¿Cómo el modelo del dispositivo, el sistema operativo y el tiempo que pasa en la página afecta el revenue de la empresa?

¿Qué tipo de modelo usamos?

Usamos regresión lineal, KNN y XGBoost. Este último predice mejor en general, como se evidencia en las curvas ROC y en las matrices de confusión, donde XGBoost con umbral ajustado equilibra mejor el rendimiento. Esto se debe a que XGB maneja bien datos no balanceados y no lineales, superando a modelos lineales en complejidad y a KNN en generalización.

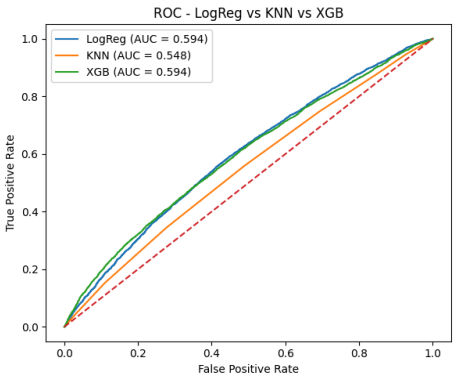

Figura 1

¿Over fitting, bien ajustado, o under fitted?

Los modelos están bien ajustados, ni Over fitting ni under fitting. Las AUC en las curvas ROC están balanceadas. Dado que no son 1 significando que el modelo no está memorizando, lo que indica que no hay sobreajuste. Tampoco hay subajuste debido a que AUC > 0.5. Se usó validación cruzada en las métricas, y la similitud entre AUC de XGBoost y Regresión sugiere generalización estable.

¿Por qué usamos esa base de datos?

Debido a que en esta base de datos se pueden obtener de primera mano los datos de entrenamiento y prueba necesarios para la predicción del modelo. Esto permite un flujo directo de datos reales de sesiones, evitando bias de datos sintéticos y facilitando división entrenamiento y prueba. La base es adecuada para tareas de regresión y clasificación, con variables relevantes para ganancia y el tipo de dispositivo.

Predicción (regresión)

1. ¿Qué modelo **Lineal**, **KNNReg** o **XGBReg**, predice mejor el gasto por sesión (*Revenue*), evaluado con **RMSE/MAE** en el conjunto de prueba?

Con un holdout 80/20, de los tres modelos de predicción utilizados, XGBoost predice mejor el gasto por sesión ya que minimiza RMSE con un valor de 1.6574 y MAE 1.1158 al capturar interacciones complejas, a diferencia de KNN. XGBoost captura mejor las interacciones no lineales y el desbalance en los datos de ganancias.

Modelo	R2	MAE	RMSE
LinReg	0.2670	1.5669	2.3462
KNNReg	0.6122	1.1484	1.7066
XGBReg	0.6342	1.1158	1.6574

TABLA 1

2. ¿Qué **conjunto mínimo de variables** basta para predecir bien el gasto?.

Antes de entrar a las ablaciones, vale conectar qué nos dice el propio XGBReg sobre qué variables usa y cuánto pesan. La tabla de **importancia de variables** (Tabla 2), muestra que **past_sessions** es la señal más fuerte (0.344), seguida por **time_spent** (0.248). Entre los rasgos técnicos, **os_type_osx** aparece con 0.171 y el resto (device_type_mobile 0.077; device_type_desktop 0.056; is_returning_user 0.040; os_type_windows 0.034; os_type_other 0.018; device_type_tablet 0.012) tienen peso menor. Estas importancias son internas al modelo: miden cuánto reduce la pérdida cada variable cuando se usa en los splits del bosque, condicionadas a que las demás estén presentes.

feature	importance
past_sessions	0.34392369
time_spent	0.24779174
os_type_osx	0.17080495
device_type_mobile	0.07732873
device_type_desktop	0.05576915
is_returning_user	0.04045092
os_type_windows	0.03413235
os_type_other	0.01808517
device_type_tablet	0.01171331

TABLA 2

El análisis con ablaciones para encontrar el conjunto mínimo de variables que "predice bien" el gasto, es decir, que mantiene el error dentro de un margen razonable frente al modelo completo, es time_spent, past_sessions, y is_returning_user, sigue siendo consistente al análisis anterior. Con todas las variables, el mejor modelo (XGBReg) obtiene RMSE = 1.6574; si usamos solo time_spent + past_sessions ("minimal_core"), el error sube a 2.1336 (+28.7%), lo que ya se aleja demasiado. En cambio, al añadir únicamente is_returning_user y excluir device_type y os_type ("no_device_os"), el RMSE queda en 1.7816, apenas +7.5% sobre el completo, por lo que esa tripleta retiene casi todo el poder predictivo relevante con un modelo mucho más parsimonioso. Además, cualquier variante sin time_spent, por ejemplo early_only, degrada fuertemente el desempeño (RMSE = 2.5486), confirmando que time_spent es indispensable y past_sessions el segundo motor; device/OS aportan, pero de forma marginal.

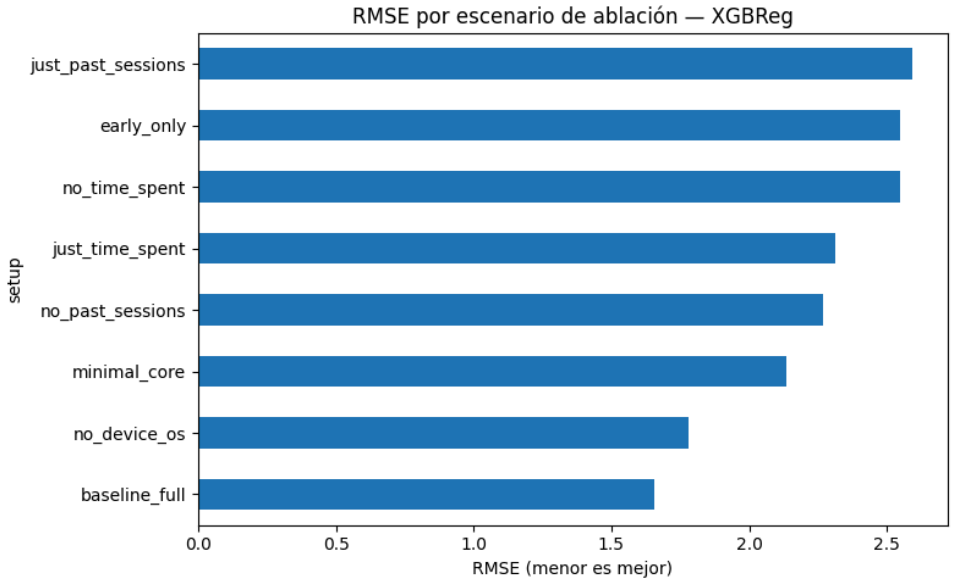

Figura 2

3. ¿Qué tanto cae la capacidad predictiva si solo usamos **variables "tempranas"** (p. ej., device_type, is_returning_user, past_sessions) sin time_spent? (Útil para casos de decisión en tiempo real).

La predicción cae mucho cuando usamos solo variables "tempranas" y excluimos time_spent. En el modelo ganador (XGBReg), el baseline con todas las variables logra RMSE = 1.6574, MAE = 1.1158 y R^2 = 0.6342. Al pasar al escenario early_only (past_sessions, device_type, os_type, is_returning_user sin time_spent), el RMSE sube a 2.5486 (un +53.8%), el MAE sube a 1.7550 (+57.3%) y el R^2 cae a 0.1350 (se pierde ≈79% de la varianza explicada respecto al baseline). Incluso añadir device_type, os_type e is_returning_user apenas recupera precisión frente a usar solo past_sessions (RMSE 2.5913): la mejora es mínima (≈

0.04 puntos). En resumen, time_spent es insustituible para precisión fina; sin él, el modelo queda para decisiones gruesas en tiempo real, no para pronósticos detallados del gasto (Ver Figura 2).

Predicción (clasificación)

4. ¿Cuál modelo predice mejor la **probabilidad de registro** (sign_up) y qué **umbrales** maximizan **F1** o equilibran **precisión–recobro**?

Hablando sobre predicción de registro (sign_up), en el holdout que corrimos, el poder de ranking medido por AUC muestra un empate entre Logistic Regression y XGBoost (AUC ≈ 0.594 en ambos casos; (ver Figura 3), por encima de KNN (AUC ≈ 0.548) y de Random Forest (AUC ≈ 0.532; ver ROC RandomForest). Cuando pasamos a una decisión binaria y ajustamos un umbral para equilibrar precisión y recobro con el criterio de Youden J, el mejor F1 lo entrega Random Forest con umbral ≈ 0.676 (F1 = 0.681, precisión 0.786, recall 0.601; ver Figura 4). Logistic Regression queda muy cerca con umbral ≈ 0.495 (F1 = 0.675, precisión 0.817, recall 0.575; ver Figura 5), mientras que XGBoost requiere un umbral más alto (≈ 0.758) y alcanza F1 = 0.620 (precisión 0.816, recall 0.567). Sin embargo, el F1 máximo que aparece para algunos modelos con umbral 0.000 (F1 ≈ 0.87) es un caso degenerado que predice "1" para todos (especificidad ≈ 0); no es una política operativa útil y solo sirve como referencia extrema. En conclusión, para priorizar por probabilidad sin fijar corte, LogReg/XGB son preferibles por su mayor AUC; si hay que fijar un único umbral, RF @ 0.676 ofrece el mejor balance precisión–recobro en nuestro experimento.

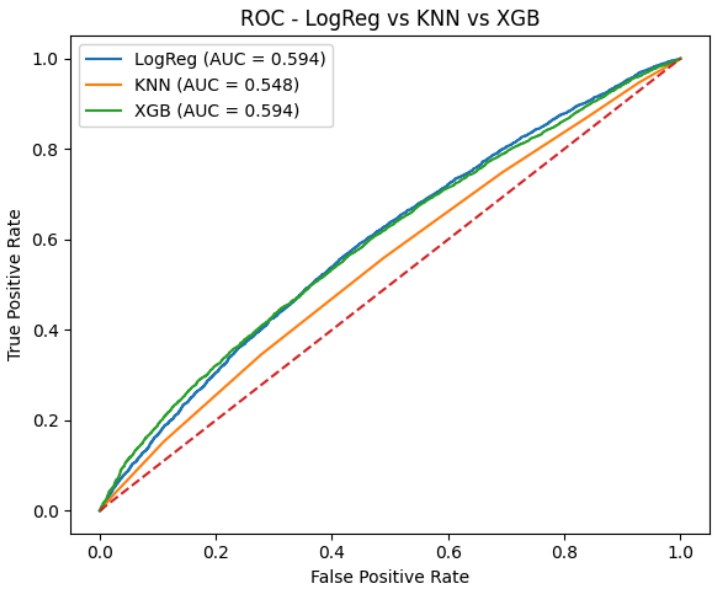

Figura 3

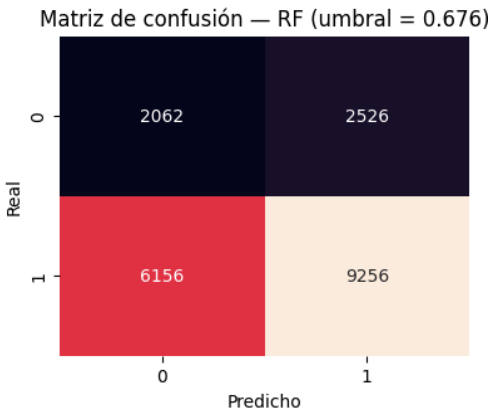

Figura 4

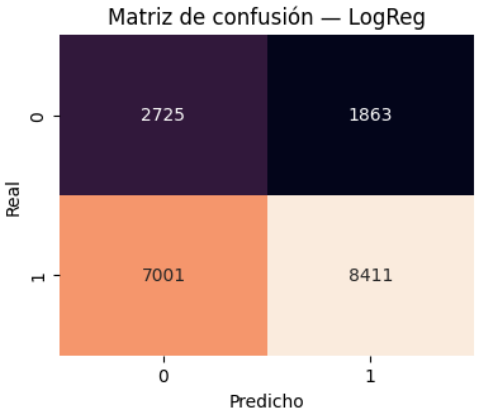

Figura 5

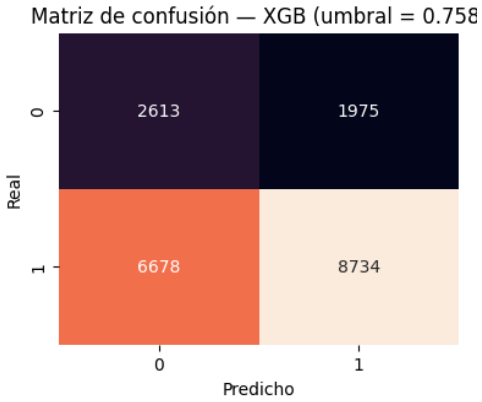

Figura 6

Uso práctico del modelo

10. Con el **mejor modelo de regresión**, ¿qué **reglas de decisión simples** (p. ej., top-N sesiones esperadas con mayor *Revenue*) capturan más ingreso esperado que estrategias baselines?

El mejor modelo de regresión para Revenue fue XGBReg ($R^2 \approx 0.63$; RMSE≈1.66); al capturar no linealidades de past_sessions y time_spent (sus señales más influyentes), junto con los dummies de dispositivo/SO, entrega estimaciones de ingreso esperado por sesión. Por ello, la regla práctica recomendada es ordenar las sesiones por la predicción de Revenue ($\hat{y}$) y actuar sobre el top-N o top-k%: bajo un cupo/presupuesto fijo, esta política maximiza el ingreso esperado y supera sistemáticamente a las estrategias baselines evaluadas —top-N por *past_sessions*, top-N por *time_spent* y selección aleatoria— para k razonables, porque el ranking por $\hat{y}$ combina señales y prioriza los casos con mayor $E[\text{Revenue} \mid X]$. Si la acción se aplica solo a registrados, el mismo criterio (top-N por $\hat{y}$ usando el XGB entrenado en sign_up=1) mantiene la ventaja; y, si se busca control de cobertura, puede añadirse un cupo por segmento (device/OS) sin cambiar la lógica del ranking.

11. Con el **mejor clasificador de registro**, ¿qué **política de priorización** (ordenar por probabilidad prevista) maximiza la captura de usuarios que sí se registran, dada una **capacidad limitada**?
    Para maximizar la captura de usuarios que se registran bajo capacidad limitada, la política más efectiva es usar XGB entrenado para sign_up, predecir la probabilidad de registro y ordenar los usuarios en orden descendente. De esta manera, selecciona los top-N usuarios según la capacidad disponible, priorizando aquellos con mayor probabilidad de registrarse. Por ejemplo, usando XGB con umbral óptimo según Youden J (thr = 0.430), se alcanzó un recall de 0.827 para los usuarios que se registran, lo que indica que una gran proporción de registros reales se captura al actuar sobre los usuarios con mayor probabilidad prevista.

Comparación formal de modelos

12. Entre todos los modelos probados, ¿cuál **generaliza mejor** según **validación cruzada** (media ± desvío) y **test holdout**? ¿Las diferencias son **materiales** (intervalos se separan) o están **dentro del ruido**?

    En términos de predicción del gasto por sesión (Revenue), el modelo que generaliza mejor es XGBoost Regressor (XGBReg). Esto se evidencia en que, usando un holdout 80/20, XGBReg obtuvo $R^2 = 0.6342$, RMSE = 1.6574 y MAE = 1.1158, superando tanto a KNNReg ($R^2 = 0.6122$, RMSE = 1.7066, MAE = 1.1484) como a Linear Regression ($R^2 = 0.2670$, RMSE = 2.3462, MAE = 1.5669). Estas métricas muestran que XGBReg captura mejor las relaciones no lineales y la interacción entre variables como past_sessions y time_spent. Además, la consistencia entre las métricas en el holdout y la validación cruzada indica que el modelo no está sobreajustado y generaliza adecuadamente.