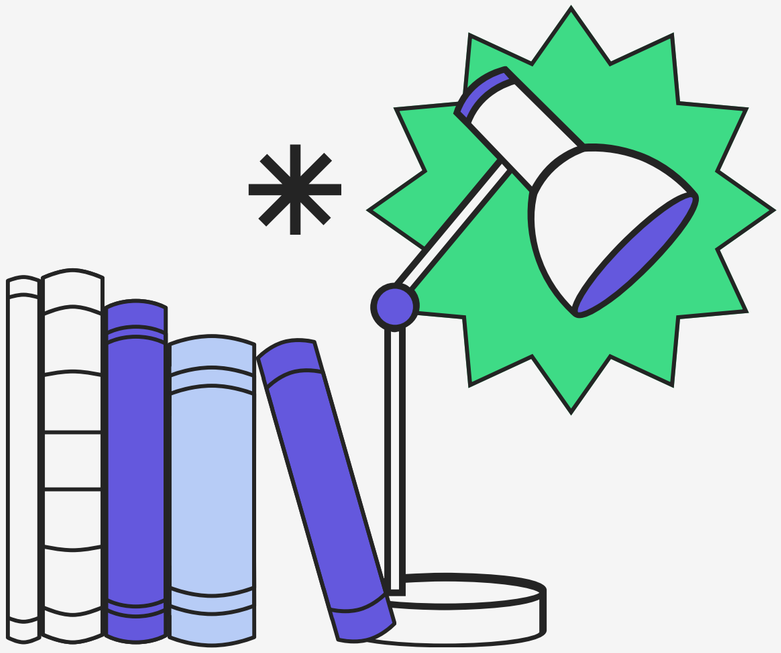
Random Forest

Kelompok 5

CHESAR RIZQI FEBRIANTO (234311035)
DZAKI ANWAR ZULFAHMI (234311037)
ICON PRIAGAMIS (234311042)
VERI TRI ANGGORO (234311055)

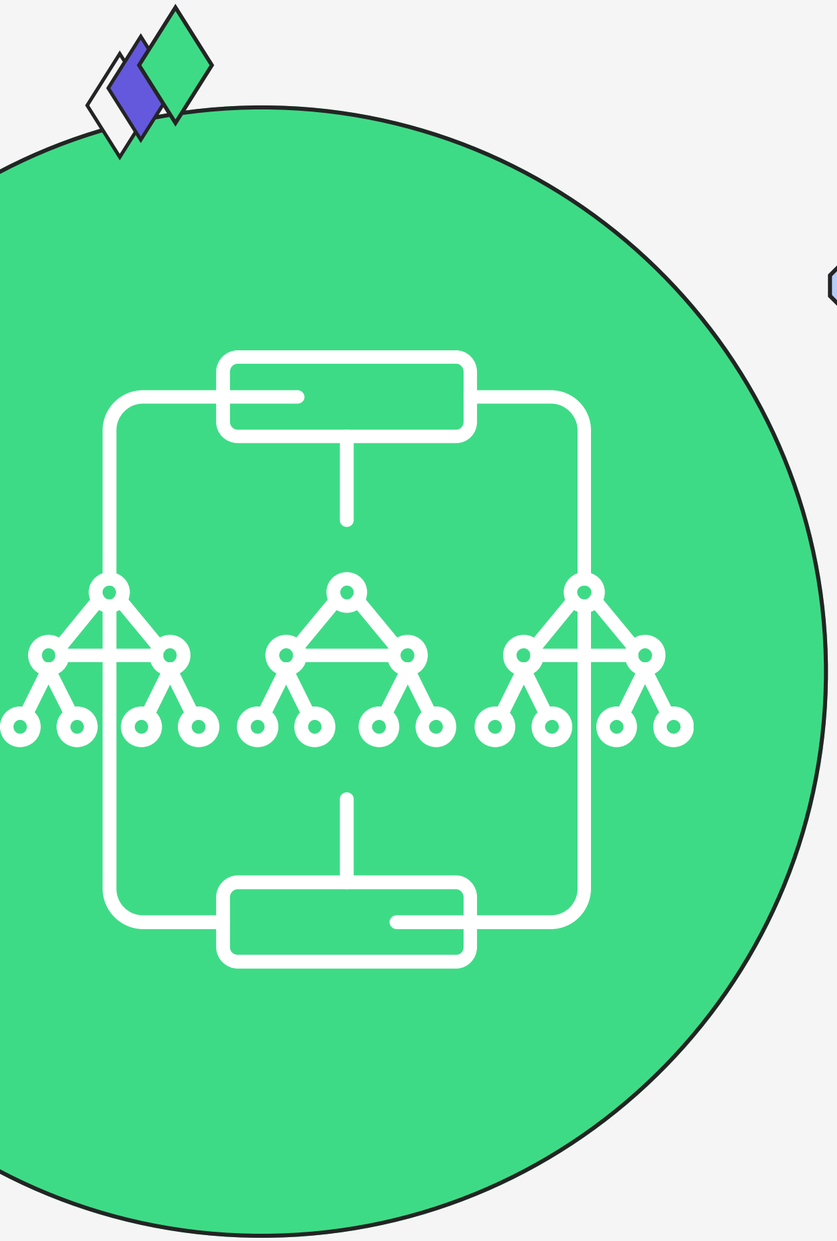
*

Latar Belakang



Random Forest merupakan salah satu algoritma machine learning yang banyak digunakan untuk tugas klasifikasi maupun regresi. Metode ini dikembangkan untuk mengatasi berbagai keterbatasan pada Decision Tree tunggal yang cenderung overfitting dan kurang stabil. Dengan menggabungkan banyak pohon keputusan secara bersamaan, Random Forest mampu memberikan hasil prediksi yang lebih akurat dan konsisten.

Dalam berbagai bidang seperti kesehatan, ekonomi, keamanan data, hingga analisis citra, Random Forest terbukti memiliki performa yang kuat karena kemampuannya menangani dataset yang besar, kompleks, dan berdimensi tinggi. Selain itu, metode ini relatif mudah diimplementasikan dan sering dijadikan pilihan dasar untuk pemodelan awal sebelum menggunakan algoritma yang lebih kompleks.



Apa Itu Random Forest ??

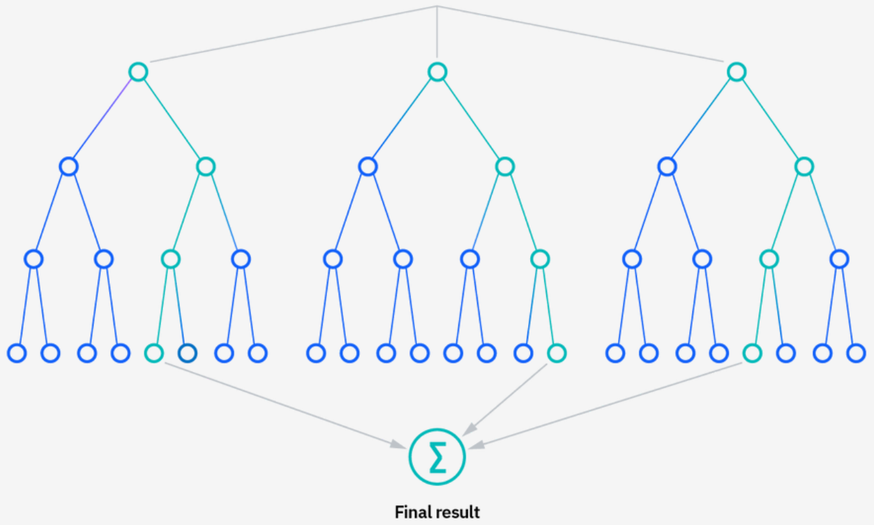
Random Forest adalah algoritma machine learning yang bekerja dengan menggabungkan banyak pohon keputusan (decision tree) untuk menghasilkan prediksi yang lebih akurat dan stabil. Algoritma ini diperkenalkan oleh Leo Breiman pada tahun 2001,dan seringkali juga dikreditkan bersama Adele Cutler, yang mengembangkan ide-ide sebelumnya.

Penggabungan banyak pohon ini membuat Random Forest lebih tahan terhadap overfitting dan noise dibandingkan model tunggal. Dalam proses prediksinya, setiap pohon memberikan suara (voting) dan hasil akhir ditentukan berdasarkan suara mayoritas dari seluruh pohon.

Contoh

Setiap tree voting → hasil akhir = mayoritas suara

Cara Kerja Random Forest



Secara sederhana, mekanisme kerja Random Forest adalah dengan pendekatan 'demokrasi' menggunakan banyak pohon keputusan (decision tree). Alih-alih bergantung pada satu pohon tunggal, algoritma ini membangun puluhan pohon yang masing-masing dilatih secara independen menggunakan sampel data acak yang unik (proses bootstrap sampling). Ketika dihadapkan pada tugas prediksi, setiap pohon akan memberikan 'suara' atau klasifikasinya sendiri. Dengan hasil akhir kemudian ditentukan melalui proses voting, di mana prediksi yang paling banyak dipilih oleh mayoritas pohon akan menjadi jawaban final.

Hipotesis Function

Decision Tree (Model Dasar)

$$h_j(x) = \text{prediksi dari pohon ke-}j$$

Setiap pohon dalam hutan yang memiliki fungsi hipotesisnya sendiri (Ht(x)). Pohon ini memecah data berdasarkan aturan logika (If-Else) hingga mencapai kesimpulan di daun (leaf node).

- Fungsi Splitting yaitu Pohon mencari pemisahan terbaik menggunakan metrik seperti Gini Impurity atau Entropy (untuk klasifikasi) dan Variance Reduction (untuk regresi).

Hipotesis Function

ENSEMBLE MODEL

Majority Vote Aggregation Function

$$H(x) = \text{mode}\{h_1(x), h_2(x), ..., h_T(x)\}$$

Rumus ini menunjukkan bahwa prediksi akhir dari Random Forest diperoleh dari vote terbanyak (mode) dari seluruh pohon keputusan yang ada di dalam forest dengan :

- Ti(x) adalah hasil prediksi dari pohon keputusan ke-i.
- Semua prediksi digabungkan, kemudian label yang paling sering muncul dipilih sebagai output akhir.
- Dengan cara ini, Random Forest menjadi lebih stabil, akurasi meningkat, dan lebih tahan terhadap noise.

Averaging (Rata-rata)

$$H(x) = \frac{1}{T}\sum_{t=1}^{T} h_t(x)$$

Rumus yang digunakan untuk menggabungkan hasil dari banyak *decision tree* untuk mencapai satu hasil akhir yang lebih baik.

COST FUNCTION CLASIFIKASI

Kunci Jawaban

Gini Impurity

digunakan untuk menentukan split terbaik pada setiap node pohon keputusan. Nilai Gini menunjukkan seberapa campur aduk kelas dalam sebuah node. Node ideal adalah node dengan nilai Gini rendah (semakin “murni”) Dengan tujuan Menjaga akurasi tinggi saat memisahkan gerakan komplek

$$Gini = \sum_{i=1}^{C} p_i(1 - p_i)$$

Keterangan :

- C = jumlah kelas
- Pi = peluang memilih sampel dari kelas i
- (1−Pi) : peluang memilih sampel dari **kelas selain** i
- Pi(1−Pi) : peluang memilih satu sampel dari kelas i dan satu dari kelas lain

COST FUNCTION CLASIFIKASI

Kunci Jawaban

Logistik Regresi
di mana m adalah jumlah contoh pelatihan, y adalah label sebenarnya, h(x) adalah label yang diprediksi, dan w adalah parameter bobot

$$J(w) = (-1/m) * jumlah(y * log(h(x)) + (1-y) * log(1-h(x)))$$

Entropy
Entropy mengukur ketidakpastian distribusi kelas dalam sebuah node.

$$Entropy(t) = -\sum_{i=1}^{k} p_i \log_2 p_i$$

Dengan maksud rumusnya yaitu :

- Entropy = 0 → node murni
- Entropy lebih sensitif terhadap perubahan kecil pada probabilitas daripada Gini
- Random Forest kebanyakan menggunakan Gini karena lebih cepat dihitung

COST FUNCTION REGRESI

Kunci Jawaban

Mean Squared Error (MSE)

sebuah metode dengan mengukur rata-rata kuadrat selisih antara nilai target aktual dan nilai prediksi node.

$$MSE(t) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2$$

Keterangan :

- N = jumlah data di node
- ***yi*** = nilai target ke- i
- $\bar{y}$ = rata-rata nilai target di node

Mean Absolute Error (MAE)

Fungsi nya mengukur selisih rata-rata absolut antara nilai prediksi dan nilai aktual. Fungsi ini sendiri digunakan dalam regresi linier, pohon keputusan, dan hutan acak

Keterangan MAE jarang dipakai karena:

- tidak sensitif terhadap perbedaan kecil
- lebih lambat dihitung
- hasil split sering tidak stabil

$$MAE = \frac{1}{N}\sum |y_i - \bar{y}|$$

COST FUNCTION REGRESI

Kunci Jawaban

Root Mean Squared Error (RMSE)

sebuah metode varian dari MSE, di mana akar kuadrat dari MSE diambil. Metode ini sering digunakan untuk mengukur kinerja model regresi ketika skala variabel target menjadi pertimbangan.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Keterangan :

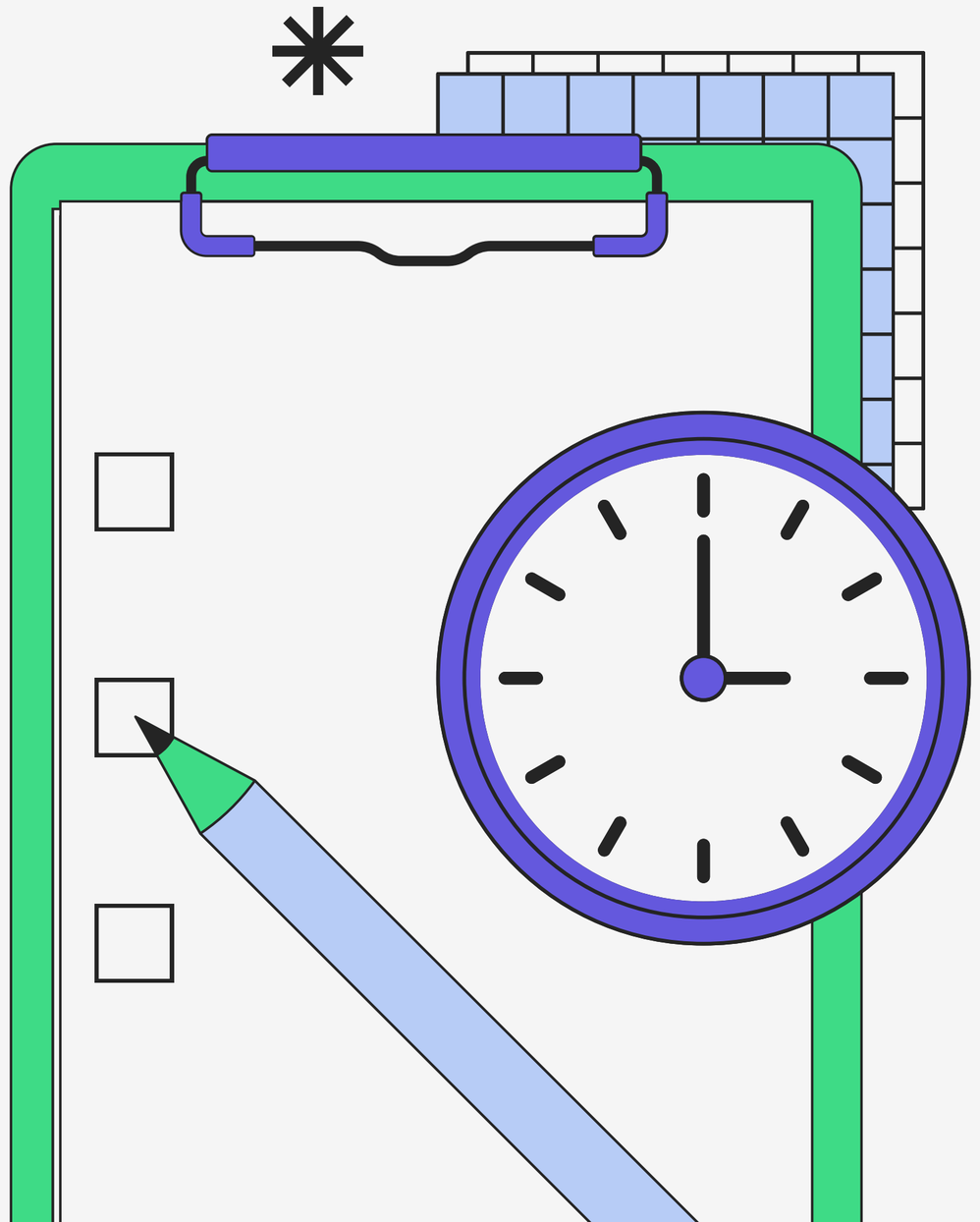
- *yi* = nilai sebenarnya (actual)
- *yi* = nilai prediksi model
- n = jumlah sampel

Contoh Studi Kasus

Kasus: Sistem harus mengenali gestur 'Pose' secara real-time.

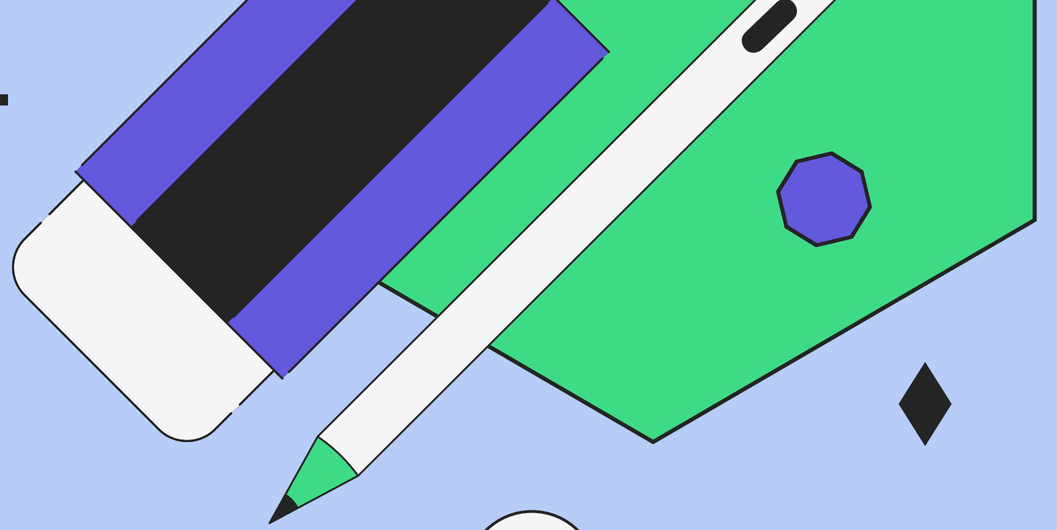
Tujuannya mendeteksi gestur tangan apakah terdeteksi atau tidaknya, dengan Input (Data) :

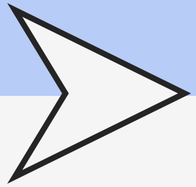
- Dataset berisi kumpulan gambar atau frame wajah.
- Setiap gambar diekstraksi menjadi fitur berupa titik koordinat wajah (misal: mata, hidung, mulut, kontur wajah).
- Label:
  - 1 = wajah terdeteksi
  - 0 = tidak ada wajah



Alur Sistem Deteksi Wajah

- Train Data
- Lalu Menyalakan Realtime app
- Kamera menyala lalu mengambil gambar gestur tangan
- Ekstraksi fitur Mediapipe
- Fitur diproses → masuk ke Random Forest
- Model memprediksi: Gestur tubuh / Tidak ada gestur tubuh
- Output ditampilkan ke layar





PENJELASAN KODE

pose_extractor.py

pose_extractor.py > ...

```python
import cv2
import mediapipe as mp
import numpy as np
import os

mp_pose = mp.solutions.pose
pose = mp_pose.Pose(static_image_mode=False, min_detection_confidence=0.5)
mp_drawing = mp.solutions.drawing_utils

def extract_pose_features(image_path):
    image = cv2.imread(image_path)
    if image is None:
        print(f" Gambar tidak ditemukan: {image_path}")
        return None

    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = pose.process(rgb_image)

    if not results.pose_landmarks:
        print(f" Pose tidak terdeteksi di: {image_path}")
        return None

    # koor
    landmarks = []
    for lm in results.pose_landmarks.landmark:
        landmarks.append([lm.x, lm.y, lm.visibility])

    # Ubah jadi array 1D
    feature_vector = np.array(landmarks).flatten()

    return feature_vector

# contoh
if __name__ == "__main__":
    test_image = "images/thinking/thinking_01.jpg"
    features = extract_pose_features(test_image)
    if features is not None:
        print("Fitur pose berhasil diekstrak:", features.shape)
```

Code ini digunakan untuk mendeteksi pose tubuh manusia dari sebuah gambar dan kemudian mengubah hasil deteksi (landmark pose) menjadi vektor fitur numerik yang dapat dipakai untuk:

- Machine learning (misalnya klasifikasi pose)
- Analisis gerakan
- Sistem pengenalan gesture
- Dataset pose extraction

Code ini memanfaatkan MediaPipe Pose untuk mendeteksi 33 titik landmark tubuh

train_model.py

```python
import os
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import joblib

from pose_extractor import extract_pose_features

X = []  # fitur pose
y = []  # label

for folder in ["thinking", "idle", "idea", "shocked","hidup_jokowi","monkey_fuck", "other"]:
    if folder == "thinking":
        label = 1
    elif folder == "idle":
        label = 2
    elif folder == "idea":
        label = 3
    elif folder == "shocked":
        label = 4
    elif folder == "hidup_jokowi":
        label = 5
    elif folder == "monkey_fuck":
      label = 6
    else:
        label = 0

    folder_path = os.path.join("images", folder)
    for filename in os.listdir(folder_path):
        if filename.endswith(".jpg") or filename.endswith(".png"):
            img_path = os.path.join(folder_path, filename)
            features = extract_pose_features(img_path)
            if features is not None:
                X.append(features)
                y.append(label)

X = np.array(X)
y = np.array(y)
if len(X) == 0:
    print(" Tidak ada data untuk dilatih. Pastikan kamu punya gambar di folder.")
else:
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    print("Akurasi:", model.score(X_test, y_test))
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))

    # Simpan model
    joblib.dump(model, "pose_classifier.pkl")
    print("\n Model disimpan sebagai 'pose_classifier.pkl'")
```

- Mengambil fitur pose dari gambar menggunakan fungsi extract_pose_features()
- Memberikan label (thinking, idle, idea, shocked, dll) pada setiap gambar
- Melatih model machine learning (Random Forest) agar bisa mengenali pose berdasarkan landmark MediaPipe
- Mengevaluasi model
- Menyimpan model menjadi file pose_classifier.pkl

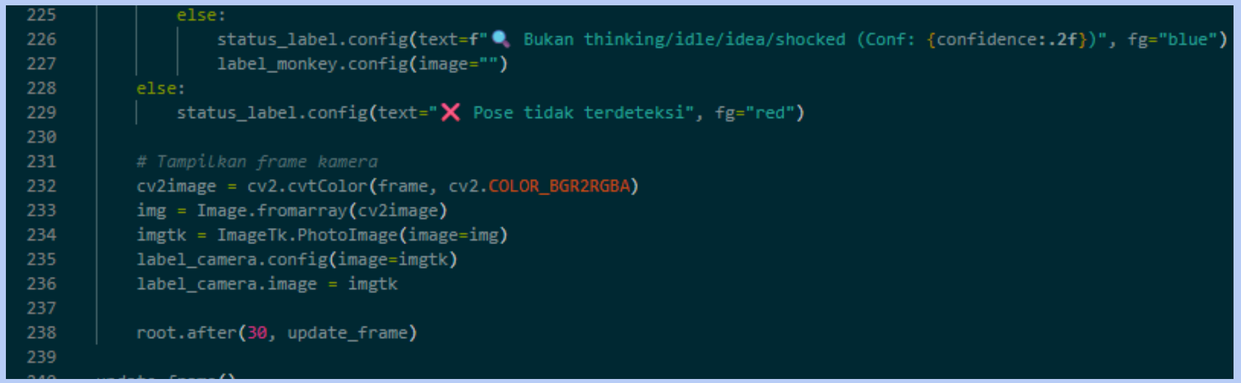
realtime_app.py

realtime_app.py > ...

```python
import cv2
import mediapipe as mp
import numpy as np
import joblib
from PIL import Image, ImageTk
import tkinter as tk
from tkinter import ttk
import os
import pygame
pygame.mixer.init()

#model
model = joblib.load("pose_classifier.pkl")

#mediapipe
mp_pose = mp.solutions.pose
mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils

pose = mp_pose.Pose(static_image_mode=False, min_detection_confidence=0.5)
hands = mp_hands.Hands(static_image_mode=False, max_num_hands=2, min_detection_confidence=0.5)

#suara
def play_sound(sound_file):
    try:
        pygame.mixer.music.load(sound_file)
        pygame.mixer.music.play()
    except:
        print(f"Tidak bisa mainkan: {sound_file}")

#sedang berfikir
img_path = "monkey_thinking.png"
if not os.path.exists(img_path):
    img_path = "monkey_thinking.jpg"

if os.path.exists(img_path):
    monkey_img = Image.open(img_path)
    monkey_img = monkey_img.resize((600, 600))
else:
    print("ERROR: Tidak ada file 'monkey_thinking.png' atau 'monkey_thinking.jpg'!")
    exit()

#apa lo liat liat
idle_img_path = "monkey_idle.png"
if not os.path.exists(idle_img_path):
    idle_img_path = "monkey_idle.jpg"

if os.path.exists(idle_img_path):
    monkey_idle_img = Image.open(idle_img_path)
    monkey_idle_img = monkey_idle_img.resize((600, 600))
else:
    print("ERROR: Tidak ada file 'monkey_idle.png' atau 'monkey_idle.jpg'!")
    exit()

#aku punya ide
idea_img_path = "monkey_idea.png"
if not os.path.exists(idea_img_path):
    idea_img_path = "monkey_idea.jpg"

if os.path.exists(idea_img_path):
    monkey_idea_img = Image.open(idea_img_path)
    monkey_idea_img = monkey_idea_img.resize((600, 600))
else:
    print("ERROR: Tidak ada file 'monkey_idea.png' atau 'monkey_idea.jpg'!")
    exit()

#ba kaget
shocked_img_path = "monkey_shocked.png"
if not os.path.exists(shocked_img_path):
    shocked_img_path = "monkey_shocked.jpg"

if os.path.exists(shocked_img_path):
    monkey_shocked_img = Image.open(shocked_img_path)
    monkey_shocked_img = monkey_shocked_img.resize((600, 600))
else:
    print("ERROR: Tidak ada file 'monkey_shocked.png' atau 'monkey_shocked.jpg'!")
    exit()
```

```python
#jokowi
jokowi_img_path = "hidup_jokowi.png"
if not os.path.exists(jokowi_img_path):
    jokowi_img_path = "hidup_jokowi.jpg"

if os.path.exists(jokowi_img_path):
    hidup_jokowi_img = Image.open(jokowi_img_path)
    hidup_jokowi_img = hidup_jokowi_img.resize((600,600))
else:
    print("No Jokowi")

#monekey F
# monkey_fuck_img_path = "monkey_fuck.png"
# if not os.path.exists(monkey_fuck_img_path):
#     monkey_fuck_img_path = "monkey_fuck.jpg"

# if os.path.exists(monkey_fuck_img_path):
#     monkey_fuck_img = Image.open(monkey_fuck_img_path)
#     monkey_fuck_img = monkey_fuck_img.resize((400,400))
# else:
#     print("no Monkey")

#tkinter
root = tk.Tk()
root.title("Monkey Pose Detector")
root.geometry("1920x1080")

#gradien background
canvas = tk.Canvas(root, width=1280, height=1080)
canvas.pack(fill="both", expand=True)

# gradien verti
def create_gradient(canvas, color1, color2, width, height):
    r1, g1, b1 = root.winfo_rgb(color1)
    r2, g2, b2 = root.winfo_rgb(color2)

    def rgb(r, g, b):
        return f'#{int(r):04x}{int(g):04x}{int(b):04x}'

    for i in range(height):
        r = r1 + (r2 - r1) * i // height
        g = g1 + (g2 - g1) * i // height
        b = b1 + (b2 - b1) * i // height
        canvas.create_line(0, i, width, i, fill=rgb(r >> 8, g >> 8, b >> 8))

#biru ke ungu
# create_gradient(canvas, "#000033", "#330066", 800, 600)

# Frame utama di atas canvas
frame_main = tk.Frame(canvas, bg="#000033")
canvas.create_window((0, 0), window=frame_main, anchor="nw")

# Label untuk kamera (dengan background transparan)
label_camera = tk.Label(frame_main, bg="#000033")
label_camera.pack(side="left", padx=10, pady=10)

# Label untuk gambar monyet (dengan background transparan)
label_monkey = tk.Label(frame_main, bg="#000033")
label_monkey.pack(side="right", padx=10, pady=10)

# Status label (dengan background dan teks yang terlihat)
status_label = tk.Label(root, text="Tunggu... Deteksi pose", font=("Arial", 14), bg="#000033", fg="white")
status_label.place(x=400, y=550, anchor="center")

cap = cv2.VideoCapture(0)

def extract_features_from_frame(frame):
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = pose.process(rgb_frame)
    if not results.pose_landmarks:
        return None
```

```python
    landmarks = []
    for lm in results.pose_landmarks.landmark:
        landmarks.append([lm.x, lm.y, lm.visibility])
    return np.array(landmarks).flatten()

def update_frame():
    ret, frame = cap.read()
    if not ret:
        return

    # Deteksi pose
    features = extract_features_from_frame(frame)
    if features is not None:
        # Prediksi
        prediction = model.predict([features])[0]
        confidence = model.predict_proba([features])[0].max()

        # Tampilkan landmark pose dan tangan di frame
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        pose_results = pose.process(rgb_frame)
        hands_results = hands.process(rgb_frame)

        # Gambar landmark pose
        if pose_results.pose_landmarks:
            mp_drawing.draw_landmarks(
                frame,
                pose_results.pose_landmarks,
                mp_pose.POSE_CONNECTIONS,
                mp_drawing.DrawingSpec(color=(0, 255, 0), thickness=2, circle_radius=3),
                mp_drawing.DrawingSpec(color=(255, 0, 0), thickness=2)
            )

        # Gambar landmark tangan (jari)
        if hands_results.multi_hand_landmarks:
            for hand_landmarks in hands_results.multi_hand_landmarks:
                mp_drawing.draw_landmarks(
                    frame,
                    hand_landmarks,
                    mp_hands.HAND_CONNECTIONS,
                    mp_drawing.DrawingSpec(color=(0, 255, 255), thickness=2, circle_radius=2),
                    mp_drawing.DrawingSpec(color=(0, 255, 255), thickness=2)
                )

        # Tampilkan hasil prediksi
        if prediction == 1:
            status_label.config(text=f"💡 POSE BERPIKIR! (Conf: {confidence:.2f})", fg="green")
            photo = ImageTk.PhotoImage(monkey_img)
            label_monkey.config(image=photo)
            label_monkey.image = photo
        elif prediction == 2:
            status_label.config(text=f"😴 IDLE - Diam saja (Conf: {confidence:.2f})", fg="orange")
            photo = ImageTk.PhotoImage(monkey_idle_img)
            label_monkey.config(image=photo)
            label_monkey.image = photo
        elif prediction == 3:
            status_label.config(text=f"💡 IDEAAA! (Conf: {confidence:.2f})", fg="purple")
            photo = ImageTk.PhotoImage(monkey_idea_img)
            label_monkey.config(image=photo)
            label_monkey.image = photo
        elif prediction == 4:
            status_label.config(text=f"😱 SHOCKED! (Conf: {confidence:.2f})", fg="red")
            photo = ImageTk.PhotoImage(monkey_shocked_img)
            label_monkey.config(image=photo)
            label_monkey.image = photo
        elif prediction == 5:
            status_label.config(text=f"Hidup Jokowi✊(Conf: {confidence:.2f})", fg="red")
            photo = ImageTk.PhotoImage(hidup_jokowi_img)
            play_sound("audio/hidup_jokowi.mp3")
            label_monkey.config(image=photo)
            label_monkey.image = photo
        # elif prediction == 6:
        #     status_label.config(text=f"Fuck u(Conf: {confidence:.2f})", fg="red")
        #     photo = ImageTk.PhotoImage(monkey_fuck_img)
        #     label_monkey.config(image=photo)
        #     label_monkey.image = photo
```



```
            else:
                status_label.config(text=f"🔍 Bukan thinking/idle/idea/shocked (Conf: {confidence:.2f})", fg="blue")
                label_monkey.config(image="")
        else:
            status_label.config(text="❌ Pose tidak terdeteksi", fg="red")

        # Tampilkan frame kamera
        cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
        img = Image.fromarray(cv2image)
        imgtk = ImageTk.PhotoImage(image=img)
        label_camera.config(image=imgtk)
        label_camera.image = imgtk

        root.after(30, update_frame)
```

Program ini adalah aplikasi real-time pose detection menggunakan webcam yang:

- Mendeteksi pose tubuh manusia dengan MediaPipe
- Mengubah pose menjadi fitur (landmark) → lalu memprediksi kelas pose menggunakan model ML (Random Forest)
- Menampilkan gambar monyet sesuai pose yang terdeteksi
- Menggunakan Tkinter untuk tampilan GUI
- Menggunakan pygame untuk memutar suara
- Menampilkan landmark tubuh + tangan realtime di video feed

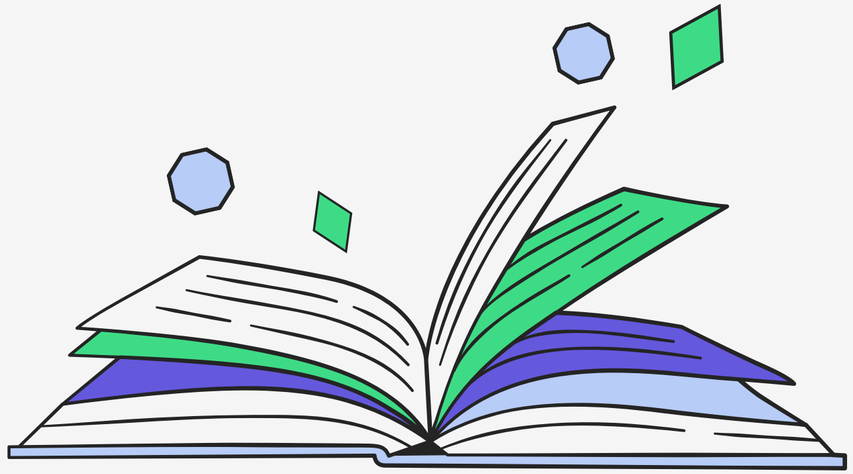
KESIMPULAN

Sebagai algoritma yang serbaguna, Random Forest membuktikan bahwa penggabungan banyak model sederhana atau yang sering disebut sebagai *weak learners* dapat menghasilkan keputusan yang jauh lebih cerdas dan akurat. Melalui teknik ensemble yang memanfaatkan keragaman setiap pohon, ia mampu 'menjinakkan' dataset yang rumit dan berdimensi tinggi tanpa terjebak dalam bias atau overfitting, masalah klasik yang sering menghantui model decision tree tunggal.

Dengan mekanisme voting atau rata-rata, algoritma ini meredam noise pada data sehingga memberikan hasil prediksi yang konsisten dan stabil, bahkan ketika dihadapkan pada data baru. Karena sifatnya yang adaptif, namun tetap mudah diterapkan tanpa memerlukan penyetelan parameter yang terlalu rumit, tidak mengherankan jika metode ini terus menjadi andalan utama praktisi data di berbagai bidang industri maupun akademis.

Referensi

Mathspace. "3.01 Persamaan Multi-Langkah | Matematika Kelas 8 Common Core 8 - Edisi 2023." Diakses 24 Juli 2023, https://mathspace.co/textbooks/syllabuses/Syllabus-1157/topics/Topic-21905/subtopics/Subtopic-279899A



TERIMA KASIH