Lab 3 Decision Tree

No.	Program
1	`# import tools` `from matplotlib import pyplot as plt` `import numpy as np` `import pandas as pd` `from sklearn.model_selection import GridSearchCV` `from sklearn.tree import DecisionTreeClassifier, plot_tree` `from sklearn.metrics import accuracy_score`
2	`# import data train` `data_train = pd.read_csv('adult_train.csv')` `data_train.tail()`
3	`# import data test` `data_test = pd.read_csv('adult_test.csv')` `data_test.tail()`
4	`# hapus data yang labelnya salah ketik` `data_test = data_test[(data_test['Target'] == ' >50K.') |` `(data_test['Target']==' <=50K.')]` `# encode target menjadi integer 0 dan 1` `data_train.loc[data_train['Target']==' <=50K', 'Target'] = 0` `data_train.loc[data_train['Target']==' >50K', 'Target'] = 1` `data_test.loc[data_test['Target']==' <=50K.', 'Target'] = 0` `data_test.loc[data_test['Target']==' >50K.', 'Target'] = 1`
5	`# cek data apakah ada yang aneh` `# data_test.describe(include='all').T` `data_train.describe(include='all').T`
6	`# hitung jumlah data target` `data_train['Target'].value_counts()`
7	`# plot untuk melihat data` `fig = plt.figure(figsize=(25, 15))` `cols = 5` `rows = np.ceil(float(data_train.shape[1]) / cols)` `for i, column in enumerate(data_train.columns):` `ax = fig.add_subplot(rows, cols, i + 1)` `ax.set_title(column)` `if data_train.dtypes[column] == np.object:`

	`` data_train[column].value_counts().plot(kind="bar", axes=ax) else: data_train[column].hist(axes=ax) plt.xticks(rotation="vertical") plt.subplots_adjust(hspace=0.7, wspace=0.2) ```
8	``` # cek tipe data train data_train.dtypes ```
9	``` # cek tipe data test data_test.dtypes ```
10	``` # ubah tipe data Age dari object menjadi int data_test['Age'] = data_test['Age'].astype(int) ```
11	``` # konversi tipe data float ke int supaya perhitungan mudah data_test['fnlwgt'] = data_test['fnlwgt'].astype(int) data_test['Education_Num'] = data_test['Education_Num'].astype(int) data_test['Capital_Gain'] = data_test['Capital_Gain'].astype(int) data_test['Capital_Loss'] = data_test['Capital_Loss'].astype(int) data_test['Hours_per_week'] = data_test['Hours_per_week'].astype(int) ```
12	``` # cek missing data dan tipe datanya data_train.info() ```
13	``` # pilih fitur categorical and continuous dari data categorical_columns = [c for c in data_train.columns if data_train[c].dtype.name == 'object'] numerical_columns = [c for c in data_train.columns if data_train[c].dtype.name != 'object'] print('categorical_columns:', categorical_columns) print('numerical_columns:', numerical_columns) ```
14	``` # impute/isi data kosong, kategorikal dengan mode, numerical dengan median for c in categorical_columns: data_train[c].fillna(data_train[c].mode()[0], inplace=True) data_test[c].fillna(data_train[c].mode()[0], inplace=True) for c in numerical_columns: data_train[c].fillna(data_train[c].median(), inplace=True) data_test[c].fillna(data_train[c].median(), inplace=True) ```

15	`# semua missing values sudah terisi` `data_train.info()`
16	`# one hot encoding untuk split fitur` `data_train = pd.concat([data_train[numerical_columns],` `    pd.get_dummies(data_train[categorical_columns])], axis=1)` `data_test = pd.concat([data_test[numerical_columns],` `    pd.get_dummies(data_test[categorical_columns])], axis=1)`
17	`# fitur/kolom jadi sangat banyak` `data_train.columns, data_test.columns`
18	`# cek apakah berbeda antara train dan test` `set(data_train.columns) - set(data_test.columns)`
19	`data_train.shape, data_test.shape`
20	`# bikin kolom holand dengan value 0` `data_test['Country_ Holand-Netherlands'] = 0`
21	`set(data_train.columns) - set(data_test.columns)`
22	`data_train.head(2)`
23	`# pisahkan X dan y` `X_train = data_train.drop(['Target'], axis=1)` `y_train = data_train['Target']` `X_test = data_test.drop(['Target'], axis=1)` `y_test = data_test['Target']`
24	`# decision tree biasa` `tree = DecisionTreeClassifier(max_depth=3, criterion='entropy' , random_state=17)` `tree.fit(X_train, y_train)`
25	`# prediksi` `tree_predictions = tree.predict(X_test)`
26	`# cek akurasi` `accuracy_score(y_test, tree_predictions)`

27	`# dengan parameter tuning( cari max depth dan menggunakan 5-fold cross validation)` `tree_params = {'max_depth': range(2, 11)}` `tree_terbaik = GridSearchCV(DecisionTreeClassifier(random_state=17,` `criterion='entropy'), tree_params, cv=5)` `tree_terbaik.fit(X_train, y_train)`
2	`# cek parameter dan cross validatio terbaik` `print("parameter terbaik:", tree_terbaik.best_params_)` `print("cross validaton score terbaik", tree_terbaik.best_score_)`
29	`# bikin tree hasil tuning` `tuned_tree = DecisionTreeClassifier(max_depth=8, criterion='entropy', random_state=17)` `tuned_tree.fit(X_train, y_train)` `prediksi_tuned_tree = tuned_tree.predict(X_test)` `accuracy_score(y_test, prediksi_tuned_tree)`
30	`# plot decision tree` `plot_tree(tuned_tree, feature_names=X_train.columns, filled=True,` `class_names=[">50k","<=50k"]);`

Feature descriptions:

- **Age** – continuous feature
- **Workclass** – continuous feature
- **fnlwgt** – final weight of object, continuous feature
- **Education** – categorical feature
- **Education_Num** – number of years of education, continuous feature
- **Martial_Status** – categorical feature
- **Occupation** – categorical feature
- **Relationship** – categorical feature
- **Race** – categorical feature
- **Sex** – categorical feature
- **Capital_Gain** – continuous feature
- **Capital_Loss** – continuous feature
- **Hours_per_week** – continuous feature
- **Country** – categorical feature

Target – earnings level