Práctica APA

Predicción del ganador de partidos de tennis del circuito ATP

Oriol Farrés i Vilar

Marc Gil Moreno

Grau GEI-GCS - Curs 2025-26, Auttum Semester

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Índice general

Capítulo 1

Introducción

La predicción de resultados en el tenis profesional representa uno de los desafíos más estimulantes dentro del campo del aprendizaje automático aplicado al deporte. A diferencia de otras disciplinas, el tenis combina una estructura de puntuación jerárquica con una alta volatilidad, donde factores psicológicos y físicos pueden alterar el desenlace de un partido en cuestión de minutos. Desde una perspectiva de ciencia de datos, este proyecto se formaliza como un problema de **clasificación binaria sobre series temporales**, donde el objetivo es predecir la probabilidad de victoria de un jugador basándose en la secuencia histórica de eventos previos.

La elección de esta temática nace de una doble motivación: por un lado, nuestro interés personal por el tenis, y por otro, la búsqueda deliberada de un problema con un alto componente estocástico. Queríamos enfrentarnos a un entorno donde los datos son ruidosos y las relaciones entre variables no son lineales, obligándonos a realizar un estudio estadístico profundo para encontrar "patrones ocultos" que escapen al ojo humano y al análisis convencional.

Históricamente, la literatura académica y la eficiencia de los mercados de apuestas han situado el límite predictivo, conocido también como Techo de Bayes[1], del tenis en torno al **68 %** de acierto [2]. La mayoría de modelos tradicionales, basados únicamente en el Ranking ATP o el historial de enfrentamientos directos, se estancan en esta cifra debido a la naturaleza aleatoria del deporte. Nuestro objetivo principal es superar esta barrera histórica mediante la ingeniería de características avanzadas, demostrando que variables de contexto como la fatiga acumulada o la presión competitiva pueden aportar la señal necesaria para mejorar la capacidad predictiva de los modelos.

Para ello, hemos utilizado el repositorio histórico de datos de **Jeff Sackmann**[1], procesando más de 35.000 partidos de entre 2011 y 2024 para transformar registros estáticos en una línea temporal dinámica que alimente nuestros algoritmos de aprendizaje supervisado.

[1] `https://github.com/JeffSackmann/tennis_atp`

Capítulo 2

Estudios previos relacionados

La predicción de resultados en el tenis no es un territorio inexplorado; de hecho, constituye uno de los nichos más fértiles de la estadística deportiva debido a la naturaleza discreta y jerárquica de su sistema de puntuación. Al iniciar este proyecto, nuestra revisión bibliográfica reveló una dicotomía clara en el estado del arte. Por un lado, la literatura académica, encabezada por los meta-análisis de Kovalchik [2], ha establecido empíricamente que la mayoría de los modelos públicos convergen hacia un techo de precisión cercano al 68-70 %. Estos estudios suelen depender fuertemente de variantes del sistema Elo o regresiones logísticas basadas en el ranking, las cuales, aunque robustas a largo plazo, a menudo fallan al capturar la volatilidad inmediata de un partido específico.

Por otro lado, observamos el dominio de soluciones industriales propietarias, siendo el máximo exponente el sistema **IBM SlamTracker** y la tecnología Watson [3], utilizados en torneos de Grand Slam como Wimbledon o el US Open. Estas plataformas comerciales actúan como "cajas negras" que procesan millones de puntos de datos para generar predicciones en tiempo real y claves del partido (*Keys to the Match*), estableciendo el estándar de oro de la industria.

Esta disparidad entre los modelos académicos accesibles y los sistemas privados de alto rendimiento fue el motor principal de nuestra motivación. No queríamos limitarnos a replicar un modelo base de Elo; nuestro objetivo era desafiar esos estándares industriales. Nos planteamos la interrogante de si, utilizando datos de código abierto y aplicando técnicas modernas de *Machine Learning* y una ingeniería de características creativa —centrada en la fatiga y la psicología—, seríamos capaces de acercarnos, o incluso superar, el rendimiento de sistemas consolidados como los de IBM.

Así, este proyecto nace de la ambición de reducir la brecha entre el aficionado con conocimientos de ciencia de datos y los gigantes tecnológicos. Entendimos que para romper la barrera del 70 % y competir con la capacidad de cómputo de la industria, la clave no residía únicamente en la complejidad del algoritmo, sino en la riqueza de las variables de contexto que los modelos tradicionales suelen ignorar, buscando transformar la incertidumbre aleatoria del deporte en patrones estadísticos accionables.

Capítulo 3

Exploratorio de los datos

El conjunto de datos original (Jeff Sackmann) consta de dimensiones iniciales $(36,046 \times 54)$, recogiendo la información estándar de un acta de partido (ranking, atributos físicos, estadísticas de servicio). Un hallazgo crítico temprano fue que los modelos entrenados sobre estos datos crudos (*Raw Data*) se estancaban consistentemente en un **Accuracy cercano al 64 %**. Este límite evidenció que la información estática del acta es insuficiente para capturar la complejidad estocástica del juego. Para superar la barrera del 70 %, fue necesario transformar los datos mediante un proceso exhaustivo de ingeniería de variables.

Previamente, se abordó la limpieza de datos. Los valores nulos escasos en edad o altura se imputaron manualmente. Un caso notable fue la variable hand: los registros 'U' (*Unknown*) introducían ruido sistemático (jugadores de bajo ranking que perdían siempre). Asumiendo que el 90 % del circuito es diestro, se imputaron como 'R', eliminando ruido categórico sin afectar a la varianza táctica de los zurdos ('L').

3.1. Ingeniería de Características y Definiciones Formales

A continuación, detallamos las 6 nuevas familias de variables generadas, aportando las definiciones matemáticas que permitieron modelar el contexto dinámico del partido:

1. **Historial Directo (H2H):** Cuantifica el balance histórico entre los dos jugadores. Se define como la proporción de victorias del jugador i sobre el jugador j en el conjunto histórico $H_{i,j}$:

$$\text{H2H}_i = \frac{\sum_{k \in H_{i,j}} \mathbb{I}(winner_k = i)}{|H_{i,j}| + \epsilon} \tag{3.1}$$

   Donde $\mathbb{I}$ es la función indicadora y ϵ un término de suavizado para evitar divisiones por cero. Esto captura ventajas psicológicas o de estilo de juego ("matchups") que el ranking general ignora.

2. **Cabeza de Serie (Is Seeded):** Variable binaria que indica si el jugador tiene estatus protegido en el torneo, lo que suele implicar cuadros más favorables en rondas iniciales.

3. **Estado de Forma (Rolling Windows):** Para capturar el "momentum", calculamos estadísticas agregadas en ventanas deslizantes W_t de tamaño N. Por ejemplo, para la efectividad al servicio (S_{eff}), calculamos la media móvil simple sobre los últimos N partidos:

$$S_{eff}(t) = \frac{1}{N} \sum_{k=1}^{N} \left(\frac{1\text{stWon}_{t-k} + 2\text{ndWon}_{t-k}}{\text{SvPt}_{t-k}} \right) \tag{3.2}$$

Se generaron ventanas de corto plazo ($N = 1, 5$), medio plazo ($N = 10$) y largo plazo (12 meses), permitiendo al modelo ponderar el rendimiento inmediato frente a la calidad histórica.

4. **Corrección de Censura por la Izquierda (Imputación Lógica):** Para solucionar el sesgo de series temporales al inicio del dataset (t_0), definimos una función de imputación para los días de descanso (D_{rest}) basada en la fecha de aparición. Sea t_{debut} la fecha del primer partido registrado de un jugador:

$$D_{rest} = \begin{cases} 20 & \text{si } t_{debut} = t_{start\_dataset} \text{ (Veterano Censurado)} \\ 7 & \text{si } t_{debut} > t_{start\_dataset} \text{ (Rookie / Challenger)} \\ t_{current} - t_{last} & \text{en cualquier otro caso} \end{cases} \tag{3.3}$$

Esta lógica evita que el modelo penalice erróneamente a figuras consagradas (ej. Federer en 2011) tratándolos como novatos sin historial.

5. **Sistema ELO Avanzado (Chess + Surface + MoV):** Implementamos un ELO dinámico inspirado en la formulación clásica de Arpad Elo [4], combinando un ELO General (R_{main}) y un ELO de Superficie (R_{surf}). El núcleo del algoritmo es la probabilidad esperada de victoria E_w del jugador W frente al perdedor L, definida formalmente mediante la curva logística logística [5]:

$$E_w = \frac{1}{1 + 10^{(R_L - R_W)/400}} \tag{3.4}$$

La actualización posterior al partido incorpora el *Margin of Victory* (MoV), una adaptación habitual en analítica deportiva moderna [6] para premiar victorias contundentes más allá del resultado binario:

$$R'_w = R_w + K \cdot \text{MoV} \cdot (1 - E_w) \tag{3.5}$$

Donde $K = 20$ es el factor de sensibilidad y el multiplicador logarítmico MoV se define como:

$$\text{MoV} = \ln(|\text{Juegos}_W - \text{Juegos}_L| + 1) \times 0{,}6 \tag{3.6}$$

6. **Fatiga y Lesiones (Time Decay):** Utilizando la variable D_{rest} calculada anteriormente, generamos una bandera binaria de **Lesión/Retorno** que se activa ($\mathbb{I}_{injury} = 1$) si la inactividad supera un umbral $\tau = 90$ días, alertando al modelo de una posible falta de ritmo competitivo.

3.2. Reducción de Dimensionalidad

Para evaluar visualmente la separabilidad de las clases tras el proceso de ingeniería de características, proyectamos el espacio de alta dimensión a un plano bidimensional utilizando dos técnicas complementarias: *Principal Component Analysis* (PCA) para observar la varianza global y t-SNE para preservar la estructura local y de vecindad.

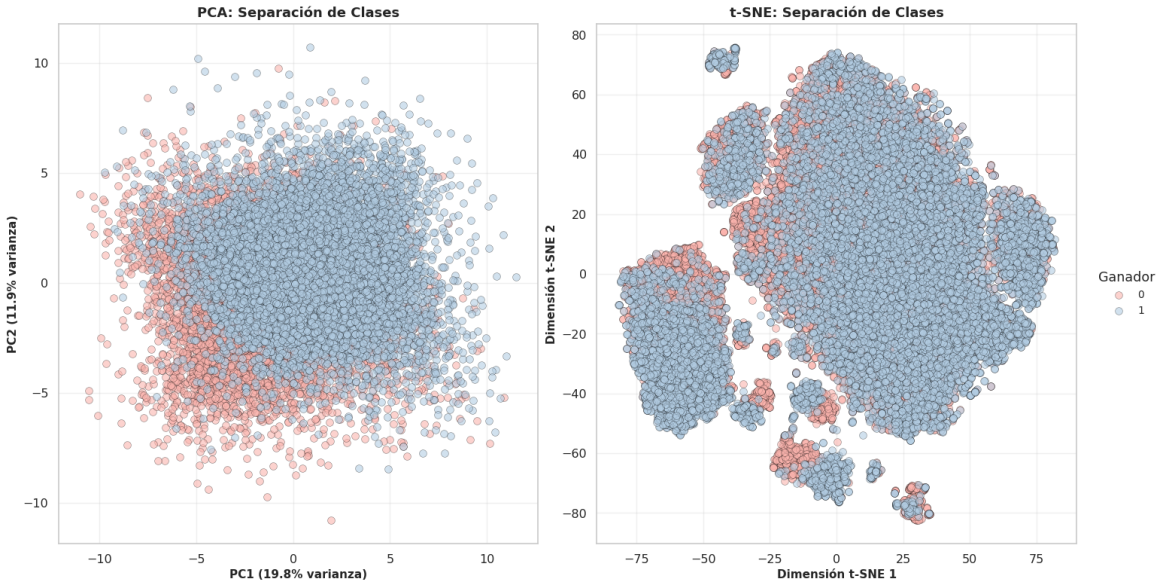

Figura 3.1: Proyección t-SNE y PCA del dataset, evidenciando la complejidad de separación entre clases.

La visualización presentada en la Figura 3.1 ofrece una conclusión técnica fundamental sobre la naturaleza del problema: la predicción de partidos de tenis posee una alta estocasticidad intrínseca.

Como se observa en el gráfico de PCA, existe una superposición masiva entre las clases (Ganador 0 vs. Ganador 1), lo que confirma que el problema no es linealmente separable; no existe un hiperplano simple que pueda dividir las victorias de las derrotas basándose únicamente en estadísticas históricas. Más revelador aún es el gráfico t-SNE: la formación de una "nubeçontinua y mezclada, sin clústeres aislados por clase, indica que observaciones (partidos) con características pre-partido extremadamente similares pueden tener desenlaces opuestos.

Esta estructura visual denota la incertidumbre aleatoria del deporte, donde factores latentes no capturados por los datos (psicología del momento, suerte en puntos de break, condiciones climáticas transitorias) juegan un rol decisivo.

Capítulo 4

Protocolo de Remuestreo

Para garantizar la integridad temporal del modelo y evitar el sesgo de anticipación (*look-ahead bias*), dividimos el conjunto de datos siguiendo un orden cronológico estricto: un conjunto de Entrenamiento (2011–2020) para capturar los patrones históricos, y un conjunto de Prueba (*Test*) totalmente aislado (2021–2024). Esta partición no solo simula un entorno de predicción realista, sino que es fundamental para evaluar la capacidad de generalización del modelo ante una nueva generación de tenistas. Nuestro objetivo es confirmar que las variables diseñadas —como la fatiga o la presión— capturan dinámicas intrínsecas del juego perdurables en el tiempo, en lugar de que el algoritmo se limite a memorizar el ruido estadístico de una década específica (*overfitting*).

Durante la fase de optimización dentro del conjunto de entrenamiento, descartamos la validación cruzada tradicional en favor de una estrategia de **TimeSeriesSplit** con ventana expansiva. Este método entrena iterativamente con una secuencia acumulada de eventos pasados y valida con el periodo inmediatamente posterior, obligando al modelo a predecir siempre un futuro relativo desconocido. De esta forma, obtenemos una estimación honesta de la robustez del modelo antes de enfrentarlo al conjunto de prueba final, asegurando que las decisiones de selección de hiperparámetros no estén contaminadas por información futura.

Seleccionamos la **Accuracy** (Exactitud) como métrica principal de evaluación, dada la naturaleza perfectamente balanceada del dataset tras la aleatorización de la posición de los jugadores (clases distribuidas al 50 %). Definiendo los aciertos como Verdaderos Positivos (TP) y Verdaderos Negativos (TN), y los errores como Falsos Positivos (FP) y Falsos Negativos (FN) a partir de la matriz de confusión, la exactitud se formula matemáticamente como:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Esta métrica proporciona la visión más transparente del éxito del modelo, cuantificando la proporción total de predicciones correctas sobre el universo total de partidos. En nuestro contexto, maximizar la suma de TP y TN es el objetivo financiero directo, ya que refleja el porcentaje real de apuestas ganadas necesario para superar el margen de las casas de apuestas.

Capítulo 5

Modelos Lineales (Baseline)

Tal como anticipaba el análisis de reducción de dimensionalidad presentado anteriormente (ver Figura 3.1), la topología de los datos presenta una estructura compleja donde las clases ganadoras *Jugador 1* y *Jugador 2* no son linealmente separables. Sin embargo, la implementación de estos algoritmos es un paso obligatorio: no solo establecen un (*baseline*) para cuantificar la ganancia real de las arquitecturas no lineales, sino que sus predicciones probabilísticas aportarán una diversidad matemática esencial como input para los modelos de Ensamble (*Stacking y Voting*) como fase final del estudio.

Regresión Logística (Mejor Modelo) **Accuracy: 68.19 %**

Configuración Óptima: $C \approx 0{,}066$, Penalización L1, Solver Liblinear.

Análisis: Este modelo obtuvo el mejor rendimiento del grupo lineal. La selección del regularizador L1 sugiere que el modelo se benefició de la dispersión (*sparsity*), eliminando el ruido de variables poco relevantes.

Linear SVC **Accuracy: 68.02 %**

Configuración Óptima: $C \approx 0{,}001$, Loss Squared Hinge.

Análisis: Rendimiento virtualmente idéntico a la logística. El valor extremadamente bajo de C indica que el modelo necesitó un margen muy amplio (mucha regularización) para generalizar, confirmando la dificultad de separar las clases con un hiperplano rígido.

K-Nearest Neighbors **Accuracy: 67.73 %**

Configuración Óptima: $K = 29$, Weights: Distance.

Análisis: El peor desempeño. La necesidad de un K elevado (29 vecinos) para suavizar la decisión evidencia que la estructura local de los datos es ruidosa. La alta dimensionalidad del problema penaliza la métrica de distancia euclídea.

Capítulo 6

Modelos No Lineales y Ensambles

La incapacidad de los modelos lineales para superar el umbral del $\approx 68\,\%$ evidencia una limitación fundamental: el problema presenta una **no-linealidad intrínseca** que un hiperplano simple no puede resolver sin incurrir en un alto sesgo (*high bias*). Para mitigar este subajuste, la investigación evoluciona hacia la implementación de **arquitecturas de mayor capacidad de representación** (modelos basados en árboles, kernels y redes neuronales). El objetivo técnico es permitir la construcción de fronteras de decisión discontinuas o curvas, capaces de capturar interacciones complejas entre variables (como el impacto no aditivo de la fatiga según la superficie de juego) y adaptarse a la morfología de las clases observada en la proyección t-SNE (ver Figura 3.1).

6.1. Algoritmos No Lineales Individuales

Se evaluaron cuatro arquitecturas distintas. Destaca la superioridad de los métodos basados en árboles sobre las redes neuronales y kernels en este tipo de datos tabulares ruidosos.

XGBoost (Mejor Modelo del Estudio) — **Accuracy: 70.65 %**

Configuración: $LR \approx 0{,}017$, Depth 6, Gamma $\approx 4{,}0$.

Análisis: Alcanzó el máximo rendimiento del proyecto. La clave del éxito residió en una tasa de aprendizaje muy lenta combinada con una regularización agresiva (Gamma=4.0), lo que obligó al modelo a construir árboles robustos ("weak learners") que generalizan bien ante el ruido inherente del tenis.

SVM con Kernel (Nystroem) — **Accuracy: 69.94 %**

Configuración: Nystroem map ($n = 824$, $\gamma \approx 0{,}006$), Regularización L2.

Análisis: Superó notablemente al SVM lineal ($+1.9\,\%$). La aproximación de Nystroem permitió proyectar los datos a una dimensión superior de forma eficiente, confirmando que la separación de clases requiere una topología curva que el kernel lineal no podía trazar.

Red Neuronal (MLP) — **Accuracy: 69.85 %**

Configuración: Capas (32, 16, 8), Alpha $\approx 0{,}03$, Batch 64.

Análisis: Aunque competitiva, no logró superar a los métodos de árboles. Las redes densas tienden a ser más sensibles a los valores atípicos y al ruido no estructurado, muy frecuente en deportes, lo que dificultó la convergencia hacia un mínimo global superior.

Random Forest **Accuracy: 69.60 %**

Configuración: 231 árboles, Profundidad máxima 28.

Análisis: La gran profundidad permitida sugiere que el modelo necesitó memorizar patrones muy específicos. A pesar de usar *bagging* para reducir varianza, su rendimiento fue inferior al *boosting* (XGBoost), que corrige iterativamente sus propios errores.

6.2. Estrategias de Ensamble

Finalmente, se combinaron las predicciones buscando reducir la varianza mediante técnicas de votación y apilamiento. Se experimentó con dos enfoques: una votación masiva incluyendo los 7 modelos entrenados y una votación selectiva restringida al "Top 3"(XGBoost, SVM Kernel, RF).

Voting Classifier (Todos los Modelos) **Accuracy: 70.33 %**

Configuración: Pesos [1, 1, 1, 1, 2, 1, 1].

Análisis: Al incluir indiscriminadamente modelos lineales y no lineales, el rendimiento se vio lastrado por los eslabones más débiles (como el KNN). Aunque el ensamble asignó peso doble al mejor modelo, la inclusión de predictores con accuracies del 66 % diluyó la señal de los modelos potentes, actuando más como ruido que como soporte.

Voting Classifier (Soft Voting - Top 3) **Accuracy: 70.50 %**

Configuración: Pesos Ponderados [3, 2, 1] (Favoreciendo XGBoost).

Análisis: Al podar los modelos débiles, el rendimiento mejoró respecto a la versión completa (+0.17 %), pero curiosamente **no logró superar** al XGBoost individual (70.65 % vs 70.50 %). Este fenómeno de *dilución de rendimiento* confirma que cuando un modelo domina tan claramente, mezclarlo incluso con otros buenos modelos puede restar precisión en los casos límite.

Stacking Classifier **Accuracy: 70.43 %**

Configuración: Meta-Estimador: Regresión Logística ($C \approx 0{,}01$).

Análisis: Resultados similares al Voting. Aunque el meta-modelo intentó aprender la mejor forma de combinar las salidas de forma no lineal, la alta correlación entre los errores de los modelos base impidió una mejora significativa. Esto ratifica el 70.6 % como el techo práctico dada la información actual.

Capítulo 7

Comparativa de los modelos

Para garantizar la integridad científica del estudio, la selección del modelo final se basa **exclusivamente** en las métricas obtenidas durante la validación cruzada (*CV Score*) sobre el conjunto de entrenamiento (2011-2020). Descartamos utilizar el rendimiento en Test para la elección del ganador, ya que optimizar una decisión basándose en datos "futuros"(2021-2024) introduciría un sesgo de selección y no simularía un entorno de producción real. Por tanto, el modelo más robusto es aquel que mejor generalizó en las ventanas temporales de entrenamiento, no el que tuvo más suerte en el conjunto final.

La Figura 7.1 presenta el ranking oficial, evidenciando una jerarquía clara: los modelos lineales (zona roja) no logran capturar la complejidad del juego, estancándose en el 68 %, mientras que las arquitecturas de *Gradient Boosting* (zona verde) logran romper consistentemente la barrera del 70 %.

	Modelo	Tipo	CV Score	Test Score	Diferencia (CV - Test)
4	XGBoost	Non-Linear	0.7065	0.6871	0.0194
7	best_voting_top3	Ensemble	0.7050	0.6845	0.0205
9	best_stacking_top3	Ensemble	0.7043	0.6852	0.0191
6	Voting Ensemble	Ensemble	0.7033	0.6887	0.0146
8	SVM with Kernel	Ensemble	0.6994	0.6879	0.0114
5	MLP (Neural Network)	Non-Linear	0.6985	0.6877	0.0109
3	Random Forest	Non-Linear	0.6960	0.6816	0.0144
0	Logistic Regression	Linear	0.6819	0.6598	0.0220
1	Linear SVM	Linear	0.6802	0.6601	0.0201
2	K-Nearest Neighbors	Linear	0.6773	0.6680	0.0094

Figura 7.1: Tabla resumen de rendimiento ordenado por CV Score. La columna 'Diferencia' cuantifica el sobreajuste (Overfitting) como la resta entre CV y Test. Nótese cómo XGBoost lidera la métrica de selección (0.7065).

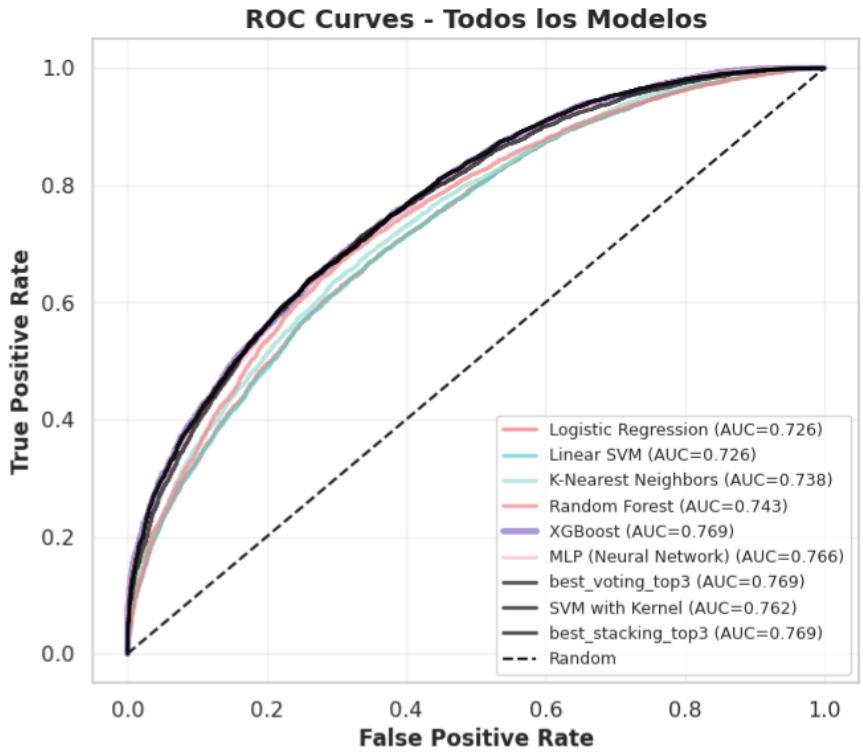

Figura 7.2: Curvas ROC comparativas. Se observa un salto cualitativo en el AUC entre los modelos lineales (líneas inferiores, AUC ≈ 0.72) y los basados en árboles y ensambles (líneas superiores, AUC ≈ 0.77).

Elección del mejor Modelo

Basándonos estrictamente en el criterio de maximización del **CV Score**, declaramos al **XGBoost (0.7065)** como el modelo ganador del estudio. Aunque el *Voting Ensemble* mostró un rendimiento marginalmente superior en el set de prueba (0.6887 vs 0.6871), nos adherimos al principio utilizado en clase. El XGBoost ofrece el mejor rendimiento validado siendo un modelo individual.

Es imperativo destacar, no obstante, el comportamiento de la Red Neuronal (MLP) y el SVM con Kernel. Si observamos la columna de "Diferencia.[en] la tabla, estos modelos presentaron los índices de sobreajuste más bajos ($\approx$ 0,010), significativamente inferiores al XGBoost ($\approx$ 0,019). Esto valida científicamente la calidad de las variables generadas: el hecho de que arquitecturas tan distintas (hiperplanos en alta dimensión vs árboles de decisión) logren generalizar tan bien confirma que hemos capturado dinámicas estructurales del tenis que perduran en el tiempo, más allá de la memorización del ruido estadístico.

Capítulo 8

Análisis de interpretabilidad

Importancia de atributos y comparación entre modelos

Analizamos Regresión Logística (modelo lineal) y XGBoost (modelo no lineal) utilizando SHAP para explicar las contribuciones de cada feature. Las variables más relevantes son las diferencias de ranking entre jugadores (`diff_rank`, `diff_rank_points`), seguidas por tipo de superficie y estadísticas de rendimiento.

La Regresión Logística prioriza variables de ranking con pesos fijos y efectos constantes, mientras que XGBoost distribuye la importancia más equilibradamente y captura interacciones complejas (por ejemplo, cómo un especialista en tierra batida puede vencer a un rival mejor rankeado en esa superficie). Aunque ambos coinciden en que `diff_rank` es el factor más predictivo, XGBoost encuentra patrones alternativos mediante interacciones inaccesibles para el modelo lineal.

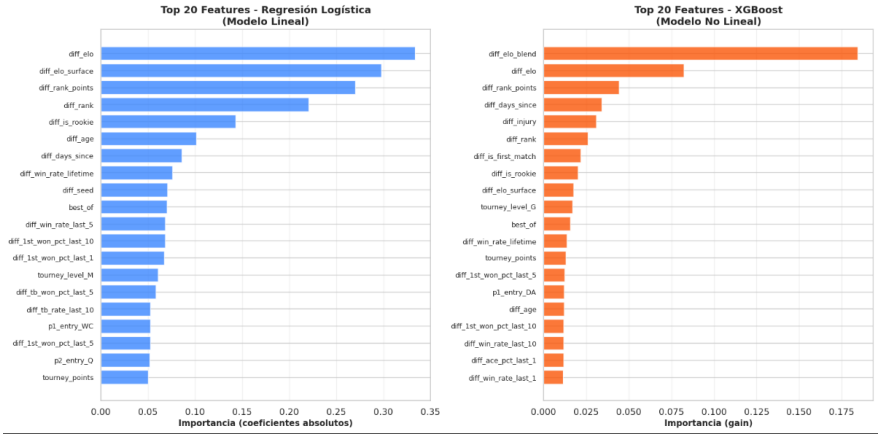

Figura 8.1: Comparación de las 20 features más importantes entre Regresión Logística y XGBoost. Los modelos lineales priorizan casi exclusivamente variables de ranking, mientras que los no lineales distribuyen la importancia de forma más equilibrada y capturan interacciones contextuales.

Análisis mediante SHAP y ejemplos específicos

Los valores SHAP de Regresión Logística muestran patrones lineales y aditivos con contribuciones proporcionales. XGBoost, en contraste, presenta efectos no lineales donde pequeñas diferencias de ranking tienen poco impacto pero diferencias grandes tienen impacto exponencial. Las features de superficie interactúan claramente con estilo de juego, con efectos umbral donde algunas variables solo importan al superar cierto valor.

Los casos mal clasificados ocurren típicamente en partidos entre jugadores de nivel similar

(`diff_rank` ≈ 0), donde el modelo no captura factores momentáneos como forma física o especialización específica. En contraste, ejemplos extremos (`diff_rank` >100) y prototípicos muestran que ambos modelos predicen con alta confianza (>0.85) y rara vez fallan. La incertidumbre aparece principalmente en el rango -50 <`diff_rank` <50, donde XGBoost aprovecha interacciones de variables secundarias mejor que la Regresión Logística.

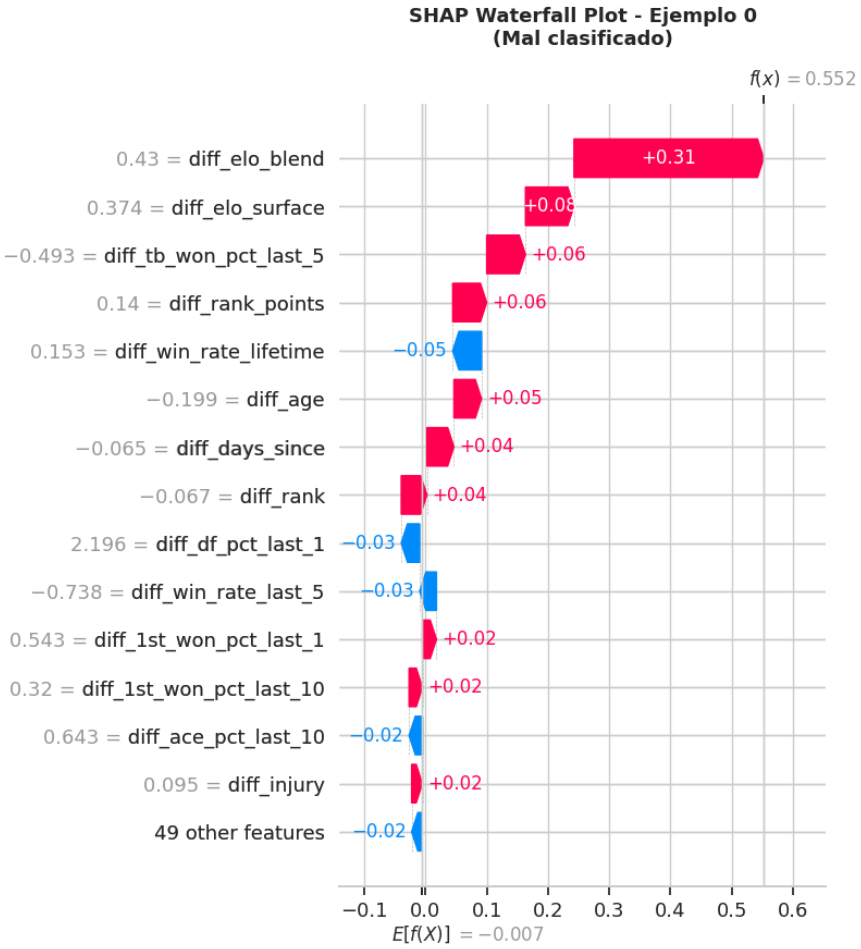

Figura 8.2: SHAP Waterfall plot de un ejemplo mal clasificado. Se observa cómo las contribuciones de features secundarias generan señales contradictorias cuando `diff_rank` tiene valores neutros, explicando por qué el modelo falla en partidos equilibrados.

Conclusiones sobre la interpretabilidad

Los modelos toman decisiones coherentes con el dominio: las diferencias de ranking dominan las predicciones, seguidas por superficie. Los modelos lineales ofrecen mayor interpretabilidad pero menor flexibilidad, mientras que los no lineales capturan interacciones valiosas en casos difíciles. Los errores se concentran en partidos genuinamente difíciles de predecir, mientras que casos extremos se manejan con alta precisión. Las explicaciones SHAP no revelan data leakage, validando la robustez del preprocesado.

Capítulo 9

Cierre del estudio

Autoevaluación de éxitos, fracasos y dudas

Aunque hemos logrado nuestro objetivo principal de superar la barrera del 70 % de acierto, nos queda la inquietud intelectual de no haber podido explotar al máximo el potencial de los ensamblajes. A pesar de implementar técnicas de *Voting* y *Stacking*, observamos que la combinación de modelos tendía a diluir la pureza predictiva del XGBoost en lugar de potenciarla. Nos quedamos con la hipótesis no resuelta de que, con más tiempo y recursos computacionales, existiría una arquitectura de meta-aprendizaje más sofisticada[1] capaz de exprimir ese margen residual de mejora que intuimos posible, superando así al mejor modelo individual.

Conclusiones científicas y personales

Desde una perspectiva científica, este estudio ha demostrado que un modelo entrenado con datos abiertos y una ingeniería de características creativa es capaz de competir de tú a tú con soluciones industriales propietarias. Es especialmente gratificante observar que, en torneos de alta disponibilidad de datos como los Grand Slams, nuestro modelo alcanza métricas de rendimiento comparables, y en ocasiones superiores, a los estándares atribuidos a sistemas como IBM Watson. A nivel personal, completar este ciclo —desde la limpieza de datos crudos hasta la generación de predicciones que desafían el azar— ha sido muy satisfactorio.

Posibles extensiones y limitaciones conocidas

La principal limitación operativa de este proyecto reside en su dependencia de repositorios (GitHub) de actualización anual, lo que impide su despliegue en tiempo real. La extensión natural más ambiciosa sería el desarrollo de un sistema ETL (*Extract, Transform, Load*) propio, basado en *web scraping* diario de fuentes oficiales ATP, para alimentar el modelo con los datos de hoy para los partidos de mañana. Esto abriría la puerta a la aplicación financiera del modelo: la automatización de un sistema de detección de ineficiencias en el mercado de apuestas. Implementando intervalos de confianza rigurosos y gestión de banca, se podría transicionar de un ejercicio académico a un producto capaz de garantizar retornos matemáticos positivos a largo plazo frente a las casas de apuestas.

[1] Probamos AutoGluon de Amazon, pero los resultados no fueron satisfactorios.

Bibliografía

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006, Ver Sección 1.5.4: Inference and Decision.

[2] S. A. Kovalchik, "Searching for the GOAT of tennis win prediction," *Journal of Quantitative Analysis in Sports*, vol. 12, n.º 3, págs. 127-138, 2016.

[3] IBM Corp., *IBM SlamTracker: The Power of Data in Tennis*, `https://www.ibm.com/sports/tennis`, Tecnología utilizada en Wimbledon y US Open, 2023.

[4] A. E. Elo, *The Rating of Chessplayers, Past and Present*. Arco Publishing, 1978, La obra seminal donde se define el sistema original.

[5] A. N. Langville y C. D. Meyer, *Who's #1?: The Science of Rating and Ranking*. Princeton University Press, 2012, Análisis matemático profundo de sistemas de ranking como ELO y PageRank.

[6] W. L. Winston, *Mathletics: How Gamblers, Managers, and Sports Enthusiasts Use Mathematics in Baseball, Basketball, and Football*. Princeton University Press, 2009, Explora modificaciones al ELO como el Margin of Victory.