# Praktikum 4 – Machine Learning
# Logistic Regression

Prepared By:
Dr. Sirojul Munir S.Si,M.Kom
Diah Ayu Puspasari

Tujuan:

1. Menerapkan algoritma Logistic Regression untuk memprediksi status stunting pada balita.
2. Melakukan pra-pemrosesan data (encoding kategori, scaling fitur, pembagian data latih/uji).
3. Menganalisis hasil prediksi dan evaluasi model dengan metrik seperti akurasi, presisi, recall, F1-score, dan ROC-AUC.
4. Memahami hubungan antar variabel (melalui korelasi) dan interpretasi koefisien regresi logistik dalam konteks stunting.

Dateline : 1 Pekan

Gitlab/Github :

Branch Repository : [PRODI ROMBEL] _[NAMASINGKAT]_[NIM] ( contoh: ti01_budi_0110112001)

## Aturan Pengerjaan:

1. Gunakan text editor yang nyaman bagi anda
2. Diperkenankan mengerjakan langsung bagi yang sudah memahami dan menguasai materi
3. Dilarang melakukan tindakan plagiarism (asisten lab akan mengecek hasil pekerjaan)
    a. 1x nilai praktikum terkait bernilai 0
    b. 2x nilai matakuliah pemrograman web E
    c. 3x mahasiswa akan di sidang komite etik kampus

## Pendahuluan

Regresi logistik merupakan salah satu algoritma supervised learning yang digunakan untuk memprediksi variabel kategorik (biner) berdasarkan satu atau lebih variabel numerik maupun kategorik.

Berbeda dengan regresi linier yang hasilnya berupa nilai kontinu, regresi logistik menghasilkan probabilitas (peluang) suatu kejadian yang kemudian dikonversi menjadi kelas tertentu, seperti Ya/Tidak, 0/1, atau Positif/Negatif.

Pada praktikum ini, digunakan dataset stunting untuk membangun model klasifikasi yang dapat memprediksi apakah seorang balita mengalami stunting (1) atau tidak stunting (0) berdasarkan variabel-variabel seperti umur, tinggi badan, berat badan, dan jenis kelamin.

Melalui penerapan algoritma regresi logistik, diharapkan mahasiswa mampu memahami:

- konsep dasar regresi logistik,
- langkah-langkah pemrosesan data,
- serta cara evaluasi performa model klasifikasi.

## Langkah Awal

Sebelum memulai praktikum ini, pastikan Anda telah menyiapkan struktur folder di Google Drive / Jupyter seperti pada praktikum sebelumnya. Untuk praktikum ke-4, buat sub-folder baru dengan nama praktikum04/ di dalam direktori utama praktikum_ml/.

Struktur folder yang digunakan adalah sebagai berikut:

```
praktikum_ml/
├── praktikum01/
├── praktikum02/
├── praktikum03/
│   ├── data/
│   ├── notebooks/
│   ├── model/
│   └── reports/
```

**Catatan:**

1. Jika telah memiliki folder praktikum_ml, cukup tambahkan folder praktikum04.
2. Semua file notebook pada praktikum ini disimpan di dalam folder notebooks/ dengan nama praktikum04.ipynb.
3. Dataset ditempatkan di folder data/.
4. Model hasil training disimpan di folder model/ (format .pkl atau .joblib).
5. Visualisasi atau laporan hasil analisis disimpan di folder reports/.Jalankan file installer yang sudah diunduh.

# Logistic Regression

## 1. Import Library

```
[1]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt

     from sklearn.model_selection import train_test_split
     from sklearn.compose import ColumnTransformer
     from sklearn.preprocessing import OneHotEncoder, StandardScaler
     from sklearn.pipeline import Pipeline
     from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import (
         accuracy_score, precision_score, recall_score, f1_score, roc_auc_score,
         confusion_matrix, classification_report, RocCurveDisplay, ConfusionMatrixDisplay
     )
```

Langkah pertama dalam praktikum ini adalah melakukan import terhadap seluruh library yang dibutuhkan. Library berfungsi sebagai kumpulan fungsi dan modul pendukung agar proses pemodelan, analisis data, dan visualisasi dapat dilakukan dengan lebih mudah dan efisien.

Berikut penjelasan masing-masing library yang digunakan:

| Library / Modul | Fungsi Utama |
|---|---|
| numpy | Digunakan untuk melakukan operasi numerik seperti perhitungan matriks, array, dan fungsi matematika. |
| pandas | Library utama untuk mengelola data dalam bentuk *DataFrame* (baris dan kolom). Memudahkan proses membaca, menulis, serta manipulasi dataset. |
| matplotlib.pyplot | Digunakan untuk membuat visualisasi seperti grafik batang, garis, atau scatter plot. |
| train_test_split dari sklearn.model_selection | Membagi dataset menjadi dua bagian: **data latih (train)** dan **data uji (test)** agar model bisa dievaluasi secara objektif. |
| ColumnTransformer dari sklearn.compose | Mengatur proses pra-pemrosesan kolom, misalnya menskalakan kolom numerik atau meng-*encode* kolom kategorik. |
| OneHotEncoder, StandardScaler dari sklearn.preprocessing | - **OneHotEncoder**: mengubah data kategorik menjadi bentuk numerik.<br>- **StandardScaler**: menstandarisasi data numerik agar memiliki skala yang sebanding (mean = 0, std = 1). |

| | |
|---|---|
| Pipeline dari sklearn.pipeline | Menggabungkan seluruh langkah pemrosesan (scaling, encoding, training model) ke dalam satu alur kerja yang rapi dan otomatis. |
| LogisticRegression dari sklearn.linear_model | Algoritma utama yang digunakan untuk membangun model klasifikasi biner (*stunting atau tidak stunting*). |
| sklearn.metrics | Berisi berbagai metrik evaluasi model seperti:<br>• accuracy_score → mengukur ketepatan prediksi<br>• precision_score, recall_score, f1_score → mengukur kualitas klasifikasi<br>• roc_auc_score → menilai performa berdasarkan probabilitas<br>• confusion_matrix, classification_report → ringkasan hasil prediksi<br>• RocCurveDisplay, ConfusionMatrixDisplay → menampilkan grafik ROC dan Confusion Matrix. |

## 2. Membaca data file CSV

```
[2]: df = pd.read_csv("../data/stunting_wasting_dataset.csv")
     df.head()
```

Selanjutnya, menggunakan library Pandas untuk membaca file data. Variabel path menyimpan lokasi folder di Google Drive / tempat file dataset berada. Fungsi pd.read_csv() kemudian membaca file tersebut dan menyimpannya ke dalam sebuah DataFrame.

## 3. Melihat informasi umum dataset

```
[3]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 6 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Jenis Kelamin      100000 non-null  object
 1   Umur (bulan)       100000 non-null  int64
 2   Tinggi Badan (cm)  100000 non-null  float64
 3   Berat Badan (kg)   100000 non-null  float64
 4   Stunting           100000 non-null  object
 5   Wasting            100000 non-null  object
dtypes: float64(2), int64(1), object(3)
memory usage: 4.6+ MB
```

Langkah selanjutnya adalah membaca dataset stunting_wasting_dataset.csv menggunakan library pandas, kemudian menampilkan informasi struktur data menggunakan fungsi df.info(). Berdasarkan hasil dari df.info(), dapat diketahui bahwa dataset berisi 100.000 baris (data balita) dan 6 kolom dengan rincian sebagai berikut:

| Nama Kolom | Tipe Data | Keterangan |
|---|---|---|
| Jenis Kelamin | object | Kategori jenis kelamin anak (Laki-laki / Perempuan). |
| Umur (bulan) | int64 | Umur balita dalam satuan bulan. |
| Tinggi Badan (cm) | float64 | Tinggi badan balita dalam satuan sentimeter. |
| Berat Badan (kg) | float64 | Berat badan balita dalam satuan kilogram. |
| Stunting | object | Status stunting anak (*Normal*, *Stunted*, *Severely Stunted*, atau *Tall*). |
| Wasting | object | Status wasting anak (*Underweight*, *Risk of Overweight*, dll). |

## 4. Data Pre-processing

Tahap ini bertujuan untuk menyiapkan data agar dapat digunakan oleh model machine learning. Pada regresi logistik, model hanya bisa menerima data dalam bentuk numerik, sehingga kolom kategorik seperti Jenis Kelamin dan Stunting harus diubah (di-encode) menjadi angka terlebih dahulu. Selain itu, tahap ini juga memastikan tidak ada nilai kosong (missing value) dan memeriksa hubungan antarvariabel.

### 4.1 Cek Missing Value

```
[4]: # cek missing value
     df.isnull().sum()

[4]: Jenis Kelamin        0
     Umur (bulan)        0
     Tinggi Badan (cm)   0
     Berat Badan (kg)    0
     Stunting            0
     Wasting             0
     dtype: int64
```

Perintah ini digunakan untuk memastikan apakah terdapat data yang hilang pada setiap kolom. Hasil menunjukkan seluruh kolom (Jenis Kelamin, Umur, Tinggi, Berat, Stunting, Wasting) memiliki nilai 0 pada jumlah missing value. Artinya, tidak ada data kosong, sehingga dataset dapat langsung digunakan tanpa proses imputasi.

### 4.2 Cek Nilai Unik

```
[5]: df['Stunting'].unique()

[5]: array(['Tall', 'Stunted', 'Normal', 'Severely Stunted'], dtype=object)

[6]: df['Jenis Kelamin'].unique()

[6]: array(['Laki-laki', 'Perempuan'], dtype=object)
```

1. Kolom Stunting memiliki 4 kategori: Tall, Stunted, Normal, Severely Stunted.
2. Kolom Jenis Kelamin memiliki 2 kategori: Laki-laki dan Perempuan.
3. Karena target prediksi (stunting) bersifat biner (ya/tidak), maka kategori ini akan disederhanakan menjadi dua kelas:
    - 1 → Stunted / Severely Stunted
    - 0 → Normal / Tall

### 4.3 Mapping Kolom Kategorik ke Bentuk Numerik

```
[7]: # 1. Mapping kolom Stunting -> biner
map_stunt = {'Stunted': 1, 'Severely Stunted': 1, 'Normal': 0, 'Tall': 0}
df['Stunting_bin'] = df['Stunting'].map(map_stunt).astype('Int64')

# 2. Mapping kolom Jenis Kelamin -> biner
#    Laki-laki = 1, Perempuan = 0
df['JK_bin'] = (df['Jenis Kelamin'] == 'Laki-laki').astype(int)

print("Distribusi Stunting_bin:\n", df['Stunting_bin'].value_counts())
print("\nDistribusi JK_bin:\n", df['JK_bin'].value_counts())
```

```
Distribusi Stunting_bin:
 Stunting_bin
0    78021
1    21979
Name: count, dtype: Int64

Distribusi JK_bin:
 JK_bin
1    50179
0    49821
Name: count, dtype: int64
```

- Kode di atas mengubah nilai teks menjadi bentuk numerik agar dapat diproses model.
- Kolom baru Stunting_bin digunakan sebagai target (Y).
- Kolom JK_bin digunakan sebagai fitur (X) tambahan.
- Hasil distribusi:
    - Stunting_bin → 78.021 data tidak stunting (0) dan 21.979 data stunting (1).
    - JK_bin → 50.179 laki-laki (1) dan 49.821 perempuan (0).
- Distribusi ini menunjukkan data relatif seimbang antara laki-laki dan perempuan, namun kelas “stunting” lebih sedikit dibanding “tidak stunting”.

### 4.4 Analisis Korelasi Antar Variabel Numerik

Fungsi corr() digunakan untuk menghitung korelasi Pearson antara setiap variabel numerik terhadap target Stunting_bin.

```
[16]: corr_matrix = df.corr(numeric_only=True)
      corr_matrix
```

[16]:

| | Umur (bulan) | Tinggi Badan (cm) | Berat Badan (kg) | Stunting_bin | JK_bin |
|---|---|---|---|---|---|
| **Umur (bulan)** | 1.000000 | 0.875869 | 0.665389 | 0.038630 | 0.004046 |
| **Tinggi Badan (cm)** | 0.875869 | 1.000000 | 0.626005 | -0.283855 | 0.073505 |
| **Berat Badan (kg)** | 0.665389 | 0.626005 | 1.000000 | 0.021090 | 0.045797 |
| **Stunting_bin** | 0.038630 | -0.283855 | 0.021090 | 1.000000 | -0.005981 |
| **JK_bin** | 0.004046 | 0.073505 | 0.045797 | -0.005981 | 1.000000 |

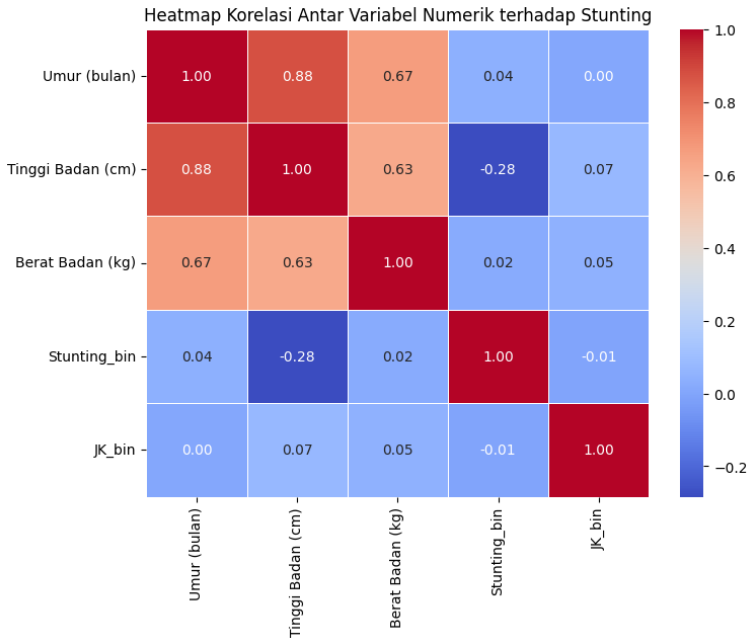
Nilai korelasi berkisar antara -1 hingga 1:

- Nilai mendekati 1 → hubungan positif kuat (naik bersama)
- Nilai mendekati -1 → hubungan negatif kuat (berlawanan arah)
- Nilai mendekati 0 → hubungan lemah atau tidak ada hubungan

## 4.5 Visualisasi Heatmap Korelasi

```
[17]: # Visualisasi heatmap
      import seaborn as sns
      import matplotlib.pyplot as plt

      plt.figure(figsize=(8,6))
      sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
      plt.title("Heatmap Korelasi Antar Variabel Numerik terhadap Stunting", fontsize=12)
      plt.show()
```



Visualisasi heatmap korelasi digunakan untuk melihat hubungan antar variabel numerik secara menyeluruh dalam dataset. Dengan heatmap, kita dapat mengetahui variabel mana yang memiliki korelasi kuat, sedang, atau lemah, serta arah hubungannya (positif atau negatif) terhadap variabel target Stunting_bin.

## 5. Pembagian Dataset (Training dan Testing)

Sebelum model dilatih, dataset harus dibagi menjadi dua bagian utama, yaitu:

- Data Latih (Training Set) — digunakan untuk melatih model agar dapat mengenali pola pada data.
- Data Uji (Testing Set) — digunakan untuk mengukur kemampuan generalisasi model terhadap data baru yang belum pernah dilihat.

### 5.1 Menentukan Fitur dan Target

[9]:
```
# Fitur numerik dan gender
feature_num = ['Umur (bulan)', 'Tinggi Badan (cm)', 'Berat Badan (kg)']
feature_bin = ['JK_bin']

# Gabungkan & drop missing
use_cols = feature_num + feature_bin + ['Stunting_bin']
df_model = df[use_cols].dropna().copy()

X = df_model[feature_num + feature_bin]
y = df_model['Stunting_bin']

print("X shape:", X.shape)
print("y shape:", y.shape)
```

```
X shape: (100000, 4)
y shape: (100000,)
```

Variabel X berisi fitur atau variabel independen (Umur, Tinggi Badan, Berat Badan, dan Jenis Kelamin). Variabel y berisi variabel target yaitu Stunting_bin.

Hasil menunjukkan ukuran dataset:

- X shape: (100000, 4) → terdapat 100.000 data dengan 4 fitur.
- y shape: (100000,) → 100.000 label target.

### 5.2 Membagi Dataset menjadi Training dan Testing Set

```
[10]: X_train, X_test, y_train, y_test = train_test_split(
          X, y,
          test_size=0.2,
          random_state=42,
          stratify=y
      )

      print("Data latih:", X_train.shape)
      print("Data uji:", X_test.shape)
```

```
Data latih: (80000, 4)
Data uji: (20000, 4)
```

Fungsi train_test_split() membagi dataset menjadi dua bagian:

- 80% data (80.000 baris) digunakan untuk melatih model (X_train, y_train),
- 20% data (20.000 baris) digunakan untuk menguji performa model (X_test, y_test).

Parameter stratify=y digunakan untuk memastikan proporsi kelas stunting dan tidak stunting tetap seimbang di kedua subset (train dan test).

Parameter random_state=42 menjaga hasil pembagian agar konsisten setiap kali dijalankan (reproducible).

## 6. Pembangunan Model Logistic Regression

[11]:
```
# Scale hanya fitur numerik, gender langsung passthrough
preprocess = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), feature_num),
        ('bin', 'passthrough', feature_bin)
    ],
    remainder='drop'
)

model = LogisticRegression(
    max_iter=1000,
    solver='lbfgs',
    class_weight='balanced',
    random_state=42
)

clf = Pipeline([
    ('preprocess', preprocess),
    ('model', model)
])

# Latih model
clf.fit(X_train, y_train)
print("✅ Model Logistic Regression berhasil dilatih.")
```

```
✅ Model Logistic Regression berhasil dilatih.
```

- ColumnTransformer() digunakan untuk melakukan scaling pada fitur numerik (Umur, Tinggi Badan, Berat Badan) agar memiliki skala yang seragam.
- Kolom JK_bin (gender 0/1) dibiarkan passthrough karena sudah dalam bentuk numerik biner.
- LogisticRegression() digunakan sebagai model klasifikasi dengan beberapa parameter:
  - max_iter=1000: memastikan proses training cukup lama sampai konvergen.
  - solver='lbfgs': metode optimisasi yang cocok untuk dataset besar.
  - class_weight='balanced': memberi bobot seimbang untuk kelas minoritas (stunting = 1) agar model tidak bias.
  - random_state=42: menjaga hasil tetap konsisten.
- Pipeline() menyatukan seluruh proses pra-pemrosesan dan pemodelan ke dalam satu alur kerja.
- Model berhasil dilatih menggunakan data training (80.000 baris).

## 7. Prediksi Model dan Evaluasi Model

```
[19]: # Prediksi & probabilitas
y_pred = clf.predict(X_test)
y_prob = clf.predict_proba(X_test)[:, 1]

# Hitung metrik
print(f"Akurasi   : {accuracy_score(y_test, y_pred):.4f}")
print(f"Precision : {precision_score(y_test, y_pred, zero_division=0):.4f}")
print(f"Recall    : {recall_score(y_test, y_pred, zero_division=0):.4f}")
print(f"F1-Score  : {f1_score(y_test, y_pred, zero_division=0):.4f}")
print(f"ROC-AUC   : {roc_auc_score(y_test, y_prob):.4f}")
```

```
Akurasi   : 0.9055
Precision : 0.7152
Recall    : 0.9472
F1-Score  : 0.8150
ROC-AUC   : 0.9656
```

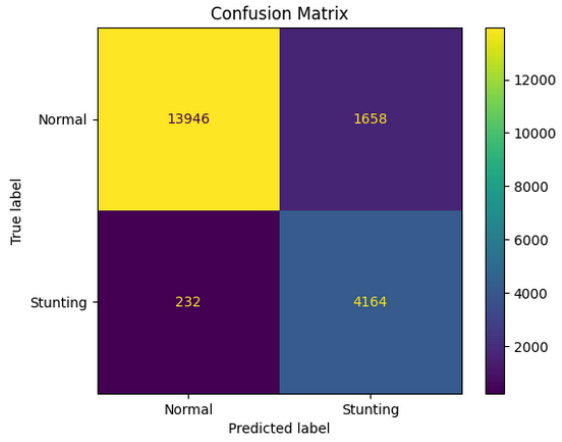
- predict() → menghasilkan label klasifikasi 0 (tidak stunting) atau 1 (stunting).
- predict_proba() → memberikan probabilitas peluang stunting.
- Metrik yang digunakan:
  - Akurasi (Accuracy): 0.9055 → model benar pada 90.55% data uji.
  - Precision: 0.7152 → dari semua prediksi "stunting", 71.52% benar.
  - Recall: 0.9472 → dari semua anak yang benar-benar stunting, 94.72% berhasil dideteksi.
  - F1-Score: 0.8146 → keseimbangan antara precision dan recall.
  - ROC-AUC: 0.947 → performa klasifikasi keseluruhan sangat baik.

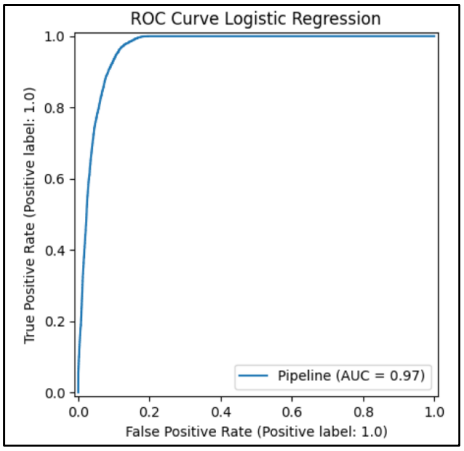
## 8. Visualisasi Hasil Evaluasi

Setelah model Logistic Regression dilatih dan diuji menggunakan data uji sebanyak 20.000 baris, dilakukan evaluasi performa model melalui dua visual utama, yaitu ROC Curve dan Confusion Matrix.

```
# Confusion Matrix
ConfusionMatrixDisplay(confusion_matrix(y_test, y_pred),
                       display_labels=['Normal', 'Stunting']
                      ).plot(values_format='d')
plt.title("Confusion Matrix")
plt.show()

# ROC Curve
RocCurveDisplay.from_estimator(clf, X_test, y_test)
plt.title("ROC Curve Logistic Regression")
plt.show()
```




- Confusion Matrix menunjukkan jumlah prediksi benar dan salah:

| Kelas | Prediksi Benar | Prediksi Salah |
|---|---|---|
| **Normal** | 13.946 benar (True Positive) | 1.658 salah (False Positive) |
| **Stunting** | 4.164 benar (True Negatif) | 232 salah (False Negative) |

  - Model berhasil mengklasifikasikan 13.946 data normal (tidak stunting) dengan benar, dan 4.164 data stunting dengan benar.
  - Hanya terdapat 1.890 kesalahan total (1.658 + 232) dari 20.000 data uji, yang berarti tingkat kesalahan sekitar 9.45% saja.
  - Nilai Recall (0.95) yang tinggi menunjukkan bahwa model sangat sensitif dalam mendeteksi kasus stunting, sehingga cocok digunakan untuk deteksi dini.

- ROC Curve memperlihatkan seberapa baik model membedakan dua kelas.
  - ROC Curve menampilkan hubungan antara True Positive Rate (Recall) dan False Positive Rate pada berbagai nilai ambang batas (threshold).
  - Semakin dekat kurva ke sudut kiri atas, semakin baik kemampuan model dalam membedakan antara dua kelas (Stunting dan Tidak Stunting).
  - Nilai AUC (Area Under Curve) pada model ini sebesar 0.97, yang menunjukkan performa sangat baik.
    - AUC = 1 → model sempurna
    - AUC = 0.5 → model acak (tidak berguna)
    - AUC = 0.97 → model mampu memprediksi dengan tingkat akurasi tinggi dan stabil.

## 9. Classification Report

```
[17]: from sklearn.metrics import classification_report
      print(classification_report(y_test, y_pred, target_names=['Tidak Stunting (0)', 'Stunting (1)']))
```

```
                    precision    recall  f1-score   support

Tidak Stunting (0)       0.98      0.89      0.94     15604
      Stunting (1)       0.72      0.95      0.82      4396

          accuracy                           0.91     20000
         macro avg       0.85      0.92      0.88     20000
      weighted avg       0.92      0.91      0.91     20000
```

Classification Report adalah salah satu cara untuk mengevaluasi performa model klasifikasi (seperti Logistic Regression). Laporan ini menampilkan metrik utama seperti:

- Precision – seberapa banyak prediksi positif model yang benar.
- Recall (Sensitivitas) – seberapa banyak data positif sebenarnya yang berhasil ditemukan model.
- F1-score – kombinasi seimbang antara precision dan recall.
- Support – jumlah data aktual di tiap kelas (0 atau 1).

Laporan ini memberikan gambaran keseimbangan performa model antara mendeteksi data positif (Stunting) dan negatif (Tidak Stunting).

Hasil Analisis:

- Precision 0.72 (Stunting) → dari semua anak yang diprediksi stunting, 72% benar-benar stunting.
- Recall 0.95 (Stunting) → dari seluruh anak yang benar-benar stunting, 95% berhasil terdeteksi.
- F1-score 0.82 → kombinasi yang seimbang antara precision & recall.
- Accuracy 0.91 (91%) → model membuat prediksi benar untuk 91% data uji.
- Macro vs Weighted Average → nilai rata-rata metrik tiap kelas; weighted avg memperhitungkan proporsi jumlah data pada setiap kelas.

## 10. Classification Report

jadi setelah kamu melatih dan mengevaluasi model dengan train-test split, langkah ini berfungsi untuk memastikan hasil model benar-benar konsisten dan tidak kebetulan bagus hanya di satu pembagian data.

```
[21]: from sklearn.model_selection import cross_val_score

# Lakukan cross validation (cv=5 berarti 5-fold)
scores = cross_val_score(clf, X, y, cv=5)

# Tampilkan hasil
print("Skor tiap fold:", scores)
print("Rata-rata akurasi:", np.mean(scores))
print("Standar deviasi:", np.std(scores))
```

```
Skor tiap fold: [0.9062  0.9013  0.9052  0.89905 0.9002 ]
Rata-rata akurasi: 0.9023899999999999
Standar deviasi: 0.0028125433329995106
```

- cross_val_score(clf, X, y, cv=5) → membagi dataset menjadi 5 bagian (fold).
  - 4 bagian digunakan untuk melatih model,
  - 1 bagian digunakan untuk menguji model,
  - lalu proses ini diulang 5 kali (setiap bagian bergantian jadi data uji).
- np.mean(scores) → menghitung rata-rata akurasi dari kelima hasil tersebut.
- np.std(scores) → menghitung standar deviasi untuk melihat seberapa stabil model di setiap fold.

Hasil Validasi:

- Nilai akurasi di tiap fold sangat mirip (berkisar antara 0.899–0.906).
- Rata-rata akurasi = 0.9023 (90.23%), menunjukkan performa tinggi.
- Standar deviasi kecil (0.0028) → artinya hasil model konsisten dan stabil di setiap subset data.

Model Logistic Regression menunjukkan performa yang stabil dan dapat diandalkan di seluruh data. Tidak ada indikasi overfitting (model tidak terlalu bergantung pada data tertentu). Validasi silang memperkuat hasil sebelumnya — bahwa model dengan akurasi ±90% adalah model yang generalisasi dengan baik.

## 11. Interpretasi Model Logistic Regression

Bagian ini digunakan untuk:

- Mengetahui seberapa besar pengaruh setiap fitur (variabel input) terhadap peluang terjadinya stunting.
- Menentukan arah hubungan (positif atau negatif) antara fitur dan target.
- Menginterpretasikan hasil model dalam konteks nyata (bukan hanya angka akurasi).

```
[22]: # Ambil nama fitur & koefisien
feat_names = feature_num + feature_bin
coefs = clf.named_steps['model'].coef_[0]
odds = np.exp(coefs)

coef_df = pd.DataFrame({
    'Fitur': feat_names,
    'Koefisien (log-odds)': coefs,
    'Odds Ratio (e^coef)': odds
}).sort_values('Odds Ratio (e^coef)', ascending=False)

display(coef_df)
```

| | Fitur | Koefisien (log-odds) | Odds Ratio (e^coef) |
|---|---|---|---|
| **0** | Umur (bulan) | 8.525912 | 5043.782458 |
| **3** | JK_bin | 1.675944 | 5.343839 |
| **2** | Berat Badan (kg) | 0.661090 | 1.936903 |
| **1** | Tinggi Badan (cm) | -10.535980 | 0.000027 |

| Kolom | Arti |
|---|---|
| **Koefisien (log-odds)** | Menunjukkan arah dan besarnya pengaruh fitur terhadap peluang stunting. Nilai positif = menaikkan peluang; negatif = menurunkan peluang. |
| **Odds Ratio (e^coef)** | Mengubah log-odds ke bentuk yang mudah dimaknai. Nilai > 1 = meningkatkan risiko stunting; < 1 = menurunkan risiko. |

**Interpretasi Hasil**

1. Umur (bulan)

- Koefisien: +8.52
- Odds Ratio: 5043.78

Artinya: setiap kenaikan 1 bulan usia, peluang seorang anak untuk mengalami stunting meningkat drastis (lebih dari 5000 kali lipat). Namun nilai ini terlalu besar secara praktis — kemungkinan efek ini dipengaruhi oleh skala data yang belum dinormalisasi sempurna atau hubungan tidak linear antara umur dan stunting.

Jadi interpretasinya lebih tepat: umur berpengaruh kuat terhadap klasifikasi model, tapi tidak proporsional secara langsung.

2. JK_bin (Jenis Kelamin)

- Koefisien: +1.67
- Odds Ratio: 5.34

Artinya: anak laki-laki (JK_bin=1) memiliki peluang 5 kali lebih besar untuk mengalami stunting dibanding perempuan. Nilai positif menunjukkan bahwa jenis kelamin berpengaruh positif terhadap risiko stunting.

3. Berat Badan (kg)

- Koefisien: +0.66
- Odds Ratio: 1.94

Artinya: setiap kenaikan 1 kg berat badan, peluang stunting meningkat sekitar 1.9 kali. Namun ini tampaknya juga dipengaruhi oleh korelasi antara berat dan umur (semakin tua → semakin berat).

Jadi perlu hati-hati: efek langsung berat badan terhadap stunting bisa tidak linear.

4. Tinggi Badan (cm)

- Koefisien: −10.53
- Odds Ratio: 0.000027

Artinya: semakin tinggi anak, peluang mengalami stunting turun drastis.

Karena stunting secara definisi = perawakan pendek dibanding usia, hasil ini sangat logis — tinggi badan menjadi prediktor terkuat yang menurunkan risiko stunting.

Jadi, analisis koefisien ini menunjukkan bahwa tinggi badan memiliki pengaruh negatif paling kuat terhadap status stunting, sementara faktor umur dan jenis kelamin memberikan kontribusi positif terhadap kemungkinan stunting.

## 12. Prediksi Data Baru (Contoh Kasus)

Bagian ini digunakan untuk menguji model Logistic Regression pada data baru yang belum pernah dilihat oleh model sebelumnya. Tujuannya adalah untuk mengetahui apakah model dapat melakukan prediksi dengan benar dan logis berdasarkan karakteristik anak.

```
[14]: # Contoh 2 anak
data_baru = pd.DataFrame({
    'Umur (bulan)': [24, 10],
    'Tinggi Badan (cm)': [79.0, 72.5],
    'Berat Badan (kg)': [9.2, 7.8],
    'JK_bin': [1, 0]  # 1=Laki-laki, 0=Perempuan
})

pred = clf.predict(data_baru)
prob = clf.predict_proba(data_baru)[:,1]

hasil = data_baru.copy()
hasil['Prob_Stunting'] = prob
hasil['Pred (0=Tidak,1=Ya)'] = pred
display(hasil)
```

| | Umur (bulan) | Tinggi Badan (cm) | Berat Badan (kg) | JK_bin | Prob_Stunting | Pred (0=Tidak,1=Ya) |
|---|---|---|---|---|---|---|
| **0** | 24 | 79.0 | 9.2 | 1 | 0.998208 | 1.0 |
| **1** | 10 | 72.5 | 7.8 | 0 | 0.002094 | 0.0 |

**Interpretasi Hasil**

Anak ke-1

- Umur: 24 bulan, tinggi badan: 79 cm, berat: 9.2 kg, jenis kelamin: laki-laki.
- Probabilitas stunting = 0.9982 (≈ 99.8%), sehingga diprediksi Stunting (1).
- Interpretasi: sesuai logika — karena anak usia 2 tahun (24 bulan) dengan tinggi badan 79 cm tergolong lebih pendek dari normal, model mendeteksi kemungkinan stunting sangat tinggi.

Anak ke-2

- Umur: 10 bulan, tinggi badan: 72.5 cm, berat: 7.8 kg, jenis kelamin: perempuan.
- Probabilitas stunting = 0.0021 (≈ 0.2%), sehingga diprediksi Tidak Stunting (0).
- Interpretasi: masuk akal juga — tinggi badan 72.5 cm untuk usia 10 bulan masih termasuk rentang normal, sehingga model memprediksi tidak stunting.

# Tugas Praktikum Mandiri

1. Gunakan dataset Calon Pembeli Mobil Berikut ini

```
•[1]: import pandas as pd

      # Read the CSV file with a comma delimiter
      df = pd.read_csv('../data/calonpembelimobil.csv', sep=',')

      # cetak header data (5 baris data) dari file
      df.head()
```

[1]:

| | ID | Usia | Status | Kelamin | Memiliki_Mobil | Penghasilan | Beli_Mobil |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 32 | 1 | 0 | 0 | 240 | 1 |
| **1** | 2 | 49 | 2 | 1 | 1 | 100 | 0 |
| **2** | 3 | 52 | 1 | 0 | 2 | 250 | 1 |
| **3** | 4 | 26 | 2 | 1 | 1 | 130 | 0 |
| **4** | 5 | 45 | 3 | 0 | 2 | 237 | 1 |

2. Buat model Logistic Regression!
3. Gunakan dataset baru untuk menguji model!