LAPORAN ANALISIS IMPLEMENTASI ALGORITMA DECISION TREE

DATA MINING

Dosen Pengampu:

Dr. Arya Adhyaksa Wakita S.Si., M.Si.

Disusun Oleh:

Hamzah Fachrudin M

Kelas : 03TPLP027 – Reguler A

Teknik Informatika

Fakultas Teknik

Universitas Pamulang

Jl. Surya Kencana No. 1 Pamulang Telp (021)7412566, Fax. (021)7412566

Tangerang Selatan, Banten

Penjelasan

1. **Import dan Baca Dataset**

```
import pandas as pd
dataset = pd.read_csv("Benign Traffic.csv")
dataset2 = pd.read_csv("DDoS ICMP Flood.csv")
dataset3 = pd.read_csv("DDoS UDP Flood.csv")
```

Pada bagian ini, kita menggunakan library pandas untuk membaca tiga file CSV yang berisi data lalu lintas jaringan:

- Benign Traffic.csv: berisi data lalu lintas normal.
- DDoS ICMP Flood.csv: berisi data lalu lintas serangan DDoS dengan metode ICMP Flood.
- DDoS UDP Flood.csv: berisi data lalu lintas serangan DDoS dengan metode UDP Flood.

Setiap file kemungkinan memiliki format kolom yang sama, sehingga bisa digabungkan di langkah selanjutnya.

2. **Gabungkan Dataset**

```
hasilgabung = pd.concat([dataset, dataset2, dataset3], ignore_index=True)
```

Ketiga dataset digabungkan menggunakan fungsi pd.concat() dari pandas. Dengan ignore_index=True, index akan direset agar tidak terjadi duplikasi index dari masing-masing file. Hasilnya adalah satu dataset besar (hasilgabung) yang berisi campuran data benign dan serangan.

3. **Pemisahan Fitur dan Label**

```
x = hasilgabung.iloc[:, 7:76]
y = hasilgabung.iloc[:, 83:84]
```

- x: Diambil dari kolom ke-7 hingga ke-75. Ini adalah fitur (input) yang akan digunakan oleh model untuk belajar. Biasanya berisi nilai-nilai seperti ukuran paket, durasi koneksi, jumlah byte, dll.
- y: Diambil dari kolom ke-83, yang merupakan label (output) atau target dari klasifikasi. Kolom ini menentukan apakah baris tersebut merupakan traffic benign, atau salah satu jenis serangan DDoS.

Pastikan bahwa indeks kolom 7:76 dan 83:84 sesuai dengan struktur data CSV kamu.

4. **Split Data: Latih dan Uji**

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

Data dibagi menjadi dua bagian:

- **80% data latih (training)** → untuk melatih model.
- **20% data uji (testing)** → untuk menguji performa model.

random_state=42 digunakan agar pembagian data bersifat *reproducible* (hasil selalu sama tiap kali dijalankan).

5. **Pelatihan Model: Decision Tree**

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier

alya = DecisionTreeClassifier(criterion='entropy', splitter='random')
alya.fit(x_train, y_train)
```

Kita melatih model klasifikasi menggunakan algoritma *Decision Tree*:

- criterion='entropy': model akan membagi data berdasarkan nilai *information gain*, yang mengukur ketidakteraturan (entropy).
- splitter='random': pemilihan fitur pembagi dilakukan secara acak (biasanya digunakan untuk menghindari overfitting dan untuk keperluan eksperimen).
- alya.fit(...): model dilatih menggunakan data latih.

6. **Prediksi dan Evaluasi Akurasi**

```
y_pred = alya.predict(x_test)

from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
```

- alya.predict(x_test): melakukan prediksi terhadap data uji menggunakan model yang telah dilatih.
- accuracy_score(...): menghitung akurasi dari hasil prediksi, yaitu seberapa banyak prediksi yang benar dibanding total prediksi.

Akurasi adalah metrik sederhana namun efektif untuk melihat performa awal model.

7. **Visualisasi Pohon Keputusan**

```
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure(figsize=(10, 7))
tree.plot_tree(alya, feature_names=x.columns.values, class_names=np.array(['Benign Traffic', 'DDos ICMP Flood', 'DDoS UDP Flood']), filled=True)
plt.show()
```

Visualisasi struktur decision tree yang telah dibuat:

- feature_names: nama-nama kolom fitur.
- class_names: label kelas target.
- filled=True: memberi warna untuk membantu interpretasi.

Pohon ini memperlihatkan bagaimana model membuat keputusan berdasarkan fitur input.

8. **Visualisasi Confusion Matrix**

```
import seaborn as lol
from sklearn import metrics
label = np.array(['Benign Traffic','DDos ICMP Flood','DDoS UDP Flood'])

conf_matrix = metrics.confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10, 10))
lol.heatmap(conf_matrix, annot=True, xticklabels=label, yticklabels=label)
plt.xlabel('Prediksi')
plt.ylabel('Fakta')
plt.show()
```

- Menggunakan *confusion matrix* untuk mengevaluasi kinerja model lebih mendetail.
- heatmap: menunjukkan jumlah prediksi benar dan salah untuk setiap kelas.
- annot=True: menampilkan angka pada tiap kotak matrix.

Confusion matrix membantu mengetahui apakah model melakukan banyak kesalahan pada kelas tertentu (misalnya sering salah mengklasifikasikan DDoS UDP sebagai benign).