



Escuela Técnica Superior de Ingeniería Informática

Grado en Ingeniería informática. Ingeniería del Software

TRABAJO FIN DE GRADO

Predicción de resultados deportivos.
Algoritmo de aprendizaje automático y aplicación android.

Autor/es:
Francisco Andrés Caro Albarrán

Tutor/es:
José Antonio Troyano Jiménez

Segunda convocatoria

Curso 2022/2023

TABLA DE CONTENIDO

1. Introducción	3
2. Planificación	8
3. Análisis de soluciones relacionadas	10
3.1 DeepTip 8	11
3.2 AI Football Analyser and Stats	13
3.3 Football AI	14
3.4 Nerdy Tips	16
3.5 Bullet Bet Predictions	17
3.6 Tabla comparativa	21
4. Introducción al aprendizaje automático	22
4.1 Librerías Python	23
4.1.1 Pandas	23
4.1.2 Sklearn (scikit-learn)	25
4.2 Kaggle	27
4.3 Algoritmo de regresión lineal	28
4.4 Algoritmo de regresión logística	30
5. Herramientas para el desarrollo	31
5.1 Flutter	33
5.2 SQLite y DBeaver	34
5.3 NodeJS	36
5.4 Visual Studio Code	37
6. Datos utilizados	39
6.1 Obtención y preprocesado de los datos	40
6.1.1 Análisis del dataset primitivo	41
6.1.2 Creación de funciones auxiliares	42
6.2 Generación del nuevo dataset	49
6.3 Problemas encontrados	52
7. Experimentación	57
7.1 Introducción	57

7.2 Dataset de entrenamiento y dataset de predicción	58
7.3 Elección del modelo	60
7.3.1 Redes neuronales	61
7.3.2 Algoritmos de regresión	63
7.4 Resultados	68
7.5 Problemas encontrados	71
8. Documentación técnica	73
8.1. Introducción	73
8.2 Requisitos y casos de uso	74
8.3 Mockups	76
8.4 Desarrollo de la aplicación	80
8.4.1. Diseño y estructura	80
8.4.2. Funcionalidades e interfaz	84
8.4.3. Problemas encontrados	89
8.5 Instrucciones para ejecutar el modelo de aprendizaje automático	91
8.6 Instrucciones para ejecutar la aplicación Flutter	92
9. Conclusiones	96
10. Referencias	98

1. Introducción

¿Imaginas cómo sería la vida sin la luz eléctrica, sin la rueda o sin internet?

Pues en un período muy corto de tiempo tampoco vamos a ser capaces de imaginarnos nuestra vida sin inteligencia artificial. A día de hoy, ya forma parte de la vida de una gran parte de la población, desde el pilotaje automático de vehículos, la detección y reconocimiento del rostro para desbloquear un smartphone o la manipulación y análisis masivo de datos para maximizar el rendimiento deportivo. Numerosos ejemplos recientes ponen en valor el potencial de la inteligencia artificial, y es que ni un deporte tan asentado en la historia de la sociedad europea ha podido resistirse a su uso.

Mientras todos los deportes evolucionan y modifican sus reglas durante el siglo XX para beneficiar así al espectáculo, el reglamento del fútbol ha permanecido prácticamente fijo. Sin embargo, cuando nos adentramos en el mundo del análisis de datos en el deporte, no podemos decir lo mismo. Numerosos han sido los equipos de fútbol que han sido pioneros en el uso de herramientas de inteligencia artificial para analizar a los rivales o evaluar el rendimiento de sus jugadores.

Resulta sorprendente cómo la inteligencia artificial ha logrado penetrar en la esencia misma del fútbol, adentrándose en las entrañas de cada partido y desvelando secretos ocultos hasta ahora. Actualmente, además de ser usada por los propios equipos para realizar análisis más profundos, es usada por numerosos espectadores con multitud de objetivos; desde la predicción del ganador de un enfrentamiento apasionante hasta el cálculo preciso de los goles anotados en un

encuentro, la inteligencia artificial se ha convertido en un aliado imprescindible para aquellos que buscan desentrañar los misterios del deporte rey en el mundo.

A través de complejos algoritmos y análisis minuciosos, se extraen patrones y tendencias que ofrecen una visión reveladora sobre qué equipo se alzará con la victoria y cómo se desarrollará el marcador. La inteligencia artificial, en su afán por potenciar la emoción del juego, se convierte en una guía infalible que nos lleva más allá de las simples suposiciones y nos sumerge en un mundo de pronósticos certeros y análisis rigurosos.

Al abordar la realización de este proyecto, teníamos en mente una serie de objetivos y requisitos que son los que nos han guiado a lo largo de su desarrollo. En primer lugar, buscamos profundizar en el campo de la inteligencia artificial aplicada al análisis de partidos de fútbol, con el propósito de comprender y explorar las posibilidades que esta disciplina ofrece en el ámbito deportivo. Nuestro objetivo principal es desarrollar una aplicación que utilice algoritmos de inteligencia artificial para realizar predicciones certeras sobre los resultados y el desarrollo de los encuentros futbolísticos. Para lograrlo, debemos cumplir con requisitos esenciales, tales como la recopilación de datos precisos y actualizados, la selección y aplicación adecuada de algoritmos de aprendizaje automático, y la implementación de un modelo de evaluación y validación riguroso para garantizar la fiabilidad de nuestras predicciones. Con estos objetivos y requisitos primitivos en mente, decidimos iniciar el desarrollo de este proyecto y con ello comenzar a explorar y descubrir el apasionante campo del análisis deportivo y la aplicación de la inteligencia artificial en el mundo del fútbol.

En los siguientes capítulos de esta memoria, profundizaremos en las distintas etapas por las que ha pasado el proyecto, asimismo comentaremos y analizaremos las distintas tecnologías utilizadas así como las decisiones que nos han hecho decantarnos por las mismas.

La planificación es una de las partes más importantes cuando hablamos de un proyecto software, en dicho capítulo expondremos las principales tareas y objetivos que tiene el proyecto así como una estimación a priori del tiempo aproximado que se va a emplear en la realización de cada una.

El siguiente capítulo es lo más parecido a un estudio de mercado realizado por cualquier empresa u organización antes del lanzamiento de algún producto. Hemos indagado en las aplicaciones relacionadas que están en el mercado y que realizan predicciones de los resultados de los partidos combinados en algunos casos con las distintas cuotas que presentan las casas de apuestas. Todo ello queda resumido finalmente en una tabla comparativa que ofrece una visión general y resumida de las características de cada aplicación analizada.

Los siguientes capítulos, introducción al aprendizaje automático y herramientas para el desarrollo, tienen como objetivo sentar las bases teóricas de las tecnologías y conceptos usados posteriormente en el desarrollo tanto del algoritmo de inteligencia artificial como de la aplicación.

Datos utilizados y experimentación reflejan cómo ha sido el tratamiento y transformación de un conjunto de datos que contenía los resultados de todos los partidos disputados en la liga española desde el año 1995 hasta el año 2020 y de qué modo hemos conseguido sacar las características deseadas para entrenar a nuestro algoritmo y cómo ha ido evolucionando y mejorando el mismo.

Seguidamente, nos encontramos con el capítulo de documentación técnica en el que se explicará todo lo relacionado con el desarrollo de la aplicación móvil, su integración con la API, los problemas a los que nos hemos enfrentado y por último una guía de instalación de la aplicación.

Este proyecto ha sido desarrollado conjuntamente con Alberto Gallego Huerta, quien ha sido el encargado de realizar el desarrollo de todo lo relacionado con el ámbito de la base de datos y la API. En su proyecto “Predicción de resultados deportivos. Backend y aplicación web” explica detalladamente cómo ha sido este proceso y cómo se ha logrado desarrollar la aplicación. Nuestros proyectos tienen parte de desarrollo común como es el tratamiento de los datos iniciales o la adquisición de conocimientos relacionados con el aprendizaje automático y otra parte que ha sido desarrollada de manera individual pero que necesita retroalimentarse de la parte común del proyecto para su correcto funcionamiento, como son los desarrollos de ambas aplicaciones, tanto móvil como web.

En conclusión, en este proyecto, hemos conseguido desarrollar una aplicación móvil que muestra una predicción tanto de la probabilidad que tiene cada equipo de ganar el partido, como el resultado esperado al descanso y al final del encuentro. Dicha predicción es realizada por un algoritmo de inteligencia artificial que ha sido entrenado con datos de todos los partidos disputados en la liga española desde el año 1995 y que mediante distintos tipos de algoritmos es capaz de realizar una predicción más que fiable del resultado de los partidos de la temporada 2020-2021, que será el conjunto de partidos sobre los que vamos a realizar dichas predicciones. Todo ello ha sido posible gracias a la integración y configuración de los diversos componentes que forman nuestra aplicación, como el modelo de inteligencia artificial, la base de datos junto a la API y la aplicación móvil.

2. Planificación

El presente proyecto tiene como objetivo el desarrollo de una aplicación móvil que sea capaz de predecir la probabilidad que tiene un equipo de fútbol de ganar un partido usando inteligencia artificial, así como predecir los goles que se marcarán antes del descanso y el resultado final del partido. La planificación de este proyecto se ha realizado considerando una serie de tareas preliminares como la búsqueda de aplicaciones relacionadas, la elección de las tecnologías a usar o el desarrollo de la propia aplicación. A dichas tareas se les ha realizado una estimación a priori de su duración y en conjunto, suman un total de unas 300 horas aproximadamente. A continuación se expondrán las diversas tareas a realizar junto con una breve descripción de cada una.

- Investigación preliminar: Realizar una búsqueda exhaustiva de aplicaciones similares a la que deseas crear, identificando las características, ventajas y limitaciones de cada una. Diseñar una tabla resumen que recoja todas las características de cada aplicación para poder realizar una comparativa rápida y eficiente con la nuestra.
- Diseño de la arquitectura de la aplicación: Definir la estructura de la aplicación, sus componentes y cómo se relacionarán entre sí.
- Elección de herramientas: Evaluar distintas herramientas disponibles en el mercado que permitan la implementación de inteligencia artificial en la aplicación móvil, así como herramientas para el desarrollo en Flutter.

- Recopilación y limpieza de datos: Identificar y recolectar los datos necesarios para el entrenamiento del modelo, y realizar una limpieza y transformación de los mismos para que sean adecuados para su procesamiento.
- Entrenamiento del modelo: Implementar y entrenar el modelo de inteligencia artificial utilizando bibliotecas de Python y los datos recolectados y transformados en la tarea anterior.
- Integración de la base de datos a través de una API: Configurar la integración e interacción de la base de datos y la aplicación móvil, definir la estructura de la base de datos y establecer la conexión entre la aplicación y la base de datos .
- Desarrollo de la interfaz de usuario: Implementar el diseño de la interfaz de usuario de la aplicación, incluyendo las pantallas y elementos necesarios para que el usuario pueda interactuar con la aplicación.
- Integración del modelo en la aplicación: Implementar la integración del modelo de inteligencia artificial en la aplicación móvil, para que pueda realizar predicciones de los resultados de los partidos.
- Elaboración de la memoria del proyecto: Redactar la memoria del trabajo fin de grado, detallando los objetivos, la metodología utilizada, los resultados obtenidos y las conclusiones a las que se llegó en el desarrollo del proyecto.

En la siguiente tabla, se realizará una comparativa entre las horas estimadas a priori y las horas reales invertidas en cada tarea contabilizadas con la herramienta Clockify:

Tarea	Estimación a priori
Investigación preliminar	10
Diseño de la arquitectura	10
Elección de herramientas	10
Recopilación y limpieza de datos	40
Entrenamiento del modelo	70
Integración de la base de datos	15
Desarrollo de la interfaz de usuario	65
Integración del modelo en la aplicación	15
Elaboración de la memoria	70

La estimación a priori de la duración aproximada de las tareas, conlleva que el proyecto tendrá una duración total de unas 305 horas.

3. Análisis de soluciones relacionadas

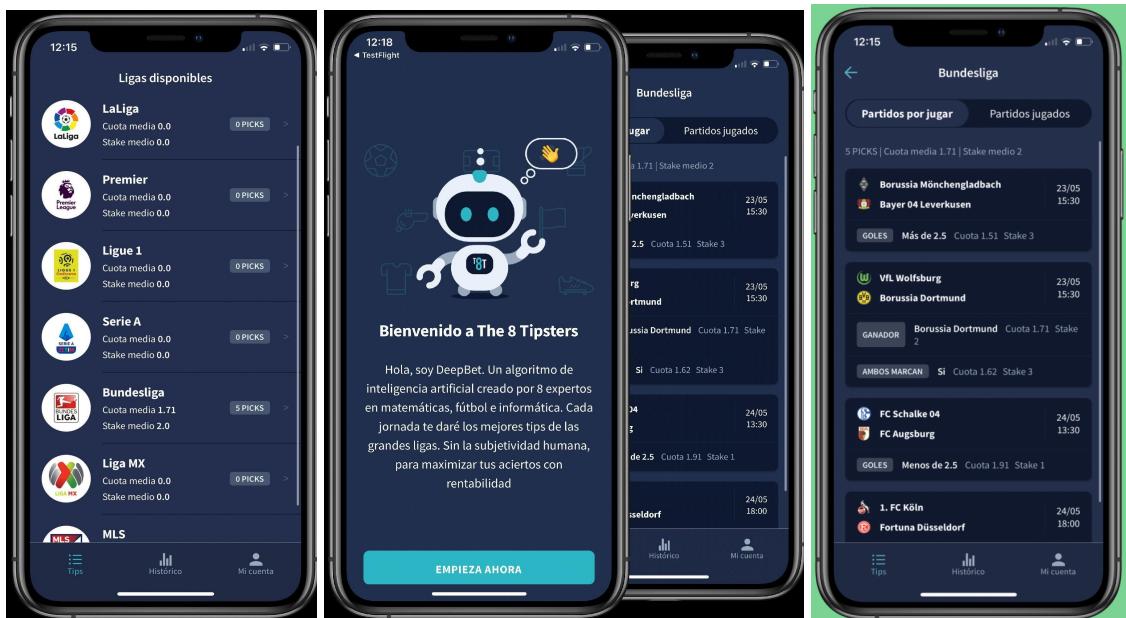
Antes de comenzar con el desarrollo de la aplicación, se ha realizado una investigación exhaustiva de las aplicaciones existentes en el mercado que tienen características similares. El objetivo de esta búsqueda fue identificar las fortalezas y debilidades de las aplicaciones existentes y encontrar oportunidades para mejorar en la experiencia del usuario. Se han evaluado diferentes aspectos, como la funcionalidad, la interfaz de usuario, la usabilidad y las características de la inteligencia artificial utilizada.

En este capítulo se describirán las principales aplicaciones encontradas y se expondrán sus características más relevantes. Esto permitirá establecer una comparativa entre la aplicación que se pretende desarrollar y las ya existentes, de modo que se pueda detectar qué aspectos deben ser mejorados para lograr una aplicación competitiva y de calidad.

3.1 DeepTip 8

DeepTip 8 es una aplicación de pago para Android y iPhone que usa inteligencia artificial para predecir el resultados de los partidos de las 5 grandes ligas europeas así como de la estadounidense y la mexicana. Es creada íntegramente por un grupo de matemáticos e informáticos asturianos y entrenada con datos del que es el mejor proveedor de datos del mundo: Stats Perform.

De cara al usuario, DeepTip8 tiene un funcionamiento muy sencillo. Al iniciar la aplicación el usuario se le muestran directamente los pronósticos disponibles, las cuotas relacionadas con los distintos resultados y los stakes, que son la recomendación de la cantidad de dinero a apostar. A mayor stake, mayor es la probabilidad de que se produzca ese resultado.



DeepTip 8 posee un apartado de “histórico” en el que puedes ver el historial de apuestas realizadas por esa cuenta y otro apartado en el que poder acceder y modificar los datos de tu perfil denominado “Mi cuenta”.

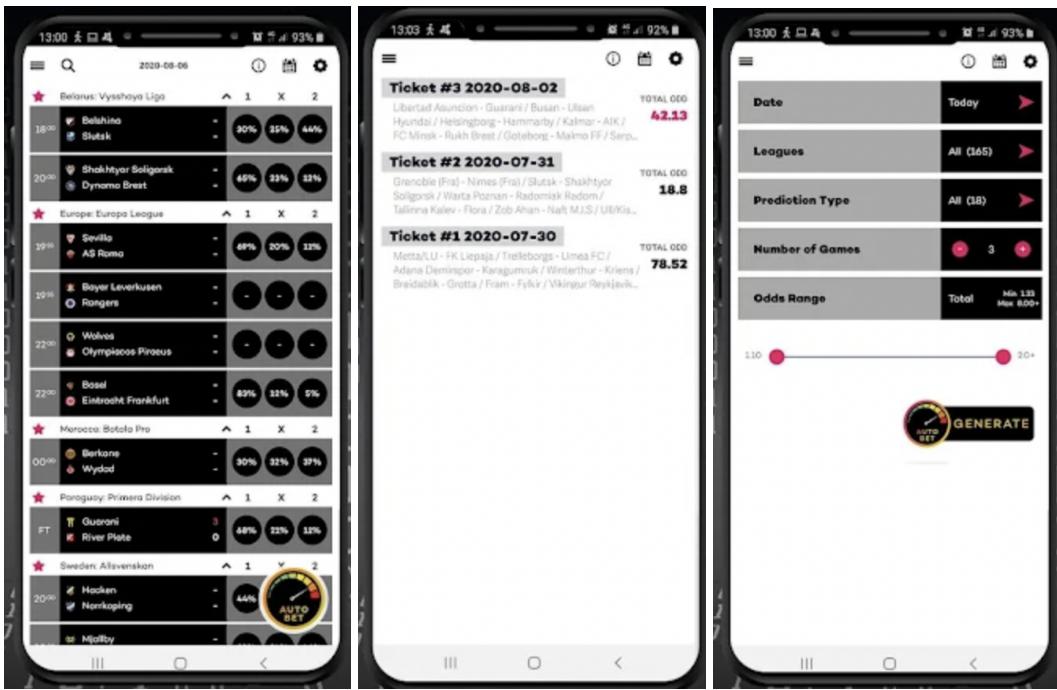
En cuanto a tecnología, esta aplicación desarrollada íntegramente por 8 expertos en matemáticas e informáticos asturianos, usa Deep Learning para analizar de forma automática el resultado de los partidos y además esto permite a los algoritmos aprender a partir de sus propias predicciones pasadas.

3.2 AI Football Analyser and Stats

AI Football Analyser and Stats es una aplicación Android lanzada en 2022 y que usa la inteligencia artificial para ofrecer porcentajes de predicción sobre distintas acciones de partidos de fútbol, por ejemplo, quién será el ganador o si se marcarán más de X goles o si marcarán ambos equipos durante el transcurso de los 90 minutos. A diferencia de DeepTip 8, esta app simplemente te muestra cual es el porcentaje estimado de que ocurra cierta acción (por ejemplo, un 66% de que gane el equipo local).

Football Analyser utiliza un algoritmo de predicción basado en inteligencia artificial. Esta app usa distintos datos estadísticos para la realización de sus cálculos como los enfrentamientos pasados, la forma actual del equipo, los goles anotados en los últimos partidos, los goles concedidos o el perfil del equipo entre otros.

Esta es la interfaz de la aplicación, donde en la página de inicio aparece una lista con los partidos más relevantes así como el porcentaje de que ocurra cierto resultado.



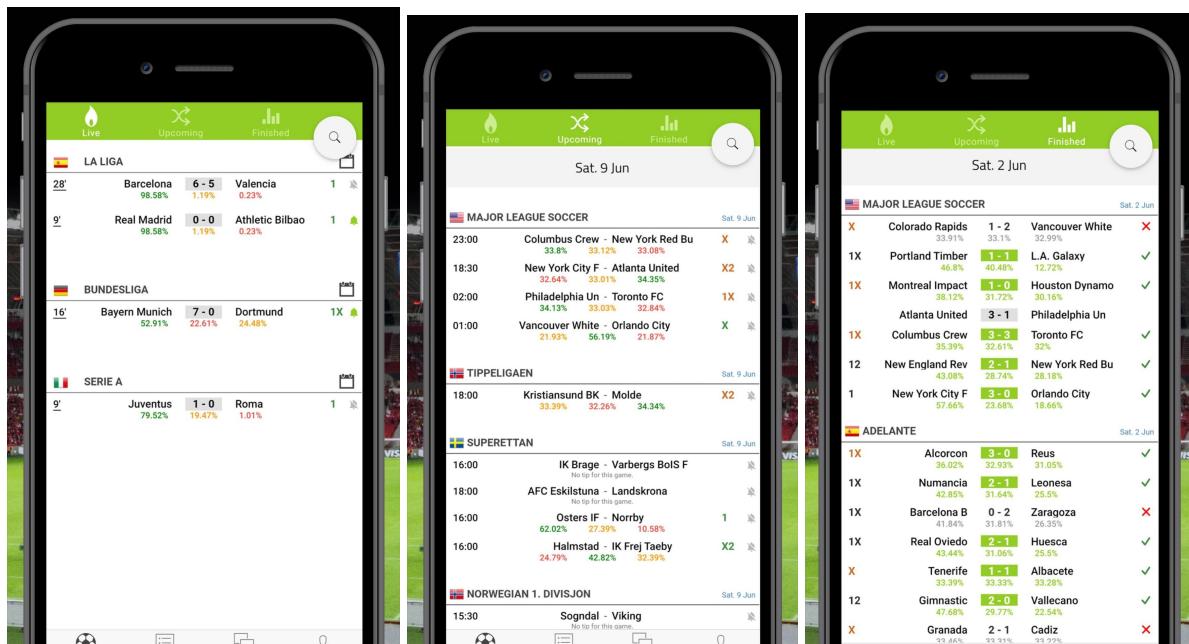
Además la aplicación permite obtener un historial de los tickets creados, así como los partidos involucrados en dichas apuestas. También se puede configurar y filtrar los partidos. Podemos filtrar por la fecha del encuentro, las distintas ligas en las que estemos interesados, el tipo de predicción o filtrar para que solo aparezcan cuotas que estén dentro de un rango seleccionado.

3.3 Football AI

Es una aplicación Android que acumula más de un millón de descargas, y que usa la inteligencia artificial para predecir el resultado de partidos de fútbol. Usa un algoritmo de machine learning que analiza y tiene en cuenta gran cantidad de factores que influyen a la hora de determinar el resultado de un partido, como pueden ser analizar la forma en la que llega un equipo, los enfrentamientos

pasados, la calidad de la plantilla, las lesiones que puedan afectar a los distintos jugadores o incluso si el equipo juega de local o de visitante.

En su interfaz se diferencian tres menús distintos, en los que se muestran los diferentes partidos. Al pulsar en “live” se mostrarán todos los partidos en directo, “upcoming” los que se jugarán próximamente y en “finished” los terminados. En todos los partidos se muestra siempre el porcentaje que Football AI calcula para la victoria de casas equipo.



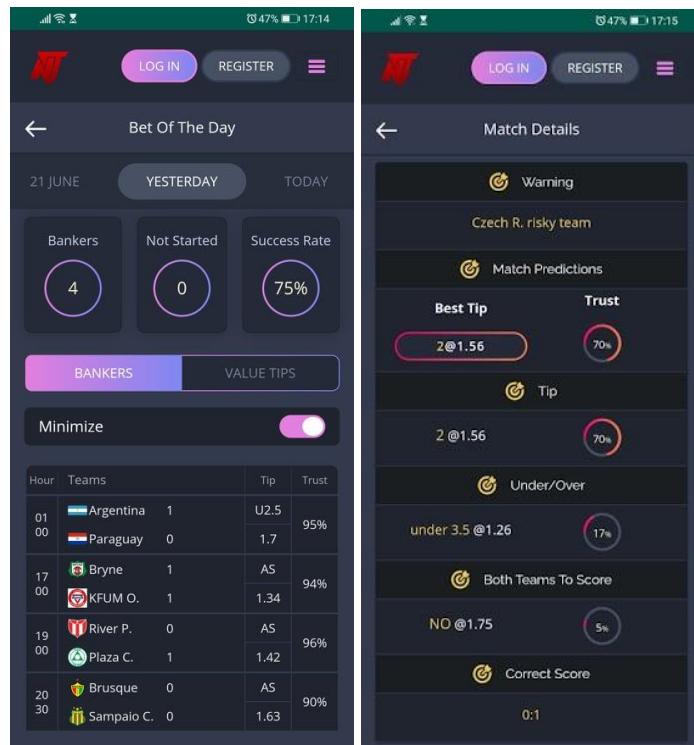
En la imagen de la izquierda, se muestran los partidos en curso, los porcentajes actualizados y en la columna derecha la apuesta seleccionada por el algoritmo como la más eficiente. En la imagen central se muestra la lista de los partidos que se disputarán próximamente, los porcentajes que tiene cada equipo de obtener la victoria y a la derecha la predicción recomendada por el algoritmo. En la última, aparece la lista de todos los partidos finalizados y muestra la predicción que

realizó el algoritmo, los porcentajes que le otorgaba a cada equipo y si finalmente se cumplió dicha predicción o no.

Esta aplicación también ofrece una versión VIP de pago y que garantiza un análisis y unas predicciones más completas de los partidos que no ofrece la versión gratuita.

3.4 Nerdy Tips

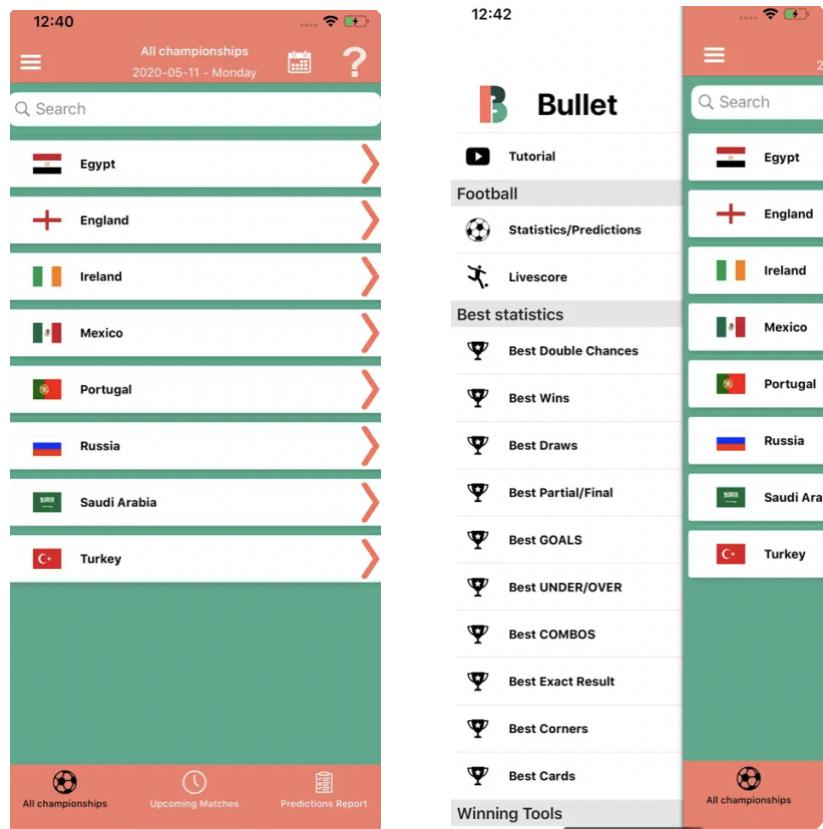
Es una aplicación de predicción de resultados deportivos mediante inteligencia artificial y está diseñada especialmente para iPad. NerdyTips recomienda la que sería la apuesta que tiene más probabilidad de que ocurra dentro de un evento deportivo, así como cuál es la probabilidad de que esta suceda.



En la imagen de la izquierda podemos apreciar la página de inicio de la aplicación. Donde podemos ver una lista de todos los partidos que han ocurrido en la fecha seleccionada, así como cuál es el evento con mayor probabilidad de que ocurra. Por ejemplo, si miramos el partido de Argentina - Paraguay, el algoritmo predice que iba a haber menos de 2.5 goles totales en el partido y le da un 95% de probabilidad. Si pulsamos sobre cualquiera de los partidos, pasaremos a ver los detalles de dicho partido, como es el caso de la imagen de la derecha. En esta imagen, se muestra tanto resultado como la fecha del partido, un leve comentario que ofrece la aplicación sobre el contexto del partido, por ejemplo, que República Checa es un equipo peligroso. A continuación se muestran distintos eventos del partido y su predicción, como el resultado del partido, cantidad de goles o si ambos equipos marcarán. Además la aplicación te muestra cual sería la más interesante en cuanto a beneficio económico basándose en la probabilidad de que ocurra dicho evento y la cuota.

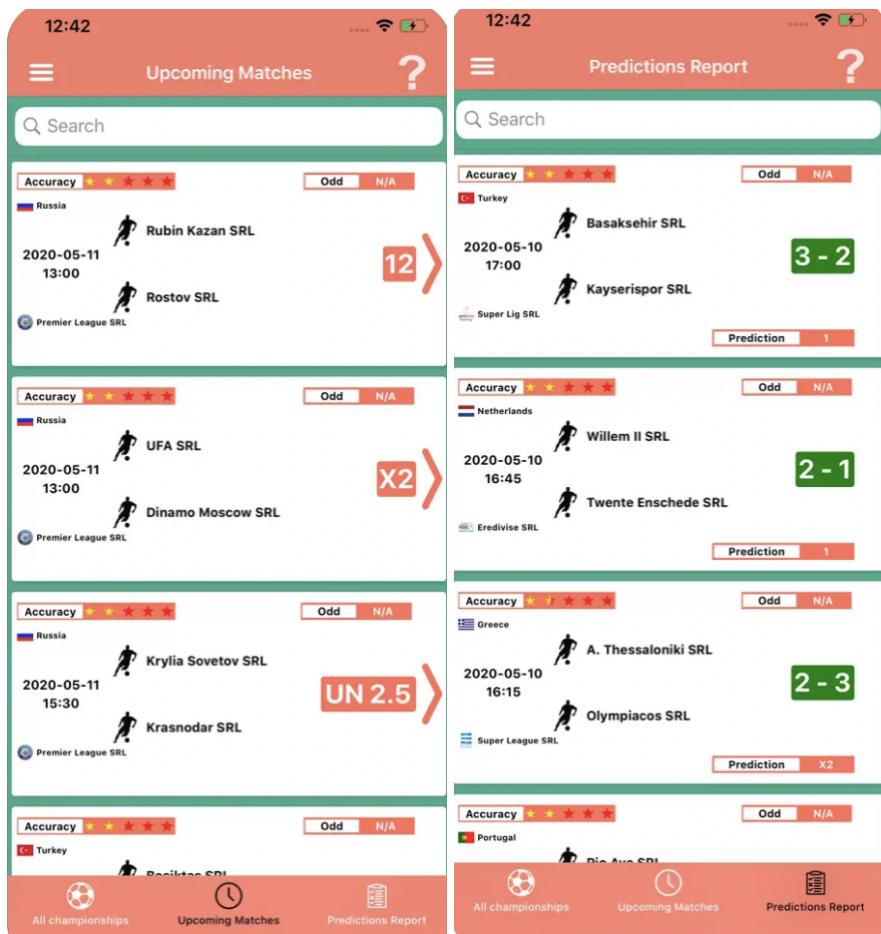
3.5 Bullet Bet Predictions

Bullet Bet Predictions es una aplicación para iphone y android que se ayuda de la inteligencia artificial para predecir tanto los resultados de los partidos, como el posible equipo ganador en el descanso, la cantidad de córners que habrá en un partido o el número de tarjetas mostradas a cada equipo. Además permite crear y recomienda cuáles serían las mejores predicciones combinadas.



La imagen de la izquierda, refleja la página de inicio de nuestra aplicación. Se muestra una primera lista con los países en los que van a suceder los distintos partidos. En el menú de la parte inferior de la pantalla, se puede cambiar entre distintas opciones, en la primera se muestran todas las competiciones, en la segunda se muestran los partidos que sucederán próximamente y en la última se muestra un informe de las predicciones de la aplicación.

En la segunda imagen, se puede ver un desplegable con información relevante tanto sobre los partidos, ya que permite seguir los resultados de los distintos encuentros que se estén disputando en directo, como una lista con los mejores tipos de predicciones.



En estas imágenes podemos observar los menús de los próximos partidos y el historial de predicciones. Para cada partido que se vaya a jugar próximamente, aparece la fecha y hora del partido y en la zona derecha de la pantalla se puede apreciar el evento que tiene mayor probabilidad de que ocurra. Por ejemplo en el primer partido, que se disputará entre el Rubin Kazan - Rostov SRL, la aplicación recomienda 1 o 2, es decir, predice que uno de los dos equipos ganará el partido y que el empate tendrá menos probabilidad de que suceda. En la parte superior de cada partido, se puede apreciar la precisión de la predicción, medida con estrellas.

En la segunda imagen vemos el historial de predicciones. Para cada partido, sale el resultado en la parte derecha y justo abajo se nos muestra cuál fue la predicción recomendada por la aplicación. Si la predicción fue correcta, el resultado del partido se nos mostrará en verde y si la aplicación falló en la predicción, dicho resultado tendrá un fondo rojo.

En conclusión y tras realizar una búsqueda de aplicaciones que usen la inteligencia artificial y el deep learning para la predicción de resultados de partidos de fútbol, el número de goles que se esperan en dicho partido. Muchas de estas aplicaciones combinan las predicciones de sus algoritmos con las cuotas de dichos eventos, buscando en una gran cantidad de casos obtener un rendimiento económico.

Hay multitud de variedad ya no sólo en las distintas funcionalidades que ofrecen las aplicaciones, sino en las distintas interfaces y la experiencia de usuario. Hay algunas aplicaciones como AI Football Analyser and Stats, que es bastante poco intuitiva y que ofrece una experiencia de usuario bastante mejorable. Sin embargo, otras como DeepTip 8 o NerdyTips, cuidan bastante este aspecto, aunque no ofrezcan un gran catálogo de funcionalidades.

En una aplicación de este tipo, no hay que olvidar que lo más importante y lo que es realmente diferenciador es la capacidad de acierto que tenga la inteligencia artificial y el algoritmo usado.

3.6 Tabla comparativa

A continuación, se muestra una tabla resumen con las distintas características de cada aplicación:

	DeepTip8	Football AI	AI Football Analyser and Stats	NerdyTips	Bullet Bet
Seguimiento en directo de partidos	NO	SI	NO	NO	SI
Predicción del resultado del partido (1 X 2)	SI	SI	SI	SI	SI
Predicción de otros eventos de un partido (tarjetas, corners, etc)	SI	SI	SI	SI	SI
Combinación entre la predicción y la cuota, para beneficio económico	SI	NO	NO	SI	SI

4. Introducción al aprendizaje automático

El presente capítulo tiene como objetivo ofrecer una exposición detallada de los fundamentos teóricos y conceptuales que se han empleado en el desarrollo del algoritmo de aprendizaje automático desarrollado durante este proyecto.

En primer lugar, se describirán las librerías usadas para el procesado y tratamiento de los datos, como Pandas, que es una herramienta fundamental para este proceso. Se explicarán las principales características de esta librería, y se mostrarán ejemplos concretos de su aplicación en el proyecto.

En segundo lugar, se describirán los entornos de desarrollo utilizados en este proyecto. Se hará especial énfasis en el uso del cuaderno de Kaggle, que es una herramienta online que permite el desarrollo y ejecución de código en Python.

En tercer lugar, se explicarán dos algoritmos ampliamente utilizados en el aprendizaje automático: la regresión lineal y la regresión logística. La regresión lineal se utiliza para modelar la relación entre una variable de entrada y una variable de salida continua, encontrando la mejor línea recta que se ajusta a los datos. Es una herramienta útil en problemas de predicción numérica.

Por otro lado, la regresión logística se utiliza para calcular probabilidades y realizar clasificaciones. Este algoritmo, que es implementado a través de la biblioteca sklearn, se utiliza en problemas de clasificación binaria o de múltiples clases, en nuestro contexto, lo usamos para predecir las probabilidades que tiene cada equipo de ganar el partido. Ambos algoritmos son fundamentales en el campo

del aprendizaje automático y permiten realizar predicciones y tomar decisiones basadas en datos.

4.1 Librerías Python

4.1.1 Pandas

La manipulación de datos es un pilar fundamental para el mundo de la ciencia, el análisis y el procesamiento de los datos. Pandas es una de las bibliotecas más populares y utilizadas en el lenguaje de programación Python para el análisis y la manipulación de datos en bruto, proporcionando herramientas para la limpieza, transformación y exploración de datos de una manera bastante sencilla.

Pandas está diseñada para trabajar con datos de tipo tabular o de hoja de cálculo, como se encuentran en bases de datos relacionales o en archivos CSV. La librería pandas se ha convertido en un elemento clave en el kit de herramientas de cualquier científico de datos o analista de datos debido a su facilidad de uso y flexibilidad.

La principal estructura de datos de pandas es el DataFrame, que es una tabla 2D que contiene una serie de columnas, cada una de las cuales puede contener un tipo de datos diferente, como números, fechas o cadenas. Además, Pandas también proporciona la estructura de datos de la Serie, que es una matriz unidimensional etiquetada que puede contener cualquier tipo de datos.

Entre las ventajas de Pandas se encuentran su facilidad de uso y velocidad en la manipulación y limpieza de grandes conjuntos de datos, así como en la agregación, transformación y análisis de datos. Con pandas, se pueden realizar fácilmente operaciones como unir, dividir y filtrar datos, y también cuenta con una gran cantidad de funciones integradas para el análisis y la manipulación de datos. Además, pandas también tiene integración con otras librerías de Python, como NumPy y Matplotlib, lo que permite realizar análisis y visualizaciones de datos más complejos.

Sin embargo, una de las desventajas de Pandas es que no es óptimo para manejar grandes conjuntos de datos debido a su uso intensivo de memoria. Además, puede ser un poco complicado para los usuarios nuevos debido a la cantidad de funciones y métodos que ofrece, lo que puede llevar a confusiones en su uso.

Además de Pandas, existen otras bibliotecas de Python que se utilizan para el análisis y manipulación de datos, como NumPy y SciPy. NumPy proporciona un conjunto de herramientas para trabajar con matrices y arrays multidimensionales, lo que lo convierte en una buena opción para cálculos numéricos y operaciones matemáticas complejas. Sin embargo, a diferencia de Pandas, no ofrece funciones específicas para el análisis de datos, como la manipulación de tablas o la selección de datos basada en etiquetas.

En resumen, Pandas es una librería de Python diseñada para trabajar con datos de tipo tabular o de hoja de cálculo. Proporciona estructuras de datos de alto rendimiento y herramientas de análisis de datos, como el DataFrame y la Serie, y es ampliamente utilizada en la comunidad de ciencia de datos debido a su facilidad de uso y flexibilidad. Aunque no es óptima para manejar grandes conjuntos de datos y puede ser un poco complicada para los usuarios nuevos, Pandas ofrece una amplia gama de funciones y métodos que permiten la manipulación y análisis de datos de manera eficiente.

4.1.2 Sklearn (scikit-learn)

La biblioteca de Python scikit-learn, también conocida como sklearn, es una de las herramientas más importantes para el análisis de datos y el aprendizaje automático. Esta biblioteca de software libre es compatible con otros paquetes de Python como NumPy, SciPy y matplotlib. El objetivo de sklearn es proporcionar una colección de herramientas para el modelado estadístico y la minería de datos de una manera fácil de usar y accesible. Es una biblioteca muy utilizada por los científicos de datos, investigadores y estudiantes de aprendizaje automático debido a su facilidad de uso y su amplia gama de funcionalidades.

La biblioteca scikit-learn ofrece una gran cantidad de algoritmos de aprendizaje automático para realizar tareas de clasificación, regresión, agrupamiento y reducción de dimensionalidad. Incluye también herramientas para la selección de características, la normalización de datos y la validación cruzada. Además, sklearn proporciona implementaciones de algoritmos de aprendizaje

automático populares, como SVM, Random Forest, K-Means, Naive Bayes, Gradient Boosting, entre otros.

La estructura de la biblioteca se basa en una serie de clases y funciones que se combinan en flujos de trabajo de aprendizaje automático. El usuario puede seleccionar y configurar las diferentes herramientas que desee utilizar en cada fase del proceso. Por ejemplo, puede seleccionar un modelo de aprendizaje automático, una técnica de selección de características, un método de validación cruzada y una medida de evaluación de rendimiento. La biblioteca permite personalizar cada uno de estos elementos para adaptarse a las necesidades específicas del usuario.

Sklearn es muy versátil y se puede utilizar en diferentes contextos, desde el ámbito educativo hasta el ámbito industrial. Se ha convertido en una herramienta popular en el aprendizaje automático debido a su facilidad de uso, documentación clara y completa, y la capacidad de integrarse con otras bibliotecas de Python. Además, su amplia variedad de algoritmos de aprendizaje automático, técnicas de preprocesamiento de datos y herramientas de evaluación de rendimiento la convierten en una biblioteca imprescindible para cualquier análisis de datos y proyecto de aprendizaje automático.

Sin embargo, una de las principales desventajas de sklearn es que está diseñada para trabajar con datos estructurados, como matrices o data frames, y no es la mejor opción para tareas de procesamiento de lenguaje natural o para datos no estructurados. Además, aunque sklearn proporciona una gran cantidad de herramientas y algoritmos, a veces es necesario utilizar bibliotecas adicionales para

tareas más específicas. A pesar de estas limitaciones, sklearn sigue siendo una biblioteca muy útil y popular en la comunidad de aprendizaje automático.

4.2 Kaggle

Los entornos de desarrollo son una de las herramientas más importantes durante el desarrollo. Un buen entorno de desarrollo, así como una buena preparación y configuración del mismo puede facilitar mucho la realización de una tarea ya que estos proporcionan un conjunto de utilidades y funcionalidades para que el programador pueda trabajar de manera eficiente y productiva. Además, suelen incluir herramientas de depuración y pruebas, facilitando la tarea de encontrar errores en el código y optimizar su rendimiento.

Kaggle es una plataforma de aprendizaje automático que permite a los científicos de datos y desarrolladores de software trabajar en proyectos de manera colaborativa y compartir soluciones. El cuaderno de Kaggle es un entorno de desarrollo en línea que permite a los usuarios crear, compartir y ejecutar código de Python en un entorno interactivo. El cuaderno de Kaggle viene con un conjunto de librerías preinstaladas, incluyendo Pandas, NumPy, Matplotlib, y Scikit-learn, que son herramientas esenciales para el análisis y procesamiento de datos y el aprendizaje automático. También ofrece la posibilidad de utilizar una GPU para entrenar modelos de aprendizaje profundo, lo que puede ser de gran ayuda para proyectos que requieren un gran poder computacional.

Kaggle se utiliza ampliamente para competencias de ciencia de datos, donde los participantes compiten por resolver un problema de aprendizaje automático. El

cuaderno permite a los usuarios cargar conjuntos de datos, analizarlos, visualizarlos y entrenar modelos. Una de las grandes ventajas de trabajar con el cuaderno de Kaggle es la facilidad de compartir y colaborar con otros usuarios. Los cuadernos se pueden compartir en línea con un enlace público, lo que permite a cualquier persona revisar y reproducir el trabajo.

4.3 Algoritmo de regresión lineal

El algoritmo de regresión lineal es una herramienta fundamental en el análisis de datos y la predicción de variables. Su funcionamiento se basa en la suposición de que existe una relación lineal entre una variable dependiente y una o más variables independientes. Esta relación lineal se representa mediante una ecuación matemática en la que los coeficientes de regresión determinan la pendiente y la intersección de la línea recta que mejor se ajusta a los datos observados.

La utilidad de utilizar un algoritmo de regresión lineal como modelo de predicción radica en su simplicidad y capacidad para interpretar los resultados. El modelo permite analizar cómo cada variable independiente afecta a la variable dependiente y proporciona una estimación cuantitativa de la relación entre ellas. Además, el modelo puede utilizarse para realizar predicciones de valores futuros de la variable dependiente en función de los valores de las variables independientes.

El algoritmo de regresión lineal se beneficia de varios supuestos importantes. En primer lugar, se asume que existe una relación lineal entre las variables involucradas. Además, se supone que los errores de predicción tienen una

distribución normal y que tienen una varianza constante. Estos supuestos permiten obtener estimaciones precisas de los coeficientes y realizar inferencias estadísticas sobre los mismos.

Para aplicar el algoritmo de regresión lineal, se deben seguir ciertos pasos. En primer lugar, se recopilan los datos de las variables dependientes e independientes. A continuación, se realiza una exploración y análisis de los datos para identificar posibles relaciones lineales y detectar valores atípicos o datos faltantes. Luego, se selecciona la forma funcional del modelo y se estiman los coeficientes utilizando técnicas como el método de mínimos cuadrados. Posteriormente, se evalúa la calidad del ajuste del modelo mediante medidas como el coeficiente de determinación y se realizan pruebas de significancia sobre los coeficientes para determinar su relevancia estadística.

El algoritmo de regresión lineal es ampliamente utilizado en diversas disciplinas, como la economía, la sociología, la ingeniería y las ciencias médicas. Se aplica en estudios de mercado para predecir el comportamiento de los consumidores, en análisis financiero para estimar el rendimiento de inversiones y en estudios epidemiológicos para investigar la relación entre factores de riesgo y enfermedades. Además, se utiliza como punto de partida para modelos más complejos y sofisticados, como la regresión lineal múltiple y los modelos de regresión no lineal.

En resumen, el algoritmo de regresión lineal es una herramienta valiosa y ampliamente utilizada para la predicción de variables y el análisis de relaciones lineales entre variables.

4.4 Algoritmo de regresión logística

El algoritmo de regresión logística es una poderosa técnica utilizada en el análisis de datos y la clasificación de variables binarias. Su funcionamiento se basa en el principio de maximizar la probabilidad de que una determinada observación pertenezca a una de las clases posibles.

La utilidad de utilizar un algoritmo de regresión logística radica en su capacidad para modelar relaciones no lineales entre las variables independientes y la probabilidad de pertenecer a una clase. A diferencia de la regresión lineal, que busca ajustar una línea recta, la regresión logística utiliza una función logística para modelar la relación entre las variables independientes y la probabilidad de pertenencia a una clase.

El algoritmo de regresión logística se beneficia de varios supuestos importantes. En primer lugar, se asume que la relación entre las variables independientes y la probabilidad de pertenecer a una clase es logarítmica. Además, se supone que las observaciones son independientes entre sí y que no existe multicolinealidad entre las variables independientes. Estos supuestos permiten obtener estimaciones precisas de los coeficientes de regresión logística y realizar inferencias estadísticas sobre los mismos.

La regresión logística es particularmente útil cuando se trata de problemas de clasificación binaria, donde se desea predecir la probabilidad de pertenecer a una clase o a otra. El modelo de regresión logística calcula estas probabilidades utilizando los coeficientes estimados y la función logística. Estas probabilidades pueden ser utilizadas para tomar decisiones, establecer umbrales de clasificación y evaluar la confiabilidad de las predicciones.

El algoritmo de regresión logística es ampliamente utilizado en campos como la medicina, la bioinformática, la ciencia de datos y la inteligencia artificial. Se aplica en problemas de clasificación, como la detección de enfermedades, el análisis de sentimientos, la detección de spam y la predicción de clientes potenciales. Además, se utiliza como punto de partida para modelos más complejos y sofisticados, como la regresión logística multinomial y los modelos de clasificación no lineal.

En resumen, el algoritmo de regresión logística es una herramienta valiosa y ampliamente utilizada en el análisis y clasificación de variables binarias. Su capacidad para modelar relaciones logarítmicas y predecir probabilidades de pertenecer a una clase o a otra lo convierte en una elección popular en diversas aplicaciones de la ciencia de datos.

5. Herramientas para el desarrollo

En cuanto al desarrollo de nuestra aplicación, es importante destacar que este será realizado en Flutter, un framework de código abierto desarrollado por Google que permite crear aplicaciones nativas para Android e iOS de manera rápida

y sencilla. Flutter utiliza el lenguaje de programación Dart, que es altamente escalable y ofrece una sintaxis limpia y fácil de entender, lo que hace que la programación en Flutter sea accesible para desarrolladores de diferentes niveles de experiencia.

Por otro lado, se utilizará SQLite como base de datos para nuestro proyecto y el programa DBeaver para la gestión de la misma. En cuanto a la estructura de una base de datos SQLite, se basa en los principios fundamentales de las bases de datos relacionales. Una base de datos SQLite está compuesta por una o más tablas, que a su vez están formadas por filas y columnas. Cada tabla representa una entidad específica y las filas representan las instancias o registros de esa entidad. Las columnas definen los atributos o propiedades de la entidad y almacenan los valores correspondientes.

DBeaver es una herramienta de administración de bases de datos que permite trabajar con diferentes sistemas de gestión de bases de datos, incluido SQLite. Proporciona una interfaz gráfica intuitiva y funcionalidades avanzadas para interactuar con bases de datos SQLite. A través de DBeaver, los usuarios pueden realizar tareas como la creación de bases de datos, diseño de tablas, ejecución de consultas SQL, importación y exportación de datos, entre otras operaciones.

Para interactuar con una base de datos SQLite desde una aplicación, se puede utilizar una API. Estas APIs permiten establecer una conexión con la base de datos SQLite, ejecutar consultas SQL, realizar operaciones de manipulación de datos y manejar transacciones. Con la ayuda de la API en Node.js, se puede crear,

leer, actualizar y eliminar registros en la base de datos SQLite desde tu aplicación, lo que te brinda flexibilidad y control sobre los datos almacenados.

Tanto la API como la base de datos, ha sido desarrollada como ya hemos mencionado con anterioridad, por mi compañero Alberto Gallego Huerta, quien refleja en el capítulo de la memoria de su proyecto, con mayor nivel de detalle, cómo ha sido desarrollado toda la parte del backend común de las aplicaciones, tanto nuestra aplicación Flutter como su aplicación web.

5.1 Flutter

Flutter es un framework de código abierto para el desarrollo de aplicaciones móviles, web y de escritorio, desarrollado por Google. Flutter utiliza el lenguaje de programación Dart, que fue creado por Google, y se caracteriza por su alto rendimiento y por permitir el desarrollo de interfaces de usuario muy atractivas y personalizadas.

La principal ventaja de Flutter es su capacidad para desarrollar aplicaciones de alta calidad y alto rendimiento en múltiples plataformas. Flutter utiliza un motor de renderizado personalizado llamado Skia para renderizar la interfaz de usuario y lograr un rendimiento suave y rápido. Además, Flutter permite la creación de widgets personalizados, lo que significa que los desarrolladores pueden diseñar interfaces de usuario únicas y atractivas sin tener que depender de bibliotecas de terceros.

Otra ventaja importante de Flutter es la velocidad y la facilidad de desarrollo. Flutter utiliza el compilador AOT (Ahead of Time) para compilar el código a lenguaje de máquina nativo, lo que significa que las aplicaciones Flutter pueden arrancar rápidamente y tener un tiempo de respuesta rápido. Además, Flutter tiene una amplia gama de widgets integrados y herramientas de depuración que facilitan el desarrollo de aplicaciones y reducen el tiempo de desarrollo.

Por otro lado, una posible desventaja de Flutter es que todavía es un framework relativamente nuevo y puede haber una curva de aprendizaje para los desarrolladores que no estén familiarizados con el lenguaje de programación Dart. Además, como Flutter es un framework completo, puede haber una sobrecarga en el tamaño de la aplicación resultante.

En resumen, Flutter es un framework de desarrollo de aplicaciones móviles, web y de escritorio de alto rendimiento y rápido desarrollo que permite a los desarrolladores crear interfaces de usuario personalizadas y atractivas en múltiples plataformas. Aunque todavía es relativamente nuevo, sus ventajas en cuanto a rendimiento y facilidad de desarrollo hacen que sea una opción popular entre los desarrolladores de aplicaciones.

5.2 SQLite y DBeaver

SQLite es un sistema de gestión de bases de datos relacional que ofrece una amplia gama de utilidades y funcionalidades para los ingenieros de software. Permite almacenar y organizar grandes volúmenes de datos de manera eficiente y

confiable. Una de las principales ventajas de SQLite es su simplicidad y su capacidad para funcionar sin un servidor dedicado, lo que lo hace adecuado para aplicaciones de escritorio y móviles.

En el funcionamiento de SQLite en DBeaver, se puede interactuar con la base de datos utilizando una interfaz gráfica intuitiva que permite realizar tareas como la creación de tablas, la modificación de esquemas, la ejecución de consultas SQL y la visualización de resultados. DBeaver proporciona herramientas y asistentes que facilitan la administración de la base de datos, como la importación y exportación de datos en varios formatos, la generación de scripts y la generación de informes.

Una de las utilidades de SQLite en DBeaver es su capacidad para soportar consultas SQL complejas y optimizadas. Esto permite a los ingenieros de software realizar operaciones de filtrado, ordenación y agrupación de datos de manera eficiente, lo que resulta en consultas rápidas y resultados precisos. Además, SQLite ofrece funciones integradas y extensiones que permiten realizar cálculos y transformaciones en los datos, lo que facilita la manipulación de la información almacenada.

Otra utilidad importante de SQLite en DBeaver es su capacidad para trabajar con transacciones. Las transacciones permiten agrupar varias operaciones en una sola unidad lógica, lo que garantiza la integridad y la consistencia de los datos. Esto es especialmente útil en aplicaciones donde se requiere mantener la integridad de los datos, como sistemas de control de inventario o aplicaciones financieras.

En resumen, SQLite en DBeaver ofrece un conjunto de herramientas y funcionalidades que facilitan la administración y el manejo de bases de datos. Asimismo, permite almacenar, organizar y consultar datos de manera eficiente y confiable. Su simplicidad y su capacidad de funcionar sin un servidor dedicado lo convierten en una opción popular para aplicaciones de escritorio y móviles.

5.3 NodeJS

Una API construida con Node.js es una poderosa herramienta, ya que permite crear y ofrecer servicios web de manera eficiente y escalable. Node.js es un entorno de tiempo de ejecución basado en JavaScript que utiliza un modelo de operaciones asincrónicas y sin bloqueo, lo que lo hace especialmente adecuado para aplicaciones de red y servidores.

El funcionamiento de una API en Node.js se basa en la capacidad de Node.js para manejar múltiples conexiones simultáneamente y de forma no bloqueante. Esto significa que puede procesar solicitudes de manera eficiente sin bloquear la ejecución de otras tareas. Utilizando módulos y bibliotecas de Node.js, los ingenieros de software pueden implementar rutas, controladores y lógica de negocio para manejar las solicitudes entrantes y enviar respuestas adecuadas.

Una de las utilidades clave de utilizar una API de Node.js es su capacidad para proporcionar servicios web altamente escalables. Node.js utiliza un enfoque basado en eventos y un bucle de eventos para gestionar las solicitudes de manera eficiente, lo que permite manejar grandes volúmenes de tráfico y escalar fácilmente

el sistema. Esto es especialmente útil en entornos donde se requiere un rendimiento y una capacidad de respuesta excepcionales, como aplicaciones en tiempo real, aplicaciones de chat o sistemas de alta concurrencia.

Además, Node.js ofrece una amplia gama de módulos y bibliotecas que facilitan el desarrollo de API robustas. Estos módulos proporcionan funcionalidades adicionales, como manejo de autenticación y autorización, integración con bases de datos, almacenamiento en caché, envío de correos electrónicos y más.

Otra ventaja de utilizar una API de Node.js es su compatibilidad con tecnologías web modernas. Node.js es compatible con protocolos web estándar como HTTP y WebSocket, lo que permite la implementación de servicios web tradicionales y de tiempo real. También es compatible con formatos de intercambio de datos como JSON, lo que facilita la comunicación con clientes y aplicaciones front-end.

5.4 Visual Studio Code

Visual Studio Code (VS Code) es un entorno de desarrollo integrado (IDE) desarrollado por Microsoft. VS Code es una aplicación de escritorio que se puede instalar en cualquier sistema operativo. Es compatible con varios lenguajes de programación, incluyendo Python, y ofrece una amplia gama de extensiones para personalizar y mejorar la funcionalidad.

Una de las grandes ventajas de trabajar con VS Code es su flexibilidad y la capacidad de personalizar la experiencia de desarrollo. Los usuarios pueden instalar y desinstalar extensiones según sea necesario para trabajar con diferentes lenguajes de programación o para añadir funcionalidades adicionales. Además, VS Code ofrece herramientas avanzadas de depuración y pruebas, así como integración con herramientas de control de versiones como Git.

VS Code también se utiliza ampliamente para el desarrollo de aplicaciones móviles con frameworks como Flutter. Flutter es un kit de herramientas de desarrollo de aplicaciones móviles de código abierto creado por Google. VS Code se integra perfectamente con Flutter, permitiendo a los desarrolladores compilar, depurar y probar aplicaciones en un entorno de desarrollo integrado. Los desarrolladores pueden también utilizar extensiones de VS Code para trabajar con otras herramientas populares de Flutter, como el administrador de paquetes de Dart y Flutter.

Además, VS Code ha sido utilizado como IDE en el desarrollo de nuestra API en Node.js. Esto nos ha facilitado la escritura de código, la depuración y la gestión de paquetes, permitiéndonos así un desarrollo mucho más eficiente y rápido.

6. Datos utilizados

En este capítulo se describe el proceso de recopilación y obtención de los datos utilizados en el proyecto. Los datos son una parte fundamental de cualquier proyecto de aprendizaje automático, y recopilar y procesar datos puede ser una tarea compleja y que requiere mucho tiempo. En este caso utilizaremos un conjunto de datos con los partidos de la liga española de fútbol desde la temporada 1995-1996 hasta la temporada 2020-2021 obtenidos en formato csv a través de la plataforma Kaggle. A lo largo del capítulo, se describirán las diferentes etapas del proceso de adquisición y procesamiento de datos, junto con los problemas que han ido surgiendo y las soluciones adoptadas.

Una vez obtenidos este conjunto de datos iniciales, es necesario realizar procesos de limpieza y preprocesamiento para que puedan ser utilizados para el análisis y construcción de modelos de aprendizaje automático. Además, a partir de los este conjunto de datos inicial y mediante distintas funciones auxiliares se crearán las variables que consideramos como más relevantes para el análisis de los partidos y que serán la entrada de nuestro modelo de aprendizaje automático

6.1 Obtención y preprocesado de los datos

En este proceso de preprocesamiento de datos hemos utilizado la librería Pandas, la cual es de gran ayuda para la manipulación y análisis de datos. Pandas es una herramienta de manipulación y análisis de datos de código abierto que proporciona estructuras de datos flexibles y eficientes para trabajar con datos estructurados. Gracias al uso de esta biblioteca, es posible realizar diferentes tareas de preprocesamiento, como limpieza, transformación y selección de datos relevantes para el análisis de una manera mucho más sencilla e intuitiva.

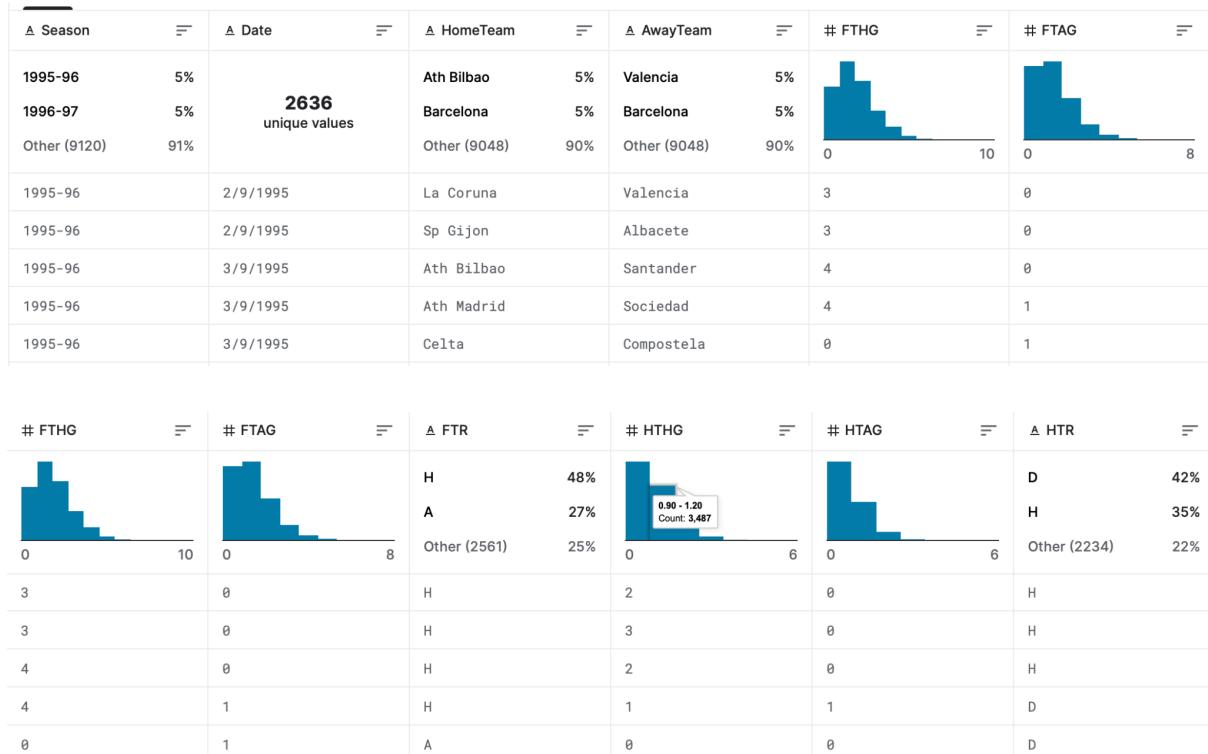
El propósito de este capítulo es proporcionar una descripción general del proceso de adquisición y preprocesamiento de datos utilizado en el proyecto, destacando las dificultades encontradas y las soluciones para superarlas. Asimismo, pretende mostrar cómo personalizar un conjunto de datos primitivo para obtener las variables deseadas, cómo por ejemplo los goles a favor marcados por un equipo en los últimos cinco partidos disputados.

6.1.1 Análisis del dataset primitivo

El conjunto de datos inicial ha sido sacado de la plataforma Kaggle. Consta de 10 columnas con información relevante acerca de los goles marcados durante el partido completo, los goles marcados en la primera parte y el equipo que acabó llevándose la victoria en cada partido disputado en la liga española de fútbol desde la temporada 1995-1996. Dichas columnas son:

- *Season*: temporada en la que se disputa el partido.
- *Date*: fecha exacta en la que se disputa el partido.
- *HomeTeam*: equipo que juega como local, es decir, que disputa el partido en su estadio.
- *AwayTeam*: equipo que juega como visitante, es decir, que disputa el partido en el campo del rival.
- *FTHG*: número de goles marcado por el equipo local tras la finalización del partido
- *FTAG*: número de goles marcados por el equipo visitante tras la finalización del partido.
- *HTHG*: número de goles marcados por el equipo local durante la primera mitad del partido.
- *HTAG*: número de goles marcados por el equipo visitante durante la primera mitad del partido.
- *FTR*: indica el ganador del partido. Tomará el valor ‘H’ si el equipo local consiguió la victoria, ‘A’ si el visitante fue el que logró la victoria y ‘D’ si el partido finalizó con empate.

Aquí podemos apreciar la apariencia de las primeras líneas de este dataset primitivo, el cual utilizaremos como base para calcular los parámetros que deseamos:



6.1.2 Creación de funciones auxiliares

Una vez analizados e importados dichos datos, procedimos a operar con ellos. Nuestro objetivo era darle importancia al estado de forma en el que llegaba cada equipo a un partido, por tanto, nuestro dataset objetivo era muy distinto a este conjunto de datos extraído de kaggle y han sido necesarias realizar diversas funciones auxiliares para poder calcular los parámetros deseados y que serán las entradas a nuestro modelo de aprendizaje.

Hemos conseguido elaborar un dataset personalizado con los parámetros que nosotros queríamos a partir del dataset inicial sacado de Kaggle, con ayuda de diversas funciones que nos permiten calcular parámetros bastante útiles para poder realizar un buen análisis a priori de un partido.

Lo primero que hemos calculado han sido los goles a favor anotados por un equipo en los últimos cinco partidos disputados a una fecha dada. Es decir, si el Real Madrid jugase un partido el 21 de octubre de 2022, esta función nos devolverá el número de goles anotados por el conjunto madrileño en los últimos cinco partidos de liga anteriores a esa fecha. Primero filtramos los partidos y nos quedamos solo en los que juegue el Real Madrid, tengan fecha anterior a la fecha dada y que pertenezcan a la misma temporada a la que pertenece la fecha por la que estamos filtrando. Después ordenaremos esos partidos en orden descendente y nos quedamos con los cinco primeros elementos de dicha lista de partidos que van a corresponder con los cinco últimos partidos jugados por ese equipo. Por último, comprobamos si el Real Madrid ejercía como local o como visitante para ir sumando a nuestra variable resultado los goles del equipo local o del visitante. Es decir, si el Real Madrid jugaba como local, sumaremos el valor de la columna ‘FTHG’ (goles del equipo local al final del partido) y si ejercía como visitante, la columna ‘FTAG’ (goles del equipo visitante al final del partido). A continuación se muestra el fragmento de código de que calcula los goles anotados por un equipo en los últimos cinco partidos, pertenece a la función “goles_a_favor_ultimos_cinco”:

```

partidos_equipo = df[((df['HomeTeam'] == equipo) | (df['AwayTeam'] == equipo)) & (df['Date'] < fecha) & (df['Season'] == temporada)]
partidos_ordenados = partidos_equipo.sort_values('Date', ascending=False)
partidos_ultimos_cinco = partidos_ordenados[(partidos_ordenados['HomeTeam'] == equipo) | (partidos_ordenados['AwayTeam'] == equipo)].head(5)
goles = 0
for i, partido in partidos_ultimos_cinco.iterrows():
    if partido['HomeTeam'] == equipo:
        goles += partido['FTHG']
    elif partido['AwayTeam'] == equipo:
        goles += partido['FTAG']
return goles

```

Para conseguir los goles que le han encajado a un equipo en los últimos cinco partidos, seguimos el mismo procedimiento para conseguir los partidos jugados por ese equipo. La diferencia estará en que cuando el equipo juegue de

local, nosotros contabilizaremos los goles que ha marcado el visitante, es decir, los goles que le han marcado al equipo local y si juega como visitante, pues sumaremos los goles que ha marcado el local. El siguiente trozo de código pertenece a la función “goles_en_contra_ultimos_cinco”:

```
'partidos_equipo = df[((df['HomeTeam'] == equipo) | (df['AwayTeam'] == equipo)) & (df['Date'] < fecha) & (df['Season'] == temporada)]
partidos_ordenados = partidos_equipo.sort_values('Date', ascending=False)
partidos_ultimos_cinco = partidos_ordenados[(partidos_ordenados['HomeTeam'] == equipo) | (partidos_ordenados['AwayTeam'] == equipo)].head(5)
goles = 0
for i, partido in partidos_ultimos_cinco.iterrows():
    if partido['HomeTeam'] == equipo:
        goles += partido['FTAG']
    elif partido['AwayTeam'] == equipo:
        goles += partido['FTHG']
return goles
```

De formas parecidas a las anteriores sacamos los goles anotados y recibidos durante las primeras partes. Estas funciones son las que nos van a permitir poder obtener información relevante sobre el marcador esperado al término de la primera mitad. Cogeremos los valores de las columnas ‘HTHG’ (goles del equipo local en la primera parte) y ‘HTAG’ (goles del equipo visitante en la primera mitad). Repetiremos el mismo proceso anterior si queremos conseguir los goles a favor o en contra de un equipo. Es decir, tendremos una función para calcular los goles a favor de un equipo, por lo que si juega en casa sumaremos el valor de la columna ‘HTHG’ y si juega como visitante, el valor de ‘HTAG’. “goles_a_favor_descanso_ultimos_cinco” es el nombre de la función encargada de realizar este cálculo:

```
'partidos_equipo = df[((df['HomeTeam'] == equipo) | (df['AwayTeam'] == equipo)) & (df['Date'] < fecha) & (df['Season'] == temporada)]
partidos_ordenados = partidos_equipo.sort_values('Date', ascending=False)
partidos_ultimos_cinco = partidos_ordenados[(partidos_ordenados['HomeTeam'] == equipo) | (partidos_ordenados['AwayTeam'] == equipo)].head(5)
goles = 0
for i, partido in partidos_ultimos_cinco.iterrows():
    if partido['HomeTeam'] == equipo:
        goles += partido['HTHG']
    elif partido['AwayTeam'] == equipo:
        goles += partido['HTAG']
return goles
```

Semejante procedimiento para calcular los goles encajados en la primera mitad pero esta vez, si el equipo juega como local sumaremos los goles del visitante y viceversa, ya que lo que queremos calcular son los goles en contra y lo hacemos con ayuda de la función `goles_en_contra_ultimos_cinco`:

```
partidos_equipo = df[((df['HomeTeam'] == equipo) | (df['AwayTeam'] == equipo)) & (df['Date'] < fecha) & (df['Season'] == temporada)]
partidos_ordenados = partidos_equipo.sort_values('Date', ascending=False)
partidos_ultimos_cinco = partidos_ordenados[(partidos_ordenados['HomeTeam'] == equipo) | (partidos_ordenados['AwayTeam'] == equipo)].head(5)
goles = 0
for i, partido in partidos_ultimos_cinco.iterrows():
    if partido['HomeTeam'] == equipo:
        goles += partido['HTAG']
    elif partido['AwayTeam'] == equipo:
        goles += partido['HTHG']
return goles
```

Otra función que hemos realizado y que es un dato realmente útil para realizar una buena valoración previa de un partido de fútbol son los puntos conseguidos por cada equipo en los últimos cinco partidos jugados. Este es uno de los valores más útiles ya que en él se puede ver reflejado el estado de forma en el que llega un equipo a un partido y eso supone un factor diferencial a la hora de analizar un encuentro.

Para el cálculo de los puntos conseguidos por un equipo en las últimas cinco jornadas, nos hemos ayudado de la función “`puntos_ultimos_cinco`” que toma como parámetros de entrada un equipo y una fecha y devuelve los puntos obtenidos por dicho equipo en los últimos cinco partidos jugados antes de esa fecha.

```

partidos_equipo = df[((df['HomeTeam'] == equipo) | (df['AwayTeam'] == equipo)) & (df['Date'] < fecha) & (df['Season'] == temporada)]
partidos_ordenados = partidos_equipo.sort_values(['Date'], ascending=[False])
partidos_ultimos_cinco = partidos_ordenados[(partidos_ordenados['HomeTeam'] == equipo) | (partidos_ordenados['AwayTeam'] == equipo)].head(5)
puntos = 0
for i, partido in partidos_ultimos_cinco.iterrows():
    if partido['HomeTeam'] == equipo:
        if(partido['FTR'] == 'H'):
            puntos += 3
        elif(partido['FTR'] == 'D'):
            puntos += 1
        elif(partido['FTR'] == 'A'):
            puntos += 0
    elif partido['AwayTeam'] == equipo:
        if(partido['FTR'] == 'A'):
            puntos += 3
        elif(partido['FTR'] == 'D'):
            puntos += 1
        elif(partido['FTR'] == 'H'):
            puntos += 0
    else:
        puntos += 0
return puntos

```

Para conseguir los cinco últimos partidos jugados por dicho equipo, hemos realizado el mismo procedimiento que el que hemos descrito con anterioridad y que ha sido usado ya para calcular los goles a favor tras la finalización de los últimos cinco partidos o los goles en contra. Una vez teníamos la lista de los últimos cinco partidos jugados, nos quedaba comprobar el valor que tomaba la variable ‘FTR’ (resultado al final del partido) y si el equipo que nos habían pasado ejercía como local o como visitante.

En resumen, si nos pedían los puntos conseguidos por el Betis antes del 21 de octubre de 2022, una vez que tengamos la lista de sus últimos partidos jugados, iremos comprobando partido a partido la situación de cada uno. Es decir, si el Betis ejerce de local y la columna ‘FTR’ tiene el valor ‘H’, es decir el local ha sido el ganador del encuentro, entonces sumamos tres puntos a nuestra variable de salida, si la columna ‘HTR’ toma el valor de ‘D’, sumaremos un punto y si es ‘A’ significará que ha ganado el visitante y por tanto no sumaremos nada. En el caso de que el Betis juegue el partido como visitante cambiarán los escenarios en los que tendríamos que sumar los puntos a nuestra variable de salida. Sumaremos tres

puntos si la columna ‘HTR’ toma el valor ‘A’, un punto si toma el valor ‘D’ ya que significaría que habrían empatado y ningún punto si aparece el valor ‘H’.

Por último, la función que tiene mayor complejidad de todas las realizadas pero que a su vez también aporta una mayor información a este análisis preliminar que estamos realizando de un partido es la de la posición que ocupa un equipo en la clasificación en una fecha dada. Esto nos permite no solo comparar y poner en valor la racha y el estado de forma de un equipo sino que además podremos ver el desempeño de dicho equipo a lo largo de toda la temporada y analizar un poco más a largo plazo las diversas actuaciones que ha venido realizando el conjunto esta campaña.

Para calcular la posición de un equipo en la tabla en una fecha concreta, vamos a utilizar dos funciones. La primera, “*clasificacion_en_una_fecha*”, que recibirá como parámetro de entrada una fecha concreta y devolverá un diccionario en el que la clave será el nombre del equipo y el valor la cantidad de puntos que atesore en ese momento.

Esta función será llamada por la función *posicion_equipo_en_una_fecha*, a la que le pasaremos un equipo y una fecha, dicha función llamará a *clasificacion_en_una_fecha*, y una vez obtenida la clasificación en una fecha buscaremos en el diccionario el par clave-valor en el cual la clave coincida con el nombre del equipo que recibe la función *posicion_equipo_en_una_fecha* como parámetro de entrada y devolveremos únicamente el valor, que será la cantidad de puntos que ese equipo ha conseguido en los últimos 5 partidos. Si introducimos una

fecha en la que todavía no se haya disputado ninguna jornada de esa temporada, la posición de todos los equipos será de -1.

La función `clasificacion_en_una_fecha` es sin duda la más compleja y distinta de todas las realizadas. Lo primero que hacemos es filtrar los partidos, nos quedamos solamente con aquellos que se hayan disputado con anterioridad a la fecha dada y que pertenezcan a la misma temporada. Una vez tenemos este conjunto de partidos disputados procedemos a ir asignando los puntos conseguidos a cada equipo. Para ello, creamos un diccionario que será nuestra clasificación final. A partir de aquí vamos recorriendo los partidos y estableciendo los puntos correspondientes a cada equipo. Primero comprobamos que el equipo local y visitante están añadidos en el diccionario y si no es así lo añadimos y le establecemos una puntuación inicial de cero puntos.

Una vez nos aseguramos de que ambos equipos estén dados de alta en el diccionario, analizamos la columna ‘FTR’ para poder saber el resultado del partido y ver quién se lleva los puntos. Si el valor de ‘FTR’ es ‘H’ significa que ha ganado el equipo local y por tanto accederemos al par clave-valor en el cual la clave sea igual que el equipo que ha ganado y le sumaremos tres puntos más. Haríamos lo mismo si el valor de ‘FTR’ es ‘A’ pero accediendo al par clave-valor que le corresponda al equipo visitante. En cambio, si el valor de dicha columna es una ‘D’ significará que habrán empatado y tendremos que sumar un punto a cada equipo. Aquí el fragmento de código encargado de realizar todo el proceso anteriormente mencionado:

```

partidos = df[(df['Date'] < fecha) & (df['Season'] == temporada)]
puntos = {}
for _, partido in partidos.iterrows():
    if partido['HomeTeam'] not in puntos:
        puntos[partido['HomeTeam']] = 0
    if partido['AwayTeam'] not in puntos:
        puntos[partido['AwayTeam']] = 0
    if partido['FTR'] == 'D':
        puntos[partido['HomeTeam']] += 1
        puntos[partido['AwayTeam']] += 1
    elif partido['FTR'] == 'H':
        puntos[partido['HomeTeam']] += 3
    else:
        puntos[partido['AwayTeam']] += 3

clasificacion = sorted(puntos.items(), key=lambda x: x[1], reverse=True)
return clasificacion

```

6.2 Generación del nuevo dataset

El código que se muestra a continuación es el encargado de generar un nuevo dataset llamado "partidos_caracteristicas.csv" a partir de los datos calculados utilizando todas las funciones previas.

```

partidos = df

datos_partidos = []
for _, partido in partidos.iterrows():
    temporada = partido['Season']
    fecha = partido['Date']
    ganador = partido['FTR']
    goles_local = partido['FTHG']
    goles_visitante = partido['FTAG']
    goles_local_primera_parte = partido['HTHG']
    goles_visitante_primera_parte = partido['HTAG']

    # EQUIPO LOCAL
    equipo_local = partido['HomeTeam']
    goles_a_favor_local_ultimos_5 = goles_a_favor_ultimos_cinco(equipo_local, fecha)
    goles_en_contra_local_ultimos_5 = goles_en_contra_ultimos_cinco(equipo_local, fecha)
    puntos_local_ultimos_5 = puntos_ultimos_cinco(equipo_local, fecha)
    posicion_local = posicion_equipo_en_una_fecha(equipo_local, fecha.strftime("%Y-%m-%d"))
    goles_a_favor_primera_parte_local_ultimos_5 = goles_a_favor_descanso_ultimos_cinco(equipo_local, fecha)
    goles_en_contra_primera_parte_local_ultimos_5 = goles_en_contra_descanso_ultimos_cinco(equipo_local, fecha)

    # EQUIPO VISITANTE
    equipo_visitante = partido['AwayTeam']
    goles_a_favor_visitante_ultimos_5 = goles_a_favor_ultimos_cinco(equipo_visitante, fecha)
    goles_en_contra_visitante_ultimos_5 = goles_en_contra_ultimos_cinco(equipo_visitante, fecha)
    puntos_visitante_ultimos_5 = puntos_ultimos_cinco(equipo_visitante, fecha)
    posicion_visitante = posicion_equipo_en_una_fecha(equipo_visitante, fecha.strftime("%Y-%m-%d"))
    goles_a_favor_primera_parte_visitante_ultimos_5 = goles_a_favor_descanso_ultimos_cinco(equipo_visitante, fecha)
    goles_en_contra_primera_parte_visitante_ultimos_5 = goles_en_contra_descanso_ultimos_cinco(equipo_visitante, fecha)

```

El proceso de generación del dataset se lleva a cabo mediante la iteración de cada fila del dataset original "partidos" usando la función `iterrows()`, para así obtener la información necesaria de cada partido.

En primer lugar, se extraen las características básicas de cada partido como la temporada, la fecha, el equipo local y visitante, los goles marcados por cada equipo y los goles marcados en la primera parte por cada equipo. Luego, se generan nuevas características de cada equipo en los últimos 5 partidos, como los goles marcados y encajados, la cantidad de puntos obtenidos y la posición en la tabla en la fecha del partido. Asimismo, se generan características adicionales de los goles marcados en la primera parte en los últimos 5 partidos.

Cada una de las características se obtiene utilizando las funciones previamente definidas, que se encargan de extraer la información necesaria de otros datasets. Por ejemplo, la función `"goles_a_favor_ultimos_cinco(equipo, fecha)"` obtiene la cantidad de goles que ha marcado el equipo dado en los 5 partidos anteriores a la fecha dada.

Posteriormente, se genera una nueva fila de datos para cada partido, que incluye las características previamente mencionadas. Estas características se agregan como columnas al dataset "df_nuevo" utilizando la función `"append"` y el argumento `"ignore_index=True"`, lo que permite que cada nueva fila se agregue al final del dataset.

```
nueva_fila = {'Temporada': temporada, 'Fecha': fecha, 'Ganador': ganador, 'Goles Local': goles_local,
    'EquipoVisitante': equipo_visitante, 'GolesFavV5': goles_a_favor_visitante_ultimos_5}

df_nuevo = df_nuevo.append(nueva_fila, ignore_index=True)
df_nuevo.to_csv('partidos_caracteristicas.csv', index=False)
```

Por último, el dataset "df_nuevo" se exporta a un archivo CSV llamado "partidos_caracteristicas.csv" utilizando la función "to_csv". El argumento "index=False" se utiliza para evitar que se agregue una columna adicional con los índices del dataset.

En resumen, el proceso de generación del dataset llamado "partidos_caracteristicas.csv" implica la iteración de cada partido en el dataset original, la extracción de características de cada equipo en los últimos 5 partidos utilizando funciones previamente definidas, la creación de una nueva fila de datos para cada partido y la exportación del nuevo dataset a un archivo CSV.

6.3 Problemas encontrados

Durante el proceso de tratamiento y limpieza de los datos primitivos para la creación de una modelo de aprendizaje automático que permita predecir resultados de partidos de fútbol, nos encontramos diversos problemas técnicos que tuvimos que ir afrontando y resolviendo de diversas maneras.

Uno de los primeros problemas importantes al que nos tuvimos que enfrentar fue el del formato de las fechas. Al principio éramos capaces de leer y operar con

todos los datos de nuestro dataset primitivo, excepto con las fechas. Rápidamente nos percatamos de que las fechas estaban siendo leídas como un tipo string y por tanto no podíamos realizar comparaciones entre ellas por ejemplo para poder comprobar si una fecha era anterior a otra. Este inconveniente se solucionó añadiendo el argumento “parse_dates” al método read_csv, para que leyera la columna ‘Date’ como un tipo fecha en lugar de tipo string.

```
pd.read_csv('/kaggle/input/la-liga-results-19952020/LaLiga_Matches_1995-2021.csv', parse_dates=['Date'])
```

Una vez conseguimos leer las fechas en formato Date para así poder hacer comparaciones y operaciones con ellas, empezamos a probar el correcto funcionamiento de las distintas funciones. Seguidamente apareció otro problema, ya que cuándo a la función ‘clasificacion_en_una_fecha’ le pedíamos la clasificación del día 25 de junio de 2015 para poder ratificar el correcto funcionamiento de la función, esta arrojaba una clasificación totalmente distinta a la real.

Tras varias comprobaciones y búsquedas del error nos dimos cuenta de que este estaba en el formato de la fecha. El formato de la fecha era yyyy-dd-mm y primero ordenaba por años, después por días y después por meses, esto implicaba que ninguno de nuestros métodos funcionase correctamente ya todos estos métodos tenían una primera parte común que era la de buscar los cinco partidos jugados más recientes, y esta lista de partidos ya era incorrecta. En resumen, si le pedíamos a cualquier función los partidos jugados antes del 2020-10-12 (10 de diciembre de 2021) nos daba los partidos que había jugado el día 9 de cada mes, después el día 8, el día 7, etc.

Esto fue solucionado con otro argumento en el método `read_csv`, “`dayfirst = TRUE`”, que cambia el formato de todas las fechas y entonces estas pasan a tener un formato `yyyy-mm-dd` que nos permitía poder realizar comparaciones correctamente.

Por otro lado, teníamos un problema que afectaba a todas las funciones realizadas hasta el momento, pues a la hora de seleccionar los últimos cinco partidos jugados por un equipo no estábamos comprobando que esos 5 partidos tenían que pertenecer a la misma temporada. Es decir, si buscábamos los últimos partidos jugados antes del 15 de agosto de 2015 (fecha que pertenece a la temporada 2015-16), las funciones estaban cogiendo partidos de mayo o junio (que pertenecen a la temporada 2014-15), ya que al ser una fecha muy temprana de la temporada todavía no se habían podido disputar cinco encuentros.

Para solucionar este conflicto y evitar así que nuestras funciones calculasen datos que no se asemejan a la realidad ni al estado de forma en el que se encuentra dicho equipo (ya que por ejemplo, en la jornada 4 de la temporada 2015-16 no importa nada el resultado obtenido en la última jornada de la temporada 2014-15 a pesar de que es uno de los últimos cinco partidos disputados por ese equipo), decidimos filtrar los partidos por temporadas, es decir si la fecha requerida era posterior al 1 de agosto de ese año, correspondería a una temporada y se era anterior, correspondería a la temporada anterior.

Veámoslo más fácil con el ejemplo anterior: la fecha es 15 de agosto de 2015. Primero vamos a calcular a qué temporada pertenece esa fecha, que como es posterior al 1 de agosto pertenece a la temporada 2015-16, así que una vez que tenemos la temporada sobre la cual estamos haciendo los cálculos vamos a añadir a la hora de filtrar y seleccionar los partidos que pertenezcan sí o sí a dicha temporada. Este proceso de comprobar a qué temporada corresponde una fecha se realiza de la siguiente forma:

```
elif fecha.strptime("%Y-%m-%d") >= f'{year}-08-01':
    temporada = f'{year}-{int(str(year+1)[-2:])}'
else:
    temporada = f'{year-1}-{int(str(year)[-2:])}'
```

Con esto se consiguió el correcto funcionamiento del código hasta que tuvimos que procesar la temporada 99-00. Cómo ya hemos comentado antes, las temporadas tienen un formato de 7 caracteres: 'XXXX-XX', todas excepto una, la temporada 99-00 que tenía el siguiente formato de 9 caracteres: 'XXXX-XXXX'. Esto nos hizo que tuviéramos que hacer un código *ad hoc*, para poder tratar las fechas que pertenecían al año 1999 o al año 2000.

El problema era que si la fecha estaba entre el 1 de agosto de 1999 o el 31 de julio del 2000, la temporada por la que iba a filtrar nuestra función era la 1999-00 pero en nuestro dataset esa temporada no existe, ya que tiene un formato distinto. Para ello teníamos que procesar de una manera distinta el año y hacer coincidir el formato de caracteres de esa temporada con el que aparecía en el dataset.

```

year = int(fecha.strftime("%Y"))

if year == 1999:
    if pd.to_datetime(fecha) >= pd.Timestamp("1999-08-01"):
        temporada = f"{year}-{int(str(year+1)[-4:])}"
    else:
        temporada = f"{year-1}-{int(str(year)[-2:])}"
elif year == 2000:
    if pd.to_datetime(fecha) <= pd.Timestamp("2000-08-01"):
        temporada = f"{year-1}-{int(str(year)[-4:])}"
    else:
        temporada = f"{year}-{int(str(year+1)[-2:])}"

```

Este código hace que si la fecha del año 1999 es posterior al 1 de agosto, se devuelva la fecha con un formato de 9 dígitos, es decir devuelve '1999-2000' y ocurrirá lo mismo si la fecha del año 2000 es anterior al 1 de agosto.

Solucionado este problema de formato, nos enfrentamos ahora a que nuestro dataset generado a partir de la temporada 2000-01 era nulo, pero a partir de la temporada 2009-2010 volvía a funcionar correctamente. Dimos por hecho que era un problema acerca del cálculo de la temporada igual que pasaba anteriormente. En este caso, cuando se ponía una fecha que estuviese entre los años 2001 y 2009, nada funcionaba. Esto se debía a que la temporada se estaba calculando mal otra vez, cuando la fecha era 20 de abril de 2003, la temporada calculada era la 2003-4 y tenía 6 caracteres en lugar de 7 ya que al sumar 03 + 1 devolvía 4 en lugar de 04. Este inconveniente fue solucionado mediante una condición que comprobaba si la temporada tenía 6 caracteres, si era así, era formateada de nuevo.

```

if len(temporada) == 6:
    temporada = f"{temporada[0:4]}-{temporada[-1]}"

```

Todas estas comprobaciones y formateos de fechas se realizan dentro de cada función ya que en todas necesitamos obtener la temporada sobre la que estamos realizando los cálculos para no obtener parámetros erróneos ni valores equivocados.

En resumen, el proceso de filtrado y tratamiento de los datos primitivos para la generación del dataset "partidos_caracteristicas.csv" implica la iteración de cada partido en el dataset original, la extracción de características de cada equipo en los últimos 5 partidos utilizando funciones previamente definidas, la creación de una nueva fila de datos para cada partido y la exportación del nuevo dataset a un archivo CSV. Durante este largo proceso se han ido resolviendo todas las dificultades técnicas encontradas para así lograr unos datos realistas y cuantitativos que posteriormente serán usados para el entrenamiento de nuestro modelo de aprendizaje automático.

7. Experimentación

7.1 Introducción

En el siguiente capítulo, abordaremos el proceso de experimentación con nuestro modelo, profundizando en las distintas pruebas realizadas y las opciones consideradas hasta llegar a la elección final de utilizar los modelos de regresión logística y regresión lineal. Se examinará en detalle la implementación de estos modelos utilizando la biblioteca de aprendizaje automático sklearn, detallando su estructura y funcionamiento.

Además, describiremos minuciosamente el proceso de desarrollo de los modelos, desde el entrenamiento y ajuste utilizando los conjuntos de datos previamente recopilados y procesados, hasta la generación de predicciones de resultados sobre los resultados de los partidos.

En resumen, proporcionará una visión detallada del proceso de experimentación llevado a cabo, permitiendo así comprender en profundidad el desarrollo del modelo y las decisiones tomadas en el camino. A través de valorar exhaustivamente las distintas alternativas consideradas, proporcionaremos argumentos de peso que razonarán el por qué de nuestra elección de los modelos

de regresión logística y regresión lineal. Asimismo, se describirá en detalle el uso de la biblioteca sklearn para desarrollar y entrenar estos modelos, y se explicará el proceso para obtener el archivo CSV que contiene los resultados de las predicciones.

7.2 Dataset de entrenamiento y dataset de predicción

Para realizar un buen entrenamiento a nuestro modelo, primero necesitamos un buen conjunto de datos, que contenga el mayor número de datos útiles posibles.

Nuestro dataset generado, ‘partidos_caracteristicas.csv’ contenía todos los partidos desde la temporada 1995 hasta 2021, pero no habíamos tenido en cuenta que las 5 primeras jornadas de cada temporada aportaba datos no reales, ya que un equipo que iba a disputar la jornada 3 de una temporada, sólo tendría datos de los dos partidos anteriores, no de los cinco últimos (ya que coger los partidos jugados en una temporada anterior no aportaría ningún valor). Para evitar entrenar a nuestro modelo con datos que no fueran reales, decidimos eliminar los primeros 50 partidos disputados de cada temporada, es decir, las primeras 5 jornadas.

Además, de este conjunto de datos que va a ser utilizado para entrenar al modelo, necesitamos eliminar los partidos pertenecientes a la temporada 2020-21, ya que esos partidos van a ser los que le pasaremos posteriormente a nuestro modelo ya entrenado para que nos de una predicción de los mismos.

```

df_nuevo = pd.read_csv('/kaggle/input/datospartidos/partidos_caracteristicas-6.csv', parse_dates=['Fecha'], dayfirst=True)

# Teniendo en cuenta que cada temporada tiene 380 partidos (50 primeras filas de cada temporada, es decir las primeras 5 jornadas)
filas_a_eliminar = []
for i in range(25):
    inicio_temporada = i * 380
    fin_temporada = inicio_temporada + 50
    filas_temporada = list(range(inicio_temporada, fin_temporada))
    filas_a_eliminar.extend(filas_temporada)

#Elimina los partidos de la temporada 20-21 del conjunto de datos de entrenamiento
#ya que es la temporada que vamos a predecir al final.
partidos_2020_2021 = df[(df['Season'] == '2020-21')]

filas_a_eliminar.extend(partidos_2020_2021.index)
df_nuevo = df_nuevo.drop(filas_a_eliminar)
df_nuevo.head()
df_nuevo.to_csv('partidos_caracteristicas_hasta_2019-20.csv', index=False)

```

Con este código obtendremos el dataset de entrenamiento con los datos más limpios posibles, sin las 5 primeras jornadas de cada temporada y sin los partidos de la temporada 2020-21 que es la temporada que vamos a predecir posteriormente.

Por otro lado, necesitamos obtener el dataset con los partidos de la temporada 2020-21, que serán los partidos sobre los cuales vamos a predecir tanto el resultado como la probabilidad de victoria de cada equipo. Para ello filtraremos aquellos partidos que pertenezcan a dicha temporada y posteriormente generaremos un archivo csv con ellos.

```

df_nuevo = pd.read_csv('/kaggle/input/datospartidos/partidos_caracteristicas-6.csv', parse_dates=['Fecha'], dayfirst=True)
partidos_a_predecir = df_nuevo[(df_nuevo['Temporada'] == '2020-21')]
partidos_a_predecir = partidos_a_predecir.drop(partidos_a_predecir.head(50).index)

```

7.3 Elección del modelo

En el proceso de desarrollo de este proyecto, se exploraron diferentes opciones de modelos de aprendizaje automático para abordar el desafío planteado. Entre las alternativas consideradas se encontraban las redes neuronales, la regresión lineal y la regresión logística. Cada uno de estos enfoques ofrece características únicas y potenciales ventajas para la solución del problema.

Las redes neuronales son conocidas por su capacidad de modelar relaciones complejas y aprender patrones a partir de los datos. Estos modelos se basan en la simulación del comportamiento de las neuronas y su interconexión en capas, lo que les permite capturar relaciones no lineales en los datos. Sin embargo, su complejidad computacional y su necesidad de grandes volúmenes de datos pueden limitar su aplicabilidad en algunos escenarios.

Por otro lado, la regresión lineal es un modelo ampliamente utilizado en análisis de datos y proporciona una representación lineal de la relación entre las variables independientes y la variable dependiente. Esta técnica es adecuada cuando se asume una relación lineal entre las variables y puede proporcionar interpretaciones claras y directas de los coeficientes de regresión. Sin embargo, puede ser limitada en su capacidad para modelar relaciones no lineales complejas.

También se consideró la regresión logística, que es un modelo de clasificación binaria que estima la probabilidad de pertenencia a una determinada clase. Este enfoque permite evaluar la influencia de las variables independientes en la probabilidad de ocurrencia de un evento. La regresión logística es especialmente útil cuando se requiere una interpretación probabilística y la capacidad de predecir la pertenencia a una clase específica.

En resumen, se exploraron diferentes opciones de modelos de aprendizaje automático, incluyendo redes neuronales, regresión lineal y regresión logística. Cada uno de estos enfoques presenta sus propias características y ventajas

potenciales. A continuación, se discutirán con más detalle las consideraciones y evaluaciones realizadas para seleccionar el modelo más apropiado para el problema en cuestión.

7.3.1 Redes neuronales

Las redes neuronales son modelos de aprendizaje automático inspirados en el funcionamiento del cerebro humano. Están compuestas por múltiples capas de neuronas interconectadas, cada una de las cuales realiza operaciones matemáticas en los datos de entrada y genera una salida. Estas conexiones entre las neuronas tienen asociados pesos que determinan la importancia relativa de cada conexión en el proceso de aprendizaje.

El funcionamiento de una red neuronal se basa en dos etapas principales: la etapa de propagación hacia adelante (forward propagation) y la etapa de retropropagación del error (backpropagation). Durante la propagación hacia adelante, los datos de entrada se pasan a través de las capas de neuronas, donde se aplican operaciones matemáticas, como multiplicaciones de matrices y funciones de activación, para generar una salida final. Durante la retropropagación del error, se calcula la diferencia entre la salida predicha y el valor objetivo, y se ajustan los pesos de las conexiones en función de este error para mejorar el rendimiento de la red.

El rendimiento de una red neuronal puede verse afectado por varios factores, como el número de capas ocultas y el número de neuronas en cada capa. El número de capas ocultas determina la profundidad de la red y su capacidad para aprender representaciones jerárquicas de los datos. A medida que se aumenta el

número de capas ocultas, la red puede capturar características más abstractas y complejas de los datos, pero también puede aumentar el riesgo de sobreajuste si no se cuenta con suficientes datos de entrenamiento.

Por otro lado, el número de neuronas en cada capa afecta la capacidad de representación y la complejidad de la red. Un mayor número de neuronas puede permitir a la red aprender relaciones más complejas, pero también puede aumentar la complejidad computacional y el riesgo de sobreajuste. Es importante encontrar un equilibrio adecuado entre el número de capas y el número de neuronas para obtener un rendimiento óptimo de la red neuronal.

A pesar de que se realizaron pruebas exhaustivas y se ajustaron diferentes valores y parámetros en la red neuronal, no se logró obtener un rendimiento satisfactorio para abordar el problema en cuestión. Esto puede deberse a la falta de datos de entrenamiento adecuados, la complejidad del problema o la dificultad para encontrar una configuración óptima de la red neuronal.

Se probaron diversas combinaciones de valores y parámetros, se ajustaron el número de capas intermedias, el número de neuronas en cada capa, se cambiaron las funciones de activación de las mismas, se modificó también la función de pérdida, así como el algoritmo de optimización pero no se llegaron a obtener ningún resultado destacable en ninguno de los casos.

```
model = Sequential()

model.add(Dense(14, input_dim=X.shape[1], activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(4, activation='linear'))
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
```

Esta imagen es un ejemplo de una de las muchas configuraciones y pruebas que se han realizado con redes neuronales. Esta concretamente predecía los goles tanto del local como del visitante en el descanso y en el final del partido.

7.3.2 Algoritmos de regresión

Los algoritmos de regresión, como la regresión lineal y la regresión logística, se basan en el análisis de datos históricos y en la identificación de relaciones y patrones entre variables. En nuestro proyecto, se ha aplicado un algoritmo de regresión logística para la predicción del porcentaje de victoria que tiene cada equipo en un partido y un algoritmo de regresión lineal para predecir el resultado al término de la primera parte y al final del partido.

Ambos algoritmos han sido entrenados por nuestro conjunto de datos obtenido anteriormente. Este conjunto está formado por información relevante acerca de todos los partidos disputados desde 1995 hasta la temporada 2019-2020 que ha sido usado para el entrenamiento de los mismos. Posteriormente se ha evaluado la precisión de los mismos con un conjunto de datos compuesto por los partidos de la temporada 2020-2021.

En primer lugar, para predecir los porcentajes de victoria de cada conjunto mediante un algoritmo de regresión logística, se carga el conjunto de datos de

entrenamiento anteriormente mencionado. La columna "Ganador" se convierte a tipo float asignando valores numéricos a las etiquetas de resultado: "D" se convierte en 0, "H" en 1 y "A" en 2. A continuación, se seleccionan las características relevantes que se utilizarán para entrenar el modelo y se almacenan en la variable X, mientras que las etiquetas de resultado se almacenan en la variable y.

```
#Dataset hasta 2019-20
df = pd.read_csv("./kaggle/input/entrenamiento-prediccion/partidos_caracteristicas_hasta_2019-20.csv")

df['Ganador'] = df['Ganador'].map({'D': 0, 'H': 1, 'A': 2}) # Convertir las etiquetas de Ganador a tipo float

# Entrenar el modelo con todos los partidos anteriores
X = df[['GolesFavL5', 'GolesConL5', 'PuntosL5', 'PosicionLocal', 'GolesFavV5', 'GolesConV5', 'PuntosV5', 'PosicionVisitante']]
y = df['Ganador']
```

Luego, se crea una instancia del modelo de regresión logística utilizando la clase LogisticRegression. En este caso, se utiliza el solver 'liblinear', que es un algoritmo eficiente para problemas de regresión logística. A continuación, se entrena el modelo utilizando los datos de entrenamiento mediante el método fit(X, y). Para evaluar la precisión del modelo, se utiliza el método score(X, y), que calcula la precisión media de las predicciones realizadas sobre los datos de entrenamiento. La precisión representa la proporción de muestras clasificadas correctamente. El resultado se almacena en la variable accuracy y se muestra por pantalla.

```
model = LogisticRegression(solver='liblinear')
model.fit(X, y)
accuracy = model.score(X, y)
print(accuracy)
```

Después de entrenar el modelo, se cargan los partidos de la temporada 2020-2021 desde un archivo CSV y se seleccionan las mismas características utilizadas durante el entrenamiento para realizar predicciones sobre estos partidos. Las predicciones se realizan utilizando el método predict_proba(X_pred), que

devuelve las probabilidades de pertenencia a cada una de las clases para cada muestra en X_pred.

```
season_2020_2021 = pd.read_csv("./kaggle/input/entrenamiento-prediccion/partidos_2020-21.csv")
# Seleccionar las columnas de interés
X_pred = season_2020_2021[['GolesFavL5', 'GolesConL5', 'PuntosL5', 'PosicionLocal', 'GolesFavV5', 'GolesConV5']]
pred = model.predict_proba(X_pred)
```

Finalmente, se crea un nuevo DataFrame llamado output_df que contiene información sobre los partidos de la temporada 2020-2021, incluyendo la fecha, los equipos locales y visitantes, así como las probabilidades de victoria local, empate y victoria visitante obtenidas a partir de las predicciones. Los valores de probabilidad se redondean a dos decimales y se convierten a enteros.

```
output_df = pd.DataFrame({
    'Fecha': season_2020_2021['Fecha'],
    'EquipoLocal': season_2020_2021['EquipoLocal'],
    'EquipoVisitante': season_2020_2021['EquipoVisitante'],
    'Prob_Victoria_Local': (pred[:, 1] * 100).round(2).astype(int), #
    'Prob_Empate': (pred[:, 0] * 100).round(2).astype(int), #
    'Prob_Victoria_Visitante': (pred[:, 2] * 100).round(2).astype(int)
})
```

Para realizar las predicciones de los goles anotados del equipo local y del equipo visitante se utiliza un algoritmo de regresión lineal. El entrenamiento de este algoritmo es similar al anterior, se usan los partidos hasta la temporada 2019-2020 como conjunto de datos de entrenamiento para posteriormente realizar predicciones sobre los partidos de la temporada 2020-2021.

Para empezar, se carga el CSV con los datos de entrenamiento y se seleccionan las columnas de interés, que son las características utilizadas para predecir los goles del equipo local y del equipo visitante. Estas columnas se almacenan en la variable 'X'. Además se crean las variables objetivo, 'y' e 'y2', que

corresponden con las variables que queremos predecir, es decir, el número de goles que anotará el equipo local y el equipo visitante.

```
df = pd.read_csv("/kaggle/input/entrenamiento-prediccion/partidos_caracteristicas_hasta_2019-20.csv")
x = df[['GolesFavL5', 'GolesConL5', 'PuntosL5', 'PosicionLocal', 'GolesFavV5', 'GolesConV5', 'PuntosV5', 'PosicionVisitante']
y = df['Goles Local']
y2 = df['Goles Visitante']
```

A continuación, se divide el conjunto de datos en conjuntos de entrenamiento y prueba utilizando la función `train_test_split()`. Se reserva el 20% de los datos para la prueba, mientras que el 80% restante se utiliza para el entrenamiento del modelo. Esto permite evaluar la capacidad de generalización del modelo. Se crea una instancia del modelo de regresión lineal utilizando la clase `LinearRegression()`. Este modelo se utilizará para predecir el número de goles del equipo local. Se entrena el modelo utilizando los datos de entrenamiento mediante el método `fit(X_train, y_train)`. Se realizan predicciones sobre los datos de prueba utilizando el método `predict(X_test)`. Las predicciones obtenidas se almacenan en la variable '`y_pred`'.

Se calcula el error cuadrático medio (MSE, por sus siglas en inglés) utilizando la función `mean_squared_error()` comparando las predicciones realizadas (`y_pred`) con los valores reales de los goles del equipo local (`y_test`). El MSE representa la diferencia promedio al cuadrado entre las predicciones y los valores reales.

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print("MSE goles local LinearRegression:", mse)
```

Se realiza el mismo procedimiento para predecir los goles que anotará el equipo visitante utilizando un segundo modelo de regresión lineal, se almacenan las predicciones en la variable 'y_pred2' y se calcula el MSE correspondiente.

Para realizar las predicciones de la temporada 2020-2021, se carga el archivo CSV que contiene dichos partidos y se seleccionan las mismas características que se han utilizado durante el entrenamiento anterior. Posteriormente, se realizan las predicciones utilizando el modelo entrenado previamente. Las predicciones se redondean al entero más cercano y se almacenan en las variables 'y_pred_local' y 'y_pred_visitante', que representan el número de goles predichos para el equipo local y el equipo visitante, respectivamente.

```
season_2020_2021 = pd.read_csv('/kaggle/input/entrenamiento-prediccion/partidos_2020-21.csv')

Z_pred = season_2020_2021[['GolesFavL5', 'GolesConL5', 'PuntosL5', 'PosicionLocal', 'GolesFavV5', 'GolesCo
y_pred_local = np.round(regressor.predict(Z_pred)).astype(int)
y_pred_visitante = np.round(regressor2.predict(Z_pred)).astype(int)
```

Finalmente, se crea una nueva columna en el DataFrame *output_df* llamada "golesLocalFinal", donde se almacenan las predicciones de goles del equipo local, y otra columna llamada "golesVisitanteFinal", donde se almacenan las predicciones de goles del equipo visitante.

```
output_df['golesLocalFinal'] = y_pred_local
output_df['golesVisitanteFinal'] = y_pred_visitante
```

Para el cálculo de los goles marcados por el equipo local y visitante durante la primera parte se realiza exactamente el mismo proceso, lo único que varía es las características que seleccionamos como entrada, ya que se cogen las que tienen

relación con los eventos sucedidos en la primera parte como por ejemplo la cantidad de goles que ha anotado un equipo durante la primera parte en los últimos 5 partidos. Todo el procedimiento posterior es igual que para calcular los goles esperados durante los 90 minutos. El resultado de estas nuevas predicciones se almacena en las columnas “golesLocalDescanso” y “golesVisitanteDescanso”.

```
output_df['golesLocalDescanso'] = y_pred_local  
output_df['golesVisitanteDescanso'] = y_pred_visitante
```

7.4 Resultados

Los algoritmos de regresión, como el de regresión lineal y el de regresión logística, son métodos utilizados en el aprendizaje automático para modelar y predecir relaciones entre variables.

El algoritmo de regresión lineal es una técnica utilizada en problemas de regresión, donde se busca predecir un valor numérico continuo. Su funcionamiento se basa en asumir una relación lineal entre la variable dependiente y una o más variables independientes. Este algoritmo será usado para predecir el número de goles anotados por cada equipo, tanto en el final como al término de la primera parte.

En cuanto a los valores del MSE a la hora de predecir los goles del local y del visitante antes del descanso, los valores obtenidos han sido los siguientes:

- MSE goles local descanso: **0.66**
- MSE goles visitante descanso: **0.475**

La interpretación de estos valores a la hora de predecir los goles marcados en la primera parte es que las predicciones realizadas por nuestro algoritmo tienden a desviarse 0.66 y 0.47 unidades cuadradas con respecto a los valores reales. Un valor bajo del MSE significará que las predicciones se asemejan y se acercan bastante a los resultados reales.

En este contexto, podemos inferir que nuestras predicciones de goles al descanso del equipo local y del equipo visitante tienden a ser bastante precisas en relación con los resultados reales.

Es importante tener en cuenta que el MSE es una medida de la diferencia promedio al cuadrado entre las predicciones y los valores reales. Por ejemplo, los goles del equipo visitante tienen un MSE de 0.4753 lo que indica que, en promedio, nuestras predicciones difieren por aproximadamente 0.69 unidades respecto a los valores reales en términos de goles al descanso del equipo visitante. Esta discrepancia relativamente baja sugiere una buena capacidad del modelo para predecir el rendimiento goleador del equipo local y visitante durante la primera mitad del partido.

En resumen, un MSE de 0.66 y 0.46 en nuestro modelo de regresión lineal para predecir los goles del equipo local y del equipo visitante al descanso implica que nuestras estimaciones son generalmente cercanas a los valores reales. Esto demuestra la capacidad de nuestro modelo para proporcionar predicciones confiables y útiles en términos de rendimiento goleador durante el período previo al descanso.

Estos valores aumentan cuando vamos a predecir los goles que anotará cada equipo al final del partido. Estas predicciones están un poco más lejos de los resultados reales pero también hay que tener en cuenta que durante los partidos de fútbol suceden numerosos acontecimientos inesperados e impredecibles que pueden condicionar fácilmente el resultado del mismo. Los valores obtenidos son los siguientes:

- MSE goles local FINAL: **1.675**
- MSE goles visitante FINAL: **1.17**

Como ya hemos mencionado anteriormente, en el contexto en el cual estamos tratando, una desviación de 1.17 unidades cuadradas, o lo que es lo mismo, una desviación promedio de 1.08 unidades con respecto al valor real de los goles anotados por el equipo visitante durante el partido, es un valor bastante aceptable, ya que en promedio nuestras predicciones van a variar en torno a un único gol. Por tanto, teniendo en cuenta el contexto de nuestro caso y la dificultad y complejidad que supone intentar calcular el resultado de un partido de fútbol, unas diferencias promedio de aproximadamente 1 y 1.5 goles hacen que nuestras predicciones sean bastante fiables y útiles.

Por otro lado, el algoritmo de regresión logística se utiliza en problemas, donde se busca predecir una variable categórica con posibles clases. A diferencia de la regresión lineal, la regresión logística utiliza una función logística para estimar las probabilidades de pertenecer a cada clase. Estas probabilidades se pueden utilizar para predecir la clase a la que pertenece una nueva muestra. Si extrapolamos esto al ámbito deportivo, este algoritmo tiene la capacidad de predecir,

dados unos valores de entrada que representan tanto el estado de forma de los equipos como los goles o los puntos obtenidos anteriormente, qué probabilidad tendrá ese partido de pertenecer a las distintas clases; 1 (victoria del local), X (empate) o 2 (victoria del visitante). Nuestro algoritmo de regresión logística tiene una precisión de 0.496, lo cual indica que predice correctamente prácticamente la mitad de los partidos. No es un porcentaje demasiado elevado de aciertos pero puede ser aceptable teniendo en cuenta que al fin y al cabo lo que ofrece son predicciones en base a unos datos de entrada proporcionados que no tienen en cuenta la totalidad de las variables que envuelven a un partido de fútbol.

7.5 Problemas encontrados

Durante la creación, desarrollo y entrenamientos de los estos algoritmos de regresión para poder predecir el resultado de los partidos de fútbol así como los porcentajes de victoria de cada equipo, se han ido afrontando y superando diversos problemas e inconvenientes.

El primero de ellos fue la elección del tipo de algoritmo a desarrollar. Se comenzó creando una red neuronal, la cual fue entrenada con diversos parámetros y valores, cómo ya se ha explicado anteriormente, pero no terminaba de arrojar resultados medianamente fiables. Además era bastante costoso computacional y temporalmente realizar predicciones, ya que tomaba bastante tiempo entrenar el modelo cada vez que se cambiaba algún parámetro para comprobar si su rendimiento era más eficiente.

A la hora de encontrar la manera de realizar predicciones probabilísticas para predecir el ganador también surgen inconvenientes, ya que los valores de precisión que se obtenían con los valores y parámetros que se estaban probando ninguno superaba el 25% de precisión a la hora de predecir el ganador. Finalmente se solventó gracias al método *predict_proba()* de la librería *sklearn*. Este método permite obtener las probabilidades que tiene un objeto de pertenecer a cada una de las clases objetivo, por lo que era justamente lo que se necesitaba.

Una vez implementados los algoritmos, se procedió a intentar reducir el ruido de los datos de entrada, es decir, filas de datos que tienen valores que no aportan todo el valor que deberían y este hecho interfiere en el entrenamiento de nuestro modelo. Por ello, se decidió eliminar las 50 primeras filas de cada temporada, es decir, los partidos correspondientes a las 5 primeras jornadas. Por ejemplo, en la jornada 3, solamente se han podido recopilar datos e información de las 2 jornadas anteriores por lo que no se puede evaluar ni el rendimiento del equipo ni los valores de las columnas de una forma igual que en jornadas venideras dónde sí se hayan disputado 5 partidos anteriores. Esto nos llevó a eliminar unas 1200 filas aproximadamente y de esa manera se eliminó todo el ruido posible de los datos de entrada y conseguimos mejorar un poco la precisión de nuestro algoritmo a la hora de calcular el ganador del partido y disminuir el error cuadrático medio cuando se trataba de predecir los goles anotados.

8. Documentación técnica

8.1. Introducción

Durante este capítulo, compartiremos los detalles sobre el diseño, los requisitos, las funcionalidades y los problemas encontrados durante el proceso de creación de la aplicación.

Esta ha sido diseñada para proporcionar a los usuarios información detallada y estadísticas sobre la predicción de los partidos de la temporada 2020-21. Utilizando datos y distintos algoritmos, nuestra aplicación muestra la probabilidad de victoria para cada equipo, así como los resultados esperados tanto al descanso como al final del partido. Esto permite a los usuarios tomar decisiones informadas y seguir de cerca el desempeño de sus equipos favoritos.

8.2 Requisitos y casos de uso

La lista de requisitos y casos de uso ayuda a comprender el alcance y las funcionalidades de nuestra aplicación y nos servirá como referencia durante el proceso de desarrollo y prueba. A continuación, se presenta dicha lista de requisitos funcionales, así como los casos de uso detallados que guiarán el uso y la interacción de los usuarios con nuestra aplicación.

1. Como usuario, quiero ver la lista de partidos de la temporada 2020-21 en la pantalla de inicio para conocer los equipos, las fechas y las probabilidades de victoria.

- Requisitos funcionales:

- Mostrar una lista de partidos con información relevante como equipos, fechas y probabilidades de victoria de cada equipo así como de empate.
 - Presentar la información de manera clara y legible en la pantalla de inicio.
2. Como usuario, quiero poder acceder a los detalles de un partido al hacer clic en él, para ver las probabilidades de victoria y los resultados esperados al descanso y al final.
- Requisitos funcionales:
 - Permitir la interacción del usuario al hacer clic en un partido para acceder a los detalles.
 - Mostrar las probabilidades de victoria y los resultados esperados al descanso y al final del partido seleccionado.
3. Como usuario, quiero poder buscar los partidos jugados por un equipo específico utilizando un buscador, para obtener información detallada sobre los partidos en los que ha participado ese equipo.
- Requisitos funcionales:
 - Implementar un buscador que permita al usuario ingresar el nombre de un equipo.
 - Filtrar la lista de partidos para mostrar solo aquellos en los que ha participado el equipo buscado.
 - Mostrar información detallada sobre los partidos encontrados, como equipos contrincantes, fechas y resultados.

4. Como usuario, quiero poder buscar los partidos disputados en una fecha específica utilizando un buscador, para revisar los resultados de los partidos en esa fecha.

- Requisitos funcionales:

- Implementar un buscador que permita al usuario ingresar una fecha.
- Filtrar la lista de partidos para mostrar solo aquellos disputados en la fecha especificada.
- Mostrar información relevante sobre los partidos encontrados, como equipos, resultados y probabilidades.

5. Como usuario, quiero poder ordenar la lista de partidos por fecha, de menor a mayor o viceversa, utilizando un ícono situado en la parte superior de la pantalla.

- Requisitos funcionales:

- Agregar un ícono en la parte superior de la pantalla para permitir al usuario cambiar el orden de la lista de partidos.
- Al pulsar en el ícono, cambiar el orden de la lista de partidos según la preferencia del usuario (de menor a mayor o viceversa).

6. Como usuario, quiero poder acceder a una pantalla de estadísticas al seleccionar el ícono correspondiente, para ver los equipos con el mejor y peor porcentaje de acierto en el algoritmo.

- Requisitos funcionales:

- Incluir un ícono en la interfaz de usuario que represente las estadísticas.

- Al seleccionar el ícono, se redirigirá al usuario a una pantalla dedicada a las estadísticas.
- Mostrar en esta pantalla las tablas con los equipos que tienen el mejor y peor porcentaje de predicciones acertadas por nuestro algoritmo, es decir, mostrar los equipos que tienen mejor y peor porcentaje de acierto en relación de las predicciones realizadas y los resultados reales.

8.3 Mockups

En el desarrollo de la aplicación de predicciones deportivas en Flutter, los mockups desempeñaron un papel fundamental en la etapa de diseño. Estos diseños visuales, representaciones gráficas de las interfaces de usuario, nos permitieron visualizar y comunicar de manera efectiva cómo se vería y funcionaría nuestra aplicación antes de comenzar el desarrollo real. En esta sección, presentamos una selección de los mockups previos creados, los cuales reflejan el enfoque y la estética de nuestra aplicación.

A continuación, presentamos una selección de los mockups previos realizados, destacando las interfaces clave y las características principales de nuestra aplicación. Estos diseños proporcionan una visión clara y concreta de cómo se espera que se vea y funcione nuestra aplicación final, y nos sirvieron como guía durante el desarrollo de la misma.



En estas imágenes se puede observar el diseño primitivo de nuestra aplicación. En la figura de la izquierda se muestra la pantalla de inicio, y que es acorde con todos los casos de usos y requisitos que han sido expuestos con anterioridad. Y la imagen de la parte derecha corresponde a los detalles de un partido. En ella se pueden observar todos los datos y valores que predice nuestro algoritmo y que están recogidos en los requisitos y casos de uso anteriormente mencionados.

← Estadísticas

Equipos con mayores porcentajes de acierto

Posición	Equipo	Predicción
1	Real Betis	21%
2	Osasuna	20%
3	Sevilla	18%
4	Granada	18%
5	Alavés	15%

Equipos con menores porcentajes de acierto

Posición	Equipo	Predicción
1	Valencia	7%
2	Cádiz	9%
3	Eibar	11%
4	Getafe	12%
5	Vallecano	12%

En la imagen anterior, se puede observar la pantalla correspondiente a las estadísticas de acierto de la predicción que realiza nuestro algoritmo en relación a cada equipo. Estos valores son una comparación entre el resultado real del partido y la predicción que hemos realizado. Con estas estadísticas se le proporciona al usuario, de manera fácil, sencilla y atractiva, cuáles son los equipos más predecibles y qué equipos son muy irregulares y por tanto, difíciles de predecir.

8.4 Desarrollo de la aplicación

8.4.1. Diseño y estructura

En el desarrollo de nuestra aplicación de predicciones deportivas en Flutter, hemos seguido una arquitectura y diseño cuidadosamente planificados para garantizar la eficiencia y la fácil mantenibilidad del código. Nuestra estructura de carpetas refleja esta organización, donde cada componente desempeña un papel específico en la funcionalidad global de la aplicación.

En la carpeta "modelos", se encuentra el archivo "match.dart", que alberga la definición del modelo de datos utilizado para almacenar la información de las predicciones realizadas para cada partido. Este modelo incluye los atributos relevantes, como los equipos que disputaron el partido, la fecha y las probabilidades de victoria de cada equipo. Así mismo contiene los métodos necesarios para serializar y deserializar los archivos JSON.

```
class Match{
    int id;
    String equipoLocal;
    String equipoVisitante;
    String fecha;
    int golesLocalDescanso;
    int golesVisitanteDescanso;
    int golesLocalFinal;
    int golesVisitanteFinal;
    int porcentajeEmpate;
    int porcentajeVictoriaLocal;
    int porcentajeVictoriaVisitante;
    int golesLocalReal;
    int golesVisitanteReal;
    String ganador;
```

En la carpeta "servicios", se encuentra el archivo "match_service.dart", que actúa como el servicio central de la lógica de nuestra aplicación y la conexión con la base de datos. Aquí se implementan los métodos y funciones necesarios para gestionar las operaciones relacionadas con los partidos, como la obtención de los datos o el cálculo de los porcentajes de acierto que tiene el algoritmo sobre cada equipo. Asimismo incluye por ejemplo, el método "sort" que permite ordenar los partidos por fecha de mayor a menor y viceversa, el método "findByName" que permite filtrar los partidos en función del equipo que lo haya disputado o el "findByDate" que filtra los partidos por la fecha en la que se disputaron.

```
List<Match> findByName(String name) {
    return matches
        .where((element) =>
            element.equipoLocal
                .trim()
                .toLowerCase()
                .contains(name.trim().toLowerCase()) ||
            element.equipoVisitante
                .trim()
                .toLowerCase()
                .contains(name.trim().toLowerCase()))
        .toList();
}

List<Match> findByDate(String date) {
    return matches
        .where((match) => match.fecha.contains(date)).toList();
}
```

```

Map<String, int> calcularPorcentajesAcierto(){
    List<Map<String, dynamic>> registros = obtenerRegistros(matches);
    Map<String, int> numeroAciertos = {};
    for (Map<String,dynamic> registro in registros){
        String equipoLocal = registro['equipo_local'];
        String equipoVisitante = registro['equipo_visitante'];
        int porcentajeVictoriaLocal = registro['porcentaje_victoria_local'];
        int porcentajeVictoriaVisitante = registro['porcentaje_victoria_visitante'];
        int porcentajeEmpate = registro['porcentaje_empate'];
        String ganador = registro['ganador'];

        String ganadorPredicho;
        if((porcentajeVictoriaLocal>=porcentajeEmpate) & (porcentajeVictoriaLocal>=porcentajeVictoriaVisitante)){
            ganadorPredicho = 'H';
        }else if((porcentajeVictoriaVisitante>=porcentajeEmpate) & (porcentajeVictoriaLocal<=porcentajeVictoriaVisitante)){
            ganadorPredicho = 'A';
        }else{
            ganadorPredicho = 'ANY';
        }
        if (ganadorPredicho == ganador) {
            // El equipo predijo correctamente
            numeroAciertos[equipoLocal] = (numeroAciertos[equipoLocal] ?? 0) + 1;
            numeroAciertos[equipoVisitante] = (numeroAciertos[equipoVisitante] ?? 0) + 1;
        }
    }
    for(int acierto in numeroAciertos.values){
        int porcentaje = (acierto/38).round()*100;
        acierto = porcentaje;
    }
    List<MapEntry<String, int>> listaOrdenada = numeroAciertos.entries.toList()
        ..sort((a, b) => b.value.compareTo(a.value));

    numeroAciertos = Map.fromEntries(listaOrdenada);
    return numeroAciertos;
}

```

En la imagen anterior podemos observar la función `calcularPorcentajesAcierto()`, en la cual devolveremos un Diccionario en el que la clave será el equipo y el valor el porcentaje de partidos que hemos predicho correctamente. Para llegar a ello previamente creamos un registro de cada partido, junto con sus predicciones y su resultado real. Una vez tenemos este registro, comprobamos que la predicción realizada por nuestro algoritmo es igual al resultado real y si es así, sumamos 1 al valor del diccionario que vamos a devolver, cuya clave sea el equipo que ha sido predicho correctamente. Por último ordenamos ese diccionario por orden de mayor a menor según sus claves para obtener el resultado ordenado de mayor a menor eficiencia en la predicción.

Por otro lado, la carpeta "delegates" contiene la implementación de los métodos delegados, que son fundamentales para el correcto funcionamiento de los buscadores en nuestra aplicación. Estos métodos permiten realizar búsquedas de partidos por equipo o por fecha, facilitando a los usuarios encontrar los resultados deseados de manera rápida y sencilla; además nos facilita la comprensión del funcionamiento de los buscadores ya que encapsula y absorbe toda la responsabilidad del buen desempeño de los mismos.

En la carpeta "widgets", se encuentran los componentes reutilizables que utilizamos para facilitar la creación de las tarjetas o "cards" que componen la pantalla principal. Estos widgets encapsulan la lógica y la presentación de cada tarjeta, permitiendo una mayor flexibilidad y mantenibilidad del código. Algunos de ellos son 'match_date' o 'match_result_widget' que se encargan de la representación tanto de la fecha como del resultado del partido dentro de la tarjeta del mismo.

Por último, en la carpeta "assets", se encuentran todos los recursos visuales necesarios para el correcto funcionamiento de nuestra aplicación, como las imágenes de los escudos de los equipos. Estos recursos son utilizados en todo el diseño de la interfaz de usuario para brindar una experiencia visual atractiva y coherente.

Además de estas carpetas, a las cuales le hemos otorgado unas responsabilidades específicas de nuestra aplicación, también seguimos las

convenciones de estructura de carpetas estándar de Flutter, como "lib" para el código fuente principal, "android" e "ios" para ejecutar en las distintas plataformas o "pubspec.yaml" para la gestión de las dependencias del proyecto.

En conjunto, esta arquitectura y diseño de carpetas nos han permitido desarrollar una aplicación estructurada, escalable y fácil de mantener. Cada componente desempeña un papel importante en la funcionalidad global de la aplicación, y la modularidad nos ha permitido realizar cambios y mejoras de manera eficiente y sin afectar otras partes del código.

8.4.2. Funcionalidades e interfaz

En este apartado, se muestra el diseño de la interfaz de usuario de nuestra aplicación. La interfaz de usuario es un componente fundamental de la aplicación, ya que es la forma en que los usuarios interactúan con la aplicación y acceden a la información sobre los partidos.

Con el objetivo de crear una interfaz de usuario efectiva, nuestra intención era desarrollar una aplicación fácil de usar, intuitiva y orientada a todo tipo de usuarios. Buscábamos un diseño sencillo y minimalista que mejorase la experiencia del usuario. Además, se prestó atención a la selección de una paleta de colores atractiva y coherente, la elección de fuentes legibles y la disposición clara y organizada de la información en cada pantalla. En este apartado, se presentan y explican la interfaz de la aplicación, así como las distintas funcionalidades que ofrece la misma.



Esta imagen se corresponde con la pantalla de inicio de nuestra aplicación. En ella, podemos observar todos los aspectos y detalles que teníamos como referencia y objetivos, como por ejemplo mostrar de forma clara y ordenada la lista de partidos disputados durante la temporada o poder identificar rápidamente los porcentajes predichos por nuestro algoritmos. Además en la parte superior de la pantalla se encuentran diversos iconos que nos llevan a nuevas pantallas, nos permiten filtrar u ordenar todos los partidos. Cabe mencionar al segundo ícono de los cuatro, que es el encargado de cambiar el orden en el que se listan los partidos.

Inicialmente están ordenados por fecha ascendente pero al pulsarlo los partidos se invierten y se ordenan por fecha descendente.

El icono situado más a la izquierda nos permite acceder a la pantalla de estadísticas.

← **Estadísticas** 

Equipos con mayores porcentajes de acierto

Posición	Equipo	Predicción
1	Sevilla	61%
2	Ath Madrid	55%
3	Barcelona	53%
4	Granada	53%
5	Valencia	50%

← **Estadísticas** 

Equipos con mayores porcentajes de acierto como LOCAL

Posición	Equipo	Predicción
1	Ath Madrid	79%
2	Sevilla	74%
3	Real Madrid	68%
4	Barcelona	58%
5	Betis	53%

Equipos con menores porcentajes de acierto

Posición	Equipo	Predicción
20	Cadiz	39%

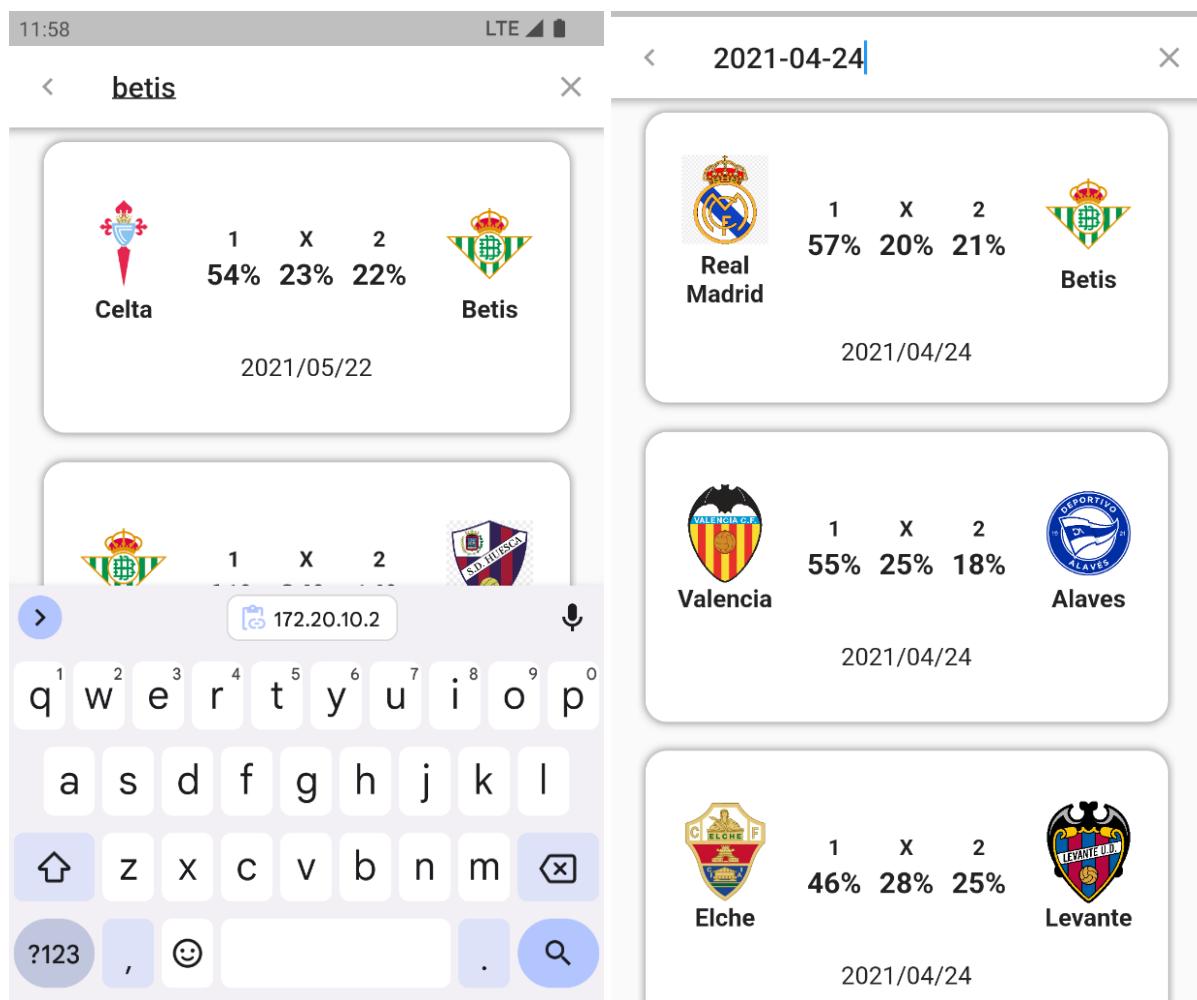
Equipos con mayores porcentajes de acierto como VISITANTE

Posición	Equipo	Predicción
1	Elche	63%

En estas figuras se pueden observar los distintos valores porcentuales obtenidos de comparar la predicción que realiza nuestro algoritmo con el que ha sido el resultado real del partido. Este cálculo se realiza en el servicio de nuestra aplicación y es muy útil ya que nos ofrece una visión global de cuáles son los equipos sobre los que se realizan predicciones más fiables, los que son menos

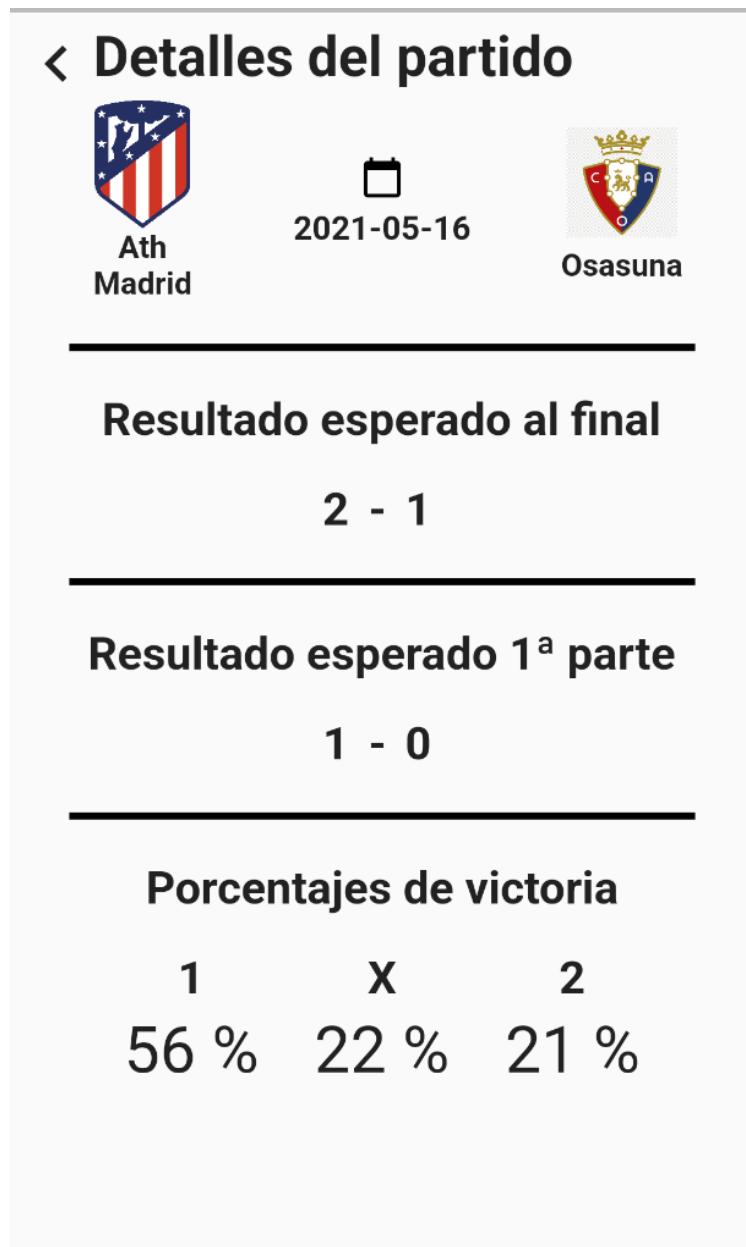
fiables o los que, cuando juegan de local, son bastante fiables pero cuando lo hacen como visitante no y viceversa.

Los dos iconos restantes son los encargados de realizar las distintas búsquedas que hemos comentado con anterioridad, esto nos permite tanto realizar búsquedas para poder así filtrar los partidos que ha disputado un equipo como filtrar los partidos disputados en un día, mes o año completo.



En las figuras anteriores se puede observar el resultado de realizar las distintas búsquedas y como se muestran los resultados obtenidos de las mismas.

Por último, al pulsar sobre cualquiera de los partidos listados, aparecerá una pantalla en la que se muestran los datos y las predicciones realizadas sobre los mismos.



En esta imagen se observa a la perfección todos los valores predichos por nuestro algoritmo y además se hace de una manera clara y fácilmente comprensible siguiendo la estructura y diseño que se planteaba en los mockups. Utilizando los goles esperados por cada equipo tanto en la primera parte como al final del partido,

realizamos una predicción del resultado esperado tanto al descanso como al final del encuentro. Además mostramos los porcentajes predichos de victoria del local, empate y derrota del visitante.

8.4.3. Problemas encontrados

Durante el desarrollo de nuestra aplicación Flutter, nos han ido surgiendo inconvenientes y problemas tanto a nivel de diseño y código, como a nivel de comunicación entre los distintos módulos que componen nuestro proyecto. Todos ellos se han ido solventando y mitigando conforme han ido apareciendo hasta lograr una implementación eficiente de la aplicación. A continuación, enumeramos algunos de los problemas e inconvenientes más destacados, ya sean por su dificultad a la hora de resolverlos o por que han sido errores poco comunes y que su aparición ha sido fruto de las características específicas que presenta nuestro proyecto.

Una de las primeras incógnitas que apareció durante el desarrollo fue el no entendimiento de por qué al llamar a la ruta de la API desde nuestra aplicación, no había ningún tipo de respuesta por parte de la misma. Finalmente, supimos que esto era debido a que la aplicación se corría en un simulador, por lo que cuando llamábamos a la API mediante “localhost:3000”, realmente era el propio dispositivo simulado el que estaba realizando esa petición sobre su propio “localhost”. Para solucionarlo, tenemos que cambiar la dirección sobre la que llamamos a la API en nuestro servicio, poniendo la dirección IP de nuestra máquina en lugar de localhost.

```
// PONER LA IP DE LA MÁQUINA PARA QUE FUNCIONE CORRECTAMENTE
final String _baseUrl = '172.20.10.2:3000';
```

Por otro lado, a nivel de desarrollo del diseño de la interfaz, hemos tenido problemas relacionados con el desbordamiento de los píxeles cuando intentábamos mostrar las distintas imágenes de los equipos de fútbol. A este error se le sumaba que cuando sí aparecía la imagen de algún equipo, era de forma aumentada y sin tener en cuenta el espacio que le estábamos asignando a la misma. Este error se solucionó estableciendo unos valores de altura y anchura a la imagen en el widget en el cual se creaba la misma. Gracias a esto, pudimos tener un control total sobre el tamaño de la imagen y así poder centrarla y alinearla en función a otros elementos de la interfaz mejorando así el aspecto y la apariencia de nuestra aplicación.

```
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children:
  [Image.asset(
    'assets/$teamName.png',
    //getImagePath(teamName),
    width: 60,
    height: 70,
```

Con el atributo “mainAxisAlignment” controlamos la alineación de la imagen y con “width” y “height” su altura y su anchura, evitando así los errores por desbordamiento de píxeles.

Otro de los inconvenientes al que nos hemos enfrentado, se encuentra en el problema que surgía cuando se agregaron nuevas columnas en la base de datos

para poder realizar otros cálculos de probabilidades. Este hecho no nos permitía ni que nuestra aplicación se ejecutara correctamente, todo esto se debía a que los campos correspondientes a las nuevas columnas no estaban incluidos en el objeto JSON que pasaba la API a nuestra aplicación. En este contexto, se evidenció una de las limitaciones de Flutter, que nos hace poner en valor su manejo deficiente de los valores nulos. Para resolver esta situación, se realizó la introducción de los nuevos valores en el JSON y se reinició la aplicación.

8.5 Instrucciones para ejecutar el modelo de aprendizaje automático

Los siguientes pasos son de obligado cumplimiento para poder ejecutar el modelo de aprendizaje automático:

Instalación de Visual Studio Code (VSCode):

- Visita el sitio web oficial de VSCode en <https://code.visualstudio.com/> y descarga la versión adecuada para tu sistema operativo.
- Sigue las instrucciones de instalación.
- Una vez instalado, asegúrate de tener acceso al ejecutable de VSCode.

Instalación de las extensiones:

- Buscar e instalar las extensiones ‘Python’ y ‘Jupyter’.

A continuación procedemos a abrir la carpeta que contiene los ficheros necesarios para realizar el procesado de datos y entrenamiento del modelo, que se corresponde con la ruta ‘ Modelo/modelo ’ y ejecutamos el archivo ‘limpieza-datos-y-modelo.ipynb’.

8.6 Instrucciones para ejecutar la aplicación Flutter

A continuación se van a detallar las instrucciones para poder ejecutar la aplicación. Antes de llevar a cabo esta acción, es necesario cumplir esta serie de requisitos previos los cuales son fundamentales para el correcto funcionamiento de la aplicación:

Instalación de Node.js:

- Visita el sitio web oficial de Node.js en <https://nodejs.org/> y descarga la versión LTS recomendada para tu sistema operativo.
- Sigue las instrucciones de instalación.
- Verifica la instalación abriendo una terminal y ejecutando el comando node -v y npm -v. Deberías ver las versiones instaladas de Node.js y npm respectivamente.

Instalación de Android Studio:

- Visita el sitio web oficial de Android Studio en <https://developer.android.com/studio> y descarga la versión adecuada para tu sistema operativo.
- Sigue las instrucciones de instalación específicas para tu sistema operativo.
- Durante la instalación, seleccionar los componentes necesarios para el desarrollo de aplicaciones Flutter, como el Android SDK y el emulador de Android.

Instalación de Visual Studio Code (VSCode):

- Visita el sitio web oficial de VSCode en <https://code.visualstudio.com/> y descarga la versión adecuada para tu sistema operativo.
- Sigue las instrucciones de instalación.
- Una vez instalado, asegúrate de tener acceso al ejecutable de VSCode.

Instalación de las extensiones:

- En VSCode, en el menú lateral seleccionamos ‘Extensiones’.
- Buscar e instalar la extensión ‘Flutter’.
- Buscar e instalar la extensión ‘Dart’.

Configuración del entorno Flutter:

- Abre una terminal y ejecuta el siguiente comando para instalar Flutter SDK: flutter doctor --android-licenses.
- Acepta las licencias necesarias para el desarrollo de aplicaciones Android.
- Ejecuta el comando flutter doctor para verificar que Flutter se haya configurado correctamente.
- Si hay algún problema detectado, sigue las instrucciones proporcionadas para solucionarlo.

Una vez tenemos se han realizado los pasos anteriores y tenemos todos los entornos y programas necesarios descargados, procedemos a inicializar la api y ejecutar la aplicación flutter siguiendo estas instrucciones:

1. Abrimos una terminal nueva y nos situamos en la ruta de la carpeta 'api/api_predicciones', una vez ahí ejecutamos el comando 'npm start'. Con esto ya deberíamos tener nuestra API funcionando.
2. Abrimos Android Studio, seleccionamos 'more actions > virtual device manager > create device' y seleccionamos el móvil llamado Pixel 2, seleccionamos Android 13 (Tiramisú) y finalizamos.
3. En una ventana de VSCode, abrir la carpeta 'AppFlutter/tfg_app'
4. En la esquina inferior derecha de la pantalla, seleccionar el dispositivo que vamos a usar como emulador
5. Iniciar el emulador
6. Buscamos cual es la ip del equipo:
 - a. En Mac, introducimos el comando 'ifconfig' en una terminal, buscamos la interfaz de red que se está usando (si es inalámbrica "en0") y junto a 'inet' nos aparece la ip
 - b. En Windows, en la ventana de comandos, escribiremos "ipconfig". Buscamos la sección que muestra la dirección IPv4 junto a "Dirección IP".
 - c. En Linux, abrimos una terminal e introducimos el comando "ip addr show".
7. En VSCode, buscamos el archivo "match_service.dart" que se encuentra dentro de las carpetas "/lib/services" y actualizaremos la variable `_baseUrl`

con la ip de nuestra máquina, de tal manera que quede de la siguiente manera: `_baseUrl = "ip_maquina:3000";`

8. Pulsamos en Ejecutar > Ejecutar sin depuración para correr la aplicación. Si aparece una ventana emergente con distintos lenguajes para compilar, elegimos “Dart y Flutter”.

9. Conclusiones

Durante el proceso de investigación y desarrollo de este proyecto, hemos evidenciado el potencial de la inteligencia artificial para desvelar patrones ocultos en los datos y proporcionar una visión reveladora sobre los resultados y el desarrollo de los encuentros futbolísticos. La utilización de algoritmos de aprendizaje automático

nos ha permitido analizar grandes volúmenes de datos de manera eficiente, identificando tendencias y generando pronósticos confiables.

La búsqueda del dataset inicial, el tratamiento y limpiado de los datos mediante las distintas funciones auxiliares ha sido desarrollado en común con Alberto Gallego Huerta. Asimismo, Alberto se ha encargado de la creación de la base de datos y del desarrollo e implementación de la API, parte fundamental del proyecto ya que es necesaria para guardar toda la información de las predicciones realizadas por nuestro algoritmo y a través de la API, poder mostrar la información en las distintas aplicaciones desarrolladas, tanto la aplicación web como la aplicación android.

La elección y experimentación con el modelo de inteligencia artificial junto con el desarrollo de la aplicación Android usando Flutter, ha presentado numerosos desafíos y dificultades durante el desarrollo del proyecto, la disponibilidad de fuentes confiables de información futbolística, así como la recopilación y limpieza de los datos requeridos, representaron un obstáculo inicial en nuestro camino. Además, el procesamiento de grandes volúmenes de datos y la selección adecuada de variables relevantes para nuestros algoritmos de inteligencia artificial también se convirtieron en desafíos técnicos a solventar.

Otra dificultad que encontramos fue la selección y configuración óptima de los algoritmos usados finalmente. La elección adecuada de estos algoritmos y la afinación de sus parámetros requerían un enfoque experimental y exhaustivo, ya que cada modelo presentaba ventajas y limitaciones específicas, tras dicha

evaluación se tomó la decisión de utilizar finalmente algoritmos de regresión, tanto logística como lineal.

Este proyecto no sólo ha enriquecido nuestro conocimiento sobre inteligencia artificial y análisis de datos en el contexto deportivo, sino que también nos ha brindado la oportunidad de aplicar nuestros conocimientos teóricos en un proyecto práctico y tangible. Hemos experimentado de primera mano los beneficios y las dificultades de trabajar en un proyecto conjunto, aprendiendo a colaborar eficientemente, comunicarnos de manera efectiva y aprovechar las fortalezas individuales para alcanzar un objetivo común.

En resumen, este proyecto nos ha permitido desarrollar habilidades técnicas, adquirir conocimientos especializados en el campo de la inteligencia artificial aplicada al fútbol y experimentar los desafíos y las recompensas de trabajar en un proyecto de investigación y desarrollo. Estamos orgullosos de los resultados obtenidos y confiamos en que nuestro trabajo contribuirá al avance y la aplicación de la inteligencia artificial en el análisis y pronóstico de partidos de fútbol.

10. Referencias

1. NerdyTips: <https://nerdylips.com/>

2. DeepTip8:

<https://www.lne.es/asturias/2021/05/20/empresa-asturiana-crea-primera-app-52048499.html>

3. Bullet Bet Predictions: <https://www.bulletbetpredictions.com/>

4. Conjunto de datos primitivo de Kaggle:

<https://www.kaggle.com/datasets/kishan305/la-liga-results-19952020>

5. Pandas. Documentación oficial de Pandas. Disponible en:

<https://pandas.pydata.org/docs/>

6. Scikit-learn. Documentación oficial de scikit-learn. Disponible en:

<https://scikit-learn.org/stable/documentation.html>

7. Flutter. Página oficial de Flutter. Disponible en: <https://flutter.dev/>

8. Node.js. Página oficial de Node.js. Disponible en: <https://nodejs.org/>

9. Visual Studio Code. Página oficial de Visual Studio Code. Disponible en:

<https://code.visualstudio.com/>