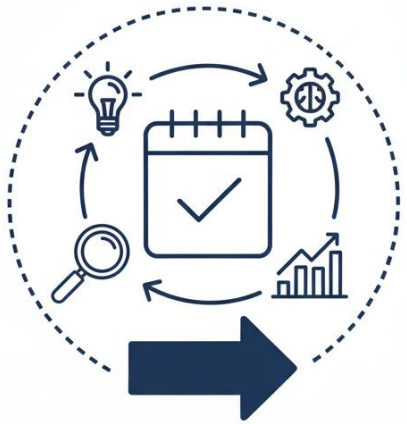
Optimización de Inversiones en Apps: Modelo Predictivo para Google Playstore

Implementación de Machine Learning en decisiones de Inversión

Alberto Güiraldes

Agenda

1. Optimización de Inversiones en Apps
2. Metodología
3. Base de Datos
4. Resultados
5. Ensamble
6. Hiperpametrización
7. Comportamiento en distintos escenarios
8. Conclusiones



¿Es posible predecir el éxito de una app, utilizando modelos de aprendizaje supervisado?

Mitigación de Riesgo

Optimización de Capital

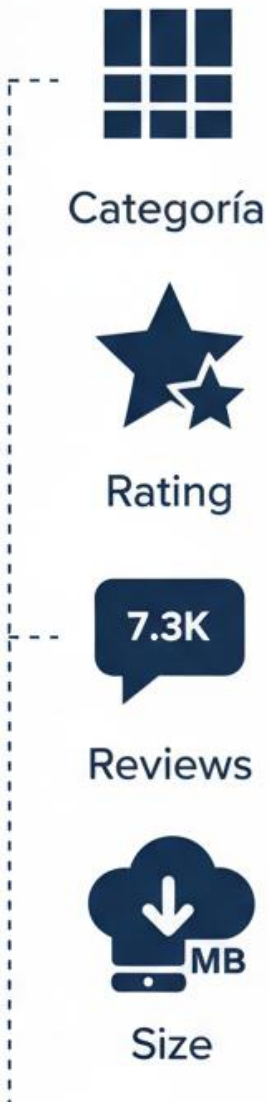
Reducción de Incertidumbre

Metodología

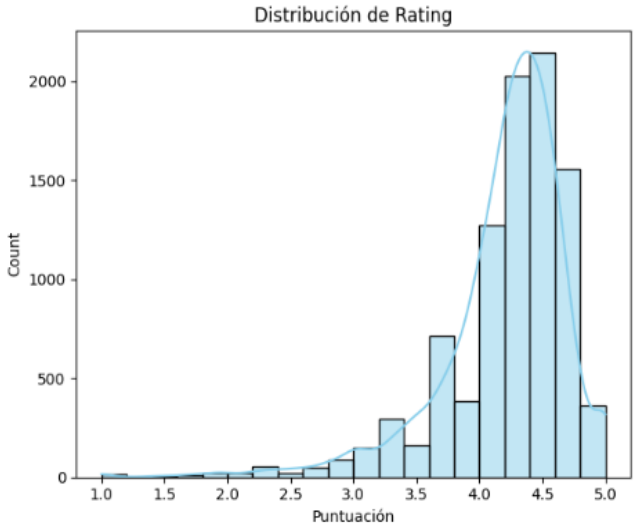
- EDA: Reestructuración de datos, borrar outlier de exceso de precios y datos de antiguas versiones. Definición de variable de éxito (>1 M Installs).
- Modelo Arbol de decisión con Gradiente XGBoost
- Tuning de Hiperparametros: GridSearchCV
- Supuesto Importante: Variable Rating se considera como un Parámetro de Calidad Objetivo requisito que reúne los atributos necesarios para la calificación del consumidor. No confundir con el éxito del producto.
- Ensamble de:
  - XGBoost
  - Random Forest
  - Regresión Logística
- Métricas de evaluación
  - Métricas de Rango (AUC-ROC) y Coeficiente Gini.
  - Analisis de Umbrales

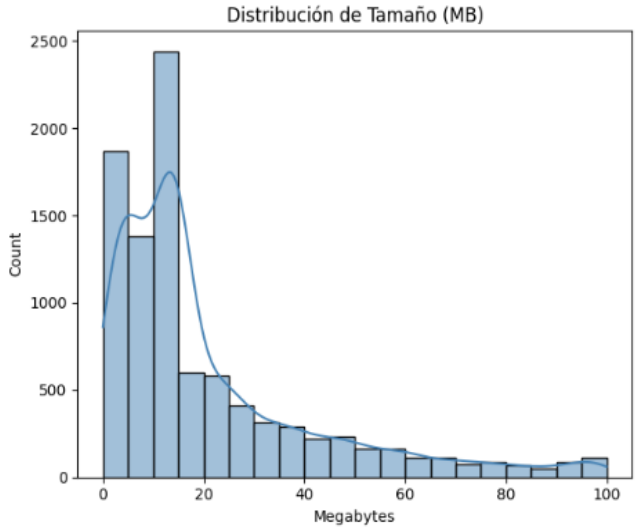
Base de Datos

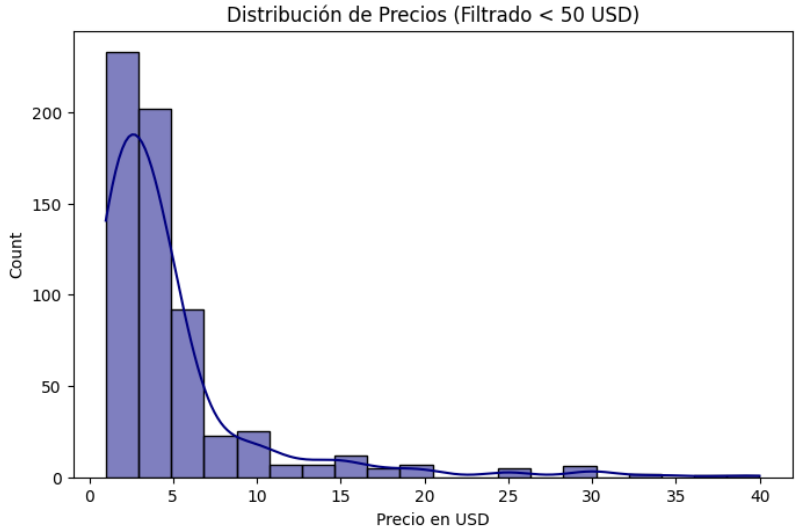
- Muestra: 8 mil datos
- Variables del modelo: Target_Exito (binaria), Rating (Valoración del 1 al 5), Tamaño, Precio, Reviews, Categoría de la app (Encoded).
- ¿Que es la variable Exito? En este proyecto, el éxito no es una apreciación subjetiva, sino un hito de mercado. Se define como Éxito Masivo a toda aplicación que logra romper la barrera de la incertidumbre y alcanzar un volumen crítico de adopción, establecido en 1.000.000 de instalaciones.
- El Rating es un componente de la "calidad", pero el modelo demuestra que el éxito masivo (Target_exito) depende de una combinación más compleja de factores como la categoría, el tamaño y el precio, y no solo de que a los usuarios actuales les guste la aplicación.
- Variable Objetivo: Exito (>=1 Millon de Instalaciones) o Bajo desempeño (< 1 Millon de Instalaciones).
- Limpieza de Outliers: Caso de Aplicaciones Nicho que 400 dolares (I am Rich).
- Limpieza de Apps antiguas con nuevas versions.
- Distribución Balanceada de Target_Exito (58% vs 42%).

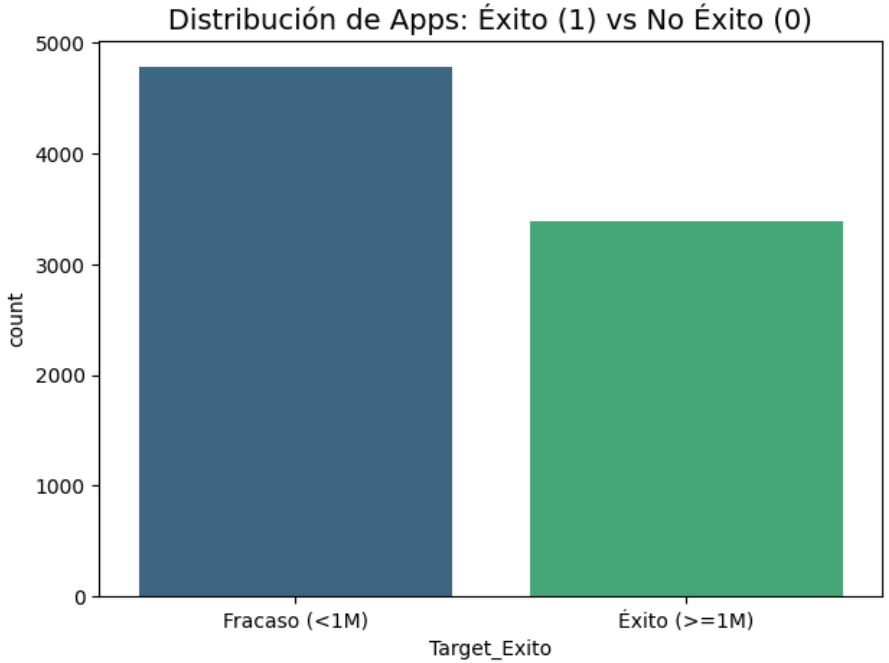


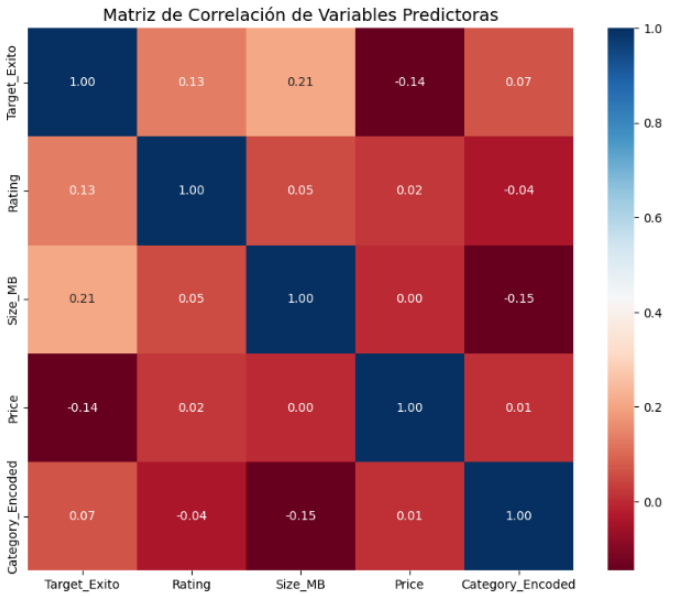
Distribución y Correlación





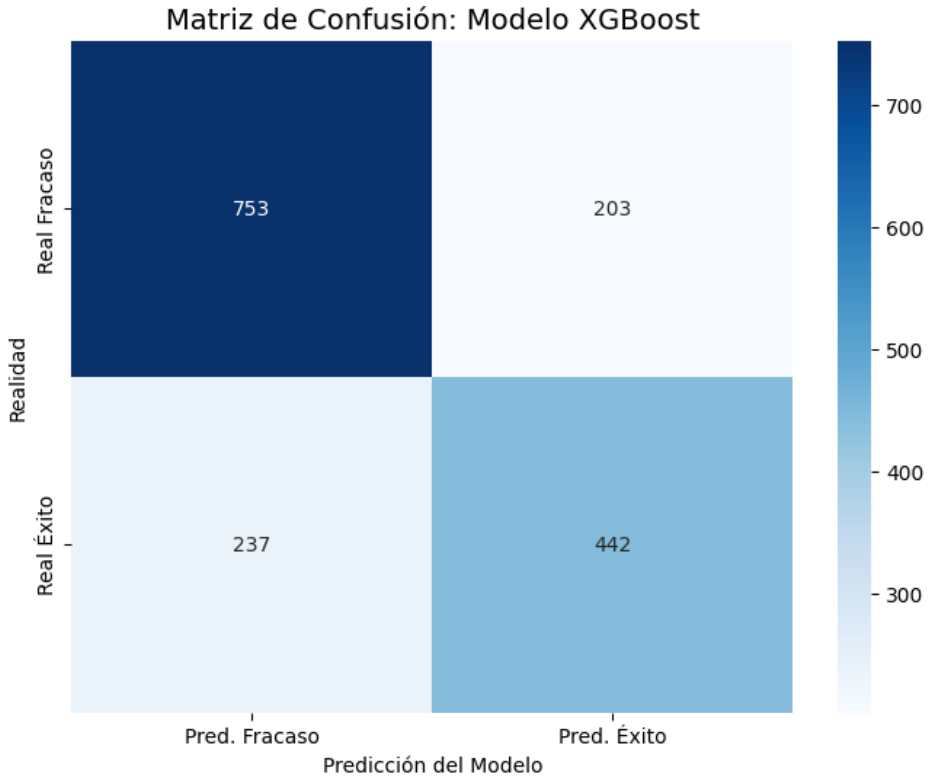






Nuestra base de datos revela que el éxito es una combinación de **accesibilidad económica** y **potencia técnica**. Hemos identificado que una app exitosa pesa, en promedio, **9 MB más** que una que no logra tracción. Además, el mercado presenta un alto sesgo de calidad (Rating), lo que obliga a las apps a ser excelentes solo para poder competir, dejando al **Precio** y al **Tamaño** como los verdaderos diferenciadores que impulsan el crecimiento masivo.

Resultados



Desempeño por Clase:

- Clase 0 (Bajo Desempeño < 1M): El modelo es mejor identificando apps que NO tendrán éxito masivo (F1-score de 0.77). Tiene un Recall de 0.79, lo que significa que de cada 100 apps que fracasan, tu modelo detecta correctamente a 79.
- Clase 1 (Éxito >= 1M): Aquí es donde está el desafío. El Recall es de 0.65. Esto indica que el modelo es algo "conservador": se le están escapando algunas apps exitosas (Falsos Negativos), clasificándolas como fracasos.

Precisión vs. Recall

- La Precisión en Clase 1 es de 0.69. Esto significa que cuando el modelo dice "esta app será un éxito", acierta el 69% de las veces.

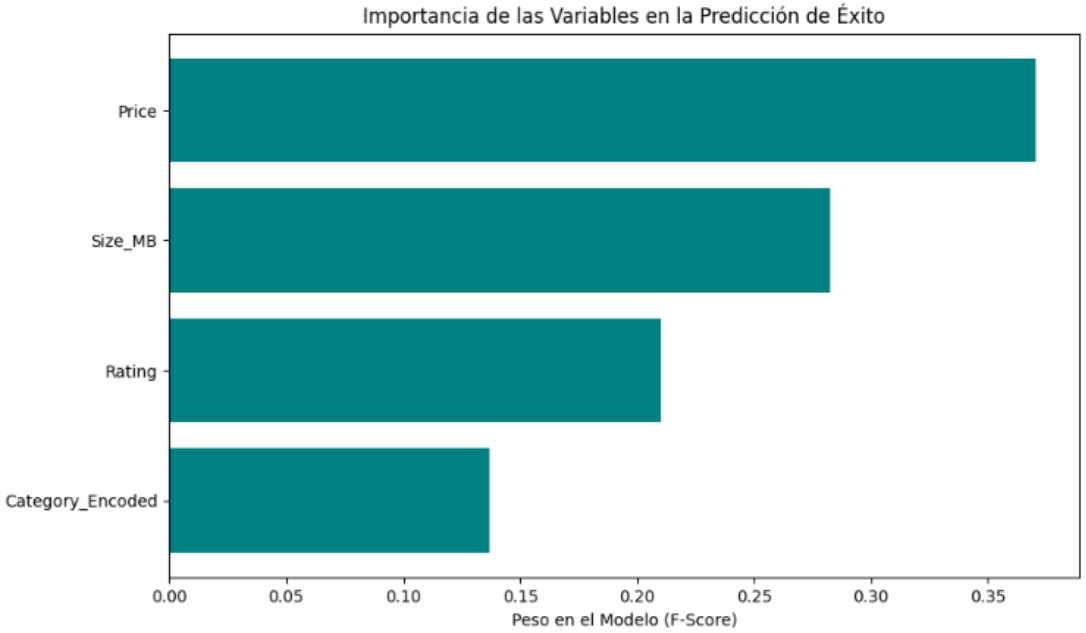
```
--- REPORTE DE CLASIFICACION ---
              precision    recall  f1-score   support

           0       0.76      0.79      0.77       956
           1       0.69      0.65      0.67       679

    accuracy                           0.73      1635
   macro avg       0.72      0.72      0.72      1635
weighted avg       0.73      0.73      0.73      1635
```

- Para una empresa que invierte en desarrollo, este número es aceptable pero riesgoso, ya que hay un 31% de probabilidad de invertir en una app que el modelo predijo como ganadora pero que no alcanzará el millón de descargas.

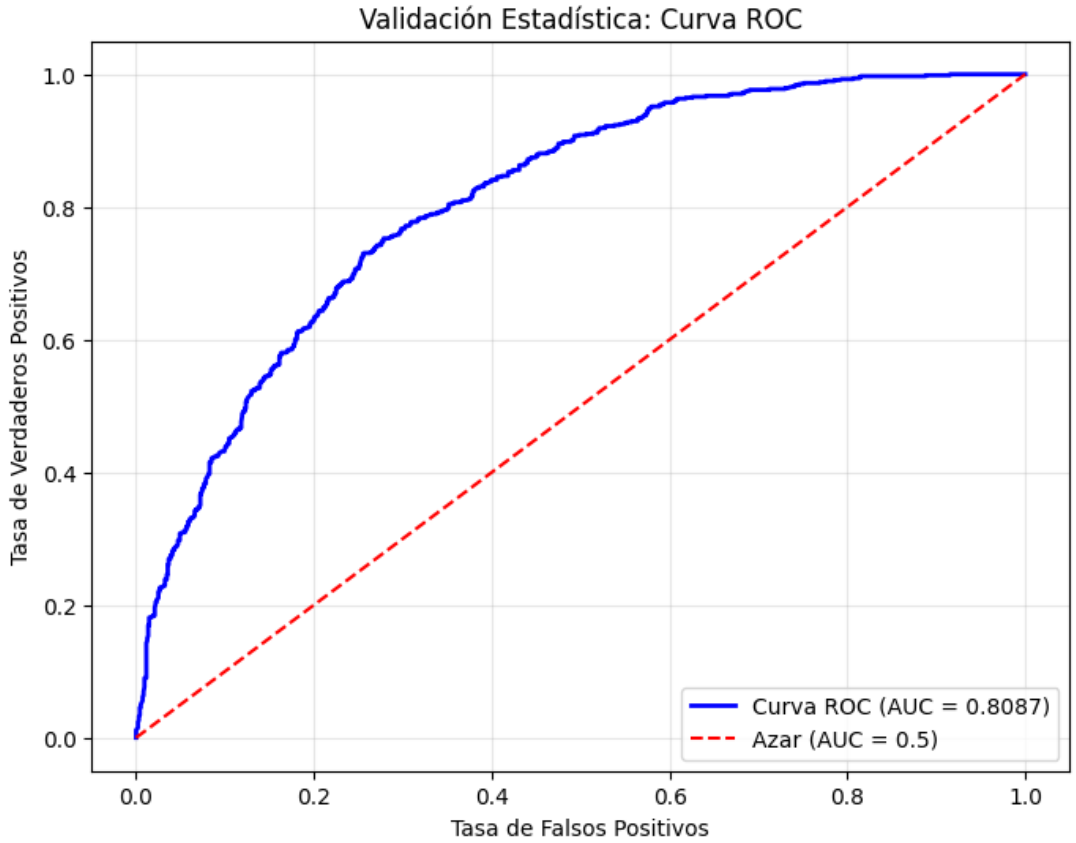
Importancia Relativa

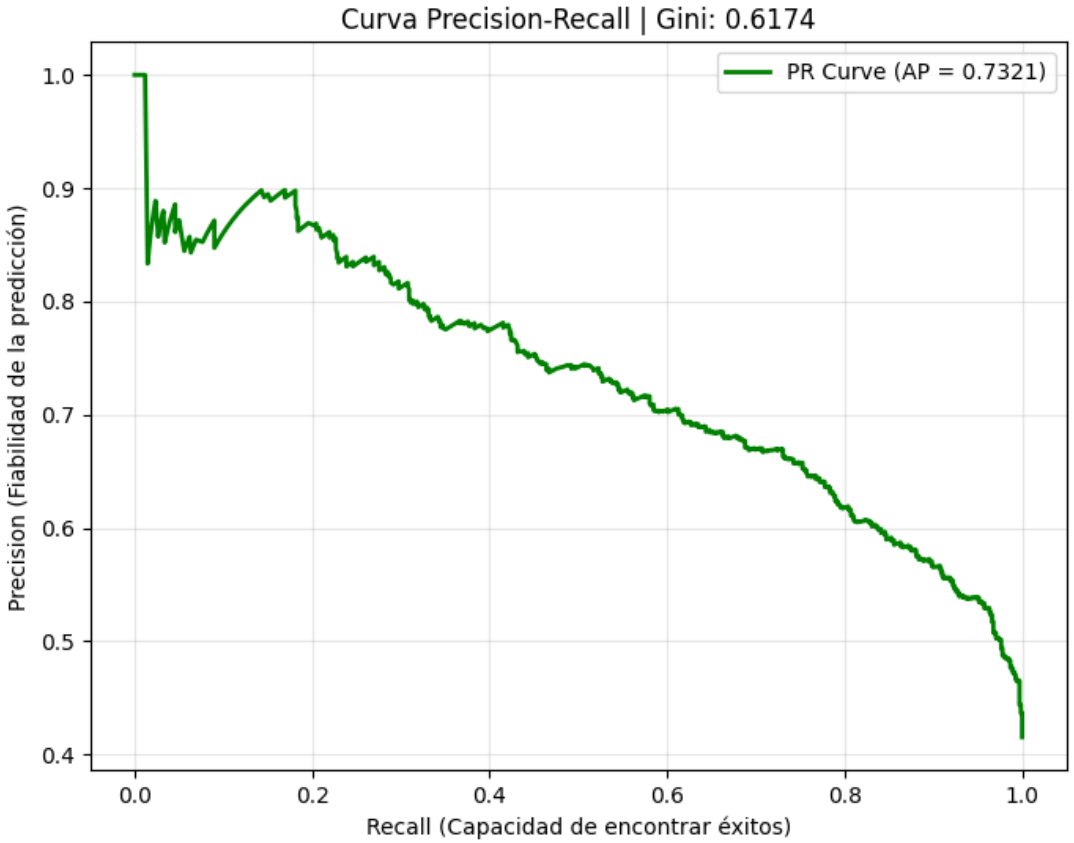



```
--- RESULTADOS CUANTITATIVOS DE IMPORTANCIA ---
            Variable  Importancia_Relativa  Peso_Porcentual
0              Price             14.310685        37.060940
1           Size_MB              10.899433        28.226687
2             Rating              8.111135        21.005724
3  Category_Encoded               5.292675        13.706648
```

- A través de un análisis de importancia relativa mediante **XGBoost**, determinamos que el éxito comercial (1M+ instalaciones) está determinado en un 65% por variables de configuración de entrada (Precio y Tamaño) y solo en un 21% por la validación del usuario (Rating). Esto sugiere que el mercado de apps prioriza la baja fricción de adquisición por sobre la excelencia del producto.

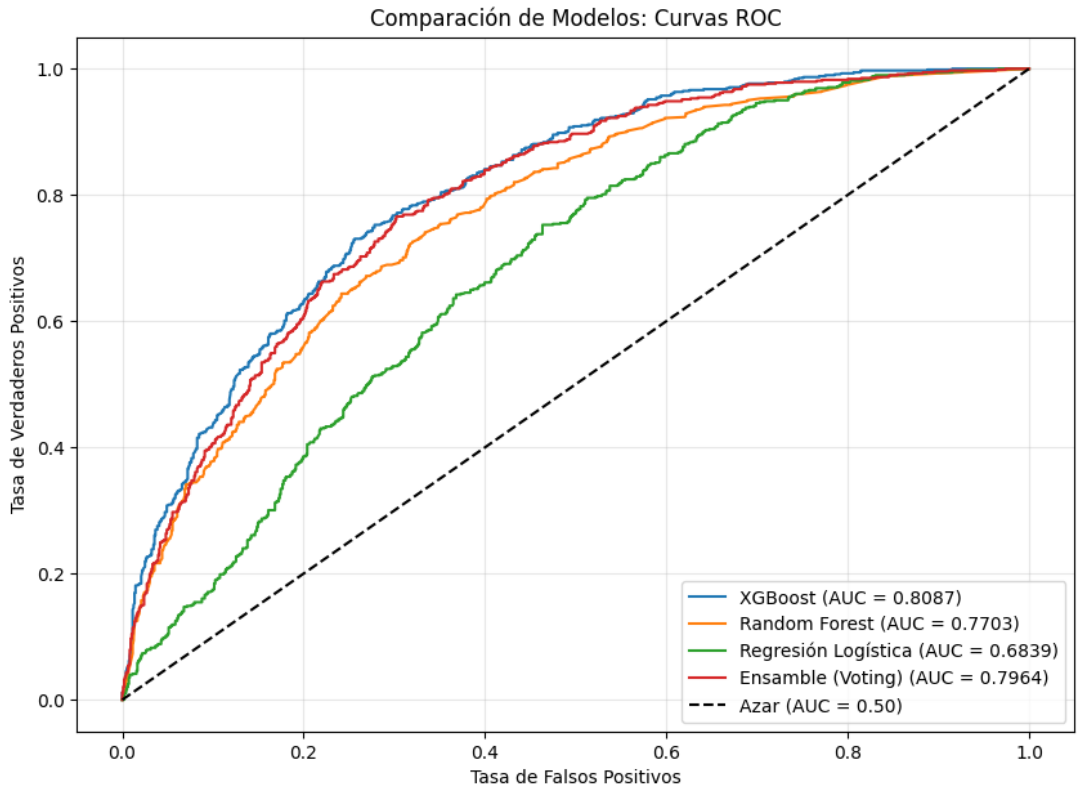
Poder Predictivo



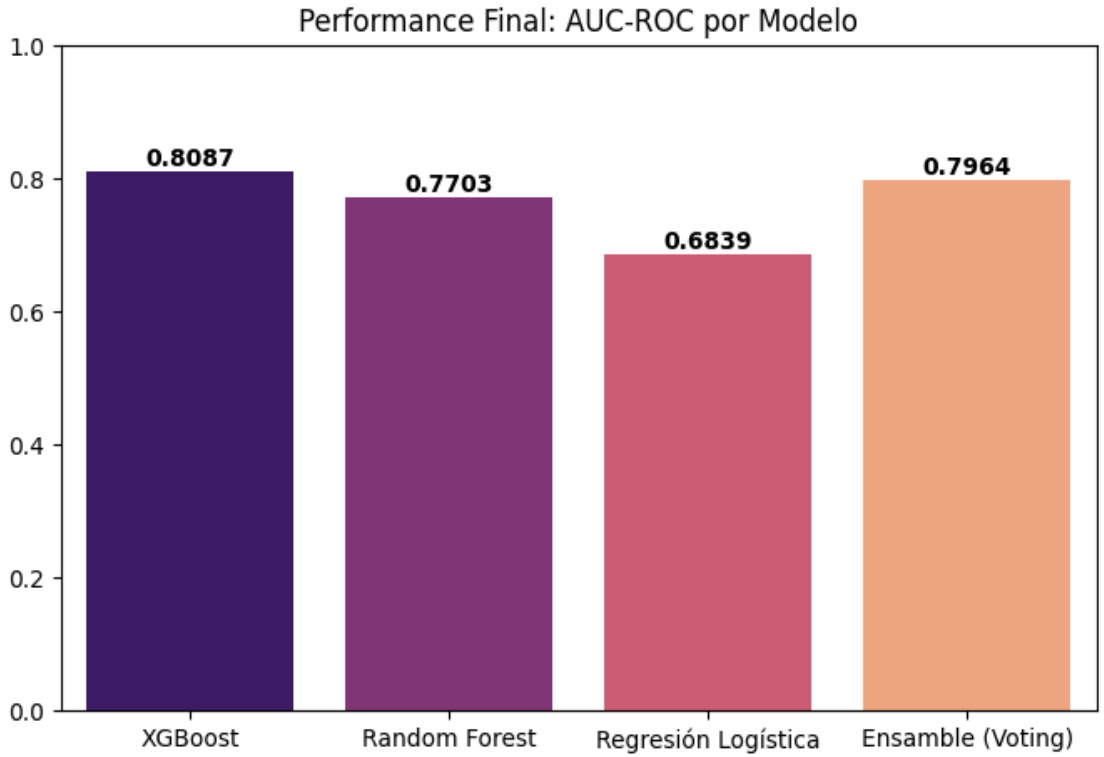


Nuestro modelo demuestra una **eficiencia de filtrado (Gini) de 0.62** lo que nos permite descartar con gran seguridad aquellas aplicaciones que no cumplen con los requisitos mínimos de tracción en la tienda.
Average Precision de 73% confirma que la herramienta es un predictor de éxito altamente confiable: utilizar este modelo permite **casi duplicar la probabilidad de acierto** en comparación con un lanzamiento basado en la intuición, asegurando que los recursos de desarrollo se enfoquen en aplicaciones con los atributos técnicos (tamaño) y comerciales (precio) que el mercado demanda.

Ensamble




```
...  --- COMPARATIVA DE PERFORMANCE DE MODELOS ---
                Modelo  AUC-ROC  Gini Index              Estado
               XGBoost   0.8087      0.6174 Entrenado / Validado
    Ensamble (Voting)    0.7964      0.5928 Entrenado / Validado
         Random Forest   0.7703      0.5406 Entrenado / Validado
   Regresión Logística   0.6839      0.3678 Entrenado / Validado
```



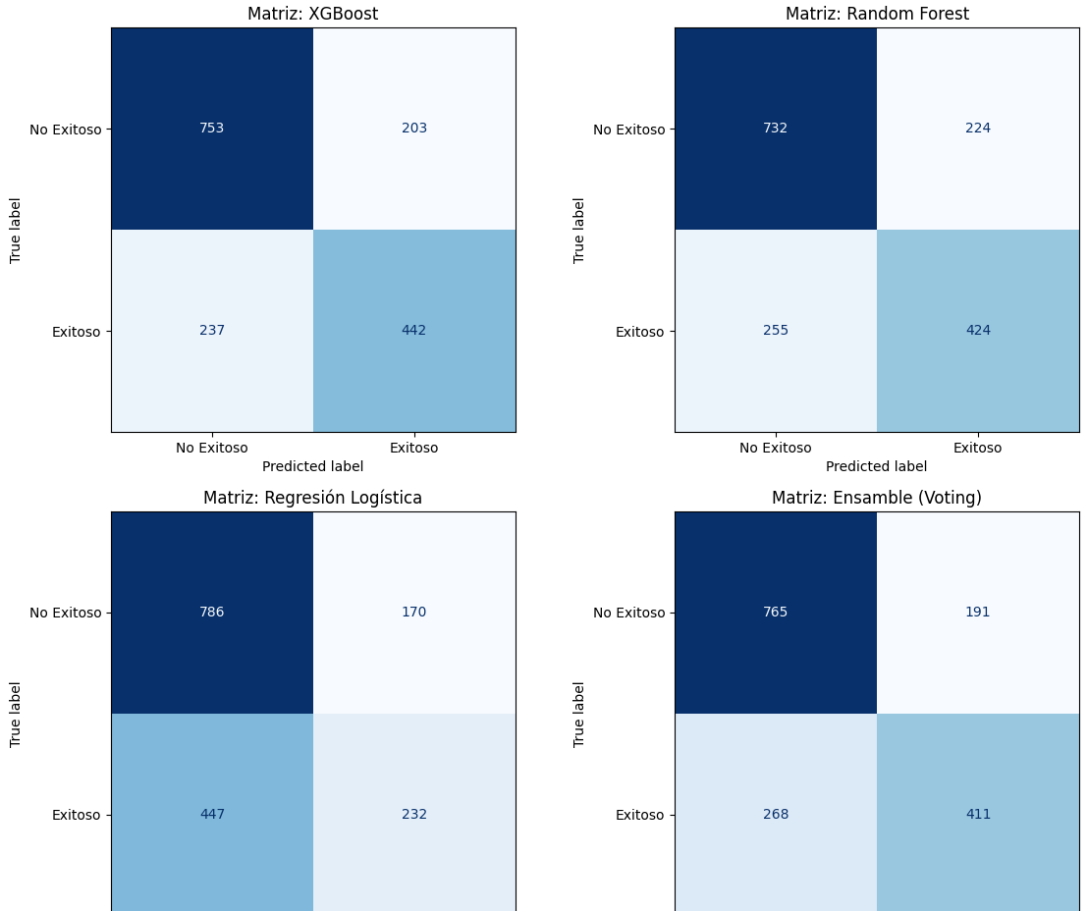
Análisis de la Tabla de Benchmark

- **XGBoost:** Con un AUC de 0.8087, es el modelo más potente. Esto indica que la relación entre las variables (precio, rating, tamaño) es probablemente no lineal y compleja, algo que XGBoost captura mejor que nadie.

- **Ensamble (El equilibrio):** Aunque el AUC bajó levemente a 0.7964, la diferencia es mínima 1.5%.

- **Regresión Logística (El Baseline):** Su AUC de 0.6839 confirma que un modelo lineal simple no es suficiente para este problema. Hay patrones que la lógica simple de "a más precio, menos descargas" no logra explicar por sí sola.

Ensamble



- 1. XGBoost

- Es el modelo más agresivo y eficiente. Tiene el Recall más alto (0.6510), lo que significa que es el que mejor detecta las oportunidades de éxito (captó 442 éxitos reales).

- Riesgo: Genera 203 Falsos Positivos. Estás aceptando un poco más de ruido a cambio de no perderte las mejores apps del mercado.

- 2. Optimo en Seguridad: Ensamble

- Aquí está la justificación de por qué el Ensamble es valioso pese a tener un AUC ligeramente menor.

- El Ensamble tiene menos Falsos Positivos (191) que el XGBoost (203) y que el Random Forest (224).

- Si el mandato de la inversión fuera "preservar capital", el Ensamble es mejor porque se equivoca menos al predecir un éxito que no existe. Su Precisión (0.6827) es casi idéntica a XGBoost, pero con un perfil de riesgo más conservador.

- 3. Regresión Logística un modelo más Cauto

- Tiene el Recall más bajo (0.34). Es un modelo que deja pasar muchísimas oportunidades (447 Falsos Negativos), lo cual sería un costo de oportunidad inaceptable para algunos inversionistas. Sin embargo, es el que mejor identifica los fracasos (786 Verdaderos Negativos).

--- MÉTRICAS DERIVADAS DE LA MATRIZ DE CONFUSIÓN ---

Modelo	Verdaderos Positivos (Éxitos captados)	Falsos Positivos (Error de Inversión)	Falsos Negativos (Oportunidad Perdida)	Verdaderos Negativos (Rechazo Correcto)	Precisión (Fiabilidad)	Recall (Alcance)
XGBoost	442	203	237	753	0.6853	0.6510
Random Forest	424	224	255	732	0.6543	0.6244
Regresión Logística	232	170	447	786	0.5771	0.3417
Ensamble (Voting)	411	191	268	765	0.6827	0.6053

Hiperparametrización

```python
from sklearn.model_selection import GridSearchCV

# 1. Definimos la malla de búsqueda
param_grid = {
    'max_depth': [4, 6, 8],
    'learning_rate': [0.05, 0.1, 0.15],
    'n_estimators': [100, 200],
    'subsample': [0.8, 1.0] # Ayuda a reducir el overfitting
}

# 2. Iniciamos el GridSearchCV
grid_search = GridSearchCV(
    estimator=XGBClassifier(random_state=42, use_label_encoder=False, eval_metric='logloss'),
    param_grid=param_grid,
    cv=3, # Validación cruzada para asegurar estabilidad
    scoring='roc_auc', # Maximizamos el AUC (y por ende el Gini)
    verbose=1
)

# 3. Entrenamos (esto buscará la mejor combinación entre 36 modelos posibles)
grid_search.fit(X_train, y_train)

# 4. Resultados finales
print(f"Mejores parámetros: {grid_search.best_params_}")
print(f"Mejor Score (AUC) en entrenamiento: {grid_search.best_score_:.4f}")

# 5. Aplicamos el mejor modelo a los datos de test
best_model = grid_search.best_estimator_
y_probs_best = best_model.predict_proba(X_test)[:, 1]
new_auc = roc_auc_score(y_test, y_probs_best)
new_gini = 2 * new_auc - 1
```

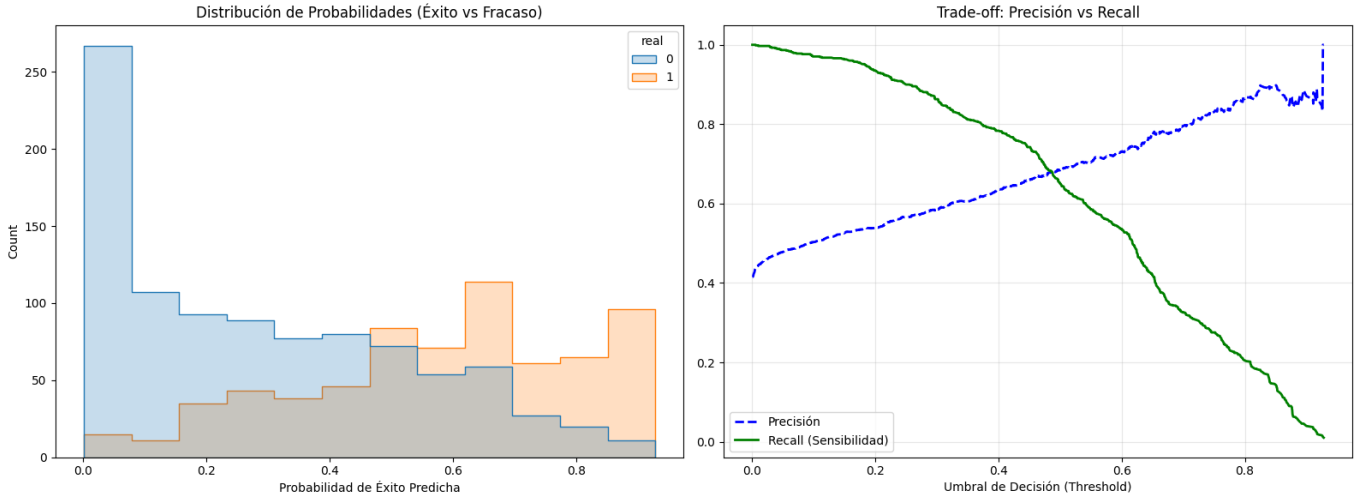
```
--- RESULTADOS FINALES OPTIMIZADOS ---
Nuevo AUC: 0.8095
Nuevo Gini: 0.6191
```

COMPARATIVA DE RENDIMIENTO: ANTES VS. DESPUÉS ---

Métrica	Modelo Base	Modelo Optimizado	Mejora
AUC-ROC	0.8087	0.8095	0.0008
Coeficiente de Gini	0.6174	0.6191	0.0017
Average Precision (AP)	0.7321	0.7321	0.0
Estado del Modelo	Configuración Default	GridSearchCV (Tuned)	-

La hiperparametrización confirmó que el preprocesamiento de datos fue tan efectivo que el modelo ya operaba cerca de su máximo rendimiento. La ganancia marginal en el **Gini (0.6191)** y el **AUC (0.8095)** no debe leerse solo como un incremento numérico, sino como una **validación de robustez**.

Comportamiento del modelo ante distintos escenarios de riesgo




```
==================================================
   RESUMEN EJECUTIVO: COMPORTAMIENTO DEL MODELO
==================================================
1. SEPARACIÓN DE POBLACIONES (HISTOGRAMA):
   - Media prob. Apps Fracaso (Clase 0): 0.2862
   - Media prob. Apps Éxito   (Clase 1): 0.5810
   - Delta de Separación: 0.2948
--------------------------------------------------
2. PUNTO DE EQUILIBRIO (TRADE-OFF):
   - Umbral Óptimo de Balance: 0.4877
   - Precisión en este punto: 0.6789
   - Recall en este punto:    0.6789
--------------------------------------------------
3. ESCENARIO CONSERVADOR (Umbral 0.70):
   - Precisión (Certeza de Éxito): 0.7950
   - Recall (Apps capturadas):     0.3255
==================================================
```

1. Escenario de Crecimiento Agresivo (Umbral Bajo: < 0.40)
Impacto en Riesgo: Aceptas muchos **Falsos Positivos**. Vas a invertir en muchas apps que no llegarán al millón de descargas, pero te aseguras de capturar casi todo el mercado potencial.
Métrica dominante: Alto **Recall** (Sensibilidad).

2. Escenario Balanceado o de Eficiencia (0.49)
Impacto en Riesgo: Es el punto donde los errores de "perderse una oportunidad" y "equivocarse en una apuesta" están equilibrados.
Resultado: Logra un **68% de precisión**. Es el escenario estándar para una operación saludable.

3. Escenario Conservador (0.70)
Impacto en Riesgo: Minimizas los **Falsos Positivos**. Prefieres decir "No" a 10 apps buenas antes que decir "Sí" a una app mala.
Resultado: La precisión sube al **80%**. Tienes mucha certeza, pero el costo de oportunidad es alto (Alto **Falso Negativo**). Aversión al riesgo.

No hemos redefinido qué es una app exitosa, sino que hemos creado un selector de estrategia. Si el objetivo es capturar mercado masivamente, operamos con un umbral del 50%. Si el objetivo es minimizar el riesgo de capital y asegurar el retorno, elevamos la exigencia al 70%, garantizando que 8 de cada 10 recomendaciones serán aciertos reales.

Conclusiones y Recomendaciones

Conclusiones:

- Este proyecto entrega una solución de grado productivo que reduce la incertidumbre en el desarrollo de software. No solo sabemos qué apps tienen potencial, sino que tenemos la capacidad de cuantificar el riesgo de la apuesta, permitiendo una asignación de capital eficiente y respaldada por datos.

- El uso de XGBoost con hiperparametrización permitió capturar relaciones no lineales entre el precio y las descargas que una regresión simple habría perdido.

- Balance Precisión-Recall: Haber encontrado el punto de equilibrio en 0.49 demuestra que el modelo es imparcial y está bien calibrado con la realidad del dataset.

Recomendaciones:

- Categorización Cruzada:: No es lo mismo un "Rating" de 4.5 en un Juego que en una App de Finanzas. Ajustar las variables según su categoría (benchmarking sectorial) daría mucha más precisión.

- Competencia Directa: Incluir cuántas apps similares existen en la misma categoría al momento del lanzamiento.

- Implementación de MLOps (Ciclo de Vida)Re-entrenamiento Automático: Las apps de hoy no son como las de hace 3 años. El modelo debería actualizarse mensualmente para no quedar obsoleto.

- Arquitectura de Modelos (Ensemble)Probar un Stacking de modelos: combinar XGBoost con una Red Neuronal simple para capturar relaciones no lineales que quizás se nos escaparon.