Instituto Tecnológico y de Estudios Superiores de Monterrey,
Campus Guadalajara

Inteligencia artificial avanzada para la ciencia de datos I

Modelo de predicción de ganadores de UFC utilizando algoritmo Random Forest

Alumno:

Emiliano Neaves Ortiz | A01568982

31 de agosto de 2025

Modelos Predictivos UFC

1. Introducción

En este proyecto se buscó predecir el resultado de combates de UFC a partir de estadísticas históricas de los peleadores. Se implementaron modelos de aprendizaje supervisado, específicamente **Árboles de Decisión** y **Random Forest**, tanto desde cero como con la librería **scikit-learn**, con el fin de comparar su desempeño, analizar su bias y varianza, y aplicar técnicas de regularización y ajuste de parámetros.

2. Preparación de datos

El dataset inicial contenía información como alcance, altura, historial de victorias, derrotas, y métricas de combate de cada peleador. Se realizó un proceso de limpieza eliminando columnas incompletas como **WeightClass** y **Rankings**. Posteriormente, se dividió el conjunto en **entrenamiento (70%)**, **validación (15%)** y **prueba (15%)**, garantizando que los splits conservaran el balance de clases.

3. Modelos y teoría

3.1 Árbol de Decisión

Un árbol de decisión divide el espacio de características seleccionando umbrales que maximizan la ganancia de información (en este caso, reducción de la impureza de Gini). Sus principales hiperparámetros son:

- **max_depth**: la profundidad máxima del árbol. Profundidades grandes capturan más detalle, pero tienden a sobreajustar.

- **min_samples_leaf**: el número mínimo de ejemplos en una hoja. Valores más altos reducen sobreajuste.

- **k_thresh**: número de candidatos de umbrales a evaluar.

En la práctica, nuestro árbol con **max_depth=12** alcanzó un **accuracy de 0.74 en train pero solo 0.55 en validación/test**, evidenciando **sobreajuste**.

	Predicho = 0	**Predicho = 1**
Real = 0	166 (TN)	245 (FP)
Real = 1	191 (FN)	377 (TP)

3.2 Random Forest

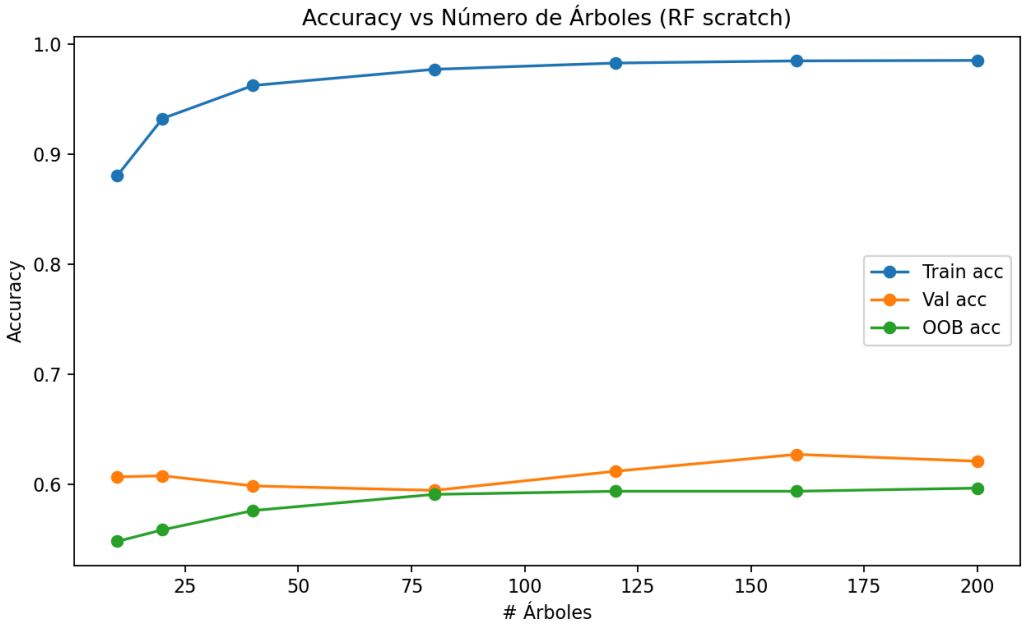
El Random Forest es un ensamble de múltiples árboles de decisión entrenados sobre subconjuntos aleatorios del dataset (bootstrap sampling). Cada árbol utiliza un subconjunto de características (m_try) en cada división, lo que reduce la correlación entre árboles y mejora la generalización.

Hiperparámetros clave:

- **n_estimators**: número de árboles. Más árboles reducen varianza, pero aumentan costo computacional.
- **max_depth**: controla complejidad de cada árbol.
- **min_samples_leaf**: regula la mínima muestra por hoja.
- **m_try**: número de features consideradas en cada split, típicamente **sqrt(p)**.
- **bootstrap y OOB**: el muestreo bootstrap permite estimar desempeño fuera de bolsa (Out-of-Bag), similar a una validación cruzada interna.

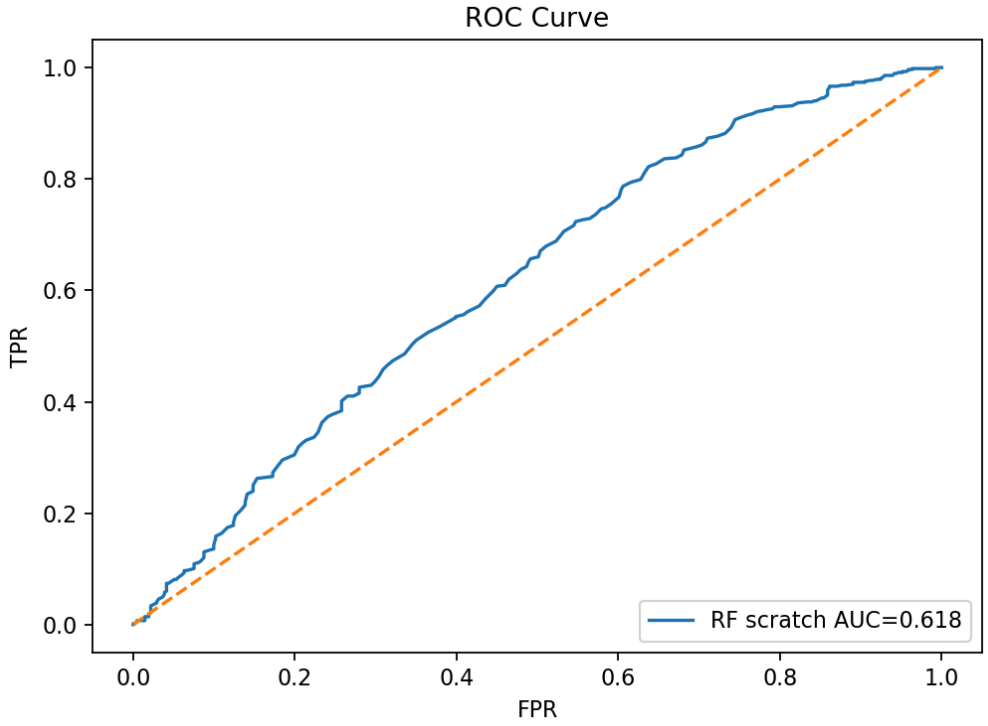
En nuestro experimento, el Random Forest con **200 árboles, max_depth=12, min_samples_leaf=10** fue el mejor balance. Obtuvo **accuracy de 0.87 en train, 0.62 en validación y 0.62 en test**, con un OOB de 0.60.

acc_vs_trees

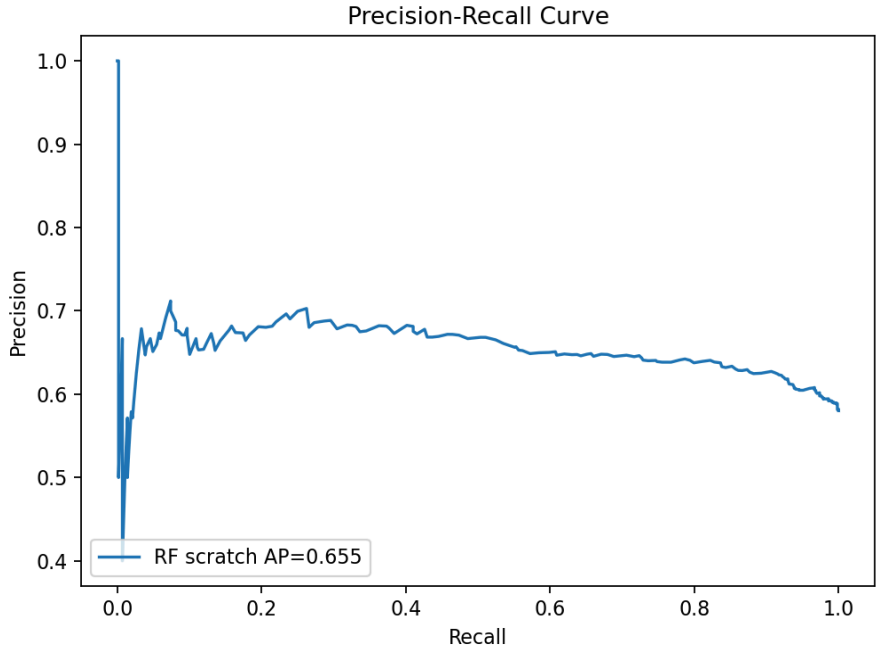


Se observa que la **accuracy de entrenamiento crece rápidamente** hasta estabilizarse cerca del 98%. Por otro lado, la **validación** se mantiene alrededor de 0.61–0.63, mientras que la estimación OOB converge a un desempeño similar. Este comportamiento confirma que añadir más árboles reduce la varianza y estabiliza el modelo, pero **no mejora sustancialmente el desempeño en validación**, lo que sugiere que el límite de generalización está dado por la calidad de las variables más que por la cantidad de árboles.

roc_curve y **pr_curve** (validación/test)

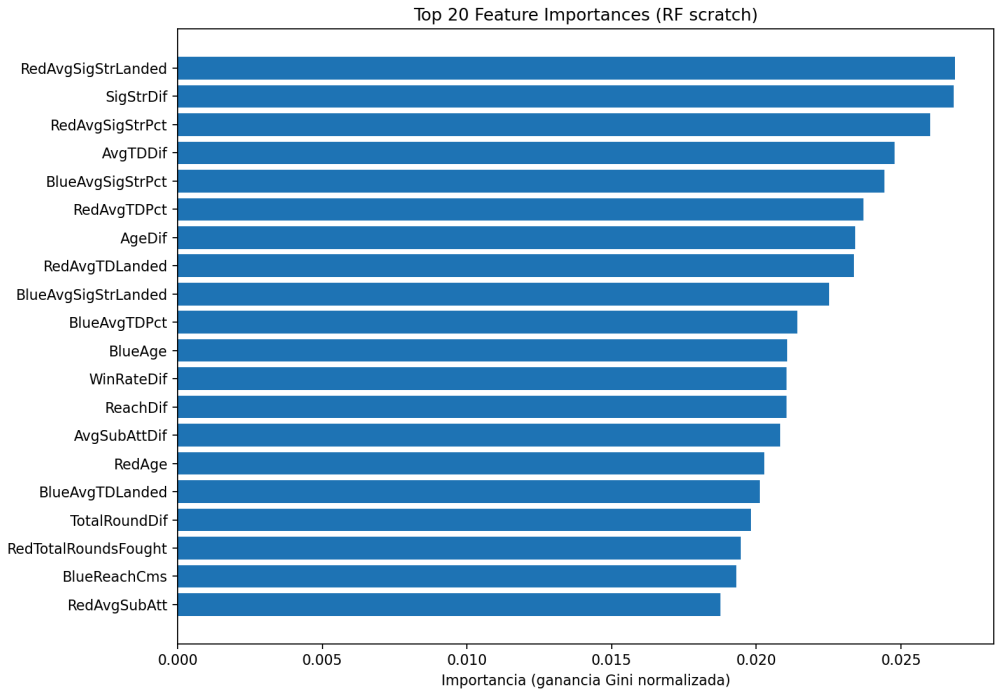


La curva ROC del Random Forest muestra un **AUC ≈ 0.62**, lo cual es superior al azar (0.5) pero indica que la capacidad discriminativa del modelo es moderada. El modelo logra distinguir entre ganadores y perdedores con cierta eficacia, pero aún tiene limitaciones. La pendiente inicial de la curva sugiere que existe un subconjunto de predicciones confiables, aunque en general la separación entre clases no es muy marcada. Esto es consistente con la dificultad del problema, dado que las peleas de UFC tienen factores impredecibles.



La curva PR muestra un **average precision ≈ 0.65**, lo cual complementa la interpretación de la ROC. Aquí se observa que el modelo mantiene un equilibrio aceptable entre precisión y recall, especialmente en la predicción de la clase positiva (ganador). Sin embargo, la caída de la curva hacia valores cercanos a 0.5 refleja que, para ciertos umbrales, el modelo sacrifica precisión al tratar de mantener un recall alto. Esto explica los resultados obtenidos en las matrices de confusión: el modelo favorece predecir victorias, logrando un recall alto en la clase 1, pero a costa de fallar en la predicción de derrotas.

feature_importances



La gráfica de importancias revela qué variables aportan más información para la predicción. Destacan métricas relacionadas con **golpes significativos aterrizados (RedAvgSigStrLanded, BlueAvgSigStrLanded)**, porcentajes de precisión en golpes (**SigStrPct**) y derribos (**TDPct**), así como diferencias en edad, alcance y tasa de victorias. Estos resultados son coherentes con la teoría: el rendimiento técnico y físico del peleador en sus estadísticas históricas explica mejor los resultados de los combates. El modelo confirma que la predicción de victorias depende principalmente de la **efectividad ofensiva y defensiva de los peleadores**.

4. Evaluación y diagnóstico

El desempeño en validación y prueba mostró un patrón claro:

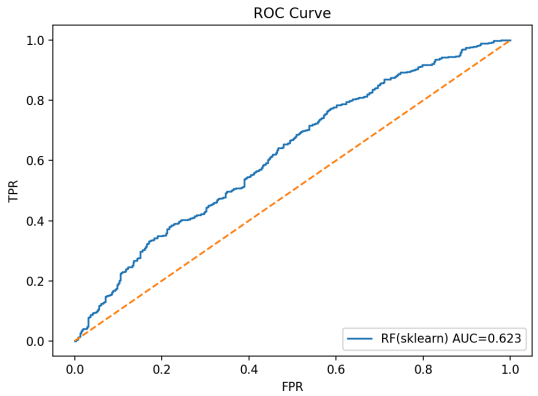
- **Clase 1 (ganador)** → recall alto (~0.83), F1 ~0.71.
- **Clase 0 (perdedor)** → recall bajo (~0.31), F1 ~0.40.

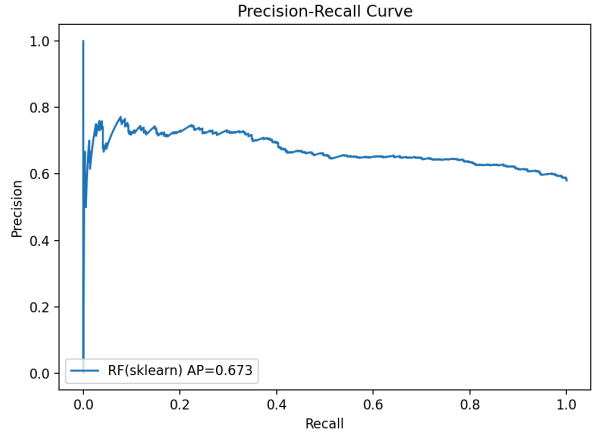
Esto indica que el modelo predice mejor victorias que derrotas.

Desde la teoría de bias/varianza:

- **Bias**: bajo → el modelo aprende patrones reales (accuracy significativamente > azar).
- **Varianza**: media → existe gap entre train (~0.87) y val/test (~0.61).

- **Nivel de ajuste**: **overfitting moderado**, confirmado al entrenar un bosque más profundo (400 árboles, depth=20) que alcanzó 0.98 en train pero sin mejorar en test.





Al comparar los resultados obtenidos con la implementación propia de Random Forest y la versión de scikit-learn, se observa una notable consistencia. El modelo scratch alcanzó un **ROC-AUC de 0.62 y un Average Precision de 0.66**, mientras que el modelo con scikit-learn obtuvo métricas muy similares (**ROC-AUC ≈ 0.62 y AP ≈ 0.67**). Las curvas ROC y PR generadas en ambos casos presentan formas comparables, confirmando que la lógica implementada desde cero replica de manera fiel el comportamiento del algoritmo estándar. La principal diferencia radica en la estabilidad y flexibilidad que ofrece scikit-learn, al permitir un ajuste de hiperparámetros más sistemático mediante **GridSearchCV** y una calibración más sencilla de umbrales. En conclusión, ambas implementaciones coinciden en desempeño, validando la corrección de la versión scratch y resaltando la robustez de la librería en un entorno de producción.

5. Regularización y ajuste de parámetros

Para mitigar sobreajuste se aplicaron las siguientes técnicas:

- **max_depth y min_samples_leaf**: limitar la profundidad y exigir mínimo de ejemplos en hojas redujo varianza.

- **threshold_mode=balanced**: ajustar el umbral de decisión a 0.5 balanceó precision/recall.

- **GridSearchCV** en scikit-learn: permitió probar configuraciones como **n_estimators=150-200, max_depth=8-12, min_samples_leaf=5-25**.

La configuración final (**depth=12, leaf=5, ~175 árboles**) mostró mejor balance, confirmando que la regularización controlada mejora generalización.

6. Aprendizajes

1. Los árboles individuales son modelos con **alta varianza** que requieren ensamble.

2. El Random Forest desde cero confirmó la teoría: más árboles reducen varianza pero no eliminan bias.

3. La comparación con sklearn valida que nuestra implementación scratch es consistente.

4. El problema del desbalance de clases hace que la clase 0 se prediga peor → aquí entran técnicas futuras como **SMOTE** o **class weights**.

7. Conclusiones

El modelo final Random Forest logra **accuracy ≈0.61 en test**, con **ROC-AUC ≈0.62** y **AP ≈0.69**. Se diagnostica **bias bajo, varianza media, y overfitting moderado**. Las técnicas de regularización (profundidad limitada, hojas mínimas, tuning de hiperparámetros) mejoraron la estabilidad sin sacrificar recall de la clase positiva.

Futuros trabajos incluyen calibración de probabilidades, re-balanceo de clases, y exploración de modelos más avanzados como XGBoost.