AI for Optimizing Artificial Blood Vessel Selection

# Step 1: Import Library dan Upload Data

Pertama-tama, kita perlu mengimpor beberapa pustaka Python yang akan digunakan untuk analisis data dan pemodelan AI. Setelah itu, kita akan mengunggah data Anda.  
Berikut adalah kode untuk mengimpor pustaka yang dibutuhkan dan membaca data dari file yang diunggah:

# Mengimpor pustaka yang dibutuhkan  
import pandas as pd # Untuk manipulasi data  
import numpy as np # Untuk perhitungan numerik  
from sklearn.model\_selection import train\_test\_split # Untuk membagi data  
from sklearn.preprocessing import StandardScaler # Untuk normalisasi data  
from sklearn.ensemble import RandomForestClassifier # Untuk model klasifikasi  
from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix # Untuk evaluasi model  
import matplotlib.pyplot as plt # Untuk visualisasi  
import seaborn as sns # Untuk visualisasi lebih lanjut  
import joblib # Untuk menyimpan model  
  
# Membaca data dari file yang diunggah  
file\_path = '/mnt/data/artificial\_blood\_vessel\_dataset.xlsx'  
df = pd.read\_excel(file\_path)  
  
# Menampilkan beberapa baris pertama untuk memeriksa struktur data  
df.head()

# Step 2: Pembersihan dan Pra-pemrosesan Data

Pada langkah ini, kita akan melakukan beberapa pemrosesan data. Ini mencakup:  
- Mengonversi kolom kategorikal (misalnya, jenis kelamin, jenis pembuluh darah, riwayat operasi) menjadi format numerik agar bisa diproses oleh model.  
- Menghapus kolom yang tidak relevan seperti `Patient\_ID`.  
- Menangani nilai yang hilang jika ada.  
Berikut adalah kode untuk pembersihan data:

# Mengonversi kolom kategorikal menjadi numerik  
df['Gender'] = df['Gender'].map({'Female': 1, 'Male': 0})  
df['Vessel\_Type'] = df['Vessel\_Type'].map({'Artery': 0, 'Vein': 1, 'Capillary': 2})  
df['Previous\_Surgery\_History'] = df['Previous\_Surgery\_History'].map({'Yes': 1, 'No': 0})  
  
# Menghapus kolom 'Patient\_ID' karena tidak relevan untuk analisis  
df = df.drop(columns=['Patient\_ID'])  
  
# Memeriksa apakah ada nilai yang hilang  
df.isnull().sum()  
  
# Jika ada nilai yang hilang, kita bisa menggantinya dengan nilai rata-rata  
df.fillna(df.mean(), inplace=True)  
  
# Menampilkan beberapa baris setelah pemrosesan  
df.head()

# Step 3: Memisahkan Fitur (X) dan Target (y)

Sekarang kita akan memisahkan kolom fitur (variabel yang digunakan untuk prediksi) dan target (variabel yang ingin kita prediksi).

# Menentukan fitur (X) dan target (y)  
X = df.drop(columns=['Selected\_for\_Artificial\_Vessel']) # Semua kolom kecuali target  
y = df['Selected\_for\_Artificial\_Vessel'] # Kolom target yang ingin diprediksi

# Step 4: Membagi Data Menjadi Data Pelatihan dan Pengujian

Kita akan membagi dataset menjadi dua bagian: data pelatihan untuk melatih model dan data pengujian untuk menguji performa model setelah dilatih. Biasanya, pembagian ini menggunakan rasio 80:20 atau 70:30.

# Membagi data menjadi data pelatihan dan data pengujian (80% pelatihan, 20% pengujian)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Normalisasi data (fitur berada dalam skala yang sama)  
scaler = StandardScaler()  
X\_train = scaler.fit\_transform(X\_train) # Normalisasi data pelatihan  
X\_test = scaler.transform(X\_test) # Normalisasi data pengujian

# Step 5: Membangun dan Melatih Model AI (Random Forest)

Pada langkah ini, kita akan menggunakan Random Forest Classifier, yang merupakan algoritma pembelajaran mesin yang kuat dan sering digunakan untuk masalah klasifikasi.

# Membuat model Random Forest  
model = RandomForestClassifier(n\_estimators=100, random\_state=42)  
  
# Melatih model dengan data pelatihan  
model.fit(X\_train, y\_train)

# Step 6: Evaluasi Model

Setelah model dilatih, langkah selanjutnya adalah mengevaluasi kinerjanya dengan menggunakan data pengujian. Ini termasuk menghitung akurasi, membuat laporan klasifikasi, dan menampilkan matriks kebingungungan.

# Membuat prediksi pada data pengujian  
y\_pred = model.predict(X\_test)  
  
# Menghitung akurasi  
accuracy = accuracy\_score(y\_test, y\_pred)  
print(f'Akurasi Model: {accuracy:.2f}')  
  
# Menampilkan laporan klasifikasi  
print(classification\_report(y\_test, y\_pred))  
  
# Matriks kebingungungan  
cm = confusion\_matrix(y\_test, y\_pred)  
  
# Visualisasi matriks kebingungungan  
plt.figure(figsize=(8, 6))  
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Tidak Terpilih', 'Terpilih'], yticklabels=['Tidak Terpilih', 'Terpilih'])  
plt.xlabel('Prediksi')  
plt.ylabel('Aktual')  
plt.title('Matriks Kebingungungan')  
plt.show()

# Step 7: Menyimpan Model untuk Penggunaan di Masa Depan

Jika Anda puas dengan kinerja model, Anda bisa menyimpan model yang sudah dilatih untuk digunakan kembali di masa depan tanpa perlu melatihnya lagi.

# Menyimpan model yang sudah dilatih ke dalam file  
joblib.dump(model, '/mnt/data/artificial\_blood\_vessel\_model.pkl')  
  
# Memuat model yang sudah disimpan untuk penggunaan selanjutnya  
loaded\_model = joblib.load('/mnt/data/artificial\_blood\_vessel\_model.pkl')

# Step 8: Prediksi dengan Data Baru

Setelah model disimpan, Anda dapat menggunakannya untuk melakukan prediksi dengan data pasien baru. Data baru harus diproses dengan cara yang sama seperti data pelatihan, termasuk normalisasi.

# Contoh data pasien baru (misalnya, seorang pasien baru)  
# Misalkan data baru memiliki fitur seperti ini: [Umur, Gender, Jenis Pembuluh Darah, Diameter, Panjang, Indeks Elastisitas, Tekanan Darah, Persentase Penyumbatan, Riwayat Operasi, Skor Kompatibilitas]  
new\_data = np.array([[69, 1, 0, 5.54, 175, 1.85, 106, 32.3, 0, 98.98]]) # Data baru pasien  
  
# Normalisasi data baru menggunakan scaler yang sama  
new\_data = scaler.transform(new\_data)  
  
# Melakukan prediksi apakah pembuluh darah tersebut terpilih atau tidak  
prediction = loaded\_model.predict(new\_data)  
print(f'Prediksi: {"Terpilih" if prediction[0] == 1 else "Tidak Terpilih"}')