KELOMPOK ROSUNTECH

NETWORK INTRUSION DETECTION

MUHAMMAD IRGIANSYAH
103012300039

BILL STEPHEN SEMBIRING
103012330197

LATAR BELAKANG

Seiring pesatnya perkembangan komputer, aktivitas komunikasi dan pertukaran data melalui jaringan komputer menjadi hal umum dalam kehidupan sehari-hari.Namun dalam kemajuan teknologi, Dalam bidang keamanan jaringan, deteksi intrusi merupakan elemen vital untuk menjaga integritas sistem informasi dari berbagai ancaman eksternal. Meningkatnya aktivitas siber yang berpotensi merusak menuntut solusi cerdas untuk mengidentifikasi lalu lintas yang mencurigakan secara otomatis. Oleh karena itu, pendekatan berbasis data, seperti klasifikasi menggunakan machine learning, menjadi semakin penting. Dataset ini memberikan landasan untuk membangun sistem pendeteksi intrusi yang mampu membedakan antara aktivitas normal dan serangan berbahaya.

RUMUSAN MASALAH

DATASET BERSIFAT SIMULASI, BUKAN DATA REAL-TIME JARINGAN.

DUA PENDEKATAN KLASIFIKASI:

- BINER (NORMAL VS INTRUSION)
- MULTI-KELAS (DOS, PROBE, R2L, U2R)

DUA ALGORITMA:

- K-NEAREST NEIGHBORS (KNN)
- FUZZY LOGIC (MAMDANI & SUGENO)

EVALUASI TERBATAS PADA DATA UJI.

TIDAK MENCAKUP IMPLEMENTASI SISTEM IDS.

TUJUAN

- Menerapkan algoritma sistem fuzzy berbasis Sugeni dan juga Mamdan, serta KNN untuk mendeteksi intrusi jaringan.
- Mengevaluasi performa model menggunakan metrik seperti akurasi, precision, recall, dan F1-score.
- Membandingkan perfofrma dari metode Fuzzy, baik Mamdani dan Sugeno, dan juga dari metode KNN

MANFAAT

- 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.

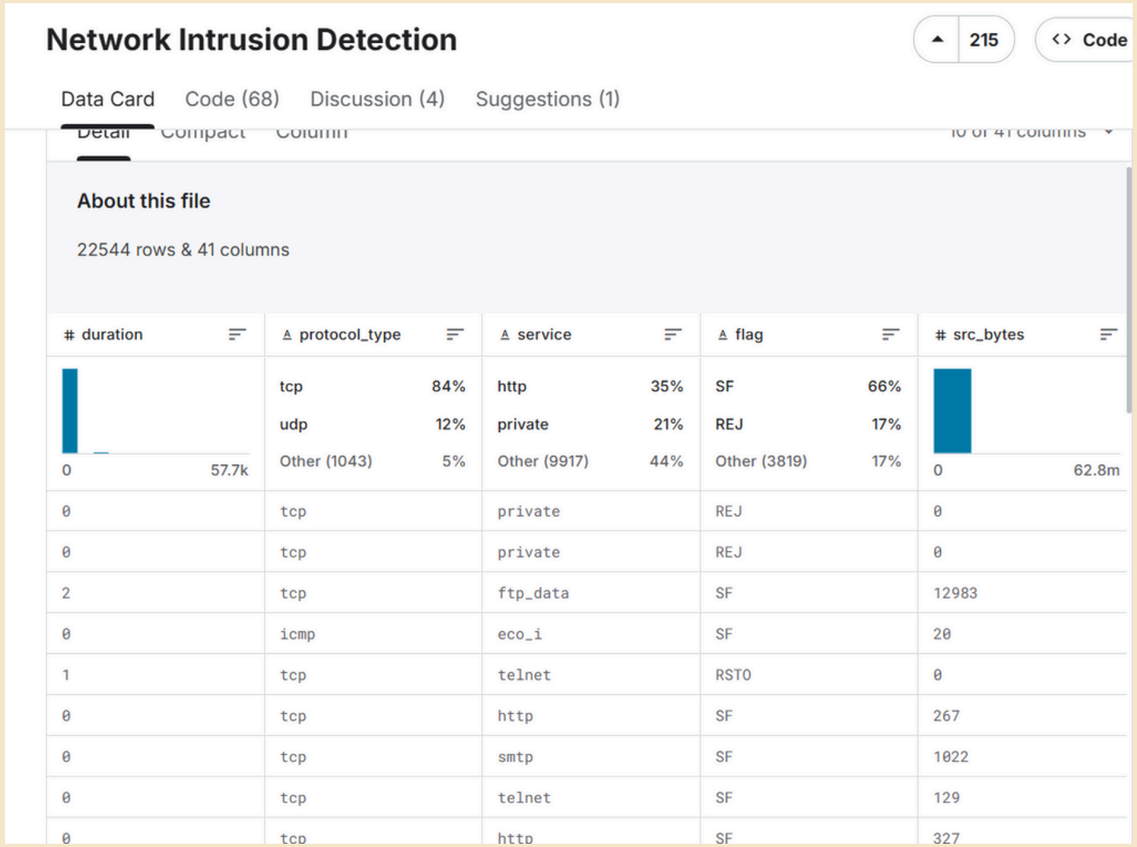
ROSUNTECH

SOLUSI

Mengembangkan sistem deteksi intrusi jaringan menggunakan kombinasi algoritma KNN dan Fuzzy Logic (Mamdani & Sugeno) dengan metodologi yang meliputi eksplorasi data, pra-pemrosesan, pelatihan (training), pengujian (testing) , dan evaluasi performa secara mendalam

Pendekatan ini memungkinkan deteksi otomatis terhadap aktivitas mencurigakan dalam jaringan dengan klasifikasi yang akurat dan evaluasi komprehensif terhadap model yang dibuat.

DESKRIPSI DATASET



Network Intrusion Detection

215 Code

Data Card | Code (68) | Discussion (4) | Suggestions (1)

Detail Compact Column 10 of 41 columns

About this file

22544 rows & 41 columns

# duration	protocol_type	service	flag	# src_bytes
0 – 57.7k	tcp 84% udp 12% Other (1043) 5%	http 35% private 21% Other (9917) 44%	SF 66% REJ 17% Other (3819) 17%	0 – 62.8m
0	tcp	private	REJ	0
0	tcp	private	REJ	0
2	tcp	ftp_data	SF	12983
0	icmp	eco_i	SF	20
1	tcp	telnet	RSTO	0
0	tcp	http	SF	267
0	tcp	smtp	SF	1022
0	tcp	telnet	SF	129
0	tcp	http	SF	327

- Dataset yang digunakan dalam penelitian ini adalah Network Intrusion Detection Dataset yang diambil dari Kaggle (https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection)

- 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

PREPROCESSING DATASET

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.

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.

PEMBAGIAN DATASET

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

METODE KLASIFIKASI

KNN

Metode klasifikasi berbasis instance yang menentukan kelas suatu data berdasarkan mayoritas kelas dari tetangga terdekatnya

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.

KNN

PRA-PEMROSESAN DATA KNN

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, precision_score, recall_score, f1_score
from scipy.stats import wilcoxon
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
```

```
# === 1. Load dan Bersihkan Data ===
df = pd.read_csv('D:/Telkom/Semester 4/DKA/tubes/Test_data.csv')
df.replace('?', np.nan, inplace=True)
df.dropna(inplace=True)
```

PRA-PEMROSESAN DATA KNN

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

# === 3. Label Biner (Normal vs Intrusion) ===
df['binary_label'] = df['src_bytes'].astype(int).apply(lambda x: 0 if x <= 500 else 1)

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

# === 5. Normalisasi untuk scoring klasifikasi multi-kelas ===
scaler_temp = MinMaxScaler()
df[['duration', 'src_bytes', 'dst_bytes', 'count', 'srv_count']] = scaler_temp.fit_transform(
    df[['duration', 'src_bytes', 'dst_bytes', 'count', 'srv_count']]
)
```

PRA-PEMROSESAN DATA KNN

```python
# === 6. Tambah Label Multi-Kelas Berdasarkan Skor dan dst_bytes ===
def compute_score(row):
    score = (
        row['duration'] * 0.2 +
        row['src_bytes'] * 0.2 +
        row['dst_bytes'] * 0.2 +
        row['count'] * 0.2 +
        row['srv_count'] * 0.2
    ) * 100
    return score

def classify(score, dst_bytes):
    if score <= 30:
        return 'Normal'
    elif score <= 60:
        return 'Probe'
    elif score <= 80:
        return 'R2L'
    else:
        return 'DoS' if dst_bytes > 0.5 else 'U2R'
```

PRA-PEMROSESAN DATA KNN

```
# === 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 KNN

```python
# === Upper Scaling: Batasi 1000 data pertama ===
df = df.head(1000).copy()

# === 7. Pisahkan Fitur dan Label ===
X = df.drop(['binary_label', 'attack_type', 'score'], axis=1)
y_bin = df['binary_label']
y_multi = df['attack_type']

# === 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)
```

IMPLEMENTASI ALGORITMA KNN

```python
# === 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}")
```

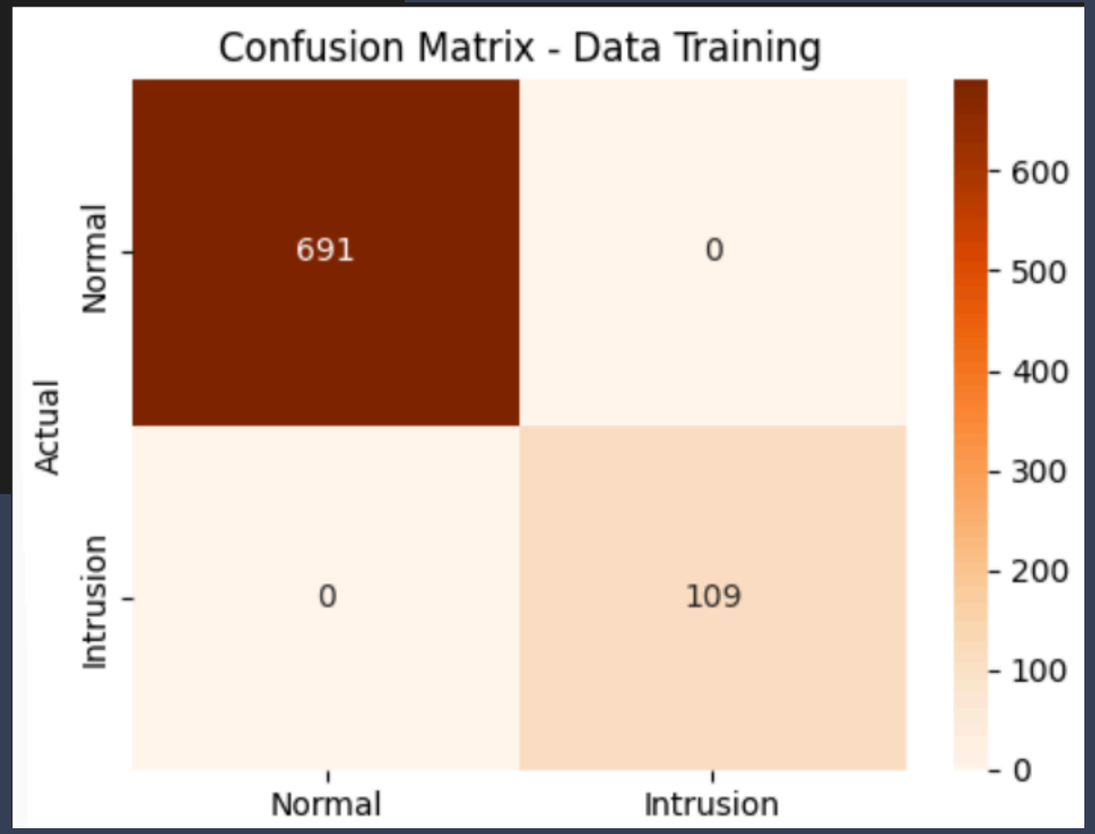
```
Nilai k optimal: 1 (Akurasi: 0.9950)

=== 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
```

IMPLEMENTASI ALGORITMA DAN EVALUASI KNN

```python
# === Evaluasi pada Data Training ===
y_train_pred_bin = knn_bin.predict(X_train_bin)
train_acc_bin = accuracy_score(y_train_bin, y_train_pred_bin)
train_prec_bin = precision_score(y_train_bin, y_train_pred_bin, zero_division=0)
train_rec_bin = recall_score(y_train_bin, y_train_pred_bin, zero_division=0)
train_f1_bin = f1_score(y_train_bin, y_train_pred_bin, zero_division=0)
print(f"\n[Training]     Akurasi: {train_acc_bin:.4f} | Precision: {train_prec_bin:.4f} | Recall: {train_rec_bin:.4f} | F1-score: {train_f1_bin:.4f}")
# Confusion Matrix - Data Training
print("\n=== Evaluasi Klasifikasi Biner - Training ===")
print(f"Akurasi: {train_acc_bin:.4f}")
print("Classification Report:\n", classification_report(y_train_bin, y_train_pred_bin))
cm_train = confusion_matrix(y_train_bin, y_train_pred_bin)
plt.figure(figsize=(5, 4))
sns.heatmap(cm_train, annot=True, fmt='d', cmap='Oranges',
            xticklabels=['Normal', 'Intrusion'], yticklabels=['Normal', 'Intrusion'])
plt.title("Confusion Matrix - Data Training")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

IMPLEMENTASI ALGORITMA DAN EVALUASI KNN

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

=== Evaluasi Klasifikasi Biner - Training ===
Akurasi: 1.0000
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       691
           1       1.00      1.00      1.00       109

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

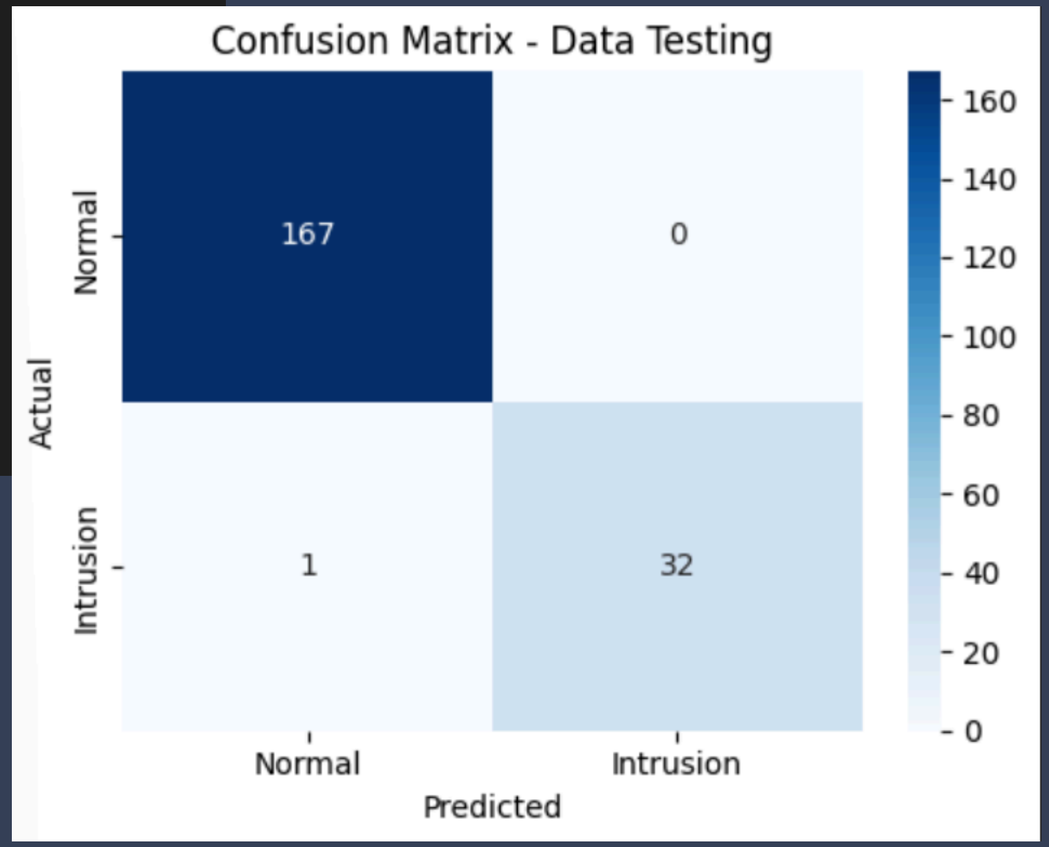


IMPLEMENTASI ALGORITMA DAN EVALUASI KNN

```python
# === 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
print("\n=== Evaluasi Klasifikasi Biner - Testing ===")
print(f"Akurasi: {test_acc_bin:.4f}")
print("Classification Report:\n", classification_report(y_test_bin, y_test_pred_bin))
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()
```

IMPLEMENTASI ALGORITMA DAN EVALUASI KNN

```
[Testing]      Akurasi: 0.9950 | Precision: 1.0000 | Recall: 0.9697 | F1-score: 0.9846

=== Evaluasi Klasifikasi Biner - Testing ===
Akurasi: 0.9950
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       167
           1       1.00      0.97      0.98        33

    accuracy                           0.99       200
   macro avg       1.00      0.98      0.99       200
weighted avg       1.00      0.99      0.99       200
```

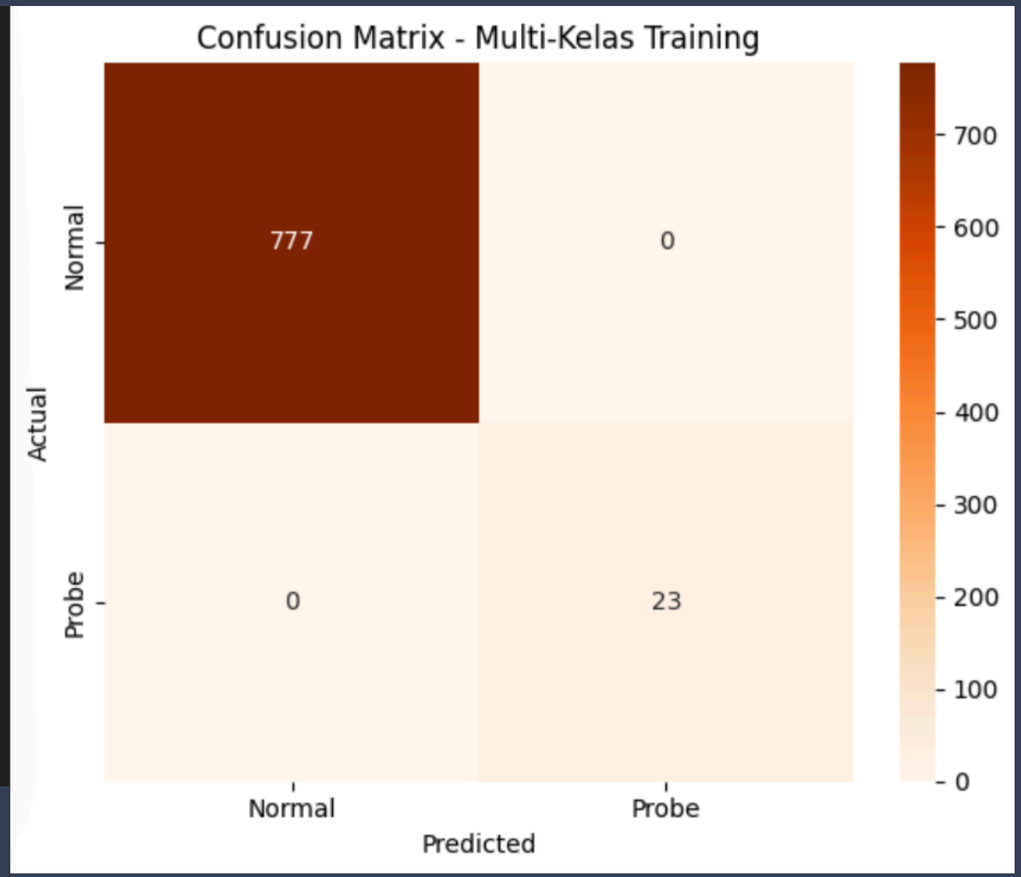


IMPLEMENTASI ALGORITMA DAN EVALUASI KNN

```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())

# === 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()
```

IMPLEMENTASI ALGORITMA DAN EVALUASI KNN

```
=== 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
```

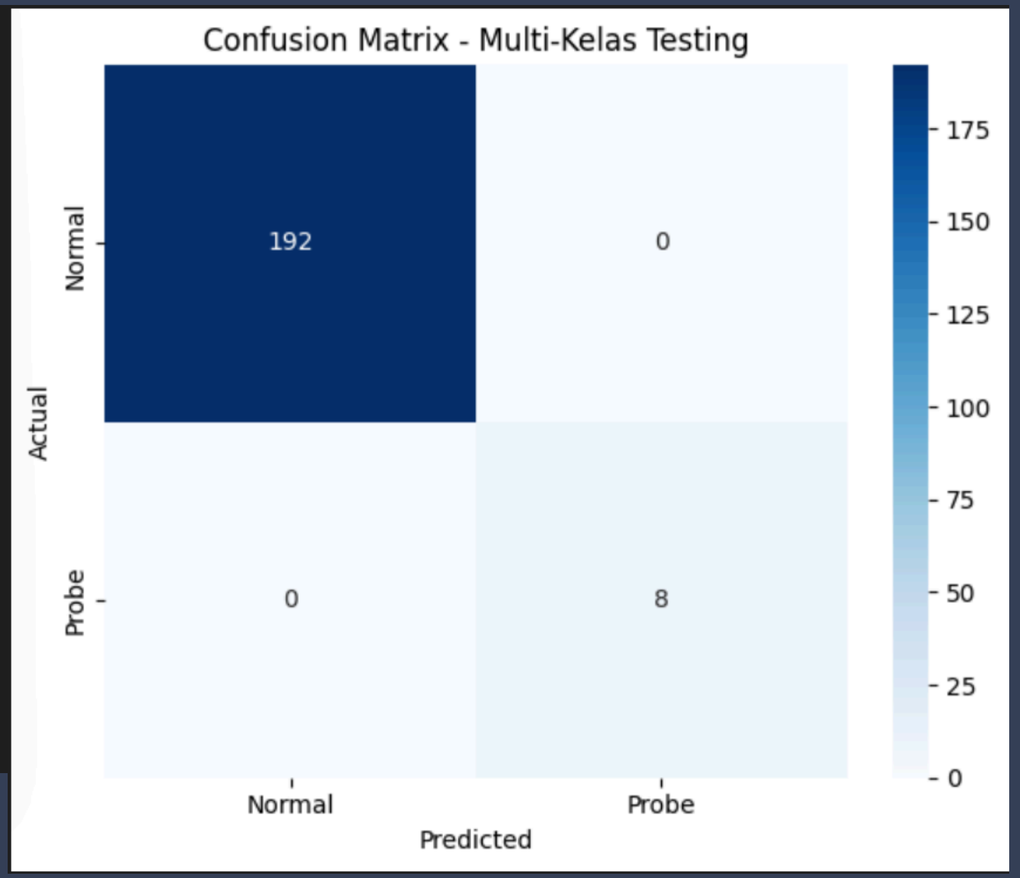


IMPLEMENTASI ALGORITMA DAN EVALUASI KNN

```
# === 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()
```

IMPLEMENTASI ALGORITMA DAN EVALUASI KNN

```
=== 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
```

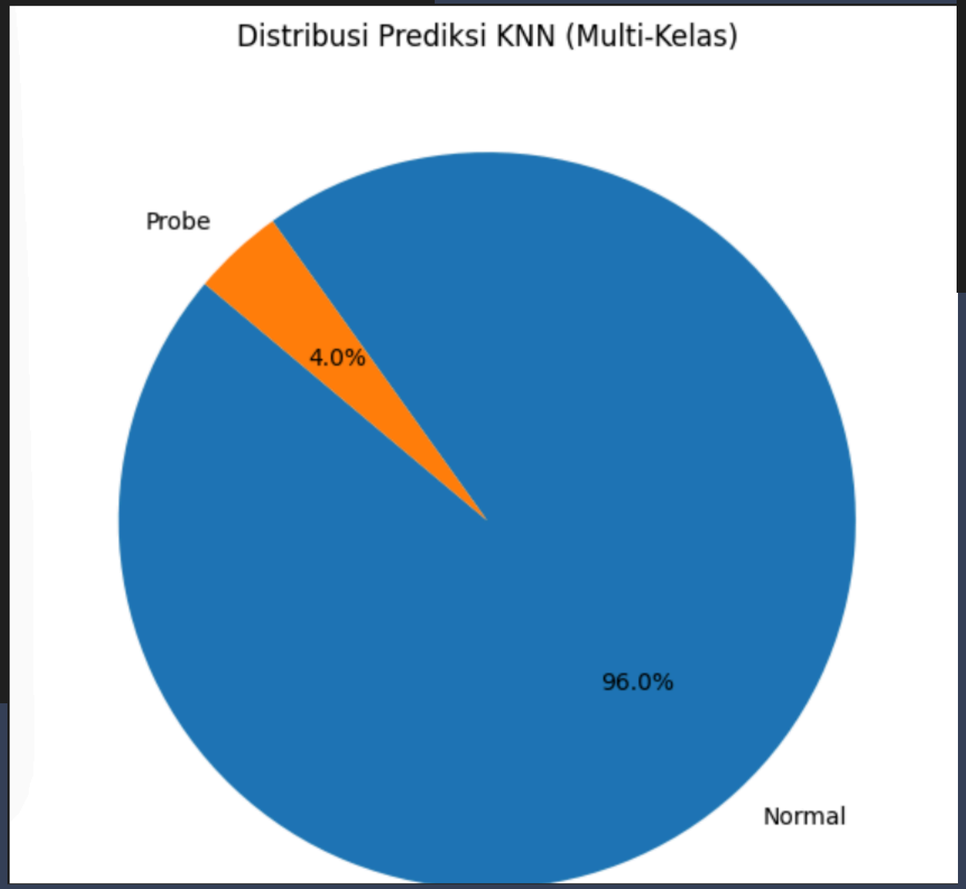


IMPLEMENTASI ALGORITMA DAN EVALUASI KNN

```python
# === Contoh Hasil Prediksi Multi-Kelas ===
y_pred_multi = knn_multi.predict(X_test_multi)

results_df = pd.DataFrame({
    'Actual': y_test_multi,
    'Predicted': y_pred_multi
})
print("\n=== Contoh Hasil Prediksi Multi-Kelas ===")
print(results_df.head(10))

pd.Series(y_pred_multi).value_counts().plot(kind='pie', autopct='%1.1f%%', startangle=140, figsize=(6, 6))
plt.title("Distribusi Prediksi KNN (Multi-Kelas)")
plt.ylabel('')
plt.tight_layout()
plt.show()

# === Evaluasi Beberapa Split Data ===
split_ratios = [0.1, 0.2, 0.3, 0.4, 0.5]
print("\n=== Evaluasi Akurasi Berbagai Split Data ===")
for ratio in split_ratios:
    X_train, X_test, y_train, y_test = train_test_split(X, y_bin, test_size=ratio, random_state=42)
    X_train[numeric_cols] = scaler.fit_transform(X_train[numeric_cols])
    X_test[numeric_cols] = scaler.transform(X_test[numeric_cols])
    model = KNeighborsClassifier(n_neighbors=3)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f"Split {int((1-ratio)*100)}:{int(ratio*100)} - Akurasi: {acc:.4f}")
```

IMPLEMENTASI ALGORITMA DAN EVALUASI KNN

```
=== Contoh Hasil Prediksi Multi-Kelas ===
     Actual Predicted
521  Normal    Normal
737  Normal    Normal
740  Normal    Normal
660  Normal    Normal
411  Normal    Normal
678  Normal    Normal
626  Normal    Normal
513  Normal    Normal
859  Normal    Normal
136  Normal    Normal
```




```
=== Evaluasi Akurasi Berbagai Split Data ===
Split 90:10 - Akurasi: 0.9900
Split 80:20 - Akurasi: 0.9900
Split 70:30 - Akurasi: 0.9900
Split 60:40 - Akurasi: 0.9925
Split 50:50 - Akurasi: 0.9760
```

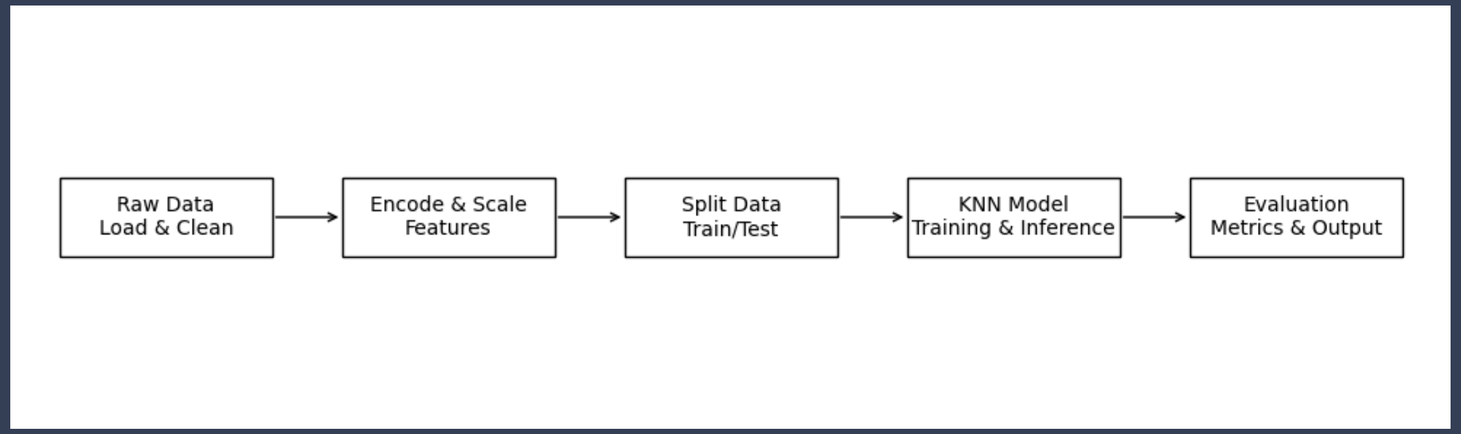
IMPLEMENTASI ALGORITMA DAN EVALUASI KNN

```python
# === Augmentasi Data dan Wilcoxon Test (dengan noise ringan) ===
df_aug = df.copy()
indices_to_modify = np.random.choice(df_aug.index, size=int(0.05 * len(df_aug)), replace=False)
df_aug.loc[indices_to_modify, 'src_bytes'] += np.random.normal(loc=0, scale=10, size=len(indices_to_modify))

print(f"\nData dimodifikasi secara acak pada {len(indices_to_modify)} baris (augmentasi dengan noise ringan)")

original = df['src_bytes'].loc[indices_to_modify].copy()
augmented = df_aug['src_bytes'].loc[indices_to_modify].copy()

if len(original) == len(augmented):
    stat, p = wilcoxon(original, augmented)
    print(f"\nWilcoxon test untuk src_bytes: statistic = {stat:.4f}, p-value = {p:.4f}")
else:
    print("\nTidak cukup data sepadan untuk Wilcoxon test.")
```

```
Data dimodifikasi secara acak pada 50 baris (augmentasi dengan noise ringan)

Wilcoxon test untuk src_bytes: statistic = 617.0000, p-value = 0.8482
```

BLOCK DIAGRAM KNN

```
# === Blok Diagram KNN ===
# Create block diagram for KNN classification pipeline
fig, ax = plt.subplots(figsize=(10, 3))
ax.set_axis_off()

# Define block positions and labels
blocks = [
    ("Raw Data\nLoad & Clean", 0.1),
    ("Encode & Scale\nFeatures", 0.3),
    ("Split Data\nTrain/Test", 0.5),
    ("KNN Model\nTraining & Inference", 0.7),
    ("Evaluation\nMetrics & Output", 0.9),
]

# Draw blocks and arrows
width, height = 0.15, 0.2
for label, x in blocks:
    rect = Rectangle((x - width/2, 0.4), width, height, fill=False)
    ax.add_patch(rect)
    ax.text(x, 0.5, label, ha='center', va='center')
# Draw arrows between blocks
for i in range(len(blocks) - 1):
    x_start = blocks[i][1] + width/2
    x_end = blocks[i+1][1] - width/2
    ax.annotate("",
                xy=(x_end, 0.5), xytext=(x_start, 0.5),
                arrowprops=dict(arrowstyle='->'))

plt.tight_layout()
plt.show()
```

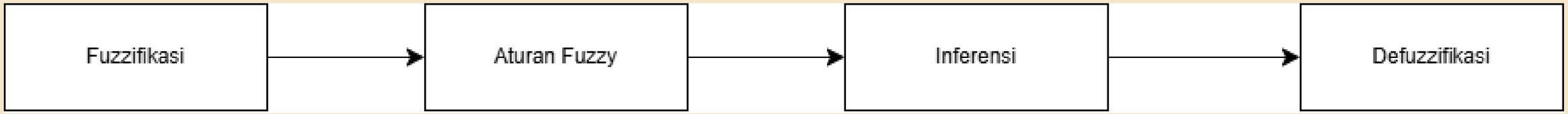


FUZZY

MENANGANI KETIDAKPASTIAN DAN KOMPLEKSITAS DALAM DATA JARINGAN YANG TIDAK DAPAT DENGAN MUDAH DIKLASIFIKASIKAN SECARA TEGAS

ROSUNTECH

FUZZY MAMDANI

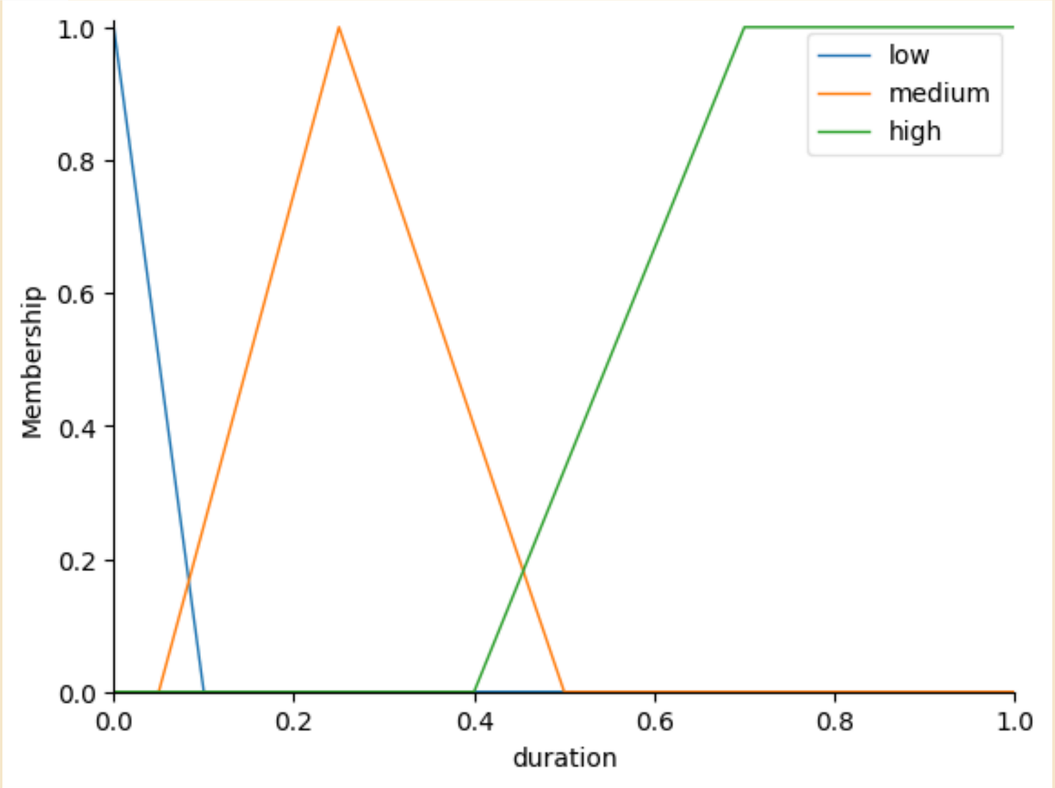


GAMBAR BLOK DIAGRAM FUZZY MAMDANI

FUZZIFIKASI

```
_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()
```

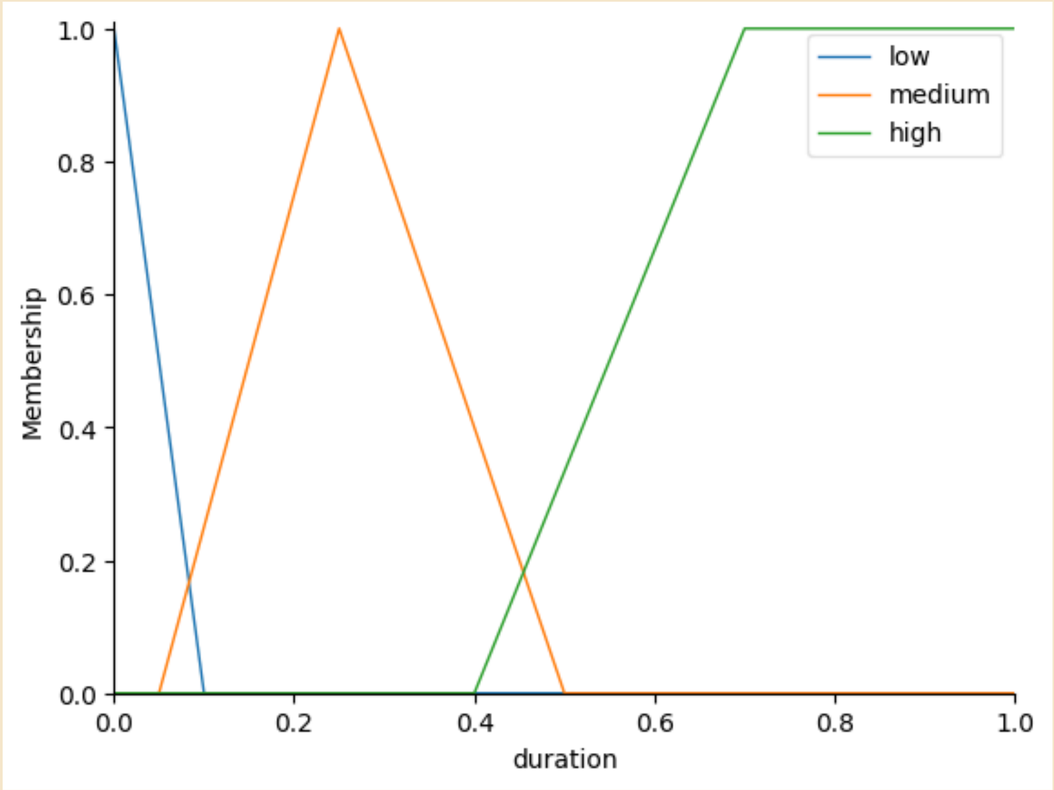
COUNT



FUZZIFIKASI

```
_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()
```

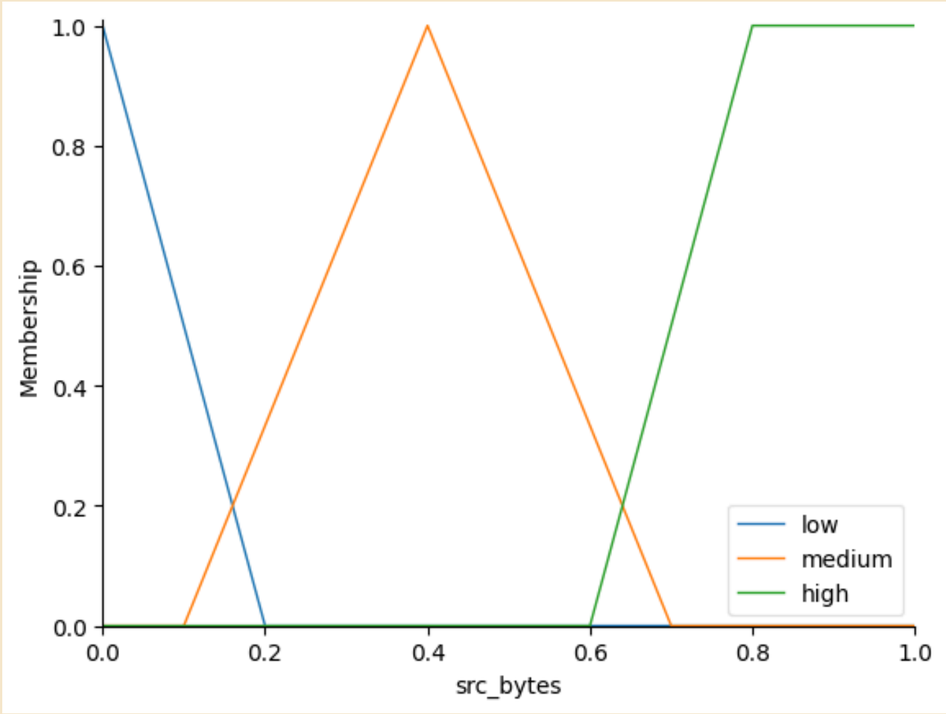
DURATION



FUZZIFIKASI

```
_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()
```

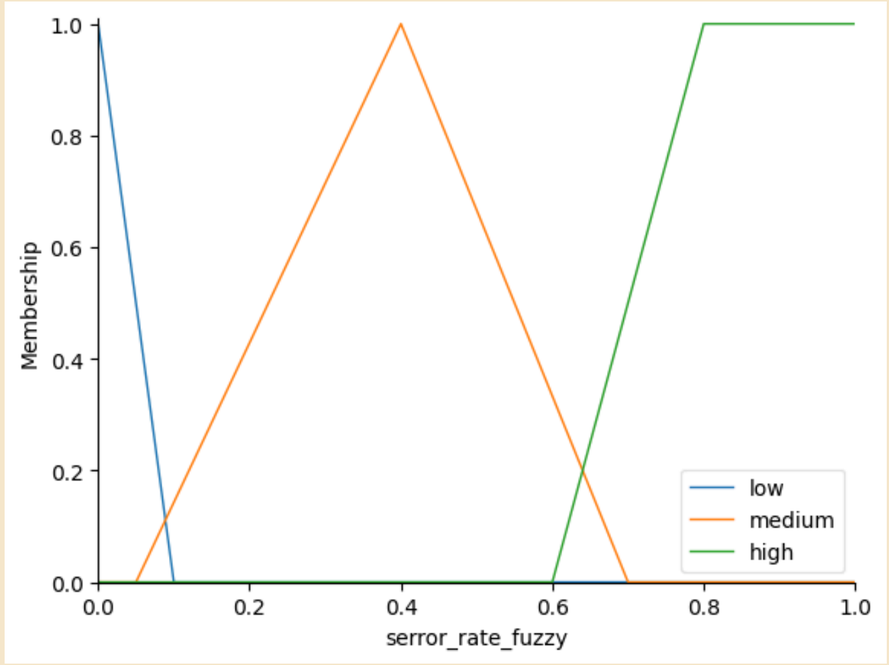
SRC_BYTES



FUZZIFIKASI

```
_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()
```

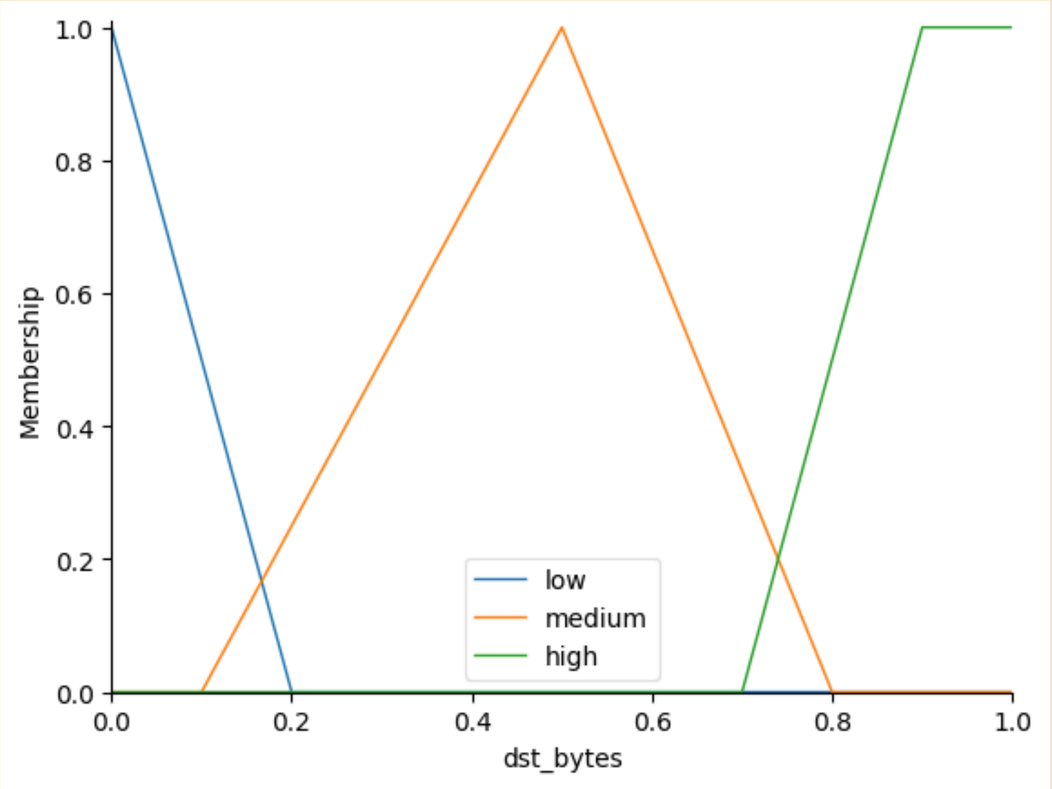
SERROR_RATE



FUZZIFIKASI

```
_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()
```

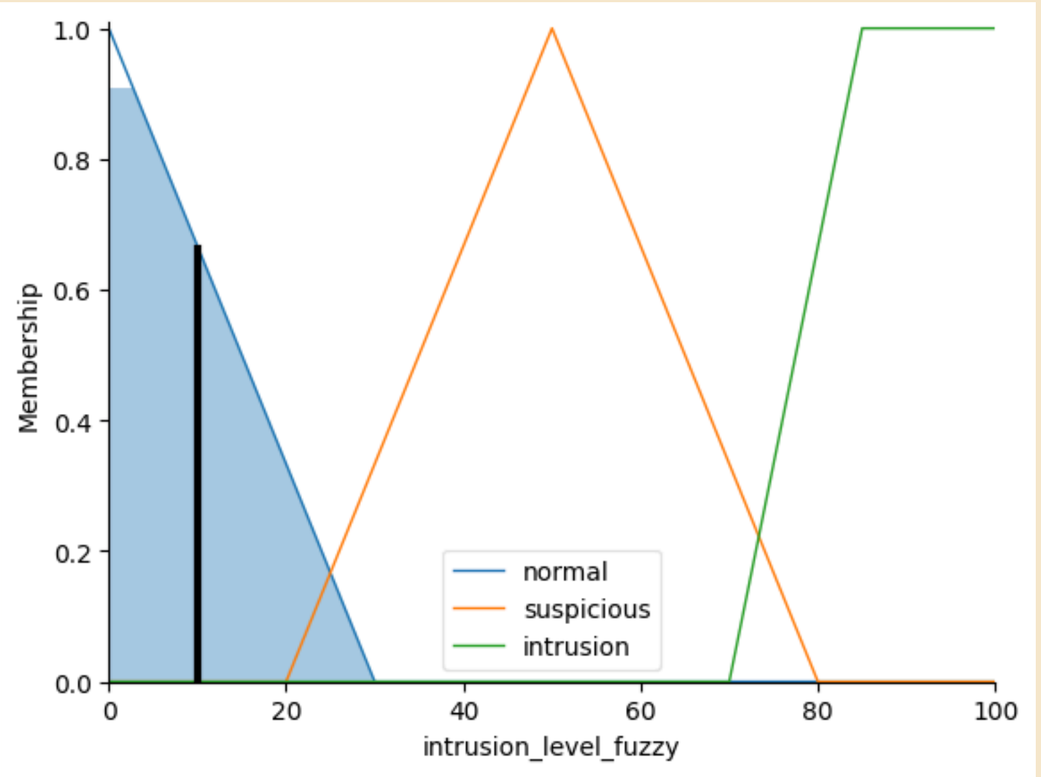
DST_BYTES



ATURAN FUZZY

```
_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()
```

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.



INTERFERENSI

SISTEM INFERENSI MAMDANI MENERAPKAN ATURAN FUZZY UNTUK MENENTUKAN DERAJAT KEBENARAN OUTPUT FUZZY BERDASARKAN INPUT.

```
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

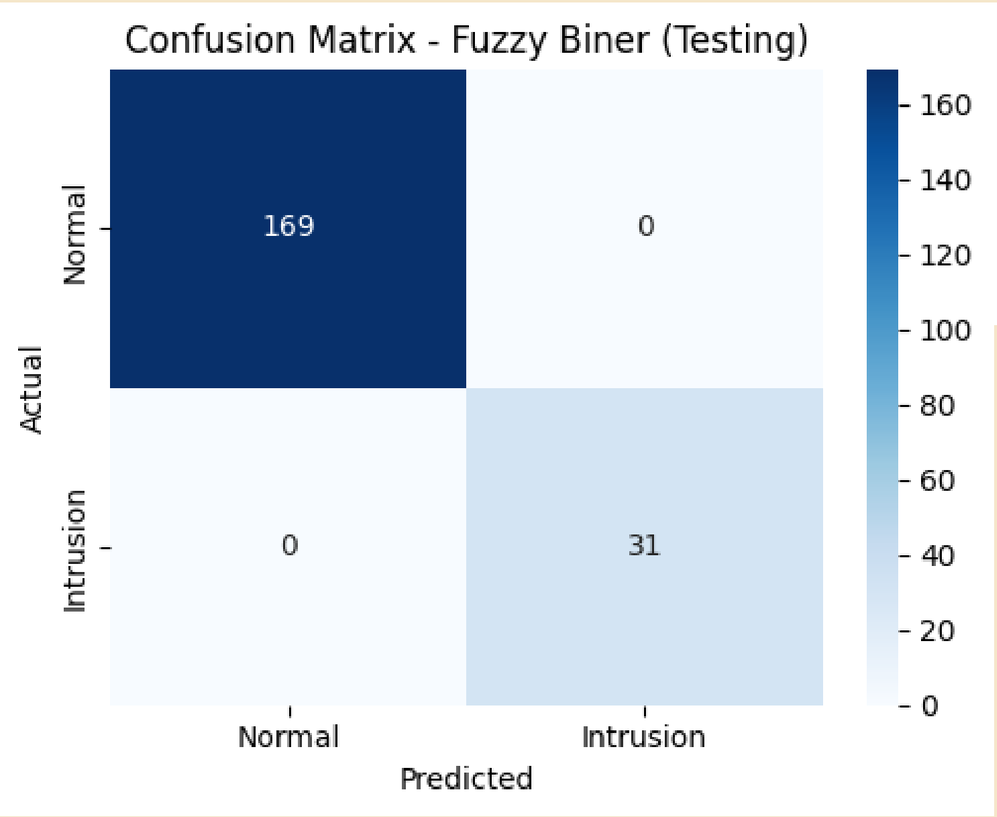
BINER

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

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
```

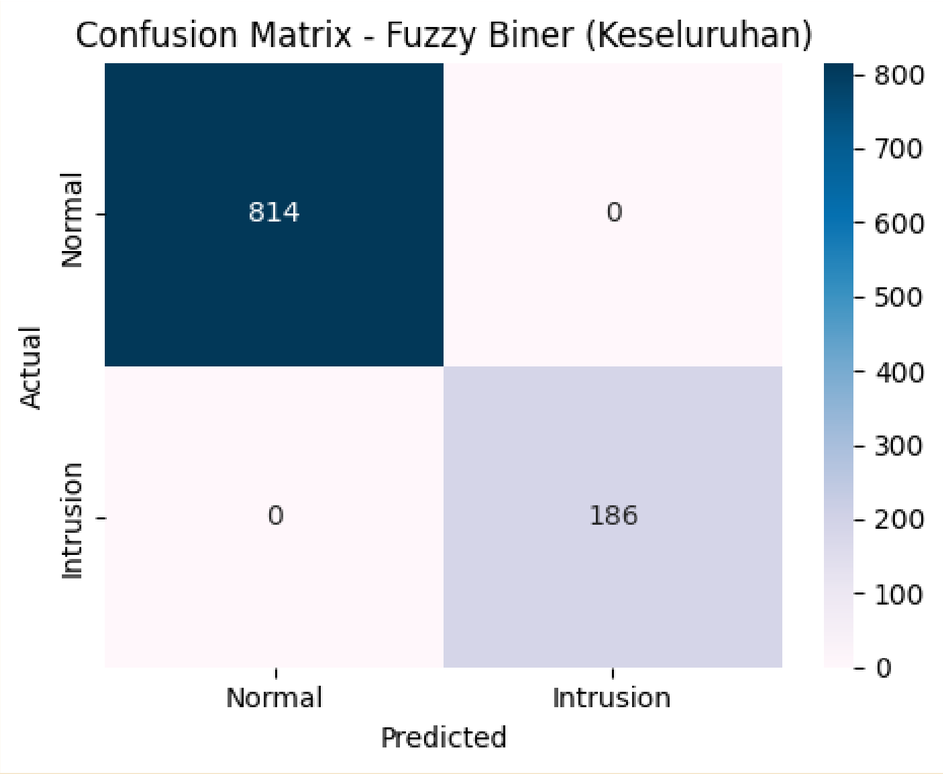
DEFUZZIFIKASI (BINARY)




```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()
```

BINER TESTING

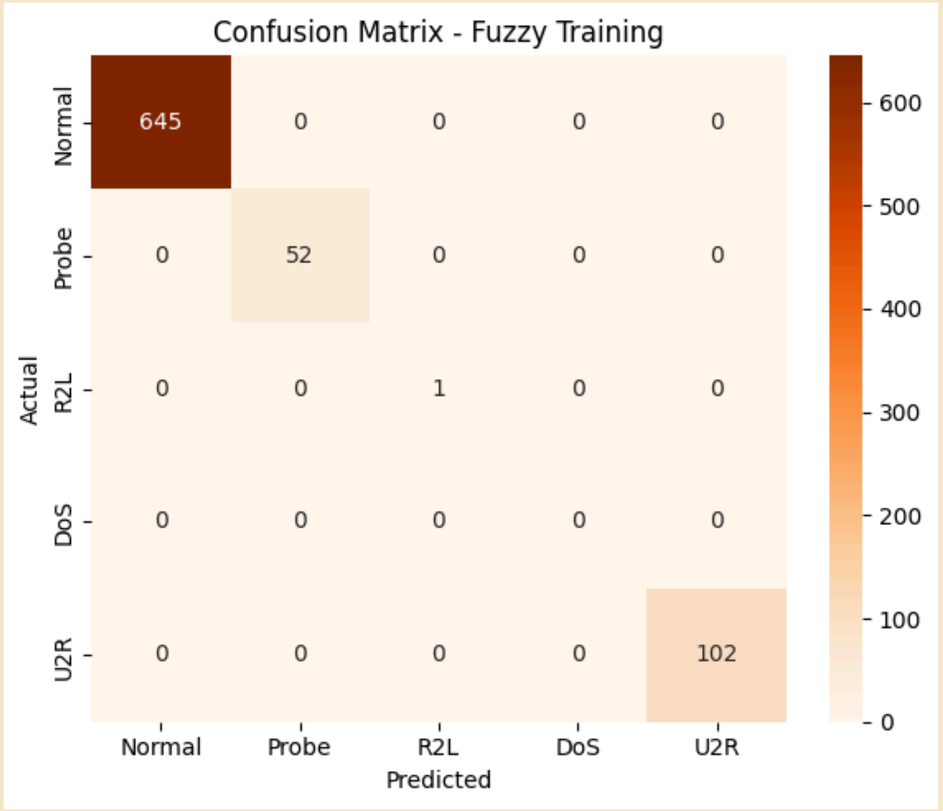
DEFUZZIFIKASI (BINARY)




```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()
```

BINER KESELURUHAN

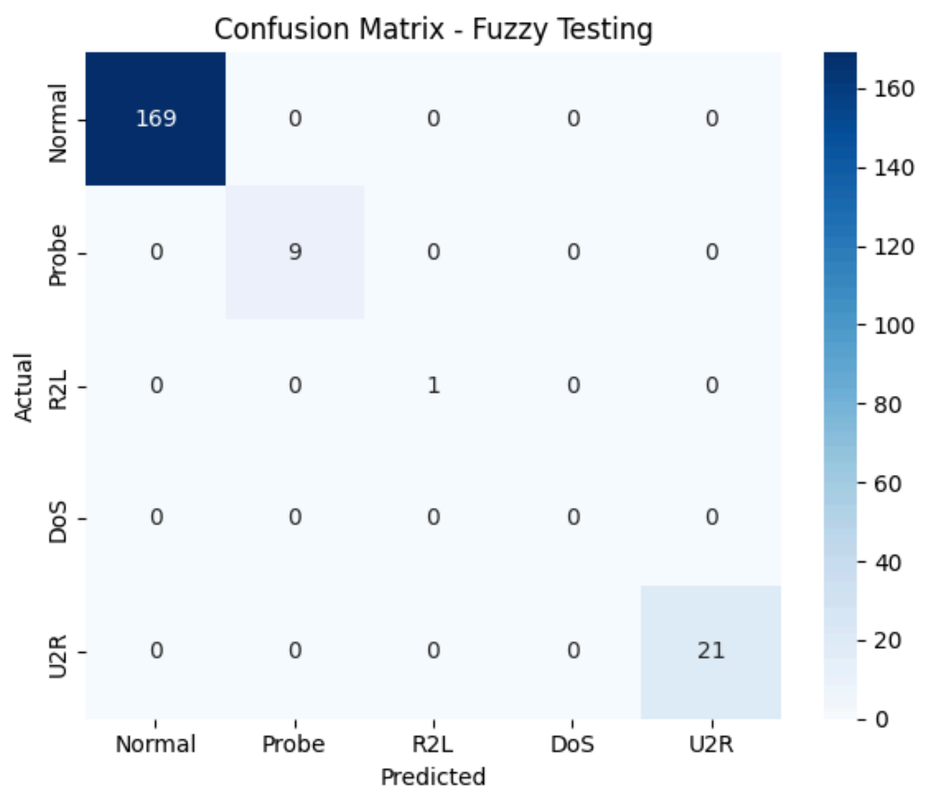
DEFUZZIFIKASI (MULTI-KELAS)




```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()
```

MULTI-KELAS TRAINING

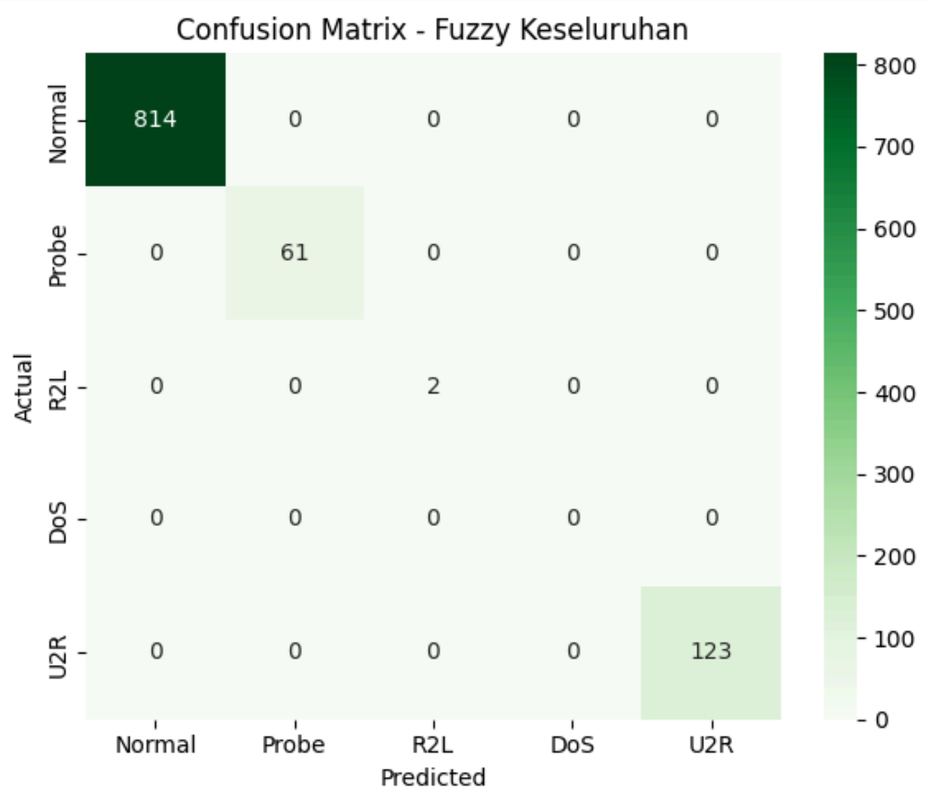
DEFUZZIFIKASI (MULTI-KELAS)




```python
# 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()
```

MULTI-KELAS TESTING

DEFUZZIFIKASI (MULTI-KELAS)

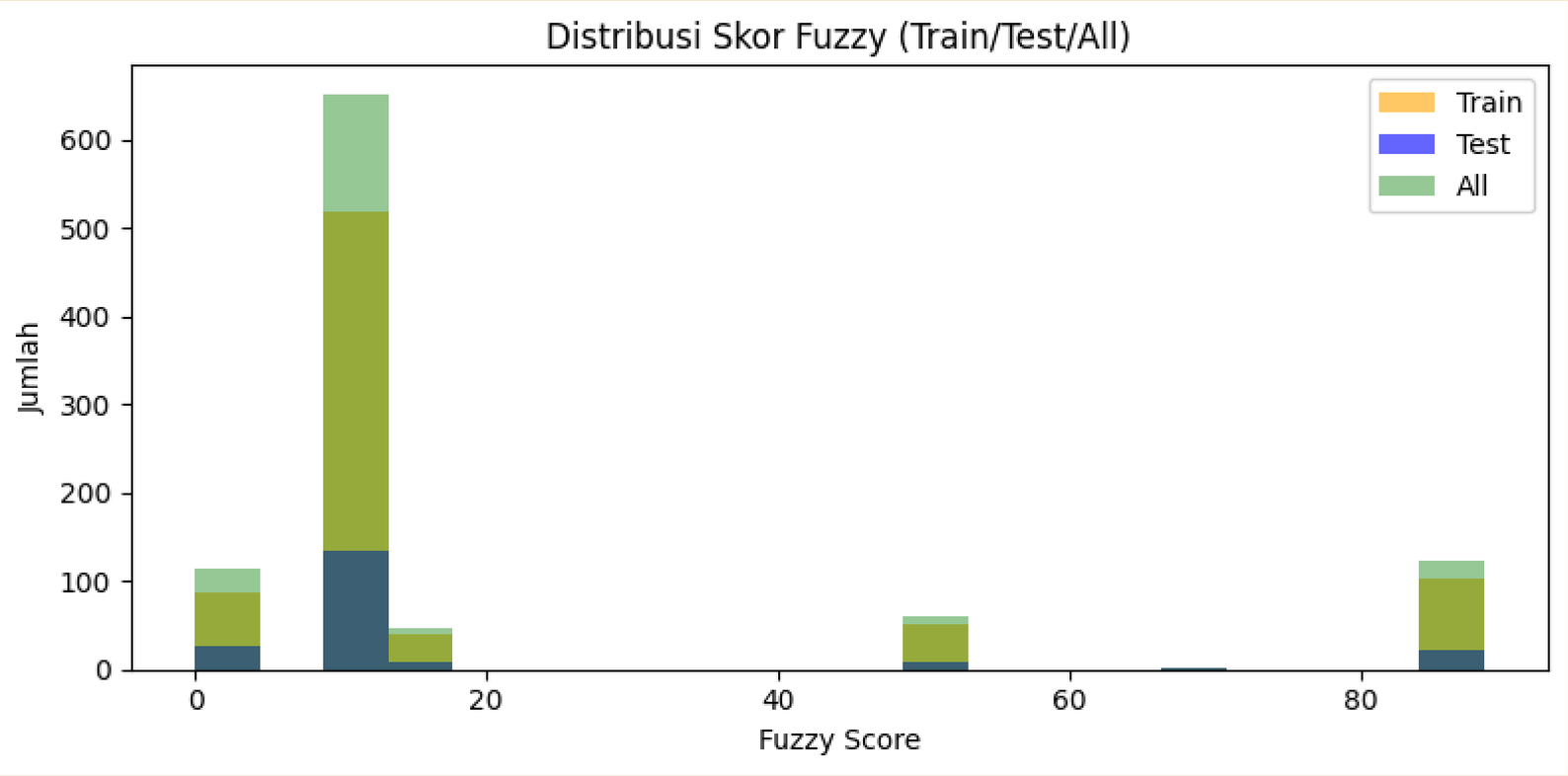



```python
# Confusion Matrix - Multi-Kelas Keseluruhan
_cm_all_fuzzy = confusion_matrix(_df_sample_fuzzy['multi_prediction_fuzzy'], _predictions_fuzzy, labels=['Normal', 'Probe', 'R2L', 'DoS', 'U2R'])
plt.figure(figsize=(6, 5))
sns.heatmap(_cm_all_fuzzy, annot=True, fmt='d', cmap='Greens', xticklabels=['Normal', '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()
```

MULTI-KELAS KESELURUHAN

DEFUZZIFIKASI (MULTI-KELAS)

DISTRIBUSI SKOR



SEBAGIAN BESAR DATA DINILAI “NORMAL” OLEH SISTEM FUZZY (SKOR RENDAH), DAN HANYA SEDIKIT DATA YANG MASUK SKOR TINGGI YANG MENANDAKAN POTENSI INTRUSI.

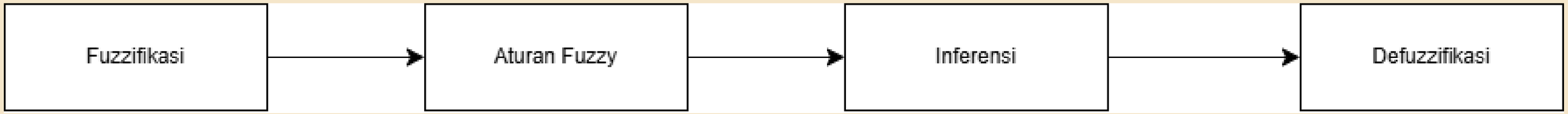
WILCOXON

```
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.

ROSUNTECH

FUZZY SUGENO

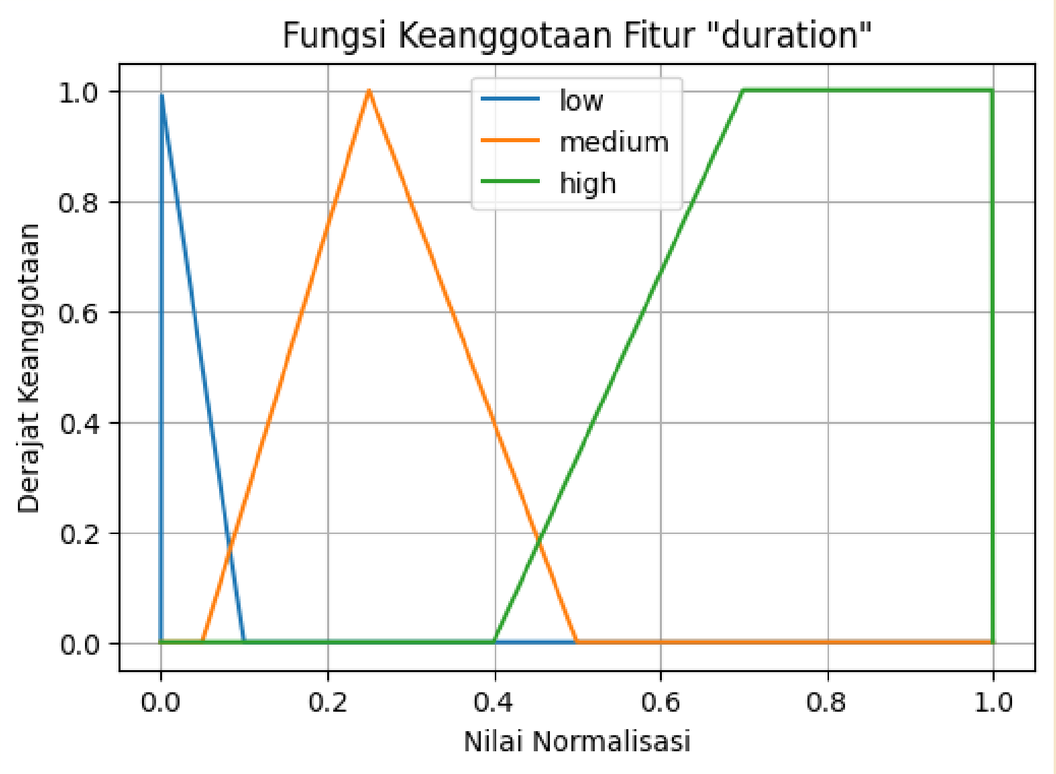


GAMBAR BLOK DIAGRAM FUZZY MAMDANI

FUZZIFIKASI

```
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'),
    },
```

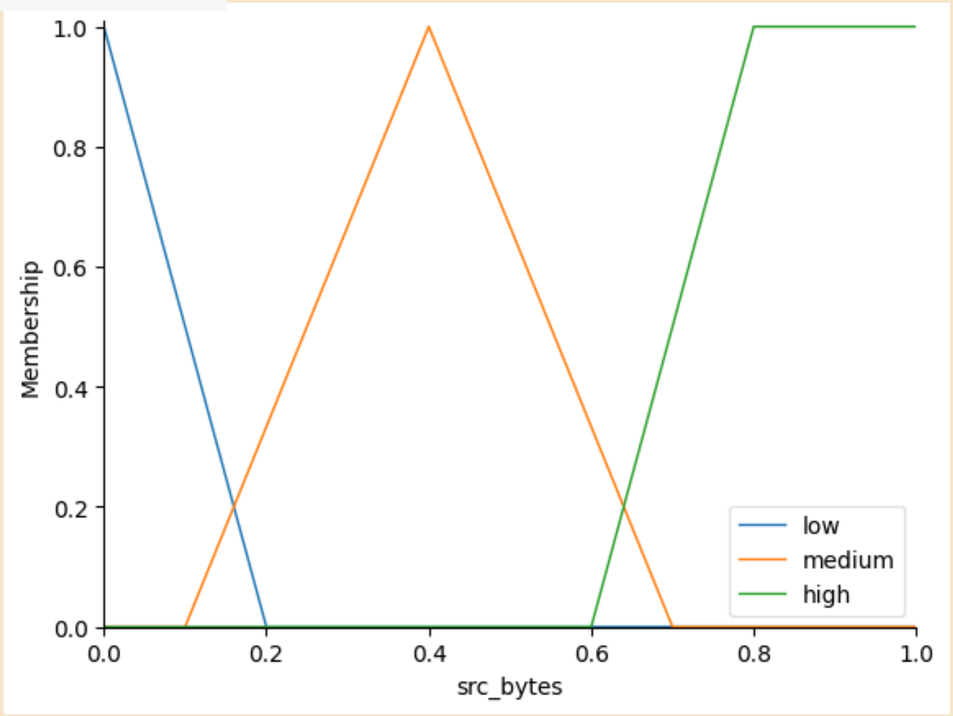
DURATION



FUZZIFIKASI

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

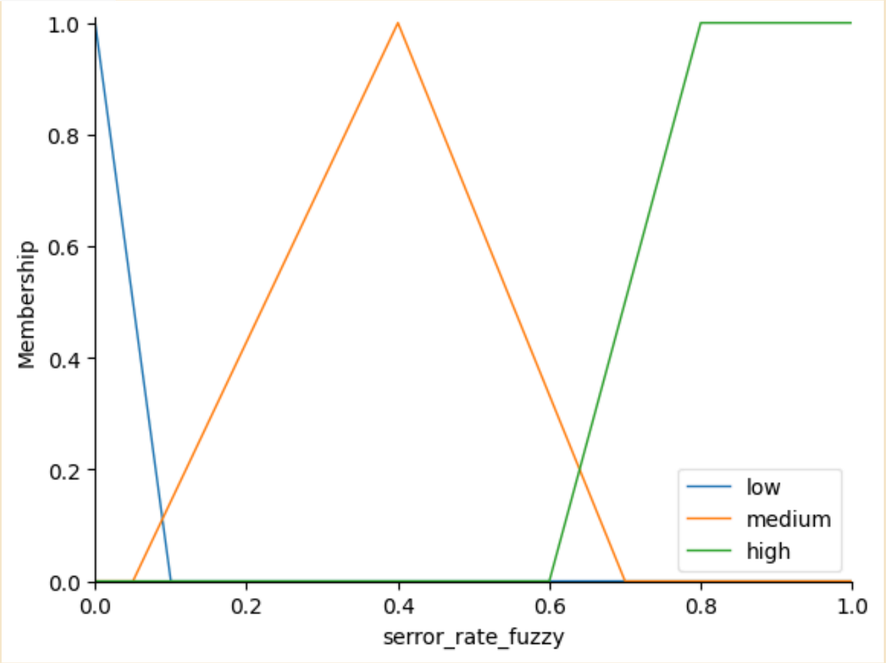
SRC_BYTES



FUZZIFIKASI

```
'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'),
}
```

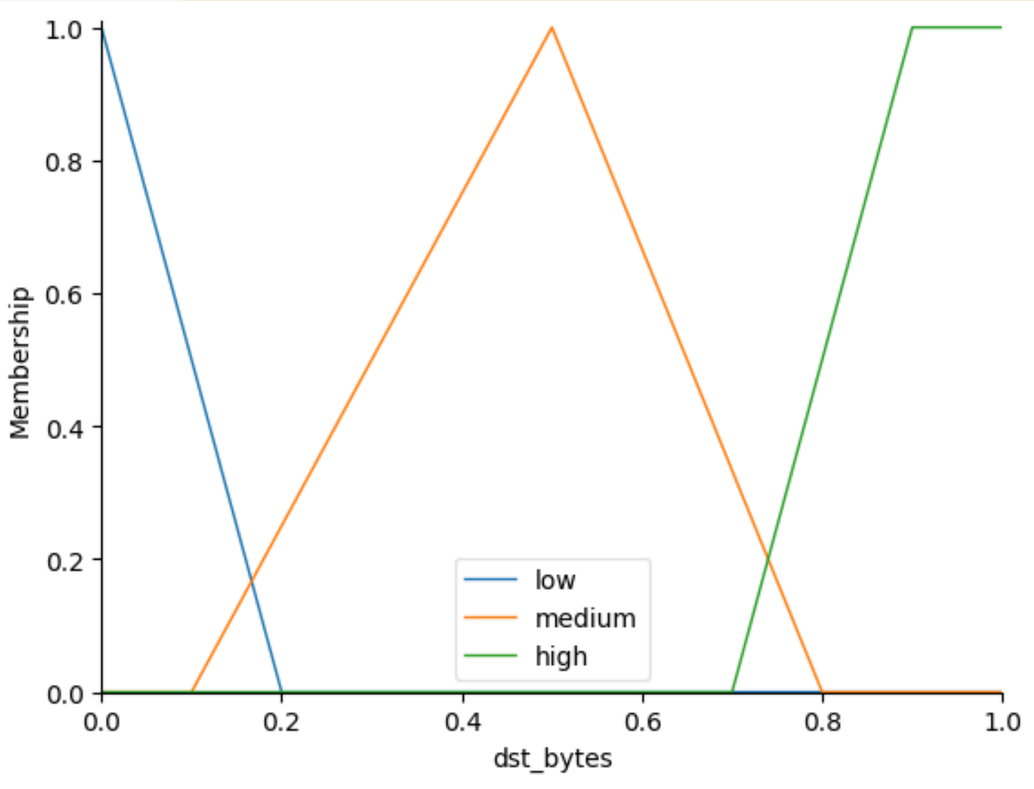
SERROR_RATE



FUZZIFIKASI

```
'dst_bytes': {
    'low':    ([0, 0, 0.2], 'trimf'),
    'medium': ([0.1, 0.5, 0.8], 'trimf'),
    'high':   ([0.7, 0.9, 1, 1], 'trapmf'),
},
```

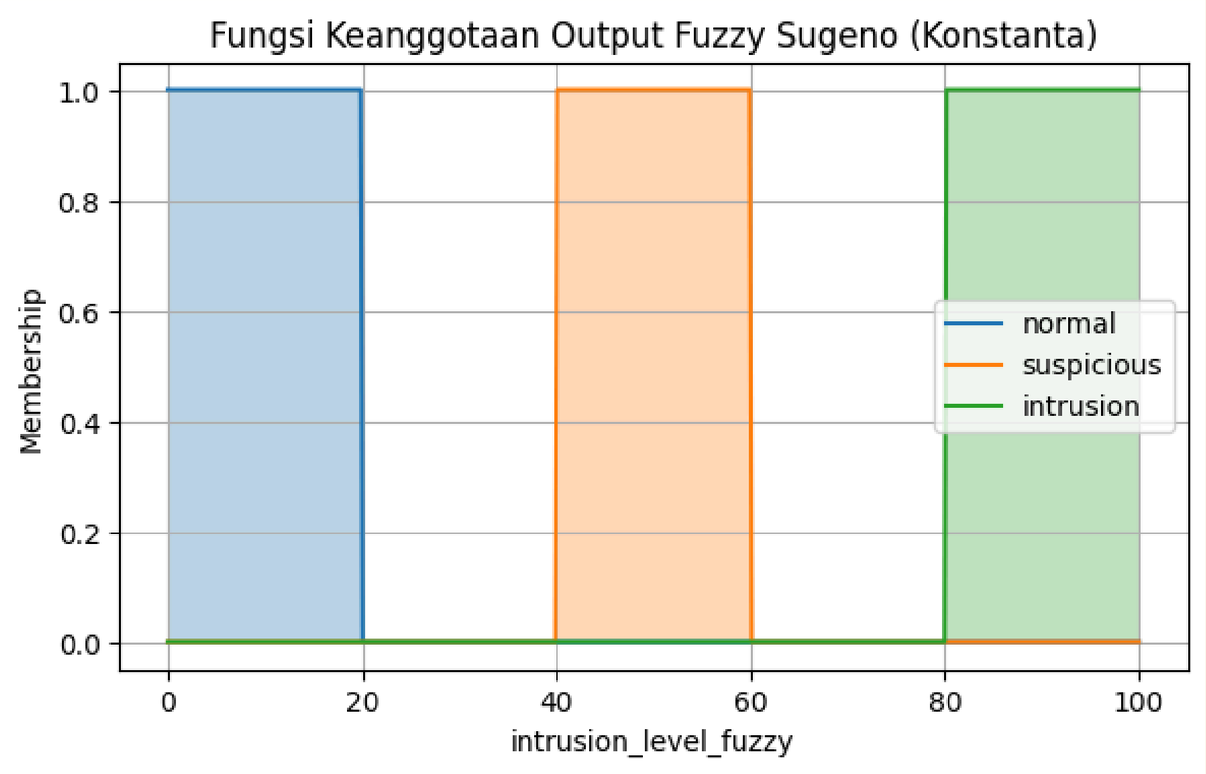
DST_BYTES



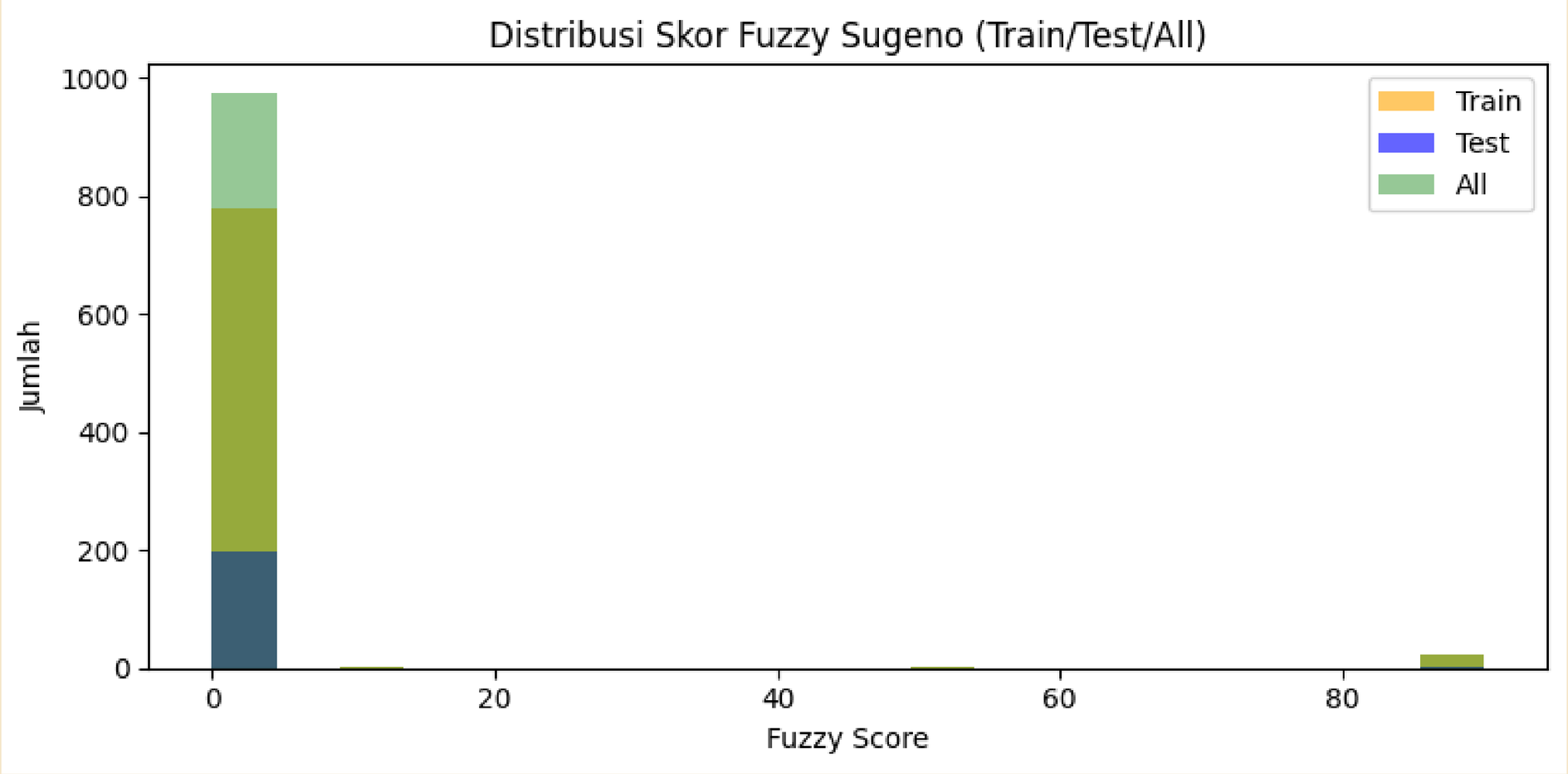
ATURAN FUZZY

```python
# 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'),
]
```

ATURAN FUZZY MENGHASILKAN OUTPUT BERUPA FUNGSI LINIER SEDERHANA DARI FITUR INPUT



INTERFERENSI DAN DEFUZZIFIKASI



PERBANDINGAN

- **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.**

```
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
```

PERBANDINGAN

```
== 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
```

- 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.

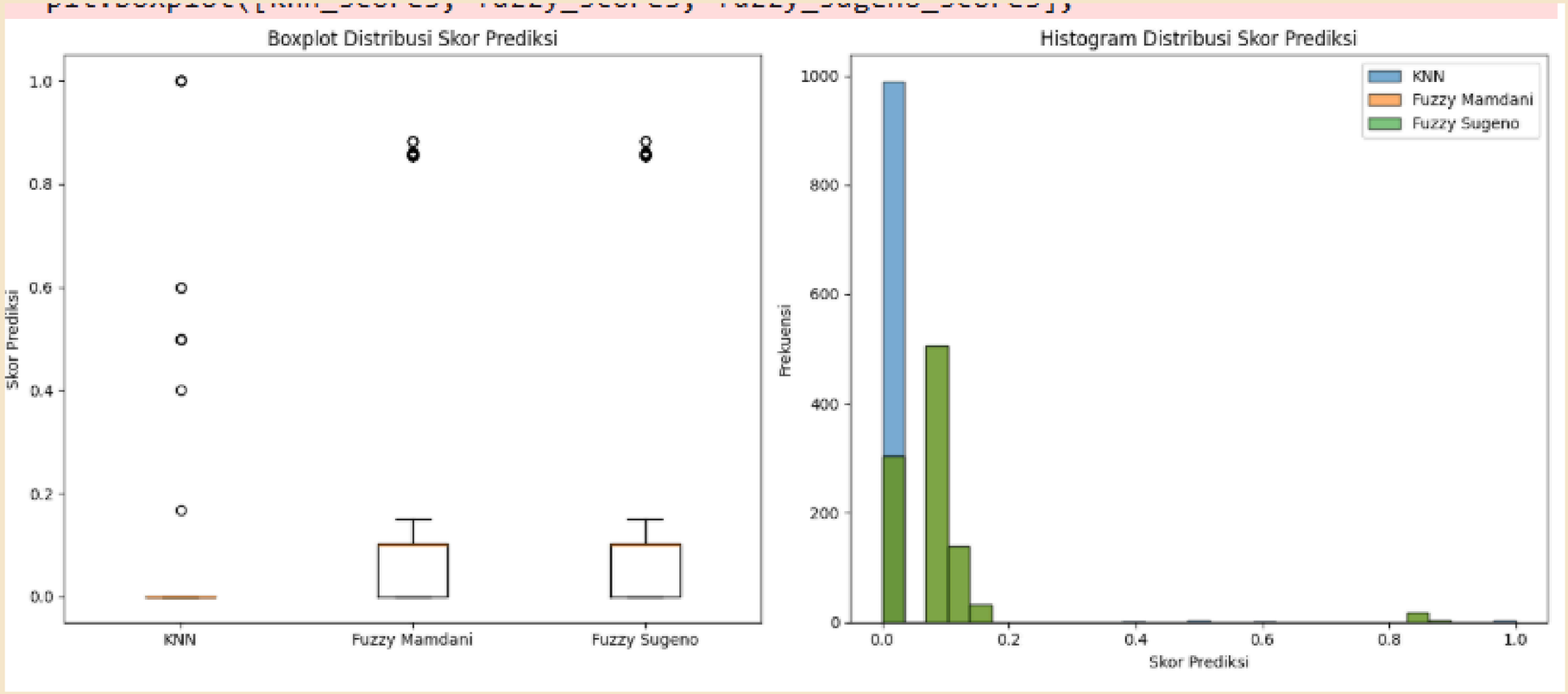
PERBANDINGAN

```
== 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
```

- **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.**

PERBANDINGAN

DISTRIBUSI SKOR



KESIMPULAN

- 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.

- 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.

REFERENSI

- Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1), 21-27.
- Takagi, T., Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. IEEE Transactions on Systems, Man, and Cybernetics, 15(1), 116-132.
- Ross, T. J. (2010). Fuzzy Logic with Engineering Applications (3rd ed.). Wiley.

THANK YOU