

Texas Hold'Em + Poker Hand (Model)

- Juan Pablo Estrada Lucero
- Max Arturo Marroquín Arango





Análisis del Enunciado - Texas Hold 'Em

Imaginen que están jugando una partida de **Texas Hold'Em** uno contra uno. Supongan que tienen un balance de 100 ciegas grandes (BBs). Su sesión de juego termina al jugar 100 manos o al perderlo todo.

Escenarios propuestos:

- Cada mano apuesta exactamente 1 BB
- Cada mano aspuesta una fracción de su balance actual
- Cada mano ganan 1 BB con probabilidad 0.70, pierden 5 BBs con probabilidad 0.10, con probabilidad 0.15 ganan 5 BBs y con probabilidad 0.05 lo apuestan todo y pierden

Bernoulli

Un intento con dos salidas (éxito o fracaso)

Ejemplo: Tirar una moneda una vez

Binomial

Número de éxitos en un número de intentos independientes.

Ejemplo: Número de caras en 10 tiros de moneda

Geométrica

Número de intentos hasta que ocurra el primer éxito.

Ejemplo: Cuantos tiros de dados se realizaron hasta obtener dos 6's.



Distribución de probabilidad para Bt luego de 100 manos

Distribución Binomial

Carecterísticas del juego:

- Cada mano = Intento independiente
- Éxito = Ganar la mano
- Fracaso = Perder la mano

Variable aleatoria = Número total de manos ganadas en la sesión de 100 manos.

Criterios esenciales:

- Número de intentos $n = 100$
- Intentos independientes
- Ganar / Perder
- Probabilidad de p constante en cada mano

Por lo tanto, B_t después de 100 manos puede modelarse como Binomial $X \sim \text{Binomial}(n = 100, p)$.





Escenarios

$B_t \sim \text{Binom}(n, p)$

Permite estimar porobabilidad de distintos balances finales

Riegos de bancarrota

Criterio de Kelly (probabilidad y manos se mantienen) $B_t \sim \text{Binom}(n, p)$

Determina apuesta de jugador

Aproximación se mantiene, diferente apuesta no altera probabilidades

PMF propia

Distintas magnitudes y probabilidades

Cada mano ya no varía en cuanto éxito o fracaso

Naturaleza de las probabilidades y valores dados

Actualización de balance (stack)





Escenario 1 - Big Blind Único

¿Qué nos decía el escenario?

En cada mano se apuesta exactamente un big blind:

Apuesta por mano: Exactamente un big blind

Balance inicial: 100 BBs

Probabilidad de ganar: $p = 0.5$

Probabilidad de perder: $1 - p = 0.5$

Ganancia por victoria: +1 BB

Pérdida por derrota: -1 BB

Condición de bancarrota: Balance ≤ 0

La probabilidad teórica de una **bancarrota** en este escenario es extremadamente baja. Para que ocurra una bancarrota, el jugador debe perder al menos **100 manos consecutivas** antes de ganar alguna, lo cual tiene una probabilidad de:

$$P(\text{bancarrota}) = (1 - p)^{100} = 0.5^{100} \approx 7.89 \times 10^{-31}$$

Planteamiento Teórico

Siguiendo la distribución binomial, tenemos lo siguiente:

Mean

$$\mu = n \cdot p$$

$$\mu = 100 \cdot 0.5 = 50$$

Variance

$$\sigma^2 = n \cdot p \cdot (1 - p)$$

$$\sigma^2 = 100 \cdot 0.5 \cdot (1 - 0.5) = 25$$



Escenario 1 - Big Blind Único

Implementación y Resultados

La implementación en Python simula 10,000 sesiones de 100 manos cada una. Los resultados obtenidos confirman la aproximación teórica:

Total de manos jugadas: 1,000,000

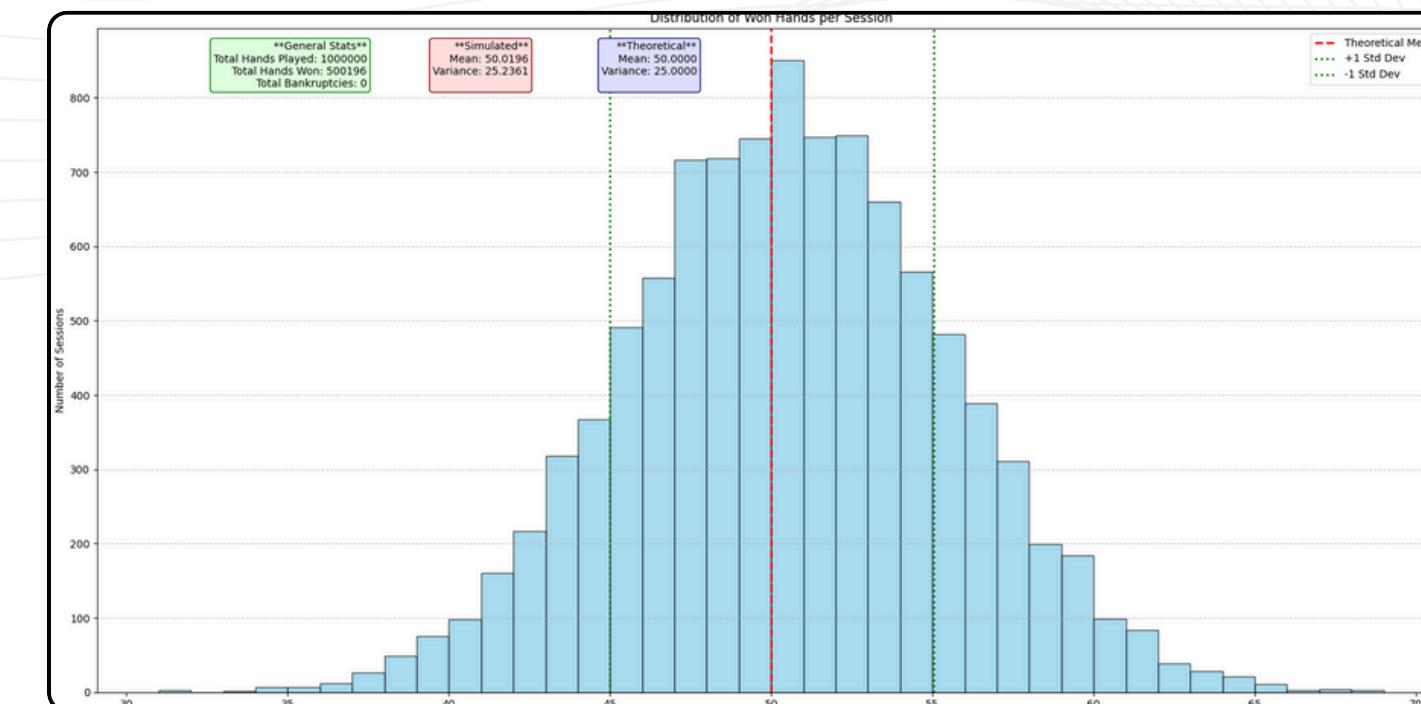
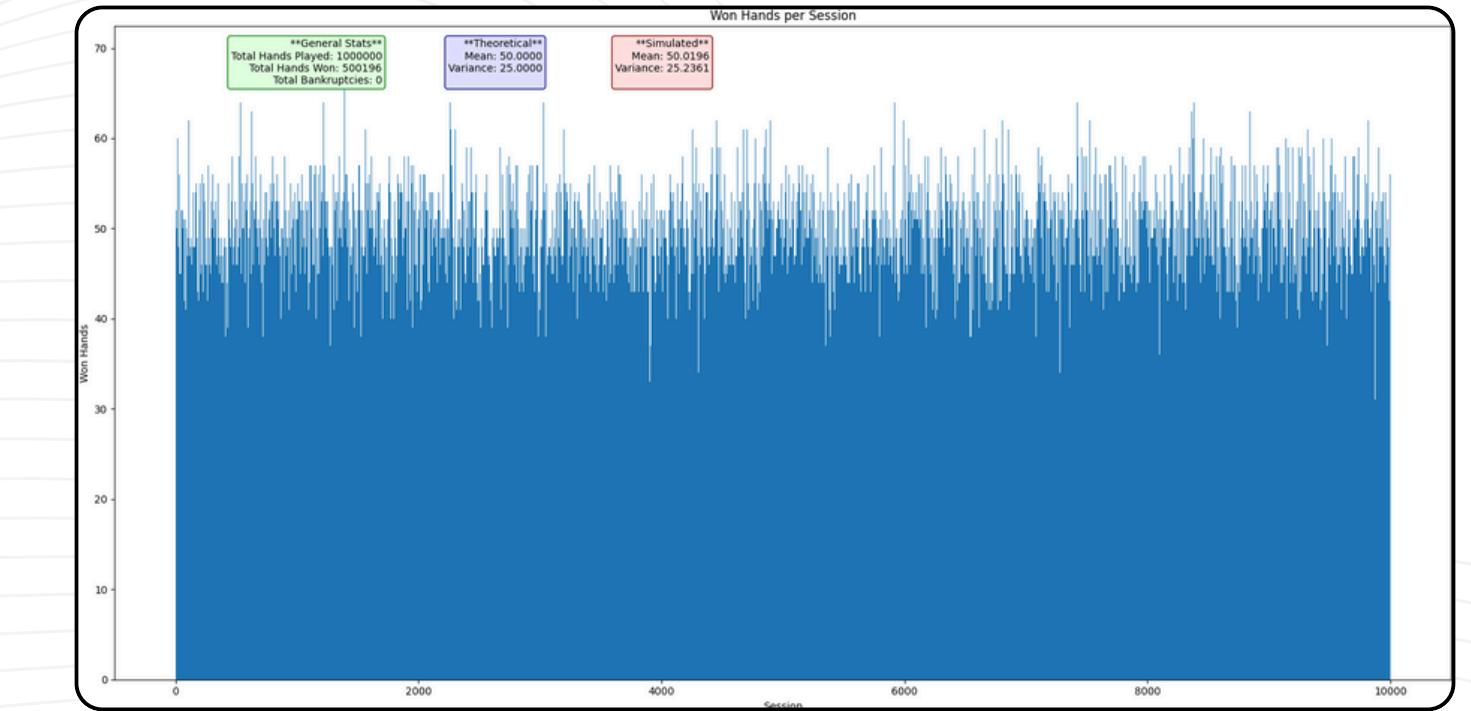
Total de manos ganadas: \approx 500,000

Media simulada: \approx 50.00 (muy cercana al valor teórico de 50)

Varianza simulada: \approx 25.00 (cercana al valor teórico de 25)

Bancarrota observadas: 0 bancarrotas observadas

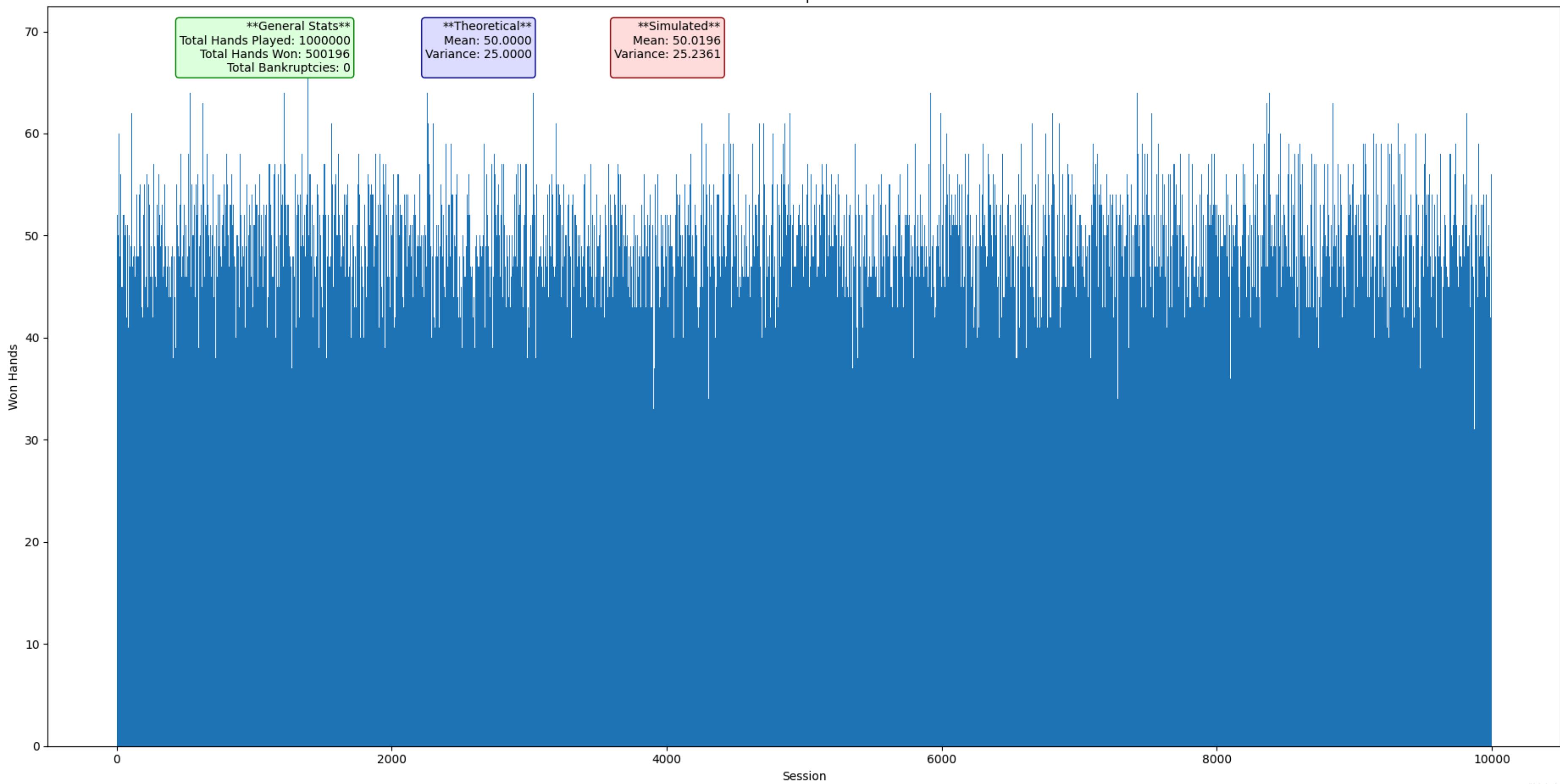
Este escenario demuestra la estabilidad de una estrategia de apuesta fija, donde la probabilidad de bancarrota es prácticamente nula y los resultados convergen consistentemente hacia los valores teóricos esperados.





Escenario 1 - Big Blind Único

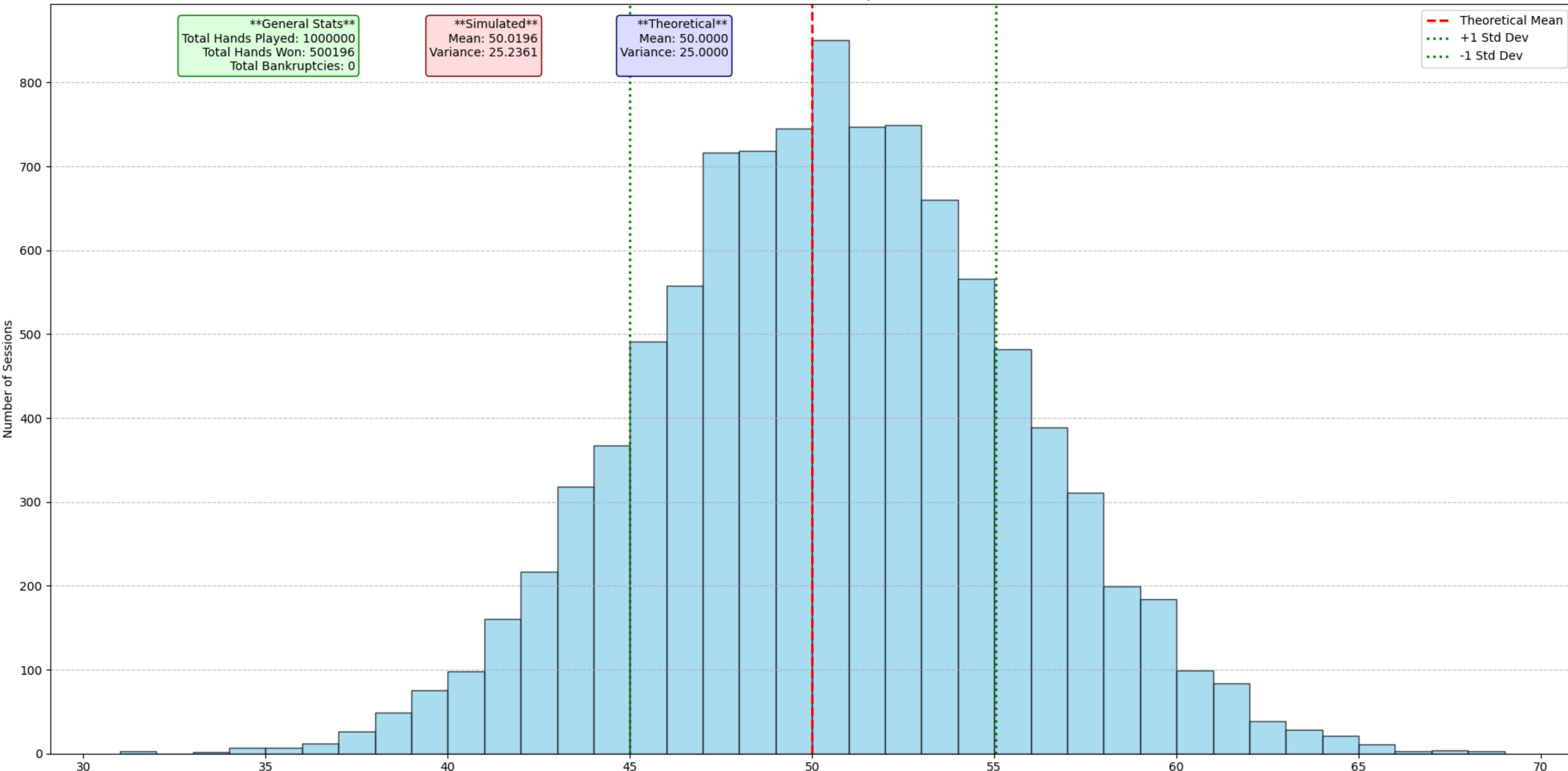
Won Hands per Session





Escenario 1 - Big Blind Único

Distribution of Won Hands per Session





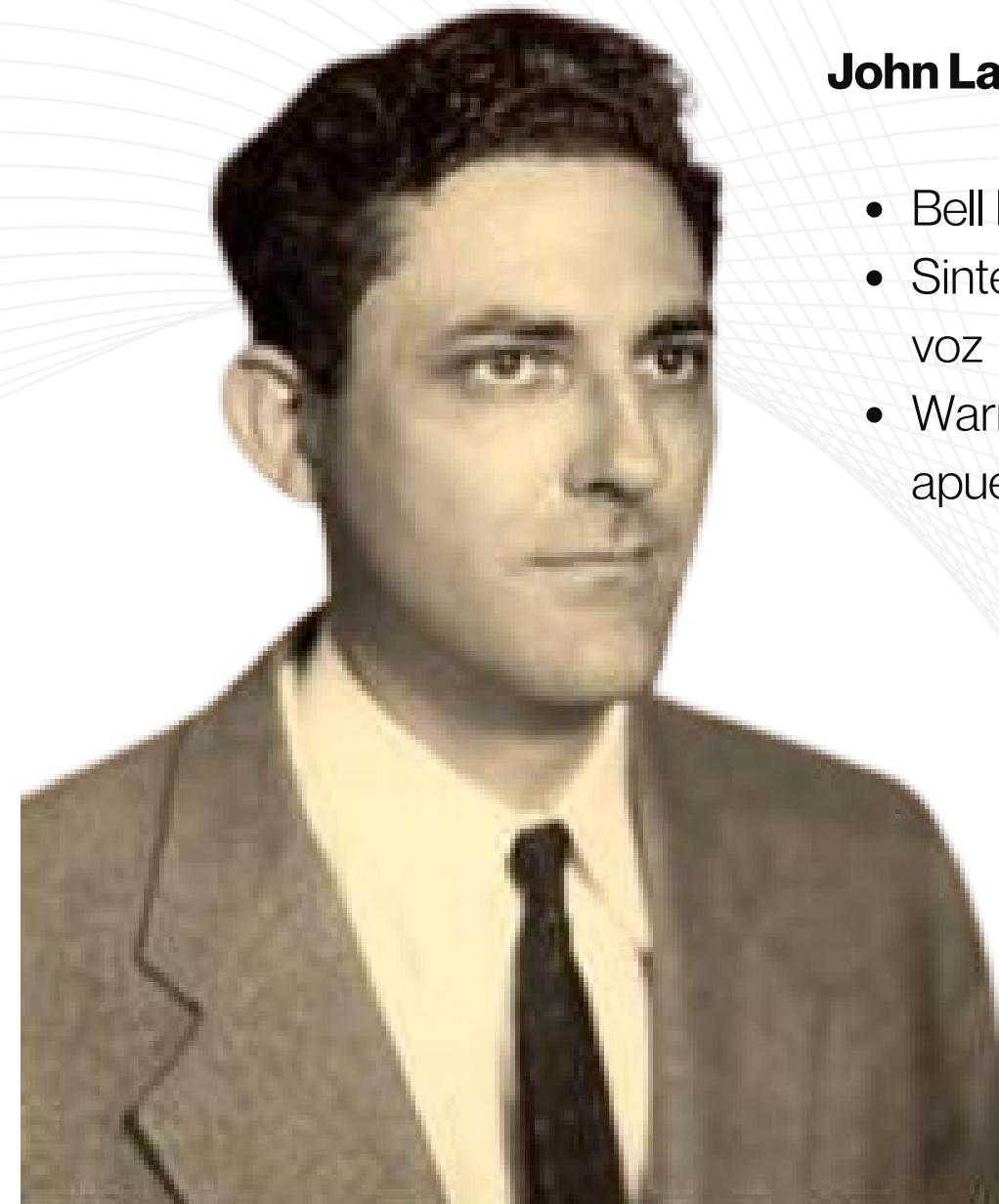
Escenario 2 - Criterio de Kelly

Qué es el criterio de Kelly?



Es una fórmula que se utiliza para estimar qué proporción de riqueza se debe arriesgar en una secuencia de apuestas para maximizar la tasa de retorno. El resultado de la fórmula es un porcentaje de Kelly, el cual nos indica la fracción de nuestro balance por apostar.

***Cada mano apuestan una fracción de su balance actual**



John Larry Kelly

- Bell Labs
- Sintetización de voz
- Warren Buffet y apuestas



Escenario 2 - Criterio de Kelly

Fórmula

$$f^* = p - \frac{q}{b} = p - \frac{1-p}{b}$$

Donde:

- f^* es la fracción de nuestro balance a apostar.
- p es la probabilidad de éxito.
- $q=1-p$ es la probabilidad de fracaso.
- b es la proporción de la apuesta ganada en caso de éxito. **E.j.** Si se apuesta \$10 en una apuesta con probabilidades de 2 a 1 (en caso de ganar se recibe \$30 y gana \$20), entonces $b = \$20/\$10 = 2.0$.

Implementación Práctica

$$f^* = p - \frac{1-p}{b}$$

$$p = 0.5, \quad b = 2$$

$$f^* = 0.5 - \frac{1-0.5}{2}$$

$$f^* = 0.25$$

Para fines de nuestro experimento, la proporción de ganancia será de **2**.

Está será la proporción a apostar en cada mano, si nuestro balance es de 100, nuestra **apuesta** será de **25**.



Escenario 2 - Criterio de Kelly

¿Qué nos decía el escenario?

Cada mano apuestan una fracción de su balance actual:

Apuesta por mano: Fracción del balance actual según Kelly
Balance inicial: 100 BBs

Probabilidad de ganar: $p = 0.5$

Probabilidad de perder: $1 - p = 0.5$

Proporción de ganancia: 2 : 1 (se duplica la apuesta al ganar)

Pérdida por derrota: Se pierde la cantidad apostada

Condición de bancarrota: Balance < 1

La probabilidad teórica de una **bancarrota** en este escenario es extremadamente baja. Para que ocurra una bancarrota, el jugador debe perder al menos **100 manos consecutivas** antes de ganar alguna, lo cual tiene una probabilidad de:

$$P(\text{bancarrota}) = (1 - p)^{100} = 0.5^{100} \approx 7.89 \times 10^{-31}$$

Análisis

A diferencia del primer escenario, la naturaleza dinámica de las apuestas hace que este sistema actúe con el objetivo de tener el retorno más óptimo utilizando el criterio de Kelly, este criterio busca:

Criterio de Kelly:

- Maximizar la tasa de retorno del capital
- Evitar la ruina completa (teóricamente)
- Balancear riesgo y retorno de manera óptima

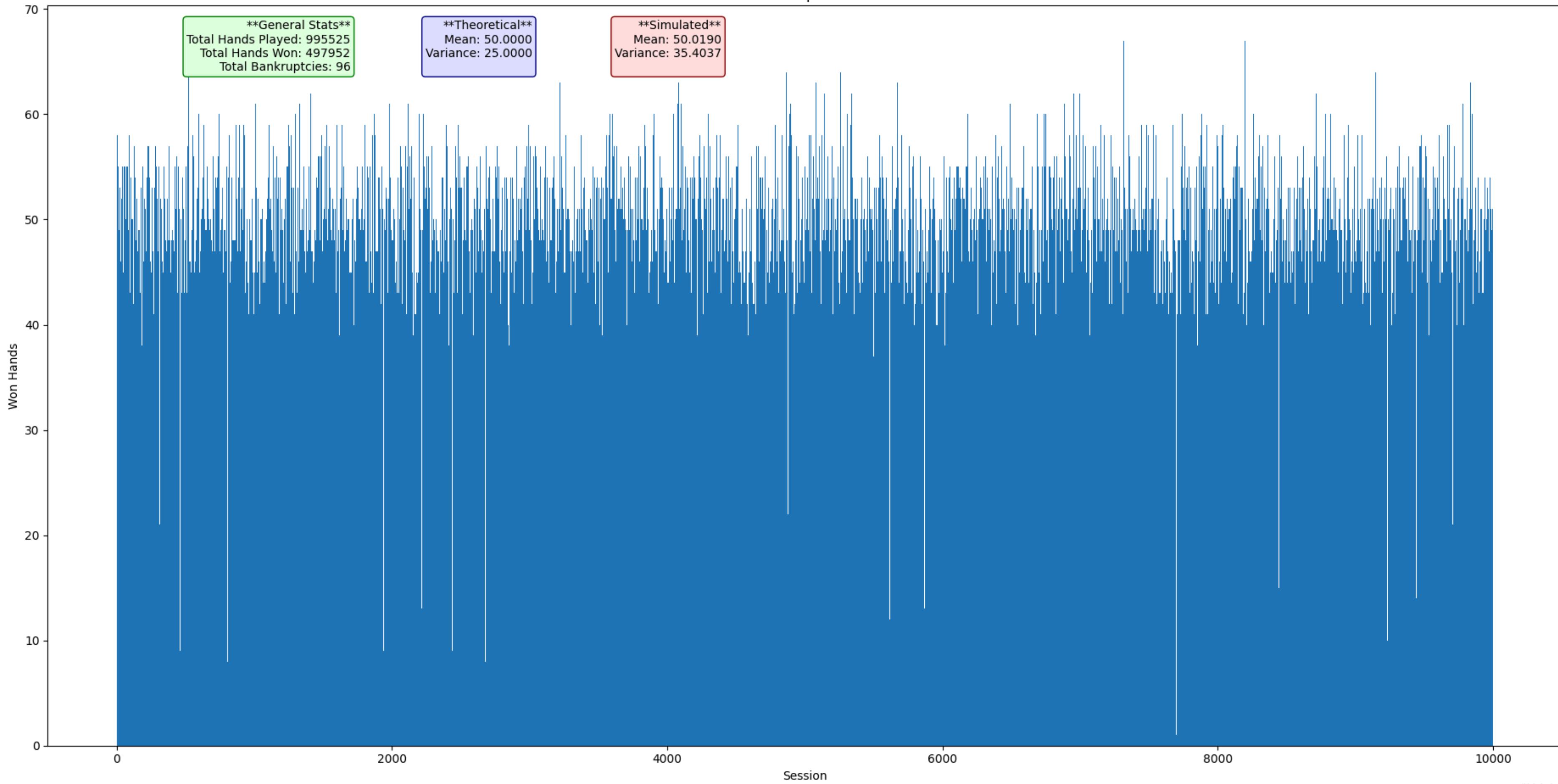
Aunque el criterio de Kelly teóricamente intenta evitar la bancarrota o al menos minimizar el riesgo de que suceda, en la práctica pueden ocurrir bancarrotas debido a:

- Condición de apuesta mínima de 1 big blind
- Secuencias adversas de resultados dada la probabilidad de éxito



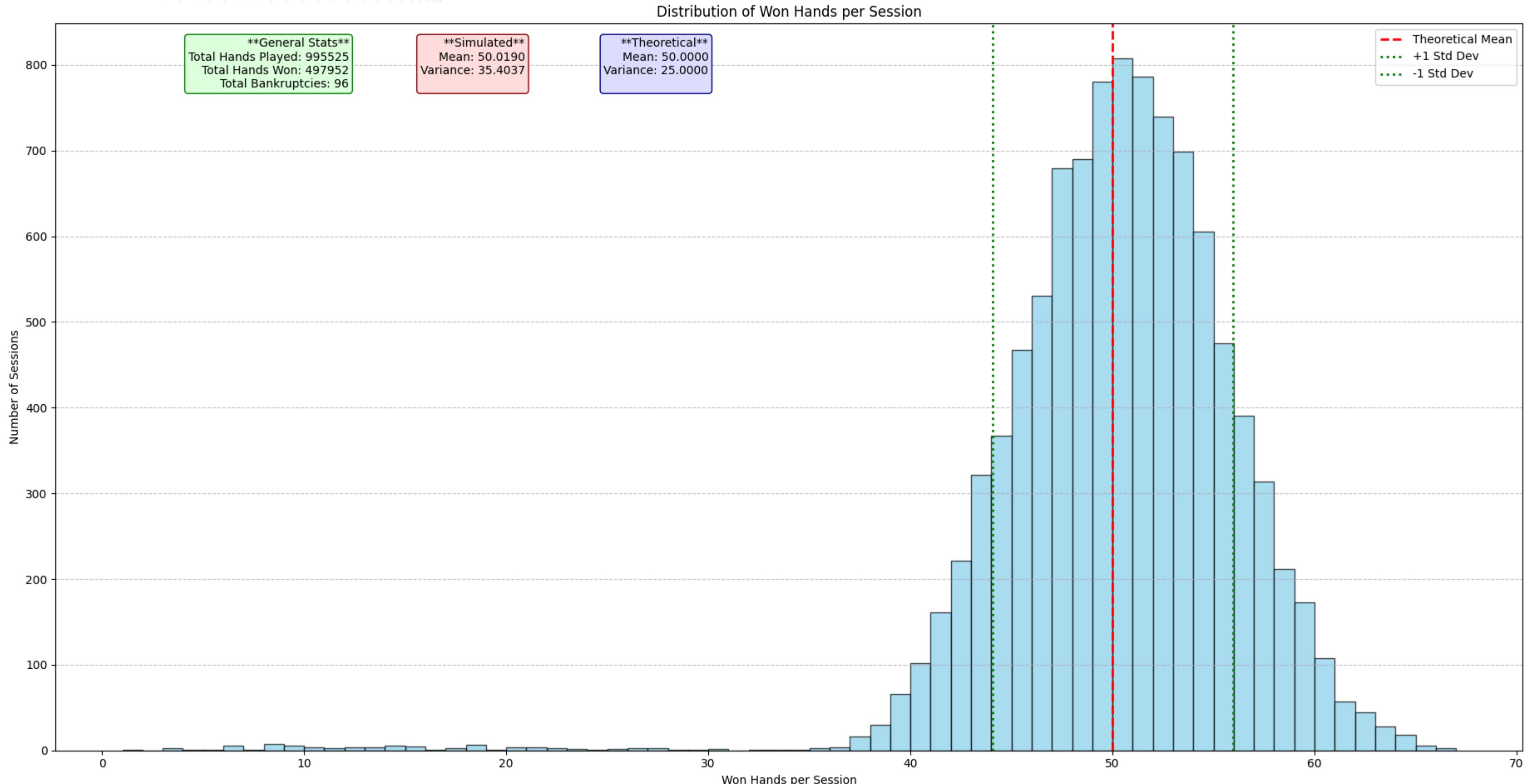
Escenario 2 - Criterio de Kelly

Won Hands per Session





Escenario 2 - Criterio de Kelly





Caso 3 - PMF propia

$$B_t = \begin{cases} 1, & \text{con probabilidad 0.70} \\ -5, & \text{con probabilidad 0.10} \\ 5, & \text{con probabilidad 0.15} \\ \text{stack}, & \text{con probabilidad 0.05} \end{cases}$$

<- Modificada del enunciado inicial

Planteamiento teórico

- Valor esperado:

$$E[X] = \sum_{i=1}^n x_i \cdot p_i \longrightarrow E[B_t] = 0.95 + 0.05\text{stack}$$

- Varianza:

$$\text{Var}[B_t] = E[B_t^2] - (E[B_t])^2 \longrightarrow \text{Var}[B_t] = 6.0475 - 0.095\text{stack} + 0.0475(\text{stack})^2$$

Planteamiento práctico

- Valor esperado:

$$E[X] = \frac{\text{cantidad de manos ganadas}}{\text{cantidad de manos jugadas}}$$

* Formúla utilizada e implementación realizada.

- Varianza:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

* Formúla utilizada

```
pl_per_hand_session = np.array([w / (w + l) for w, l in zip(wins_p1_list, wins_p2_list)])
p2_per_hand_session = np.array([w / (w + l) for w, l in zip(wins_p2_list, wins_p1_list)])
```

```
var_p1 = np.var(pl_per_hand_session, ddof=1)
var_p2 = np.var(p2_per_hand_session, ddof=1)
```

* Implementación realizada

Comparación práctico vs teórico

Práctico

- Total de manos jugadas: 200,716 (10,000 sesiones)
- Victorias Player 1: 180,680
- Victorias Player 2: 20,036
- Sesiones terminadas por *bankruptcy*: 9,970
- Proporción de *bankruptcy*: 99.70%
- Probabilidad esperada de ganar una mano:
 - Player 1: 0.9001773650331812
 - Player 2: 0.09982263496681879
- Varianza de victorias por mano:
 - Player 1: 0.005459351941939723
 - Player 2: 0.005459351941939723

Téorico

* Considerando stack = 100

$$E[B_t] = 5.95$$

$$\sigma \text{ (desv estandar)} = 32.57$$

$$\mu_{\text{simulado}} \in \mu_{\text{teórico}} \pm 1.5 \cdot \sigma$$

- $\mu_{\text{teórico}}$: valor esperado calculado teóricamente (real),
- μ_{simulado} : media obtenida a partir de la simulación,
- σ : desviación estándar de la distribución.

$$\mu_{\text{teórico}} \pm 1.5 \cdot \sigma = 5.95 \pm 32.57 \implies [-26.62, 38.52]$$

Comparación para P1

$$-26.62 \leq 0.9000064585682844 \leq 38.52$$

Como la media simulada se encuentra dentro del intervalo, se concluye que la simulación para el Jugador 1 es consistente con el valor teórico esperado.

Comparación práctico vs teórico

$$\mu_{\text{simulado}} \in \mu_{\text{teórico}} \pm 1.5 \cdot \sigma$$

- $\mu_{\text{teórico}}$: valor esperado calculado teóricamente (real),
- μ_{simulado} : media obtenida a partir de la simulación,
- σ : desviación estándar de la distribución.

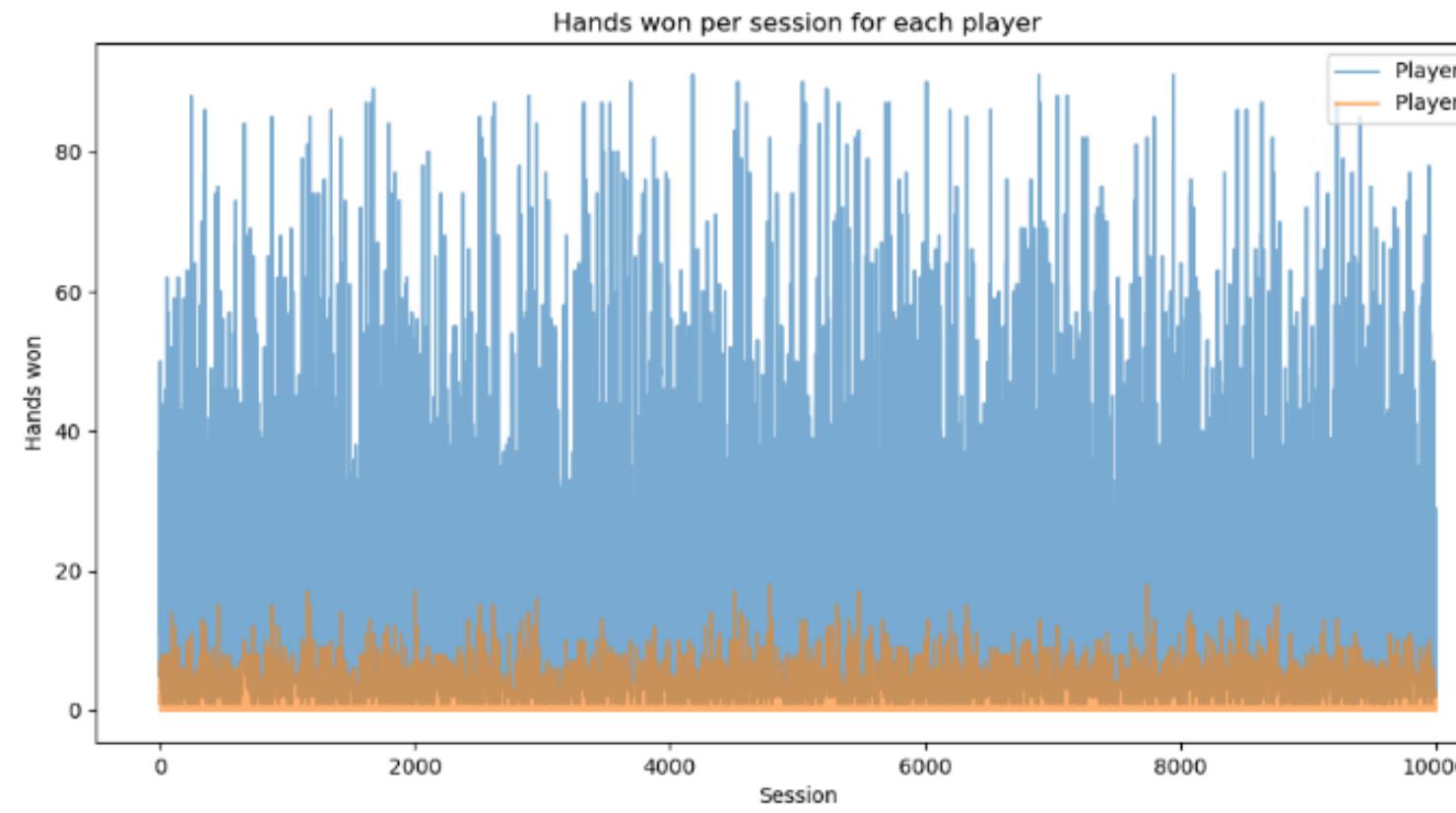


Figure 5: Manos ganadas por cada jugador

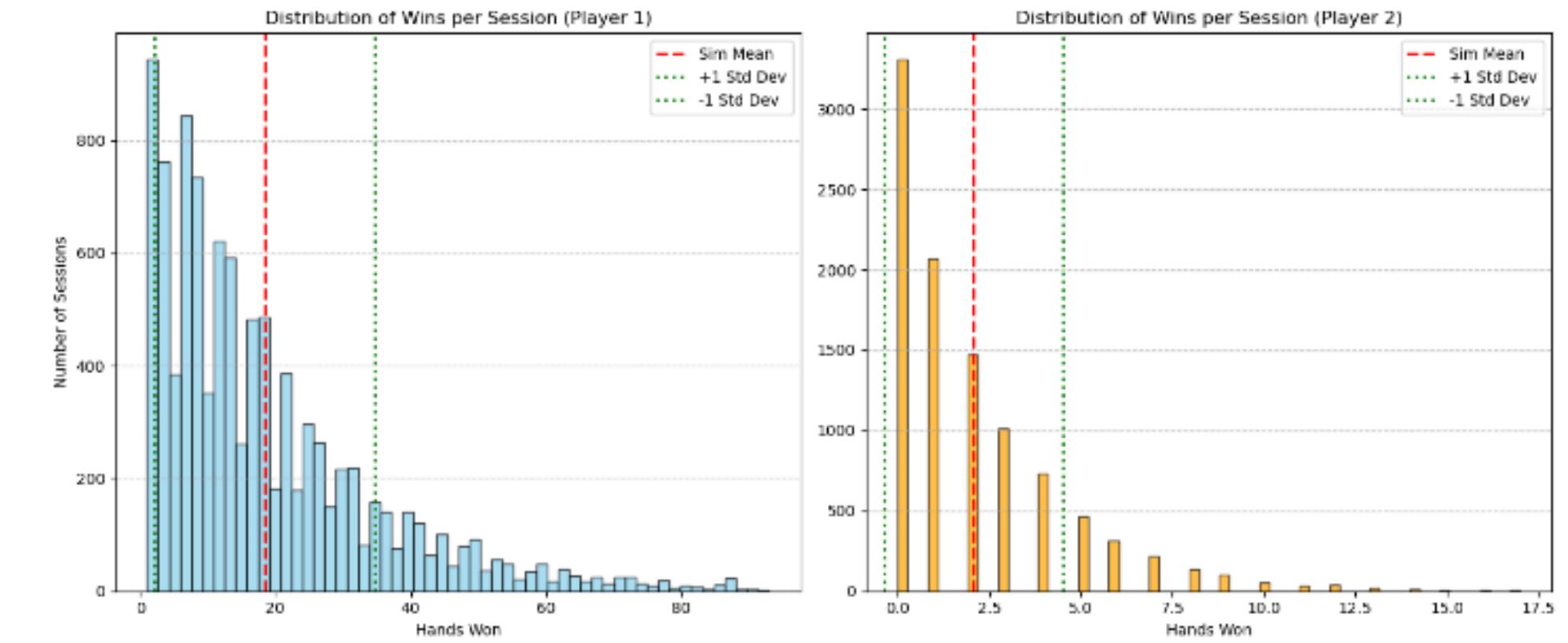
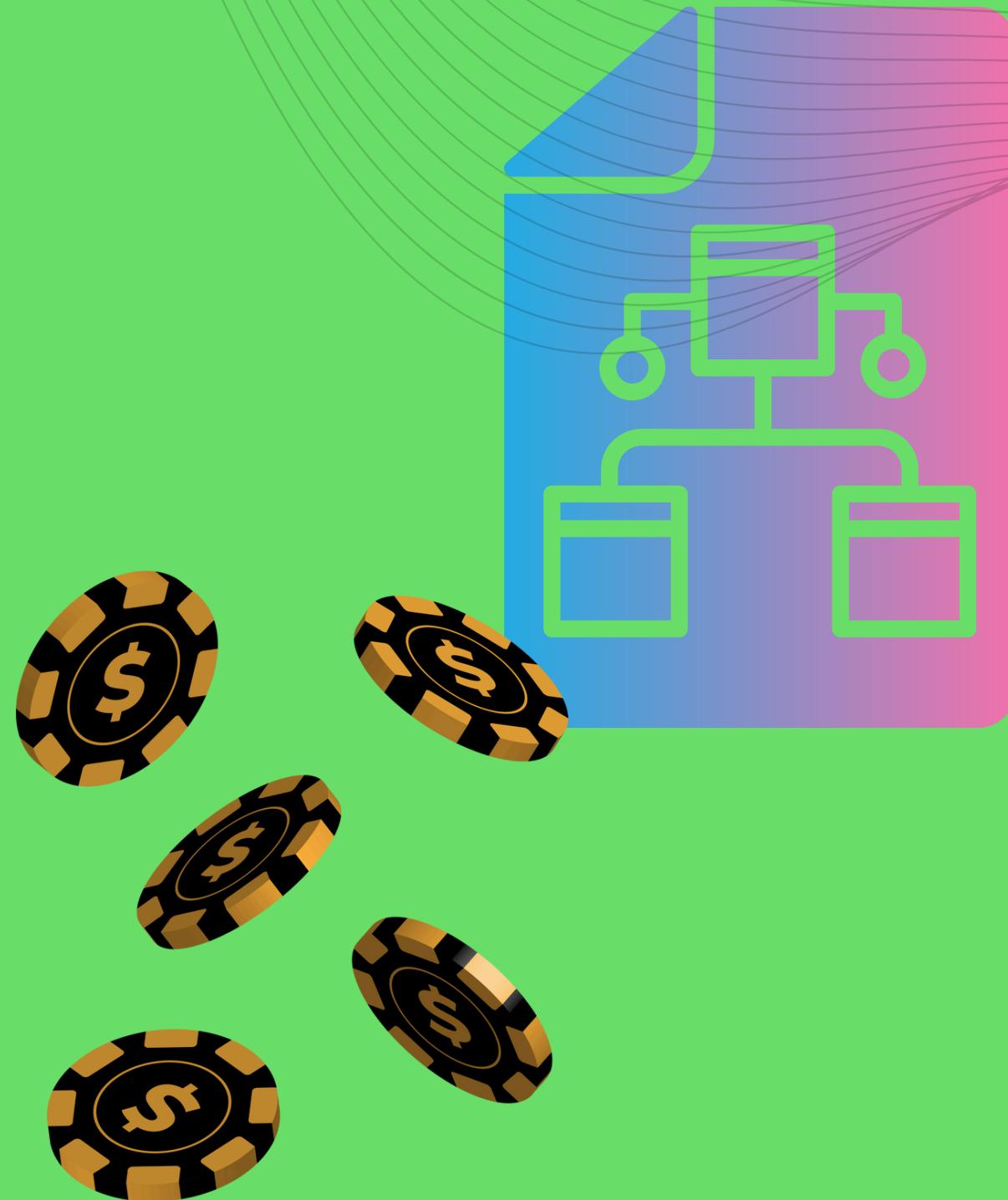


Figure 7: Histograma de las victorias por sesión de cada jugador

Poker Hand - Model

Segunda parte





Features

- Ingeniería de características = A partir de los datos base, construir nuevas variables
- Construcción de patrones lógicos

same_suit

suit_diversity

rank_diversity

max_rank_freq

has_pair

has_three_kind

has_four_kind

num_pairs

is_straight

rank_mean

rank_std

rank_range

straight_flush

full_house

is_royal_flush



same_suit

- True si todas las cartas son del mismo palo (flush).
- Detecta directamente si se tiene un color.
- Compara todos los palos y revisa si son iguales.

```
# Extraer palos y rangos por separado
suits = df[['S1', 'S2', 'S3', 'S4', 'S5']].values
ranks = df[['C1', 'C2', 'C3', 'C4', 'C5']].values
```

```
# Características de palos
df_features['same_suit'] = np.sum(suits == suits[:, 0:1], axis=1) == 5 # Flush
```



suit_diversity

- Número de palos diferentes en la mano.
- Indica si es más probable tener un flush o una mano variada.
- Cuenta los palos distintos (`len(set(...))`).

```
# Extraer palos y rangos por separado
suits = df[['S1', 'S2', 'S3', 'S4', 'S5']].values
ranks = df[['C1', 'C2', 'C3', 'C4', 'C5']].values
```

```
df_features['suit_diversity'] = np.array([len(set(row)) for row in suits]) # Diversidad de palos
```



rank_diversity

- Número de valores distintos en las cartas.
- Diferencia entre manos como [2,2,3,3,3] (poca diversidad, muchos repetidos) y [2,4,7,9,K] (muchísima diversidad).
- Cuenta los valores distintos de las cartas.

```
# Extraer palos y rangos por separado
suits = df[['S1', 'S2', 'S3', 'S4', 'S5']].values
ranks = df[['C1', 'C2', 'C3', 'C4', 'C5']].values
```

```
df_features['rank_diversity'] = np.array([len(set(row)) for row in ranks]) # Diversidad de rangos
```



max_rank_freq

- El mayor número de veces que se repite un valor.
- Identifica pares, tríos o póker.
- Ejemplo: en [K,K,K,2,3], el máximo es 3.

```
# Extraer palos y rangos por separado
suits = df[['S1', 'S2', 'S3', 'S4', 'S5']].values
ranks = df[['C1', 'C2', 'C3', 'C4', 'C5']].values
```

```
df_features['max_rank_freq'] = [max(freq.values()) for freq in rank_frequencies]
```



has_pair

- True si hay al menos un par.
- Revisa si max_rank_freq ≥ 2 .

```
# Extraer palos y rangos por separado
suits = df[['S1', 'S2', 'S3', 'S4', 'S5']].values
ranks = df[['C1', 'C2', 'C3', 'C4', 'C5']].values
```

```
df_features['has_pair'] = df_features['max_rank_freq'] >= 2
```



has_three_kind

- True si hay al menos un trío.
- Revisa si max_rank_freq ≥ 3

```
# Extraer palos y rangos por separado
suits = df[['S1', 'S2', 'S3', 'S4', 'S5']].values
ranks = df[['C1', 'C2', 'C3', 'C4', 'C5']].values
```

```
df_features['has_three_kind'] = df_features['max_rank_freq'] >= 3
```



has_four_kind

- True si hay un póker.
- Revisa si max_rank_freq >= 4

```
# Extraer palos y rangos por separado
suits = df[['S1', 'S2', 'S3', 'S4', 'S5']].values
ranks = df[['C1', 'C2', 'C3', 'C4', 'C5']].values
```

```
df_features['has_four_kind'] = df_features['max_rank_freq'] >= 4
```



num_pairs

- Cantidad de pares exactos.
 - Ejemplo: [K,K,5,5,2] tiene 2 pares.
 - Diferencia entre una mano con un par y dos pares.

```
# Extraer palos y rangos por separado
suits = df[['S1', 'S2', 'S3', 'S4', 'S5']].values
ranks = df[['C1', 'C2', 'C3', 'C4', 'C5']].values
```



is_straight

- True si los valores son consecutivos (escalera).
- Casos especiales:
- A-2-3-4-5 (escalera baja).
- 10-J-Q-K-A (escalera alta).
- Ordena los valores y revisa diferencias consecutivas.

```
# Extraer palos y rangos por separado
suits = df[['S1', 'S2', 'S3', 'S4', 'S5']].values
ranks = df[['C1', 'C2', 'C3', 'C4', 'C5']].values
```

```
# Detectar straight (escalera)
def is_straight(ranks_row):
    unique_ranks = sorted(set(ranks_row))
    if len(unique_ranks) != 5:
        return False
    # Verificar secuencia consecutiva
    for i in range(1, len(unique_ranks)):
        if unique_ranks[i] - unique_ranks[i-1] != 1:
            # Caso especial: A, 2, 3, 4, 5 (Ace low straight)
            if set(unique_ranks) == {1, 2, 3, 4, 5}:
                return True
            # Caso especial: 10, J, Q, K, A (Ace high straight)
            if set(unique_ranks) == {1, 10, 11, 12, 13}:
                return True
    return False
    return True

df_features['is_straight'] = [is_straight(row) for row in ranks]
```



rank_mean

- Promedio de los valores de las cartas.
- Mide la “altura” promedio de la mano.
- Ejemplo: [2,3,4,5,6] tiene media más baja que [10,J,Q,K,A].

```
# Extraer palos y rangos por separado
suits = df[['S1', 'S2', 'S3', 'S4', 'S5']].values
ranks = df[['C1', 'C2', 'C3', 'C4', 'C5']].values
```

```
"Calcular el promedio de los rangos"
df_features['rank_mean'] = np.mean(ranks, axis=1)
```



rank_std

- Desviación estándar de los valores.
- Indica si los valores son cercanos (baja dispersión → como en escalera) o muy variados.

```
# Extraer palos y rangos por separado
suits = df[['S1', 'S2', 'S3', 'S4', 'S5']].values
ranks = df[['C1', 'C2', 'C3', 'C4', 'C5']].values
```

```
df_features['rank_std'] = np.std(ranks, axis=1)
```



rank_range

- Diferencia entre la carta más alta y la más baja.
- Otro indicador de posibles escaleras o qué tan “alta” es la mano.

```
# Extraer palos y rangos por separado
suits = df[['S1', 'S2', 'S3', 'S4', 'S5']].values
ranks = df[['C1', 'C2', 'C3', 'C4', 'C5']].values
```

```
df_features['rank_range'] = np.max(ranks, axis=1) - np.min(ranks, axis=1)
```



straight_flush

- True si hay escalera y todas son del mismo palo.
- Detectar straight flush, una de las mejores manos.

```
# Extraer palos y rangos por separado
suits = df[['S1', 'S2', 'S3', 'S4', 'S5']].values
ranks = df[['C1', 'C2', 'C3', 'C4', 'C5']].values
```

```
df_features['straight_flush'] = df_features['is_straight'] & df_features['same_suit']
```



full_house

- True si hay un trío y un par.
- Ejemplo: [K,K,3,3,3].
- Combina has_three_kind y num_pairs.

```
# Extraer palos y rangos por separado
suits = df[['S1', 'S2', 'S3', 'S4', 'S5']].values
ranks = df[['C1', 'C2', 'C3', 'C4', 'C5']].values
```

```
df_features['full_house'] = (df_features['has_three_kind']) & (df_features['num_pairs'] >= 1)
```



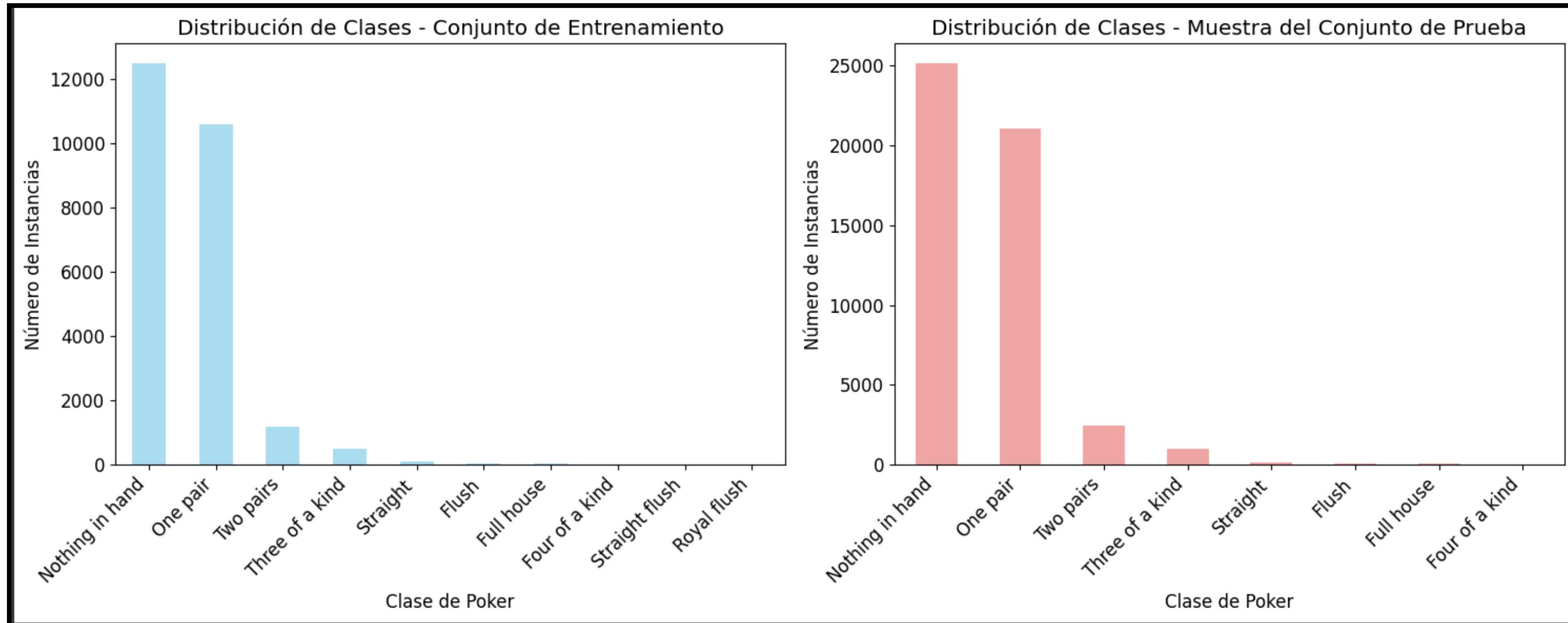
is_royal_flush

- True si hay un straight flush de [10, J, Q, K, A].
 - Detectar la mejor mano posible.
 - Revisa si todos los palos son iguales y los valores corresponden a {10,11,12,13,1}.

```
# Extraer palos y rangos por separado
suits = df[['S1', 'S2', 'S3', 'S4', 'S5']].values
ranks = df[['C1', 'C2', 'C3', 'C4', 'C5']].values
```

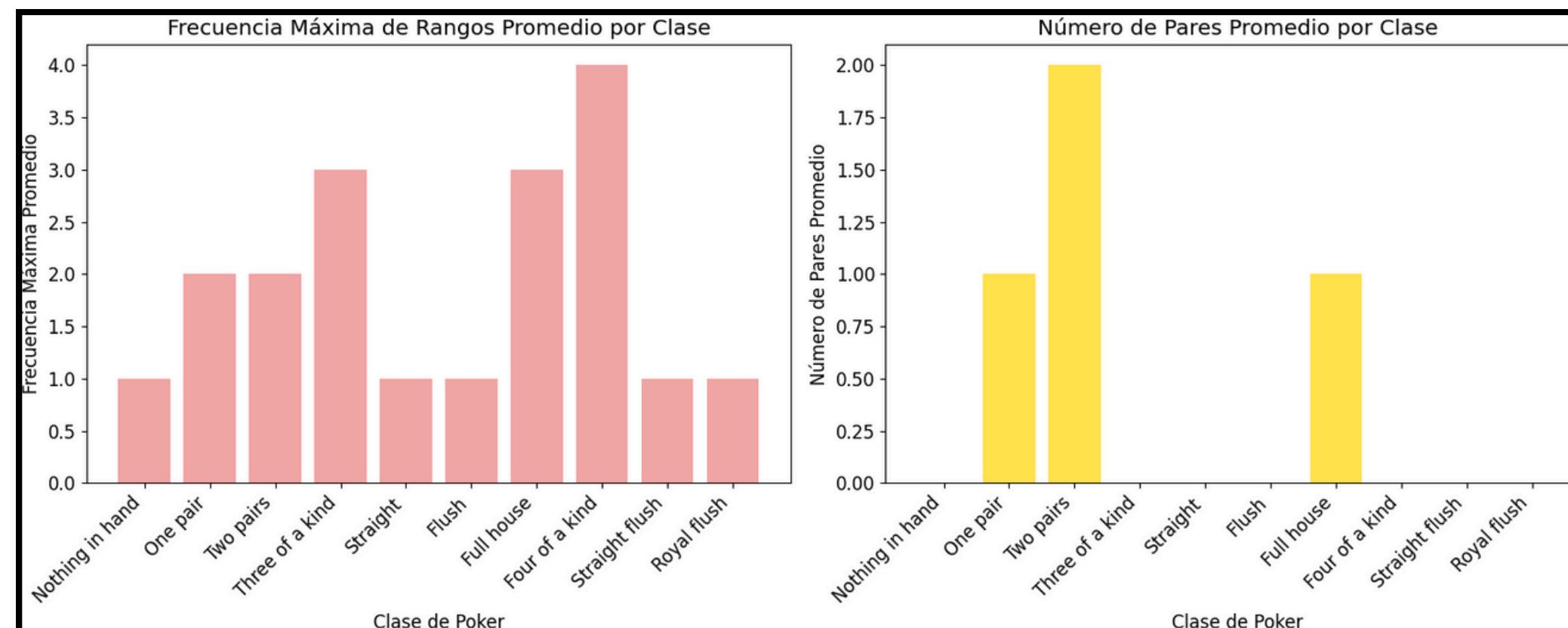
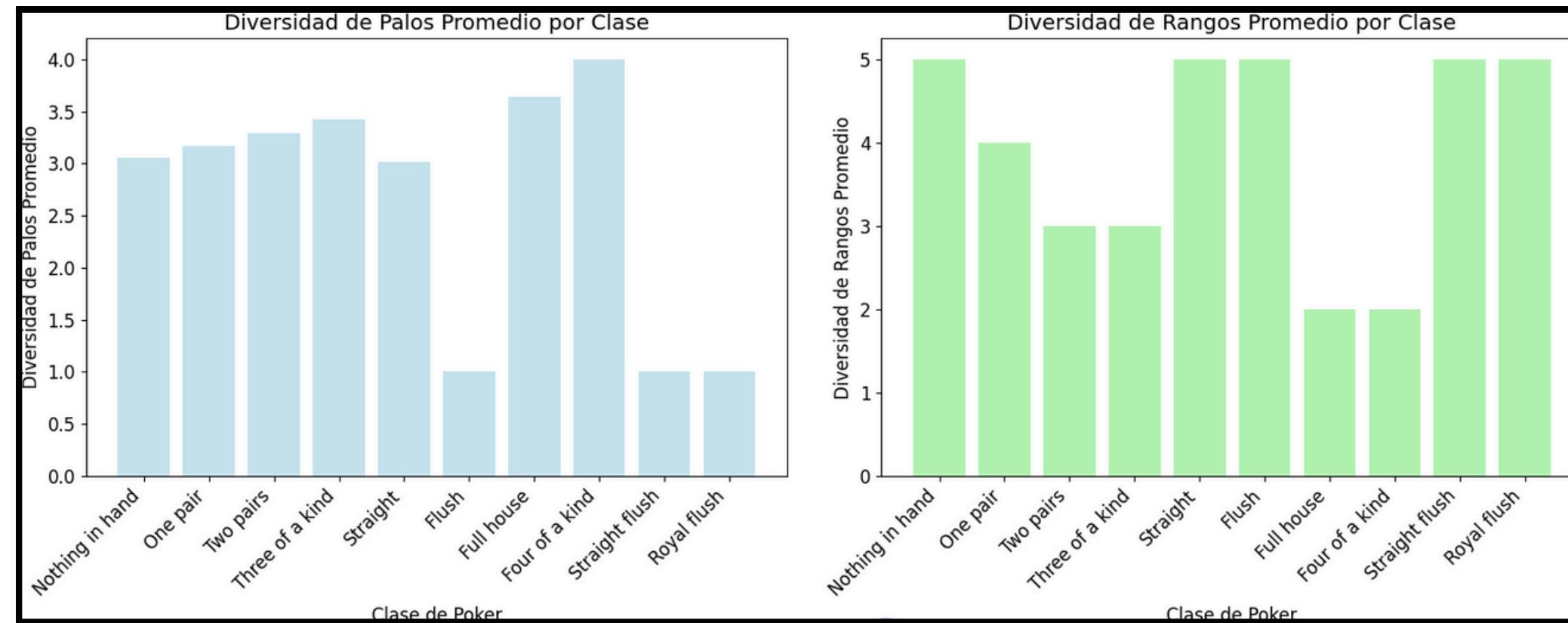


DISTRIBUCIONES DE LAS CLASES





DISTRIBUCIONES DE LAS CLASES





Elaboración de Modelos

LDA

QDA

Para la implementación y elaboración de los modelos utilizamos Linear Discriminant Analysis y Quadratic Discriminant Analysis, utilizando las características proveidas.

Accuracy = Dice qué porcentaje total de manos se clasificaron bien.

Error rate = El porcentaje de predicciones incorrectas.

Precision = De todas las veces que el modelo predijo una clase, qué fracción fue correcta.

Recall = De todos los casos reales de una clase, qué fracción el modelo logró identificar. Importante para este problema porque queremos que no se le “escapen” manos valiosas.

F1 = Combinación entre precision y recall. No sirve de nada predecir bien pocas manos si dejamos escapar muchas.

Evaluando: Linear Discriminant Analysis					

Accuracy: 0.6010					
Error Rate: 0.3990					
Precision promedio ponderada: 0.5861					
Recall promedio ponderado: 0.6010					
F1-Score promedio ponderado: 0.5806					
Reporte detallado por clase:					
	precision	recall	f1-score	support	
Nothing in hand	0.65	0.71	0.68	501209	
One pair	0.54	0.57	0.56	422498	
Two pairs	0.64	0.05	0.10	47622	
Three of a kind	0.00	0.00	0.00	21121	
Straight	0.00	0.00	0.00	3885	
Flush	0.99	0.49	0.66	1996	
Full house	0.00	0.00	0.00	1424	
Four of a kind	0.00	0.00	0.00	230	
Straight flush	0.23	0.50	0.32	12	
Royal flush	0.00	0.00	0.00	3	
accuracy				0.60	1000000
macro avg	0.31	0.23	0.23	1000000	
weighted avg	0.59	0.60	0.58	1000000	

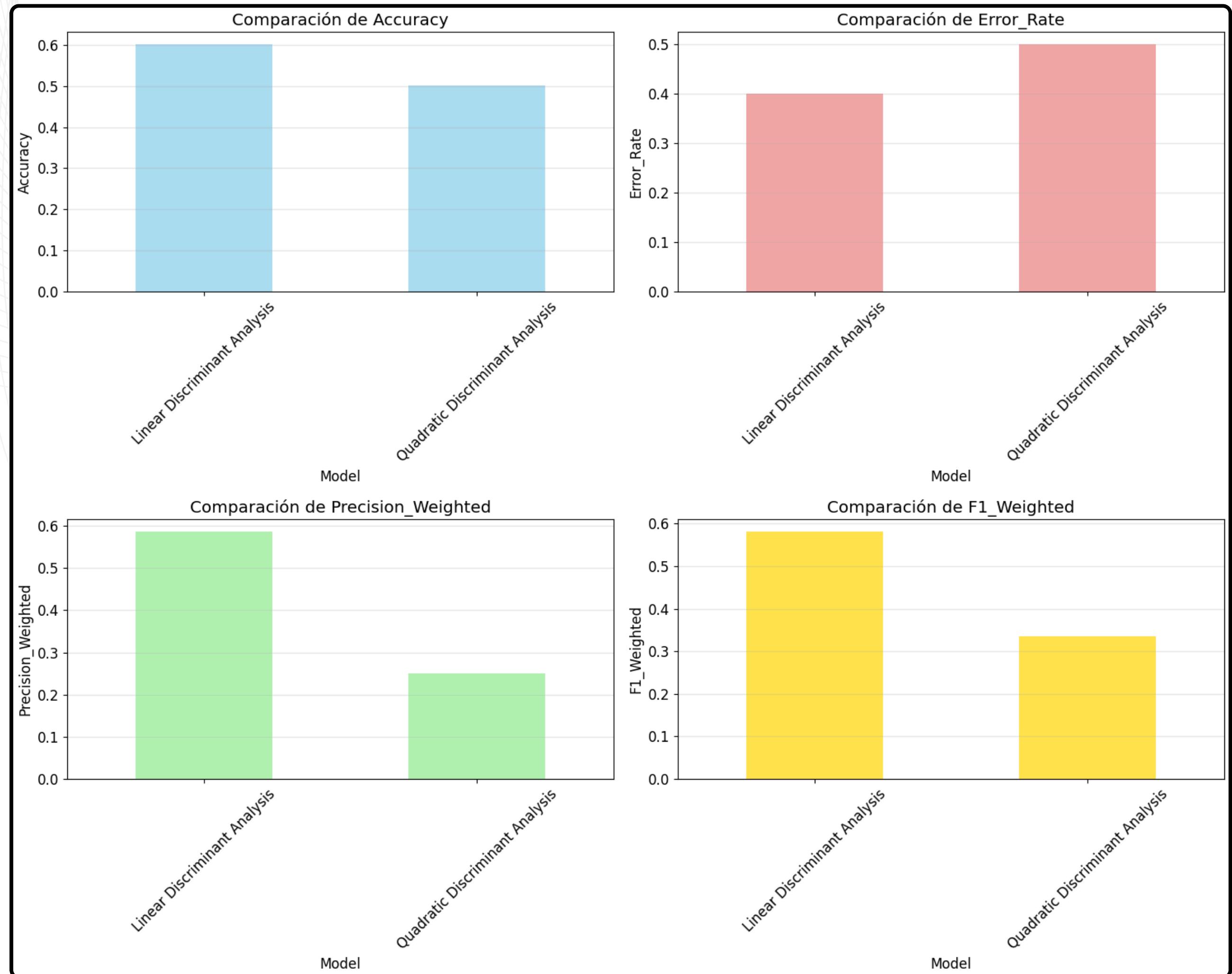
Evaluando: Quadratic Discriminant Analysis					

Accuracy: 0.5012					
Error Rate: 0.4988					
Precision promedio ponderada: 0.2512					
Recall promedio ponderado: 0.5012					
F1-Score promedio ponderado: 0.3347					

Reporte detallado por clase:					
	precision	recall	f1-score	support	
Nothing in hand	0.50	1.00	0.67	501209	
One pair	0.00	0.00	0.00	422498	
Two pairs	0.00	0.00	0.00	47622	
Three of a kind	0.00	0.00	0.00	21121	
Straight	0.00	0.00	0.00	3885	
Flush	0.00	0.00	0.00	1996	
Full house	0.00	0.00	0.00	1424	
Four of a kind	0.00	0.00	0.00	230	
Straight flush	0.00	0.00	0.00	12	
Royal flush	0.00	0.00	0.00	3	
accuracy				0.50	1000000
macro avg	0.05	0.10	0.07	1000000	
weighted avg	0.25	0.50	0.33	1000000	



Comparación de los Modelos





Evaluación de Modelos

LDA

QDA

El conjunto de datos de manos de poker presenta un gran desbalance de clases, con "Nothing in hand" representando ~50% de las instancias y clases raras como "Royal flush" con muy pocas muestras.

Linear Discriminant Analysis (LDA) - Mejor desempeño:

- Accuracy: 60.10%
- F1-Score ponderado: 58.06%
- Logra clasificar razonablemente las clases más frecuentes
- Detecta algunas clases minoritarias (Flush: 49% recall, Straight flush: 50% recall)

Quadratic Discriminant Analysis (QDA):

- Accuracy: 50.12%
- F1-Score ponderado: 33.47%
- Se comporta como un clasificador que predice solo "Nothing in hand"
- Falla completamente en clases minoritarias

1. Distribución de clases en los datos

- "Nothing in hand" = casi la mitad del dataset.
- "Royal flush" y "Straight flush" = menos del 1%.
- Si un modelo no maneja el desbalance, tenderá a favorecer las clases frecuentes, porque con eso ya gana más accuracy.

2. Métricas obtenidas

- La accuracy y el F1 ponderado se ven moderados (~58% para LDA), pero al ser ponderados están dominados por las clases grandes.
- Cuando miramos las métricas por clase (recall y precisión clase por clase), se ve que:
 - Clases frecuentes como Nothing in hand y One pair tienen recall aceptable.
 - Clases raras (Straight flush, Royal flush) casi siempre son clasificadas como clases más comunes.
- Eso significa que el modelo ignora las raras.



Evaluación de Modelos

LDA

QDA

El conjunto de datos de manos de poker presenta un gran desbalance de clases, con "Nothing in hand" representando ~50% de las instancias y clases raras como "Royal flush" con muy pocas muestras.

Linear Discriminant Analysis (LDA) - Mejor desempeño:

- Accuracy: 60.10%
- F1-Score ponderado: 58.06%
- Logra clasificar razonablemente las clases más frecuentes
- Detecta algunas clases minoritarias (Flush: 49% recall, Straight flush: 50% recall)

Quadratic Discriminant Analysis (QDA):

- Accuracy: 50.12%
- F1-Score ponderado: 33.47%
- Se comporta como un clasificador que predice solo "Nothing in hand"
- Falla completamente en clases minoritarias

1. Distribución de clases en los datos

- "Nothing in hand" = casi la mitad del dataset.
- "Royal flush" y "Straight flush" = menos del 1%.
- Si un modelo no maneja el desbalance, tenderá a favorecer las clases frecuentes.

2. Métricas obtenidas

- La accuracy y el F1 ponderado se ven moderados (~58% para LDA), pero al ser ponderados están dominados por las clases grandes.
- Cuando miramos las métricas por clase (recall y precisión clase por clase), se ve que:
 - Clases frecuentes como Nothing in hand y One pair tienen recall aceptable.
 - Clases raras (Straight flush, Royal flush) casi siempre son clasificadas como clases más comunes.
- Eso significa que el modelo ignora las raras.



Salford & Co.

Thank You!



Website

www.reallygreatsite.com

● Presentation 2030