Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Matemáticas y Física

Maestría en Ciencia de Datos

Sistema de aprendizaje interactivo de aperturas de ajedrez

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
Maestro en Ciencia de Datos

Presenta:
Alejandro Noel Hernández Gutiérrez

Director de proyecto:
Mtro. Juan Francisco Muñoz Elguezábal.

Tlaquepaque, Jalisco, 22 de agosto de 2024

Resumen

En la etapa de aprendizaje para jugar ajedrez es primordial el entendimiento de aperturas para aspirar a ser un jugador de nivel intermedio o avanzado. Esto puede resultar abrumador para una persona que a penas comienza, debido a que leer libros de ajedrez es una tarea complicada, porque se dividen en secciones donde explican apertura por apertura y movimiento por movimiento (existen al menos 1300 variantes cada una con hasta 21 movimientos), además, los recursos web y aplicaciones móviles en el mercado presentan limitaciones como que generalmente son de paga o están diseñados para estudio pasivo.

El prototipo desarrollado en este proyecto es un sistema de aprendizaje didáctico en el que el usuario puede jugar contra una computadora esta fase del ajedrez de forma aleatoria y obtiene retroalimentación constante de su posición y la del rival así como sugerencias de movimientos de aperturas.

La base de datos que alimenta el modelo clasificador [1] proviene de la plataforma online de ajedrez 'Lichess'. El archivo contiene más de veintemil juegos e incluyen los movimientos jugados, la apertura y datos estadísticos de los jugadores.

En este trabajo de grado se presenta el modelo de clasificación 8.3 que logra predecir victoria con un accuracy de 91.74 % y un brier score de 0.06, seleccionado comparando una serie de modelos propuestos a través de 2 experimentos.

Tabla de Contenidos

Índice de figuras

Página

Índice de tablas

Página

Dedico este texto a mi niño interior, que está orgulloso de quién soy. A mis padres que me mostraron como ejemplo siempre la honestidad y el trabajo duro. A mis hermanos, pilar de mi vida. A los amigos que hice en este proceso y a mi novia que sufrió conmigo días difíciles y me apoyó en todo momento.

1 Introducción

La lectura de libros de ajedrez puede resultar abrumadora sobre todo para jugadores principiantes y los recursos digitales existentes son herramientas en muchas ocasiones de paga o diseñadas para un estudio pasivo en el que se cuenta con múltiples opciones (existen más de 1327 variantes de aperturas) y es difícil y exhaustivo elegir las adecuadas y estudiarlas cuando a penas se tienen nociones fundamentales del juego.

Desde una perspectiva posicional, estos exploradores de la fase inicial del juego carecen de información cuantitativa sobre la calidad de la posición en cada movimiento y la probabilidad de victoria para cada jugador en la posición y qué tan buena es a futuro (se muestran conteos y porcentajes basados en históricos).

Según James H. McMillan[3] es probable que los estudiantes se sientan motivados si se satisfacen sus necesidades, si ven valor en lo que están aprendiendo y si creen que pueden tener éxito con un esfuerzo razonable.

La hipótesis que persigue este trabajo es que existe una forma de sistematizar la información del aprendizaje de aperturas de este deporte para que el estudiante pueda practicar contra la computadora, que cumplirá la función de entrenador, y tenga un panorama general de la posición que alcanza en cada movimiento (modelo de clasificación), así como un despliegue de información de la misma (extracción de características) a través del desarrollo de tecnología, tomando en cuenta los criterios de McMillan para su diseño.

El capítulo Contexto brinda una introducción al ajedrez y sus fases, destacando la importancia de la apertura. El análisis del problema se detalla en Problema, mientras que Objetivos define el objetivo general y especifica las metas del proyecto. La revisión de literatura y recursos tecnológicos relevantes se encuentra en Revisión de literatura y recursos tecnológicos.

El proceso desde la recolección hasta el análisis de datos, junto con la ingeniería de características y evaluación de métricas, se explica en Datos y métodos. El desarrollo del prototipo tecnológico, el Bot openings_ialex, y sus funcionalidades se muestran en Prototipo.

Los resultados de los experimentos y el análisis técnico del

prototipo se presentan en Resultados. Propuestas futuras para fortalecer el proyecto se discuten en Trabajo futuro. El capítulo Sesgos e implicaciones reflexiona sobre los sesgos potenciales y sus impactos, además de las estrategias para mitigarlos.

Detalles técnicos sobre el repositorio y el formato del proyecto se encuentran en Notas de implementación. Finalmente, el Apéndice ofrece acceso al código utilizado en el experimento 2, incluyendo la selección y optimización del modelo ganador.

2 Contexto

En esta sección se presenta de forma breve el juego de ajedrez y sus fases, con énfasis en por qué es importante la etapa de apertura.

2.1 El juego de ajedrez

El ajedrez es un juego popular y emocionante en el que *personas o máquinas* participan en enfrentamientos de dos jugadores. Una partida de este deporte cuenta con 16 piezas que inician en una posición sobre un tablero de 8x8 casillas y se mueven a través de un tablero con reglas bien definidas, por tanto, es un juego determinista en el que el único elemento estocástico es el color que con el que cada participante jugará que comúnmente se decide por azar. Por lo tanto, cuando un rey está bajo ataque o en «Jaque» debe escapar.

64	63	62	61	60	59	58	57
49	50	51	52	53	54	55	56
48	47	46	45	44	43	42	41
33	34	35	36	37	38	39	40
32	31	30	29	28	27	26	25
17	18	19	20	21	22	23	24
16	15	14	13	12	11	10	9
1	2	3	4	5	6	7	8

Figura 2.1: Casillas del tablero de ajedrez. Recuperado de chegg.com

Según Bobby Fischer [4] Gran Maestro estadounidense en su libro *Bobby Fischer Teaches Chess*, el objetivo de esta disciplina es atacar el rey del enemigo en una forma tal que no pueda escapar de su captura. Una vez que la captura se ejecuta, el rey entra en «jaque mate» y el juego se termina.

Por consenso de la comunidad, conceptualmente una partida se divide en tres fases: apertura, medio juego y final. De forma sencilla, **la apertura** es el nombre que recibe una serie de movimientos en un

Figura 2.2: Tablero tradicional de ajedrez. Recuperado de de chessfox

juego y según Di Luca [5] puede constar desde un movimiento hasta alrededor de **25**.

Pero ¿qué se busca durante la fase inicial del juego? Lev Alburt[6] en su libro *Chess openings for white explained* menciona que «con blancas se busca una posición por lo menos igual; aunque es preferible mantener cierta ventaja, sin embargo, demandar una ventaja significativa usualmente no es realista. Con negras, se busca una posición igual, o si en algún punto se vuelve ligeramente peor, se busca que al menos sea una posición que se tenga idea cómo mantener.»

Muchas aperturas se conocen como «aperturas estándar». Lo que significa que han sido jugadas y estudiadas por miles de jugadores a través de la historia. Cada apertura producida genera cierta ventaja o equilibrio posicional entre las piezas que defienden y las que atacan al oponente.

2.2 *Importancia de las aperturas*

Las aperturas son tan importantes que muchos jugadores pasan años de sus vidas simplemente estudiando aperturas [5]. Según Di Luca, en el artículo ya citado se menciona que existen razones por las que las aperturas son importantes. Algunas de ellas son:

La posición inicial del tablero. La apertura está destinada a ayudar a colocar en buena posición las piezas en el tablero y principalmente establecerlas para controlar o amenazar el centro del tablero o la mayor cantidad posible de casillas cruciales y darle forma así a la batalla que los jugadores que participan en esta fase quieren llevar.

Tomar el control del juego. Cuando un principiante se enfrenta a un jugador que domina un repertorio amplio de esta fase del juego, el comportamiento ordinario suele ser reaccionar al oponente que entiende conceptualmente los movimientos que está haciendo, sus objetivos estratégicos y posicionales o bien proponer ideas de juego.

Como en la apertura participan de forma secuencial siempre dos jugadores, entre más variantes conozcan, más posibilidades tiene un jugador de controlar el tipo de juego que quiere tener.

Evitar trucos estudiados que te pueden dejar en seria desventaja. Las aperturas, como ya se estableció, son secuencias de movimientos bien estudiadas a lo largo del tiempo y a veces esconden ventajas estratégicas, si un jugador ignora los planes o ventajas de una apertura puede caer en trampas que arruinen su posición, le cuesten piezas clave o incluso perder la partida.

Prepararse para el medio juego. El lector puede imaginar una apertura como un árbol de decisión en el que cada decisión es un tablero y las diferentes respuestas son cada rama del árbol. Entre más se avanza en una partida, más ramas y posibilidades existen, por lo que una buena apertura es importante para tener opciones solidas (Figura 2.3). Después de esta etapa, la siguiente se llama medio juego y surge cuando termina de adoptarse una posición estudiada y comienzan movimientos con el objetivo siempre de llegar a la etapa final del juego en la que se da el jaque mate.

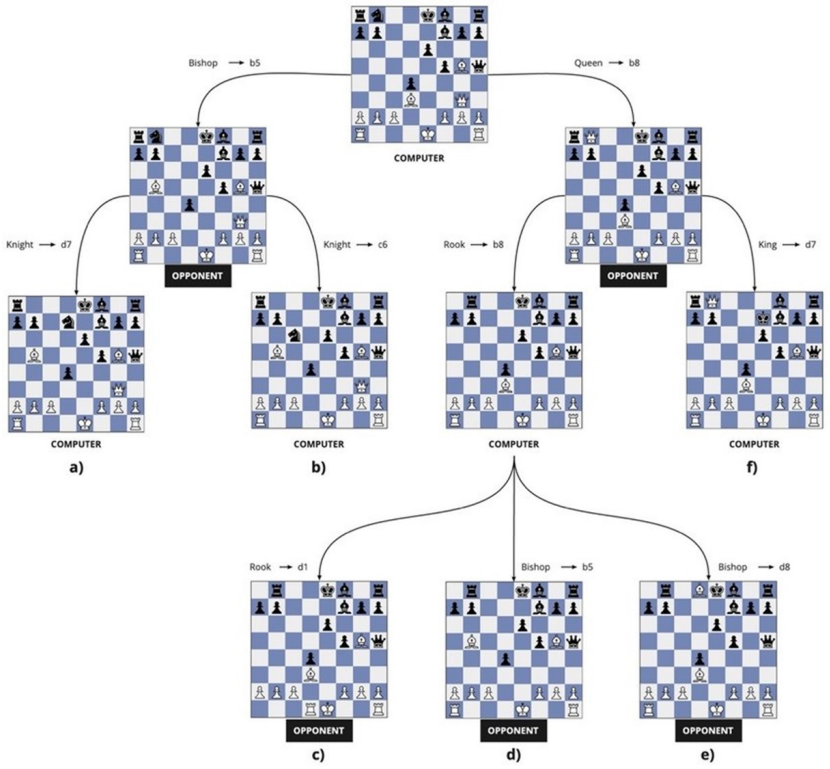

Figura 2.3: Ajedrez como árbol de decisión, ejemplo extraido de Research Gate

Según The Oxford Companion to Chess [7], que es un catálogo de conceptos de ajedrez, hasta el año de publicación existían alrededor

de 1327 aperturas o variantes de las mismas, entonces no es difícil imaginar la dificultad que enfrentan los jugadores que comienzan a jugar de forma habitual y cuando llegan a cierto nivel se enfrentan con oponentes mejor preparados y suelen entrar en desventaja.

3 Problema

En este capitulo se explica de manera explícita y breve el problema abordado.

3.1 Problema

En la etapa de aprendizaje para jugar ajedrez es primordial el estudio y entrenamiento de aperturas para aspirar a ser un jugador de nivel intermedio o superior y esto puede resultar abrumador para una persona de nivel principiante, que a penas entiende a nivel empírico las reglas más básicas, debido a que los recursos disponibles, como la lectura de libros de ajedrez o los páginas web, presentan limitaciones que pueden truncar el progreso del estudiante.

Hablando de los libros; el estudio de las aperturas es una tarea complicada porque se dividen en secciones que explican apertura por apertura y movimiento por movimiento (existen al menos 1300 variantes de apertura cada una con hasta 21 movimientos). Respecto a los recursos web y aplicaciones móviles; estos generalmente son de paga o están diseñados para estudio pasivo (presentando la misma limitante que los libros).

El prototipo desarrollado en esta investigación, por su parte, es interactivo y dinámico al jugar más de una variante de la misma partida en cada juego y brindarle al estudiante información valiosa después de cada movimiento a modo de retroalimentación.

4 Objetivos

En este capítulo se presenta el objetivo general y una lista breve de objetivos específicos cubiertos en este proyecto.

4.1 Objetivo general

Construir un sistema de aprendizaje activo para asistir el proceso de aprendizaje de apertura del ajedrez.

4.2 Objetivos específicos

- Extraer programáticamente las características posicionales de un turno específico de una partida de ajedrez.
- Implementar un codificador para construir un catálogo de datos posicionales a partir de partidas históricas.
- Diseñar e implementar un proceso de modelado predictivo para generar en cada turno un pronóstico de victoria o derrota a partir de las características de cada posición alcanzada durante una partida de ajedrez.
- Implementar una software que sea capaz de jugar aperturas al azar contra un usuario humano y le brinde en cada turno un pronóstico de victoria o derrota así como un despliegue de características posicionales en cada movimiento.

5 Revisión de literatura y recursos tecnológicos

En este capítulo se presenta una lista que puede ser consultada para profundizar el tema de aperturas de ajedrez, así como recursos tecnológicos para el mismo fin.

5.1 Literatura sobre aprendizaje de aperturas

Existen innumerables libros y recursos web para que los jugadores aprendan las aperturas. En lo que respecta a los libros, por nombrar algunos:

- Karpov, A. (2007). *El ajedrez. Aprender y progresar*. Badalona, Editorial Paidotribo.
- Chernev, I. (2001). *Ajedrez lógico jugada a jugada*. Barcelona, Editorial Paidotribo.
- Alburt, L. (2005). *Chess Openings for Black/White explained*. Ciudad: New York, Editorial Chess Information and Research Center.
- Nunn, J. (2004). *Learn Chess Tactics*. UK, Editorial Gambit.
- Wolff, P. (2005). *The complete idiot's guide to chess*. Editorial Alpha Books.
- Del Rosario, F. (2004). *A first book of morphy*. Columbia Británica, Editorial Trafford.
- Heisman, D. (2007). *Back to basic tactics*. Milford, Editorial Russell Enterprises.

La metodología de enseñanza que siguen la gran mayoría de estos libros consiste en mostrar una secuencia de ilustraciones comentadas en la que se presentan partidas de forma secuencial y los autores van narrando las ventajas de cada posición alcanzada. Esto suele ser abrumador para jugadores principiantes que apenas conocen las reglas o tienen poca experiencia.

5.2 *Recursos tecnológicos para aprendizaje de aperturas*

Dentro de los recursos tecnológicos existen algunas herramientas gratuitas o de paga que suelen tener versiones web, móvil o escritorio entre las cuales figuran:

- **Chess tempo base de datos** (gratuito o de paga según características). Es un explorador de aperturas en el que el jugador puede ver el número de veces que se ha jugado un movimiento, el porcentaje de veces que en partidas que jugaron ese movimiento ganaron blancas o negras, el nombre de las aperturas relacionadas entre otras cosas [8] (Figura 5.1).

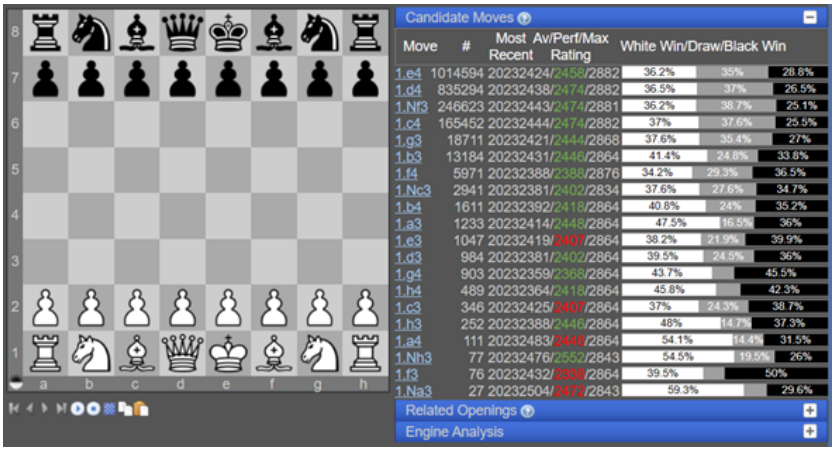

Figura 5.1: Explorador de aperturas Chesstempo

- **Chess position trainer** (de paga) es un software de escritorio que permite acceder a una base de datos de aperturas y entrenarlas una vez que fueron escogidas [9].
- **Chessbase Reader** 2017 (de paga) es un lector de partidas de ajedrez que muestra bases de datos de partidas y puede leer múltiples tipos de archivo (.cbh, .cbf, .pgn), usualmente usado por grandes maestros para sus análisis [10].
- **Chess Explorer**, de la plataforma online chess.com (de paga) Es un explorador de partidas de grandes maestros similar al ofrecido por chesstempo, pero que es actualizado en todo momento con partidas de grandes maestros en la misma plataforma. Para acceder a este explorador hay que pagar una membresía mensual [11].
- **365 Chess Opening explorer**, similar a los exploradores de chesstempo y chess.com. Es posible explorar aperturas iniciando con una posición FEN o desde la posición inicial [12].

6 Datos y métodos

En este capítulo se describen los procesos relacionados a la obtención y análisis exploratorio de los datos, que van desde la fuente de datos, licencia de uso, y una descripción contextual de su significado de acuerdo al contexto de este trabajo.

Además, se presentan las metodologías para el pre-procesamiento de los datos, ingeniería de variables, definición de modelos y su correspondiente implementación y optimización de hiperparámetros.

Finalmente, como marco de entendimiento para la presentación de resultados, se incluye una descripción de la metodología utilizada para llevar a cabo los experimentos realizados.

6.1 Datos

6.1.1 Obtención de datos

Los datos fueron obtenidos del siguiente dataset [1]
La tabla 6.1 detalla de manera técnica el conjunto de datos:

Nombre	Tipo	Valores faltantes	Valores únicos
`game_id`	Entero	no	20058
`rated`	Booleano	no	2
`turns`	Entero	no	211
`victory_status`	Objeto	no	4
`winner`	Objeto	no	3
`time_increment`	Objeto	no	400
`white_id`	Objeto	no	9438
`white_rating`	Entero	no	1516
`black_id`	Objeto	no	9331
`black_rating`	Entero	no	1521
`moves`	Objeto	no	18920
`opening_code`	Objeto	no	365
`opening_moves`	Entero	no	23
`opening_fullname`	Objeto	no	1477
`opening_shortname`	Objeto	no	128
`opening_response`	Objeto	18851	3
`opening_variation`	Objeto	5660	615

Tabla 6.1: Descripción técnica del conjunto de datos de ajedrez

6.1.2 *Descripción de las Columnas del Dataset de Ajedrez*

- **game_id**: Identificador único de cada juego. Es de tipo entero y esencialmente un índice en la base de datos.
- **rated**: Indica si el juego es clasificado (verdadero) o no (falso). Este factor puede influir en la seriedad con la que los jugadores abordan el juego, afectando el resultado final. No es tomado en cuenta para el modelo porque en los casos de uso del entrenador no son partidas clasificatorias. Sin embargo su estudio así como el de todas las columnas ayuda a entender mejor los datos, por lo que no se excluirán del análisis exploratorio 6.2.
- **turns**: Número de turnos en el juego. Entero. Este valor puede tener correlación con la duración de la partida e influir en el desempeño de los jugadores, especialmente bajo presión de tiempo. Las características posicionales pueden variar a medida que se eliminan piezas en el tablero o se mueven a posiciones en las que tengan control de más o menos casillas.
- **victory_status**: Muestra el tipo de conclusión de la partida (por ejemplo, 'Mate', 'Resign', 'Out of Time'). Esta columna es crucial para entender el contexto del resultado final y puede estar fuertemente correlacionada con 'winner', por lo que no se utiliza para el modelo final, pero si para el estudio de los datos. Se profundiza en el análisis exploratorio 6.2.
- **winner**: Esta es la variable objetivo, indica el ganador de la partida ('White', 'Black', 'Draw'). La correlación de esta columna con otras como 'white_rating' y 'black_rating' puede proporcionar información sobre las ventajas iniciales basadas en la habilidad. Esta correlación nos ayuda a entender cosas como que tipo de aperturas juegan los jugadores de alto nivel y cuales los principiantes. Profundizado en la análisis exploratorio 6.2.
- **time_increment**: Cadencia de tiempo en segundos añadidos por movimiento, lo que puede afectar la estrategia y la presión de tiempo experimentada por los jugadores. El diseño del prototipo 7 y del modelo 8.3 no toma en cuenta esta columna porque no se pone limite de tiempo al entrenamiento.
- **white_id** y **black_id**: Identificadores de los jugadores de las piezas blancas y negras, respectivamente. Aunque son principalmente etiquetas, el análisis de estos datos podría revelar jugadores con alto rendimiento. Se excluyen del modelo clasificador 8.3.
- **white_rating** y **black_rating**: Clasificación de habilidad de los jugadores blanco y negro. Estas son esenciales para predecir el

ganador, ya que un rating más alto generalmente indica un jugador más fuerte. Se excluyen del modelo clasificador 8.3 para evitar sesgos en la etapa de entrenamiento y limitar el modelo al entrenamiento con base en las características posicionales.

- **moves**: Secuencia de movimientos realizados durante el juego. El análisis de esta columna podría revelar estrategias comunes en juegos ganadores. A partir de este enfoque secuencial se genera una cadena fen, explicada en la sección 6.3.1, fundamental para el análisis posicional.
- **opening_code**: Código estándar que describe la apertura utilizada en el juego. La apertura puede estar correlacionada con la estrategia de juego y, potencialmente, con el resultado.
- **opening_moves**: Número de movimientos en la fase de apertura. Indica cuán extensa fue la apertura antes de transicionar al juego medio.
- **opening_fullname**, **opening_shortname**: Nombre completo y corto de la apertura jugada. Proporciona contexto sobre la estrategia inicial y su posible impacto en el resultado del juego.
- **opening_response** y **opening_variation**: Detalles específicos sobre la respuesta y variación de la apertura. Estas columnas son importantes para análisis detallados de estrategias de apertura.

6.2 *Análisis exploratorio de datos*

Estas son las columnas con datos faltantes.

opening_response: 18851

opening_variation: 5660

Para el caso de opening response, se elimina la columna porque no es necesaria para el análisis y es muy poco representativa (alrededor de 93.98 % de datos faltantes).

Respecto a la columna opening_variation, representa las variantes de las aperturas más tradicionales. Si no existe nombre para una variante, es porque esa fila representa la apertura en su variante 'tradicional', por tanto, se rellenan los valores faltantes con esta oración:

```
df_1['opening_variation'] =
    chess_data['opening_variation'].fillna('traditional opening')
df_1.info()
```

Después de esto, todas las columnas dejan de presentar valores faltantes:

#	Columna	Non-Null Count	Dtype
0	game_id	20058 non-null	int64
1	rated	20058 non-null	bool
2	turns	20058 non-null	int64
3	victory_status	20058 non-null	object
4	winner	20058 non-null	object
5	time_increment	20058 non-null	object
6	white_id	20058 non-null	object
7	white_rating	20058 non-null	int64
8	black_id	20058 non-null	object
9	black_rating	20058 non-null	int64
10	moves	20058 non-null	object
11	opening_code	20058 non-null	object
12	opening_moves	20058 non-null	int64
13	opening_fullname	20058 non-null	object
14	opening_shortname	20058 non-null	object
15	opening_variation	20058 non-null	object

Tabla 6.2: Descripción de las columnas del conjunto de datos

Al verificar la columnas numéricas white_rating y black_rating se decide hacerlas categóricas ya que al englobarlas en cuartiles es más fácil el estudio.

Clasificaciones:

- Beginner: Abajo de percentil 25 %
- Intermediate: Entre 25 % y 50 %
- Advanced: Entre 50 % y 75 %
- Master: Arriba de 75 %

Por tanto:

- Percentil 25 %: 1394
- Percentil 50 %: 1564
- Percentil 75 %: 1788

Con esta información se crean dos columnas: white_category y black_category

Como datos adicionales el promedio de Ratings de jugadores es de 1592.732. La mediana de Ratings es de 1564. Su desviación estandar es de 291.164 y 50 % de los datos está entre 1394 y 1788 con un Skew de la distribución: 0.280

6.2.1 *Distribuciones empíricas*

A modo informativo y para conocer más profundamente los datos. Se muestra esta visualización de las variables categóricas 6.1.

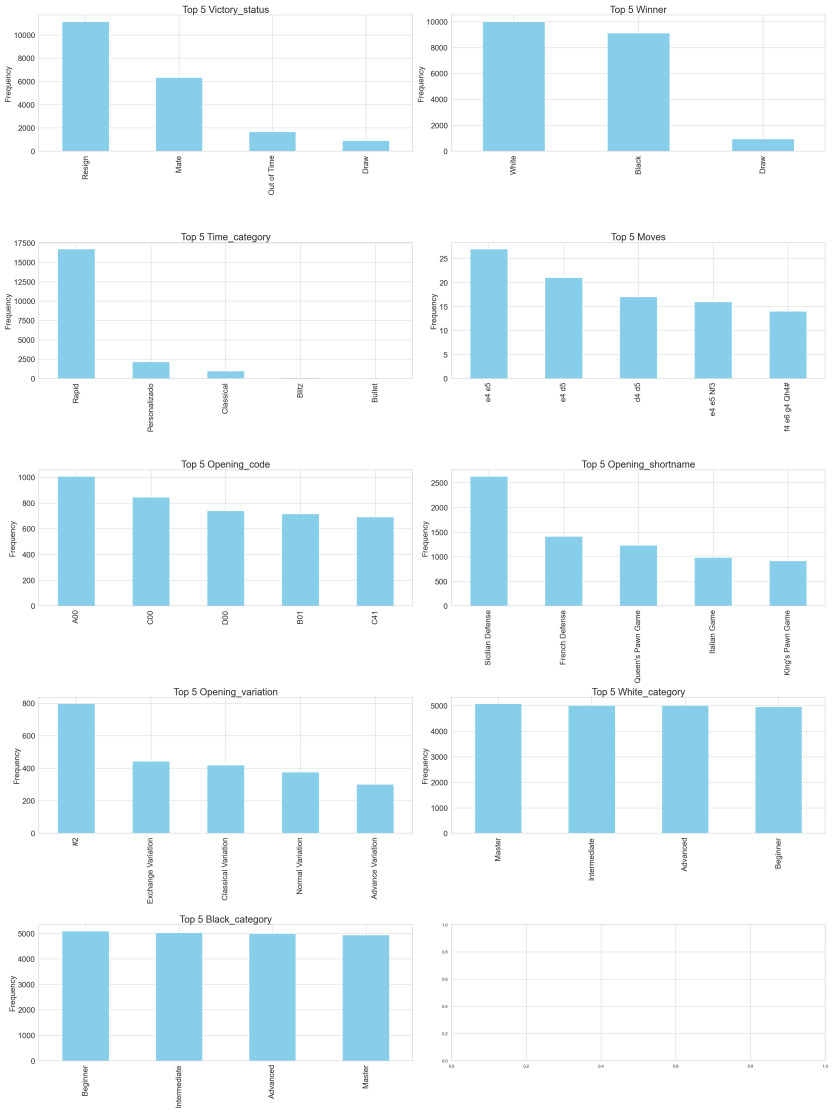

Figura 6.1: Columnas categóricas del conjunto de datos

Este gráfico 6.1 muestra algunas cosas como interesantes, como que hay más victorias por resign que por mate, lo que indica que el dataset tendrá muchas partidas que se resolvieron cuando un jugador se dio cuenta que su posición era muy mala o estaba cerca el jaque mate y concedió la victoria al oponente. También es posible observar que la probabilidad de que gane blancas es ligeramente más alta que negras. La mayoría de las partidas entraron en categoría de rápida (entre 6 y 30 minutos de tiempo base). La defensa más común es la siciliana y la

variante número 2. También es posible determinar que las categorías para blancas y negras se distribuyeron de forma correcta.

El gráfico 6.2 muestra histográmas de las variables numéricas. Es posible observar una asimetría sobre todo en opening moves y turns.

Veamos su Skewness y Kurtosis 6.3:

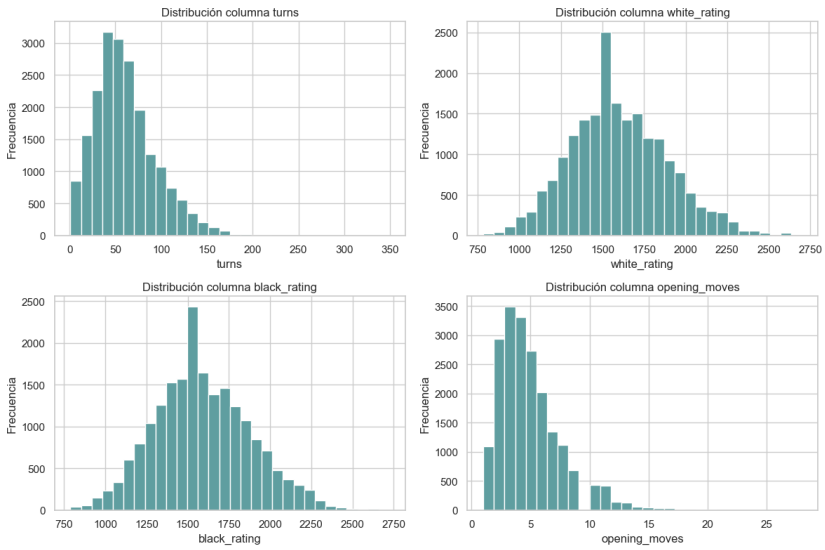

Figura 6.2: Columnas numéricas del conjunto de datos

	Asimetría	Curtosis
Turnos	0.897284	1.385161
Movimientos de apertura	1.334557	3.089694

Tabla 6.3: Asimetría y Curtosis de columnas Turns y Opening Moves

La presencia de la asimetría positiva en opening_moves indica que la distribución tiene una cola más larga hacia la derecha. Este comportamiento nos dice que hay más partidas con numero de turno superior al promedio, o lo que es lo mismo, que la mayoría de las partidas se resuelve entre 25 y 75 turnos. La curtosis es moderadamente alta, indicando que la distribución es más puntiaguda de lo normal, pero no es signo de alarma ya que refleja el comportamiento real.

La presencia de asimetría positiva en ambas variables indica un sesgo hacia partidas más largas. La curtosis elevada en opening_moves sugiere que aunque la mayoría de las partidas tienen a un patrón de más movimientos de apertura (4-8), existen casos notables que se desvían de ese patrón, esto quiere decir que hay aperturas atípicas 6.3 de muchos movimientos que son poco usadas:

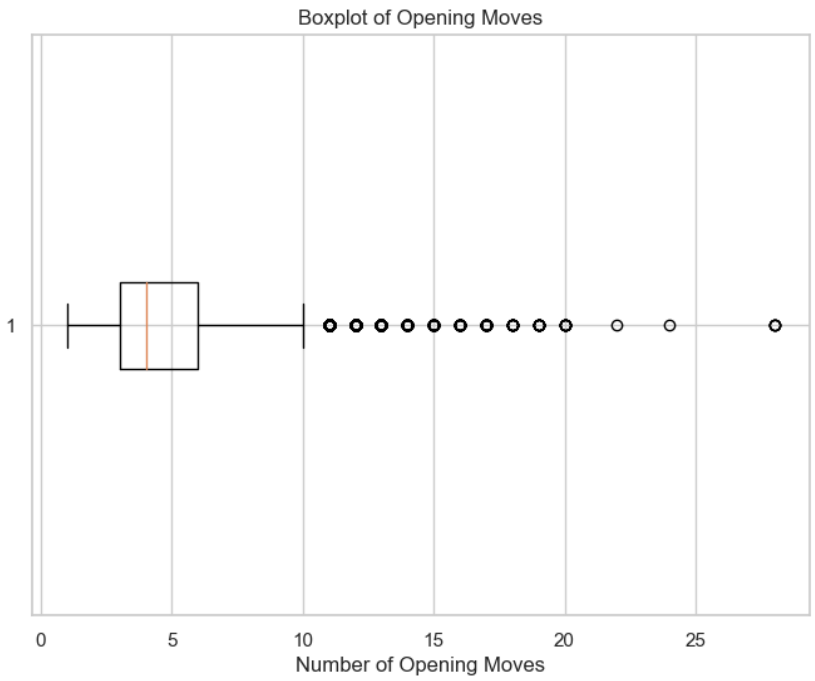

Figura 6.3: Boxplot Movimientos de apertura

En el siguiente gráfico es posible observar el elo, nivel o rating de los jugadores en este dataset, siendo 1500 predominante. Las líneas verticales discontinuas representan el percentil 25 y el 75 del conjunto combinado de rating. Estos indican que la mayoría de los jugadores tienen una rating entre estos dos valores: 1394 y 1788. Rango que es más corto que el resto de los niveles. Lo que indica que se necesite un salto de calidad para superar esta puntuación y que hay cierto estancamiento en este rango. Esta afirmación da fuerza al planteamiento de este trabajo, ya que pretende ser una herramienta didáctica para aprender aperturas de manera casi natural y reforzar el aprendizaje con datos importantes de cada posición alcanzada.

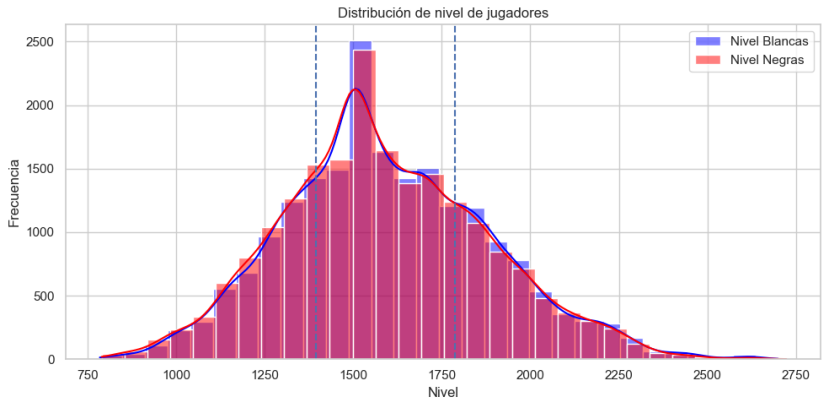

Figura 6.4: Nivel de jugador por frecuencia de partidas

En el gráfico 6.5 se visualiza con exactitud el balance de la clase objetivo: ganador. Hay un 49.9 % de probabilidad de victoria simplemente por el hecho de comenzar con blancas, situación que se ve más claramente en alto nivel. También se observa que un 55 % de las victorias fueron producidas por el medio de la rendición, situación que ya se mencionó, ayuda al modelo.

Para mitigar un poco el desbalance de las clases y dado que el objetivo es entender que tan ganadora o perdedora es la posición, en la etapa de entrenamiento se elimina el empate del entrenamiento.

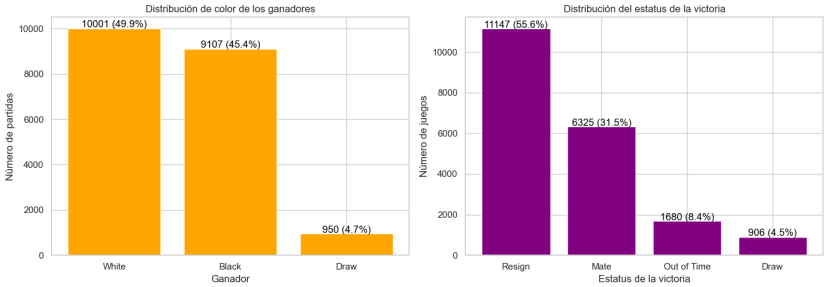

Figura 6.5: Relación a través de las partidas entre la victoria y el color

6.2.2 *Aperturas más comunes por nivel y frecuencia de juego*

Para sustentar la teoría de que los jugadores que llegan a cierto nivel necesitan aprender aperturas para dar un salto de calidad y subir, hay que explorar la frecuencia en la que se juegan las aperturas más comunes en este dataset según el rating de los jugadores en la siguiente ilustración 6.6

Sabiendo que las categorías están perfectamente balanceadas, rápidamente se puede notar que la frecuencia con que las aperturas más populares se juegan va disminuyendo, esto es porque los jugadores avanzados van conociendo y dominando más variedad de aperturas. Este gráfico da fuerza a la teoría de que para dar un salto de calidad en ajedrez hay que aprender aperturas y entender, como ya se dijo, sus ventajas y desventajas posicionales y tácticas.

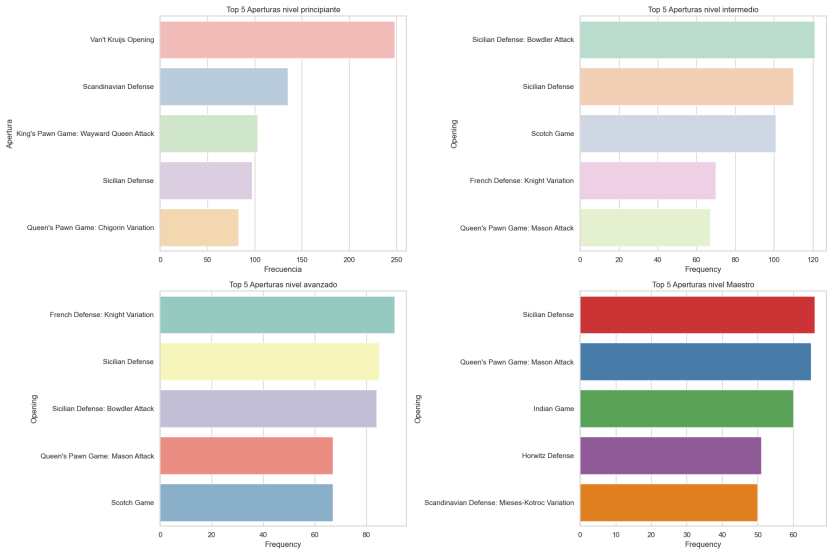

Figura 6.6: Aperturas más comunes por nivel y su frecuencia de juego

Respecto a nivel principiante, el hecho de que Van't Kruijs Opening sea la apertura más popular no indica que sea buena, para entender el problema habrá que visualizar esta apertura 6.7 donde se entiende por qué es la más común entre principiantes: los jugadores que inician en el ajedrez no conocen aperturas y juegan de forma empírica a penas conociendo las reglas. Es posible deducir esto porque habiendo tantas posiciones con nombre, esta posición es la única dentro de la secuencia de movimientos siguientes que recibió nombre en sus partidas, osea que a partir de esta posición el desarrollo del juego es caótico, con poca idea de la posición. Estos jugadores suelen aprender únicamente por memoria muscular, al perder partidas y recordar las posiciones en que se tuvo alguna desventaja. Esto no es malo, es una curva de aprendizaje, pero es allí donde el aprendizaje empírico encuentra el limite y el jugador se estanca en un nivel y se vuelve fundamental conocer las aperturas para subir.

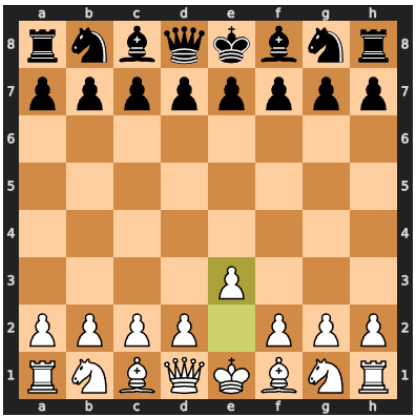

Figura 6.7: Van't Kruijs Opening

Ahora hay que echar un vistazo a las aperturas más comunes desde el punto de vista de maestros y cómo han terminado esas partidas:

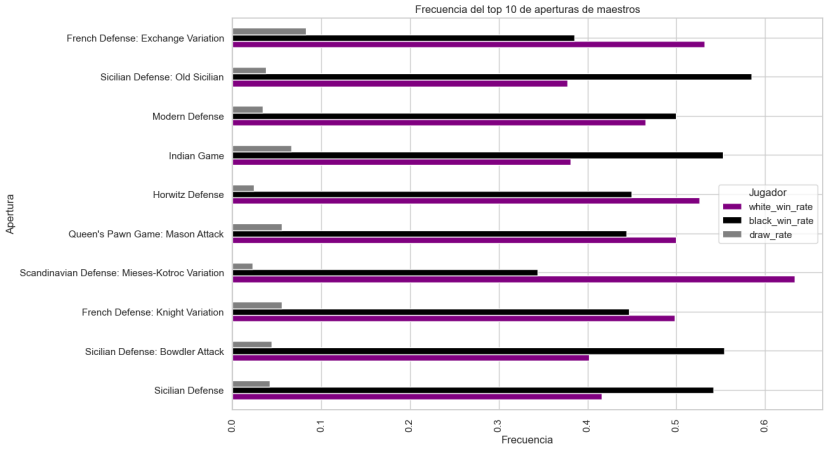

Figura 6.8: Frecuencia de juego aperturas maestros y color ganador

De este gráfico se puede concluir que:

- La Defensa Francesa, Horwitz, Peón de Reina (variante ataque Mason), así como la Defensa Escandinava (variante Mieses-Kotroc) son las defensas que mejores resultados dan para las blancas.
- Algunas variantes de la Defensa Siciliana, así como la Defensa India, son las aperturas que mejores resultados ofrecen para las negras.
- Es posible entonces decir que un buen jugador de ajedrez domina al menos estas aperturas y sus características posicionales y tácticas.

6.2.3 *Correlación de variables con la variable objetivo*

Para el modelo ganador 8.3, las variables seleccionadas incluyen las creadas con ingeniería de características 6.3.

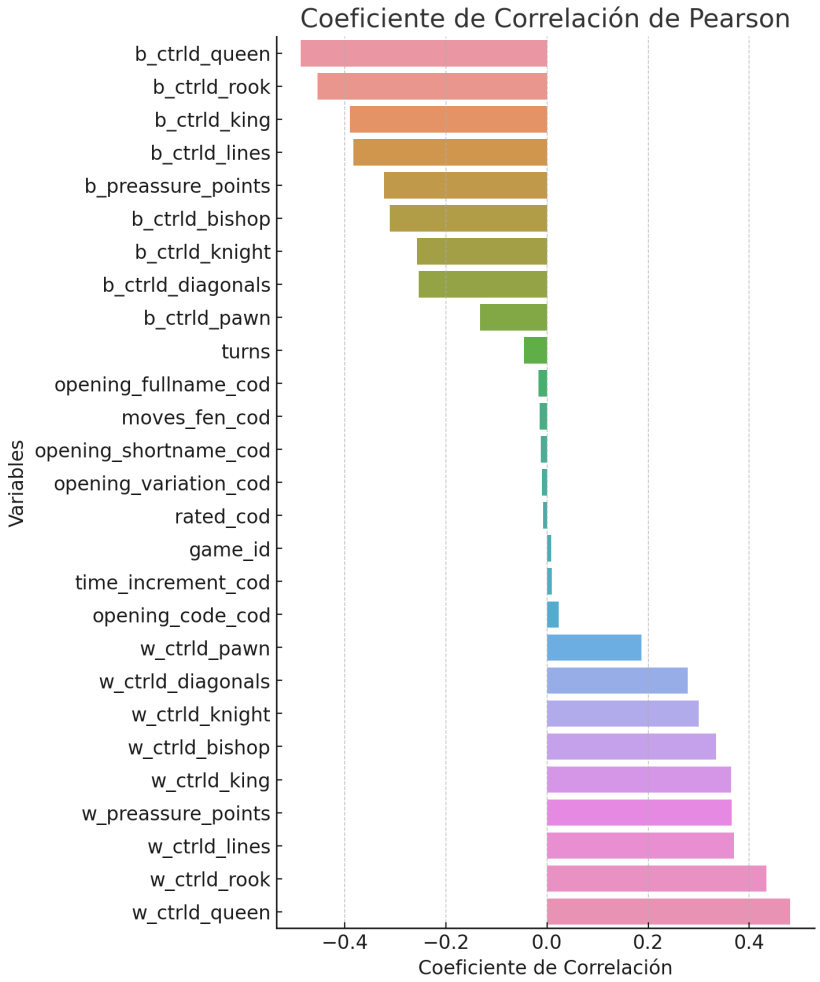

Figura 6.9: Coeficiente de correlación de pearson de variables con winner

De la imágen 6.9 se puede inferir:

- **Variables Negativamente Correlacionadas:** Las variables que están más correlacionadas negativamente con el ganador, indicando una influencia hacia la victoria de las negras cuando aumentan, incluyen varios indicadores de control del tablero por parte de las piezas negras como `b_ctrld_queen`, `b_ctrld_rook`, `b_ctrld_king`, estas representan las casillas controladas por reina, torre y rey de piezas negras.
- **Variables Positivamente Correlacionadas:** De manera similar, las variables positivamente correlacionadas indican una influencia hacia

la victoria de las blancas cuando aumentan, como `w_ctrld_queen`, `w_ctrld_rook`, etc.

- **Linealidad:** Las correlaciones muestran que ciertas variables tienen una asociación lineal moderada con el ganador. Esto sugiere que aunque algunos aspectos del control del tablero pueden predecir linealmente el resultado.

6.3 *Ingeniería de características: Descripción de los nuevas características*

Para construir un modelo que dada una posición específica clasificara victoria o derrota, era necesario extraer una serie de características que suelen considerar los grandes maestros cuando están evaluando su propia posición y la del adversario. Esta serie de características se consiguen a partir de una cadena FEN, que es una representación de la posición de todas y cada una de las piezas del tablero. Con este objetivo se creó el archivo *extract_features.py* presente en el repositorio que puede ser encontrado en 11, en el que se codificaron 22 funciones para extracción de características que se explicarán desde una perspectiva visual en este las siguiente secciones.

6.3.1 *Moves Fen*

La primera y muy importante característica es la representación de los movimientos secuenciales guardados en notación algebraica (SAN) como una cadena que represente la posición alcanzada y pueda ser utilizada para el estudio de la misma (FEN).

FEN, por sus siglas en inglés "Forsyth-Edwards Notation". Es un método estándar para describir posiciones de ajedrez usando texto. A través de cadenas FEN. Se pueden replicar estas posiciones en infinidad de herramientas computacionales y sitios web como chess.com.

Pero, ¿Cómo se describe una posición usando una cadena FEN? La primera cosa que se tiene que entender es que un tablero de ajedrez se puede descomponer en 8 filas, que en inglés reciben el nombre de ranks". Cada fila se convierte en una cadena de texto que la representa y esas 8 cadenas de texto se compactan juntas separadas por diagonales (/) para formar la cadena FEN[13].

Esto se puede ver claramente en la imagen 6.10:

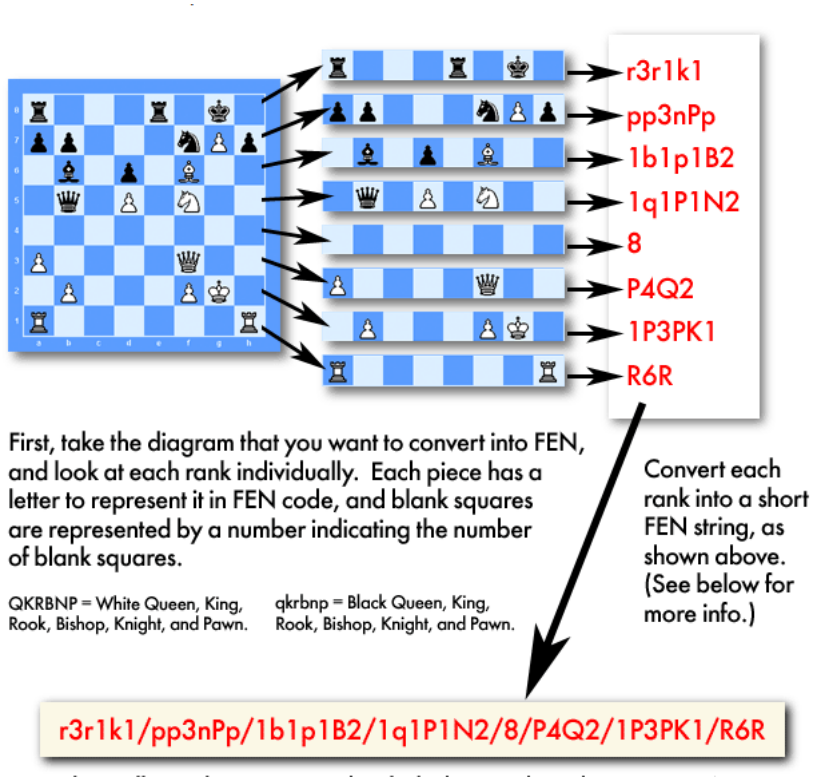

Figura 6.10: Cadena FEN, composición

Las piezas están representadas por sus iniciales en inglés, y se utiliza una letra mayúscula para Blancas y minúscula para negras 6.11

Esta cadena se extrae a partir de la columna 'moves' que representa todos los movimientos hechos en una partida, con la intención de tener

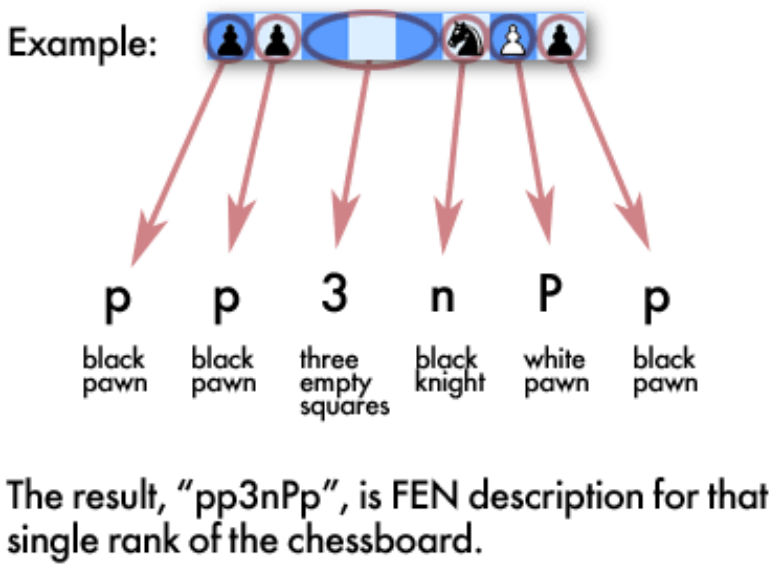

Figura 6.11: Cadena FEN, una sola fila

una representación de la posición para poder estudiarla.

6.3.2 *Turn*

La segunda característica que se extrae a partir de la cadena FEN es el turno, en el que sencillamente se guarda un 1 si el turno es de blancas y un 0 si el turno es de negras, esto para darle una perspectiva al modelo de quién evalúa la posición.

6.3.3 *Control de casillas*

Las características de control de casillas se generan con dos funciones, la que alimenta el modelo que entrega un número entero y la que informa al usuario las casillas. La función que alimenta el modelo cuenta las casillas controladas por cada pieza, esto es, las casillas en las que puede comer una pieza o bien en el siguiente turno o bien si alguna pieza enemiga se moviera a esa casilla. Esta característica es importante porque uno de los objetivos de una buena posición es controlar más casillas centrales para que el rival no se mueva libremente y entre en el territorio enemigo.

Peón: En la imagen se puede observar un equilibrio en el control de las casillas centrales, visto desde la posición del juegador blanco.

Figura 6.12: Casillas controladas por peón

Caballo: Cuenta el número de casillas que el caballo está atacando al mismo tiempo, observar que si el caballo está ubicado en casillas centrales controla más casillas que un caballo ubicado en la orilla. Dicho esto, en lo que respecta a esta característica, el jugador que controle más casillas tiene sus caballos mejor colocados.

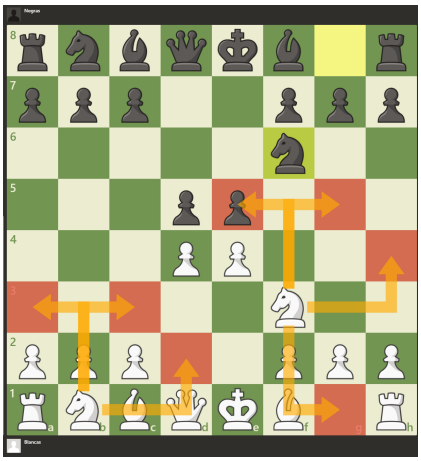

Figura 6.13: Casillas controladas por caballo

Alfil: Cuenta el número de casillas por las que el alfil puede moverse libremente para atacar y por tanto controla. Observar que en esta posición, los peones avanzados hacia el centro permiten que el alcance de los alfiles se extienda. Una característica vital para entender la calidad de la posición.

Figura 6.14: Casillas controladas por alfil

Torre: Número de casillas por las que la torre puede moverse libremente. Entre más casillas se controlen, mejor indicativo de una buena posición.

Figura 6.15: Casillas controladas por torre

Reina: Número de casillas a las que la Reina, que es la pieza más poderosa del tablero, puede acceder libremente y por tanto está atacando. Nótese que en esta posición la Reina blanca accede a una diagonal con mayor número de cuadros que la negra, por tanto ataca más piezas y se puede concluir que está mejor colocada. En lo que respecta a la reina, en el ejemplo de la imagen, la posición blancas es superior.

Figura 6.16: Casillas controladas por Reina

Rey: Casillas por las que puede moverse libremente el rey, en ocasiones funcionan como casillas de escape.

Figura 6.17: Casillas controladas por Rey

Diagonales: A diferencia de las demás características, esta cuenta el número de diagonales que son controladas por un jugador y no de casillas. Puede indicar posiciones abiertas o cerradas. Controlar mayor número de diagonales suele indicar una mejor posición porque el oponente no se puede mover libremente.

Figura 6.18: Diagonales controladas por Reina y Alfiles

Lineas: Similar a la característica de diagonales controladas, pero enfocada en las lineas horizontales y verticales. Una característica muy importante para ataques combinados en los que torres y dama pueden ser fundamentales para generar combinaciones ganadoras.

Figura 6.19: Lineas controladas por Reina y Torres

6.3.4 *Puntos de presión (>1 atacando)*

Los puntos de presión son casillas enemigas que son atacadas por más de una pieza. Son una de las características más importantes a la hora de evaluar una posición.

En el ejemplo de la imagen se indican en rojo los puntos de presión del jugador blanco. Con las flechas se ejemplifica que el peón negro de d5 está siendo atacado al mismo tiempo por la torre blanca en d1 y el alfil blanco en g2. También se ilustra con las flechas el punto de presión en c4. Los demás puntos de presión no tienen flechas para no complicar el gráfico, pero todos y cada uno son atacados por más de una pieza blanca a la vez. Generalmente los puntos de presión son debilidades del adversario, y, para mejorar la posición, el oponente atacado tiene que agregar defensores a la pieza vulnerable. En este caso, el peón negro en d5 solo tiene un defensor (el peón de c6), pero no sería buena idea que las blancas tomaran el peón porque perderían piezas de mayor valor (ya sea el alfil o la torre).

Figura 6.20: Puntos de presión

6.4 Descripción de las métricas

6.4.1 Accuracy

La métrica de *Accuracy* (exactitud) es ampliamente utilizada para evaluar la efectividad de modelos de clasificación en estadística y aprendizaje automático. Se define como la proporción de predicciones correctas (tanto positivas como negativas) entre el total de predicciones realizadas. Matemáticamente, se expresa como:

$$\text{Accuracy} = \frac{\text{NúmeroPrediccionesCorrectas}}{\text{NúmeroTotalPredicciones}}$$

Aunque es una medida intuitiva y de fácil comprensión, el *Accuracy* puede no ser una métrica fiable en escenarios donde existen clases desequilibradas. En tales casos, se recomienda complementar el análisis con otras métricas como la precisión y el recall, que pueden ofrecer una visión más detallada del rendimiento del modelo [14] [15] [16].

6.4.2 Precision

La métrica de *Precision* es fundamental en la evaluación de modelos de clasificación. Se define como la proporción de identificaciones positivas que fueron correctas, calculada por:

$$\text{Precision} = \frac{\text{Verdaderos Positivos}}{\text{Verdaderos Positivos} + \text{Falsos Positivos}}$$

Esta métrica es crucial en contextos donde los falsos positivos conllevan grandes costos. Por ejemplo, en ajedrez, una alta *Precision* significa que la mayoría de las predicciones de victoría fueron correctas[14] [16].

6.4.3 Recall

La métrica de *Recall*, también conocida como sensibilidad, tasa de verdaderos positivos o cobertura, es esencial para evaluar modelos de clasificación. El *Recall* se define como la proporción de verdaderos positivos identificados por el modelo respecto al total de casos realmente positivos. Se calcula con la fórmula:

$$\text{Recall} = \frac{\text{Verdaderos Positivos}}{\text{Verdaderos Positivos} + \text{Falsos Negativos}}$$

Esta métrica es particularmente valiosa en situaciones donde es crítico detectar todos los casos positivos, como en la detección de enfermedades o en la prevención de fraudes. En este caso la victoria en ajedrez. El *Recall* es prioritario en escenarios donde los falsos negativos representan un riesgo mayor que los falsos positivos. Que no es el caso, pero ayuda tenerlo en cuenta[14] [16].

6.4.4 *Brier Score*

El *Brier Score* es una métrica de evaluación para predicciones probabilísticas, utilizada ampliamente en la predicción de eventos. En el contexto de este documento, el *Brier Score* se utiliza para evaluar la exactitud de las predicciones sobre los resultados de las partidas, midiendo cuán cerca están las probabilidades predichas de los resultados reales y la diferencia entre las probabilidades pronosticadas y los resultados observados. Se calcula mediante la fórmula:

$$\text{Brier Score} = \frac{1}{N}\sum_{i=1}^{N}(p_i - o_i)^2$$

Donde p_i representa la probabilidad predicha de un resultado específico y o_i el resultado real (1 si ocurre, 0 de lo contrario). Esta métrica es particularmente valiosa en ajedrez para modelos predicativos que evalúan la probabilidad de diversos resultados como victorias o derrotas [17] [18].

6.4.5 *Roc Curve*

La Curva ROC (Receiver Operating Characteristic) es una herramienta esencial en el análisis de modelos de clasificación binaria. Representa gráficamente la relación entre la tasa de verdaderos positivos (sensibilidad) y la tasa de falsos positivos (1-especificidad) a varios umbrales de decisión. Es especialmente valiosa para identificar el umbral que balancea de manera óptima la sensibilidad y especificidad. La Curva ROC se acompaña típicamente del AUC (Area Under the Curve), una métrica que cuantifica el área total bajo la curva ROC. Un AUC de 1 indica un modelo perfecto, mientras que un AUC de 0.5 se asocia con un rendimiento aleatorio [19] [20].

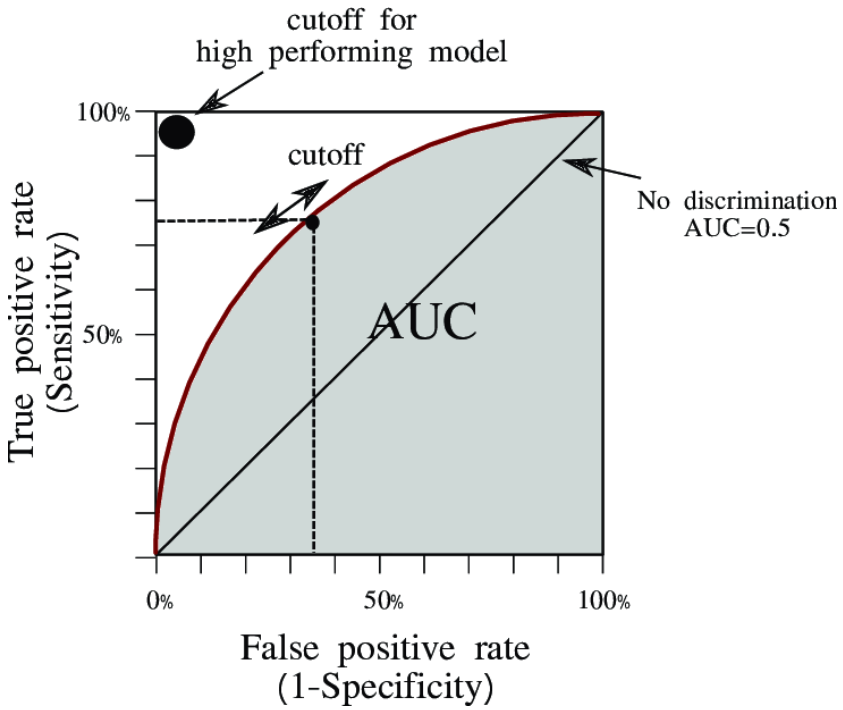

Figura 6.21: Roc curves [2]

6.5 Descripción de los modelos

6.5.1 Máquina soporte vectorial kernel lineal

Las Máquinas de Soporte Vectorial (SVM) son modelos ampliamente utilizados en aprendizaje automático para tareas de clasificación y regresión. La versión con kernel lineal de la SVM utiliza el producto escalar entre vectores de características para determinar el hiperplano óptimo de separación. Matemáticamente, el kernel lineal se define como:

$$K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$$

Este enfoque es particularmente útil cuando los datos son linealmente separables, permitiendo a la SVM maximizar el margen entre las clases adyacentes al hiperplano, lo cual es crucial para una buena generalización del modelo [21] [22].

6.5.2 Máquina soporte vectorial kernel polinomial

El kernel polinomial es una función de kernel utilizada en las máquinas de soporte vectorial para transformar los datos de entrada en un espacio de características más complejo. Se define por la fórmula:

$$K(\mathbf{x}, \mathbf{y}) = (c + \mathbf{x}^T \mathbf{y})^d$$

donde c es un coeficiente constante y d es el grado del polinomio. Este kernel es capaz de modelar relaciones complejas entre las clases al aumentar la dimensionalidad del espacio de características, lo cual es crucial para clasificaciones que no son linealmente separables en el espacio original[23] [22].

6.5.3 Máquina soporte vectorial kernel rbf

El kernel RBF (Radial Basis Function), o kernel gaussiano, es una función de kernel utilizada ampliamente en máquinas de soporte vectorial para clasificaciones donde los datos de entrada no son linealmente separables. Se define por la expresión:

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\gamma \|\mathbf{x} - \mathbf{y}\|^2\right)$$

donde γ es un parámetro que regula la amplitud de la función gaussiana. Este parámetro es crucial porque determina la velocidad con la que la función de similaridad disminuye con el aumento de la distancia entre las muestras. Un γ adecuadamente elegido permite al modelo capturar la complejidad de los datos mientras evita el sobreajuste [24] [22].

6.5.4 *LightGBM*

LightGBM es un marco de aprendizaje automático que utiliza algoritmos de aumento de gradiente para construir modelos predictivos eficientes y eficaces. La eficiencia de LightGBM proviene de su algoritmo de construcción de árboles de decisión basado en histogramas y dos técnicas principales: *GOSS* (Gradient-based One-Side Sampling) y *EFB* (Exclusive Feature Bundling).

GOSS mantiene los datos con gradientes grandes, que proporcionan información más valiosa, y muestrea aleatoriamente los datos con gradientes pequeños. Esto mantiene el equilibrio de la distribución de datos mientras reduce la cantidad de cálculos necesarios.

LightGBM mejora la eficiencia de los algoritmos de boosting de gradiente al mantener todas las instancias con gradientes grandes y realizar un muestreo aleatorio en las instancias con gradientes pequeños. Esto se realiza con la intención de utilizar instancias más informativas para el aprendizaje. El proceso se define como:

$$\text{GOSS} = \begin{cases} \text{Mantiene todas las instancias con gradientes grandes (ej., los datos con los } a \times 100\,\% \text{ mayores gradientes)} \\ \text{Muestrea aleatoriamente una porción } (1-a) \text{ de las instancias con gradientes pequeños} \end{cases} \tag{6.1}$$

donde a es un hiperparámetro del algoritmo GOSS que determina la proporción de datos conservados.

El EFB se basa en la observación de que las características en conjuntos de datos espaciados tienen una baja correlación no nula. EFB agrupa eficientemente las características para reducir la dimensionalidad con una pérdida mínima de información, lo que permite un procesamiento más rápido. El algoritmo se puede describir como:

$$\text{EFB} = \sum_{i=1}^{n} \min_{\mathcal{B}} \left\{ \sum_{f_j, f_k \in \mathcal{B}, j \neq k} \text{conflict}(f_j, f_k) \right\} \tag{6.2}$$

donde $\mathcal{B}$ representa un paquete de características y $\text{conflict}(f_j, f_k)$ es una medida de qué tan a menudo las características f_j y f_k son no nulas simultáneamente.

LightGBM construye un histograma de las características y utiliza estos histogramas para dividir los nodos del árbol, lo que es mucho más eficiente en términos de memoria y velocidad que los métodos basados en listas o matrices [25] [26].

6.5.5 *XGboost*

XGBoost (eXtreme Gradient Boosting) es una implementación de árboles de decisión potenciados por gradientes que es notable por su eficiencia y efectividad. La función objetivo en XGBoost es una combinación de una función de pérdida y una función de regularización.

La función objetivo de XGBoost se define como:

$$\text{Obj} = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{6.3}$$

donde:

- y_i son los valores reales,
- $\hat{y}_i$ son las predicciones del modelo,
- f_k son los árboles individuales,
- K es el número de árboles.

La regularización Ω de cada árbol se calcula como:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{6.4}$$

donde:

- T es el número de hojas del árbol,
- w son los pesos de las hojas,
- γ y λ son parámetros que controlan la penalización por número de hojas y los pesos de las hojas, respectivamente.

El proceso de optimización en XGBoost utiliza gradientes para minimizar la función objetivo, ajustando iterativamente los árboles para corregir errores residuales basados en el gradiente negativo de la función de pérdida.

La capacidad de XGBoost para manejar efectivamente datos desbalanceados lo hace adecuado para la clasificación de resultados de partidas de ajedrez, donde algunas partidas son más caóticas que otras.

Gracias a su regularización integrada, XGBoost evita el sobreajuste, esencial para desarrollar modelos que generalicen bien a nuevas partidas.

XGBoost mejora iterativamente, permitiendo afinar las predicciones hasta alcanzar un alto grado de precisión, ideal para clasificación de partidas de juegos de estrategia con reglas bien definidas [27] [28].

6.5.6 *Efecto de los Hiperparámetros en Modelos de Aprendizaje Automático*

Los hiperparámetros son cruciales para configurar los modelos de aprendizaje automático, afectando directamente su capacidad de aprendizaje y generalización. A continuación, se describe el impacto de tres hiperparámetros comunes, los cuales serán considerados en este trabajo de grado para optimizar los modelos xgboost:

- **Tasa de aprendizaje (learning rate)**: Un valor crucial que controla la velocidad a la que un modelo aprende. Una tasa de aprendizaje demasiado alta puede causar que el modelo converja demasiado rápido a una solución subóptima, mientras que una tasa muy baja puede ralentizar el proceso de entrenamiento, necesitando más épocas para alcanzar una convergencia efectiva [29].
- **Profundidad máxima (max depth)**: Especialmente en modelos basados en árboles, define la profundidad máxima de los árboles. Limitar la profundidad ayuda a prevenir el sobreajuste, asegurando que los árboles no creen reglas demasiado complejas que no generalicen bien a nuevos datos [30].
- **Submuestra (subsample)**: Esta proporción define cuántas instancias de datos se utilizan para entrenar cada árbol en métodos de boosting. Valores inferiores a 1 pueden aumentar la diversidad de los modelos, evitando el sobreajuste al reducir la varianza y aumentar el sesgo [31].

6.6 Descripción de los experimentos o simulaciones

6.6.1 Experimento 1

Respecto a los modelos, primero se corrieron los benchmarks con la configuración 1 mostrada en la tabla 6.4.

En esta configuración se hacía un análisis posicional relativo que solo incluía la posición del jugador en turno. Los resultados de este experimento pueden ser consultados en la sección 8, como adelanto, las métricas no fueron favorables.

Configuración	Columnas
1	turns, ctrld_pawn, ctrld_knight, ctrld_bishop, ctrld_rook, ctrld_queen, ctrld_king, preassure_points, controlled_diagonals, controlled_lines

Tabla 6.4: Configuración 1 de columnas para análisis de datos de ajedrez

Respecto al desarrollo del entrenador de ajedrez. En el primer experimento se mostraba al ser humano en cada turno únicamente datos sobre su posición, lo que dificultaba que evaluara correctamente pros y contras al no visualizar la posición del oponente (la máquina) esto se corrigió en la segunda iteración de ingeniería de características (Experimento 2).

Otro problema del experimento 1 fue que la máquina no simulaba de forma correcta la aleatoriedad con que un humano juega aperturas porque no existían pesos en esta decisión: era posible que jugara con la misma probabilidad aperturas que en el dataset de juegos históricos habían sido jugadas una sola vez (Australian Defense) o aperturas jugadas más de 2000 veces (Sicilian Defense). Lo mismo sucedía con los movimientos, dentro de una apertura había la misma probabilidad de que se jugará el movimiento más común que el menos común.

Esto se corrige en el experimento 2.

6.6.2 Experimento 2

En la configuración 2 6.5 el análisis es absoluto y toma en cuenta la posición tanto de blancas como de negras.

El mejor modelo se seleccionó considerando principalmente el accuracy porque interesa maximizar el número de clases predichas de forma correcta y el brier score porque interesa minimizar la diferencia entre las probabilidades pronosticadas y los resultados observados. Después se consideró el tiempo de entrenamiento. Tomando en cuenta la posibilidad de que este modelo pueda implementarse en producción.

Respecto al entrenador de ajedrez, el problema con las probabilidades se resolvió introduciendo el concepto de pesos relativos, sacando las probabilidades reales tanto de que un jugador juegue ciertas aperturas, como ciertos movimientos y tomándolas en cuenta para las

Configuración	Columnas
2	game_id, rated, turns, winner, time_increment, white_rating, black_rating, moves, opening_code, opening_moves, opening_fullname, opening_shortname, opening_variation, moves_fen, current_turn, w_ctrld_pawn, w_ctrld_knight, w_ctrld_bishop, w_ctrld_rook, w_ctrld_queen, w_ctrld_king, w_preassure_points, w_ctrld_diagonals, w_ctrld_lines, b_ctrld_pawn, b_ctrld_knight, b_ctrld_bishop, b_ctrld_rook, b_ctrld_queen, b_ctrld_king, b_preassure_points, b_ctrld_diagonals, b_ctrld_lines, rated_cod, winner_cod, current_turn_cod, time_increment_cod, opening_code_cod, opening_fullname_cod, opening_shortname_cod, opening_variation_cod, moves_fen_cod

Tabla 6.5: Configuración 2 de columnas para análisis de datos de ajedrez

decisiones de la máquina.

Después de introducir el concepto de pesos relativos, se tuvo el problema de que, debido a que había pesos definidos, había aperturas o movimientos cuya probabilidad de que fueran jugados era tan baja que jamás se jugarían, esto se resolvió haciendo una normalización para ajustar los pesos menores a 1 como 1, por ejemplo la relación 13.12:.0.0049 fue ajustada a 13:1 siendo el primer número el peso para la apertura más jugada (Sicilian Defense) y el segundo la menos jugada (Australian Defense). Algo similar se realizó para la elección de movimientos por parte de la máquina y está documentado en el repositorio de código que puede ser consultado en 11.

7 Prototipo

En este capitulo se describe materialmente el prototipo desarrollado y se muestran algunas capturas de la funcionalidad del mismo.

7.1 Prototipo openings_IAlex

El prototipo desarrollado recibe el nombre de openings_IAlex y es un sistema de aprendizaje activo que es capaz de jugar un juego de ajedrez iniciando alguna de las 128 aperturas entrando en cualquiera de sus 1477 variantes de ajedrez durante el juego y sirve cómo entrenador, porque brinda información de la posición y un bias de victoria después de cada movimiento jugado (Humano o máquina).

Esta es una captura de un juego 7.1 en el que la máquina escoge jugar con blancas la defensa francesa y en el transcurso se alcanzan 3 variantes de la misma y la probabilidad de victoria va fluctuando conforme la posición va cambiando.

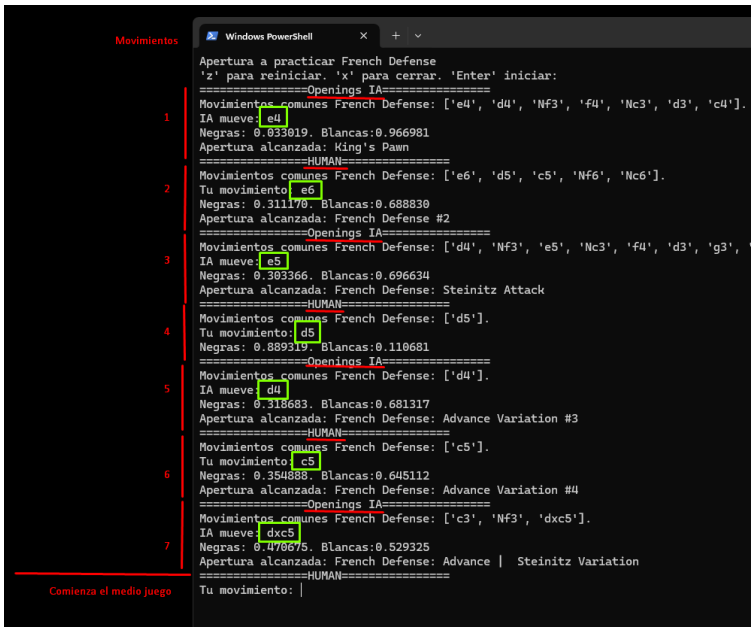

Figura 7.1: Juego de openings IAlex, interacción en consola

Y esta es una captura de pantalla en la que se aprecia la parte gráfica que se muestra al usuario durante el juego 7.2.

Figura 7.2: Juego de openings IAlex, despliegue gráfico

Y esta es la captura del archivo que muestra las características de la posición e información generada por la máquina para que el estudiante entienda las ventajas y desventajas del movimiento realizado. 7.3.

Figura 7.3: Características de la posición y de la apertura

8 Resultados

En este capítulo se enlistan los resultados obtenidos por cada experimento científico que se hizo. Se analizan todos los datos obtenidos en conjunto y se concluye con las capacidades técnicas finales del prototipo.

8.1 Resultados obtenidos experimento 1

Estos son los resultados del primer ciclo del método científico, en la primera configuración de datos, en la que se tomaba en cuenta solo la posición del jugador en turno:

Modelo	Acu (%)	Prec (%)	Reca (%)	Brier
SVM K = linear	51.60 %	26.62 %	51.60 %	0.25
SVM K = poly	51.58 %	49.40 %	51.58 %	0.25
SVM K = rbf	52.85 %	52.64 %	52.85 %	0.25
XGBoost	88.68 %	88.68 %	88.68 %	0.09
LightGBM	88.75 %	88.75 %	88.75 %	0.10

Tabla 8.1: Resultados de modelos de clasificación en el conjunto de Test (Configuración 1)

8.2 Resultados obtenidos experimento 2 y definitivo

Por otra parte, estos son los resultados de los modelos evaluando ambas posiciones, negras y blancas a la vez:

Modelo	Acu (%)	Prec (%)	Reca (%)	Brier
SVM K = linear	90.37 %	90.37 %	90.37 %	0.07
SVM K = poly	90.53 %	90.55 %	90.53 %	0.07
SVM K = rbf	82.07 %	84.52 %	82.07 %	0.11
XGBoost	91.45 %	91.45 %	91.45 %	0.06
XGBoost opt. hip.	**91.73 %**	**91.74 %**	**91.73 %**	**0.06**
LightGBM	91.49 %	91.49 %	91.49 %	0.06

Tabla 8.2: Resultados de modelos de clasificación en el conjunto de Test (Configuración 2)

8.3 Discusión de resultados

Durante el desarrollo de este prototipo, se obtienen las características posicionales descritas en los objetivos y se usan de forma satisfactoria para entrenar el modelo de clasificación.

El modelo seleccionado como el mejor es el de la tabla 'Modelo ganador' 8.3, que resulta tener un brief score satisfactiorio indicando que la diferencia entre las probabilidades pronosticadas y los resultados observados es mínima y tiene un accuracy bueno, que indica que el numero de clases predichas de forma correcta es alto.

Modelo	Acu (%)	Prec (%)	Reca (%)	Brier
XGBoost opt. hip.	**91.73 %**	**91.74 %**	**91.73 %**	**0.06**

Tabla 8.3: Modelo ganador

Estos fueron los hiperparámetros 8.4 del modelo ganador:

Hiperparámetro	Valor
Tasa de aprendizaje	0.0886
Profundidad máxima	9
Submuestra	0.7356

Tabla 8.4: Mejor conjunto de hiperparámetros

En la matriz de confusión 8.1 se observa que las Blancas predichas como negras o las negras predichas como blancas son pocas, al rededor del 10 %, mientras que el 90 % del tiempo, el modelo acierta.

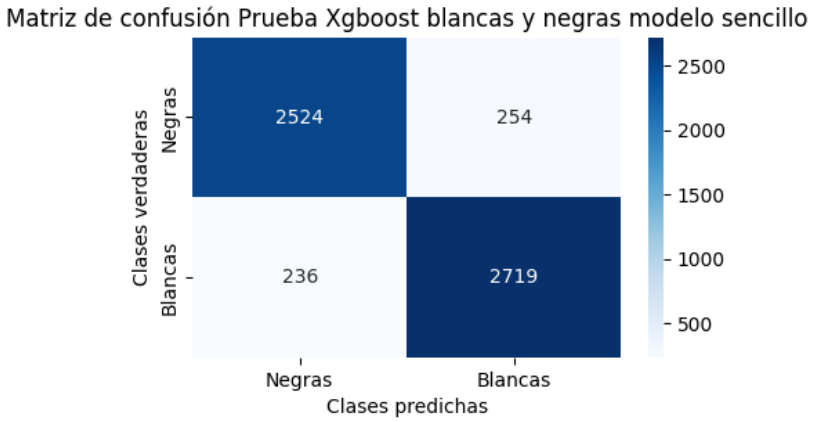

Figura 8.1: Matriz de confusión modelo ganador

En el gráfico de precisión vs recall 8.2 (derecha) ambas clases muestra una curva de precisión y recall casi perfecta a través del rango. Esto representa que el modelo tiene un excelente desempeño en la clasificación de ambas clases (victoria de blancas o negras). La intersección de las curvas de precisión y recall ocurre en un punto alto, indicando que el modelo es capaz de alcanzar un equilibrio entre precisión y recall sin sacrificar mucho.

Respecto a la curva ROC 8.2 (izquierda), esta presenta alta área bajo la curva y la curva de cada clase se acerca mucho al borde superior izquierdo del gráfico, indicando que el modelo tiene un excelente desempeño discriminativo entre las dos clases.

Se muestra una tasa de verdaderos positivos alta con una baja tasa de falsos positivos para casi todo el rango, lo que resulta en una curva ROC que tiende a la perfección y un AUC cercano a 1. Esto indica que el modelo tiene alto rendimiento.

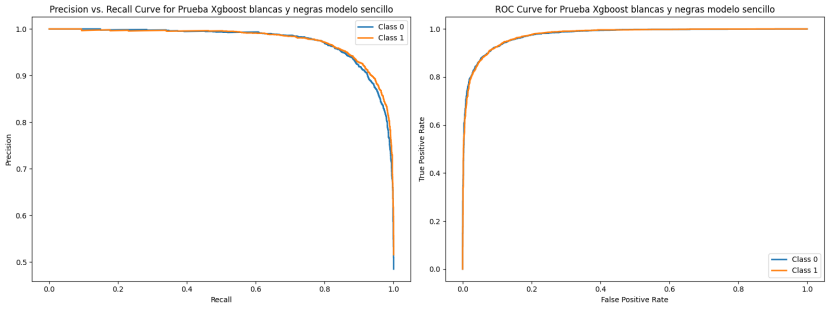

Figura 8.2: Precisión vs Recall y Curva ROC modelo ganador

Por otra parte, el sistema de aprendizaje elije de forma pseudo aleatoria una apertura e interactúa con el oponente humano emulando el comportamiento de un humano entrenador siendo capaz de responder a jugadas y mostrando correctamente las características de la posición, lo que cumple satisfactoriamente los objetivos propuestos.

9 Trabajo futuro

En este capítulo se menciona un listado de propuestas que pueden trabajarse para robustecer el proyecto.

9.1 Pasos siguientes del prototipo

- Producción de sugerencias de mejora basadas en datos: recopilar información sobre el estilo de juego del usuario al usar el sistema y producir recomendaciones personalizadas que ayuden a los jugadores principiantes a mejorar su juego durante la fase de la apertura.
- Mejorar el aspecto gráfico para que sea un sistema más intuitivo y fácil de usar. Por ejemplo, una página web con un dashboard digital que muestra las casillas controladas y la apertura alcanzada y permita mover las piezas interactuando con la interfaz gráfica.
- Entrenar el modelo con partidas más actuales usando la APIs de las páginas de ajedrez más comunes como chess.com y lichess.com.
- Incrementar el número de características que se muestran al usuario y que se usan para la predicción tales como:
  - Calidad de casillas controladas en el centro
  - Piezas defendiendo al rey
  - Piezas atacando al rey enemigo
  - Suma de la calidad de las piezas en el tablero
  - etc.

10 Sesgos e implicaciones

Este capitulo aborda la identificación, reflexión y análisis de los sesgos potenciales, se desarrolla una reflexión sobre cómo estos afectan la interpretación de los resultados y la toma de decisiones basada en el modelo y se mencionan los esfuerzos para mitigar estos sesgos, como el diseño cuidadoso del experimento, la selección equitativa de datos y la validación cruzada del modelo con varios conjuntos de datos.

En el desarrollo de este prototipo se identificaron 2 sesgos principales:

10.0.1 Sesgo en la selección de hiperparámetros

Origen del Sesgo: La selección y optimización de hiperparámetros puede introducir sesgos si se limitan a rangos estrechos basados en suposiciones o experiencias previas. Esta restricción puede llevar a que el modelo no se ajuste de manera óptima para generalizar a nuevos datos. Además, optimizar basándose exclusivamente en métricas que no consideran la variabilidad del modelo frente a diferentes conjuntos de datos puede llevar a una evaluación sesgada de su capacidad.

Implicaciones: Un modelo con hiperparámetros seleccionados de manera sesgada puede presentar un rendimiento subóptimo en situaciones reales, limitando su aplicabilidad. Las decisiones tomadas basadas en los resultados de este modelo podrían ser incorrectas si el modelo no representa adecuadamente la diversidad de contextos en los que se espera operar.

Acciones: Para evitar, en la medida de lo posible este sesgo, se probaron diferentes combinaciones de hiperparámetros. Y en cada combinación de evaluaron diferentes métricas que considereban la variabilidad del modelo, utilizando, por ejemplo, la validación cruzada comparando contra conjunto de datos de validación.

10.0.2 Sesgo en la recopilación y tratamiento de datos

Origen del Sesgo: Los sesgos en la recopilación de datos pueden ocurrir si los datos no son representativos de la población o del fenómeno estudiado. Un conjunto de datos de entrenamiento

compuesto predominantemente por ejemplos de un subgrupo específico sesgará los resultados del modelo hacia las características de ese subgrupo. Esto puede ser común cuando los datos son más accesibles o más fáciles de recopilar, pero no capturan la totalidad de la variabilidad del sistema.

Implicaciones: Un modelo entrenado con datos sesgados puede llevar a conclusiones incorrectas. Por ejemplo, en el ámbito de la ajedrez, el clasificador podría mostrar un rendimiento deficiente cuando se le presenten posiciones de partidas jugadas por aficionados o jugadores intermedios, ya que no ha aprendido las características comunes de estas partidas. Esto limita la aplicabilidad del modelo en entornos donde se requiera una evaluación equitativa y precisa de partidas de ajedrez de diversos niveles.

Acciones: Durante el desarrollo del prototipo se procuro especialmente el balance entre las clases de ganadores en partidas de diferentes niveles de juego y que existiera una cantidad suficiente de partidas y aperturas jugadas. También en la etapa de entrenamiento se eliminó el empate de la clasificación.

11 Notas de implementación

En este capitulo se encuentra una descripción de la localización del repositorio y el formato del proyecto tecnológico.

Repositorio del proyecto en GitHub disponible en la siguiente URL:

`https://github.com/AlexNoelHdz/CHESS_openings`

11.1 Estructura del Proyecto (carpetas)

- TOG: Proyecto de obtención de grado
- Data: Carpeta para almacenar los conjuntos de datos utilizados o generados.
- Notebooks: Cuadernos Jupyter para análisis exploratorio y procesamiento de datos.
- Pickles/models: Modelos entrenados y sus metadatos.
- Stockfish: La inteligencia artificial Stockfish se incorpora como auxiliar una vez que pasa la fase de apertura, para las fases de medio juego final.

11.2 Acerca de

En la etapa de aprendizaje del ajedrez, aprender las aperturas es esencial para aspirar a ser un jugador de nivel intermedio y esto puede ser abrumador para una persona de nivel principiante porque leer libros sobre este tema es una tarea complicada ya que están divididos en secciones donde explican de forma secuencial las aperturas y los movimientos con una serie de tableros (hay al menos 1,300 variantes de apertura, cada una con hasta 21 movimientos). Además, los recursos en la web y las aplicaciones móviles del mercado presentan limitaciones como que generalmente son de pago o están diseñadas para un estudio pasivo.

Este proyecto es un sistema de aprendizaje didáctico donde puedes jugar contra una computadora que juega aperturas de manera aleatoria y te brinda retroalimentación constante sobre la posición.

11.3 Getting started

Estas instrucciones te permitirán obtener una copia del proyecto en funcionamiento en tu máquina local para fines de desarrollo y prueba.

Para instalar el proyecto y sus dependencias ejecuta en consola:

```
git clone https://github.com/AlexNoelHdz/CHESS_openings
cd CHESS_openings
pip install -r requirements.txt
```

Después de clonar el proyecto simplemente ejecuta en consola:

```
py .\play_chess.py
```

11.4 Autores

- @AlexNoelHdz

12 Apéndice

En esta capitulo se muestra para su consulta rápida el fragmento de código del modelo ganador y su optimización de hiperparámetros.

12.1 Modelo ganador y optimización de hiperparámetros

```
# Libreras
import pandas as pd
from sklearn.model_selection import train_test_split
# Warnings
import warnings
warnings.filterwarnings('ignore')
# Internal tool and helpers
import model_helpers as mh
import xgboost as xgb
from hyperopt import fmin, tpe, hp, STATUS_OK, Trials
from sklearn.metrics import accuracy_score
import numpy as np
import matplotlib.pyplot as plt
CLASS_NAMES = ["Negras", "Blancas"]

data_path = "../CHESS/data/df_3_cod.csv"
df_3 = pd.read_csv(data_path)

# X e Y
X = df_3.copy()
y_name = "winner_cod"
# X es el dataframe eliminando la variable de salida. Eliminando
    tambin 'moves' que ya est representado
X = X.drop(columns=['rated','winner',y_name,
    'current_turn','time_increment','opening_code','opening_fullname','opening_shortname','opening_variation','moves_
X = X.drop(columns=['game_id', 'white_rating', 'black_rating',
    'moves', 'current_turn_cod', 'opening_moves', 'rated_cod',
    'current_turn_cod', 'time_increment_cod', 'opening_code_cod',
    'opening_fullname_cod', 'opening_shortname_cod',
    'opening_variation_cod', 'moves_fen_cod'])# Y es un array
    unidimensional (ravel) de la variable de salida
Y = df_3[y_name].ravel()
print(X.columns)
```

```
# divisin en train y test
X_train, X_test, Y_train, Y_test = train_test_split(X,
    Y,test_size=0.3)

# XGBOOST Simple
# Crear un clasificador XGBoost
xgboost = xgb.XGBClassifier()
# Entrenar el modelo en los datos de entrenamiento
xgboost.fit(X_train, Y_train)
# Evaluacin del modelo
Yhat_xgboost_test = xgboost.predict(X_test)
Yhat_xgboost_train = xgboost.predict(X_train)
Yhat_xgboost_test_prob = xgboost.predict_proba(X_test)
Yhat_xgboost_train_prob = xgboost.predict_proba(X_train)
mh.eval_perform_multi_class(Y_test,Yhat_xgboost_test,
    Yhat_xgboost_test_prob,CLASS_NAMES,"Prueba Xgboost blancas y
    negras modelo sencillo")
mh.eval_perform_multi_class(Y_train,Yhat_xgboost_train,
    Yhat_xgboost_train_prob,CLASS_NAMES,"Entrenamiento Xgboost
    blancas y negras modelo sencillo")

# Optimizacin de hiperparmetros XGboost
# Define the hyperparameter space
space = {
   'max_depth': hp.choice('max_depth', np.arange(1, 14,
       dtype=int)),
   'learning_rate': hp.loguniform('learning_rate', -5, -2),
   'subsample': hp.uniform('subsample', 0.5, 1)
}

# Define the objective function to minimize
def objective(params):
   xgb_model = xgb.XGBClassifier(**params)
   xgb_model.fit(X_train, Y_train)
   y_pred = xgb_model.predict(X_test)
   score = accuracy_score(Y_test, y_pred)
   return {'loss': -score, 'status': STATUS_OK}

trials = Trials()
# Perform the optimization
best_params = fmin(objective, space, algo=tpe.suggest,
    max_evals=250, trials=trials)
print("Best set of hyperparameters: ", best_params)
# Extraer la funcin de prdida de cada resultado en trials
losses = [x['result']['loss'] for x in trials.trials]

# Graficar la funcin de prdida
plt.figure(figsize=(10, 5))
plt.plot(losses, label='Funcin de Prdida')
plt.xlabel('Iteraciones')
plt.ylabel('Prdida Negativa de Exactitud')
```

```
plt.title('Funcin de Prdida durante la Optimizacin de
    Hiperparmetros')
plt.legend()
plt.show()

# Crear un clasificador XGBoost a partir de los mejores parmetros
xgboost_best = xgb.XGBClassifier(**best_params)
# Entrenar el modelo en los datos de entrenamiento
xgboost_best.fit(X_train, Y_train)
# Evaluacin del modelo
Yhat_xgboost_test = xgboost_best.predict(X_test)
Yhat_xgboost_train = xgboost_best.predict(X_train)
Yhat_xgboost_test_prob = xgboost_best.predict_proba(X_test)
Yhat_xgboost_train_prob = xgboost_best.predict_proba(X_train)
mh.eval_perform_multi_class(Y_test,Yhat_xgboost_test,
    Yhat_xgboost_test_prob,CLASS_NAMES,"Prueba Xgboost blancas y
    negras optimizacin de hiperparmetros")
mh.eval_perform_multi_class(Y_train,Yhat_xgboost_train,
    Yhat_xgboost_train_prob,CLASS_NAMES,"Entrenamiento Xgboost
    blancas y negras optimizacin de hiperparmetros")

# Feature importance
import matplotlib.pyplot as plt
# Feature importance
importance = xgboost.feature_importances_

df_importancia = pd.DataFrame({
   'Caracteristica': X.columns,
   'Importancia': importance
})

umbral = 0
# Filtrar los nombres de las caractersticas importantes
df_importancia = df_importancia[df_importancia['Importancia'] >
    umbral]
df_importancia = df_importancia.sort_values(by='Importancia',
    ascending=True)

features = df_importancia['Caracteristica']
importance = df_importancia['Importancia']


# Crear un grfico de barras horizontal
plt.figure(figsize=(10, 8))
plt.barh(features, importance, color='skyblue')
plt.xlabel('Importance')
plt.ylabel('Features')
plt.title('Feature importance')
plt.show()
```

```
# Validacin del modelo
import extract_features as ef
fen = '1r1q1rk1/pb2bppp/4p3/6N1/2pPQ3/4P2P/PP3PP1/R1B2RK1 w - - 1
    17'
moves = 4
new_sample_df = ef.count_all_features(fen, moves)
print(new_sample_df.T)

print("Posicin Blancas: ")
pd.set_option('display.max_columns', None)
print(ef.get_all_features_uf(fen, True))

print("Posicin Negras: ")
pd.set_option('display.max_columns', None)
print(ef.get_all_features_uf(fen, False))


# Decisin del modelo para clasificacin
prob = xgboost.predict_proba(new_sample_df)
print(xgboost.classes_) # 0: negras. 1: blancas
print(f"Negras: {'{:.6f}'.format(prob[0][0])}.
    Blancas:{'{:.6f}'.format(prob[0][1])}")
prob = xgboost_best.predict_proba(new_sample_df)
print(xgboost_best.classes_) # 0: negras. 1: blancas
print(f"Negras: {'{:.6f}'.format(prob[0][0])}.
    Blancas:{'{:.6f}'.format(prob[0][1])}")


'''
Guardar modelo elegido
En un entorno de produccin tiene menos costo computacional
    seleccionar el modelo ms sencillo para entrenar con nuevos
    datos, y la variacin es despreciable. Por tanto se selecciona
    ese modelo.
'''
import pickle
pickle.dump(xgboost,
    open("./pickles/models/xgoboost_model0416_18:06.pkl", 'wb'))
```

Bibliografía

[1] D. Sikka, "Online chess match prediction." https://www.kaggle.com/code/dhruvsikka/online-chess-match-prediction/input, Jan. 2022. Accessed on: 2024-04-14.

[2] M. R. Hernández, R. E. Pruneda, and J. M. R. Díaz, "Statistical analysis of the evolutive effects of language development in the resolution of mathematical problems in primary school education," 2021. Available at https://www.researchgate.net/publication/351506473_Statistical_Analysis_of_the_Evolutive_Effects_of_Language_Development_in_the_Resolution_of_Mathematical_Problems_in_Primary_School_Education#pf8.

[3] J. H. McMillan and D. R. Forsyth, "What theories of motivation say about why learners learn," *New Directions for Teaching and Learning*, vol. 45, no. 2, pp. 39–51, 1991.

[4] B. Fischer, S. Margulies, and D. Mosenfelder, *Bobby Fischer Teaches Chess*. Bantam Books, second ed., 1972.

[5] G. D. Luca, "How important are chess openings? (7 reasons to study)." https://chesspulse.com/how-important-are-openings-in-chess, Jan. 2012. Accessed on: 2024-01-29.

[6] L. Alburt, R. Dzindzichashvili, and E. Perelshteyn, *Chess openings for white explained*. Chess information and research center, first ed., 2007.

[7] D. Hooper and K. Whyld, *The oxford companion to chess*. Brithis Library Cataloguing, first ed., 1984.

[8] C. Tempo, "Chess tempo." https://old.chesstempo.com/game-database.html, Dec. 2023.

[9] C. position trainer, "Chess position trainer." http://www.chesspositiontrainer.com/index.php/en/buy, Dec. 2019.

[10] C. Reader, "Chessbase reader 2017." https://en.chessbase.com/pages/download, Dec. 2017.

[11] Chess.com, "Chess.com." https://www.chess.com/home, Dec. 2017.

[12] . chess, "365 chess." https://www.365chess.com/opening.php, Dec. 2023.

[13] C. S. LLC, "Learning forsyth-edwards notation." https://www.chessgames.com/fenhelp.html, 2024. Accessed on: 2024-04-15.

[14] G. James, D. Witten, T. Hastie, and R. Tibshirani, "An introduction to statistical learning," 2013.

[15] D. M. W. Powers, "Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation," 2011.

[16] J. Han, J. Pei, and M. Kamber, "Data mining: Concepts and techniques," 2011.

[17] G. W. Brier, "Verification of forecasts expressed in terms of probability," 1950.

[18] "Applying statistical methods to evaluate game outcomes: A chess perspective," 2022. Whitepaper on Statistical Methods in Games.

[19] T. Fawcett, "An introduction to roc analysis," 2006.

[20] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," 1997.

[21] C. Cortes and V. Vapnik, "Support-vector networks," 1995.

[22] B. Schölkopf and A. J. Smola, "Learning with kernels: Support vector machines, regularization, optimization, and beyond," 2002.

[23] J. Shawe-Taylor and N. Cristianini, "Kernel methods for pattern analysis," 2004.

[24] C.-W. Hsu and C.-J. Lin, "A practical guide to support vector classification." Department of Computer Science, National Taiwan University, 2003.

[25] "Lightgbm documentation." https://lightgbm.readthedocs.io, 2021. Official documentation for LightGBM.

[26] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree." https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf, 2017. Advances in Neural Information Processing Systems 30.

[27] T. Chen, "Xgboost: A scalable tree boosting system," 2016. Available at https://arxiv.org/abs/1603.02754.

[28] "Xgboost documentation." https://xgboost.readthedocs.io, 2021. Comprehensive guide and reference to XGBoost.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[30] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, 2 ed., 2009.

[31] R. E. Schapire and Y. Freund, *Boosting: Foundations and Algorithms*. MIT Press, 2013.