MODUL PRAKTIKUM
MATA KULIAH: DATA MINING
MODUL VIII
[**Algoritma Apriori dalam Analisis Asosiasi**]

`

Program Studi Sains Data

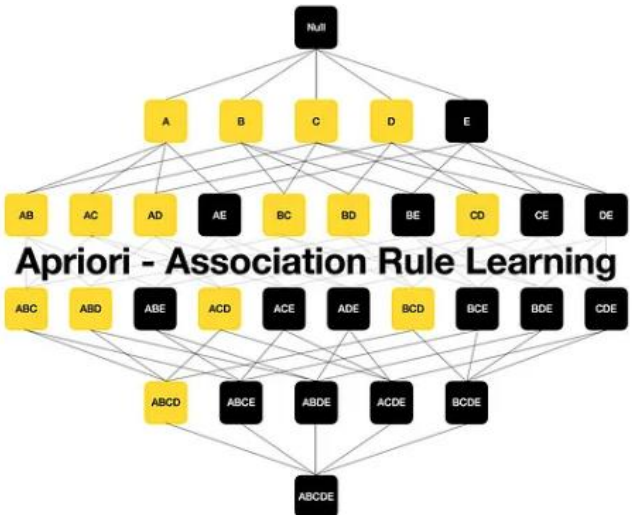
Fakultas Sains

INSTITUT TEKNOLOGI SUMATERA

Tahun Ajaran Ganjil 2024/2025.

1. **Tujuan Praktikum**
   - Dapat memahami algoritma apriori untuk *asosiasi rule learning*.
   - Dapat mengimplementasikan metode apriori dalam algoritma python.
2. **Landasan teori**

Algoritma Apriori adalah metode data mining yang digunakan untuk menemukan aturan asosiasi dalam data transaksi, seperti pola belanja pelanggan. Algoritma Apriori digunakan untuk menemukan hubungan antar item dalam dataset besar.



Gambar 1 Skema apriori- Association Rule Learning.

Association Rule Learning

Association Rule Learning atau pembelajaran aturan asosiasi adalah metode pembelajaran mesin berbasis aturan untuk menemukan hubungan menarik antara variabel dalam basis data besar. Association rule mining adalah teknik yang digunakan untuk mengidentifikasi pola dalam kumpulan data besar. Teknik ini melibatkan pencarian hubungan antara variabel dalam data dan menggunakan hubungan tersebut untuk membuat prediksi atau keputusan. Tujuan penambangan aturan asosiasi adalah untuk mengungkap aturan yang menggambarkan hubungan antara berbagai item dalam dataset. Biasanya association rule learning terdiri dari algoritma untuk menganalisis data dan mengidentifikasi hubungan.

Algoritma ini dapat didasarkan pada metode statistik atau teknik pembelajaran mesin. Aturan yang dihasilkan sering kali dinyatakan dalam bentuk pernyataan "jika-maka", di mana bagian "jika" mewakili anteseden (kondisi yang diuji) dan bagian "maka" mewakili konsekuen (hasil yang terjadi jika kondisi terpenuhi). Hal ini dapat berguna untuk berbagai keperluan, seperti mengidentifikasi tren pasar, mendeteksi aktivitas penipuan, atau memahami perilaku pelanggan. Penambangan aturan asosiasi juga dapat digunakan sebagai batu loncatan untuk jenis analisis data lainnya, seperti memprediksi hasil atau mengidentifikasi pendorong utama fenomena tertentu.

Use Cases of Association Rule Mining

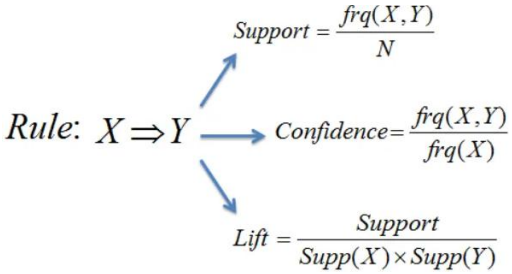
Penambangan aturan asosiasi umumnya digunakan dalam berbagai aplikasi, beberapa yang umum adalah:

- Analisis Keranjang Belanja
- Customer Segmentation
- Fraud Detection
- Social network analysis
- Recommendation systems

Metrics for Evaluating Association Rules

Dalam penambangan aturan asosiasi, beberapa metrik umumnya digunakan untuk mengevaluasi kualitas dan pentingnya aturan asosiasi yang ditemukan. Dalam penambangan aturan asosiasi, beberapa metrik umumnya digunakan untuk mengevaluasi kualitas dan pentingnya aturan asosiasi yang ditemukan.

Menafsirkan hasil metrik penambangan aturan asosiasi memerlukan pemahaman tentang makna dan implikasi setiap metrik, serta cara menggunakannya untuk mengevaluasi kualitas dan pentingnya aturan asosiasi yang ditemukan. Berikut adalah beberapa panduan untuk menafsirkan hasil metrik penambangan aturan asosiasi utama:



Gambar 2 matriks evaluasi untuk Association Rules

Dari gambar diatas dapat dijabarkan dibawah ini:

- Support

  Support merupakan Ukuran seberapa sering item atau set item muncul dalam set data. Nilai ini dihitung sebagai jumlah transaksi yang memuat item dibagi dengan jumlah total transaksi dalam kumpulan data. Nilai dukungan yang tinggi menunjukkan bahwa suatu item atau kumpulan item umum dalam kumpulan data, sedangkan nilai dukungan yang rendah menunjukkan bahwa item atau kumpulan item tersebut jarang.

$$Support(\{X\} \rightarrow \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Total\ number\ of\ transactions}$$

  Contoh: Misalnya dari 1000 transaksi, 100 transaksi berisi Saus Tomat(*Ketchup*), maka Support untuk item Saus Tomat dapat dihitung sebagai berikut:

  Support(Ketchup) = (Transactions containingKetchup)/(Total Transactions)
  Support(Ketchup) = 100/1000
  = 10%

- Confidence(Keyakinan)
  Confidence adalah Ukuran kekuatan hubungan antara dua item. Nilai ini dihitung sebagai jumlah transaksi yang memuat kedua item dibagi dengan jumlah transaksi yang memuat item pertama. Confidence yang tinggi menunjukkan bahwa keberadaan item pertama merupakan prediktor kuat keberadaan item kedua.

  $$Confidence(\{X\} \rightarrow \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Transactions\ containing\ X}$$

  Contoh: Kembali ke masalah kita, kita memiliki 50 transaksi di mana Burger dan Saus Tomat dibeli bersamaan. Sementara dalam 150 transaksi, burger dibeli. Maka kita dapat menemukan kemungkinan membeli saus tomat ketika burger dibeli dapat direpresentasikan sebagai confidence Burger -> Saus Tomat dan dapat ditulis secara matematis sebagai:

  Confidence(Burger→Ketchup) = (Transactions containing both (Burger and Ketchup))/(Transactions containing A)

  Confidence(Burger→Ketchup) = 50/150
  = 33.3%

- Lift
  Lift adalah ukuran kekuatan hubungan antara dua item, dengan mempertimbangkan frekuensi kedua item dalam kumpulan data. Nilai ini dihitung sebagai Confidence hubungan dibagi dengan dukungan item kedua. Lift digunakan untuk membandingkan kekuatan hubungan antara dua item dengan kekuatan hubungan yang diharapkan jika item-item tersebut independen.

  $$Lift(\{X\} \rightarrow \{Y\}) = \frac{(Transactions\ containing\ both\ X\ and\ Y)/(Transactions\ containing\ X)}{Fraction\ of\ transactions\ containing\ Y}$$

  Contoh: jika kita menggunakan data dari masalah diatas terkait dengan saus tomat dan burger maka kita akan dapat menghitung nilai lift nya yaitu:
  Lift(Burger→Ketchup)=(Confidence(Burger→Ketchup))/(Support(Ketchup))
  Lift(Burger→Ketchup) = 33.3/10
  = 3.33

3. Prosedur praktikum

Untuk dataset modul 8 dapat diakses, link:https://bit.ly/damin_datamodul8.
Install efficient-apriori.

```
1 pip install efficient-apriori
```

Setelah itu kita tuliskan library yang kita butuhkan untuk algotitma apriori ini

```
import pandas as pd # for data manipulation
import matplotlib.pyplot as plt # for ploting frequency distribution chart
from efficient_apriori import apriori # for association analysis
import sys
import os
# Assign main directory to a variable
main_dir=os.path.dirname(sys.path[0])
```

Masukan data Market Basket Optimisation.csv yang telah diunduh.

```
# Ingest the data
df = pd.read_csv(main_dir+'//content/Market_Basket_Optimisation.csv',
                 encoding='utf-8', header=None)
# Show dataframe
df
```

Exploration

Sebelum kita menjalankan analisis aturan asosiasi, mari kita lihat terlebih dahulu distribusi frekuensi item.

```
# Put all transactions into a single list
txns=df.values.reshape(-1).tolist()
# Create a dataframe using this single list and add a column for count
df_list=pd.DataFrame(txns)
df_list['Count']=1
# Group by items and rename columns
df_list=df_list.groupby(by=[0], as_index=False).count().sort_values(by=['Count'], ascending=True) # count
df_list['Percentage'] = (df_list['Count'] / df_list['Count'].sum()) # percentage
df_list=df_list.rename(columns={0 : 'Item'})
# Show dataframe
df_list
```

Dari output yang diperoleh tahap exploitasi menunjukan bahwa air mineral (1.788) sejauh ini merupakan barang paling populer yang dibeli di supermarket ini, diikuti oleh telur (1.348), spageti (1.306), kentang goreng (1.282), dan cokelat (1.230). Sementara itu, asparagus tua yang malang hanya dibeli satu kali. Penting juga untuk dicatat bahwa bahkan barang yang paling sering dibeli hanya muncul dalam lebih dari 6% transaksi. Kita dapat menggunakan informasi ini sebagai panduan saat menetapkan ambang batas dukungan minimum/ *minimum support threshold*.

Visualisasi distribusi frekuensi dalam diagram batang sebagai berikut:

```
# Draw a horizontal bar chart
plt.figure(figsize=(16,20), dpi=300)
plt.ylabel('Item Name')
plt.xlabel('Count')
plt.barh(df_list['Item'], width=df_list['Count'], color='black', height=0.8)
plt.margins(0.01)
plt.show()
```

Algoritma Apriori

Sebelum kita menjalankan algoritma, mari kita masukkan data kita dalam format yang diperlukan. Ini akan menjadi daftar daftar yang telah menghapus semua "NaN" yang Anda lihat di Pandas DataFrame.

```
# Create a list of lists from a dataframe
txns2=df.stack().groupby(level=0).apply(list).tolist()
# Show what it looks like
txns2
```

Terakhir, mari kita jalankan algoritma Apriori dan simpan itemset dan aturan asosiasi.

```
itemsets, rules = apriori(txns2, min_support=0.03, min_confidence=0.2, verbosity=1)
```

Algoritme tersebut telah menemukan 36 itemset berukuran 1 dan 18 itemset berukuran 2, yang memenuhi ambang batas dukungan minimum sebesar 0,03. Itemset tersebut ditampilkan di bawah ini.

Sekarang mari kita cetak aturan asosiasi yang ditemukan oleh algoritma. Perhatikan, beberapa set dikecualikan dalam tahap pembuatan aturan karena tidak memenuhi persyaratan keyakinan minimum yang kami tentukan (dalam contoh ini, min_confidence=0.2).

```
for item in sorted(rules, key=lambda item: (item.lift,item.conviction), reverse=True):
    print(item)
```

Dari hasil algoritma diatas kita dapat melihat bahwa Seperti peningkatan tertinggi dalam data kehidupan nyata ini adalah untuk kombinasi daging sapi giling dan spageti. Namun, pembeli daging sapi giling lebih mungkin untuk juga membeli spageti(confidence: 0.399, conviction: 1.374) daripada sebaliknya(confidence: 0.225, conviction: 1.164).

Latihan

Dalam *Apriori Algorithm for Association Rule Learning* kita dapat membagi kedalam beberapa tahapan:

- Libraries

```
[53]  1 import numpy as np
      2 import pandas as pd
      3 import seaborn as sns
      4 import matplotlib.pyplot as plt
      5
      6 import itertools
      7 from mlxtend.preprocessing import TransactionEncoder
      8
      9 import warnings
     10 warnings.filterwarnings("ignore")
```

- Import DataSet
  dalam Latihan modul 8 digunakan dataset GroceryStoreDataSet.csv yang dapat di unduh di https://bit.ly/damin_datamodul8.

```
data =pd.read_csv('GroceryStoreDataSet.csv',header=None)
data.columns=["Product"]
data
```

```
print("Number Of Rows is :" ,data.shape[0])
print("Number Of Columns is :" ,data.shape[1])
```

```
data.values
```

- data preparation
  dalam modul ini kita akan menggunakan algoritma Apriori untuk melakukan analisis relevansi. Untuk mengonversi data, saya akan kembali menggunakan metode konversi data dari library mlxtend

```
data =list(data["Product"].apply(lambda x:x.split(",")))
data
```

- TransactionEncoder
  Dengan menggunakan fungsi ini, kita menggunakan nilai Boolean untuk setiap transaksi dan memasukkan nama setiap produk dalam kolom.

```
te =TransactionEncoder()
te_data =te.fit_transform(data)
df =pd.DataFrame(te_data ,columns =te.columns_)
df
```

```
df_zero =df.replace(True ,1)
df_zero
```

- Visualisasi dengan menggunakan contoh Bar Plot()
  Kita mevisualisasikan dengan bar plot untuk melihat pembelian terbanyak

```
ax = sns.barplot(df_zero,estimator="sum", errorbar=None)
ax.bar_label(ax.containers[0], fontsize=10)
plt.xticks(rotation =90)
plt.show()
```

- support: menunjukkan persentase setiap pembelian produk

```
first = pd.DataFrame(df.sum() / df.shape[0],
                     columns = ["Support"]).sort_values("Support", ascending = False)
first
```

  Untuk pembelian produk diatas 15%

```
[93]  first[first['Support'] >= 0.15]
```

- Second Iteration: Temukan nilai dukungan untuk kombinasi produk berpasangan.

```
second = list(itertools.combinations(first.index, 2))
second = [list(i) for i in second]
# Sample of combinations
second
```

  Mencari nilai value dan mengeliminasi dari nilai support

```
value = []
for i in range(0, len(second)):
    temp = df.T.loc[second[i]].sum()
    temp = len(temp[temp == df.T.loc[second[i]].shape[0]]) / df.shape[0]
    value.append(temp)
# Create a data frame
secondIteration = pd.DataFrame(value, columns = ["Support"])
secondIteration["index"] = [tuple(i) for i in second]
secondIteration['length'] = secondIteration['index'].apply(lambda x:len(x))
secondIteration = secondIteration.set_index("index").sort_values("Support", ascending = False)
# Elimination by Support Value
secondIteration
```

```
df_second=pd.DataFrame()
df_second.insert(0 ,"Name Product" ,tuple(second))
df_second.insert(1 ,"Support" ,[0.20,0.20,0.20,0.20,0.20,0.20,0.20,
                                0.15,0.15,0.15,0.15,0.15,
                                0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,0.10,
                                0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,
                                0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,
                                0.00,0.00,0.00])
df_second
```

```
plt.figure(figsize =(15 ,5))
x =range(55)
y =df_second["Support"]
labels =[second]

plt.bar(x ,y)
plt.xticks(x ,labels[0] ,rotation =90)
plt.title("Product and Support")
plt.xlabel("Product")
plt.ylabel("Suppot Of Product")
plt.show()
```

Iterasi pertama

```
def ar_iterations(data, num_iter = 1, support_value = 0.1, iterationIndex = None):

    # Next Iterations
    def ar_calculation(iterationIndex = iterationIndex):
        # Calculation of support value
        value = []
        for i in range(0, len(iterationIndex)):
            result = data.T.loc[iterationIndex[i]].sum()
            result = len(result[result == data.T.loc[iterationIndex[i]].shape[0]]) / data.shape[0]
            value.append(result)
        # Bind results
        result = pd.DataFrame(value, columns = ["Support"])
        result["index"] = [tuple(i) for i in iterationIndex]
        result['length'] = result['index'].apply(lambda x:len(x))
        result = result.set_index("index").sort_values("Support", ascending = False)
        # Elimination by Support Value
        result = result[result.Support > support_value]
        return result
```

```
# First Iteration
first = pd.DataFrame(df.T.sum(axis = 1) / df.shape[0], columns = ["Support"]).sort_values("Support", ascending = False)
first = first[first.Support > support_value]
first["length"] = 1

if num_iter == 1:
    res = first.copy()

# Second Iteration
elif num_iter == 2:
    second = list(itertools.combinations(first.index, 2))
    second = [list(i) for i in second]
    res = ar_calculation(second)

# All Iterations > 2
else:
    nth = list(itertools.combinations(set(list(itertools.chain(*iterationIndex))), num_iter))
    nth = [list(i) for i in nth]
    res = ar_calculation(nth)
return res
```

Iterasi I

```
iteration1 = ar_iterations(df, num_iter=1, support_value=0.1)
iteration1
```

Iterasi II

```
iteration2 = ar_iterations(df, num_iter=2, support_value=0.1)
iteration2
```

Itersi III

```
iteration3 = ar_iterations(df, num_iter=3, support_value=0.01,
                           iterationIndex=iteration2.index)
iteration3
```

Iterasi IV

```
iteration4 = ar_iterations(df, num_iter=4, support_value=0.01,
                           iterationIndex=iteration3.index)
iteration4
```