LAPORAN PRAKTIKUM
MACHINE LEARNING
PERTEMUAN 6

Disusun oleh :

Nama : Nauval Badiul Fikri Alhadad HS

NIM : 235410017

Kelas : Informatika

PROGRAM STUDI INFORMATIKA

PROGRAM SARJANA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA

YOGYAKARTA

2024

MODUL PERTEMUAN 6

A. Praktik

1. Import library

```
        import pandas as pd
        import numpy as np
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.model_selection import train_test_split, LeaveOneOut,
        cross_val_score
        from sklearn.preprocessing import LabelEncoder
        from sklearn.metrics import accuracy_score, classification_report,
        confusion_matrix
        import seaborn as sns
import matplotlib.pyplot as plt
```

2. Buat data frame

```
        data = {
          'Cuaca': ['Cerah', 'Cerah', 'Mendung', 'Hujan', 'Hujan',
                'Cerah', 'Mendung', 'Hujan', 'Cerah', 'Mendung'],
          'Suhu':  ['Panas', 'Dingin', 'Panas', 'Dingin', 'Panas',
                'Panas', 'Dingin', 'Dingin', 'Dingin', 'Panas'],
          'Beli':  ['Yes', 'Yes', 'Yes', 'No', 'No',
                'Yes', 'Yes', 'Yes', 'No', 'Yes']
        }
        print("\n" + "="*30)
        df = pd.DataFrame(data)
        print("Dataset:")
        print(df)
print("\n" + "="*30)
```

Output :

```
==============================
Dataset:
     Cuaca    Suhu Beli
0    Cerah   Panas  Yes
1    Cerah  Dingin  Yes
2  Mendung   Panas  Yes
3    Hujan  Dingin   No
4    Hujan   Panas   No
5    Cerah   Panas  Yes
6  Mendung  Dingin  Yes
7    Hujan  Dingin  Yes
8    Cerah  Dingin   No
9  Mendung   Panas  Yes

==============================
```

Penjelasan : Kode tersebut membuat sebuah dataset sederhana menggunakan Python dan pustaka pandas. Data berisi tiga variabel yaitu Cuaca, Suhu, dan Beli, masing-masing

memiliki 10 entri yang merepresentasikan kondisi cuaca, suhu, serta keputusan seseorang untuk membeli Yes / No.

3. Encoding fitur kategorikal

```
        le_cuaca = LabelEncoder()
        le_suhu = LabelEncoder()
        le_beli = LabelEncoder()

        df['Cuaca_enc'] = le_cuaca.fit_transform(df['Cuaca'])
        df['Suhu_enc'] = le_suhu.fit_transform(df['Suhu'])
        df['Beli_enc'] = le_beli.fit_transform(df['Beli'])  # Yes=1, No=0

        # Fitur dan target
        X = df[['Cuaca_enc', 'Suhu_enc']]
y = df['Beli_enc']
```

4. Split data (80% train, 20% test)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

5. Latih Random Forest

```
        rf = RandomForestClassifier(
            n_estimators=100,
            random_state=42,
            max_depth=None
        )
rf.fit(X_train, y_train)
```

6. Prediksi

```
y_pred = rf.predict(X_test)
```

7. Evaluasi

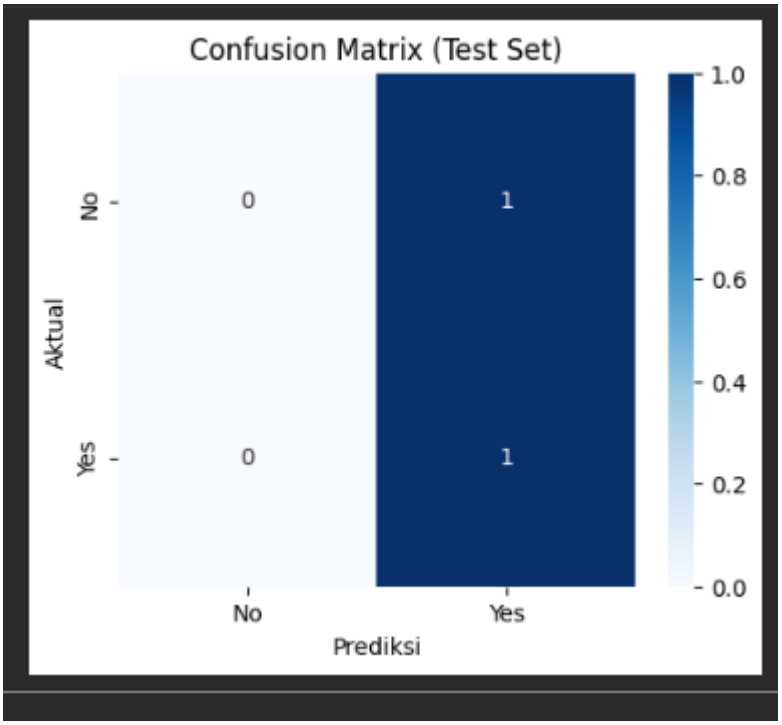
```
        accuracy = accuracy_score(y_test, y_pred)
print(f"\nAkurasi (Test Set): {accuracy:.2f}\n")
```

Output :

```
Akurasi (Test Set): 0.50
```

```
        # Confusion Matrix
        cm = confusion_matrix(y_test, y_pred)
        plt.figure(figsize=(5,4))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['No', 'Yes'], yticklabels=['No', 'Yes'])
        plt.title('Confusion Matrix (Test Set)')
        plt.xlabel('Prediksi')
        plt.ylabel('Aktual')
plt.show()
```

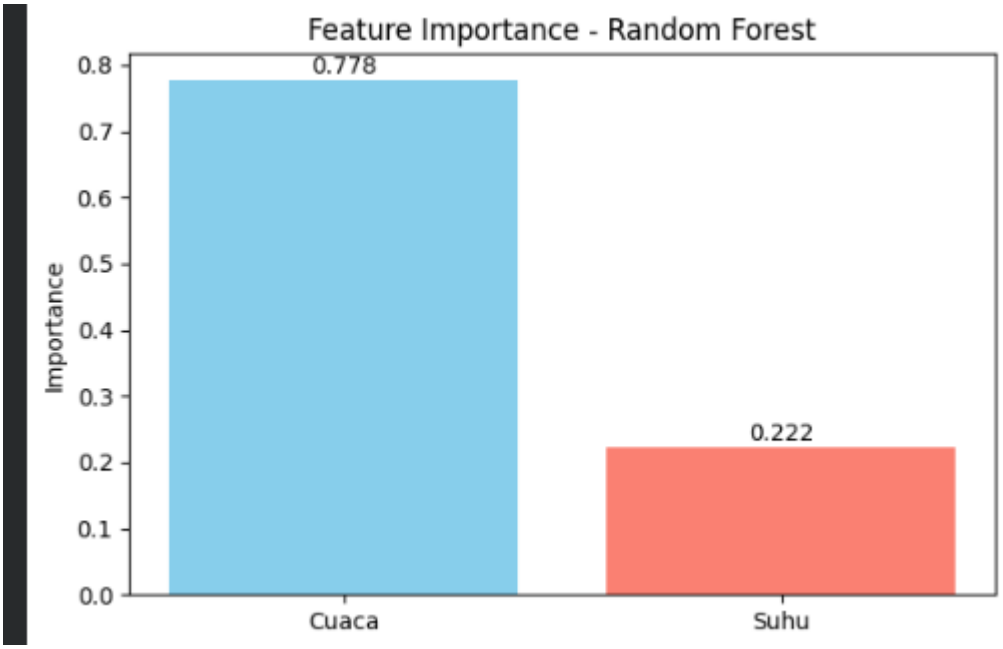
Output :



8. Feature Importance

```
        importances = rf.feature_importances_
        features = ['Cuaca', 'Suhu']

        plt.figure(figsize=(6,4))
        plt.bar(features, importances, color=['skyblue', 'salmon'])
        plt.title('Feature Importance - Random Forest')
        plt.ylabel('Importance')
        for i, v in enumerate(importances):
            plt.text(i, v + 0.01, f"{v:.3f}", ha='center')
        plt.tight_layout()
plt.show()
```

Output :



9. Prediksi data baru

```
# Data baru
data_baru = [
    ['Cerah', 'Panas'],
    ['Hujan', 'Dingin'],
    ['Mendung', 'Dingin'],
    ['Cerah', 'Dingin'],
    ['Hujan', 'Panas']
]

# Konversi ke DataFrame
df_baru = pd.DataFrame(data_baru, columns=['Cuaca', 'Suhu'])

# Encoding
df_baru['Cuaca_enc'] = le_cuaca.transform(df_baru['Cuaca'])
df_baru['Suhu_enc'] = le_suhu.transform(df_baru['Suhu'])

X_baru = df_baru[['Cuaca_enc', 'Suhu_enc']]

# Prediksi
prediksi = rf.predict(X_baru)
probabilitas = rf.predict_proba(X_baru)

# Konversi ke nama kelas
prediksi_nama = le_beli.inverse_transform(prediksi)

# Tampilkan hasil
print("PREDIKSI DATA BARU")
print(f"{'No':<3} {'Cuaca':<8} {'Suhu':<7} {'Prediksi':<8} {'Prob(No)':<8}
{'Prob(Yes)':<8}")
print("-" * 55)
```

```
for i in range(len(df_baru)):
    no = i + 1
    cuaca = df_baru['Cuaca'][i]
    suhu = df_baru['Suhu'][i]
    pred = prediksi_nama[i]
    prob_no = probabilitas[i][0]
    prob_yes = probabilitas[i][1]
    print(f"{no:<3} {cuaca:<8} {suhu:<7} {pred:<8} {prob_no:<8.3f}
{prob_yes:.3f}")
```

Output :

```
PREDIKSI DATA BARU
No  Cuaca    Suhu    Prediksi Prob(No) Prob(Yes)
-----------------------------------------------------
1   Cerah    Panas   Yes      0.020    0.980
2   Hujan    Dingin  Yes      0.463    0.537
3   Mendung  Dingin  Yes      0.314    0.686
4   Cerah    Dingin  Yes      0.046    0.954
5   Hujan    Panas   No       0.731    0.269
```

Praktik 2 **Klasifikasi untuk dataset Dataset: "Adult Income" (UCI)**

1. Import library

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

2. Load dataset

```
url = "https://archive.ics.uci.edu/ml/machine-learning-
databases/adult/adult.data"
columns = [
    'age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-
status',
    'occupation', 'relationship', 'race', 'sex', 'capital-gain', 'capital-loss',
    'hours-per-week', 'native-country', 'income'
]
```

```
        df = pd.read_csv(url, names=columns, na_values=' ?',
        skipinitialspace=True)
        print(f"Dataset loaded: {df.shape[0]} baris, {df.shape[1]} kolom")
df.head()
```

Output :

Dataset loaded: 32561 baris, 15 kolom

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba

3. Eksplorasi & preprocessing

```
        print(df.info())
print("\nMissing values:\n", df.isnull().sum())
```

Output :

```
Missing values:
 age               0
workclass          0
fnlwgt             0
education          0
education-num      0
marital-status     0
occupation         0
relationship       0
race               0
sex                0
capital-gain       0
capital-loss       0
hours-per-week     0
native-country     0
income             0
dtype: int64
```

4. Preprocessing: one-hot encoding untuk kategorikal

```
        from sklearn.preprocessing import OneHotEncoder

        preprocessor = ColumnTransformer(
            transformers=[
                ('num', 'passthrough', numerical_cols),
                ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'),
        categorical_cols)
    ])
```

5. Buat pipeline: preprocessing + random forest

```
        rf_pipeline = Pipeline(steps=[
            ('preprocessor', preprocessor),
            ('classifier', RandomForestClassifier(
                n_estimators=200,
                max_depth=15,
```

```
                min_samples_split=5,
                min_samples_leaf=2,
                max_features='sqrt',
                random_state=42,
                n_jobs=-1,
                class_weight='balanced'  # karena data tidak seimbang
            ))
])
```

6. Split data

```
        X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size=0.2, random_state=42, stratify=y
        )

print(f"Train: {X_train.shape}, Test: {X_test.shape}")
```

7. Latih model

```
        rf_pipeline.fit(X_train, y_train)

        # 8. Prediksi & evaluasi
        y_pred = rf_pipeline.predict(X_test)

        # Akurasi
        accuracy = accuracy_score(y_test, y_pred)
print(f"\nAKURASI: {accuracy:.4f}")
```

Output :

```
AKURASI: 0.8073
```

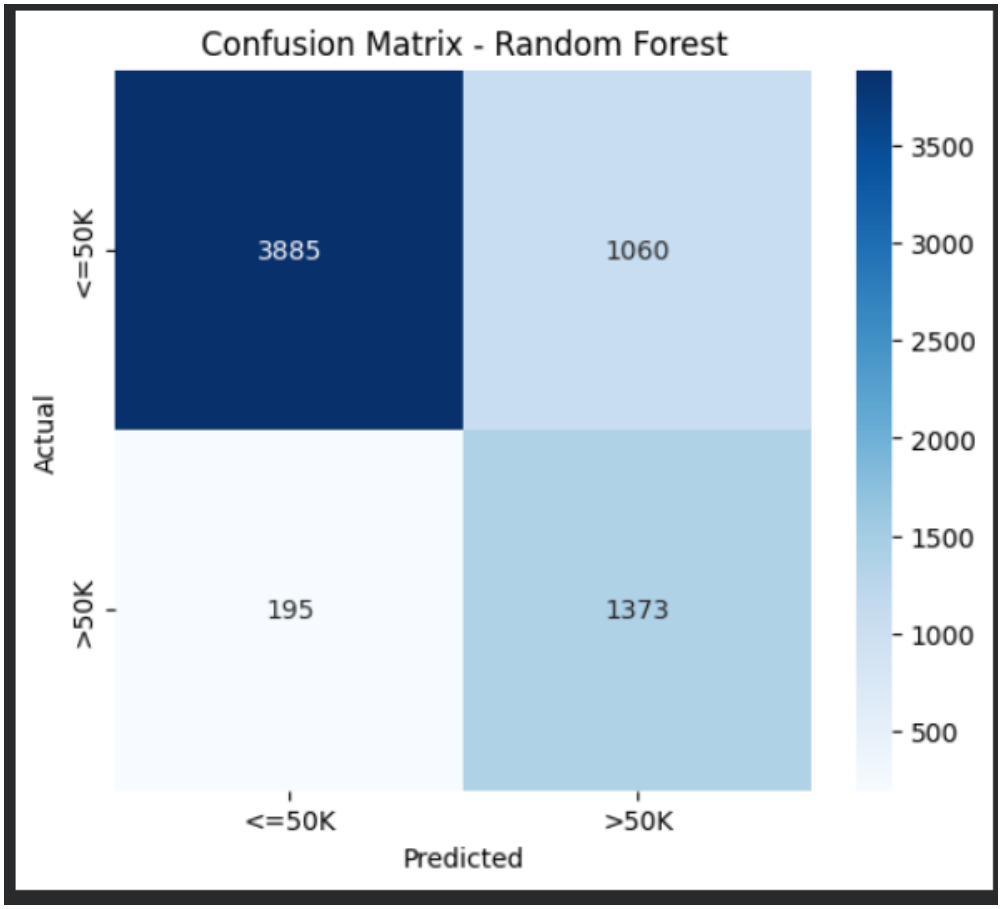
8. Praktik 8

```
        # Classification Report
        print("\nCLASSIFICATION REPORT:")
print(classification_report(y_test, y_pred, target_names=['<=50K', '>50K']))
```

```
        # Confusion Matrix
        cm = confusion_matrix(y_test, y_pred)
        plt.figure(figsize=(6,5))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['<=50K', '>50K'], yticklabels=['<=50K', '>50K'])
        plt.title('Confusion Matrix - Random Forest')
        plt.ylabel('Actual')
        plt.xlabel('Predicted')
        plt.show()
```

Output :

```
CLASSIFICATION REPORT:
              precision    recall  f1-score   support

       <=50K       0.95      0.79      0.86      4945
        >50K       0.56      0.88      0.69      1568

    accuracy                           0.81      6513
   macro avg       0.76      0.83      0.77      6513
weighted avg       0.86      0.81      0.82      6513
```



9. Feature importance (dari one-hot encoding)

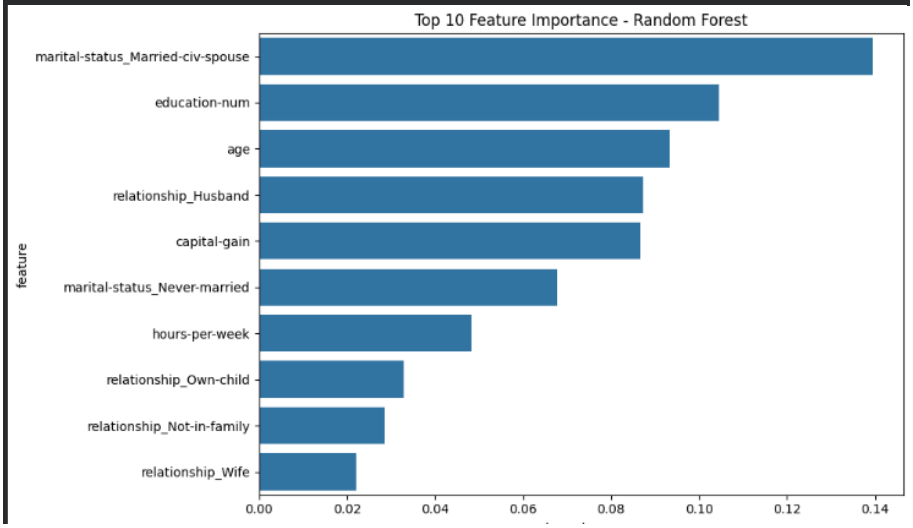
```
# Ambil nama fitur setelah encoding
ohe =
rf_pipeline.named_steps['preprocessor'].named_transformers_['cat']
ohe_feature_names = ohe.get_feature_names_out(categorical_cols)
all_feature_names = numerical_cols + ohe_feature_names.tolist()

# Importance
importances =
rf_pipeline.named_steps['classifier'].feature_importances_
feature_importance_df = pd.DataFrame({
    'feature': all_feature_names,
    'importance': importances
}).sort_values('importance', ascending=False)

# Tampilkan top 10
```

```
        print("\nTOP 10 FITUR PENTING:")
print(feature_importance_df.head(10))
```

```
        # Visualisasi
        plt.figure(figsize=(10, 6))
        top_n = 10
        sns.barplot(data=feature_importance_df.head(top_n), x='importance',
        y='feature')
        plt.title(f'Top {top_n} Feature Importance - Random Forest')
        plt.xlabel('Importance')
        plt.tight_layout()
        plt.show()
```

Output :

```
TOP 10 FITUR PENTING:
                                 feature  importance
33  marital-status_Married-civ-spouse    0.139503
2                         education-num    0.104494
0                                   age    0.093407
53                relationship_Husband    0.087245
3                          capital-gain    0.086701
35        marital-status_Never-married    0.067700
5                       hours-per-week    0.048197
56              relationship_Own-child    0.032960
54           relationship_Not-in-family    0.028545
58                    relationship_Wife    0.022123
```



10. Prediksi data baru

```
        # Contoh data orang baru
        data_baru = pd.DataFrame([{
            'age': 35,
            'workclass': 'Private',
            'fnlwgt': 215646,
            'education': 'Bachelors',
            'education-num': 13,
            'marital-status': 'Married-civ-spouse',
```

```
            'occupation': 'Exec-managerial',
            'relationship': 'Husband',
            'race': 'White',
            'sex': 'Male',
            'capital-gain': 0,
            'capital-loss': 0,
            'hours-per-week': 50,
            'native-country': 'United-States'
        }])

        prediksi = rf_pipeline.predict(data_baru)[0]
        prob = rf_pipeline.predict_proba(data_baru)[0]

        print(f"""\nPREDIKSI untuk data baru:
            age = 35,
            workclass = Private,
            fnlwgt = 215646,
            Education = Bachelors,
            education-num = 13,
            marital-status = Married-civ-spouse,
            occupation = Exec-managerial,
            Relationship =Husband,
            race =White,
            sex = Male,
            capital-gain = 0,
            capital-loss = 0,
            hours-per-week = 50,
            native-country =United-States
        """)
        print(f"Income: {['<=50K', '>50K'][prediksi]}")
print(f"Probabilitas <=50K: {prob[0]:.2%}, >50K: {prob[1]:.2%}")
```

Output :

```
PREDIKSI untuk data baru:
    age = 35,
    workclass = Private,
    fnlwgt = 215646,
    Education = Bachelors,
    education-num = 13,
    marital-status = Married-civ-spouse,
    occupation = Exec-managerial,
    Relationship =Husband,
    race =White,
    sex = Male,
    capital-gain = 0,
    capital-loss = 0,
    hours-per-week = 50,
    native-country =United-States

Income: >50K
Probabilitas <=50K: 10.29%, >50K: 89.71%
```

Praktik 3 **Klasifikasi dengan Random Forest untuk Dataset Iris**

1. Import Library

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.preprocessing import LabelEncoder
```

2. Load Dataset

```
iris = datasets.load_iris()
X = iris.data  # Fitur: sepal length, sepal width, petal length, petal
width
y = iris.target  # Target: 0=setosa, 1=versicolor, 2=virginica
```

```
        # Konversi ke DataFrame untuk eksplorasi
        df = pd.DataFrame(X, columns=iris.feature_names)
        df['species'] = y
df['species'] = df['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})
```

3. Eksplorasi data

```
        print("Info Dataset:")
        print(df.info())
        print("\n5 baris pertama:")
print(df.head())
```

Output :

```
Info Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education-num   32561 non-null  int64
 5   marital-status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital-gain    32561 non-null  int64
 11  capital-loss    32561 non-null  int64
 12  hours-per-week  32561 non-null  int64
 13  native-country  32561 non-null  object
 14  income          32561 non-null  int64
dtypes: int64(7), object(8)
memory usage: 3.7+ MB
None

5 baris pertama:
   age         workclass  fnlwgt  education  education-num  \
0   39         State-gov   77516  Bachelors             13
1   50  Self-emp-not-inc   83311  Bachelors             13
2   38           Private  215646    HS-grad              9
3   53           Private  234721       11th              7
4   28           Private  338409  Bachelors             13

       marital-status         occupation   relationship   race     sex  \
0       Never-married       Adm-clerical  Not-in-family  White    Male
1  Married-civ-spouse    Exec-managerial        Husband  White    Male
2            Divorced  Handlers-cleaners  Not-in-family  White    Male
3  Married-civ-spouse  Handlers-cleaners        Husband  Black    Male
4  Married-civ-spouse     Prof-specialty           Wife  Black  Female

   capital-gain  capital-loss  hours-per-week native-country  income
0          2174             0              40  United-States       0
1             0             0              13  United-States       0
2             0             0              40  United-States       0
3             0             0              40  United-States       0
4             0             0              40           Cuba       0
```

4. Split data: 80% train, 20% test

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)
```

5. Latih model Random Forest

```
rf_model = RandomForestClassifier(
```

```
            n_estimators=100,     # Jumlah pohon
            random_state=42,
            max_depth=None,
            min_samples_split=2,
            min_samples_leaf=1
        )

rf_model.fit(X_train, y_train)
```

6. Prediksi

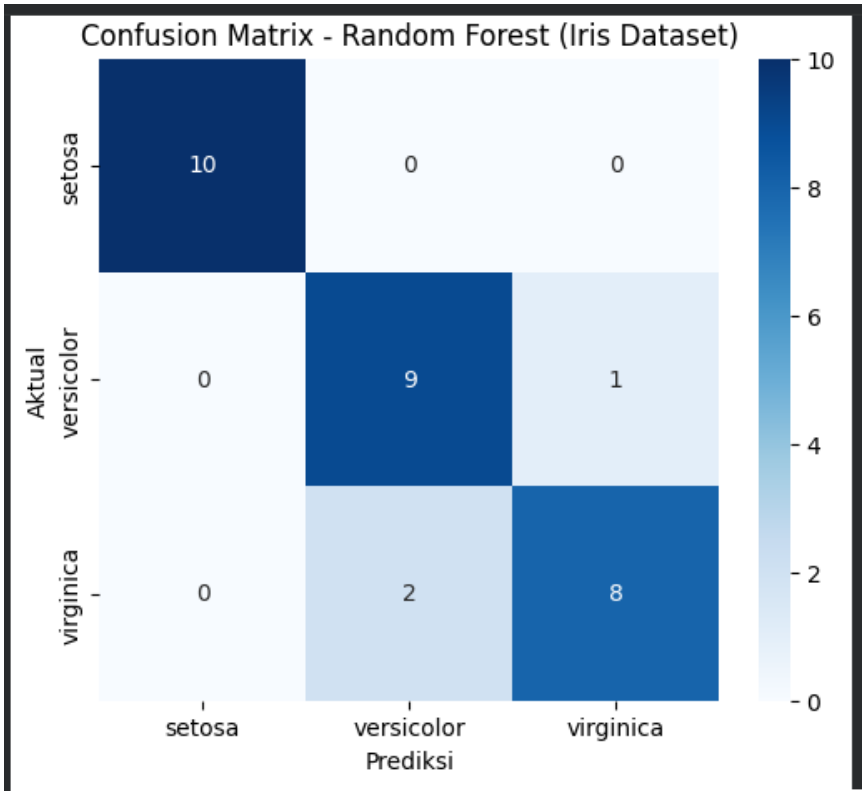
```
y_pred = rf_model.predict(X_test)
```

7. Evaluasi

```
        accuracy = accuracy_score(y_test, y_pred)
        print(f"\nAkurasi Model: {accuracy:.4f} ({accuracy*100:.2f}%)")
        print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=iris.target_names))
```

Output :

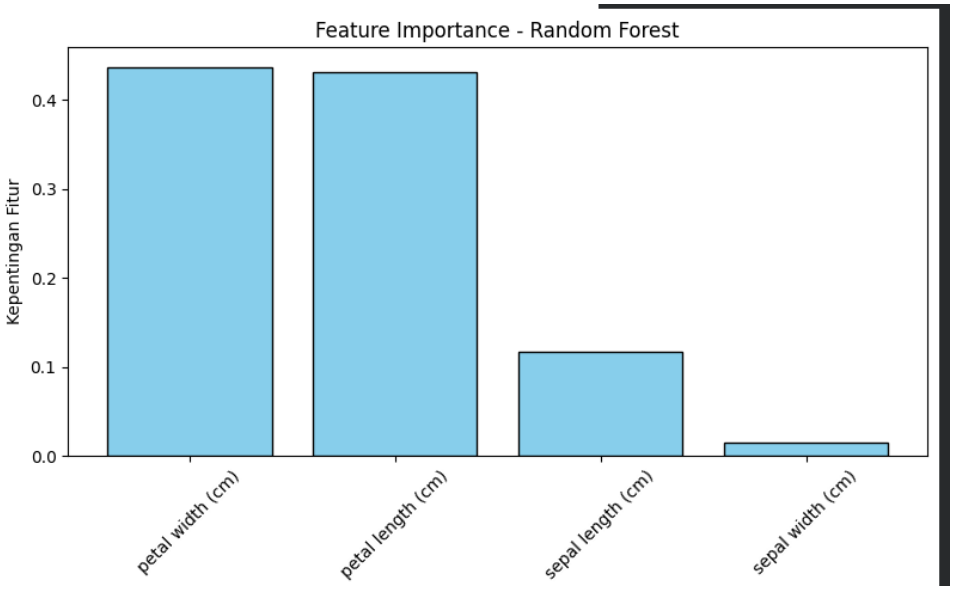
```
AKURASI: 0.9000 (90.00%)

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       0.82      0.90      0.86        10
   virginica       0.89      0.80      0.84        10

    accuracy                           0.90        30
   macro avg       0.90      0.90      0.90        30
weighted avg       0.90      0.90      0.90        30
```



8. Feature Importance

```
        importances = rf_model.feature_importances_
        feature_names = iris.feature_names

        # Urutkan berdasarkan pentingnya fitur
        indices = np.argsort(importances)[::-1]

        plt.figure(figsize=(8,5))
        plt.title("Feature Importance - Random Forest")
        plt.bar(range(X.shape[1]), importances[indices], align='center')
        plt.xticks(range(X.shape[1]), [feature_names[i] for i in indices],
        rotation=45)
        plt.tight_layout()
plt.show()
```

Output :



9. Prediksi Data Baru

```
# Konversi ke array numpy
import numpy as np

data_baru = np.array([
    [5.1, 3.5, 1.4, 0.2],
    [6.2, 2.8, 4.8, 1.8],
    [7.7, 3.0, 6.1, 2.3]
])

# Lakukan prediksi
prediksi = rf_model.predict(data_baru)
prediksi_nama = [iris.target_names[i] for i in prediksi]

# Probabilitas tiap kelas (opsional)
probabilitas = rf_model.predict_proba(data_baru)

# Tampilkan hasil
print("HASIL PREDIKSI DATA BARU")
for i in range(len(data_baru)):
    print(f"\nData {i+1}: {data_baru[i]}")
    print(f"   Prediksi     : {prediksi_nama[i]}")
    print(f"   Probabilitas  :")
    for j, prob in enumerate(probabilitas[i]):
```

```
    print(f"    - {iris.target_names[j]:10}: {prob:.4f}")
```

Output :

```
HASIL PREDIKSI DATA BARU

Data 1: [5.1 3.5 1.4 0.2]
   Prediksi      : setosa
   Probabilitas  :
     - setosa    : 1.0000
     - versicolor: 0.0000
     - virginica : 0.0000

Data 2: [6.2 2.8 4.8 1.8]
   Prediksi      : virginica
   Probabilitas  :
     - setosa    : 0.0000
     - versicolor: 0.2800
     - virginica : 0.7200

Data 3: [7.7 3.  6.1 2.3]
   Prediksi      : virginica
   Probabilitas  :
     - setosa    : 0.0000
     - versicolor: 0.0000
     - virginica : 1.0000
```

B. Latihan

1. Klasifikasi menggunakan Random Forest
   Mengambil dataset dari URL yang berisi ribuan pesan SMS dengan label *ham/spam*

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import
CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score


url =
"https://raw.githubusercontent.com/justmarkham/pycon-
2016-tutorial/master/data/sms.tsv"


df = pd.read_csv(url, sep='\t', header=None,
names=['label', 'message'])

print("="*50)
```

```
print("Dataset SMS Spam (5 Data Pertama):")
print(df.head())
print("="*50)
print(f"Jumlah Data: {len(df)}")
print(df['label'].value_counts())
print("="*50)

# ========================================
# 3. Pisahkan fitur dan label
# ========================================
X = df['message']   # fitur teks SMS
y = df['label']     # target (spam / ham)

# ========================================
# 4. Bagi data menjadi training dan testing
# ========================================
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# ========================================
# 5. Vektorisasi teks (ubah ke bentuk numerik)
# ========================================
vectorizer = CountVectorizer(stop_words='english')  #
hapus kata umum (stopwords)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# ========================================
# 6. Buat dan latih model Random Forest
# ========================================
model = RandomForestClassifier(n_estimators=100,
random_state=42)
model.fit(X_train_vec, y_train)

# ========================================
# 7. Evaluasi model
# ========================================
y_pred = model.predict(X_test_vec)

print("\n=== Confusion Matrix ===")
print(confusion_matrix(y_test, y_pred))

print("\n=== Classification Report ===")
print(classification_report(y_test, y_pred))

print("\n=== Akurasi Model ===")
```

```
print(f"{accuracy_score(y_test, y_pred):.4f}")

new_message = ["Your account has been selected for a
prize!"]
new_message_vec = vectorizer.transform(new_message)
prediction = model.predict(new_message_vec)[0]

print("\n=== Prediksi Pesan Baru ===")
print("Pesan :", new_message[0])
print("Hasil Prediksi :", prediction)
```

2. Evaluasi modelnya
   rediksi data uji menggunakan model yang sudah dilatih.

```
=== Akurasi Model ===
0.9767
```

3. Prediksi untuk data baru : Your account has been selected for a prize!

```
==================================================
Dataset SMS Spam (5 Data Pertama):
  label                                            message
0   ham  Go until jurong point, crazy.. Available only ...
1   ham                      Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3   ham  U dun say so early hor... U c already then say...
4   ham  Nah I don't think he goes to usf, he lives aro...
==================================================
Jumlah Data: 5572
label
ham     4825
spam     747
Name: count, dtype: int64
==================================================

=== Confusion Matrix ===
[[966   0]
 [ 26 123]]

=== Classification Report ===
              precision    recall  f1-score   support

         ham       0.97      1.00      0.99       966
        spam       1.00      0.83      0.90       149

    accuracy                           0.98      1115
   macro avg       0.99      0.91      0.95      1115
weighted avg       0.98      0.98      0.98      1115


=== Akurasi Model ===
0.9767

=== Prediksi Pesan Baru ===
Pesan : Your account has been selected for a prize!
Hasil Prediksi : ham
```

Penjelasan : Model mendeteksi kata seperti “selected”, “prize” yang sering muncul dalam pesan penipuan atau promosi, sehingga mengklasifikasikannya sebagai spam.

C. Tugas

a. Cari open dataset untuk klasifikasi

```
# ==========================================
# 1. Import Library
# ==========================================
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import matplotlib.pyplot as plt
import pandas as pd

# ========================================
# 2. Load Dataset Iris
# ========================================
iris = load_iris()
X = iris.data
y = iris.target

# Buat DataFrame untuk melihat data
df = pd.DataFrame(X, columns=iris.feature_names)
df['target'] = y

print("Dataset Iris (5 data pertama):")
print(df.head())
print("\nJumlah data:", len(df))
print("="*50)

# ========================================
# 3. Split Data Training dan Testing
# ========================================
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# ========================================
# 4. Decision Tree Classifier
# ========================================
dt_model = DecisionTreeClassifier(random_state=42)
```

```
dt_model.fit(X_train, y_train)
y_pred_dt = dt_model.predict(X_test)
acc_dt = accuracy_score(y_test, y_pred_dt)

print("\n=== Evaluasi Decision Tree ===")
print(confusion_matrix(y_test, y_pred_dt))
print(classification_report(y_test, y_pred_dt))
print("Akurasi Decision Tree:", acc_dt)

# ========================================
# 5. Random Forest Classifier
# ========================================
rf_model = RandomForestClassifier(n_estimators=100,
random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
acc_rf = accuracy_score(y_test, y_pred_rf)

print("\n=== Evaluasi Random Forest ===")
print(confusion_matrix(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))
print("Akurasi Random Forest:", acc_rf)

# ========================================
# 6. Visualisasi Perbandingan Akurasi
# ========================================
models = ['Decision Tree', 'Random Forest']
accuracies = [acc_dt, acc_rf]

plt.figure(figsize=(6,4))
plt.bar(models, accuracies, color=['skyblue', 'lightgreen'])
plt.title('Perbandingan Akurasi Model Klasifikasi')
plt.ylabel('Nilai Akurasi')
```

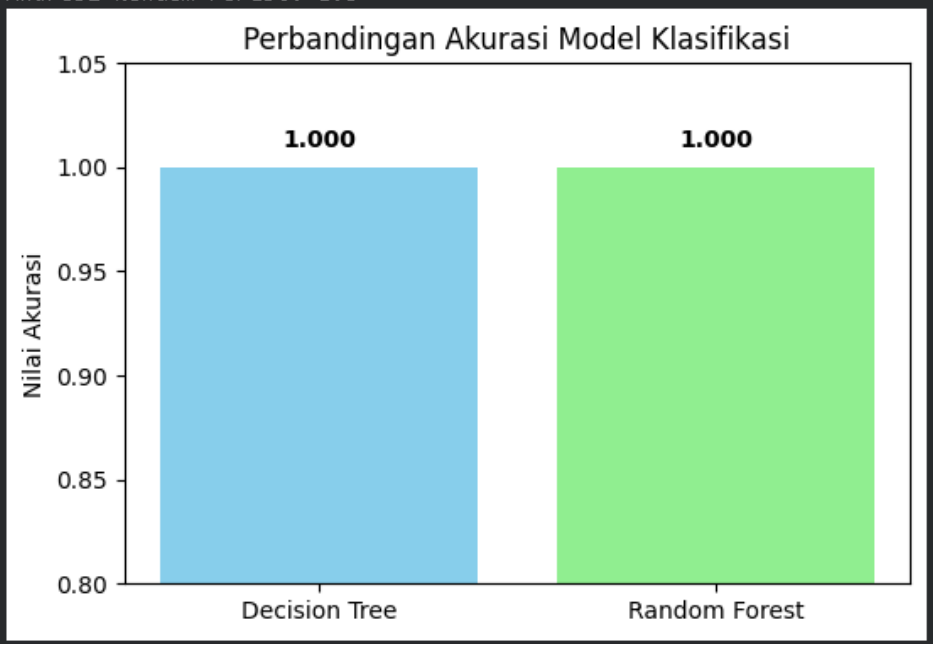
```
        plt.ylim(0.8, 1.05)

        # Tampilkan nilai akurasi di atas bar
        for i, acc in enumerate(accuracies):
            plt.text(i, acc + 0.01, f"{acc:.3f}", ha='center', fontsize=10,
        fontweight='bold')

plt.show()
```

b. Lakukan klasifikasi dengan pohon keputusan dan Random Forest

```
        models = ['Decision Tree', 'Random Forest']
        accuracies = [acc_dt, acc_rf]

        plt.figure(figsize=(6,4))
        plt.bar(models, accuracies, color=['skyblue', 'lightgreen'])
        plt.title('Perbandingan Akurasi Model Klasifikasi')
        plt.ylabel('Nilai Akurasi')
plt.ylim(0.8, 1.05)
```

c. Bandingkan Evaluasi modelnya bagus mana.

```
...                precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

Akurasi Random Forest: 1.0
```



Penjelasan :

program tersebut digunakan untuk membuat grafik batang (bar chart) yang menampilkan perbandingan akurasi antara dua model klasifikasi, yaitu Decision Tree dan Random Forest. Variabel models berisi nama kedua model, sedangkan accuracies berisi nilai akurasi masing-masing. Kemudian, plt.bar() digunakan untuk menggambar batang dengan warna berbeda (biru muda dan hijau muda), disertai judul “Perbandingan Akurasi Model Klasifikasi”, label sumbu Y “Nilai Akurasi”, dan batas tampilan nilai akurasi dari 0.8 hingga 1.05 agar grafik lebih jelas terbaca.