Prediciendo el Campeón de la Copa América 2024 usando un modelo simple con Python

Sebastian Gaviria Giraldo and Jose Luis Torres Lopez
(Universidad de Antioquia)
(Dated: August 6, 2024)

Este artículo presenta el estudio de un código diseñado para la recolección, limpieza y uso de datos históricos de la Copa América. El código integrado emplea técnicas de web scraping para extraer información detallada sobre los partidos de diferentes ediciones del torneo en wikipedia. El código no solo prepara los datos, sino que también incluye métodos que construyen un modelo simple para predecir el campeón de la Copa América 2024 utilizando la distribución de Poisson y asociando esta con el concepto de gol en este modelo simplificado.

This article presents the study of a code designed for collecting, cleaning, and using historical data from the Copa América. The integrated code employs web scraping techniques to extract detailed information about matches from different editions of the tournament on Wikipedia. The code not only prepares the data but also includes methods for building a simple model to predict the champion of Copa América 2024, using the Poisson distribution and associating it with the concept of goals in this simplified model.

Palabras clave: Web Scraping, Copa América, Limpieza de datos, Modelos Probabilísticos, Recolección de datos.

I. INTRODUCCION

Al inicio del planteamiento de el problema a resolver en el proyecto final, vimos como primera opción real de proyecto una aplicación del análisis de datos a la coyuntura de las VBG. Sin embargo, en la búsqueda de la información de bases de datos relacionados a este tema, nos dimos cuenta que no hay muchas bases de datos estandarizadas que podamos manejar adecuadamente, o que estén disponibles de forma accesible para su tratamiento y análisis. Esta situación nos hizo ver que, en algunos casos, la obtención y recolección de datos para proyectos pequeños puede ser, a su vez, un gran problema a tratar y dificil de entender mejor. Este nuevo planteamiento nos hizo descubrir, investigando un poco, una técnica de acceso a páginas web por medio de un script, comúnmente llamada WebScraping. En python, el WebScraping con BeautifulSoup y Selenium fueron librerías suficientes para aproximarnos a "construir" una pequeña base de datos históricos de la Copa América. En algún momento, cuando practicamos la técnica de WebScraping, por esas fechas se estaba jugando precisamente esta competición, la cual es una de las principales competencias futbolísticas entre las selecciones nacionales de América del Sur y conocida por ser el torneo mas antiguo del mundo.

Con todo esto, surgió una idea de lograr predecir el campeón del torneo utilizando datos de Wikipedia. Aunque al principio teníamos ideas de un modelo más complejo y completo, para que el campeón no fuese tan predecible y el modelo no fuera tan simple, tuvimos varios problemas a la hora de una recopilación de datos más completa y diversa para poder aplicar incluso técnicas de Machinne Learning. Al final, nuestro modelo fue mucho más simple, y utilizando conceptos estadísticos y de interpretación del reparto de puntos en el fútbol, logramos implementar un sencillo predictor del campeón del torneo. Cabe recalcar que, la edición de 2024 es la primera edición en la que participó la selección de Canadá, por lo tanto, para este equipo tenemos cero datos, y ya hay un pequeño sesgo en la predicción.

II. METODOS

Se puede dividir el trabajo del proyecto en tres grandes partes: La recolección de datos usando técnicas de WebScraping; la organización, limpieza y disposición de los datos recolectados; la construcción del modelo predictor del ganador, utilizando los datos limpios; y la modularización de estos tres procesos anteriores en clases que cohesionen todo el código. Para llevar a cabo todas estas tareas, al final se lograron construir cinco clases que serán explicadas en detalle a continuación:

Clase DataGetter: Se encarga de extraer información desde las páginas web de wikipedia donde se encuentran los partidos de las ediciones de Copa América a lo largo de la historia. Esta clase tiene como atributo el link (url) de la página de Wikipedia de la Copa América 2024. Sus métodos se encargan de leer el contenido html y organizarlo en un diccionario cuyo contenido son las cuatro tablas de posiciones de la Copa América 2024. Aunque una vez cargadas estas tablas, salen actualizadas con las fechas ya pasadas, posteriormente se transforman para adaptarlas a la predicción. Al tener los datos organizados en DataFrame's, a través de las tablas extraídas de la página web, podemos tener un acceso organizado a diferentes secciones de datos y

mejorar el tratamiento a nivel computacional.

Clase DataCollector: Hasta este momento del proceso, no se ha utilizado WebScraping. Pues esta es precisamente la clase encargada de utilizar esta técnica, para la extracción del total de datos históricos de todas las ediciones de la Copa América. Los métodos de esta clase se encargan de una extracción de datos en dos fases: La primera utilizando la librería bs4 y su paquete BeautifulSoup, y la segunda utilizando la librería Selenium y un webdriver para extraer los datos que no pudieron ser extraídos con bs4. Estos años de "datos perdidos" fueron revisados y se comprobó que fueron los años de las ediciones 2011 y 2015. Además de la recolección de datos utilizando WebScraping, los métodos de esta clase se encargan de recopilar el histórico de todos los datos, utilizando un patrón encontrado en el nombre de las páginas de Wikipedia (en el valor del año en los links).

Clase DataCleanner: Esta clase se encarga de leer la data total recolectada por la clase anteriormente descrita. Además los métodos de esta clase, limpia en dos partes. Por un lado, limpia, organiza, renombra algunas columnas necesarias y deja listo todo el fixture de la Copa América 2024. Con esto, a pesar que los datos de Wikipedia están actualizados y salen los marcadores de todos los partidos de la competencia, al organizar el fixture, obtenemos únicamente los datos de los partidos, sin los resultados y es como si no si hubiese jugado la copa todavía. Por otro lado, limpia, organiza, renombra y une toda la data histórica, incluyendo lo recolectado con bs4 y con Selenium. Además, separa en DataFrame's correspondientes al equipo Local y Visitante, y los goles anotados por estos equipos a lo largo de todas las ediciones de la Copa América comprendidas entre 1916 y 2019.

Clase Modeling: En esta clase encontramos una parte crucial del proyecto y de todo el código. Esta es la clase encargada de utilizar los datos recolectados con WebScraping y limpiados, para aplicar el modelo que predice el ganador de la competencia actual (Copa América 2024). Esta clase tiene dos atributos: Uno es el diccionario que contiene las cuatro tablas de posiciones de la competición actual (2024), el otro es el DataFrame que contiene la fuerza de los equipos. Este concepto de fuerza de los equipos, simplemente es un promediado histórico de todos los goles marcados y recibidos por cada equipo. En este caso, Canadá no registra fuerza de equipo, pues no hay datos de este país. Esto ya muestra una limitación del modelo, pues hay un equipo con probabilidad de ganar el torneo igual a cero.

Los métodos de esta función merecen un entendimiento mayor. Comenzando por el método *Rounds()* se encarga de organizar los partidos en las diferentes fases del torneo, obteniendo un DataFrame para la fase de grupos, cuartos de final, semi-final, y final, respectivamente. Por su parte, tenemos el método *predictPoints()* el cual utiliza un modelo probabilístico de tipo Poissson para predecir los puntos que podrían obtener dos equipos en un partido, este modelo se basa en sus estadísticas históricas particulares y calcula las probabilidades asociadas a los posibles resultados del partido retornando una tupla con los puntos esperados para ambos equipos. Esta distribución fue escogida, ya que es una distribución discreta que describe el número de eventos en un intervalo de tiempo fijo. Los partidos tienen una duración fija de tiempo, por ende, el gol puede ser un evento probabilístico que puede ocurrir en un tiempo fijo de 90 minutos. Además, el número de goles en un partido se puede contar; luego, la ocurrencia de los goles en un partido pueden aproximarse a ser independientes en un modelo simple, aunque en un modelo más completo este tema se podría variar teniendo en cuenta que en la realidad la ocurrencia de goles no siempre es independiente. Por otro lado, la tasa a la que se anotan los goles en un partido es constante, es decir, la probabilidad de que ocurra un gol en un partido de 90 minutos, es la misma de que ocurra en otro partido de 90 minutos. Y, por último, dos goles en un partido no pueden ocurrir en el mismo instante de tiempo. Con todo esto en cuenta, podemos ver que el gol es perfectamente aproximable a un evento tratable con la distribución de Poisson. Con estos conceptos en mente, podemos construir el promedio de goles de los equipos para construir el parámetro λ necesario en la implementación de la función de distribución de Poisson que existe en Python. Estos lambdas se calculan teniendo en cuenta el promedio de goles anotados por el equipo local y el promedio de goles recibidos por el equipo visitante. Una vez calculados los parámetros λ se procede a calcular la probabilidad de equipos de marcar desde 0 hasta 7 goles, y con ello ir sumando a los puntos de cada equipo.

Por su parte, la clase *getWinner()* se encarga de seleccionar el ganador de los partidos en una fase del torneo. Asignado valores en la columna Winner para los DataFrame elegidos. El método *uptadeTable()* actualiza la fase siguiente del torneo, teniendo en cuenta los ganadores de la fase anterior y dando avance a las fases de la competencia. Por último, está el método *simulateTournament()*. Este método se encarga de, primero que todo, simular toda la fase de grupos de la Copa América para cada tabla de posiciones, y actualiza los puntos y las posiciones de los grupos. A partir de esta actualización de posiciones y puntos, se extraen los dos equipos que pasan a la siguiente ronda del torneo por cada grupo. Con esto, se reemplazan los valores de los equipos que jugarán los cuartos de final, con los equipos seleccionados una vez simulado la fase de grupos. Después de actualizar los datos de los cuartos de final, se crea la columna winner y se actualiza la fase del torneo, repitiendo estos pasos hasta la final del torneo, y finalmente se obtiene al ganador del partido de la final de la competencia. Finalizando con esto a la incógnita y respondiendo a la pregunta de ¿Qué equipo es el ganador del torneo según la predicción de nuestro modelo simple?

III. RESULTADOS

El proceso de limpieza de datos permitió corregir errores en los nombres de los equipos y estandarizar los formatos de los resultados. Se eliminaron filas y columnas irrelevantes, y se organizaron los datos en DataFrames que facilitan el análisis y la comparación entre diferentes ediciones del torneo. El análisis de los datos reveló inexactitudes en la presentación de esta data, y uno de los resultados del proyecto fue encontrar formas de dejar lo más limpia posible todos los datos de goles históricos de los equipos de la Copa América 2024. En este orden de ideas, podemos decir que los archivos **ProgramacionCopaAmerica2024.csv**, **Conmebol-CopaAmericacompletedata.csv** son resultados de nuestro proyecto. Pues es una data con más de 700 partidos, totalmente organizada con python y adaptada a la resolución del problema. También las tablas de posiciones organizadas por grupo pueden ser vistas como resultados del proyecto.

También hay que mencionar que el sencillo modelo predictivo estimó los puntos esperados para los equipos en hipotéticos partidos basándose en sus estadísticas históricas. Los resultados sugieren que los equipos con altos índices de goles anotados y bajos índices de goles concedidos tienen una mayor probabilidad de ganar. Por lo tanto, otro gran resultado del proyecto son los DataFrames actualizados con las fases de cuartos de final hasta la gran final del torneo simuladas y actualizadas, dando al ganador del torneo y resumidas en la imagen que se ve al final de la página. El resultado de la simulación da como campeón del torneo a la selección de Argentina.

	home	score	away	year	Winner	Fase
0	Argentina	Match 25	Ecuador	2024	Argentina	Cuartos de Final
1	Mexico	Match 26	Peru	2024	Mexico	Cuartos de Final
2	Brazil	Match 27	United States	2024	Brazil	Cuartos de Final
3	Uruguay	Match 28	Paraguay	2024	Uruguay	Cuartos de Final
4	Argentina	Match 29	Mexico	2024	Argentina	Semi-Final
5	Brazil	Match 30	Uruguay	2024	Brazil	Semi-Final
6	Argentina	Match 32	Brazil	2024	Argentina	Final

FIG. 1. Se muestra el DataFrame que evidencia la simulación del Torneo y el ganador.

IV. CONCLUSIONES

A pesar de que no pudimos complejizar el modelo predictivo tanto como lo quisimos y planteamos en un momento, pudimos llegar a varias conclusiones importantes que servirán para dar una perspectiva a futuro del proyecto. Primero, aprendimos que utilizando técnicas medianamente avanzadas de recolección de datos como el WebScraping se pueden construir bases de datos pequeñas pero útiles para aplicar a modelos sencillos de predicción y análisis de datos. Una reflexión que hacemos es que queda pendiente perfeccionar la técnica para poder extraer información de páginas web con una arquitectura html mucho más compleja que no necesariamente está soportada por las librerías que utilizamos en nuestro proyecto. La organización y limpieza de estos datos a través de bibliotecas como pandas garantizó una observación más precisa de los datos. Por ende, para una futura mejora del modelo, podemos obtener miles de datos con mayor heterogeneidad y diversidad si logramos realizar un webScraping más avanzado.

En el desarrollo del proyecto, tratamos de complejizar mucho más el modelo, tratando de añadir datos del ranking mundial de la FIFA, jugadores destacados de la FIFA, últimos partidos de las selecciones, estado de forma de las selecciones, entre otros muchos más datos que reflejaran el estado deportivo de los equipos y con ello la predicción de los puntos de cada equipo. Lo pensado era, que una vez obtuviéramos y limpiaramos todos estos datos, utilizaramos librerías y modelos de Machinne Learning como RandomForest o Gradient Boosting para poder hacer predicciones mucho más completas y cercanas a la realidad. Sin embargo, hubo un fallo y falta de tiempo a la hora de recolectar esta cantidad de datos tan diversa. Por lo tanto, para construir un modelo mucho más complejo, haría falta una mayor aplicación de los modelos de Deep y Machinne Learning acompañado de un análisis estadístico.

A pesar de que este proyecto no tiene una relación directa con problemas de físcia, su desarrollo nos sirvió para entender mucho mejor le trabajo detrás de la recolección y construcción de bases de datos y su posterior uso en modelos de predicción de eventos. Estas habilidades sí tienen una relación directa con la resolución de algunos problemas de física.

[1] M. E. Rogelio, *Desarrollo de un sistema de Web Scraping para la obtención de datos en entornos Big Data* (Tecnologico Nacional de Mexico, 2023).

[2] T. H. Alexis, B. R. Cesar, M. G. Jorge, and M. C. Adrian, *Metodologías para análisis utilizando Web Scraping* (Universidad Politécnica de Chiapas, 2015).

[3] B. R. José, *Investigación y Desarrollo de Técnicas de Scraping* (Universidad de Alcalá, 2019).

[4] M. R. Alicia, E. E. Hugo, and H. P. Yasmín, *Uso de técnicas de Web Scraping para obtención automática de bases de datos en la Web* (Tecnológico Nacional de México, 2022).

[5] Selenium, *WebDriver Documentation.* (Selenium Developers, 2024).

[6] R. Kenneth., *Requests: HTTP for Humans.* (GitHub: https://github.com/psf/requests, 2024).

[7] L. Richardson, *Beautiful Soup Documentation.* (Source: https://www.crummy.com/software/BeautifulSoup/bs4/doc/, 2024).