TUGAS 4 DATA MINING

ADE NAFIL FIRMANSAH (3012210002)
PROGRAM STUDI INFORMATIKA
UNIVERSITAS INTERNASIONAL SEMEN INDONESIA
GRESIK

DAFTAR ISI

DAFTAR ISI.....2
1. Latar Belakang.....1
1.1 Tujuan.....1
2. Metodologi.....1
2.1 Studi Kasus.....1
2.2 Dataset.....2
2.3 Pengisian Form/Data Transaksi.....2
2.4 Algorithm.....3
3. Hasil Dan Pembahasan.....3
3.1. Exploratory Data Analysis.....4
3.2. Apriori.....6
3.3. FP-Growth.....8
3.4.Top 4 Association Rules.....11
3.5.Visualisasi Association Rules.....13

1. Latar Belakang

Analisis Association Rules adalah salah satu teknik penting dalam data mining yang digunakan untuk menemukan pola hubungan antar item dalam dataset transaksi. Teknik ini sangat berguna dalam berbagai industri, terutama retail dan Food & Beverage (F&B), untuk memahami perilaku pembelian pelanggan.

Dalam industri cafe dan restoran, memahami pola pembelian pelanggan sangat penting untuk:

a. Meningkatkan strategi cross-selling
b. Mengoptimalkan penempatan menu pada display
c. Membuat paket promosi yang menarik
d. Mengelola stok bahan baku lebih efisien

Dengan menerapkan teknik Association Rules, manajemen cafe dapat mengambil keputusan berbasis data yang lebih akurat dan efektif.

1.1 Tujuan

Tujuan dari penugasan ini adalah

1. Menerapkan algoritma Apriori dan FP-Growth untuk menganalisis pola pembelian pelanggan
2. Menemukan association rules yang berguna untuk meningkatkan strategi bisnis cafe
3. Membandingkan hasil dan efisiensi kedua algoritma
4. erikan rekomendasi bisnis yang praktis berdasarkan hasil analisis

2. Metodologi

2.1 Studi Kasus

Cafe "Kopi Senja" adalah kedai kopi lokal yang menyajikan berbagai menu minuman dan makanan. Responden dalam penelitian ini adalah 10 orang yang melakukan transaksi pembelian di cafe tersebut. Tujuannya adalah untuk memahami pola pemesanan pelanggan guna meningkatkan strategi bisnis cafe.

2.2 Dataset

a. Responden
   - Dinunaya Syuja Aryoko
   - Ahmad Arya Dwi Febriansyah
   - Vito Agatha Satritama
   - Moh. Dani Wahyudi
   - Achmad Wildan Muzaky
   - Frenky
   - Andries Nauvalentin Roestam
   - Muhammad Alvin Firdaus
   - Husni Mubarok
   - Abiyyu Valin Zavero

b. Menu Cafe
   1. Makanan
      - Roti Bakar
      - Croissant
      - Donut
      - Kue Coklat
      - Pisang Goreng
      - Nasi Goreng
      - Mie Goreng
   2. Minuman
      - Kopi Hitam
      - Kopi Susu
      - Cappuccino
      - Latte
      - Teh Manis

- Jus Jeruk
- Air Mineral

3. Snack
    - Keripik Kentang
    - Biskuit

2.3 Pengisian Form/Data Transaksi

Responden	Item yang dipesan
Dinunaya Syuja Aryoko	Kopi Susu, Roti Bakar, Keripik Kentang
Ahmad Arya Dwi Febriansyah	Cappuccino, Croissant, Air Mineral
Vito Agatha Satritama	Kopi Hitam, Roti Bakar, Keripik Kentang, Biskuit
Moh. Dani Wahyudi	Kopi Susu, Roti Bakar, Jus Jeruk
Achmad Wildan Muzaky	Teh Manis, Donut, Air Mineral
Frenky	Cappuccino, Croissant, Keripik Kentang
Andries Nauvalentin Roestam	Kopi Susu, Roti Bakar, Donut
Muhammad Alvin Firdaus	Latte, Nasi Goreng, Air Mineral
Husni Mubarok	Kopi Hitam, Roti Bakar, Keripik Kentang
Abiyyu Valin Zavero	Teh Manis, Donut, Kue Coklat

2.4 Algorithm

a. Parameter yang digunakan
    - Minimum support : 30%
    - Minimum support count : 3 transaksi
    - Minimum confidence : 70%

b. Implementasi

- Apriori Python : Menggunakan library mlxtend
- FP-Growth Python : Menggunakan library mlxtend

3. Hasil Dan Pembahasan

Sebelum melakukan clustering, saya harus bersihkan dulu datanya.

3.1. Exploratory Data Analysis

```
# Menghitung frekuensi kemunculan item
item_counts = df.sum().sort_values(ascending=False)

# Menampilkan frekuensi item
print("Frekuensi Kemunculan Item:")
for item, count in item_counts.items():
    print(f"{item}: {count}")

# Menampilkan statistik deskriptif
print("\nStatistik Deskriptif:")
print(f"Total Item Terjual: {df.sum().sum()}")
print(f"Rata-rata Item per Transaksi: {df.sum().mean():.2f}")
print(f"Minimum Item per Transaksi: {df.sum(axis=1).min()}")
print(f"Maximum Item per Transaksi: {df.sum(axis=1).max()}")
```

Penjelasan Code :

- df.sum(): Menghitung jumlah True (1) untuk setiap kolom item
- sort_values(ascending=False): Mengurutkan hasil dari yang tertinggi ke terendah
- df.sum().sum(): Menghitung total semua item yang terjual
- df.sum().mean(): Menghitung rata-rata item per transaksi
- df.sum(axis=1).min() dan .max(): Menghitung minimum dan maximum item per transaksi

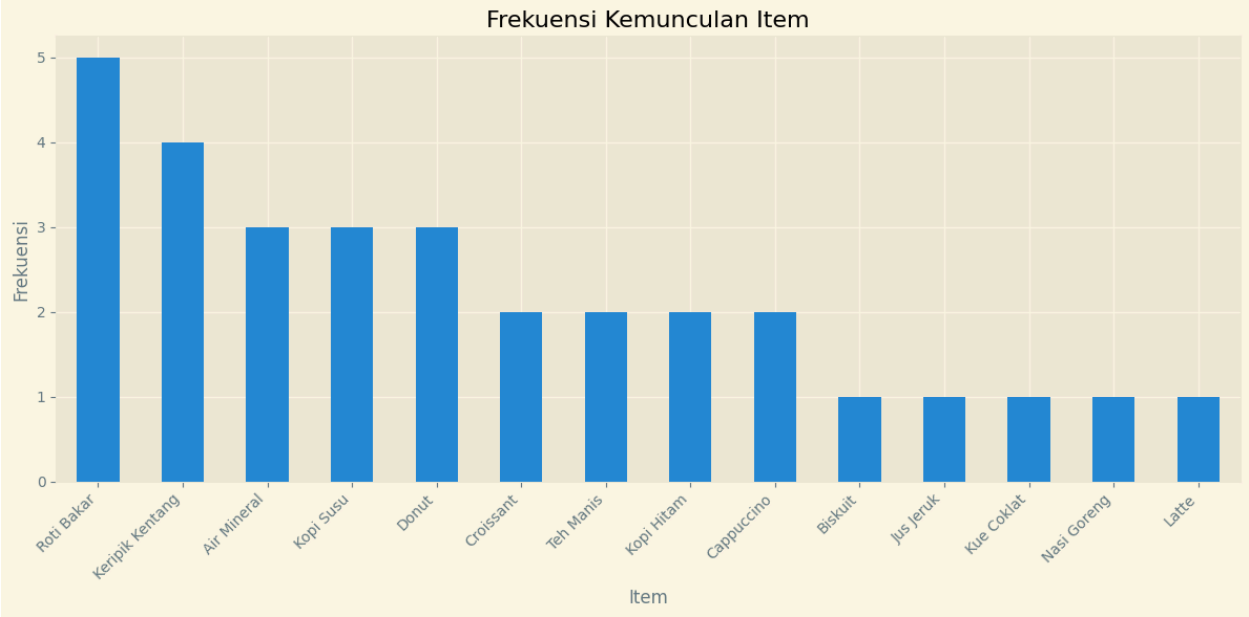
Hasil Code :

```
Frekuensi Kemunculan Item:
Roti Bakar: 5
Keripik Kentang: 4
Air Mineral: 3
Kopi Susu: 3
Donut: 3
Croissant: 2
Teh Manis: 2
Kopi Hitam: 2
Cappuccino: 2
Biskuit: 1
Jus Jeruk: 1
Kue Coklat: 1
Nasi Goreng: 1
Latte: 1
```

```python
print("\nStatistik Deskriptif:")
print(f"Total Item Terjual: {df.sum().sum()}")
print(f"Rata-rata Item per Transaksi: {df.sum().mean():.2f}")
print(f"Minimum Item per Transaksi: {df.sum(axis=1).min()}")
print(f"Maximum Item per Transaksi: {df.sum(axis=1).max()}")
```

✓ 0.0s

```
Statistik Deskriptif:
Total Item Terjual: 31
Rata-rata Item per Transaksi: 2.21
Minimum Item per Transaksi: 3
Maximum Item per Transaksi: 4
```



3.2. Apriori

Code:

```
from mlxtend.frequent_patterns import apriori

# Generate frequent itemsets
frequent_itemsets = apriori(df, min_support=0.3, use_colnames=True)

print("=== Frequent Itemsets (Apriori) ===")
print(frequent_itemsets)

# Add length column
print("\n=== Frequent Itemsets by Length ===")
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x:
len(x))
print(frequent_itemsets.sort_values(['length', 'support'], ascending=[True,
False]))

# Generate association rules
rules = association_rules(frequent_itemsets, metric="confidence",
min_threshold=0.7)

print("\n=== Association Rules (Apriori) ===")
print("Metric: Confidence >= 70%")
print(rules[['antecedents', 'consequents', 'support', 'confidence',
```

```
'lift']].to_string(index=False))

# Sort rules by lift and confidence
rules_sorted = rules.sort_values(['lift', 'confidence'], ascending=[False,
False])
print("\n=== Top 5 Rules berdasarkan Lift dan Confidence ===")
print(rules_sorted[['antecedents', 'consequents', 'support', 'confidence',
'lift']].head().to_string(index=False))
```

Penjelasan Code:

- apriori(df, min_support=0.3, use_colnames=True): Mencari frequent itemsets dengan minimum support 0.3
- frequent_itemsets['itemsets'].apply(lambda x: len(x)): Menambah kolom panjang itemset
- association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7): Membuat rules dengan minimum confidence 0.7
- sort_values(['lift', 'confidence'], ascending=[False, False]): Mengurutkan rules berdasarkan lift tertinggi, kemudian confidence

Hasil Code :

a) Frequent Itemsets (Apriori)

Support	Itemsets
0.3	Air Mineral
0.3	Donut
0.4	Keripik Kentang
0.3	Kopi Susu
0.5	Roti Bakar
0.3	Roti Bakar, Keripik Kentang
0.3	Roti Bakar, Kopi Susu

b) Frequent Itemsets by Length

Support	Itemsets	Length
0.5	Roti Bakar	1

0.4	Keripik Kentang	1
0.3	Air Mineral	1
0.3	Donut	1
0.3	Kopi Susu	1
0.3	Roti Bakar, Keripik Kentang	2
0.3	Roti Bakar, Kopi Susu	2

c) Association Rules (Apriori) - Metric: Confidence ⩾ 70%:

Antecedents	Consequents	Support	Confidence	Lift
Keripik Kentang	Roti Bakar	0.3	0.75	1.5
Kopi Susu	Roti Bakar	0.3	1.00	2.0

d) Top 5 Rules berdasarkan Lift dan Confidence:

Antecedents	Consequents	Support	Confidence	Lift
Kopi Susu)	Roti Bakar	0.3	1.00	2.0
Keripik Kentang	Roti Bakar	0.3	0.75	1.5

3.3. FP-Growth

from mlxtend.frequent_patterns import fpgrowth

```
# Generate frequent itemsets
frequent_itemsets_fpgrowth = fpgrowth(df, min_support=0.3,
use_colnames=True)

print("=== Frequent Itemsets (FP-Growth) ===")
```

```
print(frequent_itemsets_fpgrowth)

# Add length column
print("\n=== Frequent Itemsets by Length ===")
frequent_itemsets_fpgrowth['length'] =
frequent_itemsets_fpgrowth['itemsets'].apply(lambda x: len(x))
print(frequent_itemsets_fpgrowth.sort_values(['length', 'support'],
ascending=[True, False]))

# Generate association rules
rules_fpgrowth = association_rules(frequent_itemsets_fpgrowth,
metric="confidence", min_threshold=0.7)

print("\n=== Association Rules (FP-Growth) ===")
print("Metric: Confidence >= 70%")
print(rules_fpgrowth[['antecedents', 'consequents', 'support',
'confidence', 'lift']].to_string(index=False))

# Sort rules by lift and confidence
rules_fpgrowth_sorted = rules_fpgrowth.sort_values(['lift', 'confidence'],
ascending=[False, False])
print("\n=== Top 5 Rules berdasarkan Lift dan Confidence ===")
print(rules_fpgrowth_sorted[['antecedents', 'consequents', 'support',
'confidence', 'lift']].head().to_string(index=False))
```

Penjelasan Code:

- fpgrowth(df, min_support=0.3, use_colnames=True): Menggunakan algoritma FP-Growth untuk mencari frequent itemsets
- Parameter dan output yang dihasilkan sama seperti Apriori

Hasil Code :

a. Frequent Itemsets (FP-Growth)

Support	Itemsets
0.5	Roti Bakar
0.4	Keripik Kentang
0.3	Kopi Susu
0.3	Air Mineral
0.3	Donut
0.3	Roti Bakar, Keripik Kentang

0.3	Roti Bakar, Kopi Susu

b. Frequent Itemsets by Length

Support	Itemsets	Length
0.5	Roti Bakar	1
0.4	Keripik Kentang	1
0.3	Kopi Susu	1
0.3	Air Mineral	1
0.3	Donut	1
0.3	Roti Bakar, Keripik Kentang	2
0.3	Roti Bakar, Kopi Susu	2

c. Association Rules (FP-Growth) - Metric: Confidence ⩾ 70%

Antecedents	Consequents	Support	Confidence	Lift
Keripik Kentang	Roti Bakar	0.3	0.75	1.5
Kopi Susu	Roti Bakar	0.3	1.00	2.0

d. Top 5 Rules berdasarkan Lift dan Confidence

Antecedents	Consequents	Support	Confidence	Lift
Kopi Susu)	Roti Bakar	0.3	1.00	2.0
Keripik Kentang	Roti Bakar	0.3	0.75	1.5

3.4.Top 4 Association Rules

Code untuk Mengambil Top 4 Rules:

```
# Mengambil 4 rules terbaik berdasarkan lift dan confidence
top4_rules = rules.sort_values(['lift', 'confidence'], ascending=[False, False]).head(4)

print("=== TOP 4 ASSOCIATION RULES ===")
for idx, row in top4_rules.iterrows():
    antecedents = ', '.join(list(row['antecedents']))
    consequents = ', '.join(list(row['consequents']))
    print(f"\nRule {idx+1}:")
    print(f"  {antecedents} -> {consequents}")
    print(f"  Support: {row['support']:.4f} ({row['support']*100:.2f}%)")
    print(f"  Confidence: {row['confidence']:.4f} ({row['confidence']*100:.2f}%)")
    print(f"  Lift: {row['lift']:.4f}")

    if row['lift'] > 1:
        print(f"  Interpretasi: Positive correlation (lift > 1)")
    elif row['lift'] == 1:
        print(f"  Interpretasi: No correlation (lift = 1)")
    else:
        print(f"  Interpretasi: Negative correlation (lift < 1)")
```

Penjelasan Code

- sort_values(['lift', 'confidence'], ascending=[False, False]): Mengurutkan rules berdasarkan lift (descending), kemudian confidence (descending)
- head(4): Mengambil 4 rules teratas
- list(row['antecedents']): Mengubah frozenset menjadi list untuk ditampilkan
- row['support']*100: Mengonversi nilai support desimal ke persentase
- Kondisi `if row['lift'] > 1`: Memberikan interpretasi berdasarkan nilai lift

Hasil Top 4 Association Rules

a) Rule 2: Kopi Susu → Roti Bakar

Metric	Nilai

Support	0.3000 (30.00%)
Confidence	1.0000 (100.00%)
Lift	2.0000

Interpretasi :

- Support 30%: 30% dari semua transaksi mengandung Kopi Susu dan Roti Bakar bersamaan (3 dari 10 transaksi)
- Confidence 100%: 100% pelanggan yang membeli Kopi Susu juga membeli Roti Bakar (3 dari 3 transaksi yang mengandung Kopi Susu)
- Lift 2.0: Nilai lift > 1 menunjukkan **positive correlation**. Pelanggan 2 kali lebih mungkin membeli Roti Bakar jika mereka membeli Kopi Susu dibandingkan secara acak. Ini adalah korelasi positif yang sangat kuat.

b) Rule 1: Keripik Kentang → Roti Bakar

Metric	**Nilai**
Support	0.3000 (30.00%)
Confidence	0.7500 (75.00%)
Lift	1.5000

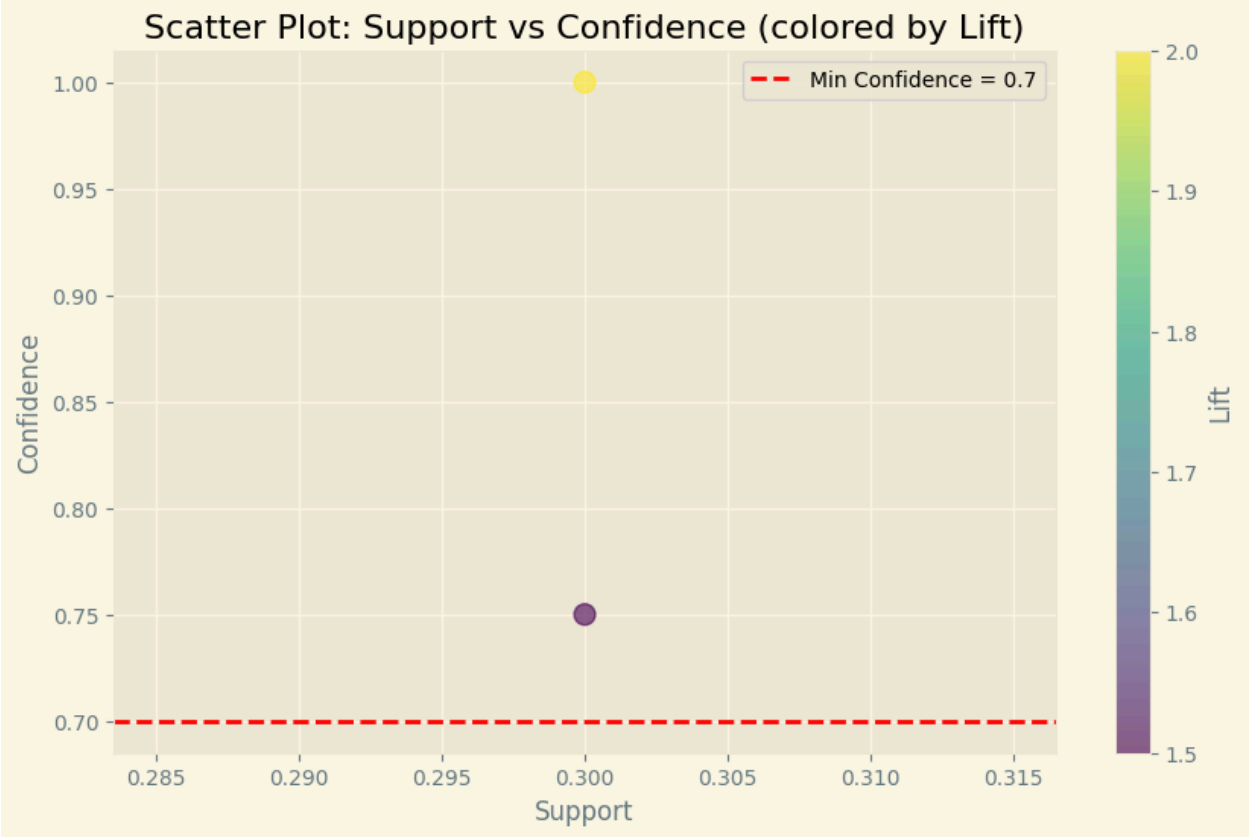
Interpretasi :

- Support 30%: 30% dari semua transaksi mengandung Keripik Kentang dan Roti Bakar bersamaan (3 dari 10 transaksi)
- Confidence 100%: Nilai lift > 1 menunjukkan **positive correlation**. Pelanggan 1.5 kali lebih mungkin membeli Roti Bakar jika mereka membeli Keripik Kentang dibandingkan secara acak.
- Lift 2.0: Keripik Kentang dan Roti Bakar juga memiliki korelasi positif yang baik. Sebagian besar pelanggan yang membeli Keripik Kentang akan membeli Roti Bakar.

3.5.Visualisasi Association Rules

Scatter Plot: Support vs Confidence (colored by Lift) :

```
plt.figure(figsize=(10, 6))
scatter = plt.scatter(rules['support'], rules['confidence'],
                      c=rules['lift'], s=100, alpha=0.6, cmap='viridis')
plt.xlabel('Support')
plt.ylabel('Confidence')
plt.title('Scatter Plot: Support vs Confidence (colored by Lift)')
plt.colorbar(scatter, label='Lift')
plt.axhline(y=0.7, color='r', linestyle='--', label='Min Confidence = 0.7')
plt.legend()
plt.show()
```



Top 4 Rules: Support, Confidence, and Lift :

```
top4_rules_plot = top4_rules.copy()
top4_rules_plot['rule'] = [f"{list(r['antecedents'])} ->
{list(r['consequents'])}"
                           for _, r in top4_rules_plot.iterrows()]
```

```
plt.figure(figsize=(12, 6))
x_pos = np.arange(len(top4_rules_plot))
plt.bar(x_pos - 0.2, top4_rules_plot['support'], 0.2, label='Support')
plt.bar(x_pos, top4_rules_plot['confidence'], 0.2, label='Confidence')
plt.xlabel('Rules')
plt.ylabel('Value')
plt.title('Top 4 Rules: Support, Confidence, and Lift')
plt.xticks(x_pos, top4_rules_plot['rule'], rotation=45, ha='right')
plt.legend()
plt.tight_layout()
plt.show()
```