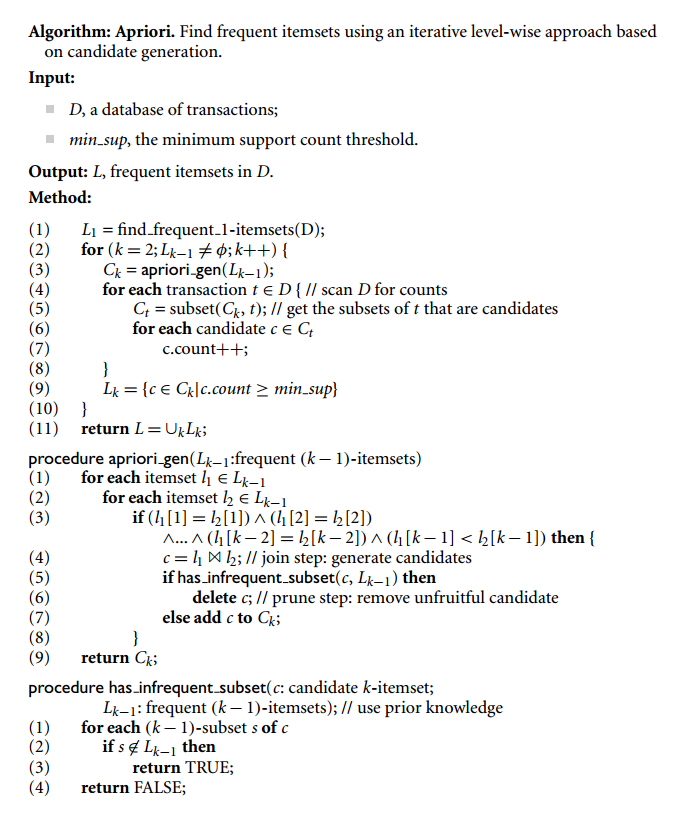
1. **Apriori**

Apriori adalah algoritma untuk mencari *frequent itemsets* dengan menghasilkan kandidat dengan restriksi tertentu. Apriori adalah suatu algoritma yang berpengaruh, yang diciptakan oleh R. Agrawal dan R. Srikant pada tahun 1994 untuk mencari *frequent itemsets* untuk *boolean association rules*. Nama dari algoritmanya terinspirasi dari fakta bahwa algoritmanya menggunakan pengetahuan sebelumnya (*prior knowledge*) dari properti *frequent itemset*.

* **Algoritma**

**Contoh**

Data Transaksi (**D**)

|  |  |
| --- | --- |
| **TID** | **Daftar Item ID** |
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

* **Langkah 1**

Pada putaran pertama dari algoritmanya, setiap item adalah bagian dari **candidate 1-itemsets**, **C**1. Algoritmanya dengan sederhana memindai semua transaksi dan menghitung jumlah kemunculan setiap item.

|  |  |
| --- | --- |
| **Item** | **Jumlah Support** |
| I1 | 6 |
| I2 | 7 |
| I3 | 6 |
| I4 | 2 |
| I5 | 2 |

* Langkah 2

Anggap saja **minimum support** yang diperlukan adalah **2**, yaitu **min\_sup** = 2. Di sini, kita merujuk kepada **minimum support absolut**, karena kita menggunakan **support count**. **Minimum support** **relatif** dari kasus demikian adalah **2/9 = 22%**. Set dari **frequent 1-itemsets**, **L1** dapat ditentukan. Ia terdiri dari **itemset** dari **C1** yang memenuhi **minimum support**. Sebagai contoh, semua **item** pada **C1** telah memenuhi **minimum support**.

|  |  |
| --- | --- |
| **Item** | **Jumlah Support** |
| I1 | 6 |
| I2 | 7 |
| I3 | 6 |
| I4 | 2 |
| I5 | 2 |

* Langkah 3

Untuk mencari tahu set dari **frequent 2-itemsets**, **L­2**, algoritmanya menggunakan **L1 join L1** untuk menghasilkan set dari **candidate 2-itemsets, C2**.

|  |
| --- |
| **Itemset** |
| I1, I2 |
| I1, I3 |
| I1, I4 |
| I1, I5 |
| I2, I3 |
| I2, I4 |
| I2, I5 |
| I3, I4 |
| I3, I5 |
| I4, I5 |

* Langkah 4

Selanjutnya, semua transaksi di **D** dipindai dan **support count** dari setiap **candidate itemset** di **C2** di akumulasikan.

|  |  |
| --- | --- |
| **Itemset** | **Jumlah Support** |
| I1, I2 | 4 |
| I1, I3 | 4 |
| I1, I4 | 1 |
| I1, I5 | 2 |
| I2, I3 | 4 |
| I2, I4 | 2 |
| I2, I5 | 2 |
| I3, I4 | 0 |
| I3, I5 | 1 |
| I4, I5 | 0 |

* Langkah 5

Set dari **frequent 2-itemsets**, **L2**, lalu ditentukan, yaitu terdiri dari **itemset** di **C2** yang memenuhi kriteria **minimum support**.

|  |  |
| --- | --- |
| **Itemset** | **Jumlah Support** |
| I1, I2 | 4 |
| I1, I3 | 4 |
| I1, I5 | 2 |
| I2, I3 | 4 |
| I2, I4 | 2 |
| I2, I5 | 2 |

* Langkah 6

Untuk menghasilkan **candidate 3-itemsets**, **C3**, kita akan menggunakan **kombinasi** dari set **L2**.

Contoh:

|  |  |
| --- | --- |
| **Itemset** | **Jumlah Support** |
| I1, I2 | 4 |
| I1, I3 | 4 |
| I1, I5 | 2 |
| I2, I3 | 4 |
| I2, I4 | 2 |
| I2, I5 | 2 |

Untuk menghasilkan set **C3**, maka **apriori generation method** akan diterapkan. Metode tersebut akan mencari **frequent itemsets** dengan item – item yang sama (dari yang **pertama** hingga yang ke **K-1**) untuk di kombinasikan. **K** adalah **jumlah item** di setiap **itemset** dari **frequent k-itemsets** yang sedang diproses, sehingga nilai **K** adalah 2.

Sebagai contoh, kita lihat itemset yang mempunyai **K-1 item pertama** yang sama, ada {I1, I2}, {I1, I3}, {I1, I5}, sehingga kita akan membuat **kombinasi** dari **itemset – itemset** tersebut. Hapus **K-1 item pertama**, karena **K** adalah 2, sehingga **K-1** = 1. Maka, untuk {I1, I2}, {I1, I3} {I1, I5}, kita menghapus **data pertama**, yaitu **I1** dan menghasilkan **kombinasi** dari {I2}, {I3}, {I5}. Hasil dari **kombinasi**nya adalah {I2, I3}, {I2, I5}, {I3, I5}. Lalu, tambahkan item yang baru saja dihapus {I1, I2, I3}, {I1, I3, I5} {I1, I2, I5}.

Lalu, lihat itemset – itemset selanjutnya dengan **K-1 item pertama** yang sama, ada {I2, I3}, {I2, I4}, {I2, I5}, sehingga kita akan membuat **kombinasi** dari **itemset – itemset** tersebut. Hapus **K-1 item pertama**, karena **K** adalah 2, sehingga **K-1** = 1. Jadi untuk {I2, I3}, {I2, I4}, {I2, I5}, kita menghapus **data pertama**, yaitu **I2** dan menghasilkan **kombinasi** dari {I3}, {I4}, {I5}. Hasil dari **kombinasi**nya adalah {I3, I4}, {I3, I5}, {I4, I5}. Lalu tambahkan item yang sebelumnya dihapus {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}.

|  |
| --- |
| **Itemset** |
| I1, I2, I3 |
| I1, I2, I5 |
| I1, I3, I5 |
| I2, I3, I4 |
| I2, I3, I5 |
| I2, I4, I5 |

* Langkah 7

Transaksi dalam **D** dipindai dan **support count** dari setiap **candidate itemset** di **C3** di akumulasikan.

|  |  |
| --- | --- |
| **Itemset** | **Jumlah Support** |
| I1, I2, I3 | 2 |
| I1, I2, I5 | 2 |
| I1, I3, I5 | 1 |
| I2, I3, I4 | 0 |
| I2, I3, I5 | 1 |
| I2, I4, I5 | 0 |

* Langkah 8

Set dari **frequent 3-itemsets**, **L3**, lalu ditentukan, terdiri dari **itemset – itemset** pada **C3** yang memenuhi **minimum support**.

|  |  |
| --- | --- |
| **Itemset** | **Jumlah Support** |
| I1, I2, I3 | 2 |
| I1, I2, I5 | 2 |

* Langkah 9

Hasilkan set dari **candidate 4-itemsets**, **C4** menggunakan **apriori generation method**. Nilai dari **K** adalah 3, karena **jumlah item** pada setiap **itemset** dari **frequent k-itemsets** sekarang adalah 3.

Lihat itemset dengan **K-1 item pertama** yang sama, ada {I1, I2, I3}, {I1, I2, I5}. Buatlah **kombinasi** dari **itemset – itemset** tersebut. Hapus **K-1 item pertama**, karena **K** adalah 3 maka **K-1** = 2. Maka, untuk {I1, I2, I3}, {I1, I2, I5}, kita menghapus **dua data pertama**, yaitu **{I1, I2}** dan menghasilkan **kombinasi** dari pasangannya {I3}, {I5}. Hasil dari **kombinasi** adalah {I3, I5}. Lalu, tambahkan item yang baru saja dihapus {I1, I2, I3, I5}.

|  |
| --- |
| **Itemset** |
| I1, I2, I3, I5 |

* Langkah 10

Transaksi pada **D** dipindai dan **support count** dari setiap **candidate itemset** pada **C4** diakumulasikan.

|  |  |
| --- | --- |
| **Itemset** | **Jumlah Support** |
| I1, I2, I3, I5 | 1 |

* Langkah 11

Set dari **frequent 4-itemsets**, **L4**, lalu ditentukan, yaitu terdiri dari **itemset – itemset** pada **C4** yang memenuhi **minimum support**.

|  |  |
| --- | --- |
| **Itemset** | **Jumlah Support** |
| Kosong | |

Karena tidak ada **itemset** dari **C4** yang memenuhi **minimum support**, maka **itemset** pada **L4** kosong, dan iterasi selesai.

* Langkah 12

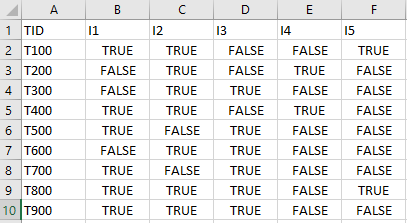
**Hasil frequent itemsets**.

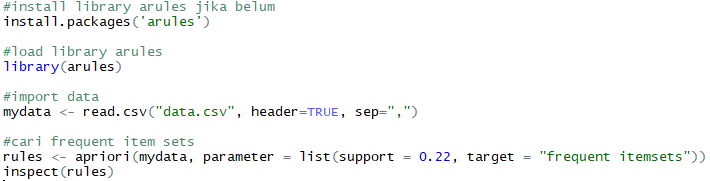
|  |  |  |
| --- | --- | --- |
| **Itemset** | **Jumlah Support** | **Support** |
| I1 | 6 | 6/9 = 0.667 |
| I2 | 7 | 7/9 = 0.778 |
| I3 | 6 | 6/9 = 0.667 |
| I4 | 2 | 2/9 = 0.222 |
| I5 | 2 | 2/9 = 0.222 |
| I1, I2 | 4 | 4/9 = 0.444 |
| I1, I3 | 4 | 4/9 = 0.444 |
| I1, I5 | 2 | 2/9 = 0.222 |
| I2, I3 | 4 | 4/9 = 0.444 |
| I2, I4 | 2 | 2/9 = 0.222 |
| I2, I5 | 2 | 2/9 = 0.222 |
| I1, I2, I3 | 2 | 2/9 = 0.222 |
| I1, I2, I5 | 2 | 2/9 = 0.222 |

* **Implementasi R**

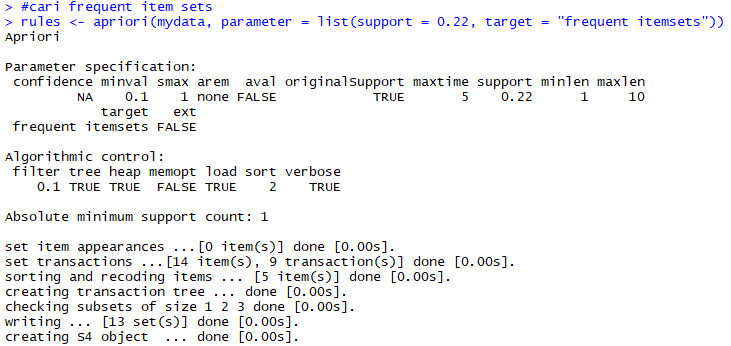
Di dalam R, kita juga bisa melakukan apriori untuk mencari *frequent itemsets*. Berikut adalah cara penggunaannya.

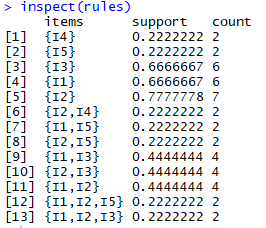
Pertama – tama, siapkan data yang akan dianalisis dalam bentuk csv. Berikut adalah contoh data dari **data.csv**:





Jika dijalankan, maka akan dihasilkan seperti berikut.

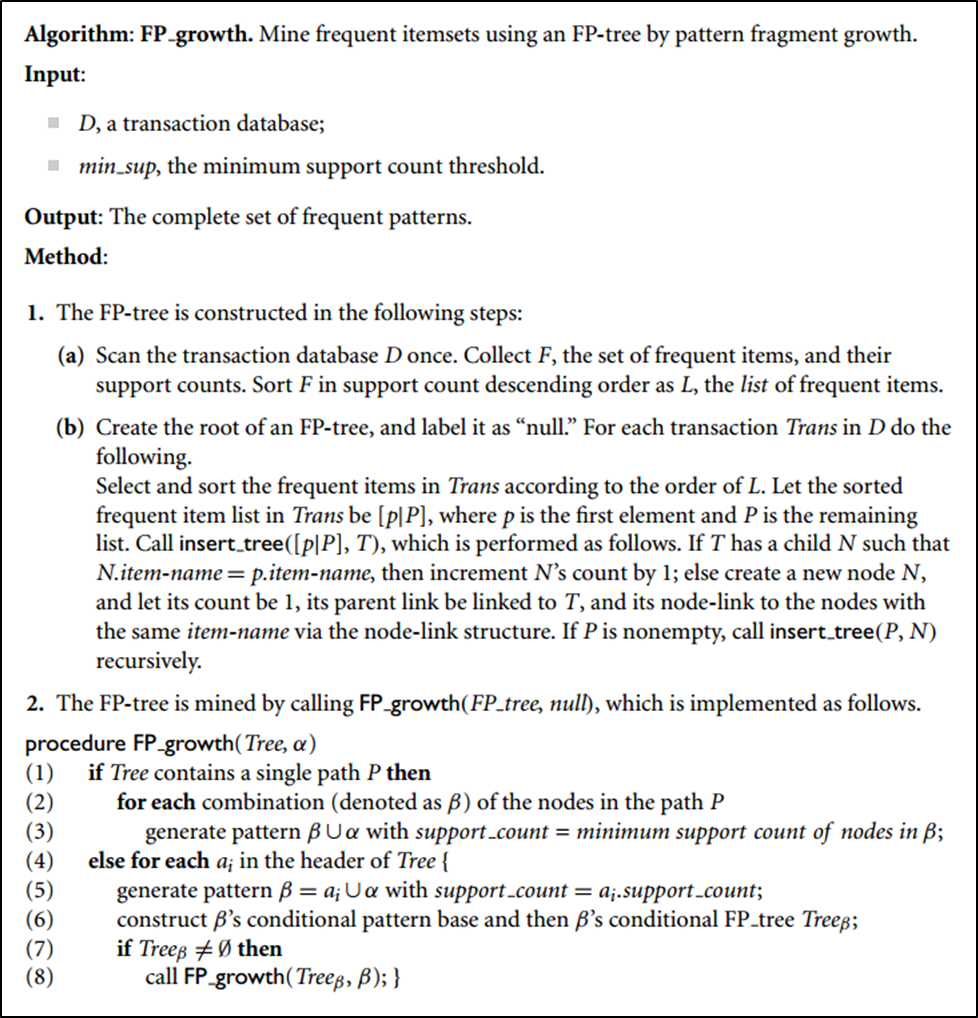




1. **FP Growth**

FP-Growth adalah algoritma untuk mencari *frequent itemsets* tanpa menghasilkan kandidat. Pada FP-Growth, *frequent itemsets* dihasilkan dari sebuah **FP-Tree** yang dikonstruksi berdasarkan data training. **FP-Tree** akan dibangun dari *root* yang ditandai *null*, lalu diikuti dengan item yang paling sering muncul berdasarkan *support count*, dan lalu bercabang dan diikuti lagi dengan item lain yang paling sering muncul bersama item tersebut.

* **Algoritma:**

****

**Contoh**

Data Transaksi

|  |  |
| --- | --- |
| **TID** | **Daftar Item ID** |
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

* Langkah 1

Pertama – tama, setiap item adalah bagian dari set **candidate 1-itemsets**, **C1**. Algoritmanya adalah dengan memindai semua transaksi untuk menghitung jumlah kemunculan tiap item.

|  |  |
| --- | --- |
| **Item** | **Support Count** |
| I1 | 6 |
| I2 | 7 |
| I3 | 6 |
| I4 | 2 |
| I5 | 2 |

* Langkah 2

Anggap saja **minimum support** yang diperlukan adalah **2**, yaitu **min\_sup** = 2. Di sini, kita merujuk kepada **minimum support absolut**, karena kita menggunakan **support count**. **Minimum support** **relatif** dari kasus demikian adalah **2/9 = 22%**. Set dari **frequent 1-itemsets**, **L1** dapat ditentukan. Ia terdiri dari **itemset** dari **C1** yang memenuhi **minimum support**. Sebagai contoh, semua **item** pada **C1** telah memenuhi **minimum support**.

|  |  |
| --- | --- |
| **Item** | **Support Count** |
| I1 | 6 |
| I2 | 7 |
| I3 | 6 |
| I4 | 2 |
| I5 | 2 |

* Langkah 3

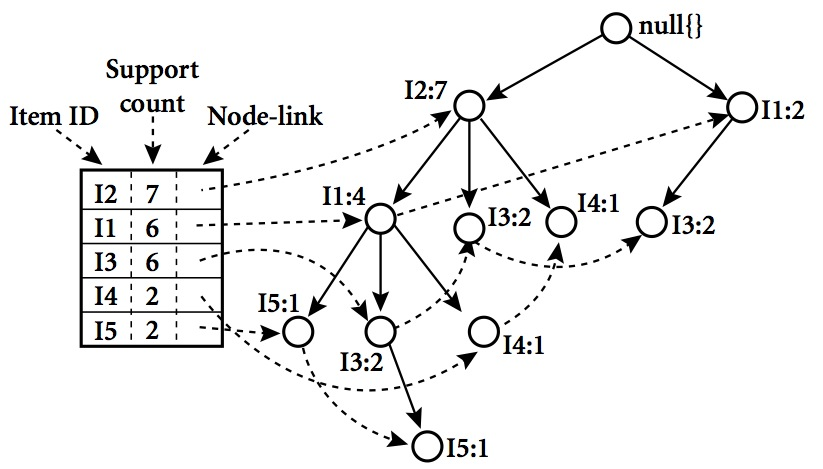
Urutkan set **L1** dengan **descending** berdasarkan **Support Count**.

|  |  |
| --- | --- |
| **Item** | **Jumlah Support** |
| I2 | 7 |
| I1 | 6 |
| I3 | 6 |
| I4 | 2 |
| I5 | 2 |

* Langkah 4

Hasilkan FP-Tree. Sebuah FP-Tree akan dibuat berdasarkan langkah – langkah berikut:

1. Buat *root* dari tree, bertandakan “*null*”.
2. Pindai setiap transaksi di dataset **D** dengan cara berikut:
   1. Item – item dalam setiap transaksi di **sort descending** berdasarkan **support count**.
   2. Tambahkan setiap item yang sudah di sort ke FP-Tree dengan aturan berikut:
      1. Jika node dari item yang dimasukkan belum dibuat, makan tambahkan node baru dari item yang dimasukkan dengan jumlah count = 1.
      2. Jika node dari item yang dimasukkan sudah pernah dibuat, maka tambahkan jumlah count dari nodenya dengan satu.



Contoh:

Pindai transaksi **T100**: {I1, I2, I5}

1. Urutkan item berdasarkan set berurut **L1**. Item – item **T100** setelah disort: {I2, I1, I5}
2. Tambahkan item yang sudah disort ke dalam FP-Tree menurut aturan di atas {*null ->* I2 -> I1 -> I5}.

Pindai transaksi **T200**: {I2, I4}

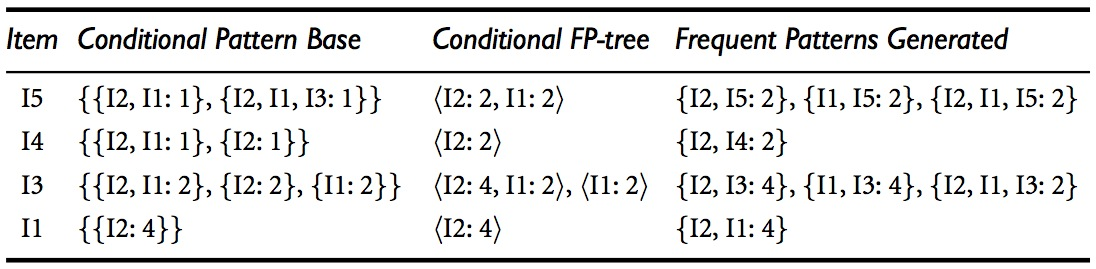
1. Urutkan item berdasarkan set berurut **L1**. Item – item **T200** setelah disort: {I2, I4}
2. Tambahkan item yang sudah disort ke dalam FP-Tree menurut aturan di atas {*null* -> I2 -> I4}.

Pindai transaksi **T300**: {I2, I3}

1. Urutkan item berdasarkan set berurut **L1**. Item – item **T300** setelah disort: {I2, I3}
2. Tambahkan item yang sudah disort ke dalam FP-Tree menurut aturan di atas {*null* -> I2 -> I3}.

* Langkah 5

Lakukan FP-Tree Mining.



Catatan:

FP-Tree Mining dimulai dari item di dalam **L1** dari **support count** yang paling kecil ke yang paling besar.

**Conditional Pattern Base** adalah **set** dari **jalur** yang **berakhir** dengan **item yang dipilih**, diikuti dengan **jumlah** berapa kali jalurnya diakhiri dengan item tersebut.

Item: **I5**

Ada 2 jalur yang diakhiri dengan I5: {I2, I1} dan {I2, I1, I3}

Jalur {I2, I1} hanya diakhiri dengan I5 sekali, maka akan ditulis seperti {I2, I1:1}

Jalur {I2, I1, I3} hanya diakhiri dengan I5 sekali, maka akan ditulis seperti {I2, I1, I3:1}

Item: **I4**

Ada 2 jalur yang diakhiri dengan I4: {I2, I1} dan {I2}

Jalur {I2, I1} hanya diakhiri dengan I4 sekali, maka akan ditulis seperti {I2, I1:1}

Jalur {I2} hanya diakhiri dengan I4 sekali, maka akan ditulis seperti {I2:1}

Item: **I3**

Ada 3 jalur yang diakhiri dengan I3: {I2, I1}, {I2} and {I1}

Jalur {I2, I1} diakhiri dengan I3 dua kali, sehingga akan ditulis seperti {I2, I1:2}

Jalur {I2} diakhiri dengan I3 dua kali, sehingga akan ditulis seperti {I2:2}

Jalur {I1} diakhiri dengan I3 dua kali, sehingga akan ditulis seperti {I1:2}

Item: **I1**

Ada 1 jalur yang diakhiri dengan I1: {I2}

Jalur {I2} diakhiri I1 sebanyak empat kali, sehingga akan ditulis seperti {I2:4}

Item: **I2**

Tidak ada jalur yang diakhiri dengan I2.

**Conditional FP-Tree** adalah **sekumpulan jalur** dari **Conditional Pattern Base** dengan **support count** lebih dari atau sama dengan **minimum support**.

Item: **I5**

Conditional Pattern Base: {{I2, I1:1}, {I2, I1, I3:1}}

Terdapat 1 jalur di mana setiap titik dari jalurnya memiliki support count lebih dari atau sama dengan minimum support. Jalurnya adalah {I2:2, I1:2}. I3 tidak termasuk, karena I3 hanya dilewati satu kali yang berarti lebih kecil daripada minimum support.

Item: **I4**

Conditional Pattern Base: {{I2, I1:1}, {I2:1}}

Terdapat 1 jalur di mana setiap titik dari jalurnya memiliki support count lebih dari atau sama dengan minimum support. Jalurnya adalah {I2:2}. I1 tidak termasuk, karena I1 hanya dilewati satu kali, yang berarti lebih kecil daripada minimum support.

Item: **I3**

Conditional Pattern Base: {{I2, I1:2}, {I2:2}, {I1:2}}

Terdapat 2 jalur di mana setiap titik dari jalurnya memiliki support count lebih dari atau sama dengan minimum support. Jalur – jalurnya adalah {I2:4, I1:2}, {I1:2}. I1 tidak dapat bergabung, karena I1 yang kedua berada di cabang yang berbeda.

Item: **I1**

Conditional Pattern Base: {I2:4}

Terdapat 1 jalur di mana setiap titik dari jalurnya memiliki support count lebih dari atau sama dengan minimum support. Jalurnya adalah {I2:4}.

**Frequent Pattern Generated** adalah **set** dari **frequent itemsets** yang dihasilkan dari FP-Tree. **Frequent itemsets** dihasilkan dari **kombinasi** dari item – item pada jalur di dalam **Conditional FP-Tree** dengan **item yang dipilih** diikuti dengan **jumlah** berapa kali kombinasi tersebut dilewati.

Item: **I5**

Conditional FP-Tree: {I2:2, I1:2}

Buatlah kombinasi dari Conditional FP-Tree. Kombinasinya adalah {I2}, {I1}, {I2, I1}. Akhiri kombinasi dengan item transaksi yang dipilih, {I2, I5:2}, {I1, I5:2}, {I2, I1, I5:2}.

Item: **I4**

Conditional FP-Tree: {I2:2}

Buatlah kombinasi dari Conditional FP-Tree. Kombinasinya adalah {I2}. Akhiri kombinasi dengan item transaksi yang dipilih, {I2, I4:2}.

Item: **I3**

Conditional FP-Tree: {I2:4, I1:2}, {I1:2}

Buatlah kombinasi dari Conditional FP-Tree. Kombinasinya adalah {I2}, {I1}, {I2, I1}. Akhiri kombinasi dengan item transaksi yang dipilih, {I2, I3:4}, {I1, I3:4}, {I2, I1, I3:2}.

Item: **I1**

Conditional FP-Tree: {I2:4}

Buatlah kombinasi dari Conditional FP-Tree. Kombinasinya adalah {I2}. Akhiri kombinasi dengan item transaksi yang dipilih, {I2, I1:4}.

* Langkah 6

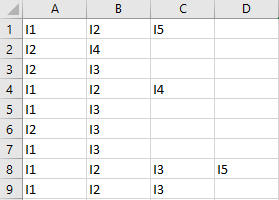
**Frequent Itemsets** result.

|  |  |  |
| --- | --- | --- |
| **Itemset** | **Support Count** | **Support** |
| I1 | 6 | 6/9 = 0.667 |
| I2 | 7 | 7/9 = 0.778 |
| I3 | 6 | 6/9 = 0.667 |
| I4 | 2 | 2/9 = 0.222 |
| I5 | 2 | 2/9 = 0.222 |
| I2, I1 | 4 | 4/9 = 0.444 |
| I2, I3 | 4 | 4/9 = 0.444 |
| I1, I3 | 4 | 4/9 = 0.444 |
| I2, I4 | 2 | 2/9 = 0.222 |
| I2, I5 | 2 | 2/9 = 0.222 |
| I1, I5 | 2 | 2/9 = 0.222 |
| I2, I1, I3 | 2 | 2/9 = 0.222 |
| I2, I1, I5 | 2 | 2/9 = 0.222 |

* **Implementasi di R**

Untuk mengimplementasikan FP Growth di R, kita dapat menggunakan program C yang telah dibuat oleh *Christian Borgelt* dan memanggilnya melalui R. Untuk mendapatkan programnya, dapat diakses site <http://www.borgelt.net//fpgrowth.html>. Kali ini, kita akan menggunakan fpgrowth.exe yang dapat diunduh dari website tersebut.

Pertama – tama, pastikan fpgrowth.exe diletakkan pada workspace R, di folder yang sama dengan script yang sedang digunakan. Siapkan **data2.csv** yang memiliki data dengan format berikut,

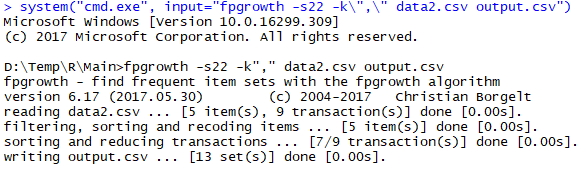


Lalu, kita dapat memanggilnya melalui R dengan cara berikut.

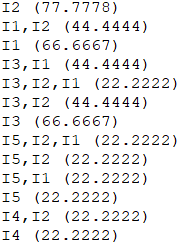


Pada contoh di atas, kita hanya memanggil *command prompt* dari Windows untuk menjalankan fpgrowth.exe dengan parameter yang diberi. **-s22** berarti **minimum support 22%**. **-k\”,\”** berarti **item separator** nya adalah lambang koma. **Data2.csv** merupakan **input**, dan **output.csv** merupakan output file untuk hasil dari frequent pattern.

Ketika dijalankan, maka akan muncul:

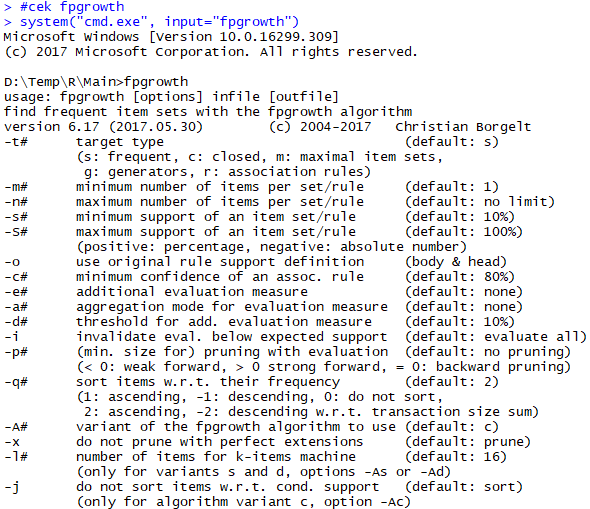


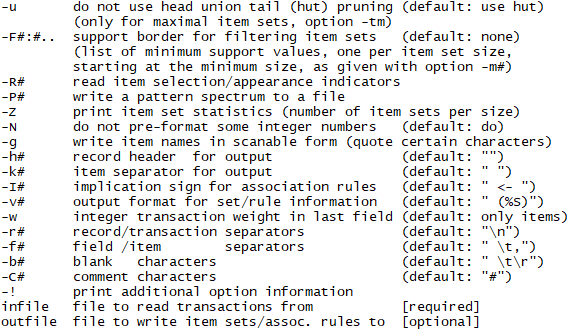
Setelah itu, **output.csv** dapat ditemukan di direktori workspace, dengan data sebagai berikut.



Hasilnya telah di sorting berdasarkan **support** terbesar, dan berbentuk persentase. Untuk melakukan modifikasi pada prosesnya, kita dapat mengubah parameternya sesuai dengan keinginan berdasarkan aturan yang telah ditentukan. Untuk melihat aturan tersebut, kita dapat memanggil:



Dengan memanggil fungsi tersebut, maka akan muncul daftar aturan parameter sebagai berikut.



1. **Association Rules**

* Langkah 1

Tentukan **minimum confidence** untuk association rule. Anggap saja **minimum confidence** adalah **0.8**.

* Langkah 2

Hasilkan rule dari setiap frequent itemset dan kalkulasikan nilai confidence untuk setiap rule yang dihasilkan.

Association rule akan dihasilkan untuk **frequent itemset** dengan 2 atau lebih item. Langkah – langkah untuk membuat association rule adalah:

1. Pisahkan itemset – itemset menjadi dua bagian (premis dan konklusi)

|  |  |  |
| --- | --- | --- |
| **Itemset** | **Premis** | **Konklusi** |
| I2, I1 | I2 | I1 |
| I2, I3 | I2 | I3 |
| I1, I3 | I1 | I3 |
| I2, I4 | I2 | I4 |
| I2, I5 | I2 | I5 |
| I1, I5 | I1 | I5 |
| I2, I1, I3 | I2 | I1, I3 |
| I1 | I2, I3 |
| I3 | I2, I1 |
| I2, I1, I5 | I2 | I1, I5 |
| I1 | I2, I5 |
| I5 | I2, I1 |

1. Rule akan dihasilkan sebagai hasil dari langkah pertama, lalu putar balikkan hasil dari step pertama.

|  |  |  |
| --- | --- | --- |
| **Itemset** | **Premis** | **Konklusi** |
| I2, I1 | I2 | I1 |
| I1 | I2 |
| I2, I3 | I2 | I3 |
| I3 | I2 |
| I1, I3 | I1 | I3 |
| I3 | I1 |
| I2, I4 | I2 | I4 |
| I4 | I2 |
| I2, I5 | I2 | I5 |
| I5 | I2 |
| I1, I5 | I1 | I5 |
| I5 | I1 |
| I2, I1, I3 | I2 | I1, I3 |
| I1, I3 | I2 |
| I1 | I2, I3 |
| I2, I3 | I1 |
| I3 | I2, I1 |
| I2, I1 | I3 |
| I2, I1, I5 | I2 | I1, I5 |
| I1, I5 | I2 |
| I1 | I2, I5 |
| I2, I5 | I1 |
| I5 | I2, I1 |
| I2, I1 | I5 |

1. Kalkulasikan nilai **support** dan **confidence** untuk setiap rule.

|  |  |  |  |
| --- | --- | --- | --- |
| **Premis** | **Konklusi** | **Support** | **Confidence** |
| I2 | I1 | 4/9 = 0.444 | 4/7 = 0.571 |
| I1 | I2 | 4/9 = 0.444 | 4/6 = 0.667 |
| I2 | I3 | 4/9 = 0.444 | 4/7 = 0.571 |
| I3 | I2 | 4/9 = 0.444 | 4/6 = 0.667 |
| I1 | I3 | 4/9 = 0.444 | 4/6 = 0.667 |
| I3 | I1 | 4/9 = 0.444 | 4/6 = 0.667 |
| I2 | I4 | 2/9 = 0.222 | 2/7 = 0.286 |
| I4 | I2 | 2/9 = 0.222 | 2/2 = 1 |
| I2 | I5 | 2/9 = 0.222 | 2/7 = 0.286 |
| I5 | I2 | 2/9 = 0.222 | 2/2 = 1 |
| I1 | I5 | 2/9 = 0.222 | 2/6 = 0.333 |
| I5 | I1 | 2/9 = 0.222 | 2/2 = 1 |
| I2 | I1, I3 | 2/9 = 0.222 | 2/7 = 0.286 |
| I1, I3 | I2 | 2/9 = 0.222 | 2/4 = 0.5 |
| I1 | I2, I3 | 2/9 = 0.222 | 2/6 = 0.333 |
| I2, I3 | I1 | 2/9 = 0.222 | 2/4 = 0.5 |
| I3 | I2, I1 | 2/9 = 0.222 | 2/6 = 0.333 |
| I2, I1 | I3 | 2/9 = 0.222 | 2/4 = 0.5 |
| I2 | I1, I5 | 2/9 = 0.222 | 2/7 = 0.286 |
| I1, I5 | I2 | 2/9 = 0.222 | 2/2 = 1 |
| I1 | I2, I5 | 2/9 = 0.222 | 2/6 = 0.333 |
| I2, I5 | I1 | 2/9 = 0.222 | 2/2 = 1 |
| I5 | I2, I1 | 2/9 = 0.222 | 2/2 = 1 |
| I2, I1 | I5 | 2/9 = 0.222 | 2/4 = 0.5 |

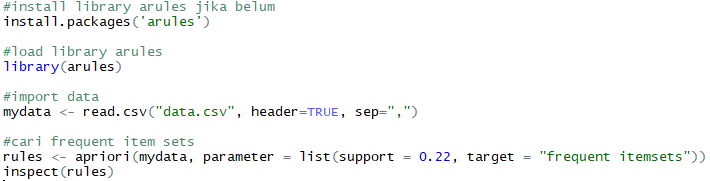
* Langkah 3

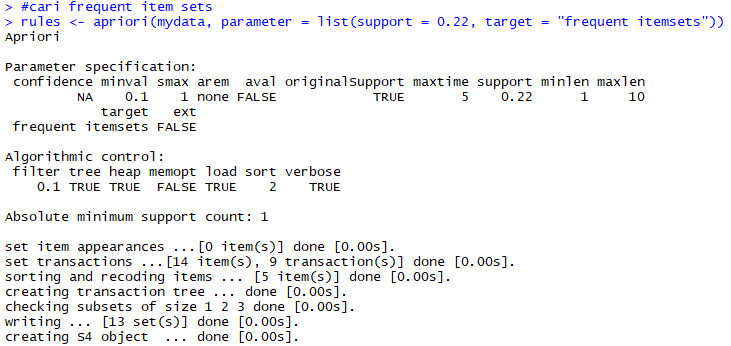
**Hapus** rule yang tidak memenuhi **minimum confidence**.

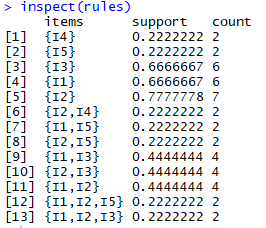
|  |  |  |  |
| --- | --- | --- | --- |
| **Premis** | **Konklusi** | **Support** | **Confidence** |
| I4 | I2 | 2/9 = 0.222 | 2/2 = 1 |
| I5 | I2 | 2/9 = 0.222 | 2/2 = 1 |
| I5 | I1 | 2/9 = 0.222 | 2/2 = 1 |
| I2, I5 | I1 | 2/9 = 0.222 | 2/2 = 1 |
| I1, I5 | I2 | 2/9 = 0.222 | 2/2 = 1 |
| I5 | I1, I2 | 2/9 = 0.222 | 2/2 = 1 |

**Implementasi di R**

Pembuatan *association rule* juga dapat dilakukan di R. Berikut adalah cara membuat association rule sebagai hasil dari pemrosesan *apriori*. Simak kembali contoh *apriori* sebelumnya.



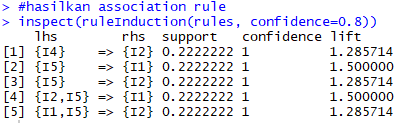




Setelah mendapatkan rules dari *apriori*, kita tinggal memanggil function **ruleInduction** untuk membuat *association rules*.



Dengan demikian, akan dihasilkan *association rules* sebagai berikut.



Perlu dicatat bahwa **arules** hanya menghasilkan satu item **Conclusion** di **RHS** (*Right Hand Side*). Oleh karena itu, {I5} => {I1, I2} tidak dihasilkan.