Perancangan Sistem Rekomendasi Buku Menggunakan Logika Fuzzy Berdasarkan Dataset *Goodbooks*

Velisia Nihan Rahmawati (103012300203)
Nabila Putri Aulia (103012330531)

Latar Belakang dan Permasalahan

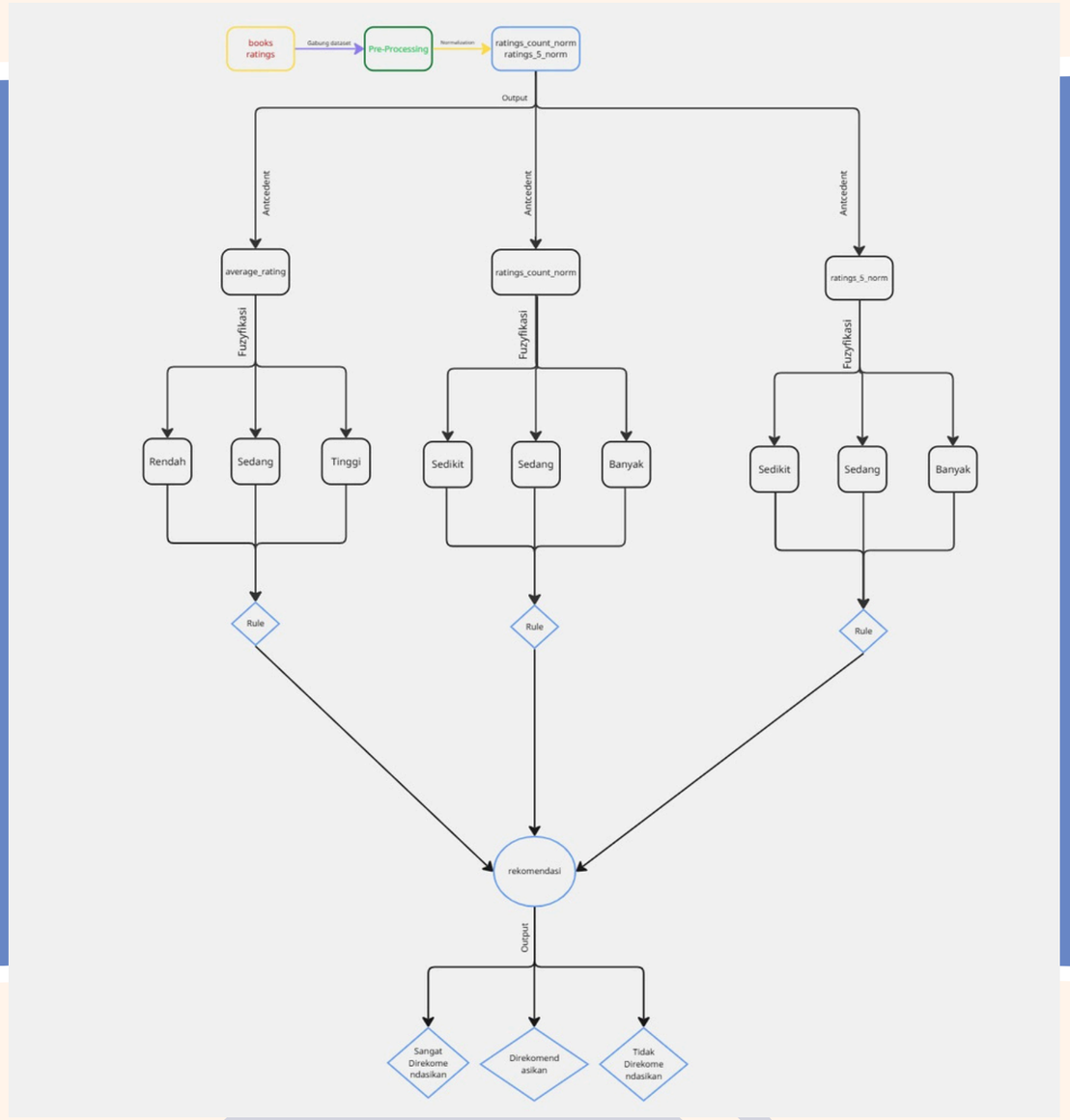
Di era digital, banyaknya pilihan buku di platform seperti Goodreads membuat proses pemilihan bacaan menjadi kompleks. Sistem rekomendasi hadir sebagai solusi, namun metode konvensional seperti collaborative filtering sering gagal menangani ketidakpastian dan subjektivitas preferensi pengguna.

Logika fuzzy, yang diperkenalkan oleh Zadeh (1965), memungkinkan pengambilan keputusan berbasis istilah linguistik seperti "tinggi", "sedang", dan "rendah", sehingga lebih menyerupai cara berpikir manusia. Penerapannya dalam sistem rekomendasi buku diharapkan dapat meningkatkan akurasi dan interpretabilitas, terutama dalam situasi dengan data yang ambigu atau tidak terstruktur.

Permasalahan utama yang diangkat adalah bagaimana merekomendasikan buku secara akurat berdasarkan rating, jumlah pembaca, dan jumlah rating bagus. Fuzzy logic dipilih karena mampu:

- Mengintegrasikan berbagai kriteria ke dalam satu sistem penilaian.
- Menangani ambiguitas dalam interpretasi data numerik.
- Mengubah data numerik menjadi istilah yang lebih mudah dipahami oleh pengguna.

Block Diagram



Normalisasi Rating ke Skala 0–100

```python
# Normalisasi ratings_count dan ratings_5 ke skala 0–100

max_rating = books["ratings_count"].max()
books["ratings_count_norm"] = (books["ratings_count"] / max_rating * 100).round(0)
books["ratings_5_norm"] = (books["ratings_5"] / max_rating * 100).round(0)
```

```python
def fuzzify_rating(r):
    if r >= 4.2:
        return "tinggi"
    elif r >= 3.5:
        return "sedang"
    else:
        return "rendah"

def fuzzify_count(c):
    if c >= 66:
        return "banyak"
    elif c >= 33:
        return "sedang"
    else:
        return "sedikit"
```

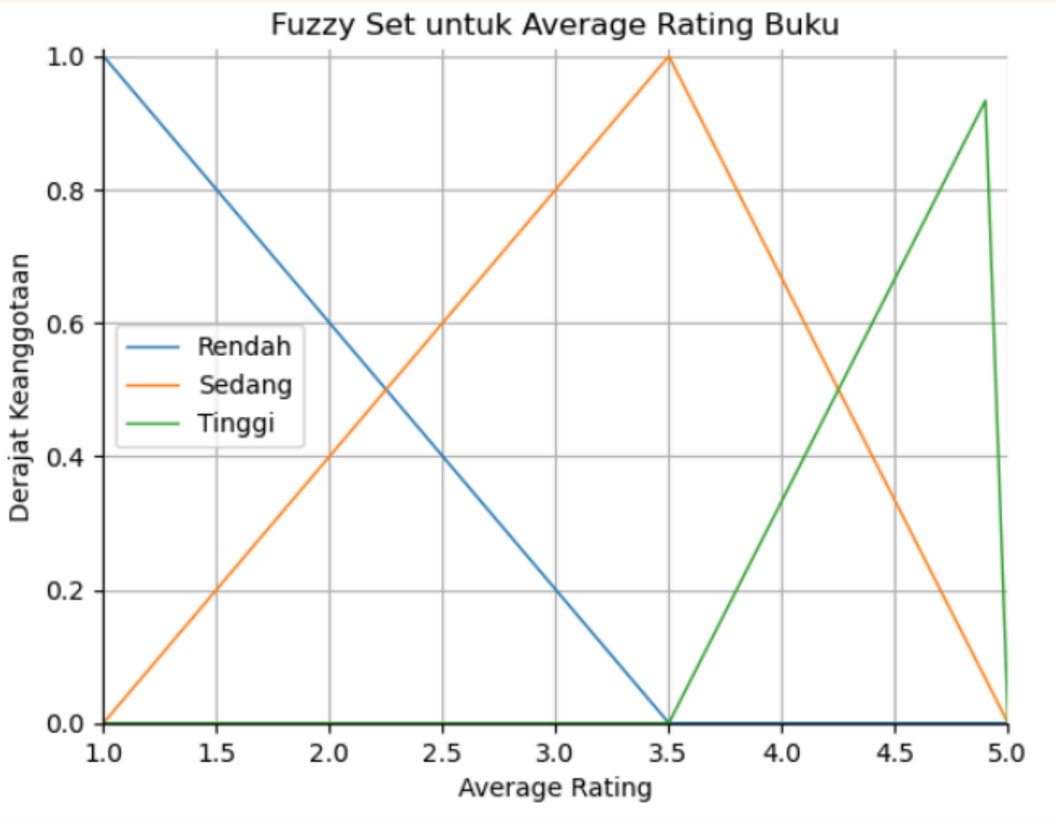
```python
def fuzzy_recommendation(row):
    rating = fuzzify_rating(row["average_rating"])
    count = fuzzify_count(row["ratings_count_norm"])
    rating5 = fuzzify_count(row["ratings_5_norm"])

    if rating == "tinggi" and count == "banyak" or rating5 == "banyak":
        return "Sangat Direkomendasikan"
    elif rating == "tinggi" and count == "sedang":
        return "Direkomendasikan"
    elif rating == "sedang" and rating5 == "banyak":
        return "Mungkin Direkomendasikan"
    else:
        return "Tidak Direkomendasikan"


# Tambahkan kolom rekomendasi
books["Rekomendasi"] = books.apply(fuzzy_recommendation, axis=1)

# Tampilkan hasil dengan kolom normalisasi
books[["title", "average_rating", "ratings_count_norm", "ratings_5_norm", "Rekomendasi"]].head(10)
```

Visual Tabel

	title	average_rating	ratings_count_norm	ratings_5_norm	Rekomendasi
0	The Hunger Games (The Hunger Games, #1)	4.34	100.0	57.0	Sangat Direkomendasikan
1	Harry Potter and the Sorcerer's Stone (Harry Potter, #1)	4.44	96.0	63.0	Sangat Direkomendasikan
2	Twilight (Twilight, #1)	3.57	81.0	28.0	Tidak Direkomendasikan
3	To Kill a Mockingbird	4.25	67.0	36.0	Sangat Direkomendasikan
4	The Great Gatsby	3.89	56.0	20.0	Tidak Direkomendasikan
5	The Fault in Our Stars	4.26	49.0	27.0	Direkomendasikan
6	The Hobbit	4.25	43.0	23.0	Direkomendasikan
7	The Catcher in the Rye	3.79	43.0	15.0	Tidak Direkomendasikan
8	Angels & Demons (Robert Langdon, #1)	3.85	42.0	14.0	Tidak Direkomendasikan
9	Pride and Prejudice	4.24	43.0	24.0	Direkomendasikan

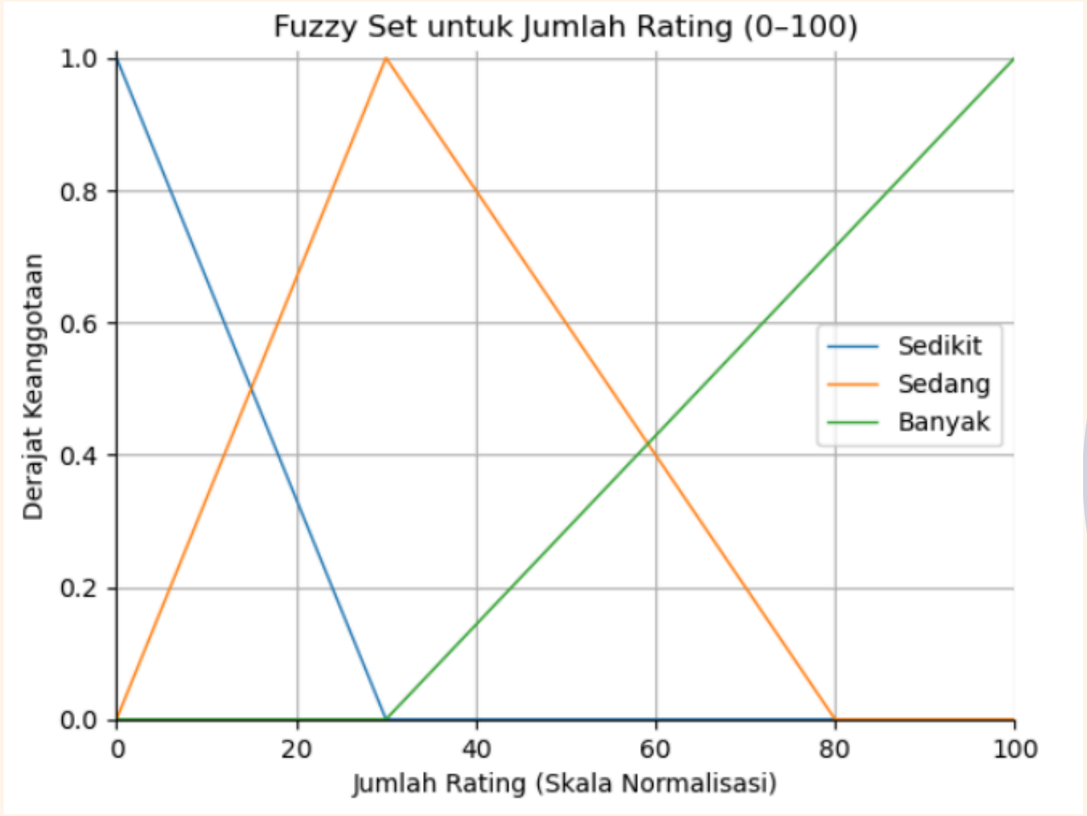
FUZZY set Average Rating

```python
#AVERAGE RATING

# Definisikan universe untuk rating
rating = ctrl.Antecedent(np.arange(1.0, 5.1, 0.1), 'rating')

# Fungsi keanggotaan fuzzy (segitiga)
rating['Rendah'] = fuzz.trimf(rating.universe, [1.0, 1.0, 3.5])
rating['Sedang'] = fuzz.trimf(rating.universe, [1.0, 3.5, 5.0])
rating['Tinggi'] = fuzz.trimf(rating.universe, [3.5, 5.0, 5.0])

# Visualisasi fuzzy set
rating.view()
plt.title('Fuzzy Set untuk Average Rating Buku')
plt.xlabel('Average Rating')
plt.ylabel('Derajat Keanggotaan')
plt.grid(True)
plt.legend(['Rendah', 'Sedang', 'Tinggi'])
plt.show()

# Buat DataFrame untuk tabel fuzzy
data = {
    'Label': ['Rendah', 'Sedang', 'Tinggi'],
    'Range': ['[1.0, 1.0, 3.5]', '[1.0, 3.5, 5.0]', '[3.5, 5.0, 5.0]'],
    'Arti Fuzzy': [
        'Mulai dari 1.0 (penuh), turun ke 0 di 3.5',
        'Mulai naik dari 1.0, puncak di 3.5, turun ke 0 di 5.0',
        'Mulai dari 3.5, puncak di 5.0, tetap tinggi'
    ]
}

df = pd.DataFrame(data)

# Tampilkan tabel rapi
print("Tabel Interpretasi Fungsi Keanggotaan Fuzzy untuk Average Rating:")
display(df)
```



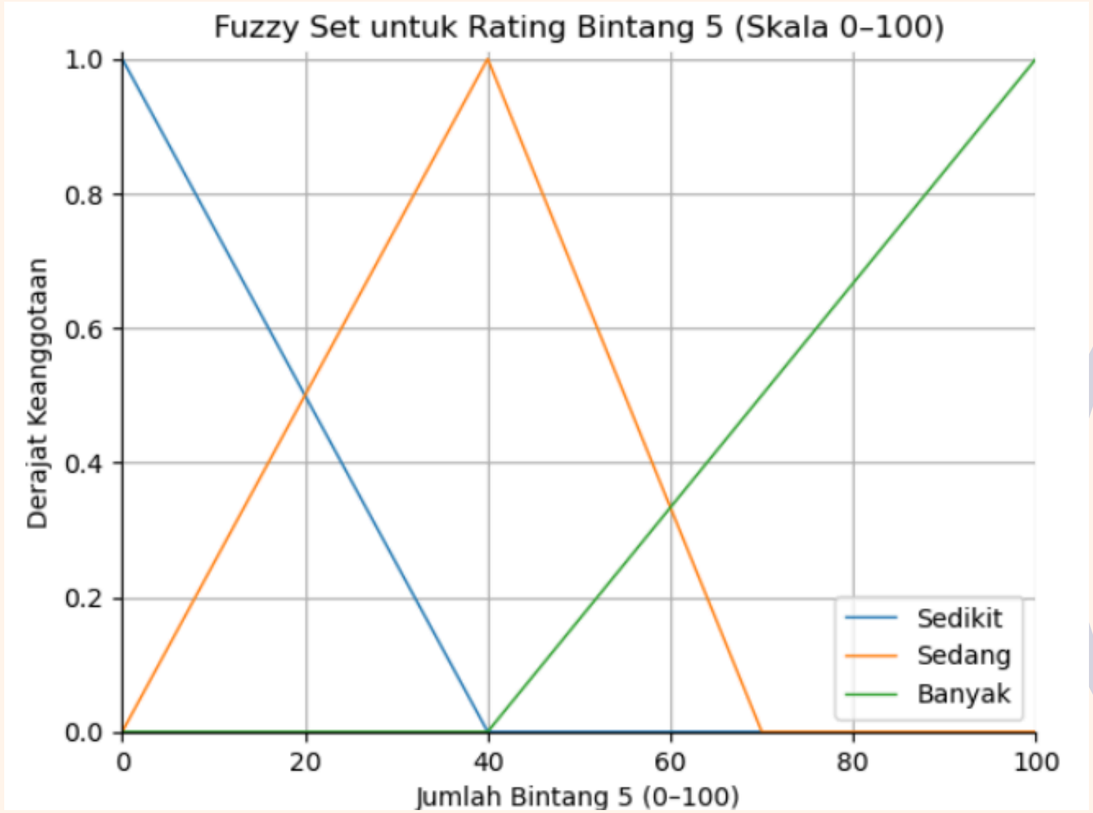
FUZZY set Rating Count

```python
# RATING COUNT NORMALIZED (0–100)
ratings_count_norm = ctrl.Antecedent(np.arange(0, 101, 1), 'ratings_count_norm')

# Fungsi keanggotaan fuzzy
ratings_count_norm['Sedikit'] = fuzz.trimf(ratings_count_norm.universe, [0, 0, 30])
ratings_count_norm['Sedang'] = fuzz.trimf(ratings_count_norm.universe, [0, 30, 80])
ratings_count_norm['Banyak'] = fuzz.trimf(ratings_count_norm.universe, [30, 100, 100])

# Visualisasi fuzzy set
ratings_count_norm.view()
plt.title('Fuzzy Set untuk Jumlah Rating (0–100)')
plt.xlabel('Jumlah Rating (Skala Normalisasi)')
plt.ylabel('Derajat Keanggotaan')
plt.grid(True)
plt.legend(['Sedikit', 'Sedang', 'Banyak'])
plt.show()

# Buat DataFrame interpretasi
data_ratings_count_norm = {
    'Label': ['Sedikit', 'Sedang', 'Banyak'],
    'Range': ['[0, 0, 30]', '[0, 30, 80]', '[30, 100, 100]'],
    'Arti Fuzzy': [
        'Jumlah rating rendah, penuh keanggotaan di 0, menurun ke 0 di 30',
        'Keanggotaan mulai naik dari 0, puncak di 30, turun ke 0 di 80',
        'Keanggotaan mulai dari 30, penuh keanggotaan dari 100 ke atas'
    ]
}

df_ratings_count_norm = pd.DataFrame(data_ratings_count_norm)

# Tampilkan tabel
print("Tabel Interpretasi Fuzzy untuk Jumlah Rating (0–100):")
display(df_ratings_count_norm)
```



FUZZY set Rating 5

```python
# RATING 5 (SKALA 0-100)
ratings_5_norm = ctrl.Antecedent(np.arange(0, 101, 1), 'ratings_5_norm')

# Fungsi keanggotaan fuzzy (segitiga) skala 0-100
ratings_5_norm['Sedikit'] = fuzz.trimf(ratings_5_norm.universe, [0, 0, 40])
ratings_5_norm['Sedang'] = fuzz.trimf(ratings_5_norm.universe, [0, 40, 70])
ratings_5_norm['Banyak'] = fuzz.trimf(ratings_5_norm.universe, [40, 100, 100])

# Visualisasi fuzzy set
ratings_5_norm.view()
plt.title('Fuzzy Set untuk Rating Bintang 5 (Skala 0-100)')
plt.xlabel('Jumlah Bintang 5 (0-100)')
plt.ylabel('Derajat Keanggotaan')
plt.grid(True)
plt.legend(['Sedikit', 'Sedang', 'Banyak'])
plt.show()

# Buat tabel fuzzy interpretasi
data_ratings_5_scaled = {
    'Label': ['Sedikit', 'Sedang', 'Banyak'],
    'Range': ['[0, 0, 40]', '[0, 40, 70]', '[40, 100, 100]'],
    'Arti Fuzzy': [
        'Jumlah rating bintang 5 rendah, keanggotaan penuh di 0, turun ke 0 di 40',
        'Naik dari 0, puncak di 40, turun ke 0 di 70',
        'Mulai dari 40, penuh keanggotaan dari 100'
    ]
}

df_ratings_5_scaled = pd.DataFrame(data_ratings_5_scaled)

# Tampilkan tabel
print("Tabel Interpretasi Fungsi Keanggotaan Fuzzy untuk Rating Bintang 5 (Skala 0-100):")
display(df_ratings_5_scaled)
```



Fuzzy Rules

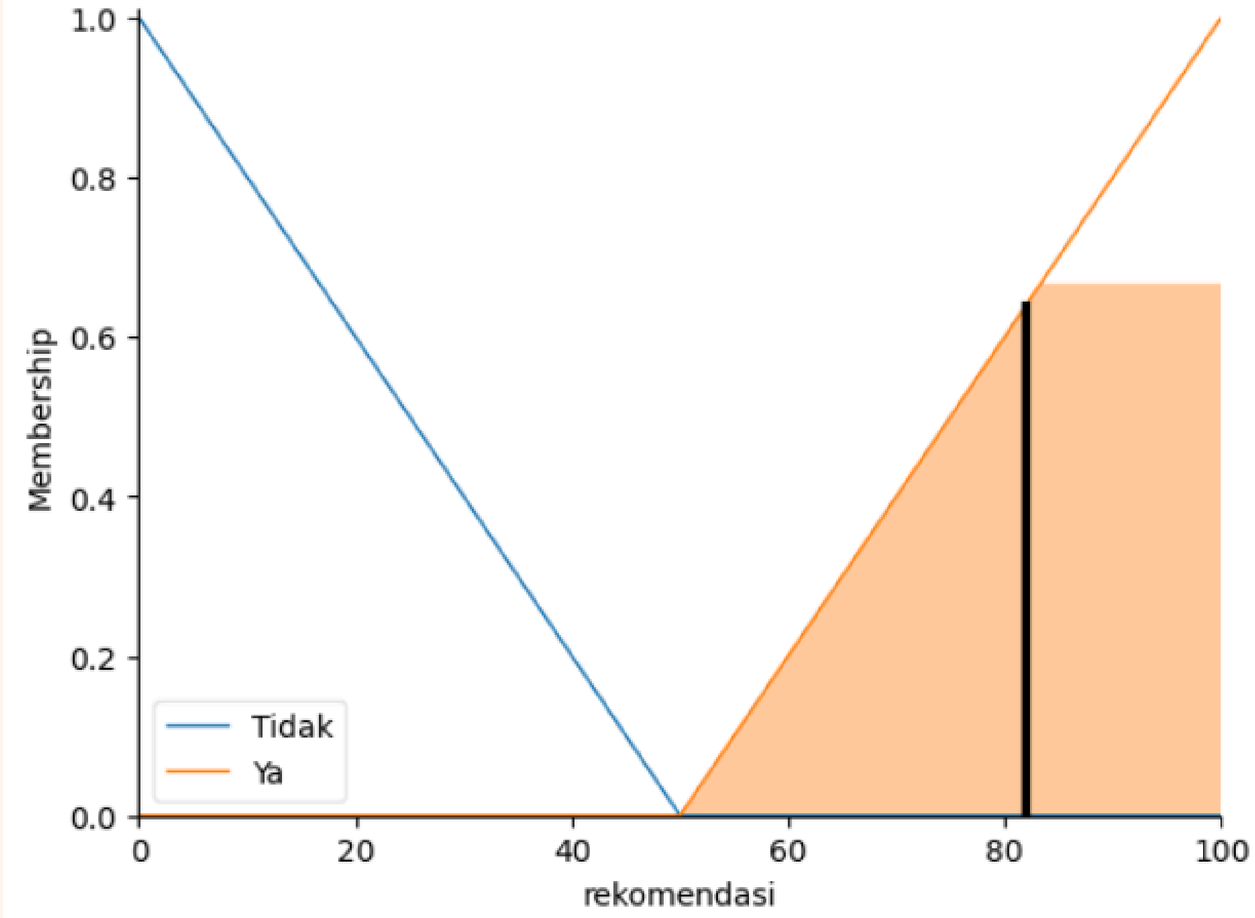
```
rekomendasi = ctrl.Consequent(np.arange(0, 101, 1), 'rekomendasi')
rekomendasi['Tidak'] = fuzz.trimf(rekomendasi.universe, [0, 0, 50])
rekomendasi['Ya'] = fuzz.trimf(rekomendasi.universe, [50, 100, 100])
```

```
rule1 = ctrl.Rule(rating['Tinggi'] & ratings_count_norm['Banyak'] & ratings_5_norm['Banyak'], rekomendasi['Ya'])
rule2 = ctrl.Rule(rating['Tinggi'] & ratings_count_norm['Sedang'] & (ratings_5_norm['Sedang'] | ratings_5_norm['Banyak']), rekomendasi['Ya'])
rule3 = ctrl.Rule(rating['Tinggi'] & ratings_count_norm['Sedikit'] & ratings_5_norm['Banyak'], rekomendasi['Ya'])
rule4 = ctrl.Rule(rating['Sedang'] & ratings_count_norm['Banyak'] & (ratings_5_norm['Sedang'] | ratings_5_norm['Banyak']), rekomendasi['Ya'])
rule5 = ctrl.Rule(rating['Sedang'] & ratings_count_norm['Sedang'] & ratings_5_norm['Banyak'], rekomendasi['Ya'])
rule6 = ctrl.Rule(rating['Sedang'] & ratings_count_norm['Sedikit'], rekomendasi['Tidak'])
rule7 = ctrl.Rule(rating['Rendah'] | ratings_5_norm['Sedikit'], rekomendasi['Tidak'])
rule8 = ctrl.Rule(rating['Rendah'] & ratings_count_norm['Banyak'], rekomendasi['Tidak'])
```

Aturan (rules) dalam sistem logika fuzzy untuk menentukan apakah sebuah buku direkomendasikan atau tidak.
Variabel output rekomendasi ada dua kategori fuzzy: 'Tidak' dan 'Ya', masing-masing dengan fungsi keanggotaan.
Delapan aturan fuzzy (rule1 sampai rule8) dibuat berdasarkan kombinasi tiga input: rating, ratings_count_norm, dan ratings_5_norm

Grafik Perhitungan Fuzzy, sangat direkomendasikan

```
Hasil fuzzy: 81.94
Rekomendasi Buku: Sangat Direkomendasikan
```



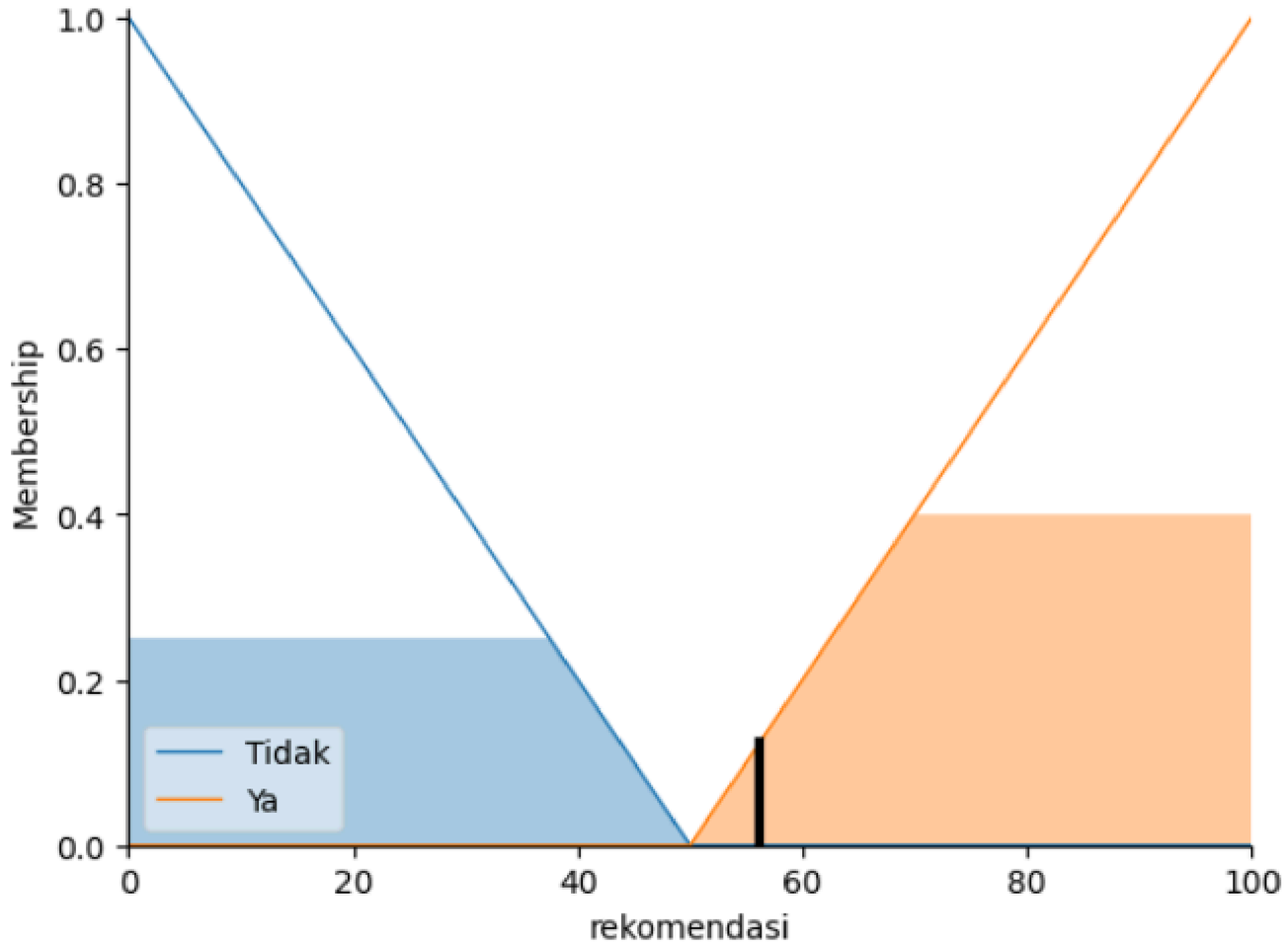

```python
# === 4. Bangun dan Simulasi Sistem Fuzzy ===
rekomendasi_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8])
rekomendasi_sim = ctrl.ControlSystemSimulation(rekomendasi_ctrl)
```

```python
# === 5. Masukkan Nilai Contoh ===
rekomendasi_sim.input['rating'] = 4.5
rekomendasi_sim.input['ratings_count_norm'] = 90
rekomendasi_sim.input['ratings_5_norm'] = 80

# === 6. Proses dan Visualisasi ===
rekomendasi_sim.compute()
rekomendasi.view(sim=rekomendasi_sim)

# === 7. Ambil Hasil Output ===
hasil = rekomendasi_sim.output['rekomendasi']

# Interpretasi hasil
if hasil >= 70:
    keputusan = "Sangat Direkomendasikan"
elif hasil >= 40:
    keputusan = "Direkomendasikan"
else:
    keputusan = "Tidak Direkomendasikan"
```

Grafik Perhitungan Fuzzy, direkomendasikan

```
# === 4. Bangun dan Simulasi Sistem Fuzzy ===
rekomendasi_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8])
rekomendasi_sim = ctrl.ControlSystemSimulation(rekomendasi_ctrl)
```

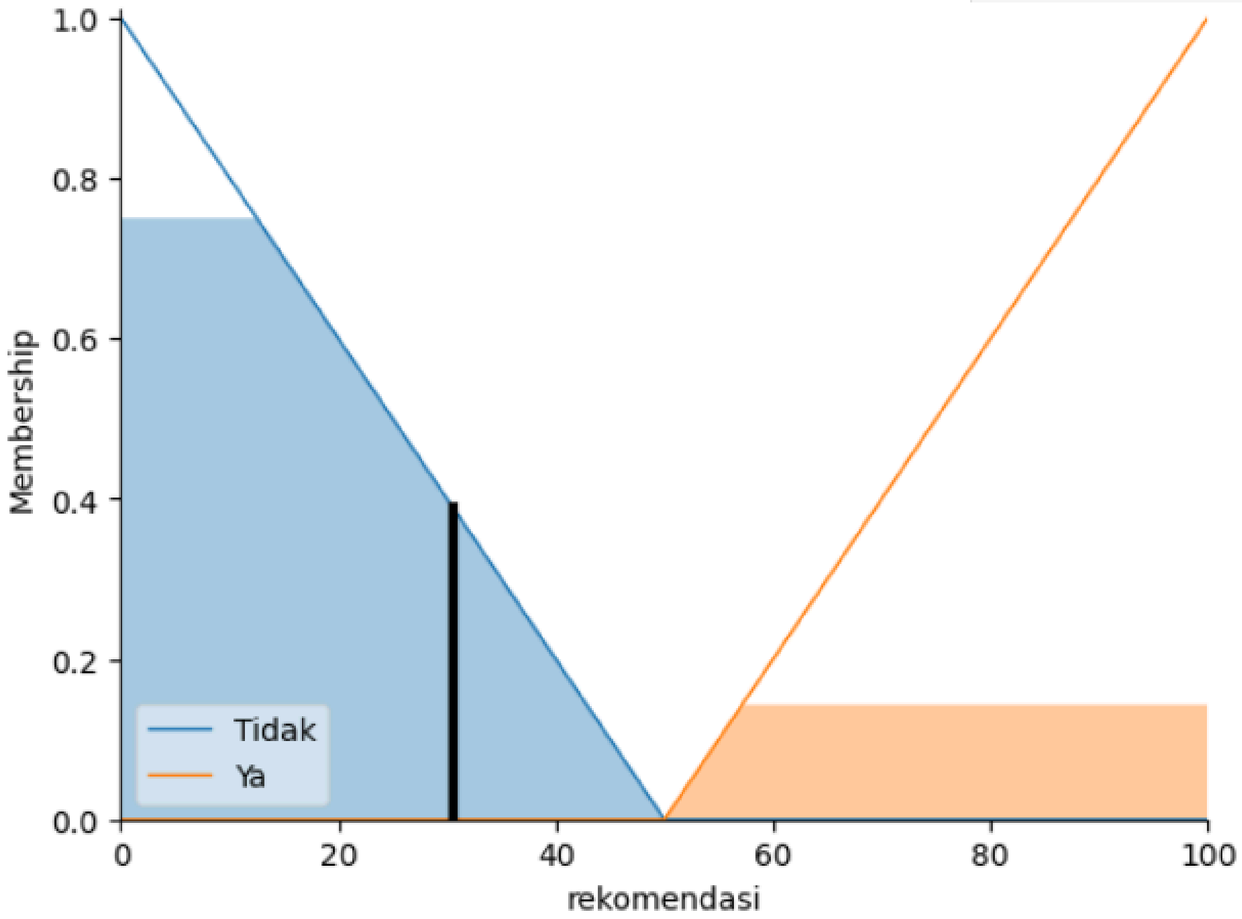
```
Hasil fuzzy: 56.21
Rekomendasi Buku: Direkomendasikan
```




```
# === 5. Masukkan Nilai Contoh ===
rekomendasi_sim.input['rating'] = 4.5
rekomendasi_sim.input['ratings_count_norm'] = 60
rekomendasi_sim.input['ratings_5_norm'] = 30

# === 6. Proses dan Visualisasi ===
rekomendasi_sim.compute()
rekomendasi.view(sim=rekomendasi_sim)

# === 7. Ambil Hasil Output ===
hasil = rekomendasi_sim.output['rekomendasi']

# Interpretasi hasil
if hasil >= 70:
    keputusan = "Sangat Direkomendasikan"
elif hasil >= 40:
    keputusan = "Direkomendasikan"
else:
    keputusan = "Tidak Direkomendasikan"
```

Grafik Perhitungan Fuzzy, tidak direkomendasikan

```
Hasil fuzzy: 30.57
Rekomendasi Buku: Tidak Direkomendasikan
```




```python
# === 4. Bangun dan Simulasi Sistem Fuzzy ===
rekomendasi_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8])
rekomendasi_sim = ctrl.ControlSystemSimulation(rekomendasi_ctrl)
```

```python
# === 5. Masukkan Nilai Contoh ===
rekomendasi_sim.input['rating'] = 2.5
rekomendasi_sim.input['ratings_count_norm'] = 40
rekomendasi_sim.input['ratings_5_norm'] = 10

# === 6. Proses dan Visualisasi ===
rekomendasi_sim.compute()
rekomendasi.view(sim=rekomendasi_sim)

# === 7. Ambil Hasil Output ===
hasil = rekomendasi_sim.output['rekomendasi']

# Interpretasi hasil
if hasil >= 70:
    keputusan = "Sangat Direkomendasikan"
elif hasil >= 40:
    keputusan = "Direkomendasikan"
else:
    keputusan = "Tidak Direkomendasikan"
```

Akurasi Sistem Fuzzy

```python
for index, row in books.iterrows():
    rating_val = row['average_rating']
    count_val = row['ratings_count_norm']
    rating5_val = row['ratings_5_norm']
    label = row['Label_Manual']

    rekomendasi_sim.input['rating'] = rating_val
    rekomendasi_sim.input['ratings_count_norm'] = count_val
    rekomendasi_sim.input['ratings_5_norm'] = rating5_val

    rekomendasi_sim.compute()
    result = rekomendasi_sim.output['rekomendasi']

    if hasil >= 60:
        pred = 2  # Sangat Direkomendasikan
    elif hasil >= 30:
        pred = 1  # Direkomendasikan
    else:
        pred = 0  # Tidak Direkomendasikan


    y_true.append(label)
    y_pred.append(pred)
```

```
Akurasi Sistem Fuzzy: 59.89%

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1912
           1       0.60      1.00      0.75      5989
           2       0.00      0.00      0.00      2099

    accuracy                           0.60     10000
   macro avg       0.20      0.33      0.25     10000
weighted avg       0.36      0.60      0.45     10000
```

Hasil akurasi sistem Fuzzy menunjukkan sebesar **59,89%**, yang artinya sistem fuzzy belum cukup baik dalam menyesuaikan rekomendasi dengan label manual. Sistem hanya efektif pada kelas 1 (Direkomendasikan), dengan precision 0.60 dan recall 1.00

Mean Squared Error (MSE)

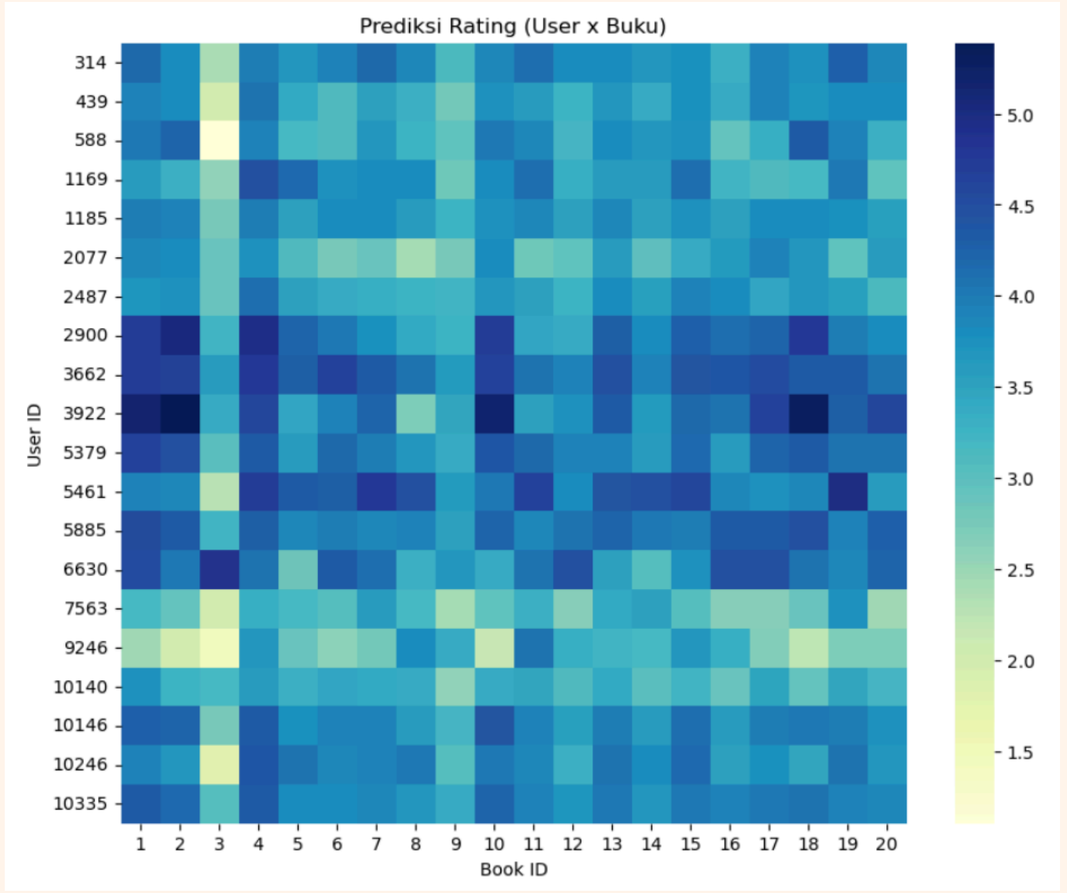
Proses perhitungan Mean Squared Error (MSE) untuk mengevaluasi performa model matrix factorization. Dataset ratings.csv diolah menjadi matriks prediksi menggunakan metode Stochastic Gradient Descent (SGD). Di akhir, MSE dihitung sebagai ukuran seberapa akurat prediksi model terhadap data asli — makin kecil MSE, makin baik performa model.

```python
# Load data
ratings = pd.read_csv('C:\\Users\\Velisia Nihan\\Downloads\\ratings.csv')
ratings = ratings.head(10000)  # subset kecil

# Mapping user_id dan book_id ke indeks integer
user_ids = ratings['user_id'].unique()
book_ids = ratings['book_id'].unique()

user_to_index = {user: i for i, user in enumerate(user_ids)}
book_to_index = {book: i for i, book in enumerate(book_ids)}

ratings['user_index'] = ratings['user_id'].map(user_to_index)
ratings['book_index'] = ratings['book_id'].map(book_to_index)
```

```python
n_users = len(user_ids)
n_books = len(book_ids)
n_factors = 30  # faktor laten lebih banyak
alpha = 0.005   # learning rate lebih kecil
n_epochs = 50
reg = 0.02      # regularisasi sedikit lebih besar

# Inisialisasi matriks P dan Q
P = np.random.normal(scale=1./n_factors, size=(n_users, n_factors))
Q = np.random.normal(scale=1./n_factors, size=(n_books, n_factors))

# Training matrix factorization dengan SGD
for epoch in range(n_epochs):
    for row in ratings.itertuples():
        u = row.user_index
        i = row.book_index
        r_ui = row.rating

        pred = np.dot(P[u, :], Q[i, :].T)
        err = r_ui - pred

        P[u, :] += alpha * (err * Q[i, :] - reg * P[u, :])
        Q[i, :] += alpha * (err * P[u, :] - reg * Q[i, :])

    # Hitung MSE per epoch untuk monitoring
    preds = np.array([np.dot(P[row.user_index], Q[row.book_index].T) for row in ratings.itertuples()])
    mse = mean_squared_error(ratings['rating'], preds)
```

Grafik Matrix

Gambar ini menunjukkan hasil prediksi rating antara pengguna dan buku menggunakan metode matrix factorization. Warna dalam grafik merepresentasikan nilai rating—semakin terang artinya rating yang diprediksi semakin tinggi

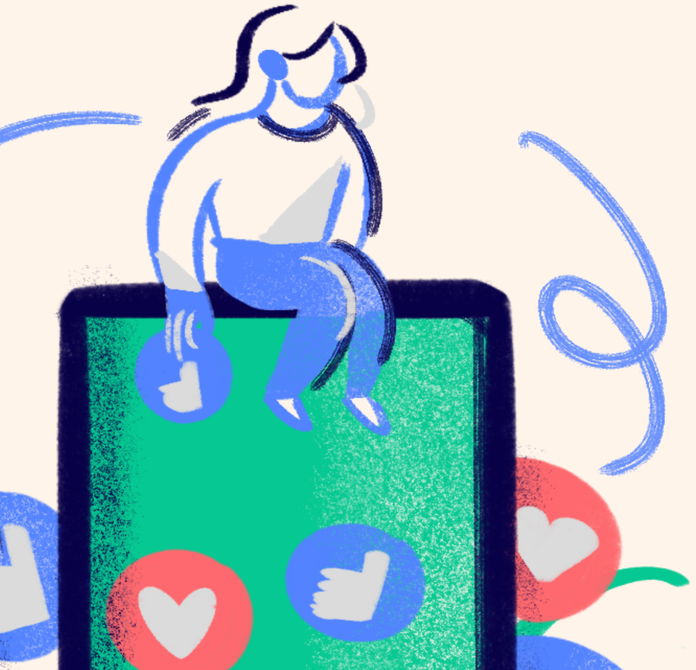


```
pred_matrix = np.dot(P, Q.T)

# Contoh visualisasi heatmap prediksi 20x20
pred_df = pd.DataFrame(pred_matrix[:20, :20], index=user_ids[:20], columns=book_ids[:20])

plt.figure(figsize=(10, 8))
sns.heatmap(pred_df, cmap='YlGnBu', annot=False)
plt.title('Prediksi Rating (User x Buku)')
plt.xlabel('Book ID')
plt.ylabel('User ID')
plt.show()
```

Akurasi Matrix

Hasilnya, akurasi klasifikasi mencapai **86,99%**, dengan performa terbaik pada prediksi (kelas 1), ditunjukkan oleh precision 0.88 dan recall 0.94.
Artinya, model cukup andal dalam memprediksi buku yang disukai pengguna, meskipun sedikit kurang akurat untuk buku yang tidak disukai.

```python
for row in ratings.itertuples():
    u = row.user_index
    i = row.book_index
    true_rating = row.rating
    pred_rating = pred_matrix[u, i]

    # Klasifikasi manual
    true_label = 1 if true_rating >= threshold else 0
    pred_label = 1 if pred_rating >= threshold else 0

    y_true.append(true_label)
    y_pred.append(pred_label)
```



```
Akurasi klasifikasi berdasarkan threshold 3.5: 86.99%

Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.73      0.79      3294
           1       0.88      0.94      0.91      6706

    accuracy                           0.87     10000
   macro avg       0.86      0.83      0.85     10000
weighted avg       0.87      0.87      0.87     10000
```

Kesimpulan

Sistem rekomendasi buku berbasis logika fuzzy menghasilkan akurasi sebesar 59,89%, dengan performa terbaik pada kelas Direkomendasikan (precision 0.60, recall 1.00). Meski mampu mengenali buku yang layak direkomendasikan, sistem masih kurang akurat untuk buku yang tidak direkomendasikan.

Hal ini bisa terjadi karena sistem fuzzy sangat bergantung pada aturan IF–THEN yang bersifat statik dan ditentukan secara manual, sehingga kurang fleksibel dalam menangkap pola kompleks dari data pengguna secara keseluruhan.

Sebaliknya, metode Matrix Factorization menunjukkan akurasi lebih tinggi (86,99%) dan performa yang lebih konsisten.

Dengan demikian, meskipun sistem fuzzy memberikan interpretabilitas dan fleksibilitas dalam menyusun aturan, Matrix Factorization unggul dari sisi akurasi, konsistensi, dan skalabilitas

Referensi

Kaggle, “Goodbooks-10k Dataset.” https://www.kaggle.com/datasets/zygmunt/goodbooks-10k

Zadeh, L. A. (1965). "Fuzzy sets." *Information and Control*, 8(3), 338–353.

Jang, J.-S. R., Sun, C.-T., & Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing*. Prentice Hall.

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

Ricci, F., Rokach, L., & Shapira, B. (2011). *Introduction to Recommender Systems Handbook*. Springer.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8), 30-37. https://doi.org/10.1109/MC.2009.263

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer.

Terima Kasih