March Madness Mania 2024

Raül Y. Dalgamonni Alonso

Resum— Este documento es un Trabajo Final de Grado (TFG) que se enmarca en la competición March Machine Learning Mania 2024, organizada por Kaggle. El objetivo es desarrollar modelos de aprendizaje automático capaces de pronosticar los resultados de los torneos masculino y femenino de la División 1 de la NCAA. Se realiza un análisis exhaustivo de los datos históricos y se aplican técnicas de vanguardia en aprendizaje automático para construir un modelo robusto y preciso. Se siguió una metodología Kanban y se utilizó el lenguaje Python para la implementación de código. Se exploraron y prepararon los datos, y se implementaron varios modelos de Machine Learning, incluyendo XGBClassifier, Random Forest, Logistic Regression y Gaussian Naive Bayes. Finalmente, se realizaron predicciones para las categorías masculina y femenina.

Paraules clau— March Machine Learning Mania 2024, Kaggle, Aprendizaje computacional, Python, Regresión Logística, NCAA, Baloncesto, March Madness, Ganador nacional.

Abstract— This document is a comprehensive analysis of the March Machine Learning Mania 2024 competition, organized by Kaggle. The competition invites participants from around the world to develop machine learning models capable of predicting the results of the NCAA Division 1 men's and women's tournaments. The document outlines the objectives, methodology, and tools used in the project, as well as a detailed exploration of the data and the development of the prediction model. The project follows the Kanban methodology and uses Python for code implementation. The prediction model is evaluated using various metrics, and the results are discussed. The document concludes with the predictions for the tournament and a reflection on the project's achievement.

Index Terms— **Index Terms**— March Machine Learning Mania 2024, Kaggle, Machine Learning, Python, Logistic Regression, NCAA, Basketball, March Madness, National Championship

1 Introducción

EN el apasionante mundo del baloncesto universitario, el torneo del Campeonato Nacional de la NCAA, conocido como March Madness es sin duda el evento más esperado del año. Millones de aficionados alrededor de los Estados Unidos y del mundo siguen con fervor cada partido, vibrando con las victorias y lamentando las derrotas de sus equipos favoritos. Un torneo trepidante en el que se juegan hasta 68 partidos en 2 semanas [1].

Este hecho lo convierte en uno de los torneos, sino el que más, fascinantes de todo el año en el deporte. Pero ¿y si pudiéramos ir más allá de la mera emoción y predecir con cierta precisión los resultados de los encuentros? Es aquí donde entra en juego la competición March Machine Learning Mania 2024, organizada por la plataforma Kaggle.

Este desafío anual invita a participantes de todo el mundo a desarrollar modelos de aprendizaje automático capaces de pronosticar los resultados de los torneos masculino y femenino de la División 1 de la NCAA. Dicha tarea, tiene una complejidad característica, pues nunca nadie ha logrado acertar los 68 cruces [2], desde el 'First Four', hasta la gran final universitaria. A pesar de dicha complejidad, en el presente Trabajo Final de Grado (TFG) se enmarca en esta apasionante competición. A través de un análisis exhaustivo de datos históricos y la aplicación de técnicas de vanguardia en aprendizaje automático, se busca construir un modelo robusto y preciso que permita anticipar con mayor certeza el desenlace de los partidos.

2 Objetivos

El objetivo general de este TFG es realizar un modelo robusto que sea capaz de a partir de datos de partidos anteriores determinar quién será el ganador de un enfrentamiento específico, pudiendo generar un entregable, donde se indiquen todos y cada uno de los enfrentamientos y quien es el ganador. Dado el objetivo general, los objetivos específicos serán:

1. Realizar un análisis básico de los data sets que se tratarán que permita extraer conclusiones sobre el comportamiento de los datos.
2. Establecer e implementar una estrategia de, preprocesado y entrenamiento para uno o varios modelos, con la finalidad de poder tratar el data set que contiene datos de la temporada Regular.
3. Implementar un modelo usando técnicas de Machine Learning que sea capaz de tener un comportamiento robusto ante datos nunca vistos, como pueden ser los de los partidos de la NCAA.

3 Metodología

A lo largo del proyecto se seguirá una metodología Kanban. Esta metodología se centra en visualizar el trabajo en curso y la limitación del trabajo en proceso. Es útil para mejorar la productividad y evitar el sobre esfuerzo, es decir, de todo el proyecto, se intentará separar las tareas de tal modo que toda tarea hecha aporte un valor o sea considerada un entregable. Para ello se deberá definir un MVP, Minimal Viable Product. Algunas de las ventajas de Kanban [3] son:

- Ayuda a tener una visualización global del trabajo a realizar.
- Limita el trabajo en paralelo, por lo que aumenta la entrega de tareas y reduce el tiempo de espera.
- Limita el sobreesfuerzo basándose en MVP.

La elección de dicha metodología viene dada por el plazo de entrega del proyecto, de cara al concurso de Kaggle. Esto ayudará a que se consiga uno de los objetivos, un entregable para el concurso.

3.1 Herramientas de desarrollo

Teniendo en cuenta que el principal objetivo del proyecto es realizar un modelo que sea capaz de predecir el ganador de un partido, se ha decidido escoger lenguaje Python para elaborar todo el proceso de implementación de código. El IDE escogido ha sido Visual Studio Code. Este entorno de desarrollo integrado resulta una opción muy útil para trabajar con lenguaje Python. No solo porque es sencillo y fácil de utilizar, sino porque además incluye la opción de añadir extensión que facilitan la implementación de código. Esta opción destaca ante otros IDE como pueden ser Pycharm o Spyder debido a su operatividad y adaptabilidad. El hecho de que puedas añadir herramientas que ayudan a una generación de código eficiente y limpia, permite también facilitar la lectura y la detección de errores.

Los principales ficheros que son utilizados para realizar exploraciones sobre el conjunto de datos y demás serán ficheros Jupyter Notebook, no obstante, también se han utilizado ficheros Python a modo de aplicación configurable mediante un JSON. El principal motivo de uso de Jupyter Notebook es que este formato permite la ejecución de código por bloques (celdas). De tal forma, no es necesario ejecutar todo el código si se hace una pequeña modificación. Además, con tal de crear código de calidad, se ha establecido como compromiso implementar el código en formato PEP8.

3.2 Planificación

Con tal de tener una guía de cuáles serán los pasos a seguir del proyecto y determinar la duración estimada de cada una de las tareas, se ha optado por realizar un diagrama de Gantt (Véase en el Anexo A1).

4 Estado Del Arte

4.1 Obtención de los datos

Tal y como se ha explicado, el contexto de este Trabajo viene dado por la competición abierta de Kaggle llamada March Machine Learning Mania 2024. Es por ello que los datos utilizados son datos ofrecidos por la organización de la competición. Kaggle incluye varias opciones de descarga de datos. Para este proyecto se ha decidido por obtener todos los datos unidos en un fichero ZIP.

4.2 Descripción de los datos

El fichero comprimido en zip contiene 33 archivos CSV con información relevante a los partidos y a los equipos. Dicha información se encuentra para los equipos masculinos y para los equipos femeninos, en el caso de los equipos masculinos tenemos información desde 1985 y desde 2003 para estadísticas avanzadas, en el caso de los equipos femeninos, tenemos información desde 1998 y desde 2010 estadísticas avanzadas.

La información que contienen los archivos de data set es la siguiente:

- ConferenceToyrneyGames: incluye información sobre partidos de conferencia.
- GamesCities: incluye la información para identificar en que ciudad se juega cada partido.
- MasseyOrdinals: ranking subjetivo de los equipos.
- CompactResults: son archivos que contienen información reducida de los partidos, incluye fecha, equipos y marcador
- NCAATourneySeeds y NCAATourneyResults: sirven para determinar cómo evolucionó el cuadro eliminatorio en años anteriors.
- Seasons: especifica información sobre qué región adopta cada conferencia

Toda esta información viene dada por el data set en cuestión. Una consideración a tener, es que la información dada en la sección masculina es mayor a la femenina. No solo a nivel histórico, si no a nivel de recogida de información, pues en el masculino se incluye información de los partidos previos al torneo de la NCAA.

5 Desarrollo

Como en todo proyecto de tratamiento de datos, este también se compone de las etapas principales (Véase Figura 1). En primer lugar, tenemos la etapa de exploración. Esta etapa tiene como objetivo familiarizarse con los datos y encontrar ciertas características significativas para el caso en cuestión.

La siguiente etapa, la de procesamiento, engloba diferentes conceptos como los siguientes: limpieza de datos, adaptación de los datos y transformación de los datos. En esta etapa, el principal objetico es poder realizar un conjunto de transformaciones a los datos que permitan dejar los datos listos para ser utilizado por un modelo.

La tercera etapa considerara en un proyecto de datos, es la de creación del modelo. Tiene como finalidad encontrar un algoritmo que sea capaz de resolver un problema o cuestión de la manera más correcta posible. Para ello, se hace uso de lo que se conoce en inglés como *hyperparameter tuning*. Este concepto hace referencia a la elección de la mejor configuración del modelo, para obtener el máximo acierto posible con el algoritmo escogido.

Finalmente, como última etapa, tenemos la etapa de Predicción. En esta etapa, una versión convincente del mode-

lo se encarga de predecir valores con tal de resolver la cuestión planteada al principio.

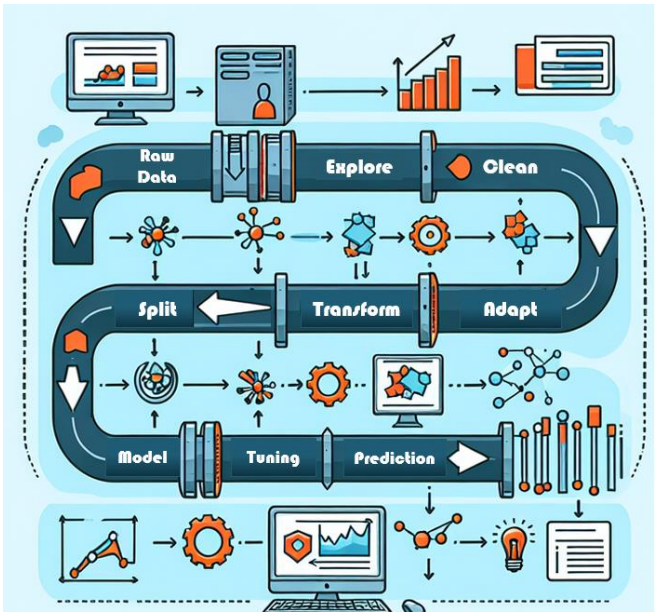


Figura 1 - Procesos de un proyecto de datos

5.1 Exploración profunda de los datos

Con tal de poder, avanzar hacia el objetivo, el primer paso fue realizar una exploración profunda sobre los datos, es decir, buscar inconsistencias, valores anormales o nulos, discrepancia entre distintos archivos.

Tras un análisis en profundo en todos los archivos del data set. Se ha podido comprobar que no existen archivos con información inconsistente, es decir, no hay archivos donde se haga referencia a una cosa y otros donde se haga referencia a algo completamente distinto. Se realizó esta comprobación para poder garantizar que todos los resultados de los partidos eran correctos. También se comprobó que no existían valores nulos, cosa que nos permite tener que evitar plantear una estrategia de tratamiento de valores nulos. Finalmente, también nos cercioramos de que toda referencia, ya sea a un equipo, una ciudad, una conferencia, o cualquier otro aspecto existía.

Es por ello por lo que no es necesario realizar ningún tipo de tarea adicional previa a la preparación de datos.

5.2 Preparación de los datos

El archivo que contienen más información en relación con los equipos son los archivos que ofrecen las estadísticas detalladas de cada partido de los equipos. El objetivo será mediante dichas estadísticas implementar algún tipo de estrategia de procesado que permita utilizar en un futuro modelo aquellas características más determinantes. Para esta sección se han seguido las siguientes estrategias:

- Agrupar por la media las estadísticas en función de los últimos partidos.
- Agrupar por la mediana las estadísticas de los últimos partidos.
- Generar y reducir el número de características haciendo combinaciones lineales entre ellas.

Para implementar la primera estrategia que consiste en la agrupación de estadísticas en función de sus X últimos partidos, se ha realizado lo siguiente. Utilizando de manera única los archivos «RegularSeasonDetailedResults» se ha procedido a agrupar para cada partido las estadísticas de sus últimos 5 y 10 partidos en función de la media. De este modo, para cada partido se tienen los siguientes datos:

- La media de cada estadística de los últimos 5 partidos para el equipo A.
- La media de cada estadística de los últimos 10 partidos para el equipo A.
- La media de cada estadística de los últimos 5 partidos para el equipo B.
- La media de cada estadística de los últimos 10 partidos para el equipo B.
- La media de cada estadística de los últimos 5 partidos de los oponentes del equipo A.
- La media de cada estadística de los últimos 10 partidos de los oponentes del equipo A.
- La media de cada estadística de los últimos 5 partidos de los oponentes del equipo B.
- La media de cada estadística de los últimos 10 partidos de los oponentes del equipo B.

Esta estrategia presenta la ventaja de que permite tener información implícita de los últimos partidos, tanto a nivel de equipo como a nivel de oponente. En otras palabras, permite saber cómo de bien esta un equipo en los últimos partidos y cuanto le están generando sus rivales. Sin embargo, puede estar sesgada debido a que el escoger 5 y 10 como valores para obtener los últimos partidos, se han escogido de manera subjetiva.

Otra estrategia empleada, ha sido la de cambiar le método de agrupación, pues en vez de la media, se ha utilizado la mediana. También se obtiene la información agregada para los propios equipos y para los rivales de dichos equipos. Esta estrategia permite hacer que las agregaciones sean menos sensibles a *outliers*[1] que puedan generar una percepción distinta a la realidad.

Finalmente, con el fin de reducir el número de características, se ha optado por implementar una tercera estrategia. La idea principal de esta es la de utilizar combinaciones lineales para reducir el número de características y generar características que representen mejor la variable Resultado.

Para la implementación de esta estrategia, se ha realizado una exploración sobre la viabilidad de la combinación de estas nuevas estrategias. Según Marcus Hagness [4], antiguo entrenador de baloncesto universitario, una de las

[1] Se denomina *outlier a aquellos valores que están fuera del rango de valores esperados, o que no se comporta con siguiendo la distribución de los datos.*

estadísticas más relevantes es el *Assist-To-Turnover* ratio. Ya que nos da una métrica de como de bien mueve un equipo el balón. Tal y como se puede apreciar en la figura 2, el sólo el 27.19% de los equipos que han perdido logran tener un *Assist-To.Turnover* mayor al del equipo ganador. Por otro lado, podemos ver que el 71.76% de los equipos ganadores tienen un *Assist-To-Turnover* ratio mayor que el rival.

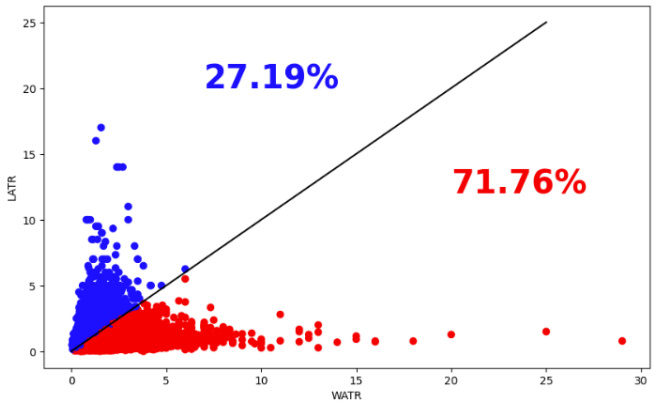


Figura 2 - Comparación Assist-To-Turnover entre 2 equipos

Otra característica, que recomienda utilizar es convertir los rebotes a ratio. De la misma forma vemos que en el 59% de las ocasiones los equipos con mayor ratio de rebotes en ataque ganados salen ganadores del partido. Otra métrica que se recomienda estudiar [5], es el porcentaje de acierto en tiros libres. Tras su estudio, se ha podido comprobar que en el 57% de las ocasiones, el equipo ganador tiene mayor porcentaje de tiro libre.

Se ha utilizado el mismo procedimiento para comprobar si el porcentaje de tiros de campo, y en efecto, en el 81% de las ocasiones el equipo con mayor precisión en tiros de campo es el ganador del partido. De la misma forma se han comprobado si el número de faltas personales también esta correlacionado con el porcentaje de victorias.

De esta forma la estrategia utilizada para el posterior modelo será esta últimos pues añade información no implícita en los datos y de carácter más táctico en lo relativo al deporte.

5.3 Modelo para la predicción

Una vez se ha creado un dataframe que sea capaz de describir de manera más correlacionada el resultado del partido y que tenga un número de características que permita un escalado del modelo óptimo, procedemos a buscar un modelo de *Machine Learning* que dada la entrada prediga 0 o 1 en función de si gana el equipo local o el visitante. Entre los distintos modelos de clasificación destacan 4 que son capaces de ser entrenados de manera eficiente y que permiten distintas configuraciones con tal de ajustar el modelo a nuestro caso. Los 4 modelos en cuestión son:

- XGBClassifier
- Random Forest
- Logistic Regression
- Gaussian Naive Bayes

Es necesario tener en cuenta que, para evaluar los modelos y determinar cuál escoger solo se han utilizado datos de la competición masculina. Esto es debido por la diferencia en estadísticas entre baloncesto masculino y femenino [5].

No se considera, una implementación con un modelo distinto, principalmente porque el preprocesado de datos es idénticos para ambas categorías, por lo que se espera que el modelo tenga un comportamiento similar.

5.3.1 Metricas

De cara a evaluar un modelo concreto, se utilizará la *confussion matrix*. Una *confussion matrix* es una matriz que permite comparar los valores predichos con lo reales. La matriz en nuestro caso tiene un aspecto de 2x2. En la coordenada 0,0 se encuentra el número de valores de una de las clases que se han predicho correctamente. En la coordenada 0,1 se encuentran los valores de esa misma clase que se han predicho de manera incorrecta. En la coordenada 1,0 por el contrario se encuentra el número de datos que pertenecen a la otra clase y los hemos predicho como la primera. Finalmente, en la coordenada 1,1 encontramos el número de elementos que pertenecen a la segunda clase y se han predicho correctamente.

Una matriz de confusión con un modelo perfecto tendría en las coordenadas 0,0 y 1,1 el número total de elementos de cada clase y en el resto de las coordenadas el valor 0.

Este formato proporciona una rápida comparación visual, sobre como funciona el modelo, y sobre todo si el modelo tiende a tener un error de predicción.

Además, con tal de poder determinar que modelo es más eficiente, se compararan las siguientes métricas: F1-Score, Recall, Accuracy de tanto del conjunto de entrenamiento como del de test.

El F1-Score nos indica cuantos datos pueden ser explicados o, mejor dicho, determinados por el modelo. Un F1-Score del 89% indicaría que el modelo es capaz de hacer predicciones con el 89% de los datos. El Recall, nos ayuda a saber con cuantos partidos en los que ha ganado el visitante hemos acertado, respecto al total de partidos ganados por el visitante.

Finalmente, el *accuracy* nos permitirá saber cuántos partidos en global predecimos correctamente, en base a todos los partidos.

5.3.2 XGBClassifier

XGBClassifier es un algoritmo de clasificación de aprendizaje automático basado en el algoritmo Gradient Boosting Tree, un algoritmo que funciona mediante arbo-

les de decisión[2] débiles. En cada nuevo árbol se intenta corregir los errores de los anteriores, mejorando la precisión general. Es conocido por su alto rendimiento y eficiencia, lo que lo convierte en una herramienta popular para una amplia gama de tareas de clasificación.

XGBClassifier funciona creando un conjunto de árboles de decisión débiles. En cada iteración, se agrega un nuevo árbol al conjunto de tal manera que minimice el error general. Este proceso se repite hasta que se alcanza un número máximo de árboles o se cumple un criterio de parada.

Entre sus fortalezas, destaca el hecho de que pude manejar un gran volumen de datos, tiene medidas para evitar el *overfitting* y es posible interpretar los resultados del modelo, para mejorar los resultados. Sin embargo, tiene aspectos negativos, como la alta tendencia al *overfitting* y su gran sensibilidad a la configuración de los hiperparámetros.

Para comprobar la validez del modelo, se han probado distintos hiperparámetros, de tal forma que la mejor configuración ha arrojado los siguientes resultados para el conjunto de entrenamiento:

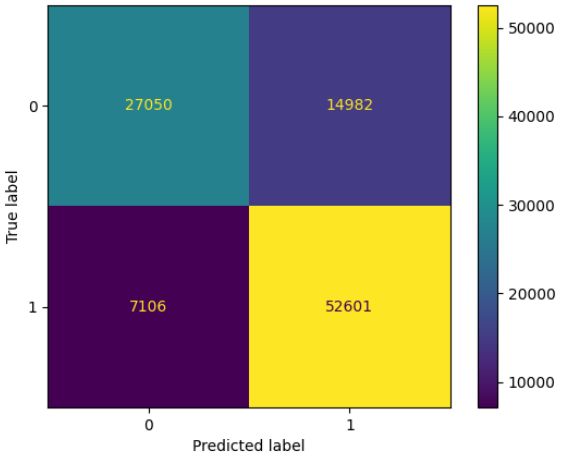


Figura 3 - Confussion Matrix de XGBClassifier en entrenamiento

Si analizamos los resultados del conjunto de entrenamiento, vemos que los resultados son esperanzadores, pues predecimos de manera correcta un 71% de los datos. Sin embargo, si apreciamos los resultados arrojados con el conjunto de test (Ver Figura 4), vemos que la precisión baja hasta el 50.7%.

En conclusión, si analizamos ambos resultados, podemos observar que existe una gran predisposición del modelo a predecir con la clase 0 (gana el equipo visitante).

5.3.3 Random Forest Classifier

El Random Forest Classifier es un algoritmo de aprendizaje automático popular para tareas de clasificación. Fun-

[2] Árbol de decisión: Herramienta similar a un diagrama de flujo utilizada para la toma de decisiones.

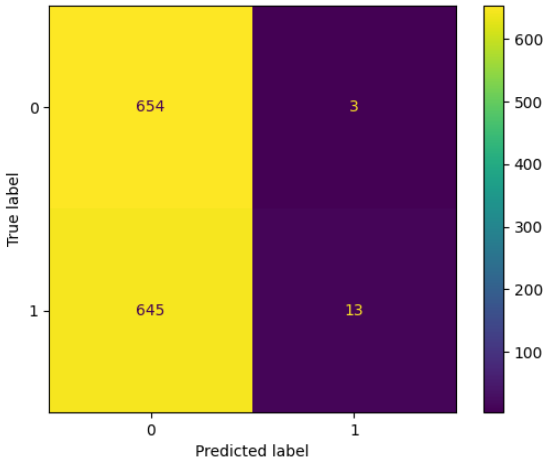


Figura 4 - Confussion Matrix de XGBClassifier en test

ciona creando un conjunto de árboles de decisión individuales y luego combinando sus predicciones para obtener un resultado final. A menudo se le considera un método de aprendizaje conjunto debido a su naturaleza colaborativa.
En primer lugar, entrena múltiples arboles de decisión de forma independiente utilizando submuestras. Cada árbol tiene una profundidad máxima, que es la predefinida. Finalmente, de cara a predecir, cada subárbol, vota por la clase predicha. Entre las fortalezas de este modelo, se puede destacar la robustez al ruido y una selección de características interna, identificando características más importantes. Sin embargo, tiene un alto coste computacional y presenta sensibilidad a los parámetros.

Entrenando el modelo con distintos hiperparámetros, obtenemos los siguientes resultados para su mejor configuración:

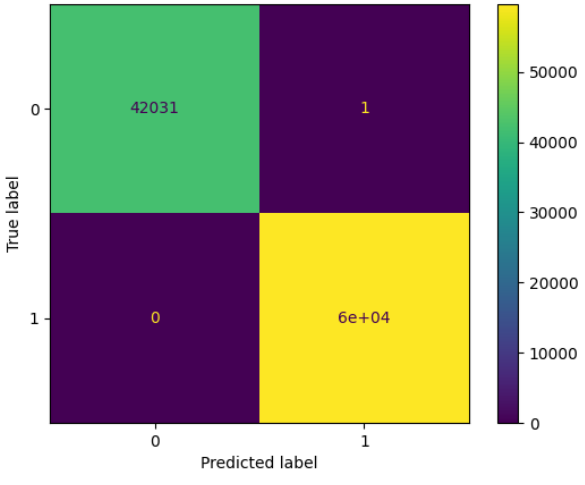


Figura 5 - Confussion Matrix del Random Forest Classifier en entrenamiento

Fácilmente, podemos apreciar que tiene un resultado prácticamente perfecto para el conjunto de entrenamiento. Lamentablemente, esta ratio baja considerablemente para el conjunto de test:

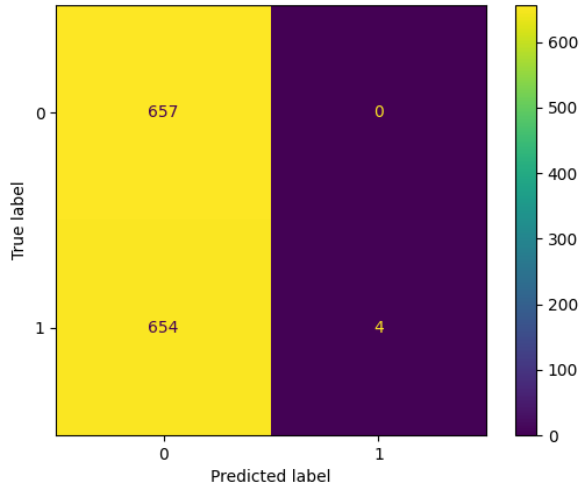


Figura 6 - Confussion Matrix del Random Forest Classifier en test

Al igual que en el modelo anterior, podemos observar que el *accuracy* es del 50%.

En definitiva, este modelo produce *overfitting* pues podemos apreciar que en el conjunto de entrenamiento apenas produce error, en cambio, en el conjunto de test vemos que predice prácticamente todas las predicciones de la misma clase.

5.3.4 Logistic Regression

La Logistic Regression, o Regresión Logística en castellano, es un algoritmo de aprendizaje automático para tareas de clasificación binaria. Es decir, predice la probabilidad de que una instancia pertenezca a una de dos clases.

Logistic Regression utiliza un modelo estadístico que representa la probabilidad de una clase en función de una combinación lineal de las características de entrada. Esta combinación lineal se calcula como la suma ponderada de las características, donde cada peso representa la importancia relativa de cada característica para la predicción.

La función de activación logística, también conocida como función sigmoide, se aplica a esta combinación lineal para obtener una salida entre 0 y 1. Esta salida representa la probabilidad estimada de que la instancia pertenezca a la clase positiva, en nuestro caso, a que el equipo ganador sea el equipo local.

Las principales ventajas de este modelo son: la interpretabilidad, la simplicidad y la eficiencia computacional. No obstante, asume una relación lineal entre las características y el resultado, y tiene problemas con escenarios de muchas dimensiones.

Tras una búsqueda de hiperparámetros del modelo, se han obtenido esta matriz de confusión para el conjunto de entrenamiento:

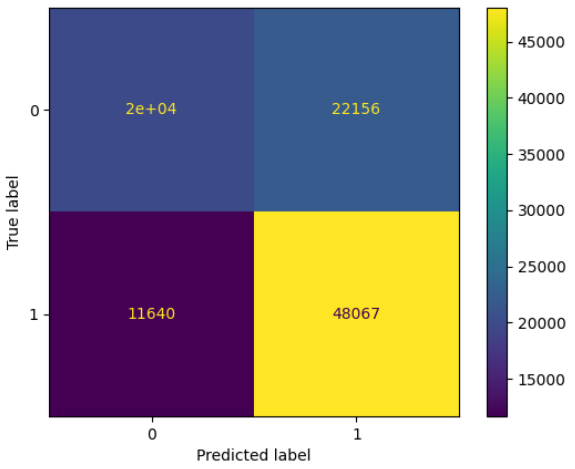


Figura 7 - Confussion Matrix del Logistic Regression en enternamiento

Podemos ver que, en esta ocasión, el *accuracy* del conjunto de entrenamiento es del 70.2%. En esta ocasión, el resultado del *accuracy* del conjunto de test es bastante similar:

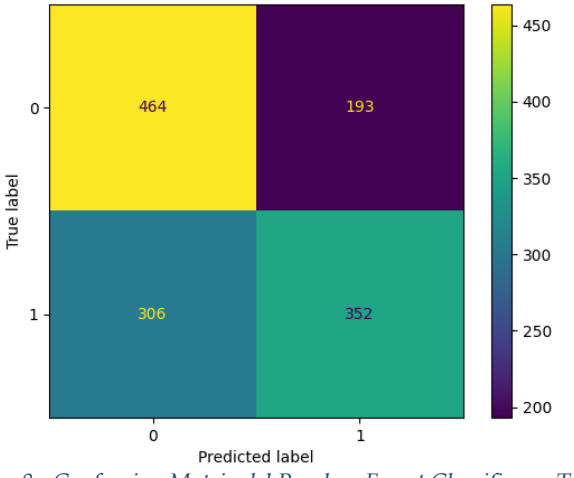


Figura 8 - Confussion Matrix del Random Forest Classifier en Test

En este caso, el *accuracy* solamente se ve decrementado en un 5%. En la comparación del conjunto de test con el conjunto de entrenamiento, podemos observar que el comportamiento es muy similar. Como es de esperar, el *accuracy* disminuye. Además, podemos observar que se equivoca de manera similar tanto para el conjunto de entrenamiento como para el de test.

5.3.5 Gaussian Naive Bayes

El Naive Bayes Gaussiano, también conocido como GaussianNB en Scikit-learn, es un algoritmo de clasificación probabilística basado en el teorema de Bayes y el supuesto de independencia de características. Se utiliza para predecir la clase más probable a la que pertenece una nueva instancia en función de las características observadas.

El algoritmo aplica el teorema de Bayes para calcular la probabilidad posterior de cada clase dada la instancia de

entrada y las características observadas. Esta probabilidad posterior se utiliza para determinar la clase más probable. Se asume que las características son independientes entre sí. Esto significa que la presencia o ausencia de una característica no afecta la probabilidad de las otras características. Se supone que cada característica en cada clase sigue una distribución normal (también conocida como distribución gaussiana). Esta distribución se utiliza para estimar la probabilidad de observar un valor particular de la característica para cada clase.

Es un modelo, simple y fácil de entender. Su algoritmo e s computacionalmente muy eficiente y además, es robusto al ruido. Sin embargo, el supuesto de independencia de características puede ser poco realista lo que afecta a la precisión del modelo. Otro contra, es que no soporta datos categóricos y puede tener problemas con datos que contienen muchas dimensiones.

Tras evaluar el conjunto de entrenamiento sobre este modelo, se han obtenido los siguientes resultados:

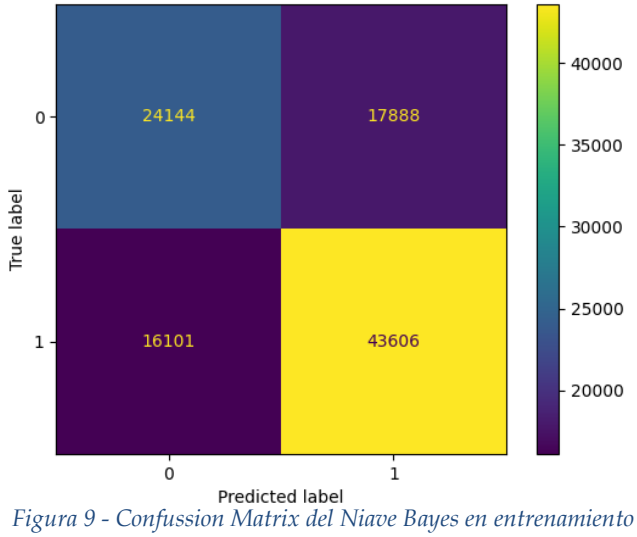


Figura 9 - Confussion Matrix del Niave Bayes en entrenamiento

Si computamos el cálculo del *accuracy*, podemos comprobar que tenemos una precisión del 66%. Es de destacar, que, el número de elementos predichos de manera errónea es similar para los partidos en los que gana el equipo visitante (cuando es 1 o es 0), como en los partidos donde gana el equipo local. En otras palabras, el modelo no se ve afectado por el desbalanceo de los datos.

Ahora bien, si comparamos los resultados obtenidos con el conjunto de entrenamiento y con el conjunto de test (véase la siguiente ilustración) se observa que el porcentaje de acierto disminuye un 10%. No obstante, podemos corroborar, que aparece un problema producido por el desbalanceo de los datos. Es decir, el modelo tiende a predecir casi todos los datos de una misma clase. Generando una bajada en el *recall* del modelo.
En conclusión, no se produce *overfitting*, aunque el resultado con el conjunto de test se ve claramente empeorado.

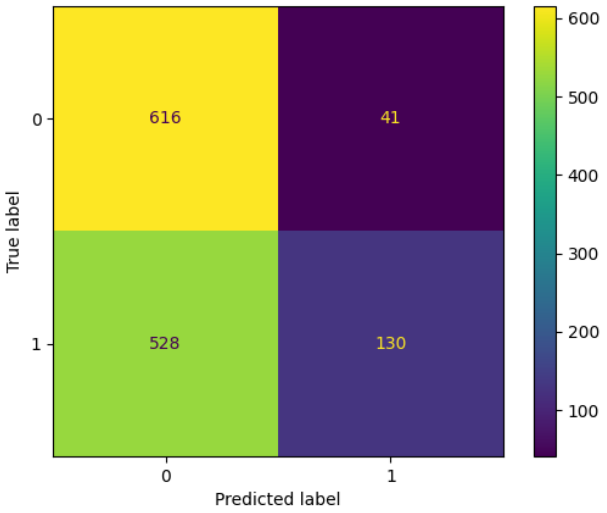


Figura 10 - Confussion Matrix del Naive Bayes en test

5.3.6 Comparación de modelos

Todas las matrices de confusión mostradas con anterioridad han sido obtenidas mediante la implementación de los distintos modelos, con sus mejores configuraciones de hiperparámetros, buscados mediante la realización de un producto vectorial con los distintos valores por cada parámetro configurable de cada modelo.

Partiendo de que se ha obtenido la mejor configuración para cada modelo, con el objetivo de poder comparar esos modelos, se ha creado este gráfico de araña:

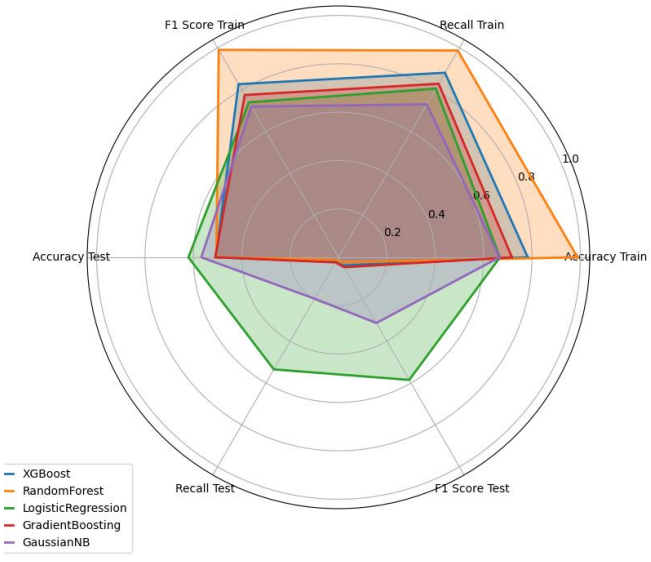


Figura 11 - Comparación de modelos

Como podemos apreciar en este gráfico de araña (Figura 11), para el conjunto de entrenamiento, el mejor modelo es el Random Forest, pues su *accuracy* es casi perfecto. Sin embargo, este modelo tiene un rendimiento muy inferior en el conjunto de test, en comparación con el conjunto de entrenamiento.

Por lo general, la mayoría de los modelos tienen este problema. Es decir, no se ha podido corregir el *overfitting* para los modelos específicos. Este ha sido el principal motivo para descartar todos aquellos modelos que sufren de este problema (XGBoost, Random Forest, Gradinet-Boosting).

Hay 2 modelos específicos que no reflejan esta problemática. Dichos modelos son Gaussian Naive Bayes y el Logistic Regression. Ambos modelos tienen una predicción similar tanto para el conjunto de entrenamiento como para el conjunto de test.
Si comparamos más profundamente estos 2 modelos, podemos observar que, en cuanto al conjunto de entrenamiento, el modelo Logistic Regression tiene mejores valores de *recall* y de F1-Score, pese a tener el mismo *accuracy*. También podemos apreciar que el modelo generaliza mejor, pues para el conjunto de test, los resultados de las 3 métricas estudiadas son mejores.

A continuación, podemos apreciar que hiperparámetros se han configurado con tal de obtener esos resultados:

Modelo	N estimators	Learning rate	Max features	C	Penalty	Max Iter	Var Smoothing
XGBoost	200	0.2					
Random Forest	10		Log2				
Logistic Regerssion				0.1	L2	10000	
Gradient Boosting	200	0.1					
Gaussian NB							1^{-10}

Tabla 1 – Mejores hiperparámetros por modelo

En la Tabla 1, podemos apreciar los distintos hiperparámetros utilizados para el mejor modelo de cada uno de los algoritmos utilizados. En el caso del XGBoost, podemos observar que el número de estimadores (N estimators) es 200. Este número controla el número máximo de árboles de decisión que se utilizan en el modelo. En este modelo, se ha definido el *learning rate* a 0.2. El *learning rate* es un hiperparámetro que controla la magnitud en la que se actualizan los pesos. Cuanto más elevado sea su valor, más drástica será la actualización de pesos.

Para el Random Forest, se ha encontrado como mejor configuración establecer el «N estimators» a 10. En cuanto al prametro Max. Features, hace referencia a el número de características que se consideran aleatoriamente en cada división de un nodo. El hecho de que este definido a «log2» implica que el número de características se basa en el logaritmo en base 2 del total de características disponibles.

Con el Logistic Regression, los hiperparámetros que mejor resultado han dado han sido determinar la C a 0.1. Este parámetro controla la regularización- Un valor alto de C conduce a una regularización más fuerte, penalizando los coeficientes más grandes. Un valor de C demasiado alto puede conducir a *underfitting*[3] y un valor de C demasiado bajo puede conducir a *overfitting*.

Para el Gradient Boosting se han utilizado unos hiperparámetros similares al XGBoost, puesto que la similitud entre estos dos algoritmos es bastante alta.

Finalmente, para el Gaussian Naive Bayes se ha obtenido que el único hiperparámetros a modificar es el «Var Smoothing». Este hiperparámetro permite añadir ruido para que se puedan evitar las multiplicaciones por 0, en caso de que una clase de las características no exista en el conjunto de entrnamiento.

En definitiva, tras probar distintos modelos con distintos hiperparámetro, se ha conseguido obtener un modelo, que se capaz de generalizar, y convertir la información dada por el conjunto de entrenamiento en algo útil para predecir sobre el conjunto de test de una manera solvente. El modelo Logistic Regression permitirá realizar las predicciones, asegurando unos resultados más certeros que con el resto de los modelos.

6 PREDICCIONES

Para poder realizar las predicciones, se debe primero implementar una preparación de datos de cara a los partidos eliminatorios.

En primer lugar, se ha generado un DataFrame que sea nos permita identificar todos y cada uno de los partidos de cada conferencia. De esta forma, podemos tener un *path* de los equipos que van avanzando rondas. Para poder construir este Data Frame, se han tenido en cuenta las normas de clasificación del campeonato. Estas normas estipulan cuáles serán los encuentros en función de la posición clasificatoria en la temporada regular. La organización define que, para los partidos de primera ronda, se utilizará una estrategia de emparejamiento «mejor-peor». Para la segunda ronda, se supone que en la primera ronda se clasifican los mejores clasificados y se vuelve a aplicar una estrategia «mejor-peor». Este proceso es iterativo hasta la final de conferencia. Para los partidos de la *final four*, cada año, se determina el nombre de la conferencia y se hace una asignación alfabética.

Gracias a la implementación de este procedimiento, podemos facilitar el proceso de predicción, haciéndolo au-

[3] La palabra *underfitting* es un anglicismo que hace referencia a cuando un modelo no está suficientemente ajustado a los datos

tomatizado y mucho más entendible.

6.1 Categoría Masculina

Nuestro modelo, se ha equivocado en distintas ocasiones en la fase de predicción. Es por ello que algunos partidos son incomparables con el *bracket* correcto debido a que no se han llegado a dar. No obstante, se pueden extraer 2 conclusiones, cuando un rival muy favorito se enfrenta a otro no tan favorito, se suele predecir que gana el favorito. Como dato curioso, informar que las universidades predichas para llegar a la final son:

- University Tenesee (4)
- Houston University (1)
- Princeton University (15)
- St. Mary's CA University (5).

Ganando St. Mary's CA en la final contra Princeton.

6.2 Categoría Femenina

Para la categoría femenina, de nuevo, no se pueden comparar de manera directa debido a distintos errores en fases previas. Los ganadores de conferencia son:

- South Carloina (1)
- Gonzaga (9)
- University Nevada, Las Vegas (11)
- University of Conneticut (2).

Es de destacar que el ganador final (University South Carolina) se ha predicho correctamente. Si ponemos en contexto la temporada de este equipo, no ha perdido ningún partido de temporada regular, se confirma que el modelo tiene en cuenta las rachas y el estado de forma de los equipos.

7 Conclusiones

A lo largo de este proyecto, se ha realizado un estudio del deporte universitario y de las estadísticas más relevantes, así como un procesado de datos, con los principales objetivos de reducir el número de características y la creación de valor mediante la combinación de algunos atributos. Esto ha permitido poder realizar análisis del modelo de manera más sencilla, y nos ha permitido identificar que variables están más correlacionadas con el resultado.

En cuanto a la creación del modelo se refiere, se han realizado en total 79 modelos distintos, al realizar la búsqueda de los mejores hiperparámetros. Se ha podido comprobar que, con este tipo de datos, los sistemas basados en Arboles de decisiones hay tendencia a *overfitting* en base a los datos de entrenamiento provocando un gran aumento de errores en la predicción con datos nunca antes vistos.

El haber tenido que enfrentarse a un reto de predicción de eventos deportivos, hace resaltar la complejidad que tiene predecir cualquier tipo de suceso en el evento. Debido a una gran cantidad de factores externos de difícil predicción.

En todo momento, se han desarrollado análisis y funcionalidades con la intención de seguir unas buenas practicas de programación. Uno de los propósitos iniciales era desarrollar código siguiendo la guía de estilos PEP8. Esta tarea no ha sido sencilla, pues las continuas adaptaciones de código y distintas decisiones de implementación han provocado tener que hacer un sobreesfuerzo para limpiar el código.

Finalmente, se ha podido crear un modelo para la predicción de partidos que presenta bastante fiabilidad. Además, para distintos tipos de categorías, el mismo tipo de modelo presenta fiabilidad.

En conclusión, partiendo de los 3 objetivos iniciales, se puede afirmar, que se han cumplido los 3. El primero de ellos, realizar un análisis de los datos, se ha llevado a cabo con la finalidad de conocer más en detalle el problema. Después, se ha trabajado con distintas estrategias para procesar los datos, finalmente escogiendo una que combina variables. Para terminar, se comprobó la eficiencia de distintos modelos de predicción.

8 Trabajo a futuro

A lo largo de este proyecto, se ha logrado desarrollar un modelo de aprendizaje automático que puede predecir con cierta precisión los resultados de los partidos del torneo de la NCAA. Sin embargo, hay varias direcciones en las que este trabajo podría expandirse y mejorar en el futuro.

1. Incorporación de más datos:
Aunque los datos utilizados en este proyecto son bastante completos, siempre hay margen para incorporar más información. Por ejemplo, se podrían incluir datos sobre las lesiones de los jugadores, el clima en el día del partido, o incluso el estado de ánimo de los jugadores (a través de análisis de sentimientos de sus publicaciones en redes sociales). Estos factores pueden tener un impacto significativo en el rendimiento de los equipos y, por lo tanto, en los resultados de los partidos.

2. Mejora del modelo de aprendizaje automático:
Aunque el modelo de regresión logística ha demostrado ser bastante eficaz, existen otros algoritmos de aprendizaje automático que podrían ofrecer mejores resultados. Por ejemplo, las redes neuronales profundas y los modelos de aprendizaje por refuerzo podrían ser explorados en futuros trabajos.

3. Realización de análisis en tiempo real:
En este proyecto, las predicciones se realizan antes de los partidos. Sin embargo, sería interesante desarrollar un sistema que pueda realizar análisis en tiempo real, utilizando datos que se generan durante el partido

(por ejemplo, estadísticas de la primera mitad del partido) para ajustar las predicciones.

4. Expansión a otros deportes:
Aunque este proyecto se ha centrado en el baloncesto universitario, los métodos y técnicas utilizados podrían aplicarse a otros deportes. Por lo tanto, un trabajo futuro podría implicar la expansión de este proyecto a otros deportes, como el fútbol, el béisbol o el baloncesto profesional.

9 Agradecimientos

En primer lugar, quiero expresar mi más profundo agradecimiento a mi tutora, Ana Oropesa. Su orientación, paciencia y experiencia han sido invaluables a lo largo de este proyecto. Sus conocimientos y sabiduría han enriquecido enormemente este trabajo, y su aliento me ha ayudado a superar numerosos desafíos.

También quiero extender mi sincero agradecimiento a mi compañero, Abel Espin. Su colaboración ha sido instrumental para la exitosa culminación de este proyecto. Sus ideas innovadoras, pensamiento crítico y apoyo inquebrantable han contribuido enormemente a nuestro trabajo.

A mis amigos, gracias por su constante aliento y por proporcionar una distracción muy necesaria durante los momentos más estresantes. Su confianza en mis habilidades y su apoyo inquebrantable han sido una fuente de fortaleza a lo largo de mi camino.

También estoy agradecido con mis compañeros de trabajo en Boehringer Ingelheim. Su experiencia profesional, consejos y disposición para ayudar han sido invaluables. Las habilidades y conocimientos que he adquirido de ellos no solo han beneficiado a este proyecto, sino que también han contribuido a mi crecimiento personal y profesional.

Bibliografia

[1] El March Madness, el torneo del K.O por excelencia. [Online] Disponible en: https://www.relevo.com/baloncesto/march-madness-torneo-excelencia20230309094751-nt.html [Acceso: Mar-2024]

[2] El March Madness: la quiniela más complicada del deporte con una probabilidad de uno entre nueve trillones. [Online] Disponible en: https://www.relevo.com/baloncesto/nba/march-madness-quiniela-complicada-deporte-20230313150852-nt.html [Acceso: Mar-2024]

[3] Beneficios de implementación Kanban. [Online] Disponible en: https://www.auxiell.com/es/kanban-beneficios/ [Acceso: Abr-2024]

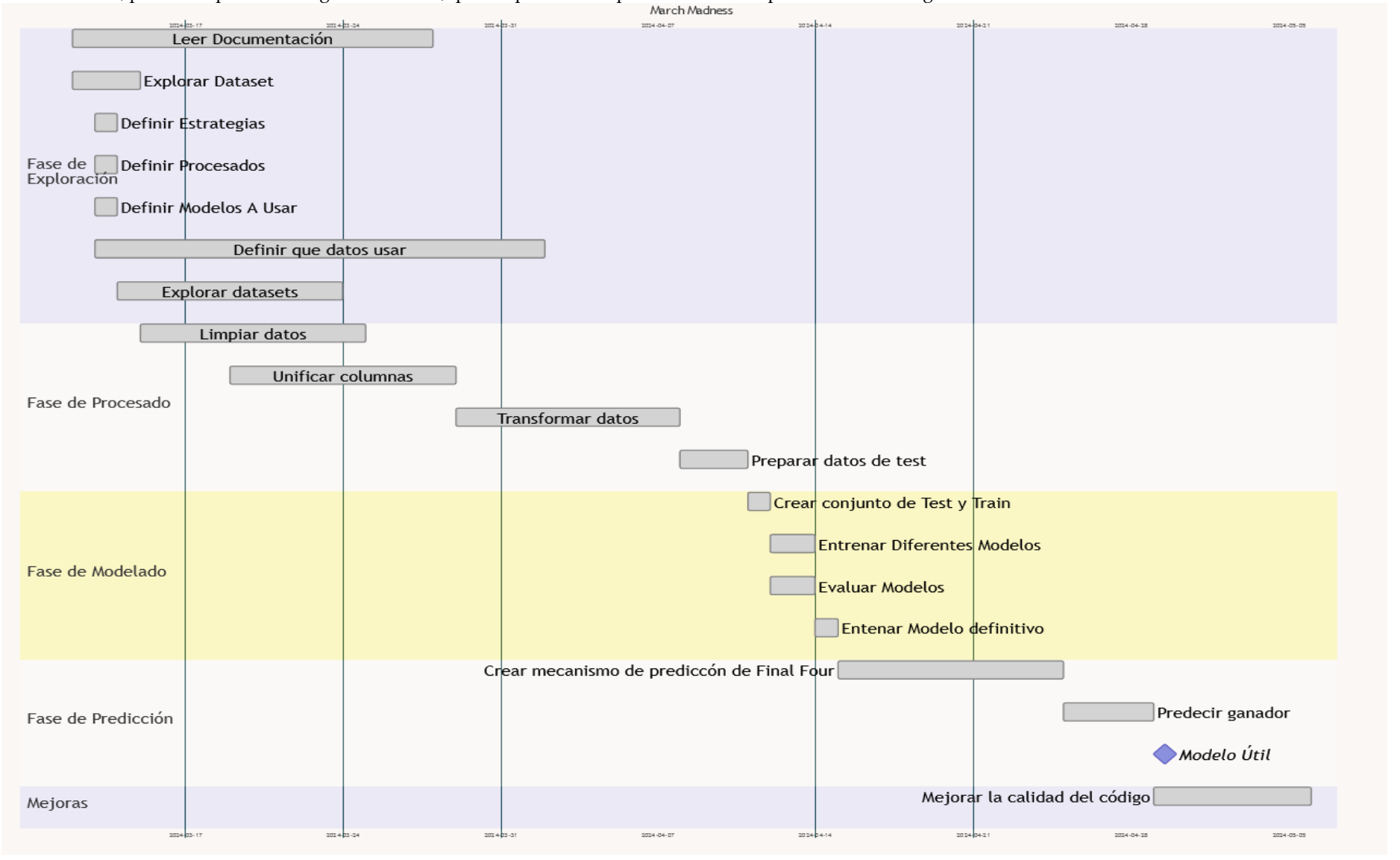
[4] The most important stats to track for your basketball team. [Online] Disponible en: https://www.breakthroughbasketball.com/stats/how-we-use-stats-Hagness.html [Acceso: Abr-2024]

[5] Diferencias en las estadísticas de juego entre bases, aleros y pívots en baloncesto femenino. [Online] Disponible en: [Acceso: May-2024]

ANNEXO

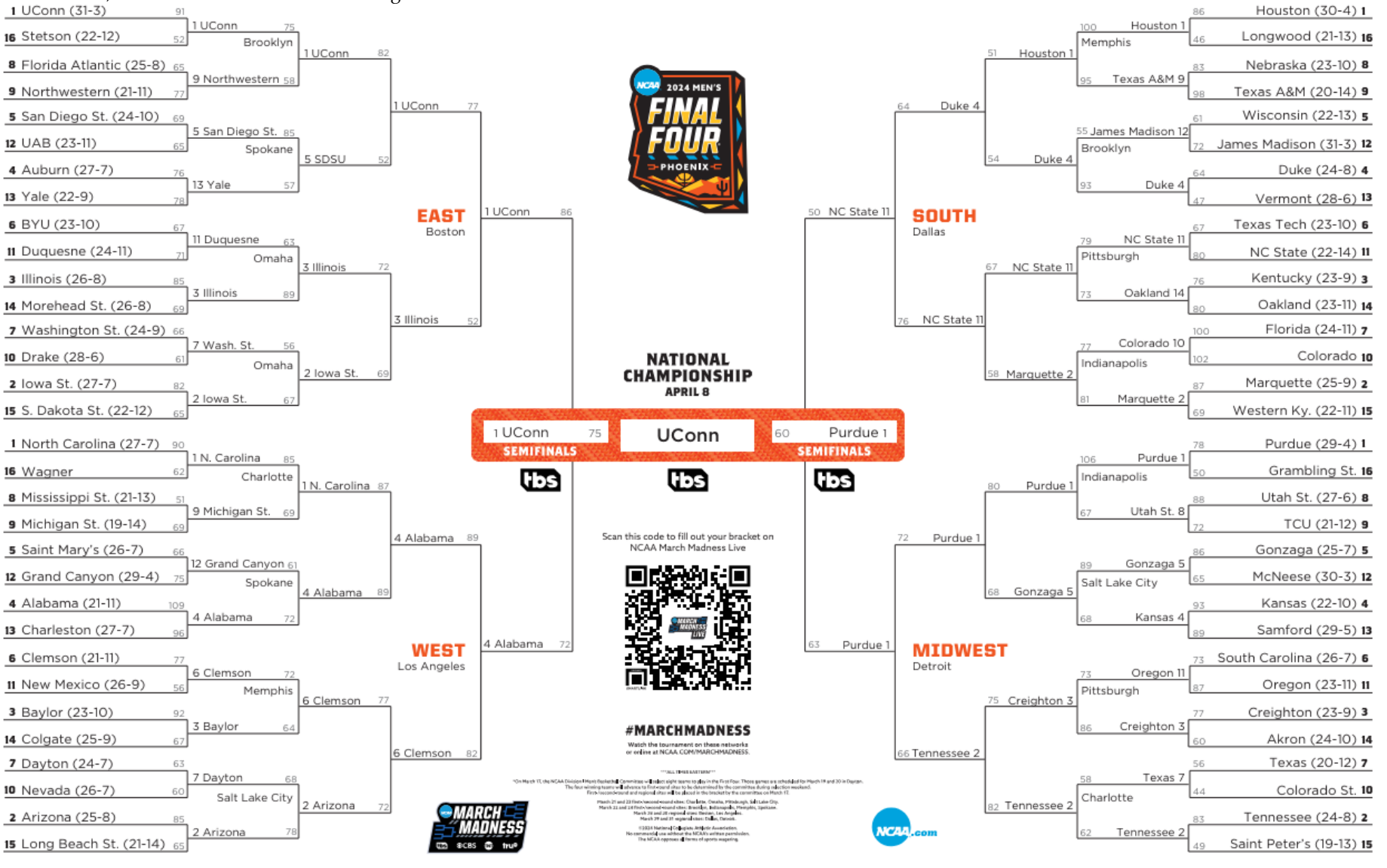
A1. Diagrama de Gantt

A continuación, podemos apreciar el Diagrama de Gantt, que nos permite interpretar como se ha aplicado la metodología:



A2. Bracket 2024 Masculino

A continuación, se muestra el bracket real de la categoría masculina:



A3. Bracket 2024 Femenino

A continuación, se muestra el bracket real de la categoría femenina: