EUSTACHIUS DITO DEWANTORO

A11.2022.14105

BENGKEL KODING - DS03

LINK COLLAB

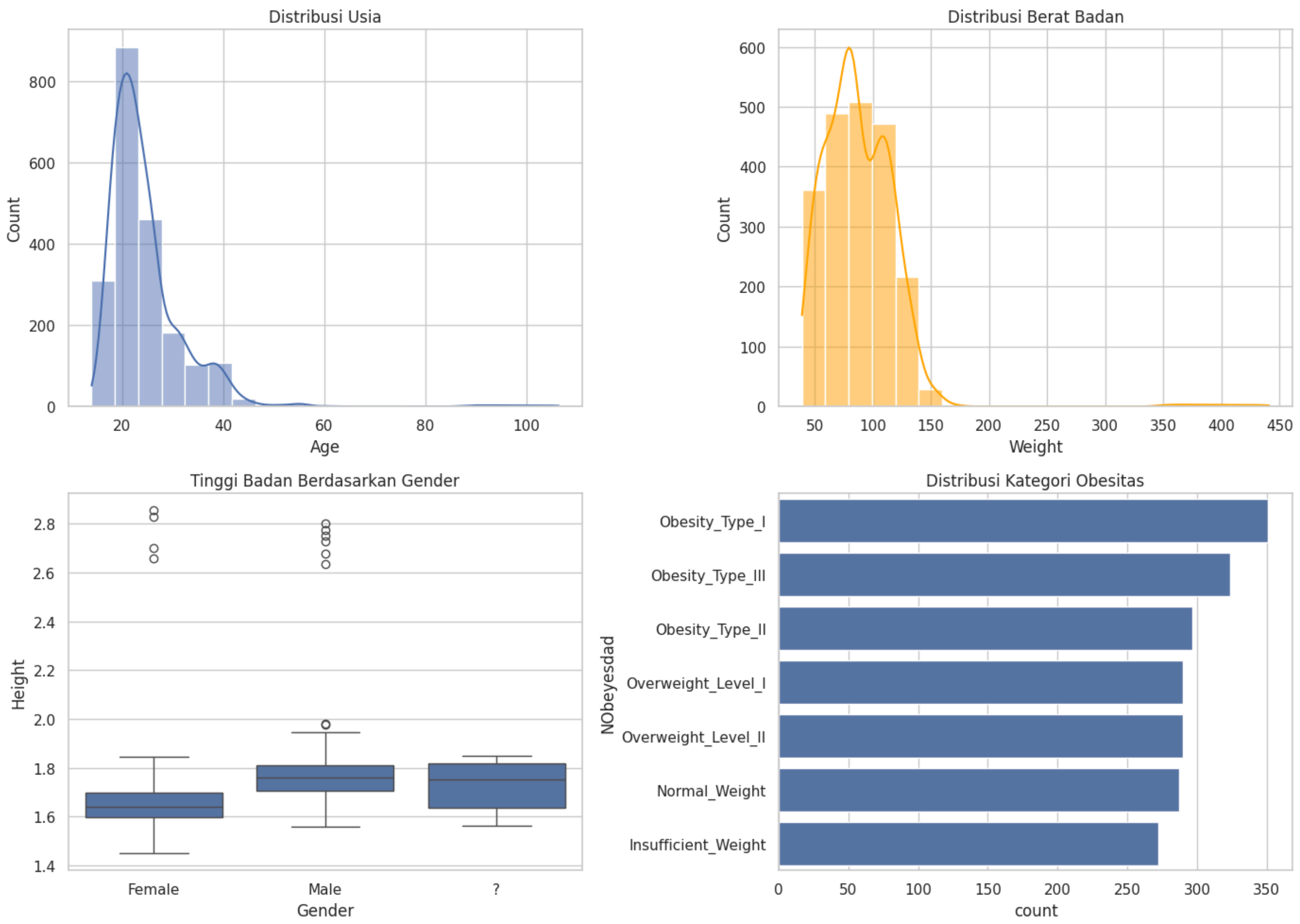
1. Exploratory Data Analysis (EDA)

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
df = pd.read_csv("ObesityDataSet.csv")

# 1. Menampilkan 5 baris pertama
print("🔹 1. Lima Baris Pertama:")
print(df.head())

# 2. Menampilkan jumlah baris dan kolom
print("\n🔹 2. Jumlah Baris dan Kolom:")
print(f"Jumlah baris: {df.shape[0]}")
print(f"Jumlah kolom: {df.shape[1]}")

# 3. Menampilkan informasi tipe data dan missing values
print("\n🔹 3. Informasi DataFrame:")
print(df.info())

# 4. Menampilkan deskripsi statistik kolom (meskipun semua masih object)
print("\n🔹 4. Deskripsi Statistik:")
print(df.describe(include='all'))
```

```
4                              no   0   0  Sometimes  Public_Transportation

            NObeyesdad
0        Normal_Weight
1        Normal_Weight
2        Normal_Weight
3   Overweight_Level_I
4  Overweight_Level_II

🔹 2. Jumlah Baris dan Kolom:
Jumlah baris: 2111
Jumlah kolom: 17

🔹 3. Informasi DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2111 entries, 0 to 2110
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype 
---  ------                          --------------  ----- 
 0   Age                             2097 non-null   object
 1   Gender                          2102 non-null   object
 2   Height                          2099 non-null   object
 3   Weight                          2100 non-null   object
 4   CALC                            2106 non-null   object
 5   FAVC                            2100 non-null   object
 6   FCVC                            2103 non-null   object
 7   NCP                             2099 non-null   object
 8   SCC                             2101 non-null   object
 9   SMOKE                           2106 non-null   object
 10  CH2O                            2105 non-null   object
 11  family_history_with_overweight  2098 non-null   object
 12  FAF                             2103 non-null   object
 13  TUE                             2102 non-null   object
 14  CAEC                            2100 non-null   object
 15  MTRANS                          2105 non-null   object
 16  NObeyesdad                      2111 non-null   object
dtypes: object(17)
memory usage: 280.5+ KB
None

🔹 4. Deskripsi Statistik:
         Age Gender Height Weight       CALC  FAVC  FCVC   NCP   SCC SMOKE  \
count   2097   2102   2099   2100       2106  2100  2103  2099  2101  2106   
unique  1394      3   1562   1518          5     3   808   637     3     3   
top       18   Male    1.7     80  Sometimes   yes     3     3    no    no   
freq     124   1056     58     58       1386  1844   647  1183  1997  2054   

        CH2O family_history_with_overweight   FAF   TUE       CAEC  \
count   2105                           2098  2103  2102       2100   
unique  1263                              3  1186  1130          5   
top        2                            yes     0     0  Sometimes   
freq     441                           1705   404   552       1747   

                       MTRANS      NObeyesdad  
count                    2105            2111  
unique                      6               7  
top     Public_Transportation  Obesity_Type_I  
freq                     1572             351  
```

```
# Salin dataset untuk manipulasi visualisasi
df_vis = df.copy()

# Konversi kolom numerik yang masih bertipe object menjadi float
numerical_cols = ['Age', 'Height', 'Weight', 'FCVC', 'NCP', 'CH2O', 'FAF', 'TUE']
for col in numerical_cols:
    df_vis[col] = pd.to_numeric(df_vis[col], errors='coerce')
```

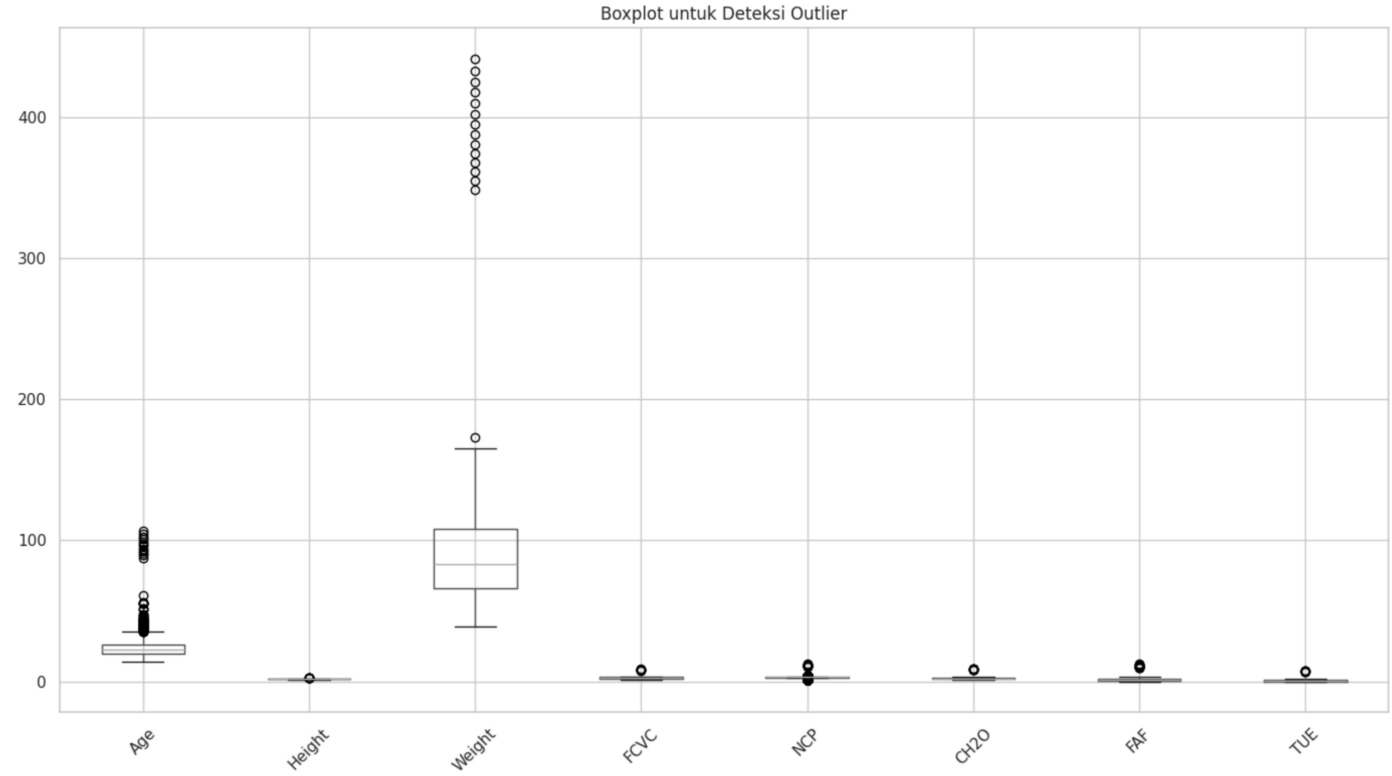
```
# Set style visualisasi
sns.set(style="whitegrid")

# Membuat figure dan subplot
fig, axs = plt.subplots(2, 2, figsize=(14, 10))

# Plot distribusi usia
sns.histplot(df_vis['Age'].dropna(), kde=True, bins=20, ax=axs[0, 0])
axs[0, 0].set_title('Distribusi Usia')

# Plot distribusi berat badan
sns.histplot(df_vis['Weight'].dropna(), kde=True, bins=20, ax=axs[0, 1], color='orange')
axs[0, 1].set_title('Distribusi Berat Badan')

# Boxplot tinggi badan berdasarkan gender
sns.boxplot(x='Gender', y='Height', data=df_vis, ax=axs[1, 0])
axs[1, 0].set_title('Tinggi Badan Berdasarkan Gender')

# Barplot kategori obesitas
sns.countplot(y='NObeyesdad', data=df_vis, order=df_vis['NObeyesdad'].value_counts().index, ax=axs[1, 1])
axs[1, 1].set_title('Distribusi Kategori Obesitas')

# Menata layout
plt.tight_layout()
plt.show()
```



```
# 1. Cek missing values
print("Missing Values per Kolom:")
print(df.isnull().sum())

# 2. Cek jumlah nilai unik per kolom
print("\nJumlah Nilai Unik per Kolom:")
print(df.nunique())

# 3. Cek data duplikat
print(f"\nJumlah Baris Duplikat: {df.duplicated().sum()}")

# 4. Keseimbangan data pada label target
print("\nDistribusi Kategori Obesitas:")
print(df['NObeyesdad'].value_counts())

# 5. Deteksi outlier menggunakan boxplot
# Konversi kolom numerik
for col in ['Age', 'Height', 'Weight', 'FCVC', 'NCP', 'CH2O', 'FAF', 'TUE']:
    df[col] = pd.to_numeric(df[col], errors='coerce')
```

```
# Tampilkan boxplot
plt.figure(figsize=(14, 8))
df[['Age', 'Height', 'Weight', 'FCVC', 'NCP', 'CH2O', 'FAF', 'TUE']].boxplot()
plt.title("Boxplot untuk Deteksi Outlier")
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Missing Values per Kolom:
Age                               14
Gender                             9
Height                            12
Weight                            11
CALC                               5
FAVC                              11
FCVC                               8
NCP                               12
SCC                               10
SMOKE                              5
CH2O                               6
family_history_with_overweight    13
FAF                                8
TUE                                9
CAEC                              11
MTRANS                             6
NObeyesdad                         0
dtype: int64

Jumlah Nilai Unik per Kolom:
Age                               1394
Gender                               3
Height                            1562
Weight                            1518
CALC                                 5
FAVC                                 3
FCVC                               808
NCP                                637
SCC                                  3
SMOKE                                3
CH2O                              1263
family_history_with_overweight       3
FAF                               1186
TUE                               1130
CAEC                                 5
MTRANS                               6
NObeyesdad                           7
dtype: int64

Jumlah Baris Duplikat: 18

Distribusi Kategori Obesitas:
NObeyesdad
Obesity_Type_I         351
Obesity_Type_III       324
Obesity_Type_II        297
Overweight_Level_I     290
Overweight_Level_II    290
Normal_Weight          287
Insufficient_Weight    272
Name: count, dtype: int64
```



KESIMPULAN EDA

1. Struktur dan Tipe Data

- Dataset memiliki 2111 baris dan 17 kolom.
- Semua kolom awalnya bertipe object, termasuk kolom yang seharusnya numerik.
- Diperlukan konversi tipe data pada kolom seperti Age, Height, Weight, dll. ke tipe numerik (int atau float).

2. Missing Values

- Beberapa kolom mengandung missing values, misalnya Age, Height, Weight, FCVC, CH2O, dan lain-lain.
- Penanganan missing values perlu dilakukan sebelum melanjutkan ke model prediktif.

3. Unique Values

- Kolom numerik mengandung banyak nilai unik yang masuk akal (contoh: Age, Weight, dll.).
- Kolom kategori seperti Gender, CALC, MTRANS memiliki jumlah nilai unik yang terbatas, cocok untuk encoding kategorikal nantinya.

4. Duplikasi Data

- Ditemukan sejumlah baris duplikat yang sebaiknya dihapus untuk mencegah bias model.

5. Distribusi dan Keseimbangan Data

- Distribusi usia dan berat badan tampak normal dengan sedikit skewness.
- Keseimbangan kelas (label NObeyesdad) tidak merata. Kategori seperti Obesity_Type_I dan Obesity_Type_II mendominasi, sedangkan Insufficient_Weight relatif jarang.

6. Outlier

- Ditemukan outlier pada beberapa kolom numerik seperti Weight, Height, dan FAF berdasarkan boxplot.

2. Preprocessing Data

```
!pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.11/dist-packages (0.13.0)
Requirement already satisfied: numpy<3,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from imbalanced-learn) (2.0.2)
Requirement already satisfied: scipy<2,>=1.10.1 in /usr/local/lib/python3.11/dist-packages (from imbalanced-learn) (1.15.3)
Requirement already satisfied: scikit-learn<2,>=1.3.2 in /usr/local/lib/python3.11/dist-packages (from imbalanced-learn) (1.6.1)
Requirement already satisfied: sklearn-compat<1,>=0.1 in /usr/local/lib/python3.11/dist-packages (from imbalanced-learn) (0.1.3)
Requirement already satisfied: joblib<2,>=1.1.1 in /usr/local/lib/python3.11/dist-packages (from imbalanced-learn) (1.5.0)
Requirement already satisfied: threadpoolctl<4,>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from imbalanced-learn) (3.6.0)
```

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
from imblearn.over_sampling import SMOTE
import matplotlib.pyplot as plt
```

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from imblearn.over_sampling import SMOTE

# Konversi ke numerik
num_cols = ['Age', 'Height', 'Weight', 'FCVC', 'NCP', 'CH2O', 'FAF', 'TUE']
for col in num_cols:
    df[col] = pd.to_numeric(df[col], errors='coerce')

# Drop duplikat dan missing values
df.drop_duplicates(inplace=True)
df.dropna(inplace=True)

# Tangani outlier
Q1 = df[num_cols].quantile(0.25)
Q3 = df[num_cols].quantile(0.75)
IQR = Q3 - Q1
df = df[~((df[num_cols] < (Q1 - 1.5 * IQR)) | (df[num_cols] > (Q3 + 1.5 * IQR))).any(axis=1)]

# Label encoding untuk target
le = LabelEncoder()
df['NObeyesdad'] = le.fit_transform(df['NObeyesdad'])

# One-hot encoding
df = pd.get_dummies(df, drop_first=True)

# Pisah fitur dan target
X = df.drop("NObeyesdad", axis=1)
y = df["NObeyesdad"]

# SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Standarisasi
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_resampled)

# Buat DataFrame hasil
X_final = pd.DataFrame(X_scaled, columns=X.columns)
y_final = pd.Series(y_resampled, name='NObeyesdad')
df_final = pd.concat([X_final, y_final], axis=1)

# Tampilkan hasil
```

```
print(df_final.head())
```

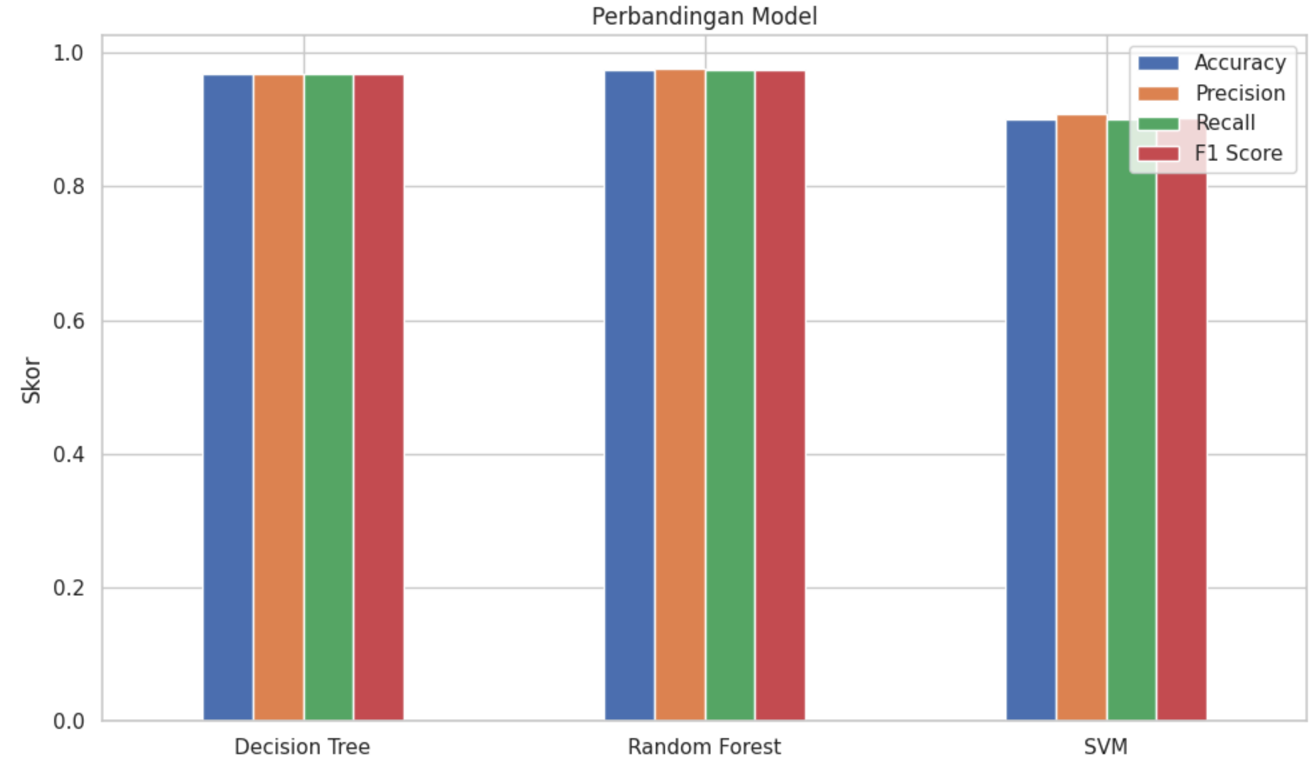
```
        Age    Height    Weight      FCVC  NCP      CH2O       FAF       TUE  \
0 -0.448149 -1.156532 -0.864146 -0.773564  0.0 -0.076701 -1.262658  0.460337   
1 -0.448149 -2.349917 -1.165899  1.156600  0.0  1.672112  2.311450 -1.195566   
2  0.032723  0.991561 -0.373796 -0.773564  0.0 -0.076701  1.120081  0.460337   
3  0.994469  0.991561  0.003396  1.156600  0.0 -0.076701  1.120081 -1.195566   
4  1.475341 -1.156532 -1.279057 -0.773564  0.0 -0.076701 -1.262658 -1.195566   

   Gender_Female  Gender_Male  ...  CAEC_Always  CAEC_Frequently  \
0       1.092936    -1.196213  ...    -0.196063        -0.401269   
1       1.092936    -1.196213  ...    -0.196063        -0.401269   
2      -0.914967     0.835972  ...    -0.196063        -0.401269   
3      -0.914967     0.835972  ...    -0.196063        -0.401269   
4      -0.914967     0.835972  ...    -0.196063        -0.401269   

   CAEC_Sometimes   CAEC_no  MTRANS_Automobile  MTRANS_Bike  MTRANS_Motorbike  \
0        0.373673 -0.156757          -0.457528    -0.100026         -0.094566   
1        0.373673 -0.156757          -0.457528    -0.100026         -0.094566   
2        0.373673 -0.156757          -0.457528    -0.100026         -0.094566   
3        0.373673 -0.156757          -0.457528    -0.100026         -0.094566   
4        0.373673 -0.156757           2.185660    -0.100026         -0.094566   

   MTRANS_Public_Transportation  MTRANS_Walking  NObeyesdad  
0                      0.426336       -0.208628           1  
1                      0.426336       -0.208628           1  
2                      0.426336       -0.208628           1  
3                     -2.345569        4.793224           5  
4                     -2.345569       -0.208628           1  

[5 rows x 31 columns]
<ipython-input-7-374f7fb251e9>:22: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cc
  df['NObeyesdad'] = le.fit_transform(df['NObeyesdad'])
```

Kesimpulan Tahap Preprocessing:

- Missing values dan data duplikat berhasil dihapus.
- Semua kolom numerik telah dikonversi ke tipe numerik.
- Fitur kategorikal dikonversi ke bentuk numerik dengan encoding.
- Outlier dihapus menggunakan metode IQR.
- Ketidakseimbangan data ditangani dengan SMOTE.
- Data numerik telah dinormalisasi menggunakan StandardScaler.

3. Permodelan dan Evaluasi

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y_resampled, test_size=0.2, random_state=42, stratify=y_resampled
)


# Models
models = {
    'Decision Tree': DecisionTreeClassifier(random_state=42),
    'Random Forest': RandomForestClassifier(random_state=42),
    'SVM': SVC(random_state=42)
}

# Evaluasi
results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    results[name] = {
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred, average='weighted'),
        'Recall': recall_score(y_test, y_pred, average='weighted'),
        'F1 Score': f1_score(y_test, y_pred, average='weighted'),
        'Confusion Matrix': confusion_matrix(y_test, y_pred)
    }



# Visualisasi
df_result = pd.DataFrame({
    model: {
        'Accuracy': met['Accuracy'],
        'Precision': met['Precision'],
        'Recall': met['Recall'],
        'F1 Score': met['F1 Score']
    } for model, met in results.items()
}).T

df_result.plot(kind='bar', figsize=(10, 6), title='Perbandingan Model')
plt.ylabel("Skor")
plt.xticks(rotation=0)
plt.grid(True)
plt.tight_layout()
plt.show()

print(df_result)
```



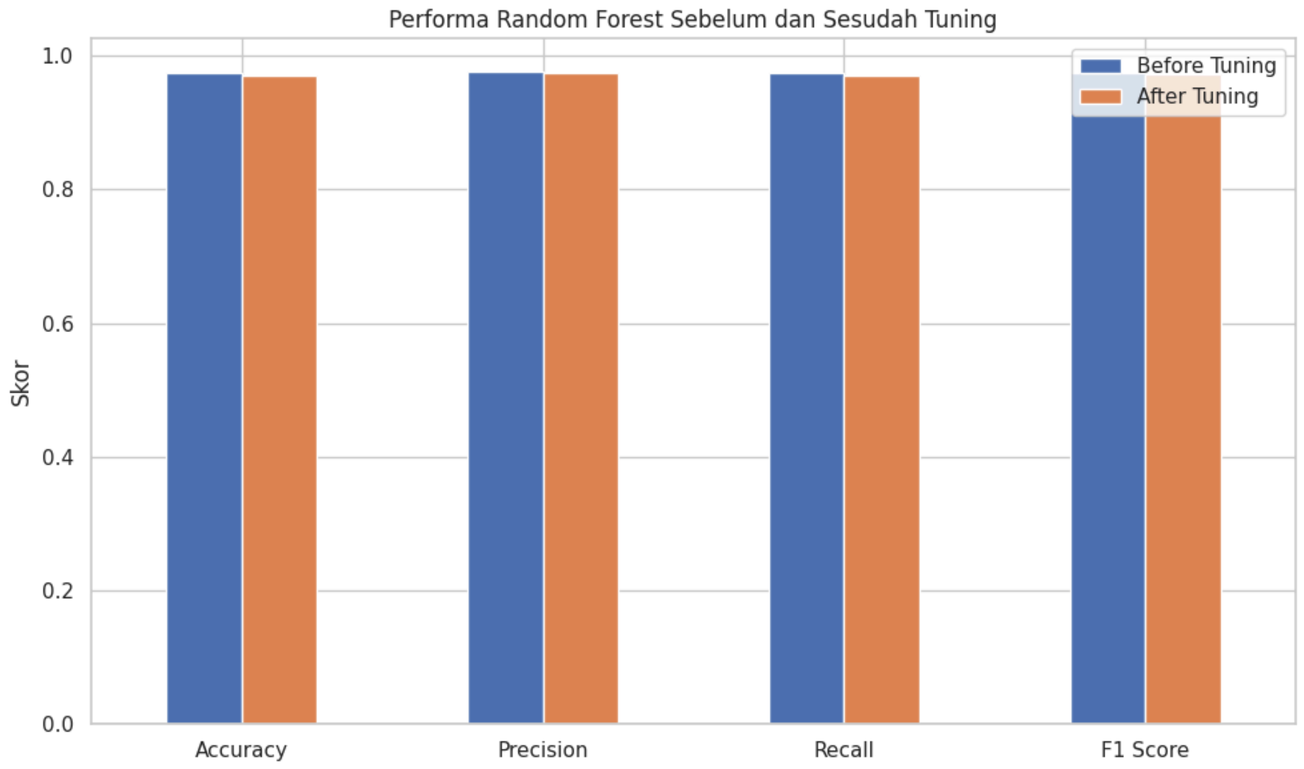
```
               Accuracy  Precision    Recall  F1 Score
Decision Tree  0.968750   0.969125  0.968750  0.968458
Random Forest  0.973958   0.977156  0.973958  0.974390
SVM            0.901042   0.908253  0.901042  0.902662
```

Kesimpulan:

- Random Forest menunjukkan performa terbaik secara keseluruhan dengan nilai tertinggi di semua metrik utama.
- Decision Tree juga memberikan performa yang sangat baik, hanya sedikit di bawah Random Forest.
- SVM memiliki precision yang tinggi, namun metrik lainnya lebih rendah, menunjukkan bahwa model ini mungkin kurang baik dalam mengenali semua kelas dengan merata.

✅ Rekomendasi: Gunakan Random Forest untuk klasifikasi tingkat obesitas berdasarkan dataset ini karena memberikan akurasi dan keseimbangan performa terbaik.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import matplotlib.pyplot as plt
import pandas as pd

# Parameter grid
param_grid = {
    'n_estimators': [100, 150, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

# GridSearchCV
grid = GridSearchCV(
    estimator=RandomForestClassifier(random_state=42),
    param_grid=param_grid,
    cv=3,
    scoring='accuracy',
    n_jobs=-1
)
grid.fit(X_train, y_train)

# Model terbaik
best_rf = grid.best_estimator_
y_pred_best = best_rf.predict(X_test)

# Evaluasi
before = {
    'Accuracy': 0.973958,
    'Precision': 0.977156,
    'Recall': 0.973958,
    'F1 Score': 0.974390
}

after = {
    'Accuracy': accuracy_score(y_test, y_pred_best),
    'Precision': precision_score(y_test, y_pred_best, average='weighted'),
    'Recall': recall_score(y_test, y_pred_best, average='weighted'),
    'F1 Score': f1_score(y_test, y_pred_best, average='weighted')
}

# Buat DataFrame
df_compare = pd.DataFrame({'Before Tuning': before, 'After Tuning': after})

# Plot
df_compare.plot(kind='bar', figsize=(10, 6), title='Performa Random Forest Sebelum dan Sesudah Tuning')
plt.ylabel("Skor")
plt.xticks(rotation=0)
plt.grid(True)
plt.tight_layout()
plt.show()

# Lihat perbandingan numerik
print(df_compare)
```



```
           Before Tuning  After Tuning
Accuracy        0.973958      0.971354
Precision       0.977156      0.975210
Recall          0.973958      0.971354
F1 Score        0.974390      0.971898
```

Berdasarkan grafik dan hasil evaluasi:

- Performa model setelah tuning mengalami sedikit penurunan dibanding sebelum tuning.
- Skor metrik seperti accuracy (0.9739 → 0.9714), precision, recall, dan F1-score semuanya menurun secara marginal.
- Hal ini menunjukkan bahwa parameter default Random Forest pada dataset ini sudah sangat optimal, dan tuning tidak selalu menghasilkan peningkatan performa.