Universidad de Lima

Facultad de Ingeniería

Carrera de Ingeniería de Sistemas

DETECCIÓN DE TRAMPAS EN PARTIDAS DE AJEDREZ EN LÍNEA UTILIZANDO CLASIFICADORES DE MACHINE LEARNING

Tesis para optar el Título Profesional de Ingeniero de Sistemas

Rodrigo Ayvar Rios

Código 20170132

Asesor

Oscar Efrain Ramos Ponce

Lima – Perú

Julio de 2024

Detección de trampas en partidas de ajedrez en línea utilizando clasificadores de Machine Learning

Rodrigo Ayvar Ríos
20170132@aloe.ulima.edu.pe
Universidad de Lima

Resumen: El ajedrez en línea ha tenido un auge de popularidad desde el año 2020, con el contexto de la pandemia. Este auge trae consigo la problemática de las trampas en torneos de ajedrez en línea, las cuales comprometen su integridad y reputación, perjudicando tanto a jugadores como organizadores. Paralelamente, se han utilizado diferentes técnicas de Machine Learning en ajedrez, en los ámbitos de clasificación y regresión, principalmente para la predicción de resultados. Sin embargo, existen pocos estudios sobre la detección de trampas en ajedrez, principalmente por la dificultad de acceso a partidas con trampas en línea. Por este motivo, el objetivo de esta investigación fue desarrollar cuatro modelos de detección de trampas con clasificadores de Machine Learning, utilizando partidas entre humanos y computadoras para caracterizar el comportamiento de jugadores haciendo trampas. Se construyó un dataset de entrenamiento a partir de las partidas de Free Internet Chess Server y un dataset de validación con partidas propias en el sitio web Chess.com. Además, se añadieron variables de análisis y de tiempo, sugeridas por investigaciones previas. El mejor modelo fue Random Forest, que obtuvo exactitudes del 96.534% con partidas del dataset de Free Internet Chess Server y del 96% con partidas en Chess.com. Esta investigación aporta al estado del arte, proporcionando modelos de detección de trampas en partidas de ajedrez con una mayor exactitud y alcance que estudios previos, siguiendo la definición de trampas hecha por la entidad reguladora de ajedrez en el mundo.

Palabras Clave: Ajedrez, Clasificación, Aprendizaje Supervisado

Abstract: Online chess has had a rise in popularity since 2020, with the context of the pandemic. This rise brings with it the problem of cheating in online chess tournaments, which compromises their integrity and reputation, harming both players and organizers. In parallel, different Machine Learning techniques have been used in chess in the classification and regression fields. However, there are few studies on the detection of cheating in chess, mainly due to the difficulty of accessing cheating games online. For this reason, the objective of this research was to develop four cheat detection models with Machine Learning classifiers, using games between humans and computers to characterize the cheaters' behavior. A dataset was built from the Free Internet Chess Server games and a validation dataset with own games on the Chess.com website. In addition, analysis and time variables were added, suggested by previous research. The best model was Random Forest, which obtained accuracies of 96.534% with games from the original dataset and 96% with games on Chess.com. This research contributes to the state of the art, providing models for detecting cheating in chess games with greater accuracy and reach than previous studies, following the definition of cheating made by the chess regulatory entity in the world.

Keywords: Chess, Classification, Supervised learning.

1. INTRODUCCIÓN

El ajedrez en línea ha tenido un gran auge en su popularidad en los últimos años: por ejemplo, chess.com, el sitio de ajedrez más popular de internet, casi triplicó su número de usuarios en 2021, con respecto a 2020. Con este aumento de popularidad, los eventos de ajedrez en línea han obtenido más auspiciadores y, por lo tanto, más dinero para los participantes. Por ejemplo, en 2021 se jugó el torneo Grand Chess Tour, con 1,275,000 USD en premios (Grand Chess Tour, 2021).

Ante la proliferación del ajedrez en línea, se hace evidente que una gran desventaja de organizar eventos de ajedrez en línea con grandes cantidades de dinero en juego es la posibilidad de realizar trampas en las partidas. FIDE (2021), en su reglamento oficial para competiciones en línea, define a las trampas como "el uso deliberado de asistencia externa por un jugador para ganar una ventaja sobre el oponente". Este artículo se centra en la detección de trampas realizadas al consultar con un motor de ajedrez que sugiera las mejores jugadas a los competidores.

Un ejemplo de trampas en ajedrez en línea es el caso del Gran Maestro Tigran Petrosian, que fue sancionado de por vida por Chess.com por hacer trampas en una partida, supuestamente utilizando un motor de ajedrez como ayuda para realizar sus jugadas. Esto sucedió en la final de la Pro Chess League, una competición entre naciones, causando que su país, Armenia, fuera descalificado (ChessBase, 2020). Casos como el de Petrosian ponen en riesgo la integridad de los torneos de ajedrez en línea y perjudican tanto a los organizadores como a los propios jugadores, por lo que es necesario desarrollar herramientas que permitan detectar posibles trampas en las partidas de ajedrez en línea.

Machine Learning (ML) es una rama de Inteligencia Artificial, que desarrolla modelos utilizando diferentes algoritmos iterativos para encontrar patrones complejos sin ser programados explícitamente (Bishop, 2006). Los modelos de ML tienen una gran aplicabilidad, principalmente, en problemas de clasificación, regresión y *clustering* (Janiesch et al., 2021).

Estos últimos años ha habido un auge en la implementación de técnicas de ML en ajedrez. Quizá el proyecto más renombrado de ML en ajedrez fue el proyecto de AlphaZero de Google, una IA basada en redes neuronales. Dicha IA tuvo un gran éxito y fue considerada la AI más precisa de ese entonces, derrotando a Stockfish 8 en 2018 (Silver et al., 2018). Anteriormente, se ha implementado regresión lineal y clasificadores (Rosales, 2016; Lehana et al., 2018; Li, 2020), redes neuronales convolucionales (Oshri, 2015), redes neuronales artificiales (Lai, 2015; Thabtah et al., 2021) y redes neuronales Long Short-Term Memory (Drezewski et al., 2021) para analizar partidas de ajedrez. Por otro lado, ha habido diferentes artículos relacionados con la implementación de técnicas de Machine Learning en ajedrez.

En cuanto a ML para la detección de trampas en ajedrez, existen dos estudios previos al respecto. Hoque (2021) desarrolló 4 clasificadores para detectar trampas y obtuvo una exactitud de entre el 85% y el 90%. Sin embargo, este trabajo consideró a los datos anómalos como posibles escenarios de trampas, ya que en el dataset utilizado no se sabía a priori si los usuarios habían realizado trampas o no. Esta limitación, sumada a la falta de variables relacionadas al tiempo por jugada y al tamaño del dataset (solo 200 partidas), fueron inconvenientes para la utilidad de los clasificadores desarrollados en un escenario real. Además, Patria et al. (2021) utilizó redes neuronales convolucionales con el mismo objetivo y obtuvo exactitudes de entre el 53% y el 57%, utilizando un dataset de cuentas vetadas por trampas en Lichess, un sitio web de ajedrez, para caracterizar a las partidas con trampas. Se utilizaron como parámetros los patrones de posiciones en el tablero e indicadores estadísticos. En ambos estudios previos se evidencian limitaciones, tanto en el alcance como en los resultados obtenidos.

Para tratar y solucionar las limitaciones de estudios previos, se proponen para este trabajo cuatro modelos de clasificación de partidas de ajedrez con el objetivo de validar si los jugadores de cada partida analizada han hecho trampas o no, de acuerdo con la definición de trampas proporcionada por FIDE (FIDE, 2021). La base de datos de partidas utilizada es FICS Database (Free Internet Chess Server, 2022), que permite obtener partidas de humanos contra humanos y de humanos contra usuarios de computadora simulado (bots), además de tener el tiempo empleado por jugador en cada jugada.

Debido a que los datasets existentes no etiquetan el uso/no uso de motores de ajedrez en cada partida. Para caracterizar el comportamiento de usuarios haciendo trampas, en esta investigación se utilizó los datos de partidas de computadoras contra humanos como escenarios de trampas. De esta manera, se entrenan a los modelos conociendo a priori el tipo de usuario de cada partida. Además, se añaden variables de análisis y relacionadas al tiempo por jugada de ambos jugadores. Para las variables de tiempo, se obtuvieron los tiempos por jugada al restar el tiempo restante actual al tiempo en el que se registró la jugada anterior. Finalmente, se implementan los clasificadores de Regresión Logística Binomial, Support Vector Machine, Random Forest y K-Nearest Neighbors para entrenar los modelos y se validan los resultados con partidas del dataset original y de partidas propias en el sitio web Chess.com con una serie de métricas de validación propuestas.

Este artículo se estructura de la siguiente manera: En la primera sección, se explica y contextualiza la problemática existente y se mencionan brevemente los objetivos de investigación. En la sección 2, se menciona en el estado del arte los estudios previos relevantes para la investigación, haciendo un énfasis en los modelos de detección de trampas en partidas de ajedrez previos. En la tercera sección, se explican los principales conceptos de ajedrez y ML en los antecedentes. En la cuarta y quinta sección, se presenta la metodología seguida, para luego mostrar los resultados de la experimentación. Finalmente, en las secciones 5 y 6 se discuten los resultados de la investigación y se exponen las conclusiones y recomendaciones para trabajos futuros de modelos de detección de trampas en ajedrez.

2. ESTADO DEL ARTE

Se han realizado diversos estudios pasados que implementan técnicas de Machine Learning en ajedrez con diferentes propósitos, como el análisis predictivo de jugadas, algoritmos de Inteligencia Artificial (IA) y la detección de trampas. Las redes neuronales fueron, junto con a la clasificación y regresión, el área de Machine Learning más utilizada para artículos relacionados al ajedrez en los últimos años. Lai (2015) utilizó redes neuronales artificiales para desarrollar una IA de ajedrez con una fortaleza de 2400 de Elo. Además, Oshri (2015), Thabtah et al. (2020) y Drezewski et al. (2021) realizaron modelos predictivos de resultados de ajedrez, que utilizaron redes neuronales convolucionales, clasificadores de redes neuronales y redes neuronales recurrentes, respectivamente.

2.1 Modelos de regresión y clasificación aplicados al ajedrez

Los modelos de regresión utilizan técnicas estadísticas con el objetivo de predecir los valores de una variable continua (variable dependiente) usando los valores de una o más variables (variables independientes) (Allen, 2004). El procedimiento estándar para el desarrollo de modelos de regresión es obtener datos, asociarlos a un modelo de regresión y evaluarlo usando técnicas estadísticas (Chatterbee, 2013). Golberg et al. (2004) indicaron que los principales usos de la regresión son: investigar la relación entre variables, interpretar el modelo obtenido y generalizar el modelo para otros conjuntos de datos.

En la Tabla 1 se muestran investigaciones que utilizaron modelos de regresión y clasificación para la predicción de resultados y numero de jugadas. De dicha tabla, se tuvo objetivos de carácter predictivo de distinto tipo, como la predicción de resultados. Además, se observa que para la regresión se utilizó Regresión Lineal Múltiple y para la clasificación solo se usó en más de una ocasión el clasificador Random Forest. La exactitud obtenida osciló entre el 60% y 65%, la cual fue calificada de "aceptable" por sus autores.

Tabla 1

Resumen de estudios de regresión y clasificación relacionados al ajedrez.

Autor	Objetivo	Base de datos	Técnica(s) de regresión o clasificación	Resultado
Rosales (2016)	Predecir el resultado de una partida de ajedrez luego de 20 movimientos realizados por cada jugador.	Gorgobase	Random forest, Support Vector Machine	Mayor exactitud con el algoritmo Support Vector Machine (61%).
Lehana et al. (2018)	Predecir resultados de partidas de ajedrez.	Shane's Chess Information Database	Regresión Lineal Múltiple, clasificadores bayesianos.	Clasificadores bayesianos con resultados "marginalmente mejores", con exactitud del 60%.
Li (2020)	Predecir el número mínimo de jugadas para obtener una posición ganadora en finales de ajedrez.	Chessbase	Regresión Lineal Múltiple, árboles de decisión, Random Forest y adaboost.	65% de exactitud

2.2 Machine Learning para la detección de trampas en ajedrez

De los trabajos expuestos anteriormente, se puede apreciar que la mayoría de ellos tiene como objetivo la predicción del resultado de una partida o desarrollar un modelo de Inteligencia Artificial que realice las mejores jugadas. Ha habido un escaso número de estudios previos que tratan sobre Machine Learning aplicado a la detección de trampas en partidas de ajedrez.

Hoque (2021), en su modelo de detección de anomalías en partidas de ajedrez, trabajó con clasificadores, los cuales indicaron la presencia o ausencia de anomalías en las partidas analizadas. Se obtuvo exactitudes de entre el 85% y el 90%, aunque la correcta clasificación de anomalías no indica necesariamente la ocurrencia de trampas. Las principales variables utilizadas fueron: posición, resultado, Elo de las piezas negras, Elo de las piezas blancas, Número de jugadas, Imprecisiones de las blancas/negras, errores graves de las blancas/negras y Average Centipawn Loss (relacionado a la exactitud de las jugadas de acuerdo con un módulo de AI). Cabe resaltar

que las variables de número de errores e imprecisiones, además de la variable de ACL, solo están presentes en este trabajo.

Patria et al. (2021) también desarrollaron un modelo de detección de trampas en partidas de ajedrez en línea, utilizando redes neuronales convolucionales y redes neuronales densas. Se utilizaron como parámetros los patrones de posiciones en el tablero e indicadores estadísticos para cada partida. Las partidas analizadas contaron con la evaluación numérica de Stockfish, un motor de ajedrez. Se obtuvieron exactitudes de entre el 53% y el 57%, destacando el modelo de redes neuronales convolucionales.

Tanto los modelos de Hoque (2021) como los de Patria et al. (2021) presentaron limitaciones. En el caso de Hoque (2021), a pesar de las exactitudes relativamente altas alcanzadas, tuvo un alcance limitado ya que se planteó a las anomalías en las partidas como posibles escenarios de trampas. Sin embargo, la presencia de anomalías no implica necesariamente que se hizo trampas, como lo menciona el autor. Además, el dataset utilizado fue limitado (solo 2012 partidas analizadas). Por otro lado, Patria et al. (2021) tuvo limitaciones en cuanto a la exactitud de los modelos, ya que esta solo alcanzó el 57% en el mejor modelo.

Para otras posibles variables significativas en modelos futuros de detección de trampas, Hoque (2021) sugirió el uso de variables relacionadas al tiempo por jugada. Esta variable también fue sugerida por Barnes (2014), en un artículo que explica las limitaciones de los métodos de detección de trampas de ese entonces. Patrias et al. (2021) utilizó la variable de tiempo restante en cada posición analizada.

2.3 Conjuntos de datos utilizados para el aprendizaje supervisado en ajedrez

Para todos los estudios expuestos en la sección 2.1 se necesitó de bases de datos de partidas de ajedrez para el entrenamiento de los modelos desarrollados. La elección de bases de datos fue variada, siendo solo una usada por más de un autor. Las bases de datos de ajedrez utilizadas, disponibles en línea con una gran cantidad de partidas, fueron: Shane’s Chess Information Database (SCID), Free Internet Chess Server (FICS), Chessbase y Gorgobase (SCID, 2020; FICS, 2020; Chessbase, 2022; Gorgonian’s Chess Site, 2015). En la Tabla 2 se consolidaron las variables utilizadas en los estudios revisados previamente.

Tabla 2

Variables numéricas utilizadas en cada base de datos de partidas de ajedrez.

	SCID	FICS	Chessbase	Gorgobase
Tiempo por jugada		x		
Elo de blancas	x	x	x	x
Elo de negras	x	x	x	x
Resultado	x	x	x	x
Ply (total de jugadas)		x	x	
Nro. de jugadas	x		x	x
Jugadas	x	x	x	x
Evaluación			x	
Variación (de apertura)			x	
Rating	x		x	x
Elo promedio	x		x	x
Duración		x	x	

Para este trabajo se utilizó la base de datos de FICS, la cual cuenta actualmente con más de 300 millones de partidas de ajedrez (FICS, 2020). Cabe mencionar que esta cuenta con tiempos de jugadas de cada partida y además permite obtener partidas entre usuarios de humanos e IA. A partir de los datos obtenidos en las distintas bases de datos expuestas en Tabla 2, se identificó una serie de variables significativas para realizar los distintos trabajos explicados anteriormente.

2.4 Principales variables utilizadas en trabajos previos

A partir de los datos obtenidos en las distintas bases de datos expuestas en la sección 2.3, se identificó una serie de variables significativas para realizar los distintos trabajos de regresión y clasificación explicados anteriormente. De acuerdo con el objetivo de trabajo, se identificaron las variables significativas utilizadas por investigaciones pasadas.

2.4.1 Variables de modelos de regresión y clasificación

Para los modelos de clasificación y regresión de estudios previos, cada autor trabajó con determinadas variables principales, mostradas en la Tabla 3.

Tabla 3

Principales variables en modelos de clasificación y regresión.

Autor	Variable(s) significativa(s)
Rosales (2016)	Jugadas, evaluación, Elo
Lehana (2018)	Jugadas, evaluación, Elo
Li (2020)	Evaluación

Tanto Rosales (2016) como Lehana et al. (2018) realizaron modelos con el objetivo de predecir el resultado de partidas de ajedrez. En ambos casos, las variables significativas fueron las jugadas, la evaluación (hecha por un motor de ajedrez) y el Elo de ambos jugadores. Li (2020) también utilizó a la evaluación como principal variable para predecir el mínimo número de jugadas para obtener una posición ganadora en finales de ajedrez.

2.4.2 Variables de modelos de detección de trampas

Hoque (2021), en su modelo de detección de anomalías en partidas de ajedrez, contó con las siguientes variables: posición, resultado, Elo de las piezas negras, Elo de las piezas blancas, Número de jugadas, Imprecisiones de las blancas/negras, errores graves de las blancas/negras y ACL (Average Centipawn Loss, relacionado a la exactitud de las jugadas de acuerdo con un módulo de AI).

Patria et al. (2020) utilizaron las variables de análisis (hechas por Stockfish 14), arreglos que representan posiciones en el tablero, el total de jugadas y el tiempo restante de cada jugador. Se obtuvo una exactitud ligeramente superior en el modelo hecho con la variable de análisis (57.50% versus 53.75%).

3. ANTECEDENTES

En esta sección, se explicarán los principales conceptos teóricos empleados en este trabajo, los cuales se dividen en dos: conceptos de ajedrez y conceptos de ML.

3.1 Conceptos de ajedrez

El ajedrez se juega en un tablero cuadrado de dos dimensiones con 64 casillas (8 casillas horizontales y 8 verticales), las cuales alternan entre colores claros y oscuros, como se expone en la Figura 1. Además, cuenta con cinco diferentes piezas: el rey, la dama, el alfil, el caballo y el peón. Cada una tiene una capacidad de movimiento en el tablero diferente y se le asigna un valor relativo (FIDE, 2017). Se juega entre dos jugadores y puede haber un ganador o “tablas” (empate).

Figura 1

El tablero de ajedrez en formato digital.

Lichess. (2020). *Analysis board*. [El tablero de ajedrez en formato digital. Se muestra la posición inicial de una partida de ajedrez.]. https://lichess.org/analysis#0

3.1.1 Notación de partidas

Como lo explica FIDE (2008), la organización reguladora del ajedrez, la notación en ajedrez es algebraica. Las filas tienen a letras como valores (de la “a” a la “h”), mientras que las columnas tienen valores numéricos (del 1 al 8). Combinando los valores de las filas y columnas, se tienen coordenadas para las 64 casillas del tablero. En la Figura 2, se muestran visualmente las coordenadas de cada casilla del tablero digital.

Figura 2

Coordenadas para cada casilla del tablero de ajedrez, siguiendo la notación algebraica.

Las piezas también poseen una notación especial. La dama se denota con D, el rey con R, el caballo con C, la torre con T y el alfil con A. El peón no lleva una letra que lo denote, sino que se identifica con la ausencia de una letra. Cada jugada se denota escribiendo en primer lugar la letra correspondiente a la pieza en mayúscula, seguido de la coordenada a la cual se ha movido. Por ejemplo, si la Dama se mueve a la casilla e4, la notación correcta sería “De4”. Además, existen excepciones en la notación para jugadas especiales, como capturas, jaques, jaque mate y promoción.

3.1.2 El formato PGN para almacenar información de partidas

Las partidas de ajedrez son anotadas de acuerdo con una nomenclatura explicada en la sección 3.1.1. Sin embargo, en las bases de datos se utilizan otros formatos de notación para captar la información de una partida. El formato PGN (Portable Game Notation) es el más común para la notación y es parecido al formato algebraico, con la adición de información del sitio, la fecha, la ronda, datos de los jugadores y el resultado de la partida, entre otros datos opcionales (International Chess Club, 2012). Este formato fue utilizado por todas las bases de datos mencionadas anteriormente. El formato SCID, utilizado en SCID y Gorgobase, se basa en PGN.

3.1.3 El sistema de puntuación Elo

La Federación Internacional de Ajedrez (FIDE, por sus siglas en francés) utiliza el sistema Elo para medir los niveles de habilidad de los jugadores competitivos con relación a sus rivales (FIDE, 2008). Dicho sistema fue desarrollado por Arpad Elo e implementado por primera vez por la Federación Estadounidense de Ajedrez (USCF, por sus siglas en inglés) en 1960 (Elo, 1978). Dependiendo del resultado de la partida, los jugadores pueden ganar o perder puntuación Elo. Mientras mayor sea la diferencia entre puntuaciones de los oponentes, mayor va a ser la probabilidad de que el jugador con mejor puntuación gane. Por ejemplo, si dos jugadores tienen la misma puntuación Elo, ambos tienen 50% de probabilidades de ganar o empatar. Sin embargo, si un jugador cuenta 100 puntos más de Elo que su contrincante, tiene el 64% de probabilidades de ganar o empatar la partida (Elo, 1978).

Los dos sitios de ajedrez en línea con mayor popularidad, Chess.com y Lichess.org, utilizan otro sistema de puntuación derivado del sistema Elo, llamado el sistema Glicko (Lichess, n.d.). Dicho sistema, desarrollado por Mark E. Glickman, propone "una mayor fiabilidad de las puntuaciones de los jugadores" ya que introduce el concepto de niveles de desviación de puntuación debido a la frecuencia de juego de los jugadores (Glickman, 1995).

3.1.4 Trampas en ajedrez en línea

FIDE publicó a inicios de 2021 el reglamento a seguir por los jugadores en las competiciones supervisadas por dicha entidad (FIDE, 2021). El reglamento se divide en tres partes: reglas básicas de juego, reglas de ajedrez online y regulaciones para competencias en línea. En esta última parte, se detalla el comportamiento que deben seguir los jugadores en las partidas.

Para esta investigación, se va a hacer énfasis en las conductas que deben seguir los jugadores en una partida, ya que se desea verificar que estos no hagan trampas durante la misma. En el artículo 9 de la última parte del documento, FIDE define el uso de ayuda externa en partidas de ajedrez en línea como el uso de otros dispositivos electrónicos, fuentes de información o consejo o análisis de la partida en otro tablero, ya sea digital o físico (FIDE, 2021).

Los dos principales sitios de ajedrez en línea, Lichess y Chess.com, también cuentan con normas de juego limpio (Lichess, 2021; Chess.com, n.d.). En ambos casos, se especifica que los jugadores no deben recibir asistencia durante las partidas, ya sea de un programa (como un motor de ajedrez), libros, bases de datos, personas, etc. En caso de no cumplir con los lineamientos de conducta, se eliminan las cuentas de dichas páginas por incumplimiento del reglamento. Para esta investigación, se considerará que un jugador ha hecho trampa si ha hecho jugadas que no son propias de su razonamiento, específicamente jugadas sacadas de un motor de ajedrez (por ejemplo, Stockfish).

3.1.5 Inteligencia artificial en ajedrez

La inteligencia artificial en ajedrez ha sido un área de investigación con un desarrollo amplio, empezando a mediados del siglo XX (Shannon, 1950). Desde entonces, con el aumento de la capacidad de procesamiento de las computadoras, los motores de fueron progresando gradualmente. En 1997, Deep Blue, un motor desarrollado por IBM, logró derrotar al campeón mundial de ajedrez, Garry Kasparov, en un enfrentamiento de 6 partidas (Campbell et al., 2002). Hoy en día, los programas de ajedrez tienen una puntuación Elo muy superior a los humanos. El actual campeón mundial, Magnus Carlsen, cuenta con un Elo de aproximadamente 2850; Stockfish, un Elo de 3500 (CCRL, 2022).

Los motores de ajedrez tienen una función de evaluación numérica, la cual expresa que jugador tiene más posibilidades de ganar. Si esta evaluación es positiva, entonces las piezas blancas tienen la ventaja; si es negativa, las negras la tienen. Cada punto de ventaja significa un peón de ventaja. Por ejemplo, si un motor de ajedrez indica una evaluación de +2.0, esto quiere decir que las blancas tienen una posición favorable equivalente a dos peones extra. De la evaluación numérica, se derivan una serie de símbolos que explican el estado de cada jugada analizada (C&O Family Chess Center, 2018). Estos símbolos, descritos en la Tabla 4, se añaden al final de la notación algebraica de cada jugada.

Tabla 4

Notación de análisis de motores de ajedrez.

Símbolo	Descripción
?!	Imprecisión
?	Error
??	Error grave
=	Posición igualada
+=	Las blancas tienen leve ventaja.
=+	Las negras tienen leve ventaja.
+-	Las blancas tienen una ventaja ganadora.
-+	Las negras tienen una ventaja ganadora.

Otra métrica derivada de la evaluación numérica es el Average Centipawn Loss (ACL), que indica el nivel de rendimiento de un jugador en una partida, medido en centésimas de peón (Barnes, 2014). Se calcula hallando el promedio de diferencia de evaluación entre las jugadas hechas por el jugador y las "mejores jugadas" calculadas por un motor de ajedrez. Mientras menor sea el ACL, mejor ha sido el rendimiento del jugador de acuerdo con las mejores jugadas halladas por un motor de ajedrez determinado.

3.2 Técnicas de clasificación de ML

Las técnicas de clasificación pertenecen al aprendizaje supervisado, que es una rama de ML que se caracteriza por el uso de datos clasificados previamente para encontrar patrones y construir modelos matemáticos (Nasteski, 2017). Los modelos supervisados cuentan con el mapeo de una serie de variables de entrada y una sola variable de salida, la cual ya ha sido clasificada. Luego, se aplica este mapeo para predecir variables de salida de datos no clasificados (Cunningham et al., 2008.)

Los modelos de clasificación, que pertenecen al aprendizaje supervisado, tienen como objetivo principal "estimar la relación entre variables que tienen una ligación de razón y resultado" (Uyanik et al. 2013). Una relación de "razón y resultado" quiere decir que la variación de una variable predictora afecta el valor de una variable predictora; es decir, una relación de causa-efecto. A continuación, se explicarán los modelos de clasificación a utilizar en esta investigación.

La Regresión Logística (RL) busca cuantificar una relación entre la variable independiente y las variables dependientes utilizando una variable dependiente de valor discreto (Lavalley, 2008). La RL asigna probabilidades de ocurrencia a eventos de tipo binario haciendo uso de la función sigmoide. El logaritmo de la posibilidad de ocurrencia (logit) de un evento es el resultado de una combinación lineal de variables independientes de tipo numérico (Peng et al., 2002).

Los modelos de Support Vector Machine (SVM) utilizan funciones separadoras en tareas de reconocimiento de patrones (clasificación) (Ukil, 2007). SVM puede ser utilizado tanto para problemas de regresión, en modelos Support Vector Regression, como para problemas de clasificación, en modelos Support Vector Classification (SVC) (Syriopoulos et al., 2020). El clasificador de SVM cuenta con una función "kernel", la cual se encarga de transformar la entrada de datos en un formato determinado (Amari, 1999). Los tipos de funciones kernel son: lineal, radial, polinomial y sigmoide. Otro parámetro importante es el valor "C" de regularización, el cual penaliza numéricamente a los errores de clasificación.

Random forest es un clasificador que consiste en una colección de árboles de decisión. Cada árbol depende de los valores de un vector elegido aleatoriamente (Breiman, 2001). Cada árbol de decisión es creado mediane la selección aleatoria de datos disponibles, que también pueden ser datos muestrales (Ali et al., 2012).

En clasificación, K-Nearest Neighbors (KNN) utiliza a los "k" valores más cercanos de los datos de entrenamiento para asignar una clase a un nuevo valor (Fix & Hodges, 1989). El número de vecinos a utilizar puede variar dependiendo del caso de uso. Además, se puede especificar el peso de los vecinos como uniforme o variable, dependiendo de su cercanía al valor a evaluar (Hall et al., 2008)

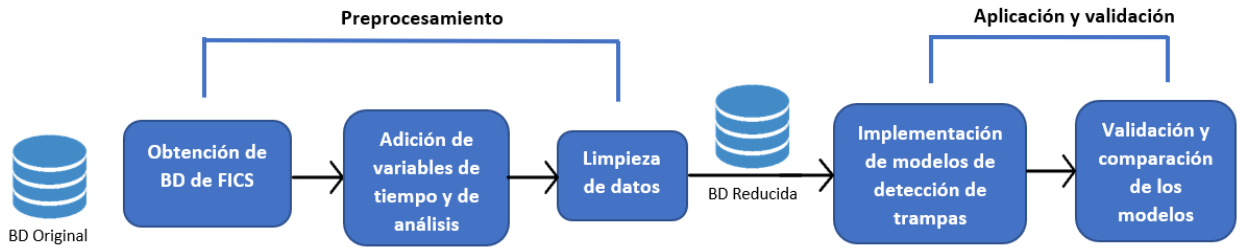
4. METODOLOGÍA

Para la metodología de este trabajo, se siguió una serie de pasos para obtener y transformar los datos de entrenamiento del modelo y ajustar las variables para la aplicación de los modelos de clasificación propuestos, como se muestra en la Figura 3. Finalmente, se implementaron y validaron dichos modelos con dos datasets de prueba, de acuerdo con las técnicas de validación propuestas.

Se plantea utilizar partidas de humanos contra computadoras en el entrenamiento de los modelos para caracterizar a las partidas en las que ocurrieron trampas. Se utilizó la definición de trampas hecha por FIDE (2021), mencionada en la sección 3.1.4. Se consideró que durante las partidas con trampas se utilizó un motor de ajedrez en el entrenamiento. Para la evaluación de la exactitud de los modelos de detección de trampas, se utilizaron dos datasets: el primero contuvo el 30% de las partidas del dataset de FICS; el segundo, partidas propias (una misma cuenta/usuario) en el sitio de ajedrez Chess.com, de las cuales en la mitad de ellas se hizo trampa, ya que se utilizaron las jugadas propuestas por el motor de ajedrez Stockfish 15 en cada turno de estas. Se utilizaron partidas propias para tener certeza a priori si es una partida con trampa o no.

Figura 3

Esquema de la metodología para la implementación del modelo de detección de trampas.



4.1 Preprocesamiento

La base de datos utilizada fue obtenida de Free Internet Chess Server (FICS), la cual contiene partidas de humanos contra IA y de humanos contra humanos. Además, esta permite el cálculo de variables relacionadas al tiempo utilizado por jugador (Free Internet Chess Server, n.d.). También se utilizaron los siguientes filtros para obtener las partidas deseadas:

i. En primer lugar, el tipo de partidas de acuerdo con el tiempo por jugador fue del formato Estándar (tiempo por jugador mayor a 10 minutos).

ii. Además, los usuarios de IA debieron tener una puntuación Elo mayor a 2800, tomando como referencia el Elo de los mejores jugadores del mundo (2700 Chess, 2022). El Elo promedio de los 10 mejores jugadores del mundo actuales es menor a 2800.

iii. Los humanos con un Elo superior a 1800, al igual que los trabajos de Hoque (2021) y Patria et al (2021).

iv. Por último, se tomaron las partidas más recientes, con fechas desde el 2015 al 2022

De la base de datos de FICS (Free Internet Chess Server, 2022), se obtuvieron partidas de tres tipos: humanos contra humanos (HvH), computadoras contra humanos (CvH) y humanos contra computadoras (HvC). Tanto las partidas HvC como las CvH fueron consideradas como escenarios de trampas. En total, se obtuvieron 3918 partidas antes de realizar la limpieza de datos. Se contó con 20 variables, de las cuales solo se utilizaron 11 variables numéricas, descritas en la Tabla 5.

Tabla 5

Variables numéricas de la base de datos de FICS.

Variable	Descripción
B – Elo	Puntuación Elo de las blancas
N – Elo	Puntuación Elo de las negras
Resultado	Resultado de la partida (0-1, 1-0, 1/2-1/2)
Ply	Número de jugadas total
Jugadas	Jugadas en notación algebraica (según la sección 3.1.1). Además, cuenta el tiempo de cada jugada.
Promedio de Elo	El promedio de Elo de ambos jugadores
Negras-Es-computadora	Indica si el jugador de las piezas negras es computadora (Booleano).
Blancas-Es-Computadora	Indica si el jugador de las piezas blancas es computadora.
Tiempo	Tiempo toal de la partida (segundos)
Tiempo-Blancas	Tiempo total con el que disponen las blancas (segundos)
Tiempo-Negras	Tiempo total con el que disponen las negras (segundos)

A continuación, se muestra un ejemplo de la notación de la variable “Jugadas” de una partida de 10 jugadas del dataset de FICS:

1.c4 {[%emt 0.0]} 1...Nf6 {[%emt 0.0]} 2.d4 {[%emt 0.1]} 2...e6 {[%emt 0.976]} 3.Nf3 {[%emt 0.1]} 3...d5 {[%emt 0.876]} 4.Nc3 {[%emt 0.1]} 4...c6 {[%emt 1.067]} 5.e3 {[%emt 0.1]} 5...h5 {[%emt 3.395]Black resigns} 1-0

De la información brindada en el ejemplo mostrado, se pueden obtener variables derivadas del tiempo empleado por ambos jugadores durante la partida. Además, ya que las jugadas siguen una notación algebraica, estas pueden ser analizadas por un motor de ajedrez. En la siguiente subsección, se exponen las variables adicionadas en el preprocesamiento, que derivan de la información brindada en la variable “Jugadas”.

Las partidas obtenidas de FICS se encontraron en formato PGN. Para poder hacer el tratamiento de variables y adaptarlas a modelos de clasificación de ML, se transformaron los datos a formato CSV (valores separados por comas, por sus siglas en inglés. Luego, se importó dicho CSV para utilizarlo como base de datos en Python.

4.1.1 Adición de variables

Hoque (2021) y Barnes et al. (2014) sugirieron la utilización de variables relacionadas al tiempo empleado por jugador para la detección de trampas en ajedrez, aunque ninguno de ellos trabajó con este tipo de variables. Por este motivo, se extrajo de la base de datos de FICS (en formato PGN convertido a txt) los tiempos de cada jugador para obtener las variables de tiempo para las piezas blancas y las negras, expuestas en la Tabla 6.

Tabla 6

Variables de tiempo propuestas para el modelo de detección de trampas.

Variable de tiempo	Descripción
Tiempo promedio por jugada	El tiempo promedio por jugada de cada jugador, obtenido con el tiempo total empleado y el número de jugadas (ply).
Desviación estándar de tiempos de jugada	La variabilidad del tiempo empleado por jugada se cuantifica al calcular la desviación estándar de los tiempos registrados por jugador. Una desviación estándar pequeña implica tiempos similares para las jugadas independientemente de su complejidad, lo cual sería un indicador de trampa.
Máximo tiempo empleado	El máximo tiempo empleado en una jugada por cada jugador.

En este trabajo, se propone la adición de variables relacionadas a la evaluación numérica de un motor de ajedrez, al igual que Rosales (2016), Hoque (2021) y Patria et al (2021). Para obtener anotaciones de un motor de ajedrez que analice las partidas del dataset, se utilizó el programa SCID, un gestor de bases de datos de ajedrez de código abierto (SCID, n.d.). En dicho programa, se importó el dataset. Luego, se usó la herramienta de análisis de motor de ajedrez de SCID para anotar las partidas con datos de la evaluación de dicho motor; en este caso, será Stockfish 15. Una vez analizadas las partidas, se importaron las variables derivadas de la evaluación mediante la lectura del archivo PGN con las anotaciones de Stockfish 15, en Python. Estas variables se presentan en la Tabla 7.

Tabla 7

Variables de análisis propuestas para el modelo de detección de trampas.

Variable de evaluación	Descripción
Número de imprecisiones	Número de imprecisiones de las blancas y las negras
Número de errores	Número de errores de las blancas y las negras
Número de errores graves	Número de errores graves de las blancas y las negras
Average Centipawn Loss (ACL)	Medida de rendimiento de cada jugador, medida en centésimas de peón.
Diferencia entre ACL real y ACL esperado	Se compara el ACL real con el esperado, luego de crear una función que calcule el mismo en función al Elo de los jugadores.

Para calcular la diferencia entre el ACL real y el ACL esperado, se siguieron los siguientes pasos:

v. Primero, se creó un dataset a partir del dataset original con únicamente partidas de humanos con las piezas blancas y negras.

vi. Luego, se seleccionaron las variables de Elo y ACL para ver la relación entre ambas y a partir de esta se creó una función que calcule el ACL esperado. Para el ACL el caso de los usuarios de IA, se generaron puntuaciones Elo aleatorias, siguiendo una distribución normal con una media y desviación estándar obtenidas de los datos de Elo de los humanos.

vii. Finalmente, se añadió la variable "Diferencia entre ACL real y ACL esperado" para todas las partidas del dataset, para las blancas y las negras. Se espera que la diferencia entre el ACL esperado y el ACL real para los usuarios humanos sea menor.

La última variable añadida fue de carácter binario, que indica si ambos jugadores son humanos o no, la cual sirvió para crear la variable objetivo.

4.1.2 Limpieza de datos

Se extrajeron del dataset de SCID las variables de Resultado de blancas, Resultado de negras y total de jugadas ("ply") utilizando la librería "pgn2data". Para la limpieza de datos, se realizaron los siguientes filtros:

i. Para el caso de partidas entre humanos y computadoras, se eliminaron las partidas que acabaron en tablas (empate), ya que en el modelo se considera que los usuarios de IA, al hacer las mejores jugadas disponibles, usualmente ganan las partidas. En caso de tablas, se asume que ningún jugador hizo trampas. Cabe destacar que ningún humano le ganó a un usuario de IA (bot) en las partidas obtenidas de FICS.

ii. Se eliminaron las partidas en las que el ACL de los usuarios de IA tuvieron un ACL mayor a 50.

iii. Se quitaron las partidas en las que la diferencia de Elo es mayor a 500, caso que ocurrió solo 3 veces.

iv. Por último, se eliminaron las partidas con un ACL no válido, debido a un número insuficiente de jugadas para calcularlo

4.2 Modelos de detección de trampas

Luego de realizar la limpieza de datos, se utilizó el dataset de FICS para entrenar y validar cuatro modelos de detección de trampas, de los cuales la variable dependiente fue el tipo de partida con trampas o sin trampas. Para el desarrollo del modelo, se usaron módulos de la librería Scikit Learn en Python (Pedregosa et al., 2011).

Los clasificadores implementados (RLB, SVM, Random Forest y KNN) tienen una serie de hiperparámetros que pueden ser modificados. Para encontrar sus valores óptimos en cada clasificador, se utilizó la técnica de *grid search* (búsqueda en rejilla), la cual prueba la exactitud para cada combinación de hiperparámetros especificada. Los hiperparámetros utilizados en la *grid search* para cada modelo se muestran en la Tabla 8.

Tabla 8

Hiperparámetros utilizados en el *grid search* de cada uno de los clasificadores implementados.

Clasificador	Hiperparámetros a modificar en la grid search
Regresión Logística Binomial	C (0.5, 1, 2), solucionador (“newton-cg”, “lbfgs”, “sag”, “saga”)
Support Vector Machine	C (0.5, 1, 2), kernel (lineal, radial, polinomial, sigmoide), función de decisión (uno versus uno, uno versus el resto)
Random Forest	Número de estimadores (50, 100, 200, 400), criterio (gini, entropía), máximo de atributos (raíz cuadrada, logaritmo de base 2, sin máximo de atributos)
K-Nearest Neighbors	Número de vecinos (3, 5, 7, 10), pesos (uniformes, distancia), algoritmo (“ball tree”, “kd tree”, “brute”), p (1, 2)

Para verificar si existe *overfitting* o *underfitting* en los modelos, se calculó la función de curva de aprendizaje (Hess et al., 2009). El *underfitting* ocurre cuando el modelo obtiene una exactitud de entrenamiento baja con relación a la exactitud de prueba; el *overfitting*, cuando la exactitud del entrenamiento es significativamente superior a la de prueba (Jabbar et al., 2015). De esta manera, se supo si tanto el tamaño de la base de datos como sus variables son adecuadas para el desarrollo de cada uno de los modelos de clasificación.

Para la validación de los modelos de clasificación resultantes, se consideraron los siguientes criterios de evaluación, propuestos por Novakovic et al. (2017):

i. La exactitud de los modelos, que se obtiene dividiendo el número de clasificaciones correctas entre el total de casos contemplados de los datos de prueba. Se implementó la técnica de validación cruzada, que consiste en la división del dataset en subconjuntos de prueba y de entrenamiento de manera iterativa, con el objetivo de reducir la varianza de la exactitud estimada (Zhong et al., 2020).

ii. La matriz de confusión, que compara las clasificaciones reales de los datos de prueba con los predichos por el modelo (Novakovic et al., 2017). De esta matriz, se obtiene el número de casos de verdaderos positivos (VP), verdaderos negativos (VN), falsos negativos (FN) y falsos positivos (FP) del modelo de RLB. Para este trabajo, es de especial relevancia la métrica de falsos positivos, ya que estos supondrían clasificar el comportamiento de un jugador humano como el de una computadora; es decir, estimar que hizo trampas cuando este no fue el caso. Las métricas derivadas de la matriz de confusión utilizada son:

   - La exactitud, calculada dividiendo los casos clasificados correctamente entre el total de casos contemplados.
   - La sensibilidad, que indica la proporción entre los VP y la suma de VP y FN.

iii. La curva ROC (Curva característica de funcionamiento del receptor, por siglas en inglés) para las dos clases: Trampas y Sin trampas. En esta curva, el eje X representa la tasa de falsos positivos; el eje Y, el de verdaderos positivos. Un clasificador óptimo minimiza el eje de falsos positivos y maximiza el eje de verdaderos positivos. Gráficamente, una curva adecuada maximiza el AUC (Área bajo la curva, por sus siglas en inglés) (Marzban, 2004).

Para validar la exactitud del modelo desarrollado se utilizaron dos datasets: el primero conteniendo el 30% de las partidas del dataset de FICS; luego para validar en un escenario real, se jugó 50

partidas en el sitio de ajedrez en línea de Chess.com como segundo dataset. En 25 de ellas, se hizo trampas utilizando el motor de ajedrez Stockfish 15 como ayuda externa para realizar las jugadas. En las otras 25, se jugó sin hacer trampa. Luego, se hizo uso de estas partidas como datos de entrada de los modelos ya desarrollados y se midió su desempeño, utilizando las métricas de exactitud y sensibilidad.

5. RESULTADOS

5.1 Preprocesamiento

Se utilizó la librería de Python "pgn2data", disponible en el Índice de paquetes de Python (Qureshi, 2021), para transformar el formato a CSV, el cual fue importado en un dataframe en Python. Sin embargo, esta transformación no incluyó la variable de tiempo por jugada. Por este motivo, se tuvo que crear una función en Python que extraiga dichos valores de la variable "Jugadas" del PGN original para obtener el tiempo promedio por jugada de ambos jugadores y su desviación estándar.

Algunas variables no fueron contempladas por la librería, por lo que se obtuvieron manualmente mediante la lectura del PGN de la base de datos en Python, ya que este está en formato de texto. Estas fueron: Blancas-Es-Computadora, Negras-Es-Computadora y Jugadas (que incluyen los tiempos por jugada).

5.1.1 Adición de variables de tiempo

Las variables de tiempo, mencionadas en la metodología, fueron obtenidas manualmente en Python extrayendo los tiempos, presentes en la variable Jugadas, del archivo PGN de la base de datos. Luego, se calculó el tiempo promedio por jugada, la desviación estándar de tiempos de jugada y la jugada de mayor tiempo. Cada uno de estos valores se dividió entre la variable Tiempo para que todos los valores sean proporcionales al tiempo por partida.

De la Tabla 9, se observó que los humanos se demoraron más tiempo por jugada y tienen la jugada con mayor tiempo empleado. Estos también tuvieron una mayor desviación estándar de tiempo por jugada, por lo que existe una mayor variabilidad en el tiempo empleado para cada jugada.

Tabla 9

Porcentaje de tiempo empleado usuarios de humanos e IA.

Tipo de jugador	Tiempo promedio por jugada (%)	Desv. Estándar de tiempo por jugada (%)	Máximo tiempo empleado (%)
Humanos	1.64	1.98	8.89
IA	0.61	0.89	4.39

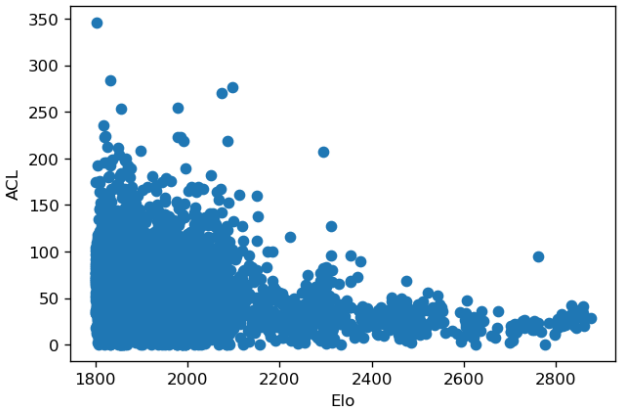
5.1.2 Adición de variables de análisis

La implementación de las variables de evaluación descritas en la tabla 5 requirió de un tiempo de procesamiento de aproximadamente 10 días, utilizando un solo núcleo de un procesador de 4.7 GHz de frecuencia y memoria RAM a 3600 MHz de velocidad. Para cada partida, se utilizó un tiempo de evaluación de 2 segundos por jugada. La extracción de dichas variables se dio de acuerdo con la notación de análisis de motores de ajedrez, explicada en la sección 3.4.1.

Las variables de número de imprecisiones, número de errores, número de errores graves y ACL se obtuvieron directamente del archivo PGN generado de la herramienta de análisis de partidas de SCID Database, con el motor de ajedrez Stockfish 15. Para la variable de "Diferencia entre ACL esperado y ACL real", se encontró una relación inversa entre el ACL real y el Elo de los humanos, como se muestra en la Figura 4.

Figura 4

Gráfico de dispersión del ACL de los humanos con respecto a su Elo.



Se observó que existe una relación inversamente proporcional entre el Elo y el ACL de los humanos. Quiere decir que, a mayor Elo de un jugador, más precisas son sus jugadas de acuerdo con Stockfish 15, por lo que su ACL esperado es menor. Luego de probar con polinomios de distinto grado, la función resultante de esta relación se ajusta mejor manera a la siguiente ecuación polinomial de grado 2:

$$y = 0.000188x^2 - 0.140452x + 263.169259 \quad (1)$$

donde y es el ACL esperado y x es el Elo del jugador. Por ejemplo, un jugador de 2000 de Elo tiene un ACL esperado de 57.46.

Al dividir el dataset en datos de humanos y datos de usuarios de IA, se obtuvo que los humanos tienen medias de imprecisiones, errores, errores graves y ACL mayores, como se muestra en la Tabla 10.

Tabla 10

Porcentaje de jugadas con imprecisiones, errores y errores graves y ACL de humanos e IA.

Tipo de jugador	Imprecisiones (%)	Errores (%)	Errores graves (%)	ACL
Humanos	7.83	2.99	2.74	59.54
IA	1.97	1.02	0.04	13.65

De las 29 variables iniciales del dataset, se descartaron aquellas con valores no numéricos. Las variables restantes del modelo, sumadas a las variables de tiempo y de análisis, fueron las siguientes: Ply, Resultado-Blancas, Resultado-Negras, Blancas-Es-Computadora y Negras-Es-Computadora. Las variables de B-Elo y N-Elo fueron descartadas luego de ser utilizadas para calcular el ACL esperado de cada jugador. Después de la limpieza de datos, en base a los criterios explicados en la metodología, se eliminaron 616 partidas, resultando en 3171 partidas en la base de datos.

5.2 Implementación y validación de los modelos de detección de trampas

5.2.1 Implementación de los modelos de detección de trampas

Luego de implementar la técnica de *grid search* en cada modelo, se identificaron los hiperparámetros óptimos. Utilizando los hiperparámetros expuestos en la Tabla 11, se crearon los cuatro modelos de detección de trampas. Para la aplicación de los modelos, se dividió el dataset en 2 partes: los datos de entrenamiento (70%) y los de validación (30%). Se contó con 2219 datos de prueba y 952 datos de validación. De las 3171 partidas, 1039 fueron HvC, 1056 fueron CvH y 1076 fueron HvH. Adicionalmente se utilizó un segundo dataset de validación con 50 partidas propias en Chess.com, de las cuales en 25 se hizo trampa.

Tabla 11

Hiperparámetros óptimos de cada modelo hallados con grid search

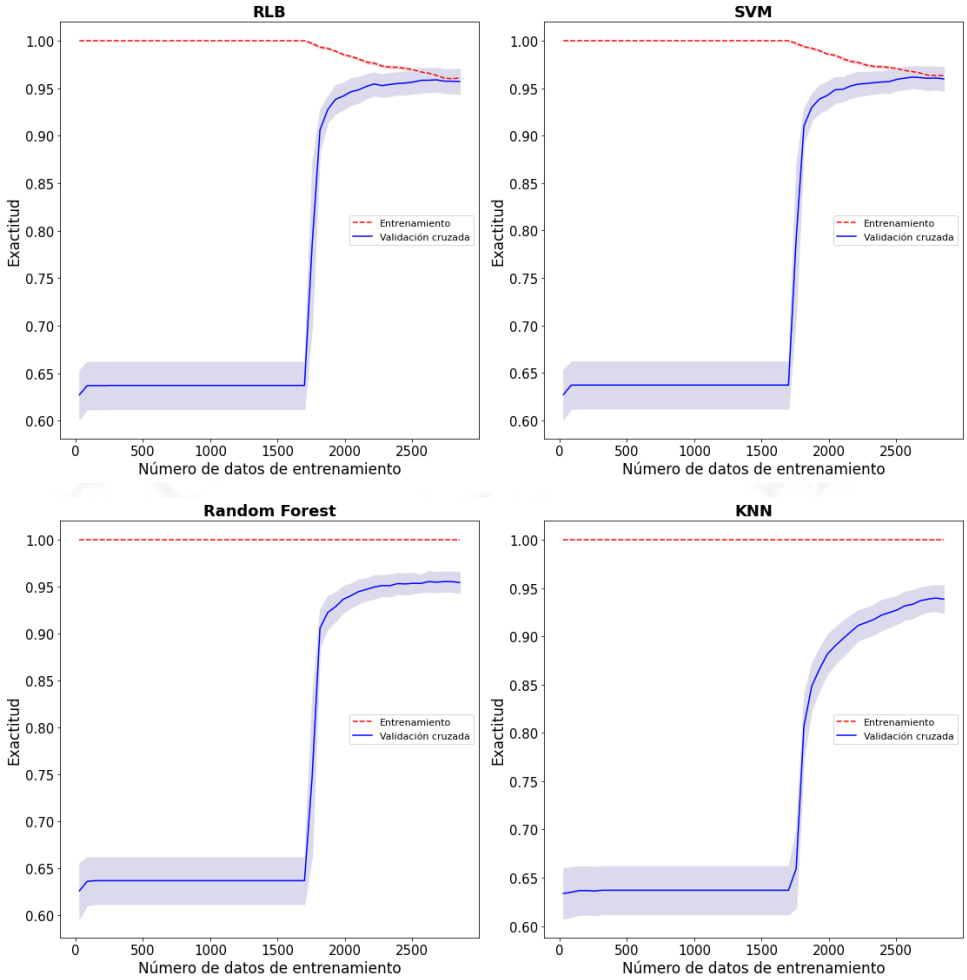
Clasificador	Hiperparámetros óptimos
RLB	C = 1, solucionador = “newton-cg”
SVM	C = 0.5, kernel = lineal, función de decisión = uno versus uno
Random Forest	Número de estimadores = 200, criterio = entropía, máximo de atributos = raíz cuadrada
KNN	Número de vecinos = 7, pesos = distancia, algoritmo = *ball tree*, p = 1

5.2.2 Validacion de los modelos de detección de trampas

Para comprobar que no existió overfitting ni underfitting en los modelos, se graficaron las curvas de aprendizaje, expuestas en la Figura 5. En dicha figura, se observa que las exactitudes de entrenamiento y las de validación cruzada fueron muy similares con el número de datos de entrenamiento actual. Esto quiere decir que no ocurrió ni *overfitting* ni *underfitting* en ninguno de los modelos y que el número de datos del dataset no supone una limitación en la exactitud de los modelos.

Figura 5

Curvas de aprendizaje de los modelos de detección de trampas



5.2.2 Validación de los modelos de detección de trampas

Para el primer dataset de validación (30% de dataset FICS, 952 datos), las exactitudes obtenidas de cada modelo se muestran en la Tabla 12. Los modelos de RLB, SVM y Random Forest obtuvieron exactitudes similares, mientras que KNN tuvo una exactitud ligeramente menor.

Tabla 12

Exactitud de cada modelo de detección de trampas

	RLB	SVM	Random Forest	KNN
Exactitud (%)	96.429	96.324	96.534	94.223

Las matrices de confusión de la Figura 6 confirman los resultados de exactitud expuestos en la Tabla 12. Además, se puede observar que, para todos los modelos, la mayoría de los errores de clasificación ocurren cuando hay casos de falsos positivos de trampas; es decir, cuando se clasifica como “trampas” a una partida en la cual no las hubo. Esto se hace evidente con el valor de sensibilidad de la clase “Sin trampas” en todos los modelos, presente en la Tabla 13.

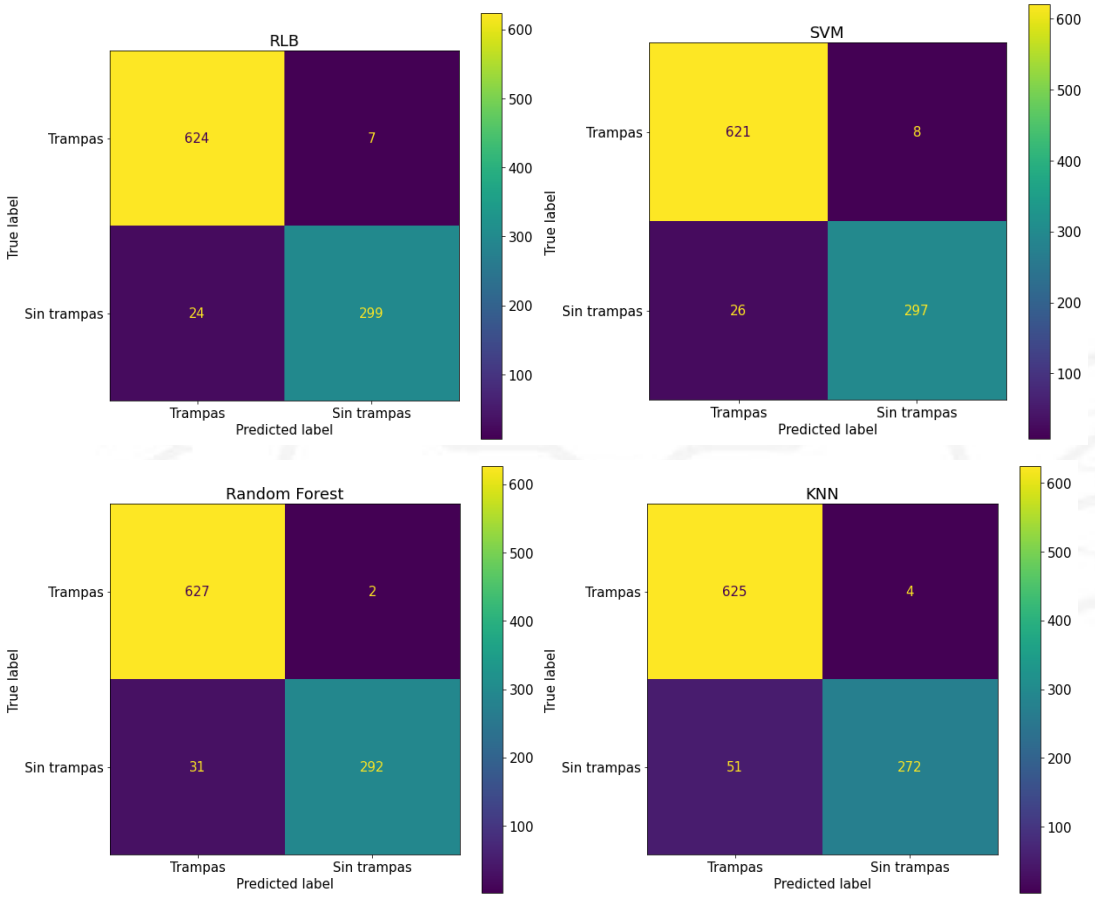
Tabla 13

Sensibilidad (%) de cada clase de los modelos de detección de trampas.

	RLB	SVM	Random Forest	KNN
Trampas	98.559	98.722	99.521	99.359
Sin trampas	92.260	92.260	90.093	83.591

Figura 6

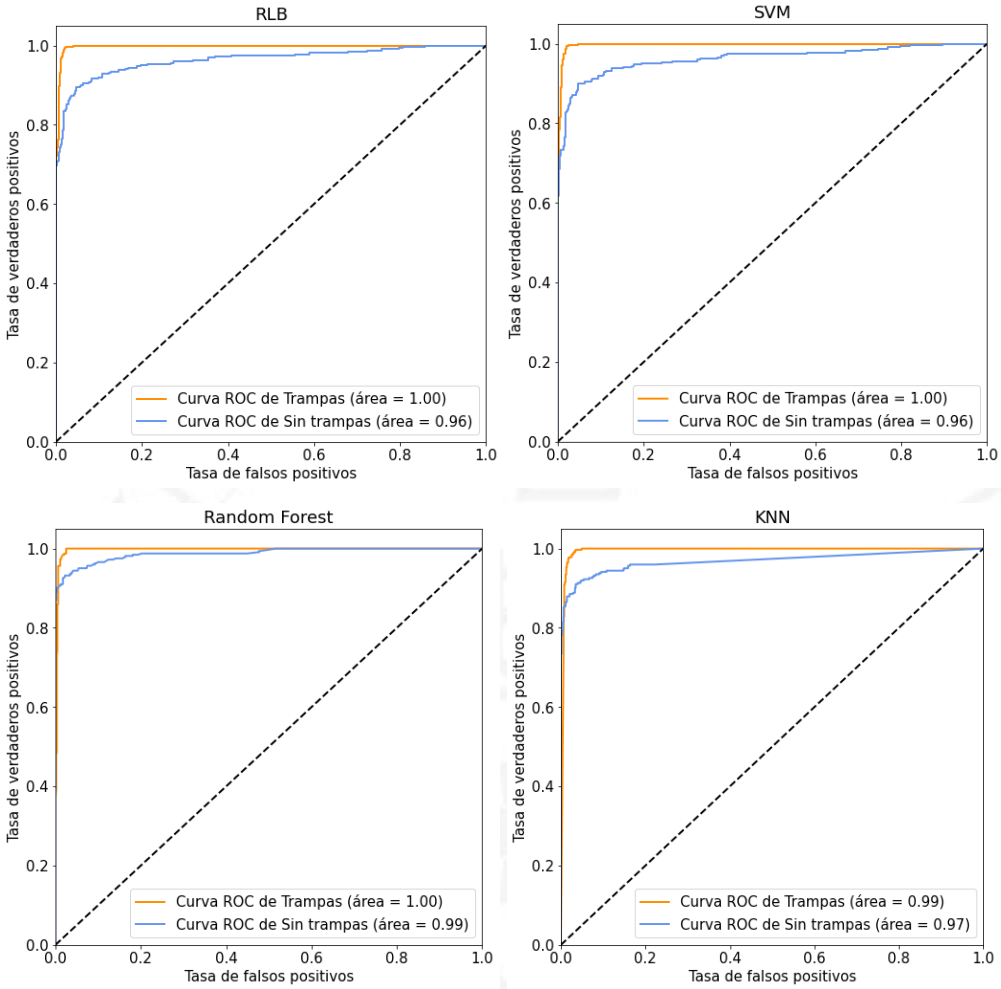
Matrices de confusión de los modelos de detección de trampas.



Los resultados de las curvas ROC se observan en la Figura 7. La curva ROC de la clase “Con trampas” tuvo un área bajo la curva de 1.00, lo que indica que los casos de falsos negativos en este tipo de partidas son ínfimos. La clase Trampas tuvo una menor AUC, de entre 0.96 y 0.97, lo que indica que esta tuvo un mayor número de casos de falsos negativos. Esto concuerda con los resultados de las matrices de confusión, que mostraron un mayor número de errores de clasificación en casos de falsos negativos en partidas con trampas.

Figura 7

Curvas ROC para cada clase de cada modelo.



Para el segundo dataset de validación luego de jugar y procesar 50 partidas en Chess.com, haciendo trampas en 25 de ellas al consultar con Stockfish 15 la mejor jugada para cada posición durante las partidas. Se evaluó este dataset en cada uno de los modelos desarrollados previamente, entrenados con el dataset original. Las métricas de exactitud y sensibilidad de los modelos se muestran en las Tablas 14 y 15, respectivamente.

Tabla 14

Exactitud (%) de cada modelo de detección de trampas con partidas propias en Chess.com.

	RLB	SVM	Random Forest	KNN
Exactitud (%)	94	94	96	88

Tabla 15

Sensibilidad (%) de cada clase de los modelos de detección de trampas con partidas propias en Chess.com.

	RLB	SVM	Random Forest	KNN
Con trampas	100	100	100	100
Sin trampas	88.461	88.461	92.307	76.923

La Tabla 14 muestra exactitudes similares a las obtenidas de la validación del dataset original. Los modelos de RLB, SVM y Random Forest obtuvieron exactitudes del 94%, 94% y 96%, respectivamente. Nuevamente, la exactitud del modelo de KNN fue inferior a la del resto de modelos. Además, de la Tabla 14 se evidencia que la sensibilidad de la clase “Sin trampas” tiene un valor menor en cada uno de los modelos. Esto mismo sucedió con la validación del modelo original.

6. DISCUSIÓN

En la experimentación, se utilizó un dataset de partidas de FICS (Free Internet Chess Server, n.d.), el cual tuvo partidas de HvH (sin trampas), HvC y CvH (trampas). Luego, se desarrollaron cuatro modelos de detección de trampas con dicho dataset, utilizando técnicas de clasificación. Finalmente, se validaron los resultados con partidas del dataset original y con partidas propias en Chess.com utilizando exactitud, validación cruzada, matriz de confusión y métricas derivadas de ella, función de pérdida y curvas ROC.

En el preprocesamiento, se tuvo que eliminar el 15% de las partidas por contar con un número insuficiente de jugadas para el análisis o porque el resultado fue de tablas, el cual no se ha considerado en el modelo. A pesar de que se redujo el número de partidas a analizar, se comprobó con las curvas de aprendizaje obtenidas (Figura 5) que no existieron problemas de overfitting ni de underfitting en los modelos. Esto indicó que el número de partidas utilizado no ha supuesto una limitación en las exactitudes obtenidas.

En el gestor de bases de datos SCID (SCID, n.d.), para realizar anotaciones con el motor de ajedrez de Stockfish 15, se utilizó el tiempo de análisis por defecto de 2 segundos por jugada. Esto impactó en el tiempo de análisis de las 3302 partidas, que conllevó 10 días de tiempo de procesamiento. Si se hubiera utilizado un mayor tiempo de análisis por jugada, se pudo haber obtenido evaluaciones más precisas, lo que quizás pudo tener como consecuencia un aumento de exactitud en los modelos desarrollados.

El dataset utilizado contuvo 1273 partidas HvH, 1016 partidas CvH y 1013 partidas HvC. Cabe destacar que, para las partidas CvH y HvC (con trampas), se seleccionaron aquellas en las que la IA contaba con un Elo mayor a 2800, tomando como referencia el Elo de los mejores jugadores del mundo (2700 Chess, 2022). Esto limitó el número de partidas obtenidas, ya que un porcentaje muy bajo de usuarios de computadora contó con este Elo. Para el segundo dataset de validación, con partidas en Chess.com, se consideró a 50 partidas: 25 con trampas y 25 sin trampas. Cada partida fue realizada a los más por dos jugadores humanos, con un tiempo por jugador de 30 minutos, lo cual tuvo como consecuencia un número limitado de partidas para la validación. Adicionalmente, las partidas fueron di señadas para ser clasificadas en partidas con trampas y sin trampas, en un entorno controlado. Esta etapa de validación no fue contemplada por investigaciones previas (Hoque, 2021; Patria et al., 2021), que solo utilizaron un dataset de prueba, obtenido del dataset original.

Para el desarrollo del modelo, se definió a las trampas como el uso de un motor de ajedrez como ayuda externa en partidas de ajedrez en línea. Esta definición coincide con la proporcionada por FIDE (2021); por lo tanto, el modelo propuesto puede utilizarse en escenarios reales para la detección de trampas. En cambio, Hoque (2021) consideró a las anomalías estadísticas presentes en algunas partidas como posibles escenarios de trampas. Este enfoque no tiene mucha utilidad práctica, ya que el hecho de que una partida sea anómala no significa que haya habido trampas por parte de algún jugador. Por otro lado, Patria et al. (2021) consideraron como escenarios de trampas a un dataset con partidas de las cuentas suspendidas por incumplimiento de términos de servicio en el sitio web Lichess.org. Sin embargo, no existe la certeza de que se hayan realizado trampas

en dichas partidas, ya que el juicio de si se hizo trampas o no depende del sistema de detección de trampas de Lichess.

La adición de variables de tiempo y de análisis fue significativa para la exactitud del modelo, como se muestra en la Tabla 16. Al eliminar estas variables, la exactitud del modelo bajó significativamente, lo que implica la pérdida de un 30% de exactitud en la detección de trampas. Por otro lado, la exactitud del modelo fue mayor cuando las variables fueron separadas para las blancas y las negras. También se intentó reducir el número de variables independientes (de 19 a 11) al utilizar una misma variable para las blancas y las negras. Por ejemplo, en lugar de utilizar el ACL de las Blancas y el ACL de negras, se planteó utilizar la diferencia de ACL. Los resultados de esta eliminación de variables no fueron adecuados, ya que la exactitud fue del 86.7%; es decir, un 8.6% menor a la original. Esto puede deberse a que, al separar las variables, los modelos cuentan con más información de cada partida. Autores pasados también separaron las variables para las blancas y las negras para sus modelos de clasificación (Hoque, 2021; Patria et al., 2021; Lehana, 2018; Li, 2020; Rosales, 2016).

Tabla 16

Exactitudes (%) de modelos alternos para cada clasificador, utilizando el dataset original para la validación.

Tipo de modelo	RLB	SVM	Random Forest	KNN
Original	96.429	96.524	96.534	94.223
Reducido	84.979	85.399	89.811	87.080
Sin variables de análisis ni de tiempo	65.126	66.071	66.807	66.702
Modelo con certeza mayor al 90%	88.445	88.866	85.084	83.929

De las métricas de sensibilidad para la validación con el dataset original y el de partidas propias en Chess.com (Tablas 13 y 15), se observó que la mayoría de los errores de clasificación de los modelos desarrollados fueron casos de falsos positivos de trampas. Para intentar reducir este número de falsos positivos, se realizaron modelos alternativos, en los cuales solo se consideró la existencia de trampas en aquellas partidas en las que se tenga una certeza mayor al 90%. Las exactitudes de estos modelos fueron ligeramente inferiores a las de los modelos originales (Tabla 16) pero los casos de falsos positivos se redujeron. En la Tabla 17, se observa que la sensibilidad para la clase “Sin trampas” de los modelos alternativos aumentó con respecto a los modelos originales (Tabla 15), debido al menor número de casos de falsos positivos por trampas, pero aumentó el número de falsos negativos. La existencia de este balance entre la maximización de la exactitud y minimización de falsos positivos deja al criterio propio si se debe o no implementar un intervalo de confianza, y si este debe ser del 90%.

Tabla 17

Sensibilidad (%) de cada clase de los modelos con un nivel de confianza del 90% para los casos de trampas.

	RLB	SVM	Random Forest	KNN
CvH	86.084	85.090	81.129	81.230
HvC	82.500	83.125	75.313	74.375
HvH	96.594	96.285	98.452	95.975

En las Tablas 18 y 19, se exponen las exactitudes de los modelos de detección de trampas de Patria et al. (2021) y Hoque (2021). Se puede observar que, en comparación con investigaciones previas, los modelos propuestos tienen una mejor exactitud (Tabla 12) que los de estudio previos. Cabe destacar que en los modelos de Hoque (2021) las exactitudes fueron basadas en la detección de anomalías, no de trampas. Incluso el modelo con menor exactitud (KNN) obtuvo una mejor exactitud que cualquiera de los revisados en otros trabajos. Además, en este trabajo también se ha validado el modelo con datos ajenos al dataset original al probar su exactitud con datos de 50 partidas en Chess.com. Los resultados obtenidos de esta prueba adicional fueron similares a los obtenidos con el dataset original de FICS. El uso de un dataset que permite obtener variables relacionadas al tiempo utilizado por cada jugador en las partidas, sumado a las variables relacionadas al ACL propuestas tuvieron un impacto significativo en las exactitudes de los modelos desarrollados.

Tabla 18

Exactitudes de los modelos de redes neuronales de Patria et al. (2021)

	CNN	Red neuronal densa	CNN (analizada)	Red neuronal densa (analizada)
Exactitud	53.75	48.75	57.50	56.25

Tabla 19

Exactitudes de los modelos de clasificación de Hoque (2021)

	LDA	QDA	RLB	KNN
Exactitud	90.62	87.50	87.50	85.94

7. CONCLUSIONES

En la investigación propuesta, se propuso realizar modelos de clasificación para la detección de trampas en partidas de ajedrez en línea, logrando detectar trampas con exactitudes del 96.534% y del 96%, utilizando partidas del propio dataset y de partidas propias en Chess.com. Ambas exactitudes fueron obtenidas con Random Forest, el clasificador implementado que obtuvo los mejores resultados. El impacto de la adición de variables de análisis (imprecisiones, errores, errores graves y ACL) y de tiempo en su exactitud fue significativa, ya que esta aumentó en aproximadamente un 30% con su inclusión. Cabe destacar que los modelos desarrollados operan con partidas terminadas, no en curso. En los modelos con certeza mayor al 90% para casos de trampas, se observó un balance entre la reducción de falsos positivos de trampas y la exactitud general de los modelos. Además, el alcance del modelo de detección de trampas fue mayor a los modelos pasados, ya que se obtuvo una exactitud mayor, se consideró una definición de trampas que coincide con la de FIDE (2021), se utilizó un número significativamente mayor de partidas para el entrenamiento y la validación, y se validaron los modelos con partidas propias en un entorno controlado.

Se consiguió desarrollar modelos que obtuvieron una mejor exactitud que todos los modelos contemplados en el estado del arte, con validaciones tanto con partidas del dataset original como con partidas en Chess.com. Además, se propuso y se comprobó la utilidad de la utilización de partidas de humanos contra computadoras para caracterizar a las partidas con trampas. También se consideraron las variables de tiempo, las cuales fueron sugeridas por autores pasados, pero solo implementadas por Patria et al. (2021). Estos aportes, sumados a un alcance mayor que en estudios

previos, suponen un avance en el campo de detección de trampas en ajedrez. El modelo propuesto puede ser de ayuda para asegurar la integridad competitiva de los torneos de ajedrez en línea, lo cual beneficia tanto a los organizadores como a los propios jugadores, ya que se logró detectar trampas con una exactitud y alcance superiores a los de estudios previos.

8. TRABAJOS FUTUROS

Para trabajos futuros, se recomienda aumentar el número de partidas de validación en entornos controlados. También se podría realizar el análisis de partidas con un mayor tiempo de análisis por jugada o con una capacidad de computación superior, para potencialmente mejorar su exactitud. Finalmente, se sugiere considerar a más de una partida por jugador para determinar la presencia de trampas.

REFERENCIAS

Ali, J., Khan, R., Ahmad, N., & Maqsood, I. (2012). Random Forests and Decision Trees. *International Journal of Computer Science Issues, Vol.9*(5), 272-278. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.3863&rep=rep1&type=pdf

Amari, S., & Wu, S. (1999). Improving support vector machine classifiers by modifying kernel functions. En *Neural Networks* (Vol. 12, Issue 6, pp. 783–789). Elsevier BV. https://doi.org/10.1016/s0893-6080(99)00032-5

Becker, B., Kohawi, R., & Sommerfield, D. (1997, Agosto 17). *Visualizing the Simple Bayesian Classifier* [Acta de conferencia]. Issues in the Integration of Data Mining and Data Visualization, Newport Beach, California, E.E.U.U. http://robotics.stanford.edu/~ronnyk/vizNB.pdf

Breiman, L. (2001). Random Forests. Machine Learning, Vol 45, 5-32. https://link.springer.com/content/pdf/10.1023/A:1010933404324.pdf

Drezewski, R., & Wator, G. (2021). Chess as Sequential Data in a Chess Match Outcome Prediction Using Deep Learning with Various Chessboard Representations. *Procedia Computer Science,* Vol. *192(1),* 1760-1769. https://doi.org/10.1016/j.procs.2021.08.180

Elo, A. (1978). *The Rating of Chess Players: Past and Present* (2da ed.). FIDE. https://www.gwern.net/docs/statistics/comparison/1978-elo-theratingofchessplayerspastandpresent.pdf

FIDE (2017). *FIDE Laws of Chess.* https://www.fide.com/FIDE/handbook/LawsOfChess.pdf

FIDE (2021). *FIDE Online Chess Regulations*. https://rcc.fide.com/wp-content/uploads/2021/01/FideOnlineChessRegulations.pdf

Fix, E., & Hodges, J. L. (1989). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties. En *International Statistical Review / Revue Internationale de Statistique* (Vol. 57, Issue 3, p. 238). JSTOR. https://doi.org/10.2307/1403797

Free Internet Chess Server. (2022). scid [Data set]. https://www.ficsgames.org/download.html

Free Internet Chess Server. (2022). *Rating distribution of FICS standard players (22,405,688 games)*. https://www.ficsgames.org/stats/9999_rdistr_3.html

Glickman, M. E. (1995). *The Glicko System*. http://www.glicko.net/glicko/glicko.pdf

Golberg, M. A., Cho, H. A. (2004). *Introduction to Regression Analysis (1ra ed.)*. WIT Press. https://www.witpress.com/books/978-1-85312-624-6

Gorgonian's Chess Site. (2015). GorgoBase [Data set]. http://gorgonian.weebly.com/pgn.html

Grand Chess Tour (2021). 2021 *Grand Chess Tour*. https://grandchesstour.org/2021-grand-chess-tour/overiew

Hall, P., Park, B. U., & Samworth, R. J. (2008). Choice of neighbor order in nearest-neighbor classification. En *The Annals of Statistics (Vol. 36, Issue 5)*. Institute of Mathematical Statistics. https://doi.org/10.1214/07-aos537

Hess, K. R., & Wei, C. (2010). Learning Curves in Classification with Microarray Data. En *Seminars in Oncology* (Vol. 37, Issue 1, pp. 65-68). Elsevier BV. https://doi.org/10.1053/j.seminoncol.2009.12.002

Hoque, M. (2021). *Classification of chess games: An exploration of classifiers for anomaly detection in chess* [Master's thesis, Minnesota State University, Mankato]. Cornerstone: A Collection of Scholarly and Creative Works for Minnesota State University, Mankato. https://cornerstone.lib.mnsu.edu/etds/1119

International Chess Club. (2012). *Portable Game Notation Specification and Implementation Guide*. ICC. https://www.chessclub.com/help/PGN-spec

Jabbar, H., & Khan, R. Z. (2015). Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). Computer Science, Communication and Instrumentation Devices.

Karthika, S., & Sairam, N. (2015). A Naïve Bayesian Classifier for Educational Qualification. En *Indian Journal of Science and Technology* (Vol. 8, Issue 16). Indian Society for Education and Environment. https://doi.org/10.17485/ijst/2015/v8i16/62055

LaValley, M. P. (2008). Logistic Regression. En *Circulation* (Vol. 117, Issue 18, pp. 2395–2399). Ovid Technologies (Wolters Kluwer Health). https://doi.org/10.1161/circulationaha.106.682658

Lai, M. (2015). *Giraffe: Using Deep Reinforcement Learning to Play Chess*. Londres: Imperial College London.

Lehana, P., Kulshrestha, P., Thakur, N., & Asthana, P. (2018). Statistical Analysis on Result Prediction in Chess. En *International Journal of Information Engineering and Electronic Business (IJIEEB),* Vol.10(4), pp. 25-32. https://doi.org/10.5815/ijieeb.2018.04.04

Li, H. (2020). The Application of Machine Learning in Chess Endgames Prediction. En *Proceedings of the 2020 2nd International Conference on Big Data and Artificial Intelligence. ISBDAI '20: 2020 2nd International Conference on Big Data and Artificial Intelligence*. ACM. https://doi.org/10.1145/3436286.3436289

Lin, Y. (2004). A note on margin-based loss functions in classification. En *Statistics & Probability Letters* (Vol. 68, Issue 1, pp. 73–82). Elsevier BV. https://doi.org/10.1016/j.spl.2004.03.002

Lichess. (n.d.) *Chess rating systems*. Lichess.org. https://lichess.org/page/rating-systems

Lichess. (n.d.) *Analysis board*. Lichess.org. https://lichess.org/analysis

Lichess. (2021, Diciembre 28) *Terms of Service*. Lichess.org. https://lichess.org/terms-of-service

Marzban, C. (2004). The ROC Curve and the Area under It as Performance Measures. En *Weather and Forecasting* (Vol. 19, Issue 6, pp. 1106-1114). American Meteorological Society. https://doi.org/10.1175/825.1

Nasteski, V. (2017). An overview of the supervised machine learning methods. En *HORIZONS.B* (Vol. 4, pp. 51–62). University St. Kliment Ohridski - Bitola. https://doi.org/10.20544/horizons.b.04.1.17.p05

Novaković, J. D., Veljović, A., Ilić, S. S., Papić, Željko, & Milica, T. (2017). Evaluation of Classification Models in Machine Learning. En *Theory and Applications of Mathematics & Computer Science*, 7(1), Pages: 39 -. Obtenido de https://www.uav.ro/applications/se/journal/index.php/TAMCS/article/view/158

Osbourne, J. W. (2014). *Best Practices in Logistic Regression*. SAGE Publications. https://books.google.com.pe/books?id=5UgXBAAAQBAJ&dq=logistic+regression&lr=&hl=es&source=gbs_navlinks_s

Oshri, B., & Khandwala, N. (2015). *Predicting Moves in Chess using Convolutional Neural Networks*. Stanford: Universidad de Stanford.

Patria, R., Favian, S., Caturdewa, A., & Suhartono, D. (2021). *Cheat Detection on Online Chess Games using Convolutional and Dense Neural Network*. En 2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI). 2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI). IEEE. https://doi.org/10.1109/isriti54043.2021.9702792

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research* (Volume 12, pp. 2825-2830). JMLR.

Peng, C.-Y. J., Lee, K. L., & Ingersoll, G. M. (2002). An Introduction to Logistic Regression Analysis and Reporting. En *The Journal of Educational Research* (Vol. 96, Issue 1, pp. 3–14). Informa UK Limited. https://doi.org/10.1080/00220670209598786

Qureshi, Z. (2021). *Pgn2data 0.0.4*. Python Index Package. https://pypi.org/project/pgn2data/#description

Rosales, H. A. (2016). *Predicting the Outcome of a Chess Game by Statistical and Machine Learning techniques*. Universidad Politécnica de Barcelona.

SCID. (n.d.) *Wiki*. SCID. https://sourceforge.net/p/scid/wiki/StartHere/

SCID. (2020). scid (Version 4.7) [Data set]. https://sourceforge.net/projects/scid/files/Scid/Scid%20Latest/

Shannon, C. E. (1950). XXII. Programming a computer for playing chess. En *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* (Vol. 41, Issue 314, pp. 256–275). Informa UK Limited. https://doi.org/10.1080/14786445008521796

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, D., Lillicrap, T., Simonyan, K., …Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. En *Science*, Vol 362(6419), 1140-1144. https://doi.org/10.1126/science.aar6404

Smirnov, I. (2016). *Chess Rules: The Ultimate Guide for Beginners. Remote Chess Academy*. http://chess-teacher.com/wp-content/uploads/2016/05/Chess-rules-ultimate-guide.pdf

Syriopoulos, T., Tsatsaronis, M., & Karamanos, I. (2020). Support Vector Machine Algorithms: An Application to Ship Price Forecasting. En *Computational Economics* (Vol. 57, Issue 1, pp. 55–87). Springer Science and Business Media LLC. https://doi.org/10.1007/s10614-020-10032-2

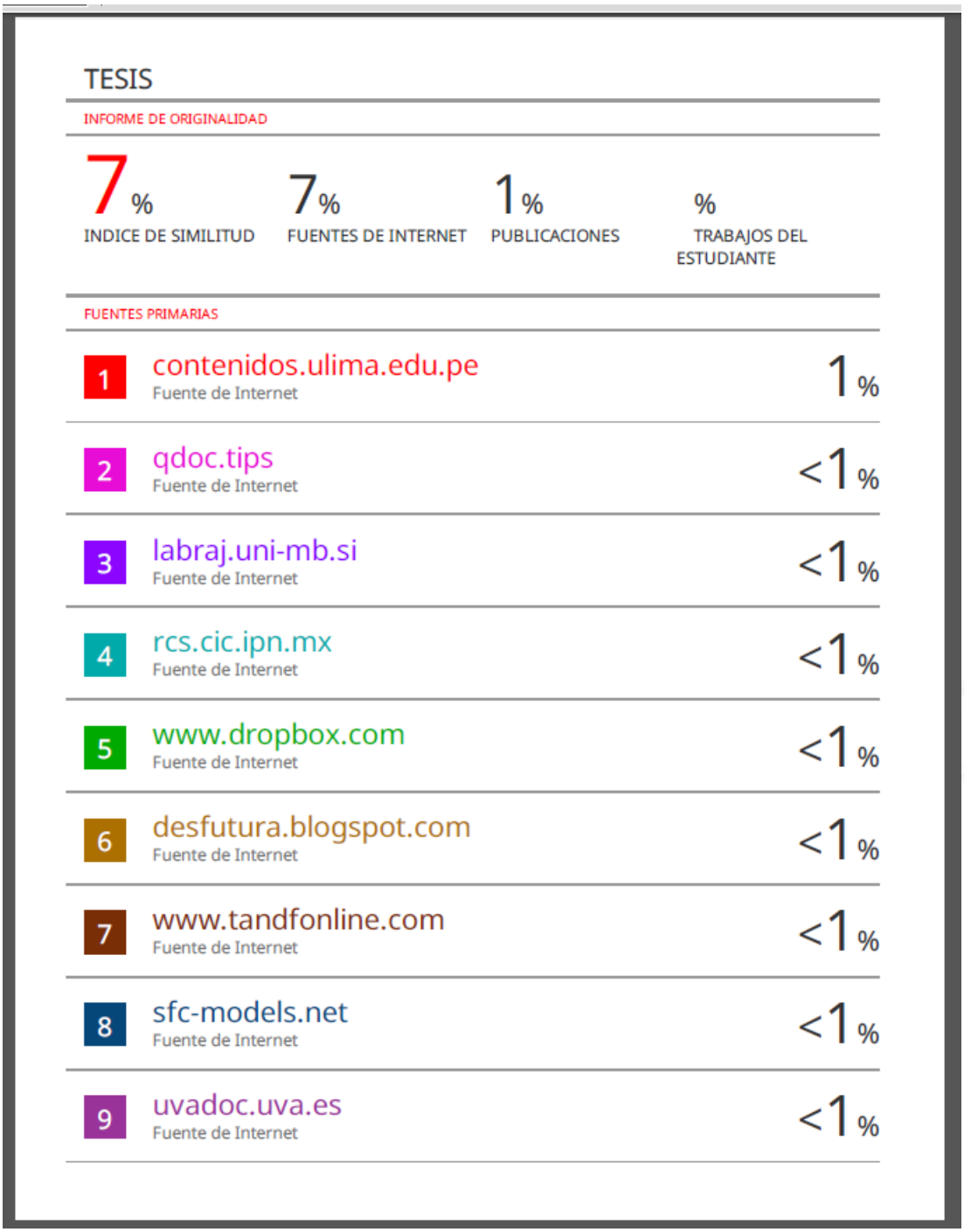
Ukil, A. (2007). *Support Vector Machine. Tomado de Power Systems* (pp. 161–226). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-73170-2_4

Uyanik, G. K., & Güler, N. (2013). A Study on Multiple Linear Regression Analysis. En *Procedia - Social and Behavioral Sciences* (Vol. 106, pp. 234–240). Elsevier BV. https://doi.org/10.1016/j.sbspro.2013.12.027

Yan, X., & Su, X. (2009). *Linear Regression Analysis*. World Scientific. https://books.google.com.pe/books/about/Linear_Regression_Analysis.html?id=AYS1mAEACAAJ&redir_esc=y

Zhong, Y., He, J., & Chalise, P. (2020). Nested and Repeated Cross Validation for Classification Model With High-Dimensional Data. En *Revista Colombiana de Estadística* (Vol. 43, Issue 1, pp. 103–125). Universidad Nacional de Colombia. https://doi.org/10.15446/rce.v43n1.80000

2700 Chess. (2022, 06 de Julio). *Live Chess Ratings.* 2700 Chess. https:// https://2700chess.com/



TESIS

INFORME DE ORIGINALIDAD

7%	7%	1%	%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	contenidos.ulima.edu.pe Fuente de Internet	1%
2	qdoc.tips Fuente de Internet	<1%
3	labraj.uni-mb.si Fuente de Internet	<1%
4	rcs.cic.ipn.mx Fuente de Internet	<1%
5	www.dropbox.com Fuente de Internet	<1%
6	desfutura.blogspot.com Fuente de Internet	<1%
7	www.tandfonline.com Fuente de Internet	<1%
8	sfc-models.net Fuente de Internet	<1%
9	uvadoc.uva.es Fuente de Internet	<1%