DAFTAR LAMPI˜RAN

Lampiran. 1 Kode Library yang Dibutuhkan

```
# Import Library yang Dibutuhkan
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Import metrik tambahan
from sklearn.metrics import classification_report
# import StratifiedKFold
from sklearn.model_selection import StratifiedKFold
# Library untuk pemodelan dan evaluasi
from sklearn.model_selection import train_test_split,
StratifiedKFold, GridSearchCV
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score,
precision_score, recall_score, f1_score,
confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
# Import hyperparameter tuning
from sklearn.model_selection import GridSearchCV
# Penanganan data tidak seimbang
from imblearn.over_sampling import RandomOverSampler
```

Lampiran. 2 Kode Data Selection

```
# Membaca Dataset
import pandas as pd
df = pd.read_csv("Dataset.csv")
df.head()
```

```
df.info()
# Menghapus Kolom yang Tidak Relevan
df.drop(['ID', 'No_Pation'], axis=1, inplace=True)

# Menghitung jumlah masing-masing kelas
df['CLASS'].value_counts()
```

Lampiran. 3 Kode Analisis Data Eksplorasi (EDA)

```
# Statistik deskriptif fitur numerik
# Menampilkan count, mean, std, min, Q1, median, Q3,
max

from IPython.display import display

desc = df.describe().T
desc.columns = ['Count', 'Mean', 'Std', 'Min', 'Q1
(25%)', 'Median (50%)', 'Q3 (75%)', 'Max']

print('Statistik Deskriptif Fitur Numerik:')
display(desc.round(3))

# Menampilkan distribusi label CLASS sebelum
standardisasi

print('Distribusi CLASS:')
for k, v in df['CLASS'].value_counts().items():
    print(f'  {repr(k):8s} : {v} data
({v/len(df)*100:.1f}%)')

print(f'\nTotal data: {len(df)}

# Mengecek nilai unik pada kolom kategorikal untuk
mendeteksi inkonsistensi
print('Distribusi CLASS:')
for k, v in df['CLASS'].value_counts().items():
    print(f'  {repr(k):8s} : {v} data
({v/len(df)*100:.1f}%)')

print(f'\nTotal data: {len(df)}
```

```
# Heatmap korelasi untuk melihat hubungan antar fitur
numerik
corr = df[numeric_cols].corr()

plt.figure(figsize=(12, 9))
sns.heatmap(corr, annot=True, fmt='.2f',
cmap='RdYlBu_r',
            center=0, square=True, linewidths=0.5,
annot_kws={'size': 9})
plt.title('Heatmap Korelasi Antar Fitur Numerik',
fontsize=13, fontweight='bold', pad=15)
plt.tight_layout()
plt.show()
```

Lampiran. 4 Kode Data Pre-Processing

```
# Mengecek dan menghapus data duplikat
n_sebelum = len(df)
n_dup     = df.duplicated().sum()

print(f'Jumlah data sebelum hapus duplikat :
{n_sebelum}')
print(f'Jumlah baris duplikat                : {n_dup}
({n_dup/n_sebelum*100:.1f}%)')
print()

df = df.drop_duplicates().reset_index(drop=True)

n_sesudah = len(df)
print(f'Jumlah data sesudah hapus duplikat :
{n_sesudah}')
print()

# Standardisasi nilai kategorikal
# Menghapus spasi tersembunyi dan mengubah semua nilai
ke huruf kapital

gender_before = df['Gender'].value_counts().to_dict()
class_before  = df['CLASS'].value_counts().to_dict()



df['Gender'] = df['Gender'].str.strip().str.upper()
df['CLASS']  = df['CLASS'].str.strip().str.upper()

gender_after = df['Gender'].value_counts().to_dict()
```

```
class_after  = df['CLASS'].value_counts().to_dict()

print('Hasil Standardisasi:')
print()
print('[ Gender ]')
print(f'  Sebelum : {gender_before}')
print(f'  Sesudah : {gender_after}')
print()

print('[ CLASS ]')
print(f'  Sebelum : {class_before}')
print(f'  Sesudah : {class_after}')
print()


# Mengambil semua kolom numerik
numeric_cols =
df.select_dtypes(include=np.number).columns

# Fungsi untuk menghitung jumlah outlier pada satu
kolom menggunakan metode IQR

def cek_outlier_iqr(df, kolom):
    Q1    = df[kolom].quantile(0.25)
    Q3    = df[kolom].quantile(0.75)
    IQR   = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    outlier = df[(df[kolom] < lower) | (df[kolom] >
upper)]
    return len(outlier)
# Menghitung jumlah outlier pada setiap fitur numerik

numerik = ['AGE', 'Urea', 'Cr', 'HbA1c', 'Chol', 'TG',
'HDL', 'LDL', 'VLDL', 'BMI']

print('Jumlah outlier per fitur (metode IQR):')
print()
for kolom in numerik:
    n = cek_outlier_iqr(df, kolom)
    print(f'  {kolom:<8} : {n} outlier')
# Eliminasi baris yang tidak valid secara medis
berdasarkan hasil validasi di atas
mask_cr     = (df['Cr'] == 6)
lb_hba1c, _ = iqr_bounds['HbA1c']
mask_hba1c  = df['HbA1c'] < lb_hba1c    # hanya outlier
IQR ekstrem (mis. 0.9), BUKAN < 5.7
```

```
mask_chol   = df['Chol'] <= 1.2
mask_hdl    = df['HDL'] > 4.0
lb_ldl, ub_ldl = iqr_bounds['LDL']
mask_ldl = df['LDL'] > ub_ldl     # pakai batas IQR,
bukan angka tetap 4.9

invalid_mask = mask_cr | mask_hba1c | mask_chol |
mask_hdl | mask_ldl

print('Rincian eliminasi per fitur:')
print()
kriteria = [
    ('Cr',    mask_cr,    'Nilai = 6 (sangat rendah,
tidak mungkin secara klinis; nilai 800
dipertahankan)'),
    ('HbA1c', mask_hba1c, 'Nilai < batas bawah IQR
({lb_hba1c:.2f}%) 4 outlier ekstrem seperti 0.9%, bukan
seluruh rentang normal'),
    ('Chol',  mask_chol,  'Nilai f 1.2 mmol/L (di bawah
batas hipolipidemia 1.3 mmol/L)'),
    ('HDL',   mask_hdl,   'Nilai > 4.0 mmol/L (perlu
diverifikasi ulang)'),
    ('LDL', mask_ldl, 'Nilai > batas atas IQR
({ub_ldl:.2f} mmol/L) 4 outlier statistik ekstrem'),
]
for col, mask, alasan in kriteria:
    n    = mask.sum()
    vals = sorted(df[mask][col].unique())
    print(f'  [{col}] ³ {n} baris dieliminasi')
    print(f'    Alasan : {alasan}')
    print(f'    Nilai  : {[round(v,2) for v in vals]}')
    print()


print(f'Total baris dieliminasi  :
{invalid_mask.sum()}')

df_clean =
df[~invalid_mask].copy().reset_index(drop=True)

print(f'Jumlah data sebelum      : {len(df)}')
print(f'Jumlah data sesudah      : {len(df_clean)}')
print()
```

Lampiran. 5 Kode Tranformasi Data

```
# Encoding Gender
df_clean['Gender'] = df_clean['Gender'].map({'M': 1,
'F': 0})


# Encoding CLASS (target)
df_clean['CLASS'] = df_clean['CLASS'].map({'Y': 2, 'P':
1, 'N': 0})
```

Lampiran. 6 Kode Data Splinting

```
# Pisahkan fitur dan target
# Pemisahan fitur (X) dan label (y)
X = df_clean.drop('CLASS', axis=1)
y = df_clean['CLASS']

# Split data (80% train, 20% test)
# Train-test split dengan stratifikasi (80:20)
X_train, X_test, y_train, y_test = train_test_split( X,
y,test_size=0.2,random_state=42,stratify=y)
```

Lampiran. 7 Kode membangun Model Random Forest

```
from sklearn.model_selection import StratifiedKFold,
GridSearchCV
from sklearn.ensemble import RandomForestClassifier

# 1. Definisikan Cross Validation
cv = StratifiedKFold(n_splits=5, shuffle=True,
random_state=42) # strafidkfoldKarena dataset Anda
multi-kelas dan kemungkinan imbalance., proporsi tiap
kelas dijaga tetap seimbang di setiap fold

# 2. Grid Hyperparameter
param_grid_rforest = {
    'n_estimators': [100, 200, 300],
    'max_depth': [5, 7, 15, None],
    'criterion': ['gini', 'entropy'],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2']
}
```

```
# 3. Definisikan Kamus Metrik Evaluasi untuk 3 Kelas
(Multi-kelas)
scoring_metrics = {
    'accuracy': 'accuracy',
    'precision': 'precision_macro',
    'recall': 'recall_macro',
    'f1': 'f1_macro'
}

# 4. Inisialisasi Model Dasar
rforest_model = RandomForestClassifier(random_state=42)

# 5. Konfigurasi GridSearchCV dengan Banyak Metrik
# Parameter 'refit' diatur ke 'f1' agar pencarian model
terbaik mengutamakan F1-macro tertinggi
grid_rforest = GridSearchCV(
    estimator=rforest_model,
    param_grid=param_grid_rforest,
    scoring=scoring_metrics,
    refit='f1', # F1-score macro digunakan sebagai
dasar pemilihan model terbaik karena mampu mengevaluasi
keseimbangan performa model pada setiap kelas secara
adil tanpa dipengaruhi distribusi jumlah data, sehingga
lebih sesuai untuk klasifikasi multi-kelas dengan
kondisi data tidak seimbang.
    cv=cv,
    n_jobs=-1, #Menggunakan semua core CPU agar
training lebih cepat.
    verbose=1
)

# 6. Proses Training Model
grid_rforest.fit(X_train, y_train)

# 7. Mengambil Model Terbaik
best_rforest = grid_rforest.best_estimator_

# 8. Menampilkan Hasil Parameter Terbaik
print("Hyperparameter Terbaik:")
print(grid_rforest.best_params_)
```

Lampiran. 8 Kode membangun model XGBoost

```
from xgboost import XGBClassifier
from sklearn.model_selection import StratifiedKFold,
GridSearchCV

# CROSS VALIDATION
cv = StratifiedKFold(
    n_splits=5,
    shuffle=True,
    random_state=42
)

# METRIK EVALUASI
scoring_metrics = {
    'accuracy': 'accuracy',
    'precision': 'precision_macro',
    'recall': 'recall_macro',
    'f1': 'f1_macro'
}

# MODEL XGBOOST
xgboost_model =
XGBClassifier(objective='multi:softprob', num_class=3,
random_state=42)

# GRID HYPERPARAMETER
param_grid_xgb = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5],
    'min_child_weight': [1, 3],
    'learning_rate': [0.05, 0.1, 0.2],
    'gamma': [0, 0.2],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}
# GRID SEARCH CV
grid_xgb = GridSearchCV(
    estimator=xgboost_model,
    param_grid=param_grid_xgb,
    scoring=scoring_metrics,
    refit='f1',
    cv=cv,
    n_jobs=-1,
    verbose=1
)
```

```
# TRAINING MODEL
grid_xgb.fit(X_train, y_train)

best_xgbost = grid_xgb.best_estimator_

# HASIL PARAMETER TERBAIK
print("Best Parameter XGBoost:")
print(grid_xgb.best_params_)
```

Lampiran. 9 Kode Prediksi Model Random Forest dan XGBoost

```
# Prediksi pada test set
y_pred_rforest  = best_rforest.predict(X_test)
y_pred_xgbost = best_xgbost.predict(X_test)


print('Prediksi selesai.')
```

Lampiran. 10 Kode Evaluasi Model Random Forest dan XGBoost

```
from sklearn.metrics import (accuracy_score,
classification_report, confusion_matrix,
precision_score, recall_score, f1_score

)

# RANDOM FOREST

print("===== EVALUASI RANDOM FOREST=====")

acc_rforest = accuracy_score(y_test, y_pred_rforest) *
100

precision_rforest = precision_score(y_test,
y_pred_rforest, average='macro') * 100

precision, recall, dan f1-score dari seluruh kelas

recall_rforest = recall_score(y_test, y_pred_rforest,
average='macro') * 100

f1_rforest = f1_score(y_test, y_pred_rforest,
average='macro') * 100

cm_rforest = confusion_matrix(y_test, y_pred_rforest
```

```
print(f"Accuracy  : {acc_rforest:.2f}%")
print(f"Precision  : {precision_rforest:.2f}%")
print(f"Recall     : {recall_rforest:.2f}%")
print(f"F1-Score   : {f1_rforest:.2f}%")


print("\nClassification Report:\n")
print(classification_report(y_test, y_pred_rforest))

print("Confusion Matrix:\n", cm_rforest)

# XGBOOST
print("\n===== EVALUASI XGBOOST =====")

acc_xgbost = accuracy_score(y_test, y_pred_xgbost) *
100
precision_xgbost = precision_score(y_test,
y_pred_xgbost, average='macro') * 100
recall_xgbost = recall_score(y_test, y_pred_xgbost,
average='macro') * 100
f1_xgbost = f1_score(y_test, y_pred_xgbost,
average='macro') * 100
cm_xgbost = confusion_matrix(y_test, y_pred_xgbost)
print(f"Accuracy  : {acc_xgbost:.2f}%")
print(f"Precision  : {precision_xgbost:.2f}%")
print(f"Recall     : {recall_xgbost:.2f}%")
print(f"F1-Score   : {f1_xgbost:.2f}%")


print("\nClassification Report:\n")
print(classification_report(y_test, y_pred_xgbost))
print("Confusion Matrix:\n", cm_xgbost)
# PERBANDINGAN MODEL
```

```
print("\n===== PERBANDINGAN MODEL =====")

print(
    f"Random Forest -> "
    f"Accuracy: {acc_rforest:.2f}% | "
    f"F1: {f1_rforest:.2f}%"
)

print(
    f"XGBoost -> "
    f"Accuracy: {acc_xgbost:.2f}% | "
    f"F1: {f1_xgbost:.2f}%"
```

Lampiran. 11 Kode confusion matrix XGBoost dan Random Forest

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

# Random Forest
plt.figure()
sns.heatmap(cm_rforest, annot=True, fmt='d')
plt.title("Confusion Matrix Random Forest")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# XGBoost
plt.figure()
sns.heatmap(cm_xgbost, annot=True, fmt='d')
plt.title("Confusion Matrix XGBoost")
plt.xlabel("Predicted")
```

```
plt.ylabel("Actual")
plt.show()/
```

Lampiran. 12 Jadwal Penelitian

No	Kegiatan	Bulan																															
		Okt 2025				Nov 2025				Des 2025				Jan 2026				Feb 2026				Mar 2026				Apr 2026				Mei 2026				Jun 2026				Jul 2026
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1
1	Studi Literatur																																					
2	Data Selection																																					
3	Pra-pemrosesan Data																																					
4	Transformasi Data																																					
5	Data Mining																																					
6	Evaluation																																					
7	Penulisan Laporan																																					