

Tugas Besar Dasar Kecerdasan Artifisial

# **Prediksi Cuaca Harian Menggunakan Fuzzy Inference: Mamdani**

Nabila Putri Azhari (103012300316)

Jeany Ferliza Nayla (103012300357)



# PENDAHULUAN



- Perubahan cuaca yang tak terduga menjadi tantangan bagi sektor penting seperti pertanian.
- Tugas ini menggunakan Fuzzy Inference System (FIS) metode Mamdani untuk memprediksi cuaca harian berdasarkan suhu, kelembapan, dan angin.
- Contoh kasus difokuskan di Kabupaten Karawang, daerah sentra padi yang sangat bergantung pada cuaca. Sistem ini diharapkan membantu petani mengambil keputusan yang lebih tepat dan adaptif terhadap perubahan iklim.

# METODE

## PAPARAN, STATISTIK, DAN SUMBER DARI DATASET YANG DIGUNAKAN

Berikut ini adalah cuplikan (preview) dari dataset yang digunakan, yang memuat beberapa kolom untuk prediksi cuaca harian

	date	precipitation	temp_max	temp_min	wind	weather
0	2012-01-01	0.0	12.8	5.0	4.7	drizzle
1	2012-01-02	10.9	10.6	2.8	4.5	rain
2	2012-01-03	0.8	11.7	7.2	2.3	rain
3	2012-01-04	20.3	12.2	5.6	4.7	rain
4	2012-01-05	1.3	8.9	2.8	6.1	rain
5	2012-01-06	2.5	4.4	2.2	2.2	rain
6	2012-01-07	0.0	7.2	2.8	2.3	rain
7	2012-01-08	0.0	10.0	2.8	2.0	sun
8	2012-01-09	4.3	9.4	5.0	3.4	rain
9	2012-01-10	1.0	6.1	0.6	3.4	rain

Sumber Dataset cuaca harian Indonesia diambil dari Kaggle melalui tautan:

<https://www.kaggle.com/datasets/ananthr1/weather-prediction>

# METODE



Dataset ini terdiri dari data harian cuaca dengan atribut-atribut sebagai berikut:

Date : Tanggal pencatatan data cuaca (format: YYYY-MM-DD)

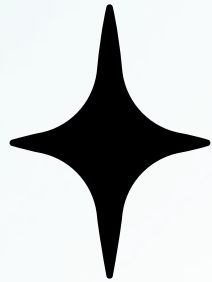
precipitation : Curah hujan dalam milimeter (mm)

temp max : Suhu maksimum harian dalam derajat celcius(°C)

temp\_min : Suhu minimum harian dalam derajat celcius(°C)

wind : Kecepatan angin harian dalam km/jam

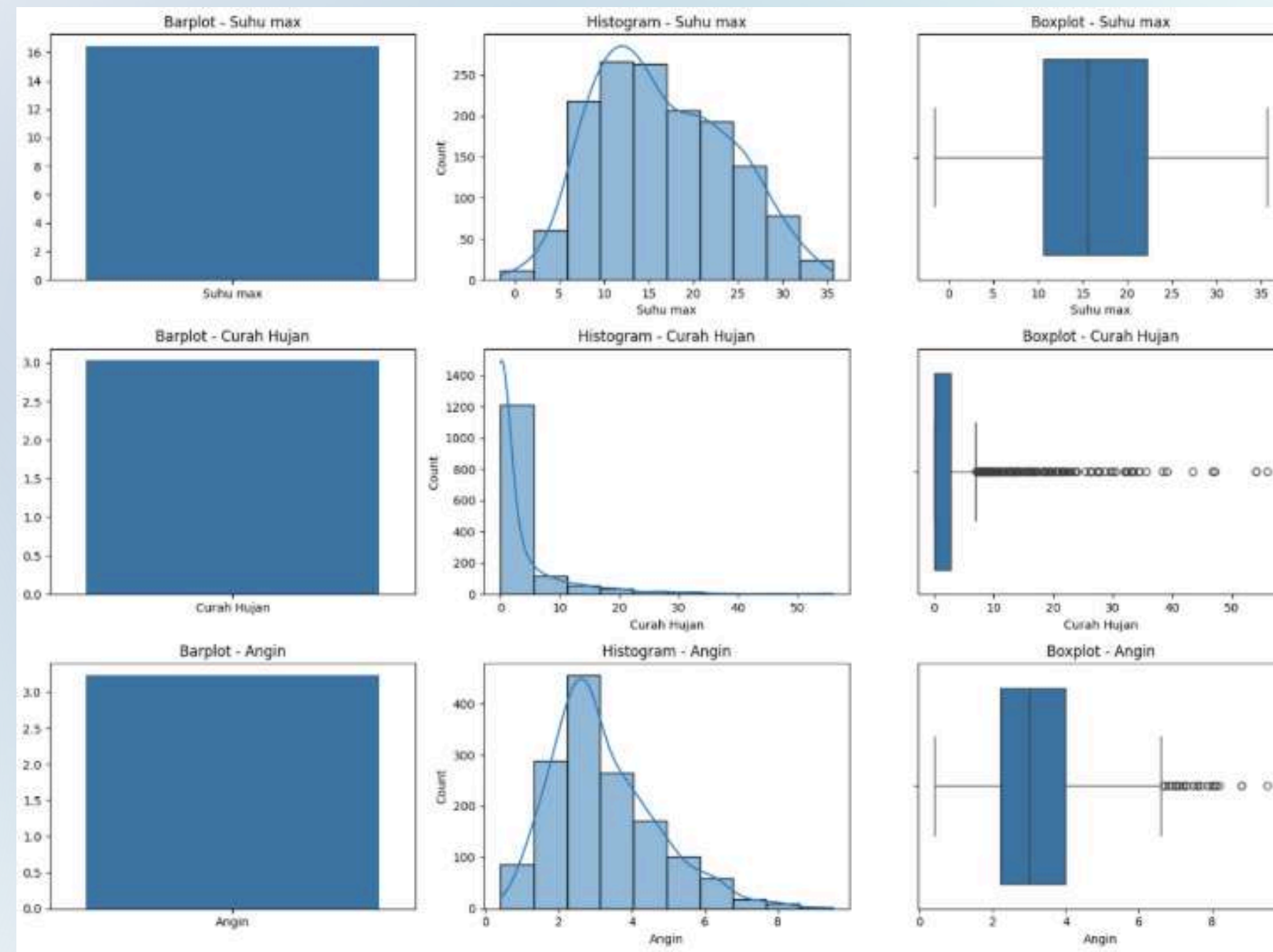
weather : Kategori kondisi cuaca(drizzle, rain, sun) sebagai output





# STATISTIK

## Statistik Visualisasi: Suhu max, Curah Hujan, dan Angin



# A. PAPARAN PREPROCESSING DATASET

Dataset sudah bersih

Cleaning (Drop Nan dan Drop Duplicate)

```
df.isna().sum() #nan  
df = df.drop_duplicates() #duplicate
```

```
date      0  
Curah Hujan 0  
Suhu max  0  
Suhu min  0  
Angin     0  
Cuaca Dataset Asli 0  
dtype: int64
```

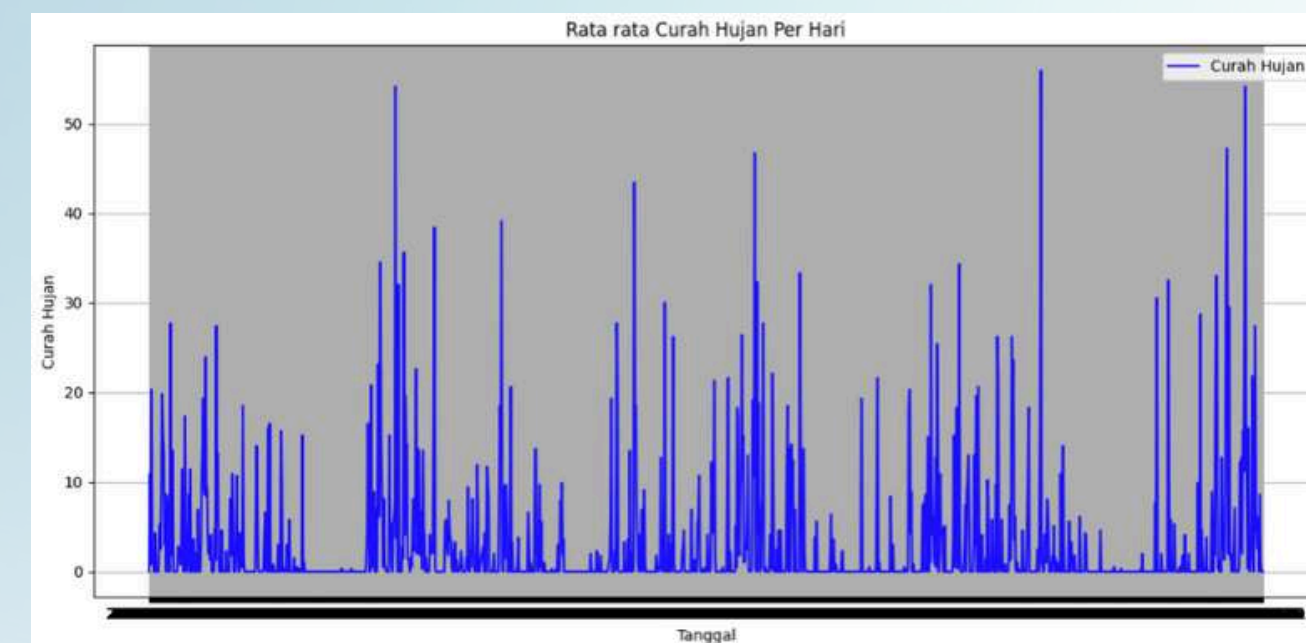
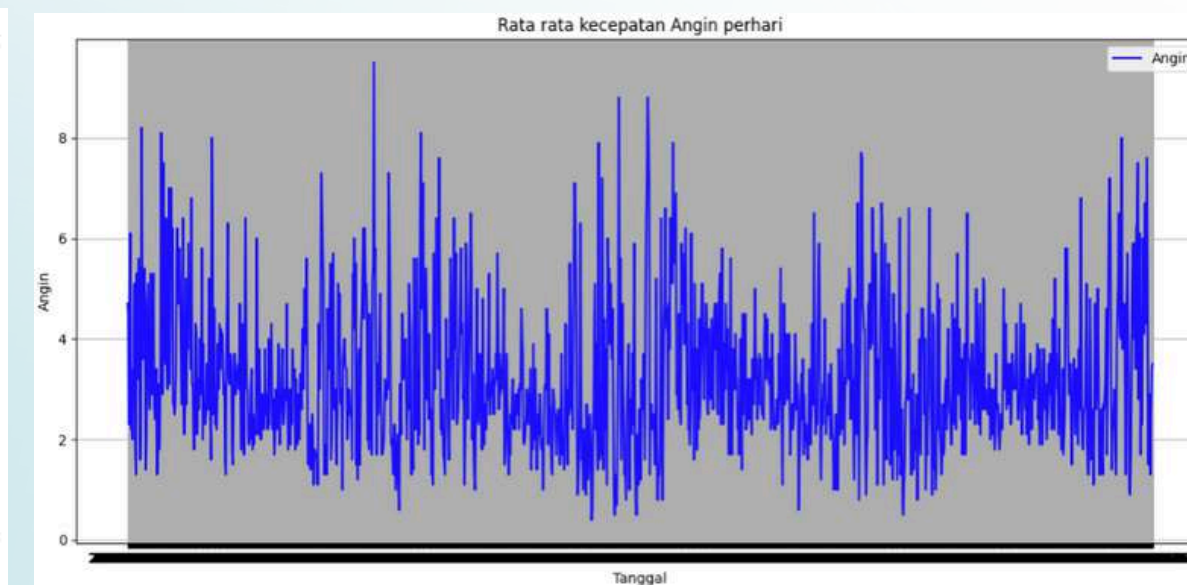
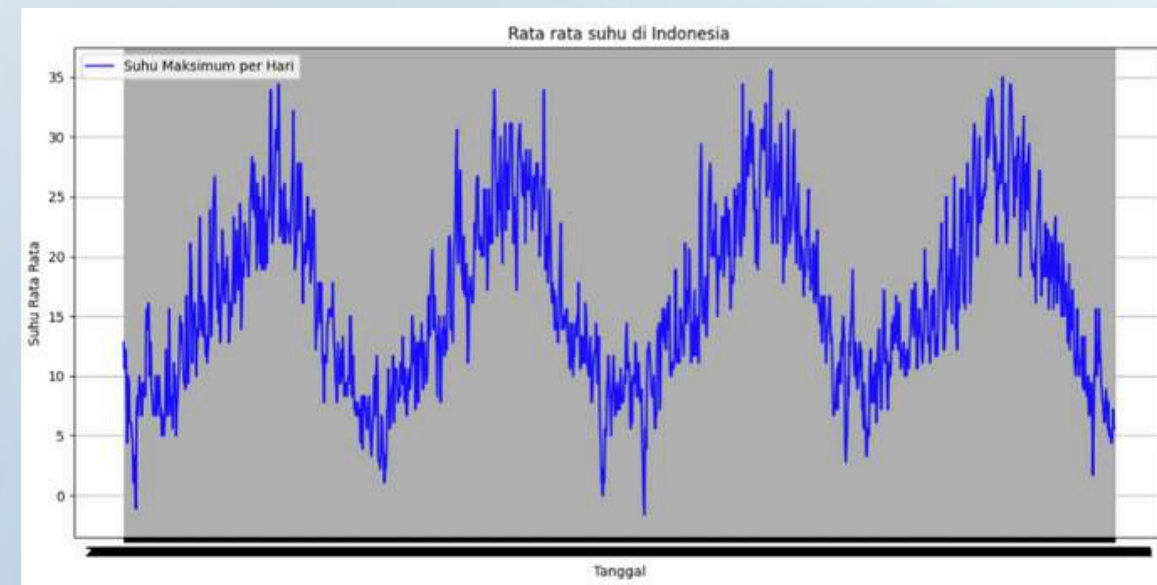
Feature Selection

```
fitur = ["Suhu max", "Curah Hujan", "Angin"]  
df = df[fitur]  
df.head()
```

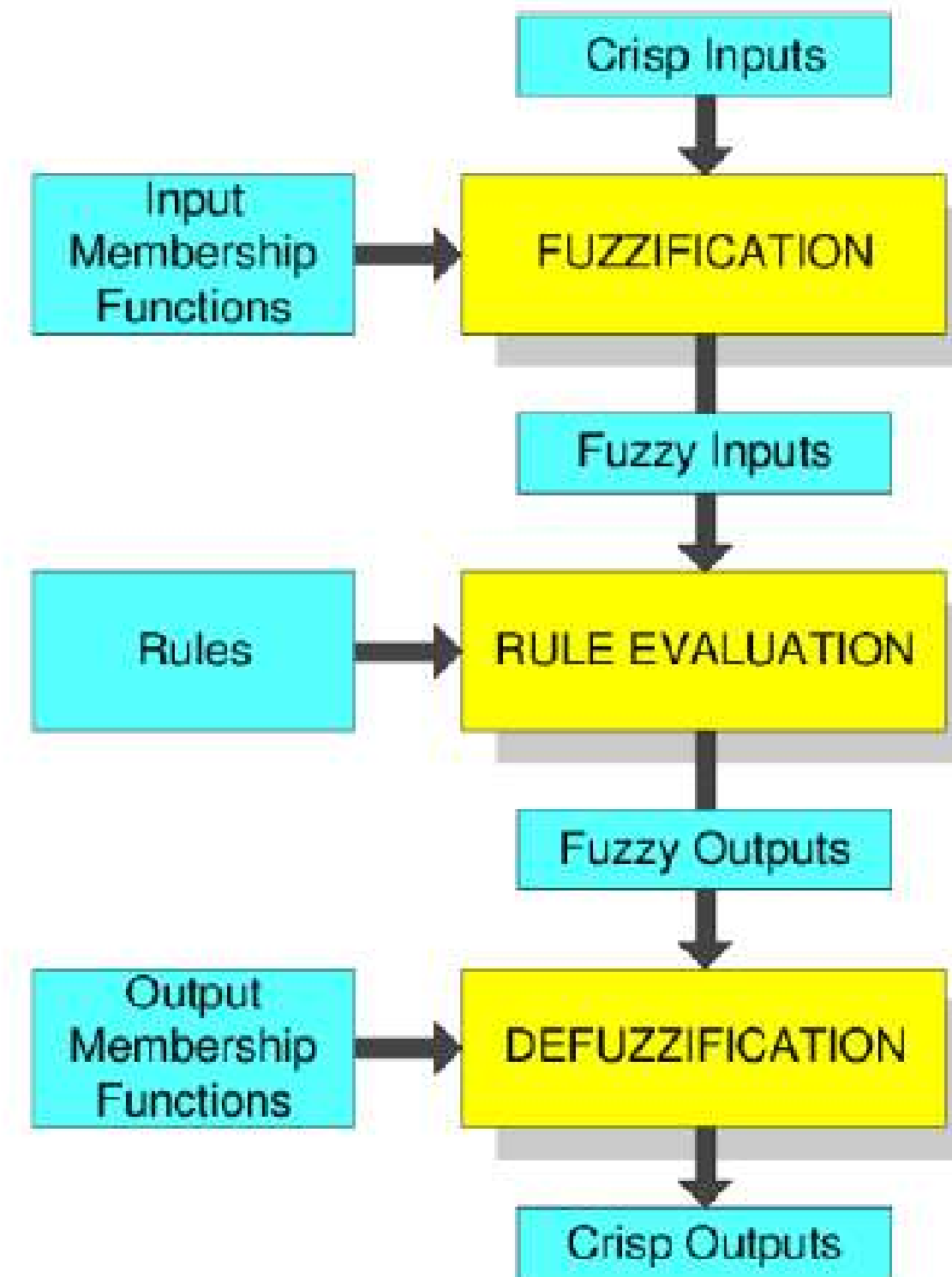
```
✓ 0.0s
```

	Suhu max	Curah Hujan	Angin
0	12.8	0.0	4.7
1	10.6	10.9	4.5
2	11.7	0.8	2.3
3	12.2	20.3	4.7
4	8.9	1.3	6.1

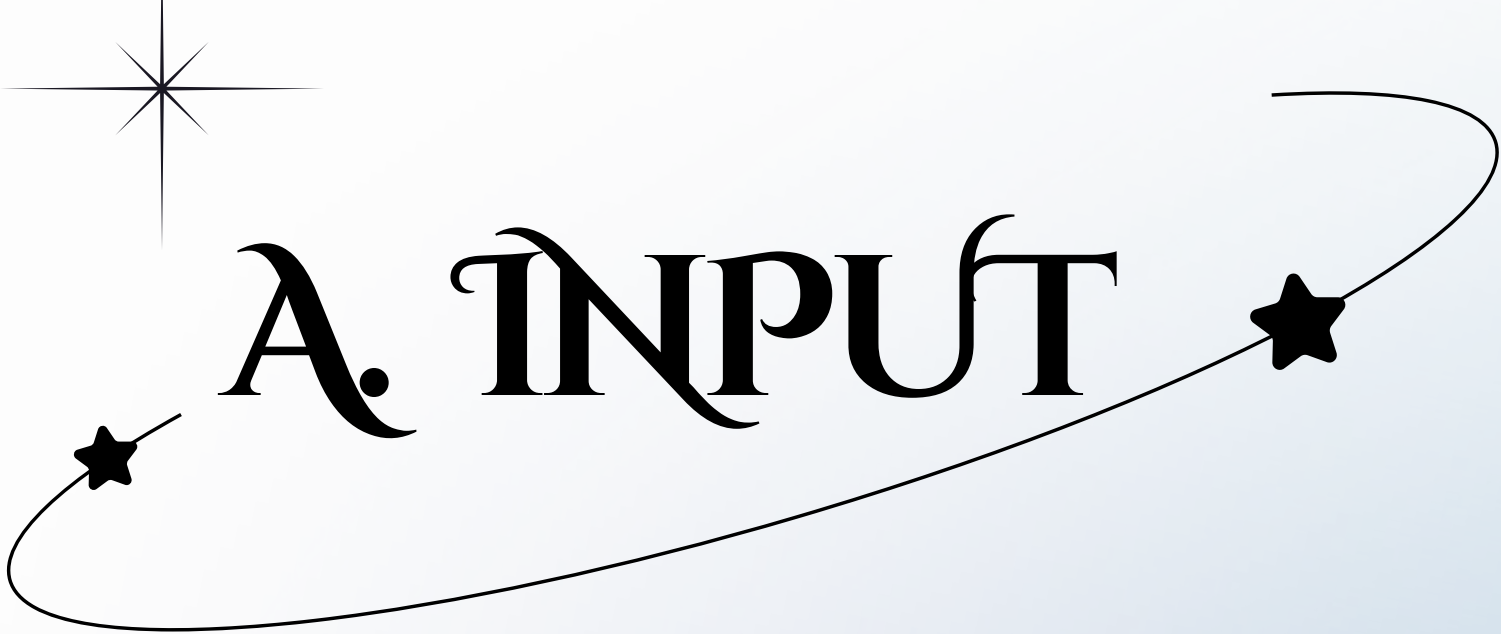
# B. EXPLORATORY DATA ANALYSIS (EDA)



# PROSES FUZZY







# A. INPUT

- Suhu : Dingin, Normal, Panas
- Curah Hujan : Rendah, Sedang, Tinggi
- Angin : Pelan, Sedang, Kencang

## 1. Linguistik Variabel

- Batas Batas Linguistik:

1. suhu\_dingin:  $\leq 20$
2. suhu\_normal: 15 - 30
3. suhu\_panas:  $\geq 24$
4. hujan\_rendah:  $\leq 10$
5. hujan\_sedang: 15-25
6. hujan\_tinggi:  $\geq 20$
7. angin\_pelan:  $\leq 3$
8. angin\_sedang: 2 - 6.5
9. angin\_kencang:  $\geq 5.5$



# A. INPUT

## 2. Fuzzifikasi Input

Proses ini mengubah input yang jelas (crisp input) menjadi nilai fuzzy dengan menggunakan fungsi keanggotaan.

Suhu

```
def suhu_dingin(suhu):  
    if suhu <= 0:  
        return 1  
    elif 0 < suhu <= 20:  
        return (20 - suhu) / (20 - 0)  
    else:  
        return 0  
  
def suhu_normal(suhu):  
    if suhu <= 15 or suhu >= 30:  
        return 0  
    elif 15 < suhu <= 22:  
        return (suhu - 15) / (22 - 15)  
    elif 22 < suhu < 30:  
        return (30 - suhu) / (30 - 22)  
    else:  
        return (30 - suhu) / (30 - 22)  
  
def suhu_panas(suhu):  
    if suhu <= 24:  
        return 0  
    elif 24 < suhu <= 30:  
        return (suhu - 24) / (30 - 24)  
    else:  
        return 1
```

Hujan

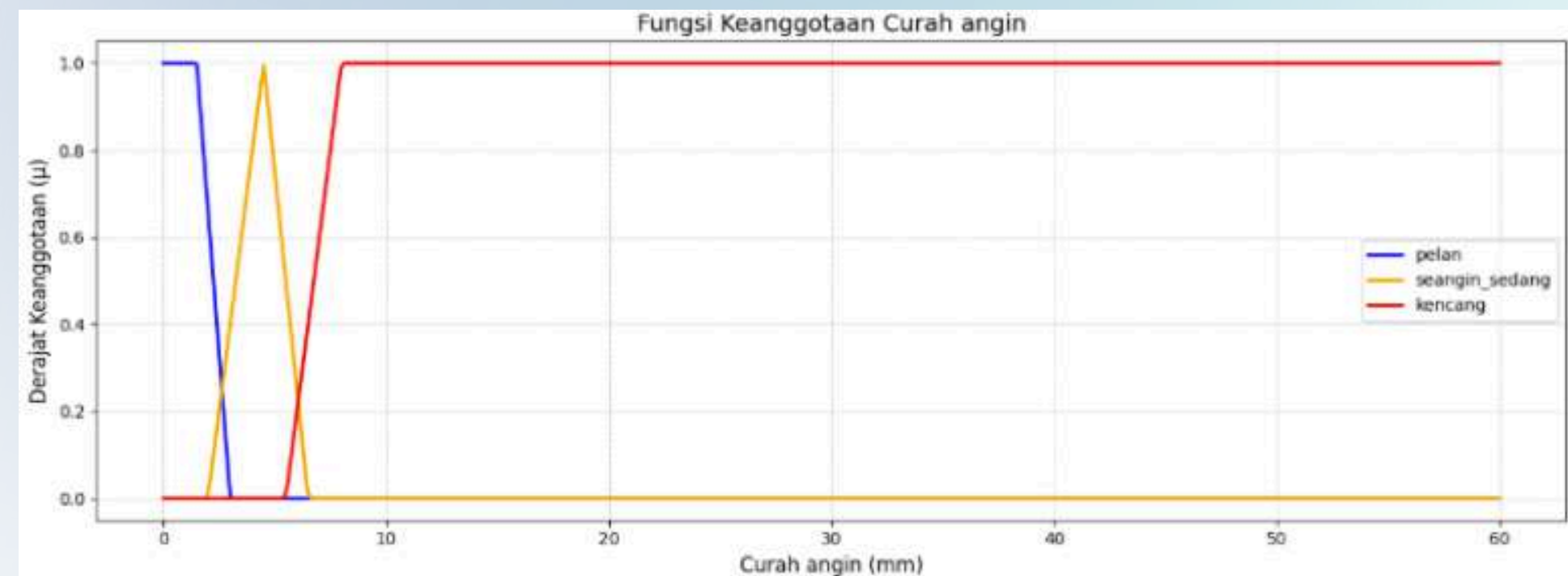
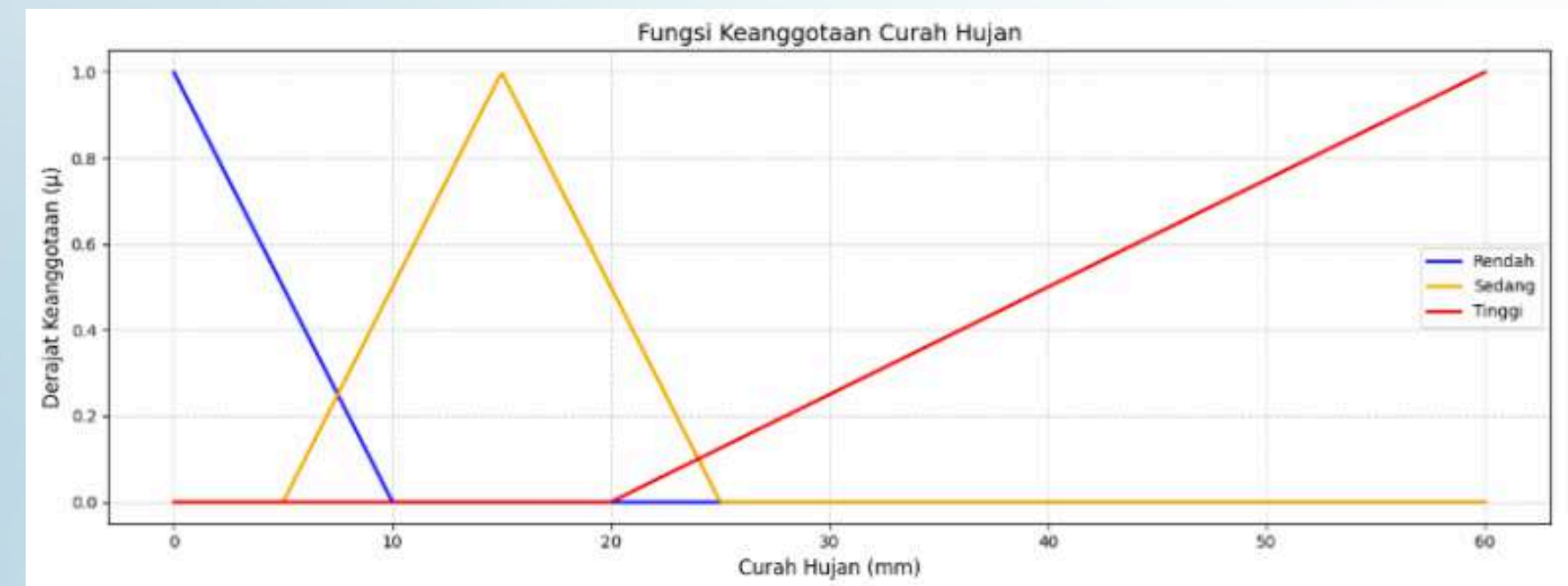
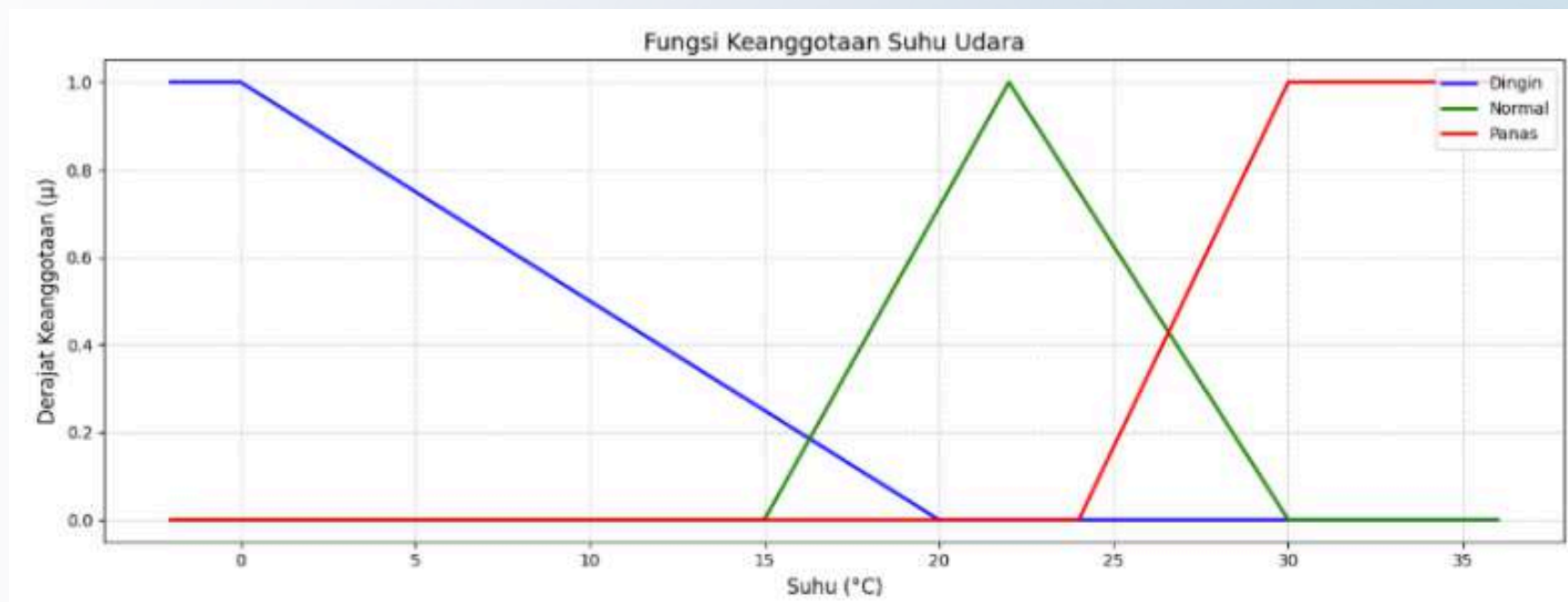
```
#CURAH HUJAN  
def hujan_rendah(hujan):  
    if hujan <= 0:  
        return 1  
    elif 0 < hujan <= 10:  
        return (10 - hujan) / (10 - 0)  
    else:  
        return 0  
  
def hujan_sedang(hujan):  
    if hujan <= 5 or hujan >= 25:  
        return 0  
    elif 5 < hujan <= 15:  
        return (hujan - 5) / (15 - 5)  
    elif 15 < hujan < 25:  
        return (25 - hujan) / (25 - 15)  
    else:  
        return 1  
  
def hujan_tinggi(hujan):  
    if hujan <= 20:  
        return 0  
    elif hujan <= 60:  
        return (hujan - 20) / (60 - 20)  
    else:  
        return 1
```

Angin

```
#ANGIN  
def angin_pelan(kecepatan):  
    if kecepatan <= 1.5:  
        return 1.0  
    elif 1.5 < kecepatan < 3:  
        return (3 - kecepatan) / (3 - 1.5)  
    else:  
        return 0.0  
  
def angin_sedang(kecepatan):  
    if 2 < kecepatan < 4.5:  
        return (kecepatan - 2) / (4.5 - 2)  
    elif 4.5 <= kecepatan <= 6.5:  
        return (6.5 - kecepatan) / (6.5 - 4.5)  
    else:  
        return 0.0  
  
def angin_kencang(kecepatan):  
    if kecepatan <= 5.5:  
        return 0.0  
    elif 5.5 < kecepatan < 8:  
        return (kecepatan - 5.5) / (8 - 5.5)  
    else:  
        return 1.0
```

# A. INPUT

## Visualisasi Fungsi Keanggotaan Input





# FUZZY ROLE EVALUATION



```

rules = [
    # Rain kuat saat suhu dingin/normal, hujan tinggi/sedang, angin pelan/sedang
    ("dingin", "tinggi", "pelan", "Rain"),
    ("dingin", "tinggi", "sedang", "Rain"),
    ("dingin", "sedang", "pelan", "Rain"),
    ("dingin", "sedang", "sedang", "Rain"),
    ("normal", "tinggi", "pelan", "Rain"),
    ("normal", "tinggi", "sedang", "Rain"),
    ("normal", "sedang", "pelan", "Rain"),
    ("normal", "sedang", "sedang", "Rain"),

    # Drizzle saat hujan sedang + suhu normal/panas + angin pelan
    ("normal", "sedang", "pelan", "Drizzle"),
    ("normal", "sedang", "sedang", "Drizzle"),
    ("panas", "sedang", "pelan", "Drizzle"),
    ("panas", "sedang", "sedang", "Drizzle"),

    # Snow hanya jika suhu dingin dan angin kencang
    ("dingin", "tinggi", "kencang", "Snow"),

    # Fog dari kombinasi suhu dingin/normal dan hujan rendah
    ("dingin", "rendah", "pelan", "Fog"),
    ("dingin", "rendah", "sedang", "Fog"),
    ("normal", "rendah", "pelan", "Fog"),
    ("normal", "rendah", "sedang", "Fog"),

    # Sun jika suhu panas dan hujan rendah
    ("panas", "rendah", "pelan", "Sun"),
    ("panas", "rendah", "sedang", "Sun"),
    ("panas", "rendah", "kencang", "Sun"),
    ("normal", "rendah", "sedang", "Sun"),
    ("normal", "rendah", "kencang", "Sun"),
    ("normal", "rendah", "sedang", "Sun"),
    ("normal", "rendah", "kencang", "Sun"),
    ("dingin", "rendah", "kencang", "Sun"),
]

for suhu_label, hujan_label, angin_label, output_label in rules:
    μ = min(
        suhu_vals[suhu_label],
        hujan_vals[hujan_label],
        angin_vals[angin_label]
    )
    result[output_label] = max(result[output_label], μ)

return result

```

SuhuMax	CurahHujan	Angin	Prediksi Cuaca
Dingin	Tinggi	Pelan	Rain
Dingin	Tinggi	Sedang	Rain
Dingin	Tinggi	Kencang	Snow
Dingin	Sedang	Pelan	Fog
Dingin	Sedang	Sedang	Fog
Dingin	Sedang	Kencang	Fog
Dingin	Rendah	Pelan	Fog
Dingin	Rendah	Sedang	Fog
Dingin	Rendah	Kencang	Fog
Normal	Tinggi	Pelan	Rain
Normal	Tinggi	Sedang	Rain
Normal	Tinggi	Kencang	Rain
Normal	Sedang	Pelan	Drizzle
Normal	Sedang	Sedang	Drizzle
Normal	Sedang	Kencang	Drizzle
Normal	Rendah	Pelan	Fog
Normal	Rendah	Sedang	Fog
Normal	Rendah	Kencang	Fog
Panas	Tinggi	Pelan	Rain
Panas	Tinggi	Sedang	Rain
Panas	Tinggi	Kencang	Rain
Panas	Sedang	Pelan	Drizzle
Panas	Sedang	Sedang	Drizzle
Panas	Sedang	Kencang	Drizzle
Panas	Rendah	Pelan	Sun
Panas	Rendah	Sedang	Sun
Panas	Rendah	Kencang	Sun

OUTPUT



# DEFINISI OUTPUT

```
def output_snow(x):  
    if x <= 5:  
        return 1  
    elif 5 < x < 15:  
        return (15 - x) / (15 - 5)  
    else:  
        return 0
```

```
def output_rain(x):  
    if x <= 20 or x >= 40:  
        return 0  
    elif 20 < x < 30:  
        return (x - 20) / (30 - 20)  
    elif 30 <= x < 40:  
        return (40 - x) / (40 - 30)  
    else:  
        return 0
```

```
def output_drizzle(x):  
    if x <= 35 or x >= 65:  
        return 0  
    elif 35 < x < 50:  
        return (x - 35) / (50 - 35)  
    elif 50 <= x < 65:  
        return (65 - x) / (65 - 50)  
    else:  
        return 0
```

```
def output_fog(x):  
    if x <= 40:  
        return 0  
    elif 40 < x <= 60:  
        return (x - 40) / (60 - 40)  
    elif 60 < x < 80:  
        return 1  
    elif 80 <= x <= 90:  
        return (90 - x) / (90 - 80)  
    else:  
        return 0
```

```
def output_sun(x):  
    if x <= 60:  
        return 0  
    elif 60 < x <= 75:  
        return (x - 60) / (75 - 60)  
    elif 75 < x < 90:  
        return 1  
    elif 90 <= x <= 100:  
        return (100 - x) / (100 - 90)  
    else:  
        return 0
```



# DEFUZZIFIKASI

- Proses ini mengubah output fuzzy menjadi nilai yang jelas (crisp output atau weight output).  
Menggunakan rumus Centroid:

$$\text{Centroid} = \frac{\sum_{i=1}^N x_i \cdot \mu(x_i)}{\sum_{i=1}^N \mu(x_i)}$$

# PROCESS DEFUZZIFIKASI

```
def defuzzifikasi(hasil):
    x = np.linspace(0, 100, 500)
    numerator = 0
    denominator = 0

    for xi in x:
        μ = max(
            min(hasil["Snow"], output_snow(xi)),
            min(hasil["Rain"], output_rain(xi)),
            min(hasil["Drizzle"], output_drizzle(xi)),
            min(hasil["Fog"], output_fog(xi)),
            min(hasil["Sun"], output_sun(xi))
        )
        numerator += xi * μ
        denominator += μ

    return numerator / denominator if denominator != 0 else 0
```

```
def interpretasi_fuzzy(skor):
    sn = output_snow(skor)
    r = output_rain(skor)
    d = output_drizzle(skor)
    f = output_fog(skor)
    su = output_sun(skor)

    max_μ = max(sn, r, d, f, su)
    if max_μ == sn:
        return "Snow"
    elif max_μ == r:
        return "Rain"
    elif max_μ == d:
        return "Drizzle"
    elif max_μ == f:
        return "Fog"
    else:
        return "Sun"
```

```
def classify_weather(row):
    hasil_inferensi = fuzzy_inference(row['Suhu max'], row['Curah Hujan'], row['Angin'])
    return defuzzifikasi(hasil_inferensi)

df['Model Mamdani'] = df.apply(classify_weather, axis=1)
df['Kategori Cuaca Mamdani'] = df['Model Mamdani'].apply(interpretasi_fuzzy)

df[['Suhu max', 'Curah Hujan', 'Angin', 'Cuaca Dataset Asli', 'Model Mamdani', 'Kategori Cuaca Mamdani']].tail(20)
```

**HASIL**



# HASIL MAMDANI

	Suhu max	Curah Hujan	Angin	Cuaca Dataset Asli	Model Mamdani	Kategori Cuaca Mamdani
1441	8.9	16.0	5.6	rain	30.000329	Rain
1442	7.8	1.3	6.1	rain	71.577187	Fog
1443	7.8	0.0	1.7	sun	66.410882	Fog
1444	6.7	1.5	2.9	rain	65.863549	Fog
1445	6.1	3.6	2.3	rain	66.103090	Fog
1446	6.7	21.8	6.0	rain	26.994022	Rain
1447	8.9	18.5	5.1	rain	30.000382	Rain
1448	8.3	0.0	4.1	fog	66.358534	Fog
1449	7.8	4.3	6.7	rain	80.564460	Sun
1450	5.6	27.4	4.3	rain	29.999484	Rain
1451	7.8	4.6	5.0	rain	66.262876	Fog
1452	5.0	6.1	7.6	rain	80.464243	Sun
1453	5.6	2.5	4.3	rain	66.634560	Fog
1454	5.0	5.8	1.5	rain	63.219381	Fog
1455	4.4	0.0	2.5	sun	65.801800	Fog
1456	4.4	8.6	2.9	rain	48.796425	Drizzle
1457	5.0	1.5	1.3	rain	66.693333	Fog
1458	7.2	0.0	2.6	fog	65.646896	Fog
1459	5.6	0.0	3.4	sun	66.305638	Fog
1460	5.6	0.0	3.5	sun	66.390112	Fog

# AKURASI

SMAPE (*Symmetric Mean Absolute Percentage Error*)

$$SMAPE = \frac{1}{n} \times \sum \frac{|forecast\ value - actual\ value|}{(|actual\ value| + |forecast\ value|)/2}$$

```
cuaca_ke_angka = {
    'snow': 10,
    'rain': 30,
    'drizzle': 45,
    'fog': 60,
    'sun': 90
}

df['weather_num'] = df['Cuaca Dataset Asli'].str.lower().map(cuaca_ke_angka)
df['pred_mamdani_num'] = df['Kategori Cuaca Mamdani'].str.lower().map(cuaca_ke_angka)
df_valid_mamdani = df.dropna(subset=['weather_num', 'pred_mamdani_num'])

# Hitung SMAPE
def smape(y_true, y_pred):
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)
    denominator = (np.abs(y_true) + np.abs(y_pred)) / 2
    diff = np.abs(y_true - y_pred) / denominator
    diff[denominator == 0] = 0
    return np.mean(diff) * 100

nilai_smape_mamdani = smape(df_valid_mamdani['weather_num'], df_valid_mamdani['pred_mamdani_num'])
print(f'SMAPE Model Mamdani: {nilai_smape_mamdani:.2f}%')
```

SMAPE Model Mamdani: 43.27%

MAE (*Mean Absolute Error*)

$$MAE = \frac{1}{n} \sum_{i=1}^n |x_i - x|$$

```
def mae(acutal, predicted):
    return np.mean(np.abs(acutal-predicted))

print(f"Mean Abosulte Error Mamdani: {mae(df['weather_num'], df['pred_mamdani_num'])}")
```

Mean Abosulte Error Mamdani: 23.870636550308006



# ANALISIS

## Analisis

### 1. Beberapa Faktor Penyebab mendapat akurasi tersebut

- Karakteristik Data
  - Data yang digunakan mungkin memiliki variabilitas tinggi atau terdapat outlier yang signifikan sehingga mempengaruhi performa model.
- Model dan Parameter Fuzzy
  - Fungsi keanggotaan (membership functions) yang digunakan dalam model fuzzy mungkin kurang representatif atau tidak teroptimasi dengan baik.
  - Rule base atau aturan fuzzy yang digunakan bisa jadi kurang lengkap atau tidak sesuai dengan pola data yang ada.

### 2. Kesimpulan Karakteristik Fuzzy Logic

- Berbasis aturan linguistik, misalnya: "Jika suhu panas dan curah hujan tinggi maka cuaca kemungkinan hujan."
- Tidak memerlukan data latih dalam jumlah besar.
- Sangat bagus untuk interpretabilitas dan logika manusia.
- Tidak belajar dari data → hanya mengandalkan aturan statis buatan manusia.
- Tidak ada proses optimasi parameter otomatis.



MACHINE LEARNING

XGBOOST

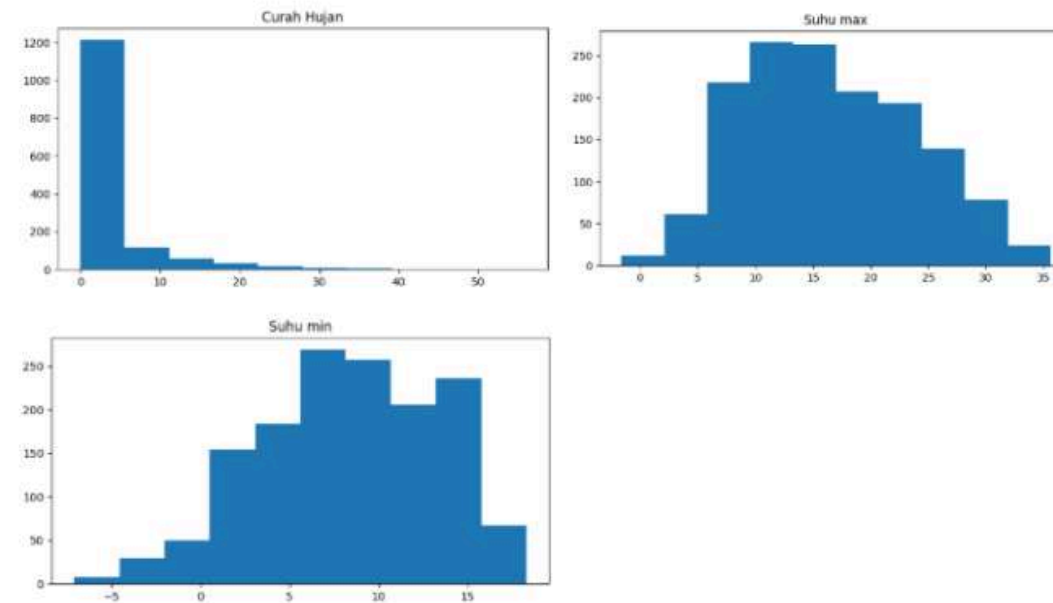
# ML Process

## I. Import Library

```
from sklearn.preprocessing import LabelEncoder
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

## II. Visualisasi

```
num=['Curah Hujan', 'Suhu max', 'Suhu min', 'Angin']
for col in num:
    plt.figure(figsize=(8,4))
    plt.hist(df[col])
    plt.title(col)
    plt.show()
```



## III. Label Encoding

```
df['Cuaca Dataset Asli']=LabelEncoder().fit_transform(df['Cuaca Dataset Asli'])
```

## IV. Split Data

```
x=df[['Curah Hujan', 'Suhu max', 'Suhu min', 'Angin']]
y=df[['Cuaca Dataset Asli']]
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8,random_state=10)
```

## V. Modelling

```
from xgboost import XGBClassifier

model1=XGBClassifier()
model1.fit(x_train,y_train)
predict=model1.predict(x_test)
```

## VI. Akurasi

```
print(accuracy_score(y_test,predict)*100)
```

83.61774744027304

```
from sklearn.metrics import mean_absolute_error
import numpy as np

# Hitung MAE
mae = mean_absolute_error(y_test, predict)
print("MAE:", mae)

# Hitung SMAPE
def smape(actual, predicted):
    actual, predicted = np.array(actual), np.array(predicted)
    denominator = (np.abs(actual) + np.abs(predicted)) / 2
    diff = np.abs(actual - predicted) / denominator
    diff[denominator == 0] = 0.0 # untuk menghindari pembagian dengan nol
    return np.mean(diff) * 100
```

```
smape_score = smape(y_test, predict)
print("SMAPE:",smape_score)
```

✓ 0.0s

MAE: 0.44368600682593856  
SMAPE: 46.42101940894006

**PERBANDINGAN**

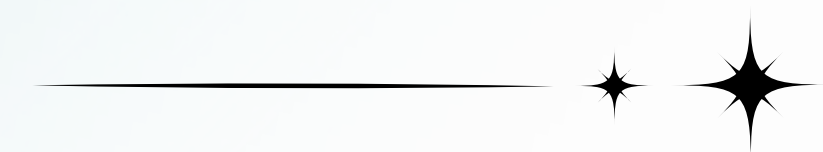
**FUZZY LOGIC**

**DAN**

**MACHINE LEARNING**



1. Jika kondisi cuaca kompleks, fuzzy tidak mampu mengikuti korelasi fitur-fitur seperti curah hujan, suhu, dan angin secara dinamis.
2. Jumlah aturan terbatas:
  - Semakin banyak kombinasi input, semakin kompleks aturan fuzzy yang diperlukan.
  - Jika jumlah aturan terlalu sedikit  $\rightarrow$  sistem jadi underfitting.
3. Tidak ada mekanisme koreksi kesalahan:
  - Fuzzy Mamdani tidak bisa “belajar dari kesalahan” seperti halnya XGBoost yang melakukan boosting.



# SOURCE CODE

[HTTPS://GITHUB.COM/NABILA-AZHARI/TUBES\\_DK&BLOB/main/WEATHERPREDICTIONFUZZYFIX.IPYNB](https://github.com/Nabila-Azhari/Tubes_DK&BLOB/main/WeatherPredictionFuzzyFix.ipynb)



# KESIMPULAN

- **Prediksi cuaca di Indonesia sangat penting**, mengingat iklim tropis yang dinamis dan berdampak pada sektor pertanian, transportasi, dan pariwisata.
- **Fuzzy Logic Mamdani** digunakan karena mampu menangani ketidakpastian data cuaca dan menghasilkan output linguistik yang mudah dipahami.
- **Kelebihan Mamdani:**
  - Interpretasi hasil yang jelas dan transparan.
  - Cocok untuk sistem yang membutuhkan penjelasan berbasis aturan.
- **Kekurangan Mamdani:**
  - Tidak mampu menangkap hubungan kompleks antar fitur (suhu, hujan, angin) secara dinamis.
  - Jumlah aturan yang terbatas menyebabkan risiko underfitting pada data yang kompleks.
  - Tidak memiliki kemampuan untuk belajar atau memperbaiki kesalahan seperti model machine learning.
- **Machine learning (XGBoost):**
  - Memiliki akurasi yang lebih tinggi dalam prediksi.
  - Dapat menangani hubungan fitur yang kompleks dan belajar dari kesalahan.
- Fuzzy cocok digunakan jika dibutuhkan interpretasi yang mudah dan sistem sederhana, sedangkan machine learning lebih unggul untuk akurasi dan adaptasi terhadap data kompleks.
- **Tujuan utama penelitian** adalah membandingkan efektivitas dan kepraktisan metode Fuzzy Mamdani dengan machine learning untuk menentukan pendekatan terbaik bagi prediksi cuaca di Indonesia.





# REFERENSI

Kurniawan, D., Haryanto, D., & Susilo, R. (2019). *Prediksi Curah Hujan Menggunakan Metode Fuzzy Mamdani*. Jurnal Informatika, 15(2), 85-92.

Astuti, R. & Wijaya, H. (2020). *Perbandingan Metode Fuzzy Mamdani untuk Prediksi Kelembaban Udara*. Jurnal Sains dan Informatika, 6(1), 25–31.

Prasetyo, D., Nugroho, R. A., & Saraswati, P. (2022). *Penerapan Fuzzy dan Machine Learning dalam Prediksi Cuaca Harian di Surabaya*. Prosiding Seminar Nasional Teknologi Informasi dan Komunikasi.

Ross, T. J. (2010). *Fuzzy Logic with Engineering Applications* (3rd ed.). Wiley.

Kaggle (2023). *Indonesia Climate Dataset*. Diakses dari:  
<https://www.kaggle.com/datasets/greegtitan/indonesia-climate>

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.

THANK YOU