PRAKTIKUM 4

Nama : Fayza Noval Fausta
NIM : G.231.22.0102
Kelas : TI SORE B
Mata Kuliah : Data Mining

```
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

df = pd.read_excel('http://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx')
df.head()
```

Penjelasan coding:

1. `import pandas as pd`: library untuk manipulasi dan analisis data dalam Python.
2. `from mlxtend.frequent_patterns import apriori`: Baris ini mengimport fungsi `apriori` dari submodul `frequent_patterns` dalam library MLxtend. Fungsi `apriori` digunakan untuk menghasilkan frequent item sets dari dataset.
3. `from mlxtend.frequent_patterns import association_rules`: Baris ini mengimport fungsi `association_rules` dari submodul `frequent_patterns` dalam library MLxtend. Fungsi `association_rules` digunakan untuk menghasilkan aturan asosiasi dari frequent item sets.
4. `df = pd.read_excel('http://archive.ics.uci.edu/ml/machine-learning databases/00352/Online%20Retail.xlsx')`: digunakan untuk membaca file Excel dari URL yang diberikan. Data ini berasal dari UCI Machine Learning Repository dan berisi informasi.
5. `df.head()`: menampilkan lima baris pertama dari DataFrame df, sehingga kita bisa melihat struktur dan contoh data.

```
[2] df['Description'] = df['Description'].str.strip()
    df.dropna(axis=0, subset=['InvoiceNo'], inplace=True)
    df['InvoiceNo'] = df['InvoiceNo'].astype('str')
    df = df[~df['InvoiceNo'].str.contains('C')]
```

Penjelasan coding:

1. `df['Description'] = df['Description'].str.strip()`: Baris ini menghilangkan spasi di awal dan akhir setiap nilai dalam kolom 'Description'. Hal ini dilakukan dengan menggunakan metode `strip()` dari objek string di dalam kolom 'Description'. Tujuannya adalah untuk membersihkan data dan menghapus karakter kosong yang tidak diinginkan.
2. `df.dropna(axis=0, subset=['InvoiceNo'], inplace=True)`: Baris ini menghapus baris dari DataFrame `df` di mana nilai dalam kolom 'InvoiceNo' hilang (NaN). Parameter `axis=0` menunjukkan bahwa yang dihapus adalah baris, dan `subset=['InvoiceNo']` menunjukkan bahwa hanya kolom 'InvoiceNo' yang akan diperiksa keberadaan nilai hilangnya.

`inplace=True` digunakan untuk mengubah DataFrame asli, tanpa perlu membuat salinan baru.

3. `df['InvoiceNo'] = df['InvoiceNo'].astype('str')`: Baris ini mengonversi nilai dalam kolom 'InvoiceNo' menjadi tipe data string. Ini berguna jika ada angka faktur yang seharusnya berupa string tetapi terbaca sebagai angka. Dengan mengubahnya menjadi string, kita memastikan bahwa data akan diproses dengan benar saat penggunaan selanjutnya.
4. `df = df[~df['InvoiceNo'].str.contains('C')]`: Baris ini memfilter DataFrame `df` sehingga hanya menyertakan transaksi yang bukan merupakan transaksi kredit (diberi tanda 'C' pada nomor faktur). Ini dilakukan dengan menggunakan metode `str.contains()` untuk mencari transaksi yang mengandung 'C' dalam nomor faktur, dan tanda `~` digunakan untuk mengambil kebalikan dari hasil pencarian tersebut, sehingga hanya transaksi yang tidak mengandung 'C' yang disertakan dalam DataFrame.

```
[3] basket = (df[df['Country'] =="France"]
              .groupby(['InvoiceNo', 'Description'])['Quantity']
              .sum().unstack().reset_index().fillna(0)
              .set_index('InvoiceNo'))
```

Penjelasan coding:

1. `df[df['Country'] =="France"]`: Baris ini melakukan filtering pada DataFrame `df`, sehingga hanya baris-baris yang memiliki nilai "France" pada kolom "Country" yang disertakan. Ini berarti hanya data transaksi dari Prancis yang akan diproses selanjutnya.
2. `.groupby(['InvoiceNo', 'Description'])['Quantity'].sum()`: Setelah filtering, data dikelompokkan berdasarkan nomor faktur (`InvoiceNo`) dan deskripsi produk (`Description`). Kemudian, jumlah produk yang dibeli dihitung untuk setiap kelompok menggunakan fungsi `sum()` pada kolom 'Quantity'. Ini akan menghasilkan total jumlah setiap produk yang dibeli dalam setiap transaksi.
3. `.unstack()`: Langkah ini mengubah struktur data dari hasil groupby sebelumnya menjadi bentuk yang lebih lebar, di mana setiap produk akan menjadi kolomnya sendiri. Ini akan membuat setiap transaksi direpresentasikan sebagai satu baris dengan setiap produk dijadikan kolom.
4. `.reset_index()`: Setelah `unstack()`, indeks DataFrame akan direset agar nomor faktur menjadi kolom biasa, bukan indeks. Ini diperlukan agar kita dapat menggunakan nomor faktur sebagai indeks di langkah selanjutnya.
5. `.fillna(0)`: Jika ada nilai yang hilang (NaN), langkah ini akan menggantinya dengan nilai 0. Ini dilakukan untuk memastikan bahwa kita memiliki nilai yang valid untuk setiap sel di DataFrame.
6. `.set_index('InvoiceNo')`: Terakhir, nomor faktur dijadikan indeks DataFrame, sehingga setiap transaksi memiliki nomor faktur sebagai identifikasi uniknya.

```python
def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1

basket_sets = basket.applymap(encode_units)
basket_sets.drop('POSTAGE', inplace=True, axis=1)
```

Penjelasan coding:

1. `def encode_units(x):`: Ini adalah definisi dari sebuah fungsi bernama `encode_units` yang menerima satu argumen `x`. Fungsi ini akan diterapkan pada setiap sel dalam DataFrame `basket_sets` pada langkah selanjutnya. Tujuan dari fungsi ini adalah untuk mengonversi nilai-nilai dalam DataFrame menjadi bentuk yang sesuai untuk analisis asosiasi.
2. ```python if x <= 0: return 0 if x >= 1: return 1 ```: Dalam fungsi `encode_units`, setiap nilai `x` yang kurang dari atau sama dengan 0 akan diubah menjadi 0, dan setiap nilai `x` yang lebih besar dari atau sama dengan 1 akan diubah menjadi 1. Ini berarti kita akan melakukan binarization data, di mana nilai-nilai yang semula mungkin beragam akan diubah menjadi 0 (tidak ada pembelian) atau 1 (pembelian terjadi).
3. `basket_sets = basket.applymap(encode_units)`: Di sini, fungsi `encode_units` diterapkan pada setiap sel dalam DataFrame `basket_sets` menggunakan metode `applymap()`. Ini berarti setiap nilai dalam DataFrame `basket_sets` akan diubah sesuai dengan aturan yang didefinisikan dalam fungsi `encode_units`. Hasilnya adalah DataFrame baru yang telah diubah dalam bentuk yang sesuai untuk analisis asosiasi.
4. `basket_sets.drop('POSTAGE', inplace=True, axis=1)`: Setelah transformasi, kita biasanya perlu melakukan pembersihan tambahan. Di sini, kita menghapus kolom 'POSTAGE' dari DataFrame untuk analisis asosiasi, karena itu adalah biaya pengiriman dan bukan produk yang dibeli. Parameter `inplace=True` digunakan untuk mengubah DataFrame asli tanpa perlu membuat salinan baru, dan `axis=1` digunakan untuk menunjukkan bahwa kita akan menghapus kolom, bukan baris.

```python
[5] frequent_itemsets = apriori(basket_sets, min_support=0.07, use_colnames=True)
```

Penjelasan coding:

1. `basket_sets`: Argumen pertama adalah DataFrame yang berisi data transaksi yang telah diproses sebelumnya, di mana setiap transaksi direpresentasikan sebagai satu baris dengan setiap produk dijadikan kolom dan jumlah produk yang dibeli dijadikan nilai.
2. `min_support=0.07`: Argumen ini menentukan nilai minimum support yang digunakan untuk menentukan frequent itemsets. Support adalah

proporsi dari transaksi yang mengandung suatu itemset. Dalam konteks ini, nilai 0.07 berarti itemset akan dianggap frequent jika muncul dalam setidaknya 7% dari transaksi. Nilai support yang dipilih akan bergantung pada kebutuhan analisis dan dapat disesuaikan sesuai keinginan.
3. `use_colnames=True`: Argumen ini menentukan apakah nama-nama kolom dalam DataFrame `basket_sets` harus digunakan sebagai label untuk item dalam frequent itemsets.

```
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules.head()
```

Penjelasan coding:

1. `frequent_itemsets`: Argumen pertama adalah frequent itemsets yang dihasilkan sebelumnya menggunakan fungsi `apriori`. Frequent itemsets adalah kumpulan itemset yang muncul dengan frekuensi yang memenuhi kriteria support minimum yang telah ditetapkan.
2. `metric="lift"`: Argumen ini menentukan metrik yang digunakan untuk menilai kualitas aturan asosiasi yang dihasilkan. Dalam hal ini, kita menggunakan metrik "lift". Lift adalah ukuran kekuatan asosiasi antara dua item. Nilai lift yang lebih besar dari 1 menunjukkan bahwa kemunculan dua item tersebut bersama-sama lebih sering daripada yang diharapkan secara acak, yang menunjukkan adanya hubungan yang signifikan antara keduanya.
3. `min_threshold=1`: Argumen ini menentukan nilai ambang minimum untuk metrik yang dipilih. Dalam hal ini, nilai ambang minimum yang dipilih adalah 1. Ini berarti aturan asosiasi yang dihasilkan harus memiliki nilai lift minimal 1. Aturan dengan lift lebih dari 1 menunjukkan adanya asosiasi yang signifikan antara item-item tersebut.

```
rules[ (rules['lift'] >= 6) &
       (rules['confidence'] >= 0.8) ]
```

Penjelasan coding:
1. `rules['lift'] >= 6`: Ini adalah bagian pertama dari filter, di mana kita memilih aturan asosiasi yang memiliki nilai lift setidaknya 6. Lift yang tinggi menunjukkan hubungan yang kuat antara item-item dalam aturan tersebut. Dengan menetapkan ambang 6, kita memastikan bahwa hanya aturan dengan hubungan yang sangat signifikan yang akan dipertimbangkan.
2. `rules['confidence'] >= 0.8`: Bagian kedua dari filter, di mana kita memilih aturan-asosiasi yang memiliki nilai confidence setidaknya 0.8. Confidence adalah ukuran seberapa sering aturan tersebut terbukti benar. Dengan menetapkan ambang 0.8, kita memastikan bahwa hanya aturan dengan tingkat kepercayaan yang tinggi yang akan dipertimbangkan.

3. `rules[...]`: Kode ini menggabungkan kedua filter di atas menggunakan operator `&` (AND), yang berarti aturan-asosiasi harus memenuhi kedua kriteria tersebut untuk disertakan dalam hasil filter.

```
basket['ALARM CLOCK BAKELIKE GREEN'].sum()
340.0
basket['ALARM CLOCK BAKELIKE RED'].sum()
316.0
```

Penjelasan coding:

1. `basket['ALARM CLOCK BAKELIKE GREEN'].sum()`: Kode ini menghitung jumlah pembelian untuk jam weker berwarna hijau dengan nama "ALARM CLOCK BAKELIKE GREEN" dalam DataFrame `basket`. Ini dilakukan dengan menggunakan metode `sum()` pada kolom yang sesuai dalam DataFrame `basket`. Hasilnya, yaitu 340.0, adalah total jumlah pembelian untuk jam weker berwarna hijau tersebut.
2. `basket['ALARM CLOCK BAKELIKE RED'].sum()`: Kode ini menghitung jumlah pembelian untuk jam weker berwarna merah dengan nama "ALARM CLOCK BAKELIKE RED" dalam DataFrame `basket`. Seperti sebelumnya, ini dilakukan dengan menggunakan metode `sum()` pada kolom yang sesuai dalam DataFrame `basket`. Hasilnya, yaitu 316.0, adalah total jumlah pembelian untuk jam weker berwarna merah tersebut.

```
basket2 = (df[df['Country'] =="Germany"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))

basket_sets2 = basket2.applymap(encode_units)
basket_sets2.drop('POSTAGE', inplace=True, axis=1)
frequent_itemsets2 = apriori(basket_sets2, min_support=0.05, use_colnames=True)
rules2 = association_rules(frequent_itemsets2, metric="lift", min_threshold=1)

rules2[ (rules2['lift'] >= 4) &
        (rules2['confidence'] >= 0.5)]
```

Penjelasan coding:

1. `basket2 = (df[df['Country'] =="Germany"]`: Kode ini melakukan filtering pada DataFrame `df` untuk memilih hanya baris-baris yang memiliki nilai "Germany" pada kolom "Country", sehingga hanya data transaksi dari Jerman yang akan diproses selanjutnya.
2. `.groupby(['InvoiceNo', 'Description'])['Quantity'].sum()`: Setelah filtering, data dikelompokkan berdasarkan nomor faktur (`InvoiceNo`) dan deskripsi produk (`Description`). Kemudian, jumlah produk yang dibeli dihitung untuk setiap kelompok menggunakan fungsi `sum()` pada kolom 'Quantity'. Ini akan menghasilkan total jumlah setiap produk yang dibeli dalam setiap transaksi.

3. `.unstack()`: Langkah ini mengubah struktur data dari hasil groupby sebelumnya menjadi bentuk yang lebih lebar, di mana setiap produk akan menjadi kolomnya sendiri. Ini akan membuat setiap transaksi direpresentasikan sebagai satu baris dengan setiap produk dijadikan kolom.
4. `.reset_index()`: Setelah `unstack()`, indeks DataFrame akan direset agar nomor faktur menjadi kolom biasa, bukan indeks. Ini diperlukan agar kita dapat menggunakan nomor faktur sebagai indeks di langkah selanjutnya.
5. `.fillna(0)`: Jika ada nilai yang hilang (NaN), langkah ini akan menggantinya dengan nilai 0. Ini dilakukan untuk memastikan bahwa kita memiliki nilai yang valid untuk setiap sel di DataFrame.
6. `.set_index('InvoiceNo'))`: Terakhir, nomor faktur dijadikan indeks DataFrame, sehingga setiap transaksi memiliki nomor faktur sebagai identifikasi uniknya.
7. `basket_sets2 = basket2.applymap(encode_units)`: Ini adalah langkah untuk menerapkan fungsi `encode_units` pada setiap sel dalam DataFrame `basket2`, yang bertujuan untuk mengonversi nilai-nilai dalam DataFrame menjadi bentuk yang sesuai untuk analisis asosiasi, sama seperti yang dilakukan sebelumnya.
8. `basket_sets2.drop('POSTAGE', inplace=True, axis=1)`: Langkah ini bertujuan untuk menghapus kolom 'POSTAGE' dari DataFrame `basket_sets2`, karena kolom 'POSTAGE' mungkin tidak relevan untuk analisis asosiasi, sama seperti yang telah dilakukan sebelumnya.
9. `frequent_itemsets2 = apriori(basket_sets2, min_support=0.05, use_colnames=True)`: Kode ini menghasilkan frequent itemsets dari DataFrame `basket_sets2` yang telah dihasilkan sebelumnya, dengan menggunakan nilai minimum support 0.05. Ini akan menghasilkan kumpulan itemset yang muncul dengan frekuensi yang memenuhi kriteria support minimum yang telah ditetapkan.
10. `rules2 = association_rules(frequent_itemsets2, metric="lift", min_threshold=1)`: Langkah ini menghasilkan aturan asosiasi dari frequent itemsets yang dihasilkan sebelumnya, dengan menggunakan metrik "lift" dan nilai ambang minimum 1. Ini akan menghasilkan aturan asosiasi yang memenuhi kriteria yang telah ditetapkan.
11. `rules2[ (rules2['lift'] >= 4) & (rules2['confidence'] >= 0.5)]`: Baris terakhir ini adalah langkah untuk memfilter aturan asosiasi yang dihasilkan berdasarkan kriteria tertentu, yaitu nilai lift yang lebih besar dari atau sama dengan 4 dan nilai confidence yang lebih besar dari atau sama dengan 0.5. Ini bertujuan untuk mempersempit jumlah aturan-asosiasi yang dihasilkan menjadi hanya yang paling relevan dan signifikan dalam analisis kita.