Universidad del Valle de Guatemala
Deep Learning
Sec. 20

Proyecto Final

AI Black Jack

Javier Chávez 21016
Andres Quezada 21085
Mario Cristales 21631

Introducción

En este proyecto, desarrollamos un modelo de inteligencia artificial capaz de jugar al blackjack de manera óptima utilizando aprendizaje automático. El interés radica en explorar cómo las técnicas de Deep Learning pueden aplicarse a juegos de cartas tradicionales para tomar decisiones estratégicas. Empleamos una red neuronal profunda para predecir las mejores acciones basadas en el estado actual del juego, con el objetivo de maximizar las ganancias y minimizar las pérdidas. Los resultados obtenidos demuestran que el modelo es capaz de aprender patrones complejos y mejorar su rendimiento en comparación con estrategias básicas.

Antecedentes

Blackjack es un juego de cartas popular en casinos, donde el objetivo es obtener una mano con un valor lo más cercano posible a 21 sin pasarse. Los jugadores compiten contra el crupier y deben decidir si "piden" más cartas, se "plantan", "doblan" su apuesta o "dividen" sus cartas. La estrategia óptima en blackjack puede ser compleja debido a la probabilidad y la incertidumbre inherentes al juego.

Para este proyecto, utilizamos un conjunto de datos que contiene información detallada de partidas de blackjack, incluyendo las cartas restantes en el mazo, la carta visible del crupier, la mano inicial del jugador, conteos de cartas y las acciones tomadas. La tecnología empleada incluye Python, TensorFlow para construir y entrenar la red neuronal, y herramientas de preprocesamiento de datos como pandas y scikit-learn.

El conjunto de datos fue extraído con una simulación de 100 millones de iteraciones de una mano de blackjack. Siguiendo las reglas más comunes dentro de los casinos más populares del mundo:

- Shoe de 8 barajas
- Blackjack paga 3:2.
- Se permite doblar con cualquier combinación de las primeras 2 cartas.
- Se permite doblar después de dividir.
- Se pueden dividir las primeras 2 cartas iguales hasta un máximo de 4 manos.
- No se permite volver a dividir Ases.
- Dividir Ases solo da 1 carta adicional por As y no se considera Blackjack.
- Se permite la rendición tardía.
- No se permite rendirse después de dividir.
- El crupier pide carta en un 17 suave (soft 17).

Aspectos a tener en cuenta:

- Todas las manos se juegan "uno contra uno" (un solo jugador contra el crupier).
- El jugador siempre juega según la estrategia básica correcta.
- La apuesta inicial para cada mano siempre es 1.
- Los 10, J, Q, K se consideran iguales en Blackjack y se representan como 10.
- Los Ases se representan como 11.
- Los palos (♠, ♥, ♦, ♣) no son relevantes y no se consideran.
- Los valores de cuenta en marcha (*Run count*) y cuenta verdadera (*True count*) (redondeados a precisión de enteros) se registran según el sistema Hi-Lo al inicio de cada ronda, antes de repartir cualquier carta.
- El número de cartas restantes en el zapato se registra al inicio de cada ronda antes de repartir cartas. Este número incluye cartas que no se jugarán debido a una penetración de baraja menor al 100%.

La lista de acciones que puede tomar el jugador y sus descripciones son las básicas del juego:

Acción	Descripción
H	Pedir carta (Hit)
S	Plantarse (Stand)
D	Doblar la apuesta (Double Down)
P	Dividir (Split)
N	No Insurance
R	Rendirse (Surrender)

Análisis exploratorio

El conjunto de datos cuenta con 100 millones de iteraciones de una hand de blackjack de 1 jugador contra el crupier. Las columnas que lo conforman son:

- **shoe_id**: Identificador del zapato (shoe).
- **cards_remaining**: Cartas restantes en el zapato.
- **dealer_up**: Carta visible del crupier.
- **initial_hand**: Mano inicial del jugador.
- **dealer_final**: Mano final del crupier.
- **dealer_final_value**: Valor final de la mano del crupier.
- **player_final**: Mano final del jugador.
- **player_final_value**: Valor final de la mano del jugador.
- **actions_taken**: Acciones realizadas por el jugador.
- **run_count**: Cuenta en marcha (*Running count*).
- **true_count**: Cuenta verdadera (*True count*).
- **win**: Resultado de la mano (indicador de victoria).

De estas las que decidimos extraer son las 7 marcadas de azul. Y para las pruebas se utilizaron las primeras 500,000 manos. Las columnas seleccionadas corresponden a los siguientes tipos de datos:

- Int64: Para las variables numéricas enteras (cards_remaining, dealer_up, run_count, true_count).

- Float64: Para la variable win, que representa ganancias o pérdidas y puede incluir valores decimales.

- object: Para las variables que contienen cadenas o estructuras complejas (initial_hand, actions_taken).

No se encontraron valores nulos en ninguna de las columnas, lo que indica que el conjunto de datos está completo y listo para el análisis sin necesidad de tratamiento adicional para datos faltantes.

Las primeras cinco filas del conjunto de datos nos permiten observar ejemplos concretos de cómo están estructurados los registros:

- cards_remaining: Varía entre 395 y 416 en estas muestras iniciales, reflejando diferentes etapas del mazo durante las partidas.

- dealer_up: Muestra valores altos como 10 y 8, representando cartas visibles del crupier que pueden influir en las decisiones del jugador.

- initial_hand: Se presenta como una lista de cartas, por ejemplo, [10, 11] o [5, 5], indicando las cartas iniciales que recibió el jugador.

- actions_taken: Refleja las decisiones del jugador, como [['S']] para "Stand" o [['H', 'S']] para "Hit" seguido de "Stand".

- win: Indica el resultado de la mano, con valores como 1.5 (posible blackjack), 1.0 (victoria), 0.0 (empate) y -1.0 (derrota).

Para las variables numéricas podemos ver las siguientes estadísticas:

- cards_remaining:

    - Media: Aproximadamente 248.68 cartas restantes.
    - Desviación Estándar: 98.27, indicando una variabilidad significativa en el número de cartas restantes entre manos.
    - Mínimo y Máximo: Desde 79 hasta 416 cartas, lo que corresponde a diferentes momentos en el uso del mazo durante el juego.

- dealer_up:

    - Media: 7.30, sugiriendo que las cartas visibles del crupier suelen ser intermedias.
    - Desviación Estándar: 2.92, reflejando una distribución uniforme de las cartas del crupier.
    - Rango: Valores entre 2 y 11 (considerando que el As puede valer 11).

- run_count y true_count:

    - Media: Cercanas a cero (0.195 y 0.044 respectivamente), lo cual es común en conteos donde las cartas altas y bajas se equilibran a lo largo del tiempo.
    - Desviación Estándar: 7.91 para *run_count* y 2.04 para *true_count*, indicando fluctuaciones en el conteo de cartas.
    - Valores Extremos: *run_count* oscila entre -35 y 36, mientras que *true_count* varía de -19 a 16.

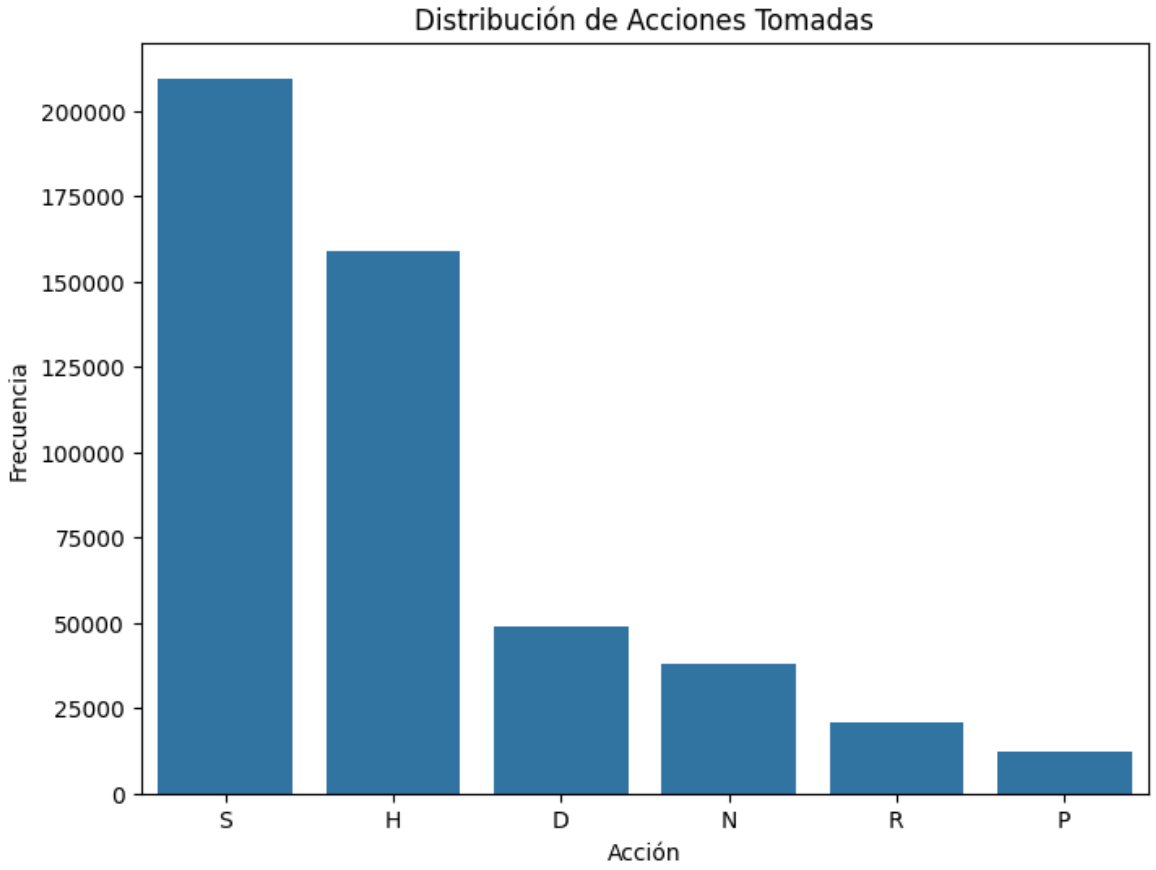
- win:

    - Media: -0.0058, sugiriendo que las pérdidas y ganancias están casi equilibradas, con una ligera tendencia hacia la pérdida.
    - Desviación Estándar: 1.1459, mostrando variabilidad en los resultados de las manos.

- Valores Mínimo y Máximo: Desde -7.0 hasta 7.0, lo que puede indicar ganancias o pérdidas excepcionales, posiblemente debido a dobles, splits o blackjack.

Para entender mejor el conjunto de datos, realizamos los siguientes gráficos estadísticos:

Gráfico #1: Distribución de las Acciones Tomadas por el jugador



Cómo se puede observar la mano más repetida en el conjunto de datos es Stand, seguido por Hit. Lo cual hace sentido ya que estos son las dos jugadas básicas del juego.

Gráfico #2: Distribución de las ganancias y pérdidas en unidades de 1

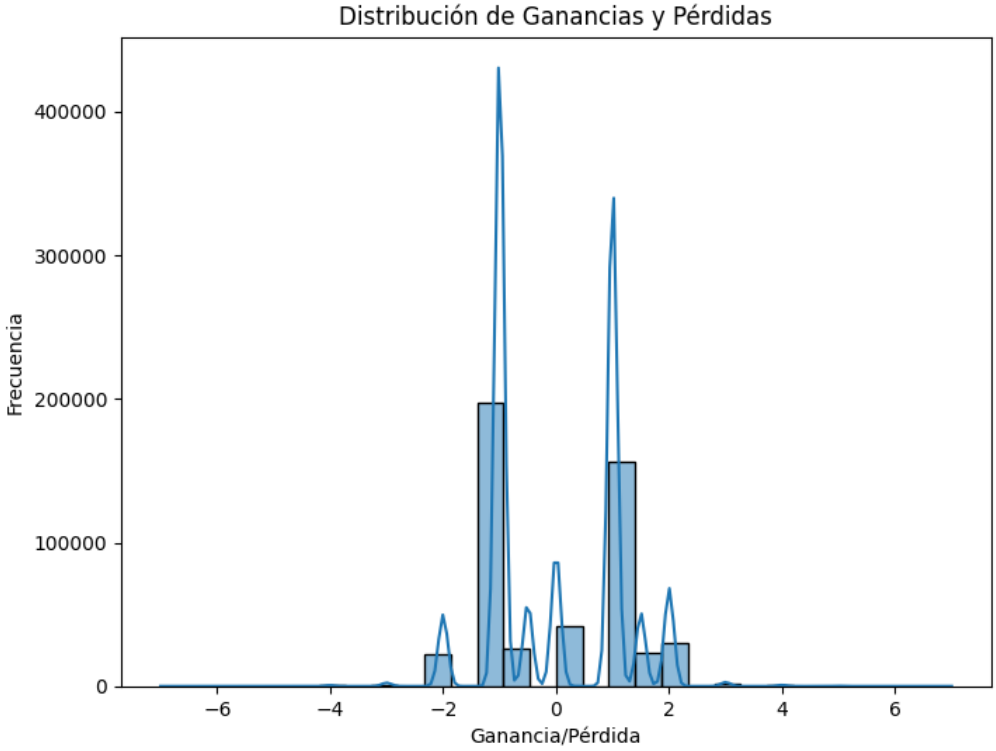
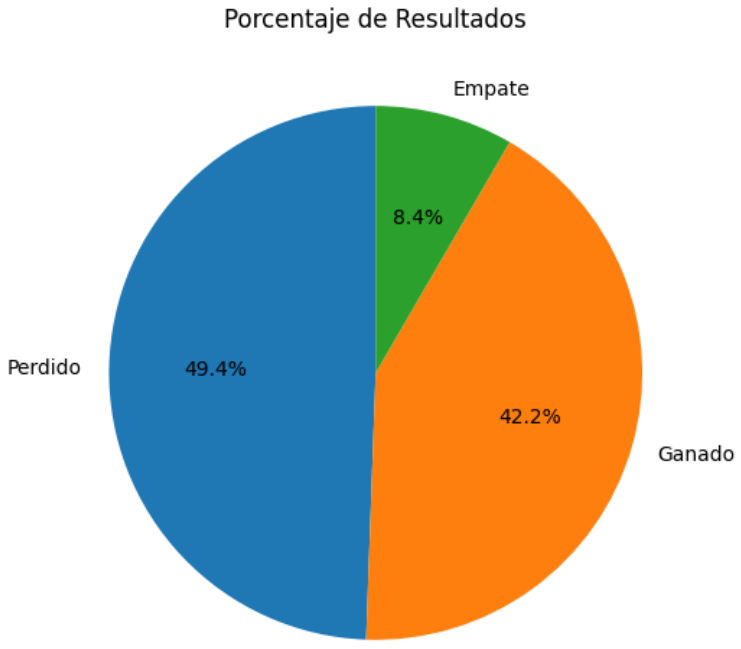
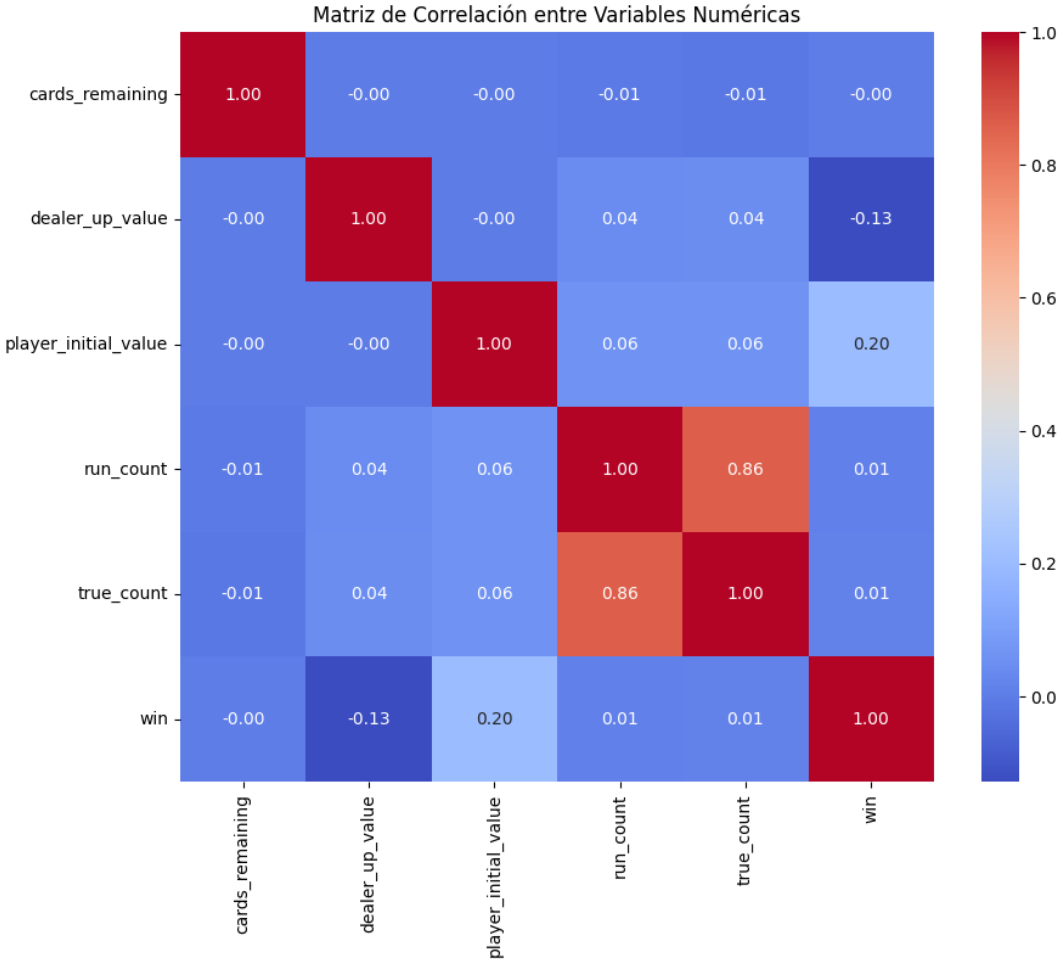


Gráfico #3: Porcentaje de Resultados de ganancia en total



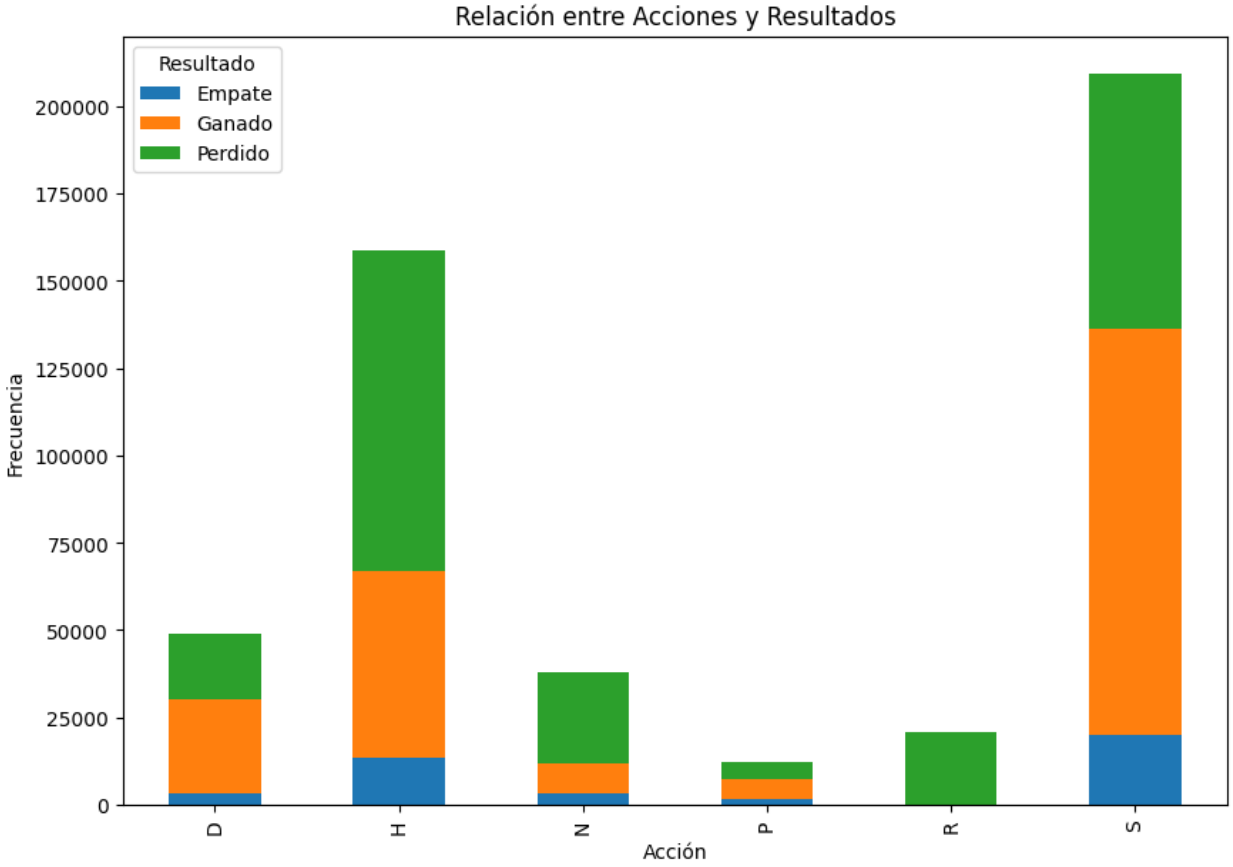
Luego, evaluando las ganancias y pérdidas del conjunto de datos podemos ver que sigue una distribución estándar en cuanto a las estadísticas de Blackjack, ya que obviamente la casa tiene la ventaja. Sin embargo el 42% de las partidas ganadas indica que el dataset es una simulación correcta ya que el porcentaje de victorias esperado en blackjack utilizando la estrategia básica generalmente está entre el 42% y el 44%.

Gráfico #4: Matriz de correlación entre las variables numéricas



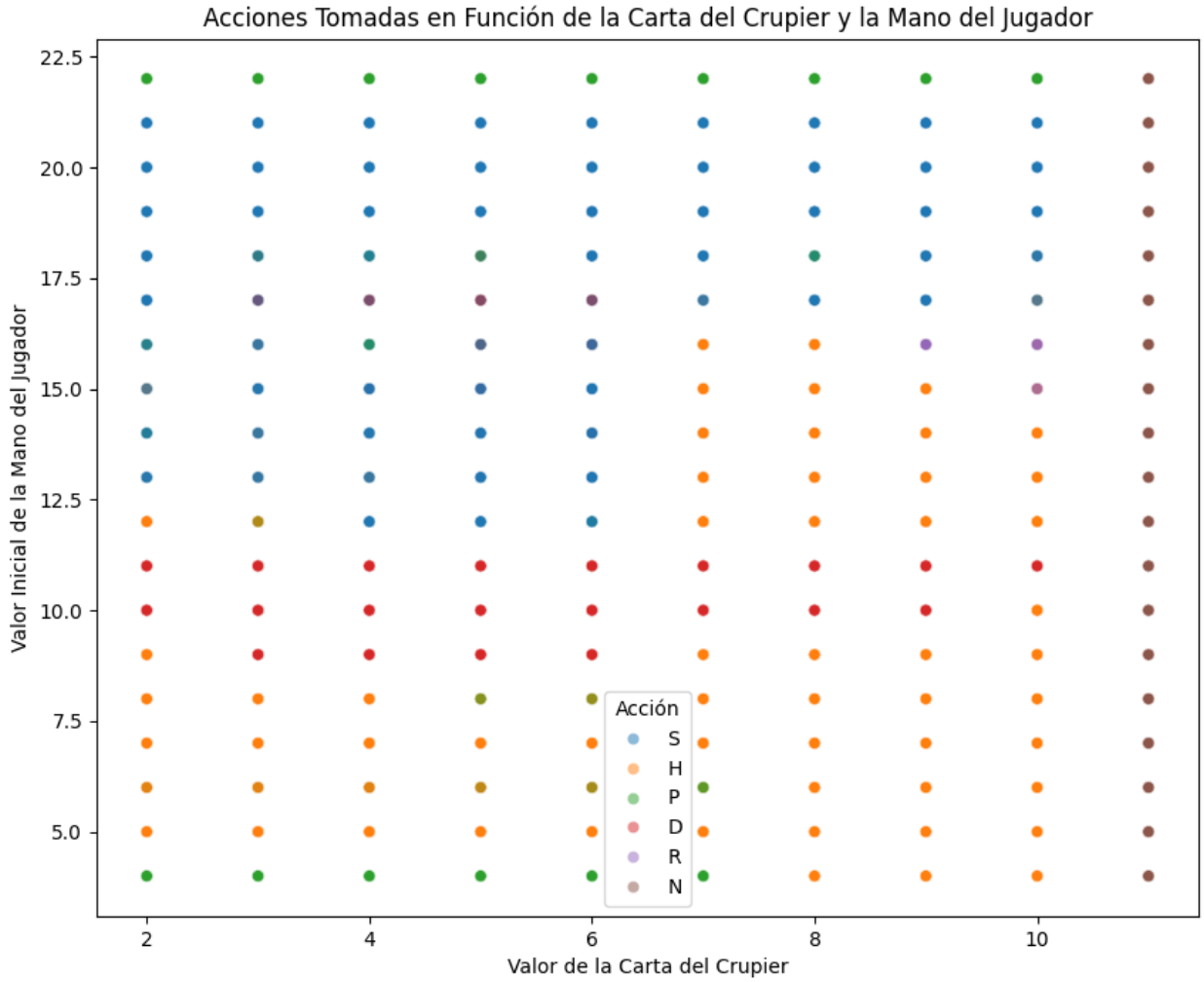
Al analizar la matriz de correlación podemos notar que no hay mayor correlación entre las variables numéricas realmente importantes o significativas para el análisis de la construcción del modelo, ya que la relación entre run_count y true_count es a causa de que están midiendo lo mismo.

Gráfico #5: Relación entre la acción tomada y el resultado de la hand



En este otro gráfico podemos observar el resultado de la partida dependiendo de la decisión del jugador. Lo significativo de esta gráfica lo obtuvimos al comparar los resultados de Stand vs Hit, ya que se puede observar que en la mayoría de los hits, se pierde la partida, mientras que stand gana la mayoría de sus partidas. Otro caso particular es el double, que gana más de lo que pierde.

Gráfico #6: Mapeo de la distribución de las acciones tomadas dependiendo el valor de la carta inicial tanto del jugador como del Crupier



En el gráfico se reflejan claramente las decisiones del jugador según la estrategia básica de Blackjack, con algunos patrones a destacar:

1. Manos bajas del jugador (valores entre 5 y 11):
    - En este rango, el jugador predominantemente elige H (Hit), es decir, pedir carta. Esto es consistente con la estrategia básica, ya que con valores bajos es necesario intentar mejorar la mano para acercarse al 21.
2. Manos medias del jugador (valores entre 12 y 16):
    - En este rango, las decisiones varían dependiendo del valor de la carta visible del crupier:

        - Si el crupier tiene una carta alta (como un 10), el jugador suele elegir H (Hit) para intentar mejorar su mano y evitar perder automáticamente.
        - Si el crupier tiene una carta baja (como 2, 3 o 4), el jugador tiende a plantarse (S, Stand). Esto se debe a que, según la estrategia básica, existe una mayor probabilidad de que el crupier supere los 21 debido a las reglas del Blackjack, donde el crupier debe pedir carta hasta alcanzar al menos 17.

3. Manos altas del jugador (valores de 17 en adelante):

    - Aquí la mayoría de las acciones corresponden a S (Stand), lo que significa que el jugador se planta. Esto es lógico según la estrategia básica, ya que con una mano alta el riesgo de pasarse de 21 es muy alto, y se prefiere dejar que el crupier asuma el riesgo de perder.

4. Valores específicos como 9, 10 y 11:

    - En estos casos, el jugador tiende a elegir D (Double Down), doblando la apuesta. Esto es parte de la estrategia básica, ya que existe una alta probabilidad de que la carta adicional sea un 10, resultando en una mano fuerte (19, 20 o 21).

5. Cuando el jugador tiene una mano alta y el crupier una carta baja:

    - En estas situaciones, el jugador casi siempre elige S (Stand). Esto ocurre porque la probabilidad de que el crupier supere 21 (se pase) es significativa al tener una carta baja. Por ejemplo, si el crupier tiene un 2 o un 3, necesitará pedir varias cartas para alcanzar al menos 17, aumentando el riesgo de pasarse. Plantarse en estas condiciones es una estrategia que maximiza las probabilidades de ganar sin asumir riesgos innecesarios.

Metodología

Preprocesamiento de Datos

El preprocesamiento de datos es una etapa esencial para garantizar que el modelo de aprendizaje automático pueda interpretar y aprender eficazmente de los datos proporcionados. A continuación, se detallan los pasos realizados en esta etapa:

1. Selección de Variables:

   Nos centramos en las variables más relevantes para nuestro modelo, seleccionando las siguientes columnas del conjunto de datos:

   - cards_remaining: Número de cartas restantes en el mazo.
   - dealer_up: Carta visible del crupier.
   - initial_hand: Cartas iniciales del jugador.
   - run_count: Conteo acumulativo de cartas altas y bajas (para estrategias de conteo de cartas).
   - true_count: Conteo verdadero ajustado por el número de mazos restantes.
   - actions_taken: Acciones realizadas por el jugador.
   - win: Resultado de la mano (ganancia o pérdida).

2. Transformación de Cartas:

   Para que el modelo pueda procesar las cartas, es necesario convertirlas en valores numéricos:

   - Carta del Crupier (dealer_up): Convertimos la carta visible del crupier a su valor numérico correspondiente. Las cartas J (Jack), Q Queen) y K (King) se asignan un valor de 10, y el As (A) inicialmente se asume como 11.

   - Mano Inicial del Jugador (initial_hand): Convertimos las cartas de la mano inicial del jugador a sus valores numéricos utilizando la misma función card_value.

3. Extracción de Características:

   - Valores de las Cartas Iniciales: Extraemos los valores numéricos de las dos primeras cartas del jugador para crear nuevas características. Separando las cartas en 2 columnas.
   - Valor de la Mano del Jugador: Calculamos el valor total de la mano inicial del jugador, teniendo en cuenta el valor flexible del As (11 o 1).
   - Extracción de la Primera Acción: Utilizamos expresiones regulares para extraer la primera acción tomada por el jugador, ya que actions_taken puede contener una lista de acciones.

- Eliminación de Registros Nulos: Eliminamos las filas donde no se pudo extraer una acción válida.

4. Modificación de Acciones (Versión 2 del Proyecto):

En la segunda versión del proyecto, exploramos cómo la modificación de acciones en manos perdedoras afecta al aprendizaje del modelo:

- Identificación de Manos Perdedoras: Creamos una máscara para identificar las manos en las que el jugador perdió.
- Selección Aleatoria del 50% de Manos Perdedoras: Seleccionamos aleatoriamente la mitad de las manos perdedoras para modificar sus acciones.
- Función para Modificar Acciones: Definimos una función que cambia la acción original a otra alternativa. Por ejemplo, si la acción original fue 'H' (Hit), la cambiamos a 'S' (Stand).

5. Codificación y Normalización:

- Codificación de Acciones: Utilizamos LabelEncoder para convertir las acciones categóricas en valores numéricos.
- Normalización de Características Numéricas: Aplicamos StandardScaler para normalizar las características numéricas y mejorar el rendimiento del modelo.

Implementación del Modelo

Desarrollamos una red neuronal profunda utilizando TensorFlow y Keras para predecir la acción óptima que el jugador debe tomar en función del estado del juego.

Arquitectura de la Red Neuronal:

- Capa de Entrada:
    - Características de Entrada: La red recibe un vector de entrada con las siguientes características preprocesadas:
        - cards_remaining: Número de cartas restantes en el mazo.
        - dealer_up: Valor numérico de la carta visible del crupier.
        - initial_card_1: Valor numérico de la primera carta del jugador.
        - initial_card_2: Valor numérico de la segunda carta del jugador.
        - run_count: Conteo acumulativo de cartas.
        - true_count: Conteo verdadero.
        - player_hand_value: Valor total de la mano inicial del jugador.
- Capas Ocultas:
    1. Primera Capa Densa:
        - Número de Neuronas: 64
        - Función de Activación: ReLU (Rectified Linear Unit)
        - Descripción: Esta capa captura patrones iniciales y relaciones no lineales entre las características de entrada.
    2. Segunda Capa Densa:
        - Número de Neuronas: 128
        - Función de Activación: ReLU
        - Descripción: Aumenta la capacidad de la red para aprender representaciones más complejas y abstracciones de los datos.
    3. Tercera Capa Densa:
        - Número de Neuronas: 64
        - Función de Activación: ReLU
        - Descripción: Reduce la dimensionalidad y ayuda a consolidar el aprendizaje antes de la capa de salida.
- Capa de Salida:

- Número de Neuronas: Igual al número de acciones posibles según las clases del mapeo.
- Función de Activación: Softmax
- Descripción: Genera una distribución de probabilidad sobre las acciones posibles, permitiendo seleccionar la acción con mayor probabilidad.

Parámetros e Hiperparámetros:

- Optimizador: Adam
    - Justificación: Adam es un optimizador eficiente que combina las ventajas de los algoritmos AdaGrad y RMSProp. Es adecuado para problemas con grandes conjuntos de datos y parámetros.

- Función de Pérdida: sparse_categorical_crossentropy
    - Justificación: Se utiliza para problemas de clasificación multicategoría con etiquetas enteras. Es apropiada cuando las etiquetas están codificadas como enteros en lugar de one-hot.

- Métrica de Evaluación: Accuracy
    - Justificación: La precisión es una métrica adecuada para evaluar el rendimiento en problemas de clasificación.

- Épocas: 20
    - Justificación: Se eligieron 20 épocas luego de un par de pruebas para permitir que el modelo aprenda de los datos sin sobreajustarse, a partir de este punto se estabiliza la pérdida.

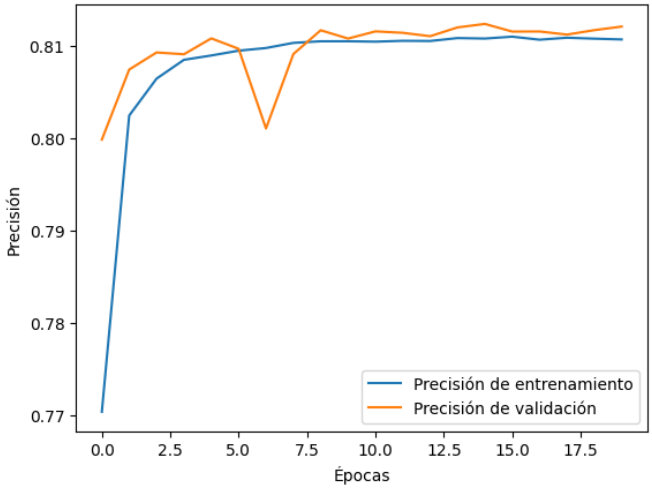
- Batch Size: 32
    - Justificación: Un tamaño de lote de 32 es un valor común que ofrece un equilibrio entre la estabilidad del entrenamiento y la eficiencia computacional.

Resultados

Entrenamiento:

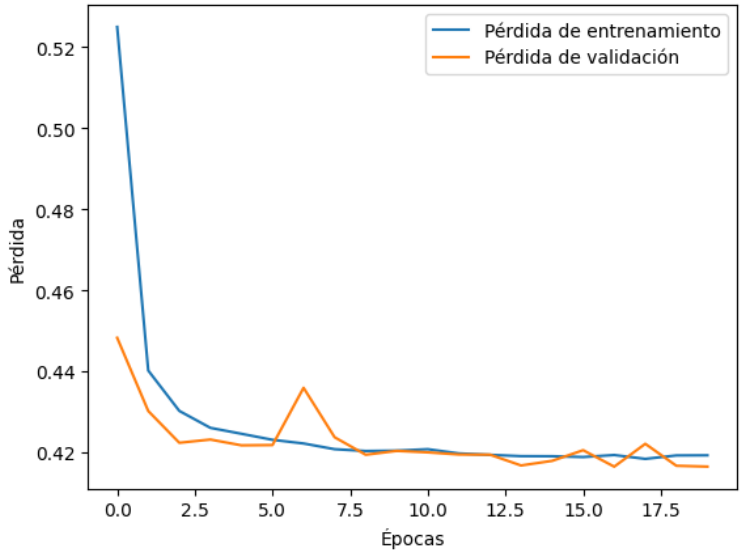
Los gráficos presentados muestran el desempeño del modelo durante el entrenamiento y la validación a lo largo de las épocas. En el primer gráfico se observa la evolución de la precisión, mientras que en el segundo se muestra la pérdida (error). Estos indicadores permiten analizar tanto la capacidad del modelo para aprender a partir de los datos de entrenamiento como su generalización a datos no vistos (validación). El objetivo es minimizar la pérdida y maximizar la precisión en ambos conjuntos, asegurando que el modelo no presente problemas de sobreajuste o subajuste.

Gráfico #7: Precisión de entrenamiento y validación a lo largo de las épocas



En este gráfico, se observa un aumento constante en la precisión tanto para los datos de entrenamiento como de validación en las primeras épocas, alcanzando un valor cercano al 81% después de unas pocas iteraciones. La precisión de validación muestra pequeñas fluctuaciones, lo cual es normal debido a la variabilidad de los datos. Sin embargo, ambas curvas se estabilizan hacia el final, lo que indica que el modelo ha convergido y no presenta signos evidentes de sobreajuste, ya que la precisión de validación no disminuye significativamente ni diverge de la de entrenamiento.

Gráfico #8: Pérdida durante el entrenamiento y la validación

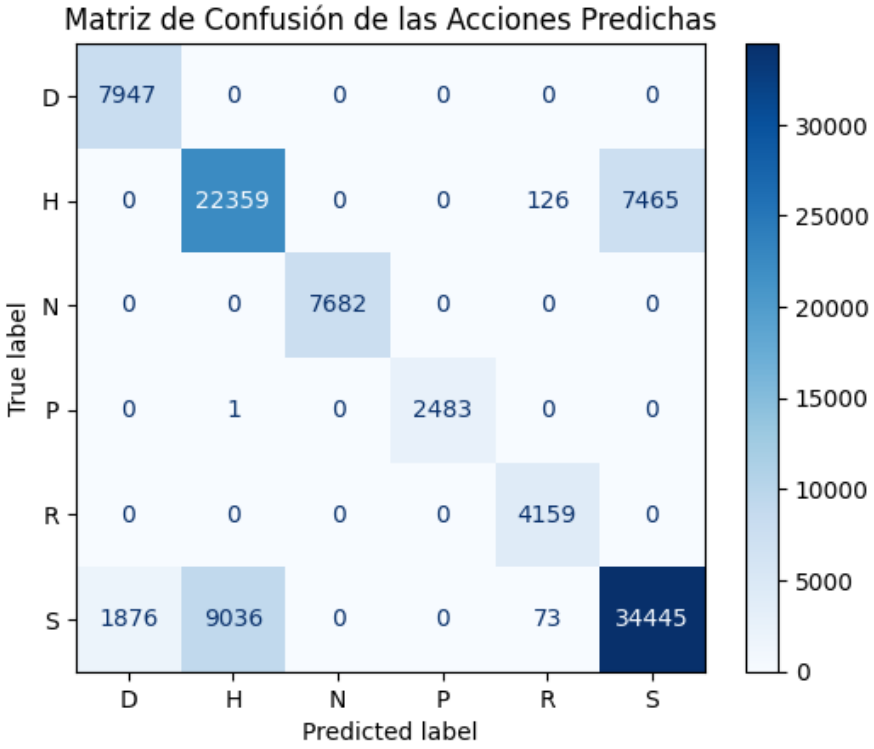


La pérdida muestra una disminución rápida en las primeras épocas, lo que indica un aprendizaje efectivo al inicio del entrenamiento. Posteriormente, las curvas de pérdida de entrenamiento y validación se estabilizan en valores bajos y similares, alrededor de 0.42. Esto sugiere que el modelo ha alcanzado un equilibrio entre aprendizaje y generalización. Aunque hay pequeñas oscilaciones en la pérdida de validación, estas son esperables y no indican un problema grave de sobreajuste o inestabilidad.

Análisis de Resultados

Lo primero a evaluar es la precisión de las acciones predichas, a comparación de los valores reales.

Gráfica #9: Matriz de confusiones de las acciones predichas



Acciones correctamente clasificadas (diagonal principal):

- D (Double Down): Se predijeron correctamente 7947 de las acciones.
- H (Hit): Es la acción más frecuente, con 22,359 clasificaciones correctas. Sin embargo, hay 7465 mal clasificadas como S (Stand).
- N (No Insurance): El modelo clasificó correctamente 7682 de estas acciones, lo cual es bastante preciso.
- P (Split): Aunque menos frecuentes, 2483 se clasificaron correctamente.
- R (Surrender): Todas las 4159 acciones de rendición fueron clasificadas correctamente.
- S (Stand): Con 34,445 clasificaciones correctas, esta acción también tiene un alto grado de precisión.

Errores frecuentes:

- Las acciones "H (Hit)" presentan una confusión significativa con "S (Stand)", con 7465 casos mal clasificados. Esto podría deberse a similitudes contextuales en los valores de las manos del jugador y del crupier, donde estas acciones son comúnmente intercambiables.

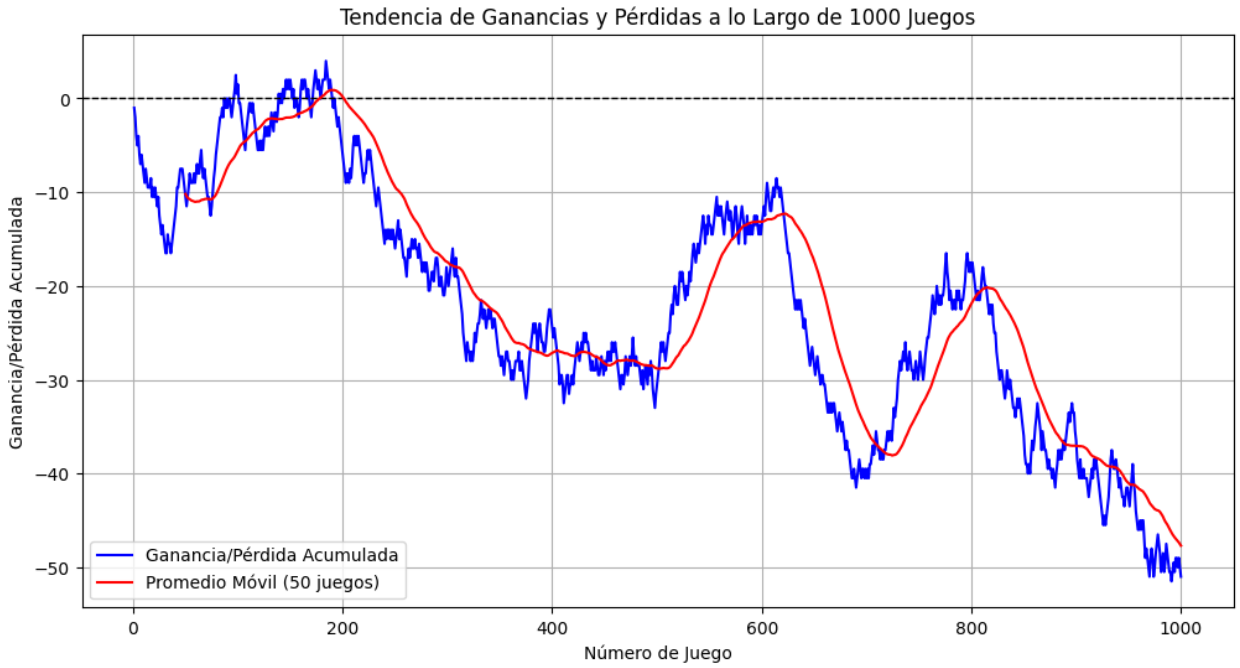
- Para "S (Stand)", 9036 casos fueron incorrectamente clasificados como "H (Hit)". Esto puede indicar que el modelo no logra distinguir correctamente los contextos en los que es mejor plantarse en lugar de pedir carta.

Acciones menos problemáticas:

- R (Surrender) y P (Split) muestran una excelente precisión con prácticamente ningún error de clasificación.
- D (Double Down) también tiene muy pocos errores, con solo 1876 mal clasificadas como "S (Stand)", lo cual puede ser atribuible a similitudes en situaciones estratégicas.

Luego queremos ver cómo se desempeñaría nuestro modelo luego de 1000 manos de blackjack, para esto, simulamos las jugadas y podemos observar el siguiente análisis de las ganancias y pérdidas en base a unidades de apuesta:

Gráfico #10: Tendencia de ganancias y pérdidas a lo largo de 1000 manos de blackjack



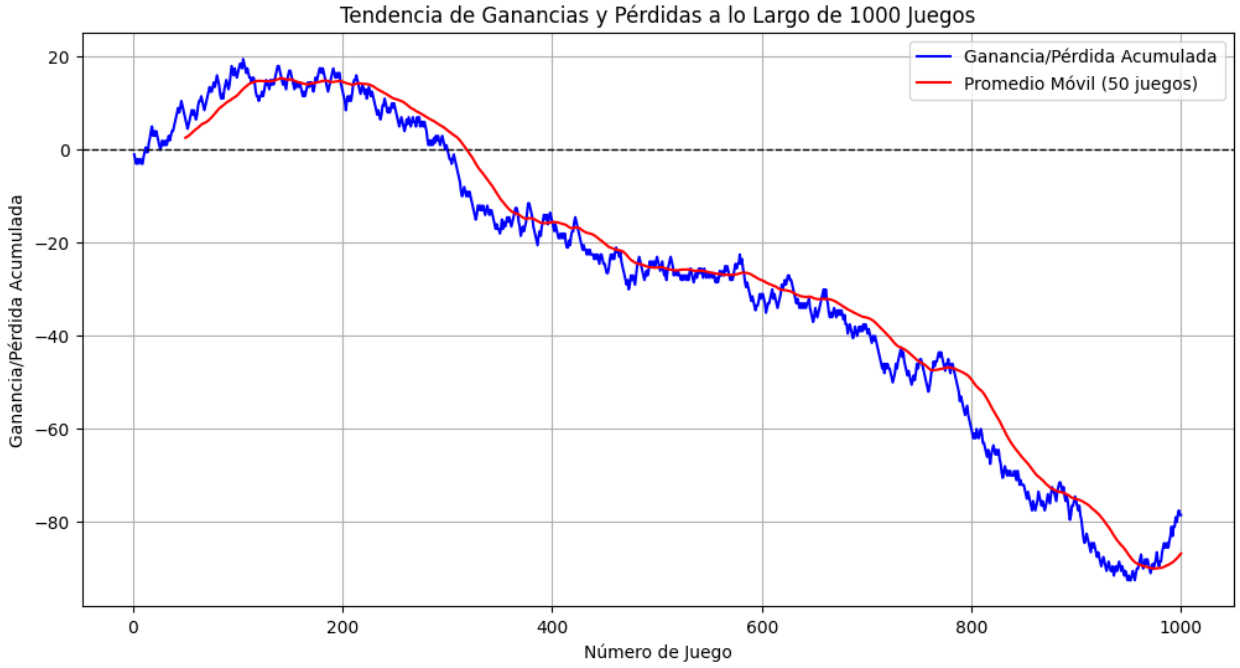
En los primeros 200 juegos, el modelo parece tomar decisiones favorables que generan ganancias acumuladas. Durante esta fase inicial, el desempeño del modelo se mantiene por encima del punto de equilibrio, lo que indica que está logrando ejecutar estrategias que lo colocan en una posición ventajosa. Esto sugiere que el modelo puede ser eficaz en el corto plazo, al menos bajo las condiciones iniciales del juego.

Sin embargo, a partir del juego 200, se observa una caída en el desempeño. Las pérdidas comienzan a acumularse de manera más consistente, lo que sugiere que el modelo no logra adaptarse adecuadamente a las estrategias del crupier o a las variaciones en las manos que se presentan a medida que avanza el juego. Este cambio en la tendencia podría deberse a una falta de flexibilidad en las decisiones del modelo o a limitaciones en su capacidad para manejar escenarios más complejos.

En términos generales, la línea roja del promedio móvil muestra una tendencia decreciente clara, lo que confirma que el modelo tiene dificultades para mantener un desempeño positivo a largo plazo. Esto evidencia que, aunque el modelo puede ofrecer buenos resultados inicialmente, su estrategia no es sostenible, lo que lleva a un rendimiento global negativo en el transcurso de los 1000 juegos.

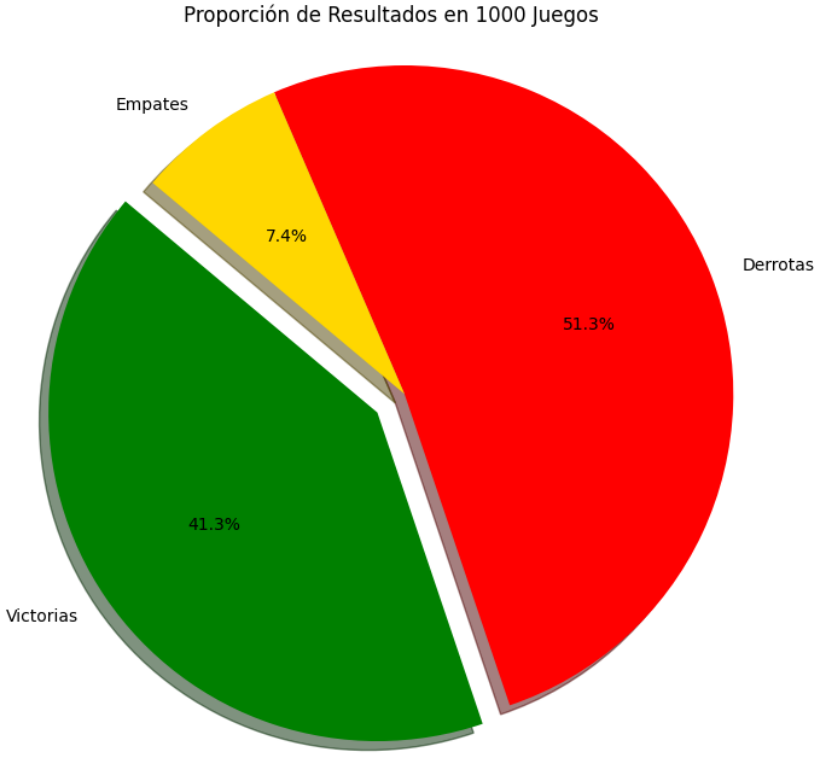
Para verificar los datos realizamos una segunda simulación:

Gráfico #11: Segunda iteración de la simulación de 1000 juegos



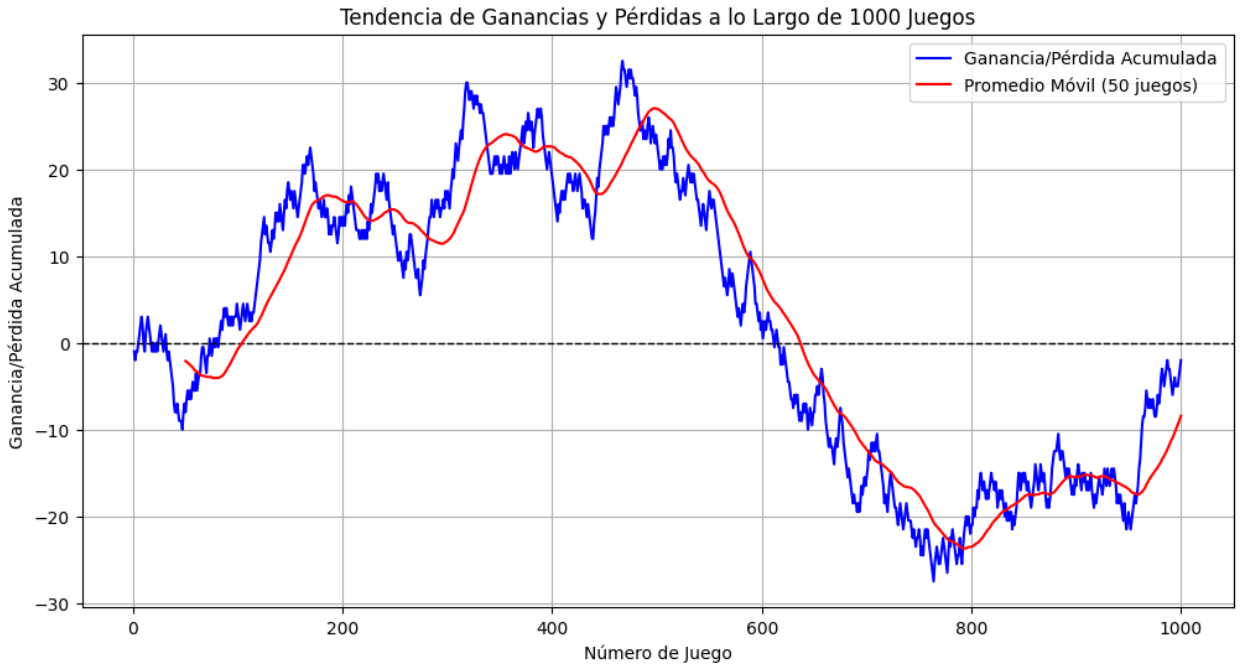
En este caso podemos ver que durante las primeras 300 iteraciones el modelo fue positivo, obteniendo ganancias de hasta 20 unidades de apuesta. Sin embargo, luego empieza un declive que lo desciende hasta las 80 unidades de apuesta en pérdida. Para esta segunda iteración podemos visualizar la tasa de victorias:

Gráfico #13: Tasa de victoria durante la segunda simulación de 1000 iteraciones



Aunque la tasa de victorias (41.3%) no alcanza el rango superior de la estrategia básica (42%-44%), el modelo se encuentra cerca de este umbral. Esto sugiere que está utilizando un enfoque razonablemente sólido basado en la estrategia básica. Sin embargo, el alto porcentaje de derrotas (51.3%) destaca áreas de mejora potencial para optimizar el modelo, como una mejor adaptación a escenarios desfavorables o ajustes en las decisiones para maximizar las probabilidades de ganar. A pesar de estas limitaciones, el desempeño general no es malo y muestra un buen acercamiento a los principios estratégicos del Blackjack.

Siguiendo el dicho de que la tercera es la vencida, realizamos una tercera simulación de 1000 juegos:



En está el modelo logró alcanzar hasta 32 unidades de apuesta en positivo, a lo largo de aproximadamente 500 juegos. Y luego empezó en una racha de pérdida hasta recuperarse al final para terminar con una pérdida neta de 2 unidades de apuesta. En esta iteración tuvo un 43.30% de tasa de victoria, lo que sirvió para demostrarnos que en realidad el modelo nunca va a poder romper la regla del juego, donde la casa siempre tiene la ventaja y al final es un juego de probabilidades y suerte.

Discusión

En este proyecto, desarrollamos un modelo de red neuronal profunda con el objetivo de predecir la acción óptima que un jugador debe tomar en una partida de blackjack, basándonos en el estado del juego. Tras llevar a cabo el preprocesamiento de datos y diseñar una arquitectura de red adecuada, procedimos a entrenar el modelo y evaluar su desempeño.

Los resultados obtenidos muestran que el modelo alcanzó una precisión aproximada del 81% tanto en los datos de entrenamiento como en los de validación. Este nivel de precisión indica que el modelo logró aprender patrones significativos a partir de los datos sin presentar signos evidentes de sobreajuste, ya que la precisión en el conjunto de validación se mantuvo estable y cercana a la del entrenamiento a lo largo de las épocas. Esto fue lo que se buscó y no intentamos conseguir una precisión tan perfecta ya que la idea es tener los principios de la estrategia básica del juego pero con el ajuste a las manos pérdidas.

Al analizar la matriz de confusión de las acciones predichas, observamos que el modelo clasifica correctamente la mayoría de las acciones, especialmente aquellas como "Rendirse" (R) y "Dividir" (P), que presentan una alta tasa de aciertos y casi ningún error de clasificación. Sin embargo, identificamos una confusión significativa entre las acciones "Pedir" (H) y "Plantarse" (S). Específicamente, hay un número considerable de casos donde el modelo predice "Plantarse" cuando la acción real era "Pedir", y viceversa. Esta confusión podría deberse a similitudes en las situaciones de juego donde ambas acciones son estratégicamente viables, lo que dificulta al modelo distinguir entre ellas con mayor precisión. También está relacionado a que la versión utilizada para los resultados es la que presenta la modificación del 50% de las jugadas perdidas, usualmente cambiando Hit por Stand y viceversa.

En las simulaciones realizadas, donde el modelo jugó 1000 manos de blackjack, inicialmente observamos un desempeño favorable, con ganancias acumuladas en los primeros 200 juegos. Esto sugiere que el modelo puede tomar decisiones efectivas en el corto plazo. Sin embargo, a medida que avanzan las partidas, se evidenció una tendencia negativa en las ganancias, resultando en pérdidas acumuladas a largo plazo. La línea de tendencia del promedio móvil confirma esta disminución constante en el desempeño del modelo. Este comportamiento indica que, aunque el modelo es capaz de implementar estrategias razonables en fases iniciales, carece de adaptabilidad para mantener un rendimiento positivo sostenido en el tiempo. Aparte de que es un comportamiento normal en cuanto al juego Blackjack, donde por el pequeño porcentaje de ventaja que siempre tiene la casa, conforme aumentan las iteraciones el comportamiento natural matemático es que el jugador empiece a perder.

Las tasas de victorias obtenidas fueron desde 41.3% a 43.33%, lo cual, aunque cercana al rango de la estrategia básica del blackjack (entre 42% y 44%), no logra superarla. Esto implica que el modelo está aplicando principios fundamentales de la estrategia básica, pero no incorpora mejoras significativas que le permitan obtener una ventaja más notable sobre estrategias convencionales. Además, el porcentaje relativamente alto de derrotas sugiere que el modelo

podría beneficiarse de ajustes que le permitan manejar mejor situaciones desfavorables o decisiones más complejas.

Estos resultados significan que el modelo tiene una comprensión general de las estrategias básicas del blackjack y es capaz de aplicarlas hasta cierto punto. No obstante, las limitaciones observadas, especialmente en la confusión entre acciones críticas y en el desempeño a largo plazo, indican que el modelo no alcanza un nivel óptimo de rendimiento. Esto se traduce en la incapacidad de mantener ganancias consistentes en simulaciones prolongadas y en no superar efectivamente estrategias básicas ya establecidas.

Considerando estos hallazgos, podemos afirmar que los resultados no son los mejores posibles. Existen áreas claras de mejora, como el refinamiento del preprocesamiento de datos para abordar el desbalance en la frecuencia de las acciones y la incorporación de características adicionales que puedan proporcionar más contexto al modelo. Asimismo, ajustar la arquitectura de la red neuronal, quizás incorporando más capas ocultas o neuronas, o implementando técnicas de regularización como el dropout, podría mejorar la capacidad del modelo para capturar patrones más complejos y reducir la confusión entre acciones similares.

Un descubrimiento significativo obtenido de este análisis es la influencia crítica que tiene el preprocesamiento de datos en el rendimiento del modelo. La forma en que se manipularon las acciones en manos perdedoras y cómo se representaron las características influyó directamente en la capacidad del modelo para aprender y generalizar. Además, identificamos que las acciones "Hit" y "Stand" son áreas problemáticas que requieren una atención especial, posiblemente debido a que representan decisiones estratégicas que dependen de contextos muy específicos y sutiles del juego. Esto sugiere que el modelo podría beneficiarse de un enfoque que considere secuencias de acciones o que utilice modelos más adecuados para datos secuenciales, como redes neuronales recurrentes o modelos de aprendizaje por refuerzo.

En conclusión, aunque el modelo desarrollado muestra un desempeño aceptable y es capaz de replicar en cierta medida estrategias básicas del blackjack, no alcanza un nivel de excelencia. Los resultados obtenidos resaltan la complejidad de modelar decisiones estratégicas en juegos de azar y la necesidad de enfoques más avanzados y datos más ricos para mejorar el rendimiento. Este proyecto nos permitió comprender mejor las limitaciones y desafíos asociados con el aprendizaje automático en contextos de toma de decisiones complejas y nos da las bases para futuras investigaciones que busquen superar estas barreras.

Conclusiones

- El modelo de red neuronal profunda alcanzó una precisión del 81%, demostrando su capacidad para aprender patrones significativos en el juego de blackjack sin evidentes signos de sobreajuste.

- Se observó una confusión notable entre las acciones 'Pedir' (Hit) y 'Plantarse' (Stand), lo que indica que el modelo no distingue completamente las sutilezas estratégicas necesarias para diferenciar estas decisiones clave.

- Las simulaciones de 1000 manos revelaron que el modelo no mantiene un desempeño positivo a largo plazo, acumulando pérdidas y sin superar la tasa de victorias de la estrategia básica, lo que resalta la necesidad de mejoras en el preprocesamiento de datos y en la arquitectura del modelo para obtener resultados óptimos.

- La arquitectura de red neuronal elegida resultó efectiva para capturar los aspectos esenciales de la toma de decisiones en blackjack, validando la idoneidad de los modelos de aprendizaje profundo para este tipo de problemas.

- El conjunto de datos proporcionó una representación integral de escenarios de blackjack, permitiendo que el modelo aprendiera y generalizara estrategias fundamentales de manera eficiente.

Apéndice

Repositorio:

https://github.com/andresquez/AIBlackJack

Para ejecutar en línea:

- Versión 2: *https://colab.research.google.com/drive/1R-NNLTNms0Yh-NkTfNOIll_1LSM7xujv#scrollTo=3QHZgiZvk-Q6*
- EDA:https://colab.research.google.com/drive/1D2PFGO7AlUkKCpniPnoTCoT2jhyYIaii#scrollTo=DZdSvoR0UQME