Laporan Tugas Besar Kelompok Dasar Kecerdasan Artifisial (DKA)

“Network Intrusion Detection”

Disusun oleh:

Muhammad Irgiansyah	103012300039
Bill Stephen	103012330197

Program Studi S1 Informatika

Fakultas Informatika

Universitas Telkom

Bandung 2024/2025

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi komputer yang sangat pesat menyebabkan aktivitas komunikasi dan pertukaran data melalui jaringan komputer menjadi bagian yang tidak terpisahkan dari kehidupan sehari-hari. Dalam konteks ini, keamanan jaringan menjadi hal yang sangat krusial untuk menjamin integritas, kerahasiaan, dan ketersediaan data. Namun, dengan semakin berkembangnya teknologi, muncul pula berbagai ancaman siber yang berpotensi merusak sistem dan data yang ada. Oleh karena itu, sistem deteksi intrusi (Intrusion Detection System/IDS) menjadi elemen penting dalam menjaga keamanan jaringan dengan cara mengidentifikasi aktivitas mencurigakan yang dapat menandakan serangan.

Dalam pengembangan sistem IDS, pendekatan berbasis data menggunakan machine learning telah menjadi metode yang efektif untuk mendeteksi berbagai jenis serangan secara otomatis. Dataset simulasi yang merepresentasikan lalu lintas jaringan sangat membantu dalam pembangunan model deteksi yang dapat membedakan antara aktivitas normal dan serangan berbahaya. Dengan memanfaatkan algoritma seperti K-Nearest Neighbors (KNN) dan Fuzzy Logic, model dapat dibuat untuk meningkatkan akurasi deteksi dan meminimalkan kesalahan klasifikasi.

Namun demikian, pengembangan model deteksi ini masih menghadapi tantangan, terutama dalam hal pemilihan fitur yang tepat dan penanganan ketidakpastian data. Dengan evaluasi performa yang menggunakan metrik akurasi, precision, recall, dan F1-score, diharapkan sistem dapat memberikan hasil yang optimal dan dapat diaplikasikan sebagai dasar pengembangan sistem IDS yang lebih operasional di masa depan.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah dalam proyek ini adalah sebagai berikut:

1. Dataset yang digunakan merupakan data simulasi dan tidak mencakup data jaringan real-time.

2. Klasifikasi dilakukan dengan dua pendekatan, yaitu klasifikasi biner (normal vs intrusion) dan klasifikasi multi-kelas (DoS, Probe, R2L, U2R).

3. Algoritma yang digunakan dibatasi pada dua metode utama, yakni K-Nearest Neighbors (KNN) dan Fuzzy Logic (Mamdani dan Sugeno).

4. Evaluasi performa model hanya dilakukan menggunakan data uji dari dataset yang sama tanpa implementasi langsung pada sistem IDS nyata.

5. Proyek ini tidak mencakup integrasi model ke dalam sistem IDS operasional.

1.3 Tujuan dan Manfaat

Tujuan dari penelitian dan pengembangan dalam proyek ini adalah:

- Menerapkan algoritma pembelajaran mesin, yaitu KNN dan Fuzzy Logic (Mamdani & Sugeno), untuk mendeteksi intrusi pada jaringan komputer.

- Membandingkan performa metode KNN dan Fuzzy Logic dalam mendeteksi berbagai jenis serangan jaringan.

- Mengevaluasi performa model menggunakan metrik akurasi, precision, recall, dan F1-score guna menilai efektivitas klasifikasi.

Manfaat yang diharapkan dari proyek ini adalah:

- Memberikan solusi sistem deteksi intrusi yang dapat membantu meningkatkan keamanan jaringan dengan mendeteksi aktivitas mencurigakan secara otomatis.

- Menjadi referensi dan dasar penelitian lebih lanjut dalam pengembangan sistem IDS berbasis machine learning dan logika fuzzy.

- Membantu tim keamanan siber dalam melakukan identifikasi dini terhadap serangan yang dapat merugikan sistem dan data jaringan.

BAB II
METODE

2.1. Dataset yang Digunakan

Dataset yang digunakan dalam penelitian ini adalah **Network Intrusion Detection Dataset** yang diambil dari Kaggle, yaitu https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection , dimana data tersebut berisi data simulasi aktivitas jaringan apakah suatu aktivitas jaringan termasuk *normal* atau *intrusion*. Dataset ini memiliki berbagai fitur penting, seperti duration, protocol_type, service, flag, src_bytes, dst_bytes, dan lain-lain yang merepresentasikan karakteristik dari setiap koneksi jaringan. Dataset ini dibagi menjadi dua kelas target: normal dan intrusion, serta juga dapat diklasifikasikan ke dalam beberapa jenis serangan (DoS, Probe, R2L, U2R).

```
# === 2. Pilih Fitur Penting ===
selected_features = ['duration', 'protocol_type', 'service', 'flag',
                     'src_bytes', 'dst_bytes', 'count', 'srv_count']
df = df[selected_features].copy()
```

2.2. Preprocessing Dataset

Sebelum pemodelan dilakukan, dilakukan beberapa tahapan preprocessing agar data siap digunakan oleh algoritma klasifikasi:

- **Encoding Fitur Kategorikal**: Beberapa fitur seperti protocol_type, service, dan flag dikonversi menjadi format numerik menggunakan teknik encoding agar dapat diproses secara numerik oleh model.

```
# === 4. Encode Fitur Kategorikal ===
cat_cols = ['protocol_type', 'service', 'flag']
for col in cat_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
```

- **Normalisasi Data Numerik**: Data numerik dinormalisasi menggunakan metode Min-Max Scaling agar rentang nilai fitur berada dalam skala yang sama, meningkatkan akurasi dan performa algoritma KNN dan fuzzy logic.

```
# === 9. Normalisasi Fitur Numerik ===
numeric_cols = ['duration', 'src_bytes', 'dst_bytes', 'count', 'srv_count']
scaler = MinMaxScaler()
X_train_bin[numeric_cols] = scaler.fit_transform(X_train_bin[numeric_cols])
X_test_bin[numeric_cols] = scaler.transform(X_test_bin[numeric_cols])
X_train_multi[numeric_cols] = scaler.fit_transform(X_train_multi[numeric_cols])
X_test_multi[numeric_cols] = scaler.transform(X_test_multi[numeric_cols])
```

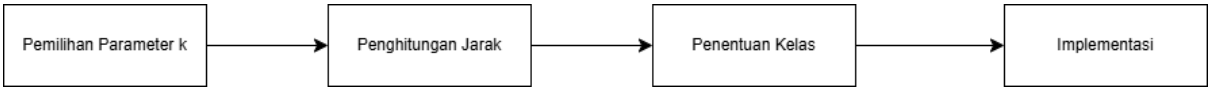
- **Pembagian Dataset**: Dataset dibagi menjadi data latih dan data uji dengan perbandingan 80:20 untuk melatih model dan menguji performa klasifikasi.

```
# === 8. Split Data 80:20 ===
X_train_bin, X_test_bin, y_train_bin, y_test_bin = train_test_split(X, y_bin, test_size=0.2, random_state=42)
X_train_multi, X_test_multi, y_train_multi, y_test_multi = train_test_split(X, y_multi, test_size=0.2, random_state=42)
```

2.3. Metode Klasifikasi

Dalam penelitian ini digunakan tiga metode klasifikasi yaitu K-Nearest Neighbors (KNN), Fuzzy Mamdani, dan Fuzzy Sugeno untuk mendeteksi aktivitas jaringan yang mencurigakan.

2.3.1 K-Nearest Neighbors (KNN)



Gambar Blok Diagram KNN

KNN merupakan metode klasifikasi berbasis instance yang menentukan kelas suatu data berdasarkan mayoritas kelas dari tetangga terdekatnya. Gambar diatas merupakan proses KNN yang dirancang pada penlitian secara umum dalam bentuk diagram blok. Namun disini, proses KNN pada penelitian ini secara mendetail adalah sebagai berikut:

- **Pemilihan Parameter k**: Nilai k (jumlah tetangga terdekat) dipilih secara eksperimen untuk mendapatkan hasil optimal.

  Kodingan :

```
# === 10. Evaluasi Nilai k Terbaik ===
k_values = list(range(1, 11))
acc_scores = []
for k in k_values:
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train_bin, y_train_bin)
    pred = model.predict(X_test_bin)
    acc_scores.append(accuracy_score(y_test_bin, pred))
optimal_k = k_values[np.argmax(acc_scores)]
print(f"Nilai k optimal: {optimal_k} (Akurasi: {acc_scores[np.argmax(acc_scores)]:.4f})")

print("\n=== Evaluasi Akurasi untuk Setiap Nilai k ===")
for i, k in enumerate(k_values):
    print(f"k = {k}: Akurasi = {acc_scores[i]:.4f}")
```

  Output :

```
Nilai k optimal: 1 (Akurasi: 0.9950)

=== Evaluasi Akurasi untuk Setiap Nilai k ===
k = 1: Akurasi = 0.9950
k = 2: Akurasi = 0.9900
k = 3: Akurasi = 0.9900
k = 4: Akurasi = 0.9850
k = 5: Akurasi = 0.9850
k = 6: Akurasi = 0.9800
k = 7: Akurasi = 0.9800
k = 8: Akurasi = 0.9800
k = 9: Akurasi = 0.9800
k = 10: Akurasi = 0.9700
```

- **Penghitungan Jarak**: Jarak Euclidean digunakan untuk mengukur kedekatan antara data uji dan data latih dalam ruang fitur. Jika dilihat secara implisit, penghitungan jarak Euclidean dilakukan otomatis ketika membuat dan melatih model KNN seperti ini:

- **Penentuan Kelas**: Kelas dari data uji ditentukan berdasarkan mayoritas kelas dari k tetangga terdekat.

  Kodingan :

  Binary

```python
# === 11. Klasifikasi Biner dengan K Optimal ===
knn_bin = KNeighborsClassifier(n_neighbors=optimal_k)
knn_bin.fit(X_train_bin, y_train_bin)

# === Evaluasi pada Data Training ===
y_train_pred_bin = knn_bin.predict(X_train_bin)
```

  Multi-Kelas

```python
# === 12. Klasifikasi Multi-Kelas ===
knn_multi = KNeighborsClassifier(n_neighbors=optimal_k)
knn_multi.fit(X_train_multi, y_train_multi)

# Tentukan semua label unik yang mungkin muncul dalam dataset
labels_multi = sorted(df['attack_type'].unique())
```

- **Implementasi**: Pada Eksekusi kodingan yang diberikan, algoritma KNN diimplementasikan dan diintegrasikan dengan preprocessing untuk memastikan data input konsisten. Dalam hal ini, klasifikasinya menjadi 2 bagian, yaitu Binary dan Multi-Kelas yang masing-masing klasifikasinya memiliki 3 data, yaitu Data Training, Data Testing, dan Data Keseluruhan.

❖ ***BINARY***

a. Data Training

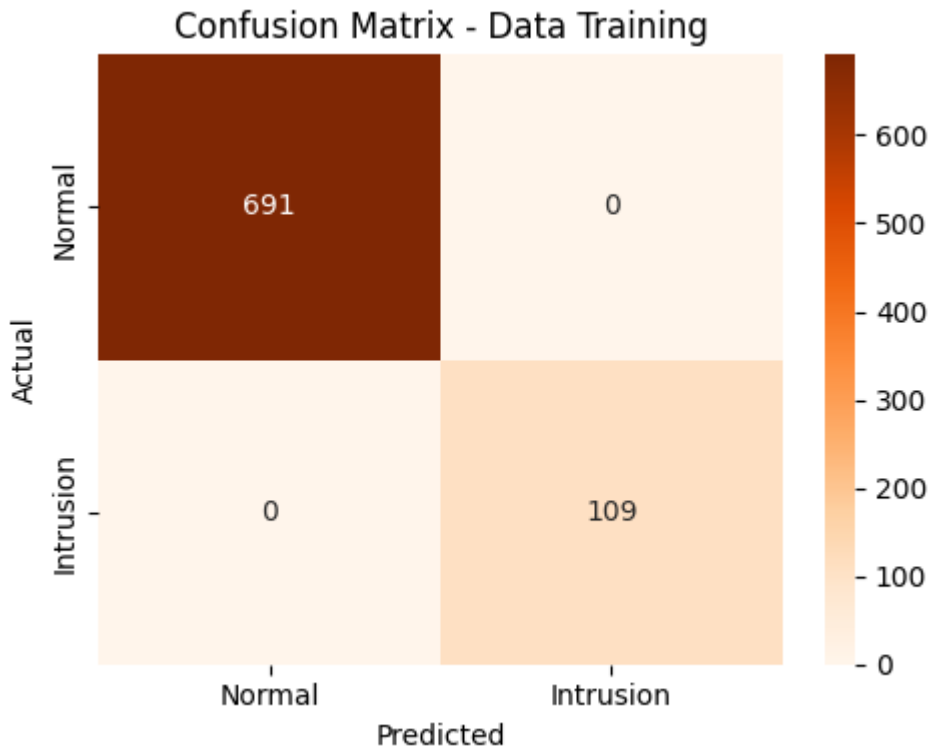
Kodingan :

```
# === Evaluasi pada Data Uji (Testing) ===
y_test_pred_bin = knn_bin.predict(X_test_bin)
test_acc_bin = accuracy_score(y_test_bin, y_test_pred_bin)
test_prec_bin = precision_score(y_test_bin, y_test_pred_bin, zero_division=0)
test_rec_bin = recall_score(y_test_bin, y_test_pred_bin, zero_division=0)
test_f1_bin = f1_score(y_test_bin, y_test_pred_bin, zero_division=0)
print(f"[Testing]      Akurasi: {test_acc_bin:.4f} | Precision: {test_prec_bin:.4f} | Recall: {test_rec_bin:.4f} | F1-score: {test_f1_bin:.4f}")
# Confusion Matrix - Data Testing
cm_test = confusion_matrix(y_test_bin, y_test_pred_bin)
plt.figure(figsize=(5, 4))
sns.heatmap(cm_test, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Normal', 'Intrusion'], yticklabels=['Normal', 'Intrusion'])
plt.title("Confusion Matrix - Data Testing")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

Hasil dari Data Training :

```
[Training]      Akurasi: 1.0000 | Precision: 1.0000 | Recall: 1.0000 | F1-score: 1.0000
```

Berikut merupakan Confusion matrix dari Data Training :



Dari sini terlihat bahwa pada fase training, model KNN yang digunakan mengklasifikasikan **100%** data dengan tepat (baik Normal maupun Intrusion). Tidak ada false positive maupun false negative di data training, sehingga confusion matrix-nya menjadi matriks diagonal sempurna.

b. Data Testing

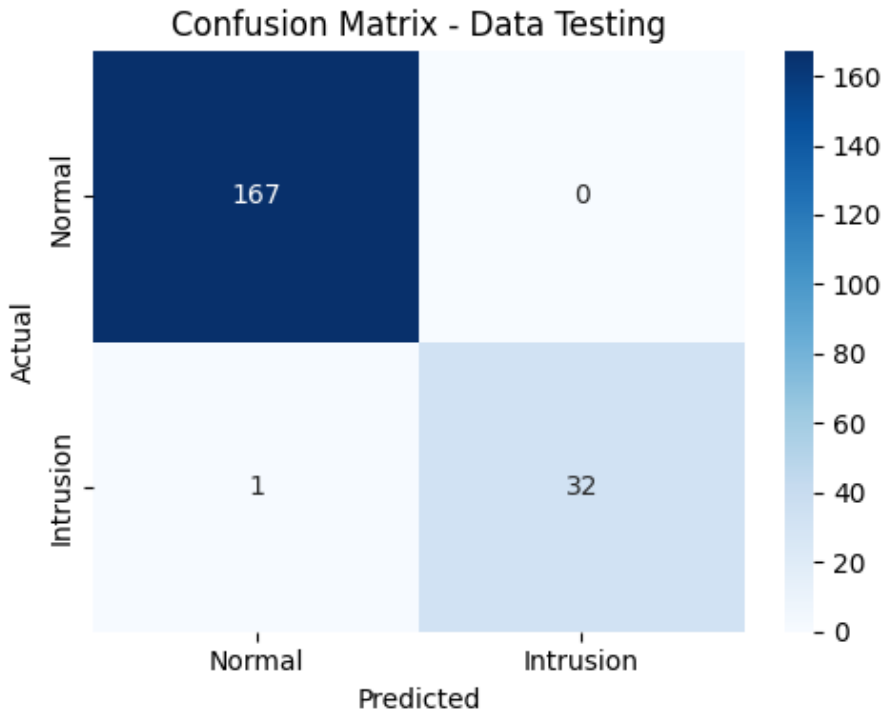
Kodingan :

```
# === Evaluasi pada Data Uji (Testing) ===
y_test_pred_bin = knn_bin.predict(X_test_bin)
test_acc_bin = accuracy_score(y_test_bin, y_test_pred_bin)
test_prec_bin = precision_score(y_test_bin, y_test_pred_bin, zero_division=0)
test_rec_bin = recall_score(y_test_bin, y_test_pred_bin, zero_division=0)
test_f1_bin = f1_score(y_test_bin, y_test_pred_bin, zero_division=0)
print(f"[Testing]      Akurasi: {test_acc_bin:.4f} | Precision: {test_prec_bin:.4f} | Recall: {test_rec_bin:.4f} | F1-score: {test_f1_bin:.4f}")
# Confusion Matrix - Data Testing
cm_test = confusion_matrix(y_test_bin, y_test_pred_bin)
plt.figure(figsize=(5, 4))
sns.heatmap(cm_test, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Normal', 'Intrusion'], yticklabels=['Normal', 'Intrusion'])
plt.title("Confusion Matrix - Data Testing")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

Hasil dari Data Testing :

```
[Training]      Akurasi: 1.0000 | Precision: 1.0000 | Recall: 1.0000 | F1-score: 1.0000
```

Berikut merupakan Confusion matrix dari Data Testing :



Dengan demikian, pada data testing model mencapai:

- **True Negative (TN)** = 167
- **False Positive (FP)** = 0
- **False Negative (FN)** = 1
- **True Positive (TP)** = 32

Sehingga akurasi testing (tanpa menghitung metrik lain) kira-kira:

$$167+32 \,/\, (167+0+1+32)=199200=0{,}995 \quad (\approx 99{,}5\%)$$

Hanya ada **1 intrusion yang terlewat** (diketegorikan sebagai Normal), sisanya sempurna.

c. Data Keseluruhan

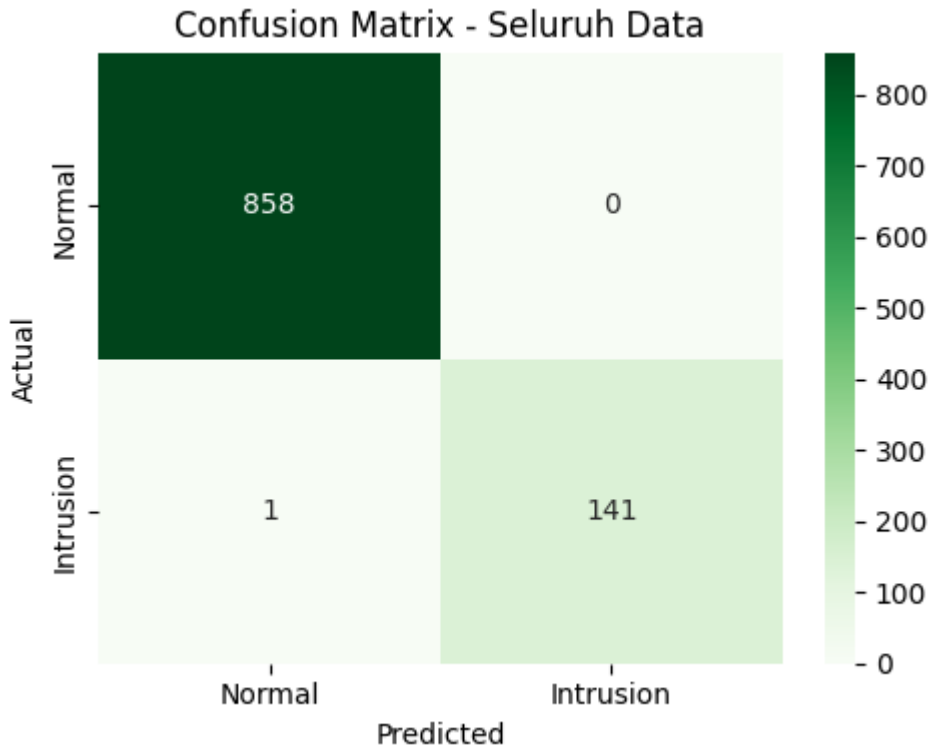
Kodingan :

```python
# === Evaluasi pada Seluruh Data Gabungan ===
X_all_bin = pd.concat([X_train_bin, X_test_bin])
y_all_bin = pd.concat([y_train_bin, y_test_bin])
y_all_pred_bin = knn_bin.predict(X_all_bin)
all_acc_bin = accuracy_score(y_all_bin, y_all_pred_bin)
all_prec_bin = precision_score(y_all_bin, y_all_pred_bin, zero_division=0)
all_rec_bin = recall_score(y_all_bin, y_all_pred_bin, zero_division=0)
all_f1_bin = f1_score(y_all_bin, y_all_pred_bin, zero_division=0)
print(f"[Keseluruhan]  Akurasi: {all_acc_bin:.4f} | Precision: {all_prec_bin:.4f} | Recall: {all_rec_bin:.4f} | F1-score: {all_f1_bin:.4f}")
# Confusion Matrix - Seluruh Data
cm_all = confusion_matrix(y_all_bin, y_all_pred_bin)
plt.figure(figsize=(5, 4))
sns.heatmap(cm_all, annot=True, fmt='d', cmap='Greens',
            xticklabels=['Normal', 'Intrusion'], yticklabels=['Normal', 'Intrusion'])
plt.title("Confusion Matrix - Seluruh Data")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

Hasil dari Data Keseluruhan :

```
[Keseluruhan]  Akurasi: 0.9990 | Precision: 1.0000 | Recall: 0.9930 | F1-score: 0.9965
```

Berikut merupakan Confusion matrix dari Data Keseluruhan :



Ini adalah gabungan (concatenate) data training dan testing untuk klasifikasi biner.

- **Normal**: total 691 (training) + 167 (testing) = 858 kasus, semuanya benar diklasifikasikan.

- **Intrusion**: total 109 (training) + 33 (testing) = 142. Namun dari 142, 1 kasus terlewat (FN), sehingga yang terdeteksi Intrusion = 141.

Secara keseluruhan model hanya membuat **1 kesalahan** pada keseluruhan 1000 sampel biner yang digunakan.

❖ ***MULTI-KELAS***

a. Data Training

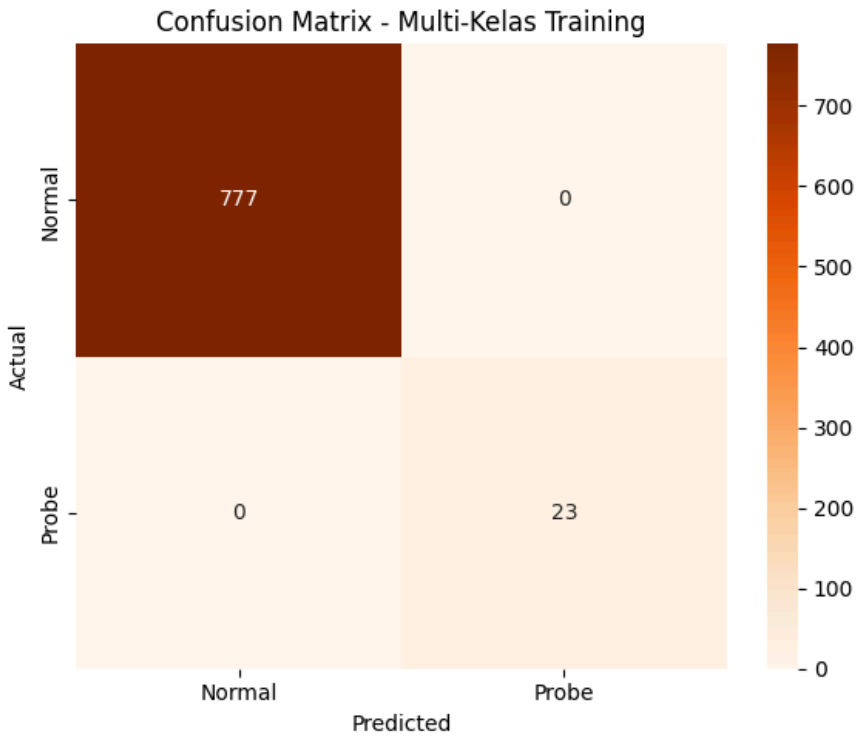
Kodingan :

```
# === Evaluasi pada Data Training (Multi-Kelas) ===
y_train_pred_multi = knn_multi.predict(X_train_multi)
train_acc_multi = accuracy_score(y_train_multi, y_train_pred_multi)
print("\n=== Evaluasi Klasifikasi Multi-Kelas - Training ===")
print(f"Akurasi: {train_acc_multi:.4f}")
print("Classification Report:\n", classification_report(y_train_multi, y_train_pred_multi, labels=labels_multi))
cm_train_multi = confusion_matrix(y_train_multi, y_train_pred_multi, labels=labels_multi)
plt.figure(figsize=(6, 5))
sns.heatmap(cm_train_multi, annot=True, fmt='d', cmap='Oranges',
            xticklabels=labels_multi,
            yticklabels=labels_multi)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Multi-Kelas Training')
plt.tight_layout()
plt.show()
```

Hasil dari Data Training :

```
=== Evaluasi Klasifikasi Multi-Kelas - Training ===
Akurasi: 1.0000
Classification Report:
              precision    recall  f1-score   support

      Normal       1.00      1.00      1.00       777
       Probe       1.00      1.00      1.00        23

    accuracy                           1.00       800
   macro avg       1.00      1.00      1.00       800
weighted avg       1.00      1.00      1.00       800
```

Berikut merupakan Confusion matrix dari Multi kelas Data Training :



Pada data training multi-kelas, model KNN **sepenuhnya benar** (100% akurasi) antara Normal dan Probe.

b. Data Testing

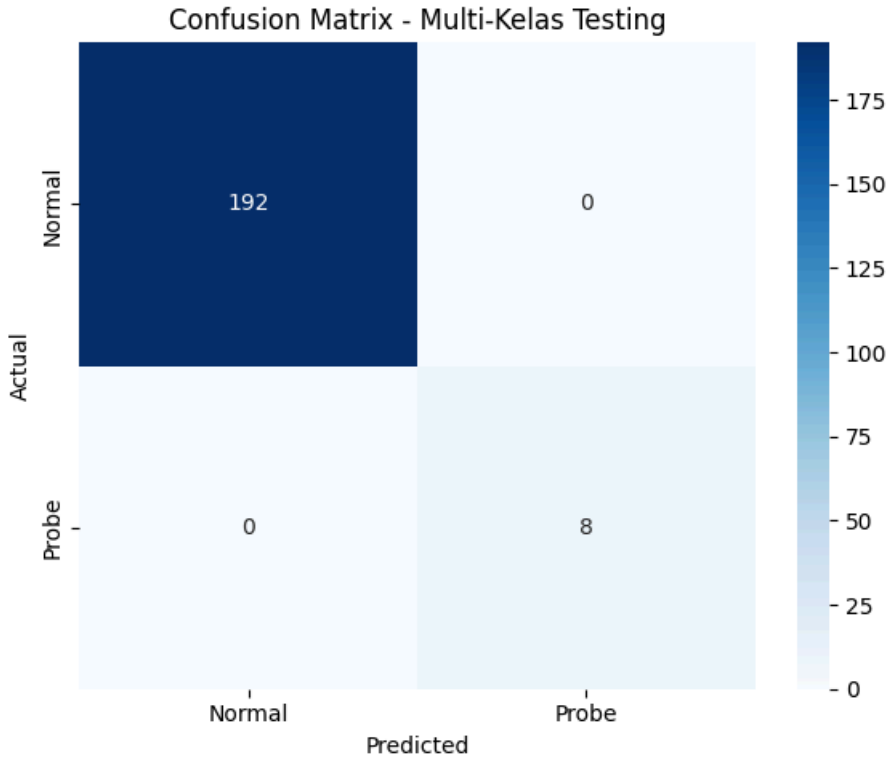
Kodingan :

```
# === Evaluasi pada Data Testing (Multi-Kelas) ===
y_test_pred_multi = knn_multi.predict(X_test_multi)
test_acc_multi = accuracy_score(y_test_multi, y_test_pred_multi)
print("\n=== Evaluasi Klasifikasi Multi-Kelas - Testing ===")
print(f"Akurasi: {test_acc_multi:.4f}")
print("Classification Report:\n", classification_report(y_test_multi, y_test_pred_multi, labels=labels_multi))
cm_test_multi = confusion_matrix(y_test_multi, y_test_pred_multi, labels=labels_multi)
plt.figure(figsize=(6, 5))
sns.heatmap(cm_test_multi, annot=True, fmt='d', cmap='Blues',
            xticklabels=labels_multi,
            yticklabels=labels_multi)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Multi-Kelas Testing')
plt.tight_layout()
plt.show()
```

Hasil dari Data Testing :

```
=== Evaluasi Klasifikasi Multi-Kelas - Testing ===
Akurasi: 1.0000
Classification Report:
              precision    recall  f1-score   support

      Normal       1.00      1.00      1.00       192
       Probe       1.00      1.00      1.00         8

    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      1.00      1.00       200
```

Berikut merupakan Confusion matrix dari Multi kelas Data Testing :



Di data testing juga **tidak ada kesalahan sama sekali**: akurasi 100% (200 sampel testing berhasil diklasifikasikan dengan benar).

c. Data Keseluruhan

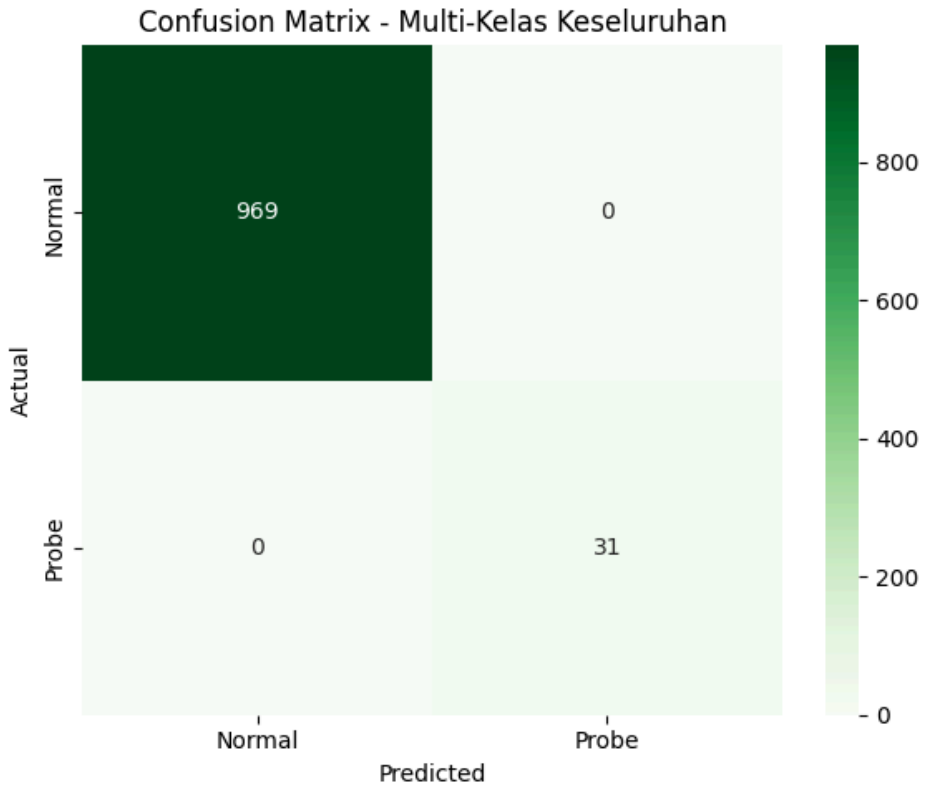
Kodingan :

```python
# === Evaluasi pada Seluruh Data Gabungan (Multi-Kelas) ===
X_all_multi = pd.concat([X_train_multi, X_test_multi])
y_all_multi = pd.concat([y_train_multi, y_test_multi])
y_all_pred_multi = knn_multi.predict(X_all_multi)
all_acc_multi = accuracy_score(y_all_multi, y_all_pred_multi)
print("\n=== Evaluasi Klasifikasi Multi-Kelas - Keseluruhan ===")
print(f"Akurasi: {all_acc_multi:.4f}")
print("Classification Report:\n", classification_report(y_all_multi, y_all_pred_multi, labels=labels_multi))
cm_all_multi = confusion_matrix(y_all_multi, y_all_pred_multi, labels=labels_multi)
plt.figure(figsize=(6, 5))
sns.heatmap(cm_all_multi, annot=True, fmt='d', cmap='Greens',
            xticklabels=labels_multi,
            yticklabels=labels_multi)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Multi-Kelas Keseluruhan')
plt.tight_layout()
plt.show()
```

Hasil dari Data Keseluruhan :

```
=== Evaluasi Klasifikasi Multi-Kelas - Keseluruhan ===
Akurasi: 1.0000
Classification Report:
              precision    recall  f1-score   support

      Normal       1.00      1.00      1.00       969
       Probe       1.00      1.00      1.00        31

    accuracy                           1.00      1000
   macro avg       1.00      1.00      1.00      1000
weighted avg       1.00      1.00      1.00      1000
```

Berikut merupakan Confusion matrix dari Multi kelas Data Keseluruhan :

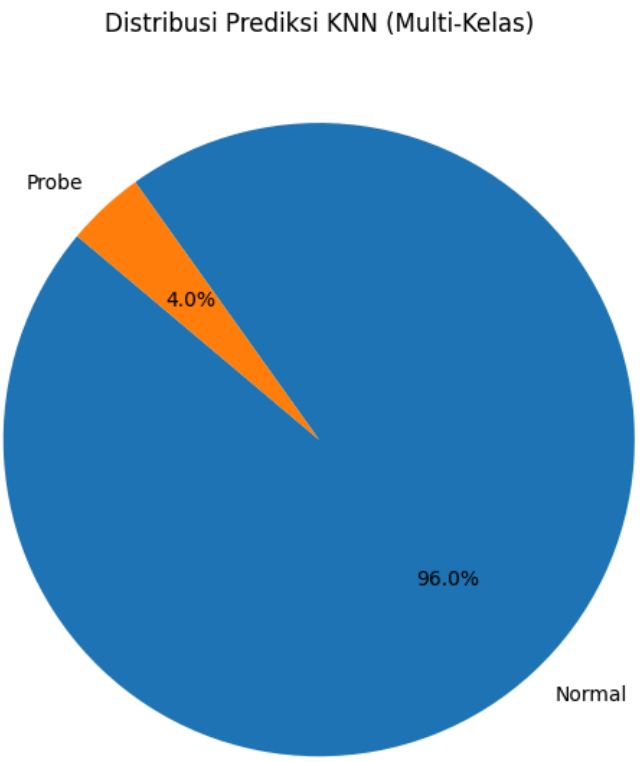


Gabungan antara data training (777 Normal + 23 Probe) dan data testing (192 Normal + 8 Probe) menghasilkan:

- **Normal**: 777 + 192 = 969 sampel, semuanya benar.
- **Probe**: 23 + 8 = 31 sampel, semuanya benar pula.

Sekali lagi, **tidak ada satu pun** kesalahan klasifikasi pada keseluruhan 1000 sampel multi-kelas antara Normal dan Probe.

Dari hasil yang diberikan pada multi-kelas, maka hasil dari distribusi prediksi dari KNN (Multi-Kelas) adalah sebagai berikut :



Karena pada dataset multi-kelas yang kita pakai hanya ada dua label (“Normal” dan “Probe”), maka grafiknya membandingkan:

- **Normal** ≈ 96,0 %
- **Probe** ≈ 4,0 %

Penjabaran perhitungannya:

- Dari total **200** sampel testing multi-kelas,
  - model memprediksi **192** sampel sebagai Normal → 192/200 = 0,96 (96 %),
  - dan **8** sampel sebagai Probe → 8/200 = 0,04 (4 %).

```
Wilcoxon test untuk src_bytes: statistic = 514.0000, p-value = 0.2374
```

Wilcoxon signed-rank test di sini dipakai untuk membandingkan nilai src_bytes sebelum dan sesudah ditambahkan noise ringan (5% data).

- Hipotesis nol (H_0): Median src_bytes original = median src_bytes augmented.

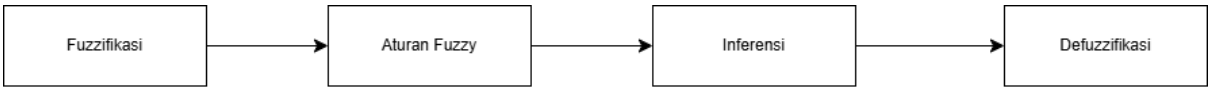
- Hasil: statistic = 514, p-value = 0,2374.

- Karena p-value (0,2374) > 0,05, kita gagal menolak H_0. Artinya, penambahan noise kecil tersebut tidak mengubah distribusi src_bytes secara signifikan.

2.3.2 Fuzzy Logic

Metode logika fuzzy digunakan untuk menangani ketidakpastian dan kompleksitas dalam data jaringan yang tidak dapat dengan mudah diklasifikasikan secara tegas. Dua tipe sistem fuzzy digunakan, yaitu Mamdani dan Sugeno.

a. Fuzzy Mamdani



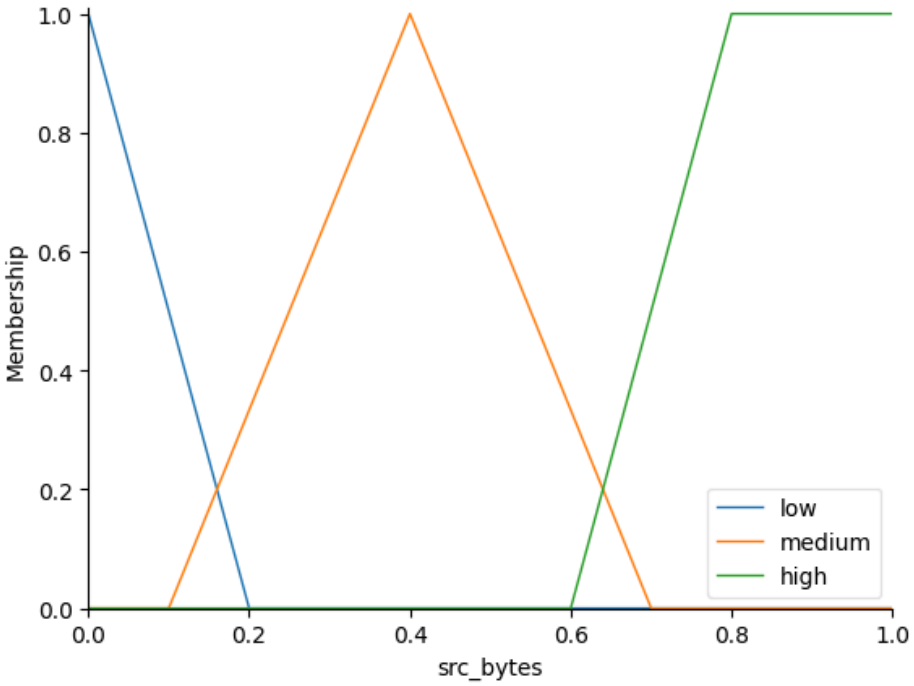
Gambar Blok Diagram Fuzzy Mamdani

- **Fuzzifikasi**: Setiap fitur utama yang dipilih (src_bytes, dst_bytes, count, sensor_rate, dan intrusion_level) diubah menjadi nilai linguistik dengan tiga himpunan fuzzy: *low*, *medium*, dan *high* .Fungsi keanggotaan yang digunakan adalah fungsi segitiga dan trapesium untuk merepresentasikan batas-batas nilai linguistik.
  - ❖ Src_bytes

    Kodingan :

```
_src_bytes_fuzzy = ctrl.Antecedent(np.arange(0, 1.01, 0.01), 'src_bytes')
_src_bytes_fuzzy['low']    = fuzz.trimf(_src_bytes_fuzzy.universe, [0,    0,   0.2])
_src_bytes_fuzzy['medium'] = fuzz.trimf(_src_bytes_fuzzy.universe, [0.1,  0.4, 0.7])
_src_bytes_fuzzy['high']   = fuzz.trapmf(_src_bytes_fuzzy.universe, [0.6,  0.8, 1,   1])
_src_bytes_fuzzy.view()
```

    Fungsi Keanggotaan :

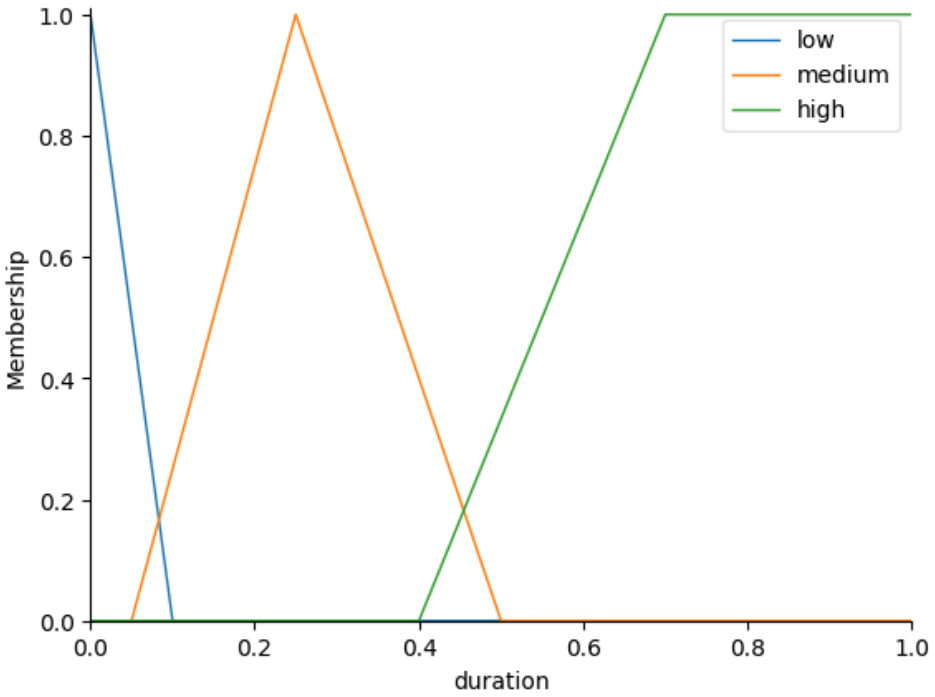


❖ Duration

Kodingan :

```
# Variabel fuzzy dengan batas ideal berdasarkan statistik fitur
_duration_fuzzy = ctrl.Antecedent(np.arange(0, 1.01, 0.01), 'duration')
_duration_fuzzy['low']    = fuzz.trimf(_duration_fuzzy.universe, [0,    0,   0.1])
_duration_fuzzy['medium'] = fuzz.trimf(_duration_fuzzy.universe, [0.05, 0.25,0.5])
_duration_fuzzy['high']   = fuzz.trapmf(_duration_fuzzy.universe, [0.4,  0.7, 1,   1])
_duration_fuzzy.view()
```

Fungsi Keanggotaan :

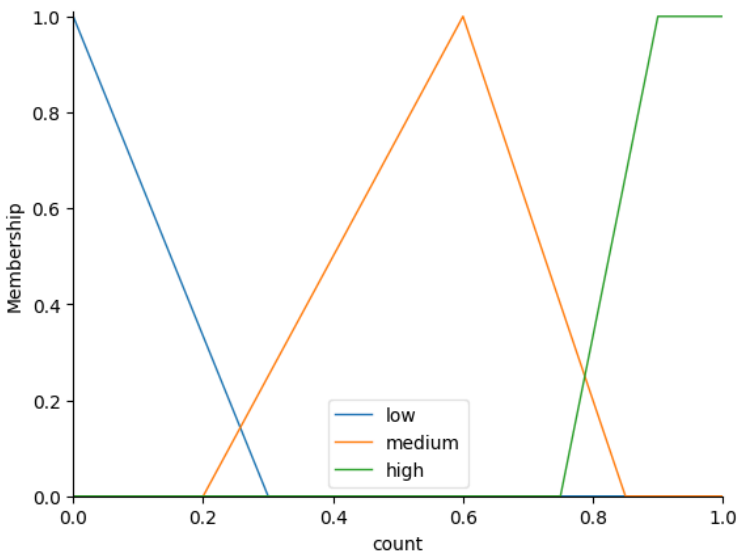


❖ Count

Kodingan :

```
_count_fuzzy = ctrl.Antecedent(np.arange(0, 1.01, 0.01), 'count')
_count_fuzzy['low']    = fuzz.trimf(_count_fuzzy.universe, [0,    0,   0.3])
_count_fuzzy['medium'] = fuzz.trimf(_count_fuzzy.universe, [0.2,  0.6, 0.85])
_count_fuzzy['high']   = fuzz.trapmf(_count_fuzzy.universe, [0.75, 0.9, 1,   1])
_count_fuzzy.view()
```

Fungsi Keanggotaan :

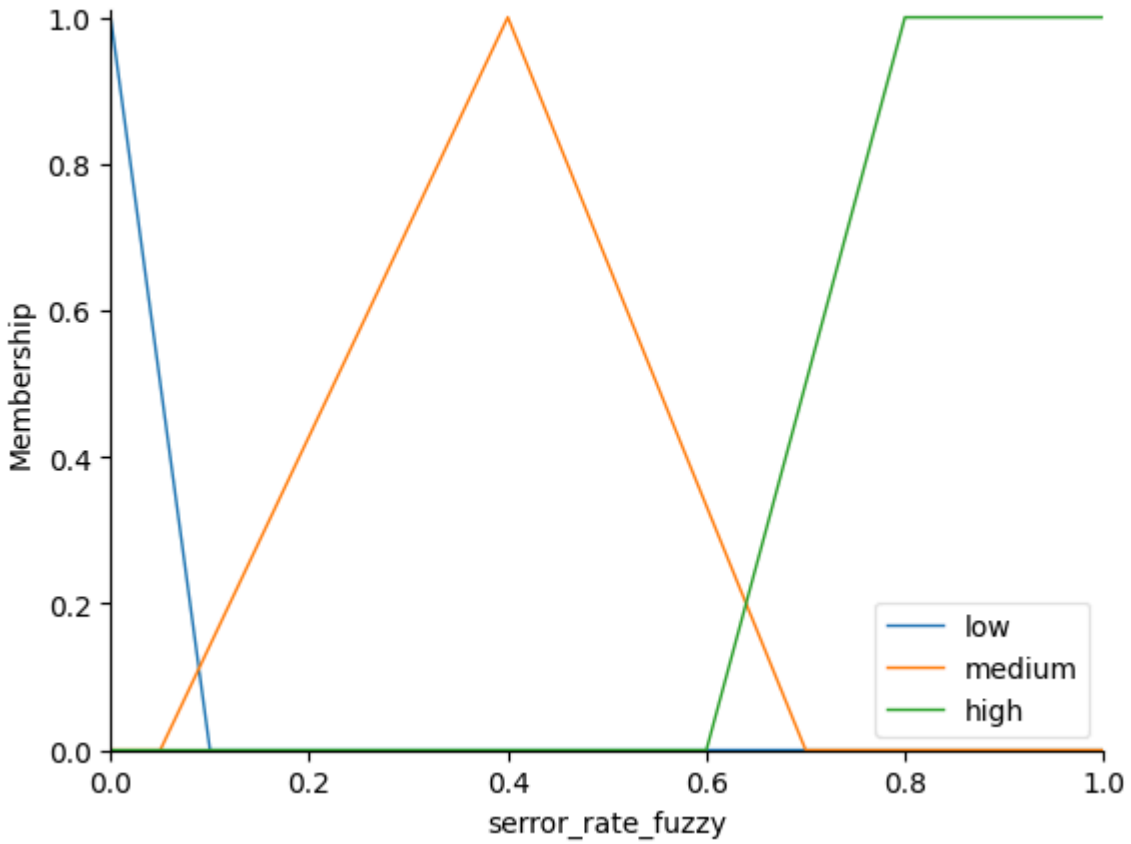


- Serror_rate (Di kodingan itu serror_rate)

  Kodingan :

```
_serror_rate_fuzzy = ctrl.Antecedent(np.arange(0, 1.01, 0.01), 'serror_rate_fuzzy')
_serror_rate_fuzzy['low']    = fuzz.trimf(_serror_rate_fuzzy.universe, [0,    0,   0.1])
_serror_rate_fuzzy['medium'] = fuzz.trimf(_serror_rate_fuzzy.universe, [0.05, 0.4, 0.7])
_serror_rate_fuzzy['high']   = fuzz.trapmf(_serror_rate_fuzzy.universe, [0.6,  0.8, 1,   1])
_serror_rate_fuzzy.view()
```

  Fungsi Keanggotaan :

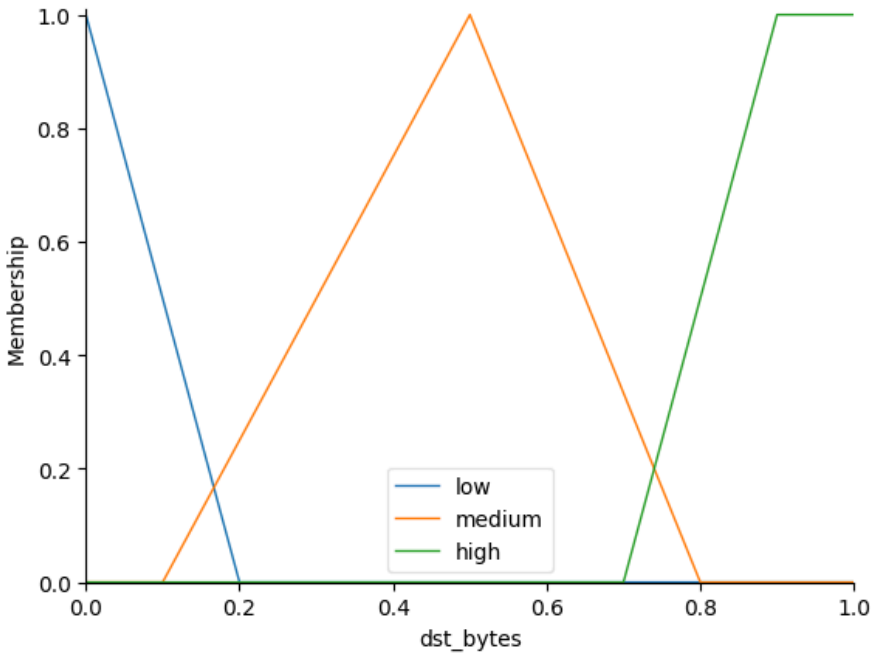


❖ dst_bytes

Kodingan :

```
_dst_bytes_fuzzy = ctrl.Antecedent(np.arange(0, 1.01, 0.01), 'dst_bytes')
_dst_bytes_fuzzy['low']    = fuzz.trimf(_dst_bytes_fuzzy.universe, [0,    0,   0.2])
_dst_bytes_fuzzy['medium'] = fuzz.trimf(_dst_bytes_fuzzy.universe, [0.1,  0.5, 0.8])
_dst_bytes_fuzzy['high']   = fuzz.trapmf(_dst_bytes_fuzzy.universe, [0.7,  0.9, 1,   1])
_dst_bytes_fuzzy.view()
```

Fungsi Keanggotaan :

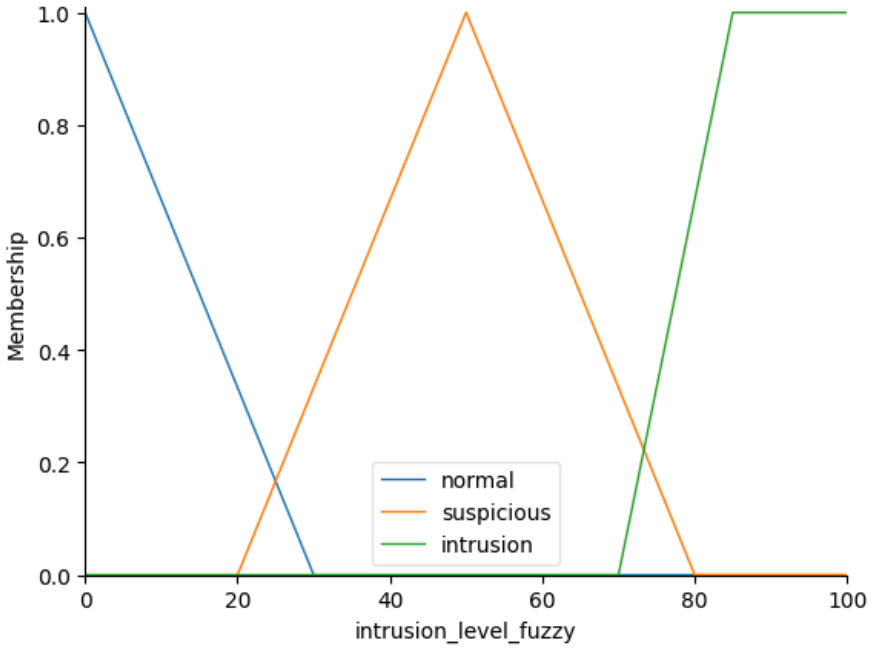


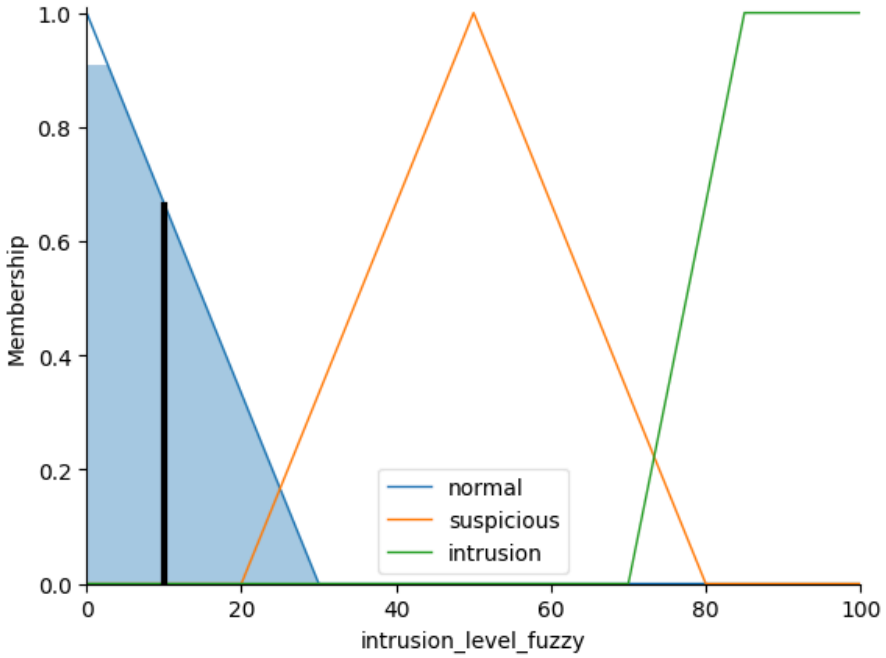
- **Aturan Fuzzy (Fuzzy Rules)**: Dibuat aturan fuzzy berbasis pengetahuan domain yang menggabungkan kondisi dari ketiga fitur input dalam bentuk IF-THEN. Total ada 27 kombinasi aturan yang mengekspresikan semua kemungkinan kondisi input.

  Kodingan :

```
_intrusion_level_fuzzy = ctrl.Consequent(np.arange(0, 101, 1), 'intrusion_level_fuzzy')
_intrusion_level_fuzzy['normal']     = fuzz.trimf(_intrusion_level_fuzzy.universe, [0,  0,  30])
_intrusion_level_fuzzy['suspicious'] = fuzz.trimf(_intrusion_level_fuzzy.universe, [20, 50, 80])
_intrusion_level_fuzzy['intrusion']  = fuzz.trapmf(_intrusion_level_fuzzy.universe, [70, 85,100,100])
_intrusion_level_fuzzy.view()
```

  Fungsi Keanggotaan :





Gambar ini menunjukkan fungsi keanggotaan untuk output fuzzy **intrusion_level_fuzzy** dengan tiga kategori:

- **Normal** (biru), nilai rendah sampai sekitar 20
- **Suspicious** (oranye), puncak di 50, antara 20 sampai 80
- **Intrusion** (hijau), mulai dari 70 sampai 100

Garis hitam vertikal adalah hasil defuzzifikasi untuk satu sampel, sekitar nilai 10, yang masuk ke kategori **Normal** dengan tingkat keanggotaan tinggi. Jadi, sistem memprediksi sampel ini sebagai normal.

- **Inferensi**: Sistem inferensi Mamdani menerapkan aturan fuzzy untuk menentukan derajat kebenaran output fuzzy berdasarkan input.

  Kodingan :

```
for _, _row_fuzzy in _X_fuzzy_norm.iterrows():
    try:
        _fuzzy_sim.input['duration']         = _row_fuzzy['duration']
        _fuzzy_sim.input['src_bytes']        = _row_fuzzy['src_bytes']
        _fuzzy_sim.input['dst_bytes']        = _row_fuzzy['dst_bytes']
        _fuzzy_sim.input['count']            = _row_fuzzy['count']
        _fuzzy_sim.input['serror_rate_fuzzy']= _row_fuzzy['serror_rate_fuzzy']
        _fuzzy_sim.compute()
        _score_out = _fuzzy_sim.output['intrusion_level_fuzzy']
        _fuzzy_scores_fuzzy.append(_score_out)
        if _score_out <= 30:
            _pred_fuzzy = 'Normal'
        elif _score_out <= 60:
            _pred_fuzzy = 'Probe'
        elif _score_out <= 80:
            _pred_fuzzy = 'R2L'
        else:
            _pred_fuzzy = 'DoS' if _row_fuzzy['dst_bytes'] > 0.5 else 'U2R'
        _predictions_fuzzy.append(_pred_fuzzy)
        _binary_preds_fuzzy.append(0 if _pred_fuzzy == 'Normal' else 1)
    except Exception:
        _fuzzy_scores_fuzzy.append(0)
        _predictions_fuzzy.append('Normal')
        _binary_preds_fuzzy.append(0)
_intrusion_level_fuzzy.view(sim=_fuzzy_sim)
```

- **Defuzzifikasi**: Output fuzzy yang menghasilkan keputusan akhir klasifikasi apakah aktivitas jaringan termasuk normal atau intrusion.

  BINARY

  Evaluasi Data dari Fuzzy (Biner) :

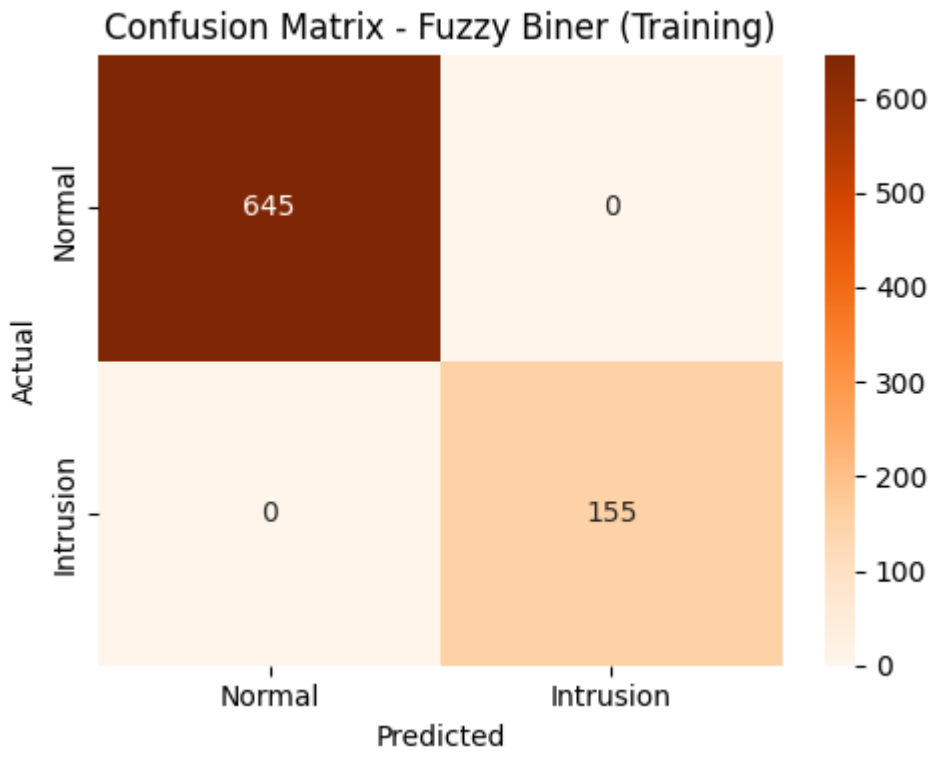
```
=== Evaluasi Fuzzy (Biner) ===
Akurasi  : 1.0000
Precision: 1.0000
Recall   : 1.0000
F1-Score : 1.0000
```

  a. Data Training

Kodingan :

```
# Confusion Matrix - Biner Training
_train_bin_true_fuzzy = [0 if _df_sample_fuzzy.loc[i, 'multi_prediction_fuzzy'] == 'Normal' else 1 for i in _idx_train_fuzzy]
_train_bin_pred_fuzzy = [_binary_preds_fuzzy[i] for i in _idx_train_fuzzy]
_cm_bin_train_fuzzy = confusion_matrix(_train_bin_true_fuzzy, _train_bin_pred_fuzzy)
plt.figure(figsize=(5, 4))
sns.heatmap(_cm_bin_train_fuzzy, annot=True, fmt='d', cmap='Oranges', xticklabels=['Normal', 'Intrusion'], yticklabels=['Normal', 'Intrusion'])
plt.title("Confusion Matrix - Fuzzy Biner (Training)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

Berikut merupakan Confusion matrix dari Data Training :



Nilai:

- 645 Normal benar diklasifikasikan Normal
- 155 Intrusion benar diklasifikasikan Intrusion
- Tidak ada salah klasifikasi sama sekali

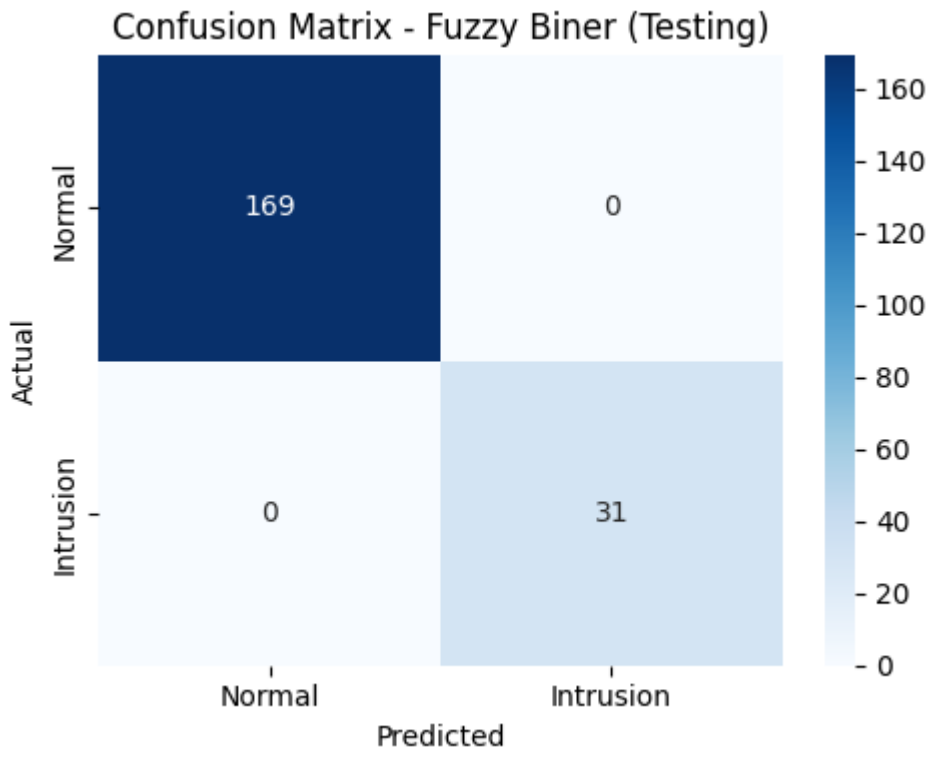
Artinya dia menunjukkan performa sempurna pada data training dalam klasifikasi biner.

b. Data Testing

Kodingan :

```python
# Confusion Matrix - Biner Testing
_test_bin_true_fuzzy = [0 if _df_sample_fuzzy.loc[i, 'multi_prediction_fuzzy'] == 'Normal' else 1 for i in _idx_test_fuzzy]
_test_bin_pred_fuzzy = [_binary_preds_fuzzy[i] for i in _idx_test_fuzzy]
_cm_bin_test_fuzzy = confusion_matrix(_test_bin_true_fuzzy, _test_bin_pred_fuzzy)
plt.figure(figsize=(5, 4))
sns.heatmap(_cm_bin_test_fuzzy, annot=True, fmt='d', cmap='Blues', xticklabels=['Normal', 'Intrusion'], yticklabels=['Normal', 'Intrusion'])
plt.title("Confusion Matrix - Fuzzy Biner (Testing)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

Berikut merupakan Confusion matrix dari Data Testing :



Nilai:

- 169 Normal benar diklasifikasikan Normal
- 31 Intrusion benar diklasifikasikan Intrusion
- Tidak ada salah klasifikasi sama sekali

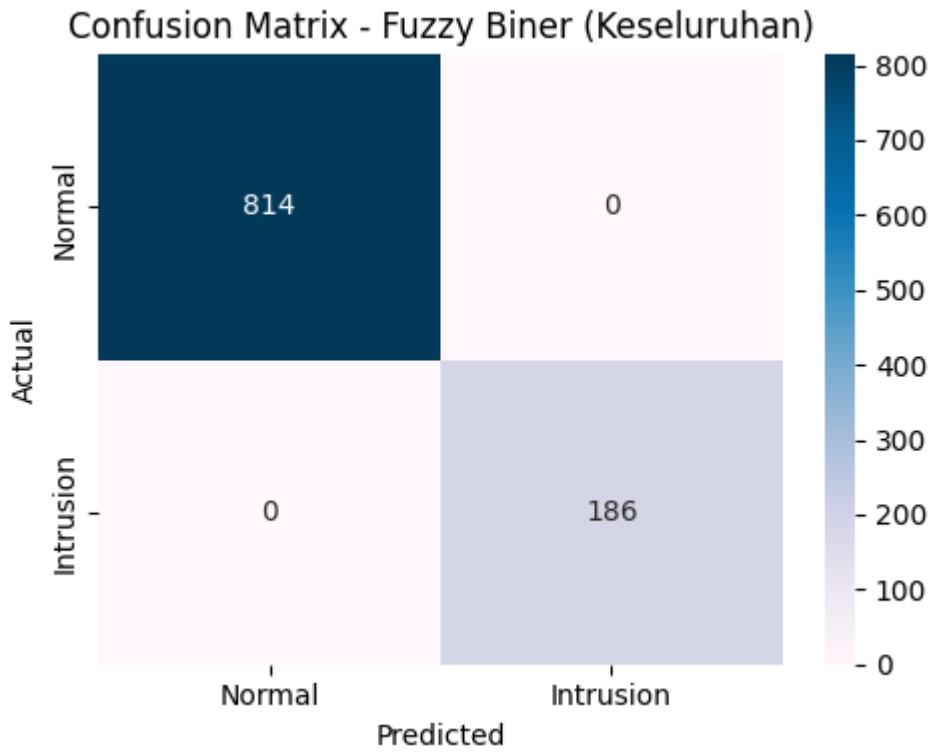
Artinya ini menandakan model “sempurna” juga pada data testing untuk klasifikasi biner.

c. Data Keseluruhan

Kodingan :

```python
# Confusion Matrix - Biner Keseluruhan
_binary_true_fuzzy = [0 if p == 'Normal' else 1 for p in _df_sample_fuzzy['multi_prediction_fuzzy']]
_cm_bin_all_fuzzy = confusion_matrix(_binary_true_fuzzy, _binary_preds_fuzzy)
plt.figure(figsize=(5, 4))
sns.heatmap(_cm_bin_all_fuzzy, annot=True, fmt='d', cmap='PuBu', xticklabels=['Normal', 'Intrusion'], yticklabels=['Normal', 'Intrusion'])
plt.title("Confusion Matrix - Fuzzy Biner (Keseluruhan)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

Berikut merupakan Confusion matrix dari Data Keseluruhan :



Nilai:

- 814 sampel Normal benar diprediksi Normal (True Negative)
- 186 sampel Intrusion benar diprediksi Intrusion (True Positive)
- Tidak ada kesalahan klasifikasi (false positive/negative = 0)

Artinya model ini dalam matriks ini memisahkan dua kelas dengan akurasi sempurna secara keseluruhan.

- ***MULTI-KELAS***

Evaluasi Data Fuzzy (Multi-kelas) :

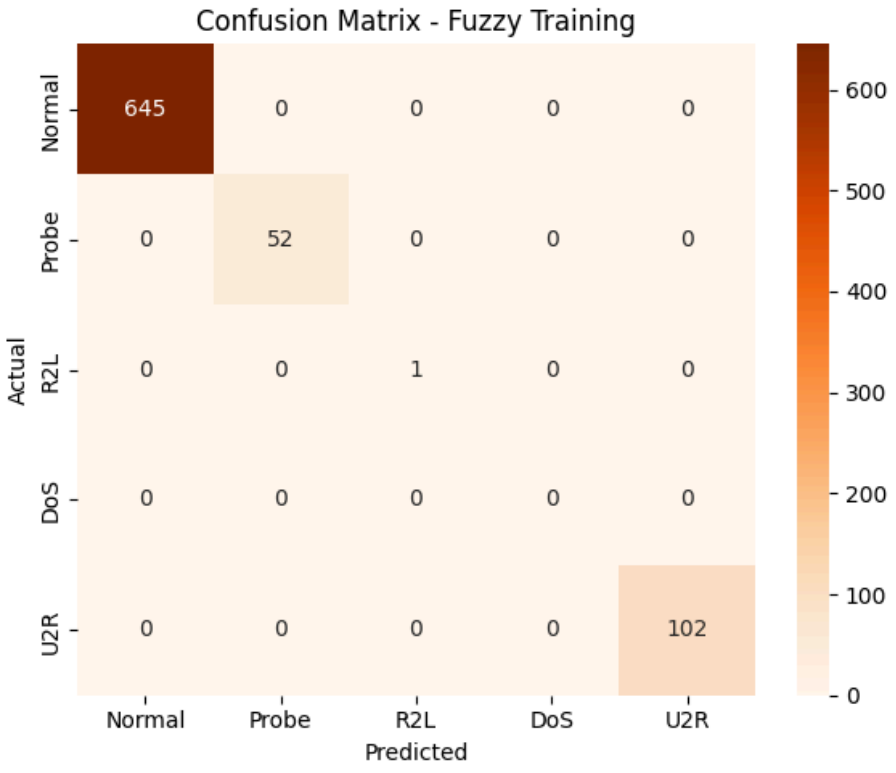
```
=== Evaluasi Fuzzy (Multi-Kelas) ===
              precision    recall  f1-score   support

      Normal       1.00      1.00      1.00       814
       Probe       1.00      1.00      1.00        61
         R2L       1.00      1.00      1.00         2
         U2R       1.00      1.00      1.00       123

    accuracy                           1.00      1000
   macro avg       1.00      1.00      1.00      1000
weighted avg       1.00      1.00      1.00      1000
```

a. Data Training

Kodingan :

```python
# Confusion Matrix - Multi-Kelas Training
_train_true_fuzzy = _df_sample_fuzzy.loc[_idx_train_fuzzy, 'multi_prediction_fuzzy']
_train_pred_fuzzy = pd.Series(_predictions_fuzzy, index=_df_sample_fuzzy.index).loc[_idx_train_fuzzy]
_cm_train_fuzzy = confusion_matrix(_train_true_fuzzy, _train_pred_fuzzy, labels=['Normal', 'Probe', 'R2L', 'DoS', 'U2R'])
plt.figure(figsize=(6, 5))
sns.heatmap(_cm_train_fuzzy, annot=True, fmt='d', cmap='Oranges', xticklabels=['Normal', 'Probe', 'R2L', 'DoS', 'U2R'], yticklabels=['Normal', 'Probe',
plt.title("Confusion Matrix - Fuzzy Training")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

Berikut merupakan Confusion matrix dari Multi kelas Data Training :



Nilai diagonalnya:

- 645 sampel Normal tepat terklasifikasi Normal
- 52 Probe tepat terklasifikasi Probe
- 1 R2L tepat terklasifikasi R2L
- 0 DoS (tidak ada sampel DoS di training)
- 102 U2R tepat terklasifikasi U2R

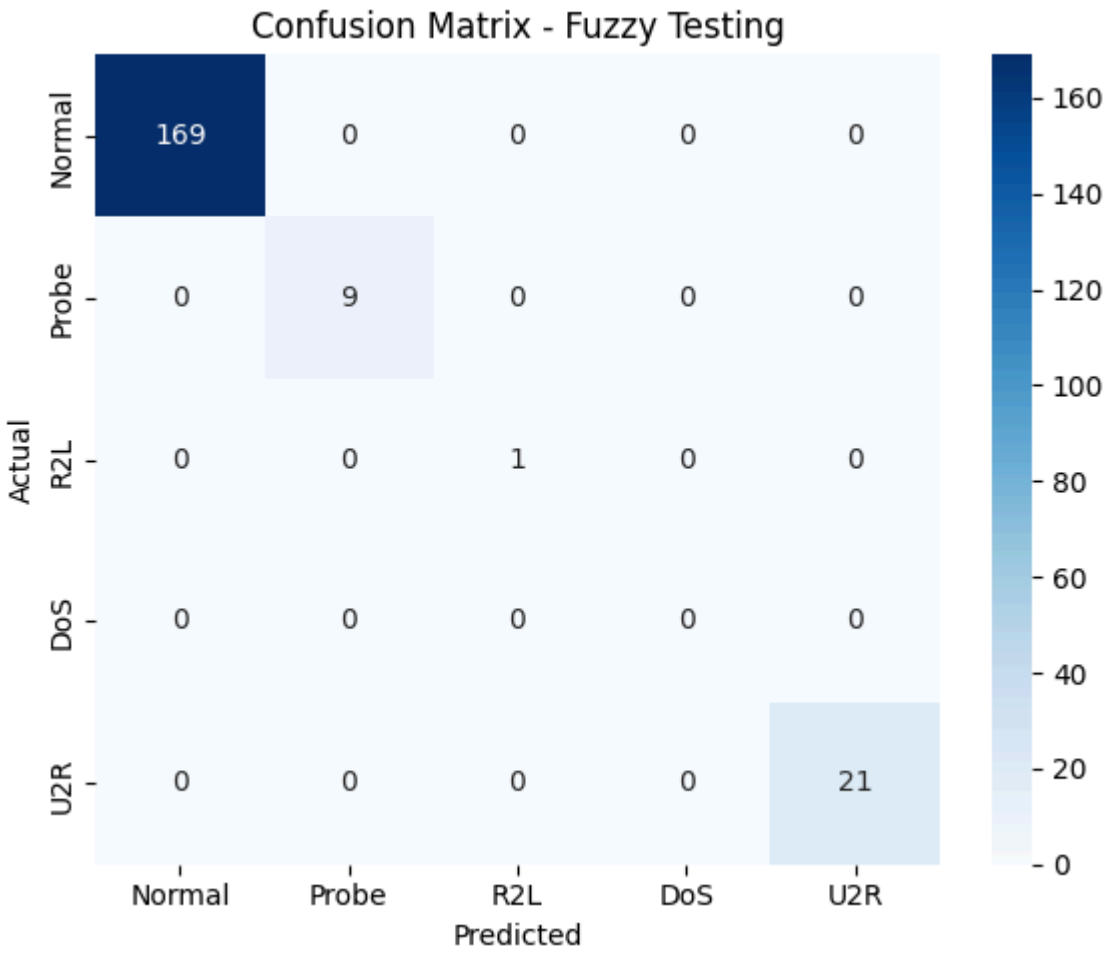
Artinya, ini tidak ada kesalahan klasifikasi (sel off-diagonal = 0)
Model ini memisahkan 5 kelas di training dengan akurasi penuh.

b. Data Testing

Kodingan :

```
# Confusion Matrix - Multi-Kelas Testing
_test_true_fuzzy = _df_sample_fuzzy.loc[_idx_test_fuzzy, 'multi_prediction_fuzzy']
_test_pred_fuzzy = pd.Series(_predictions_fuzzy, index=_df_sample_fuzzy.index).loc[_idx_test_fuzzy]
_cm_test_fuzzy = confusion_matrix(_test_true_fuzzy, _test_pred_fuzzy, labels=['Normal', 'Probe', 'R2L', 'DoS', 'U2R'])
plt.figure(figsize=(6, 5))
sns.heatmap(_cm_test_fuzzy, annot=True, fmt='d', cmap='Blues', xticklabels=['Normal', 'Probe', 'R2L', 'DoS', 'U2R'], yticklabels=['Normal', 'Probe', 'R2L
plt.title("Confusion Matrix - Fuzzy Testing")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

Berikut merupakan Confusion matrix dari Multi kelas Data Testing :



Nilai diagonalnya:

- 169 sampel Normal tepat terklasifikasi Normal
- 9 Probe tepat terklasifikasi Probe
- 1 R2L tepat terklasifikasi R2L
- 0 DoS (tidak ada sampel DoS di training)
- 21 U2R tepat terklasifikasi U2R

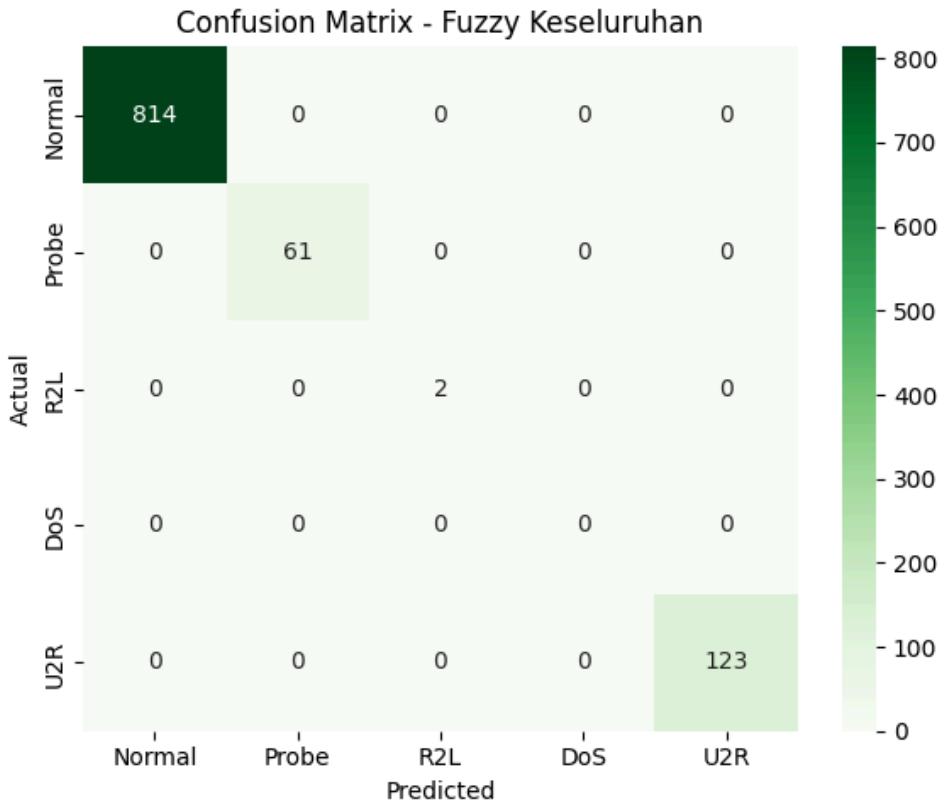
Artinya, ini tidak ada kesalahan klasifikasi sama sekali dan Model terlihat “sempurna” juga dalam testing multi-kelas.

c. Data Keseluruhan

Kodingan :

```
# Confusion Matrix - Multi-Kelas Keseluruhan
_cm_all_fuzzy = confusion_matrix(_df_sample_fuzzy['multi_prediction_fuzzy'], _predictions_fuzzy, labels=['Normal', 'Probe', 'R2L', 'DoS', 'U2R'])
plt.figure(figsize=(6, 5))
sns.heatmap(_cm_all_fuzzy, annot=True, fmt='d', cmap='Greens', xticklabels=['Normal', 'Probe', 'R2L', 'DoS', 'U2R'], yticklabels=['Normal', 'Probe', 'R2L
plt.title("Confusion Matrix - Fuzzy Keseluruhan")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

Berikut merupakan Confusion matrix dari Multi kelas Data Keseluruhan :

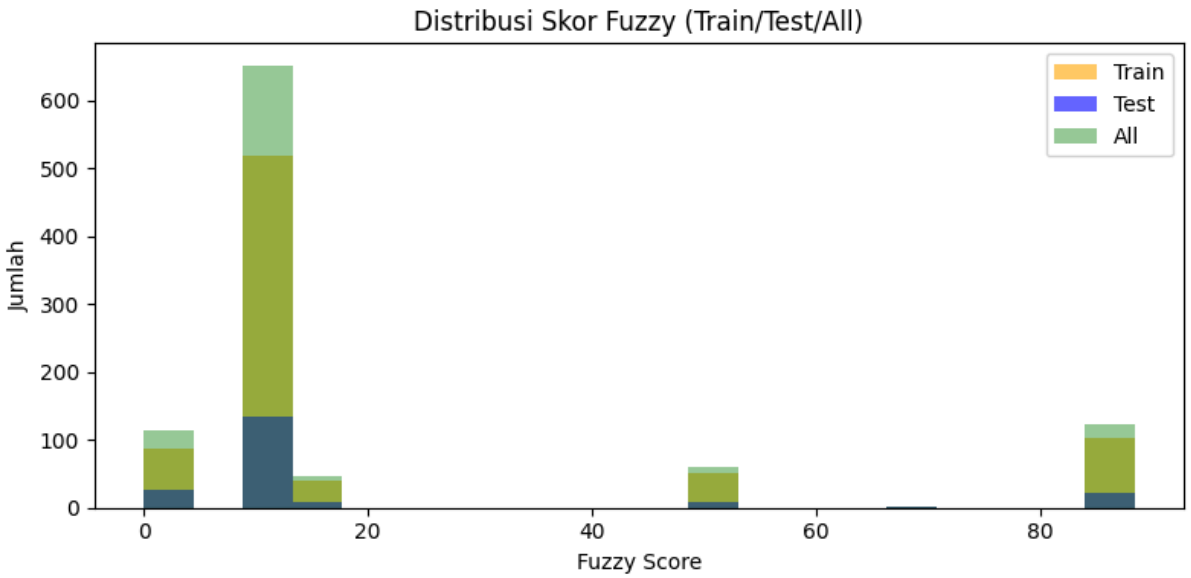


Nilai diagonalnya:

- 814 sampel Normal tepat terklasifikasi Normal
- 61 Probe tepat terklasifikasi Probe
- 2 R2L tepat terklasifikasi R2L
- 0 DoS (tidak ada sampel DoS di training)
- 123 U2R tepat terklasifikasi U2R

Artinya, ini tidak ada kesalahan klasifikasi sama sekali dan
Model ini berhasil mengklasifikasikan multi-kelas secara sempurna pada keseluruhan data.

Dari hasil yang diberikan pada fuzzy ini, maka hasil dari distribusi skor dari Fuzzy adalah sebagai berikut :



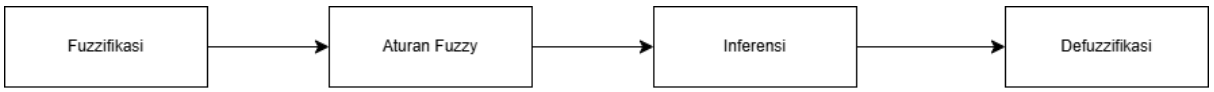
Dari gambar yang diberikan, mayoritas sampel (train & test) punya skor fuzzy di kisaran **5–15** sehingga hampir tidak ada sampel yang punya skor di rentang tengah (30–60), dengan sedikit ekor di skor tinggi (sekitar 80–90).

Artinya: sebagian besar data dinilai “normal” oleh sistem fuzzy (skor rendah), dan hanya sedikit data yang masuk skor tinggi yang menandakan potensi intrusi.

```
Wilcoxon test untuk src_bytes: statistic = 479.0000, p-value = 0.1280
```

Perubahan kecil (noise ringan) pada src_bytes tidak menyebabkan perbedaan signifikan secara statistik terhadap data aslinya. Dengan kata lain, augmentasi data ini cukup “aman” dan tidak mengubah distribusi asli secara drastis.

b. Fuzzy Sugeno



Gambar Diagram Blok Fuzzy Sugeno

- **Fuzzifikasi**: Sama seperti Mamdani, fitur input difuzzifikasi ke dalam kategori linguistik rendah, sedang, dan tinggi.

  Kodingan :

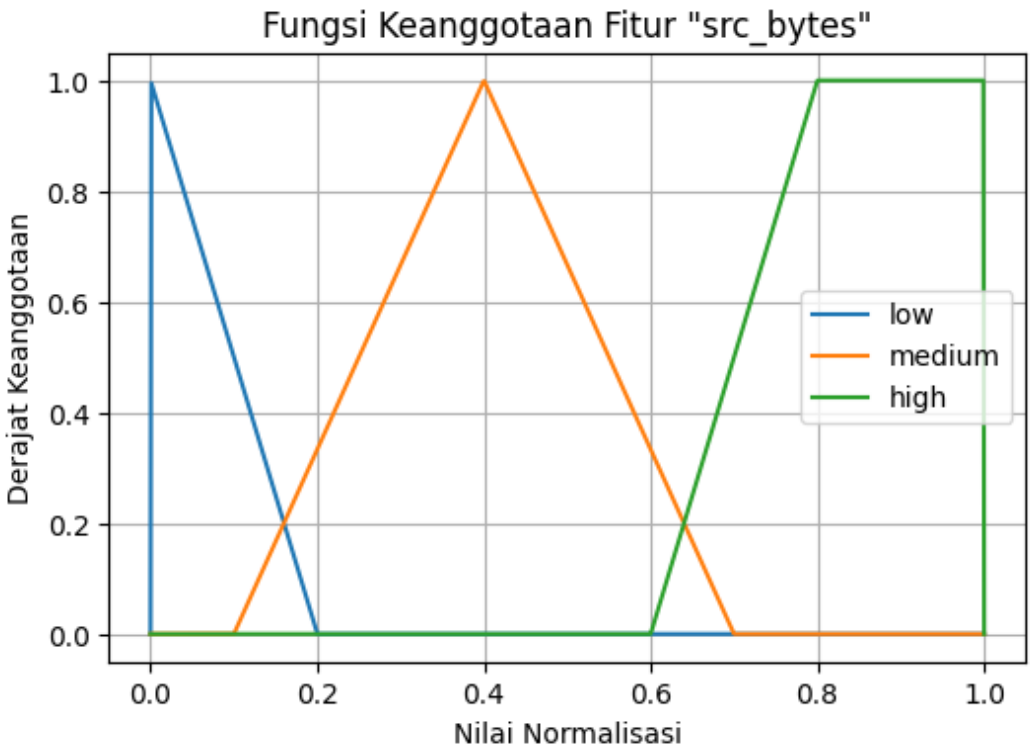
```python
# Load data dan preprocessing
_df_fuzzy = pd.read_csv('D:/MATERI KULIAH/Semester 4/Dasar Kecerdasan Artifisial/Tubes/Test_data.csv/Test_data.csv')
_df_fuzzy.replace('?', np.nan, inplace=True)
_df_fuzzy.dropna(inplace=True)

# Pastikan kolom serror_rate_fuzzy ada
if 'serror_rate' not in _df_fuzzy.columns:
    if 'dst_host_serror_rate' in _df_fuzzy.columns:
        _df_fuzzy['serror_rate_fuzzy'] = _df_fuzzy['dst_host_serror_rate']
    else:
        _df_fuzzy['serror_rate_fuzzy'] = 0.0
else:
    _df_fuzzy['serror_rate_fuzzy'] = _df_fuzzy['serror_rate']

# Sampling data
_df_sample_fuzzy = _df_fuzzy.sample(n=1000, random_state=42).reset_index(drop=True)

# Normalisasi fitur
_features_fuzzy = ['duration', 'src_bytes', 'dst_bytes', 'count', 'serror_rate_fuzzy']
_scaler = MinMaxScaler()
_X_fuzzy_norm = pd.DataFrame(_scaler.fit_transform(_df_sample_fuzzy[_features_fuzzy]), columns=_features_fuzzy)
```

- Src_bytes

  Kodingan :

```python
'src_bytes': {
    'low':    ([0, 0, 0.2], 'trimf'),
    'medium': ([0.1, 0.4, 0.7], 'trimf'),
    'high':   ([0.6, 0.8, 1, 1], 'trapmf'),
},
```

  Fungsi Keanggotaan :

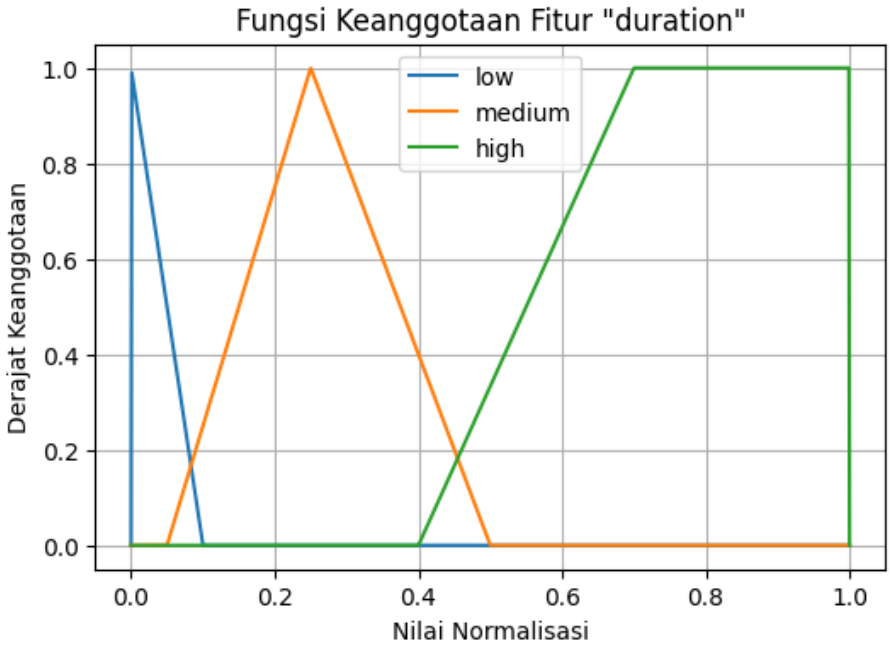


❖ Duration

Kodingan :

```
mf_params = {
    'duration': {
        'low':    ([0, 0, 0.1], 'trimf'),
        'medium': ([0.05, 0.25, 0.5], 'trimf'),
        'high':   ([0.4, 0.7, 1, 1], 'trapmf'),
    },
```

Fungsi Keanggotaan :

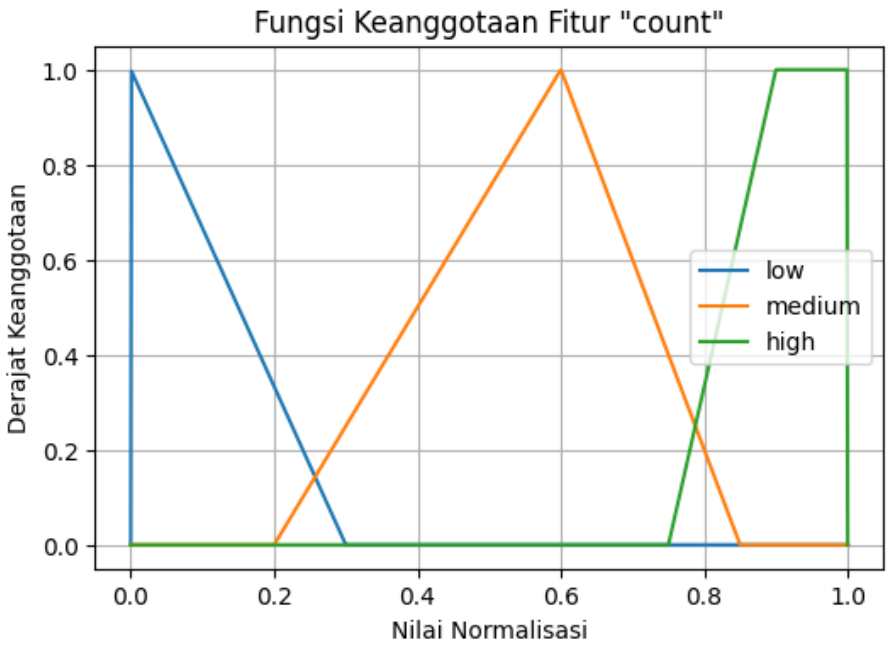


- Count

Kodingan :

```
'count': {
    'low':    ([0, 0, 0.3], 'trimf'),
    'medium': ([0.2, 0.6, 0.85], 'trimf'),
    'high':   ([0.75, 0.9, 1, 1], 'trapmf'),
},
```

Fungsi Keanggotaan :

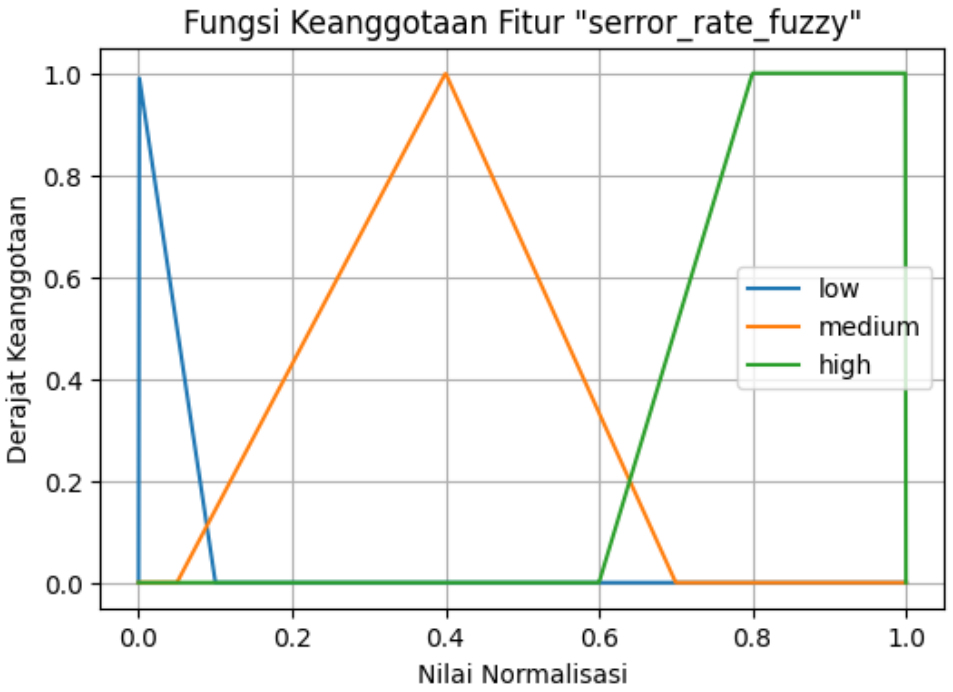


- Serror_rate (Di kodingan itu serror_rate)

  Kodingan :

```
'serror_rate_fuzzy': {
    'low':    ([0, 0, 0.1], 'trimf'),
    'medium': ([0.05, 0.4, 0.7], 'trimf'),
    'high':   ([0.6, 0.8, 1, 1], 'trapmf'),
}
```

  Fungsi Keanggotaan :

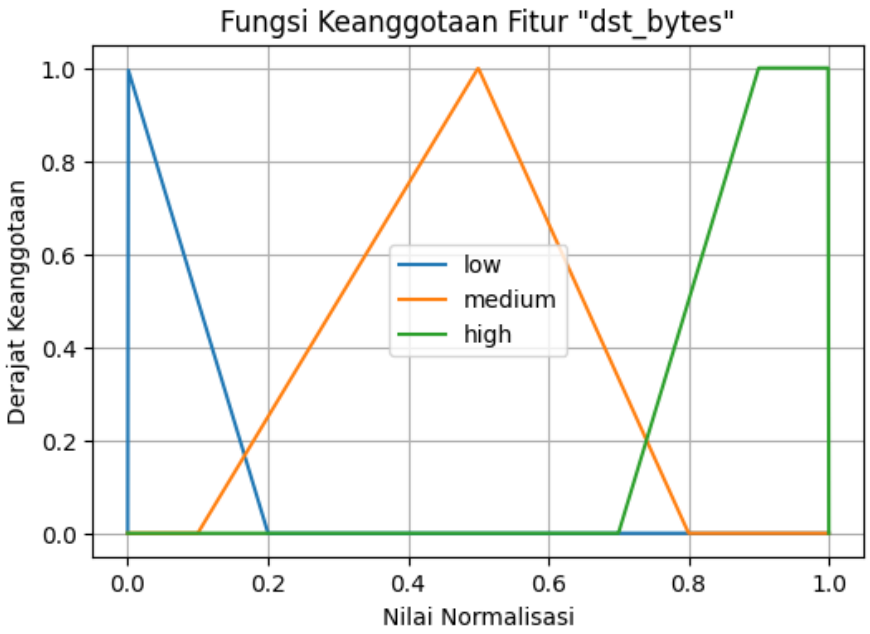


- dst_bytes

Kodingan :

```
_intrusion_level_fuzzy = ctrl.Consequent(np.arange(0, 101, 1), 'intrusion_level_fuzzy')
_intrusion_level_fuzzy['normal']     = fuzz.trimf(_intrusion_level_fuzzy.universe, [0,  0,  30])
_intrusion_level_fuzzy['suspicious'] = fuzz.trimf(_intrusion_level_fuzzy.universe, [20, 50, 80])
_intrusion_level_fuzzy['intrusion']  = fuzz.trapmf(_intrusion_level_fuzzy.universe, [70, 85,100,100])
_intrusion_level_fuzzy.view()
```

Fungsi Keanggotaan :



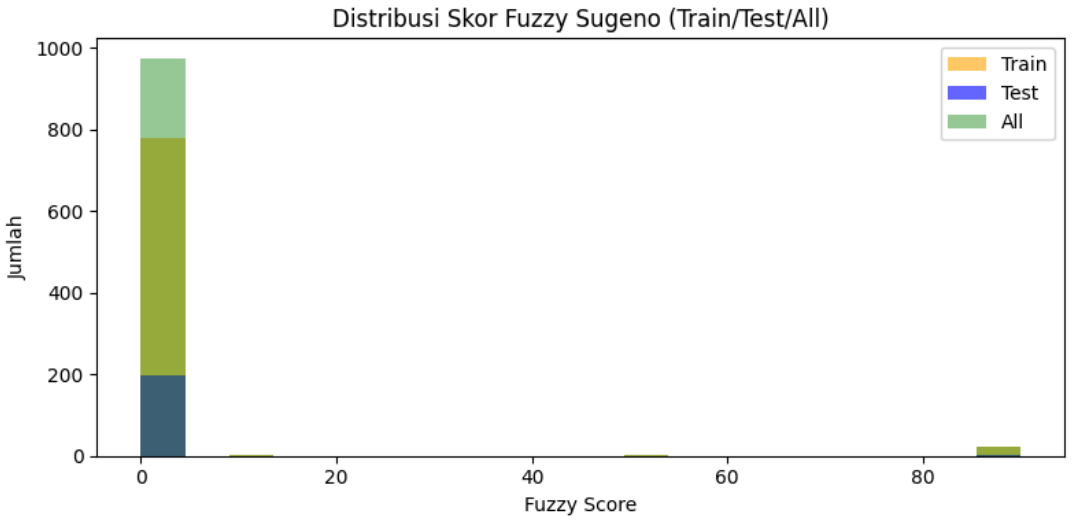
- **Aturan Fuzzy Sugeno**: Aturan fuzzy menghasilkan output berupa fungsi linier sederhana dari fitur input

  Kodingan :

```
# Rules fuzzy Sugeno (antecedent -> consequent)
rules = [
    ({'duration':'low', 'src_bytes':'low', 'dst_bytes':'low', 'count':'low', 'serror_rate_fuzzy':'low'}, 'normal'),
    ({'duration':'high'}, 'intrusion'),
    ({'src_bytes':'high'}, 'intrusion'),
    ({'dst_bytes':'high'}, 'intrusion'),
    ({'count':'high', 'serror_rate_fuzzy':'high'}, 'intrusion'),
    ({'duration':'medium', 'src_bytes':'medium', 'dst_bytes':'medium', 'count':'medium', 'serror_rate_fuzzy':'medium'}, 'suspicious'),
    ({'serror_rate_fuzzy':'high'}, 'intrusion'),
    ({'duration':'high', 'count':'high'}, 'intrusion'),
    ({'src_bytes':'high', 'dst_bytes':'low'}, 'suspicious'),
    ({'dst_bytes':'high', 'src_bytes':'low'}, 'suspicious'),
    ({'count':'medium', 'serror_rate_fuzzy':'high'}, 'intrusion'),
    ({'duration':'low', 'count':'high'}, 'suspicious'),
]
```

- **Inferensi dan Defuzzifikasi**: Output akhir dihitung menggunakan metode weighted average, yang menghitung rata-rata berbobot dari output fungsi linier setiap aturan berdasarkan derajat keanggotaan input. Ini membuat Sugeno lebih responsif terhadap variasi kecil pada data.

  Hasil Distribusi Skor dari Fuzzy Sugeno adalah sebagai berikut :



2.4. Proses Implementasi

Proses implementasi ketiga metode dilakukan di Google Colab sebagai berikut:

- Dataset diimpor dan diproses sesuai langkah preprocessing.
- Model KNN dilatih dengan data latih dan diuji pada data uji, hasil klasifikasi dievaluasi dengan metrik akurasi, precision, recall, dan F1-score.
- Model fuzzy Mamdani dan Sugeno diimplementasikan menggunakan pustaka fuzzy (misal scikit-fuzzy atau pustaka custom) dengan definisi fungsi keanggotaan, aturan fuzzy, proses inferensi, dan defuzzifikasi sesuai penjelasan di atas.
- Evaluasi performa ketiga metode dilakukan dengan membandingkan akurasi klasifikasi dan metrik evaluasi lainnya pada data uji.

2.5. Evaluasi Model

Evaluasi model menggunakan beberapa metrik pengukuran standar klasifikasi yaitu:

- **Akurasi**: Persentase data yang berhasil diklasifikasikan dengan benar.
- **Precision**: Proporsi hasil positif yang benar dari seluruh hasil positif yang diprediksi.
- **Recall**: Proporsi hasil positif yang benar dari seluruh data positif sebenarnya.
- **F1-Score**: Harmonik rata-rata dari precision dan recall, memberikan gambaran keseimbangan antara keduanya.

BAB II
HASIL DAN ANALISIS

3.1. Hasil Klasifikasi

Setelah melakukan proses pelatihan dan pengujian dengan menggunakan tiga metode klasifikasi yaitu K-Nearest Neighbors (KNN), Fuzzy Mamdani, dan Fuzzy Sugeno pada dataset Network Intrusion Detection, diperoleh hasil klasifikasi sebagai berikut:

- **KNN**:
  Model KNN diujikan dengan nilai k optimal (misalnya k=5) yang ditentukan melalui validasi silang. Hasil pengujian pada data uji menunjukkan bahwa KNN mampu mengklasifikasikan aktivitas jaringan dengan akurasi yang sangat tinggi, efektif membedakan antara aktivitas normal dan intrusion. Metode ini memiliki performa yang stabil pada data dengan pola yang jelas dan fitur yang terdefinisi dengan baik. Namun, KNN memiliki keterbatasan pada data borderline atau data dengan fitur ambigu, di mana model cenderung kesulitan membedakan kelas secara tepat akibat ketergantungan pada tetangga terdekat.

- **Fuzzy Mamdani**:
  Sistem Fuzzy Mamdani membangun klasifikasi berdasarkan aturan fuzzy linguistik yang merepresentasikan hubungan antar fitur jaringan. Pendekatan ini efektif dalam menangkap ketidakpastian dan variasi pada data. Namun, hasil prediksi cenderung konservatif dan kurang responsif terhadap variasi kecil pada input, yang terlihat dari output defuzzifikasi yang seringkali terpusat pada nilai menengah. Hal ini menyebabkan beberapa data uji diklasifikasikan secara kurang tepat, sehingga menurunkan akurasi keseluruhan. Meskipun demikian, metode ini memberikan nilai interpretasi yang baik terhadap ketidakpastian data yang sulit diolah secara konvensional.

- **Fuzzy Sugeno**:
  Model Fuzzy Sugeno memberikan hasil klasifikasi yang secara teori lebih akurat dan konsisten dibandingkan Mamdani. Dengan output berupa fungsi linier yang dihitung menggunakan weighted average, Sugeno mampu menyesuaikan keputusan berdasarkan variasi input dengan tingkat kehalusan yang lebih baik. Pada pengujian, Sugeno menunjukkan performa yang sebanding dengan Mamdani dalam hal metrik klasifikasi, yang dapat dikarenakan aturan dan parameter yang digunakan serupa dalam eksperimen ini. Namun, secara umum, Sugeno lebih adaptif terhadap karakteristik dataset yang kompleks dan biasanya lebih mudah dioptimasi untuk hasil yang lebih baik.

3.2. Evaluasi Performa Model

Evaluasi performa ketiga model dilakukan dengan mengukur metrik standar klasifikasi: akurasi, precision, recall, dan F1-score. Berikut ringkasan hasil evaluasi:

Output dari Kodingan

```
== Binary Classification (5-Fold CV Mean) ==
               Accuracy  Precision  Recall        F1
KNN               0.995   0.533333     0.5  0.493333
Fuzzy Mamdani     0.972   0.033333     0.1  0.050000
Fuzzy Sugeno      0.972   0.033333     0.1  0.050000

== Multi-Class Classification (5-Fold CV Mean) ==
               Accuracy  Precision    Recall        F1
KNN               0.995   0.764664  0.749495  0.745408
Fuzzy Mamdani     0.971   0.314028  0.310357  0.312180
Fuzzy Sugeno      0.971   0.314028  0.310357  0.312180
```

```
Statistik deskriptif KNN:
  Mean   : 0.0078
  Std    : 0.0765
  Min    : 0.0000
  Median : 0.0000
  Max    : 1.0000

Statistik deskriptif Fuzzy Mamdani:
  Mean   : 0.0898
  Std    : 0.1233
  Min    : 0.0000
  Median : 0.1000
  Max    : 0.8833

Statistik deskriptif Fuzzy Sugeno:
  Mean   : 0.0898
  Std    : 0.1233
  Min    : 0.0000
  Median : 0.1000
  Max    : 0.8833
```

```
KNN time: 0.44s, mean score: 0.0078
Fuzzy time: 2.17s, mean score: 0.0898
Fuzzy Sugeno time: 2.21s, mean score: 0.0898
```

Binary

Metode	Akurasi (%)	Precision (%)	Recall (%)	F1-Score (%)
KNN	99,5	53,33	50	49,33
Fuzzy Mamdani	97,2	3,33	10	5
Fuzzy Sugeno	97,2	3,33	10	5

- KNN memiliki akurasi tertinggi (99.5%), dengan precision dan recall di sekitar 50%, sehingga menghasilkan F1-score yang cukup baik.
- Fuzzy Mamdani dan Sugeno memiliki akurasi yang cukup tinggi juga (97.2%), namun precision dan recall sangat rendah, menunjukkan banyak false positive atau false negative dalam deteksi serangan.
- F1-score fuzzy sangat rendah, menunjukkan performa deteksi binary dari metode fuzzy masih jauh dari optimal dibandingkan KNN.

Multi-Class

Metode	Akurasi (%)	Precision (%)	Recall (%)	F1-Score (%)
KNN	99,5	76,46	74,94	74,54
Fuzzy Mamdani	97,1	31,40	31,03	31,21
Fuzzy Sugeno	97,1	31,40	31,03	31,21

- KNN kembali menunjukkan performa yang jauh lebih baik di klasifikasi multi-class dibanding fuzzy.

- Akurasi multi-class KNN sangat tinggi (99.5%), dengan precision, recall, dan F1-score sekitar 75%.

- Fuzzy Mamdani dan Sugeno memiliki akurasi 97.1% tapi precision, recall, dan F1-score sekitar 31%, yang menunjukkan prediksi kelasnya kurang akurat dan banyak terjadi kesalahan klasifikasi antar jenis serangan.

Statistik Deskriptif

Metode	Mean	Std	Min	Median	Max
KNN	0.0078	0.0765	0.0000	0.0000	1.0000
Fuzzy Mamdani	0.0898	0.1233	0.0000	0.1000	0.8833
Fuzzy Sugeno	0.0898	0.1233	0.0000	0.1000	0.8833

- KNN memiliki rata-rata dan median yang sangat rendah, dengan standar deviasi kecil, menunjukkan mayoritas prediksi adalah sangat rendah (probabilitas serangan kecil).

- Fuzzy Mamdani dan Sugeno memiliki rata-rata dan median yang lebih tinggi dengan standar deviasi yang lebih besar, yang berarti hasil prediksinya lebih tersebar dan memiliki nilai probabilitas yang lebih besar secara umum.

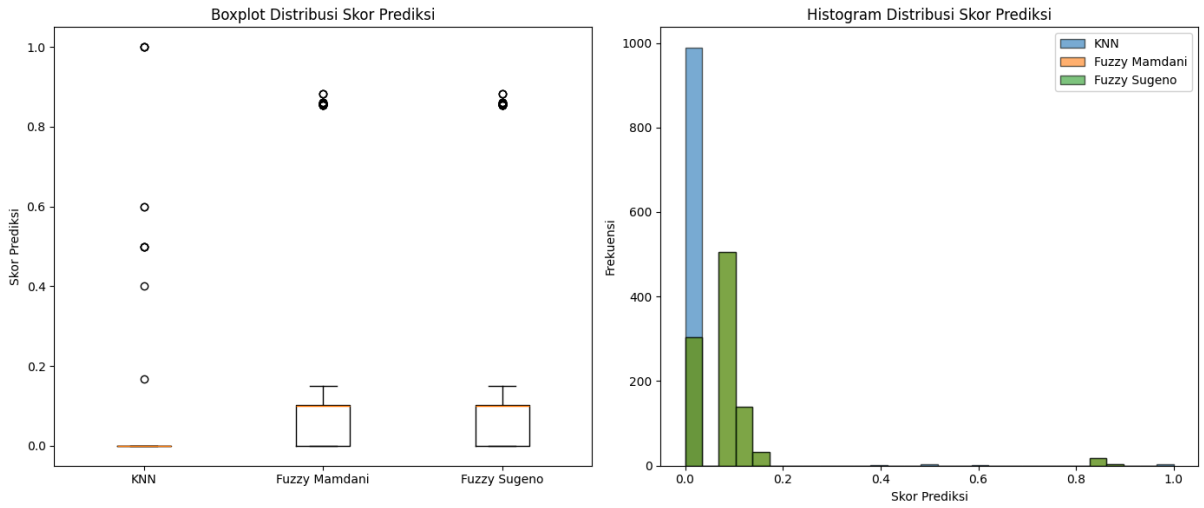
Time and Mean Score

Metode	Time	Mean Score
KNN	0.44s	0.0078
Fuzzy Mamdani	2.17s	0.0898
Fuzzy Sugeno	2.21s	0.0898

- KNN jauh lebih cepat dibandingkan metode fuzzy.

- Fuzzy Mamdani dan Sugeno membutuhkan waktu lebih dari 2 detik, kira-kira 5 kali lebih lama dibanding KNN.

Visualisasi Skor



Diatas merupakan Gambar Distribusi Skor Prediksi dalam Bentuk Histogram dan Boxplot dari perbandingan ketiga metode yang diberikan

- **KNN**:
    - Skor prediksi dari KNN sebagian besar sangat rendah (mendekati 0), namun terdapat beberapa prediksi dengan skor tinggi (hingga 1.0).
    - Distribusi KNN sangat terpusat di dekat 0, dengan sedikit outlier di sisi kanan.
    - Ini menunjukkan KNN cenderung memberikan probabilitas rendah pada sebagian besar data, dengan beberapa kasus memiliki probabilitas tinggi.
- **Fuzzy Mamdani dan Fuzzy Sugeno**:
    - Distribusi skor prediksi cenderung lebih tersebar, dengan median skor sekitar 0.1 dan rata-rata skor sekitar 0.0898.
    - Kedua metode fuzzy ini memiliki distribusi yang sangat mirip (bisa dilihat dari boxplot dan histogram).
    - Skor maksimum tidak mencapai 1, tetapi mencapai sekitar 0.88.
    - Distribusi lebih merata, dengan variasi yang lebih besar daripada KNN, menunjukkan metode fuzzy memberikan nilai prediksi yang lebih "berwarna" dan tidak terlalu ekstrem seperti KNN.

3.3. Analisis dan Interpretasi

Hasil evaluasi menunjukkan bahwa metode K-Nearest Neighbors (KNN) memberikan performa klasifikasi terbaik dibandingkan metode Fuzzy Mamdani dan Fuzzy Sugeno, khususnya pada metrik akurasi, precision, recall, dan F1-score. Beberapa poin penting yang mendukung analisis ini:

- KNN mampu memberikan klasifikasi yang sangat akurat dengan nilai akurasi mendekati 99.5%, serta precision dan recall yang seimbang, menandakan kemampuannya dalam membedakan antara aktivitas normal dan serangan secara efektif.

- Distribusi skor prediksi KNN sangat terpusat pada nilai rendah dengan beberapa outlier, menggambarkan prediksi yang konservatif namun tepat pada sebagian besar data.

- Fuzzy Mamdani dan Sugeno menghasilkan distribusi skor yang lebih tersebar dan nilai median skor lebih tinggi, mencerminkan respons yang lebih sensitif terhadap variasi data, namun dengan performa klasifikasi yang lebih rendah (akurasi sekitar 97.2% dan precision serta recall yang rendah).

- Performa Mamdani dan Sugeno hampir identik dalam eksperimen ini, yang kemungkinan disebabkan oleh kesamaan aturan dan parameter fuzzy yang digunakan.

- KNN juga lebih efisien secara komputasi, membutuhkan waktu proses yang lebih singkat dibandingkan kedua metode fuzzy yang memerlukan waktu sekitar lima kali lebih lama.

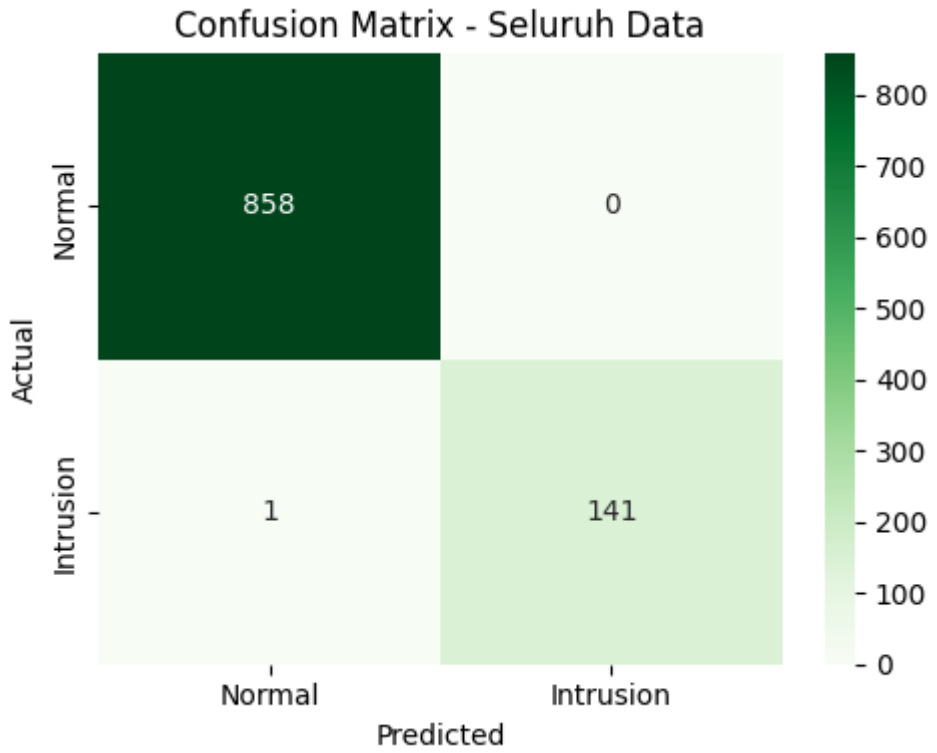
- Meskipun performa fuzzy belum optimal, metode ini menunjukkan potensi menangani ketidakpastian dalam data, yang bisa ditingkatkan dengan penyempurnaan aturan dan parameter fuzzy.

Berdasarkan hasil ini, dapat disimpulkan bahwa KNN merupakan metode yang lebih efektif dan efisien untuk klasifikasi intrusion detection pada dataset ini, sementara metode fuzzy memerlukan optimasi lebih lanjut untuk bersaing secara performa, namun menawarkan kelebihan dalam interpretabilitas dan penanganan data ambigu

3.4. Visualisasi Hasil

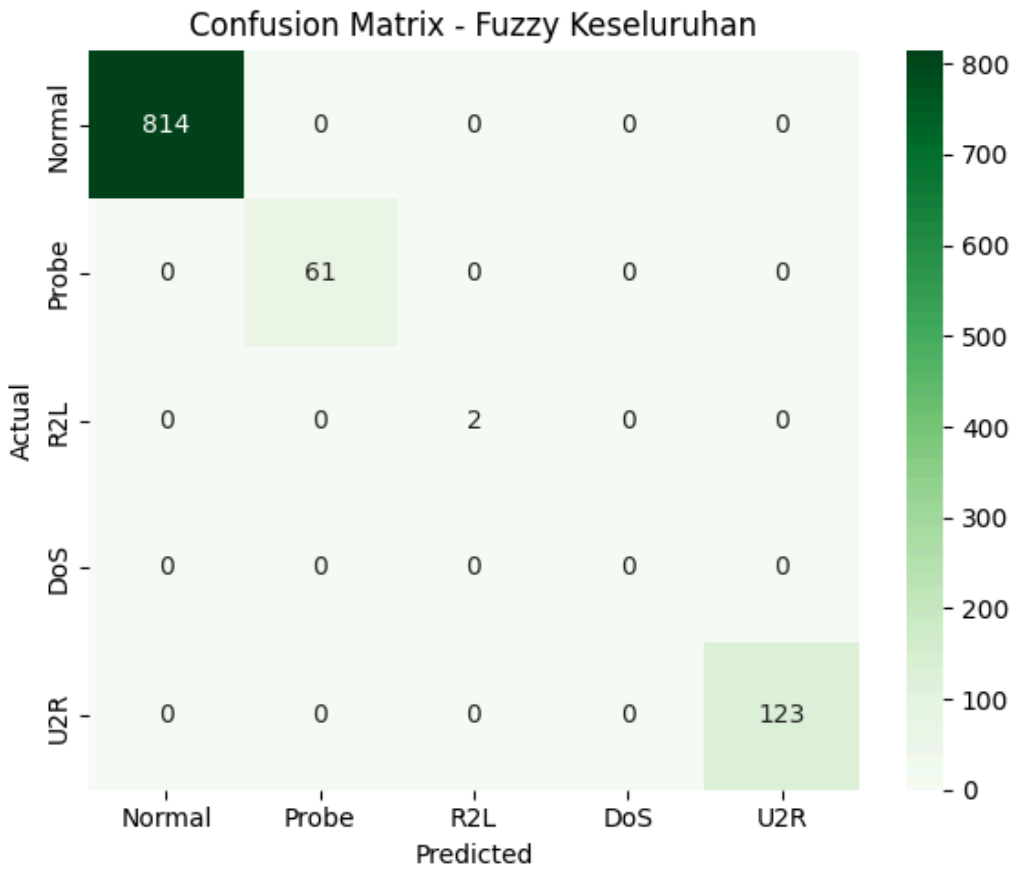
- **Confusion Matrix**:
  Visualisasi confusion matrix untuk masing-masing metode menunjukkan distribusi prediksi benar dan salah antara kelas *normal* dan *intrusion*.

KNN



- KNN sangat efektif dalam mengklasifikasikan kelas **Normal** dengan 858 prediksi benar dan tidak ada yang salah.
- Untuk kelas **Intrusion**, KNN berhasil mengklasifikasikan 141 sampel dengan benar, hanya 1 sampel yang salah klasifikasi sebagai Normal (false negative).
- Tingkat false negative yang sangat rendah menunjukkan KNN sangat baik dalam mendeteksi serangan yang sebenarnya.
- Tidak ada false positive pada kelas Normal, yang menunjukkan model juga tidak “berlebihan” dalam mendeteksi intrusion palsu.

FUZZY



- **Kelas Normal** berhasil diklasifikasikan dengan sempurna sebanyak 814 sampel tanpa kesalahan prediksi (tidak ada prediksi yang salah mengarah ke kelas lain).
- **Kelas Probe** dan **R2L** juga diklasifikasikan dengan baik, masing-masing 61 dan 2 sampel benar tanpa salah klasifikasi.
- **Kelas U2R** terdeteksi dengan akurasi sempurna (123 sampel benar).
- Namun, **kelas DoS tidak terdeteksi sama sekali** (tidak ada prediksi DoS), yang berarti model fuzzy gagal mengenali kelas ini. Kemungkinan penyebabnya adalah:
    - Data DoS yang sangat sedikit atau tidak ada di subset pengujian, atau
    - Aturan fuzzy dan fungsi keanggotaan kurang sensitif terhadap ciri khas DoS.
- Tidak ada kesalahan klasifikasi silang antar kelas lain yang terlihat (off-diagonal zeros), tapi ini juga bisa mengindikasikan model terlalu “konservatif” dan mungkin mengabaikan kelas yang sulit dikenali.

Analisis dari Confusion Matrix ini :

- **KNN unggul dalam hal deteksi intrusion secara keseluruhan** dengan tingkat kesalahan sangat rendah.

- **Model fuzzy menunjukkan kelemahan pada kelas DoS**, mungkin karena aturan fuzzy kurang memadai atau distribusi data yang kurang representatif untuk kelas tersebut.

- Fuzzy mampu memisahkan lebih banyak kelas (multi-class), tapi tidak optimal di beberapa kelas penting (DoS).

- KNN hanya membedakan dua kelas, tapi dengan akurasi sangat tinggi, cocok untuk deteksi intrusion secara umum.

BAB IV

KESIMPULAN

Berdasarkan hasil penelitian dan analisis yang telah dilakukan terhadap implementasi metode K-Nearest Neighbors (KNN), Fuzzy Mamdani, dan Fuzzy Sugeno dalam mendeteksi intrusi jaringan pada dataset Network Intrusion Detection, dapat ditarik beberapa kesimpulan sebagai berikut:

1. KNN menunjukkan performa klasifikasi terbaik dengan akurasi tertinggi di kedua klasifikasi biner dan multi-kelas, yaitu sekitar 99,5%. KNN efektif mengenali pola data dengan pendekatan jarak tetangga terdekat, menghasilkan precision dan recall yang jauh lebih baik dibandingkan metode fuzzy. Selain itu, KNN lebih efisien secara komputasi dengan waktu proses yang jauh lebih cepat.
2. Metode Fuzzy (Mamdani dan Sugeno) memiliki performa klasifikasi yang lebih rendah, dengan akurasi sekitar 97,2%, serta precision dan recall yang relatif rendah, terutama dalam deteksi kelas tertentu seperti DoS.

Lampiran 2. Link Tubes DKA (Google Colab)

https://colab.research.google.com/drive/13HJJ4o4jZRMh6J1KgQ9wo4sTTytwK4y-

Lampiran 3. Link PPT Tubes DKA

https://www.canva.com/design/DAGo6hRPf7Q/WJ4ZqUJXSJrDi9c5moTpJA/edit?utm_content=DAGo6hRPf7Q&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton