LAMPIRAN

Lampiran 1 *Source Code* Python

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,
OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from imblearn.over_sampling import SMOTE
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

stroke_data = pd.read_csv('stroke_train_data.csv')

stroke_data = stroke_data.drop(columns=['id'])

X= stroke_data.drop(columns=['stroke'])
y= stroke_data['stroke']

num_cols = X.select_dtypes(include=['float64',
'int64']).columns.tolist()
cat_cols =
X.select_dtypes(include=['object']).columns.tolist()

num_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])
cat_pipeline = Pipeline([
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])
preprocessor = ColumnTransformer([
    ('num', num_pipeline, num_cols),
    ('cat', cat_pipeline, cat_cols)
])

X_processed = preprocessor.fit_transform(X)

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

X_train, X_test, y_train, y_test =
train_test_split(X_resampled, y_resampled, test_size=0.2,
random_state=42)

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
model = Sequential([
    Dense(64, activation='relu',
input_shape=(X_train.shape[1],)),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Training
history = model.fit(X_train, y_train, epochs=100,
batch_size=32, validation_split=0.2)

# Plot Accuracy dan Loss
plt.figure(figsize=(12, 5))

# Akurasi
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val
Accuracy')
plt.title("Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()

# Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title("Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()

plt.tight_layout()
plt.show()


from sklearn.metrics import classification_report,
confusion_matrix

# Prediksi
y_pred = (model.predict(X_test) > 0.5).astype(int)

# Evaluasi
print("Classification Report:\n",
classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
#input data untuk prediksi testing
def prediksi_pasien_stroke(input_data):
    """
    input_data: tuple berisi data pasien dalam format:
    (gender, age, hypertension, heart_disease, ever_married,
work_type,
     Residence_type, avg_glucose_level, bmi, smoking_status)
    """
    import numpy as np

    # Ubah ke dataframe satu baris (sesuai format training)
    input_df = pd.DataFrame([input_data], columns=[
        'gender', 'age', 'hypertension', 'heart_disease',
        'ever_married', 'work_type', 'Residence_type',
        'avg_glucose_level', 'bmi', 'smoking_status'
    ])

    # Lewat preprocessing pipeline
    input_processed = preprocessor.transform(input_df)

    # Prediksi
    prediction = model.predict(input_processed)

    # Output
    print("Probabilitas stroke:", prediction[0][0])
    if prediction[0][0] < 0.5:
        print("🟢 Pasien Tidak Terkena Stroke")
    else:
        print("🔴 Pasien Terkena Stroke")

prediksi_pasien_stroke((
    'Famale',         # gender
    57.0,           # age
    1,              # hypertension
    0,              # heart_disease
    'Yes',          # ever_married
    'Private',      # work_type
    'Rural',        # Residence_type
    129.54,         # avg_glucose_level
    60.9,           # bmi
    'smokes'         # smoking_status
))

import pickle
# Simpan
model.save("stroke_mlp_model.h5")

# Load
# from tensorflow.keras.models import load_model
# model = load_model("stroke_mlp_model.h5")

filename = 'Stroke.sav'
pickle.dump(model, open(filename,'wb'))

import joblib
joblib.dump(preprocessor, "preprocessor.pkl")
```

Lampiran 2 *Source code Streamlit*

```
import streamlit as st
import pandas as pd
import numpy as np
import joblib
from tensorflow.keras.models import load_model

# ===== Load model dan preprocessor =====
model = load_model("stroke_mlp_model.h5")
preprocessor = joblib.load("preprocessor.pkl")  # simpan
pipeline preprocessor sebelumnya
st.title("🧠 Deteksi Penyakit Stroke")
st.write("Masukkan data pasien untuk memprediksi apakah pasien
terkena stroke.")
# ===== Form input =====
gender = st.selectbox("Jenis Kelamin", ["Male", "Female",
"Other"])
age = st.number_input("Umur",min_value = 0,max_value= 100)
hypertension = st.selectbox("Hipertensi", [0, 1])
heart_disease = st.selectbox("Penyakit Jantung", [0, 1])
ever_married = st.selectbox("Pernah Menikah", ["Yes", "No"])
work_type = st.selectbox("Pekerjaan", ["Private", "Self-
employed", "Govt_job", "children", "Never_worked"])
residence_type = st.selectbox("Tipe Tempat Tinggal", ["Urban",
"Rural"])
avg_glucose_level = st.number_input("Rata-rata Glukosa",
min_value=50.0, max_value=300.0, value=100.0)
bmi = st.number_input("BMI", min_value=10.0, max_value=100.0,
value=25.0)
smoking_status = st.selectbox("Status Merokok", ["formerly
smoked", "never smoked", "smokes", "Unknown"])

# ===== Prediksi saat tombol ditekan =====
if st.button("Prediksi Stroke"):
    # Buat DataFrame satu baris
    input_data = pd.DataFrame([[
        gender, age, hypertension, heart_disease,
ever_married,
        work_type, residence_type, avg_glucose_level, bmi,
smoking_status
    ]], columns=[
        'gender', 'age', 'hypertension', 'heart_disease',
'ever_married',
        'work_type', 'Residence_type', 'avg_glucose_level',
'bmi', 'smoking_status'
    ])

    # Preprocessing
    input_processed = preprocessor.transform(input_data)

    # Prediksi
    prediction = model.predict(input_processed)[0][0]

    # Output
    st.subheader("Hasil:")
    st.write(f"Probabilitas stroke: **{prediction:.2f}**")

    if prediction < 0.5:
        st.success("🟢 Pasien Tidak Terkena Stroke")
    else:
        st.error("🔴 Pasien Terkena Stroke")
```