

# Linear • Logistic Regression

## Fokus:

- Linear Regression (regresi)
- Logistic Regression (klasifikasi biner)
- **Regularisasi ringan** (L2/L1), scaling, dan tips anti-overfitting

## Tujuan Sesi

- Memahami **fungsi tujuan** dan **solusi** Linear & Logistic
- Mengenal **loss**: MSE (regresi) dan log-loss (klasifikasi)
- Memahami **regularisasi** (L2/L1): efeknya pada bobot
- Praktik aman: **scaling, validasi, threshold, imbalance**

# Bagian A — Linear Regression

## Intuisi Linear Regression

Memprediksi target numerik  $y$  dari fitur  $\mathbf{x}$ :

$$\hat{y} = \mathbf{w}^\top \mathbf{x} + b$$

**Tujuan:** meminimalkan MSE:

$$J(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^n \left( y_i - (\mathbf{w}^\top \mathbf{x}_i + b) \right)^2$$

Garis/hiperbidang terbaik dalam arti **kuadrat error minimum**.

## Solusi Tertutup (Normal Equation)

Tanpa regularisasi, dalam bentuk matriks:

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

**Catatan:**

- Perlu  $\mathbf{X}^\top \mathbf{X}$  **invertible** (atau gunakan **pseudo-inverse**)
- Untuk fitur banyak/kolinear  $\rightarrow$  lebih aman pakai **Regresi Ridge** (L2)

## Evaluasi Regresi

- **MAE:** rata-rata  $|y - \hat{y}|$
- **MSE:** rata-rata  $(y - \hat{y})^2$
- **RMSE:**  $\sqrt{\text{MSE}}$
- $R^2$ : proporsi variasi yang dijelaskan model

Pilih metrik sesuai konteks (MAE robust, RMSE penalti error besar).

## Bagian B — Logistic Regression

## Intuisi Logistic Regression

Klasifikasi biner dengan probabilitas:

$$p(y=1 \mid \mathbf{x}) = \sigma(z) = \frac{1}{1 + e^{-z}}, \quad z = \mathbf{w}^\top \mathbf{x} + b$$

**Prediksi label** pakai ambang (default 0.5):

$\hat{y} = 1$  jika  $p \geq 0.5$ , else 0.

## Loss: Log-Loss (Binary Cross-Entropy)

Untuk satu contoh:

$$\ell(\mathbf{w}) = -[y \log p + (1 - y) \log(1 - p)]$$

Total:

$$J(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^n \left( -y_i \log p_i - (1 - y_i) \log(1 - p_i) \right)$$

Dioptimasi numerik (gradient-based). Tidak ada solusi tertutup.

## Evaluasi Klasifikasi

- **Accuracy** (mudah, bisa bias kalau imbalance)
- **Precision / Recall / F1**
- **ROC-AUC** (kurva TPR vs FPR), **PR-AUC** (untuk imbalance)
- **Confusion Matrix, Calibration** (opsional)

Threshold bisa diubah sesuai **biaya salah** (precision vs recall).

## Bagian C — Regularisasi Ringan

## Kenapa Perlu Regularisasi?

- Mengontrol **kompleksitas** model
- Mengurangi **overfitting**, menangani **multikolinearitas**
- Mendorong **bobot kecil/jarang** → generalisasi lebih baik

## L2 (Ridge) & L1 (Lasso)

Ridge (L2) menambah penalti kuadrat:

$$J_{\text{ridge}} = \text{Loss} + \lambda \|\mathbf{w}\|_2^2$$

Lasso (L1) menambah penalti absolut:

$$J_{\text{lasso}} = \text{Loss} + \lambda \|\mathbf{w}\|_1$$

Efek:

- L2 → **mengecilkan** bobot mulus (semua fitur tetap, lebih stabil)
- L1 → mendorong **banyak bobot ke 0** (seleksi fitur otomatis)

## Catatan Praktis (scikit-learn)

- **LinearRegression** (tanpa regular) vs **Ridge/Lasso**
- **LogisticRegression** default pakai **L2**  
Parameter  $C = \frac{1}{\lambda} \rightarrow C \text{ kecil} = \text{regular lebih kuat}$
- **StandardScaler** sebelum regular (skala memengaruhi penalti!)

## Bagian D — Contoh Mini & Rumus

## Linear + L2: Rumus Matrix

**Ridge** solution (tertutup):

$$\hat{\mathbf{w}}_{\text{ridge}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$$

Menstabilkan inversi saat  $\mathbf{X}^\top \mathbf{X}$  nyaris singular (kolinear).

## Logistic + L2: Loss Terregularisasi

Log-loss + penalti L2:

$$J(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n \left( -y_i \log p_i - (1 - y_i) \log(1 - p_i) \right) + \lambda \|\mathbf{w}\|_2^2$$

Gradien ringkas (skalar):

$$\nabla_{\mathbf{w}} \approx \frac{1}{n} \sum_i (p_i - y_i) \mathbf{x}_i + 2\lambda \mathbf{w}$$

## Bias–Variance (Singkat)

- Tanpa reguler → bias kecil, **variance besar** (overfit)
- Dengan reguler → bias naik sedikit, variance turun → **total error bisa turun**

## Bagian E — Praktik Aman

## Pipeline & Scaling (Anti-Leakage)

- **Split dulu** → fit **scaler** & model **di train** saja
- Gunakan **Pipeline** agar transformasi **ikut CV**
- Simpan **seed** (reprodusibilitas)

## Koding Minimal (Regresi)

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import Ridge, LogisticRegression
from sklearn.metrics import mean_squared_error, classification_report
import numpy as np

# X: fitur, y: target (numerik untuk Ridge; biner untuk Logistic)
X_tr, X_te, y_tr, y_te = train_test_split(X, y, test_size=0.2, random_state=42, stratify=None)

# Linear dengan L2 (Ridge)
ridge = make_pipeline(StandardScaler(), Ridge(alpha=1.0)) # alpha =  $\lambda$ 
ridge.fit(X_tr, y_tr)
rmse = mean_squared_error(y_te, ridge.predict(X_te), squared=False)
print("RMSE (Ridge):", rmse)

# Logistic dengan L2
logreg = make_pipeline(StandardScaler(), LogisticRegression(C=1.0, max_iter=1000)) # C=1/ $\lambda$ 
logreg.fit(X_tr, y_tr)
print(classification_report(y_te, logreg.predict(X_te)))
```

## Grid Search Ringan

```
from sklearn.model_selection import GridSearchCV, StratifiedKFold

pipe = make_pipeline(StandardScaler(), LogisticRegression(max_iter=1000))
param_grid = {"logisticregression__C": [0.1, 1.0, 10.0]}
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

gs = GridSearchCV(pipe, param_grid, scoring="f1", cv=cv, n_jobs=-1)
gs.fit(X_tr, y_tr)
print(gs.best_params_, gs.best_score_)
```

Untuk Ridge/Lasso: ganti estimator & hyperparameter ( `alpha` ).

## Tips Kelas Tidak Seimbang

- Gunakan **class\_weight='balanced'** pada LogisticRegression
- Atur **threshold** berdasarkan **precision–recall** tradeoff
- Gunakan **PR-AUC** atau **F1** per kelas

```
logreg = LogisticRegression(class_weight="balanced", max_iter=1000)
```

## Diagnostik Ringan

- **Learning curve:** cek data cukup atau tidak
- **Validation curve:** efek  $C/\alpha$  terhadap skor
- **Calibration curve:** cek probabilitas logistic “well-calibrated”

Jangan lupa **feature importance** (koefisien) setelah scaling.

## Bagian F — Ringkasan & Tugas

## Ringkasan

- **Linear**: MSE, solusi tertutup; tambahkan **L2** untuk stabilitas
- **Logistic**: probabilitas via sigmoid, log-loss; **L2** default
- **Regularisasi ringan**: L2 mengecilkan bobot, L1 bisa nol-kan bobot
- **Scaling & Pipeline** wajib untuk evaluasi adil

## Tugas Mini

1. Buat **Ridge** dengan  $\alpha \in \{0.1, 1, 10\}$ , bandingkan **RMSE** (CV=5).
2. Buat **Logistic** dengan  $C \in \{0.1, 1, 10\}$ , bandingkan **F1** (CV=5).
3. Coba **class\_weight='balanced'** dan laporkan perubahan **precision/recall**.
4. Tulis satu contoh **fitur interaksi** yang *masuk akal* di dataset kalian.