Laporan Praktikum Decision Tree

Nama : Surya Dwi Satria
Kelas : C7
NIM : 434231048

1. Import Library

```
import pickle
from pathlib import Path

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, ConfusionMatrixDisplay
from pandas.api.types import is_numeric_dtype
```

2. Baca Data & Tentukan Target kolom

```
df = pd.read_excel("BlaBla.xlsx")  # ganti dengan nama file Anda
target_col = "penyakit"  # ganti dengan nama kolom target Anda

df.columns = df.columns.str.lower()
true_target = [c for c in df.columns if c == target_col][0]

print("Jumlah baris, kolom:", df.shape)
df.head()
```

```
Jumlah baris, kolom: (2308, 15)
```

	kategori	umur_tahun	jenis_kelamin	demam	batuk	sesak_napas	nyeri_kepala	lemas	mual	diare	nyeri_otot	sakit_tenggorokan	ruam_kulit	hilang_penciuman	penyakit
0	1	17	0	1	0	NaN	0	0	0	1	0	0	0	1	0
1	5	70	0	0	0	0.0	0	0	0	1	1	1	0	1	1
2	3	39	0	0	0	0.0	0	1	0	0	0	0	0	1	0
3	5	63	0	0	0	0.0	0	0	0	0	1	0	0	1	0
4	3	40	0	0	0	0.0	0	1	0	0	0	1	0	1	0

```
Jumlah baris, kolom: (2308, 15)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2308 entries, 0 to 2307
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   kategori           2308 non-null   int64
 1   umur_tahun         2307 non-null   float64
 2   jenis_kelamin      2308 non-null   int64
 3   demam              2308 non-null   int64
 4   batuk              2308 non-null   int64
 5   sesak_napas        2307 non-null   float64
 6   nyeri_kepala       2308 non-null   int64
 7   lemas              2308 non-null   int64
 8   mual               2308 non-null   int64
 9   diare              2308 non-null   int64
 10  nyeri_otot         2308 non-null   int64
 11  sakit_tenggorokan  2308 non-null   int64
 12  ruam_kulit         2308 non-null   int64
 13  hilang_penciuman   2308 non-null   int64
 14  penyakit           2308 non-null   int64
dtypes: float64(2), int64(13)
memory usage: 270.6 KB
```

Penjelasan : Ada missing Value di umur_tahun dan sesak napas, sehingga datatype float bukan integer

3. Preprocessing (Imputasi & One-Hot Encoding)

```python
obj_cols = [c for c in df.columns if df[c].dtype == "object" and c != true_target]
for c in obj_cols:
    # strip spasi/placeholder kosong -> NaN, lalu coerce ke angka
    df[c] = (
        df[c].astype(str).str.strip().replace({"": np.nan, "NA": np.nan, "NaN": np.nan, "-": np.nan})
    )
    df[c] = pd.to_numeric(df[c], errors="coerce")

# --- 1) Tentukan kolom numerik vs kategorik (robust untuk Int64 nullable) ---
num_cols = [c for c in df.columns if c != true_target and is_numeric_dtype(df[c])]
cat_cols = [c for c in df.columns if c != true_target and not is_numeric_dtype(df[c])]

# --- 2) Imputasi & casting kolom numerik (biner vs non-biner) ---
def is_binary(series: pd.Series) -> bool:
    u = pd.to_numeric(series.dropna(), errors="coerce").unique()
    if len(u) == 0:
        return False
    try:
        return set(pd.Series(u).dropna().astype(float).round().astype(int).unique()) <= {0, 1}
    except Exception:
        return False

for c in num_cols:
    s = pd.to_numeric(df[c], errors="coerce")
    if is_binary(s):
        fill_val = s.mode().iloc[0] if not s.mode().empty else 0
        df[c] = s.fillna(fill_val).astype("int64")            # biner -> int64
    else:
        med = s.median()
        df[c] = s.fillna(med)
        # jika semuanya bilangan bulat setelah imputasi, rapikan ke int64
        if np.isclose(df[c] % 1, 0).all():
            df[c] = df[c].astype("int64")

# --- 3) Imputasi kategorikal (jika ada) ---
for c in cat_cols:
    if df[c].isna().any():
        df[c] = df[c].fillna(df[c].mode().iloc[0])

# --- 4) Rebuild daftar kolom (setelah cleaning) & buat ColumnTransformer ---
num_cols = [c for c in df.columns if c != true_target and is_numeric_dtype(df[c])]
cat_cols = [c for c in df.columns if c != true_target and not is_numeric_dtype(df[c])]

numeric_pipe = Pipeline([("imputer", SimpleImputer(strategy="median"))])  # safety net
categorical_pipe = Pipeline([
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("onehot", OneHotEncoder(handle_unknown="ignore", sparse_output=False)),
])

preprocess = ColumnTransformer(
    transformers=[
        ("num", numeric_pipe, num_cols),
        ("cat", categorical_pipe, cat_cols),
    ],
    remainder="drop",
    verbose_feature_names_out=False,
)

print("Kolom numerik  :", num_cols)
print("Kolom kategorik:", cat_cols)
```

```
Kolom numerik  : ['kategori', 'umur_tahun', 'jenis_kelamin', 'demam', 'batuk', 'sesak_napas', 'nyeri_kepala', 'lemas', 'mual', 'diare', 'nyeri_otot', 'sakit_tenggorokan', 'ruam_kulit', 'hilang_penciuman']
Kolom kategorik: []
```

4. Pisahkan Fitur (X) dan Target (y), lalu Split Train/Test

```
TEST_SIZE = 0.2  # 20% untuk data uji dan 80% untuk data latih

X = df.drop(columns=[true_target]) # fitur yaitu semua kolom kecuali target kolom
y = df[true_target] # target yaitu kolom target yang akan digunakan sebagai label

# Stratify hanya jika kelas > 1 agar tidak error di dataset 1 label
strat = y if y.nunique() > 1 else None #

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=TEST_SIZE, stratify=strat
) # stratify agar proporsi kelas di train & test sama (jika klasifikasi)

print("X_train:", X_train.shape, "X_test:", X_test.shape) # Menampilkan jumlah baris dan kolom pada data latih dan data uji
print("y_train:", y_train.shape, "y_test:", y_test.shape) # Menampilkan jumlah baris pada label data latih dan data uji
```

```
X_train: (1846, 14) X_test: (462, 14)
y_train: (1846,) y_test: (462,)
```

Penjelasan :

`X` adalah semua kolom kecuali target

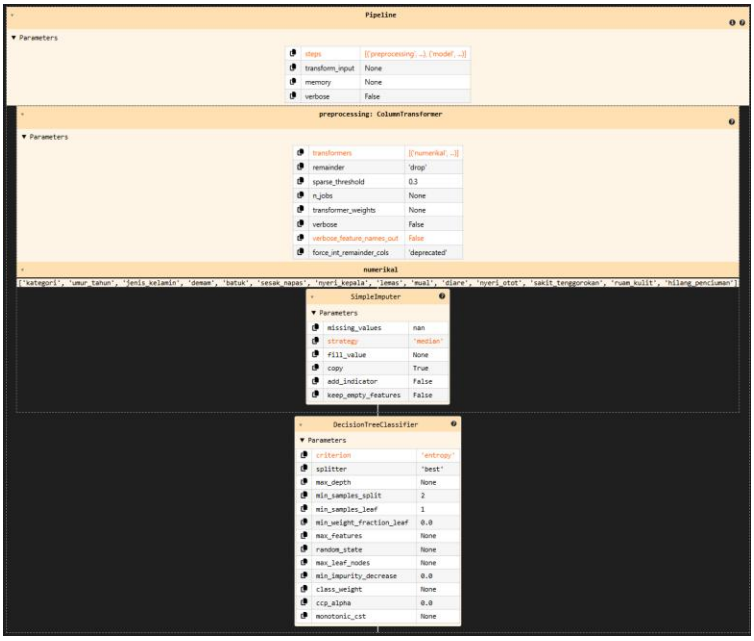
`y` adalah kolom target (label)

`train_test_split` membagi data jadi 80% train dan 20% test (bisa diubah)

`stratify=y` menjaga proporsi kelas tetap seimbang antara train dan test (jika label punya >1 kelas)

5. Bangun Model Decision Tree dengan Pipeline

```
# Hyperparameter sederhana (bisa kamu ubah & coba-coba):
CRITERION = "gini"     # opsi: "gini", "entropy", "log_loss"
MAX_DEPTH = None       # None = kedalaman bebas. Ubah ke angka (mis. 5) untuk membatasi kompleksitas
MIN_SAMPLES_SPLIT = 2  # minimum sampel untuk memecah node

model = DecisionTreeClassifier(
    criterion=CRITERION,
    max_depth=MAX_DEPTH,
    min_samples_split=MIN_SAMPLES_SPLIT,

)

clf = Pipeline([
    ("prep", preprocess),
    ("model", model),
])

clf
```



Penjelasan : **Pipeline**

- Berisi 2 langkah: preprocessing (prep data) dan model (Decision Tree).

preprocessing: ColumnTransformer

- Semua kolom fitur dianggap **numerik**.
- Missing value diisi dengan **median** (SimpleImputer(strategy='median')).

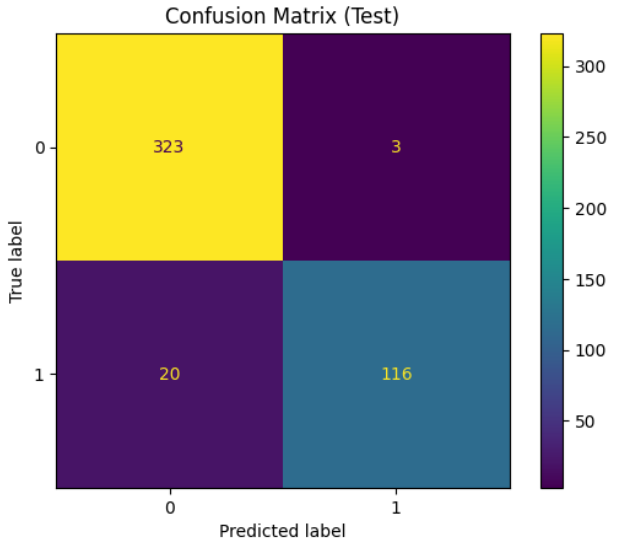
DecisionTreeClassifier

- criterion='entropy' → pohon pakai **Information Gain** untuk memilih split.
- max_depth=None → pohon bisa tumbuh tanpa batas (berpotensi overfitting).
- min_samples_split=2, min_samples_leaf=1 → aturan default pemisahan node.
- Parameter lain tetap default.

6. Latih dan Evaluasi Model

```python
# Latih model
clf.fit(X_train, y_train)

# Prediksi di test
y_pred = clf.predict(X_test)

# Metrik
acc = accuracy_score(y_test, y_pred)
print(f"Accuracy (test): {acc:.3f}\n")

print("Classification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix (plot)
fig = plt.figure()
ConfusionMatrixDisplay.from_estimator(clf, X_test, y_test)
plt.title("Confusion Matrix (Test)")
plt.show()
```

```
Accuracy (test): 0.950

Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.99      0.97       326
           1       0.97      0.85      0.91       136

    accuracy                           0.95       462
   macro avg       0.96      0.92      0.94       462
weighted avg       0.95      0.95      0.95       462
```



Penjelasan :

1. **Accuracy = 0.95 (95%)**
   Artinya 95% prediksi di data uji benar.
2. **Classification Report**
   - **Kelas 0** → precision 0.94, recall 0.99 (model hampir selalu benar saat mendeteksi kelas 0).
   - **Kelas 1** → precision 0.97, recall 0.86 (model sangat tepat mendeteksi kelas 1, tapi kadang ada yang miss → recall lebih rendah).
   - Macro avg & weighted avg ≈ 0.95 → performa seimbang antar kelas.
3. **Confusion Matrix**
   - 323 benar terklasifikasi sebagai 0.
   - 116 benar terklasifikasi sebagai 1.
   - 3 salah klasifikasi (0 jadi 1).
   - 20 salah klasifikasi (1 jadi 0).

Model Decision Tree dengan entropy bekerja dengan baik, punya akurasi tinggi, dan relatif seimbang performanya di kedua kelas, meski sedikit lebih sering meleset saat mendeteksi kelas

7. Ekspor dataset_test.csv dan model.pkl

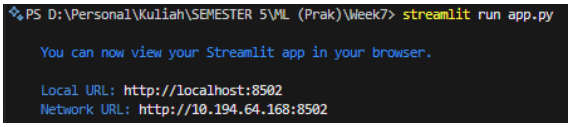
```python
# Simpan dataset_test.csv
test_df = X_test.copy()
test_df[true_target] = y_test.values
test_csv_path = Path("dataset_test.csv")
test_df.to_csv(test_csv_path, index=False)

# Simpan model.pkl
model_path = Path("model.pkl")
with open(model_path, "wb") as f:
    pickle.dump(clf, f)

print("Tersimpan:", test_csv_path.resolve())
print("Tersimpan:", model_path.resolve())
```

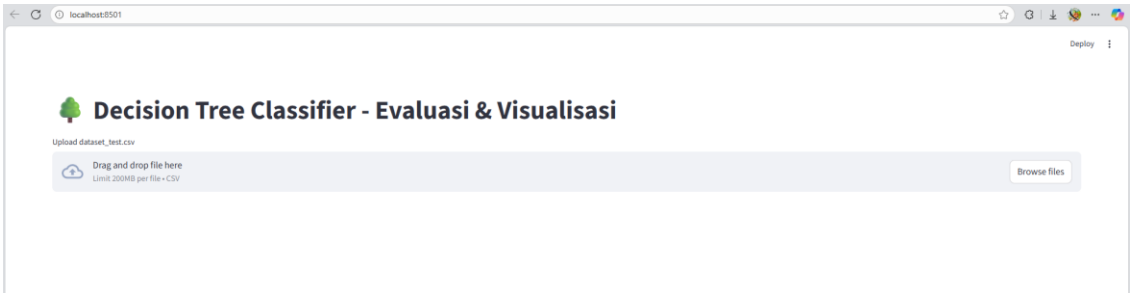
```
Tersimpan: D:\Personal\Kuliah\SEMESTER 5\ML (Prak)\Week7\dataset_test.csv
Tersimpan: D:\Personal\Kuliah\SEMESTER 5\ML (Prak)\Week7\model.pkl
```

8. Buat file app.py untuk streamlit

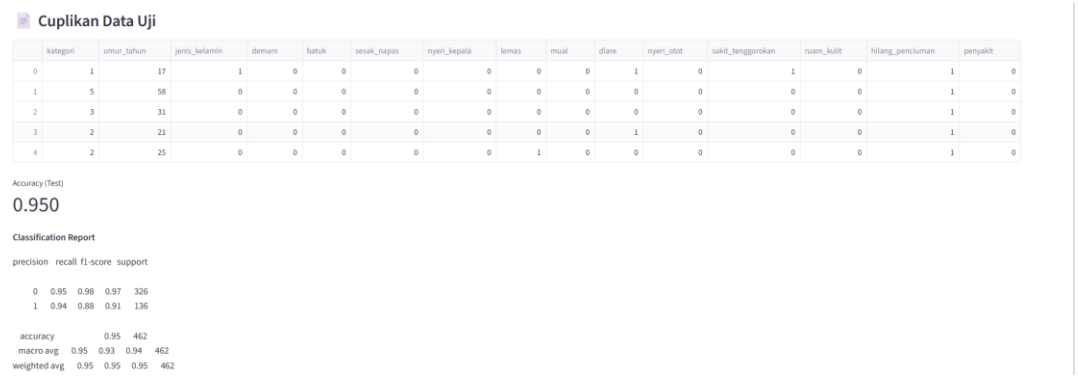
```python
import streamlit as st
import pandas as pd
import pickle
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, classification_report, ConfusionMatrixDisplay
from sklearn import tree

# ----------------------------
# Konfigurasi awal Streamlit
# ----------------------------
st.set_page_config(page_title="Decision Tree Classifier", layout="wide")
st.title("🌳 Decision Tree Classifier - Evaluasi & Visualisasi")

# ----------------------------
# Load model
# ----------------------------
with open("model.pkl", "rb") as f:
    clf = pickle.load(f)

# ----------------------------
# Upload dataset test
# ----------------------------
uploaded = st.file_uploader("Upload dataset_test.csv", type=["csv"])
if uploaded is not None:
    df_test = pd.read_csv(uploaded)

    st.subheader("📄 Cuplikan Data Uji")
    st.dataframe(df_test.head())

    # Kolom target (ubah sesuai datasetmu)
    target_col = "penyakit"
    if target_col not in df_test.columns:
        st.error(f"Kolom target '{target_col}' tidak ada di dataset!")
        st.stop()

    X_test = df_test.drop(columns=[target_col])
    y_test = df_test[target_col]

    # ----------------------------
    # Prediksi & Evaluasi
    # ----------------------------
    y_pred = clf.predict(X_test)
    acc = accuracy_score(y_test, y_pred)

    st.metric("Accuracy (Test)", f"{acc:.3f}")

    st.write("**Classification Report**")
    st.text(classification_report(y_test, y_pred))

    st.write("**Confusion Matrix**")
    fig, ax = plt.subplots()
    ConfusionMatrixDisplay.from_estimator(clf, X_test, y_test, ax=ax)
    st.pyplot(fig)

    # ----------------------------
    # Visualisasi Pohon
    # ----------------------------
    st.subheader("🌳 Visualisasi Pohon Keputusan")

    max_depth_vis = st.slider("Pilih kedalaman visualisasi pohon", 1, 10, 3)

    fig, ax = plt.subplots(figsize=(20, 10))
    tree.plot_tree(
        clf.named_steps["model"],
        filled=True,
        feature_names=X_test.columns,
        class_names=[str(c) for c in sorted(y_test.unique())],
        fontsize=10,
        max_depth=max_depth_vis  # kendali kedalaman visualisasi
    )
    st.pyplot(fig)
```

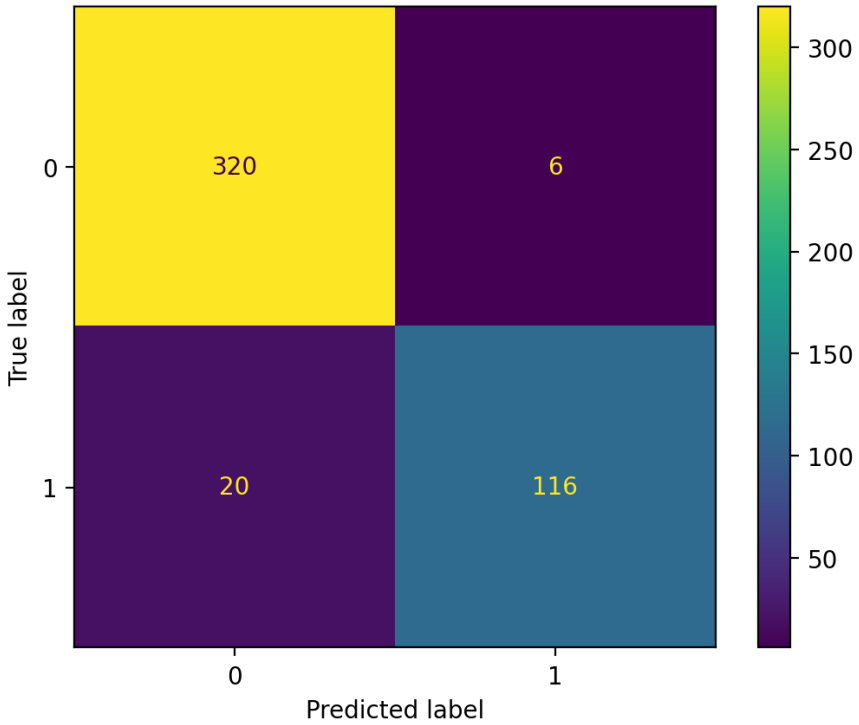
9. Run Streamlit dan akan di arahkan ke halaman browser streamlit

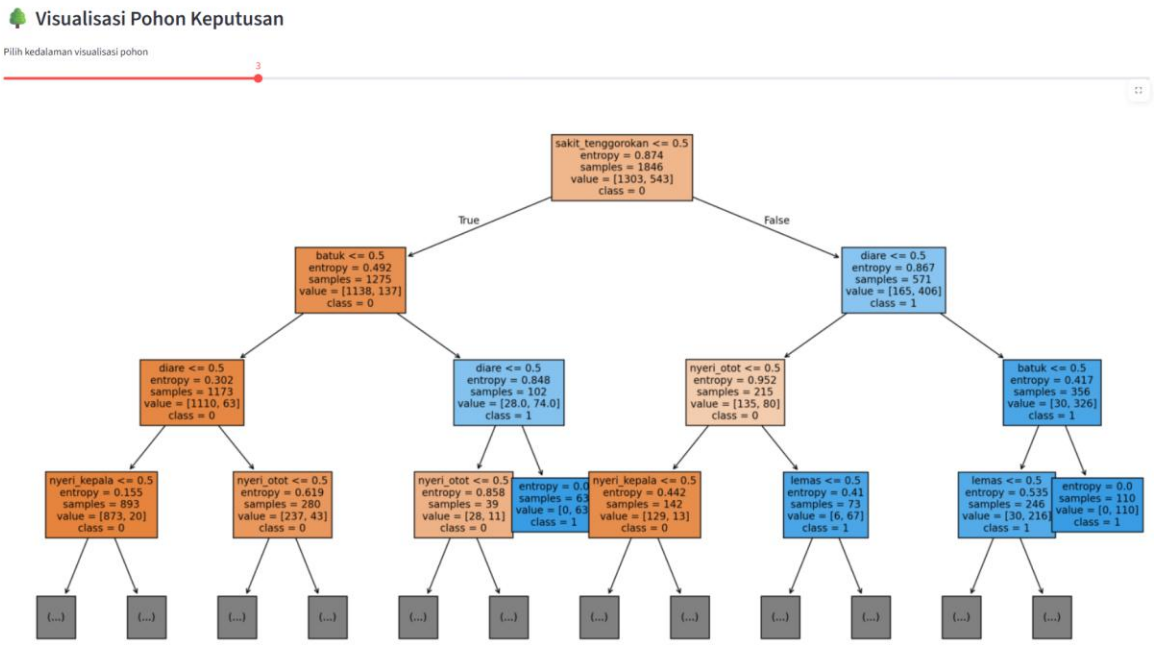




10. Masukkan dataset_test.csv







Penjelasan :

- **Confusion Matrix**: model benar memprediksi 320 data kelas 0 dan 116 data kelas 1, salah 6 kali (0 → 1) dan 20 kali (1 → 0). Artinya akurasi tinggi, tapi kelas 1 kadang terlewat.
- **Pohon Keputusan**: tiap node menunjukkan aturan split (misal sakit_tenggorokan <= 0.5). Warna oranye = dominan kelas 0, biru = dominan kelas 1. Semakin pekat warnanya, node makin murni.

Kesimpulan: model sudah baik, pohon menjelaskan logika keputusan berdasarkan gejala.