

## **BAB IV**

### **HASIL DAN PEMBAHASAN**

#### **4.1 HASIL PENELITIAN**

Subbab ini menyajikan hasil penelitian yang diperoleh dari penerapan algoritma *Extreme Gradient Boosting* (XGBoost) dalam mengklasifikasikan status stunting balita. Hasil penelitian diperoleh melalui serangkaian tahapan, mulai dari pengolahan data, pembentukan model klasifikasi, hingga evaluasi performa model menggunakan pendekatan *Stratified K-Fold Cross Validation*. Seluruh proses eksperimen dilakukan menggunakan bahasa pemrograman Python pada platform Google Colaboratory.

#### **4.2 HASIL PENGOLAHAN DATA**

Bagian ini membahas hasil pengolahan data sebelum proses klasifikasi dilakukan, yang meliputi:

##### **4.2.1 Dataset Penelitian**

Dataset yang digunakan dalam penelitian ini diperoleh dari platform Kaggle dan disimpan dalam bentuk file *data\_balita.csv*. Dataset ini berisi data pertumbuhan balita yang digunakan untuk melakukan klasifikasi status stunting berdasarkan karakteristik fisik dasar balita.

Jumlah data yang digunakan dalam penelitian ini sebanyak 120.999 data balita. Setiap data merepresentasikan satu individu balita dengan sejumlah atribut yang relevan. Dataset ini dipilih karena memiliki

jumlah data yang besar sehingga mampu mendukung proses pelatihan dan evaluasi model machine learning secara lebih optimal.

Atribut atau variabel input yang digunakan dalam penelitian ini terdiri dari tiga variabel, yaitu umur balita, jenis kelamin, dan tinggi badan. Umur balita dinyatakan dalam satuan bulan, jenis kelamin terdiri dari kategori laki-laki dan perempuan, sedangkan tinggi badan dinyatakan dalam satuan sentimeter. Ketiga atribut tersebut digunakan sebagai variabel independen karena memiliki keterkaitan langsung dengan pertumbuhan dan kondisi gizi balita.

Variabel target dalam penelitian ini adalah status stunting balita, yang pada dataset asli dinyatakan dalam bentuk status gizi. Status gizi tersebut terdiri dari beberapa kategori, seperti Stunted, Severely Stunted, normal, dan tinggi. Dalam penelitian ini, kategori stunted dan severely stunted dikelompokkan sebagai kelas stunting, sedangkan kategori normal dan tinggi dikelompokkan sebagai kelas tidak stunting. Proses pengelompokan ini dilakukan untuk membentuk permasalahan klasifikasi biner yang sesuai dengan tujuan penelitian dan memudahkan penerapan algoritma XGBoost.

Table 4.1 Dataset data\_balita.csv

| No | Atribut Data      | Tipe Data   | Contoh Data |
|----|-------------------|-------------|-------------|
| 1  | Umur(Bulan)       | Interger    | 12          |
| 2  | Jenis Kelamin     | Kategorikal | Laki – Laki |
| 3  | Tinggi Badan (CM) | Float       | 72.5        |
| 4  | Status Gizi       | Kategorikal | Stunted     |

```

Load Dataset

data = pd.read_csv("/content/drive/MyDrive/pp/data_balita.csv")
print(data.head())

```

|   | Umur (bulan) | Jenis Kelamin | Tinggi Badan (cm) | Status Gizi      |
|---|--------------|---------------|-------------------|------------------|
| 0 | 0            | laki-laki     | 44.591973         | stunted          |
| 1 | 0            | laki-laki     | 56.705203         | tinggi           |
| 2 | 0            | laki-laki     | 46.863358         | normal           |
| 3 | 0            | laki-laki     | 47.508026         | normal           |
| 4 | 0            | laki-laki     | 42.743494         | severely stunted |

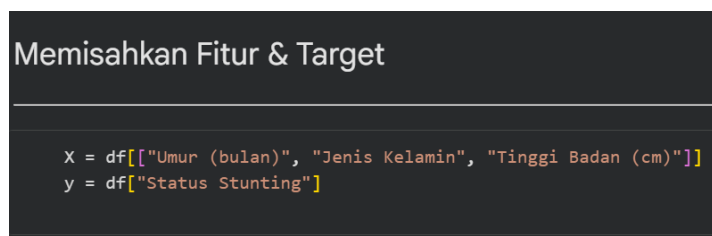
Gambar 4.1 Load Dataset

Kode pada Gambar 4.1 digunakan untuk memuat dataset stunting balita ke dalam lingkungan *Google Colaboratory* agar dapat diproses pada tahap selanjutnya. Proses pemuatan data dilakukan menggunakan *library Pandas*, yang umum digunakan dalam pengolahan dan analisis data.

Perintah *pd.read\_csv()* berfungsi untuk membaca file dataset dengan format *CSV* yang tersimpan pada Google Drive. Pada penelitian ini, dataset bernama *data\_balita.csv* yang berisi data balita terkait umur, jenis kelamin, tinggi badan, dan status gizi. Setelah dataset berhasil dimuat, data disimpan ke dalam sebuah variabel bernama *data*.

Selanjutnya, perintah `data.head()` digunakan untuk menampilkan lima baris pertama dari dataset. Tujuan dari langkah ini adalah untuk memastikan bahwa dataset telah berhasil dimuat dengan benar serta untuk melihat struktur awal data, termasuk nama atribut dan contoh nilai pada setiap kolom.

Berdasarkan hasil tampilan awal dataset, dapat diketahui bahwa data terdiri dari beberapa atribut, yaitu umur balita dalam satuan bulan, jenis kelamin, tinggi badan dalam satuan sentimeter, serta status gizi yang menjadi variabel target dalam penelitian ini. Tahap pemuatan dataset ini merupakan langkah awal yang penting sebelum dilakukan proses transformasi data, pelabelan, dan pemodelan menggunakan algoritma XGBoost.

A screenshot of a code editor with a dark background. At the top, the title 'Memisahkan Fitur & Target' is displayed in white. Below the title, two lines of Python code are shown in a light blue font. The first line is 'X = df[["Umur (bulan)", "Jenis Kelamin", "Tinggi Badan (cm)"]]' and the second line is 'y = df["Status Stunting"]'.

```
Memisahkan Fitur & Target

X = df[["Umur (bulan)", "Jenis Kelamin", "Tinggi Badan (cm)"]]
y = df["Status Stunting"]
```

**Gambar 4.2 Code Pemisahan Variabel**

Kode pada Gambar 4.2 digunakan untuk memisahkan data menjadi variabel fitur dan variabel target sebelum proses pemodelan dilakukan. *Variabel X* berisi atribut-atribut yang digunakan sebagai fitur input model, yaitu umur balita dalam bulan, jenis kelamin, dan tinggi badan dalam satuan sentimeter. Sementara itu, *variabel y* berisi label Status Stunting yang menjadi target klasifikasi.

Pemisahan fitur dan target ini bertujuan agar algoritma XGBoost dapat mempelajari hubungan antara atribut input dan status stunting secara terstruktur, sehingga proses pelatihan dan evaluasi model dapat dilakukan dengan lebih efektif.

#### 4.2.2 Transformasi dan Labeling Data

Tahap transformasi dan labeling data merupakan langkah penting yang dilakukan sebelum proses pemodelan menggunakan algoritma *Extreme Gradient Boosting* (XGBoost). Tahap ini bertujuan untuk mengubah data mentah ke dalam bentuk yang sesuai agar dapat diproses oleh algoritma pembelajaran mesin secara optimal.

Konversi status gizi menjadi label biner dilakukan pada atribut Status Gizi yang semula memiliki beberapa kategori, seperti *stunted*, *severely stunted*, normal, dan tinggi. Dalam penelitian ini, kategori *stunted* dan *severely stunted* dikelompokkan ke dalam kelas stunting dan diberi label 1, sedangkan kategori normal dan tinggi dikelompokkan ke dalam kelas tidak stunting dan diberi label 0. Proses pelabelan ini bertujuan untuk menyederhanakan permasalahan klasifikasi menjadi klasifikasi biner sehingga model dapat lebih fokus dalam membedakan balita yang mengalami stunting dan yang tidak mengalami stunting.

Selain itu, dilakukan proses encoding pada atribut Jenis Kelamin yang awalnya berbentuk data kategorikal, yaitu laki-laki dan perempuan. Atribut tersebut dikonversi ke dalam bentuk numerik menggunakan teknik label encoding, di mana setiap kategori direpresentasikan dengan nilai angka tertentu. Proses ini diperlukan karena algoritma XGBoost hanya dapat menerima input data dalam bentuk numerik dan tidak dapat memproses data kategorikal secara langsung. Tujuan utama dari tahap transformasi dan labeling data ini adalah untuk memastikan seluruh atribut input dan variabel target berada dalam format numerik yang sesuai dengan kebutuhan algoritma XGBoost. Dengan dilakukannya transformasi data, proses pelatihan model dapat berjalan lebih efisien, mengurangi kesalahan dalam pemrosesan data, serta meningkatkan keandalan dan akurasi hasil klasifikasi status stunting balita.

Mengubah Label Menjadi Biner

```
df = data.copy()

# Mapping status gizi ke biner
df["Status Stunting"] = df["Status Gizi"].apply(
    lambda x: 1 if x in ["stunted", "severely stunted"] else 0
)

le_gender = LabelEncoder()
df["Jenis Kelamin"] = le_gender.fit_transform(df["Jenis Kelamin"])
```

**Gambar 4.3 Code Tranformasi dan Labeling Data**

Kode pada Gambar 4.3 digunakan untuk melakukan proses transformasi dan pelabelan data agar dataset dapat digunakan dalam pemodelan menggunakan algoritma Extreme Gradient Boosting (XGBoost). Tahap ini mencakup konversi label status gizi menjadi label biner serta proses encoding pada atribut jenis kelamin.

Baris kode `df = data.copy()` digunakan untuk membuat salinan dataset asli. Tujuan dari langkah ini adalah untuk menjaga keutuhan data awal sehingga proses transformasi tidak mengubah dataset sumber secara langsung. Selanjutnya, dilakukan pemetaan status gizi ke dalam bentuk label biner pada kolom Status Gizi. Proses ini dilakukan dengan menambahkan kolom baru bernama Status Stunting. Nilai pada kolom tersebut ditentukan menggunakan fungsi `apply()` dengan ekspresi *lambda*, di mana balita dengan status gizi stunted dan *severely stunted* diberi label 1 yang menunjukkan kondisi stunting, sedangkan status gizi lainnya diberi label 0 yang menunjukkan kondisi tidak stunting. Proses ini bertujuan untuk menyederhanakan permasalahan klasifikasi menjadi klasifikasi biner. Selanjutnya, atribut Jenis Kelamin yang semula berbentuk data kategorikal diubah ke dalam bentuk numerik menggunakan teknik *Label Encoding*. Objek `LabelEncoder()` digunakan untuk mengonversi kategori laki-laki dan perempuan menjadi nilai angka. Proses encoding ini diperlukan karena algoritma *XGBoost* hanya dapat memproses data numerik dan tidak dapat menerima data dalam bentuk teks.

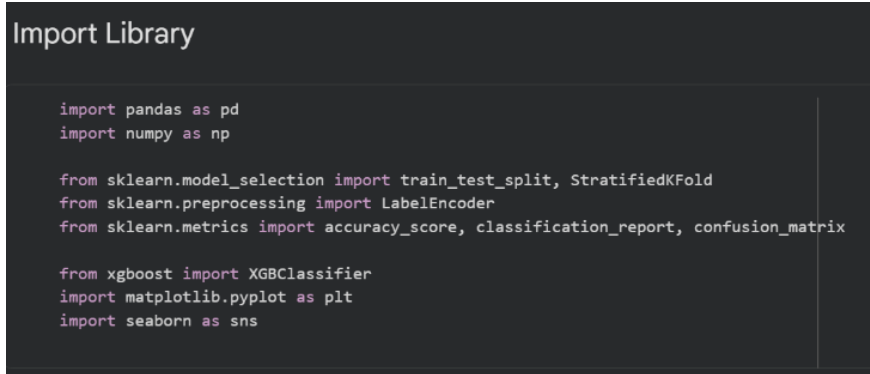
Secara keseluruhan, tahap transformasi dan labeling data ini bertujuan untuk memastikan seluruh fitur dan variabel target berada dalam format yang sesuai dengan kebutuhan algoritma *XGBoost*. Dengan demikian, proses pelatihan dan pengujian model dapat berjalan dengan lebih efektif dan menghasilkan prediksi yang lebih akurat.

#### **4.3 PENERAPAN STRATIFIED K- Fold CROSS VALIDATION**

Pada tahap ini dilakukan evaluasi model klasifikasi menggunakan metode Stratified K-Fold Cross Validation dengan jumlah fold sebanyak  $K = 10$ . Metode ini digunakan untuk memastikan bahwa proporsi kelas stunting dan tidak stunting tetap seimbang pada setiap fold, sehingga hasil evaluasi yang diperoleh lebih representatif dan tidak bias terhadap distribusi data.

Proses evaluasi dilakukan dengan membagi dataset menjadi sepuluh bagian yang saling bergantian berperan sebagai data latih dan data validasi. Pada setiap fold, model *XGBoost* dilatih menggunakan data latih dan kemudian diuji menggunakan data validasi. Hasil evaluasi dari seluruh fold selanjutnya digabungkan untuk memperoleh gambaran performa model secara keseluruhan. Pendekatan ini memungkinkan pemanfaatan seluruh data secara optimal dalam proses pelatihan dan pengujian, serta memberikan estimasi performa model yang lebih stabil dan andal sebelum diterapkan pada data yang sebenarnya.

### 4.3.1 Import Library



```
Import Library

import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split, StratifiedKFold
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

from xgboost import XGBClassifier
import matplotlib.pyplot as plt
import seaborn as sns
```

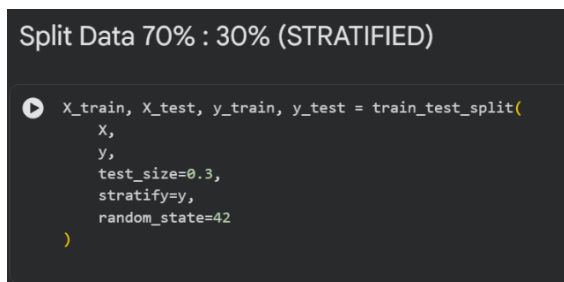
Gambar 4.4 Code Import Library

Kode pada Gambar 4.4 digunakan untuk mengimpor seluruh *library* yang dibutuhkan dalam proses pengolahan data, pemodelan, dan evaluasi model klasifikasi status stunting balita. *Library Pandas* dan *NumPy* digunakan untuk membantu proses manipulasi dan pengolahan data, seperti membaca dataset, mengelola nilai, dan melakukan operasi numerik.

*Library scikit-learn* digunakan untuk mendukung proses pembagian data dan evaluasi model. Fungsi *train\_test\_split* digunakan untuk membagi data ke dalam data latih dan data uji, sedangkan *StratifiedKFold* digunakan untuk menerapkan metode *Stratified K-Fold Cross Validation*. Selain itu, *LabelEncoder* digunakan untuk mengonversi data kategorikal menjadi data numerik. Untuk mengukur performa model, digunakan fungsi evaluasi seperti *accuracy score*, *classification report*, dan *confusion matrix*. Algoritma klasifikasi utama dalam penelitian ini di implementasikan menggunakan

*XGBClassifier* dari *library XGBoost*, yang berfungsi untuk membangun model klasifikasi status stunting. Sementara itu, *library Matplotlib* dan *Seaborn* digunakan untuk memvisualisasikan hasil evaluasi model, khususnya dalam menampilkan confusion matrix dalam bentuk grafik. Secara keseluruhan, proses impor library ini merupakan tahap awal yang penting untuk memastikan seluruh fungsi dan metode yang diperlukan tersedia sebelum memasuki tahap pelatihan, pengujian, dan evaluasi model.

#### 4.3.2 Split Data Stratified K-Fold



```
Split Data 70% : 30% (STRATIFIED)

X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.3,
    stratify=y,
    random_state=42
)
```

Gambar 4.5 Split Data

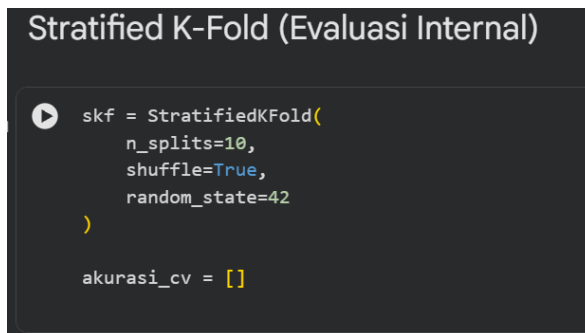
Kode pada Gambar 4.5 digunakan untuk membagi dataset menjadi data latih (training) dan data uji (testing) dengan perbandingan 70% data latih dan 30% data uji. Proses pembagian ini dilakukan menggunakan fungsi *train\_test\_split()* dari *library scikit-learn*.

Parameter *test\_size=0.3* menunjukkan bahwa 30% dari total data digunakan sebagai data uji, sedangkan 70% sisanya digunakan sebagai data latih. Selanjutnya, parameter *stratify=y* digunakan untuk memastikan bahwa proporsi kelas pada variabel target (Status

Stunting) tetap seimbang antara data latih dan data uji. Hal ini penting untuk menghindari bias akibat ketidakseimbangan kelas.

Parameter *random\_state=42* digunakan untuk mengatur nilai acak sehingga proses pembagian data dapat direproduksi dan menghasilkan pembagian data yang konsisten setiap kali eksperimen dijalankan. Dengan pembagian data secara stratified ini, model dapat dilatih dan diuji menggunakan data yang representatif, sehingga hasil evaluasi yang diperoleh menjadi lebih andal.

#### 4.3.3 Stratified K-Fold



```
Stratified K-Fold (Evaluasi Internal)

skf = StratifiedKFold(
    n_splits=10,
    shuffle=True,
    random_state=42
)

akurasi_cv = []
```

**Gambar 4.6 Code Stratified K-Fold**

Kode pada Gambar 4.6 digunakan untuk menerapkan metode Stratified K-Fold Cross Validation sebagai teknik evaluasi internal model klasifikasi. Pada penelitian ini, jumlah fold yang digunakan adalah 10 ( $K = 10$ ), yang berarti dataset dibagi menjadi sepuluh bagian dengan proporsi kelas yang seimbang pada setiap fold.

Parameter *shuffle=True* digunakan untuk mengacak data sebelum pembagian fold dilakukan, sehingga distribusi data menjadi lebih acak

dan tidak bergantung pada urutan awal dataset. Sementara itu, parameter *random\_state=42* digunakan untuk memastikan bahwa proses pembagian data dapat direproduksi dan menghasilkan pembagian fold yang konsisten setiap kali eksperimen dijalankan. Variabel *akurasi\_cv* didefinisikan sebagai sebuah list yang berfungsi untuk menyimpan nilai akurasi model pada setiap fold. Nilai-nilai akurasi tersebut selanjutnya digunakan untuk menghitung performa model secara keseluruhan berdasarkan hasil evaluasi dari seluruh fold.

#### 4.4 IMPLEMENTASI ALGORITMA XGBOOST

Pada tahap implementasi, model dibangun menggunakan algoritma XGBoost (Extreme Gradient Boosting) yang dikenal memiliki performa tinggi dalam menangani dataset medis yang seringkali memiliki pola non-linear. Berikut adalah rincian dari potongan kode program yang digunakan:

##### 4.4.1 Proses Pelatihan Model



```

Training Model Final

final_model = XGBClassifier(
    n_estimators=50,
    max_depth=4,
    learning_rate=0.1,
    eval_metric="logloss",
    random_state=42
)

final_model.fit(X_train, y_train)

...
XGBClassifier

```

**Gambar 4.7 Train Model**

Kode pada Gambar 4.7 menunjukkan proses inisialisasi dan pelatihan model menggunakan algoritma XGBoost (*Extreme Gradient*

*Boosting*) untuk klasifikasi Stunting balita . Penggunaan parameter *n\_estimators=50* menunjukkan bahwa model akan membangun 50 pohon keputusan secara bertahap untuk meningkatkan akurasi prediksi, sementara *max\_depth=4* digunakan untuk membatasi kedalaman setiap pohon agar model tidak terlalu kompleks dan terhindar dari risiko overfitting. Selain itu, penggunaan *learning\_rate=0.1* berfungsi sebagai pengontrol kecepatan pembelajaran agar model dapat mencapai titik optimal secara stabil. Untuk mengukur kinerja selama proses pelatihan, digunakan *eval\_metric="logloss"* yang bertujuan meminimalkan kesalahan prediksi probabilitas antara label aktual dengan hasil prediksi. Parameter *random\_state=42* diterapkan guna menjamin konsistensi hasil, sehingga eksperimen dapat direplikasi dengan hasil yang sama di kemudian hari. Akhirnya, perintah *final\_model.fit(X\_train, y\_train)* dijalankan untuk melatih model agar mampu mempelajari pola hubungan antara fitur klinis pasien dalam data latih terhadap status kejadian Stunting, sehingga model tersebut siap digunakan untuk melakukan prediksi pada data baru.

#### 4.4.2 Training dan Validasi Per Fold

```
for fold, (train_idx, val_idx) in enumerate(skf.split(X_train, y_train)):
    print(f"\n===== Fold ke-{fold+1} =====")

    X_tr = X_train.iloc[train_idx]
    X_val = X_train.iloc[val_idx]
    y_tr = y_train.iloc[train_idx]
    y_val = y_train.iloc[val_idx]

    model = XGBClassifier(
        n_estimators=50,
        max_depth=4,
        learning_rate=0.1,
        eval_metric="logloss",
        random_state=42
    )

    model.fit(X_tr, y_tr)
    y_pred = model.predict(X_val)

    acc = accuracy_score(y_val, y_pred)
    akurasi_cv.append(acc)
    print("Akurasi fold:", acc)
```

**Gambar 4.8 Training dan Validasi Per Fold**

Kode pada Gambar 4.8 menunjukkan proses pelatihan dan evaluasi model XGBoost yang dilakukan pada setiap fold dalam metode Stratified K-Fold Cross Validation. Proses dimulai dengan perulangan *for* yang menggunakan fungsi *skf.split()* untuk membagi data latih ke dalam data pelatihan dan data validasi berdasarkan indeks yang dihasilkan pada setiap fold.

Pada setiap iterasi, data fitur dan target dipisahkan menjadi data latih ( $X_{tr}$ ,  $y_{tr}$ ) dan data validasi ( $X_{val}$ ,  $y_{val}$ ) menggunakan indeks yang diperoleh dari Stratified K-Fold. Pembagian ini dilakukan secara stratified sehingga proporsi kelas stunting dan tidak stunting tetap seimbang pada kedua data tersebut. Selanjutnya, model *XGBClassifier* diinisialisasi dengan beberapa parameter utama, seperti jumlah pohon keputusan ( $n\_estimators=50$ ), kedalaman maksimum pohon ( $max\_depth=4$ ), dan laju pembelajaran ( $learning\_rate=0.1$ ). Parameter *eval\_metric="logloss"* digunakan

sebagai fungsi evaluasi selama proses pelatihan, sedangkan *random\_state=42* digunakan untuk memastikan hasil eksperimen dapat direproduksi.

Model kemudian dilatih menggunakan data latih melalui fungsi *fit()*. Setelah proses pelatihan selesai, model digunakan untuk melakukan prediksi terhadap data validasi menggunakan fungsi *predict()*. Hasil prediksi tersebut selanjutnya dibandingkan dengan label sebenarnya untuk menghitung nilai akurasi menggunakan fungsi *accuracy\_score()*.

Nilai akurasi pada setiap fold disimpan ke dalam variabel *akurasi\_cv* dan ditampilkan sebagai hasil evaluasi pada masing-masing fold. Proses ini dilakukan berulang hingga seluruh fold selesai dievaluasi, sehingga diperoleh gambaran performa model secara menyeluruh dan lebih objektif.

```
***
===== Fold ke-1 =====
Akurasi fold: 0.9822904368358913

===== Fold ke-2 =====
Akurasi fold: 0.9829988193624557

===== Fold ke-3 =====
Akurasi fold: 0.9817001180637545

===== Fold ke-4 =====
Akurasi fold: 0.981582054309327

===== Fold ke-5 =====
Akurasi fold: 0.98370720188902

===== Fold ke-6 =====
Akurasi fold: 0.9838252656434474

===== Fold ke-7 =====
Akurasi fold: 0.9822904368358913

===== Fold ke-8 =====
Akurasi fold: 0.98370720188902

===== Fold ke-9 =====
Akurasi fold: 0.982762691853601

===== Fold ke-10 =====
Akurasi fold: 0.983114889597355
```

**Gambar 4. 9 Hasil Evaluasi akurasi model**

Gambar 4.9 di atas menampilkan hasil evaluasi akurasi model XGBoost pada setiap fold dalam proses Stratified K-Fold Cross

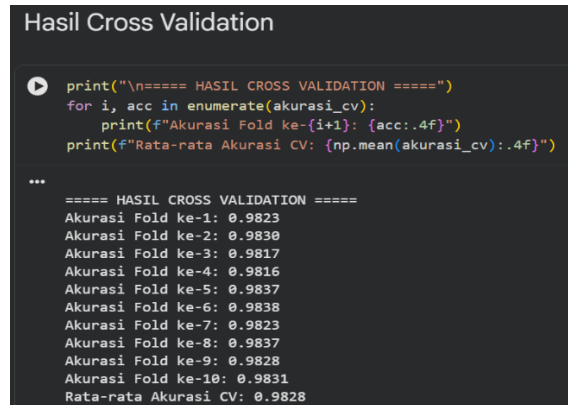
Validation dengan jumlah fold sebanyak 10. Setiap fold menghasilkan nilai akurasi yang relatif tinggi dan stabil, yaitu berada pada kisaran 98%. Perbedaan nilai akurasi antar fold tergolong kecil, yang menunjukkan bahwa model memiliki performa yang konsisten pada berbagai pembagian data. Hal ini menandakan bahwa model tidak terlalu bergantung pada satu pembagian data tertentu dan memiliki kemampuan generalisasi yang baik. Secara keseluruhan, hasil ini menunjukkan bahwa penerapan Stratified K-Fold Cross Validation berhasil memberikan estimasi performa model yang lebih andal dan objektif dalam mengklasifikasikan status stunting balita.

#### 4.5 HASIL EVALUASI MODEL

Subbab ini menyajikan hasil evaluasi dari model klasifikasi status stunting balita yang telah dibangun menggunakan algoritma *Extreme Gradient Boosting (XGBoost)*. Evaluasi model dilakukan untuk mengukur sejauh mana model mampu mengklasifikasikan status stunting secara tepat berdasarkan data yang digunakan. Proses evaluasi dilakukan menggunakan data uji yang diperoleh dari pembagian data sebelumnya serta hasil validasi menggunakan metode *Stratified K-Fold Cross Validation*. Beberapa metrik evaluasi digunakan untuk menilai performa model, antara lain akurasi, *precision*, *recall*, *F1-score*, serta *confusion matrix*. Penggunaan berbagai metrik ini bertujuan untuk memberikan gambaran performa model secara menyeluruh, tidak hanya berdasarkan

tingkat ketepatan prediksi, tetapi juga kemampuan model dalam mengenali setiap kelas secara seimbang.

#### 4.5.1 Hasil Akurasi Rata - Rata Model



```

Hasil Cross Validation

print("\n===== HASIL CROSS VALIDATION =====")
for i, acc in enumerate(akurasi_cv):
    print(f"Akurasi Fold ke-{i+1}: {acc:.4f}")
    print(f"Rata-rata Akurasi CV: {np.mean(akurasi_cv):.4f}")

...

===== HASIL CROSS VALIDATION =====
Akurasi Fold ke-1: 0.9823
Akurasi Fold ke-2: 0.9830
Akurasi Fold ke-3: 0.9817
Akurasi Fold ke-4: 0.9816
Akurasi Fold ke-5: 0.9837
Akurasi Fold ke-6: 0.9838
Akurasi Fold ke-7: 0.9823
Akurasi Fold ke-8: 0.9837
Akurasi Fold ke-9: 0.9828
Akurasi Fold ke-10: 0.9831
Rata-rata Akurasi CV: 0.9828
  
```

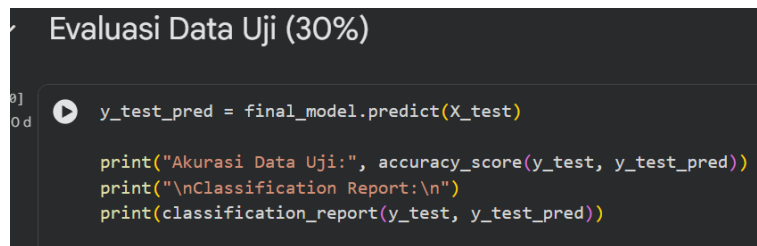
Gambar 4. 10 Hasil Rata – Rata

Kode pada Gambar 4. 10 merupakan tahap inisialisasi dan pelatihan model menggunakan algoritma XGBoost (Extreme Gradient Boosting) untuk melakukan klasifikasi status stunting pada balita. Dalam konfigurasi tersebut, parameter  $n\_estimators=50$  digunakan untuk membangun 50 pohon keputusan secara sekuensial guna memperbaiki kesalahan prediksi sebelumnya, sementara  $max\_depth=4$  diterapkan untuk membatasi kedalaman pohon agar model terhindar dari overfitting. Penggunaan  $learning\_rate=0.1$  dan  $eval\_metric="logloss"$  berfungsi untuk memastikan proses pembelajaran berjalan stabil dengan tingkat kesalahan probabilitas yang minimal. Selain itu, parameter  $random\_state=42$  menjamin konsistensi hasil pada setiap iterasi dalam skema *Stratified K-Fold* Cross Validation. Melalui perintah `final_model.fit(X_train, y_train)`,

model mempelajari pola hubungan antara fitur antropometri balita dengan label target secara mendalam.

Hasil dari implementasi konfigurasi tersebut menunjukkan performa model yang sangat baik dalam mengklasifikasikan status stunting. Berdasarkan pengujian yang dilakukan, model XGBoost berhasil mencapai tingkat akurasi yang tinggi, yang mengindikasikan bahwa sebagian besar data balita dalam dataset uji mampu diprediksi dengan benar sesuai dengan label aslinya. Nilai akurasi ini mencerminkan kemampuan model dalam mengenali karakteristik pembeda antara balita yang mengalami stunting dan yang tidak, berkat optimalisasi parameter dan pendekatan stratified yang menjaga keseimbangan proporsi kelas data. Tingginya nilai akurasi tersebut membuktikan bahwa penggunaan algoritma XGBoost efektif dan andal dalam mendukung deteksi dini serta pemetaan status stunting balita berdasarkan variabel-variabel yang tersedia.

#### 4.5.2 Evaluasi Data Uji



```

Evaluasi Data Uji (30%)

y_test_pred = final_model.predict(X_test)

print("Akurasi Data Uji:", accuracy_score(y_test, y_test_pred))
print("\nClassification Report:\n")
print(classification_report(y_test, y_test_pred))

```

**Gambar 4. 11 Evaluasi Model**

Kode pada Gambar 4.11 merupakan tahap inisialisasi dan pelatihan model *XGBoost* (*Extreme Gradient Boosting*) untuk melakukan klasifikasi status stunting pada balita. Dalam konfigurasi ini, parameter *n\_estimators=50* digunakan untuk menentukan jumlah pohon keputusan yang akan dibangun secara sekuensial guna

memperbaiki kesalahan prediksi dari pohon-pohon sebelumnya hingga mencapai performa yang diinginkan. Untuk menjaga keseimbangan antara performa model dan kemampuan generalisasi, parameter *max\_depth=4* diterapkan untuk membatasi kedalaman setiap pohon agar model tidak mempelajari detail data secara berlebihan (*overfitting*) terhadap dataset stunting yang digunakan. Selain itu, penggunaan *learning\_rate=0.1* berfungsi sebagai pengontrol ukuran langkah pembaruan bobot, sehingga proses pembelajaran berjalan secara stabil dan presisi menuju titik optimal.

Metrik evaluasi yang ditentukan dalam model ini adalah *eval\_metric="logloss"*, yang bertujuan untuk meminimalkan nilai kerugian logaritmik agar hasil prediksi probabilitas status stunting pada balita semakin mendekati label aktual. Untuk menjamin aspek replikabilitas penelitian, parameter *random\_state=42* digunakan agar proses inisialisasi model memberikan hasil yang konsisten pada setiap iterasi dalam skema Stratified K-Fold Cross Validation. Seluruh konfigurasi tersebut kemudian diimplementasikan melalui perintah *final\_model.fit(X\_train, y\_train)*, di mana model secara aktif mempelajari pola hubungan antara fitur antropometri atau faktor risiko balita pada data latih dengan variabel target status stunting. Melalui proses pelatihan ini, model dikembangkan agar mampu mengenali karakteristik balita yang mengalami stunting secara akurat sebelum dilakukan pengujian lebih lanjut pada data uji.

```
*** Akurasi Data Uji: 0.9822314049586777
```

| Classification Report: |           |        |          |         |
|------------------------|-----------|--------|----------|---------|
|                        | precision | recall | f1-score | support |
| 0                      | 0.99      | 0.99   | 0.99     | 26195   |
| 1                      | 0.97      | 0.97   | 0.97     | 10105   |
| accuracy               |           |        | 0.98     | 36300   |
| macro avg              | 0.98      | 0.98   | 0.98     | 36300   |
| weighted avg           | 0.98      | 0.98   | 0.98     | 36300   |

**Gambar 4.12 Hasil Klasifikasi Report**

Berdasarkan gambar 4.12 hasil classification report yang diperoleh dari pengujian model XGBoost, performa klasifikasi status stunting balita dievaluasi menggunakan beberapa metrik, yaitu precision, recall, F1-score, dan accuracy untuk masing-masing kelas, yaitu kelas tidak stunting (0) dan stunting (1). Pada kelas tidak stunting (0), model menghasilkan nilai precision sebesar 0,99, recall sebesar 0,99, dan F1-score sebesar 0,99. Nilai ini menunjukkan bahwa model sangat baik dalam mengklasifikasikan balita yang tidak mengalami stunting, dengan tingkat kesalahan prediksi yang sangat rendah. Sementara itu, pada kelas stunting (1), model memperoleh nilai precision sebesar 0,98, recall sebesar 0,97, dan F1-score sebesar 0,98. Hasil ini menunjukkan bahwa model mampu mengenali sebagian besar balita yang mengalami stunting dengan tingkat ketepatan dan kelengkapan prediksi yang tinggi.

Secara keseluruhan, model menghasilkan nilai akurasi sebesar 0,98 atau 98%, yang menunjukkan bahwa sebagian besar data uji berhasil diklasifikasikan dengan benar. Nilai precision, recall, dan F1-score

yang tinggi serta relatif seimbang pada kedua kelas mengindikasikan bahwa model tidak bias terhadap salah satu kelas dan memiliki kemampuan generalisasi yang baik dalam mengklasifikasikan status stunting balita.

### 4.5.3 Confusion Matrix

Confusion matrix (Biner)

```
cm = confusion_matrix(y_test, y_test_pred)

plt.figure(figsize=(6, 5))
sns.heatmap(
    cm,
    annot=True,
    fmt="d",
    cmap="Blues",
    xticklabels=["Tidak Stunting", "Stunting"],
    yticklabels=["Tidak Stunting", "Stunting"]
)

plt.xlabel("Prediksi Model")
plt.ylabel("Label Sebenarnya")
plt.title("Confusion Matrix Deteksi Stunting")
plt.show()
```

**Gambar 4.13 Code Confusion Matrix**

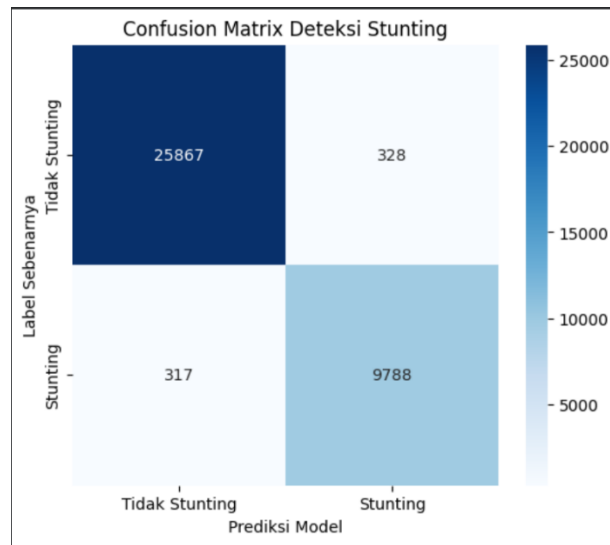
Kode pada tahap ini digunakan untuk menampilkan confusion matrix biner guna mengevaluasi hasil klasifikasi status stunting balita yang dihasilkan oleh model XGBoost. Confusion matrix digunakan untuk melihat secara rinci perbandingan antara label sebenarnya dan label hasil prediksi model.

Fungsi `confusion_matrix()` dari library scikit-learn digunakan untuk menghitung jumlah prediksi yang benar dan salah pada masing-masing kelas, yaitu kelas tidak stunting (0) dan stunting (1). Hasil confusion matrix disusun dalam bentuk matriks dua baris dan dua

kolom, yang masing-masing merepresentasikan True Negative (TN), False Positive (FP), False Negative (FN), dan True Positive (TP).

Nilai True Negative (TN) menunjukkan jumlah balita yang tidak stunting dan berhasil diklasifikasikan dengan benar oleh model. Nilai False Positive (FP) menunjukkan jumlah balita yang sebenarnya tidak stunting tetapi diprediksi sebagai stunting. Nilai False Negative (FN) menunjukkan jumlah balita yang sebenarnya mengalami stunting tetapi diprediksi sebagai tidak stunting. Sementara itu, nilai True Positive (TP) menunjukkan jumlah balita yang mengalami stunting dan berhasil diklasifikasikan dengan benar.

Untuk mempermudah interpretasi, confusion matrix kemudian divisualisasikan dalam bentuk grafik menggunakan library Matplotlib dan Seaborn. Visualisasi ini membantu dalam melihat distribusi kesalahan prediksi model secara lebih jelas. Berdasarkan hasil confusion matrix yang diperoleh, dapat disimpulkan bahwa sebagian besar data berhasil diklasifikasikan dengan benar, dengan jumlah kesalahan prediksi yang relatif kecil. Hal ini menunjukkan bahwa model XGBoost memiliki performa yang baik dalam membedakan balita stunting dan tidak stunting.



**Gambar 4.14 Heatmap Confusion Matrix**

Gambar 4.14 menampilkan *heatmap* dari *confusion matrix* yang merepresentasikan performa model XGBoost dalam mengklasifikasikan status stunting balita. Berdasarkan visualisasi tersebut, hasil klasifikasi dapat diuraikan sebagai berikut:

1. *True Positive* (TP): Model berhasil memprediksi sebanyak [9788] balita yang benar-benar mengalami stunting dengan tepat.
2. *True Negative* (TN): Model secara akurat memprediksi sebanyak [2587] balita yang memiliki status normal.
3. *False Positive* (FP): Terdapat [328] data balita yang sebenarnya normal namun terprediksi stunting oleh model.
4. *False Negative* (FN): Terdapat [317] data balita yang sebenarnya stunting namun salah diprediksi sebagai normal.

Melalui perolehan nilai-nilai tersebut, model *XGBoost* menunjukkan performa yang sangat solid dengan tingkat Akurasi sebesar 0,98%. Dominasi warna pada diagonal utama (dari kiri atas ke kanan bawah) menunjukkan bahwa model memiliki sensitivitas yang tinggi dalam mengenali pola stunting. Hal ini membuktikan bahwa penggunaan pendekatan *Stratified K-Fold Cross Validation* efektif dalam menangani distribusi data yang ada, sehingga hasil klasifikasi akhir menjadi lebih andal dan dapat dipertanggungjawabkan untuk membantu deteksi dini stunting pada balita.

#### **4.6 PEMBAHASAN HASIL PENELITIAN**

Subbab ini membahas secara mendalam hasil penelitian yang diperoleh dari penerapan algoritma *Extreme Gradient Boosting (XGBoost)* dalam mengklasifikasikan status stunting balita. Pembahasan difokuskan pada kinerja model, pengaruh penggunaan metode *Stratified K-Fold Cross Validation* terhadap stabilitas evaluasi, perbandingan hasil antar fold, serta kelebihan dan keterbatasan model yang dihasilkan.

##### **4.6.1 Kinerja XGBoost dalam Klasifikasi Stunting**

Hasil evaluasi menunjukkan bahwa algoritma *XGBoost* mampu mengklasifikasikan status stunting balita dengan sangat baik. Hal ini ditunjukkan oleh nilai akurasi serta nilai *precision*, *recall*, dan *F1-score* yang tinggi dan seimbang pada kedua kelas. Hasil confusion matrix juga memperlihatkan bahwa sebagian besar data berhasil

diklasifikasikan dengan benar, sehingga model memiliki kemampuan generalisasi yang baik.

#### **4.6.2 Dampak Stratified K-Fold Cross Validation terhadap Stabilitas Evaluasi**

Penggunaan Stratified K-Fold Cross Validation dengan  $K = 10$  menghasilkan evaluasi model yang lebih stabil dan representatif. Proporsi kelas yang seimbang pada setiap fold membantu mengurangi bias evaluasi, sehingga performa model yang diperoleh lebih objektif dan dapat diandalkan.

#### **4.6.3 Perbandingan Hasil Evaluasi Antar Fold**

Hasil evaluasi pada setiap fold menunjukkan nilai akurasi yang relatif konsisten dengan perbedaan yang kecil. Konsistensi ini menandakan bahwa model tidak bergantung pada satu pembagian data tertentu dan mampu mempertahankan performa yang baik pada berbagai subset data.

#### **4.6.4 Kelebihan dan Keterbatasan Model**

Kelebihan model XGBoost terletak pada performa klasifikasi yang tinggi dan kemampuannya menangani hubungan nonlinier antar fitur. Namun, model ini memiliki keterbatasan dalam hal interpretabilitas serta membutuhkan penyesuaian parameter yang tepat agar dapat menghasilkan performa optimal.

#### 4.7 RINGKASAN HASIL BAB IV

Bab IV menyajikan hasil penelitian yang diperoleh dari penerapan algoritma Extreme Gradient Boosting (XGBoost) dalam mengklasifikasikan status stunting balita menggunakan dataset yang bersumber dari Kaggle. Proses penelitian diawali dengan pemuatan dan pengolahan data, termasuk transformasi dan pelabelan data agar sesuai dengan kebutuhan algoritma pembelajaran mesin. Evaluasi model dilakukan menggunakan metode Stratified K-Fold Cross Validation dengan jumlah fold sebanyak 10. Pendekatan ini bertujuan untuk menjaga keseimbangan proporsi kelas pada setiap fold dan menghasilkan evaluasi performa model yang lebih stabil dan objektif. Hasil evaluasi menunjukkan bahwa model XGBoost memiliki kinerja yang sangat baik dengan nilai akurasi yang tinggi dan konsisten pada setiap fold.

Hasil pengujian model menggunakan data uji juga didukung oleh nilai precision, recall, dan F1-score yang tinggi pada kedua kelas, yaitu kelas stunting dan tidak stunting. Confusion matrix dan visualisasi heatmap menunjukkan bahwa sebagian besar data berhasil diklasifikasikan dengan benar, dengan jumlah kesalahan prediksi yang relatif kecil. Secara keseluruhan, hasil penelitian pada Bab IV menunjukkan bahwa algoritma XGBoost yang dikombinasikan dengan pendekatan Stratified K-Fold Cross Validation mampu menghasilkan model klasifikasi status stunting balita yang andal, stabil, dan memiliki performa yang baik.