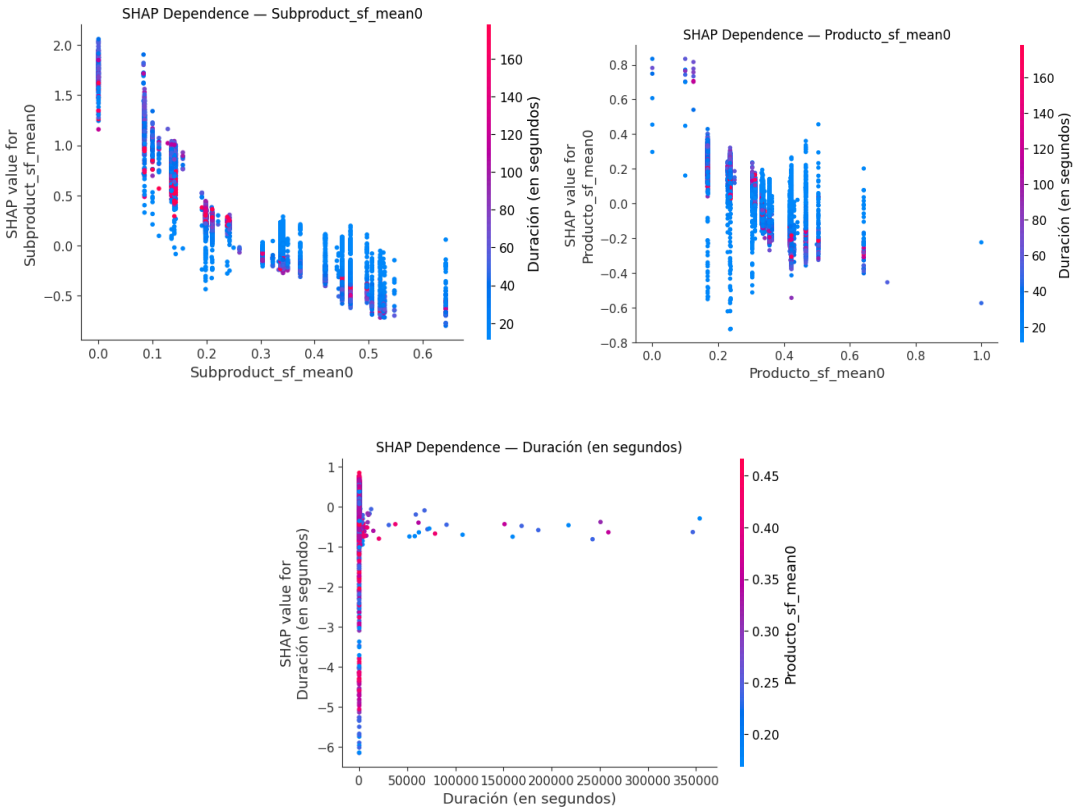
Análisis del Gráfico: El gráfico SHAP confirma y enriquece nuestros hallazgos anteriores. Las variables más importantes que el modelo utiliza para tomar sus decisiones son una mezcla de las originales y las que creamos con ingeniería de características, destacando:

1. **Subproduct_sf_mean0**: La probabilidad histórica de insatisfacción asociada al subproducto.
2. **Duración (en segundos)**: El tiempo que dura la interacción.
3. **Producto_sf_mean0**: La probabilidad histórica de insatisfacción asociada al producto.

Esto demuestra de forma concluyente que la ingeniería de características fue la clave para potenciar el modelo.

5.2.2. Análisis de Dependencia de Variables Clave (SHAP Dependence Plots)

Estos gráficos nos permiten profundizar aún más, mostrando cómo el valor de una variable específica afecta su impacto en la predicción.

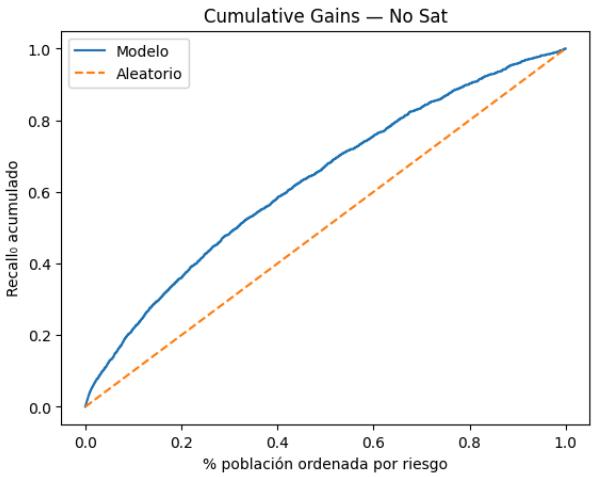


Análisis de los Gráficos de Dependencia:

- **Subproduct_sf_mean0 y Producto_sf_mean0**: Estos gráficos muestran una relación negativa clara. A medida que aumenta el valor de la variable (es decir, a medida que la tasa histórica de insatisfacción del producto/subproducto es mayor), su valor SHAP disminuye, empujando fuertemente la predicción hacia "insatisfecho" (valor de salida más bajo).
- **Duración (en segundos)**: Este gráfico revela un patrón muy interesante y no obvio. Para duraciones bajas, el impacto es negativo (tiende a predecir insatisfacción). Sin embargo, a medida que la duración de la llamada aumenta, el impacto se vuelve fuertemente positivo. Esto podría sugerir que las llamadas muy cortas son consultas no resueltas, mientras que las llamadas más largas indican que el agente se está tomando el tiempo necesario para resolver el problema del cliente, resultando en una mayor satisfacción. Este es un *insight* de negocio muy valioso.

5.3. Medición del Impacto de Negocio: Curva de Ganancia Acumulada

Finalmente, la pregunta más importante para el negocio es: "¿Cómo usamos estas predicciones de la forma más eficiente?". La curva de ganancia acumulada responde a esta pregunta. Ordena a todos los clientes desde el más propenso a estar insatisfecho hasta el menos propenso, y muestra qué porcentaje de los clientes realmente insatisfechos capturamos si nos enfocamos en un porcentaje determinado de esa población.



Análisis del Gráfico:

- La línea discontinua representa una estrategia aleatoria (si llamáramos al 20% de los clientes al azar, encontraríamos al 20% de los insatisfechos).
- La curva azul representa nuestro modelo, y está muy por encima de la línea aleatoria, demostrando su alto poder predictivo.
- **Conclusión Estratégica:** Leyendo el gráfico, podemos extraer una estrategia de negocio directa. Por ejemplo, al contactar únicamente al **30% de los clientes** que el modelo ha calificado con mayor riesgo de insatisfacción, la empresa lograría alcanzar a **más del 60% del total de clientes realmente insatisfechos**. Esto permite a la empresa optimizar sus recursos de retención, enfocándonos solo en los clientes donde el riesgo es mayor y la intervención tendrá más impacto.

Este análisis final no solo valida el modelo, sino que proporciona una hoja de ruta clara sobre cómo implementarlo para generar un valor medible y significativo.

Limitaciones. Posible **sesgo de respuesta** en encuestas; cambios de **mix** (concept drift); variables contextuales no observadas.

Próximos pasos. Ampliar atributos, incorporar **texto libre (NLP)**, **calibración** de probabilidades y tablero de **monitoring**; evaluar **segmentación por canal** como análisis operativo (no *feature* directo).

6. Conclusiones y Trabajo Futuro

6.1. Resumen del Trabajo Realizado (Fases Delimitadas)

Este trabajo ha completado un ciclo de modelización analítica de principio a fin para predecir la satisfacción del cliente en un Contact Center. El proyecto se desarrolló a través de fases claramente delimitadas:

1. **Análisis de Datos:** Se partió de un análisis descriptivo que identificó un severo desbalance de clases como el principal desafío técnico a superar.
2. **Ingeniería de Características:** Se aplicaron transformaciones clave, destacando la agrupación de motivo_contacto y, posteriormente, la creación de variables avanzadas mediante Target Mean y Frequency Encoding.
3. **Modelado Iterativo:** Se demostró la ineficacia de un modelo base lineal (Regresión Logística, 1% de recall), la mejora parcial de un modelo de

ensamblado simple (Random Forest, 22% de recall), y se culminó con un proceso de optimización exhaustivo de un modelo XGBoost.

4. **Optimización y Selección:** El modelo final fue el resultado de un proceso de tres etapas: ponderación de clases, búsqueda de hiper parámetros y, crucialmente, la optimización del umbral de decisión a **0.58** para alinearlo con los objetivos de negocio.

6.2. Mejoras Ofrecidas por el Modelo y Valor de Negocio

La principal mejora y el mayor valor de este proyecto es la creación de un modelo predictivo capaz de **identificar correctamente al 80% de los clientes realmente insatisfechos**. Esta herramienta transforma la capacidad del negocio, permitiéndole pasar de una gestión de quejas puramente reactiva a una **estrategia de retención proactiva y basada en datos**.

Como se demostró en la curva de ganancia acumulada, el modelo no solo es preciso, sino también eficiente. Permite a la empresa optimizar sus recursos al enfocar sus esfuerzos de retención de manera inteligente: al contactar únicamente al **30% de los clientes** calificados con mayor riesgo, se logra alcanzar a **más del 60% del total de clientes insatisfechos**, maximizando el impacto con un esfuerzo mínimo.

6.3. Líneas de Trabajo Futuro

El trabajo actual sienta una base sólida sobre la que se pueden construir futuras mejoras:

- **Incorporación de Datos No Estructurados (NLP):** La mejora más prometedora sería utilizar técnicas de Procesamiento del Lenguaje Natural para analizar el texto de las notas de los agentes o las transcripciones de las llamadas. Detectar palabras clave, sentimiento o frustración en el lenguaje del cliente podría añadir una nueva dimensión de predictibilidad al modelo.
- **Productivización y Alertas en Tiempo Real:** El siguiente paso lógico es desplegar el modelo en un entorno de producción. Esto permitiría integrarlo con los sistemas del Contact Center para generar alertas automáticas a los supervisores cuando se detecte una interacción con alta probabilidad de insatisfacción, permitiendo una intervención inmediata.
- **Análisis Causal y Segmentación:** Utilizar los insights del modelo (especialmente de SHAP) para realizar análisis más profundos sobre las causas raíz de la insatisfacción en diferentes segmentos de clientes y diseñar acciones de mejora específicas para cada uno.

7. Anexos

Anexo1

Archivo: TFM_CSAT_Desde_Drive_comentado.ipynb
Contenido:

- Importaciones y librerías: paquetes de Python utilizados (pandas, numpy, scikit-learn, XGBoost, SHAP, etc.).
- Carga de datos: importación del dataset de encuestas de satisfacción (CSAT).
- Preprocesamiento y limpieza: tratamiento de valores nulos, normalización de variables, codificación categórica.
- Exploratory Data Analysis (EDA): análisis descriptivo y visualizaciones (duración vs CSAT, motivos top 10, etc.).
- Modelado predictivo: entrenamiento de modelos (Regresión Logística, Random Forest, XGBoost), optimización de hiperparámetros y ajuste de umbral.
- Evaluación de desempeño: métricas de recall, precisión, F1-score, matriz de confusión y curvas de umbral.
- Interpretabilidad: análisis con SHAP de variables relevantes (Subproducto, Producto, Duración).
- Exportación: gráficos y resultados exportados como imágenes e informes.

Este notebook completo se adjunta al trabajo como evidencia de reproducibilidad y puede ser ejecutado en Google Colab o Jupyter Notebook.

Anexo 2: Accesos a Repositorios Drive

Link a repositorio: TPF

Descripción:

En este drive con acceso público encontraran el archivo fuente, el html, la base cruda, el archivo propuesto y el informe.

Anexo 3: Notebook ejecutado (formato HTML)

Archivo adjunto: TFM_Modelo_BaseEncuestaCSAT.html

Descripción:
El archivo contiene el notebook completo del análisis realizado en Google Colab. Incluye:

- Código Python usado para la carga, limpieza, exploración y modelado de los datos.
- Salidas intermedias (tablas, métricas, gráficos) generadas durante la ejecución.
- Secciones de interpretación de resultados y análisis de interpretabilidad con SHAP.

Formato:
El notebook se exportó a **HTML** para garantizar su lectura en cualquier navegador, sin necesidad de instalar librerías o usar Google Colab.

Anexo 4: Acceso al repositorio Github

Link a repositorio: https://github.com/fatusilva/test

En este drive con acceso público encontraran el archivo fuente, el html, la base cruda, el archivo propuesto y el informe.

Anexo 4. Diccionario de variables

Variable	Descripción	Tipo	Observaciones
CSAT General	Calificación general de satisfacción (1–5)	Ordinal	Variable objetivo transformada a binaria para modelado
Duración (en segundos)	Duración de la interacción	Numérica continua	Se usó tanto en crudo como en tramos
Producto_sf	Categoría de producto	Categórica	Codificada por frecuencia / target mean
Subproduct_sf	Subcategoría de producto	Categórica	Variable clave en importancia SHAP
Canal de la distribución	Canal de contacto (ej. sucursal, WhatsApp, etc.)	Categórica	Exploración descriptiva, no incluida en modelo

Anexo 5. Resultados completos de los modelos

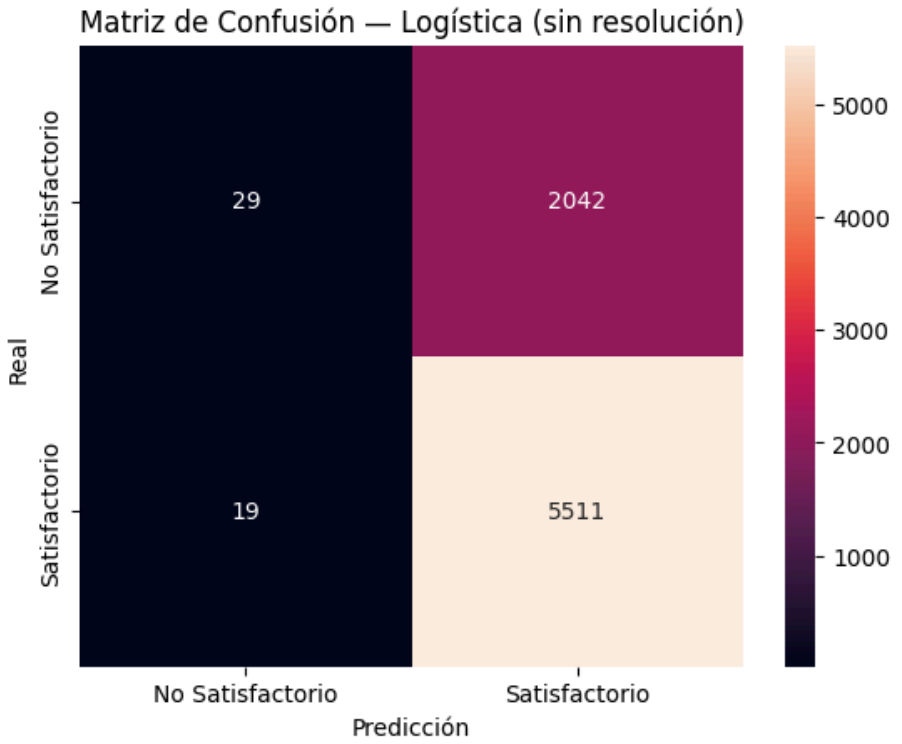
- Tabla comparativa con métricas (precisión, recall, F1, AUC) para Regresión Logística, Random Forest y XGBoost en diferentes configuraciones.

Modelo (setup)	Umbral	Recall clase 0	Precisión clase 0	F1 clase 0	AUC	Nota
Regresión Logística (base)	0.50	0.01	—	0.03	0.62	Línea base insuficiente (desbalance).
Random Forest	0.50	0.30	0.81	0.35	—	Mejora fuerte vs base; aún bajo recall.
XGBoost (Exp. A) + *class weight + early stopping*	0.56	0.61	—	—	—	Umbral optimizado desde 0.50→0.56.
XGBoost (Exp. B) + *RandomizedSearchCV* (refit=F1_0)	0.54	0.73	—	—	—	Ajuste fino de HP; consolida *recall_0*.
XGBoost (Exp. C) + *Freq/Target Mean Encoding*	0.58	0.80	—	—	—	Salto cualitativo por ingeniería de variables (modelo ganador).

- Matrices de confusión para cada modelo.

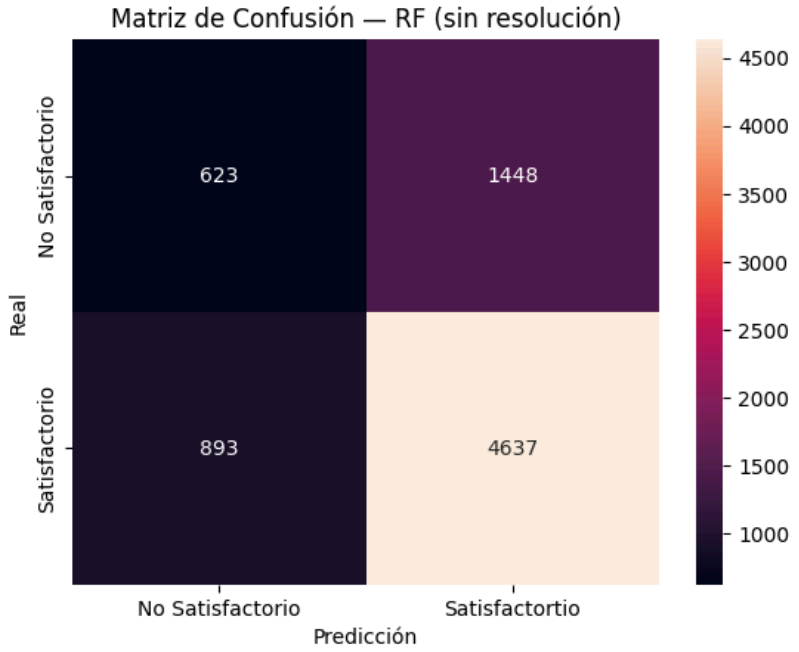
Modelo Logistica

1. **Regresión Logística (base)**
   - Umbral 0.50
   - Línea de base muy floja: prácticamente no detecta insatisfechos (recall clase 0 ≈ 0%).



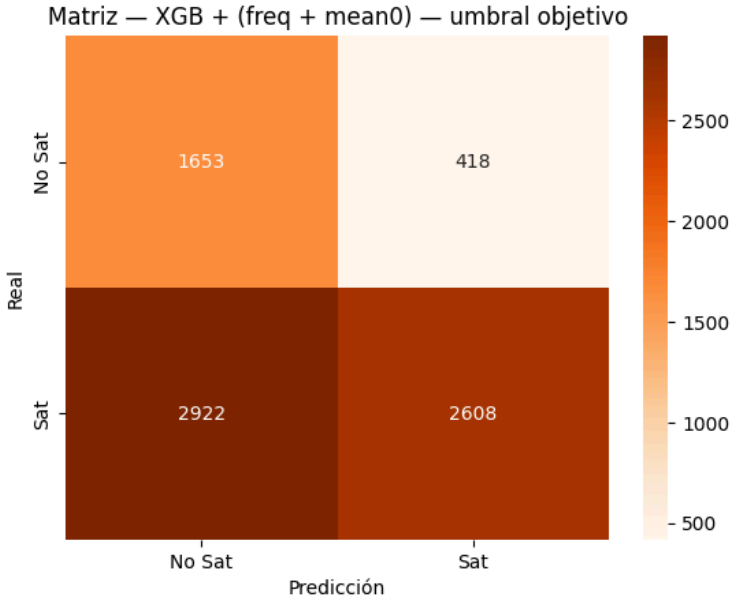
Modelo Random Forest

2. **Random Forest**
    - **Umbral 0.50**
    - **Mejora frente a la logística, recall clase 0 ≈ 30%, precisión aceptable.**

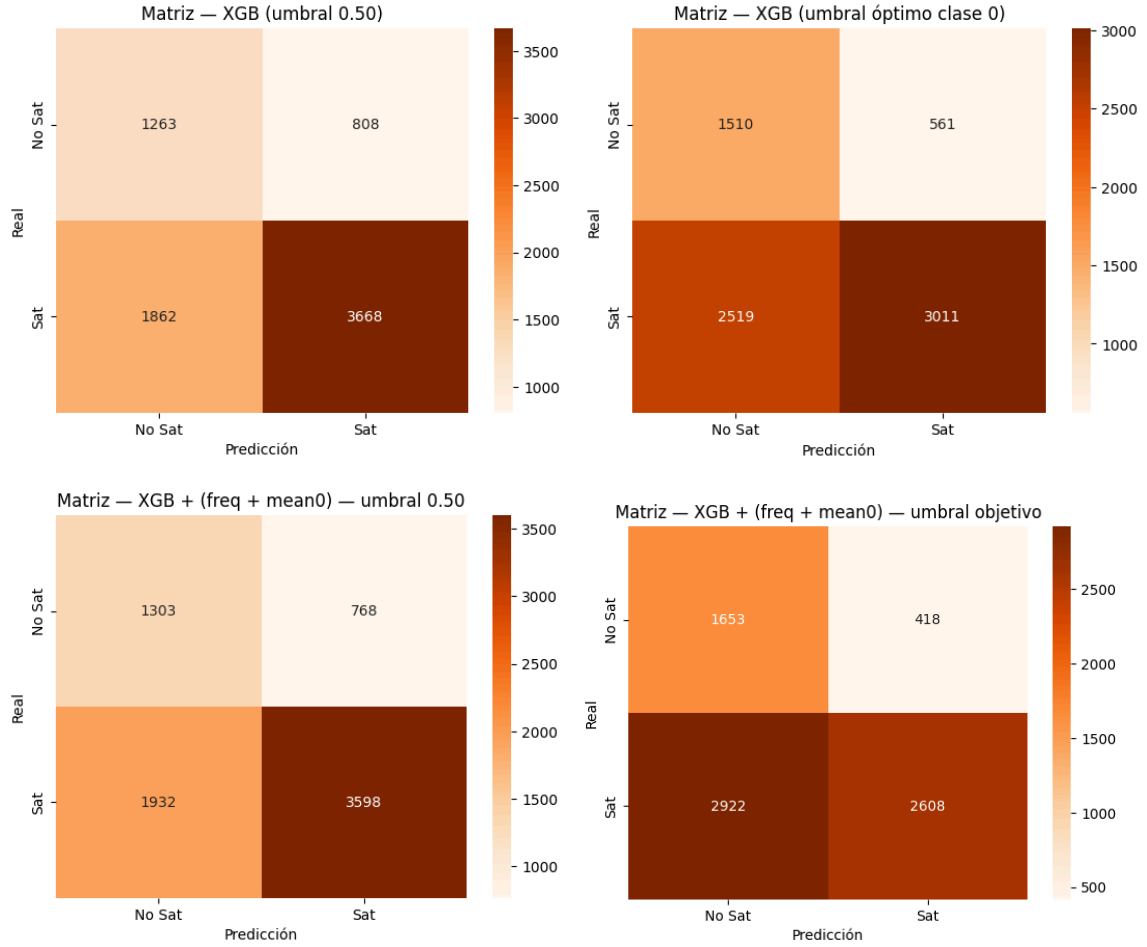


Modelo XGBoost - final

3. **XGBoost – Experimentos**
    - **Exp. A: XGBoost con class weight + early stopping → recall ≈ 61%.**
    - **Exp. B: XGBoost con RandomizedSearchCV (refit=F1_0) → recall ≈ 73%.**
    - **Exp. C: XGBoost con Frequency/Target Mean Encoding + umbral 0.58 → recall ≈ 80%, salto cualitativo gracias a ingeniería de variables.**



Evolución hasta la mejora del modelo objetivo - Matriz de confusion



Modelos comparados

XGBoost – Experimentos

- **Exp. A:** XGBoost con class weight + early stopping → recall ≈ 61%.
- **Exp. B:** XGBoost con `RandomizedSearchCV` (refit=F1_0) → recall ≈ 73%.
- **Exp. C:** XGBoost con Frequency/Target Mean Encoding + umbral 0.58 → recall ≈ 80%, salto cualitativo gracias a ingeniería de variables.

Anexo 6. Ingeniería de variables y parámetros de modelos

🔹 Transformaciones aplicadas

1. Target Mean Encoding (Producto / Subproducto)
   - A cada categoría de Producto_sf y Subproduct_sf se le asignó la tasa histórica de insatisfacciónCalculada en train.
   - Esto permitió transformar variables categóricas en numéricas con alto poder predictivo.
2. Frequency Encoding
   - Se incorporó la frecuencia relativa de aparición de cada categoría como variable adicional.
3. Binning de duración
   - La variable Duración (en segundos) se normaliza y luego se agruparon tramos (ej. <1 min, 1–3 min, >3 min).

    - Se detectó que llamadas muy cortas (<1 min) tienden a asociarse a insatisfacción.
4. Normalización y limpieza
    - Tratamiento de valores nulos (fillna con media o categoría “otros”).
    - Eliminación de duplicados.
    - Codificación robusta para evitar *data leakage* (aplicada siempre sobre train).

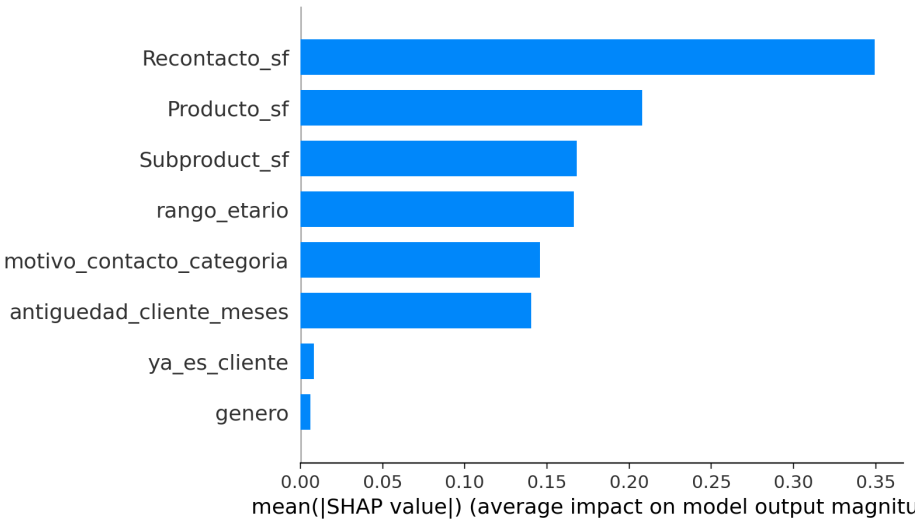
🔹 Parámetros finales de los modelos

- Regresión Logística
    - Solver: liblinear
    - Penalización: l2
    - C: 1.0 (default)
- Random Forest
    - n_estimators=300
    - max_depth=None (crece hasta que cada hoja es pura)
    - min_samples_split=2, min_samples_leaf=1
    - class_weight=balanced
- XGBoost (modelo final Exp. C)
    - max_depth=6
    - learning_rate=0.1
    - n_estimators=300
    - subsample=0.8
    - colsample_bytree=0.8
    - scale_pos_weight ajustado para el desbalance
    - Early stopping con n=30

🔹 Justificación de la elección

- XGBoost con Target/Freq Encoding fue el modelo seleccionado porque:
    - Logró un recall en la clase 0 (insatisfechos) de ~80%, muy superior a los otros.
    - El trade-off fue un aumento en falsos positivos, considerado aceptable dado el objetivo de negocio: capturar insatisfechos aunque implique más revisiones.
    - La mejora se debió más a la ingeniería de variables que al simple ajuste de hiperparámetros.
- Umbral de decisión optimizado (0.58)
    - Con el umbral por defecto (0.50), el recall_0 quedaba más bajo (~73%).
    - Ajustando a 0.58 se logró mejorar el recall sin perder demasiado en precisión.
    - La selección del umbral se basó en la curva Precision–Recall y la evaluación de falsos negativos (FN) como el error más crítico.

Anexo 7. Explicabilidad del modelo



El análisis SHAP confirma que las variables más influyentes en la predicción de insatisfacción son aquellas derivadas de ingeniería de variables: Subproduct_sf_mean0 y Producto_sf_mean0 (tasas históricas de insatisfacción). Esto evidencia que la mejora del modelo no se debe sólo al ajuste de hiperparámetros, sino al enriquecimiento del dataset con nuevas features.

Anexo 8. Reproducibilidad

Entorno de ejecución:

- Plataforma: Google Colab (basado en Jupyter Notebook).
- Lenguaje: Python 3.10.
- GPU: No requerida.

Principales librerías utilizadas:

- `pandas` → manipulación de datos.
- `numpy` → operaciones numéricas.
- `scikit-learn` → modelos base, métricas y validación cruzada.
- `xgboost` → modelo principal de boosting.
- `shap` → interpretabilidad del modelo.
- `matplotlib` y `seaborn` → generación de gráficos.

Versión de librerías:

- Se incluye archivo `requirements.txt` en el repositorio / carpeta compartida, generado con `pip freeze`.
- Ejemplo de instalación en Colab:

```
!pip install -r requirements.txt
```

Instrucciones para reejecutar el notebook en Colab desde Github:

Se puede acceder directamente desde el link de gitHub que ya está enlazado con colab para su uso

Instrucciones para reejecutar el notebook en Colab:

1. Abrir Google Colab y montar Google Drive:

```
from google.colab import drive

drive.mount('/content/drive')
```

2. Subir el archivo `TFM_CSAT_Desde_Drive.ipynb` (o abrirlo directamente desde tu carpeta de Drive).
3. Verificar que el archivo de datos (`BaseEncuestasClientes.xlsx`) esté en la ruta indicada dentro del notebook.
4. Instalar dependencias adicionales si fuera necesario:

```
!pip install xgboost shap imbalanced-learn
```

5. Ejecutar todas las celdas en orden (Menú → “Entorno de ejecución” → “Ejecutar todo”).
6. Los resultados reproducen:
    - Gráficos exploratorios (EDA).
    - Entrenamiento y validación de modelos (Logística, Random Forest, XGBoost).

- Métricas comparativas y matrices de confusión.
- Gráficos de interpretabilidad con SHAP.

8. Bibliografía

A continuación, se citan las principales librerías y frameworks de Python que constituyeron la base tecnológica para el desarrollo de este proyecto.

- **Chen, T., & Guestrin, C.** (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- **Lundberg, S. M., & Lee, S. I.** (2017). A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30*. (Referencia para la librería SHAP).
- **McKinney, W.** (2010). Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*. (Referencia para la librería Pandas).
- **Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E.** (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830