# Hiperparámetros en regresión logística

## 1. Introducción

La regresión logística es un modelo supervisado que se utiliza para predecir la probabilidad de que un evento ocurra (binaria), la categoría de una variable con varias clases (multinomial) o un valor ordenado (ordinal).

Nota importante sobre la librería:

scikit-learn **no implementa regresión logística ordinal de forma nativa**. Aunque se puede tratar una variable ordinal como multinomial, esto **ignora el orden natural de las clases**.

Para modelos ordinales reales se recomienda usar librerías específicas como:

- **mord** (Python): LogisticIT, LogisticAT, LogisticSE
- **statsmodels** (OrderedModel)

En cualquiera de estas últimas librerías, el manejo de hiperparámetros se usa con otros nombres difrentes a los de la scikit-learn.

Para que el modelo funcione bien, hay que definir **hiperparámetros**, que son valores que **no aprende el modelo automáticamente** sino que tú decides antes de entrenar.

Los principales hiperparámetros son:

- C: fuerza de regularización
- penalty: tipo de regularización (L1, L2, ElasticNet)
- solver: algoritmo de optimización
- max_iter: número máximo de iteraciones del solver
- class_weight: ponderación de clases desbalanceadas

## 2. Regularización: L1 (Lasso) vs L2 (Ridge) vs ElasticNet

### L2 – Ridge

- Penaliza los coeficientes grandes mediante el **cuadrado** de su valor.

- Todos los coeficientes se mantienen pero se reducen : evita sobreajuste:
- Útil cuando:
    - Hay variables correlacionadas
    - No se necesita eliminar variables
    - Dataset pequeño o mediano

### L1 – Lasso

- Penaliza los coeficientes mediante el **valor absoluto.**
- Puede reducir algunos coeficientes **exactamente a 0**, eliminando variables irrelevantes.
- Útil cuando:
    - Muchas variables, algunas irrelevantes
    - Se busca **interpretabilidad**
    - Dataset mediano o grande

### ElasticNet

- Mezcla de L1 y L2, controlada por l1_ratio
- Combina selección de variables (L1) y suavizado de coeficientes (L2)
- Útil cuando:
    - Muchas variables correlacionadas
    - Se quiere reducir dimensionalidad y suavizar coeficientes al mismo tiempo

# 3. C – Intensidad de la regularización

- **C es el inverso de lambda**:
    - C pequeño → regularización fuerte → coeficientes más conservadores
    - C grande → regularización débil → coeficientes más libres

**Orientación según tamaño de dataset:**

| Tamaño de dataset | C recomendado | Nota |
|---|---|---|
| Muy pequeño (<200 filas) | 0.01–0.1 | Evitar sobreajuste |
| Pequeño (200–1,000) | 0.1–0.5 | Regularización moderada |
| Mediano (1,000–10,000) | 0.5–1.0 | Regularización ligera |
| Grande (10,000–100,000) | 1–10 | Dataset grande → coeficientes más libres |
| Muy grande (>100,000) | 5–10 | Considerar solver optimizado (saga) |

# 4. Solver – algoritmo de optimización

| Solver | Cuándo usar | Limitaciones |
|---|---|---|
| liblinear | Binaria, pequeña, L1 o L2 | No multinomial |
| lbfgs | Binaria o multinomial, pequeño/mediano | Solo L2 |
| saga | Multinomial, L1/L2/ElasticNet, datasets grandes | Muy eficiente |
| newton-cg | Multinomial, L2 | Estable pero lento, no L1 |

**Regla:** cambiar solver solo si necesitas L1/ElasticNet o tienes datasets grandes/multinomial.

# 5. max_iter – número máximo de iteraciones

- Controla cuántos pasos da el solver para ajustar coeficientes
- **Recomendación:**
  - Default = 100 → subir a 500–1000 si el modelo **no converge**
  - Datasets muy grandes → 1000+

# 6. class_weight – clases desbalanceadas

- Ajusta la penalización de las clases minoritarias
- 'balanced' → ajusta automáticamente según frecuencia
- Útil cuando hay desbalance fuerte entre clases

# 7. Recomendación según tipo de regresión

| Tipo de regresión | Qué usar | Notas |
|---|---|---|
| Binaria | L1 (Lasso) o L2 (Ridge) | Odds ratios claros, coeficientes interpretables |
| Multinomial | L2 o ElasticNet | Coeficientes comparan cada clase contra la clase de referencia |

| | | |
|---|---|---|
| Ordinal | L2 o L1 (según librería) | Aprovecha el orden natural de las clases; usar statsmodels o mord |

# 8. Combinación de hiperparámetros según tamaño y problema

| Tamaño | Tipo de dataset | Penalty | C | Solver | max_iter | class_weight |
|---|---|---|---|---|---|---|
| Muy pequeño (<200) | Binaria | L2 | 0.01–0.1 | Liblinear | 500 | balanced si desbalanceada |
| Pequeño (200–1,000) | Binaria | L2/L1 | 0.1–0.5 | Liblinear/lbfgs | 500 | balanced si desbalanceada |
| Mediano (1,000–10,000) | Binaria o Multinomial | L2/ElasticNet | 0.5–1.0 | Lbfgs/saga | 500 | según necesidad |
| Grande (10,000–100,000) | Binaria o Multinomial | L2/ElasticNet | 1–10 | Saga | 500–1000 | según necesidad |
| Muy grande (>100,000) | Binaria o Multinomial | L2/ElasticNet | 5–10 | saga | 1000+ | según necesidad |

# 9. Flujo de trabajo recomendado

1. **EDA:** explorar datos, detectar valores faltantes y outliers
2. **Dividir dataset:** entrenamiento y test (evitar fuga de datos)
3. **Preprocesamiento:** imputación de faltantes y escalado solo con entrenamiento → aplicar a test
4. **Entrenar modelo:** seleccionar penalty, C, solver y max_iter según tamaño y tipo de regresión
5. **Evaluar modelo:** métricas de test (accuracy, F1, matriz de confusión, ROC/AUC si binaria)

6. **Ajuste de hiperparámetros:** GridSearchCV o RandomizedSearchCV, solo con entrenamiento / validación cruzada (etapa más avanzada de manejo de datos)
7. **Uso final:** predecir nuevos datos aplicando mismos preprocesamientos

# 10. Resumen conceptual simple de hiperparámetros

- **L2/Ridge:** suaviza coeficientes → no elimina variables → buen generalizador
- **L1/Lasso:** puede eliminar variables → útil para selección y datasets con muchas variables
- **ElasticNet:** mezcla de ambos → selección + suavizado
- **C:** controla fuerza de regularización → bajo C = más conservador; alto C = más flexible
- **Solver:** define cómo se ajustan los coeficientes → elegir según penalty y tamaño del dataset
- **max_iter:** sube si no converge
- **class_weight:** ajustar si hay desbalance de clases

# 11. Métricas de evaluación y su interpretación

Cuando entrenamos un modelo de clasificación, como la **regresión logística**, queremos medir cuán **bien predice**. Para eso usamos varias métricas:

## A) Accuracy (Exactitud)

- **Qué mide:** el porcentaje de predicciones correctas sobre el total.
- **Interpretación:**
    - Si las clases están **balanceadas**, es un buen indicador general.
    - Comparación entrenamiento vs test:
        - **Entrenamiento ≈ Test** → buen ajuste, modelo generaliza bien
        - **Entrenamiento >> Test** → sobreajuste (memoriza los datos de entrenamiento)

Precaución: si las clases están muy desbalanceadas (ej. 90% clase A, 10% clase B), la accuracy puede ser engañosa.

## B) Precision (Precisión)

- **Qué mide:** de todas las predicciones positivas que hizo el modelo, cuántas fueron correctas.
- **Interpretación:**
  - Alta precision → pocas falsas alarmas (pocos falsos positivos)
  - Útil si un **falso positivo es costoso**, por ejemplo diagnosticar enfermedad cuando no hay.

## C) Recall (Sensibilidad / True Positive Rate)

- **Qué mide:** de todos los positivos reales, cuántos predijo correctamente.
- **Interpretación:**
  - Alto recall → pocos falsos negativos
  - Útil si es crítico **no perder positivos**, por ejemplo detectar pacientes enfermos.

## D) F1-score

- **Qué mide:** combinación de precision y recall mediante la **media armónica**.
- **Interpretación:**
  - Equilibra precision y recall
  - Muy útil cuando las clases están desbalanceadas, porque considera tanto falsos positivos como falsos negativos

## E) Matriz de confusión

- **Qué mide:** conteo de predicciones correctas e incorrectas por clase:

| | **Predicho Positivo** | **Predicho Negativo** |
|---|---|---|
| **Real Positivo** | Verdadero Positivo | Falso Negativo |
| **Real Negativo** | Falso Positivo | Verdadero Negativo |

- **Interpretación:**
  - Permite ver **qué clases se confunden**

- Útil para decidir si hay que **cambiar threshold** de clasificación o ajustar hiperparámetros

### F) ROC / AUC

- **Qué mide:** capacidad del modelo de discriminar entre clases.
- **Interpretación:**
  - **AUC cercano a 1 → excelente**
  - **AUC ≈ 0.5 → aleatorio**
  - Útil para comparar modelos y ajustar threshold de decisión

## 12. Cómo interpretar métricas combinadas con hiperparámetros

| Situación | Qué indica | Qué ajustar |
|---|---|---|
| Entrenamiento muy alto, test bajo | Sobreajuste (memoriza entrenamiento) | Disminuir C (más regularización), aumentar penalty, considerar más datos |
| Entrenamiento bajo, test bajo | Subajuste (modelo no aprende) | Aumentar C (menos regularización), probar L1/L2 diferente, revisar features |
| Accuracy y otras métricas similares en entrenamiento y test | Modelo estable | Hiperparámetros adecuados, buen ajuste |

### Regla general:

- Las métricas en **entrenamiento** sirven para ver si el modelo **aprende** los patrones.
- Las métricas en **test** sirven para ver si el modelo **generaliza a datos nuevos**.
- **Nunca ajustes hiperparámetros usando test**, solo entrenamiento o validación cruzada.