Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 1
Pembahasan : Decision Tree (C.45)
Pokok Pemb : - Membangun Model Decision Tree
- Simulasi Algoritma Decision Tree
- Evaluasi Algoritma Decision Tree
- Aplikasi menggunakan algoritma Decision Tree

1. Membangun Model Decision Tree

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
```

```
df = pd.read_csv('heart.csv')
```

```
df.head()
```

	Age	Cholesterol Level	Blood Pressure	Smoking	Physical Activity	BMI	Heart Disease Risk
0	55	High	High	Yes	Low	Overweight	High
1	43	Normal	Normal	No	Moderate	Normal	Low
2	60	High	High	Yes	Low	Obese	High
3	35	Normal	Low	No	High	Normal	Low
4	50	High	High	No	Low	Overweight	Medium

```python
# 1. Histogram of Age
plt.figure(figsize=(10, 6))
plt.hist(df['Age'], bins=10, edgecolor='black')
plt.title("Distribution of Age")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```

```python
# 2. Bar plot of Cholesterol Level Counts
plt.figure(figsize=(10, 6))
df['Cholesterol Level'].value_counts().plot(kind='bar')
plt.title("Count of Cholesterol Levels")
plt.xlabel("Cholesterol Level")
plt.ylabel("Count")
plt.show()
```

```python
# 3. Bar plot of Blood Pressure Counts
plt.figure(figsize=(10, 6))
df['Blood Pressure'].value_counts().plot(kind='bar', color='orange')
plt.title("Count of Blood Pressure Levels")
plt.xlabel("Blood Pressure Level")
plt.ylabel("Count")
plt.show()
```

```python
# 4. Pie chart of Smoking Status
plt.figure(figsize=(8, 8))
df['Smoking'].value_counts().plot(kind='pie', autopct='%1.1f%%', startangle=140)
plt.title("Proportion of Smoking Status")
plt.ylabel("")  # Removes 'Smoking' label from pie
plt.show()
```

```python
# 5. Count plot of BMI categories
plt.figure(figsize=(10, 6))
df['BMI'].value_counts().plot(kind='bar', color='purple')
plt.title("Count of BMI Categories")
plt.xlabel("BMI Category")
plt.ylabel("Count")
plt.show()
```

```python
# 6. Violin Plot of Age by Cholesterol Level
plt.figure(figsize=(10, 6))
sns.violinplot(x='Cholesterol Level', y='Age', data=df)
plt.title("Violin Plot of Age by Cholesterol Level")
plt.xlabel("Cholesterol Level")
plt.ylabel("Age")
plt.show()
```

```python
# 7. Swarm Plot of Age by BMI Category
plt.figure(figsize=(10, 6))
sns.swarmplot(x='BMI', y='Age', data=df)
plt.title("Swarm Plot of Age by BMI Category")
plt.xlabel("BMI Category")
plt.ylabel("Age")
plt.show()
```

```python
# 8. Box Plot of Age by Blood Pressure
plt.figure(figsize=(10, 6))
sns.boxplot(x='Blood Pressure', y='Age', data=df)
plt.title("Box Plot of Age by Blood Pressure Level")
plt.xlabel("Blood Pressure Level")
plt.ylabel("Age")
plt.show()
```

```python
# 9. Scatter Plot of Age vs Physical Activity with BMI Category as Hue
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Age', y='Physical Activity', hue='BMI')
plt.title("Scatter Plot of Age vs Physical Activity by BMI Category")
plt.xlabel("Age")
plt.ylabel("Physical Activity")
plt.legend(title="BMI Category")
plt.show()
```

```python
# membuat fungsi untuk mengkategorikan usia
def categorize_age(age):
    if age < 35:
        return "Young"
    elif age < 55:
        return "Middle-aged"
    else:
        return "Senior"
```

```python
# terapkan fungsi pada kolom 'Age (Usia)' dan buat kolom baru 'Age Category'
df['Age'] = df['Age'].apply(categorize_age)
```

```
df.head()
```

	Age	Cholesterol Level	Blood Pressure	Smoking	Physical Activity	BMI	Heart Disease Risk
0	Senior	High	High	Yes	Low	Overweight	High
1	Middle-aged	Normal	Normal	No	Moderate	Normal	Low
2	Senior	High	High	Yes	Low	Obese	High
3	Middle-aged	Normal	Low	No	High	Normal	Low
4	Middle-aged	High	High	No	Low	Overweight	Medium

```
# Encode semua variabel kategori
label_encoders = {}
for column in df.columns:
    if df[column].dtype == 'object':
        le = LabelEncoder()
        df[column] = le.fit_transform(df[column])
        label_encoders[column] = le
```

```
df.head()
```

	Age	Cholesterol Level	Blood Pressure	Smoking	Physical Activity	BMI	Heart Disease Risk
0	1	0	0	1	1	2	0
1	0	1	2	0	2	0	1
2	1	0	0	1	1	1	0
3	0	1	1	0	0	0	1
4	0	0	0	0	1	2	2

```python
# Split the dataset
X = df.drop('Heart Disease Risk', axis=1)
y = df['Heart Disease Risk']
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```python
# Train the C4.5 model (Decision Tree)
model = DecisionTreeClassifier(criterion='entropy')
model.fit(X_train, y_train)
```

```python
# Make predictions
y_pred = model.predict(X_test)
```

2. Simulasi Algoritma Decision Tree

```
# Simulate new input data
new_data = pd.DataFrame({
    'Age': [label_encoders['Age'].transform(['Middle-aged'])[0]],
    'Cholesterol Level': [label_encoders['Cholesterol Level'].transform(['High'])[0]],
    'Blood Pressure': [label_encoders['Blood Pressure'].transform(['High'])[0]],
    'Smoking': [label_encoders['Smoking'].transform(['Yes'])[0]],
    'Physical Activity': [label_encoders['Physical Activity'].transform(['Low'])[0]],
    'BMI': [label_encoders['BMI'].transform(['Overweight'])[0]]
})
```

```
new_prediction = model.predict(new_data)
```

```
# Decode the new prediction
new_prediction_decoded = label_encoders['Heart Disease Risk'].inverse_transform(new_prediction)
```

```
new_prediction_decoded
```

3. Evaluasi Algoritma Decision Tree

```
# Calculate accuracy and confusion matrix
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
```

```
accuracy, conf_matrix
```

4. Plot Hasil Decision Tree

```python
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

plt.figure(figsize=(20, 20))
plot_tree(model, feature_names=['Age',
                                'Cholesterol Level',
                                'Blood Pressure',
                                'Smoking',
                                'Physical Activity',
                                'BMI'], class_names=['High Risk',
                                                     'Low Risk',
                                                     'Medium Risk'],
                                                     filled=True, rounded=True)
plt.title("pohon keputusan")
plt.show()
```

5. Aplikasi menggunakan algoritma Decision Tree

```python
import streamlit as st
import pickle
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

# Load the trained model from the .pkl file
model_path = 'c45_pinjam_mod.pkl'
with open(model_path, 'rb') as model_file:
    loaded_model = pickle.load(model_file)

st.title('C4.5 Decision Tree Model for Heart Disease Prediction')

# Input for new data
st.sidebar.header('Input Features')
age = st.sidebar.selectbox('Age', ('Senior', 'Middle-aged', 'Young'))
cholesterol = st.sidebar.selectbox('Cholesterol Level', ('High', 'Normal', 'Low'))
blood_pressure = st.sidebar.selectbox('Blood Pressure', ('High', 'Normal', 'Low'))
smoking = st.sidebar.selectbox('Smoking', ('Yes', 'No'))
physical_activity = st.sidebar.selectbox('Physical Activity', ('Low', 'Moderate', 'High'))
bmi = st.sidebar.selectbox('BMI', ('Overweight', 'Normal', 'Obese'))

# Create a mapping for the labels to numbers (as per how the model was trained)
label_encodings = {
    'Age': {'Senior': 1, 'Middle-aged': 0, 'Young': 2},
    'Cholesterol Level': {'High': 0, 'Normal': 1, 'Low': 2},
    'Blood Pressure': {'High': 0, 'Normal': 1, 'Low': 2},
    'Smoking': {'Yes': 1, 'No': 0},
    'Physical Activity': {'Low': 0, 'Moderate': 1, 'High': 2},
    'BMI': {'Overweight': 1, 'Normal': 0, 'Obese': 2}
}

# Encode the input data
new_data = np.array([[label_encodings['Age'][age],
                      label_encodings['Cholesterol Level'][cholesterol],
                      label_encodings['Blood Pressure'][blood_pressure],
                      label_encodings['Smoking'][smoking],
                      label_encodings['Physical Activity'][physical_activity],
                      label_encodings['BMI'][bmi]]])

# Predict using the loaded model
if st.button('Predict'):
    prediction = loaded_model.predict(new_data)
    risk_mapping = {0: 'High Risk', 1: 'Low Risk', 2: 'Medium Risk'}
    st.write(f'The predicted heart disease risk is: **{risk_mapping[prediction[0]]}**')

    # Plot the decision tree
    st.write("### Decision Tree Visualization")
    fig = plt.figure(figsize=(20, 10))
    plot_tree(loaded_model, feature_names=['Age',
                                           'Cholesterol Level',
                                           'Blood Pressure',
                                           'Smoking',
                                           'Physical Activity',
                                           'BMI'],
                                           class_names=['High Risk',
                                                        'Low Risk',
                                                        'Medium Risk'],
                                                        filled=True, rounded=True)
    st.pyplot(fig)
```