Demo Aplikasi

```
9) Save trained model ke file .h5
def save trained model():
    keras model = tf.keras.Sequential([
        tf.keras.layers.Input(shape=(len(FEATURE_COLS),)),
        tf.keras.layers.Dense(64, activation="relu"),
        tf.keras.layers.Dense(32, activation="relu"),
tf.keras.layers.Dense(1, activation="sigmoid"),
   # Latih model dengan dataset global gabungan
X_all = pd.concat([X_dinsos, X_dukcapil, X_kemenkes], axis=0).values.astype("float32")
    y_all = np.concatenate([y_dinsos, y_dukcapil, y_kemenkes]).astype("float32")
    keras model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=0.01),
        loss = tf.keras.losses.Binary Crossentropy (from\_logits = False),\\
        metrics=[tf.keras.metrics.BinaryAccuracy()]
    keras_model.fit(X_all, y_all, epochs=10, batch_size=32, verbose=1)
    keras_model.save("trained_model.h5")
    print("☑ Model disimpan ke trained model.h5")
if __name_
    save_trained_model()
```

- Membuat arsitektur neural network sederhana (2 hidden layer).
- Menggabungkan dataset dari beberapa sumber.
- Melatih model dengan optimizer Adam + loss BinaryCrossentropy.
- Menyimpan model ke file trained model.h5 agar bisa digunakan lagi.

```
import joblib
import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

import joblib

i
```

Kode ini menyimpan metadata preprocessing ke file preprocess.pkl menggunakan **joblib**. File ini nantinya akan dipakai bersamaan dengan model .h5 supaya pipeline prediksi berjalan mulus:

- 1. Load preprocess.pkl → terapkan normalisasi dengan mins dan rng.
- 2. Load trained model.h5 \rightarrow lakukan prediksi pada data baru.

```
Epoch 1/10
141/141 [=
                                Epoch 2/10
141/141 [=
                                   ==] - 0s 2ms/step - loss: 0.0941 - binary_accuracy: 0.9549
Epoch 3/10
141/141 [==
                             ======] - 0s 2ms/step - loss: 0.0877 - binary_accuracy: 0.9596
Epoch 4/10
                                ====] - 0s 2ms/step - loss: 0.0739 - binary_accuracy: 0.9662
141/141 [==
Epoch 5/10
                               ======] - 0s 2ms/step - loss: 0.0702 - binary_accuracy: 0.9653
141/141 [=:
Epoch 6/10
141/141 [=
                               =====] - 0s 2ms/step - loss: 0.0644 - binary_accuracy: 0.9691
Epoch 7/10
141/141 [==
                                   ==] - 0s 2ms/step - loss: 0.0667 - binary_accuracy: 0.9696
Epoch 8/10
                                ====] - 0s 2ms/step - loss: 0.0674 - binary_accuracy: 0.9696
141/141 [==
Epoch 9/10
141/141 [==
                                  ====] - 0s 2ms/step - loss: 0.0623 - binary_accuracy: 0.9722
Epoch 10/10
                       ========] - 0s 2ms/step - loss: 0.0564 - binary_accuracy: 0.9724
141/141 [===
/mnt/c/KP/MATERI/4.5. Laporan & Video Demo/venv/lib/python3.11/site-packages/keras/src/engine/trai
considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.
 saving_api.save_model(
☑ Model disimpan ke trained_model.h5
 Preprocessing info disimpan ke preprocess.pkl
```

- 1. Model sudah dilatih 10 epoch dengan hasil akurasi tinggi (97.24%).
- 2. File yang dihasilkan:
 - trained model.h5 → model machine learning yang siap dipakai.
 - **preprocess.pkl** → metadata preprocessing untuk normalisasi data baru.

Menggunakan Flask untuk menjalankan atau mencoba model yang sudah dibuat :

App.py

1. Import Library

```
papp.py > ♥ preprocess_input
   import tensorflow as tf
   import pandas as pd
   import numpy as np
   import joblib
   from flask import Flask, render_template, request
```

- **TensorFlow** \rightarrow load model .h5 hasil training.
- pandas, numpy → manipulasi data dan preprocessing.
- **joblib** → load preprocessing info (preprocess.pkl).
- Flask → framework web Python untuk membuat form input & API prediksi.

2. Inisialisasi Flask

- Membuat aplikasi Flask
- 3. Load model dan Pre-Processing Info

```
# === Load trained model ===
keras_model = tf.keras.models.load_model("trained_model.h5", compile=False)

# === Load preprocessing info ===
preproc = joblib.load("preprocess.pkl")
FEATURE_COLS = preproc["FEATURE_COLS"]
mins = preproc["mins"]
rng = preproc["rng"]
```

- **trained model.h5** \rightarrow model hasil training.
- **preprocess.pkl** → menyimpan metadata preprocessing (nama kolom fitur, nilai minimum & range untuk normalisasi).
- compile=False → hanya load model untuk inference (tidak butuh compile ulang optimizer/loss).
- 4. Fungsi Pre-Processing Input

```
def preprocess_input(form_data):
    df = pd.DataFrame([form_data])

# Tambahkan kolom kosong untuk semua FEATURE_COLS yang tidak ada di input
for col in FEATURE_COLS:
    if col not in df:

df[col] = 0.0

# Scale sesuai global
df_scaled = (df[FEATURE_COLS] - mins) / rng
df_scaled = df_scaled.fillna(0.0)
return df_scaled.values.astype("float32")
```

- Mengubah dictionary dari form menjadi DataFrame.
- Menambahkan kolom kosong untuk fitur yang hilang (0.0).
- Normalisasi sesuai parameter training (mins dan rng).
- Output berupa array float32 siap diprediksi.

5. Routing Halaman Utama

```
30
31 @app.route("/")
32 def index():
33 return render_template("form.html")
```

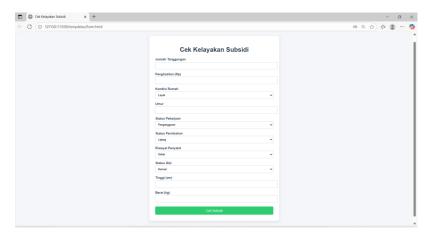
- Saat buka / (homepage), akan menampilkan form.html (form input user).
- 6. Routing Aplikasi

```
@app.route("/predict", methods=["POST"])
def predict():
    form = request.form
    input_data = {
         "jumlah_tanggungan": int(form["jumlah_tanggungan"]),
        "penghasilan": float(form["penghasilan"]),
        "umur": int(form["umur"]),
        "tinggi_cm": float(form["tinggi_cm"]),
        "berat_kg": float(form["berat_kg"]),
    input_data[f"kondisi_rumah__{form['kondisi_rumah']}"] = 1.0
    input_data[f"status_pekerjaan__{form['status_pekerjaan']}"] = 1.0
    input_data[f"status_pernikahan__{form['status_pernikahan']}"] = 1.0
    input_data[f"riwayat_penyakit__{form['riwayat_penyakit']}"] = 1.0
    input_data[f"status_gizi__{form['status_gizi']}"] = 1.0
    X_user = preprocess_input(input_data)
    pred = keras_model.predict(X_user)[0][0]
label = "▼ Layak Subsidi" if pred >= 0.5 else "X Tidak Layak Subsidi"
    return render_template("result.html", result=label, score=pred)
```

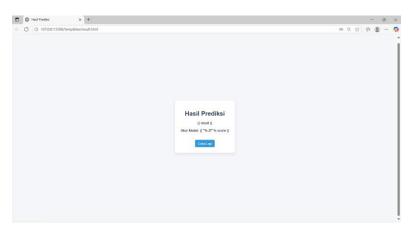
Alurnya:

- 1. Ambil **input numerik** (tanggungan, penghasilan, umur, tinggi, berat).
- 2. Encode kategori menjadi one-hot (status, kondisi rumah, gizi, dll).
- 3. Preprocessing (normalize) \rightarrow array float32.
- 4. Prediksi dengan model Keras.
- 5. Jika hasil $\geq =0.5 \rightarrow \text{Layak Subsidi}$, kalau $<0.5 \rightarrow \text{Tidak Layak}$.
- 6. Render hasil ke **result.html**.
- 7. Menjalankan Server

Form.html

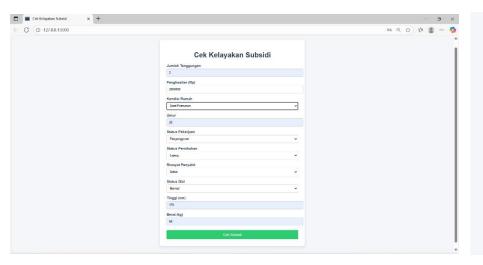


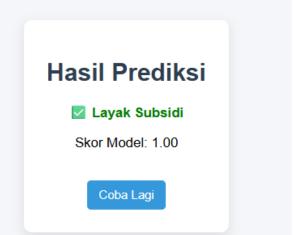
Result.html



UJI COBA

1. Kategori Layak





2. Kategori Tidak Layak

