```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
# Gunakan row ke-1 (index 0) sebagai header
url = "/content/drive/MyDrive/PRAK KECERDASAN/DATA SET/diabetes.csv"
cols = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
        'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
df = pd.read_csv(url, names=cols, skiprows=1)  # Lewati baris pertama (yang berisi nama kolom)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
data = pd.read_csv("/content/drive/MyDrive/PRAK KECERDASAN/DATA SET/diabetes.csv")
print("Dataset berhasil dimuat. Dimensi dataset:", data.shape)
print(data.head())
```

```
Dataset berhasil dimuat. Dimensi dataset: (768, 9)
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

PREPROCESING DATA

```
# Pisahkan fitur dan label
X = data.drop('Outcome', axis=1)
y = data['Outcome']

# Normalisasi fitur
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split ke data latih dan uji
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

```

BANGUN MODEL ANN

```
model = Sequential([
    Dense(16, input_shape=(8,), activation='relu'),
    Dense(8, activation='relu'),
    Dense(1, activation='sigmoid')  # binary classification
```

```
])

```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

COMPILE DAN LATIH MODEL

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

model.fit(X_train, y_train, epochs=100, verbose=1)
```

```
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8052 - loss: 0.3971
Epoch 73/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8190 - loss: 0.3843
Epoch 74/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8484 - loss: 0.3617
Epoch 75/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8128 - loss: 0.3972
Epoch 76/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8478 - loss: 0.3489
Epoch 77/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8213 - loss: 0.3824
Epoch 78/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8358 - loss: 0.3656
Epoch 79/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8134 - loss: 0.3812
Epoch 80/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8142 - loss: 0.3930
Epoch 81/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8390 - loss: 0.3616
Epoch 82/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8368 - loss: 0.3769
Epoch 83/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8427 - loss: 0.3521
Epoch 84/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8094 - loss: 0.3984
Epoch 85/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8292 - loss: 0.3681
Epoch 86/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8158 - loss: 0.3957
Epoch 87/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8506 - loss: 0.3531
Epoch 88/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8434 - loss: 0.3584
Epoch 89/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8242 - loss: 0.3825
Epoch 90/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8283 - loss: 0.3923
Epoch 91/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8410 - loss: 0.3651
Epoch 92/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8291 - loss: 0.3608
Epoch 93/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8277 - loss: 0.3613
Epoch 94/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8084 - loss: 0.3845
Epoch 95/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8269 - loss: 0.3630
Epoch 96/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8323 - loss: 0.3596
Epoch 97/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8238 - loss: 0.3695
Epoch 98/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8152 - loss: 0.4021
Epoch 99/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8331 - loss: 0.3766
Epoch 100/100
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7945 - loss: 0.4133
<keras.src.callbacks.history.History at 0x7808ad6141d0>
```

EVALUASI AKURASI MODEL

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Akurasi model: {accuracy:.2f}")
```

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7249 - loss: 0.5581
Akurasi model: 0.73
```

PREDIKSI PASIEN

```
# Contoh data pasien: [6, 148, 72, 35, 0, 33.6, 0.627, 50]
sample = np.array([[6, 148, 72, 35, 0, 33.6, 0.627, 50]])
sample_scaled = scaler.transform(sample)

prediction = model.predict(sample_scaled)
hasil = "Diabetes" if prediction[0][0] > 0.5 else "Tidak Diabetes"
print("Prediksi:", hasil)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step
Prediksi: Diabetes
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Star
  warnings.warn(
```

PREDIKSI DIABETES DENGAN FITUR INPUT OUTPUT

```
7# Input dari user
print("Masukkan data pasien untuk prediksi diabetes:")
print("Gunakan angka sesuai petunjuk di bawah ini!\n")

# Daftar fitur dan jenis nilainya
features = [
    ('Pregnancies', 'Jumlah kehamilan (int, 0-20)', int),
    ('Glucose', 'Kadar glukosa (int, 0-200)', int),
    ('BloodPressure', 'Tekanan darah (int, 0-140)', int),
    ('SkinThickness', 'Ketebalan kulit (int, 0-100)', int),
    ('Insulin', 'Tingkat insulin (int, 0-900)', int),
    ('BMI', 'Indeks massa tubuh (float, 0.0 - 70.0)', float),
    ('DiabetesPedigreeFunction', 'Faktor keturunan diabetes (float, 0.0 - 2.5)', float),
    ('Age', 'Umur (int, 0-120)', int)
]

user_input = []

# Ambil input dengan validasi
for feat_name, desc, typ in features:
    while True:
        try:
            val = typ(input(f"{feat_name} - {desc}: "))
            user_input.append(val)
            break
        except ValueError:
            print(f"⚠️ Masukkan angka yang valid untuk {feat_name}!")

# Ubah jadi array dan skalakan
sample = np.array([user_input])
sample_scaled = scaler.transform(sample)

# Prediksi
prediction = model.predict(sample_scaled)
hasil = "🩺 Diabetes" if prediction[0][0] > 0.5 else "✅ Tidak Diabetes"
print("\nHasil Prediksi:", hasil)

```

```
Masukkan data pasien untuk prediksi diabetes:
Gunakan angka sesuai petunjuk di bawah ini!

Pregnancies - Jumlah kehamilan (int, 0-20): 3
Glucose - Kadar glukosa (int, 0-200): 20
BloodPressure - Tekanan darah (int, 0-140): 100
SkinThickness - Ketebalan kulit (int, 0-100): 10
Insulin - Tingkat insulin (int, 0-900): 500
BMI - Indeks massa tubuh (float, 0.0 - 70.0): 60.0
DiabetesPedigreeFunction - Faktor keturunan diabetes (float, 0.0 - 2.5): 1.2
Age - Umur (int, 0-120): 50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step

Hasil Prediksi: ✅ Tidak Diabetes
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Star
  warnings.warn(
```