TUBES DKA

PRESENTASI TUBES DKA

PENINJAUAN KELAYAKAN PENJUALAN MOBIL

Arfian Ghifari Mahya - 103012300337
Muhammad Rafif Taufiq - 103012300318

PENDAHULUAN

- Meninjau Penjualan Mobil saat ini, menilai layak apakah tidak mobil tersebut untuk dijual
- Data yang dipakai meliputi tahun produksi, kondisi kendaraan, dan jarak tempuh dari dataset kaggle.
- Sistem diharapkan dapat membantu penjual dan membuat keputusan yang lebih akurat dan adaptif terhadap preferensi pasar yang berbeda.

TUBES DKA

I. METODE

PAPARAN, STATISTIK DAN SUMBER DATASET

Berikut adalah preview dari dataset kami, yang menampilkan beberapa kolom tentang penjualan mobil

Sumber dataset kami, diambil dari tautan kaggle

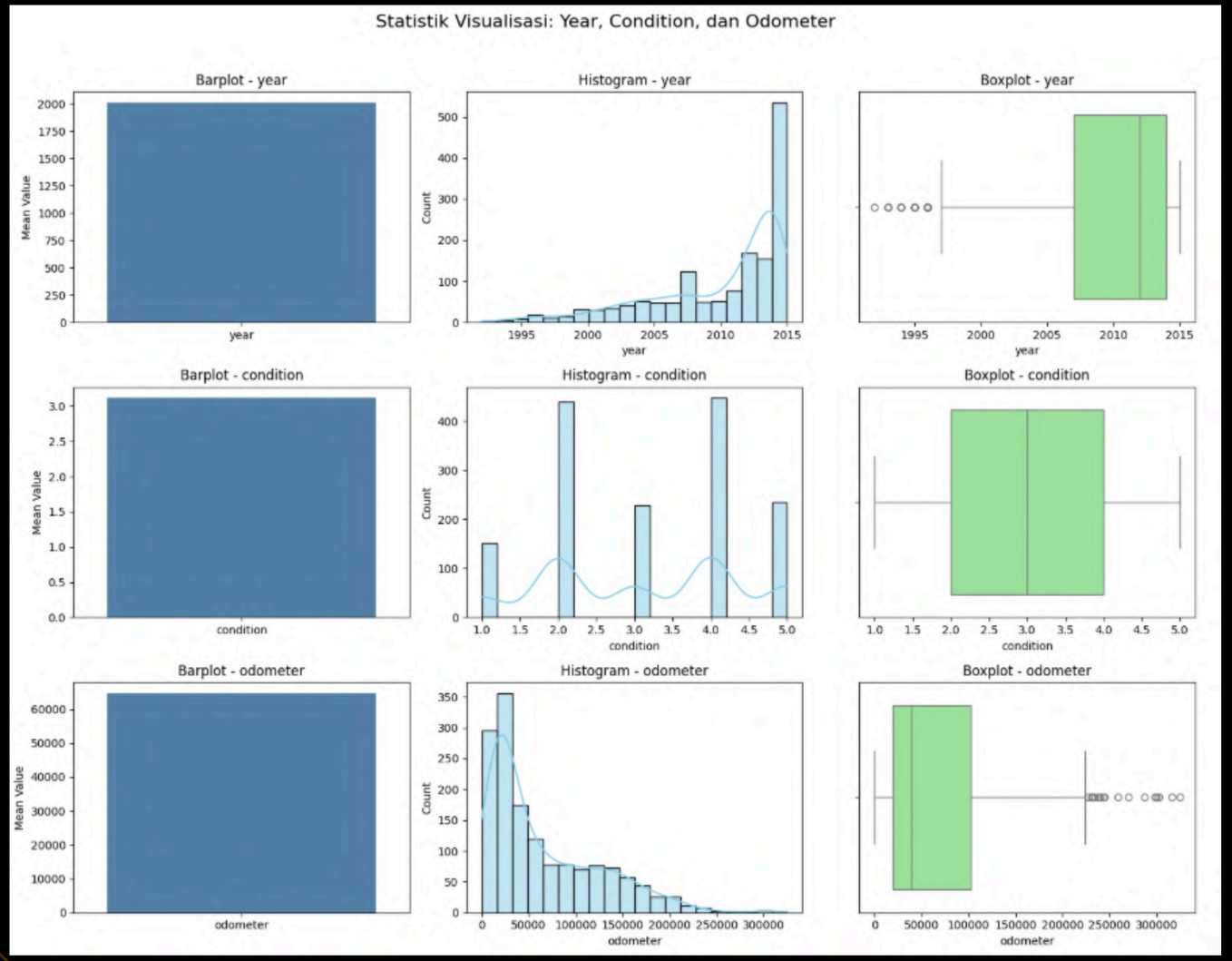
https://www.kaggle.com/datasets/syedanwarafridi/vehicle-sales-data/data

	year	make	model	trim	body	transmission	vin	state	condition	odometer	color	interior
0	2011	Chevrolet	Equinox	LTZ	SUV	automatic	2cnalfecxb6336309	il	3.0	64680.0	gray	black
1	2011	BMW	3 Series	328i xDrive	Sedan	automatic	wbapk5c57bf127600	nv	4.0	14505.0	blue	tan
2	2014	Chrysler	Town and Country	Touring	Minivan	automatic	2c4rc1bg9er255287	mi	5.0	20738.0	gold	black
3	2006	Subaru	B9 Tribeca	Limited 5-Passenger	SUV	automatic	4s4wx83c564409540	nc	3.0	133320.0	green	tan
4	2005	Dodge	Stratus	SXT	Sedan	automatic	1b3el46x15n653909	pa	2.0	137046.0	silver	gray
5	2006	Acura	TSX	Base	Sedan	manual	jh4cl95846c019471	nj	2.0	125744.0	red	beige
6	2011	Toyota	Camry	SE	Sedan	automatic	4t1bf3ek2bu759646	fl	2.0	45861.0	black	tan
7	2004	Nissan	Armada	LE	SUV	automatic	5n1aa08a24n700539	fl	2.0	125211.0	gray	gray
8	2012	Nissan	Altima	2.5 S	Sedan	automatic	1n4al2apxcn459486	nc	2.0	59224.0	gray	black
9	2003	Pontiac	Grand Am	GT1	Coupe	automatic	1g2nv12e93c134728	tx	1.0	1.0	silver	gray

TUBES DKA

STATISTIK

Statistik Visualisai; year, condition, odometer

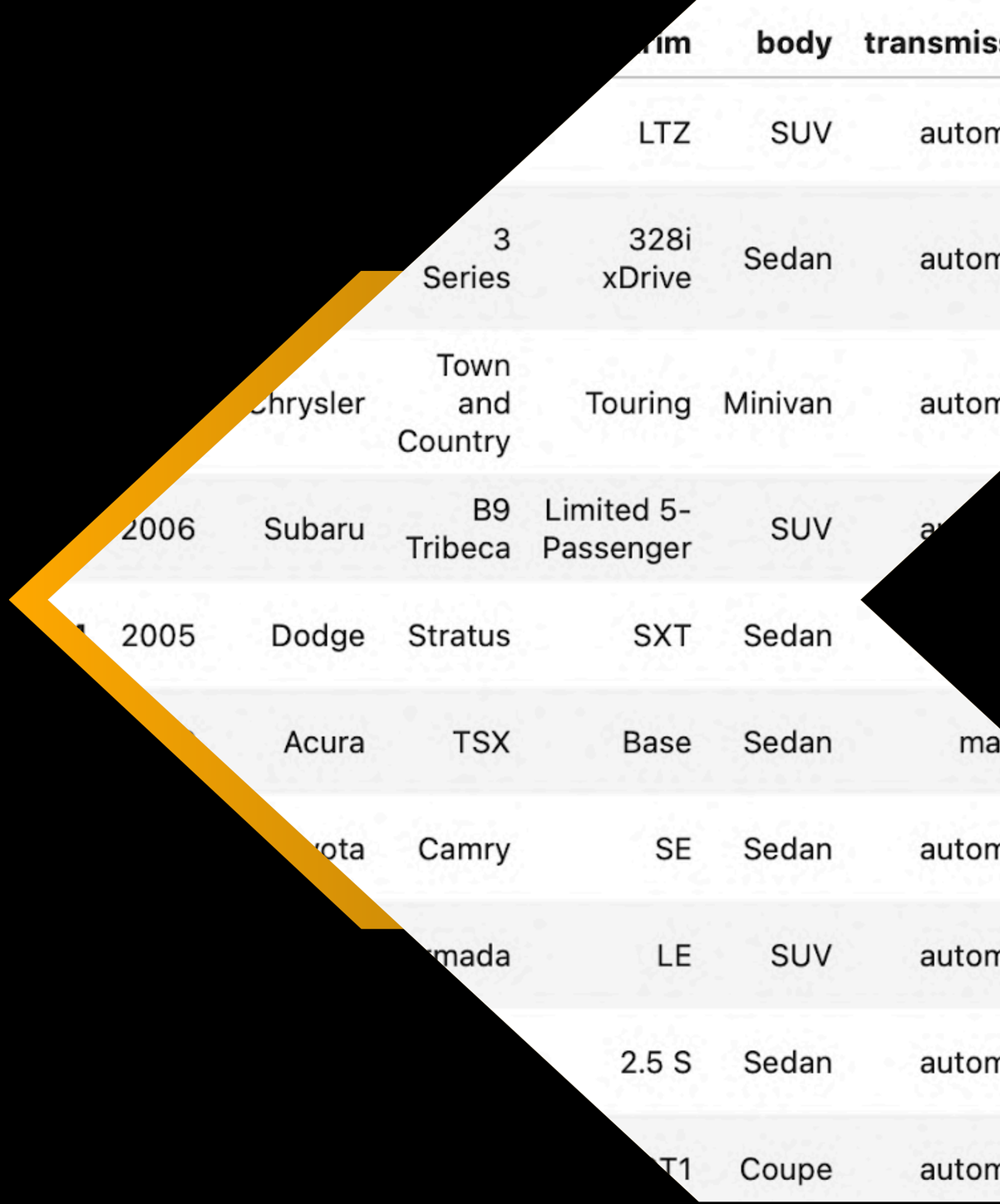


TUBES DKA

PAPARAN PROCESSING DATA

- Menghapus data duplikat
- Mengecek nilai kosong

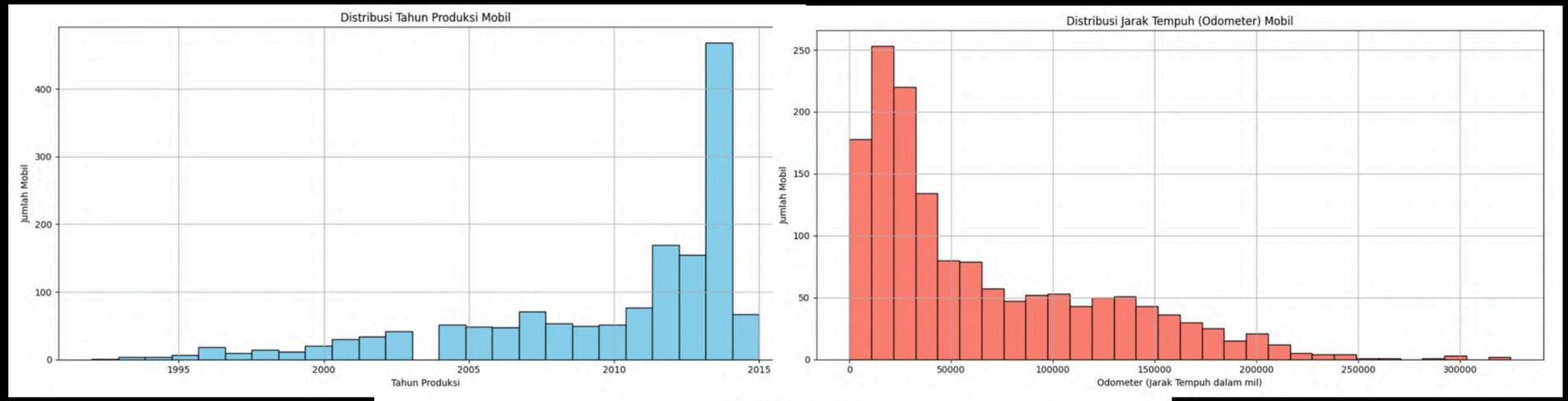
```
df.isna().sum()
df = df.drop_duplicates()
```

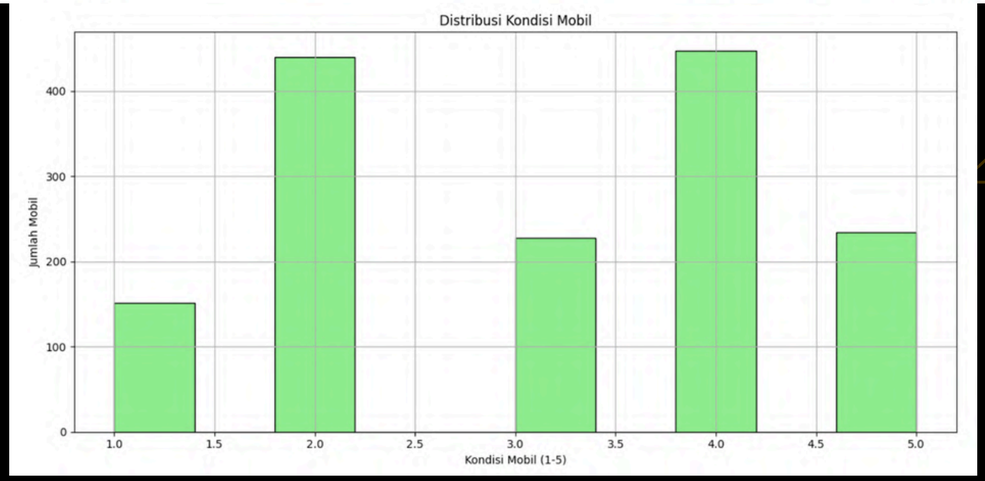


			im	body	transmis
			LTZ	SUV	autom
		3 Series	328i xDrive	Sedan	autom
	hrysler	Town and Country	Touring	Minivan	autom
2006	Subaru	B9 Tribeca	Limited 5-Passenger	SUV	a
2005	Dodge	Stratus	SXT	Sedan	
	Acura	TSX	Base	Sedan	ma
	yota	Camry	SE	Sedan	autom
	mada		LE	SUV	autom
			2.5 S	Sedan	autom
			T1	Coupe	autom

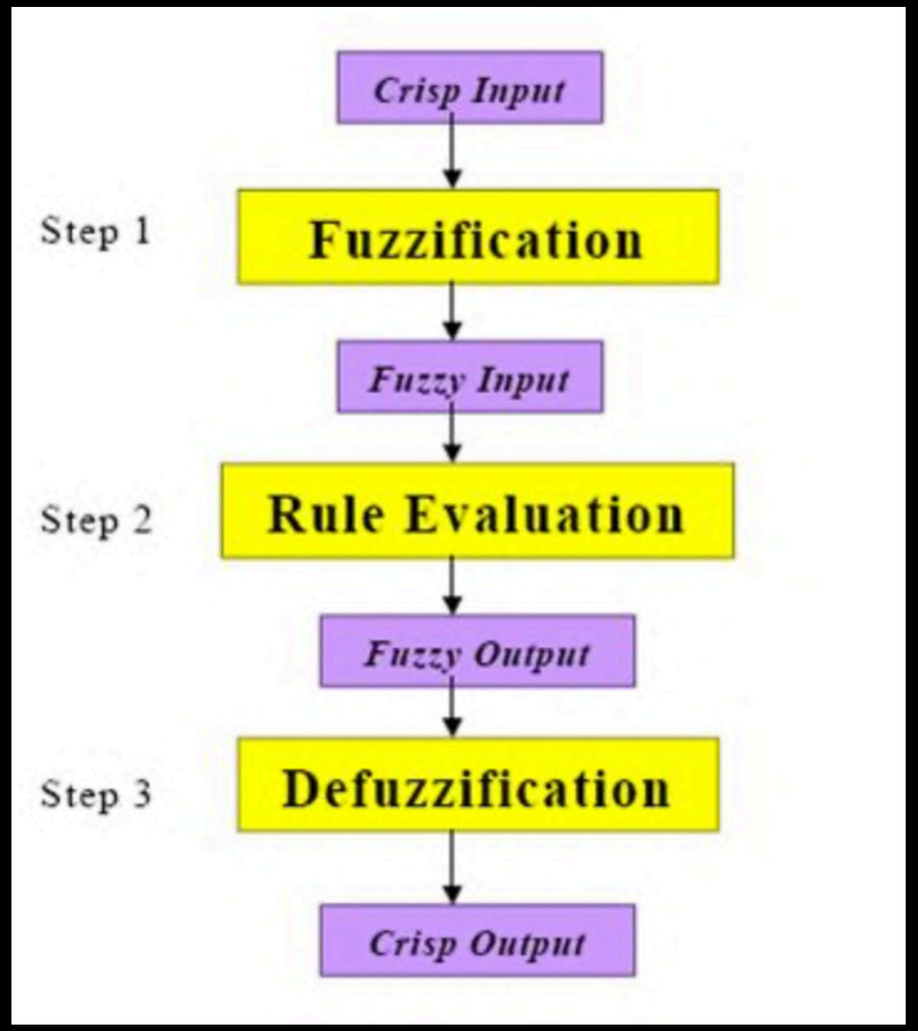
TUBES DKA

EXPLORATORY DATA ANALYSIS (EDA)





II. PROSES FUZZY



A. INPUT

Variabel Linguistik

Tahun Produksi: Tua, Sedang, Baru
Kondisi Mobil: Buruk, Sedang, Baik
Odometer: Tinggi, Sedang, Rendah

FUZZIFIKASI

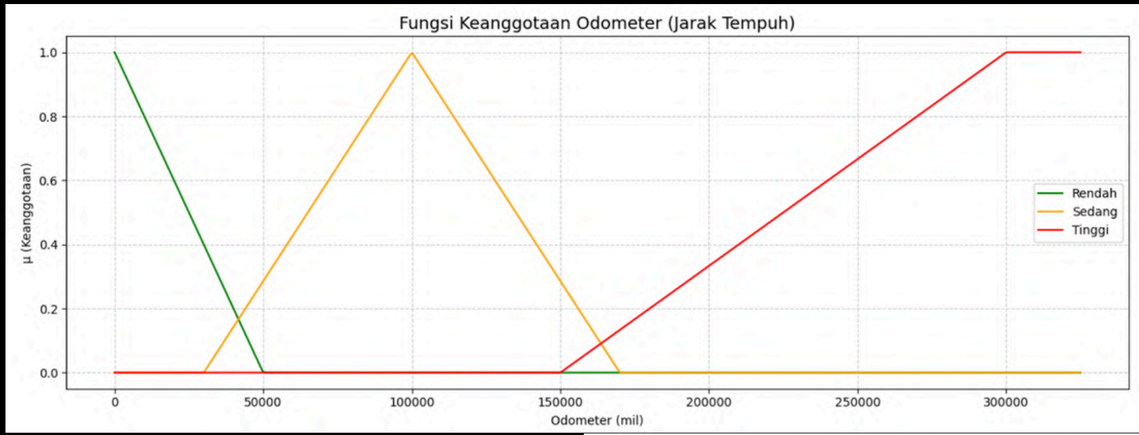
Proses ini mengubah input yang jelas (crisp input) menjadi nilai fuzzy dengan menggunakan fungsi keanggotaan.

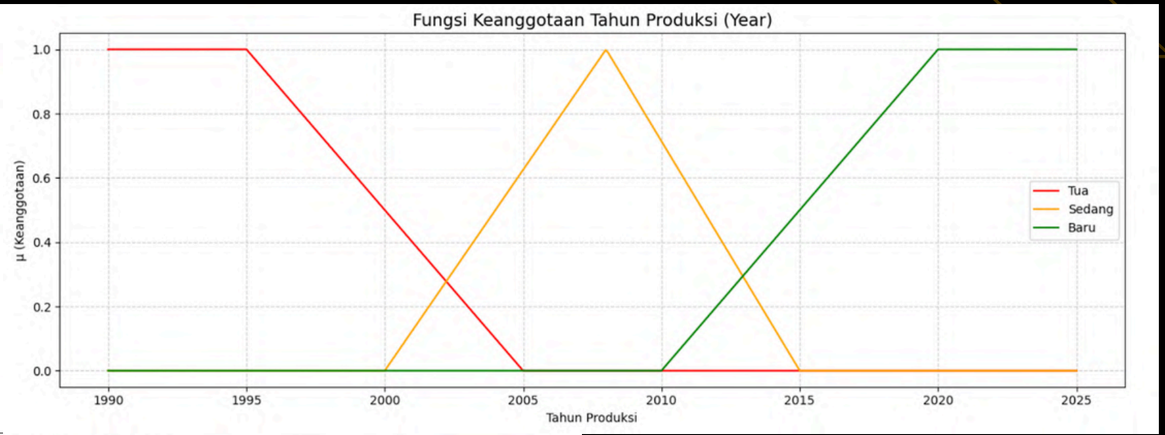
```python
#year
def year_tua(year):
    if year <= 1995:
        return 1
    elif 1995 < year <= 2005:
        return (2005 - year) / (2005 - 1995)
    else:
        return 0

def year_sedang(year):
    if year <= 2000 or year >= 2015:
        return 0
    elif 2000 < year <= 2008:
        return (year - 2000) / (2008 - 2000)
    elif 2008 < year < 2015:
        return (2015 - year) / (2015 - 2008)
    else:
        return 0

def year_baru(year):
    if year <= 2010:
        return 0
    elif 2010 < year <= 2020:
        return (year - 2010) / (2020 - 2010)
    else:
        return 1
```

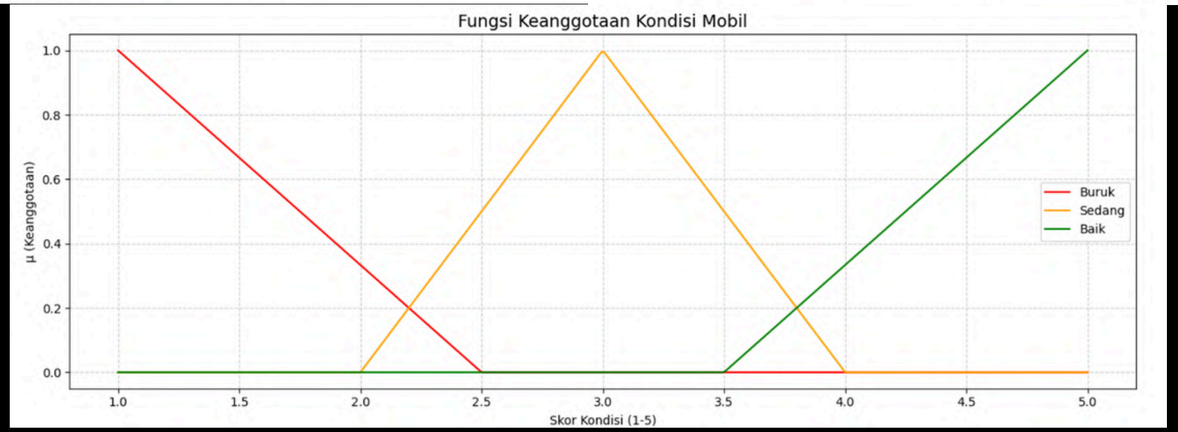
```python
def condition_buruk(cond):
    if cond <= 1.0:
        return 1
    elif 1.0 < cond <= 2.5:
        return (2.5 - cond) / (2.5 - 1.0)
    else:
        return 0

#condition
def condition_sedang(cond):
    if cond <= 2 or cond >= 4:
        return 0
    elif 2 < cond <= 3:
        return (cond - 2) / (3 - 2)
    elif 3 < cond < 4:
        return (4 - cond) / (4 - 3)
    else:
        return 0

def condition_baik(cond):
    if cond <= 3.5:
        return 0
    elif 3.5 < cond <= 5.0:
        return (cond - 3.5) / (5.0 - 3.5)
    else:
        return 1
```

```python
#odometer
def odometer_rendah(km):
    if km <= 0:
        return 1
    elif 0 < km <= 50000:
        return (50000 - km) / 50000
    else:
        return 0

def odometer_sedang(km):
    if km <= 30000 or km >= 170000:
        return 0
    elif 30000 < km <= 100000:
        return (km - 30000) / (100000 - 30000)
    elif 100000 < km < 170000:
        return (170000 - km) / (170000 - 100000)
    else:
        return 0

def odometer_tinggi(km):
    if km <= 150000:
        return 0
    elif 150000 < km <= 300000:
        return (km - 150000) / (300000 - 150000)
    else:
        return 1
```

Visualisasi Fungsi Keanggotaan Input :







B. RULE BASE

Aturan fuzzy dibangun berdasarkan kombinasi input dan akanmenghasilkan output:

- Layak
- Cukup Layak
- Tidak Layak

Mendesain rule fuzzy logic :

```python
import pandas as pd
import matplotlib.pyplot as plt

# Fungsi fuzzy prediksi kelayakan berdasarkan rule base
def prediksi_kelayakan(year, condition, odo):
    # HIGH PRIORITY RULES
    if year == "Tua" and condition == "Buruk" and odo == "Tinggi":
        return "Tidak Layak"
    if year == "Sedang" and condition == "Buruk" and odo == "Tinggi":
        return "Tidak Layak"

    if year == "Tua" and condition == "Sedang" and odo == "Sedang":
        return "Cukup Layak"
    if year == "Tua" and condition == "Baik" and odo == "Rendah":
        return "Cukup Layak"
    if year == "Sedang" and condition == "Sedang" and odo == "Sedang":
        return "Cukup Layak"
    if year == "Baru" and condition == "Baik" and odo == "Tinggi":
        return "Cukup Layak"
    if year == "Sedang" and condition == "Baik" and odo == "Tinggi":
        return "Cukup Layak"

    if year == "Baru" and condition == "Baik" and odo == "Rendah":
        return "Layak"
    if year == "Baru" and condition == "Sedang" and odo == "Sedang":
        return "Layak"
    if year == "Sedang" and condition == "Baik" and odo == "Rendah":
        return "Layak"

    return "Cukup Layak"

# Daftar kategori
year_list = ["Tua", "Sedang", "Baru"]
condition_list = ["Buruk", "Sedang", "Baik"]
odometer_list = ["Tinggi", "Sedang", "Rendah"]
```

```python
rules = []
for year in year_list:
    for cond in condition_list:
        for odo in odometer_list:
            rules.append({
                "Year": year,
                "Condition": cond,
                "Odometer": odo,
                "Kelayakan": prediksi_kelayakan(year, cond, odo)
            })

df_rules = pd.DataFrame(rules)

fig, ax = plt.subplots(figsize=(12, 8))
ax.axis('off')

cell_text = df_rules.values.tolist()
col_labels = df_rules.columns.tolist()

tbl = ax.table(
    cellText=cell_text,
    colLabels=col_labels,
    loc='center',
    cellLoc='center',
    colWidths=[0.2] * len(col_labels)
)

tbl.auto_set_font_size(False)
tbl.set_fontsize(10)
tbl.scale(1.2, 1.2)

plt.title('Tabel Aturan Fuzzy Kelayakan Mobil', fontsize=14, weight='bold')
plt.tight_layout()
plt.show()
```

Tabel Aturan Fuzzy Kelayakan Mobil

Tabel Aturan Fuzzy Kelayakan Mobil

Year	Condition	Odometer	Kelayakan
Tua	Buruk	Tinggi	Tidak Layak
Tua	Buruk	Sedang	Cukup Layak
Tua	Buruk	Rendah	Cukup Layak
Tua	Sedang	Tinggi	Cukup Layak
Tua	Sedang	Sedang	Cukup Layak
Tua	Sedang	Rendah	Cukup Layak
Tua	Baik	Tinggi	Cukup Layak
Tua	Baik	Sedang	Cukup Layak
Tua	Baik	Rendah	Cukup Layak
Sedang	Buruk	Tinggi	Tidak Layak
Sedang	Buruk	Sedang	Cukup Layak
Sedang	Buruk	Rendah	Cukup Layak
Sedang	Sedang	Tinggi	Cukup Layak
Sedang	Sedang	Sedang	Cukup Layak
Sedang	Sedang	Rendah	Cukup Layak
Sedang	Baik	Tinggi	Cukup Layak
Sedang	Baik	Sedang	Cukup Layak
Sedang	Baik	Rendah	Layak
Baru	Buruk	Tinggi	Cukup Layak
Baru	Buruk	Sedang	Cukup Layak
Baru	Buruk	Rendah	Cukup Layak
Baru	Sedang	Tinggi	Cukup Layak
Baru	Sedang	Sedang	Layak
Baru	Sedang	Rendah	Cukup Layak
Baru	Baik	Tinggi	Cukup Layak
Baru	Baik	Sedang	Cukup Layak
Baru	Baik	Rendah	Layak

C. MENDEFINISIKAN OUTPUT

Mendefinisikan Output

```python
# Output: Layak
def output_layak(x):
    if x <= 60:
        return 0
    elif 60 < x <= 80:
        return (x - 60) / (80 - 60)
    elif 80 < x <= 100:
        return 1
    else:
        return 0
```

```python
# Output: Cukup Layak
def output_cukup_layak(x):
    if x <= 30 or x >= 70:
        return 0
    elif 30 < x <= 50:
        return (x - 30) / (50 - 30)
    elif 50 < x < 70:
        return (70 - x) / (70 - 50)
    else:
        return 0
```

```python
# Output: Tidak Layak
def output_tidak_layak(x):
    if x <= 20:
        return 1
    elif 20 < x <= 40:
        return (40 - x) / (40 - 20)
    else:
        return 0
```

D. DEFUZZIFIKASI

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

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

Melakukan Defuzzifikasi

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

    for xi in x:
        μ = max(
            min(hasil["Tidak Layak"], output_tidak_layak(xi)),
            min(hasil["Cukup Layak"], output_cukup_layak(xi)),
            min(hasil["Layak"], output_layak(xi))
        )
        numerator += xi * μ
        denominator += μ

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

```python
def interpretasi_kelayakan(skor):
    μ_tidak = output_tidak_layak(skor)
    μ_cukup = output_cukup_layak(skor)
    μ_layak = output_layak(skor)

    max_μ = max(μ_tidak, μ_cukup, μ_layak)
    if max_μ == μ_tidak:
        return "Tidak Layak"
    elif max_μ == μ_cukup:
        return "Cukup Layak"
    else:
        return "Layak"
```

E. HASIL

Hasil dari Fuzzy Mamdani menampilkan dataset ditambah kolom nilai model mamdani dan kategori kelayakan yang dihasilkan.

Hasil Dari Fuzzy

	year	condition	odometer	Model Mamdani	Kategori Kelayakan Mamdani
0	2011	3.0	64680.0	55.962515	Cukup Layak
1	2011	4.0	14505.0	81.666293	Layak
2	2014	5.0	20738.0	81.976123	Layak
3	2006	3.0	133320.0	50.000000	Cukup Layak
4	2005	2.0	137046.0	50.000000	Cukup Layak
5	2006	2.0	125744.0	50.000000	Cukup Layak
6	2011	2.0	45861.0	50.000000	Cukup Layak
7	2004	2.0	125211.0	50.000000	Cukup Layak
8	2012	2.0	59224.0	50.000000	Cukup Layak
9	2003	1.0	1.0	50.000000	Cukup Layak
10	2012	5.0	2835.0	82.107308	Layak
11	2014	5.0	7723.0	81.976123	Layak
12	2013	3.0	42801.0	65.955832	Layak
13	2014	4.0	23882.0	81.666293	Layak
14	2015	3.0	9065.0	50.000000	Cukup Layak

IV. HASIL DAN ANALISIS KINERJA FUZZY

Membuat table kelayakan berdasarkan selling price

Sebagai pengganti data aktual, yang digunakan sebagai data pembanding dengan hasil metode fuzzy

```
def buat_label_kelayakan(row):
    if row['sellingprice'] > 10000:
        return 'layak'
    elif row['sellingprice'] > 6000:
        return 'cukup layak'
    else:
        return 'tidak layak'

df['kelayakan_aktual'] = df.apply(buat_label_kelayakan, axis=1)
```

Hasil Akurasi

1. SMAPE (Symmetric Mean Absolute Percentage Error)

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

```python
kelayakan_ke_angka = {
    'tidak layak': 20,
    'cukup layak': 50,
    'layak': 80
}

df['kelayakan_aktual'] = df['kelayakan_aktual'].str.lower().map(kelayakan_ke_angka)
df['pred_mamdani_num'] = df['Kategori Kelayakan Mamdani'].str.lower().map(kelayakan_ke_angka)
df['pred_sugeno_num'] = df['Kategori Kelayakan Sugeno'].str.lower().map(kelayakan_ke_angka)

# Fungsi 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

df_valid_sugeno = df.dropna(subset=['kelayakan_aktual', 'pred_sugeno_num'])
nilai_smape_sugeno = smape(df_valid_sugeno['kelayakan_aktual'], df_valid_sugeno['pred_sugeno_num'])

df_valid_mamdani = df.dropna(subset=['kelayakan_aktual', 'pred_mamdani_num'])
nilai_smape_mamdani = smape(df_valid_mamdani['kelayakan_aktual'], df_valid_mamdani['pred_mamdani_num'])

print(f"SMAPE Model Sugeno: {nilai_smape_sugeno:.2f}%")
print(f"SMAPE Model Mamdani: {nilai_smape_mamdani:.2f}%")
```

```
SMAPE Model Mamdani: 30.48%
```

Hasil Akurasi

2. MAE (Mean Absolute Eror)

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

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

print(f"Mean Absolute Error Mamdani: {mae(df['kelayakan_aktual'], df['pred_mamdani_num']):.2f}")
print(f"Mean Absolute Error Sugeno: {mae(df['kelayakan_aktual'], df['pred_sugeno_num']):.2f}")
```

```
Mean Absolute Error Mamdani: 14.12
```

Evaluasi Akurasi

Setelah proses inferensi fuzzy dan defuzzifikasi dilakukan menggunakan metode Mamdani, model dievaluasi dengan dua metrik umum:

- SMAPE (Symmetric Mean Absolute Percentage Error): 30.48%
- MAE (Mean Absolute Error): 14.12

Berdasarkan hasil evaluasi tersebut, model Fuzzy Mamdani menunjukkan performa yang cukup stabil, meskipun tidak seakurat metode machine learning dalam hal prediksi numerik. Keunggulan utama metode ini terletak pada kemampuannya dalam menangani data linguistik, serta memberikan hasil yang mudah diinterpretasikan.

Fuzzy Mamdani bekerja berdasarkan rule base yang bersifat eksplisit, sehingga cocok untuk sistem yang mengedepankan transparansi dalam pengambilan keputusan, seperti dalam skenario sistem pendukung keputusan (Decision Support System) untuk menilai kelayakan mobil secara umum.

V. MACHINE LEARNING (XGBOOST)

Langkah – Langkah

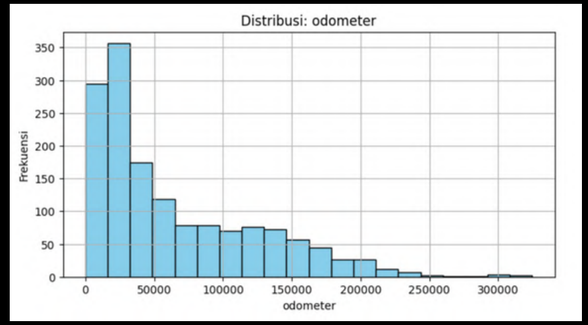
1. Import Library

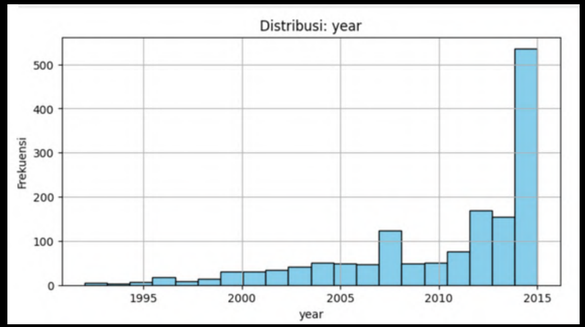
```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import xgboost as xgb
import matplotlib.pyplot as plt
import pandas as pd
from xgboost import XGBClassifier
```

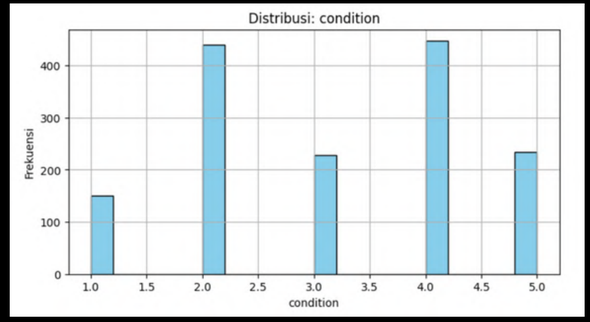
2. Visualisasi

```
num_cols = ['year', 'condition', 'odometer']

for col in num_cols:
    plt.figure(figsize=(8, 4))
    plt.hist(df[col], bins=20, color='skyblue', edgecolor='black')
    plt.title(f'Distribusi: {col}')
    plt.xlabel(col)
    plt.ylabel('Frekuensi')
    plt.grid(True)
    plt.show()
```

Hasil Visualisasi







Langkah - Langkah

3. Label Encoding

```
df = df.dropna(subset=['kelayakan_aktual'])

label_encoder = LabelEncoder()
df['kelayakan_encoded'] = label_encoder.fit_transform(df['kelayakan_aktual'].astype(str).str.lower())
```

4. Split Data

```
x = df[['year', 'condition', 'odometer']]
y = df[['kelayakan_encoded']]

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

5. Modelling

```
model1 = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')
model1.fit(x_train, y_train)

predict = model1.predict(x_test)
```

Langkah - Langkah

7. Akurasi

```
print(f"Akurasi menggunakan XGBoost: {accuracy_score(y_test, predict) * 100:.2f}%")
```

```
Akurasi menggunakan XGBoost: 79.00%
```

VI. PERBANDINGAN FUZZY LOGIC DENGAN MACHINE LEARNING (XGBOOST)

Hasil Evaluasi Model XGBoost

Sebagai pembanding, model XGBoost juga diterapkan pada data yang sama untuk memprediksi kelayakan kendaraan. Hasil yang diperoleh:

- Akurasi Model XGBoost: 79.00%
- SMAPE Fuzzy Mamdani: 30.48%
- MAE Fuzzy Mamdani: 14.12

Aspek	Fuzzy Mamdani	XGBoost
Interpretasi	Sangat tinggi (berbasis aturan)	Rendah (black-box model)
Akurasi Prediksi	Sedang	Tinggi
Adaptif terhadap Data	Tidak	Ya (belajar dari data historis)
Kecepatan Proses	Lambat (banyak aturan)	Cepat dan efisien
Kompleksitas	Sederhana, transparan	Kompleks, sulit diinterpretasi

Kesimpulan Perbandingan

- Fuzzy Mamdani unggul dalam aspek interpretabilitas dan cocok untuk kasus di mana transparansi dan logika linguistik dibutuhkan.
- XGBoost lebih unggul secara statistik, karena mampu mengenali pola dari data besar dan kompleks secara otomatis.
- Jika tujuan utama adalah prediksi akurat, maka XGBoost adalah pilihan tepat. Namun, jika diperlukan kejelasan dalam penjelasan keputusan, Mamdani lebih sesuai.

VII. KESIMPULAN

Kesimpulan Penelitian

Fuzzy logic terbukti mampu menyelesaikan permasalahan klasifikasi berbasis rule dengan cukup baik, terutama untuk interpretasi yang bersifat linguistik dan fleksibel. Namun, dalam konteks dataset numerik berskala besar seperti penilaian kelayakan mobil, algoritma machine learning seperti XGBoost memberikan akurasi yang lebih tinggi dan efisiensi prediksi yang lebih baik.

Model fuzzy tetap unggul dalam aspek interpretabilitas, sedangkan XGBoost unggul dalam performa prediktif. Pemilihan metode sangat tergantung pada kebutuhan sistem—apakah lebih mengutamakan transparansi atau akurasi.

Kesimpulan Penelitian

Beberapa kesimpulan yang dapat diambil:

1. Fuzzy Mamdani mampu menghasilkan sistem penilaian yang transparan dan mudah dipahami, dengan hasil linguistik yang sesuai untuk pengambilan keputusan manusia.
2. Meskipun akurasi prediksi Mamdani lebih rendah dibandingkan XGBoost, kelebihan utamanya adalah pada interpretasi dan fleksibilitas terhadap logika linguistik.
3. XGBoost menghasilkan akurasi lebih tinggi (79.00%), namun kurang interpretatif dan bersifat black-box.
4. Sistem Fuzzy Mamdani cocok diterapkan untuk sistem pendukung keputusan manual/semi-otomatis, sedangkan XGBoost lebih cocok untuk sistem otomatisasi berskala besar.

VIII. REFERENSI

Referensi

- Ross, T. J. (2010). Fuzzy Logic with Engineering Applications (3rd ed.). Wiley.
- Kaggle Dataset: https://www.kaggle.com/datasets/syedanwarafridi/vehicle-sales-data/data
- Scikit-learn Documentation: https://scikit-learn.org
- XGBoost Documentation: https://xgboost.readthedocs.io
- Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

IX. SOURCE CODE

SOURCE CODE :
HTTPS://GITHUB.COM/ARVIANN/TUBESDKA/BLOB/MAIN/TUBES_DKA_KELAYAKAN_MOBIL.IPYNB

TUBES DKA

THANK YOU

FOR YOUR ATTENTION