Proyecto de Análisis de Datos

UCI Poker Hand — EDA y Modelado

Autores: *Carlos Solares*

Fecha: *30/09/2025*

Objetivos

- Analizar el dataset **Poker Hand** de UCI.
- Realizar **EDA** para entender variables y clases.
- **Construir nuevas características** que expliquen la etiqueta (mano de póker).
- Entrenar y comparar **dos modelos**: Regresión Logística y Random Forest.
- Evaluar con métricas adecuadas (Accuracy y **Macro-F1**) por el fuerte desbalance.

Definiciones y variables del dataset

- **R1..R5**: Rangos de las 5 cartas en la mano
  - Valores 1–13 → A=1, J=11, Q=12, K=13
- **S1..S5**: Suits de las 5 cartas
  - Valores 1–4 (cada número representa un suit distinto)

Definiciones y variables del dataset

- **Etiqueta (Clase 0–9): tipo de mano de póker**
  - 0 = Nada en especial
  - 1 = One Pair
  - 2 = Two Pairs
  - 3 = Three of a Kind
  - 4 = Straight
  - 5 = Flush
  - 6 = Full House
  - 7 = Four of a Kind
  - 8 = Straight Flush
  - 9 = Royal Flush (Straight Flush al As)

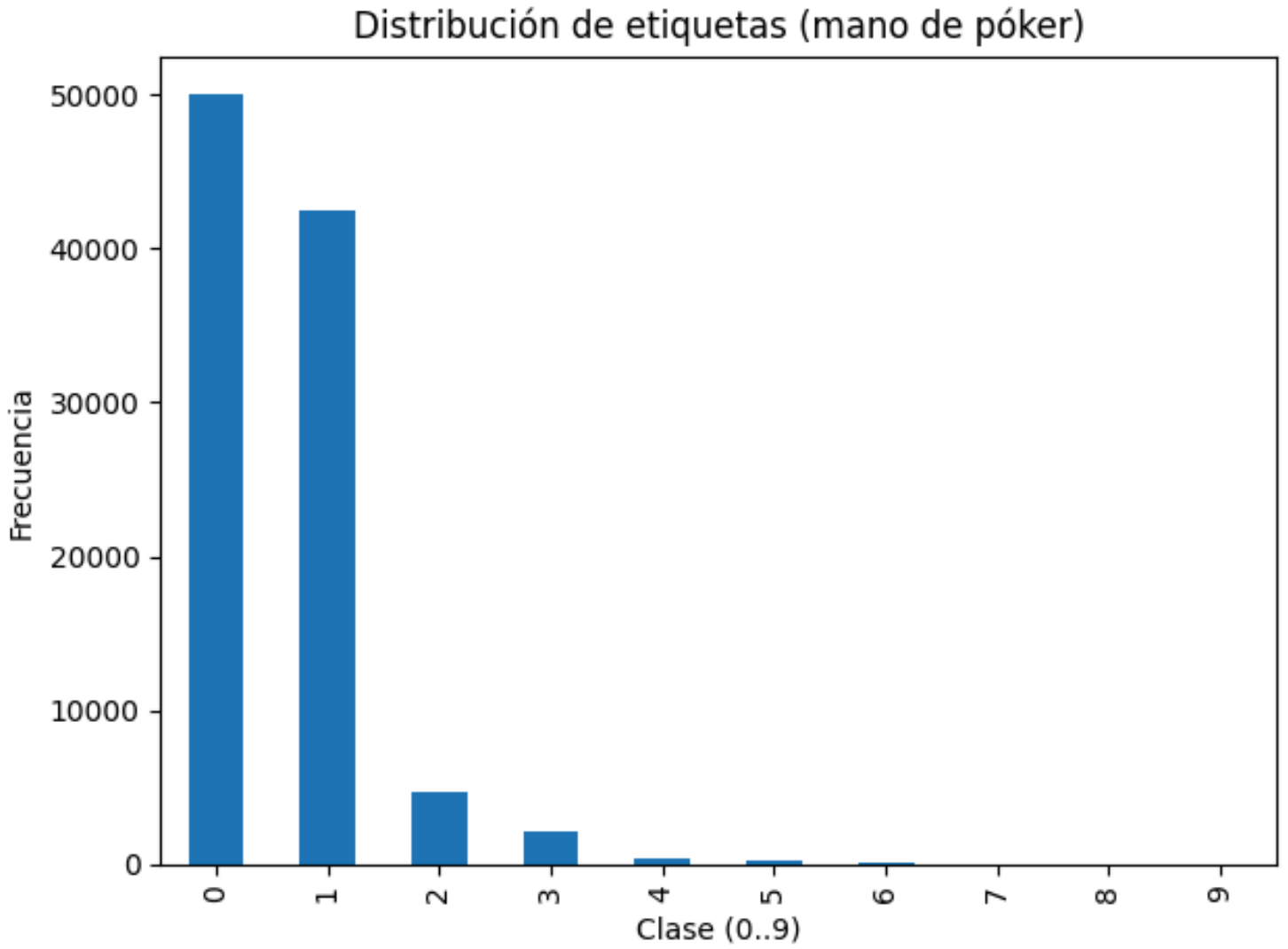
Definiciones y variables del dataset

- **Flush** = todas las cartas del mismo suit
- **Straight** = cartas consecutivas en rango
- **Macro-F1** = métrica que promedia el F1-score por clase, útil en datasets desbalanceados
- **Holdout 20%** = Separamos un **20% de los datos para prueba final**, no usados en el entrenamiento ni en cross-validation.
    - Sirve para evaluar la **generalización real del modelo**.
    - Las matrices de confusión están basadas en este conjunto.

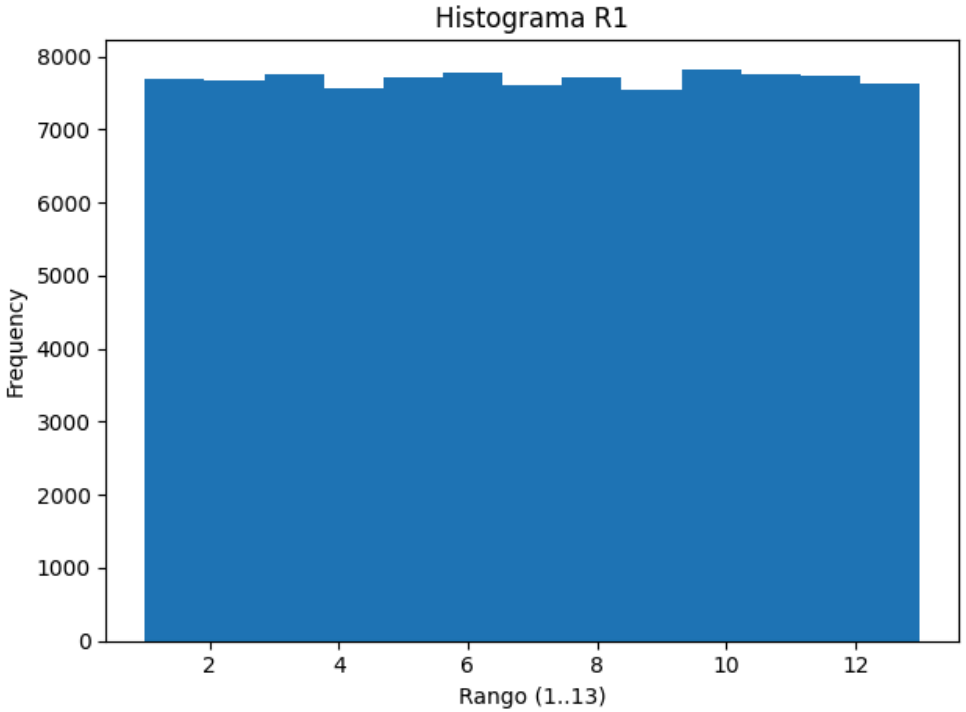
Flujo del script (`poker_analysis.py`)

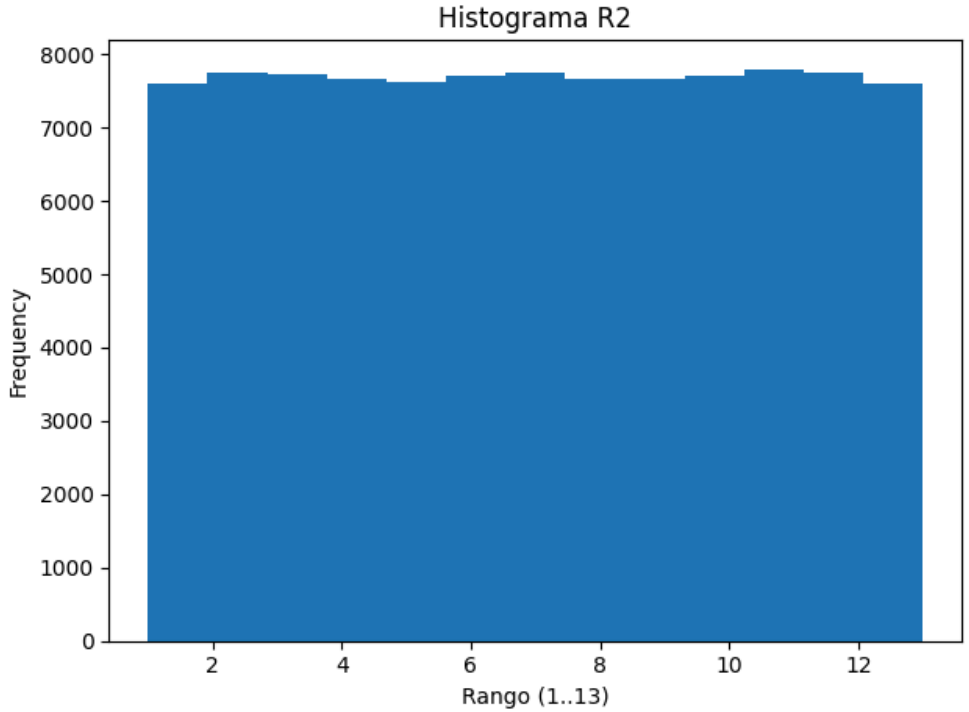
1. **Carga de datos** (train + test de UCI).
2. **EDA**: distribución de etiquetas y variables originales (R1..R5, S1..S5).
3. **Ingeniería de características**: flush, straight, pares, trío, póker, etc.
4. **Modelos**:
    - Regresión Logística (multiclase, balanced).
    - Random Forest (balanced_subsample).
5. **Evaluación**:
    - Stratified K-Fold, Accuracy y Macro-F1.
    - Holdout 20% para matrices de confusión.
6. **Artefactos** en `outputs/` : figuras, reportes y CSVs.

Distribución de etiquetas (y)

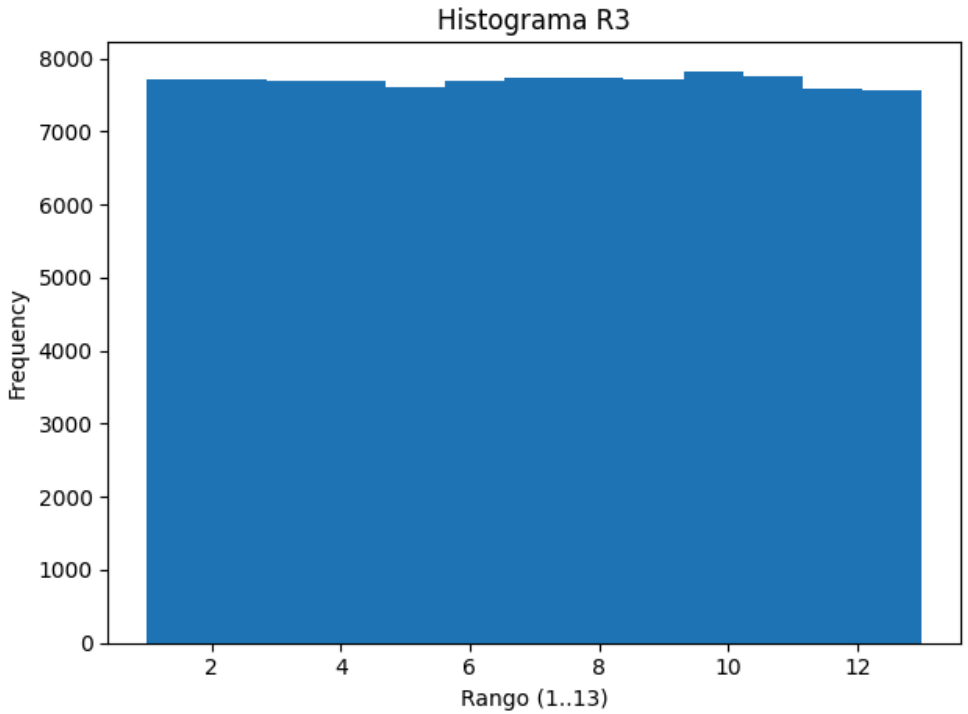


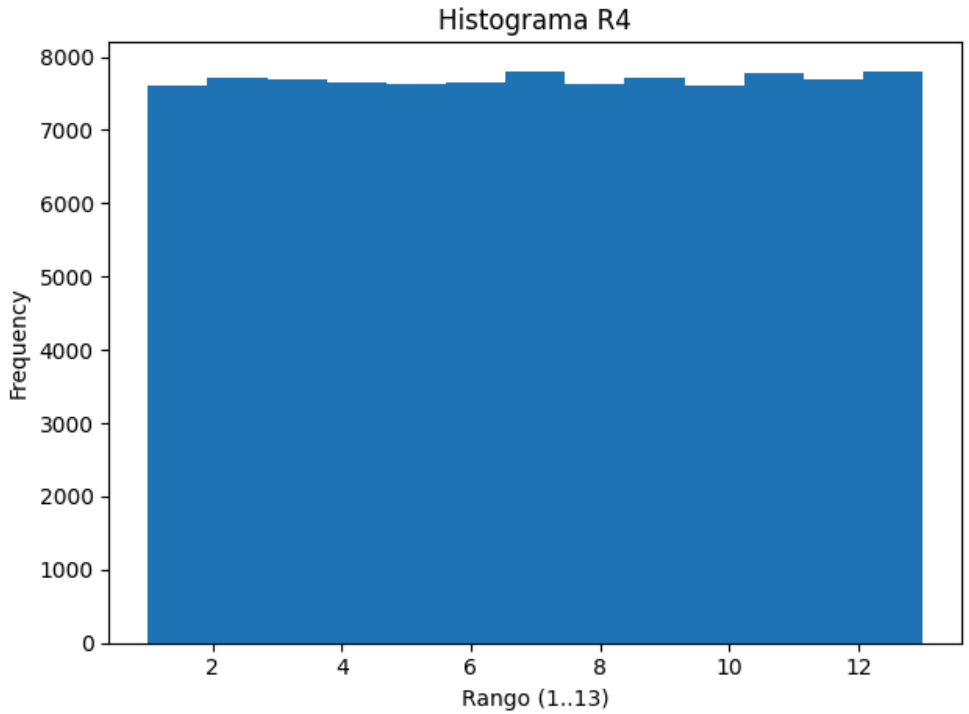
Histogramas de rangos (R1..R2)



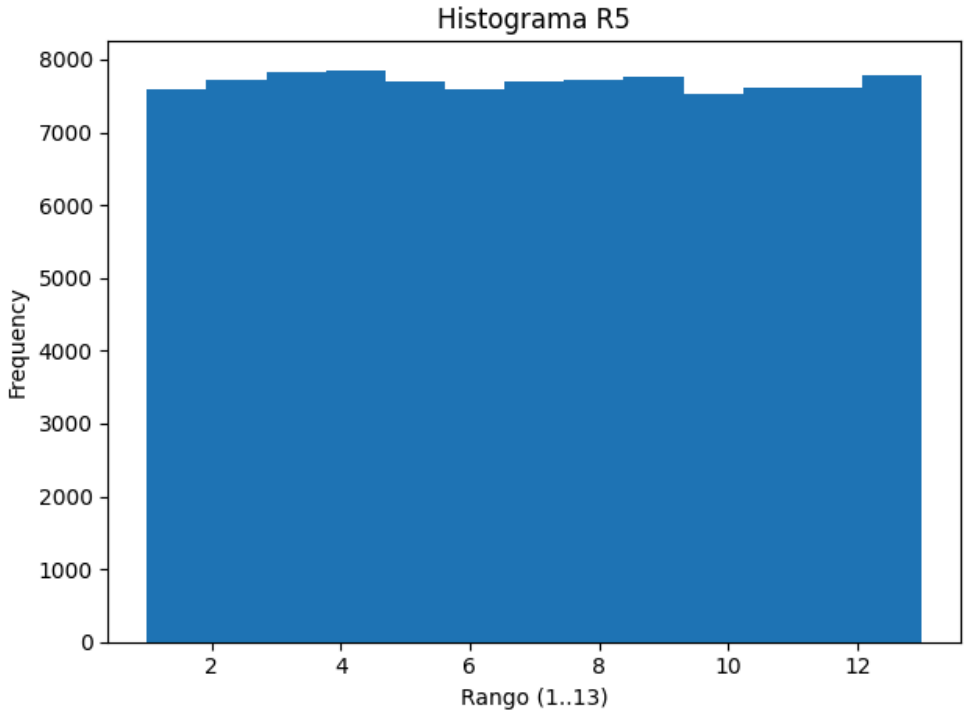


Histogramas de rangos (R3..R4)



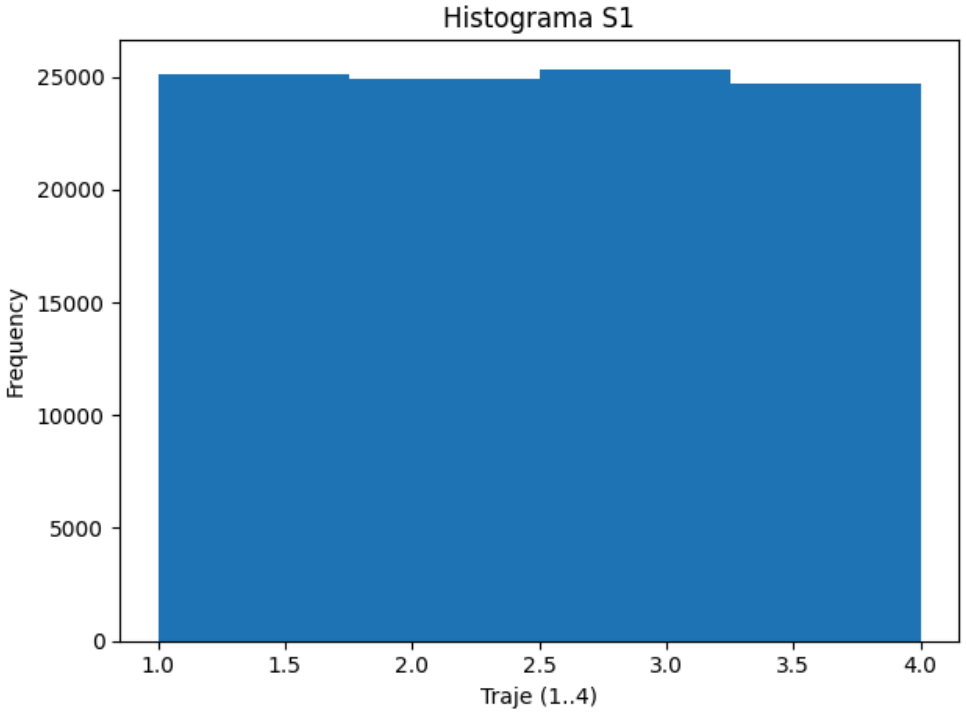


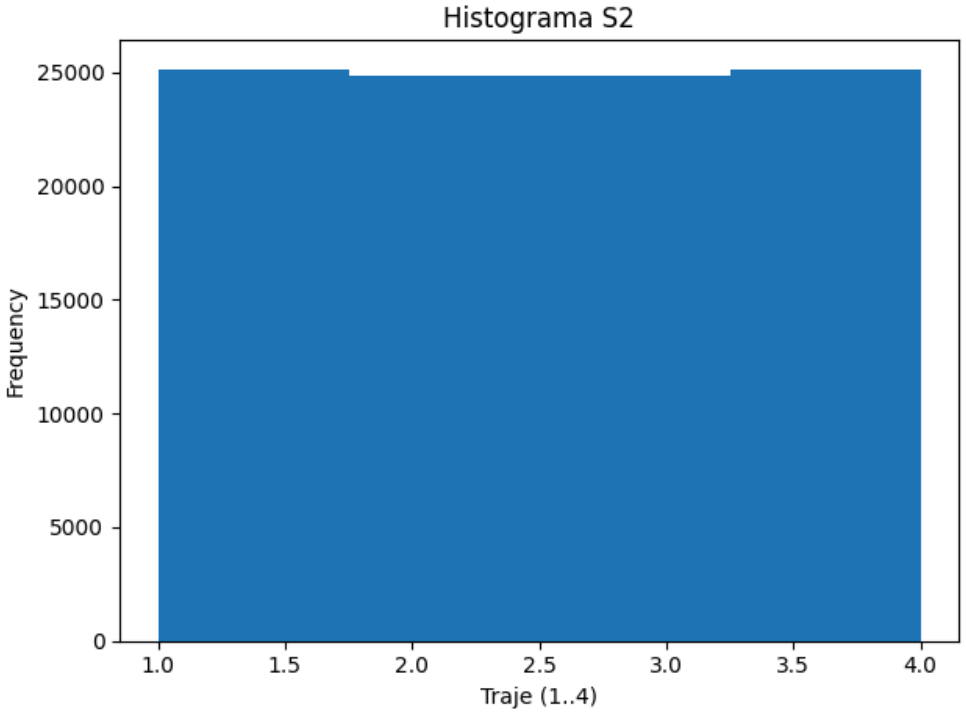
Histograma de rangos (R5)



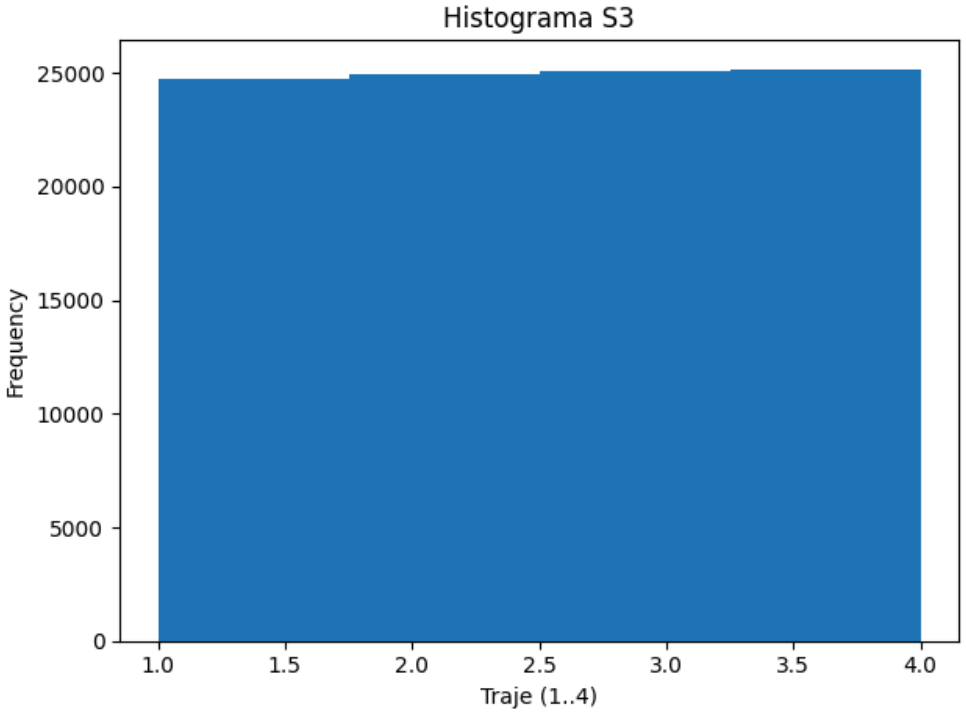
Lectura: Distribuciones ~uniformes por carta → no hay sesgo de valor.

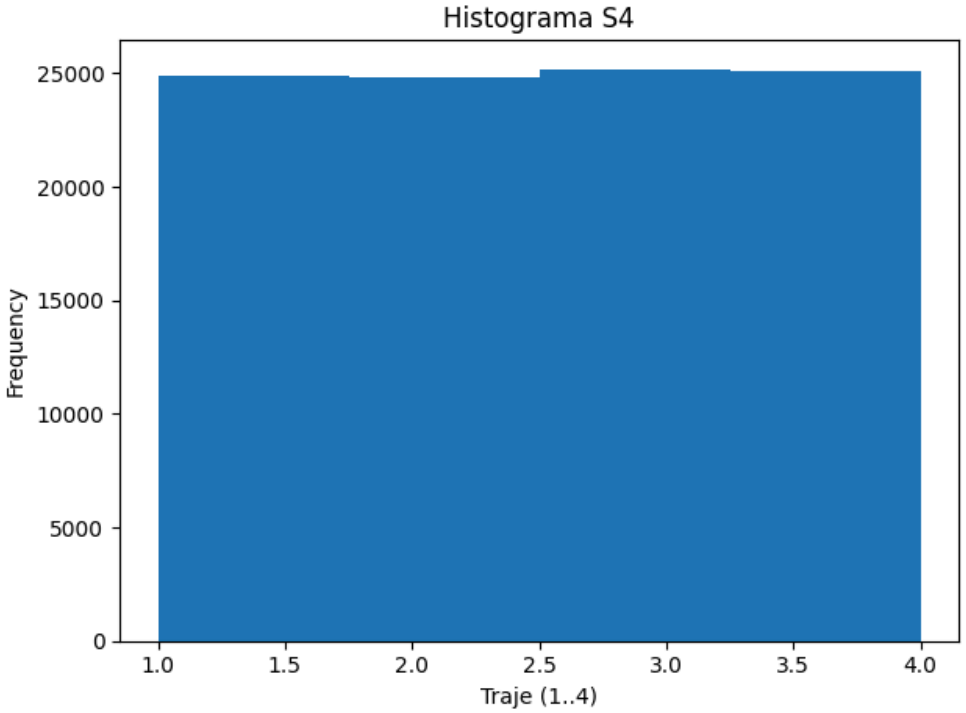
Histogramas de Suits (S1..S2)



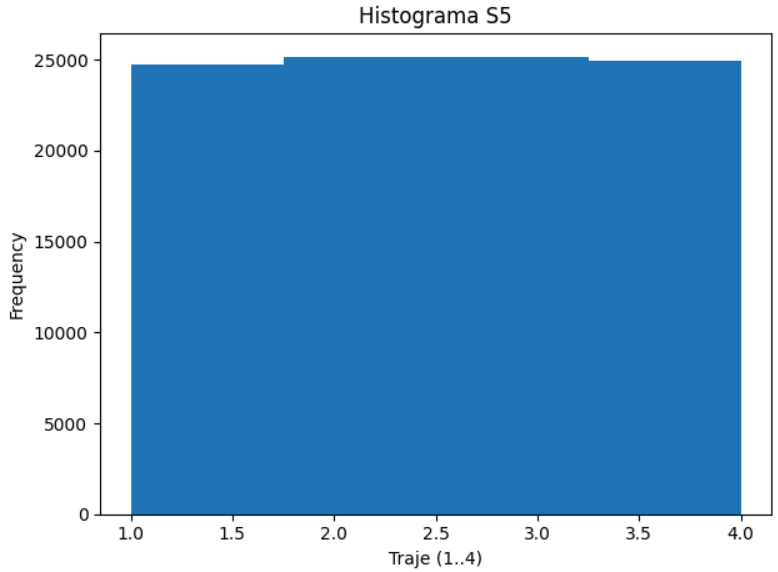


Histogramas de Suits (S3..S4)





Histograma de Suit (S5)



Lectura:

- Distribuciones prácticamente uniformes.
- No hay sesgo hacia un Suit particular.
- La dificultad del modelo proviene del **desbalance de clases**, no de las variables de entrada.

Ingeniería de características (resumen)

Se diseñaron features para capturar **patrones de póker**:

- `is_flush` (5 cartas mismo Suit), `is_straight` (rango consecutivo).
- Multiplicidades de rango: `num_pairs`, `has_three`, `has_four`, `max_count_rank`.
- **Cardinality**: `unique_ranks`, `unique_suits`.
- Estadísticos de ranks: `rank_sum`, `rank_mean`, `rank_std`, `top1_rank..top3_rank`.
- **Gaps** entre cartas: `rank_gap12..rank_gap45`.

Objetivo: acercar las variables a la semántica real de la etiqueta.

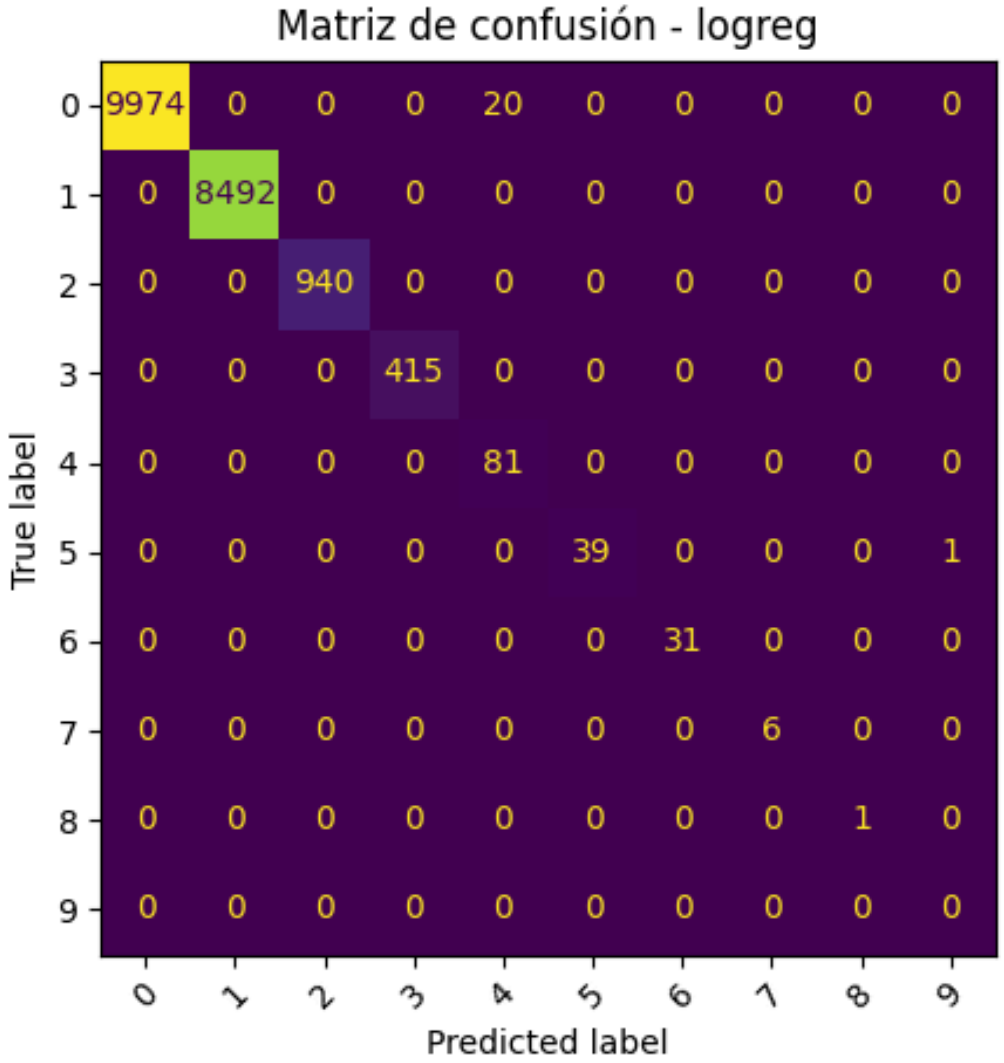
Resultados de Cross-Validation

Modelo	Accuracy (media ± std)	F1-macro (media ± std)
Logistic Regression	0.9988 ± 0.0003	0.92 ± 0.06
Random Forest	0.9999 ± 0.0000	0.96 ± 0.06

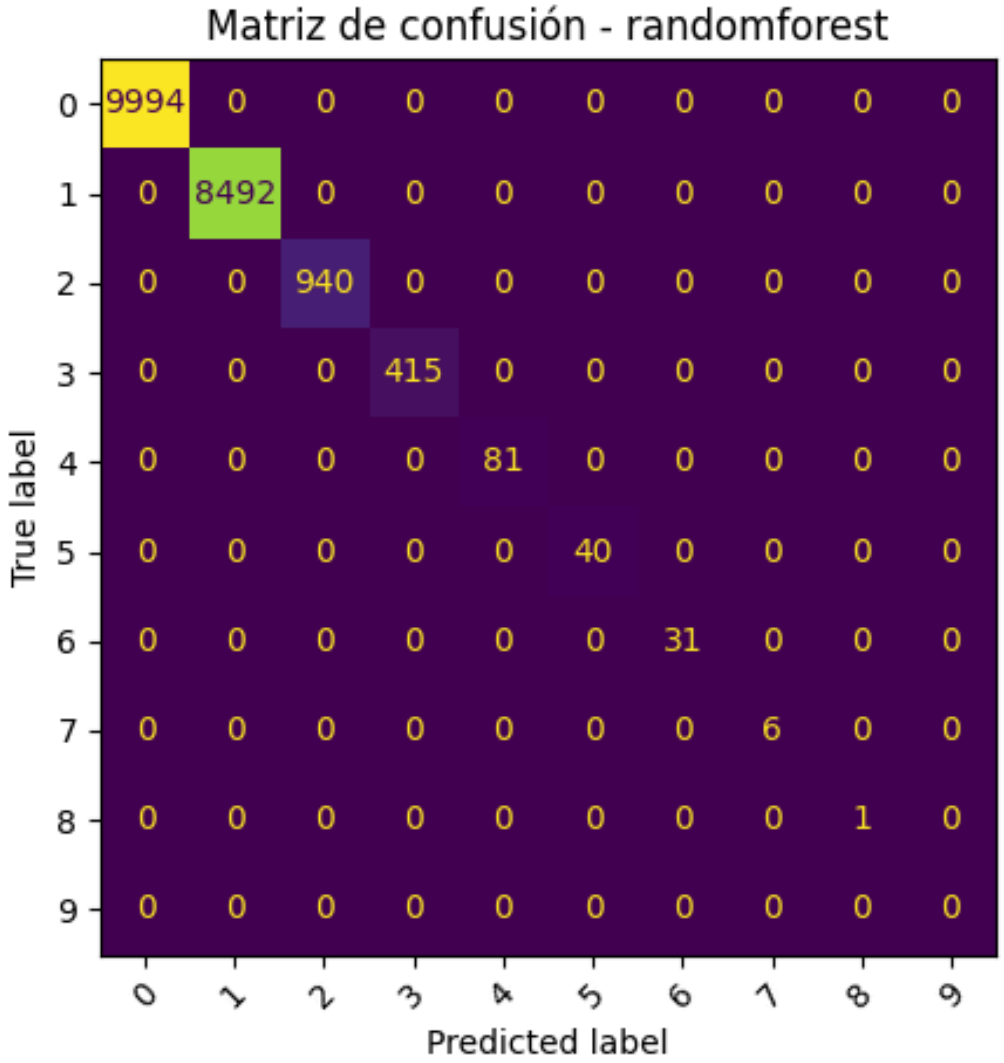
Lectura:

- Ambos modelos alcanzan **accuracy casi perfecta** → dominan la clase mayoritaria.
- **Macro-F1** diferencia mejor:
  - LogReg ~0.92, peor en clases raras.
  - Random Forest ~0.96, mejor desempeño en minoritarias.
- Persisten problemas en manos raras por su escasez.

Matriz de confusión — Logistic Regression (holdout 20%)



Matriz de confusión — Random Forest (holdout 20%)



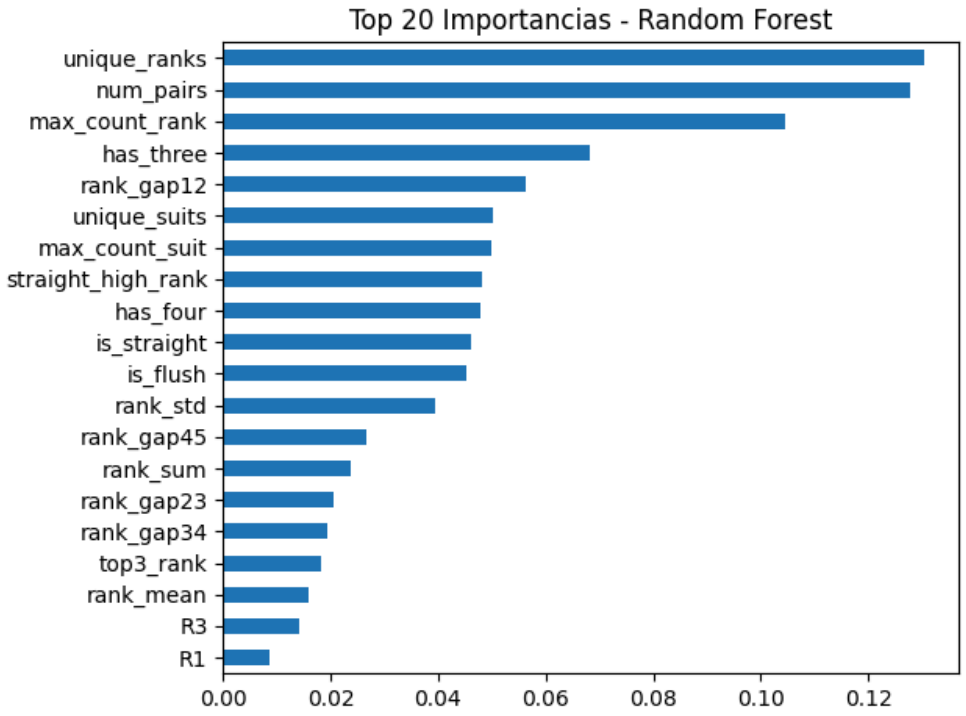
Explicación de la gráfica de Importancia de características (RF)

- El **Random Forest** calcula qué variables son más útiles para separar las clases.
- En el **Top-20** aparecen principalmente las **features derivadas** que diseñamos:
  - `unique_ranks` → cuántos rangos distintos hay en la mano (clave para detectar pares, tríos o full house).
  - `num_pairs`, `has_three`, `has_four` → permiten identificar jugadas concretas.
  - `max_count_rank` → frecuencia máxima de un rango (útil para diferenciar par, trío o póker).
- También aparecen métricas globales como `rank_sum` o `rank_std`, que ayudan a detectar secuencias (straights).

Conclusión:

Las variables **ingeniadas a partir del dominio del póker** resultan más importantes que los valores crudos de cartas (R1..R5, S1..S5).

Importancia de características — Random Forest (Top-20)



Lectura:

- `unique_ranks`, `num_pairs`, `max_count_rank`, `has_three` son claves.
- Refleja que las features de dominio capturan bien los patrones de póker.

Hallazgos y Recomendaciones

- RF mejora frente a LogReg, pero aún falla en clases ultra raras.
- Recomendaciones:
  - Técnicas de **re-muestreo** o **ajuste de pesos por clase**.
  - Probar **Gradient Boosting / XGBoost**.
  - Features específicas para *full house* y *straight-flush*.

Aplicación práctica:
Un modelo así podría clasificar **manos comunes de póker**, pero aún no es confiable para detectar jugadas raras en contextos reales.

Apéndice

- `cv_results.csv`, `cv_summary.csv` → métricas de CV.
- `classification_report_*.txt` → precision/recall/F1 por clase en holdout.
- `dataset_with_features.csv` → datos originales + derivadas.

Pipeline resumido:

Carga → EDA → Features → Modelos → Validación → Resultados

¡Gracias!