ml-project

October 26, 2025

[26]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from numpy import array
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, confusion_matrix, classification_report
```

[27]:
```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

[28]:
```
!ls
```

```
drive  sample_data  sample_submission.csv  test.csv  train.csv
```

[29]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[30]:
```
trainData=pd.read_csv("train.csv")
trainData
```

[30]:
```
          id  Gender        Age    Height     Weight  \
0          0    Male  24.443011  1.699998  81.669950
1          1  Female  18.000000  1.560000  57.000000
2          2  Female  18.000000  1.711460  50.165754
```

```
3          3  Female  20.952737  1.710730  131.274851
4          4    Male  31.641081  1.914186   93.798055
…        …      …          …         …          …
15528  15528    Male  18.000000  1.700000   50.000000
15529  15529    Male  18.000000  1.763101   55.523481
15530  15530  Female  19.010211  1.686936   49.660995
15531  15531    Male  22.777890  1.805445   85.228116
15532  15532    Male  39.371523  1.770278   79.677930

      family_history_with_overweight FAVC      FCVC       NCP        CAEC  \
0                                yes  yes  2.000000  2.983297   Sometimes
1                                yes  yes  2.000000  3.000000  Frequently
2                                yes  yes  1.880534  1.411685   Sometimes
3                                yes  yes  3.000000  3.000000   Sometimes
4                                yes  yes  2.679664  1.971472   Sometimes
…                                  …    …         …         …           …
15528                             no  yes  2.000000  3.000000  Frequently
15529                            yes  yes  2.786008  3.000000   Sometimes
15530                             no  yes  1.053534  3.452590   Sometimes
15531                            yes  yes  2.000000  2.092179   Sometimes
15532                            yes  yes  2.407817  1.097312   Sometimes

      SMOKE      CH2O  SCC       FAF       TUE        CALC  \
0        no  2.763573   no  0.000000  0.976473   Sometimes
1        no  2.000000   no  1.000000  1.000000          no
2        no  1.910378   no  0.866045  1.673584          no
3        no  1.674061   no  1.467863  0.780199   Sometimes
4        no  1.979848   no  1.967973  0.931721   Sometimes
…     …         …   …        …         …           …
15528    no  2.000000   no  1.000000  2.000000   Sometimes
15529    no  1.962646  yes  0.028202  1.561272   Sometimes
15530    no  1.000000   no  2.001230  1.000000   Sometimes
15531    no  2.452986   no  0.796770  0.000000   Sometimes
15532    no  2.205911   no  0.977929  0.000000  Frequently

                      MTRANS       WeightCategory
0      Public_Transportation  Overweight_Level_II
1                 Automobile        Normal_Weight
2      Public_Transportation  Insufficient_Weight
3      Public_Transportation     Obesity_Type_III
4      Public_Transportation  Overweight_Level_II
…                          …                    …
15528  Public_Transportation  Insufficient_Weight
15529  Public_Transportation  Insufficient_Weight
15530  Public_Transportation  Insufficient_Weight
15531  Public_Transportation   Overweight_Level_I
15532             Automobile  Overweight_Level_II
```

```
[15533 rows x 18 columns]
```

[31]:
```
trainData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15533 entries, 0 to 15532
Data columns (total 18 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              15533 non-null  int64
 1   Gender                          15533 non-null  object
 2   Age                             15533 non-null  float64
 3   Height                          15533 non-null  float64
 4   Weight                          15533 non-null  float64
 5   family_history_with_overweight  15533 non-null  object
 6   FAVC                            15533 non-null  object
 7   FCVC                            15533 non-null  float64
 8   NCP                             15533 non-null  float64
 9   CAEC                            15533 non-null  object
 10  SMOKE                           15533 non-null  object
 11  CH2O                            15533 non-null  float64
 12  SCC                             15533 non-null  object
 13  FAF                             15533 non-null  float64
 14  TUE                             15533 non-null  float64
 15  CALC                            15533 non-null  object
 16  MTRANS                          15533 non-null  object
 17  WeightCategory                  15533 non-null  object
dtypes: float64(8), int64(1), object(9)
memory usage: 2.1+ MB
```

[32]:
```
trainData.columns
```

[32]:
```
Index(['id', 'Gender', 'Age', 'Height', 'Weight',
       'family_history_with_overweight', 'FAVC', 'FCVC', 'NCP', 'CAEC',
       'SMOKE', 'CH2O', 'SCC', 'FAF', 'TUE', 'CALC', 'MTRANS',
       'WeightCategory'],
      dtype='object')
```

[33]:
```
trainData.dtypes
```

[33]:
```
id                                  int64
Gender                             object
Age                               float64
Height                            float64
Weight                            float64
family_history_with_overweight     object
```

```
FAVC                              object
FCVC                             float64
NCP                              float64
CAEC                              object
SMOKE                             object
CH2O                             float64
SCC                               object
FAF                              float64
TUE                              float64
CALC                              object
MTRANS                            object
WeightCategory                    object
dtype: object
```

[34]:
```
trainData.describe()
```

[34]:
```
                 id           Age        Height        Weight          FCVC  \
count  15533.000000  15533.000000  15533.000000  15533.000000  15533.000000
mean    7766.000000     23.816308      1.699918     87.785225      2.442917
std     4484.135201      5.663167      0.087670     26.369144      0.530895
min        0.000000     14.000000      1.450000     39.000000      1.000000
25%     3883.000000     20.000000      1.630927     66.000000      2.000000
50%     7766.000000     22.771612      1.700000     84.000000      2.342220
75%    11649.000000     26.000000      1.762921    111.600553      3.000000
max    15532.000000     61.000000      1.975663    165.057269      3.000000

                NCP          CH2O           FAF           TUE
count  15533.000000  15533.000000  15533.000000  15533.000000
mean       2.760425      2.027626      0.976968      0.613813
std        0.706463      0.607733      0.836841      0.602223
min        1.000000      1.000000      0.000000      0.000000
25%        3.000000      1.796257      0.007050      0.000000
50%        3.000000      2.000000      1.000000      0.566353
75%        3.000000      2.531456      1.582675      1.000000
max        4.000000      3.000000      3.000000      2.000000
```

[35]:
```
trainData.describe(include='object')
```

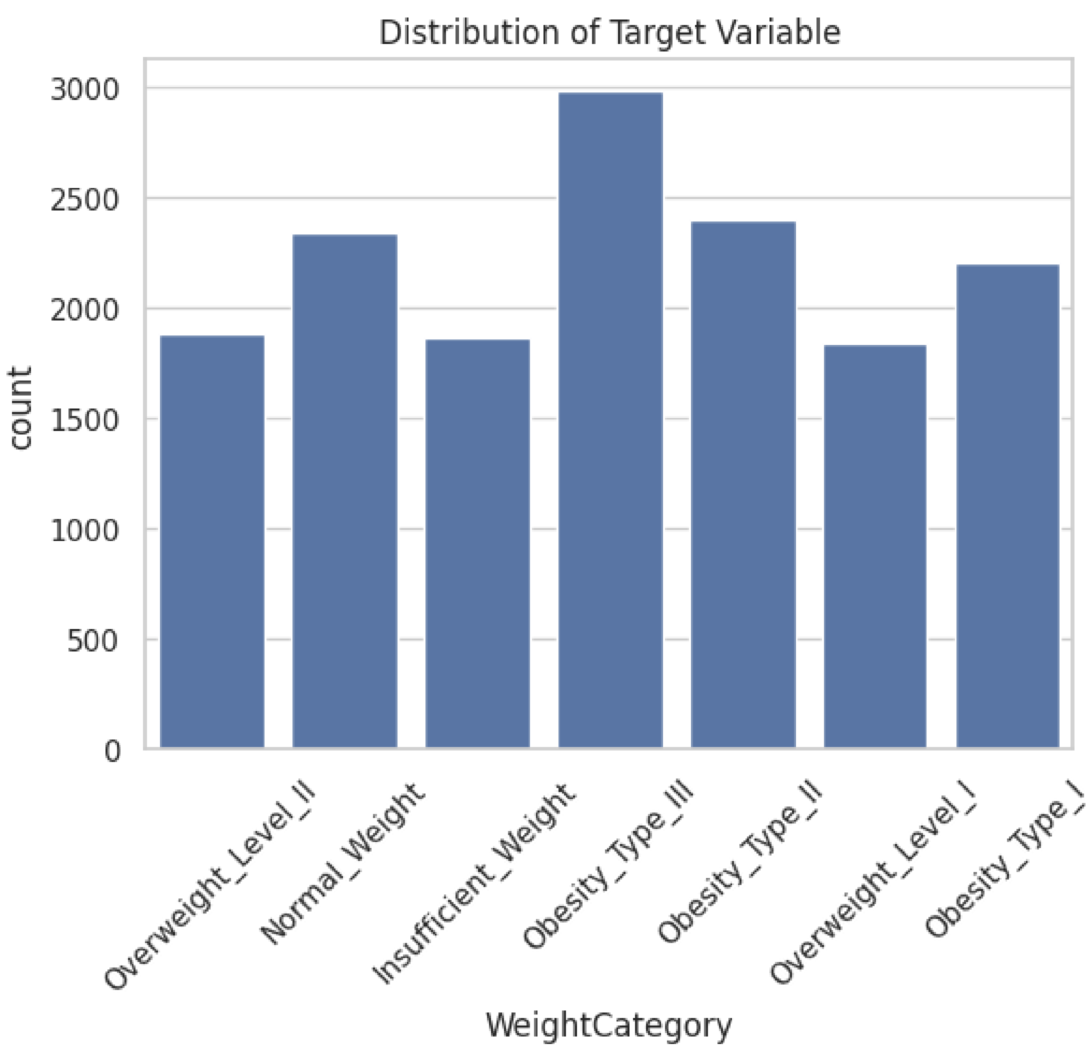
[35]:
```
       Gender family_history_with_overweight   FAVC       CAEC  SMOKE    SCC  \
count   15533                          15533  15533      15533  15533  15533
unique      2                              2      2          4      2      2
top      Male                            yes    yes  Sometimes     no     no
freq     7783                          12696  14184      13126  15356  15019

             CALC                 MTRANS    WeightCategory
count       15533                  15533             15533
unique          3                      5                 7
```

```
top      Sometimes  Public_Transportation  Obesity_Type_III
freq         11285                  12470              2983
```

1 EXPLORATORY DATA ANALYSIS

Target Variable Analysis

[36]:
```
sns.countplot(x='WeightCategory', data=trainData)
plt.title("Distribution of Target Variable")
plt.xticks(rotation=45)
plt.show()

trainData['WeightCategory'].value_counts(normalize=True)
```

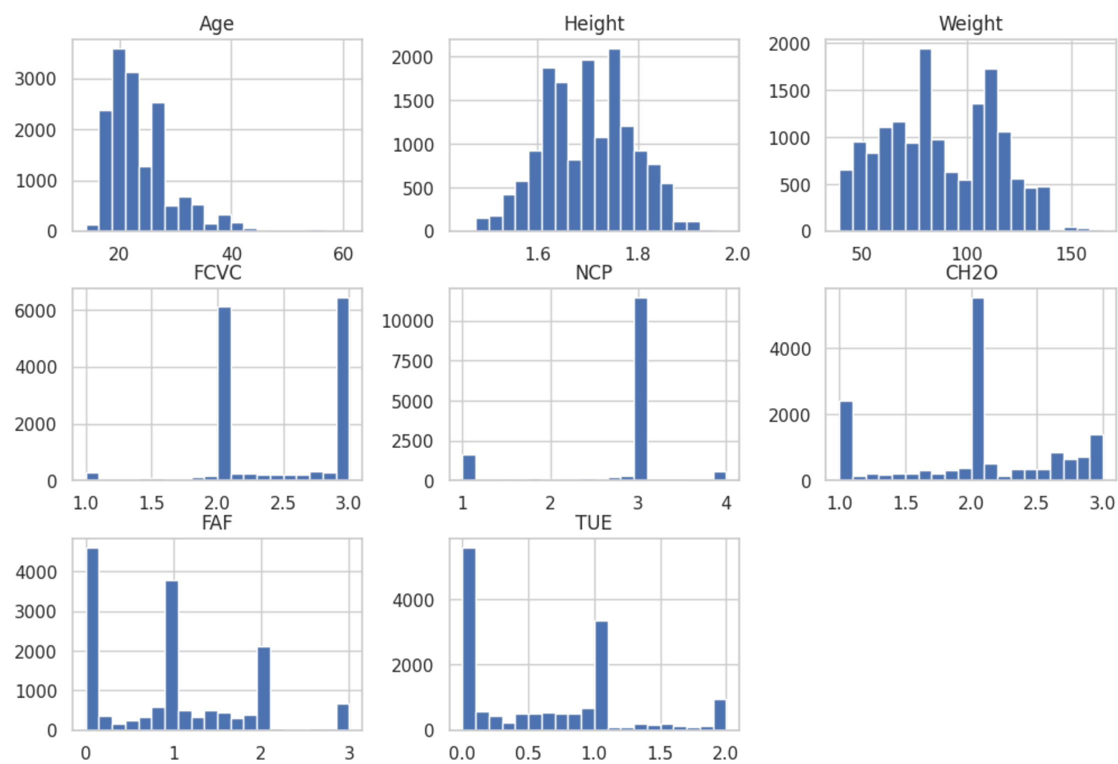


```
[36]: WeightCategory
Obesity_Type_III       0.192043
Obesity_Type_II        0.154703
Normal_Weight          0.150969
Obesity_Type_I         0.142085
Overweight_Level_II    0.121097
Insufficient_Weight    0.120389
Overweight_Level_I     0.118715
Name: proportion, dtype: float64
```

The target variable WeightCategory shows a fairly uniform distribution across all seven categories, with class proportions ranging from 11.9% to 19.2%. The maximum-to-minimum ratio is approximately 1.6:1, indicating that the dataset is reasonably balanced. Hence, no major imbalance handling is required.

Univariate Analysis for Numerical Variables

```
[37]: numeric_cols = ['Age', 'Height', 'Weight', 'FCVC', 'NCP', 'CH2O', 'FAF', 'TUE']
trainData[numeric_cols].hist(bins=20, figsize=(12, 8))
plt.show()
```



- The **Age** distribution is **right-skewed**, with the majority of participants falling between **18 and 25 years**, indicating that the dataset is dominated by young adults.

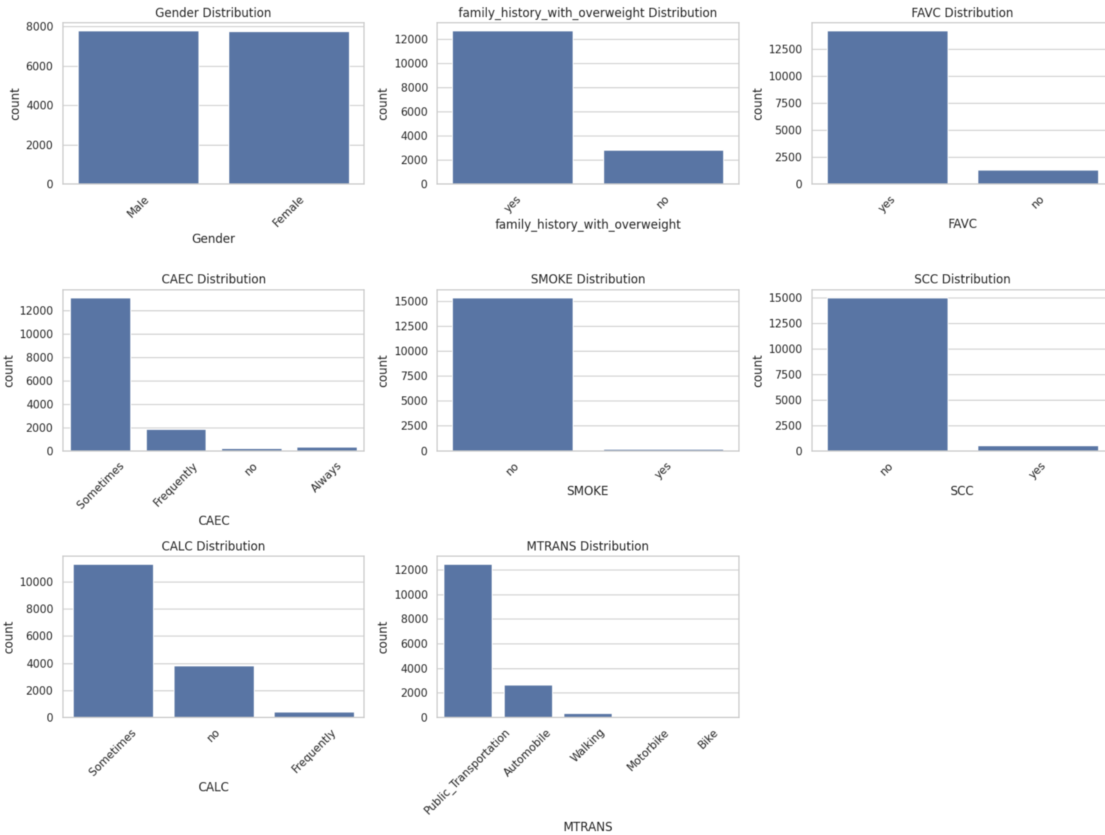
- **Height** follows an approximately **normal distribution**, centered around **1.65 to 1.75 meters**, suggesting a realistic spread across participants.
- **Weight** is **moderately right-skewed**, with most individuals weighing between **50 and 100 kilograms**, and a few higher values that likely represent genuinely obese participants.
- **FCVC (Frequency of Vegetable Consumption)** shows a **bimodal pattern**, where most individuals have values around **2 or 3**, implying regular vegetable consumption among participants.
- **NCP (Number of Main Meals)** displays a strong concentration at **3 meals per day**, which aligns with typical eating habits.
- **CH2O (Daily Water Intake)** is **mildly right-skewed**, with most participants consuming about **2 liters of water per day**.
- **FAF (Physical Activity Frequency)** has a **multimodal distribution**, suggesting a wide range of exercise habits, from no activity to frequent exercise.
- **TUE (Time Using Technology Devices)** is **highly right-skewed**, showing that most participants spend **less than one hour per day** using technology.

A few outliers are present in Age and Weight, but they appear to be genuine rather than data entry errors.

Overall, all numerical features show logical and interpretable distributions, with only mild skewness in some variables that can be handled during model preparation.

Univariate Analysis for Categorical Variables

[38]:
```
cat_cols = ['Gender', 'family_history_with_overweight', 'FAVC', 'CAEC',
            'SMOKE', 'SCC', 'CALC', 'MTRANS']

plt.figure(figsize=(16, 12))
for i, col in enumerate(cat_cols, 1):
    plt.subplot(3, 3, i)
    sns.countplot(x=col, data=trainData)
    plt.title(f"{col} Distribution")
    plt.xticks(rotation=45)
    plt.tight_layout()

plt.show()
```



[39]:
```python
cat_cols = ['Gender', 'family_history_with_overweight', 'FAVC', 'CAEC',
            'SMOKE', 'SCC', 'CALC', 'MTRANS']

for col in cat_cols:
    print(f"\n{'='*40}")
    print(f"  {col} Distribution")
    print(f"{'='*40}")
    counts = trainData[col].value_counts(normalize=True) * 100
    for cat, val in counts.items():
        print(f"{cat:<25}: {val:5.2f}%")
```

```
========================================
  Gender Distribution
========================================
Male                     : 50.11%
Female                   : 49.89%

========================================
  family_history_with_overweight Distribution
========================================
```

```
yes                       : 81.74%
no                        : 18.26%

========================================
  FAVC Distribution
========================================
yes                       : 91.32%
no                        :  8.68%

========================================
  CAEC Distribution
========================================
Sometimes                 : 84.50%
Frequently                : 11.96%
Always                    :  2.23%
no                        :  1.31%

========================================
  SMOKE Distribution
========================================
no                        : 98.86%
yes                       :  1.14%

========================================
  SCC Distribution
========================================
no                        : 96.69%
yes                       :  3.31%

========================================
  CALC Distribution
========================================
Sometimes                 : 72.65%
no                        : 24.73%
Frequently                :  2.62%

========================================
  MTRANS Distribution
========================================
Public_Transportation     : 80.28%
Automobile                : 17.18%
Walking                   :  2.19%
Motorbike                 :  0.19%
Bike                      :  0.15%
```

- The **Gender distribution is nearly equal**, with 50.11% males and 49.89% females, indicating a perfectly balanced gender representation in the dataset.
- A large proportion **(81.74%) of individuals reported having a family history of over-**

weight, suggesting a strong hereditary influence among participants.

- The **majority (91.32%) of participants responded "yes"** to **FAVC (Frequent Consumption of High-Calorie Food)**, implying that most individuals regularly consume calorie-dense foods.
- For **CAEC (Consumption of Food Between Meals)**, **84.5% of respondents reported "Sometimes"**, while only 1.31% reported "no," indicating that snacking between meals is a common habit.
- **SMOKE shows a clear pattern**, with **98.86% non-smokers** and only 1.14% smokers, suggesting that smoking is relatively uncommon in this population.
- In **SCC (Calories Monitoring)**, **96.69% of individuals do not monitor their calorie intake**, showing that calorie tracking is rare among participants.
- For **CALC (Alcohol Consumption)**, **72.65% reported drinking "Sometimes"**, while 24.73% said "no", indicating that moderate alcohol consumption is fairly common.
- Regarding **MTRANS (Mode of Transportation)**, the **vast majority (80.28%) use public transportation**, followed by automobiles (17.18%), while walking and biking are much less common.

Overall Insights

Most categorical variables are highly imbalanced, with one dominant category (e.g., FAVC = "yes", SMOKE = "no").

However, these distributions reflect realistic lifestyle behaviors rather than data issues.

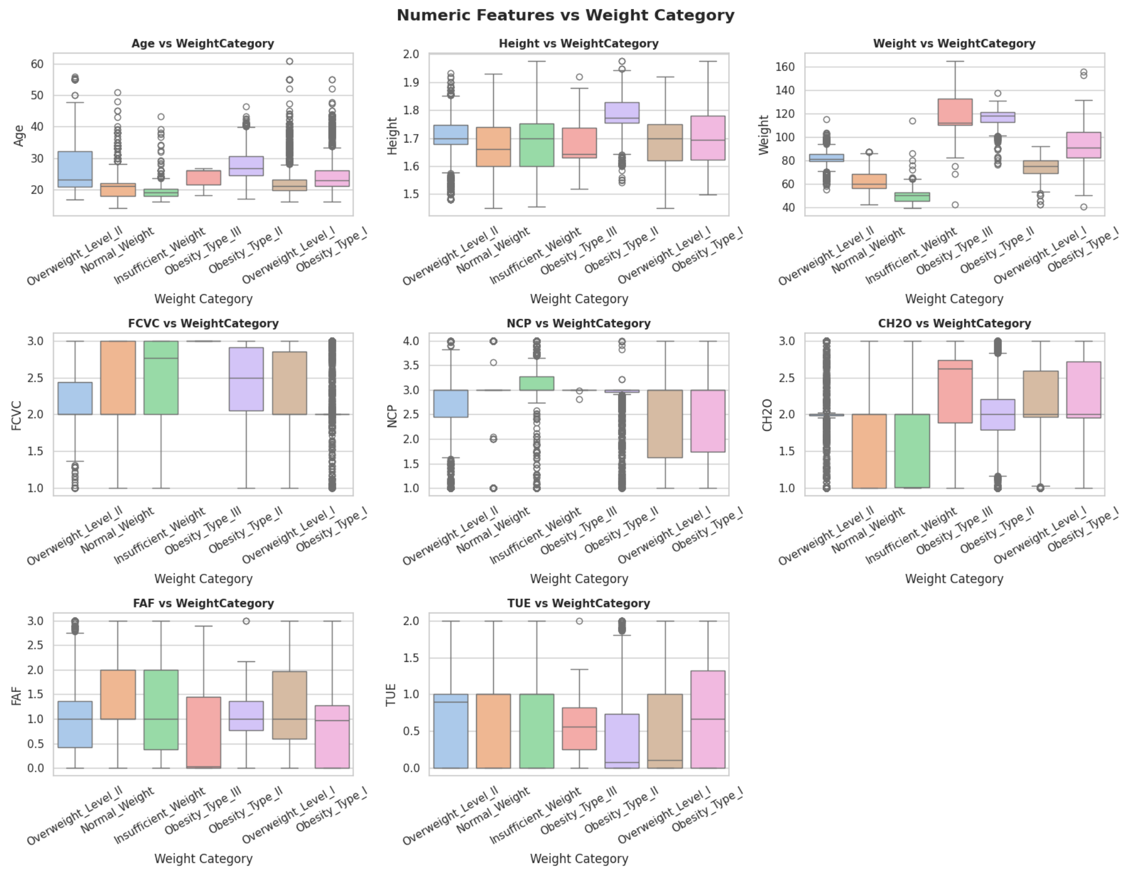
Variables such as Gender are well-balanced, while others (e.g., SMOKE, SCC, MTRANS) may provide limited variability for model training.

The categorical data appears clean, with no evidence of inconsistent or erroneous category values.

Bivariate Analysis for Numerical Variables

[40]:
```
sns.set_theme(style="whitegrid")

plt.figure(figsize=(16, 12))

for i, col in enumerate(numeric_cols, 1):
    plt.subplot(3, 3, i)
    sns.boxplot(
        x='WeightCategory',
        y=col,
        hue='WeightCategory',
        data=trainData,
        palette='pastel',
        legend=False
    )
    plt.title(f"{col} vs WeightCategory", fontsize=11, weight='bold')
    plt.xlabel("Weight Category")
```

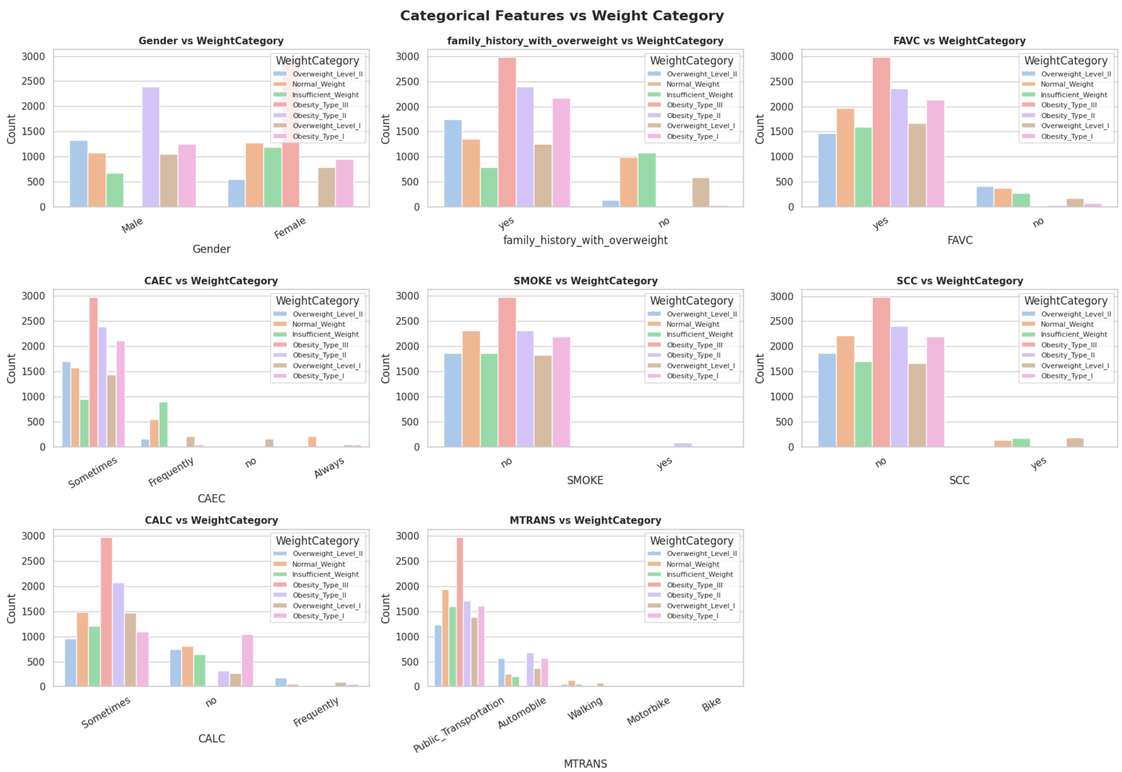
```
    plt.ylabel(col)
    plt.xticks(rotation=30)
    plt.tight_layout()

plt.suptitle("Numeric Features vs Weight Category", fontsize=16, weight='bold',␣
 ↪y=1.02)
plt.show()
```



Bivariate Analysis for Categorical Variables

[41]:
```
import math

sns.set_theme(style="whitegrid")

n_cols = 3
n_rows = math.ceil(len(cat_cols) / n_cols)

plt.figure(figsize=(6 * n_cols, 4 * n_rows))

for i, col in enumerate(cat_cols, 1):
```

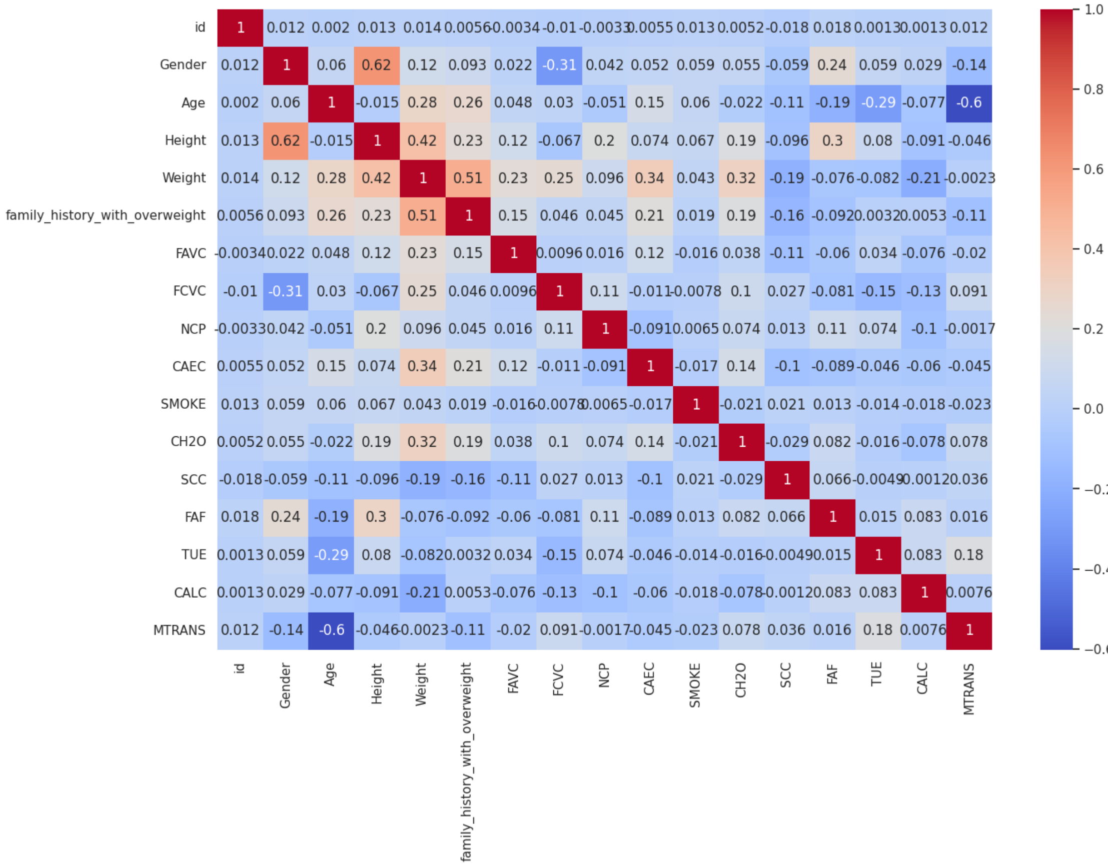
```
    plt.subplot(n_rows, n_cols, i)
    sns.countplot(
        x=col,
        hue='WeightCategory',
        data=trainData,
        palette='pastel'
    )
    plt.title(f"{col} vs WeightCategory", fontsize=11, weight='bold')
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.xticks(rotation=30)
    plt.legend(title="WeightCategory", loc="upper right", fontsize=8)
    plt.tight_layout()

plt.suptitle("Categorical Features vs Weight Category", fontsize=16,␣
↪weight='bold', y=1.02)
plt.show()
```



Correlation Analysis

[42]:
```
x=trainData.drop('WeightCategory',axis=1)
target=trainData["WeightCategory"]
target.head()
```

```
df_copy = x.copy()
categorical_cols = df_copy.select_dtypes(include='object').columns

for col in categorical_cols:
    df_copy[col] = LabelEncoder().fit_transform(df_copy[col])

plt.figure(figsize=(15,10))
sns.heatmap(df_copy.corr(), annot=True, cmap='coolwarm')
plt.show()
```

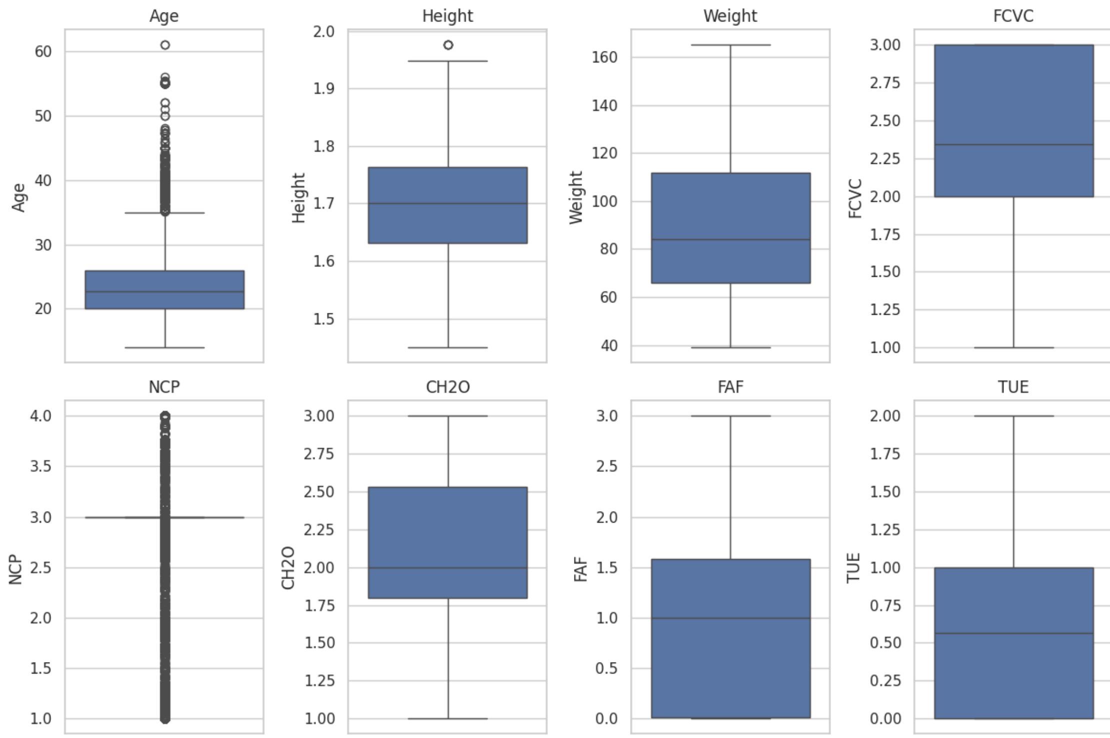


The above heatmap depicts the correlation between the features. There are no two features having high correlation between them. Hence none of the features were dropped.

Outlier Detection

[43]:
```
plt.figure(figsize=(12,8))
for i,col in enumerate(numeric_cols,1):
  plt.subplot(2,4,i)
  sns.boxplot(trainData[col])
  plt.title(col)
plt.tight_layout()
```

```
plt.show()
```



[44]:
```
# IQR method to flag outliers
def outlier_indices(series):
    Q1 = series.quantile(0.25)
    Q3 = series.quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5*IQR
    upper = Q3 + 1.5*IQR
    return series[(series < lower) | (series > upper)].index

outlier_dict = {col: outlier_indices(trainData[col]) for col in numeric_cols}

# check which columns have too many outliers
for col, idx in outlier_dict.items():
    print(f"{col}: {len(idx)} outliers")
```

```
Age: 792 outliers
Height: 4 outliers
Weight: 0 outliers
FCVC: 0 outliers
NCP: 4548 outliers
CH2O: 0 outliers
FAF: 0 outliers
```

```
TUE: 0 outliers
```

[45]:
```python
# Age had 792 outliers but its Maximum value was 61 and Minimum value was 14
# which can't be ignored.


# Compute IQR for Height
Q1 = trainData['Height'].quantile(0.25)
Q3 = trainData['Height'].quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5*IQR
upper = Q3 + 1.5*IQR

# Show the outlier values
trainData['Height'][(trainData['Height'] < lower) | (trainData['Height'] >
 upper)]

#Conculsion - The outlier heights are under 2m. Hence these can't be called
# bad outliers.
```

[45]:
```
1271     1.975663
4084     1.975663
7272     1.975663
11718    1.975663
Name: Height, dtype: float64
```

2 Data Preprocessing

[46]:
```python
trainData.isnull().any()
```

[46]:
```
id                                False
Gender                            False
Age                               False
Height                            False
Weight                            False
family_history_with_overweight    False
FAVC                              False
FCVC                              False
NCP                               False
CAEC                              False
SMOKE                             False
CH2O                              False
SCC                               False
FAF                               False
TUE                               False
CALC                              False
```

```
MTRANS                            False
WeightCategory                    False
dtype: bool
```

[47]:
```
trainData.duplicated().any()
```

[47]:
```
np.False_
```

[48]:
```
x=trainData.drop('WeightCategory',axis=1)
target=trainData["WeightCategory"]
target.head()
```

[48]:
```
0    Overweight_Level_II
1          Normal_Weight
2    Insufficient_Weight
3       Obesity_Type_III
4    Overweight_Level_II
Name: WeightCategory, dtype: object
```

[49]:
```
x.head()
```

[49]:
```
   id  Gender        Age    Height      Weight family_history_with_overweight  \
0   0    Male  24.443011  1.699998   81.669950                            yes
1   1  Female  18.000000  1.560000   57.000000                            yes
2   2  Female  18.000000  1.711460   50.165754                            yes
3   3  Female  20.952737  1.710730  131.274851                            yes
4   4    Male  31.641081  1.914186   93.798055                            yes

  FAVC      FCVC       NCP        CAEC SMOKE      CH2O  SCC       FAF  \
0  yes  2.000000  2.983297   Sometimes    no  2.763573   no  0.000000
1  yes  2.000000  3.000000  Frequently    no  2.000000   no  1.000000
2  yes  1.880534  1.411685   Sometimes    no  1.910378   no  0.866045
3  yes  3.000000  3.000000   Sometimes    no  1.674061   no  1.467863
4  yes  2.679664  1.971472   Sometimes    no  1.979848   no  1.967973

        TUE       CALC                 MTRANS
0  0.976473  Sometimes  Public_Transportation
1  1.000000         no             Automobile
2  1.673584         no  Public_Transportation
3  0.780199  Sometimes  Public_Transportation
4  0.931721  Sometimes  Public_Transportation
```

[50]:
```
numeric_cols = x.select_dtypes(include=[np.number]).columns.tolist()
categorical_cols = x.select_dtypes(include=['object', 'category']).columns.
 ↪tolist()
from sklearn.preprocessing import OneHotEncoder
X_encoded = pd.get_dummies(x, drop_first=True)  # simple one-hot encoding
```

LOADING DATASET FOR TRAINING AND TESTING MODELS

[51]:
```
trainData = pd.read_csv("train.csv")
testData = pd.read_csv("test.csv")

X = trainData.drop(columns=['WeightCategory', 'id'])
y = trainData['WeightCategory']

X_test = testData.drop(columns=['id'])
```

1. DECISION TREE

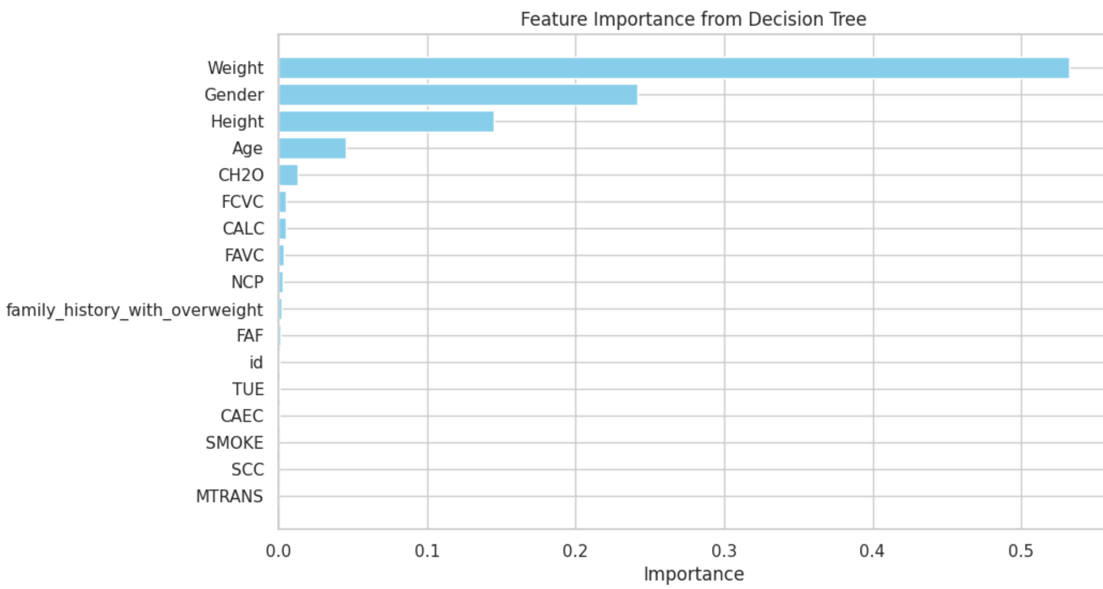
[52]:
```
cat_cols =␣
 ↪['Gender','family_history_with_overweight','FAVC','CAEC','SMOKE','SCC','CALC','MTRANS']
le = LabelEncoder()
for col in cat_cols:
    x[col] = le.fit_transform(x[col])
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
 ↪random_state=42)
```

[53]:
```
dt_clf = DecisionTreeClassifier(random_state=42)

dt_clf.fit(X_train, y_train)

y_pred = dt_clf.predict(X_test)

dt_clf = DecisionTreeClassifier(
    criterion='gini',
    max_depth=7,
    min_samples_split=5,
    min_samples_leaf=2,
    random_state=42
)

dt_clf.fit(X_train, y_train)

y_pred = dt_clf.predict(X_test)

test_accuracy_dtt = accuracy_score(y_test, y_pred)
test_precision_dtt = precision_score(y_test, y_pred, average='weighted',␣
 ↪zero_division=0)
test_recall_dtt = recall_score(y_test, y_pred, average='weighted',␣
 ↪zero_division=0)
test_f1_dtt = f1_score(y_test, y_pred, average='weighted', zero_division=0)

print("--- Decision Tree Model Evaluation ---")
print("Test Accuracy:", test_accuracy_dtt)
print("Test Precision (Weighted):", test_precision_dtt)
```

```python
print("Test Recall (Weighted):", test_recall_dtt)
print("Test F1-Score (Weighted):", test_f1_dtt)
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))


importances = dt_clf.feature_importances_

feature_names = X_train.columns

feat_importances = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)


plt.figure(figsize=(10,6))
plt.barh(feat_importances['Feature'], feat_importances['Importance'],␣
 ↪color='skyblue')
plt.gca().invert_yaxis()
plt.xlabel("Importance")
plt.title("Feature Importance from Decision Tree")
plt.show()
```

```
--- Decision Tree Model Evaluation ---
Test Accuracy: 0.8651432249758609
Test Precision (Weighted): 0.8659294193764477
Test Recall (Weighted): 0.8651432249758609
Test F1-Score (Weighted): 0.8654772440534064

Confusion Matrix:
 [[307  36   1   1   0   0   1]
 [ 29 413   1   0   0  50   3]
 [  0   0 393  15   1  10  29]
 [  0   0  26 436   0   0   2]
 [  0   0   3   1 602   0   0]
 [  3  49   8   0   0 268  56]
 [  0   6  27   1   0  60 269]]

Classification Report:
                      precision    recall  f1-score   support

Insufficient_Weight       0.91      0.89      0.90       346
      Normal_Weight       0.82      0.83      0.83       496
     Obesity_Type_I       0.86      0.88      0.87       448
    Obesity_Type_II       0.96      0.94      0.95       464
   Obesity_Type_III       1.00      0.99      1.00       606
```

```
 Overweight_Level_I       0.69      0.70      0.69       384
Overweight_Level_II       0.75      0.74      0.74       363

           accuracy                           0.87      3107
          macro avg       0.85      0.85      0.85      3107
       weighted avg       0.87      0.87      0.87      3107
```



ACCURACY WITH DECISION TREE IS 0.8651432249758609

Hyperparameter Tuning Using Optuna on Decision Tree

[54]:
```
!pip install optuna
```

```
Collecting optuna
  Downloading optuna-4.5.0-py3-none-any.whl.metadata (17 kB)
Collecting alembic>=1.5.0 (from optuna)
  Downloading alembic-1.17.0-py3-none-any.whl.metadata (7.2 kB)
Collecting colorlog (from optuna)
  Downloading colorlog-6.10.1-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages
(from optuna) (2.0.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.12/dist-packages (from optuna) (25.0)
Collecting sqlalchemy>=1.4.2 (from optuna)
  Downloading sqlalchemy-2.0.44-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_
x86_64.whl.metadata (9.5 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages
(from optuna) (4.67.1)
```

```
Requirement already satisfied: PyYAML in /usr/local/lib/python3.12/dist-packages
(from optuna) (6.0.3)
Requirement already satisfied: Mako in /usr/lib/python3/dist-packages (from
alembic>=1.5.0->optuna) (1.1.3)
Requirement already satisfied: typing-extensions>=4.12 in
/usr/local/lib/python3.12/dist-packages (from alembic>=1.5.0->optuna) (4.15.0)
Collecting greenlet>=1 (from sqlalchemy>=1.4.2->optuna)
  Downloading greenlet-3.2.4-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x8
6_64.whl.metadata (4.1 kB)
Downloading optuna-4.5.0-py3-none-any.whl (400 kB)
                         400.9/400.9 kB
12.4 MB/s eta 0:00:00
Downloading alembic-1.17.0-py3-none-any.whl (247 kB)
                         247.4/247.4 kB
24.5 MB/s eta 0:00:00
Downloading
sqlalchemy-2.0.44-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(3.3 MB)
                         3.3/3.3 MB
100.3 MB/s eta 0:00:00
Downloading colorlog-6.10.1-py3-none-any.whl (11 kB)
Downloading
greenlet-3.2.4-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl (607
kB)
                         607.6/607.6 kB
47.9 MB/s eta 0:00:00
Installing collected packages: greenlet, colorlog, sqlalchemy, alembic,
optuna
Successfully installed alembic-1.17.0 colorlog-6.10.1 greenlet-3.2.4
optuna-4.5.0 sqlalchemy-2.0.44
```

[55]:
```
import optuna
```

[56]:
```
def objective(trial):
    criterion = trial.suggest_categorical('criterion', ['gini', 'entropy',␣
 ↪'log_loss'])
    max_depth = trial.suggest_int('max_depth', 2, 20)
    min_samples_split = trial.suggest_int('min_samples_split', 2, 20)
    min_samples_leaf = trial.suggest_int('min_samples_leaf', 1, 10)
    dt_clf = DecisionTreeClassifier(
        criterion=criterion,
        max_depth=max_depth,
        min_samples_split=min_samples_split,
        min_samples_leaf=min_samples_leaf,
        random_state=42
    )
```

```
    score = cross_val_score(dt_clf, X_train, y_train, cv=5, scoring='accuracy').
↪mean()
    return score

study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=50, show_progress_bar=True)

best_params = study.best_params
dt_clf_best = DecisionTreeClassifier(**best_params, random_state=42)
dt_clf_best.fit(X_train, y_train)
y_pred = dt_clf_best.predict(X_test)

test_accuracy_dtt_hp = accuracy_score(y_test, y_pred)
test_precision_dtt_hp = precision_score(y_test, y_pred, average='weighted',␣
↪zero_division=0)
test_recall_dtt_hp = recall_score(y_test, y_pred, average='weighted',␣
↪zero_division=0)
test_f1_dtt_hp = f1_score(y_test, y_pred, average='weighted', zero_division=0)

print("Best parameters:", study.best_params)
print("Best cross-validation accuracy:", study.best_value)
print("--- Decision Tree Model Evaluation (Optimized) ---")
print("Test Accuracy:", test_accuracy_dtt_hp)
print("Test Precision (Weighted):", test_precision_dtt_hp)
print("Test Recall (Weighted):", test_recall_dtt_hp)
print("Test F1-Score (Weighted):", test_f1_dtt_hp)
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

importances = dt_clf_best.feature_importances_
feature_names = X_train.columns
feat_importances = pd.DataFrame({'Feature': feature_names, 'Importance':␣
↪importances}).sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10,6))
plt.barh(feat_importances['Feature'], feat_importances['Importance'],␣
↪color='skyblue')
plt.gca().invert_yaxis()
plt.xlabel("Importance")
plt.title("Feature Importance from Optimized Decision Tree")
plt.show()
```

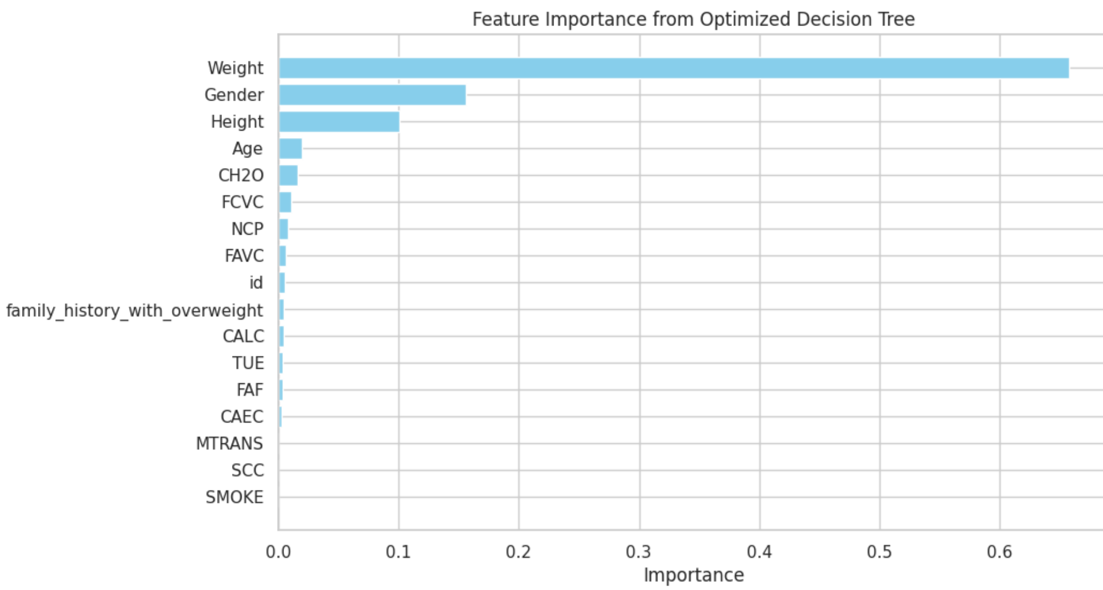
```
[I 2025-10-26 16:26:14,114] A new study created in memory with name: no-
name-e4d43aab-97e3-42da-bdb1-b5154228b207

  0%|          | 0/50 [00:00<?, ?it/s]

[I 2025-10-26 16:26:14,530] Trial 0 finished with value: 0.8670532608361352 and
```

```
parameters: {'criterion': 'log_loss', 'max_depth': 16, 'min_samples_split': 13,
'min_samples_leaf': 9}. Best is trial 0 with value: 0.8670532608361352.
[I 2025-10-26 16:26:14,901] Trial 1 finished with value: 0.8665707195708443 and
parameters: {'criterion': 'gini', 'max_depth': 15, 'min_samples_split': 20,
'min_samples_leaf': 6}. Best is trial 0 with value: 0.8670532608361352.
[I 2025-10-26 16:26:15,320] Trial 2 finished with value: 0.8619832915433066 and
parameters: {'criterion': 'entropy', 'max_depth': 16, 'min_samples_split': 15,
'min_samples_leaf': 5}. Best is trial 0 with value: 0.8670532608361352.
[I 2025-10-26 16:26:15,710] Trial 3 finished with value: 0.8604543754886519 and
parameters: {'criterion': 'gini', 'max_depth': 17, 'min_samples_split': 10,
'min_samples_leaf': 5}. Best is trial 0 with value: 0.8670532608361352.
[I 2025-10-26 16:26:16,035] Trial 4 finished with value: 0.8658457907541791 and
parameters: {'criterion': 'entropy', 'max_depth': 7, 'min_samples_split': 18,
'min_samples_leaf': 7}. Best is trial 0 with value: 0.8670532608361352.
[I 2025-10-26 16:26:16,436] Trial 5 finished with value: 0.8673749981789369 and
parameters: {'criterion': 'log_loss', 'max_depth': 10, 'min_samples_split': 8,
'min_samples_leaf': 1}. Best is trial 5 with value: 0.8673749981789369.
[I 2025-10-26 16:26:16,859] Trial 6 finished with value: 0.8660069184212273 and
parameters: {'criterion': 'entropy', 'max_depth': 18, 'min_samples_split': 5,
'min_samples_leaf': 8}. Best is trial 5 with value: 0.8673749981789369.
[I 2025-10-26 16:26:17,328] Trial 7 finished with value: 0.8627076376197653 and
parameters: {'criterion': 'entropy', 'max_depth': 14, 'min_samples_split': 19,
'min_samples_leaf': 3}. Best is trial 5 with value: 0.8673749981789369.
[I 2025-10-26 16:26:17,764] Trial 8 finished with value: 0.8630296987071262 and
parameters: {'criterion': 'gini', 'max_depth': 17, 'min_samples_split': 12,
'min_samples_leaf': 6}. Best is trial 5 with value: 0.8673749981789369.
[I 2025-10-26 16:26:17,998] Trial 9 finished with value: 0.7824720163296756 and
parameters: {'criterion': 'log_loss', 'max_depth': 4, 'min_samples_split': 6,
'min_samples_leaf': 10}. Best is trial 5 with value: 0.8673749981789369.
[I 2025-10-26 16:26:18,437] Trial 10 finished with value: 0.8687432074344701 and
parameters: {'criterion': 'log_loss', 'max_depth': 10, 'min_samples_split': 8,
'min_samples_leaf': 2}. Best is trial 10 with value: 0.8687432074344701.
[I 2025-10-26 16:26:18,855] Trial 11 finished with value: 0.8673749981789369 and
parameters: {'criterion': 'log_loss', 'max_depth': 10, 'min_samples_split': 8,
'min_samples_leaf': 1}. Best is trial 10 with value: 0.8687432074344701.
[I 2025-10-26 16:26:19,278] Trial 12 finished with value: 0.8599711867342428 and
parameters: {'criterion': 'log_loss', 'max_depth': 11, 'min_samples_split': 2,
'min_samples_leaf': 1}. Best is trial 10 with value: 0.8687432074344701.
[I 2025-10-26 16:26:19,641] Trial 13 finished with value: 0.8672947419027439 and
parameters: {'criterion': 'log_loss', 'max_depth': 8, 'min_samples_split': 9,
'min_samples_leaf': 3}. Best is trial 10 with value: 0.8687432074344701.
[I 2025-10-26 16:26:19,798] Trial 14 finished with value: 0.5961694867515633 and
parameters: {'criterion': 'log_loss', 'max_depth': 2, 'min_samples_split': 5,
'min_samples_leaf': 3}. Best is trial 10 with value: 0.8687432074344701.
[I 2025-10-26 16:26:20,234] Trial 15 finished with value: 0.8561885229316365 and
parameters: {'criterion': 'log_loss', 'max_depth': 13, 'min_samples_split': 7,
'min_samples_leaf': 2}. Best is trial 10 with value: 0.8687432074344701.
[I 2025-10-26 16:26:20,667] Trial 16 finished with value: 0.852567407664005 and
```

```
parameters: {'criterion': 'log_loss', 'max_depth': 20, 'min_samples_split': 3,
'min_samples_leaf': 4}. Best is trial 10 with value: 0.8687432074344701.
[I 2025-10-26 16:26:21,030] Trial 17 finished with value: 0.8664093976570607 and
parameters: {'criterion': 'log_loss', 'max_depth': 8, 'min_samples_split': 15,
'min_samples_leaf': 1}. Best is trial 10 with value: 0.8687432074344701.
[I 2025-10-26 16:26:21,336] Trial 18 finished with value: 0.8532912033747133 and
parameters: {'criterion': 'log_loss', 'max_depth': 6, 'min_samples_split': 10,
'min_samples_leaf': 2}. Best is trial 10 with value: 0.8687432074344701.
[I 2025-10-26 16:26:21,753] Trial 19 finished with value: 0.8615003294100889 and
parameters: {'criterion': 'log_loss', 'max_depth': 12, 'min_samples_split': 4,
'min_samples_leaf': 4}. Best is trial 10 with value: 0.8687432074344701.
[I 2025-10-26 16:26:22,091] Trial 20 finished with value: 0.8680996032510429 and
parameters: {'criterion': 'gini', 'max_depth': 10, 'min_samples_split': 8,
'min_samples_leaf': 2}. Best is trial 10 with value: 0.8687432074344701.
[I 2025-10-26 16:26:22,430] Trial 21 finished with value: 0.8680996032510429 and
parameters: {'criterion': 'gini', 'max_depth': 10, 'min_samples_split': 8,
'min_samples_leaf': 2}. Best is trial 10 with value: 0.8687432074344701.
[I 2025-10-26 16:26:22,749] Trial 22 finished with value: 0.8700311927882662 and
parameters: {'criterion': 'gini', 'max_depth': 9, 'min_samples_split': 11,
'min_samples_leaf': 2}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:22,969] Trial 23 finished with value: 0.8343789850931819 and
parameters: {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 12,
'min_samples_leaf': 4}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:23,287] Trial 24 finished with value: 0.8694677477576642 and
parameters: {'criterion': 'gini', 'max_depth': 9, 'min_samples_split': 14,
'min_samples_leaf': 2}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:23,584] Trial 25 finished with value: 0.8676975448831363 and
parameters: {'criterion': 'gini', 'max_depth': 8, 'min_samples_split': 15,
'min_samples_leaf': 3}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:23,952] Trial 26 finished with value: 0.8675361905948968 and
parameters: {'criterion': 'gini', 'max_depth': 12, 'min_samples_split': 13,
'min_samples_leaf': 2}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:24,266] Trial 27 finished with value: 0.8689850122456381 and
parameters: {'criterion': 'gini', 'max_depth': 9, 'min_samples_split': 17,
'min_samples_leaf': 4}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:24,429] Trial 28 finished with value: 0.7259775547897198 and
parameters: {'criterion': 'gini', 'max_depth': 3, 'min_samples_split': 17,
'min_samples_leaf': 4}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:24,677] Trial 29 finished with value: 0.8381618755169795 and
parameters: {'criterion': 'gini', 'max_depth': 6, 'min_samples_split': 17,
'min_samples_leaf': 5}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:24,995] Trial 30 finished with value: 0.8692263314399673 and
parameters: {'criterion': 'gini', 'max_depth': 9, 'min_samples_split': 14,
'min_samples_leaf': 3}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:25,315] Trial 31 finished with value: 0.8692263314399673 and
parameters: {'criterion': 'gini', 'max_depth': 9, 'min_samples_split': 14,
'min_samples_leaf': 3}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:25,591] Trial 32 finished with value: 0.8653632171144323 and
```

```
parameters: {'criterion': 'gini', 'max_depth': 7, 'min_samples_split': 14,
'min_samples_leaf': 3}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:25,956] Trial 33 finished with value: 0.8665704282007411 and
parameters: {'criterion': 'gini', 'max_depth': 12, 'min_samples_split': 13,
'min_samples_leaf': 3}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:26,279] Trial 34 finished with value: 0.8699506451419701 and
parameters: {'criterion': 'gini', 'max_depth': 9, 'min_samples_split': 11,
'min_samples_leaf': 1}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:26,556] Trial 35 finished with value: 0.8654437000118167 and
parameters: {'criterion': 'gini', 'max_depth': 7, 'min_samples_split': 11,
'min_samples_leaf': 1}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:26,954] Trial 36 finished with value: 0.8595690312429687 and
parameters: {'criterion': 'gini', 'max_depth': 14, 'min_samples_split': 11,
'min_samples_leaf': 1}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:27,268] Trial 37 finished with value: 0.8693068467118075 and
parameters: {'criterion': 'gini', 'max_depth': 9, 'min_samples_split': 16,
'min_samples_leaf': 2}. Best is trial 22 with value: 0.8700311927882662.
[I 2025-10-26 16:26:27,670] Trial 38 finished with value: 0.8717206537697626 and
parameters: {'criterion': 'entropy', 'max_depth': 11, 'min_samples_split': 16,
'min_samples_leaf': 8}. Best is trial 38 with value: 0.8717206537697626.
[I 2025-10-26 16:26:28,072] Trial 39 finished with value: 0.8726867075340214 and
parameters: {'criterion': 'entropy', 'max_depth': 11, 'min_samples_split': 20,
'min_samples_leaf': 8}. Best is trial 39 with value: 0.8726867075340214.
[I 2025-10-26 16:26:28,481] Trial 40 finished with value: 0.8684210815981974 and
parameters: {'criterion': 'entropy', 'max_depth': 15, 'min_samples_split': 19,
'min_samples_leaf': 8}. Best is trial 39 with value: 0.8726867075340214.
[I 2025-10-26 16:26:28,878] Trial 41 finished with value: 0.8726867075340214 and
parameters: {'criterion': 'entropy', 'max_depth': 11, 'min_samples_split': 20,
'min_samples_leaf': 8}. Best is trial 39 with value: 0.8726867075340214.
[I 2025-10-26 16:26:29,271] Trial 42 finished with value: 0.8726867075340214 and
parameters: {'criterion': 'entropy', 'max_depth': 11, 'min_samples_split': 20,
'min_samples_leaf': 8}. Best is trial 39 with value: 0.8726867075340214.
[I 2025-10-26 16:26:29,671] Trial 43 finished with value: 0.8726867075340214 and
parameters: {'criterion': 'entropy', 'max_depth': 11, 'min_samples_split': 20,
'min_samples_leaf': 8}. Best is trial 39 with value: 0.8726867075340214.
[I 2025-10-26 16:26:30,068] Trial 44 finished with value: 0.8726867075340214 and
parameters: {'criterion': 'entropy', 'max_depth': 11, 'min_samples_split': 20,
'min_samples_leaf': 8}. Best is trial 39 with value: 0.8726867075340214.
[I 2025-10-26 16:26:30,473] Trial 45 finished with value: 0.8699503861463228 and
parameters: {'criterion': 'entropy', 'max_depth': 13, 'min_samples_split': 20,
'min_samples_leaf': 9}. Best is trial 39 with value: 0.8726867075340214.
[I 2025-10-26 16:26:30,882] Trial 46 finished with value: 0.8689042056036944 and
parameters: {'criterion': 'entropy', 'max_depth': 13, 'min_samples_split': 19,
'min_samples_leaf': 7}. Best is trial 39 with value: 0.8726867075340214.
[I 2025-10-26 16:26:31,279] Trial 47 finished with value: 0.8727671904314057 and
parameters: {'criterion': 'entropy', 'max_depth': 11, 'min_samples_split': 20,
'min_samples_leaf': 9}. Best is trial 47 with value: 0.8727671904314057.
[I 2025-10-26 16:26:31,689] Trial 48 finished with value: 0.8683408253220044 and
```

```
parameters: {'criterion': 'entropy', 'max_depth': 15, 'min_samples_split': 19,
'min_samples_leaf': 10}. Best is trial 47 with value: 0.8727671904314057.
[I 2025-10-26 16:26:32,101] Trial 49 finished with value: 0.8696284869312414 and
parameters: {'criterion': 'entropy', 'max_depth': 14, 'min_samples_split': 20,
'min_samples_leaf': 9}. Best is trial 47 with value: 0.8727671904314057.
Best parameters: {'criterion': 'entropy', 'max_depth': 11, 'min_samples_split':
20, 'min_samples_leaf': 9}
Best cross-validation accuracy: 0.8727671904314057
--- Decision Tree Model Evaluation (Optimized) ---
Test Accuracy: 0.8725458641776633
Test Precision (Weighted): 0.8733532739578848
Test Recall (Weighted): 0.8725458641776633
Test F1-Score (Weighted): 0.8728714678590848

Confusion Matrix:
 [[312  31   1   1   0   0   1]
 [ 27 414   2   0   0  49   4]
 [  0   1 383  16   1  16  31]
 [  0   0  24 436   0   0   4]
 [  0   0   2   1 603   0   0]
 [  3  40  10   0   0 276  55]
 [  0   6  29   3   0  38 287]]

Classification Report:
                      precision    recall  f1-score   support

Insufficient_Weight       0.91      0.90      0.91       346
      Normal_Weight       0.84      0.83      0.84       496
     Obesity_Type_I       0.85      0.85      0.85       448
    Obesity_Type_II       0.95      0.94      0.95       464
   Obesity_Type_III       1.00      1.00      1.00       606
 Overweight_Level_I       0.73      0.72      0.72       384
Overweight_Level_II       0.75      0.79      0.77       363

           accuracy                           0.87      3107
          macro avg       0.86      0.86      0.86      3107
       weighted avg       0.87      0.87      0.87      3107
```



ACCURACY WITH DECISION TREE USING OPTUNA IS 0.8651432249758609

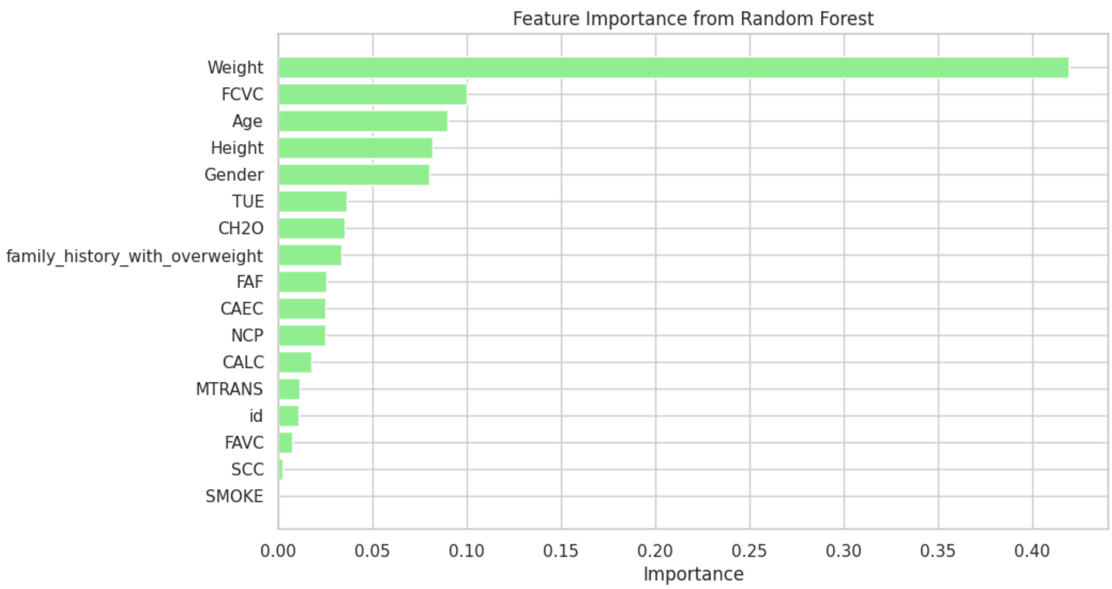
2. RANDOM FOREST

[57]:
```
rf_clf = RandomForestClassifier(
    n_estimators=200,
    criterion='gini',
    max_depth=10,
    min_samples_split=5,
    min_samples_leaf=2,
    max_features='sqrt',
    random_state=42,
    n_jobs=-1
)

rf_clf.fit(X_train, y_train)

y_pred = rf_clf.predict(X_test)

test_accuracy_rf = accuracy_score(y_test, y_pred)
test_precision_rf = precision_score(y_test, y_pred, average='weighted',␣
  ↪zero_division=0)
test_recall_rf = recall_score(y_test, y_pred, average='weighted',␣
  ↪zero_division=0)
test_f1_rf = f1_score(y_test, y_pred, average='weighted', zero_division=0)

print("--- Random Forest Model Evaluation ---")
print("Test Accuracy:", test_accuracy_rf)
```

```
print("Test Precision (Weighted):", test_precision_rf)
print("Test Recall (Weighted):", test_recall_rf)
print("Test F1-Score (Weighted):", test_f1_rf)
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

importances = rf_clf.feature_importances_
feature_names = X_train.columns
feat_importances = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)


plt.figure(figsize=(10,6))
plt.barh(feat_importances['Feature'], feat_importances['Importance'],␣
 ↪color='lightgreen')
plt.gca().invert_yaxis()
plt.xlabel("Importance")
plt.title("Feature Importance from Random Forest")
plt.show()
```

```
--- Random Forest Model Evaluation ---
Test Accuracy: 0.8918570968780174
Test Precision (Weighted): 0.8934727995317859
Test Recall (Weighted): 0.8918570968780174
Test F1-Score (Weighted): 0.891523274112166

Confusion Matrix:
 [[308  35   0   1   0   1   1]
 [ 19 441   1   0   0  28   7]
 [  0   1 395  13   1  10  28]
 [  0   0  12 449   0   0   3]
 [  0   0   1   1 603   1   0]
 [  2  51   8   0   0 267  56]
 [  0  12  20   1   0  22 308]]

Classification Report:
                      precision    recall  f1-score   support

Insufficient_Weight       0.94      0.89      0.91       346
      Normal_Weight       0.82      0.89      0.85       496
     Obesity_Type_I       0.90      0.88      0.89       448
    Obesity_Type_II       0.97      0.97      0.97       464
   Obesity_Type_III       1.00      1.00      1.00       606
 Overweight_Level_I       0.81      0.70      0.75       384
Overweight_Level_II       0.76      0.85      0.80       363
```

```
    accuracy                           0.89      3107
   macro avg       0.89      0.88      0.88      3107
weighted avg       0.89      0.89      0.89      3107
```



THE ACCURACY FROM RANDOM FOREST IS 0.8918570968780174

Hyperparameter Tuning Using Optuna On Random Forest

[58]:
```python
def objective(trial):
    n_estimators = trial.suggest_int('n_estimators', 100, 500)
    max_depth = trial.suggest_int('max_depth', 2, 20)
    min_samples_split = trial.suggest_int('min_samples_split', 2, 20)
    min_samples_leaf = trial.suggest_int('min_samples_leaf', 1, 10)
    max_features = trial.suggest_categorical('max_features', ['sqrt', 'log2',␣
↪None])
    criterion = trial.suggest_categorical('criterion', ['gini', 'entropy',␣
↪'log_loss'])

    rf_clf = RandomForestClassifier(
        n_estimators=n_estimators,
        max_depth=max_depth,
        min_samples_split=min_samples_split,
        min_samples_leaf=min_samples_leaf,
        max_features=max_features,
        criterion=criterion,
        random_state=42,
```

```
        n_jobs=-1
    )

    score = cross_val_score(rf_clf, X_train, y_train, cv=5, scoring='accuracy').
↪mean()
    return score

study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=50, show_progress_bar=True)

best_params = study.best_params
rf_clf_best = RandomForestClassifier(**best_params, random_state=42, n_jobs=-1)
rf_clf_best.fit(X_train, y_train)
y_pred = rf_clf_best.predict(X_test)

test_accuracy_rf_hp = accuracy_score(y_test, y_pred)
test_precision_rf_hp = precision_score(y_test, y_pred, average='weighted',␣
↪zero_division=0)
test_recall_rf_hp = recall_score(y_test, y_pred, average='weighted',␣
↪zero_division=0)
test_f1_rf_hp = f1_score(y_test, y_pred, average='weighted', zero_division=0)

print("Best parameters:", study.best_params)
print("Best cross-validation accuracy:", study.best_value)
print("--- Random Forest Model Evaluation (Optimized) ---")
print("Test Accuracy:", test_accuracy_rf_hp)
print("Test Precision (Weighted):", test_precision_rf_hp)
print("Test Recall (Weighted):", test_recall_rf_hp)
print("Test F1-Score (Weighted):", test_f1_rf_hp)
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

importances = rf_clf_best.feature_importances_
feature_names = X_train.columns
feat_importances = pd.DataFrame({'Feature': feature_names, 'Importance':␣
↪importances}).sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10,6))
plt.barh(feat_importances['Feature'], feat_importances['Importance'],␣
↪color='lightgreen')
plt.gca().invert_yaxis()
plt.xlabel("Importance")
plt.title("Feature Importance from Optimized Random Forest")
plt.show()
```

```
[I 2025-10-26 16:26:33,349] A new study created in memory with name: no-
name-09421bbf-a836-42aa-aab8-94d59390f898
```

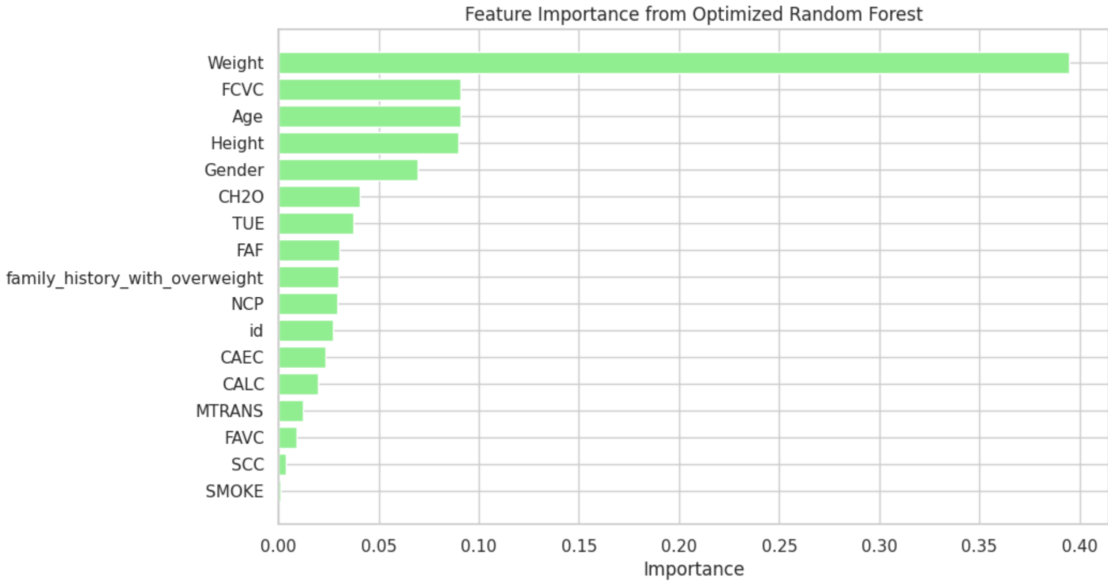
```
  0%|          | 0/50 [00:00<?, ?it/s]

[I 2025-10-26 16:26:46,860] Trial 0 finished with value: 0.8913566677620024 and
parameters: {'n_estimators': 185, 'max_depth': 17, 'min_samples_split': 16,
'min_samples_leaf': 4, 'max_features': None, 'criterion': 'log_loss'}. Best is
trial 0 with value: 0.8913566677620024.
[I 2025-10-26 16:26:54,075] Trial 1 finished with value: 0.8937714136791788 and
parameters: {'n_estimators': 283, 'max_depth': 12, 'min_samples_split': 15,
'min_samples_leaf': 1, 'max_features': 'sqrt', 'criterion': 'entropy'}. Best is
trial 1 with value: 0.8937714136791788.
[I 2025-10-26 16:27:06,595] Trial 2 finished with value: 0.891195637218322 and
parameters: {'n_estimators': 196, 'max_depth': 19, 'min_samples_split': 17,
'min_samples_leaf': 4, 'max_features': None, 'criterion': 'entropy'}. Best is
trial 1 with value: 0.8937714136791788.
[I 2025-10-26 16:27:12,097] Trial 3 finished with value: 0.8944956626322698 and
parameters: {'n_estimators': 255, 'max_depth': 14, 'min_samples_split': 7,
'min_samples_leaf': 3, 'max_features': 'sqrt', 'criterion': 'log_loss'}. Best is
trial 3 with value: 0.8944956626322698.
[I 2025-10-26 16:27:15,310] Trial 4 finished with value: 0.7907624022493772 and
parameters: {'n_estimators': 267, 'max_depth': 4, 'min_samples_split': 12,
'min_samples_leaf': 4, 'max_features': 'sqrt', 'criterion': 'entropy'}. Best is
trial 3 with value: 0.8944956626322698.
[I 2025-10-26 16:27:17,784] Trial 5 finished with value: 0.8894260818329123 and
parameters: {'n_estimators': 125, 'max_depth': 10, 'min_samples_split': 9,
'min_samples_leaf': 6, 'max_features': 'sqrt', 'criterion': 'entropy'}. Best is
trial 3 with value: 0.8944956626322698.
[I 2025-10-26 16:27:19,391] Trial 6 finished with value: 0.7732178104831725 and
parameters: {'n_estimators': 151, 'max_depth': 4, 'min_samples_split': 3,
'min_samples_leaf': 4, 'max_features': 'sqrt', 'criterion': 'gini'}. Best is
trial 3 with value: 0.8944956626322698.
[I 2025-10-26 16:27:26,445] Trial 7 finished with value: 0.8959441281639962 and
parameters: {'n_estimators': 320, 'max_depth': 16, 'min_samples_split': 11,
'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'entropy'}. Best is
trial 7 with value: 0.8959441281639962.
[I 2025-10-26 16:27:31,083] Trial 8 finished with value: 0.8859652524964752 and
parameters: {'n_estimators': 303, 'max_depth': 11, 'min_samples_split': 6,
'min_samples_leaf': 10, 'max_features': 'sqrt', 'criterion': 'gini'}. Best is
trial 7 with value: 0.8959441281639962.
[I 2025-10-26 16:27:40,822] Trial 9 finished with value: 0.8966683447426312 and
parameters: {'n_estimators': 429, 'max_depth': 15, 'min_samples_split': 4,
'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'log_loss'}. Best is
trial 9 with value: 0.8966683447426312.
[I 2025-10-26 16:27:49,131] Trial 10 finished with value: 0.8822635248336358 and
parameters: {'n_estimators': 467, 'max_depth': 8, 'min_samples_split': 3,
'min_samples_leaf': 7, 'max_features': 'log2', 'criterion': 'log_loss'}. Best is
trial 9 with value: 0.8966683447426312.
[I 2025-10-26 16:27:57,780] Trial 11 finished with value: 0.8947372084477905 and
parameters: {'n_estimators': 407, 'max_depth': 16, 'min_samples_split': 20,
```

```
'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'log_loss'}. Best is
trial 9 with value: 0.8966683447426312.
[I 2025-10-26 16:28:05,720] Trial 12 finished with value: 0.895461619273161 and
parameters: {'n_estimators': 371, 'max_depth': 20, 'min_samples_split': 12,
'min_samples_leaf': 2, 'max_features': 'log2', 'criterion': 'entropy'}. Best is
trial 9 with value: 0.8966683447426312.
[I 2025-10-26 16:28:14,398] Trial 13 finished with value: 0.8966683771170871 and
parameters: {'n_estimators': 376, 'max_depth': 15, 'min_samples_split': 7,
'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'log_loss'}. Best is
trial 13 with value: 0.8966683771170871.
[I 2025-10-26 16:28:23,904] Trial 14 finished with value: 0.8891040531200073 and
parameters: {'n_estimators': 493, 'max_depth': 14, 'min_samples_split': 6,
'min_samples_leaf': 10, 'max_features': 'log2', 'criterion': 'log_loss'}. Best
is trial 13 with value: 0.8966683771170871.
[I 2025-10-26 16:28:31,259] Trial 15 finished with value: 0.8830679976884639 and
parameters: {'n_estimators': 418, 'max_depth': 8, 'min_samples_split': 3,
'min_samples_leaf': 2, 'max_features': 'log2', 'criterion': 'log_loss'}. Best is
trial 13 with value: 0.8966683771170871.
[I 2025-10-26 16:28:38,348] Trial 16 finished with value: 0.890954933138655 and
parameters: {'n_estimators': 357, 'max_depth': 14, 'min_samples_split': 8,
'min_samples_leaf': 7, 'max_features': 'log2', 'criterion': 'log_loss'}. Best is
trial 13 with value: 0.8966683771170871.
[I 2025-10-26 16:28:46,248] Trial 17 finished with value: 0.8959442900362756 and
parameters: {'n_estimators': 435, 'max_depth': 18, 'min_samples_split': 5,
'min_samples_leaf': 2, 'max_features': 'log2', 'criterion': 'gini'}. Best is
trial 13 with value: 0.8966683771170871.
[I 2025-10-26 16:29:06,627] Trial 18 finished with value: 0.8897472040610518 and
parameters: {'n_estimators': 360, 'max_depth': 13, 'min_samples_split': 10,
'min_samples_leaf': 8, 'max_features': None, 'criterion': 'log_loss'}. Best is
trial 13 with value: 0.8966683771170871.
[I 2025-10-26 16:29:14,924] Trial 19 finished with value: 0.8875747485718819 and
parameters: {'n_estimators': 449, 'max_depth': 9, 'min_samples_split': 2,
'min_samples_leaf': 3, 'max_features': 'log2', 'criterion': 'log_loss'}. Best is
trial 13 with value: 0.8966683771170871.
[I 2025-10-26 16:29:23,028] Trial 20 finished with value: 0.894254278689029 and
parameters: {'n_estimators': 393, 'max_depth': 16, 'min_samples_split': 5,
'min_samples_leaf': 5, 'max_features': 'log2', 'criterion': 'log_loss'}. Best is
trial 13 with value: 0.8966683771170871.
[I 2025-10-26 16:29:30,847] Trial 21 finished with value: 0.8959442900362756 and
parameters: {'n_estimators': 433, 'max_depth': 18, 'min_samples_split': 5,
'min_samples_leaf': 2, 'max_features': 'log2', 'criterion': 'gini'}. Best is
trial 13 with value: 0.8966683771170871.
[I 2025-10-26 16:29:39,923] Trial 22 finished with value: 0.8960246758102921 and
parameters: {'n_estimators': 491, 'max_depth': 20, 'min_samples_split': 5,
'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'gini'}. Best is
trial 13 with value: 0.8966683771170871.
[I 2025-10-26 16:29:48,743] Trial 23 finished with value: 0.8966684742404547 and
parameters: {'n_estimators': 485, 'max_depth': 20, 'min_samples_split': 8,
```

```
'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'gini'}. Best is
trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:29:56,676] Trial 24 finished with value: 0.8933689020688895 and
parameters: {'n_estimators': 463, 'max_depth': 15, 'min_samples_split': 8,
'min_samples_leaf': 3, 'max_features': 'log2', 'criterion': 'gini'}. Best is
trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:30:16,042] Trial 25 finished with value: 0.890692975228687 and
parameters: {'n_estimators': 342, 'max_depth': 18, 'min_samples_split': 9,
'min_samples_leaf': 1, 'max_features': None, 'criterion': 'gini'}. Best is trial
23 with value: 0.8966684742404547.
[I 2025-10-26 16:30:19,738] Trial 26 finished with value: 0.6356837404151376 and
parameters: {'n_estimators': 397, 'max_depth': 2, 'min_samples_split': 13,
'min_samples_leaf': 2, 'max_features': 'log2', 'criterion': 'log_loss'}. Best is
trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:30:27,854] Trial 27 finished with value: 0.8937714460536348 and
parameters: {'n_estimators': 484, 'max_depth': 12, 'min_samples_split': 7,
'min_samples_leaf': 3, 'max_features': 'log2', 'criterion': 'gini'}. Best is
trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:30:35,791] Trial 28 finished with value: 0.893449384966274 and
parameters: {'n_estimators': 386, 'max_depth': 20, 'min_samples_split': 4,
'min_samples_leaf': 5, 'max_features': 'log2', 'criterion': 'log_loss'}. Best is
trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:30:56,904] Trial 29 finished with value: 0.8918396622696759 and
parameters: {'n_estimators': 334, 'max_depth': 17, 'min_samples_split': 10,
'min_samples_leaf': 1, 'max_features': None, 'criterion': 'log_loss'}. Best is
trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:31:04,577] Trial 30 finished with value: 0.895059075288416 and
parameters: {'n_estimators': 427, 'max_depth': 17, 'min_samples_split': 7,
'min_samples_leaf': 2, 'max_features': 'log2', 'criterion': 'gini'}. Best is
trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:31:13,800] Trial 31 finished with value: 0.8957831299947715 and
parameters: {'n_estimators': 491, 'max_depth': 20, 'min_samples_split': 4,
'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'gini'}. Best is
trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:31:22,585] Trial 32 finished with value: 0.8957831623692275 and
parameters: {'n_estimators': 460, 'max_depth': 19, 'min_samples_split': 2,
'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'gini'}. Best is
trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:31:31,247] Trial 33 finished with value: 0.8965879913430704 and
parameters: {'n_estimators': 475, 'max_depth': 19, 'min_samples_split': 6,
'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'gini'}. Best is
trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:31:55,312] Trial 34 finished with value: 0.8912763467368976 and
parameters: {'n_estimators': 455, 'max_depth': 15, 'min_samples_split': 8,
'min_samples_leaf': 3, 'max_features': None, 'criterion': 'gini'}. Best is trial
23 with value: 0.8966684742404547.
[I 2025-10-26 16:32:04,446] Trial 35 finished with value: 0.8949785600165757 and
parameters: {'n_estimators': 415, 'max_depth': 19, 'min_samples_split': 6,
```

'min_samples_leaf': 2, 'max_features': 'log2', 'criterion': 'log_loss'}. Best is trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:32:12,065] Trial 36 finished with value: 0.8936102860121308 and parameters: {'n_estimators': 439, 'max_depth': 13, 'min_samples_split': 9, 'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'gini'}. Best is trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:32:17,045] Trial 37 finished with value: 0.8923226891518053 and parameters: {'n_estimators': 238, 'max_depth': 18, 'min_samples_split': 16, 'min_samples_leaf': 4, 'max_features': 'sqrt', 'criterion': 'entropy'}. Best is trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:32:46,623] Trial 38 finished with value: 0.8922419472587739 and parameters: {'n_estimators': 479, 'max_depth': 15, 'min_samples_split': 7, 'min_samples_leaf': 2, 'max_features': None, 'criterion': 'log_loss'}. Best is trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:32:54,488] Trial 39 finished with value: 0.8952198468364492 and parameters: {'n_estimators': 378, 'max_depth': 17, 'min_samples_split': 13, 'min_samples_leaf': 3, 'max_features': 'sqrt', 'criterion': 'entropy'}. Best is trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:32:58,801] Trial 40 finished with value: 0.8874944922956889 and parameters: {'n_estimators': 274, 'max_depth': 11, 'min_samples_split': 4, 'min_samples_leaf': 9, 'max_features': 'log2', 'criterion': 'gini'}. Best is trial 23 with value: 0.8966684742404547.
[I 2025-10-26 16:33:08,011] Trial 41 finished with value: 0.8967489571378392 and parameters: {'n_estimators': 500, 'max_depth': 19, 'min_samples_split': 6, 'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'gini'}. Best is trial 41 with value: 0.8967489571378392.
[I 2025-10-26 16:33:17,268] Trial 42 finished with value: 0.8967489571378392 and parameters: {'n_estimators': 500, 'max_depth': 19, 'min_samples_split': 6, 'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'gini'}. Best is trial 41 with value: 0.8967489571378392.
[I 2025-10-26 16:33:26,265] Trial 43 finished with value: 0.8953004268572011 and parameters: {'n_estimators': 497, 'max_depth': 16, 'min_samples_split': 7, 'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'gini'}. Best is trial 41 with value: 0.8967489571378392.
[I 2025-10-26 16:33:34,237] Trial 44 finished with value: 0.8956222613233706 and parameters: {'n_estimators': 450, 'max_depth': 19, 'min_samples_split': 10, 'min_samples_leaf': 2, 'max_features': 'log2', 'criterion': 'gini'}. Best is trial 41 with value: 0.8967489571378392.
[I 2025-10-26 16:33:40,073] Trial 45 finished with value: 0.8532920451105668 and parameters: {'n_estimators': 471, 'max_depth': 6, 'min_samples_split': 4, 'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'gini'}. Best is trial 41 with value: 0.8967489571378392.
[I 2025-10-26 16:33:44,730] Trial 46 finished with value: 0.8924032691725575 and parameters: {'n_estimators': 214, 'max_depth': 17, 'min_samples_split': 19, 'min_samples_leaf': 2, 'max_features': 'sqrt', 'criterion': 'entropy'}. Best is trial 41 with value: 0.8967489571378392.
[I 2025-10-26 16:33:55,867] Trial 47 finished with value: 0.8960247081847481 and parameters: {'n_estimators': 500, 'max_depth': 20, 'min_samples_split': 8,

```
'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'log_loss'}. Best is
trial 41 with value: 0.8967489571378392.
[I 2025-10-26 16:33:59,263] Trial 48 finished with value: 0.8935299002381141 and
parameters: {'n_estimators': 159, 'max_depth': 13, 'min_samples_split': 6,
'min_samples_leaf': 4, 'max_features': 'log2', 'criterion': 'log_loss'}. Best is
trial 41 with value: 0.8967489571378392.
[I 2025-10-26 16:34:06,649] Trial 49 finished with value: 0.895380683133394 and
parameters: {'n_estimators': 407, 'max_depth': 18, 'min_samples_split': 3,
'min_samples_leaf': 2, 'max_features': 'log2', 'criterion': 'gini'}. Best is
trial 41 with value: 0.8967489571378392.
Best parameters: {'n_estimators': 500, 'max_depth': 19, 'min_samples_split': 6,
'min_samples_leaf': 1, 'max_features': 'log2', 'criterion': 'gini'}
Best cross-validation accuracy: 0.8967489571378392
--- Random Forest Model Evaluation (Optimized) ---
Test Accuracy: 0.8982941744448021
Test Precision (Weighted): 0.8989410211193476
Test Recall (Weighted): 0.8982941744448021
Test F1-Score (Weighted): 0.898414759101369

Confusion Matrix:
 [[310  33   0   1   0   1   1]
 [ 20 433   2   0   0  35   6]
 [  0   1 400  13   1  10  23]
 [  0   0  12 449   0   0   3]
 [  0   0   1   1 603   1   0]
 [  2  34   8   0   0 292  48]
 [  0   9  19   2   0  29 304]]

Classification Report:
                      precision    recall  f1-score   support

Insufficient_Weight       0.93      0.90      0.91       346
      Normal_Weight       0.85      0.87      0.86       496
     Obesity_Type_I       0.90      0.89      0.90       448
    Obesity_Type_II       0.96      0.97      0.97       464
   Obesity_Type_III       1.00      1.00      1.00       606
 Overweight_Level_I       0.79      0.76      0.78       384
Overweight_Level_II       0.79      0.84      0.81       363

           accuracy                           0.90      3107
          macro avg       0.89      0.89      0.89      3107
       weighted avg       0.90      0.90      0.90      3107
```



THE ACCURACY FROM RANDOM FOREST USING OPTUNA IS 0.8999034438364982

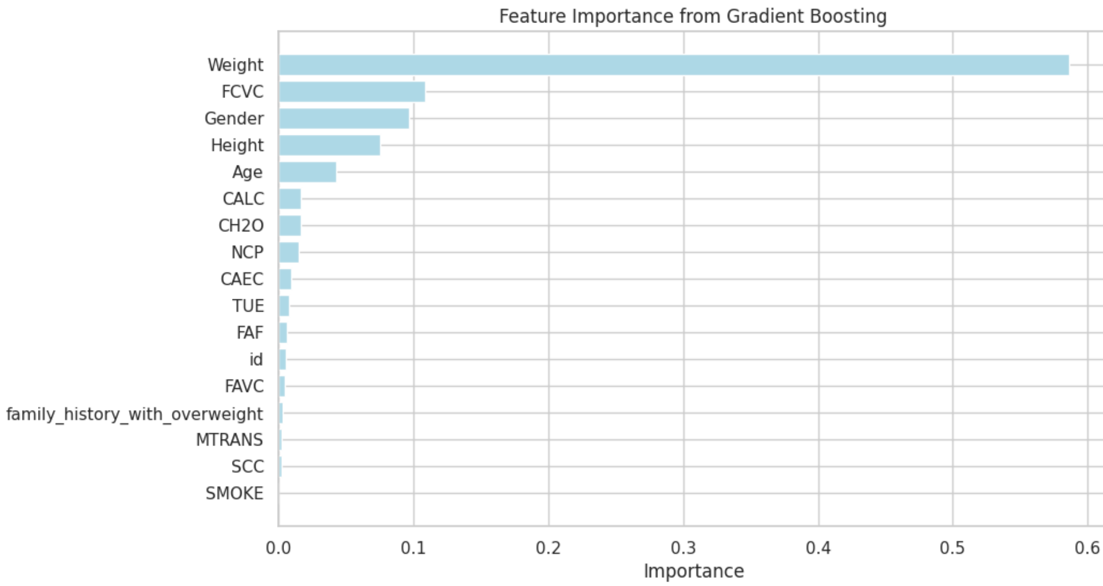
3. GRADIENT BOOSTING

[59]:
```
gb_clf = GradientBoostingClassifier(
    n_estimators=200,
    learning_rate=0.1,
    max_depth=3,
    min_samples_split=5,
    min_samples_leaf=2,
    subsample=0.8,
    random_state=42
)

gb_clf.fit(X_train, y_train)

y_pred = gb_clf.predict(X_test)

test_accuracy_gb = accuracy_score(y_test, y_pred)
test_precision_gb = precision_score(y_test, y_pred, average='weighted',␣
 ↪zero_division=0)
test_recall_gb = recall_score(y_test, y_pred, average='weighted',␣
 ↪zero_division=0)
test_f1_gb = f1_score(y_test, y_pred, average='weighted', zero_division=0)

print("--- Gradient Boosting Model Evaluation ---")
print("Test Accuracy:", test_accuracy_gb)
```

```python
print("Test Precision (Weighted):", test_precision_gb)
print("Test Recall (Weighted):", test_recall_gb)
print("Test F1-Score (Weighted):", test_f1_gb)
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

importances = gb_clf.feature_importances_
feature_names = X_train.columns
feat_importances = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10,6))
plt.barh(feat_importances['Feature'], feat_importances['Importance'],␣
 ↪color='lightblue')
plt.gca().invert_yaxis()
plt.xlabel("Importance")
plt.title("Feature Importance from Gradient Boosting")
plt.show()
```

```
--- Gradient Boosting Model Evaluation ---
Test Accuracy: 0.8979723205664628
Test Precision (Weighted): 0.8982407446122279
Test Recall (Weighted): 0.8979723205664628
Test F1-Score (Weighted): 0.8980677084050355

Confusion Matrix:
 [[312  31   1   1   0   0   1]
 [ 25 432   2   0   0  34   3]
 [  1   0 397  11   1  12  26]
 [  0   0  16 448   0   0   0]
 [  0   0   2   1 603   0   0]
 [  2  34   6   0   0 299  43]
 [  0   7  21   3   0  33 299]]

Classification Report:
                     precision    recall  f1-score   support

Insufficient_Weight       0.92      0.90      0.91       346
      Normal_Weight       0.86      0.87      0.86       496
     Obesity_Type_I       0.89      0.89      0.89       448
    Obesity_Type_II       0.97      0.97      0.97       464
   Obesity_Type_III       1.00      1.00      1.00       606
 Overweight_Level_I       0.79      0.78      0.78       384
Overweight_Level_II       0.80      0.82      0.81       363
```

```
    accuracy                           0.90      3107
   macro avg       0.89      0.89      0.89      3107
weighted avg       0.90      0.90      0.90      3107
```



THE ACCURACY USING GRADIENT BOOSTING IS 0.8979723205664628

[60]:
```
!pip install xgboost
```

```
Collecting xgboost
  Downloading xgboost-3.1.1-py3-none-manylinux_2_28_x86_64.whl.metadata (2.1 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages
(from xgboost) (2.0.2)
Collecting nvidia-nccl-cu12 (from xgboost)
  Downloading nvidia_nccl_cu12-2.28.7-py3-none-
manylinux_2_18_x86_64.whl.metadata (2.0 kB)
Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-packages
(from xgboost) (1.16.2)
Downloading xgboost-3.1.1-py3-none-manylinux_2_28_x86_64.whl (115.9 MB)
                          115.9/115.9 MB
11.0 MB/s eta 0:00:00
Downloading nvidia_nccl_cu12-2.28.7-py3-none-manylinux_2_18_x86_64.whl
(296.8 MB)
                          296.8/296.8 MB
1.3 MB/s eta 0:00:00
Installing collected packages: nvidia-nccl-cu12, xgboost
Successfully installed nvidia-nccl-cu12-2.28.7 xgboost-3.1.1
```

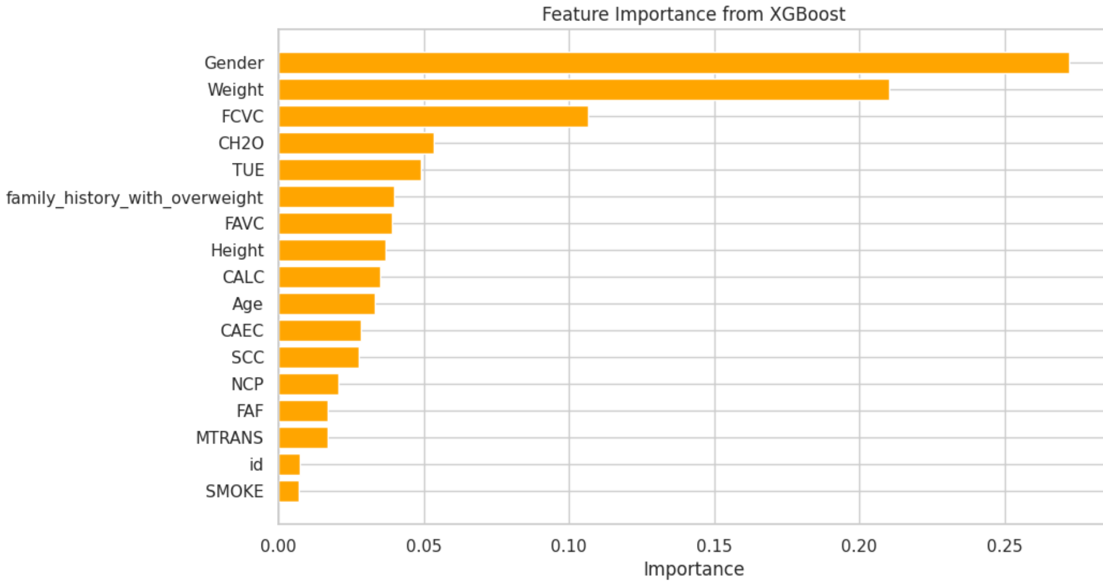
4. XGBOOST

[61]:
```python
from xgboost import XGBClassifier
```

[62]:
```python
y_enc = le.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(
    x, y_enc, test_size=0.2, random_state=42
)

xgb_clf = XGBClassifier(
    n_estimators=200,
    max_depth=4,
    learning_rate=0.1,
    subsample=0.8,
    colsample_bytree=0.8,
    gamma=0,
    reg_alpha=0,
    reg_lambda=1,
    random_state=42,
    eval_metric='mlogloss'
)

xgb_clf.fit(X_train, y_train)

y_pred_enc = xgb_clf.predict(X_test)

y_pred = le.inverse_transform(y_pred_enc)
y_test_orig = le.inverse_transform(y_test)

test_accuracy_xgb = accuracy_score(y_test_orig, y_pred)
test_precision_xgb = precision_score(y_test_orig, y_pred, average='weighted',␣
 ↪zero_division=0)
test_recall_xgb = recall_score(y_test_orig, y_pred, average='weighted',␣
 ↪zero_division=0)
test_f1_xgb = f1_score(y_test_orig, y_pred, average='weighted', zero_division=0)

print("--- XGBoost Model Evaluation ---")
print("Test Accuracy:", test_accuracy_xgb)
print("Test Precision (Weighted):", test_precision_xgb)
print("Test Recall (Weighted):", test_recall_xgb)
print("Test F1-Score (Weighted):", test_f1_xgb)
print("\nConfusion Matrix:\n", confusion_matrix(y_test_orig, y_pred))
print("\nClassification Report:\n", classification_report(y_test_orig, y_pred))

importances = xgb_clf.feature_importances_
feature_names = X_train.columns
feat_importances = pd.DataFrame({
    'Feature': feature_names,
```

```python
    'Importance': importances
}).sort_values(by='Importance', ascending=False)


plt.figure(figsize=(10,6))
plt.barh(feat_importances['Feature'], feat_importances['Importance'],
 ↪color='orange')
plt.gca().invert_yaxis()
plt.xlabel("Importance")
plt.title("Feature Importance from XGBoost")
plt.show()
```

```
--- XGBoost Model Evaluation ---
Test Accuracy: 0.9034438364982298
Test Precision (Weighted): 0.9039893335772461
Test Recall (Weighted): 0.9034438364982298
Test F1-Score (Weighted): 0.9036474236074407

Confusion Matrix:
 [[313  30   1   1   0   0   1]
 [ 19 435   2   0   0  35   5]
 [  0   0 403  10   1  12  22]
 [  0   0  11 452   0   0   1]
 [  0   0   1   1 603   0   1]
 [  2  33   4   0   0 301  44]
 [  0   7  22   1   0  33 300]]

Classification Report:
                      precision    recall  f1-score   support

Insufficient_Weight       0.94      0.90      0.92       346
      Normal_Weight       0.86      0.88      0.87       496
     Obesity_Type_I       0.91      0.90      0.90       448
    Obesity_Type_II       0.97      0.97      0.97       464
   Obesity_Type_III       1.00      1.00      1.00       606
 Overweight_Level_I       0.79      0.78      0.79       384
Overweight_Level_II       0.80      0.83      0.81       363

           accuracy                           0.90      3107
          macro avg       0.90      0.89      0.89      3107
       weighted avg       0.90      0.90      0.90      3107
```



THE ACCURACY USING XGBOOST IS 0.9034438364982298

Hyperparameter Tuning Using Optuna On XGBoost

[63]:
```
# ============================================================
#   Optuna Optimization + Final XGBoost + Kaggle Submission
# ============================================================

import optuna
from optuna.samplers import TPESampler
from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score, StratifiedKFold,␣
↪train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# -----------------------------
# 1  Load Data
# -----------------------------
train_df = pd.read_csv("train.csv")
test_df = pd.read_csv("test.csv")

# Separate features and target
X = train_df.drop('WeightCategory', axis=1)
y = train_df['WeightCategory']
```

```
# Drop ID if present
if 'id' in X.columns:
    X = X.drop('id', axis=1)

test_ids = test_df['id'] if 'id' in test_df.columns else None
test_features = test_df.drop('id', axis=1, errors='ignore')

# -----------------------------
# 2 Encode categorical variables
# -----------------------------
le_target = LabelEncoder()
y_enc = le_target.fit_transform(y)

# Label encode categorical columns for both train and test
X_encoded = X.copy()
test_encoded = test_features.copy()

for col in X_encoded.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    X_encoded[col] = le.fit_transform(X_encoded[col])
    test_encoded[col] = le.transform(test_encoded[col])  # same encoder on test

# Split data for evaluation
X_train, X_valid, y_train, y_valid = train_test_split(
    X_encoded, y_enc, test_size=0.2, random_state=42, stratify=y_enc
)

print(f" Training samples: {X_train.shape[0]} | Features: {X_train.shape[1]}")

# -----------------------------
# 3 Your three best parameter sets
# -----------------------------
base_params_list = [
   {'n_estimators': 802, 'learning_rate': 0.01376727840442897, 'max_depth': 10,␣
 ↪'min_child_weight': 4, 'subsample': 0.7294665487567288, 'colsample_bytree':␣
 ↪0.5546856654158943, 'gamma': 0.16215372378274814, 'reg_alpha': 0.
 ↪1275266644210375, 'reg_lambda': 1.6207767848777366},

   {'n_estimators': 771, 'learning_rate': 0.013703494452479268, 'max_depth':␣
 ↪10, 'min_child_weight': 4, 'subsample': 0.7002200873954504,␣
 ↪'colsample_bytree': 0.5528037754800043, 'gamma': 0.1951947752165459,␣
 ↪'reg_alpha': 0.1397555185137831, 'reg_lambda': 1.6147146257448057},
```

```
    {'n_estimators': 809, 'learning_rate': 0.013314881450029495, 'max_depth':␣
 ↪9, 'min_child_weight': 3, 'subsample': 0.7675430727257503,␣
 ↪'colsample_bytree': 0.5599981247033479, 'gamma': 0.1416646576710503,␣
 ↪'reg_alpha': 0.12140295992038691, 'reg_lambda': 1.5505437116668446}
]


# Compute mean center point
mean_params = {k: np.mean([p[k] for p in base_params_list]) for k in␣
 ↪base_params_list[0]}

# -----------------------------
# 4  Optuna objective function
# -----------------------------
def objective(trial):
    params = {
        'n_estimators': trial.suggest_int('n_estimators', 680, 820),
        'learning_rate': trial.suggest_float('learning_rate', 0.010, 0.016,␣
 ↪log=True),
        'max_depth': trial.suggest_int('max_depth', 8, 10),
        'min_child_weight': trial.suggest_int('min_child_weight', 3, 6),
        'subsample': trial.suggest_float('subsample', 0.70, 0.80),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.55, 0.60),
        'gamma': trial.suggest_float('gamma', 0.13, 0.20),
        'reg_alpha': trial.suggest_float('reg_alpha', 0.07, 0.14),
        'reg_lambda': trial.suggest_float('reg_lambda', 1.5, 1.7),
        'eval_metric': 'mlogloss',
        'tree_method': 'hist',
        'random_state': 42,
        'n_jobs': -1
    }

    model = XGBClassifier(**params)
    cv = StratifiedKFold(n_splits=7, shuffle=True, random_state=42)
    scores = cross_val_score(model, X_encoded, y_enc, cv=cv,␣
 ↪scoring='accuracy', n_jobs=-1)
    return scores.mean()


# -----------------------------
# 5  Run Optuna (fewer than 100 trials)
# -----------------------------
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=60, show_progress_bar=True)
print("Best params:", study.best_params)
print("Best accuracy:", study.best_value)
```

```python
# ------------------------------
# 6  Train final model
# ------------------------------
best_params = study.best_trial.params
final_model = XGBClassifier(**best_params)
final_model.fit(X_encoded, y_enc)

# ------------------------------
#   6.1 Evaluate Final Model
# ------------------------------
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, confusion_matrix, classification_report

# Predict on validation set
y_pred_valid = final_model.predict(X_valid)

# Compute metrics
test_accuracy_xgb_hp = accuracy_score(y_valid, y_pred_valid)
test_precision_xgb_hp = precision_score(y_valid, y_pred_valid,␣
↪average='weighted')
test_recall_xgb_hp = recall_score(y_valid, y_pred_valid, average='weighted')
test_f1_xgb_hp = f1_score(y_valid, y_pred_valid, average='weighted')
cm = confusion_matrix(y_valid, y_pred_valid)

# Display results
print("\n  Model Evaluation on Validation Set")
print("=====================================")
print("Accuracy:",test_accuracy_xgb_hp)
print("Precision:",test_precision_xgb_hp)
print("Recall:",test_recall_xgb_hp)
print("F1 Score:",test_f1_xgb_hp)
print("\nConfusion Matrix:\n", cm)
print("\nClassification Report:\n", classification_report(y_valid,␣
↪y_pred_valid))

# ------------------------------
# 7  Make Predictions on Test Data
# ------------------------------
test_preds_enc = final_model.predict(test_encoded)
test_preds = le_target.inverse_transform(test_preds_enc)

# ------------------------------
# 8  Prepare Submission File
# ------------------------------
submission = pd.DataFrame({
    "id": test_ids,
    "WeightCategory": test_preds
```

```
})

submission.to_csv("submission.csv", index=False)
print("\n  Submission file saved as 'submission.csv'")

# -----------------------------
# 9  Optional: Feature Importance Plot
# -----------------------------
feat_imp = pd.DataFrame({
    "Feature": X_encoded.columns,
    "Importance": final_model.feature_importances_
}).sort_values(by="Importance", ascending=False)

plt.figure(figsize=(10,6))
plt.barh(feat_imp["Feature"][:20], feat_imp["Importance"][:20],␣
 ↪color="darkorange")
plt.gca().invert_yaxis()
plt.title("Top 20 Feature Importances - Final XGBoost Model")
plt.show()
```

```
[I 2025-10-26 16:34:58,955] A new study created in memory with name: no-name-5aa55680-6bdc-475b-98b7-52225fdc09eb

 Training samples: 12426 | Features: 16

  0%|          | 0/60 [00:00<?, ?it/s]

[I 2025-10-26 16:35:18,549] Trial 0 finished with value: 0.9074229060709458 and parameters: {'n_estimators': 767, 'learning_rate': 0.011630836979826885, 'max_depth': 10, 'min_child_weight': 6, 'subsample': 0.7802913074065987, 'colsample_bytree': 0.5581257000430231, 'gamma': 0.15183002639795884, 'reg_alpha': 0.13868042954928156, 'reg_lambda': 1.6163301843232656}. Best is trial 0 with value: 0.9074229060709458.
[I 2025-10-26 16:35:35,052] Trial 1 finished with value: 0.9074229060709457 and parameters: {'n_estimators': 795, 'learning_rate': 0.013231395935522297, 'max_depth': 9, 'min_child_weight': 6, 'subsample': 0.7121614722908453, 'colsample_bytree': 0.5755879996239438, 'gamma': 0.1502205983473868, 'reg_alpha': 0.0850250059080261, 'reg_lambda': 1.613935063799916}. Best is trial 0 with value: 0.9074229060709458.
[I 2025-10-26 16:35:52,593] Trial 2 finished with value: 0.9080666967102299 and parameters: {'n_estimators': 768, 'learning_rate': 0.01477392524322971, 'max_depth': 10, 'min_child_weight': 4, 'subsample': 0.7074642147494602, 'colsample_bytree': 0.5571800655486081, 'gamma': 0.18105857858206562, 'reg_alpha': 0.13247485721429145, 'reg_lambda': 1.6920336068973931}. Best is trial 2 with value: 0.9080666967102299.
[I 2025-10-26 16:36:08,989] Trial 3 finished with value: 0.9074229060709458 and parameters: {'n_estimators': 706, 'learning_rate': 0.010941544935194423, 'max_depth': 10, 'min_child_weight': 4, 'subsample': 0.718175497563846, 'colsample_bytree': 0.5559928939284153, 'gamma': 0.1917246458534284,
```

'reg_alpha': 0.10025139126185875, 'reg_lambda': 1.6636251539060338}. Best is trial 2 with value: 0.9080666967102299.
[I 2025-10-26 16:36:24,539] Trial 4 finished with value: 0.9066503573038049 and parameters: {'n_estimators': 702, 'learning_rate': 0.015390765182408157, 'max_depth': 10, 'min_child_weight': 5, 'subsample': 0.7656988075043349, 'colsample_bytree': 0.5853334714001831, 'gamma': 0.175799191457729, 'reg_alpha': 0.13901504918596808, 'reg_lambda': 1.6915434754057168}. Best is trial 2 with value: 0.9080666967102299.
[I 2025-10-26 16:36:43,388] Trial 5 finished with value: 0.9072297688791605 and parameters: {'n_estimators': 800, 'learning_rate': 0.015039492050992073, 'max_depth': 10, 'min_child_weight': 3, 'subsample': 0.7264218378566091, 'colsample_bytree': 0.5679230030822933, 'gamma': 0.17123185504873917, 'reg_alpha': 0.12761671247996187, 'reg_lambda': 1.5953633316988693}. Best is trial 2 with value: 0.9080666967102299.
[I 2025-10-26 16:37:00,300] Trial 6 finished with value: 0.9077448013905878 and parameters: {'n_estimators': 820, 'learning_rate': 0.010985927747306435, 'max_depth': 8, 'min_child_weight': 3, 'subsample': 0.7169722254983124, 'colsample_bytree': 0.5565111770993991, 'gamma': 0.18312221486301894, 'reg_alpha': 0.10722523950021116, 'reg_lambda': 1.554281738763959}. Best is trial 2 with value: 0.9080666967102299.
[I 2025-10-26 16:37:15,733] Trial 7 finished with value: 0.9073585270070171 and parameters: {'n_estimators': 726, 'learning_rate': 0.010560700241351742, 'max_depth': 8, 'min_child_weight': 3, 'subsample': 0.7511627853364868, 'colsample_bytree': 0.5931182564312805, 'gamma': 0.1713135919769499, 'reg_alpha': 0.1125422418726002, 'reg_lambda': 1.6412260312263287}. Best is trial 2 with value: 0.9080666967102299.
[I 2025-10-26 16:37:34,272] Trial 8 finished with value: 0.9081310757741583 and parameters: {'n_estimators': 789, 'learning_rate': 0.0137885644388524, 'max_depth': 10, 'min_child_weight': 3, 'subsample': 0.7383481993859331, 'colsample_bytree': 0.5588749158964819, 'gamma': 0.16596979347090798, 'reg_alpha': 0.079093945187112, 'reg_lambda': 1.6659216524948643}. Best is trial 8 with value: 0.9081310757741583.
[I 2025-10-26 16:37:49,500] Trial 9 finished with value: 0.9071010107513037 and parameters: {'n_estimators': 707, 'learning_rate': 0.010961879386390034, 'max_depth': 9, 'min_child_weight': 6, 'subsample': 0.7250552074803447, 'colsample_bytree': 0.581871959328018, 'gamma': 0.13504834191013523, 'reg_alpha': 0.08852638143497504, 'reg_lambda': 1.6756209138827185}. Best is trial 8 with value: 0.9081310757741583.
[I 2025-10-26 16:38:06,001] Trial 10 finished with value: 0.9072297688791605 and parameters: {'n_estimators': 744, 'learning_rate': 0.013135953991326246, 'max_depth': 9, 'min_child_weight': 4, 'subsample': 0.7986786918572174, 'colsample_bytree': 0.5702593039142229, 'gamma': 0.15752652851131055, 'reg_alpha': 0.07108673090219104, 'reg_lambda': 1.5074371916627796}. Best is trial 8 with value: 0.9081310757741583.
[I 2025-10-26 16:38:22,927] Trial 11 finished with value: 0.9080023176463013 and parameters: {'n_estimators': 764, 'learning_rate': 0.014212444922629104, 'max_depth': 10, 'min_child_weight': 4, 'subsample': 0.700938661408883, 'colsample_bytree': 0.5514307141832163, 'gamma': 0.19585221218596743,

```
'reg_alpha': 0.1228234540829291, 'reg_lambda': 1.6950996401026215}. Best is
trial 8 with value: 0.9081310757741583.
[I 2025-10-26 16:38:39,494] Trial 12 finished with value: 0.9078091804545163 and
parameters: {'n_estimators': 782, 'learning_rate': 0.01457485850283542,
'max_depth': 9, 'min_child_weight': 5, 'subsample': 0.7381763609130155,
'colsample_bytree': 0.5636674925978187, 'gamma': 0.18437340228400206,
'reg_alpha': 0.07155290334453497, 'reg_lambda': 1.6503358528268959}. Best is
trial 8 with value: 0.9081310757741583.
[I 2025-10-26 16:38:56,813] Trial 13 finished with value: 0.907487285134874 and
parameters: {'n_estimators': 742, 'learning_rate': 0.01599308018824757,
'max_depth': 10, 'min_child_weight': 3, 'subsample': 0.7525930973110482,
'colsample_bytree': 0.5617780611155483, 'gamma': 0.1622316389821933,
'reg_alpha': 0.09346052038323627, 'reg_lambda': 1.576258916341387}. Best is
trial 8 with value: 0.9081310757741583.
[I 2025-10-26 16:39:13,446] Trial 14 finished with value: 0.9085817292216571 and
parameters: {'n_estimators': 772, 'learning_rate': 0.013777621813274443,
'max_depth': 9, 'min_child_weight': 4, 'subsample': 0.700758741960762,
'colsample_bytree': 0.5506331704907971, 'gamma': 0.18128173192348174,
'reg_alpha': 0.1180197235430194, 'reg_lambda': 1.6322892137130458}. Best is
trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:39:30,809] Trial 15 finished with value: 0.9083242129659436 and
parameters: {'n_estimators': 820, 'learning_rate': 0.01224945331017878,
'max_depth': 9, 'min_child_weight': 5, 'subsample': 0.7352843509286481,
'colsample_bytree': 0.551183179328206, 'gamma': 0.14258824683050322,
'reg_alpha': 0.11719713773561674, 'reg_lambda': 1.635609819709131}. Best is
trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:39:48,355] Trial 16 finished with value: 0.9083242129659436 and
parameters: {'n_estimators': 819, 'learning_rate': 0.012135547341759746,
'max_depth': 9, 'min_child_weight': 5, 'subsample': 0.7389653529966709,
'colsample_bytree': 0.5508418890459085, 'gamma': 0.1312807636876619,
'reg_alpha': 0.11812487858987344, 'reg_lambda': 1.6190130260525823}. Best is
trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:40:02,435] Trial 17 finished with value: 0.9070366316873751 and
parameters: {'n_estimators': 685, 'learning_rate': 0.01258423537284347,
'max_depth': 8, 'min_child_weight': 5, 'subsample': 0.7699884942216029,
'colsample_bytree': 0.5502452960168863, 'gamma': 0.14568280391598226,
'reg_alpha': 0.11471749868208526, 'reg_lambda': 1.5603528431291487}. Best is
trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:40:19,908] Trial 18 finished with value: 0.9074229060709458 and
parameters: {'n_estimators': 810, 'learning_rate': 0.011897412455849415,
'max_depth': 9, 'min_child_weight': 5, 'subsample': 0.7011594936820111,
'colsample_bytree': 0.5966727104948413, 'gamma': 0.13941297159912241,
'reg_alpha': 0.10353501508216349, 'reg_lambda': 1.6360667620620788}. Best is
trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:40:36,832] Trial 19 finished with value: 0.907165389815232 and
parameters: {'n_estimators': 782, 'learning_rate': 0.013402893844101394,
'max_depth': 9, 'min_child_weight': 4, 'subsample': 0.7300901972211253,
'colsample_bytree': 0.566869204875136, 'gamma': 0.19997278575643865,
```

'reg_alpha': 0.12241341992441451, 'reg_lambda': 1.5236728604516985}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:40:51,526] Trial 20 finished with value: 0.9072297688791605 and parameters: {'n_estimators': 731, 'learning_rate': 0.012566859813634252, 'max_depth': 8, 'min_child_weight': 5, 'subsample': 0.7607855872452357, 'colsample_bytree': 0.5745251498746516, 'gamma': 0.14364714428876812, 'reg_alpha': 0.11016289717226099, 'reg_lambda': 1.5888407245401743}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:41:08,822] Trial 21 finished with value: 0.9080023176463013 and parameters: {'n_estimators': 808, 'learning_rate': 0.011848588195725377, 'max_depth': 9, 'min_child_weight': 5, 'subsample': 0.7395463090325409, 'colsample_bytree': 0.5510414434663317, 'gamma': 0.1309236566974316, 'reg_alpha': 0.12062875579473815, 'reg_lambda': 1.6188302729421482}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:41:26,223] Trial 22 finished with value: 0.9081954548380866 and parameters: {'n_estimators': 819, 'learning_rate': 0.012236941894999135, 'max_depth': 9, 'min_child_weight': 5, 'subsample': 0.7330594642727395, 'colsample_bytree': 0.5500509710253221, 'gamma': 0.1305308608681446, 'reg_alpha': 0.11772531382010322, 'reg_lambda': 1.6311418844682866}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:41:44,023] Trial 23 finished with value: 0.9077448013905878 and parameters: {'n_estimators': 806, 'learning_rate': 0.011474265199106054, 'max_depth': 9, 'min_child_weight': 4, 'subsample': 0.7433642117184942, 'colsample_bytree': 0.5542781472155464, 'gamma': 0.13829436522333158, 'reg_alpha': 0.12907725537310033, 'reg_lambda': 1.6063230138362083}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:42:00,733] Trial 24 finished with value: 0.9072297688791605 and parameters: {'n_estimators': 777, 'learning_rate': 0.010010667137984385, 'max_depth': 9, 'min_child_weight': 6, 'subsample': 0.7821106523029575, 'colsample_bytree': 0.5626032741809803, 'gamma': 0.15718044991030514, 'reg_alpha': 0.09718558815639165, 'reg_lambda': 1.652371047651585}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:42:16,839] Trial 25 finished with value: 0.9080666967102299 and parameters: {'n_estimators': 757, 'learning_rate': 0.012832721397970771, 'max_depth': 9, 'min_child_weight': 5, 'subsample': 0.7467943790470368, 'colsample_bytree': 0.554185301992586, 'gamma': 0.14391224539165628, 'reg_alpha': 0.1078306553969138, 'reg_lambda': 1.624392419537255}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:42:32,447] Trial 26 finished with value: 0.9080666967102298 and parameters: {'n_estimators': 793, 'learning_rate': 0.013891756897909354, 'max_depth': 8, 'min_child_weight': 4, 'subsample': 0.7226489862631477, 'colsample_bytree': 0.5604620676968505, 'gamma': 0.18870148381370716, 'reg_alpha': 0.11571888256750756, 'reg_lambda': 1.5800144179102864}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:42:49,939] Trial 27 finished with value: 0.9078735595184446 and parameters: {'n_estimators': 820, 'learning_rate': 0.012270999491364594, 'max_depth': 9, 'min_child_weight': 5, 'subsample': 0.7576847739367353, 'colsample_bytree': 0.5541598965752882, 'gamma': 0.13520338355589495,

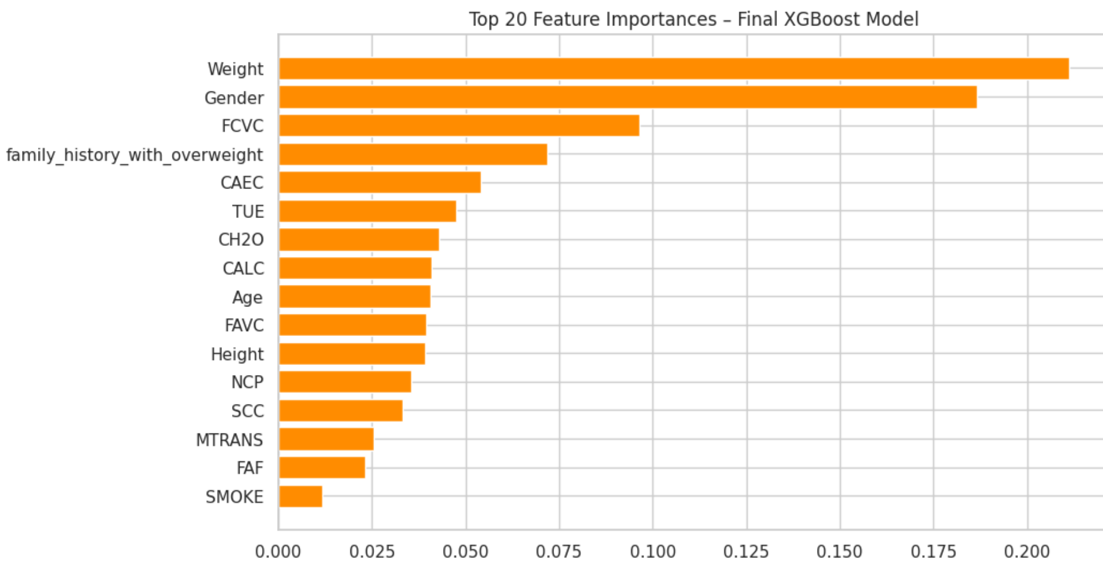
'reg_alpha': 0.12869047250929894, 'reg_lambda': 1.600734779944352}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:43:05,596] Trial 28 finished with value: 0.9074229060709458 and parameters: {'n_estimators': 804, 'learning_rate': 0.013671415414302918, 'max_depth': 8, 'min_child_weight': 6, 'subsample': 0.7147310005139375, 'colsample_bytree': 0.566042062466504, 'gamma': 0.15113645428418648, 'reg_alpha': 0.12387874394435817, 'reg_lambda': 1.6461965796039095}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:43:22,952] Trial 29 finished with value: 0.9080023176463013 and parameters: {'n_estimators': 773, 'learning_rate': 0.011378105767047331, 'max_depth': 9, 'min_child_weight': 4, 'subsample': 0.7768652870853008, 'colsample_bytree': 0.5728729809645302, 'gamma': 0.1771865238204776, 'reg_alpha': 0.11741722360965041, 'reg_lambda': 1.6240613084633746}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:43:39,087] Trial 30 finished with value: 0.907165389815232 and parameters: {'n_estimators': 757, 'learning_rate': 0.012945864896470614, 'max_depth': 9, 'min_child_weight': 5, 'subsample': 0.7315911216586517, 'colsample_bytree': 0.5812238318411875, 'gamma': 0.15757743529760676, 'reg_alpha': 0.13420269179463812, 'reg_lambda': 1.6767001618630644}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:43:58,301] Trial 31 finished with value: 0.9080666967102299 and parameters: {'n_estimators': 814, 'learning_rate': 0.012135953386674564, 'max_depth': 9, 'min_child_weight': 5, 'subsample': 0.7333373509774926, 'colsample_bytree': 0.5500386094082713, 'gamma': 0.1308616651361332, 'reg_alpha': 0.11889895810119998, 'reg_lambda': 1.6295886357735738}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:44:14,735] Trial 32 finished with value: 0.9074229060709458 and parameters: {'n_estimators': 796, 'learning_rate': 0.012198204785471824, 'max_depth': 9, 'min_child_weight': 6, 'subsample': 0.7067338798281085, 'colsample_bytree': 0.552803526572798, 'gamma': 0.1350427892740649, 'reg_alpha': 0.11319287820208287, 'reg_lambda': 1.6098297843565674}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:44:32,126] Trial 33 finished with value: 0.907487285134874 and parameters: {'n_estimators': 819, 'learning_rate': 0.012496981199118434, 'max_depth': 9, 'min_child_weight': 5, 'subsample': 0.7461059190897739, 'colsample_bytree': 0.5583480445623331, 'gamma': 0.13886129769808922, 'reg_alpha': 0.10400196159252442, 'reg_lambda': 1.6133068096704366}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:44:48,597] Trial 34 finished with value: 0.9076160432627309 and parameters: {'n_estimators': 791, 'learning_rate': 0.01184222754107582, 'max_depth': 9, 'min_child_weight': 6, 'subsample': 0.7112113716461834, 'colsample_bytree': 0.5564858641726516, 'gamma': 0.1478116947628444, 'reg_alpha': 0.13470431676178277, 'reg_lambda': 1.6335185791107714}. Best is trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:45:06,062] Trial 35 finished with value: 0.9078735595184446 and parameters: {'n_estimators': 811, 'learning_rate': 0.013397555420207253, 'max_depth': 9, 'min_child_weight': 4, 'subsample': 0.7238283523559882, 'colsample_bytree': 0.5543287365672405, 'gamma': 0.13016835867751153,

```
'reg_alpha': 0.1260816570763907, 'reg_lambda': 1.6644693237433377}. Best is
trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:45:24,002] Trial 36 finished with value: 0.907487285134874 and
parameters: {'n_estimators': 802, 'learning_rate': 0.011293129463408895,
'max_depth': 9, 'min_child_weight': 5, 'subsample': 0.7341811056964159,
'colsample_bytree': 0.557728865582524, 'gamma': 0.14115094039847892,
'reg_alpha': 0.11851857656098498, 'reg_lambda': 1.6576176030955585}. Best is
trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:45:43,271] Trial 37 finished with value: 0.9078735595184445 and
parameters: {'n_estimators': 788, 'learning_rate': 0.013136455384422518,
'max_depth': 10, 'min_child_weight': 4, 'subsample': 0.757138860562867,
'colsample_bytree': 0.552812866397269, 'gamma': 0.13401250777986673,
'reg_alpha': 0.11062596333281233, 'reg_lambda': 1.641810553254157}. Best is
trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:46:00,226] Trial 38 finished with value: 0.9077448013905878 and
parameters: {'n_estimators': 798, 'learning_rate': 0.01428277711522705,
'max_depth': 9, 'min_child_weight': 5, 'subsample': 0.7067971961600117,
'colsample_bytree': 0.5638854814147001, 'gamma': 0.15374634220455852,
'reg_alpha': 0.10007696069846139, 'reg_lambda': 1.5933736014560633}. Best is
trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:46:17,375] Trial 39 finished with value: 0.9074229060709457 and
parameters: {'n_estimators': 814, 'learning_rate': 0.011688432243476226,
'max_depth': 9, 'min_child_weight': 5, 'subsample': 0.7203961456554931,
'colsample_bytree': 0.558970195935141, 'gamma': 0.16702648021126107,
'reg_alpha': 0.12495688424039242, 'reg_lambda': 1.678594748778006}. Best is
trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:46:34,586] Trial 40 finished with value: 0.907294147943089 and
parameters: {'n_estimators': 718, 'learning_rate': 0.0123075920080818,
'max_depth': 10, 'min_child_weight': 4, 'subsample': 0.7418550926769639,
'colsample_bytree': 0.5896565948137471, 'gamma': 0.17925358679421355,
'reg_alpha': 0.13208596081044585, 'reg_lambda': 1.6195343693589481}. Best is
trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:46:52,958] Trial 41 finished with value: 0.9078735595184446 and
parameters: {'n_estimators': 786, 'learning_rate': 0.013896926101093552,
'max_depth': 10, 'min_child_weight': 3, 'subsample': 0.7386147848429295,
'colsample_bytree': 0.5500322579832365, 'gamma': 0.16908099345007013,
'reg_alpha': 0.08078347024991563, 'reg_lambda': 1.6648680173658428}. Best is
trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:47:11,179] Trial 42 finished with value: 0.9080666967102299 and
parameters: {'n_estimators': 766, 'learning_rate': 0.012895120707151323,
'max_depth': 10, 'min_child_weight': 3, 'subsample': 0.7283471737642192,
'colsample_bytree': 0.556075217289419, 'gamma': 0.16371931843320234,
'reg_alpha': 0.08066323274860213, 'reg_lambda': 1.6337599520286907}. Best is
trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:47:28,769] Trial 43 finished with value: 0.9085817292216571 and
parameters: {'n_estimators': 820, 'learning_rate': 0.014859722339390204,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.7359775951758664,
'colsample_bytree': 0.5599166707299679, 'gamma': 0.18747857936605333,
```

```
'reg_alpha': 0.10659966418097244, 'reg_lambda': 1.6865147766424577}. Best is
trial 14 with value: 0.9085817292216571.
[I 2025-10-26 16:47:46,363] Trial 44 finished with value: 0.9087748664134423 and
parameters: {'n_estimators': 816, 'learning_rate': 0.014710138612677975,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.7180557357572586,
'colsample_bytree': 0.5522858830234695, 'gamma': 0.18769981291348767,
'reg_alpha': 0.10935646298325649, 'reg_lambda': 1.6041904508339415}. Best is
trial 44 with value: 0.9087748664134423.
[I 2025-10-26 16:48:03,679] Trial 45 finished with value: 0.9081954548380866 and
parameters: {'n_estimators': 803, 'learning_rate': 0.015282154577699185,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.7113709169962045,
'colsample_bytree': 0.5604413393759144, 'gamma': 0.18814302771919605,
'reg_alpha': 0.10685879241752422, 'reg_lambda': 1.683609662339871}. Best is
trial 44 with value: 0.9087748664134423.
[I 2025-10-26 16:48:21,304] Trial 46 finished with value: 0.9081954548380865 and
parameters: {'n_estimators': 814, 'learning_rate': 0.01473459386467209,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.721363503127408,
'colsample_bytree': 0.5525166550189564, 'gamma': 0.19357227665154295,
'reg_alpha': 0.11039745149211673, 'reg_lambda': 1.5710620083574405}. Best is
trial 44 with value: 0.9087748664134423.
[I 2025-10-26 16:48:38,558] Trial 47 finished with value: 0.9087104873495139 and
parameters: {'n_estimators': 798, 'learning_rate': 0.01575735720160836,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.7164719182481573,
'colsample_bytree': 0.5555593098808913, 'gamma': 0.18440382043842254,
'reg_alpha': 0.10044706818651947, 'reg_lambda': 1.6035694057678818}. Best is
trial 44 with value: 0.9087748664134423.
[I 2025-10-26 16:48:54,732] Trial 48 finished with value: 0.9083242129659433 and
parameters: {'n_estimators': 797, 'learning_rate': 0.01583107735092103,
'max_depth': 8, 'min_child_weight': 3, 'subsample': 0.7161809692127379,
'colsample_bytree': 0.5554530091778943, 'gamma': 0.18411695642716797,
'reg_alpha': 0.09259031511983017, 'reg_lambda': 1.5865439816780615}. Best is
trial 44 with value: 0.9087748664134423.
[I 2025-10-26 16:49:12,907] Trial 49 finished with value: 0.9068434944955899 and
parameters: {'n_estimators': 777, 'learning_rate': 0.015063921369040733,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.7027114039327237,
'colsample_bytree': 0.5704011368447073, 'gamma': 0.1748046828797527,
'reg_alpha': 0.09789371712692718, 'reg_lambda': 1.541666464446408}. Best is
trial 44 with value: 0.9087748664134423.
[I 2025-10-26 16:49:30,802] Trial 50 finished with value: 0.9079379385823729 and
parameters: {'n_estimators': 809, 'learning_rate': 0.015611341792978498,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.7106780203676939,
'colsample_bytree': 0.5603426033818818, 'gamma': 0.18850596888092314,
'reg_alpha': 0.10196046626933455, 'reg_lambda': 1.6026590015475703}. Best is
trial 44 with value: 0.9087748664134423.
[I 2025-10-26 16:49:48,431] Trial 51 finished with value: 0.9086461082855856 and
parameters: {'n_estimators': 815, 'learning_rate': 0.014250328221111297,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.726708667290631,
'colsample_bytree': 0.553164192782055, 'gamma': 0.18118846314787646,
```

```
'reg_alpha': 0.11302216140032322, 'reg_lambda': 1.6998626703145898}. Best is
trial 44 with value: 0.9087748664134423.
[I 2025-10-26 16:50:06,413] Trial 52 finished with value: 0.9085817292216571 and
parameters: {'n_estimators': 814, 'learning_rate': 0.014434516061140375,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.7275372722743714,
'colsample_bytree': 0.5567816073768916, 'gamma': 0.18215767999854976,
'reg_alpha': 0.10597588171766244, 'reg_lambda': 1.6966008919659878}. Best is
trial 44 with value: 0.9087748664134423.
[I 2025-10-26 16:50:24,112] Trial 53 finished with value: 0.9088392454773707 and
parameters: {'n_estimators': 814, 'learning_rate': 0.014308476269248808,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.7263200600339932,
'colsample_bytree': 0.5576317868512216, 'gamma': 0.18072420150582985,
'reg_alpha': 0.10641751805672392, 'reg_lambda': 1.6915597753551928}. Best is
trial 53 with value: 0.9088392454773707.
[I 2025-10-26 16:50:42,055] Trial 54 finished with value: 0.9068434944955902 and
parameters: {'n_estimators': 800, 'learning_rate': 0.014936519470056867,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.7171132516985788,
'colsample_bytree': 0.5645389710961846, 'gamma': 0.17335715408392338,
'reg_alpha': 0.09407036449849945, 'reg_lambda': 1.6880672211176833}. Best is
trial 53 with value: 0.9088392454773707.
[I 2025-10-26 16:50:59,461] Trial 55 finished with value: 0.9087748664134423 and
parameters: {'n_estimators': 806, 'learning_rate': 0.015223596621572908,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.719221404564782,
'colsample_bytree': 0.5588956631557571, 'gamma': 0.1866579386744173,
'reg_alpha': 0.10898606895529972, 'reg_lambda': 1.6963435237892088}. Best is
trial 53 with value: 0.9088392454773707.
[I 2025-10-26 16:51:17,529] Trial 56 finished with value: 0.9081310757741583 and
parameters: {'n_estimators': 807, 'learning_rate': 0.015345927457087438,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.705788849313502,
'colsample_bytree': 0.5526704914280967, 'gamma': 0.17900235246355486,
'reg_alpha': 0.11291487375106399, 'reg_lambda': 1.6996907204311178}. Best is
trial 53 with value: 0.9088392454773707.
[I 2025-10-26 16:51:34,597] Trial 57 finished with value: 0.9078091804545163 and
parameters: {'n_estimators': 785, 'learning_rate': 0.014112625434305994,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.7196943840617852,
'colsample_bytree': 0.5621392683229055, 'gamma': 0.18575490085485052,
'reg_alpha': 0.10951861109901326, 'reg_lambda': 1.6931570329587995}. Best is
trial 53 with value: 0.9088392454773707.
[I 2025-10-26 16:51:52,138] Trial 58 finished with value: 0.9063928410480911 and
parameters: {'n_estimators': 791, 'learning_rate': 0.014574943772794833,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.7137923218359338,
'colsample_bytree': 0.5999426989965128, 'gamma': 0.1911333143743522,
'reg_alpha': 0.10141616314317219, 'reg_lambda': 1.668265322836546}. Best is
trial 53 with value: 0.9088392454773707.
[I 2025-10-26 16:52:10,413] Trial 59 finished with value: 0.9087748664134424 and
parameters: {'n_estimators': 804, 'learning_rate': 0.015553131209879384,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.7241905987127736,
'colsample_bytree': 0.5550458839234195, 'gamma': 0.1975868606950675,
```

```
'reg_alpha': 0.1148692930663168, 'reg_lambda': 1.6714265650161926}. Best is
trial 53 with value: 0.9088392454773707.
Best params: {'n_estimators': 814, 'learning_rate': 0.014308476269248808,
'max_depth': 9, 'min_child_weight': 3, 'subsample': 0.7263200600339932,
'colsample_bytree': 0.5576317868512216, 'gamma': 0.18072420150582985,
'reg_alpha': 0.10641751805672392, 'reg_lambda': 1.6915597753551928}
Best accuracy: 0.9088392454773707

  Model Evaluation on Validation Set
=====================================
Accuracy: 0.964274219504345
Precision: 0.9642615720784604
Recall: 0.964274219504345
F1 Score: 0.9642404609827702

Confusion Matrix:
 [[365   8   0   0   0   1   0]
 [  9 449   0   0   0  10   1]
 [  0   0 417   4   3   7  10]
 [  0   0   3 477   0   0   1]
 [  0   0   0   0 597   0   0]
 [  2  12   4   0   0 336  15]
 [  0   2   8   0   0  11 355]]

Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.98      0.97       374
           1       0.95      0.96      0.96       469
           2       0.97      0.95      0.96       441
           3       0.99      0.99      0.99       481
           4       0.99      1.00      1.00       597
           5       0.92      0.91      0.92       369
           6       0.93      0.94      0.94       376

    accuracy                           0.96      3107
   macro avg       0.96      0.96      0.96      3107
weighted avg       0.96      0.96      0.96      3107


  Submission file saved as 'submission.csv'
```



XGBOOST ACCURACY USING OPTUNA IS **0.964274219504345**

---

5. ADABOOST

[71]:
```
base_estimator = DecisionTreeClassifier(
    criterion='gini',
    max_depth=7,
    min_samples_split=5,
    min_samples_leaf=2,
    random_state=42
)

ada_clf = AdaBoostClassifier(
    estimator=base_estimator,
    n_estimators=100,
    learning_rate=0.5,
    random_state=42
)

ada_clf.fit(X_train, y_train)
y_pred_enc = ada_clf.predict(X_test)

y_pred = le.inverse_transform(y_pred_enc)
y_test_orig = le.inverse_transform(y_test)

test_accuracy_ab = accuracy_score(y_test_orig, y_pred)
test_precision_ab = precision_score(y_test_orig, y_pred, average='weighted',␣
↪zero_division=0)
```

```python
test_recall_ab = recall_score(y_test_orig, y_pred, average='weighted',␣
↪zero_division=0)
test_f1_ab = f1_score(y_test_orig, y_pred, average='weighted', zero_division=0)

print("--- AdaBoost Model Evaluation ---")
print("Test Accuracy (AdaBoost):", test_accuracy_ab)
print("Test Precision (Weighted):", test_precision_ab)
print("Test Recall (Weighted):", test_recall_ab)
print("Test F1-Score (Weighted):", test_f1_ab)
print("\nConfusion Matrix:\n", confusion_matrix(y_test_orig, y_pred))
print("\nClassification Report:\n", classification_report(y_test_orig, y_pred))
```

```
--- AdaBoost Model Evaluation ---
Test Accuracy (AdaBoost): 0.8847763115545543
Test Precision (Weighted): 0.8860478578886273
Test Recall (Weighted): 0.8847763115545543
Test F1-Score (Weighted): 0.8852893045389144

Confusion Matrix:
 [[306  37   1   1   0   0   1]
 [ 26 417   1   0   0  43   9]
 [  0   0 389  13   1  14  31]
 [  0   0  16 447   0   0   1]
 [  0   0   2   1 603   0   0]
 [  2  34   4   0   0 296  48]
 [  0   4  24   3   0  41 291]]

Classification Report:
                      precision    recall  f1-score   support

Insufficient_Weight       0.92      0.88      0.90       346
      Normal_Weight       0.85      0.84      0.84       496
     Obesity_Type_I       0.89      0.87      0.88       448
    Obesity_Type_II       0.96      0.96      0.96       464
   Obesity_Type_III       1.00      1.00      1.00       606
 Overweight_Level_I       0.75      0.77      0.76       384
Overweight_Level_II       0.76      0.80      0.78       363

           accuracy                           0.88      3107
          macro avg       0.88      0.87      0.88      3107
       weighted avg       0.89      0.88      0.89      3107
```

THE ACCURACY USING ADABOOST ON DECISION TREE IS 0.8847763115545543

Hyperparameter Tuning Using Optuna On AdaBOOST

```
[ ]: def objective(trial):
    max_depth = trial.suggest_int('max_depth', 1, 10)
    min_samples_split = trial.suggest_int('min_samples_split', 2, 20)
    min_samples_leaf = trial.suggest_int('min_samples_leaf', 1, 10)
    n_estimators = trial.suggest_int('n_estimators', 50, 500)
    learning_rate = trial.suggest_float('learning_rate', 0.01, 1.0)

    base_estimator = DecisionTreeClassifier(
        criterion='gini',
        max_depth=max_depth,
        min_samples_split=min_samples_split,
        min_samples_leaf=min_samples_leaf,
        random_state=42
    )

    ada_clf = AdaBoostClassifier(
        estimator=base_estimator,
        n_estimators=n_estimators,
        learning_rate=learning_rate,
        random_state=42
    )

    score = cross_val_score(ada_clf, X_train, y_train, cv=5,␣
 ↪scoring='accuracy').mean()
    return score

study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=50, show_progress_bar=True)

best_params = study.best_params
base_estimator = DecisionTreeClassifier(
    criterion='gini',
    max_depth=best_params['max_depth'],
    min_samples_split=best_params['min_samples_split'],
    min_samples_leaf=best_params['min_samples_leaf'],
    random_state=42
)

ada_clf_best = AdaBoostClassifier(
    estimator=base_estimator,
    n_estimators=best_params['n_estimators'],
    learning_rate=best_params['learning_rate'],
    random_state=42
)
ada_clf_best.fit(X_train, y_train)
y_pred_enc = ada_clf_best.predict(X_test)
```

```
y_pred = le.inverse_transform(y_pred_enc)
y_test_orig = le.inverse_transform(y_test)

test_accuracy_ab_hp = accuracy_score(y_test_orig, y_pred)
test_precision_ab_hp = precision_score(y_test_orig, y_pred, average='weighted',␣
↪zero_division=0)
test_recall_ab_hp = recall_score(y_test_orig, y_pred, average='weighted',␣
↪zero_division=0)
test_f1_ab_hp = f1_score(y_test_orig, y_pred, average='weighted',␣
↪zero_division=0)

print("Best parameters:", study.best_params)
print("Best cross-validation accuracy:", study.best_value)
print("--- AdaBoost Model Evaluation (Optimized) ---")
print("Test Accuracy (AdaBoost):", test_accuracy_ab_hp)
print("Test Precision (Weighted):", test_precision_ab_hp)
print("Test Recall (Weighted):", test_recall_ab_hp)
print("Test F1-Score (Weighted):", test_f1_ab_hp)
print("\nConfusion Matrix:\n", confusion_matrix(y_test_orig, y_pred))
print("\nClassification Report:\n", classification_report(y_test_orig, y_pred))
```

THE ACCURACY USING ADABOOST ON DECISION TREE USING OPTUNA IS 0.8937882201480528

6. KNN

[68]:
```
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Now split encoded labels
X_train, X_test, y_train, y_test = train_test_split(x, y_encoded, test_size=0.
↪2, random_state=42)
knn_clf = KNeighborsClassifier(n_neighbors=5)

# Fit model
knn_clf.fit(X_train, y_train)

# Predict
y_pred_enc = knn_clf.predict(X_test)

# Decode predictions back to original labels
y_pred = le.inverse_transform(y_pred_enc)
y_test_orig = le.inverse_transform(y_test)

# --- Individual Classification Metrics Calculation ---
test_accuracy_knn = accuracy_score(y_test_orig, y_pred)
test_precision_knn = precision_score(y_test_orig, y_pred, average='weighted',␣
↪zero_division=0)
```

```
test_recall_knn = recall_score(y_test_orig, y_pred, average='weighted',␣
    ↪zero_division=0)
test_f1_knn = f1_score(y_test_orig, y_pred, average='weighted', zero_division=0)

# Evaluate
print("--- KNN Model Evaluation ---")
print("Test Accuracy (KNN):", test_accuracy_knn)
print("Test Precision (Weighted):", test_precision_knn)
print("Test Recall (Weighted):", test_recall_knn)
print("Test F1-Score (Weighted):", test_f1_knn)
print("\nConfusion Matrix:\n", confusion_matrix(y_test_orig, y_pred))
print("\nClassification Report:\n", classification_report(y_test_orig, y_pred))
```

```
--- KNN Model Evaluation ---
Test Accuracy (KNN): 0.4354682973929836
Test Precision (Weighted): 0.4292489701413928
Test Recall (Weighted): 0.4354682973929836
Test F1-Score (Weighted): 0.43105678373536

Confusion Matrix:
 [[215 122   1   0   1   6   1]
 [148 235  11   0   0  75  27]
 [  0  12 156  38  95  68  79]
 [  0   0  34 246 181   2   1]
 [  0   0  59 240 306   0   1]
 [ 12 133  56   0   1 106  76]
 [  3  33 125   8   7  98  89]]

Classification Report:
                      precision    recall  f1-score   support

Insufficient_Weight       0.57      0.62      0.59       346
      Normal_Weight       0.44      0.47      0.46       496
     Obesity_Type_I       0.35      0.35      0.35       448
    Obesity_Type_II       0.46      0.53      0.49       464
   Obesity_Type_III       0.52      0.50      0.51       606
 Overweight_Level_I       0.30      0.28      0.29       384
Overweight_Level_II       0.32      0.25      0.28       363

           accuracy                           0.44      3107
          macro avg       0.42      0.43      0.42      3107
       weighted avg       0.43      0.44      0.43      3107
```

THE ACCURACY USING KNN IS 0.4354682973929836

Hyperparameter Tuning Using Optuna On KNN

[69]:
```
import optuna
from sklearn.model_selection import cross_val_score

def objective(trial):
    n_neighbors = trial.suggest_int('n_neighbors', 1, 30)
    weights = trial.suggest_categorical('weights', ['uniform', 'distance'])
    metric = trial.suggest_categorical('metric', ['euclidean', 'manhattan',␣
 ↪'minkowski'])
    model = KNeighborsClassifier(
        n_neighbors=n_neighbors,
        weights=weights,
        metric=metric
    )
    score = cross_val_score(model, X_train, y_train, cv=5, scoring='accuracy').
 ↪mean()
    return score

study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=50, n_jobs=-1)

best_params = study.best_params
print("Best Parameters:", best_params)

knn_clf = KNeighborsClassifier(**best_params)
knn_clf.fit(X_train, y_train)

y_pred_enc = knn_clf.predict(X_test)
y_pred = le.inverse_transform(y_pred_enc)
y_test_orig = le.inverse_transform(y_test)

test_accuracy_knn_hp = accuracy_score(y_test_orig, y_pred)
test_precision_knn_hp = precision_score(y_test_orig, y_pred,␣
 ↪average='weighted', zero_division=0)
test_recall_knn_hp = recall_score(y_test_orig, y_pred, average='weighted',␣
 ↪zero_division=0)
test_f1_knn_hp = f1_score(y_test_orig, y_pred, average='weighted',␣
 ↪zero_division=0)

print("--- Tuned KNN Model Evaluation ---")
print("Test Accuracy (KNN):", test_accuracy_knn_hp)
print("Test Precision (Weighted):", test_precision_knn_hp)
print("Test Recall (Weighted):", test_recall_knn_hp)
print("Test F1-Score (Weighted):", test_f1_knn_hp)
print("\nConfusion Matrix:\n", confusion_matrix(y_test_orig, y_pred))
print("\nClassification Report:\n", classification_report(y_test_orig, y_pred))
```

```
[I 2025-10-26 16:57:35,221] A new study created in memory with name: no-
```

```
name-668063b0-867a-43d7-855a-4ea887a4455c
[I 2025-10-26 16:57:43,854] Trial 4 finished with value: 0.37035040492350724 and
parameters: {'n_neighbors': 18, 'weights': 'uniform', 'metric': 'minkowski'}.
Best is trial 4 with value: 0.37035040492350724.
[I 2025-10-26 16:57:44,349] Trial 21 finished with value: 0.42282166045346903
and parameters: {'n_neighbors': 5, 'weights': 'uniform', 'metric': 'euclidean'}.
Best is trial 21 with value: 0.42282166045346903.
[I 2025-10-26 16:57:44,755] Trial 8 finished with value: 0.3668904496973798 and
parameters: {'n_neighbors': 19, 'weights': 'uniform', 'metric': 'euclidean'}.
Best is trial 21 with value: 0.42282166045346903.
[I 2025-10-26 16:57:45,648] Trial 13 finished with value: 0.3668904496973798 and
parameters: {'n_neighbors': 19, 'weights': 'uniform', 'metric': 'euclidean'}.
Best is trial 21 with value: 0.42282166045346903.
[I 2025-10-26 16:57:45,855] Trial 15 finished with value: 0.3520027971529903 and
parameters: {'n_neighbors': 27, 'weights': 'uniform', 'metric': 'minkowski'}.
Best is trial 21 with value: 0.42282166045346903.
[I 2025-10-26 16:57:46,065] Trial 22 finished with value: 0.35948683249942126
and parameters: {'n_neighbors': 21, 'weights': 'uniform', 'metric':
'minkowski'}. Best is trial 21 with value: 0.42282166045346903.
[I 2025-10-26 16:57:47,348] Trial 16 finished with value: 0.3483812286429761 and
parameters: {'n_neighbors': 28, 'weights': 'uniform', 'metric': 'euclidean'}.
Best is trial 21 with value: 0.42282166045346903.
[I 2025-10-26 16:57:48,256] Trial 20 finished with value: 0.48084448768232885
and parameters: {'n_neighbors': 8, 'weights': 'uniform', 'metric': 'manhattan'}.
Best is trial 20 with value: 0.48084448768232885.
[I 2025-10-26 16:57:48,352] Trial 11 finished with value: 0.44881562909233363
and parameters: {'n_neighbors': 19, 'weights': 'distance', 'metric':
'minkowski'}. Best is trial 20 with value: 0.48084448768232885.
[I 2025-10-26 16:57:48,847] Trial 19 finished with value: 0.4996769029300502 and
parameters: {'n_neighbors': 5, 'weights': 'uniform', 'metric': 'manhattan'}.
Best is trial 19 with value: 0.4996769029300502.
[I 2025-10-26 16:57:49,261] Trial 12 finished with value: 0.45525299827929766
and parameters: {'n_neighbors': 14, 'weights': 'uniform', 'metric':
'manhattan'}. Best is trial 19 with value: 0.4996769029300502.
[I 2025-10-26 16:57:49,460] Trial 18 finished with value: 0.4629788384368966 and
parameters: {'n_neighbors': 11, 'weights': 'uniform', 'metric': 'manhattan'}.
Best is trial 19 with value: 0.4996769029300502.
[I 2025-10-26 16:57:49,574] Trial 10 finished with value: 0.4637039938747529 and
parameters: {'n_neighbors': 9, 'weights': 'distance', 'metric': 'minkowski'}.
Best is trial 19 with value: 0.4996769029300502.
[I 2025-10-26 16:57:49,575] Trial 6 finished with value: 0.4617719834696028 and
parameters: {'n_neighbors': 13, 'weights': 'uniform', 'metric': 'manhattan'}.
Best is trial 19 with value: 0.4996769029300502.
[I 2025-10-26 16:57:49,655] Trial 23 finished with value: 0.4229024347209565 and
parameters: {'n_neighbors': 29, 'weights': 'uniform', 'metric': 'manhattan'}.
Best is trial 19 with value: 0.4996769029300502.
[I 2025-10-26 16:57:49,953] Trial 7 finished with value: 0.5087711789643735 and
parameters: {'n_neighbors': 22, 'weights': 'distance', 'metric': 'manhattan'}.
```

```
Best is trial 7 with value: 0.5087711789643735.
[I 2025-10-26 16:57:50,659] Trial 5 finished with value: 0.53935183101829 and
parameters: {'n_neighbors': 4, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:57:51,547] Trial 2 finished with value: 0.5087711789643735 and
parameters: {'n_neighbors': 22, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:57:51,758] Trial 9 finished with value: 0.5226934252336223 and
parameters: {'n_neighbors': 12, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:57:52,267] Trial 0 finished with value: 0.5054713477971611 and
parameters: {'n_neighbors': 23, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:57:52,553] Trial 1 finished with value: 0.5064371101913168 and
parameters: {'n_neighbors': 24, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:57:52,758] Trial 17 finished with value: 0.5107024771314936 and
parameters: {'n_neighbors': 21, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:57:52,958] Trial 14 finished with value: 0.5087711789643735 and
parameters: {'n_neighbors': 22, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:57:53,457] Trial 3 finished with value: 0.5087711789643735 and
parameters: {'n_neighbors': 22, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:57:54,848] Trial 27 finished with value: 0.35650977465759964
and parameters: {'n_neighbors': 23, 'weights': 'uniform', 'metric':
'euclidean'}. Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:57:55,262] Trial 24 finished with value: 0.48470957684967403
and parameters: {'n_neighbors': 1, 'weights': 'distance', 'metric':
'euclidean'}. Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:57:55,772] Trial 26 finished with value: 0.4791559979345097 and
parameters: {'n_neighbors': 3, 'weights': 'distance', 'metric': 'minkowski'}.
Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:57:56,354] Trial 28 finished with value: 0.4847900597470584 and
parameters: {'n_neighbors': 2, 'weights': 'distance', 'metric': 'euclidean'}.
Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:57:57,056] Trial 25 finished with value: 0.4229024347209565 and
parameters: {'n_neighbors': 29, 'weights': 'uniform', 'metric': 'manhattan'}.
Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:57:57,648] Trial 30 finished with value: 0.3830663789656685 and
parameters: {'n_neighbors': 13, 'weights': 'uniform', 'metric': 'minkowski'}.
Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:57:59,949] Trial 29 finished with value: 0.44776935142633756
and parameters: {'n_neighbors': 21, 'weights': 'distance', 'metric':
'minkowski'}. Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:58:01,562] Trial 31 finished with value: 0.4448718376226789 and
parameters: {'n_neighbors': 26, 'weights': 'distance', 'metric': 'minkowski'}.
```

```
Best is trial 5 with value: 0.53935183101829.
[I 2025-10-26 16:58:02,258] Trial 36 finished with value: 0.5420888970184745 and
parameters: {'n_neighbors': 1, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 36 with value: 0.5420888970184745.
[I 2025-10-26 16:58:02,547] Trial 32 finished with value: 0.5227734872630797 and
parameters: {'n_neighbors': 11, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 36 with value: 0.5420888970184745.
[I 2025-10-26 16:58:02,859] Trial 39 finished with value: 0.5422498951876991 and
parameters: {'n_neighbors': 2, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 39 with value: 0.5422498951876991.
[I 2025-10-26 16:58:02,960] Trial 38 finished with value: 0.5419270247389405 and
parameters: {'n_neighbors': 3, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 39 with value: 0.5422498951876991.
[I 2025-10-26 16:58:03,051] Trial 37 finished with value: 0.5419270247389405 and
parameters: {'n_neighbors': 3, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 39 with value: 0.5422498951876991.
[I 2025-10-26 16:58:04,047] Trial 33 finished with value: 0.5227734872630797 and
parameters: {'n_neighbors': 11, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 39 with value: 0.5422498951876991.
[I 2025-10-26 16:58:04,049] Trial 34 finished with value: 0.534281667478726 and
parameters: {'n_neighbors': 5, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 39 with value: 0.5422498951876991.
[I 2025-10-26 16:58:04,554] Trial 41 finished with value: 0.5422498951876991 and
parameters: {'n_neighbors': 2, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 39 with value: 0.5422498951876991.
[I 2025-10-26 16:58:04,560] Trial 35 finished with value: 0.5422498951876991 and
parameters: {'n_neighbors': 2, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 39 with value: 0.5422498951876991.
[I 2025-10-26 16:58:04,748] Trial 42 finished with value: 0.5420888970184745 and
parameters: {'n_neighbors': 1, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 39 with value: 0.5422498951876991.
[I 2025-10-26 16:58:05,254] Trial 40 finished with value: 0.5064371101913168 and
parameters: {'n_neighbors': 24, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 39 with value: 0.5422498951876991.
[I 2025-10-26 16:58:05,548] Trial 46 finished with value: 0.5420888970184745 and
parameters: {'n_neighbors': 1, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 39 with value: 0.5422498951876991.
[I 2025-10-26 16:58:05,559] Trial 43 finished with value: 0.5422498951876991 and
parameters: {'n_neighbors': 2, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 39 with value: 0.5422498951876991.
[I 2025-10-26 16:58:05,650] Trial 45 finished with value: 0.5420888970184745 and
parameters: {'n_neighbors': 1, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 39 with value: 0.5422498951876991.
[I 2025-10-26 16:58:05,655] Trial 47 finished with value: 0.5420888970184745 and
parameters: {'n_neighbors': 1, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 39 with value: 0.5422498951876991.
[I 2025-10-26 16:58:05,665] Trial 44 finished with value: 0.5422498951876991 and
parameters: {'n_neighbors': 2, 'weights': 'distance', 'metric': 'manhattan'}.
```

```
Best is trial 39 with value: 0.5422498951876991.
[I 2025-10-26 16:58:05,758] Trial 48 finished with value: 0.5422498951876991 and
parameters: {'n_neighbors': 2, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 39 with value: 0.5422498951876991.
[I 2025-10-26 16:58:05,798] Trial 49 finished with value: 0.5420888970184745 and
parameters: {'n_neighbors': 1, 'weights': 'distance', 'metric': 'manhattan'}.
Best is trial 39 with value: 0.5422498951876991.

Best Parameters: {'n_neighbors': 2, 'weights': 'distance', 'metric':
'manhattan'}
--- Tuned KNN Model Evaluation ---
Test Accuracy (KNN): 0.5468297392983585
Test Precision (Weighted): 0.5414058059931212
Test Recall (Weighted): 0.5468297392983585
Test F1-Score (Weighted): 0.5415818219119306

Confusion Matrix:
 [[226 109   2   0   1   7   1]
 [112 234  13   1   0 104  32]
 [  0   7 166  42  75  54 104]
 [  0   0  31 291 138   0   4]
 [  0   0  16  71 519   0   0]
 [  2 108  38   0   0 132 104]
 [  2  24  96   9   7  94 131]]

Classification Report:
                      precision    recall  f1-score   support

Insufficient_Weight       0.66      0.65      0.66       346
      Normal_Weight       0.49      0.47      0.48       496
     Obesity_Type_I       0.46      0.37      0.41       448
    Obesity_Type_II       0.70      0.63      0.66       464
   Obesity_Type_III       0.70      0.86      0.77       606
 Overweight_Level_I       0.34      0.34      0.34       384
Overweight_Level_II       0.35      0.36      0.35       363

           accuracy                           0.55      3107
          macro avg       0.53      0.53      0.52      3107
       weighted avg       0.54      0.55      0.54      3107
```

KNN ACCURACY USING OPTUNA IS

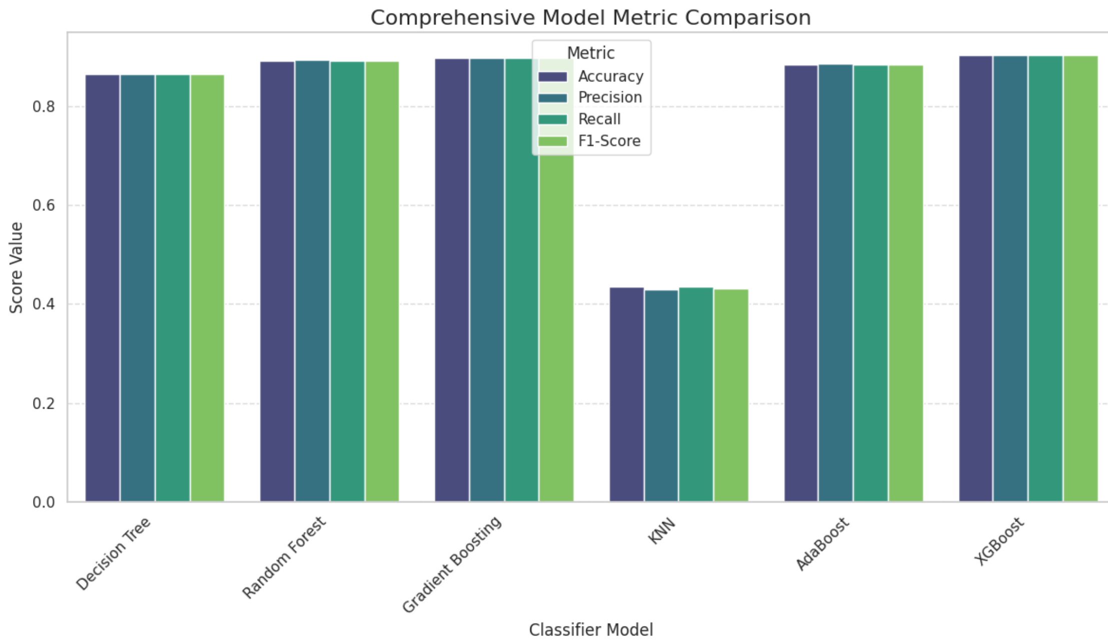
3 CONCLUSION

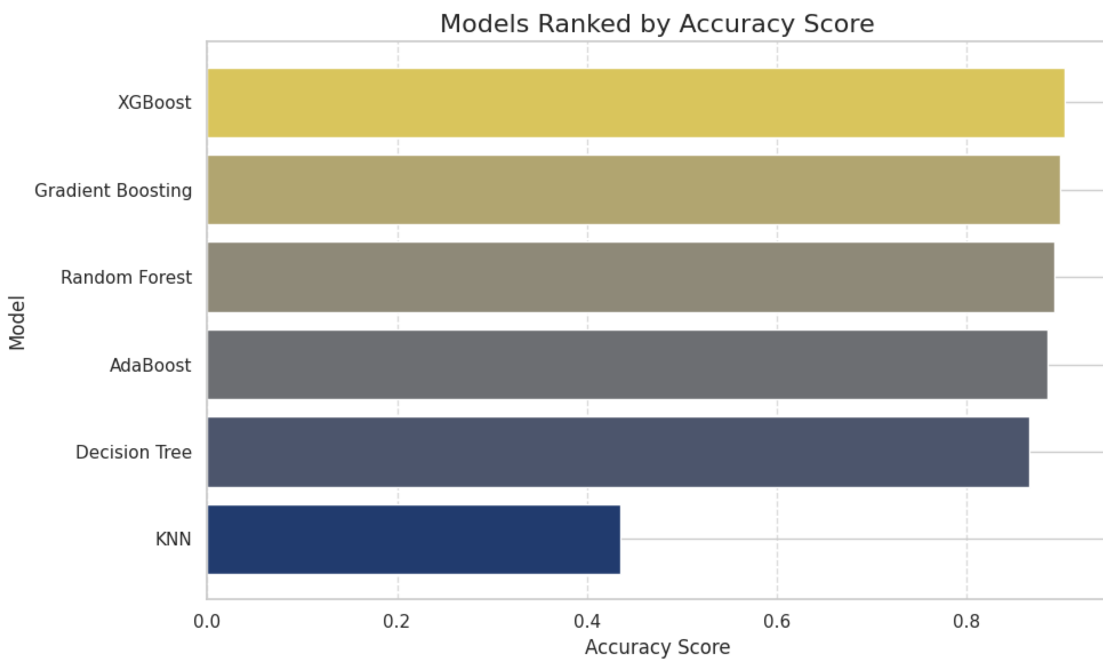
[74]:
```
results = [
    {"Model": "Decision Tree",
     "Accuracy": test_accuracy_dtt,
     "Precision": test_precision_dtt,
     "Recall": test_recall_dtt,
     "F1-Score": test_f1_dtt},

    {"Model": "Random Forest",
     "Accuracy": test_accuracy_rf,
     "Precision": test_precision_rf,
     "Recall": test_recall_rf,
     "F1-Score": test_f1_rf},

    {"Model": "Gradient Boosting",
     "Accuracy": test_accuracy_gb,
     "Precision": test_precision_gb,
     "Recall": test_recall_gb,
     "F1-Score": test_f1_gb},

    {"Model": "KNN",
     "Accuracy": test_accuracy_knn,
     "Precision": test_precision_knn,
     "Recall": test_recall_knn,
     "F1-Score": test_f1_knn},

    {"Model": "AdaBoost",
     "Accuracy": test_accuracy_ab,
     "Precision": test_precision_ab,
     "Recall": test_recall_ab,
     "F1-Score": test_f1_ab},

    {"Model": "XGBoost",
     "Accuracy": test_accuracy_xgb,
     "Precision": test_precision_xgb,
     "Recall": test_recall_xgb,
     "F1-Score": test_f1_xgb},
]

results_df = pd.DataFrame(results).round(4)

print("  Model Comparison DataFrame (results_df) Created:")
display(results_df)

# --- 2. Descriptive Statistics (MPA) ---
```

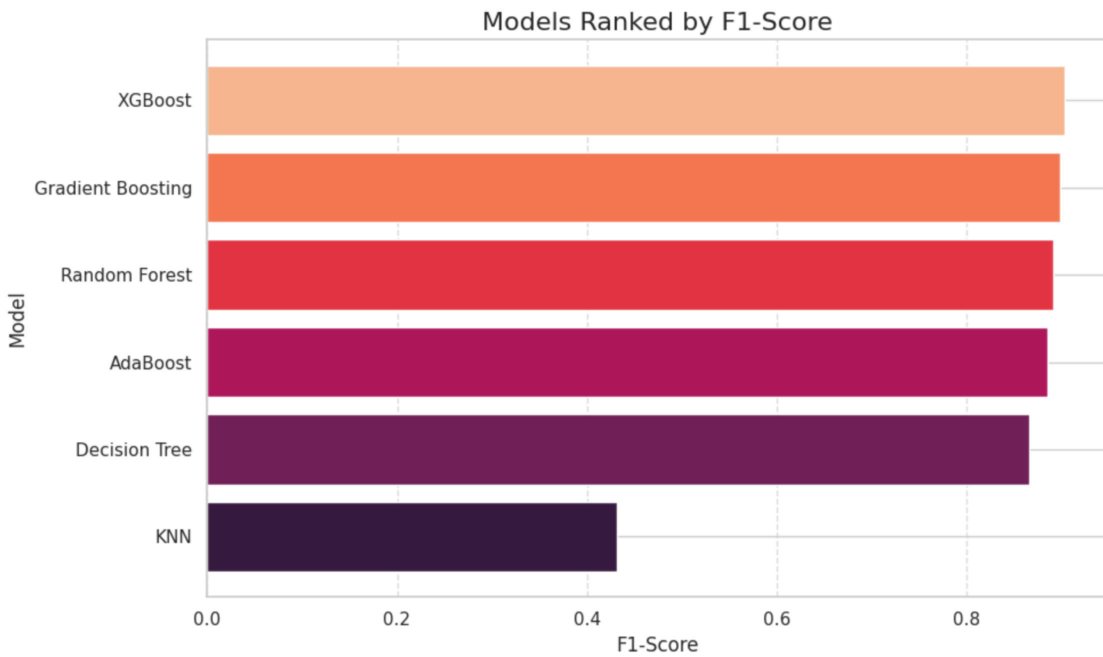
```
print("\n---  Descriptive Statistics of Model Performance Metrics ---")
display(results_df.set_index('Model').describe().round(4))

# --- 3. Key Insights ---
print("\n---  Key Insights from Model Metrics ---")

best_model = results_df.sort_values(by="F1-Score", ascending=False).iloc[0]
print(f"  Best Overall Model (by F1-Score): **{best_model['Model']}** (F1-Score:
 ↪ {best_model['F1-Score']:.4f})")
print(f"  Highest Accuracy Observed: {results_df['Accuracy'].max():.4f}")

# --- 4. Visualizations (MPA) ---

# Prepare data for Seaborn (melting wide format to long format)
results_melted = results_df.melt(id_vars='Model',
                                 var_name='Metric',
                                 value_name='Score',
                                 value_vars=['Accuracy', 'Precision', 'Recall',␣
 ↪'F1-Score'])

# Visualization 1: Comprehensive Metric Comparison
plt.figure(figsize=(12, 7))
sns.barplot(x='Model', y='Score', hue='Metric', data=results_melted,␣
 ↪palette='viridis')
plt.title('Comprehensive Model Metric Comparison', fontsize=16)
plt.ylabel('Score Value', fontsize=12)
plt.xlabel('Classifier Model', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.legend(title='Metric')
plt.tight_layout()
plt.show()

# --- NEW VISUALIZATION: Accuracy Ranking Plot ---
results_accuracy_ranked = results_df.sort_values(by='Accuracy', ascending=True)

plt.figure(figsize=(10, 6))
plt.barh(results_accuracy_ranked['Model'], results_accuracy_ranked['Accuracy'],␣
 ↪color=sns.color_palette("cividis", len(results_accuracy_ranked)))
plt.title('Models Ranked by Accuracy Score', fontsize=16)
plt.xlabel('Accuracy Score', fontsize=12)
plt.ylabel('Model', fontsize=12)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# Visualization 2: Ranked F1-Score Plot
```

```python
results_ranked = results_df.sort_values(by='F1-Score', ascending=True)

plt.figure(figsize=(10, 6))
plt.barh(results_ranked['Model'], results_ranked['F1-Score'], color=sns.
 ↪color_palette("rocket", len(results_ranked)))
plt.title('Models Ranked by F1-Score', fontsize=16)
plt.xlabel('F1-Score', fontsize=12)
plt.ylabel('Model', fontsize=12)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
 Model Comparison DataFrame (results_df) Created:

               Model  Accuracy  Precision  Recall  F1-Score
0      Decision Tree    0.8651     0.8659  0.8651    0.8655
1      Random Forest    0.8919     0.8935  0.8919    0.8915
2  Gradient Boosting    0.8980     0.8982  0.8980    0.8981
3                KNN    0.4355     0.4292  0.4355    0.4311
4           AdaBoost    0.8848     0.8860  0.8848    0.8853
5            XGBoost    0.9034     0.9040  0.9034    0.9036


---  Descriptive Statistics of Model Performance Metrics ---

       Accuracy  Precision  Recall  F1-Score
count    6.0000     6.0000  6.0000    6.0000
mean     0.8131     0.8128  0.8131    0.8125
std      0.1855     0.1884  0.1855    0.1873
min      0.4355     0.4292  0.4355    0.4311
25%      0.8700     0.8709  0.8700    0.8704
50%      0.8884     0.8898  0.8884    0.8884
75%      0.8965     0.8970  0.8965    0.8964
max      0.9034     0.9040  0.9034    0.9036


---  Key Insights from Model Metrics ---
 Best Overall Model (by F1-Score): **XGBoost** (F1-Score: 0.9036)
 Highest Accuracy Observed: 0.9034
```


Comprehensive Model Metric Comparison
Metric
Accuracy
Precision
Recall
F1-Score
Score Value
0.0
0.2
0.4
0.6
0.8
Decision Tree
Random Forest
Gradient Boosting
KNN
AdaBoost
XGBoost
Classifier Model


Models Ranked by Accuracy Score
XGBoost
Gradient Boosting
Random Forest
AdaBoost
Decision Tree
KNN
Model
0.0
0.2
0.4
0.6
0.8
Accuracy Score



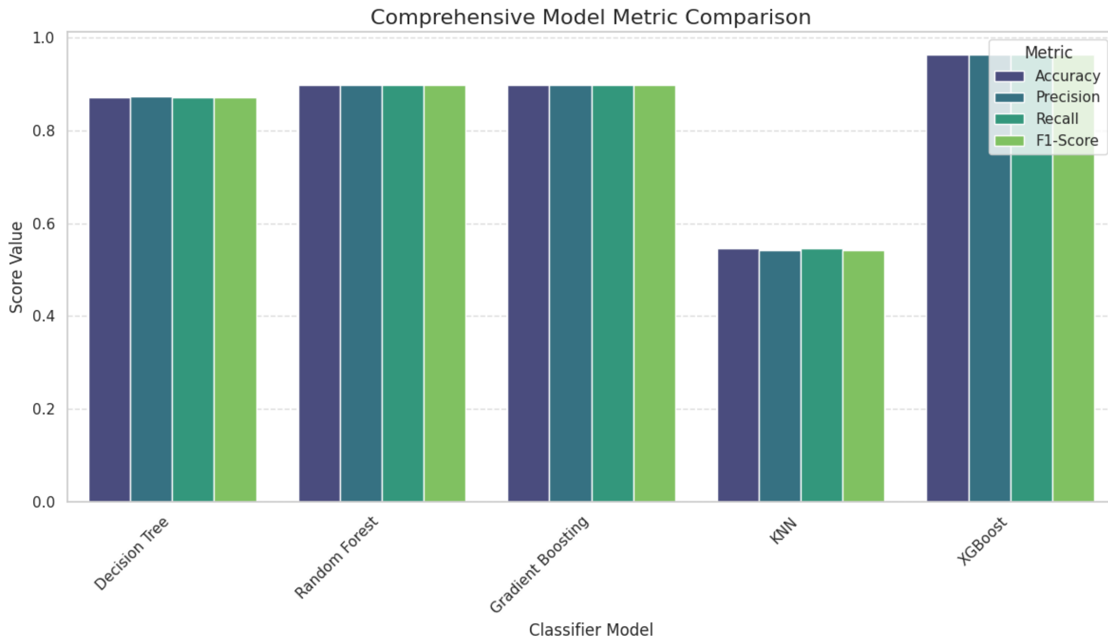
[75]:
```
results = [
    {"Model": "Decision Tree",
     "Accuracy": test_accuracy_dtt_hp,
     "Precision": test_precision_dtt_hp,
     "Recall": test_recall_dtt_hp,
     "F1-Score": test_f1_dtt_hp},

    {"Model": "Random Forest",
     "Accuracy": test_accuracy_rf_hp,
     "Precision": test_precision_rf_hp,
     "Recall": test_recall_rf_hp,
     "F1-Score": test_f1_rf_hp},

    {"Model": "Gradient Boosting",
     "Accuracy": test_accuracy_gb,
     "Precision": test_precision_gb,
     "Recall": test_recall_gb,
     "F1-Score": test_f1_gb},

    {"Model": "KNN",
     "Accuracy": test_accuracy_knn_hp,
     "Precision": test_precision_knn_hp,
     "Recall": test_recall_knn_hp,
     "F1-Score": test_f1_knn_hp},
```

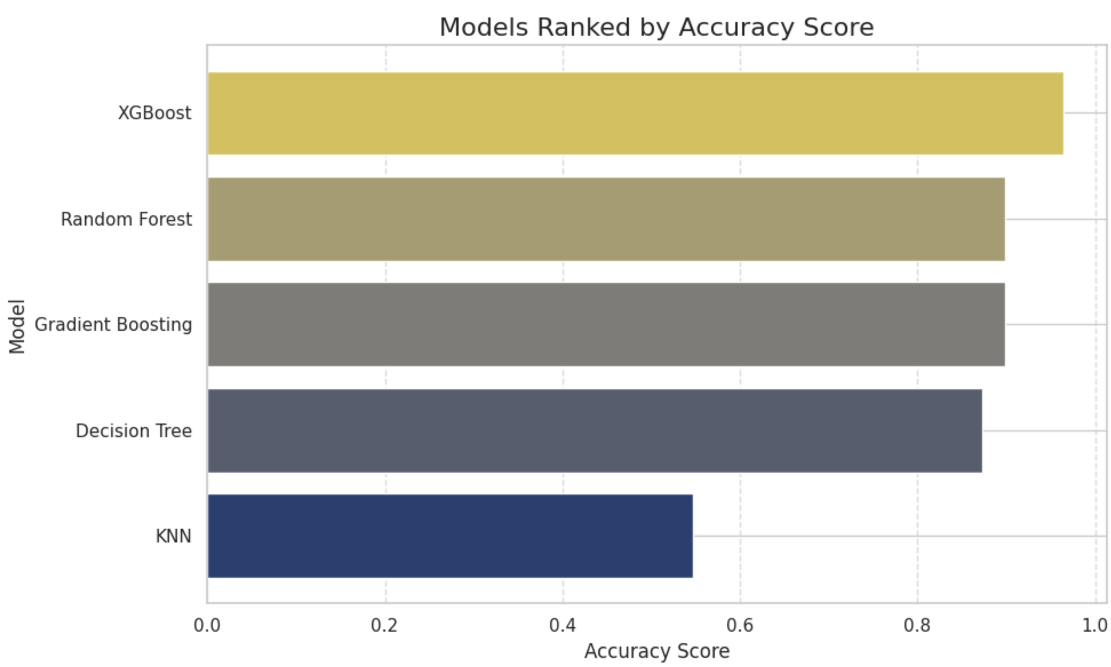
```
    {"Model": "XGBoost",
     "Accuracy": test_accuracy_xgb_hp,
     "Precision": test_precision_xgb_hp,
     "Recall": test_recall_xgb_hp,
     "F1-Score": test_f1_xgb_hp},
]

results_df = pd.DataFrame(results).round(4)

print("  Model Comparison DataFrame (results_df) Created:")
display(results_df)

# --- 2. Descriptive Statistics (MPA) ---

print("\n---  Descriptive Statistics of Model Performance Metrics ---")
display(results_df.set_index('Model').describe().round(4))

# --- 3. Key Insights ---
print("\n---  Key Insights from Model Metrics ---")

best_model = results_df.sort_values(by="F1-Score", ascending=False).iloc[0]
print(f"  Best Overall Model (by F1-Score): **{best_model['Model']}** (F1-Score:
↪ {best_model['F1-Score']:.4f})")
print(f"  Highest Accuracy Observed: {results_df['Accuracy'].max():.4f}")

# --- 4. Visualizations (MPA) ---

# Prepare data for Seaborn (melting wide format to long format)
results_melted = results_df.melt(id_vars='Model',
                                 var_name='Metric',
                                 value_name='Score',
                                 value_vars=['Accuracy', 'Precision', 'Recall',␣
↪'F1-Score'])

# Visualization 1: Comprehensive Metric Comparison
plt.figure(figsize=(12, 7))
sns.barplot(x='Model', y='Score', hue='Metric', data=results_melted,␣
↪palette='viridis')
plt.title('Comprehensive Model Metric Comparison', fontsize=16)
plt.ylabel('Score Value', fontsize=12)
plt.xlabel('Classifier Model', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.legend(title='Metric')
plt.tight_layout()
plt.show()
```

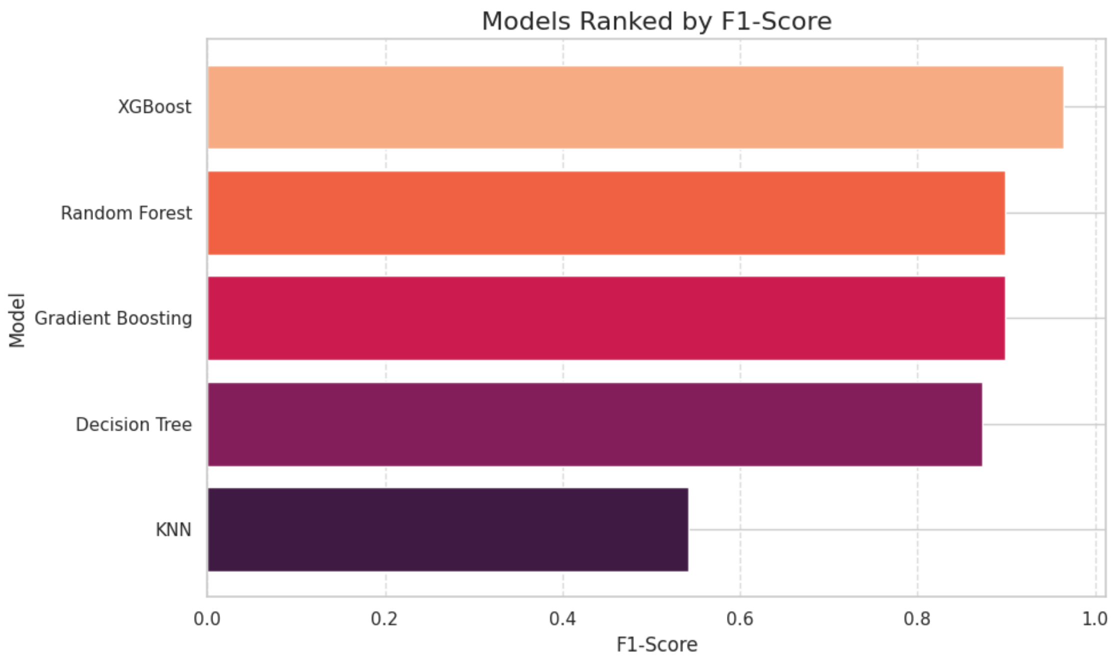
```python
# --- NEW VISUALIZATION: Accuracy Ranking Plot ---
results_accuracy_ranked = results_df.sort_values(by='Accuracy', ascending=True)

plt.figure(figsize=(10, 6))
plt.barh(results_accuracy_ranked['Model'], results_accuracy_ranked['Accuracy'],␣
↪color=sns.color_palette("cividis", len(results_accuracy_ranked)))
plt.title('Models Ranked by Accuracy Score', fontsize=16)
plt.xlabel('Accuracy Score', fontsize=12)
plt.ylabel('Model', fontsize=12)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# Visualization 2: Ranked F1-Score Plot
results_ranked = results_df.sort_values(by='F1-Score', ascending=True)

plt.figure(figsize=(10, 6))
plt.barh(results_ranked['Model'], results_ranked['F1-Score'], color=sns.
↪color_palette("rocket", len(results_ranked)))
plt.title('Models Ranked by F1-Score', fontsize=16)
plt.xlabel('F1-Score', fontsize=12)
plt.ylabel('Model', fontsize=12)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
 Model Comparison DataFrame (results_df) Created:

               Model  Accuracy  Precision  Recall  F1-Score
0      Decision Tree    0.8725     0.8734  0.8725    0.8729
1      Random Forest    0.8983     0.8989  0.8983    0.8984
2  Gradient Boosting    0.8980     0.8982  0.8980    0.8981
3                KNN    0.5468     0.5414  0.5468    0.5416
4            XGBoost    0.9643     0.9643  0.9643    0.9642


---  Descriptive Statistics of Model Performance Metrics ---

       Accuracy  Precision  Recall  F1-Score
count    5.0000     5.0000  5.0000    5.0000
mean     0.8360     0.8352  0.8360    0.8350
std      0.1652     0.1677  0.1652    0.1675
min      0.5468     0.5414  0.5468    0.5416
25%      0.8725     0.8734  0.8725    0.8729
50%      0.8980     0.8982  0.8980    0.8981
75%      0.8983     0.8989  0.8983    0.8984
max      0.9643     0.9643  0.9643    0.9642
```

```
---  Key Insights from Model Metrics ---
 Best Overall Model (by F1-Score): **XGBoost** (F1-Score: 0.9642)
 Highest Accuracy Observed: 0.9643
```






Models Ranked by F1-Score
XGBoost
Random Forest
Gradient Boosting
Decision Tree
KNN
Model
0.0
0.2
0.4
0.6
0.8
1.0
F1-Score