Memoria del Proyecto: Aplicación de Machine Learning en Ajedrez

Autor: Jesús González Suárez

20 de noviembre de 2024

Índice

Índice de figuras

1. Introducción

El presente proyecto tiene como objetivo explorar la integración de técnicas de *Machine Learning* (ML) en el análisis del ajedrez. Se desarrollaron modelos capaces de:

- Sugerir movimientos en base a una posición del tablero.
- Predecir resultados de partidas (ganador y duración).
- Analizar patrones de juego mediante técnicas de agrupamiento (*clustering*).

El proyecto se fundamenta en el uso de bibliotecas modernas de Python como `TensorFlow`, `scikit-learn`, y `pandas`. A continuación, se describe el flujo de trabajo, los métodos implementados y los resultados obtenidos.

2. Metodología

El proyecto se divide en las siguientes etapas:

1. Preprocesamiento de datos y tokenización de movimientos.
2. Entrenamiento de modelos supervisados y no supervisados.
3. Desarrollo de una interfaz gráfica para sugerencia de movimientos.
4. Evaluación y visualización de resultados.

2.1. Datos Utilizados

Los datos provienen de un conjunto de partidas en formato PGN, almacenados en el archivo `games.csv`. Las características principales extraídas incluyen:

- Número de turnos (`turns`).
- Ventaja de rating entre jugadores (`rating_advantage_white`).
- Duración de la partida en minutos (`game_duration`).
- Jugador ganador (`winner`).

2.2. Modelos Implementados

Los siguientes modelos fueron utilizados en distintas tareas del proyecto:

- **Sugerencia de movimientos:** Una red neuronal entrenada con `TensorFlow` para predecir el próximo movimiento basado en una secuencia previa.
- **Predicción de duración de la partida:** Modelos de regresión como `RandomForestRegressor`
- **Predicción de ganador:** Clasificadores como `GradientBoostingClassifier` y `RandomForestClassifier`.
- **Clustering de patrones de juego:** Algoritmo `KMeans`.

3. Desarrollo

3.1. Tokenización y Sugerencia de Movimientos

El primer paso fue preparar un tokenizador para representar movimientos en un formato que pueda ser procesado por el modelo. A continuación se muestra el código usado:

```
1 def prepare_and_save_tokenizer(moves, save_path):
2     tokenizer = Tokenizer()
3     tokenizer.fit_on_texts(moves)
4     with open(save_path, 'wb') as f:
5         joblib.dump(tokenizer, f)
6     return tokenizer
```

El modelo utiliza una red neuronal para sugerir el siguiente movimiento, basada en secuencias de longitud máxima de 20 movimientos. La predicción se integra con un motor de ajedrez para verificar la validez del movimiento.

3.2. Predicción del Ganador y Duración de la Partida

Para la predicción de resultados, se extrajeron características como el `rating` de ambos jugadores, la duración y el número de turnos. El pipeline incluye el uso de modelos supervisados. Un ejemplo de la evaluación de `RandomForestRegressor` para duración es:

```
def predict_game_duration(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y,
        test_size=0.3, random_state=42)
    model = RandomForestRegressor(random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"MAE: {mean_absolute_error(y_test, y_pred):.2f}")
    print(f"RMSE: {mean_squared_error(y_test, y_pred, squared=
        False):.2f}")
    return model
```

3.3. Clustering

El análisis no supervisado se realizó con `KMeans`, identificando clusters de partidas según las características de los jugadores (`rating` blanco y negro). Este enfoque ayuda a identificar patrones de juego comunes.

```
def evaluate_clustering(data, n_clusters=3):
    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(data)
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans.fit(scaled_data)
    data['Cluster'] = kmeans.labels_
    return kmeans, data
```

3.4. Interfaz Gráfica

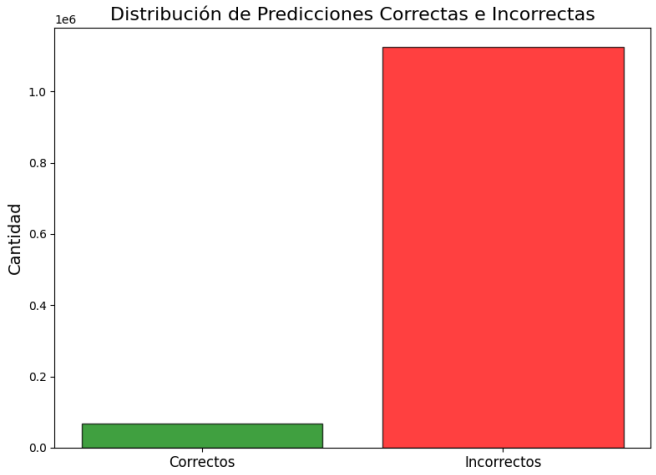
Se desarrolló una interfaz gráfica en `Tkinter` que muestra el tablero de ajedrez y sugiere movimientos. La integración con la biblioteca `chess.svg` permite visualizar el tablero de manera interactiva.

Figura 1: Interfaz Modulo Ajedrez

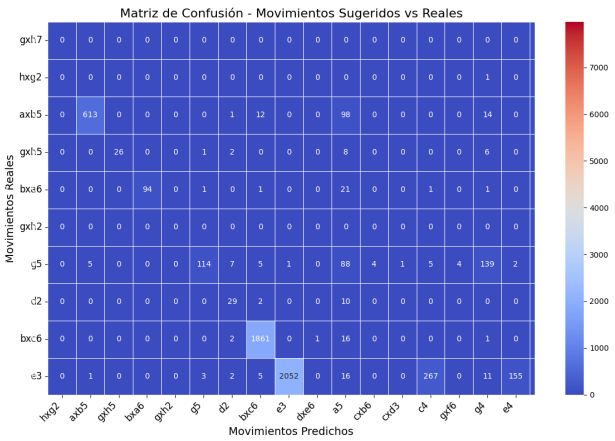
4. Resultados

4.1. Sugerencia de Movimientos

El modelo de red neuronal logró una precisión del 6 % en movimientos sugeridos en el conjunto de validación. La siguiente figura muestra la comparación entre movimientos sugeridos y reales:



(a) Distribución Predicciones Correctas e Incorrectas



(b) Ejemplo Matriz de Confusión

Figura 2: Comparación de Predicciones y Matriz de Confusión.

4.2. Predicción de Ganador

El modelo de `GradientBoostingClassifier` obtuvo una precisión promedio del 84 %. Para entrenar este modelo, se utilizaron las siguientes *features* extraídas del conjunto de datos:

- **Número de turnos:** Representa la cantidad total de movimientos realizados en la partida.
- **Puntuación del jugador blanco (`white_rating`):** Valoración del jugador blanco al momento de jugar la partida.
- **Puntuación del jugador negro (`black_rating`):** Valoración del jugador negro al momento de jugar la partida.
- **Ventaja de puntuación blanca:** Diferencia entre la puntuación del jugador blanco y la del jugador negro (`white_rating - black_rating`).
- **Duración de la partida:** Tiempo total que tomó la partida, medido en minutos.
- **Apertura (`opening_ply`):** Número de movimientos realizados en la fase inicial de la partida.

La matriz de confusión presentada a continuación ilustra la comparación entre las etiquetas reales y las predicciones del modelo:

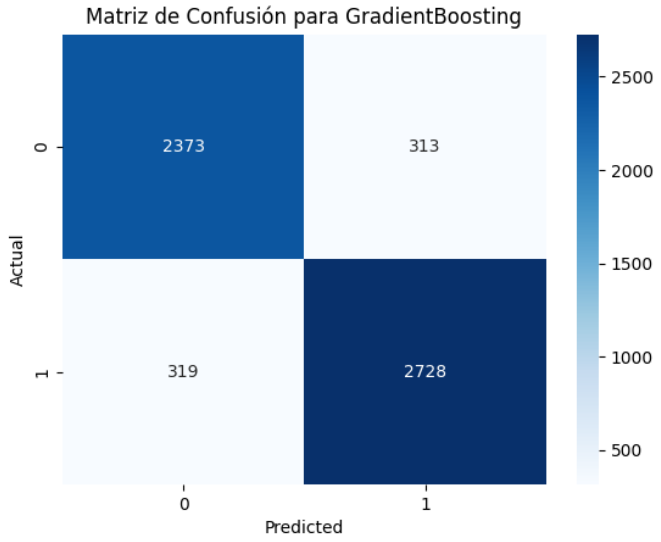


Figura 3: Matriz de Confusión GradientBoostingClassifier

4.3. Predicción de Duración de Partidas

El objetivo de esta tarea fue construir un modelo capaz de predecir la duración de las partidas en minutos. Los datos utilizados incluyen características como:

- **Diferencia de puntuaciones:** Diferencia entre las puntuaciones de los jugadores al inicio de la partida.
- **Número de turnos:** Total de movimientos realizados durante la partida.
- **Duración previa:** Basada en partidas similares en términos de apertura y estilo de juego.

Para este propósito, se utilizó un `RandomForestRegressor` con los siguientes hiperparámetros optimizados:

- **n_estimators:** 200
- **max_depth:** 10
- **min_samples_leaf:** 2

El modelo alcanzó las siguientes métricas:

- **Error absoluto medio (MAE):** 17.76 minutos.
- **Error cuadrático medio (RMSE):** 65.01 minutos.

La figura 4 muestra un gráfico de dispersión que compara las predicciones del modelo con los valores reales. La línea roja representa el ideal donde predicción y realidad coinciden perfectamente.

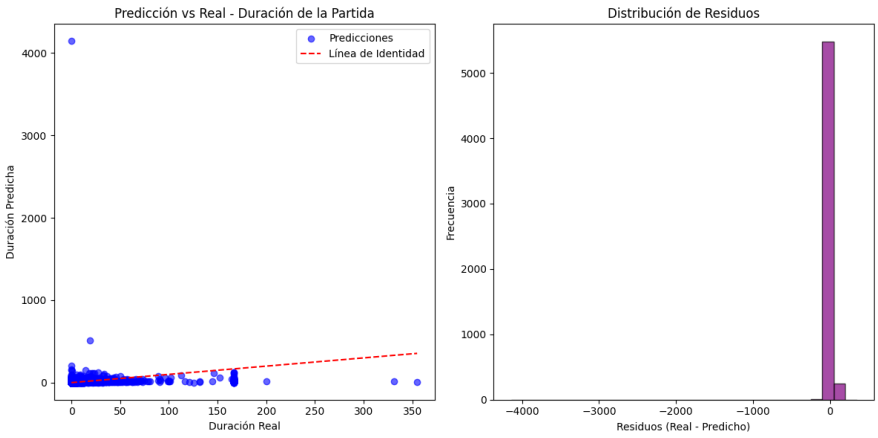


Figura 4: Predicción vs Real - Duración de Partidas

Interpretación de Resultados: El modelo predice con un margen razonable de error en partidas de duración media, pero tiene dificultades con partidas extremadamente largas o cortas, lo que se refleja en el RMSE significativamente mayor al MAE. Esto sugiere que el modelo podría beneficiarse de:

- Tratamiento de outliers en las partidas con duración extrema.
- Inclusión de características adicionales relacionadas con la estrategia de apertura.

4.4. Análisis de Clustering

El análisis de clustering fue realizado para identificar patrones subyacentes en las partidas de ajedrez que no son evidentes a simple vista. Usamos el algoritmo de *K-Means* para agrupar las partidas en función de características clave, como las puntuaciones de los jugadores y la duración de las partidas. El objetivo era descubrir grupos naturales de partidas basados en similitudes en las estadísticas del juego.

Características utilizadas:

- **Puntuación del jugador blanco (`white_rating`):** Representa la habilidad del jugador blanco según el sistema de puntuación del ajedrez.
- **Puntuación del jugador negro (`black_rating`):** Similar al anterior, pero para el jugador negro.
- **Duración de la partida:** Tiempo total que tomó cada partida.
- **Número de turnos:** La cantidad de movimientos realizados en cada partida.

Estas características fueron seleccionadas porque proporcionan una visión integral del nivel de los jugadores y la complejidad de las partidas.

Resultados del clustering: El análisis reveló **tres grupos principales de partidas**:

- **Cluster 1:** Partidas rápidas con jugadores de puntuación baja a media, lo que sugiere juegos casuales o menos competitivos.
- **Cluster 2:** Partidas de duración intermedia con jugadores de puntuación media, representando una mezcla de partidas semicompetitivas.
- **Cluster 3:** Partidas largas con jugadores de puntuación alta, lo que indica juegos de alto nivel, posiblemente en torneos.

Interpretación: La separación en tres clusters destaca cómo el nivel de los jugadores y la duración de las partidas influyen en los patrones de juego. Los jugadores de mayor nivel tienden a tener partidas más largas, ya que analizan más profundamente sus movimientos, mientras que los jugadores casuales suelen jugar partidas más rápidas y menos estratégicas.

El siguiente gráfico ilustra esta clasificación, mostrando cómo los grupos se distribuyen en función de las puntuaciones de los jugadores y otras características clave:

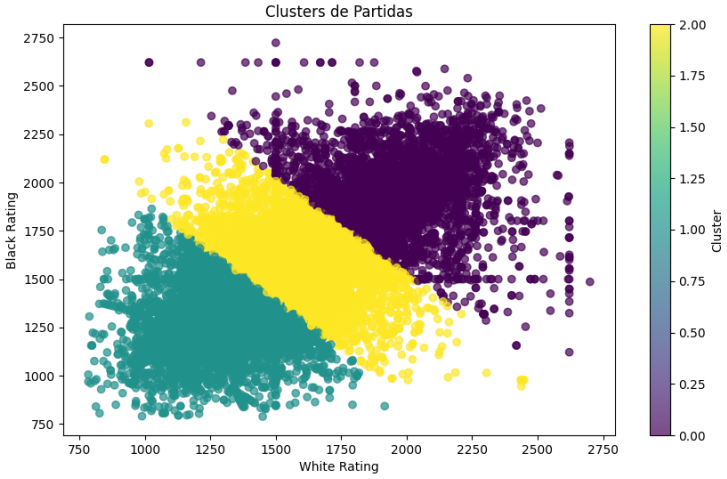


Figura 5: Clasificación de partidas mediante clustering.

5. Conclusión

Este proyecto demuestra el potencial de aplicar *Machine Learning* en el análisis del ajedrez. Las áreas principales de mejora incluyen:

- Entrenar el modelo de sugerencia con conjuntos de datos más amplios.
- Explorar arquitecturas avanzadas como *transformers*.
- Refinar las métricas para análisis de patrones.

Referencias

- TensorFlow Documentation
- Scikit-learn Documentation
- Chess.com Data API