Logistic Regression – Feature Engineering & Modell-Feintuning Cheat Sheet

1. Feature Engineering mit np.column_stack([...])

Verfügbare Transformationen (inkl. kurzer Beschreibung & Wirkung):

- Linear: x1, x2
  → Trennt entlang gerader Linien parallel zu x- oder y-Achse.
  → Beispiel: linke Hälfte Klasse 0, rechte Hälfte Klasse 1

- Quadratisch: x1**2, x2**2
  → Trennt entlang parabelförmiger Kurven.
  → Beispiel: Klasse 1 nur in Randbereichen, Mitte = Klasse 0

- Interaktionen: x1 * x2
  → Erkennt rechteckige oder L-förmige Bereiche.
  → Beispiel: Klasse 1 nur wenn x1 groß UND x2 klein

- Höhere Grade: x1**3, x2**3
  → Modelliert starke Krümmungen oder asymmetrische Trennung.
  → Vorsicht: Kann overfitten!

- Radiale Trennung: x1**2 + x2**2
  → Trennt konzentrisch vom Mittelpunkt aus.
  → Beispiel: Klasse 1 bildet Kreisring um Klasse 0

- Wurzel: np.sqrt(x1), np.sqrt(x2)
  → Dämpft Unterschiede bei kleinen Werten.
  → Nützlich wenn Punkte sich im Zentrum stark ballen.

- Logarithmisch: np.log(x1 + 1), np.log(x2 + 1)
  → Verhält sich ähnlich wie Wurzel, robuster bei Ausreißern.

- Exponentiell: np.exp(-x1), np.exp(-x2)
  → Modelliert abfallende Einflüsse (z. B. Aktivitätsabnahme)

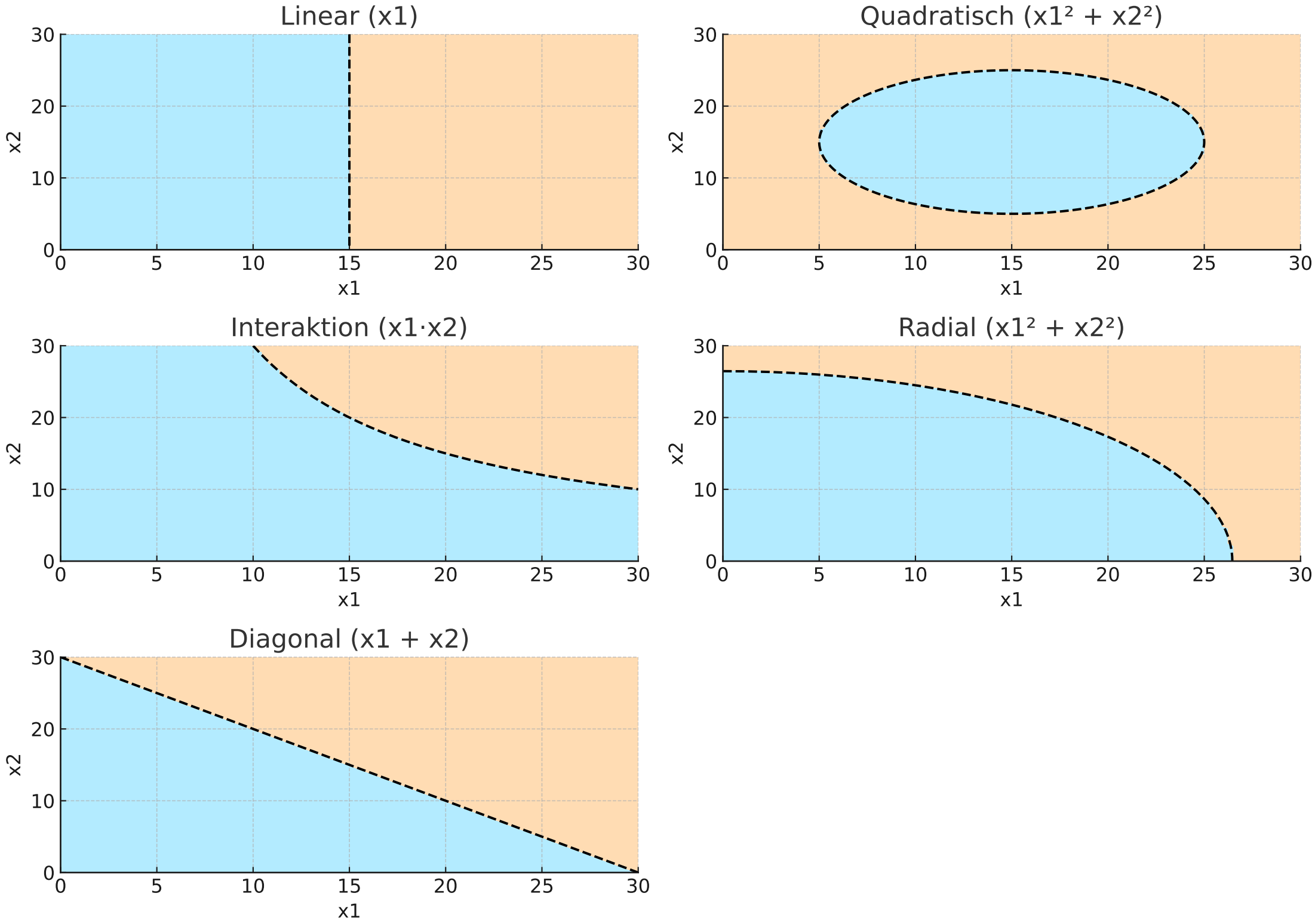
- Kombinationen: x1 + x2, x1 - x2
  → Trennt entlang diagonaler Linien.
  → Beispiel: unterhalb der Linie x1 + x2 = 15 ist Klasse 0

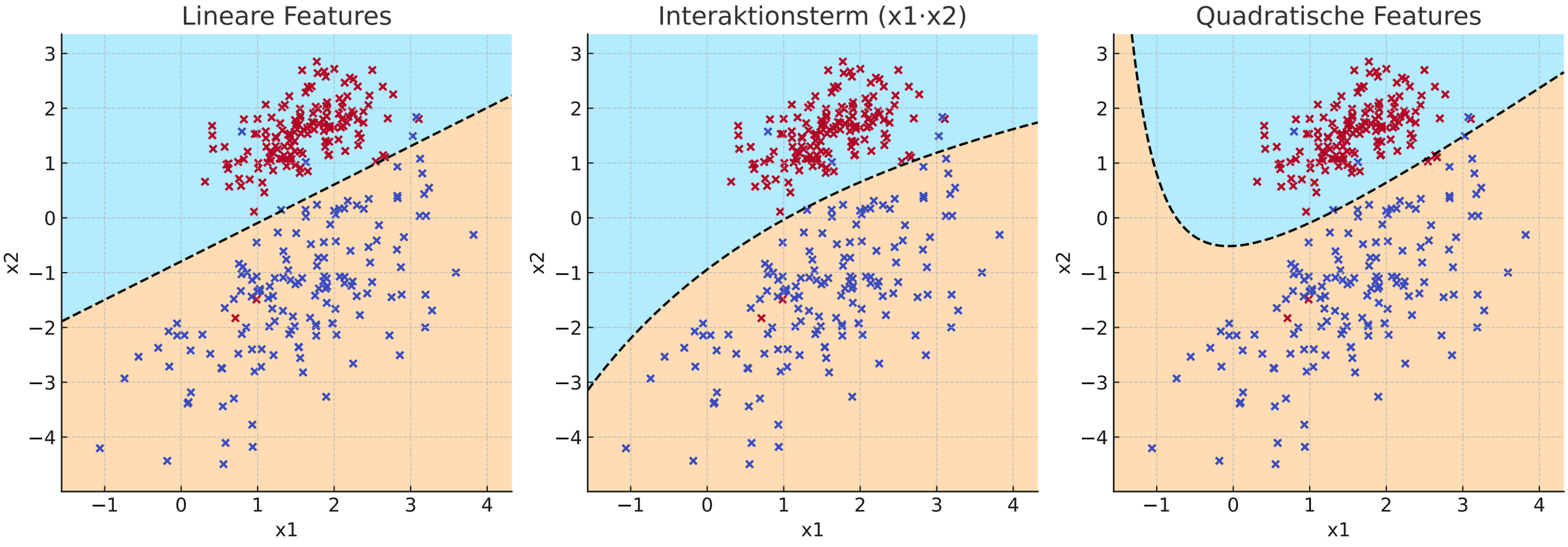
Beispiel:

```
return np.column_stack([
    x1,
    x2,
    x1 * x2,
    x1**2,
    x2**2
])
```

2. Parameter für LogisticRegression (sklearn)

```
def logistic_regression_params_sklearn():
    return {
        'penalty': 'l2',         # 'l2' (Standard), 'none' (keine Einschränkung)
        'C': 1.0,                # Stärke der Regularisierung (kleiner = stärker)
        'solver': 'lbfgs',       # Für kleine/mittlere Daten geeignet
        'max_iter': 10000        # Erhöhen, falls Training zu früh abbricht
    }

------------------------------------------------------------
3. Troubleshooting & Modellanalyse
------------------------------------------------------------

Fall 1: Niedrige Accuracy (Train & Test)
→ Modell zu simpel → Komplexere Features ausprobieren

Fall 2: Train gut, Test schlecht
→ Overfitting → Features reduzieren oder C verkleinern

Fall 3: Loss hoch trotz guter Accuracy
→ Wahrscheinlichkeiten nicht gut → besseres Feature-Mapping

 Nutze plot_datapoints(...) & plot_logistic_regression(...) zum Visualisieren!
```


Linear (x1)
Quadratisch (x1² + x2²)
Interaktion (x1·x2)
Radial (x1² + x2²)
Diagonal (x1 + x2)
x1
x2
0
5
10
15
20
25
30


Lineare Features
Interaktionsterm (x1·x2)
Quadratische Features
x1
x2