Predicción de Resultados en la Premier League usando Machine Learning Multiclase

I. Introducción

El análisis de resultados deportivos mediante Machine Learning (ML) ha ganado un enorme interés en los últimos años. En el contexto del fútbol profesional, la predicción de resultados se ha convertido en una de las tareas más desafiantes y ambiciosas dentro del campo del aprendizaje supervisado, debido a la complejidad de este deporte, la cantidad de variables en juego, la escasez de goles, que es la variable definitoria al fin y al cabo y el alto grado de aleatoriedad. La capacidad de anticipar resultados puede ofrecer ventajas estratégicas tanto para analistas deportivos como para departamentos técnicos, casas de apuestas o incluso aficionados avanzados. La predicción del resultado de un partido (victoria local, empate o victoria visitante) representa un desafío clásico en ciencia de datos, que combina múltiples factores numéricos, categóricos y temporales, con un alto grado de incertidumbre.

El presente proyecto se enfoca en la creación de un modelo de clasificación multiclase que intente predecir el resultado de los partidos de la segunda mitad de la temporada 2023/2024 de la Premier League, utilizando como entrenamiento no solo la primera mitad de dicha temporada, sino también todas las temporadas anteriores disponibles. La necesidad surge del deseo de construir un modelo que no solo se ajuste bien al pasado, sino que demuestre capacidad de generalización **temporal**.

El objetivo principal de este trabajo es desarrollar un modelo de Machine Learning capaz de predecir el resultado de partidos futuros en la Premier League, trabajando con múltiples temporadas históricas, incluyendo estadísticas como posesión, xG, tiros a puerta y rendimiento del portero. Para lograr esto, se ha llevado a cabo un riguroso proceso de análisis, ingeniería de características, validación cruzada estructurada por temporada y técnicas avanzadas como stacking de modelos

El proyecto no se limita a encontrar el modelo con mayor precisión, sino a entender qué variables aportan valor predictivo y cómo se comportan los algoritmos ante cambios estructurales en las temporadas. Esto habilita futuras líneas de investigación sobre dinámica de juego, impacto del calendario y análisis basado en rendimiento reciente.

II. Dataset y Análisis Exploratorio (EDA)

El dataset base utilizado fue un archivo consolidado llamado "df\_top5.csv", que incluye partidos de las cinco grandes ligas europeas desde la temporada 2000/2001. Se filtraron los partidos correspondientes a la Premier League, dado que era la liga con mas temporadas, y mas partidos en general en el dataset, un total de más de 9.000.

Una columna clave es 'season\_year', que permite ordenar cronológicamente los partidos e implementar validaciones cruzadas temporales (ej. entrenar con temporadas anteriores y testear con una temporada futura). Además, se agregaron columnas específicas para distinguir métricas del equipo local y visitante (por ejemplo, Shots\_on\_Goal\_Home vs Shots\_on\_Goal\_Away).

Otras columnas disponibles incluyen estadísticas como:

* Posesión de balón (local y visitante)
* Goles esperados (xG) (local y visitante)
* Tiros a puerta (local y visitante)
* Faltas cometidas (local y visitante)
* Tarjetas amarillas y rojas (local y visitante)
* Offsides (local y visitante)
* Pases totales (local y visitante)
* Atajadas del portero y save ratio (local y visitante)

Durante el análisis exploratorio, se estudiaron distribuciones, tendencias y relaciones relevantes. Algunos de los hallazgos clave:

* **Ventaja local**: Se confirmó que los equipos locales ganan aproximadamente el 45-50% de los partidos, mientras que los empates representan entre un 20-25%, lo que genera un pequeño desbalance, pero realista y objetivo, en la variable a predecir.
* **Distribución de xG**: Se observó que los goles esperados (xG) tienen una fuerte correlación con el resultado, siendo generalmente más altos en los equipos ganadores.
* **Tiros a puerta** y **paradas del portero** mostraron ser variables predictoras útiles al correlacionarse tanto con el rendimiento ofensivo como defensivo.
* Se detectaron valores atípicos (outliers) en partidos con resultados atípicos (como 7-0 o 0-6), pero no se eliminaron dado que representan casos reales relevantes para el aprendizaje del modelo.
* Se estudiaron también relaciones entre tarjetas, faltas y posesión, sin encontrar patrones claros que predijeran directamente el resultado.

Como parte del EDA avanzado, también se realizaron visualizaciones específicas como histogramas, gráficos de densidad y mapas de calor para examinar correlaciones y evaluar si las variables eran redundantes o informativas.

III. Preprocesamiento y Feature Engineering

Uno de los principales desafíos fue construir un dataset realista. Se optó por eliminar completamente la última temporada registrada (2024-2025) ya que estaba incompleta y no tenía sentido, dado que la temporada a predecir era la (2023-2024). Las temporadas anteriores fueron usadas para calcular estadísticas agregadas por equipo.

Antes de iniciar la construcción del modelo, se realizó una verificación exhaustiva de la calidad del dataset. Entre las acciones llevadas a cabo se incluyen:

* **Eliminación de partidos de la temporada 2024/2025**, dado que aún no había datos completos al momento del análisis y no tenía sentido, dado que la temporada a predecir era la (2023-2024).
* Se validaron tipos de datos, formatos de fecha y columnas faltantes. En particular, se unificaron fechas en el formato Date\_day, construyendo una columna temporal fiable para ordenar cronológicamente los partidos.
* Se detectaron columnas con valores nulos y se rellenaron con medias especificas para cada equipo y su localia en cada caso.

**Decisiones, imputaciones y transformación de variables**

Durante el preprocesamiento, se tomaron una serie de decisiones clave que impactan directamente en la capacidad predictiva del modelo:

* **Cálculo de estadísticas acumuladas (feature engineering)**:  
  Se implementaron métricas por equipo local y visitante usando medias históricas y formas móviles:
  + save\_ratio, expected\_goals\_xg, Shots\_on\_Goal, Goalkeeper\_Saves, etc.
  + Cada una se calculó separadamente para el equipo local y visitante, sumando contexto táctico individual.
* **Método de cálculo de estadísticas.**

Se aplico la mediana en vez de la media, dado que hay cierto partidos y equipos que pueden desvirtuar la realidad de algunas métricas.

* + Luego se agrupo por equipo y por localia, y si ese equipo tenía algún dato faltante o algún “0” que no fuera lógico (por ejemplo, posesión de balón) se hace la mediana de ese mismo equipo jugando en esa misma localia.
  + En caso de no haber datos o ser todos 0, se hace la media de ese equipo jugando tanto de local como de visitante, para que al menos mantenga la esencia del equipo, aunque pueda variar la mediana local a la mediana general de un equipo.
  + En muy pocos caso en los que pudiera no haber nada de eso, se buscaba rellenar con la mediana según la localia en esa métrica, todos los locales por un lado y los visitantes por otro, sin importar el equipo.
  + Y como ultimo si no era posible eso tampoco, se buscaba rellenar con una media general de toda la métrica sin importar la localia ni el equipo.

Todo esto para buscar un dataset mas robusto y en el que cada equipo tuviera una métrica “razonable” dentro de lo que cabía.

* **Corrección de datos del propio dataset**.

Se encontraron ciertos valores muy extraños, que se comprobaron de forma manual, viendo que algunos eran ciertos y otros estaba muy equivocados.

Se corrigió los que estaban equivocados por los valores reales encontrados.

* **Separación entrenamiento / test**: se dividió el dataset respetando completamente el eje temporal. Se entrenó con:
  + Todas las temporadas anteriores a la temporada 2024/2024
  + Primera mitad de la temporada 2023/2024
  + Y se testea con la **segunda mitad de la temporada 2023/2024**.

Este enfoque garantiza que no se “filtren” datos del futuro al pasado, evitando fugas de información que sesgarían la evaluación del modelo.

IV. Modelado

Los modelos entrenados fueron:

* **Random Forest**: una primera aproximación robusta, especialmente útil por su facilidad de interpretación y resistencia al overfitting en configuraciones bien ajustadas.
* **XGBoost**: modelo de boosting altamente eficiente, que permite mayor control del sesgo y la varianza. Se utilizó tanto con parámetros por defecto como con varias configuraciones específicas (más restrictivas y más flexibles).
* **LightGBM**: modelo de boosting alternativo más rápido, con buena tolerancia al ruido y grandes volúmenes de datos.
* **StackingClassifier**: combinación de los tres modelos anteriores mediante un meta-modelo final (Regresión Logística), para intentar capturar la fuerza de cada uno de forma complementaria.

**Evaluación de los diferentes modelos e iteraciones**

Durante el entrenamiento, se utilizaron dos estrategias de validación:

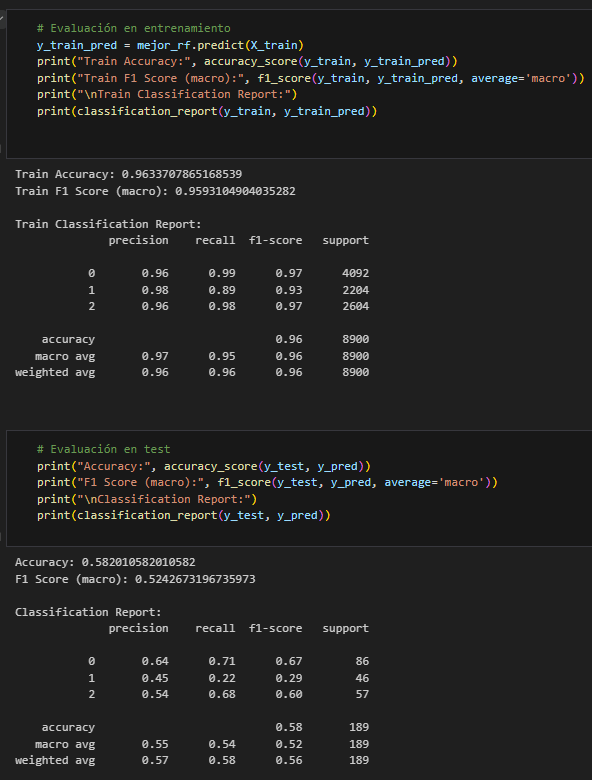
* **Cross-validation clásica (k=5) y (k=10)** sobre el conjunto completo de entrenamiento (X\_train), para evaluar consistencia general del modelo.
* **Validación cruzada temporal (GroupKFold)**, dividiendo por temporada (season\_year) para evitar filtración de información futura y testear la estabilidad año a año del modelo.

Los modelos fueron entrenados con escalado previo de features, validación cruzada y ajuste de hiperparámetros. La métrica principal fue el Accuracy y otras métricas que se miraron fueron el F1 Score macro y el ROC AUC, por el desbalance inherente entre las clases (la mayoría de partidos terminan en victoria local).

Además, se aplicaron visualizaciones como matrices de confusión, reportes de clasificación por clase, y gráficos de importancia de variables.

**Implementación de modelos.**

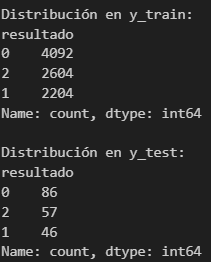
Primero probamos un modelo de Random Forest básico que nos dio un Accuracy en test de 0,582. Lo cual pensé que era una buena base de partida.

Hasta que pensé en testear también con el entrenamiento. Ahí empezaron los problemas porque me di cuenta que tenia un Overfitting enorme.

Los resultado con otros hiperparametros y en otros modelos incluso fueron los siguientes:

Para combatir el Overfitting se probaron las siguientes cosas:

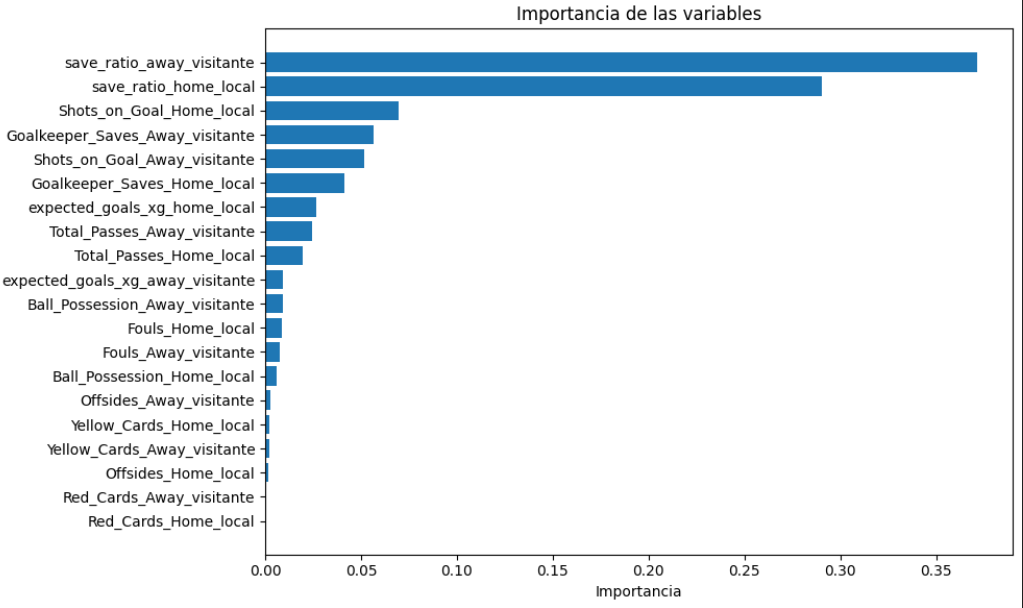
Se hizo un conteo del target tanto en Train como en Test para ver si había un desbalance que justificara el overfitting.

Por lo que vemos, parece que si puede haber un pequeño desbalanceo, pero algo totalmente lógico y normal en el mundo del futbol.

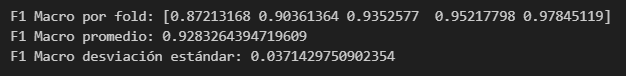
Como pudimos ver en mi EDA el equipo local tiende a ganar y tener ventaja por jugar en casa.

Por lo cual no me pareció un desbalanceo para preocuparme.

Una vez vemos esto, decidí mirar el feature importance, para ver que estaba tomando el modelo como importante y me encontré con esto:

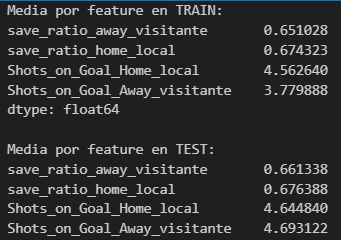


Esto me sorprendió muchísimo porque el modelo interpreta que las 3 primeras variables (sin importar su localia) son las mas importantes, pero no tiene en cuenta otras métricas que en la realidad son muy valoradas como pueden ser los XG o Excpected Goals (que miden la calidad de una ocasión de gol sin importar si el disparo va a puerta o fuera o es gol, mide otras variables y saca una cuantía de cuan peligroso fue esa ocasión). Otra métrica que no tiene en cuenta son por ejemplo las rojas, que es una situación que se da muy muy poco en el futbol, pero es sumamente condicionante.

Después realizamos un Cross\_val\_score sobre todo X para ver si “MIRAR EN CHAT GPT PARA QUE HACIAMOS ESTA FUNCION”

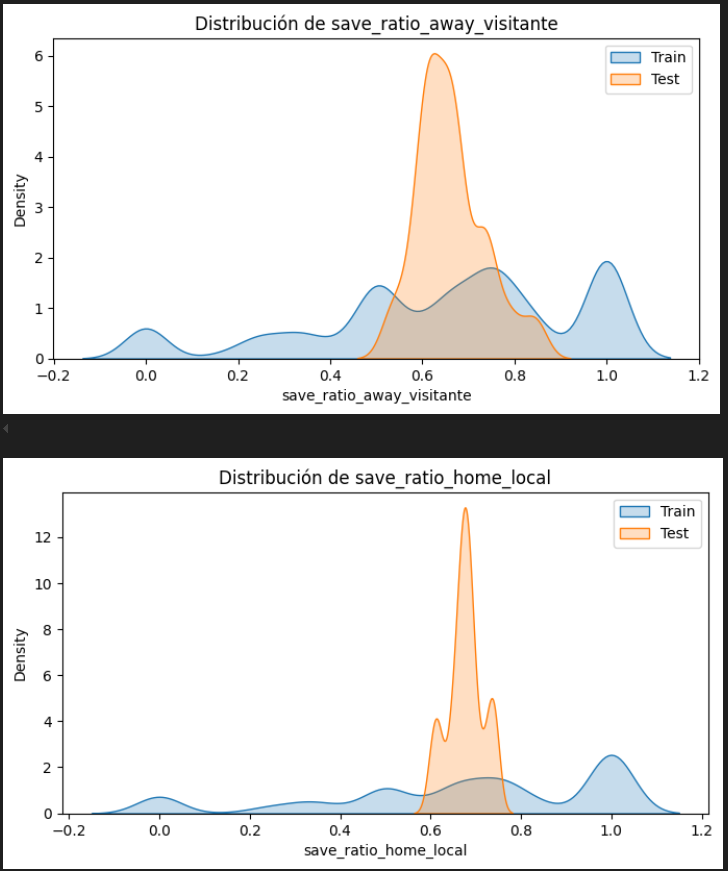
Vemos que todos los F1 Macro son similares, que el promedio es bueno y que no hay apenas desviación estándar, otro razonamiento de que el dataset no está desbalanceado.

También se nos ocurrió ver la media en train y test de las 4 features más importantes para el dataset, dependiendo su localia. Para ver si había mucha discrepancia entre lo que había en train y test y a ver si podía ser eso lo que causara el overfitting.

Vemos que las medias son muy similares tanto en train como en test.

Puede que haya una pequeña variación en la última métrica entre test y train, pero no es nada llamativo.

Por lo cual vamos a ver gráficamexnte la relación entre estas métricas de train y test



Aquí vemos algo muy interesante, y es que las 2 métricas mas importantes para el modelo tienen bastante discrepancia. Vemos que en ambas el train tiene muchos picos, subidas y bajadas y una densidad muy baja, mientras que, en test, tenemos algo parecido a una Campana de Gauss, con una densidad muy alta.

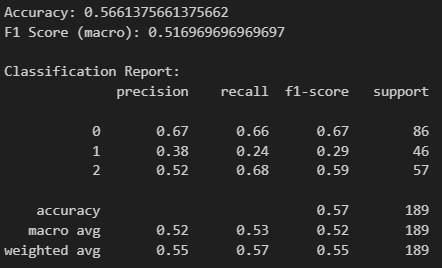
Esto se puede deber a que el Train tiene mucha cantidad de datos y muy disperso, lo cual hace que se genere esa forma alargada pero bajita.

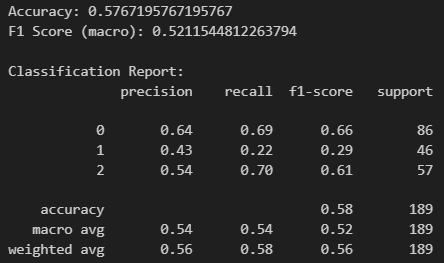
Y el Test al ser una media por equipo de la primera mitad de la temporada, se termina agrupando muchos datos en un rango muy pequeño, por lo cual sube la densidad y termina siendo una figura corta pero muy alta

Esto puede afectar a ese overfitting dado que son las métricas mas importantes para el y esta entrenando con unos datos y en el test tiene otros muy diferentes. Esto nos abre una línea de investigación.

Para seguir con nuestro intento de reducir el Overfitting, procedemos a eliminar algunas features y entrenar los modelos de nuevo.

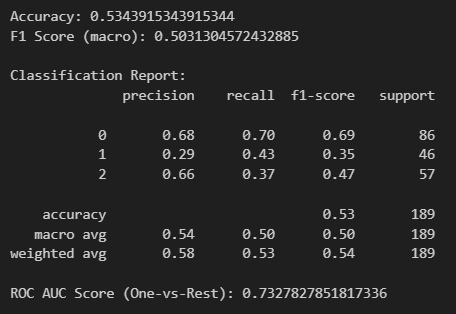
* Features importantes -> Primero decidimos eliminar las 4 features más importantes, para ver si podía aprender del resto de features y eso nos llevo a bajar mucho el Accuracy en Train y mantenerlo en Test. Pero me daba un Recall de 0.02 en el caso de los empates.
* Features insignificantes -> También eliminamos las 6 features menos importantes y conseguimos algo mas decente y con un mejor Recall en los empates.



Luego pensé en probar escalando las features, para reducir las diferencia que pueden generar el total de pases (ej: 650) con los disparos a puerta (ej: 3) con los xgoals (ej: 1.14). De esta forma mejoraron un poquitito las cosas.

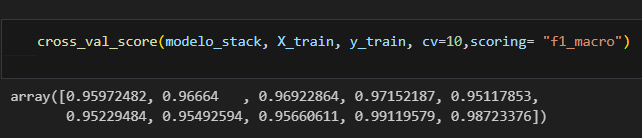
Una de las ultimas cosas que probamos fue el “Stacking de modelos”.

¿Qué es? -> Es una técnica de ensamblado en la que se entrenan múltiples modelos base sobre los mismos datos. Luego, sus predicciones se utilizan como nuevas features para entrenar un modelo adicional, en este caso, una regresión logística. El meta-modelo no ve los datos originales, sino únicamente las salidas de los modelos base, aprendiendo a combinar sus errores y aciertos para mejorar la predicción final.



Sus resultados no son los mejores en Accuracy, pero vemos que mejora el f1-score de los empates, a costa de las victorias visitantes.

Y por último se hizo un train test Split, para ver que tal predecía el modelo sobre los datos mismos. Se cogió todo el train, se dividió en 80/20 y se entreno y se vio que el modelo predice bien en general. Para tener mas seguridad de esto, se realizo un Cross\_Val\_Score con el Stacking, sobre el train y los resultados fueron similares.



Con esto podemos ver que el modelo predice muy bien con los datos de train, pero el problema viene con los datos que tenemos en test. Ya que como vimos arriba esa variaciones en la feature mas importante hace que el modelo no entienda que esta pasando, al entrenar con unos ciertos datos y de pronto tener que predecir con otros bastante distintos. Por ende, concluyo con que el modelo funciona, pero en esta ocasión los datos para el test no le ayudan.

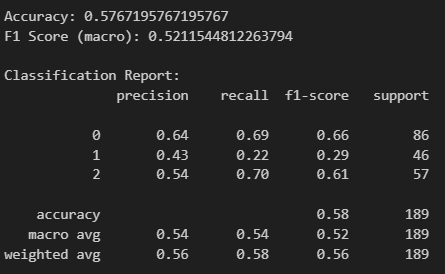
**Selección e interpretación del modelo final**

Durante las pruebas, se identificó un patrón repetido: los modelos lograban un rendimiento aceptable en la predicción de victorias (clase 0 y 2), pero tenían **grandes dificultades con los empates (clase 1)**.

El modelo final seleccionado fue Random Forest con el escalado de las features.

V. Evaluación Final

Se evaluó el modelo con matriz de confusión, curva ROC multicategoría, Acuraccy, F1 Score y análisis de importancia de variables.



Vemos que tenemos un f1 score decente en las victorias, pero los empates se nos escapan mucho.

VI. Conclusiones

Este proyecto demuestra una cierta viabilidad en construir un sistema predictivo de resultados futbolísticos usando únicamente información disponible previo al partido.

Los resultados son razonablemente buenos dado el nivel de aleatoriedad del fútbol. El enfoque puede ampliarse a múltiples ligas o a otras competiciones, integrando forma reciente del equipo (rolling average), valoraciones de plantillas, y embeddings contextuales.

VII. Futuros pasos

* Agregar features con promedios de los últimos 3-5 partidos.
* Explorar métricas relativas entre equipos (e.g. diferencia de xG histórico).
* Usar SMOTE o estrategias de balanceo avanzadas.
* Probar modelos de secuencia (LSTM, Transformers).
* Crear un sistema de scoring probabilístico calibrado para apuestas o simuladores.

Este proyecto sentó las bases de una arquitectura realista, escalable y analíticamente válida para la predicción de resultados futbolísticos con aprendizaje automático.