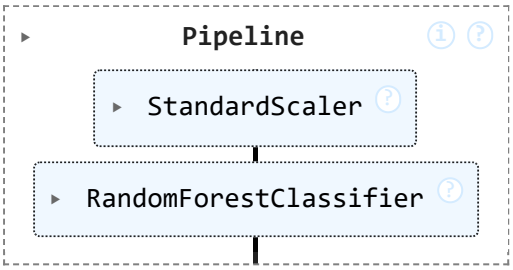
Demo

In [5]:
```
import joblib
import gradio as gr
import pandas as pd
from sklearn.metrics import accuracy_score
```

In [6]:
```
model = joblib.load('heart_attack_prediction_model.joblib')
```

In [7]:
```
model
```

Out[7]:



In [8]:
```
import pandas as pd
import gradio as gr

def predict_heart_attack(
    gender,
    region,
    EKG_results,
    age,
    income_level,
    hypertension,
    diabetes,
    cholesterol_level,
    obesity,
    waist_circumference,
    family_history,
    smoking_status,
    alcohol_consumption,
    physical_activity,
    dietary_habits,
    air_pollution_exposure,
    stress_level,
    sleep_hours,
    blood_pressure_systolic,
    blood_pressure_diastolic,
    fasting_blood_sugar,
    cholesterol_hdl,
    cholesterol_ldl,
    triglycerides,
    previous_heart_disease,
    medication_usage,
    participated_in_free_screening,
):
    smoking_status_map = {'Never': 1, 'Past':2, 'Current':3}
    air_pollution_exposure_map = {'Low': 1, 'Moderate': 2, 'High': 3}
    income_level_map = {'Low': 1, 'Middle': 2, 'High': 3}
    alcohol_consumption_map = {'None': 1, 'Moderate': 2, 'High': 3}
```

```
    physical_activity_map = {'Low': 1, 'Moderate': 2, 'High': 3}
    stress_level_map = {'Low': 1, 'Moderate': 2, 'High': 3}

    # Convert categorical inputs to numerical
    input_data = pd.DataFrame({
        'gender_encoded': [1 if gender == 'Female' else 0],
        'region_encoded': [1 if region == 'Urban' else 0],
        'EKG_results_encoded': [1 if EKG_results == 'Normal' else 0],
        'age': [float(age)],
        'income_level': [income_level_map[income_level]],
        'hypertension': [1 if hypertension == 'Yes' else 0],
        'diabetes': [1 if diabetes == 'Yes' else 0],
        'cholesterol_level': [float(cholesterol_level)],
        'obesity': [1 if obesity == 'Yes' else 0],
        'waist_circumference': [float(waist_circumference)],
        'family_history': [1 if family_history == 'Yes' else 0],
        'smoking_status': [smoking_status_map[smoking_status]],
        'alcohol_consumption': [alcohol_consumption_map[alcohol_consumption]],
        'physical_activity': [physical_activity_map[physical_activity]],
        'dietary_habits': [2 if dietary_habits == 'Unhealthy' else 1],
        'air_pollution_exposure': [air_pollution_exposure_map[air_pollution_expo
        'stress_level': [stress_level_map[stress_level]],
        'sleep_hours': [float(sleep_hours)],
        'blood_pressure_systolic': [float(blood_pressure_systolic)],
        'blood_pressure_diastolic': [float(blood_pressure_diastolic)],
        'fasting_blood_sugar': [float(fasting_blood_sugar)],
        'cholesterol_hdl': [float(cholesterol_hdl)],
        'cholesterol_ldl': [float(cholesterol_ldl)],
        'triglycerides': [float(triglycerides)],
        'previous_heart_disease': [1 if previous_heart_disease == 'Yes' else 0],
        'medication_usage': [1 if medication_usage == 'Yes' else 0],
        'participated_in_free_screening': [1 if participated_in_free_screening =
        'age_group': [1 if age < 40 else 2 if age < 55 else 3 if age < 64 else 4
        'health_risk_score': [
            (1 if hypertension == 'Yes' else 0) +
            (1 if diabetes == 'Yes' else 0) +
            (1 if obesity == 'Yes' else 0) +
            (1 if family_history == 'Yes' else 0) +
            smoking_status_map[smoking_status] +
            alcohol_consumption_map[alcohol_consumption] +
            (3 - physical_activity_map[physical_activity]) +  # Inverted so high
            (2 if dietary_habits == 'Unhealthy' else 1) +
            air_pollution_exposure_map[air_pollution_exposure] +
            stress_level_map[stress_level] +
            (4 - income_level_map[income_level])  # Inverted so lower income = h
        ],
        'obesity_risk_score': [
            (1 if obesity == 'Yes' else 0) +
            (waist_circumference *
            (0.5 if ((gender == 'Male' and waist_circumference > 102) or
                    (gender == 'Female' and waist_circumference > 88))
            else 0))
        ],
        'stress_to_sleep_ratio': [
            (stress_level_map[stress_level]) / sleep_hours if sleep_hours != 0 e
        ],
        'mean_arterial_pressure': [
            (2 * blood_pressure_systolic + blood_pressure_diastolic) / 3
        ],
        'triglyceride-hdl-ratio': [
```

```
            triglycerides / cholesterol_hdl if cholesterol_hdl != 0 else 0
        ]
    })

    # Make prediction (ensure your model is loaded)
    prediction = model.predict(input_data)
    if prediction [0] == 1:
        proba = model.predict_proba(input_data)[0][1]
    else:
        proba = model.predict_proba(input_data)[0][0]

    # Return more user-friendly output
    result = "Heart Attack" if prediction[0] == 1 else "No Heart Attack"
    return f"Prediction: {result} (Probability: {proba:.2f})"

with gr.Blocks(title='Heart Attack Prediction in Indonesia') as demo:
    gr.Markdown("# Heart Attack Prediction in Indonesia")
    with gr.Row():
        with gr.Column():
            gender = gr.Radio(['Male', 'Female'], label='Gender')
            region = gr.Radio(['Urban', 'Rural'], label='Region')
            EKG_results = gr.Radio(['Normal', 'Abnormal'], label='EKG Results')
            obesity = gr.Radio(['Yes', 'No'], label='Obesity')
            family_history = gr.Radio(['Yes', 'No'], label='Family History of He
            alcohol_consumption = gr.Radio(['None', 'Moderate', 'High'], label='
            physical_activity = gr.Radio(['Low', 'Moderate', 'High'], label='Phy
            dietary_habits = gr.Radio(['Healthy', 'Unhealthy'], label='Dietary H
            medication_usage = gr.Radio(['Yes', 'No'], label='Medication Usage')
            participated_in_free_screening = gr.Radio(['Yes', 'No'], label='Part
            income_level = gr.Radio(['Low', 'Middle', 'High'], label='Income Lev
            previous_heart_disease = gr.Radio(['Yes', 'No'], label='Previous Hea
            hypertension = gr.Radio(['Yes', 'No'], label='Hypertension')
            diabetes = gr.Radio(['Yes', 'No'], label='Diabetes')
            smoking_status = gr.Radio(['Never', 'Past', 'Current'], label='Smoki
            air_pollution_exposure = gr.Radio(['Low', 'Moderate', 'High'], label
            stress_level = gr.Radio(['Low', 'Moderate', 'High'], label='Stress L

            age = gr.Slider(25, 90, step=0.5, label='Age (minimum: 25)')
            cholesterol_level = gr.Slider(100, 318, step=0.1, label='Cholesterol
            waist_circumference = gr.Slider(45, 142, step=0.1, label='Waist Circ
            sleep_hours = gr.Slider(3, 9, step=0.01, label='sleep_hours (minimum
            blood_pressure_systolic = gr.Slider(85, 174, step=0.1, label='Systol
            blood_pressure_diastolic = gr.Slider(50, 109, step=0.1, label='Diast
            fasting_blood_sugar = gr.Slider(70, 193, step=0.1, label='Fasting Bl
            cholesterol_hdl = gr.Slider(20, 79, step=0.1, label='Cholesterol HDL
            cholesterol_ldl = gr.Slider(26, 235, step=0.1, label='Cholesterol LD
            triglycerides = gr.Slider(50, 298, step=0.1, label='Triglycerides (m

            predict_button = gr.Button("Predict Heart Attack")
            prediction_output = gr.Textbox(label='Prediction Result', value='Cli

        predict_button.click(
            fn = predict_heart_attack,
            inputs=[
            gender,
            region,
            EKG_results,
            age,
            income_level,
            hypertension,
```

```
            diabetes,
            cholesterol_level,
            obesity,
            waist_circumference,
            family_history,
            smoking_status,
            alcohol_consumption,
            physical_activity,
            dietary_habits,
            air_pollution_exposure,
            stress_level,
            sleep_hours,
            blood_pressure_systolic,
            blood_pressure_diastolic,
            fasting_blood_sugar,
            cholesterol_hdl,
            cholesterol_ldl,
            triglycerides,
            previous_heart_disease,
            medication_usage,
            participated_in_free_screening,
        ],

            outputs = prediction_output
        )

if __name__ == "__main__":
    demo.launch()
```

```
* Running on local URL:  http://127.0.0.1:7861

To create a public link, set `share=True` in `launch()`.
```

In [ ]: