2025

Análisis predictivo de partidas en League of Legends

Comisión 67395- Data Science II

Docente: Gustavo Benitez

Tutor: Carlos Davoli

Alumno: Esteban Goyeneche

Alcance y Nivel de Usuario

League of Legends, es uno de los juegos más populares y competitivos del mundo, con millones de jugadores y un ecosistema de deportes electrónicos altamente profesionalizado. A medida que los jugadores ascienden en el sistema de clasificación, los factores que determinan el éxito en una partida se vuelven más complejos y menos evidentes.

Este proyecto busca analizar más de 9,000 partidas de jugadores en elo Diamante alto a Maestro bajo, un rango donde la toma de decisiones, la macroestrategia y la coordinación de equipo tienen un peso significativo.

Audiencia

- Jugadores competitivos que desean optimizar su toma de decisiones.
- Entrenadores o analistas de esports.
- Científicos de datos interesados en aplicar analítica predictiva a juegos multijugador.
- Cualquier persona interesada en entender cómo los datos pueden explicar el rendimiento en videojuegos complejos.

Contexto Analítico

El dataset incluye métricas detalladas por partida de ambos equipos (rojo y azul), como:

- Estadísticas de visión (wards)
- KDA (kills, deaths, assists)
- Objetivos tomados (dragones, heraldos, torres)
- Diferencias de oro y experiencia
- Ritmo de farmeo y oro por minuto
- La información está agrupada a nivel de equipo, no por jugador individual.
- Esto nos permite construir modelos que evalúan el rendimiento colectivo y predecir el resultado de una partida basado en estas métricas objetivas.

nota: en secciones posteriores habrá un diccionario básico que contiene la descripcióon de cada variable

Objetivos

Objetivo General
Analizar partidas en el elo Diamante–Maestro para identificar los factores clave que determinan la victoria y construir un modelo predictivo basado en datos objetivos de equipo.

Objetivos Específicos
- Realizar un análisis exploratorio comparativo entre equipos ganadores y perdedores.
- Identificar las métricas más correlacionadas con el resultado.
- Diseñar un modelo de clasificación que prediga la victoria del equipo azul o rojo.
- Evaluar la importancia relativa de cada variable para el resultado.

Hipótesis y preguntas clave

Hipótesis a probar:

- Hipótesis 1 (Análisis Exploratorio):"Existen ciertos indicadores clave que aumentan significativamente la probabilidad de victoria, como el control de objetivos neutrales (dragones/heraldos) y la visión (wards)."
- Hipótesis 2 (Modelado Predictivo):"Es posible predecir el equipo ganador con buena precisión usando solo los datos objetivos de rendimiento del equipo a lo largo de los 10 primero minutos de la partida."

El planteamiento de dos hipótesis está definido de tal modo que una se aplique al análisis exploratorio, análisis univariado, bivariado y demás del dataset, y que su respuesta sea el insumo para el modelado de la hipótesis 2.

Preguntas que guían el análisis Exploración:

- ¿Qué diferencias estadísticas existen entre equipos que ganan y pierden?
- ¿Cuáles son los factores más asociados con una victoria?
- ¿Controlar dragones/heraldos/towers tiene más impacto que kills?
- ¿Hay diferencias sistemáticas entre los equipos rojo y azul?

Predicción:

- ¿Podemos construir un modelo que prediga el resultado de una partida con base en estadísticas finales?
- ¿Cuáles son las variables más importantes para el modelo?
- ¿Qué tan precisa puede ser una predicción sin información sobre campeones o picks?

Posibilidad de mejora en la información

- Se puede conectar la API de Riot games para obtener más información, por ejemplo de partidas en elo mas bajo, pára así realizar un análisis comparativo de lo que diferencia un jugador clasificado más alto que uno mas bajo.
- Para efectos académicos el dataset está muy "limpio", cosa de la que me percaté avanzando en el trabajo, por lo que no fueron necesarias aplicar muchas técnicas de data wranglin o limpieza de datos e identificación y reemplazo o eliminación de valores nulos.

Detalles del dataset

El dataset presenta casi 10 mil filas, contra 40 columnas, divididas casi en su totalidad en dos partes, las variables del equipo azul y las del equipo rojo.

- gameId: ID único de RIOT para la partida, puede usarse con la API de RIOT.
- blueWins: La columna objetivo. 1 Si el equipo azul ha ganado, 0 de lo contrario.
- blueWardsPlaced: Número de tótems de visión (wards) colocados por el equipo azul en el mapa.
- blueWardsDestroyed: Número de wards enemigos que el equipo azul ha destruido.
- blueFirstBlood: Primera sangre del juego. 1 Si el equipo azul hizo el primer asesinato, 0 de lo contrario.
- blueKills: Número de enemigos asesinados por el equipo azul.
- blueDeaths: Número de muertes del equipo azul.
- blueAssists: Número de asistencias en muertes.
- blueEliteMonsters: Número de monstruos de élite asesinados por el equipo azul (dragones y heraldos), oscila entre 0 y 2, ya que se compone del indicador de dragón y heraldo asesinado.
- blueDragons: Indicador de primer dragón tomado por el equipo azul.
- blueHeralds: Indicador de primer heraldo tomado por el equipo azul.
- blueTowersDestroyed: Número de torres destruidas por el equipo azul.
- blueTotalGold: Total de oro del equipo azul.
- blueAvgLevel: Nivel de campeón promedio del equipo azul.
- blueTotalExperience: Experiencia total del equipo azul.
- blueTotalMinionsKilled: Total de minions asesinados por el equipo azul.
- blueTotalJungleMinionsKilled: Total de monstruos de jungla muertos por el equipo azul.
- blueGoldDiff: Diferencia de oro del equipo azul en comparación con el equipo enemigo.
- blueExperienceDiff: Diferencia de experiencia del equipo azul en comparación con el equipo enemigo.
- blueCSPerMin: Creep Score por minuto del equipo azul (monstruos de junga y minions de línea asesinados).
- blueGoldPerMin: Oro por minuto del equipo azul.
- redWardsPlaced: Número de tótems de visión (wards) colocados por el equipo rojo en el mapa.
- redWardsDestroyed: Número de wards enemigos que el equipo rojo ha destruido.
- redFirstBlood: Primera sangre del juego. 1 Si el equipo rojo hizo el primer asesinato, 0 de lo contrario.

Detalles del dataset

- redKills: Número de enemigos asesinados por el equipo rojo.
- redDeaths: Número de muertes del equipo rojo.
- redAssists: Número de asistencias en muertes.
- redEliteMonsters: Número de monstruos de élite asesinados por el equipo rojo (dragones y heraldos), oscila entre 0 y 2, ya que se compone del indicador de dragón y heraldo asesinado.
- redDragons: Indicador de primer dragón tomado por el equipo rojo.
- redHeralds: Indicador de primer heraldo tomado por el equipo rojo.
- redTowersDestroyed: Número de torres destruidas por el equipo rojo.
- redTotalGold: Total de oro del equipo rojo.
- redAvgLevel: Nivel de campeón promedio del equipo rojo.
- redTotalExperience: Experiencia total del equipo rojo.
- redTotalMinionsKilled: Total de minions asesinados por el equipo rojo.
- redTotalJungleMinionsKilled: Total de monstruos de jungla muertos por el equipo rojo.
- redGoldDiff: Diferencia de oro del equipo rojo en comparación con el equipo enemigo.
- redExperienceDiff: Diferencia de experiencia del equipo rojo en comparación con el equipo enemigo.
- redCSPerMin: Creep Score por minuto del equipo rojo (monstruos de junga y minions de línea asesinados).
- redGoldPerMin: Oro por minuto del equipo rojo.

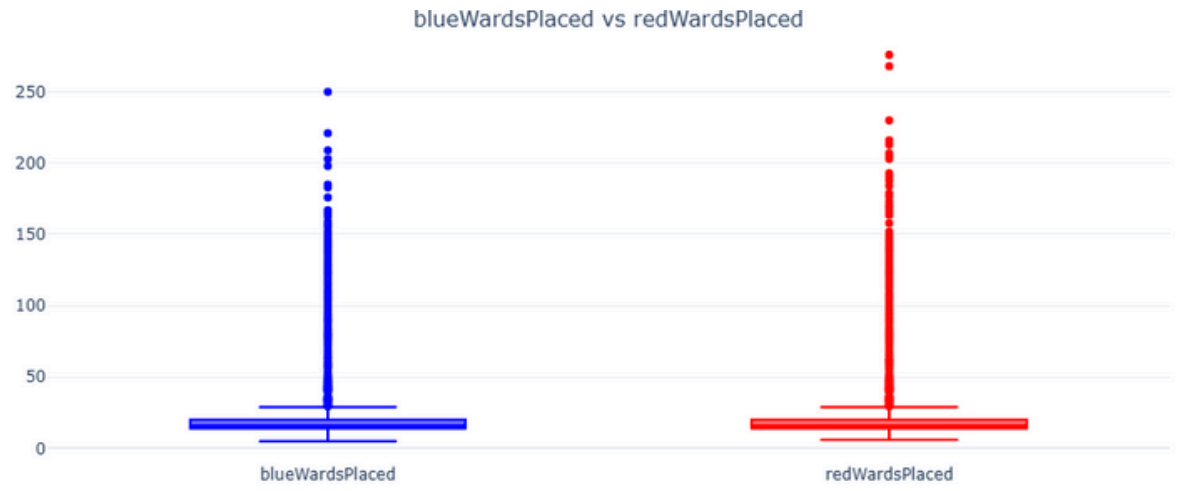
Valores nulos y Outliers

El dataset no presenta valores nulos, y los valores que vienen con "0" son válidos de acuerdo al contexto de cada una de las variables, por ejemplo, la variable blueDragons. A su vez, tampoco hay variables tipo object, ya que las variables categóricas del dataset ya se encuentran codificadas, por ejemplo la variable objetivo blueWins.

No hay valores duplicados en el dataset por la naturaleza del mismo, puede que hayan estadisticas iguales en dos partidas, pero al ser partidas con identificadores diferentes no se cuentan como valores duplicados, de hecho es practicamente imposible encontrar dos partidas con estadisticas diferentes

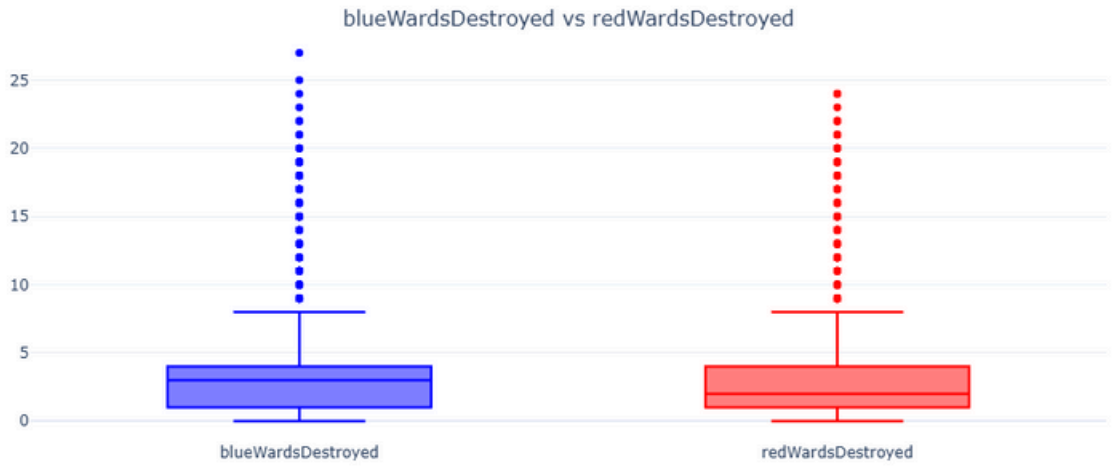
En cuanto a la estadistica básica, en general tiene valores normales para las las variables numéricas, a excepción de algunos valores outliers que pueden llegar a afectar la media de algunas variables, por ejemplo:

- blueWardsPlaced: Tiene una media de 22,28 para los 10 primeros minutos de partida, debido a valores outliers que rondan los 250
- redWardsPlaced: Tiene una media de 22,36 para los 10 primeros minutos de partida, debido a valores outliers que rondan los 250



- blueWardsDestroyed: Tiene una media de 2,82 para los 10 primeros minutos de partida, con outliers cercanos a los 27
- redWardsDestroyed: Tiene una media de 2,73 para los 10 primeros minutos de partida, con outliers cercanos a los 24

Valores nulos y Outliers



Los demás valores outliers, relacionados a variables como muertes, asesinatos y asistencias, no son tan pronunciados, y además pueden tener una explicación relacionada al desarrollo de la partida, por ejemplo, se presenten casos de snowball en la partida, donde un equipo arrasa al otro desde el inicio de la partida, lo cual genera diferencias abismales a su vez entre oro y experiencia ganada.

Con esto en cuenta, y dado que la informacion de los boxplot es congruente en su mayoría con los rangos suministrados, se identifican múltiples outliers en varibales como la cantidad de wards colocados, donde hay valores como 250, lo cual es practicamente imposible, ya que implica que cada jugador tuvo que haber puesto 5 wards por minuto, de la mano con la cantidad de wards destruidos que también presenta outliers.

Tratamiento de Outliers
Se usa el IQR(Rango intercuartílico) para eliminar los registros con los outliers más evidentes e incongruentes de acuerdo al desarrollo normal de una partida. Se toma el valor 3 en el IQR para que no sea tan ácido, la idea es eliminar las variables menos lógicas (por ejemplo las de wards), pero sin reducir demasiado el dataset con otras variables que son más posibles (como la cantidad de minions asesinados).

```
IQR = Q3 - Q1
lower_bound = Q1 - 3 * IQR
upper_bound = Q3 + 3 * IQR
```

Valores nulos y Outliers

Posterior a la depuración de valores outliers tenemos las siguientes cifras en las variables con valores outliers más críticos:

- blueWardsPlaced: Tiene una media de 17,25, con valores máximos de la variable de 38
- redWardsPlaced: Tiene una media de 17,24, con valores máximos de la variable de 38
- blueWardsDestroyed: Tiene una media de 2,63 con valores máximos de la variable de 13
- redWardsDestroyed: Tiene una media de 2,54 con valores máximos de la variable de 13

Como se evidencia, se depuraron los valores outliers, y se presentan valores mas "normales" de acuerdo al contexto de una partida de league of legends, en total, de 9879 registros, tras la depuración vamos a trabajar con 7720, si hubieramos usado un valor mas ácido en el IQR, probablemente la cifra rondaria menos de los 6500.

Creación de nuevas variables

Con miras a enriquecer el análisis y el modelado, se crean dos variables nuevas, que están compuestas a partir de variables ya existentes, estas variables son:

- KDA: Es una métrica que a nivel individual nos permite evaluar el desempeño de un jugador en la partida y su contribución al equipo, se calcula a partir de los asesinatos, asistencias y muertes. Sin embargo en este caso, las variables mencionadas anteriormente están a nivel de equipo y no de jugador, esto nos permitirá tener a grandes rasgos un vistazo general del control del mapa, ventaja en peleas, etc. La formula de cálculo del KDA es:

$$KDA = Kills + Assists / Deaths$$

- VisionScore: Esta variable nos puede ayudar a evaluar de forma más general el control de la visión en el mapa, ya que se compone de los wards colocados y los wards destruidos, lo cual nos permite tener mayor control del mapa y de los objetivos. Este se calcula de la siguiente forma:

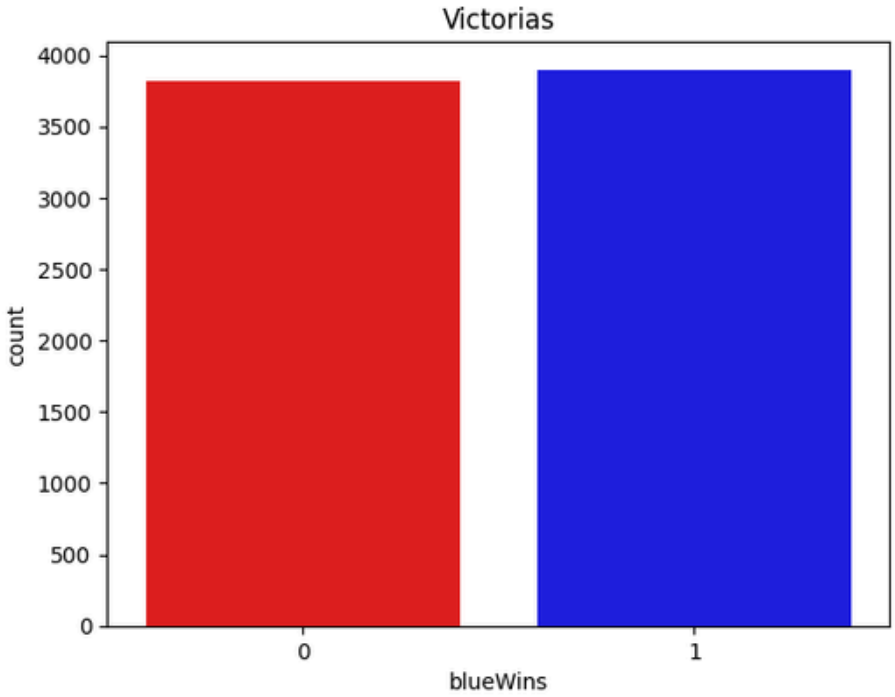
$$VisionScore = WardsPlaced + WardsDestroyed$$

Análisis exploratorio de datos EDA

Análisis Univariado

- **Distribución de la variable objetivo**

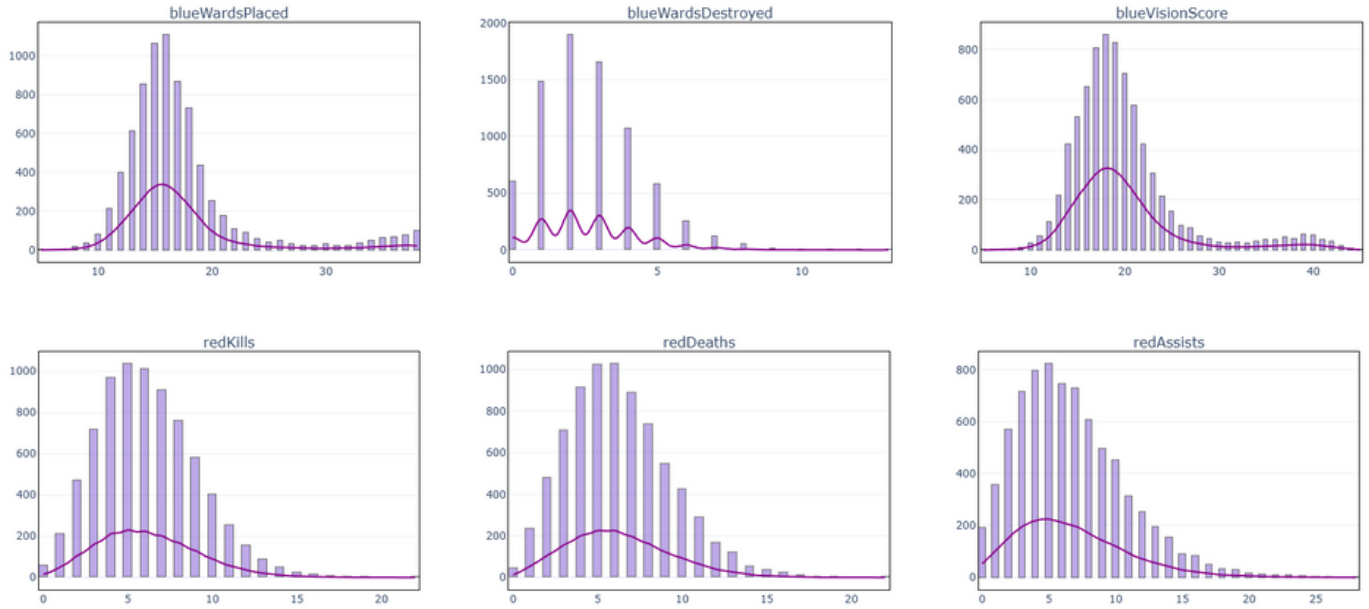
Lo que vamos a revisar primero, es validar la distribución de nuestra variable objetivo, con miras a identificar posibles desbalances en la variable o sesgos de algún tipo, como se evidencia a continuación el el gráfico, la variable está practicamente equilibrada en su distribución, lo cual nos ayuda a tener un margen mas preciso por ejemplo a la hora de evaluar el Accuracy o F1 Score



- **Distribución de variables numéricas**

Las distribuciones de las variables numéricas muestran patrones bien definidos y, en su mayoría, presentan formas unimodales con ligeras asimetrías. Algunas variables como blueKills, redKills y blueAssists tienen una distribución sesgada a la derecha, lo que indica que la mayoría de los valores se concentran en rangos bajos, pero existen casos menos frecuentes con valores mucho mayores (posibles "partidas destacadas"). Otras, como blueWardsPlaced y redWardsDestroyed, presentan sesgo a la izquierda, sugiriendo que los valores altos son más comunes, mientras que los valores bajos son menos frecuentes. La forma no plana y bien definida de las distribuciones sugiere que estas variables siguen comportamientos consistentes, lo que puede ser útil para predicción o modelado. A continuación se presentan algunos ejemplos gráficos:

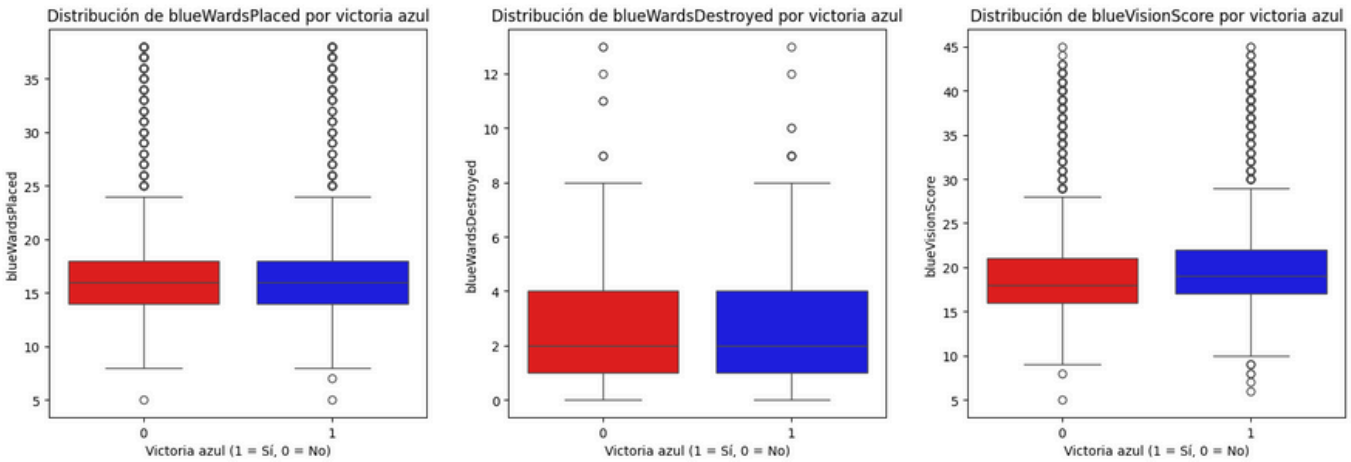
Análisis exploratorio de datos EDA



Análisis Bivariado

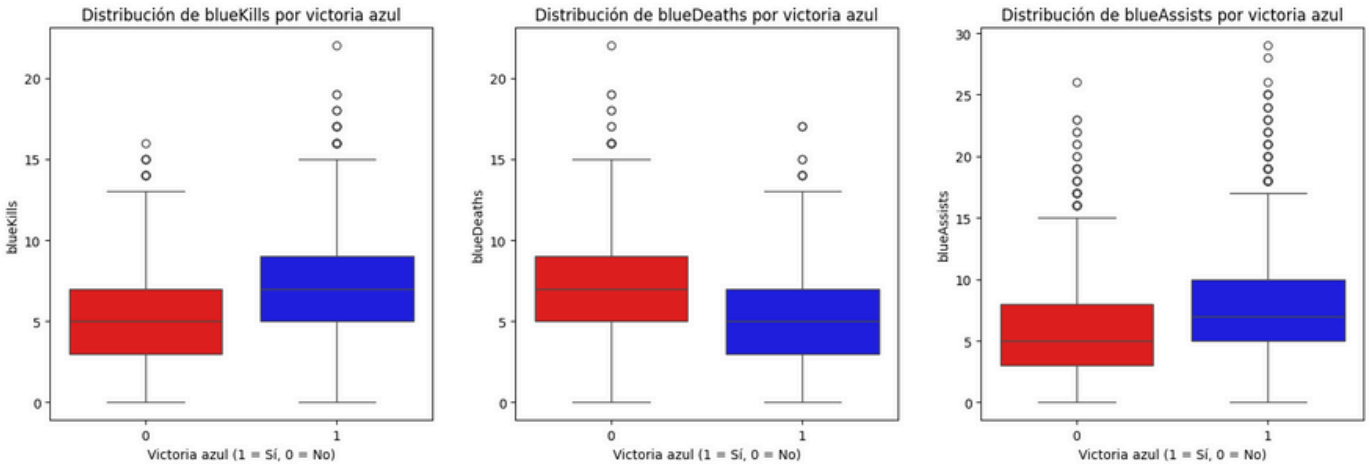
Dentro del análisis bivariado tomamos como base la variable objetivo, contra otra variable, de tal forma que podamos tener un indicador gráfico de la influencia de dicha variable respecto al resultado final de victoria del equipo rojo o azul. Del análisis realizado podemos extraer los siguientes insights principales:

- Aunque los wards colocados y destruidos por separado no tienen una gran implicancia en las victorias, la nueva variable que creamos anteriormente, VisionScore, sí proporciona una mejora significativa en las probabilidades de victoria, tanto para el equipo azul como para el rojo.

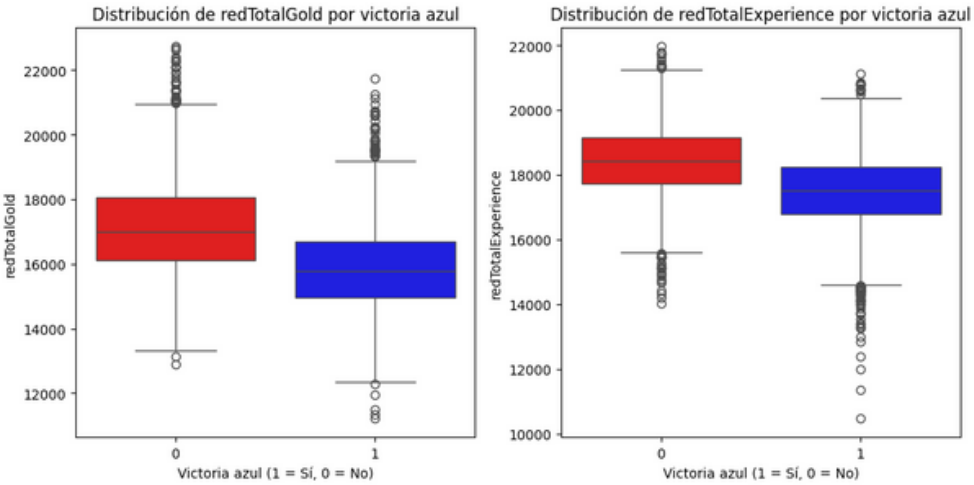


Análisis exploratorio de datos EDA

- A mayor cantidad de asesinatos y asistencias, hay más victorias finales, dependiendo del equipo que las haya conseguido. Esto está relacionado con la variable KDA que creamos previamente, la cual refleja una mayor presencia de victorias del equipo azul a medida que aumenta su KDA. Por el contrario, cuando el equipo azul pierde, se presenta una mayor cantidad de muertes dentro del equipo.

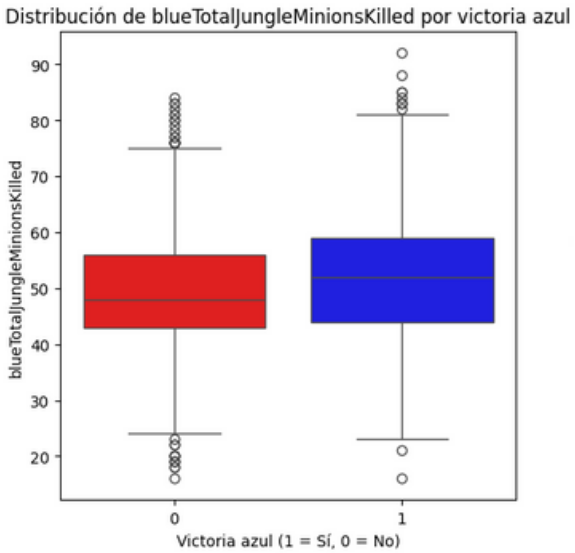


- La experiencia y el oro ganados tienen una de las principales influencias en la victoria del equipo. Por razones evidentes, generar más experiencia en los inicios de la partida puede desencadenar un efecto snowball, donde el equipo supera progresivamente al rival. Lo mismo ocurre con el oro: alcanzar objetos clave en menor tiempo otorga una ventaja sustancial.

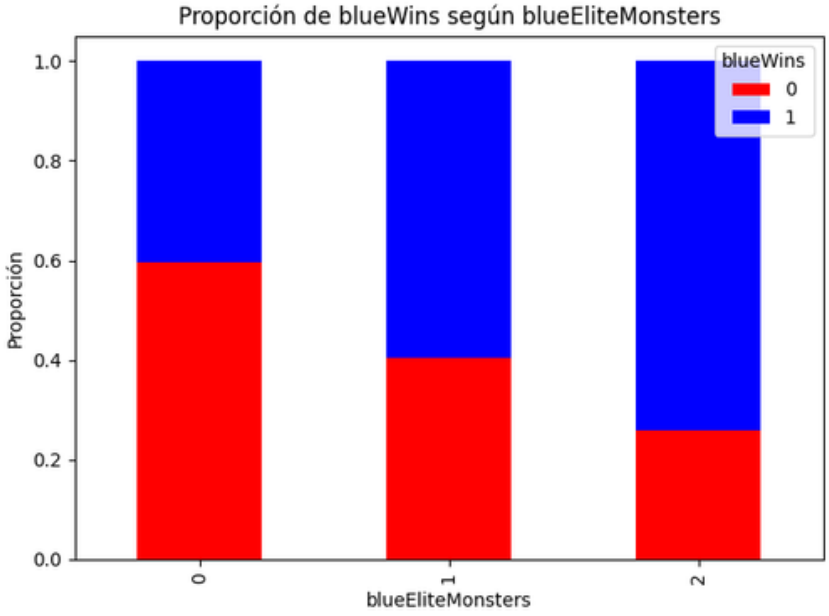


Análisis exploratorio de datos EDA

- La cantidad de minions de jungla asesinados tiene un gran impacto en los niveles iniciales, ya que permite al jungla realizar ataques a líneas vulnerables de forma más rápida y efectiva, lo que puede derivar en primeras sangres y marcar una diferencia temprana en oro y experiencia.



- En cuanto a la obtención de los primeros objetivos importantes (dragón y heraldo), estos pueden tener cierto impacto en la cantidad de victorias del equipo que los consigue. Sin embargo, observamos que la influencia de tomar el dragón es significativamente mayor que la de tomar el heraldo de la grieta. En general, los equipos que logran asegurar ambos objetivos presentan una clara ventaja y mayor cantidad de victorias sobre su rival. No obstante, no es común que un equipo tome ambos durante los primeros 10 minutos de partida. Por el contrario, no tomar ningún objetivo puede tener un impacto negativo significativo en la posibilidad de alcanzar la victoria.



Análisis exploratorio de datos EDA

Análisis Multivariado

Se realizó un gráfico de pares con las siguientes variables clave del equipo azul: blueKills, blueAssists, blueDeathslueKDA y blueWins.

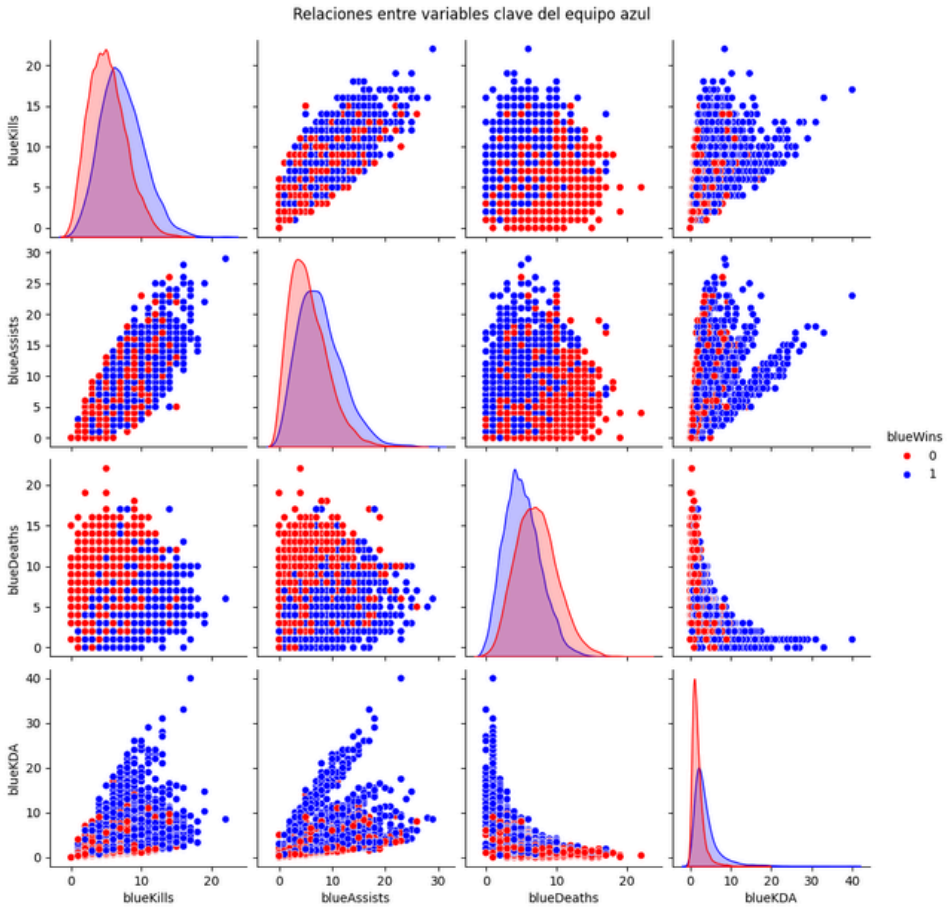
Las distribuciones muestran que los equipos azules que ganan suelen tener:

- Mayor cantidad de asesinato (blueKills) y asistencias (blueAssists).
- Menor cantidad de muertes (blueDeaths).
- Un KDA más alto (blueKDA), lo cual refleja un buen desempeño general del equipo.

Existe una correlación positiva fuerte entre asesinatos y asistencias: los equipos que matan más, también asisten más. A su vez, las combinaciones de altas kills y assists con pocas muertes son características predominantes en las victorias del equipo azul.

Finalmente, El KDA destaca como un buen indicador sintético del desempeño: valores más altos se asocian consistentemente con victorias.

Este análisis multivariado confirma la importancia de un desempeño coordinado y eficiente en combate (kills, assists, baja mortalidad) para lograr la victoria en el juego, tanto para el equipo azul como rojo.



Matriz de correlación

Al realizar un análisis de correlaciones entre variables podemos identificar variables altamente correlacionadas, que en realidad son opuestas entre si (correlación de -1), y podriamos depurarlas del dataset que usemos para el modelado y asi evitar redundancia de variables, algunos ejemplos son:

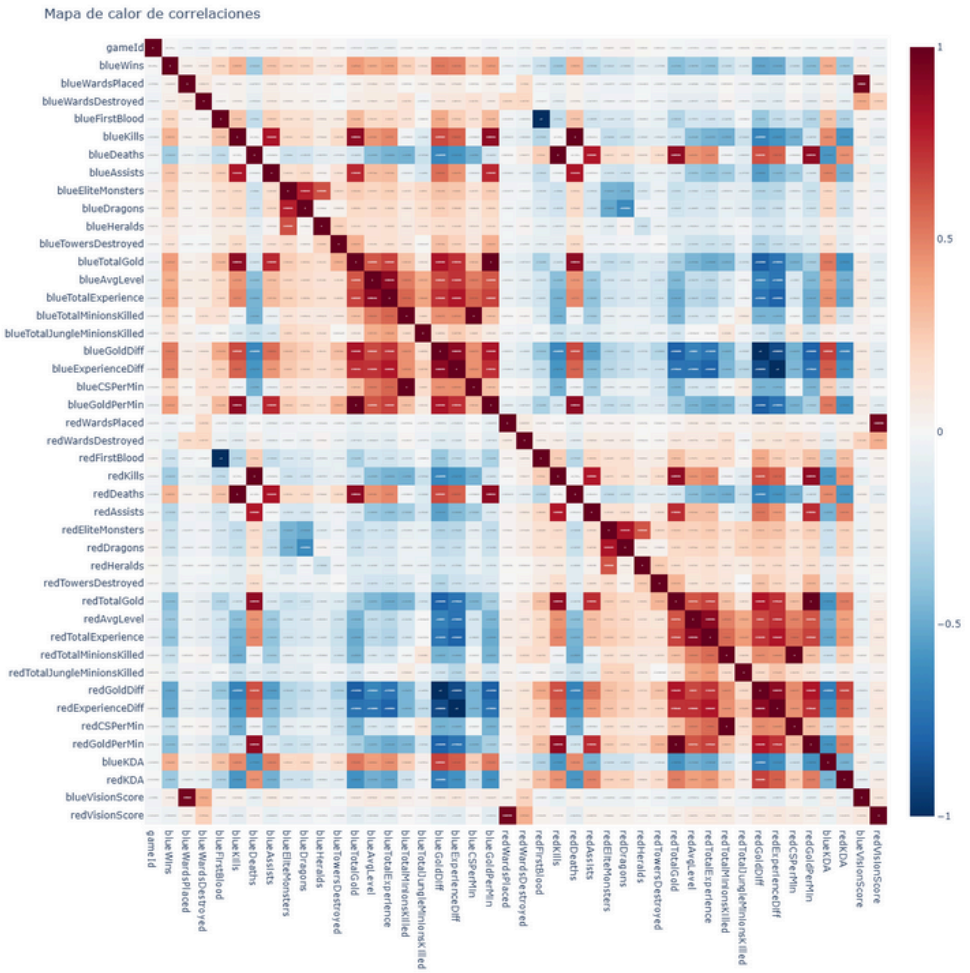
- blueFirstBlood vs redFirstBlood
- blueGoldDiff vs redGoldDiff
- blueExperienceDiff vs redExperienceDiff

Para ir acorde a nuestra variable objetivo blueWins, podemos antes de la etapa de modelado eliminar las variables redFirstBlood, redGoldDiff y redExperienceDiff.

A su vez se presentan variables con correlacion positiva perfecta (1), donde al subir una también la otra en la misma proporción, un ejemplo de esto son:

- blueKills vs redDeaths
- blueDeaths vs redKills

Al igual que en el caso anterior, antes de la etapa de modelado podemos depurar estas variables del equipo rojas y mantener unicamente las del equipo azul, o usar variables mas complejas como el KDA de cada equipo.



Matriz de correlación

Para resumir, las siguientes son las variables altamente correlacionadas:

- blueCSPerMin correlaciona fuertemente con: ['blueTotalMinionsKilled']
- blueGoldPerMin correlaciona fuertemente con: ['blueTotalGold']
- redFirstBlood correlaciona fuertemente con: ['blueFirstBlood']
- redKills correlaciona fuertemente con: ['blueDeaths']
- redDeaths correlaciona fuertemente con: ['blueKills']
- redGoldDiff correlaciona fuertemente con: ['blueGoldDiff']
- redExperienceDiff correlaciona fuertemente con: ['blueExperienceDiff']
- redCSPerMin correlaciona fuertemente con: ['redTotalMinionsKilled']
- redGoldPerMin correlaciona fuertemente con: ['redTotalGold']}

El tratamiento previo al paso de modelado fue depurar estas variables altamente correlacionadas, protegiendo un par de variables que a criterio personal me parecen que pueden aportar valor al modelado, que son blueVisionScore y redVisionScore

```
vars_protegidas = ['blueVisionScore', 'redVisionScore']

# Matriz de correlación
corr_matrix = df_clean.corr(numeric_only=True).abs()
upper_triangle = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))

# Variables con correlación > 0.95
high_corr_vars = [
    column for column in upper_triangle.columns
    if any(upper_triangle[column] > 0.95) and column not in vars_protegidas
]

# Revisamos con quién se correlacionan esas variables
for var in high_corr_vars:
    correladas = upper_triangle[upper_triangle[var] > 0.95].index.tolist()
    print(f"{var} correlaciona fuertemente con: {correladas}")

# Eliminamos solo las que no están protegidas
df_sin_corr = df_clean.drop(columns=high_corr_vars)

df_sin_corr
```

Modelado

Para esta etapa vamos a utilizar 3 modelos diferentes y así comparar su comportamiento de acuerdo al dataset a utilizar, explicando sus hiperparámetros basicos y su desempeño respecto de los demás. Los modelos a utilizar para clasificación son:

- Regresión Logística
- Random Forest
- XGBoost

Para los 3, las métricas de evaluación son las mismas, teniendo un análisis homogeneo del desempeño de cada uno. Las métricas son las siguientes:

- Accuracy: Proporción de predicciones correctas.
- Precision: Qué proporción de predicciones positivas son verdaderamente positivas.
- Recall: Qué proporción de verdaderos positivos fueron correctamente identificados.
- F1 Score: Media armónica entre precisión y recall.
- AUC-ROC: Capacidad del modelo para discriminar entre clases.

Se grafican además:

- La matriz de confusión, para visualizar los aciertos y errores del modelo.
- La curva ROC, que muestra la sensibilidad frente a la tasa de falsos positivos.

Modelo 1: Regresión Logística

La regresión logística es un modelo de clasificación lineal que estima la probabilidad de que una observación pertenezca a una clase, en este caso si el equipo azul gana (blueWins = 1).

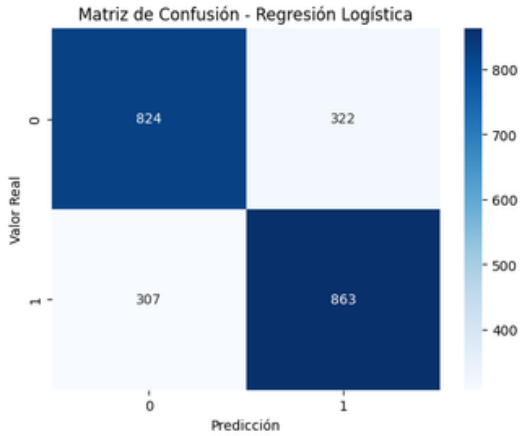
Hiperparámetros definidos:

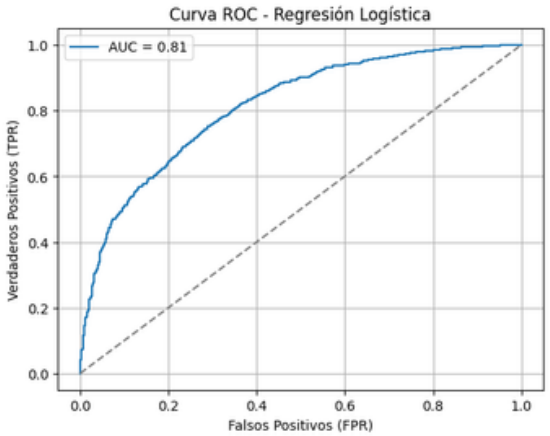
- C = 1.0: Controla la regularización del modelo. Un valor más alto reduce la penalización, un valor más bajo la aumenta. Usamos 1.0 como valor base.
- penalty = 'l2': Se aplica regularización L2 (Ridge), que penaliza grandes coeficientes y ayuda a evitar el sobreajuste.
- solver = 'liblinear': Algoritmo de optimización eficiente para conjuntos de datos pequeños y para problemas binarios como este.
- random_state=42: Se asegura reproducibilidad de resultados.

Modelado

Resultado

- Accuracy: 0.7284
- Precision: 0.7283
- Recall: 0.7376
- F1 Score: 0.7329
- AUC-ROC: 0.8139





Análisis Regresión Logística

- Accuracy: 0.7390 → El modelo acierta el 73.9% del total.
- Precision: 0.7347 → De las veces que predijo "victoria azul", acertó el 73.5%.
- Recall: 0.7564 → Detectó el 75.6% de las veces que ganó azul.
- F1 Score: 0.7454 → Balance adecuado entre precisión y recall.
- AUC-ROC: 0.8192 → Buen poder discriminativo entre clases.
- En cuanto a la matriz de confusión, los resultados que se muestran en el gráfico significan que el modelo acierta más del 70% en ambos casos, pero aún comete errores importantes, especialmente confundiendo 0 con 1.
- Para la curva ROC, el AUC (Área bajo la curva) = 0.81, nos indica que en términos generales el modelo tiene un poder predictivo consistente entre victorias y derrotas.

Modelo 2: Random Forest

Random Forest es un modelo de ensamble basado en la construcción de múltiples árboles de decisión que votan para generar la predicción final. Esta técnica mejora la generalización y reduce el sobreajuste comparado con un solo árbol.

Este modelo suele ofrecer una mejora frente a árboles individuales o regresión logística, especialmente si hay relaciones no lineales o interacciones entre variables.

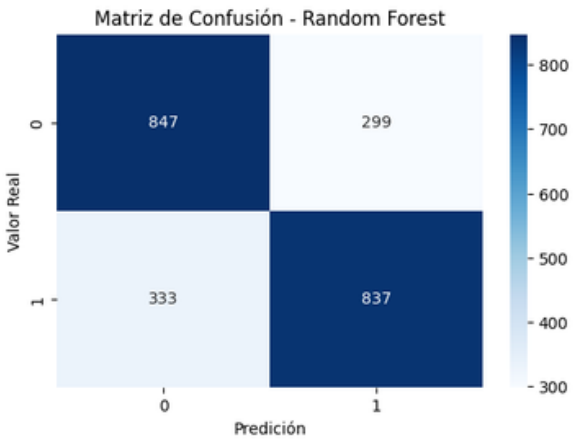
Modelado

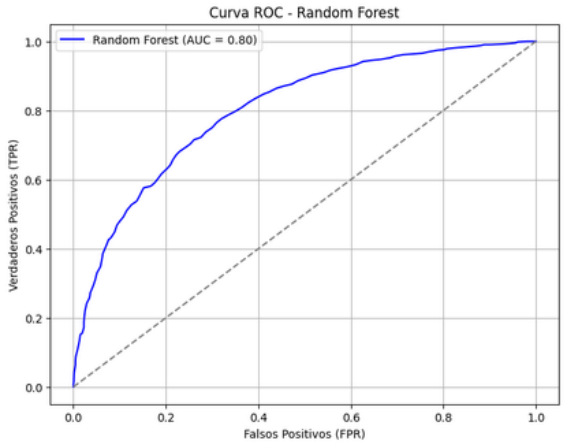
Hiperparámetros definidos:

- n_estimators=100: Se entrenan 100 árboles.
- max_depth=None: Cada árbol crece sin límite de profundidad.
- random_state=42: Se asegura reproducibilidad de resultados.
- min_samples_split=2: Define el número mínimo de muestras requeridas para dividir un nodo interno.
- min_samples_leaf=1: Define el número mínimo de muestras requeridas para estar en una hoja (nodo final).

Resultado

- Accuracy: 0.7271
- Precision: 0.7368
- Recall: 0.7154
- F1 Score: 0.7259
- AUC-ROC: 0.8032





Análisis Random Forest

- Accuracy: 0.7390 → El modelo acierta el 73.9% del total.
- Precision: 0.7347 → De las veces que predijo "victoria azul", acertó el 73.5%.
- Recall: 0.7564 → Detectó el 75.6% de las veces que ganó azul.
- F1 Score: 0.7454 → Balance adecuado entre precisión y recall.
- AUC-ROC: 0.8192 → Buen poder discriminativo entre clases.
- En cuanto a la matriz de confusión y la curva ROC, los resultados que se muestran en los gráficos son muy similares con respecto al modelo anterior.

Modelado

Modelo 3: XGBoost (Gradient Boosted Trees)

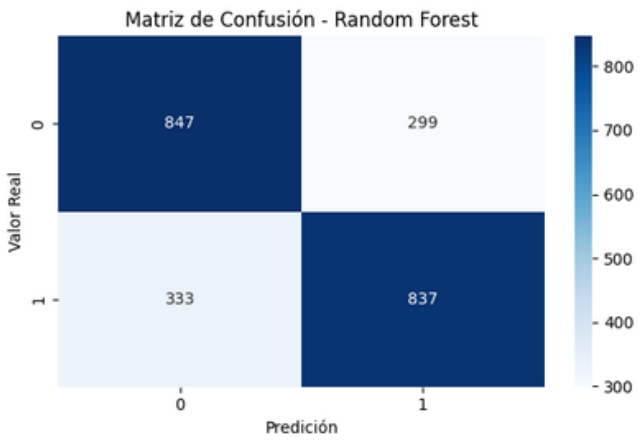
XGBoost es un algoritmo de boosting basado en árboles que construye modelos secuenciales donde cada nuevo árbol trata de corregir los errores del anterior. Es altamente eficiente y suele tener un gran desempeño en problemas de clasificación.

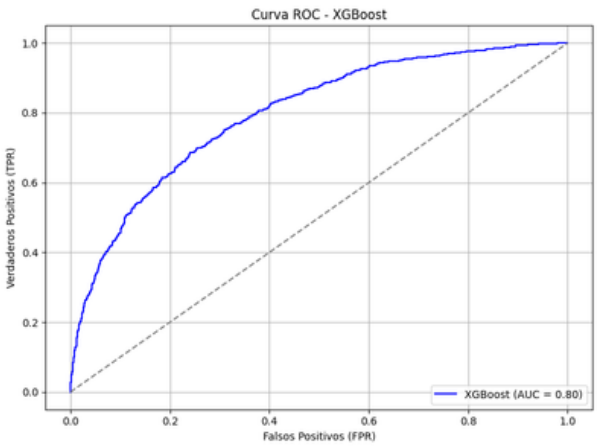
Hiperparámetros principales:

- n_estimators=300: número de árboles que se construirán (boosting rounds). se entrenan 300 árboles.
- max_depth=5: profundidad máxima de cada árbol. Aumentar puede llevar a overfitting. Típico: 3–10.
- learning_rate=0.05: tasa de aprendizaje que controla el impacto de cada árbol. Más bajo = mejor generalización pero más lento. Típico: 0.01–0.3.
- subsample=0.8: fracción de datos utilizada para entrenar cada árbol. Controla overfitting. Típico: 0.5–1. se usa el 80% de los datos en cada árbol.
- colsample_bytree=0.8: fracción de columnas usadas al construir cada árbol. Típico: 0.5–1. Se usa el 80% de las características en cada árbol.
- gamma=1: se requiere una mejora mínima para dividir nodos.
- reg_alpha=0.1 y reg_lambda=1: regularización L1 y L2.
- random_state: fija la semilla para reproducibilidad.

Resultado

- Accuracy: 0.7189
- Precision: 0.7247
- Recall: 0.7154
- F1 Score: 0.7200
- AUC-ROC: 0.7979





Modelado

Análisis XGBoost

- Accuracy: 0.7189 → El modelo acierta el 71.9% del total.
- Precision: 0.7247 → De las veces que predijo "victoria azul", acertó el 72.4%.
- Recall: 0.7154 → Detectó el 71.5% de las veces que ganó azul.
- F1 Score: 0.7200 → Balance adecuado entre precisión y recall.
- AUC-ROC: 0.7979 → Buen poder discriminativo entre clases.
- En cuanto a la matriz de confusión y la curva ROC, los resultados que se muestran en los gráficos son muy similares con respecto al modelo anterior.

Conclusiones Generales

Con las cifras de desempeño de cada modelo y los graficos añadidos podemos tener algunas conclusiones rápidas:
Regresión Logística

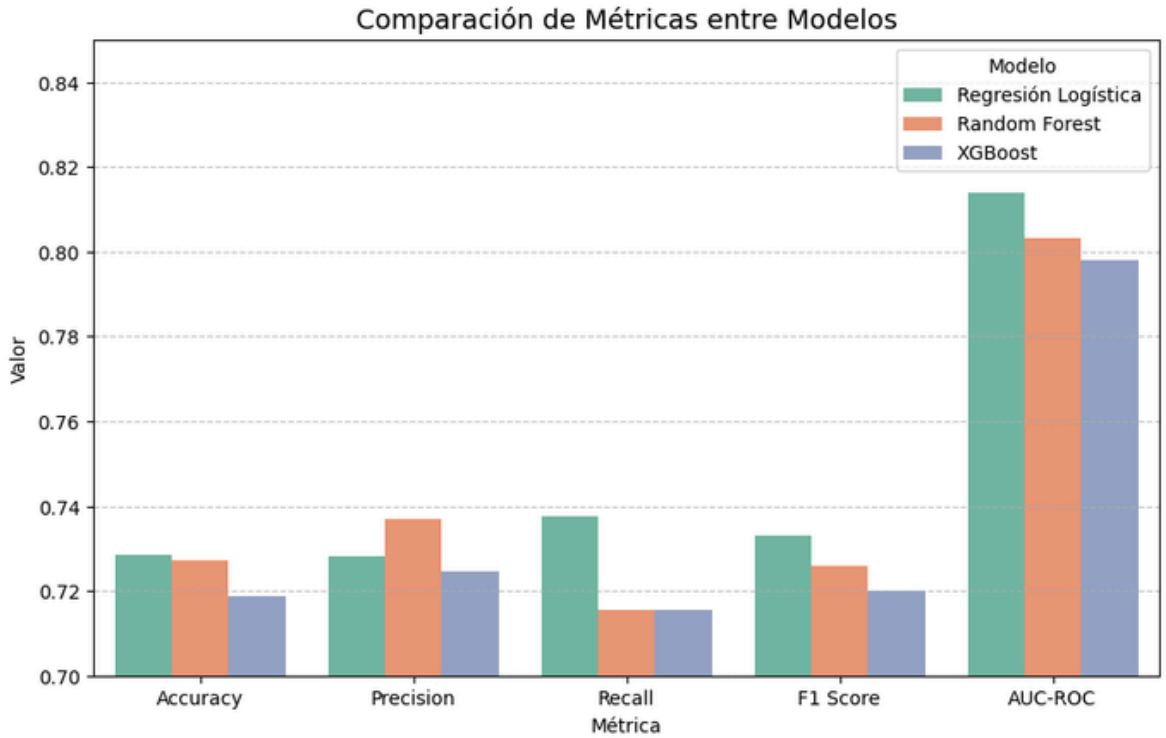
- Mejor Recall (0.7376) y mejor AUC-ROC (0.8139): esto indica que detecta correctamente más positivos y generaliza bien.
- Accuracy y F1 Score más altos: representa un mejor balance entre precisión y recall.

Random Forest

- Mejor Precisión (0.7368): indica que comete menos falsos positivos.
- Aunque tiene una ligera pérdida en Recall, su desempeño es muy equilibrado.
- F1 Score y AUC-ROC son cercanos a los de la regresión logística.

XGBoost

- Tiene el desempeño más bajo en casi todas las métricas, aunque sigue siendo competitivo.
- Es posible que el modelo necesite mayor ajuste de hiperparámetros o más datos para destacar.



Optimización de modelos

Con el objetivo de mejorar el rendimiento predictivo de los modelos desarrollados, en esta etapa se realiza un proceso de optimización de hiperparámetros mediante el uso de GridSearchCV, una técnica de búsqueda exhaustiva con validación cruzada. Este proceso permite identificar la mejor combinación de parámetros para cada algoritmo, maximizando su capacidad de generalización sobre datos no vistos.

A continuación, se optimizarán los siguientes modelos:

- Regresión Logística: Se ajustarán parámetros como la penalización (penalty) y la constante de regularización (C), con el fin de encontrar un balance entre simplicidad del modelo y capacidad predictiva.
- Random Forest: Se explorarán diferentes configuraciones del número de árboles (n_estimators), la profundidad máxima de los árboles (max_depth), el número mínimo de muestras por hoja (min_samples_leaf), entre otros. El objetivo es mejorar la precisión del modelo sin incurrir en sobreajuste.
- XGBoost: Se realizará un ajuste fino de parámetros como la tasa de aprendizaje (learning_rate), el número de iteraciones (n_estimators), la profundidad de los árboles (max_depth), entre otros. Este modelo será evaluado con la métrica logloss durante la validación cruzada.

Cada modelo optimizado será evaluado utilizando las mismas métricas usadas durante el proceso de modelado, permitiendo comparar su desempeño de manera objetiva y evidenciar que tanta mejora relativa presentan posterior a la optimización. Posterior al proceso los resultados fueron los siguientes:

Modelo	Accuracy Antes	Accuracy Después	Precision Antes	Precision Después	Recall Antes	Recall Después	F1 Score Antes	F1 Score Después	AUC-ROC Antes	AUC-ROC Después
Regresión Logística (GridSearchCV)	0.7284	0.73	0.7283	0.73	0.7376	0.73	0.7329	0.73	0.8139	0.8148
Regresión Logística (RandomizedSearchCV)	0.7284	0.72	0.7283	0.73	0.7376	0.72	0.7329	0.72	0.8139	0.8106
Random Forest	0.7271	0.73	0.7368	0.73	0.7154	0.73	0.7259	0.73	0.8032	0.8087
XGBoost	0.7189	0.73	0.7247	0.73	0.7154	0.73	0.7200	0.73	0.7979	0.8070

Dada la alta complejidad de búsqueda, la optimización no resulta rentable salvo en contextos donde se necesite maximizar cada decimal de rendimiento. Por lo tanto si vale la pena aplicar optimización si se dispone de tiempo y recursos computacionales para la búsqueda exhaustiva, de lo contrario podemos quedarnos con los modelos sin optimizar dado que los resultados no cambian en gran medida.

Conclusiones Generales

Importancia de la limpieza
La limpieza de datos y el análisis estadístico y de caracteristicas de los datos jugó un papel fundamental no solo en el desempeño de los modelos, sino también en el total entendimiento de las métricas importantes, de la relación que puede existir entre variables y como estas pueden explicar fenómenos e hipótesis planteadas al inicio del proyecto.

Desempeño Global de los Modelos
Los tres algoritmos evaluados —Regresión Logística, Random Forest y XGBoost— mostraron un desempeño competitivo, con resultados similares en términos de precisión, recall, F1-score y AUC-ROC, tanto antes como después de la optimización. La regresión logística, a pesar de su simplicidad, logró resultados sólidos comparables a modelos más complejos.

Impacto de la Optimización
La optimización de hiperparámetros mediante GridSearchCV y RandomizedSearchCV permitió mejoras discretas pero valiosas en todos los modelos, especialmente en la métrica AUC-ROC. Las mejoras más consistentes se observaron con GridSearchCV en el caso de la Regresión lineal, sin embargo los tiempos de ejecución de la optimización son considerables .

Modelo Recomendado
Aunque no hay un único modelo claramente superior, la regresión logística optimizada con regularización L1 representa una alternativa válida a pesar de ser el modelo "menos robusto", sin embargo, cualquiera de los modelos utilizados es perfectamente viable como modelo principal dadas las pocas diferencias en su desempeño final.

Hipótesis a probar:

Hipótesis 1 (Análisis Exploratorio): "Existen ciertos indicadores clave que aumentan significativamente la probabilidad de victoria, como el control de objetivos neutrales (dragones/heraldos) y la visión (wards)."

Durante el análisis exploratorio, se identificaron correlaciones consistentes entre la adquisición temprana de dragones y heraldos con una mayor probabilidad de victoria. Equipos que obtienen el primer dragón o heraldo muestran una ventaja temprana y suelen traducirla en control de mapa y oro.
En cuanto a la visión (wards colocados y destruidos), si bien hubo diferencias entre equipos ganadores y perdedores, estas no fueron tan marcadas como otros indicadores (oro, kills, objetivos mayores). La visión puede complementar el rendimiento, pero no se comportó como un predictor fuerte por sí sola.
Pero definitivamente, las métricas que mandan a la hora de definir una victoria, son la cantidad de oro y experiencia obtenida, en particular la variable que define la diferencia entre un equipo y otro, esta se deriva a su vez de la consecusión de objetivos y el KDA por equipo, que fué una métrica que se creó dentro del análisis realizado.

Conclusiones Generales

Hipótesis 2 (Modelado Predictivo): "Es posible predecir el equipo ganador con buena precisión usando solo los datos objetivos de rendimiento del equipo a lo largo de los 10 primeros minutos de la partida."

Los modelos predictivos desarrollados con variables objetivas de los primeros 10 minutos lograron precisiones cercanas al 73%, con AUC-ROC superiores al 0.80 en los mejores casos, lo que indica un buen poder discriminativo.
Esto demuestra que el rendimiento temprano del equipo, medido en objetivos como asesinatos, oro acumulado, control de dragones/heraldos y torres, es un indicador suficientemente fuerte para predecir el resultado final de la partida en una etapa temprana.

Nota importante: Aunque el rendimiento temprano predice bien la victoria, aún existe un margen de error, lo cual refleja la naturaleza dinámica del juego (comebacks, errores estratégicos, etc.). Por eso, el modelo no puede ser perfecto, pero sí útil para predicción temprana y toma de decisiones en tiempo real.

Posibles puntos de mejora

- Podemos realizar una conexión permanente a la API de RIOT para tratar de obtener una muestra más significativa y completa, incluso si es posible con información de la partida entera y no de los primeros 10 minutos, con miras a tener mejor calidad de datos que nos permita aumentar la precisión del modelado.
- Validar más a detalle los casos en que los modelos definen falsos positivos y negativos, con miras a evaluar la caracteristicas que ofrecen los datos y los límites a los que puede llegar cada uno de los modelos usados.