Casa de Apuestas para la UEFA Champions League

Carlota Sánchez González

Abril 2024

Resumen

Índice

1. Introduction

En la intersección de la tecnología y el deporte, la Inteligencia Artificial y el Machine Learning han emergido como herramientas fundamentales que han transformado la forma en que se abordan y se comprenden los eventos deportivos de alto nivel, como la UEFA Champions League. Estas tecnologías han permitido un análisis más profundo de los datos, desde el rendimiento de los jugadores hasta las estrategias de equipo, brindando una ventaja competitiva tanto dentro como fuera del campo.

Las apuestas deportivas han ganado una enorme popularidad en las últimas décadas, convirtiéndose en una industria multimillonaria con un impacto significativo en la economía global y en la sociedad en general. Con el avance de la tecnología y la proliferación de plataformas de apuestas en línea, la forma en que los aficionados interactúan con los deportes ha evolucionado considerablemente.

En este trabajo, analizaremos el campo del Machine Learning, examinando una variedad de técnicas y algoritmos utilizados para resolver diferentes problemas, que van desde la predicción de resultados de partido de fútbol hasta la clasificación de imágenes de escudos de fútbol. Este trabajo se centra en el análisis de datos y el uso de técnicas avanzadas de machine learning para predecir los resultados de los partidos de la UEFA Champions League, con el objetivo de desarrollar un modelo predictivo que pueda ser utilizado en casas de apuestas.

Comenzamos por explorar la recopilación, preparación y visualización de los datos. Posteriormente, nos enfocaremos en los diferentes tipos de aprendizaje supervisado y no supervisado, analizando algoritmos como regresión, clasificación y modelos de series temporales. También examinaremos el mundo de los modelos profundos, como las Redes Neuronales Artificiales y las Convolutional Neural Networks, conocidas por su alto rendimiento en diversas tareas. Finalizaremos nuestra exploración al investigar el aprendizaje por refuerzo y su aplicación en entornos de toma de decisiones.

En este trabajo explicaremos todos los puntos que he explorado en mi proyecto para conseguir los modelos con los mejores resultados y así unas predicciones mejores.

2. Datos

2.1. Recopilación

Para comenzar la creación de nuestra base de datos, hemos realizado una búsqueda exhaustiva en la web para encontrar fuentes de información ajustada a nuestro objetivo. Estas fuentes debían ser accesibles, confiables y libres de licencias restrictivas que pudieran limitar su uso. Aunque encontré varias páginas web que ofrecen datos sobre la UEFA Champions League, muchas de ellas presentaban información incompleta, no permitían la extracción automatizada de datos o no contaban con la información necesaria para nuestros fines. Entre las páginas web revisadas estaban:

- https://es.uefa.com/uefachampionsleague/ Aunque tiene información valiosa y confiable que podía utilizar, presentaba limitaciones de acceso, lo que limitó mi capacidad para obtener información directamente de esta fuente.
- https://www.thesportsdb.com/ A pesar de ser una fuente prometedora debido a su amplia base de datos y su enfoque en proporcionar información detallada sobre una variedad de eventos deportivos, encontramos que tenía datos incompletos a partir de cierto año, lo que limitaba su utilidad para nuestro proyecto.
- https://es.wikipedia.org/wiki/Liga_de_Campeones_de_la_UEFA Fue la página que inicialmente empecé a utilizar, pero me di cuenta de que no tenía la suficiente información que estaba buscando. Además, a partir de cierto año los datos cambiaban de estructura y iba a ser menos eficiente de scrapear.
- https://fbref.com/es/comps/8/stats/Estadisticas-de-Champions-League Esta página fue útil, tiene mucha información valiosa y variada y se mantiene actualizada. De aquí saqué los datos de los partidos de las últimas 20 temporadas de la UEFA y las estadísticas de los jugadores.

- https://www.bdfutbol.com/es/ De esta página saqué el listado de equipos participantes de la UEFA en todos los años con sus estadísticas. También saqué los entrenadores y la trayectoria de cada entrenador.
- https://www.sofascore.com/es/ Aunque tiene información valiosa y confiable que podía ser de utilidad, presentaba limitaciones de acceso, lo que limitó mi capacidad para obtener información directamente de esta fuente.

Para obtener información detallada sobre entrenadores, jugadores y partidos, implementé el algoritmo de Web Scraping. Inicialmente, intenté usar solo BeautifulSoup, pero debido a la naturaleza dinámica de la tabla de datos de los jugadores, también incorporé Selenium para lograr la extracción completa y precisa de los datos. Al finalizar el proceso, los datos extraídos se almacenaron en archivos .csv en la carpeta 'data' de mi proyecto. Las tablas resultantes incluyen información sobre equipos, entrenadores, trayectorias de entrenadores, partidos y jugadores. Cabe destacar que, debido a cambios en las estadísticas a partir del año 2017-2018, se creó una segunda tabla para los jugadores que en un futuro ajustaremos para tener una tabla de jugadores.

2.2. Preparación

Considerando la naturaleza de los datos, primero creé unos archivos .json donde almacené los IDs de los equipos, entrenadores y jugadores. Posteriormente, llevé a cabo un procesamiento para garantizar la integridad de las tablas, asegurando la ausencia de valores faltantes y estructurándolas de manera óptima. Durante este proceso, ajusté las tablas según mi criterio, incorporando estadísticas que consideré relevantes y potencialmente beneficiosas para mi investigación. Finalmente, estas nuevas tablas tratadas fueron guardadas en la carpeta 'dataframe' de mi proyecto. Estos son los dataframes que utilizaremos para el trabajo.

Tabla Equipo.csv

- idEquipo
- Nombre: Nombre del equipo
- Pais
- T: Temporadas jugadas por ese equipo
- PJ: Partidos jugados en la historia de la UEFA por el equipo.
- PG: Partidos ganados en la historia de la UEFA por el equipo.
- PE: Partidos empatados en la historia de la UEFA por el equipo.
- PP: Partido perdido en la historia de la UEFA po rel equipo.
- %: Porcentaje de partidos ganados, empatados y perdidos.
- Títulos: Titulos de la Champions conseguidos por el equipo.
- F: Temporadas en las que el equipo quedó subcampeón.
- 1/2: Temporadas en las que el equipo se quedoó en semifinales.
- 1/4: Temporadas en las que el equipo se quedó en cuartos.
- 1/8: Temporadas en las que el equipo se quedó en octavos.
- 1/16: Temporadas en las que el equipo se quedó en fase de grupos.
- GF: Goles a favor del equipo en todas las temporadas jugadas.
- GC: Goles en contra del equipo en todas las temporadas jugadas.
- TA: Tarjetas amarillas.
- TR: Tarjetas rojas.

Tabla Entrenadores

- idEntrenador
- Apodo: Apodo del entrenador
- Nombre: Nombre completo del entrenador.
- T: Temporadas jugadas por el entrenador.
- PJ: Partidos jugados por el entrenador.
- PG: Partidos ganados por el entrenador.
- PE: Partidos empatados por el entrenador.
- PP: Partidos perdidos por el entrenador.
- %: Porcentaje de partidos ganados, empatados y perdidos.

Tabla Trayectoria Entrenadores

- idEntrenador: Corresponde con el id de la tabla de entrenadores
- Temporada: Temporada del entrenador.
- Equipo: Equipo del que fue entrenador esa temporada.
- PJ: Partidos jugados con ese equipo esa temporada.
- PG: Partidos ganados con ese equipo esa temporada.
- PE: Partidos empatados con ese equipo esa temporada.
- PP: Partidos perdidos con ese equipo esa temporada.

Tabla Jugadores

- Temporada
- idJugador
- Jugador: Nombre completo del jugador
- Equipo
- Edad
- PJ: Partidos jugados
- Titular: Partidos en los que fue titular.
- Mín: Minutos jugados por ese jugador esa temporada.
- Gls.: Goles anotados por el jugador esa temporada.
- Ass: Asistencias del jugador esa temporada.
- TP: Penaltis metidos.
- TPint: Penaltis intentados.
- TA: Tarjetas amarillas.
- TR: Tarjetas rojas.
- xG: Goles esperados.
- npxG: Goles esperados sin penaltis.

- PrgC : Acarreos de balón que avanzan hacia la línea de gol del oponente al menos 10 yardas desde su punto más alejado en los últimos seis pases, o cualquier acarreo al área penal. Se excluyen los acarreos que terminan en la defensa del 50 % del campo.
- PrgP: Pases completos que avanzan el balón hacia la línea de gol del oponente al menos 10 yardas desde su punto más alejado en los últimos seis pases, o cualquier pase completo al área penal. Se excluyen los pases del 40 % de la defensa del campo.
- PrgR: Pases progresivos recibidos, pases completos que recibe un jugador y que avanzan el balón hacia la línea de gol del oponente al menos 10 yardas desde su punto más alejado en los últimos seis pases, o cualquier pase completo al área penal.Excluye los pases del 40 % de la defensa del campo.
- Valoracion: Puntuación del jugador según el algoritmo creado por mí.

Tabla Champions

La tabla partidos la separé en dos tablas: Champions (partidos de la temporada 2003/2004-2022/2023) y Champions_23_24 (partidos champions 2023/2024). La tabla champions será la que usaremos para entrenar los modelos y la otra para las predicciones.

- idPartido
- Temporada: Temporada en la que se jugó el partido. Para brindar una mayor claridad y facilidad de manipulación de los datos, la información sobre la temporada en la que se disputó el partido se convirtió al formato datetime, utilizando el estándar año-mes-día, siendo el año, el año en el que se incia la temporada correspondiente.
- Ronda: Ronda en la que se disputó el partido.
- Local: ID del equipo que jugó como local ese partido.
- Visitante: ID del equipo que jugó como visitante ese partido.
- Evento: Donde se jugó el partido.
- GolesLocal: Goles que anotó el equipo local.
- GolesVisitante: Goles que anotó el equipo visitante.
- VictoriaLocal: Variable binaria que indica 1 si ganó el local.
- Emapte: Variable binaria que indica 1 si el partido acabó en empate.
- VictoriaVisitante: Variable binaria que indica 1 si ganó el equipo visitante.
- %_Victorias_Local: Porcentaje de victorias del equipo local cuando esos dos equipos jugaron anteriormente, independientemente de quien fuera el local.
- %_Empate: Porcentaje de partidos que acabaron en empate cuando estos dos equipos se enfrentaron.
- %_Victoria_Visitante:Porcentaje de victorias del equipo visitante cuando esos dos equipos jugaron anteriormente, independientemente de quien fuera el visitante.
- %_Equipo1_Ganado: Porcentaje de partidos que ha ganado el equipos 1 (equipo local de este partido) independientemente de si fue local o visitante cuando jugó contra el equipo 2 (equipo visitante de este partido).
- %_Equipo2_Ganado: Porcentaje de partidos que ha ganado el equipos 2 (equipo visitante de este partido) independientemente de si fue local o visitante cuando jugó contra el equipo 1 (equipo local de este partido).
- %_1_G_Temporada: Gandos por el equipo 1 en esa temporada.
- %_1_G_Temporada_L: Ganados por el equipo 1 en esa temporada, pero solo como local.

- %_1_E_Temporada_L: Empatados por el equipo 1 en esa temporada, pero solo como local.
- %_1_P_Temporada_L: Perdidos por el equipo 1 en esa temporada, pero solo como local.
- 1_Media_G: Media de partidos ganados frente a jugador por el equipo 1.
- 1_Media_G_Local: Media de partidos ganados como local frente a jugados como local.
- 1_Media_Goles_PP: Media de goles por partido del equipo 1 esa temporada.
- 1_ValorJugadores: Suma de la columna 'Valoracion' de la tabla jugadores de los jugadores del equipo 1.
- 1_MediaJugadores: Media de la valoración de los jugadores del equipo 1.
- %_2_G_Temporada: Gandos por el equipo 2 en esa temporada.
- %_2_G_Temporada_L: Ganados por el equipo 2 en esa temporada, pero solo como local.
- %_2_E_Temporada_L: Empatados por el equipo 2 en esa temporada, pero solo como local.
- %_2_P_Temporada_L: Perdidos por el equipo 2 en esa temporada, pero solo como local.
- 2_Media_G: Media de partidos ganados frente a jugador por el equipo 2.
- 2_Media_G_Local: Media de partidos ganados como local frente a jugados como local.
- 2_Media_Goles_PP: Media de goles por partido del equipo 2 esa temporada.
- 2_ValorJugadores: Suma de la columna 'Valoracion' de la tabla jugadores de los jugadores del equipo 2.
- 2_MediaJugadores: Media de la valoración de los jugadores del equipo 2.

2.2.1. Algoritmo de valoración del jugador

Para obtener una evaluación global que incluyera las estadísticas más relevantes de un jugador, diseñé una ponderación utilizando los siguientes criterios:

- 'Mín': 0.05
- 'PJ': 0.15
- 'Gls.': 0.40
- 'Ass': 0.20
- 'TR': -0.15
- 'PrgP': 0.15
- 'PrgR': 0.15

La ecuación utilizada es esta:

$$\begin{aligned} \text{ValoracionNoNormalizada} = & \text{Mín} \times \text{peso['Mín']} \\ & + \text{PJ} \times \text{peso['PJ']} \\ & + \text{Gls.} \times \text{peso['Gls.']} \\ & + \text{Ass} \times \text{peso['Ass']} \\ & + \text{TR} \times \text{peso['TR']} \\ & + \text{PrgP} \times \text{peso['PrgP']} \\ & + \text{PrgR} \times \text{peso['PrgR']} \end{aligned} \tag{1}$$

Para tener la puntuación en un rango del 0 al 100 hemos normalizado los datos de esta manera:

$$\text{Valoracion} = 100 \times \frac{\text{ValoracionNoNormalizada} - \text{min\_valoracion}}{\text{max\_valoracion} - \text{min\_valoracion}} \tag{2}$$

ValoracionNoNormalizada: Este término representa la valoración original del jugador antes de la normalización.

min_valoracion: Es el valor mínimo de las valoraciones de todos los jugadores en el conjunto de datos.

max_valoracion: Es el valor máximo de las valoraciones de todos los jugadores en el conjunto de datos.

De esta manera, se generó una puntuación única para cada jugador, que reflejaba su desempeño en base a estas métricas. Por ejemplo para los jugadores del Real Madrid de la temporada 2022/2023 tendríamos estas puntuaciones:

	Jugador	Valoracion
18839	David Alaba	44.688458
18862	Marco Asensio	24.821246
18894	Karim Benzema	52.655771
18928	Eduardo Camavinga	44.484168
18939	Dani Carvajal	56.996936
18942	Dani Ceballos	6.792646
18969	Thibaut Courtois	47.293156
19103	Eden Hazard	9.601634
19144	Vinicius Júnior	78.345250
19186	Toni Kroos	59.754852
19222	Andriy Lunin	9.295199
19233	Mariano	0.561798
19264	Ferland Mendy	23.340143
19272	Éder Militão	42.798774
19280	Luka Modrić	52.247191
19304	Nacho	26.404494
19396	Rodrygo	61.695608
19401	Antonio Rüdiger	38.661900
19489	Aurélien Tchouaméni	29.673136
19516	Jesús Vallejo	1.021450
19517	Federico Valverde	69.765066
19521	Lucas Vázquez	15.372829

Figura 1: Valoración jugadores Real Madrid 2022/2023

2.3. Visualización

La exploración visual de los datos al inicio del análisis es importante porque permite identificar patrones, tendencias y posibles anomalías de manera intuitiva. La visualización ha sido útil para realizar una comprensión inicial de los datos. Esta comprensión ha sido de ayuda para tomar las decisiones posteriores en el análisis de datos.

Asimismo, al explorar gráficamente los datos, es posible identificar las características más relevantes para el análisis o modelado posterior, lo que ayuda a priorizar ciertas características y descartar otras.

La visualización del número de títulos de la Champions League que tiene cada equipo de esta temporada proporciona una comprensión rápida y clara del historial y el rendimiento de los equipos en esta competición. Esto permite identificar a los equipos con más éxito en la UEFA. Además, nos permite comparar el rendimiento de diferentes equipos. Con nuestros datos de los equipos, he creado un gráfico de barras con la cantidad de títulos que tiene cada equipo.

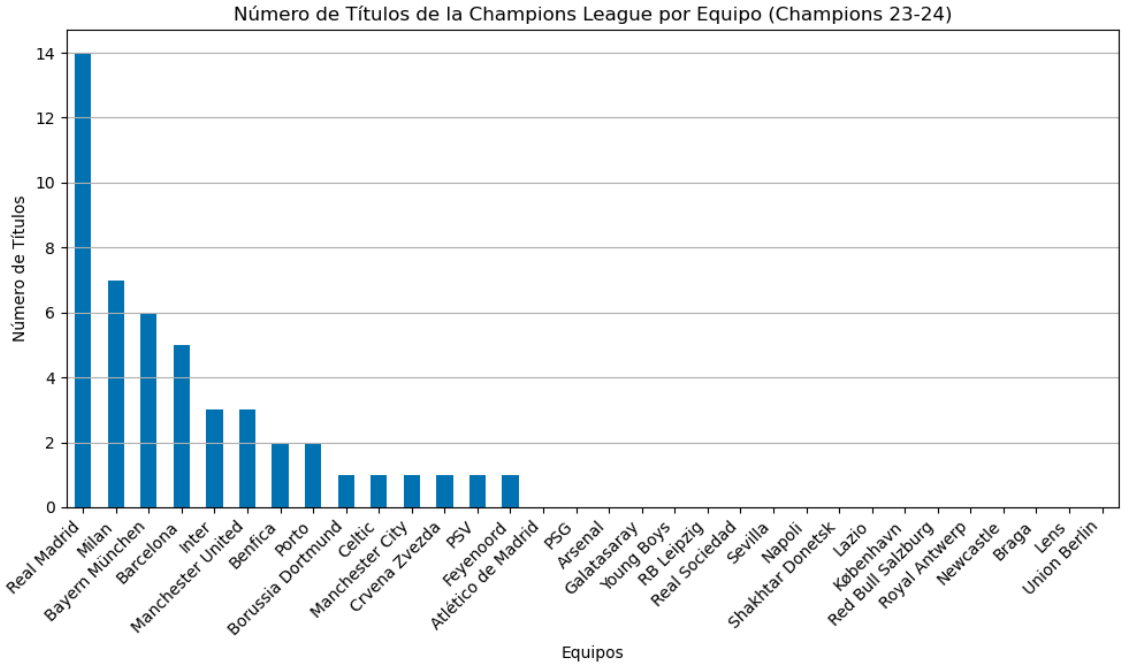

Figura 2: Títulos por equipo

Con este gráfico vemos que el Real Madrid encabeza la lista con la mayor cantidad de títulos, seguido por otros equipos destacados como Bayern München y Barcelona.

Luego de visualizar los títulos de los equipos, también consideré relevante examinar la media de goles, lo que proporciona una visión adicional sobre el rendimiento de cada equipo tanto en sus partidos en casa como fuera de ella. Para esta visualización, utilizamos un gráfico de barras donde se muestra la media de goles por equipo como local y visitante.

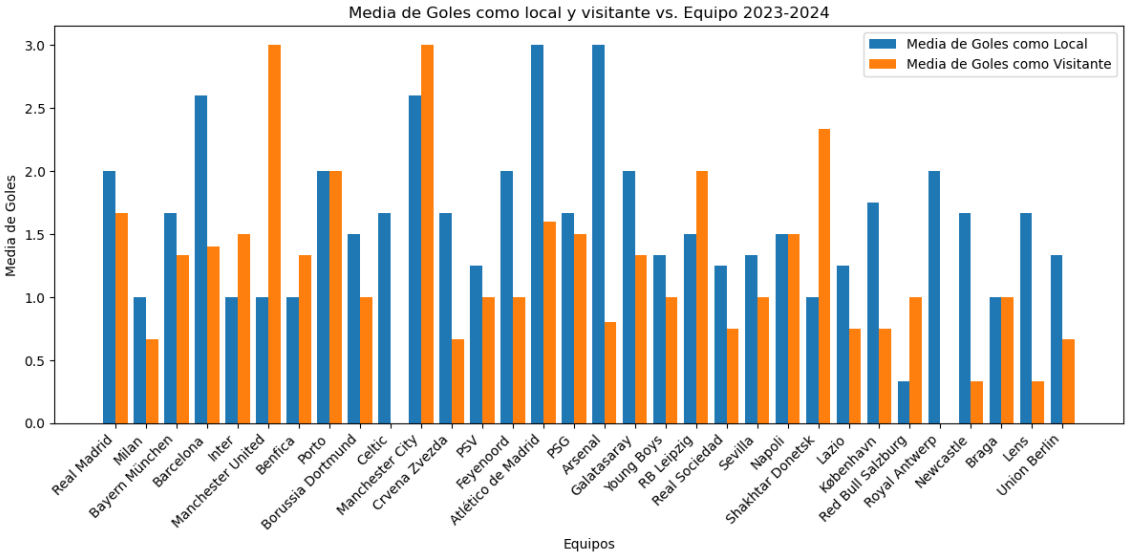

Figura 3: Media de goles por equipo como local y visitante

También he explorado el número de victorias de cada equipo a lo largo de las temporadas. Para esto, calculamos el número total de temporadas en nuestros datos y distribuimos los resultados en una matriz de subgráficos. Cada subgráfico muestra un histograma de barras que representa el número de victorias de cada equipo en una temporada específica. Este enfoque nos permite visualizar las tendencias de victorias de cada equipo a lo largo del tiempo, lo que puede ser útil para identificar equipos dominantes en diferentes temporadas y observar cambios en el rendimiento a lo largo del tiempo.

Además, exploramos la distribución de resultados de los partidos mediante un gráfico circular. Esta representación visual nos proporciona una visión rápida de la proporción de victorias locales, empates y victorias visitantes en nuestros datos. Cada segmento del círculo muestra el porcentaje de cada resultado en relación con el total de partidos, lo que nos permite comprender rápidamente cómo se distribuyen los resultados de los partidos en nuestra muestra de datos y qué proporción de los partidos terminan en victoria local, empate o victoria visitante.

Aparte de eso, también hemos explorado la distribución de resultados de los partidos mediante un gráfico circular. Este gráfico nos proporciona una representación visual de la proporción de victorias locales, empates y victorias visitantes en nuestros datos. Esto nos permite comprender rápidamente cómo se distribuyen los resultados de los partidos en nuestra muestra de datos y qué proporción de los partidos terminan en victoria local, empate o victoria visitante.

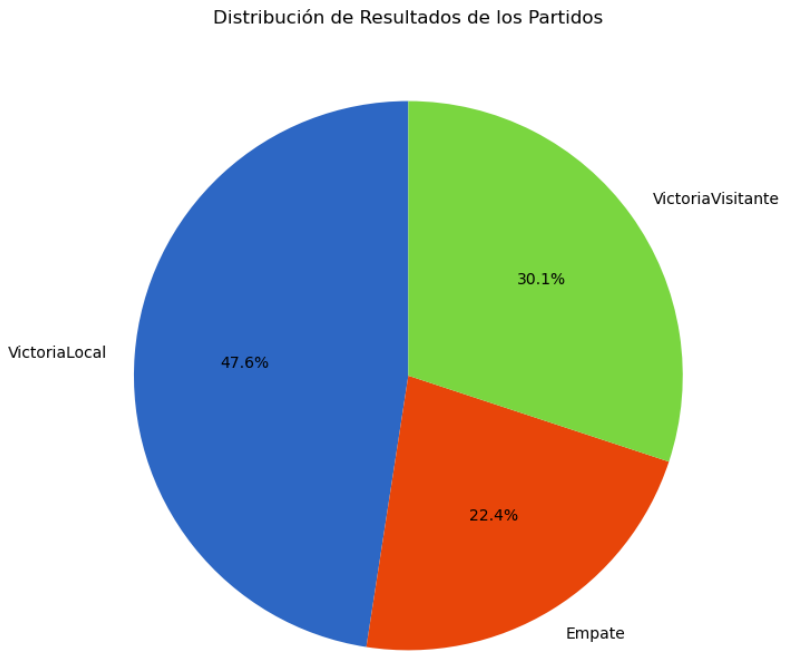

Figura 4: Distribución de Resultados de los Partidos

Observamos que la proporción de victorias locales es significativamente mayor en comparación con los empates y las victorias del equipo visitantes. Este hallazgo sugiere que los equipos locales tienen una ventaja competitiva en los partidos de nuestra muestra de datos de la Champions League.

Por último, una observación muy útil es el rendimiento de los jugadores. Aquí tenemos una representación visual de cómo cambia el promedio de goles por partido en relación con el valor promedio de los jugadores para cada equipo en cada temporada de la competición. En cada gráfico, el eje x representa el valor promedio de los jugadores, mientras que el eje y representa el promedio de goles por partido. Cada punto en el gráfico representa un equipo en una temporada específica. He realizado un gráfico por cada temporada. Por ejemplo, el gráfico de la última temporada es este.

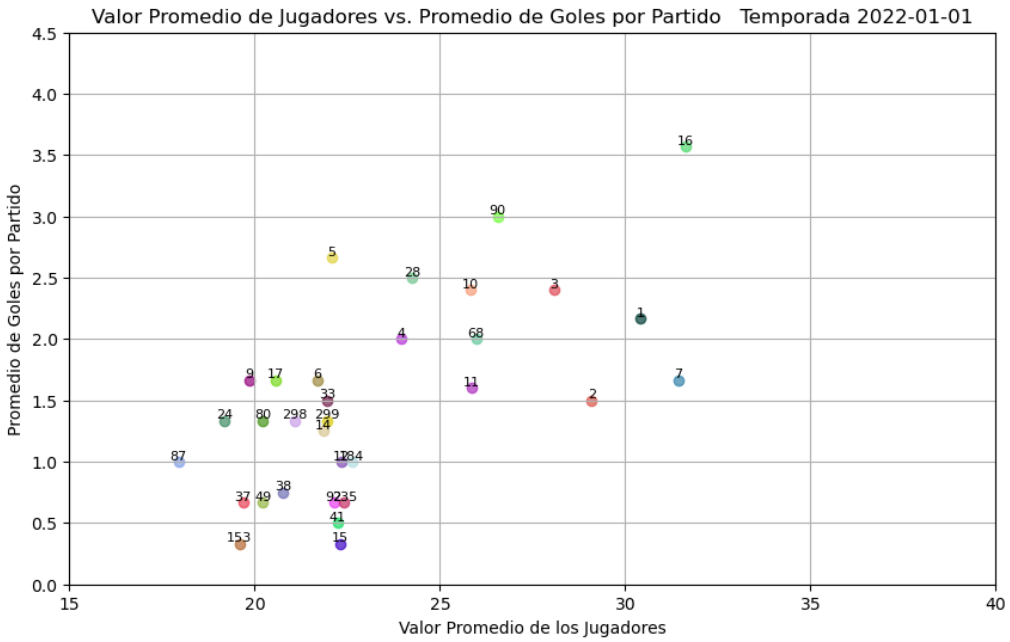

Figura 5: Relación entre el Valor Promedio de los Jugadores y el Promedio de Goles por Partido para cada Equipo en cada Temporada

Observamos que en la última temporada, el equipo que más goles anotó por partido es el 16, que según nuestros datos de equipos corresponde al Manchester City, ganador de esa temporada y también el que tiene el mayor valor promedio de jugadores. Sin embargo, también podemos notar que el equipo con el ID 7, el Inter de Milán, tiene un valor promedio muy similar al del Manchester City, pero el número promedio de goles no destaca.

3. Aprendizaje supervisados

En el aprendizaje supervisado dividimos los datos en conjunto de entrenamiento y prueba. El conjunto de entrenamiento enseña a los modelos a producir la salida deseada, incluye tanto las entradas como las salidas correctas, lo que permite que el modelo aprenda con el tiempo.

El aprendizaje supervisado es la tarea de aprender una función que asigna una salida a una entrada basándose en ejemplos de pares entrada y salida. Cada dato es un par que consta de un objeto de entrada (usualmente un vector) y un valor de salida deseado. Un algoritmo de aprendizaje supervisado analiza los datos de entrenamiento y genera una función, que puede usarse para asignar valores de salida a nuevos ejemplos. Se requiere que el algoritmo generalice, a partir de los datos de entrenamiento, las situaciones que todavía no ha visto.

Este enfoque de aprendizaje se divide en dos tipos principales en la minería de datos: clasificación y regresión. La clasificación consiste en asignar con precisión datos de prueba a categorías específicas, identificando entidades dentro del conjunto de datos y sacando conclusiones sobre cómo deben etiquetarse o definirse. Entre los algoritmos de clasificación más utilizados se encuentran la regresión logística, los árboles de decisión, el método de los k vecinos más cercanos y métodos de algoritmos combinados como el bosque aleatorio y el aumento del gradiente, cada uno empleado en este trabajo.

Por otro lado, la regresión se emplea para comprender la relación entre variables dependientes e independientes, a menudo utilizada para realizar proyecciones basadas en datos históricos o conocidos. Algunos algoritmos de regresión comúnmente utilizados incluyen la regresión lineal, regresión Lasso, árboles de decisión, bosques aleatorios y aumento del gradiente.

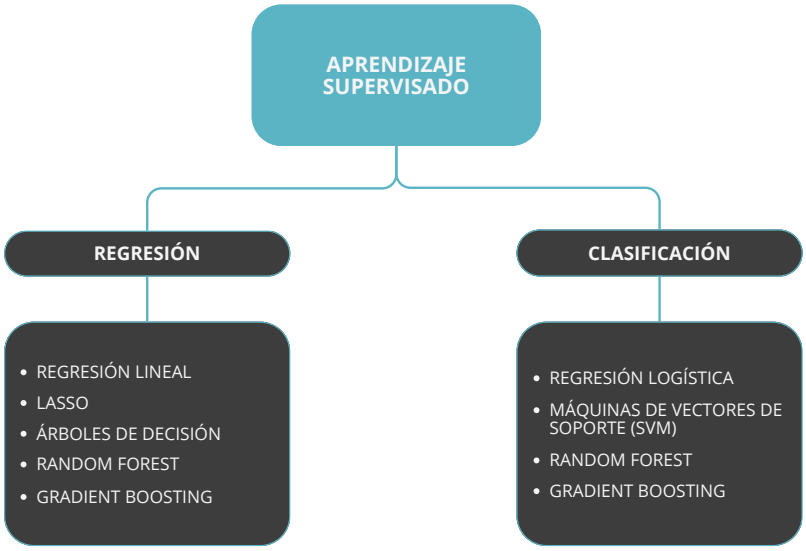

Figura 6: Modelos utilizados de aprendizaje supervisados

3.1. Regresión

Para el modelo de regresión, utilicé los modelos de la Figura 1. Inicialmente, visualicé la matriz de correlación para ver qué características no iban a ser útiles para mi modelo y visualicé la distribución de mis características. Para la regresión, establecí como etiquetas los goles del equipo local y los goles del equipo visitante. La regresión es el caso más complicado de acertar en todo el proyecto, puesto que implica estimar una cifra de goles cuando estos dependen de muchos factores que, además, no todos están en mis datos.

Para el preprocesamiento de los datos, en primer lugar, transformé y normalicé las características numéricas del conjunto de datos mediante la función que creé llamada 'preprocesamiento'. En segundo lugar, utilicé un Pipeline para ver cómo variaban mis modelos mediante estos dos preprocesamientos.

Aparte de esto, para todos los modelos tenemos una función 'evaluate_regression_model' que nos dará las métricas de evaluación: error cuadrático medio (MSE), raíz del error cuadrático medio (RMSE) y R2; y la función 'visualize_regression', donde podemos visualizar la calidad del modelo de regresión al observar cómo se comparan las etiquetas reales con las predicciones generadas por el modelo. En el gráfico, el eje x representa las etiquetas reales y el eje y representa las etiquetas predichas por el modelo. Cada punto en el gráfico representa una instancia del conjunto de prueba, donde la posición del punto indica la etiqueta real y la predicción correspondiente para esa instancia. Además del gráfico de dispersión, la función también añade una línea de regresión, también conocida como hiperplano, que muestra la relación entre las etiquetas reales y las predicciones. Esta línea se ajusta a los datos para mostrar la tendencia general de las predicciones en relación con las etiquetas reales. La pendiente y la intersección de esta línea proporcionan información sobre la precisión y el sesgo del modelo de regresión.

3.1.1. Regresión lineal

El análisis de regresión lineal es uno de los algoritmos más simples en el aprendizaje supervisado, y se utiliza para encontrar una línea recta que describa la tendencia de un conjunto de datos determinado. Calcula una función a la que, pasándole los parámetros de entrada, nos da el de salida. En mi caso el parámetro de salida que quiero que me devuelva son los goles que meterá el equipo local y los goles que meterá el equipo visitante en un partido futuro.

3.1.2. Lasso

Lasso, que significa "Least Absolute Shrinkage and Selection Operator", es una técnica de regularización utilizada en modelos de regresión lineal. Agrega una penalización L1 a la función

de pérdida, lo que conduce a la contracción de algunos coeficientes a cero y, por lo tanto, a la selección automática de características. Esto hace que Lasso sea útil en conjuntos de datos con muchas características como el nuestro, ya que simplifica el modelo al eliminar características menos importantes. Sin embargo, la elección del parámetro de regularización es crítica para evitar el subajuste o el sobreajuste del modelo.

En mi caso, los parámetros de salida que deseo obtener son los goles que marcará tanto el equipo local como el equipo visitante en un partido futuro.

3.1.3. Árboles de decisión

Un árbol de regresión es una técnica de modelado que utiliza la suma de cuadrados y el análisis de regresión para predecir valores continuos. Cada nodo en el árbol representa un valor medio pronosticado, y el árbol se construye dividiendo los nodos en función de las variables predictoras que minimizan la suma de cuadrados. El proceso comienza con un nodo raíz que abarca todos los datos y se divide en nodos hijo hasta que no se puede mejorar más la precisión, o el tamaño del árbol se vuelve demasiado grande. La fuerza predictiva del árbol se evalúa mediante R2; si es superior al 10 %, indica que el modelo predictivo es confiable.

3.1.4. Random Forest

En lugar de depender de un solo modelo, los algoritmos ensamblados utilizan la idea de que la combinación de múltiples modelos puede producir predicciones más precisas y robustas que cualquier modelo individual. En mi trabajo, he entrenado modelos de regresión utilizando el algoritmo Random Forest y Gradient Boosting.

Random Forest consiste en construir varios árboles y combinar sus predicciones basadas en las características de entrada para seleccionar una ruta óptima de predicción.

Este modelo nos da mejores valores en las métricas utilizando Pipeline. Por lo tanto, será el modelo que guardaremos.

3.1.5. Gradient Boosting

Como he mencionado antes, Gradient Boosting consiste en construir un modelo predictivo a partir de una serie de modelos de regresión más débiles y combinar sus predicciones para mejorar la precisión del modelo final. En lugar de construir cada árbol de forma independiente, el algoritmo ajusta secuencialmente cada nuevo árbol para corregir los errores del modelo anterior. Esto se logra mediante la optimización de una función de pérdida que mide la diferencia entre las predicciones del modelo actual y las etiquetas verdaderas. Con cada iteración, el algoritmo se centra en los puntos de datos que el modelo actual predice incorrectamente, ajustando así el modelo para mejorar su rendimiento.

Para Gradient Boosting, he establecido un GridSearchCV para encontrar la mejor combinación de los parámetros de este modelo que nos dé un mejor resultado. Esta optimización de los hiperparámetros nos proporciona los mejores valores para este modelo.

3.1.6. Resultados

Para seleccionar los mejores modelos, he evaluado los modelos con las mejores métricas, preprocesando los datos de las dos formas y realizando la optimización de los hiperparámetros en el caso de Gradient Boosting.

Modelo	MSE	RMSE	R2
RL Local	1.626235	1.275239	0.168486
GB Local	1.595392	1.263088	0.184256
RF Local	1.567793	1.252115	0.198368
RL Visitante	1.315570	1.146983	0.114506
GB Visitante	1.258920	1.122016	0.152636
RF Visitante	1.308848	1.144049	0.119031

Cuadro 1: Resultados de las métricas de evaluación para la predicción de goles

Podemos interpretar el RMSE como la diferencia promedio de goles entre las predicciones del modelo y los goles reales, es decir, nuestras futuras predicciones tendrán un error máximo de 1.27 goles. Así se verían los gráficos de los errores y el R2 para mis modelos de regresión.

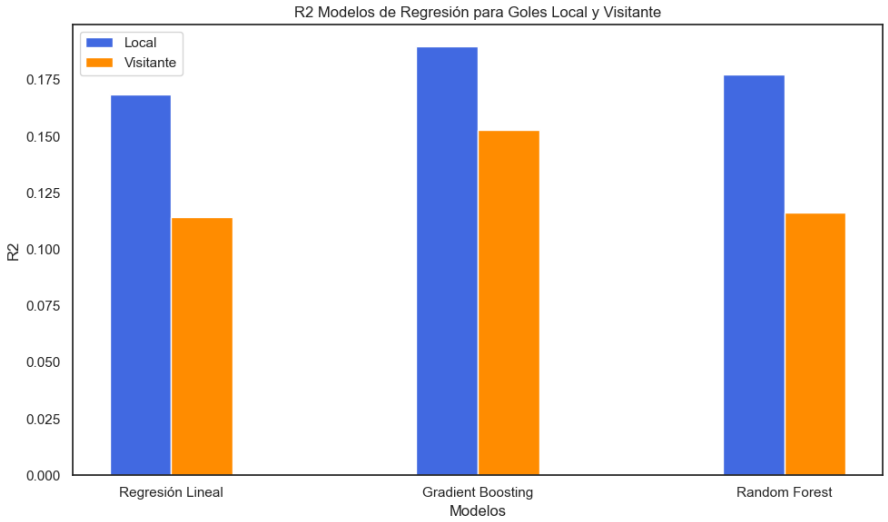

Figura 7: Coeficiente de determinación

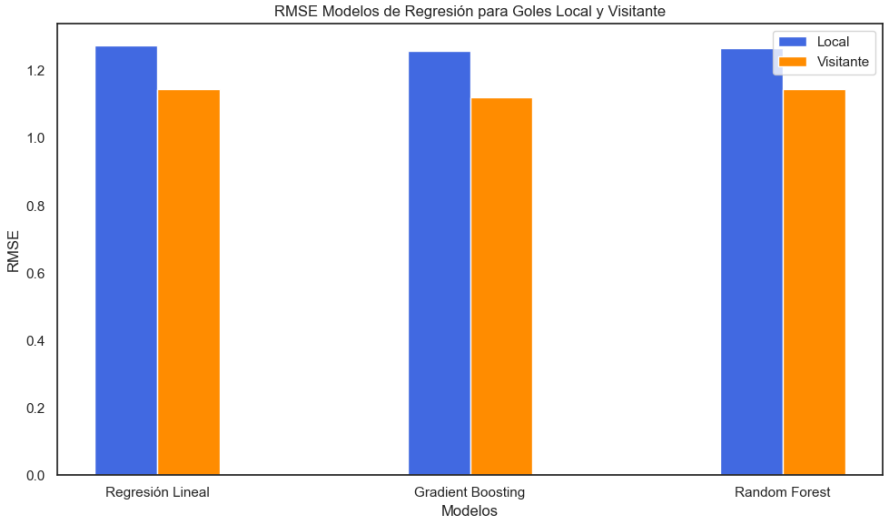

Figura 8: Raíz del error cuadrático medio

3.2. Clasificación

Para mi trabajo, he utilizado clasificación multiclase ya que quiero predecir si gana el local, el visitante o si hay empate. Primero, he tenido que crear una columna llamada 'Victoria' con los valores 1 si el partido lo ganó el equipo local, 2 si lo ganó el equipo visitante y 3 si empataron; esta será nuestra etiqueta con esas tres clases. Después, hemos visualizado las instancias que tenemos por cada clase con un gráfico de barras de los registros por clase. En este caso, he utilizado Pipeline para el preprocesado de datos y la creación de los modelos.

En este caso, he creado una clase de evaluación con las métricas:

- Accuracy: Es la proporción de predicciones correctas sobre el total de predicciones. Indica qué tan bien el modelo clasifica correctamente las instancias en general.
- Precision: Proporción de instancias correctamente identificadas como positivas sobre todas las instancias clasificadas como positivas por el modelo. Indica qué tan precisas son las predicciones positivas del modelo.

- Recall: Proporción de instancias correctamente identificadas como positivas sobre todas las instancias positivas en el conjunto de datos. Indica la capacidad del modelo para identificar correctamente todas las instancias relevantes.
- AUC: El Área bajo la Curva ROC (AUC) es una medida de la capacidad de discriminación del modelo. Un valor de AUC cercano a 1 indica un modelo con un buen rendimiento en la clasificación, mientras que un valor cercano a 0.5 indica un rendimiento similar al azar.

La clase también tiene funciones para graficar la matriz de confusión para todos los modelos y la curva ROC. En el caso de un modelo de clasificación multiclase, no es posible una única curva ROC que muestre la tasa de verdaderos positivos en comparación con la tasa de falsos positivos. Sin embargo, podemos utilizar las tasas de cada clase en una comparación Uno vs Resto (OVR) para crear un gráfico ROC para cada clase.

3.2.1. Regresión Logística

La regresión logística adopta la función sigmoide para analizar datos y predecir clases discretas en un conjunto de datos, convirtiéndola en una estrategia de clasificación. Como hemos mencionado, funciona aplicando una función sigmoide a una combinación lineal de las características de entrada, lo que produce una salida en el rango de 0 a 1 que se interpreta como la probabilidad de pertenencia a una clase particular.

3.2.2. SVM

SVM es una técnica avanzada para clasificar datos, similar a la regresión logística pero más estricta. Destaca en trazar límites de clasificación en datos linealmente separables. Utiliza un hiperplano para dividir los puntos de datos en clases, maximizando la distancia entre él y los subconjuntos.

El margen entre el hiperplano y los puntos de datos es una característica clave, proporcionando un respaldo para clasificar nuevos datos y siendo menos sensible a anomalías que otros métodos como la regresión logística. SVM también es efectivo en detectar y mitigar anomalías, minimizando su impacto en la ubicación del límite de decisión. Su verdadera fuerza radica en datos de alta dimensión, utilizando múltiples características con variantes como SVC lineal, polinomial y Kernel Trick, este último mapeando datos a espacios dimensionales más altos para una mejor separación.

En mi implementación, he utilizado la estrategia 'ovr', esto significa que SVM construye múltiples clasificadores binarios, uno para cada clase, y luego combina sus resultados. Además, he configurado el parámetro 'probability' en True para obtener estimaciones de probabilidad de pertenencia a cada clase y el 'random_state' en 42 para asegurar la reproducibilidad de los resultados.

3.2.3. Random Forest

Un Random Forest consta de múltiples árboles de decisión, cada uno de los cuales genera una predicción. Al realizar una tarea de clasificación, cada árbol de decisión en el Random Forest vota por una de las clases a las que pertenece la entrada. Por ejemplo, en nuestro conjunto de datos sobre partidos de fútbol y quisiéramos determinar el resultado de un partido, cada uno de los árboles de decisión en un random forest emitirá un voto por el resultado al que cree que pertenece un partido. Una vez que todos los árboles hayan llegado a una conclusión, el random forest contará qué clase (gana local, empate o gana visitante) tuvo el voto más poblado y esta clase será lo que el Random Forest genere como predicción.

3.2.4. Gradient Boosting

El Boosting es una técnica ampliamente adoptada que se centra en corregir los errores y los datos clasificados incorrectamente por iteraciones anteriores para construir un modelo final. Entre sus variantes más populares se encuentran el aumento de gradiente y AdaBoost. Este método implica el entrenamiento de múltiples modelos algorítmicos que difieren entre sí, utilizando los mismos datos de entrenamiento. Luego, se selecciona aquel que demuestre un mejor rendimiento en los datos de prueba.

Modelo	Accuracy	Precision	Recall	AUC
Logistic Regression	0.619493	0.450295	0.516732	0.745174
Random Forest	0.624833	0.542712	0.543529	0.743458
Gradient Boosting	0.658211	0.593534	0.578173	0.771693
SVM	0.627503	0.478725	0.523986	0.730540

Cuadro 2: Resultados de evaluación de modelos

3.2.5. Resultados

Después de aplicar y entrenar los modelos, los resultados de la evaluación han sido estos:

Podemos ver que mirando el mejor modelo que es el aumento del gradiente, llegamos a tener un 65 % de exactitud.

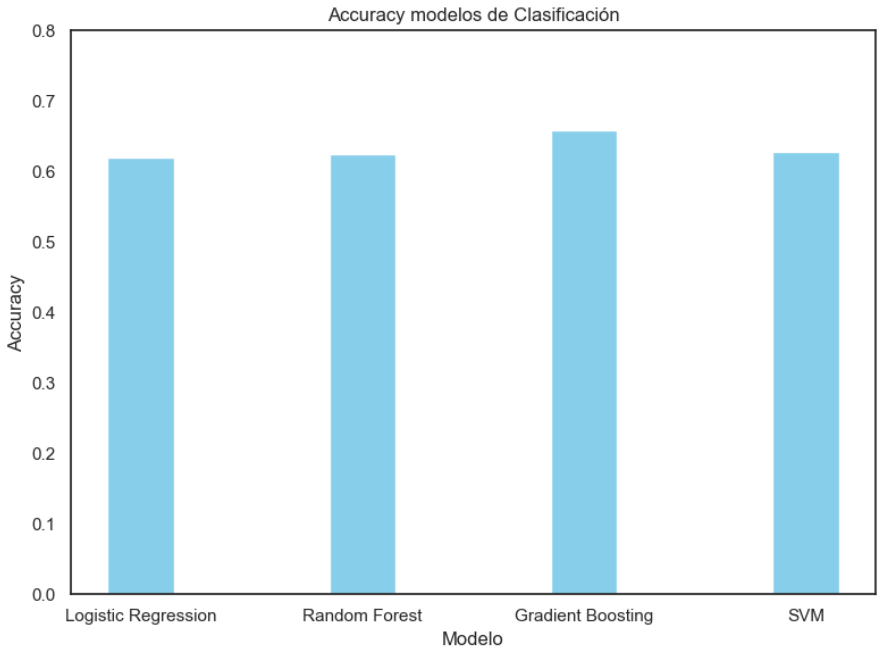

Figura 9: Rendimiento modelos de clasificación

3.3. Series Temporales

Una serie temporal es una sucesión de observaciones de una variable tomadas en varios instantes de tiempo. Nos interesa estudiar los cambios de los goles a lo largo del tiempo.

El modelo ARIMA, que significa "Promedio Móvil Integrado Autorregresivo", se divide en tres componentes: AR (Autorregresivo), I (Integrado) y MA (Promedio Móvil). El componente autorregresivo (AR) se refiere a cómo cada valor en una serie temporal depende de los valores previos en la misma serie. El componente de promedio móvil (MA) describe cómo cada valor en la serie temporal depende de los errores de predicción anteriores. El componente integrado (I) se utiliza para hacer que la serie temporal sea estacionaria.

El modelo ARIMA combina estos tres componentes para proporcionar una predicción precisa de una serie temporal. Es particularmente útil para modelar datos que muestran tendencias y estacionalidad. Además del ARIMA básico, existen variantes como SARIMA, que permite modelar series estacionarias, así como series estacionales y no estacionarias. Una serie estacionaria es una secuencia de datos en el tiempo que tiene propiedades estadísticas constantes a lo largo del tiempo. Esto significa que sus características, como la media, la varianza y la autocovarianza, no cambian con el tiempo. En otras palabras, las estadísticas de la serie no dependen del momento específico en el que se observan, lo cual es una condición importante en muchos análisis y modelos estadísticos y econométricos.

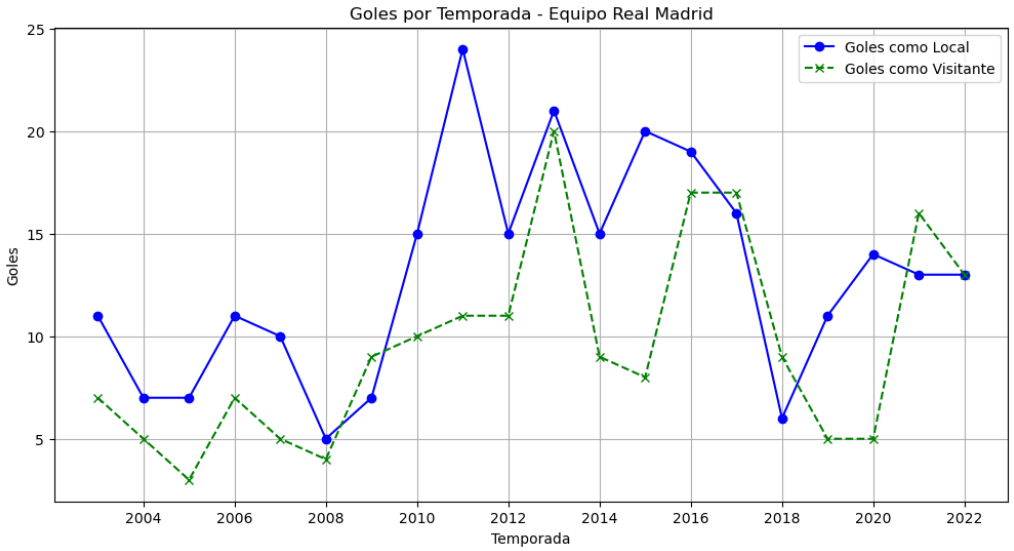

Figura 10: Serie Temporal goles por temporada Real Madrid

4. Modelos no supervisados

El aprendizaje no supervisado se vale de algoritmos de machine learning para analizar conjuntos de datos sin etiquetar y organizarlos en clústeres. Consigue producir conocimiento únicamente de los datos que se proporcionan como entrada, sin necesidad de explicarle al sistema qué resultado se quiere obtener. Estos algoritmos tienen la capacidad de identificar patrones subyacentes y agrupaciones de datos de forma automática, sin requerir intervención directa del ser humano. Los modelos de aprendizaje no supervisado se utilizan para tres tareas principales: agrupación en clústeres, asociación y reducción de dimensionalidad. Para nuestro estudio, hemos aplicado agrupaciones por clústeres basadas en los equipos para examinar sus características y composiciones, de esta manera podremos ver los diferentes niveles de equipos que hay y a qué nivel pertenece cada uno.

Para aplicar los modelos de clusterización, inicialmente he normalizado mis datos con MinMaxScaler y he reducido su dimensionalidad con la técnica PCA. En total, en mis tenemos 12 características que podríamos interpretarlo como coordenadas que describen una instancia en un espacio de doce dimensiones. Como el espacio de 12 dimensiones es difícil de visualizar, he utilizado la técnica PCA (Análisis de Componentes Principales) para analizar las relaciones entre características y así reducir la dimensionalidad a 2 dimensiones.

La agrupación en clústeres es una estrategia de análisis de datos que organiza conjuntos de datos no etiquetados basándose en sus semejanzas o disparidades. Se emplea para agrupar objetos de datos sin procesar y sin clasificar en conjuntos representados por patrones o estructuras en la información. Dentro de la clusterización hay diferentes algoritmos que pueden ser clasificados en diferentes tipos, como exclusivos, superpuestos, jerárquicos y probabilísticos.

En mi trabajo he intentado explorar varios tipos de modelos de clusterización para quedarme con el que más me encaje. Para evaluar los diferentes algoritmos de clusterización, he creado una función que nos devuelve estas métricas:

- Silhouette Score: Esta métrica cuantifica la cohesión y la separación de los clústeres. Un valor cercano a 1 indica que los puntos están bien agrupados dentro de sus clústeres y están separados de otros clústeres. Un valor cercano a -1 indica que los puntos están mal asignados a sus clústeres y podrían pertenecer a otros clústeres. Un valor cercano a 0 indica una superposición de clústeres. En general, valores más altos son mejores.
- Calinski-Harabasz Index: También conocido como criterio de varianza entre grupos, es una medida de cuánto mejor son los clústeres en comparación con los puntos distribuidos aleatoriamente. Valores más altos indican clústeres más densos y bien separados.
- Davies-Bouldin Score: Esta métrica mide la compacidad dentro de los clústeres y la separación entre los clústeres. Un valor más bajo indica clústeres mejor definidos.

4.1. Clusterización

Algoritmo K-means

El objetivo del algoritmo de clustering K-Means es dividir un conjunto de observaciones en un número predefinido de grupos, donde cada observación se asigna al grupo cuyo centroide está más cercano. K-Means requiere solo el conjunto de datos de entrada y el número de clústeres deseado (k) para operar.

En K-Means, primero se colocan k puntos llamados centroides en el espacio de datos, representando los centros iniciales de los clusters. Luego, cada punto de datos se asigna al cluster cuyo centroide está más cercano. Después, se recalcula la posición de cada centroide tomando la media de las posiciones de todos los puntos asignados a su cluster. Este proceso se repite hasta que los centroides dejan de cambiar significativamente de posición.
Este proceso continúa hasta que los centroides convergen y no hay cambios significativos en sus posiciones, lo que indica que se ha alcanzado una solución estable.

Para encontrar el número de clusters óptimo para mis datos, he utilizado la técnica llamada "método del codo", este método implica ajustar el modelo de agrupamiento con diferentes números de clústers y calcular el WCSS para cada modelo. Luego, se traza un gráfico del número de clusters en el eje x y el valor de WCSS en el eje y. WCSS es una métrica para medir la compacidad de los grupos formados por el modelo.

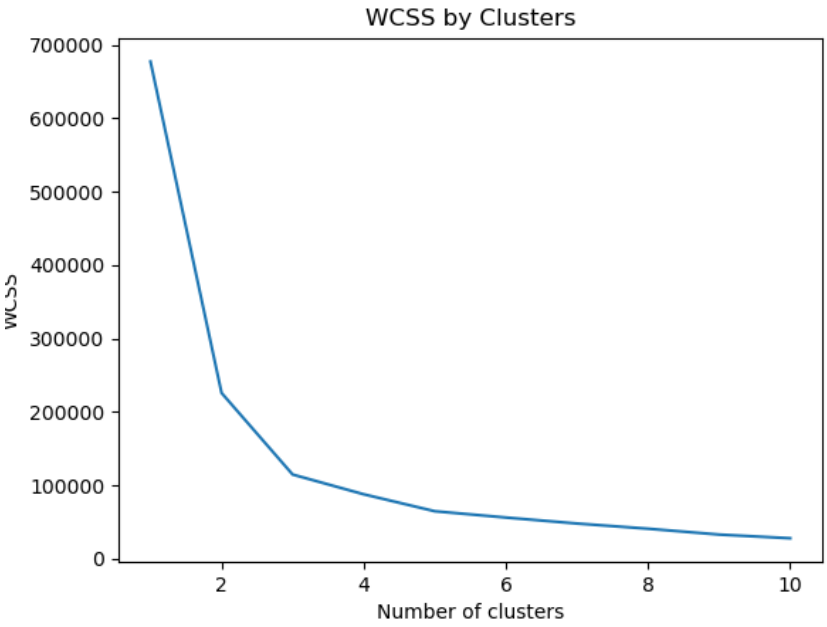

Figura 11: Método del codo

El 'codo' es el punto en el que la tasa de cambio de la variabilidad intraclúster comienza a disminuir significativamente, indicando que agregar más clústeres no proporciona beneficios significativos en términos de separación de los datos. Podemos ver un cambio notable de pendiente en 2 y 3 clústers.

Dado que con el método del codo hemos comprobado que a partir de 3 clúster la curva empieza a ser más suave, hemos utilizado ese número de clústers para el algoritmo K-means.

Clusterización jerárquica

Es un algoritmo de agrupación en clústeres no supervisado que se puede categorizar de dos formas: aglomerante o divisivo. En nuestro caso, hemos hecho uso del aglomerante, ya que la agrupación en clústeres divisiva no se suele utilizar. Este método implica que los puntos de datos se

aíslan inicialmente como agrupaciones separadas y luego se fusionan iterativamente en función de su similitud hasta que se alcanza un clúster. Normalmente, se emplean cuatro métodos distintos para calcular la similitud: enlace de Ward, promedio, completo y único. En mi caso, he hecho uso del enlace completo, ya que prefería cómo se organizaban los equipos en comparación con los enlaces.

Algoritmo Mean-Shift

Mean Shift es un algoritmo de clustering basado en densidad, busca los picos de densidad en un conjunto de datos para encontrar los centroides de los clusters. Es un método de agrupación no paramétrico, lo que significa que no asume el número de clusters a priori. En lugar de eso, el algoritmo Mean Shift utiliza un rango de búsqueda variable (parámetro bandwidth) sobre el conjunto de datos. Este rango de búsqueda determina cuánto se extiende la influencia de cada punto de datos en la determinación de los clusters. A medida que el algoritmo avanza, se desplaza hacia las regiones de mayor densidad, ajustándose hasta que converge a los centroides de los clusters.

He desarrollado una función para encontrar el ancho de banda óptimo basado en los datos y una serie de valores de ancho de banda predefinidos. Esta función calcula los puntajes de silueta (métrica Silhouette Score) para diferentes valores de ancho de banda y selecciona aquel que maximiza la cohesión dentro de los clústeres y la separación entre ellos. Una vez determinado el ancho de banda óptimo, utilizo este valor para crear el modelo de clustering Mean Shift.

DBSCAN

Este tipo de clustering se basa en la densidad de los puntos de datos. Los clusters se forman alrededor de las regiones de alta densidad de puntos, mientras que los puntos aislados se consideran ruido. En este tipo tampoco hace falta pasarle el número de clusters. Este algoritmo es especialmente útil donde hay presencia de ruido y outliers.

Los parámetros que hay que pasarle a este modelo es eps, que define la dsitancia entre dos puntos es menor o igual a 'eps', entonces se consideran vecinos, y min_samples que es número mínimo de vecinos (puntos de datos) dentro del radio de eps. En mi trabajo 0, 1, 2 y 3 son los tres clusters diferentes y -1 es el ruido.

GMM

Los modelos de mezcla gaussiana (GMM) son una categoría de modelos estadísticos mixtos. Estos están compuestos por una combinación de diferentes distribuciones de probabilidad, en el caso de un GMM, distribuciones gaussianas. Un GMM no está limitado a un número fijo de clústeres, sino que el modelo puede adaptarse para ajustarse a la cantidad de grupos que se ajusten mejor a los datos. En lugar de tener una sola forma de distribución para tus datos, como una campana de Gauss típica, el GMM permite tener múltiples formas de distribución, cada una representando un posible grupo o 'clúster' en los datos.

4.2. Resultados

Mirando las métricas de evaluación y los gráficos de los clústers, considero que el mejor modelo que se ajusta a mis datos es el modelo con el algoritmo K-Means. De esta manera tenemos 3 agrupaciones, una con equipos de alto nivel (clúster 1), otro con equipos nivel medio (clúster 2) y otro equipos nivel bajo (clúster 0).

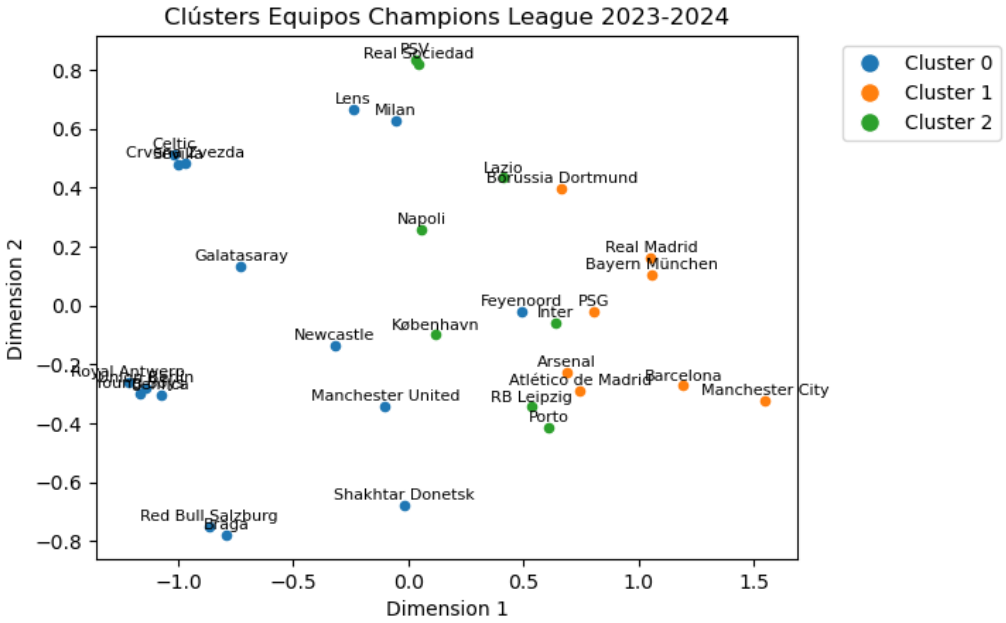

Figura 12: Clúster equipos

5. Modelos profundos

El aprendizaje profundo es un método de la inteligencia artificial (IA) que enseña a las computadoras a procesar datos de una manera que se inspira en el cerebro humano. Los modelos de aprendizaje profundo son capaces de reconocer patrones complejos en imágenes, textos, sonidos y otros datos, a fin de generar información y predicciones precisas. Es posible utilizar métodos de aprendizaje profundo para automatizar tareas que habitualmente requieren inteligencia humana, como la descripción de imágenes o la transcripción a texto de un archivo de sonido.

El aprendizaje profundo, una rama del machine learning, se distingue por su capacidad para mejorar su rendimiento al acceder a grandes volúmenes de datos. A diferencia de los algoritmos tradicionales de aprendizaje automático, que tienen una capacidad de aprendizaje limitada, los sistemas de aprendizaje profundo pueden adquirir más experiencia al procesar una cantidad significativa de datos. Las redes neuronales artificiales aprenden mediante la detección de estructuras complejas en los datos que reciben. Al crear modelos computacionales compuestos por varias capas de procesamiento, las redes pueden crear varios niveles de abstracción que representen los datos. Una vez que las máquinas han acumulado suficiente experiencia a través del aprendizaje profundo, pueden emplearse para realizar tareas específicas con mayor eficacia.

5.1. Redes Neuronales Artificiales

En las redes neuronales artificiales, el aprendizaje se organiza en capas, con las capas iniciales encargadas de detectar patrones simples y las capas posteriores encargadas de combinar y abstraer esos patrones en conceptos más complejos. Con cada capa adicional, la red puede aprender representaciones cada vez más abstractas y sofisticadas de los datos de entrada.

Las redes neuronales artificiales son un tipo de machine learning que pretende imitar el funcionamiento del cerebro humano. Para ello, se emplean nodos (neuronas) que se conectan entre ellas en distintas capas y que intentan imitar el modelo de aprendizaje del cerebro humano. Cada nodo individual es como su propio modelo de regresión lineal, formado por un vector de entrada, un vector de pesos, sesgo y una salida.

El entrenamiento de una Red Neuronal Artificial (RNA) se lleva a cabo utilizando un algoritmo llamado backpropagation, que se basa en la regla de la cadena. Básicamente, después de cada paso hacia adelante a través de la red, backpropagation realiza un paso hacia atrás mientras ajusta los pesos y sesgos del modelo. Este ajuste se realiza utilizando información sobre el gradiente de la

función de costo con respecto a los pesos de la red.

El gradiente es una medida de la inclinación o la pendiente de una función en un punto dado. En el contexto del entrenamiento de una RNA, el gradiente de la función de costo con respecto a los pesos indica cómo cambiar los pesos para disminuir la función de costo. Este gradiente se calcula utilizando el algoritmo de retropropagación, que propaga el error desde la capa de salida hacia atrás hasta la capa de entrada, calculando el gradiente de la función de costo con respecto a cada peso mediante la regla de la cadena.

El objetivo del algoritmo backpropagation es minimizar la función de costo durante el proceso de aprendizaje, mejorando así la precisión de las predicciones de la red neuronal. Para lograr esto, backpropagation ajusta los pesos y sesgos de cada nodo de la red en función del gradiente calculado. Este proceso se repite iterativamente, actualizando los pesos y sesgos de la red hasta que se alcanza un mínimo local o global de la función de costo y la red neuronal alcanza un nivel aceptable de precisión en sus predicciones.

Debido a la linealidad que presenta la ecuación de cada nodo, si se intentara enlazar neuronas una detrás de otra, no se obtendría mayor ventaja que usando una única neurona, porque la composición de funciones lineales es lineal. Para introducir no linealidad en la red y permitirle aprender patrones más complejos, se utilizan funciones de activación. Estas funciones se aplican a la salida de cada neurona y pueden incluir la función sigmoide, la función ReLU (Rectified Linear Unit) y la función tangente hiperbólica.

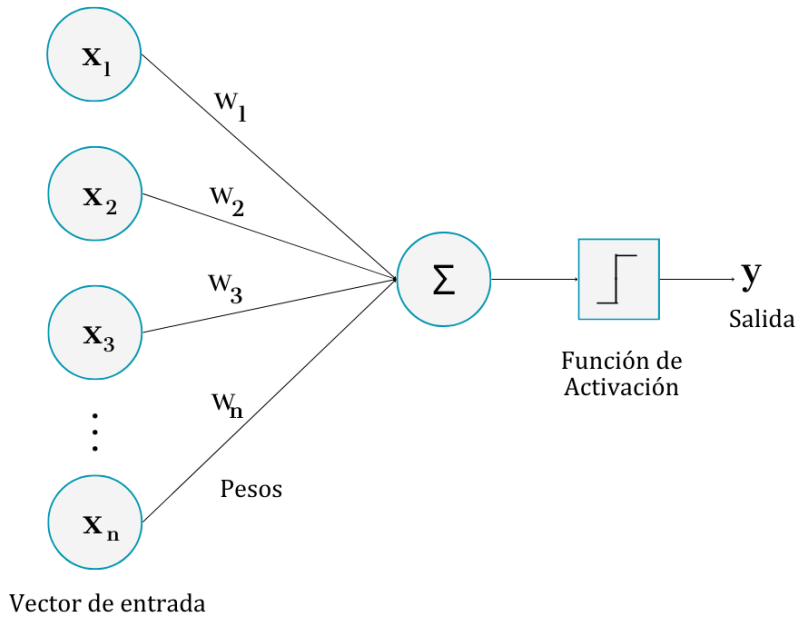

Figura 13: Red Neuronal Artificial

5.2. Funciones de Activación

- Sigmoide: Esta función de activación toma cualquier rango de valores a la entrada y los mapea al rango de 0 a 1 a la salida. En la actualidad esta función de activación tiene un uso limitado, y realmente su principal aplicación es la clasificación binaria.

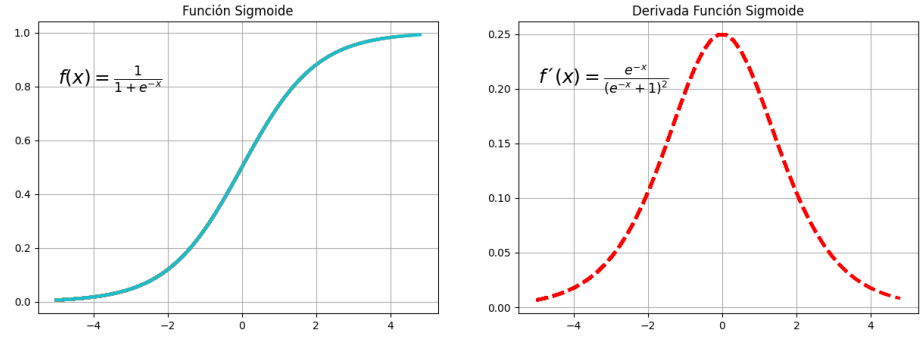

Figura 14: Función Sigmoide. Fuente: https://medium.com/escueladeinteligenciaartificial/funciones-de-activacion-para-redes-neuronales-de00fefb7150

- Tangente hiperbólica: Esta función tiene un comportamiento muy similar a la sigmoidal, con la diferencia de que los valores de salida estarán en el rango de -1 a 1. La ventaja que tienen las tangenciales hiperbólicas (tanh) tienen la ventaja de que las entradas negativas se transforman en valores negativos fuertemente acentuados, y las entradas positivas se convierten en valores positivos fuertemente acentuados. Esto conduce a resultados que tienden a los extremos. Al igual que la función sigmoide, la función tanh es diferenciable y monótona, aunque su derivada no lo es.

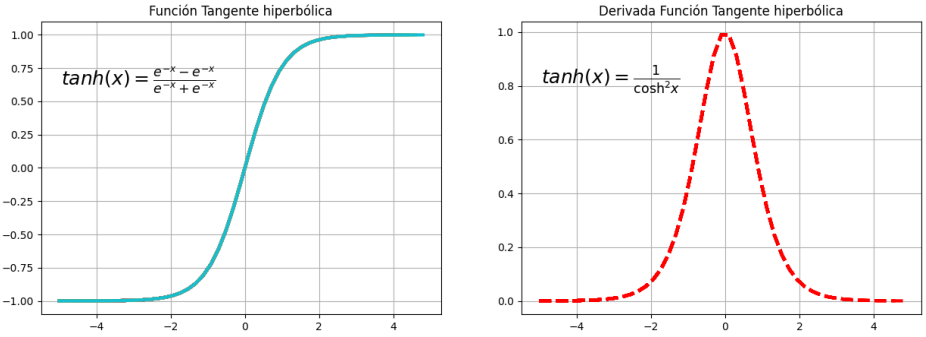

Figura 15: Función Tangente Hiperbólica. Fuente: https://medium.com/escueladeinteligenciaartificial/funciones-de-activacion-para-redes-neuronales-de00fefb7150

- ReLU: Es la función de activación más utilizada en el mundo en estos momentos. Esta función generará una salida igual a cero cuando la entrada sea negativa y una salida igual a la entrada cuando dicha esta última sea positiva. Esto puede ayudar mucho en la simplificación computacional ya que todos los valores iguales a 0 son inmediatamente descartados (dichas neuronas son irrelevantes).

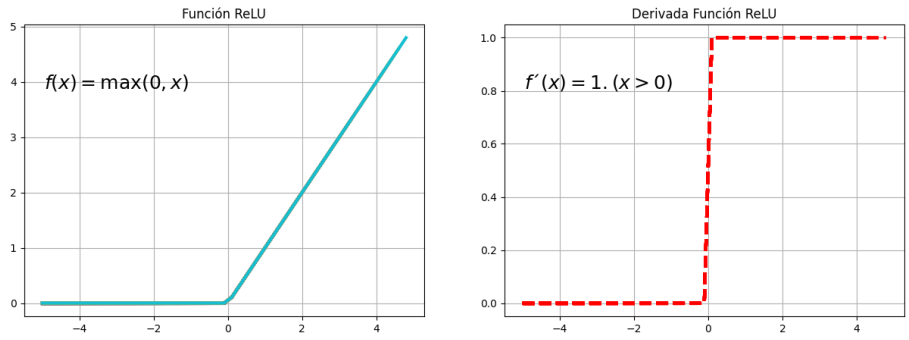

Figura 16: Función ReLU. Fuente: https://medium.com/escueladeinteligenciaartificial/funciones-de-activacion-para-redes-neuronales-de00fefb7150

- SoftMax: Es una función de activación importante en la clasificación multiclase. Esta función se utiliza para convertir las salidas de la capa anterior en probabilidades que suman uno. La función aplica una función exponencial para cada valor de z (vector de entrada) y lo normaliza dividiéndolo por el sumatoria de todos los exponenciales del vector.

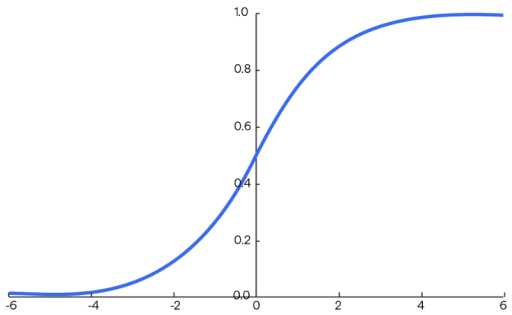

Figura 17: Función SoftMax. Fuente: https://botpenguin.com/glossary/softmax-function

5.3. Algoritmos de Optimización

- Stochastic Gradient Descent with Momentum, SGDM: Su objetivo es encontrar el valor mínimo de una función de costos determinada. La idea principal detrás del descenso de gradiente estocástico es minimizar la función de costo al ajustar los parámetros del modelo de manera iterativa en función de la retroalimentación que se obtiene del conjunto de datos de entrenamiento. En lugar de actualizar los parámetros del modelo con todos los datos de entrenamiento en cada iteración (como ocurre en el descenso de gradiente regular), el descenso de gradiente estocástico utiliza pequeñas muestras aleatorias de los datos de entrenamiento (conocidas como "mini-batch") en cada iteración para actualizar los parámetros del modelo.

- Propagación cuadrática media, RMSProp: Es un algoritmo de optimización de la tasa de aprendizaje adaptativo. De hecho, es una extensión del descenso de gradiente y del popular algoritmo AdaGrad y está diseñado para reducir drásticamente la cantidad de esfuerzo computacional utilizado en el entrenamiento de redes neuronales. El algoritmo funciona reduciendo exponencialmente la tasa de aprendizaje cada vez que el gradiente al cuadrado es inferior a un determinado umbral.

- Adaptive Moment Estimation, Adam: Su función principal es ajustar de manera iterativa los pesos de la red en función de los datos de entrenamiento, con el objetivo de minimizar la función de pérdida. Adam combina dos técnicas de optimización, la capacidad de AdaGrad para manejar gradientes dispersos y la capacidad de RMSProp para manejar objetivos no estacionarios. Utiliza una media móvil exponencial de los gradientes para ajustar las tasas de aprendizaje. Además, para calcular los ajustes de los pesos durante el entrenamiento, utiliza el algoritmo backpropagation antes mencionado.

- Adadelta: La optimización Adadelta es una técnica de descenso de gradiente estocástico que ajusta la tasa de aprendizaje por dimensión, resolviendo dos problemas clave: la disminución continua de las tasas de aprendizaje y la necesidad de seleccionar manualmente una tasa global. Mejora sobre Adagrad al utilizar una ventana móvil de actualizaciones de gradientes en lugar de acumular todos los gradientes previos. Esto permite que el aprendizaje continúe siendo efectivo incluso después de muchas actualizaciones. Adadelta no requiere una tasa de aprendizaje inicial predefinida, lo que simplifica su configuración y lo hace más robusto y adaptable en diversas aplicaciones de aprendizaje automático.

5.4. DNN

Hemos aplicado redes neuronales profundas con tres objetivos: predecir el resultado de un partido con la probabilidad de victoria local, empate y victoria visitante, predecir el número de goles de cada equipo y predecir si marcarán o no marcarán ambos.

Red neuronal para 1X2

En las capas ocultas de la red hemos utilizado la función ReLU, que es una función de activación comúnmente utilizada en las capas ocultas de una red neuronal debido a su capacidad para mitigar el problema de la desaparición del gradiente y acelerar el proceso de convergencia durante el entrenamiento.

Para este modelo, dado que es una clasificación multiclase y queremos la probabilidad de que se dé cada escenario, utilizaremos la función de activación SoftMax para la capa de salida. Utilizamos SoftMax en vez de sigmoide porque la sigmoide tiene un problema. La función sigmoide es monótona, pero la derivada de la función no lo es. Ocasionalmente podemos encontrarnos con que este tipo de funciones de activación, llegado a un determinado porcentaje de precisión y con muchas clases, en el entrenamiento se trabe y no mejore. En este sentido, la función SoftMax es una función de activación logística más generalizada que se utiliza para la clasificación multiclase. La SoftMax tiene forma sigmoidea y clasifica entre valores de 0 y 1 (probabilidad), pero a diferencia de la sigmoide o de la tanh, es muy diferenciable y no empuja la salida hacia los extremos 0 y 1.

Las capas dropout las utilizamos para evitar el sobreajuste. La regularización con dropout aplicada después de cada capa densa ayuda a prevenir el sobreajuste al .ªpagar.ªleatoriamente algunas neuronas durante el entrenamiento, lo que obliga al modelo a aprender representaciones más robustas y generalizables.

En cuanto al optimizador usado, Adam, es una elección sólida como optimizador debido a su eficacia y adaptabilidad en una variedad de problemas de optimización. Es particularmente útil en problemas con grandes conjuntos de datos y/o espacios de parámetros de alta dimensión, ya que ajusta dinámicamente la tasa de aprendizaje para cada parámetro de la red. Esto puede ayudar a acelerar el proceso de convergencia y mejorar el rendimiento del modelo.

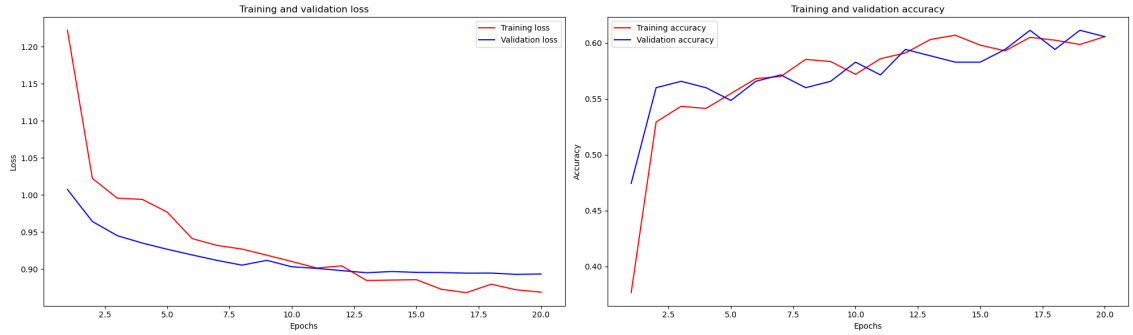

Figura 18: Curva de aprendizaje para el modelo 1X2

La curva de aprendizaje proporciona una visión detallada del rendimiento del modelo durante el proceso de entrenamiento y validación. Sirve como herramienta para identificar problemas de ajuste, como el sobreajuste o subajuste, al observar la evolución de la pérdida y la métrica de

rendimiento a lo largo de las épocas.

Idealmente, la curva de pérdida debe disminuir continuamente en ambos conjuntos a medida que avanza el entrenamiento. Con pérdida nos referimos a la función de pérdida, medida de la discrepancia entre las predicciones del modelo y los valores reales. El objetivo principal es minimizar esta pérdida. La curva de precisión ideal mostraría un aumento constante en la precisión del modelo, con una estabilización eventual que indica que el modelo ha alcanzado su capacidad máxima de aprendizaje y la diferencia entre la precisión en el conjunto de entrenamiento y en el conjunto de validación debería ser pequeña.

Red neuronal Marcan ambos o no

En este modelo de red neuronal, se utiliza la función de activación ReLU en las capas ocultas para introducir no linealidad y facilitar el aprendizaje de características complejas, y la función de activación sigmoide en la capa de salida dado que la salida es binaria.

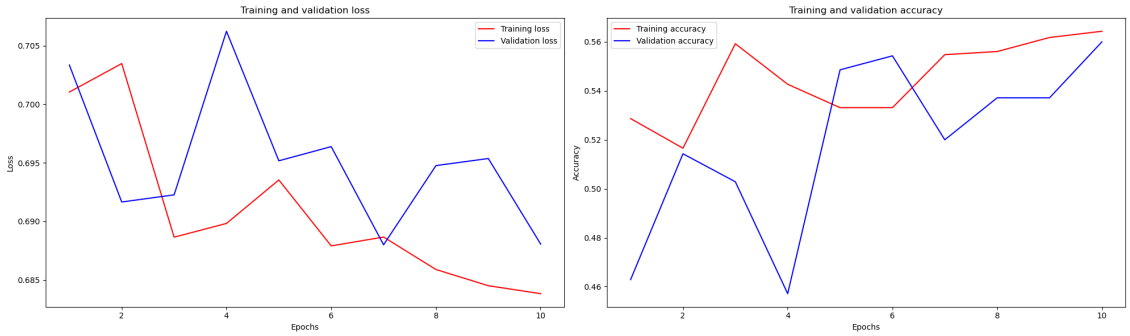

Figura 19: Curva de aprendizaje para el modelo marcan ambos

Como vemos, estas curvas son menos suaves. El valor de pérdida y de precisión para estos dos modelos de redes neuronales de clasificación tiene estos valores.

Modelo	Loss	Accuracy
1X2	0.83	0.65
Marcan ambos	0.69	0.56

Red neuronal Goles

Al aplicar ReLU como función de activación en las capas permite que la red aprenda características no lineales de los datos de entrada, lo que puede mejorar la capacidad de predicción del modelo. Tras probar varios tipos de funciones de activación, la que mejor se ajusta es ReLU. Aquí, nuevamente hemos utilizado el optimizador Adam y capas dropout para la regularización.

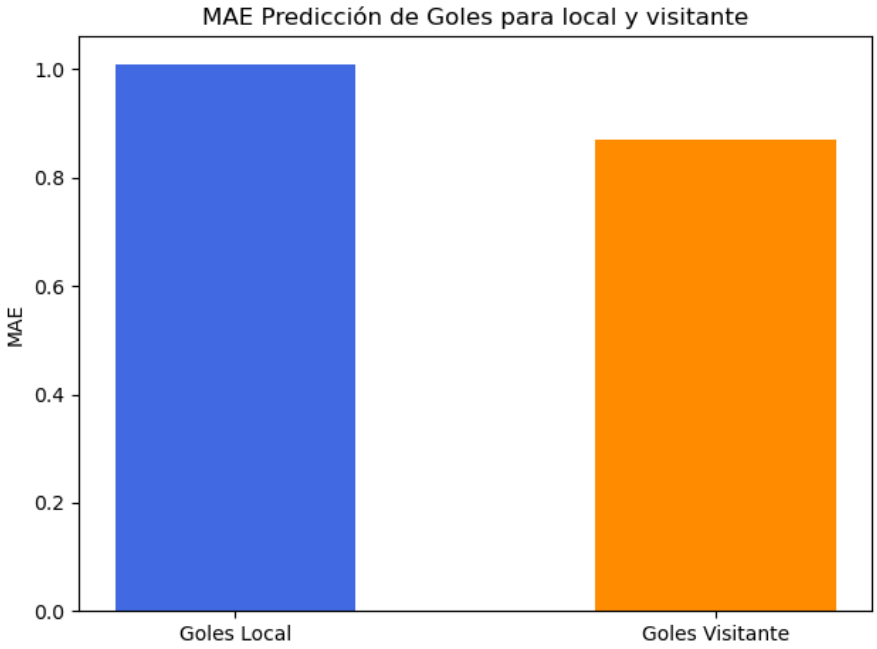

Figura 20: MAE para los goles local y goles visitante

5.5. CNN

Además de trabajar con redes neuronales densamente conectadas (DNN), también utilicé redes neuronales convolucionales (CNN) para procesar imágenes de los escudos de fútbol. Las CNN son especialmente poderosas en tareas de visión por computadora debido a su capacidad para capturar patrones espaciales locales como bordes, texturas y formas en diferentes regiones de la imagen.

Si bien puede usar modelos de aprendizaje profundo para cualquier tipo de aprendizaje automático, resultan especialmente útiles para trabajar con datos que constan de grandes matrices de valores numéricos, como imágenes. Los modelos de aprendizaje automático que funcionan con imágenes son la base de una inteligencia artificial de área denominada Computer Vision. El motivo del éxito del aprendizaje profundo en esta área es un tipo de modelo denominado red neuronal convolucional, CNN. Por lo general, una CNN funciona extrayendo características de las imágenes y enviándolas a una red neuronal completamente conectada para generar una predicción. Las capas de extracción de características de la red reducen la cantidad de características en la matriz potencialmente enorme de valores de píxeles individuales a un conjunto más pequeño de características que ayuda a la predicción de etiquetas.

Las CNN constan de varias capas, cada una de las cuales realiza una tarea específica en la extracción de características o la predicción de etiquetas. En general, las CNN están compuestas por capas convolucionales, de agrupación (pooling) y completamente conectadas. Las capas convolucionales aplican filtros a las imágenes de entrada para extraer características relevantes, mientras que las capas de agrupación reducen la dimensionalidad de las características conservando la información más importante. Finalmente, las capas completamente conectadas clasifican las características extraídas para realizar predicciones.

Modelo	**Loss**	**Accuracy**
CNN	0.0007	1.0

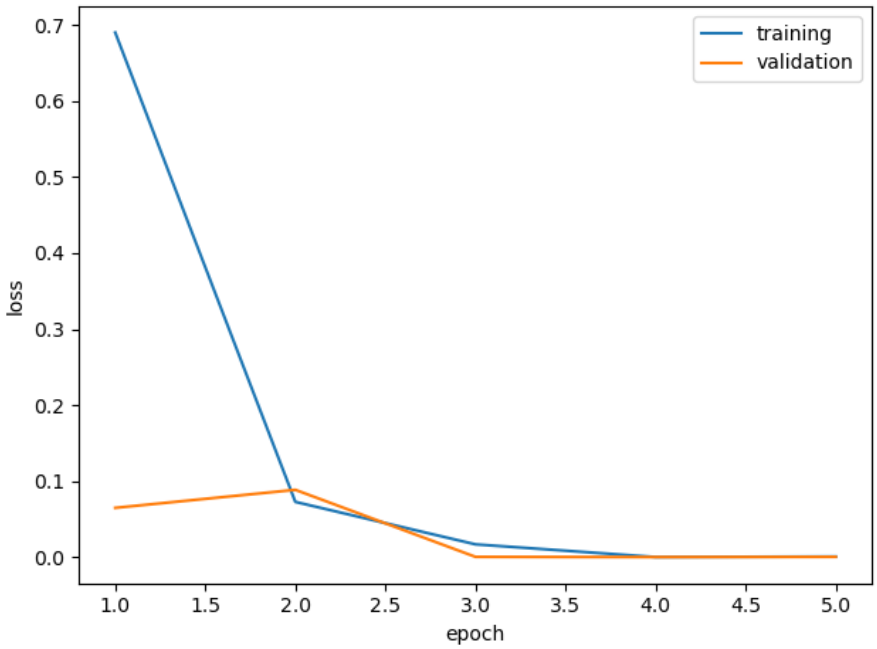

Figura 21: Curva de aprendizaje para el modelo CNN

5.6. Aprendizaje por Transferencia

Transferencia de aprendizaje es un enfoque en el aprendizaje automático donde se aprovechan los conocimientos adquiridos por un modelo previamente entrenado en una tarea específica para mejorar el rendimiento en una tarea relacionada pero diferente. En lugar de entrenar un modelo desde cero, se utiliza un modelo pre-entrenado como punto de partida y se ajusta o se extiende para adaptarse al nuevo problema.

En mi caso, he utilizado la transferencia de aprendizaje para entrenar un modelo de clasificación de imágenes de escudos de equipos deportivos al igual que CNN pero partiendo de un modelo base. Inicialmente, cargamos un modelo base pre-entrenado, en este caso, he utilizado la arquitectura ResNet50 que ha sido entrenada en el conjunto de datos ImageNet. Esta arquitectura es conocida por su efectividad en tareas de visión artificial y puede capturar características útiles en imágenes. He preparado los datos de imágenes de escudos de equipos para el entrenamiento, utilizando un generador de imágenes que escala los valores de los píxeles y divide los datos en conjuntos de entrenamiento y validación. Amplío el modelo base añadiendo una capa de clasificación en la parte superior y congelo las capas del modelo base. Luego, compilo el modelo con una función de pérdida y un optimizador específico, Adam, y lo entreno utilizando datos preparados durante un número definido de épocas. Este enfoque de transferencia de aprendizaje me permite adaptar eficazmente un modelo pre-entrenado, como ResNet50, a mi problema específico de clasificación de escudos de equipos deportivos, lo que puede acelerar el entrenamiento y mejorar el rendimiento del modelo.

Modelo	Loss	Accuracy
Aprendizaje por Transferencia	0.17	0.93

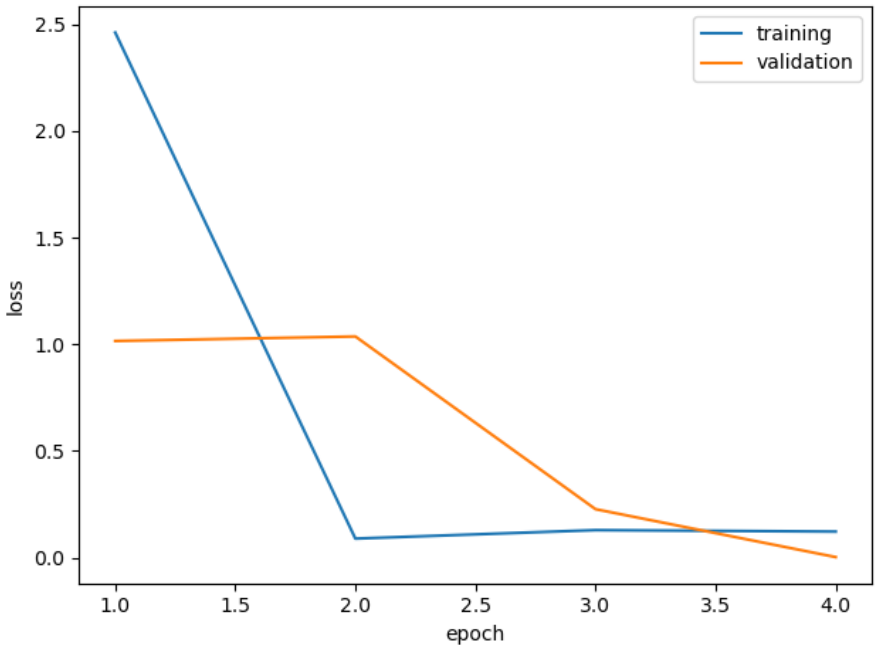

Figura 22: Curva de aprendizaje para el modelo Aprendizaje por Transferencia

6. Aprendizaje por Refuerzo

El aprendizaje por refuerzo es un paradigma dentro del campo del aprendizaje automático donde un agente interactúa con un entorno con el objetivo de aprender a tomar decisiones que maximicen alguna noción de recompensa acumulada a lo largo del tiempo. En este proceso, el agente toma decisiones secuenciales, observa el resultado de esas decisiones y recibe retroalimentación sobre la calidad de sus acciones en forma de recompensas o castigos.

En el contexto del aprendizaje por refuerzo, se utilizan modelos matemáticos llamados Procesos de Decisión de Markov (MDP). Estos modelos representan situaciones de toma de decisiones secuenciales en entornos estocásticos. En un MDP, un agente toma decisiones en un entorno donde las transiciones entre estados son probabilísticas y están influenciadas por las acciones del agente. Cada estado en el MDP representa una situación en la que se encuentra el agente, y las acciones que el agente puede tomar tienen efectos probabilísticos en el siguiente estado del entorno. Los MDPs proporcionan un marco formal para modelar problemas de toma de decisiones secuenciales y permiten a los algoritmos de aprendizaje por refuerzo encontrar políticas óptimas que maximicen la recompensa acumulada a lo largo del tiempo.

Los elementos esenciales del aprendizaje por refuerzo incluyen el agente, que es la entidad que aprende y toma decisiones basadas en su interacción con el entorno, y el entorno, que representa el mundo en el que opera el agente y en el que ocurren las acciones y se reciben las recompensas. El estado describe la situación actual del entorno en un momento dado y puede influir en las decisiones del agente, mientras que las acciones son las decisiones que el agente puede tomar en un estado determinado del entorno.

La recompensa es la señal de retroalimentación que el agente recibe del entorno después de realizar una acción. Indica cuán buena o mala fue esa acción en términos de alcanzar los objetivos del agente. La política es la estrategia que guía al agente para tomar decisiones y puede ser determinista o estocástica. En una política determinista, la acción que elige el agente en un estado dado está completamente definida y es única. Esto significa que, dadas las mismas condiciones o estado del entorno, el agente siempre tomará la misma acción. Por otro lado, en una política estocástica, la acción que elige el agente en un estado dado se determina mediante una distribución de probabilidad. Esto significa que, en lugar de una acción única y definida, el agente elige una acción de acuerdo con una distribución de probabilidad sobre todas las posibles acciones. Además, el equilibrio entre la exploración y la explotación es crucial en el aprendizaje por refuerzo, ya que el agente debe probar nuevas acciones para descubrir mejores estrategias (exploración) al tiempo

que aprovecha las acciones conocidas que han resultado exitosas en el pasado (explotación).

6.1. Cadenas de Markov

Una cadena de Markov es un proceso evolutivo que consiste en un número finito de estados en el cual la probabilidad de que ocurra un evento depende únicamente del evento inmediatamente anterior, con unas probabilidades que están fijas. He creado un algoritmo para generar matrices de transición para los 4 equipos de la semifinal de la Champions 2023/2024, considerando los estados de ganar, empatar y perder. La idea detrás de esto es determinar la probabilidad que tiene un equipo de ganar, perder o empatar en función únicamente del resultado del último partido jugado. Esta metodología cobra sentido dado que, en el contexto de eventos secuenciales como los partidos de fútbol, los equipos pueden experimentar cambios en su rendimiento, motivación y estrategias basándose en sus resultados previos. Por lo tanto, analizar la probabilidad condicional de resultados futuros en función del último resultado puede proporcionar información valiosa sobre el estado actual y las tendencias de los equipos en la competición.

$$\begin{pmatrix} 0{,}0 & 1{,}0 & 0{,}0 \\ 0{,}16 & 0{,}33 & 0{,}5 \\ 0{,}2 & 0{,}4 & 0{,}4 \end{pmatrix}$$

Figura 23: Matriz de transción Real Madrid

Por ejemplo, mirando la matriz podemos saber que el Real Madrid tendrá un 20 % de probabilidad de que gane el próximo partido habiendo perdido el último partido jugado.

7. Predicción

Para la predicción de los resultados de un partido futuro, he creado una función donde al usuario se le dará una lista de los equipos con sus IDs y podrá ingresar los equipos que quiere enfrentar. Esto proporcionará la probabilidad de que gane el local, de que empate o de que gane el visitante, los goles y si marcan o no ambos. Las predicciones se realizan con las redes neuronales creadas para el resultado de partidos, goles y si marcan o no ambos. Un ejemplo para la predicción del partido entre Bayern de Múnich y Real Madrid sería:

Figura 24: Resultado partido

Cabe destacar que el resultado tiene una precisión del 65 %, los goles pueden variar 1 gol para el local y 0.87 para el visitante, y que el resultado de si marcan o no ambos tiene una precisión del 56 %.

Al predecir resultados de partidos, podemos prever quién pasará de fase. Cuando se trata de predecir los resultados de los partidos y, por consiguiente, determinar qué equipo avanzará a la siguiente fase, se debe considerar el formato de ïda y vueltaçomúnmente utilizado en muchas competiciones, excepto en la final. En este formato, cada equipo juega un partido en su propio campo y otro en el campo del equipo contrario. Para calcular las probabilidades de que cada equipo avance, se evalúan las probabilidades de victoria para ambos equipos en sus respectivos encuentros como local y como visitante. Una vez obtenidas estas probabilidades para cada escenario, se calcula un promedio de las probabilidades de victoria para cada equipo en ambos partidos. El equipo con la media más alta, es decir, el que tiene una probabilidad más alta de ganar en ambos partidos, es el que se considera más probable que avance a la siguiente fase. Este enfoque permite tomar decisiones informadas sobre qué equipo se espera que pase de fase, basadas en análisis estadísticos y probabilísticos de los resultados posibles en los encuentros de ida y vuelta.

Este proceso lo hemos realizado con las parejas de cuartos, a partir de estos resultados hicimos lo mismo con la semifinal y después con la final para predecir quién será el ganador de la UEFA Champions League. Cabe destacar que para el siguiente cuadro se realizará con los datos actualizados después de la fase de octavos. Actualmente, tenemos los datos actualizados hasta la semifinal, incluida, que es la fase que acaba de jugarse. Posteriormente, procederemos a realizar la predicción una vez se hayan disputado los partidos de cuartos de final y se conozcan los equipos clasificados para las semifinales, utilizando los datos actualizados de esta etapa de la competición.

Figura 25: Resultado a partir de cuartos para la UEFA Champions League

Durante la fase de cuartos de final, nuestro análisis indicaba que el Manchester City sería el equipo ganador. Sin embargo, dado que estábamos realizando este trabajo de predicción simultáneamente con el desarrollo de la competición, pudimos comprobar nuestras predicciones con los resultados reales. Resultó que nuestras predicciones habían fallado para la pareja Real Madrid-Manchester City y para Atlético de Madrid-Borussia Dortmund, ya que tanto el Borussia como el Real Madrid lograron avanzar a semifinales. Una vez jugada la fase de cuartos, actualizamos los datos y seguimos el mismo procedimiento para predecir el ganador, y nuestro resultado fue este.

Figura 26: Resultado a partir de semifinales para la UEFA Champions League

Vemos cómo nuestro modelo de predicción indicaba que el Paris Saint-Germain (PSG) sería el

ganador de la UEFA Champions League. Sin embargo, ahora que la semifinal ya se ha jugado y solo queda la final por disputarse, hemos observado que nuestro modelo ha fallado al predecir qué equipo avanzaría a la final. En lugar del PSG, el Borussia Dortmund ha asegurado su lugar en la final. Podemos utilizar nuestro modelo para predecir el resultado de dicho encuentro; actualmente, mi base de datos está actualizada con los datos de la semifinal.

Figura 27: Resultado final UEFA Champions League

Referencias

[1] *¿Qué es el aprendizaje supervisado? — IBM.* (n.d.). https://www.ibm.com/es-es/topics/supervised-learning

[2] *Regresión lasso — Interactive Chaos.* (n.d.). https://interactivechaos.com/es/manual/tutorial-de-machine-learning/regresion-lasso

[3] *IBM Cognos Analytics 11.1.x.* (2024, February 29). https://www.ibm.com/docs/es/cognos-analytics/11.1.0?topic=tests-regression-tree

[4] E, T. G. (2022, May 21). Una guía práctica para implementar un Random Forest Classifier en Python. *Medium.* https://tutegomeze.medium.com/una-gu%C3%ADa-pr%C3%A1ctica-para-implementar-un-random-forest-classifier-en-python-5e38c290ae03

[5] *¿Qué es el aprendizaje no supervisado? — IBM.* (n.d.). https://www.ibm.com/es-es/topics/unsupervised-learning

[6] Keytrendsadmin. (2024, April 18). *Clustering - KeyTrends.* KeyTrends. https://keytrends.ai/es/academy/glosario/inteligencia-artificial/clustering

[7] *¿Qué es el aprendizaje profundo? - Explicación del aprendizaje profundo - AWS.* (n.d.). https://aws.amazon.com/es/what-is/deep-learning/

[8] Javi. (2023, March 24). *Función SoftMax: Activación para la clasificación.* Jacar. https://jacar.es/funcion-softmax-activacion-para-la-clasificacion/

[9] Franco, F. (2024, March 30). What is RMSprop? (with code!) - The Deep Hub - Medium. *Medium.* https://medium.com/thedeephub/what-is-rmsprop-0f54effc47e4

[10] Team, K. (n.d.). Keras documentation: Adadelta. https://keras.io/api/optimizers/adadelta/

[11] Del Barrio González, D. (2022). Aplicación del aprendizaje automático en modelos de materia activa (Trabajo de Fin de Grado, Universidad Politécnica de Madrid, España). https://oa.upm.es/70193/3/TFG_Diego_del_Barrio_Gonzalez.pdf

[12] Durán, J. (2021, December 13). Qué es la Transferencia de Aprendizaje y Cómo Aplicarla a tu Red Neuronal. *Medium.* https://medium.com/metadatos/qu%C3%A9-es-la-transferencia-de-aprendizaje-y-c%C3%B3mo-aplicarla-a-tu-red-neuronal-e0e120156e40