BAB III

METODOLOGI DAN PERANCANGAN

Pada bab ini berisi pembahasan mengenai metode dan langkah-langkah perancangan yang dilakukan dalam penelitian tentang*Association rule* dengan Algoritma *Frequent Pattern Growth (FP-Growth)* pada Data Spesifikasi *Smartphone.* Langkah-langkah yang dilakukan dalam penelitian ini dijelaskan pada Gambar 3.1 sebagai berikut:



Gambar 3.1 Langkah-langkah penelitian

Berdasarkan Gambar 3.1 tersebut, langkah-langkah yang dilakukan dalam penelitian ini adalah sebagai berikut :

1. Melakukan studi literatur yang berhubungan dengan spesifikasi *smartphone* dan metode yang digunakan yaitu *association rule* dengan algoritma *Frequent Pattern Growth (FP-Growth)*.
2. Mengumpulkan data-data yang diperlukan dalam penelitian ini yaitu data-data spesifikasi *smartphone*.
3. Melakukan analisa dan perancangan sistem.
4. Implementasi rancangan sistem yang telah dibuat pada tahap sebelumnya menjadi perangkat lunak.
5. Melakukan uji coba terhadap perangkat lunak yang telah dibuat.
6. Melakukan evaluasi hasil uji yang dilakukan oleh sistem.
7. **Pengumpulan Data**

Data yang digunakan dalam penelitian ini adalah data hasil *survey* mengenai spesifikasi-spesifikasi pada *smartphone* yang memiliki sistem operasi Android dan Windows Phone yang dimiliki oleh pengguna. *Survey* dilakukan pada tanggal 1 mei sampai dengan 14 mei 2014. *Dataset* yang diperoleh terdiri dari beberapa atribut, antara lain: memori internal, RAM, Core CPU, Clock speed CPU, dan GPU.

1. **Analisis dan Perancangan Sistem**

**3.2.1 Deskripsi Umum Sistem**

Secara umum sistem yang dibuat adalah sistem yang mengimplementasikan *association rule* dengan algoritma *Frequent Pattern Growth (FP-Growth)* untuk menganalisa terhadap data spesifikasi *smartphone* untuk mengetahui pola hubungan antar spesifikasi *smartphone* yang digunakan oleh pengguna saat ini. Metode *association rule* dalam sistem ini digunakan untuk mencari pola hubungan antar-*item (rule)* dengan menentukan nilai *support* dan *confidence* dalam proses pencarian pola hubungannya*.* Sedangkan algoritma *Frequent Pattern Growth (FP-Growth)* digunakan untuk mencari *frequent itemset* sebelum itemset tersebut dibangkitkan menjadi *rule*.

Dalam sistem yang dibuat ini parameter yang digunakan untuk melakukan analisa adalah nilai dari *minimum support*, dan *minimum confidence*. Dengan menggunakan algoritma *FP-Growth* sistem akan mencari *frequent itemset* terlebih dahulu. Dimana *itemset* berupa himpunan data spesifikasi *smartphone* yaitu kombinasi antar spesifikasi sehingga *frequent itemset*-nya berupa kombinasi antar spesifikasi *smartphone* yang mempunyai frekuensi kemunculan yang banyak. Frekuensi tersebut sebelumnya didapatkan dari pemindaian dalam *database* dan yang telah memenuhi nilai dari *minimum support*.

Setelah didapatkan *frequent itemset*, selanjutnya sistem akan melakukan proses *generate rule*. Dalam proses *generate rule* tersebut dilakukan perhitungan nilai *confidence* pada setiap *frequent itemset.* Setelah didapatkan nilai *confidence*, sistem akan melakukan pembangkitan *frequent itemset* menjadi *rule*. Dimana *frequent itemset* yang akan dibangkitkan menjadi *rule* adalah *frequent itemset* yang mempunyai nilai *confidence* yang memenuhi nilai *minimum confidence* yang telah dimasukkan dalam sistem.

Setelah melalui proses tersebut maka akan terbentuk *rule-rule*, kemudian selanjutnya sistem akan menguji kekuatan *rule* yang telah terbentuk dengan melakukan perhitungan *lift ratio.* Hasil *rule* yang telah terbentuk beserta hasil *lift ratio* nya akan ditampilkan sebagai hasil keluaran oleh sistem.

**3.2.2 Rancangan Pembuatan Sistem**

Dalam perancangan sistem pada penelitian ini akan mengimplementasikan metode serta algoritma yang digunakan. Proses implementasi yang akan dilakukan secara umum digambarkan menggunakan *flowchart* pada Gambar 3.2.



Gambar 3.2 *Flowchart* rancangan umum sistem

Secara umum, *input* yang dibutuhkan sistem adalah *minimum support, minimum confidence*, dan data spesifikasi *smartphone* berdasarkan masing-masing sistem operasi. Proses awal dalam sistem yaitu proses *association rule* dengan algoritma *FP-Growth*. Selanjutnya hasil *rule* yang didapatkan dilakukan proses perhitungan *lift ratio* sehingga dihasilkan keluaran berupa *rule* beserta nilai *lift ratio* nya.

Selanjutnya setiap proses dalam sistem dijabarkan sesuai yang tertera dalam *flowchart* pada Gambar 3.3 sampai dengan Gambar 3.9.



Gambar 3.3 *Flowchart* proses *association rule* dengan algoritma *FP-Growth*

Pada Gambar 3.3 dijelaskan mengenai proses *association rule* dengan menggunakan algoritma *frequent pattern growth (FP-Growth)* dimanayang pertama kali dilakukan oleh sistem ialah mencari *frequent 1-itemset*. Lalu dilakukan seleksi data spesifikasi *smartphone*, kemudian hasil seleksi digunakan sebagai *input* dalam pembangunan *tree*. Hasil pembangunan *tree* tersebut dimasukkan ke dalam proses *FP-Growth* untuk mendapatkan *frequent itemset*. Hasil *frequent itemset* tersebut lalu dilakukan proses *generate rule*. Dalam proses *generate rule,* *frequent itemset* yang memenuhi *minimum confidence* akan dibangkitkan menjadi *rule*.



Gambar 3.4 *Flowchart* proses *frequent 1-itemset*

Pada Gambar 3.4 dijelaskan mengenai proses f*requent 1-itemset,* dimana *frequent 1-itemset* didapatkan dari *item-item* yang memiliki nilai *support item* yang memenuhi nilai *minimum support.*



Gambar 3.5 *Flowchart* proses seleksi data

Pada Gambar 3.5 dijelaskan mengenai proses seleksi data dimana setelah didapatkan *frequent 1-itemset* kemudian dilakukan seleksi data. Data spesifikasi yang memenuhi *frequent 1-itemset* saja yang akan diambil.



Gambar 3.6 *Flowchart* proses pembangunan *tree*

Pada Gambar 3.6 dijelaskan mengenai proses pembangunan *tree* dimana setiap data hasil seleksi akan dimasukkan ke proses *AddTransactionTree* untuk mendapatkan data lintasan *FP-tree.*



Gambar 3.7 *Flowchart* proses *AddTransactionTree*

Pada proses *AddTransactionTree* dibuat *root* terlebih dahulu dan diberi label *null*. Lalu dilakukan pembuatan node untuk data spesifikasi dimana jika pada *tree* belum pernah ada node dengan label data tersebut maka dibuat node baru dan diberi nilai *count* 1, namun jika sebelumnya pada *tree* tersebut sudah ada node dengan label yang sama maka tidak membuat node baru melainkan hanya menambah 1 nilai *count* nya.



Gambar 3.8 *Flowchart* proses *FP-Growth*

Pada Gambar 3.8 dijelaskan mengenai proses algoritma *FP-Growth* dimana data lintasan hasil dari pembangunan *FP-Tree* akan diolah melalui dua proses yaitu proses *conditional pattern base* dan *conditional FP-Tree* untuk mendapatkan hasil *frequent itemset*.



Gambar 3.9 *Flowchart* proses *conditional pattern base*

Pada proses *conditional pattern base* ini akan dicari data *prefix* dan data *suffix* dari lintasan *FP-Tree* yang telah terbentuk melalui proses pembangunan *FP-Tree.* Hasil keluaran proses *conditional pattern base* ini adalah data-data *prefix* tiap *suffix.*



Gambar 3.10 *Flowchart* proses *conditional FP-Tree*

Pada proses *conditional FP-Tree* ini akan dicari frekuensi dan nilai *support* tiap item *prefix* dan *suffix* yang mempunyai lintasan yang sama. Dan jika nilai *support* dari kombinasi item *prefix* dan *suffix* tersebut memenuhi *minimum support* maka akan dijadikan *frequent itemset*. Keluaran dari proses ini adalah data *frequent itemset*.



Gambar 3.11 *Flowchart* proses *generate rule*

Pada Gambar 3.11 dijelaskan mengenai proses *generate rule,* dimana yang pertama dilakukan adalah mencari nilai *confidence.* Nilai *confidence* didapatkan dari nilai *support* (A,B) dibagi nilai *support* A. Kemudian apabila nilai *confidence* yang dimiliki oleh *frequent itemset* tersebutmemenuhi *minimum confidence* maka akan dibangkitkan menjadi *rule*. Hasil keluaran proses ini adalah *rule* beserta nilai *confidence* nya.



Gambar 3.12 *Flowchart* proses perhitungan *Lift Ratio*

Setelah melalui proses *generate rule* maka hasil *rule* akan diperhitungkan nilai *lift ratio*-nya. Nilai *lift ratio* dari masing-masing *rule* didapatkan dari perbandingan nilai *confidence* dengan nilai benchmark *confidence*. Jika nilai *lift ratio* dari *rule* lebih dari sama dengan 1 maka *rule* tersebut menunjukkan ada manfaat dan *rule* akan disimpan.

1. **Perhitungan Manual**

Perhitungan manual dilakukan pada *sample data* yang diambil dari data hasil survey mengenai spesifikasi *smartphone*. *Sample data* yang diambil sebanyak 20 data spesifikasi *smartphone* yang memiliki sistem operasi Android. Nilai *minimum support* yang digunakan dalam proses perhitungan manual ini yaitu 20% sedangkan nilai *minimum confidence*-nya yaitu 40%.

Dalam proses perhitungan manual ini, *sample data* yang dipakai dilakukan proses *preprocessing* terlebih dulu sampai bentuk data sesuai dengan yang dibutuhkan kemudian baru dilakukan penerapan metode *association rule* dengan algoritma *FP-Growth*. Pada proses *preprocessing*, dilakukan proses transformasi pada data kedalam bentuk kode-kode untuk mempermudah dalam proses perhitungan. *Sample data* yang dipakai dalam perhitungan manual ini dapat dilihat pada Tabel 3.1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Memory Internal** | **RAM** | **Core CPU** | **Clock Speed CPU** | **GPU** |
| 1 | 4 GB | 512 MB RAM | Single-core | 1 GHz | Adreno 200 |
| 2 | 16 GB | 2 GB RAM | Quad-core | 1.5 GHz | Adreno 320 |
| 3 | 4 GB | 512 MB RAM | Single-core | 1 GHz | Mali 300 |
| 4 | 4 GB | 512 MB RAM | Single-core | 1 GHz | Adreno 200 |
| 5 | 2.9 GB | 512 MB RAM | Single-core | 800 MHz | Adreno 200 |
| 6 | 4 GB | 512 MB RAM | Single-core | 1 GHz | Adreno 200 |
| 7 | 4 GB | 512 MB RAM | Single-core | 850 MHz | VideoCore IV HW / - |
| 8 | 4 GB | 512 MB RAM | Single-core | 1 GHz | Mali 300 |
| 9 | 180 MB | 290 MB RAM | Single-core | 830 MHz | Mali 400MP |
| 10 | 4 GB | 512 MB RAM | Single-core | 1 GHz | PowerVR SGX531 |
| 11 | 4 GB | 512 MB RAM | Dual-core | 1 GHz | Mali 400 |
| 12 | 110 MB | 256 MB RAM | Single-core | 528 MHz | Adreno 130 |
| 13 | 4 GB | 512 MB RAM | Dual-core | 1 GHz | Adreno 203 |
| 14 | 1 GB | 512 MB RAM | Single-core | 800 MHz | Adreno 200 |
| 15 | 4 GB | 512 MB RAM | Single-core | 1 GHz | PowerVR SGX531 |
| 16 | 8 GB | 1 GB RAM | Dual-core | 1.2 GHz | Adreno 203 |
| 17 | 4 GB | 768 MB RAM | Dual-core | 1 GHz | Adreno 203 |
| 18 | 11 GB | 1 GB RAM | Dual-core | 1.5 GHz | Adreno 220 |
| 19 | 8 GB | 1 GB RAM | Dual-core | 1.2 GHz | Adreno 203 |
| 20 | 16 GB | 1 GB RAM | Quad-core | 1.4 GHz | Mali 400MP4 |

Tabel 3.1 *Sample data* spesifikasi *smartphone* dengan sistem operasi Android

Langkah pertama yang dilakukan yaitu proses *preprocessing* data dimana dilakukan pemberian kode untuk setiap item data spesifikasi memori internal, RAM, core CPU, clock speed CPU, dan GPU *smartphone* Android. Pemberian kode untuk setiap item spesifikasi dapat dilihat pada Tabel 3.2 sampai dengan Tabel 3.6.

Tabel 3.2 Kode memori internal

|  |  |
| --- | --- |
| Memory Internal | Kode Memory Internal |
| 4 GB | 1 |
| 16 GB | 2 |
| 2.9 GB | 3 |
| 180 MB | 4 |
| 110 MB | 5 |
| 1 GB | 6 |
| 8 GB | 7 |
| 11 GB | 8 |

Tabel 3.3 Kode RAM

|  |  |
| --- | --- |
| Jumlah RAM | Kode Jumlah RAM |
| 512 MB RAM | 9 |
| 2 GB RAM | 10 |
| 290 MB RAM | 11 |
| 256 MB RAM | 12 |
| 1 GB RAM | 13 |
| 768 MB RAM | 14 |

Tabel 3.4 Kode core CPU

|  |  |
| --- | --- |
| Core CPU | Kode Core CPU |
| Single Core | 15 |
| Quad Core | 16 |
| Dual Core | 17 |

Tabel 3.5 Kode clock speed CPU

|  |  |
| --- | --- |
| Clock Speed CPU | Kode Clock CPU |
| 1 Ghz | 18 |
| 1.5 Ghz | 19 |
| 800 MHz | 20 |
| 850 MHz | 21 |
| 830 MHz | 22 |
| 528 MHz | 23 |
| 1.2 Ghz | 24 |
| 1.4 Ghz | 25 |

Tabel 3.6 Kode GPU

|  |  |
| --- | --- |
| GPU | Kode GPU |
| Adreno 200 | 26 |
| Adreno 320 | 27 |
| Mali 300 | 28 |
| VideoCore IV HW / - | 29 |
| Mali 400MP | 30 |
| PowerVR SGX531 | 31 |
| Mali 400 | 32 |
| Adreno 130 | 33 |
| Adreno 203 | 34 |
| Adreno 220 | 35 |
| Mali 400MP4 | 36 |

Langkah kedua, setelah dilakukan pemberian kode untuk setiap item data spesifikasi proses selanjutnya yaitu melakukan transformasi data spesifikasi *smartphone* Android menjadi kode-kode seperti yang telah dibuat sebelumnya. Hasil proses transformasi data dalam bentuk kode dapat dilihat pada Tabel 3.7.

Tabel 3.7 Hasil proses transformasi data ke dalam kode

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Memory Internal** | **RAM** | **Core CPU** | **Clock Speed CPU** | **GPU** |
| 1 | 1 | 9 | 15 | 18 | 26 |
| 2 | 2 | 10 | 16 | 19 | 27 |
| 3 | 1 | 9 | 15 | 18 | 28 |
| 4 | 1 | 9 | 15 | 18 | 26 |
| 5 | 3 | 9 | 15 | 20 | 26 |
| 6 | 1 | 9 | 15 | 18 | 26 |
| 7 | 1 | 9 | 15 | 21 | 29 |
| 8 | 1 | 9 | 15 | 18 | 28 |
| 9 | 4 | 11 | 15 | 22 | 30 |
| 10 | 1 | 9 | 15 | 18 | 31 |
| 11 | 1 | 9 | 17 | 18 | 32 |
| 12 | 5 | 12 | 15 | 23 | 33 |
| 13 | 1 | 9 | 17 | 18 | 34 |
| 14 | 6 | 9 | 15 | 20 | 26 |
| 15 | 1 | 9 | 15 | 18 | 31 |
| 16 | 7 | 13 | 17 | 24 | 34 |
| 17 | 1 | 14 | 17 | 18 | 34 |
| 18 | 8 | 13 | 17 | 19 | 35 |
| 19 | 7 | 13 | 17 | 24 | 34 |
| 20 | 2 | 13 | 16 | 25 | 36 |

Langkah ketiga yaitu menghitung frekuensi kemunculan tiap spesifikasi memori internal, RAM, core CPU, clock speed CPU, dan GPU. Hasil perhitungan frekuensi diurutkan berdasarkan nilai frekuensi terbanyak seperti yang tertera pada Tabel 3.8.

Tabel 3.8 Frekuensi kemunculan tiap spesifikasi *smartphone*

|  |  |
| --- | --- |
| **Kode Spesifikasi** | **Frekuensi** |
| 9 | 12 |
| 15 | 12 |
| 1 | 11 |
| 18 | 10 |
| 17 | 6 |
| 26 | 5 |
| 13 | 4 |
| **Kode Spesifikasi** | **Frekuensi** |
| 34 | 4 |
| 2 | 2 |
| 7 | 2 |
| 16 | 2 |
| 19 | 2 |
| 20 | 2 |
| 24 | 2 |
| 28 | 2 |
| 31 | 2 |
| 3 | 1 |
| 4 | 1 |
| 5 | 1 |
| 6 | 1 |
| 8 | 1 |
| 10 | 1 |
| 11 | 1 |
| 12 | 1 |
| 14 | 1 |
| 21 | 1 |
| 22 | 1 |
| 23 | 1 |
| 25 | 1 |
| 27 | 1 |
| 29 | 1 |
| 30 | 1 |
| 32 | 1 |
| 33 | 1 |
| 35 | 1 |
| 36 | 1 |

Langkah keempat yaitu penghitungan *support* pada kandidat *frequent 1-itemset*. Nilai *support* didapatkan dari perbandingan frekuensi dengan total data spesifikasi. Hasil perhitungan *support* diurutkan berdasarkan nilai *support* tertinggi seperti yang tertera pada Tabel 3.9.

Tabel 3.9 *Support* kandidat *frequent 1-itemset*

|  |  |  |
| --- | --- | --- |
| **Kode Spesifikasi** | **Frekuensi** | ***Support*** |
| 9 | 12 | = 12/20 = 0.6 |
| 15 | 12 | = 12/20 = 0.6 |
| 1 | 11 | = 11/20 = 0.55 |
| 18 | 10 | = 10/20 = 0.5 |
| 17 | 6 | = 6/20 = 0.3 |
| 26 | 5 | = 5/20 = 0.25 |
| 13 | 4 | = 4/20 = 0.2 |
| 34 | 4 | = 4/20 = 0.2 |
| 2 | 2 | = 2/20 = 0.1 |
| 7 | 2 | = 2/20 = 0.1 |
| 16 | 2 | = 2/20 = 0.1 |
| 19 | 2 | = 2/20 = 0.1 |
| 20 | 2 | = 2/20 = 0.1 |
| 24 | 2 | = 2/20 = 0.1 |
| 28 | 2 | = 2/20 = 0.1 |
| 31 | 2 | = 2/20 = 0.1 |
| 3 | 1 | = 1/20 = 0.05 |
| 4 | 1 | = 1/20 = 0.05 |
| 5 | 1 | = 1/20 = 0.05 |
| 6 | 1 | = 1/20 = 0.05 |
| 8 | 1 | = 1/20 = 0.05 |
| 10 | 1 | = 1/20 = 0.05 |
| 11 | 1 | = 1/20 = 0.05 |
| 12 | 1 | = 1/20 = 0.05 |
| 14 | 1 | = 1/20 = 0.05 |
| 21 | 1 | = 1/20 = 0.05 |
| 22 | 1 | = 1/20 = 0.05 |
| 23 | 1 | = 1/20 = 0.05 |
| 25 | 1 | = 1/20 = 0.05 |
| 27 | 1 | = 1/20 = 0.05 |
| 29 | 1 | = 1/20 = 0.05 |
| 30 | 1 | = 1/20 = 0.05 |
| 32 | 1 | = 1/20 = 0.05 |
| 33 | 1 | = 1/20 = 0.05 |
| 35 | 1 | = 1/20 = 0.05 |
| 36 | 1 | = 1/20 = 0.05 |

Langkah kelima yaitu pencarian *frequent 1-itemset* dari kandidat *frequent 1-itemset*. Pada tahap ini *item* yang tidak memenuhi *minimum support* akan dihilangkan. Dan *item* yang telah *frequent* (memenuhi *minimum support*) dapat dilihat dalam Tabel 3.10.

Tabel 3.10 *Frequent 1-itemset*

|  |  |  |
| --- | --- | --- |
| **Frequent 1-*itemset*** | | ***Support*** |
| 9 | 0.6 | |
| 15 | 0.6 | |
| 1 | 0.55 | |
| 18 | 0.5 | |
| 17 | 0.3 | |
| 26 | 0.25 | |
| 13 | 0.2 | |
| 34 | 0.2 | |

Langkah keenam yaitu melakukan seleksi data spesifikasi *smartphone* Android yang telah memenuhi *frequent 1-itemset*. Item spesifikasi yang tidak memenuhi *frequent 1-itemset* dihilangkan. Item dari masing-masing data spesifikasi tersebut sebelumnya diurutkan sesuai dengan nilai *support* terbesar berdasarkan Tabel 3.10. Proses seleksi data spesifikasi dapat dilihat dalam Tabel 3.11.

Tabel 3.11 Proses seleksi *sample data* spesifikasi *smartphone* Android

|  |  |
| --- | --- |
| **No** | **Data Spesifikasi** |
| 1 | 9, 15, 1, 18, 26 |
| 2 | 2, 16, 19, 10, 27 |
| 3 | 9, 15, 1, 18, 28 |
| 4 | 9, 15, 1, 18, 26 |
| 5 | 9, 15, 26, 20, 3 |
| 6 | 9, 15, 1, 18, 26 |
| 7 | 9, 15, 1, 21, 29 |
| 8 | 9, 15, 1, 18, 28 |
| 9 | 15, 4, 11, 22, 30 |
| 10 | 9, 15, 1, 18, 31 |
| 11 | 9, 1, 18, 17, 32 |
| 12 | 15, 5, 12, 23, 33 |
| 13 | 9, 1, 18, 17, 34 |
| **No** | **Data Spesifikasi** |
| 14 | 9, 15, 26, 20, 6 |
| 15 | 9, 15, 1, 18, 31 |
| 16 | 17, 13, 34, 7, 24 |
| 17 | 1, 18, 17, 34, 14 |
| 18 | 17, 13, 19, 8, 35 |
| 19 | 17, 13, 34, 7, 24, |
| 20 | 13, 2, 16, 25, 36 |

Perubahan pada *sample data* spesifikasi *smartphone* Android dapat dilihat dalam Tabel 3.12.

Tabel 3.12 Hasil proses seleksi data spesifikasi *smartphone* Android

|  |  |
| --- | --- |
| **No** | **Data Spesifikasi** |
| 1 | 9, 15, 1, 18, 26 |
| 3 | 9, 15, 1, 18 |
| 4 | 9, 15, 1, 18, 26 |
| 5 | 9, 15, 26 |
| 6 | 9, 15, 1, 18, 26 |
| 7 | 9, 15, 1 |
| 8 | 9, 15, 1, 18 |
| 9 | 15 |
| 10 | 9, 15, 1, 18 |
| 11 | 9, 1, 18, 17 |
| 12 | 15 |
| 13 | 9, 1, 18, 17, 34 |
| 14 | 9, 15, 26 |
| 15 | 9, 15, 1, 18 |
| 16 | 17, 13, 34 |
| 17 | 1, 18, 17, 34 |
| 18 | 17, 13 |
| 19 | 17, 13, 34 |
| 20 | 13 |

Langkah ketujuh yaitu pembuatan *FP-Tree* dari hasil proses seleksi data spesifikasi *smartphone* seperti yang tertera dalam Gambar 3.13 sampai Gambar 3.18. Pada pembuatan *tree* awal dibuatlah *root* dan diberi nilai null. Dan pada *Head of table link* untuk semua kode *itemset* diberi nilai null.

*Header table*

|  |  |
| --- | --- |
| ***Itemset*** | ***Head of table link*** |
| 9 | null |
| 15 | null |
| 1 | null |
| 18 | null |
| 17 | null |
| 26 | null |
| 13 | null |
| 34 | null |



Gambar 3.13 *FP-Tree* awal

Pembuatan *FP-Tree* selanjutnya adalah pembuatan *FP-Tree* untuk data spesifikasi 1 yaitu {9, 15, 1, 18, 26}. Pembacaan yang pertama untuk kode *itemset* 9, dimana untukdibuat node baru yang merupakan node anak dari *root* dan diberi label 9 serta *count* diberi nilai 1. Dan pada setiap pembuatan node baru pada *tree* akan dibuat suatu *link* dari *itemset* pada tabel ke node tersebut, dimana *link* ini menunjukkan informasi mengenai *itemset* tersebut disimpan. Setelah pembuatan node pertama lalu pembuatan node untuk kode spesifikasi selanjutnya yaitu 15. Karena pada node 9 belum memiliki anak yang mempunyai label 15 maka dibuat node anak baru dengan label 15 dan diberi nilai *count* 1. Begitu juga untuk kode spesifikasi selanjutnya yaitu 1, 18 dan 26, dibuat menjadi node-node anak baru dan diberi nilai *count* 1.



*Header table*

|  |  |
| --- | --- |
| ***Itemset*** | ***Head of table link*** |
| 9 |  |
| 15 |  |
| 1 |  |
| 18 |  |
| 17 | null |
| 26 |  |
| 13 | null |
| 34 | null |

Gambar 3.14 *FP-Tree* pembacaan data spesifikasi 1

Langkah pembuatan *FP-Tree* yang kedua adalah untuk data spesifikasi 3 yaitu {9, 15, 1, 18}. Karena pada *FP-Tree* sebelumnya sudah ada node-node dengan urutan label yang sama maka tidak membuat node-node baru melainkan hanya menambah 1 nilai *count* dari node-node yang telah ada tersebut sehingga menjadi bernilai 2.



*Header table*

|  |  |
| --- | --- |
| ***Itemset*** | ***Head of table link*** |
| 9 |  |
| 15 |  |
| 1 |  |
| 18 |  |
| 17 | null |
| 26 |  |
| 13 | null |
| 34 | null |

Gambar 3.15 *FP-Tree* pembacaan data spesifikasi 3

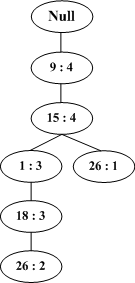
Langkah pembuatan *FP-Tree* yang ketiga adalah untuk data spesifikasi 4 yaitu {9, 15, 1, 18, 26}. Karena pada *FP-Tree* sebelumnya sudah ada node-node dengan urutan label yang sama maka tidak membuat node-node baru melainkan hanya menambah 1 nilai *count* dari node-node yang telah ada tersebut.

*Header table*

|  |  |
| --- | --- |
| ***Itemset*** | ***Head of table link*** |
| 9 |  |
| 15 |  |
| 1 |  |
| 18 |  |
| 17 | null |
| 26 |  |
| 13 | null |
| 34 | null |

Gambar 3.16 *FP-Tree* pembacaan data spesifikasi 4

Langkah pembuatan *FP-Tree* yang keempat adalah untuk data spesifikasi 5 yaitu {9, 15, 26}. Karena pada *FP-Tree* sebelumnya sudah ada node orang tua dengan label 9 kemudian anak dari node 9 dengan label 15, maka tidak membuat node baru melainkan hanya menambah 1 nilai *count* dari label-label tersebut. Dan untuk *itemset* 26, karena node 15 belum memiliki anak dengan label 26 maka dibuat node anak baru dari node 15 dan diberi label 26 serta nilai *count* 1.

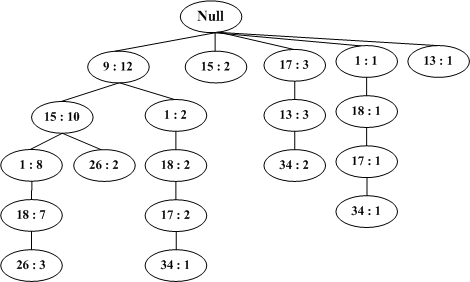


*Header table*

|  |  |
| --- | --- |
| ***Itemset*** | ***Head of table link*** |
| 9 |  |
| 15 |  |
| 1 |  |
| 18 |  |
| 17 | null |
| 26 |  |
| 13 | null |
| 34 | null |

Gambar 3.17 *FP-Tree* pembacaan data spesifikasi 5

Pembangunan *FP-Tree* terus berlanjut sampai data spesifikasi ke-20. Bentuk *FP-Tree* yang telah lengkap dapat dilihat pada Gambar 3.18.



Gambar 3.18 *FP-Tree* pembacaan data spesifikasi 20

Setelah pembuatan *FP-Tree*, langkah ke-delapan yaitu *Conditional Pattern Base*, yakni mencari *suffix* pattern dan juga *prefix* *path*-nya pada *FP-Tree* yang telah terbentuk. Dari *FP-Tree* yang telah terbentuk sebelumnya maka didapatkan *conditional pattern base*-nya yang dapat dilihat pada Tabel 3.13.

Tabel 3.13 *Conditional pattern base* dari *sample data* spesifikasi

|  |  |
| --- | --- |
| ***Suffix*** | ***Conditional* Pattern Base** |
| 26 | {(9, 15, 1, 18) : 3,  (9, 15) : 2 } |
| 34 | {(9, 1, 18, 17) : 1,  (17, 13) : 2,  (1, 18, 17) : 1} |
| 18 | {(9, 15, 1) : 7,  (9, 1) : 2,  (1) : 1} |
| 17 | {(9, 1, 18) : 2,  (1, 18) : 1} |
| 1 | {(9, 15) : 8,  (9) : 2} |
| 15 | {(9) : 10} |
| 13 | {(17) : 3} |

Langkah ke-sembilan yaitu tahap *Conditional* *FP-Tree* dimana dicari nilai *support* dari masing-masing *itemset* tiap *suffix*. *Itemset* yang memiliki jumlah *support* lebih dari atau sama dengan nilai *minimum support* dan yang memiliki satu item di kedua bagian *prefix* dan *suffix* akan dijadikan *frequent itemset*.

Hasil *frequent itemset* dari proses *Conditional* *FP-Tree* digabung menjadi satu dalam Tabel 3.14.

.

Tabel 3.14 Hasil *frequent itemset*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | ***Frequent itemset*** | **Frekuensi** | ***Support*** |
| 1 | 9, 26 | 5 | 0.25 |
| 2 | 15, 26 | 5 | 0.25 |
| 3 | 9, 15, 26 | 5 | 0.25 |
| 4 | 17, 34 | 4 | 0.2 |
| 5 | 1, 18 | 10 | 0.5 |
| 6 | 9, 18 | 9 | 0.45 |
| 7 | 15, 18 | 7 | 0.35 |
| 8 | 9, 1, 18 | 9 | 0.45 |
| 9 | 15, 1, 18 | 7 | 0.35 |
| 10 | 9, 15, 18 | 7 | 0.35 |
| 11 | 9, 15, 1, 18 | 7 | 0.35 |
| 12 | 9, 1 | 10 | 0.5 |
| 13 | 15, 1 | 8 | 0.4 |
| 14 | 9, 15, 1 | 8 | 0.4 |
| 15 | 9, 15 | 10 | 0.5 |

Langkah selanjutnya yaitu *frequent itemset* yang telah didapat kemudian dilakukan proses *generate rule* . Dalam proses *generate rule* ini dilakukan penghitungan nilai *confidence* dari *frequent itemset*. Nilai *confidence* didapatkan dari perbandingan nilai *support* (A,B) dengan nilai *support* (A). Nilai *confidence* ini digunakan untuk menentukan apakah *frequent itemset* tersebut dapat dibangkitkan menjadi *rule*. Jika nilai *confidence* yang didapatkan adalah lebih besar atau sama dengan nilai *minimum confidence* maka *frequent itemset* tersebut dapat dibangkitkan menjadi *rule*. Untuk penghitungan nilai *confidence* dari *frequent itemset* dapat dilihat pada Tabel 3.15.

* 9 → 26

Tabel 3.15 Nilai *confidence* dari *frequent itemset*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | ***Frequent itemset*** | ***Support* (A,B)** | ***Support* A** | ***Confidence*** |
| 1 | 9, 26 | 0.25 | 0.6 | 0.42 |
| 2 | 15, 26 | 0.25 | 0.6 | 0.42 |
| 3 | 9, 15, 26 | 0.25 | 0.5 | 0.5 |
| 4 | 17, 34 | 0.2 | 0.3 | 0.67 |
| 5 | 1, 18 | 0.5 | 0.55 | 0.91 |
| 6 | 9, 18 | 0.45 | 0.6 | 0.75 |
| 7 | 15, 18 | 0.35 | 0.6 | 0.58 |
| 8 | 9, 1, 18 | 0.45 | 0.5 | 0.9 |
| 9 | 15, 1, 18 | 0.35 | 0.4 | 0.88 |
| 10 | 9, 15, 18 | 0.35 | 0.5 | 0.7 |
| 11 | 9, 15, 1, 18 | 0.35 | 0.4 | 0.88 |
| 12 | 9, 1 | 0.5 | 0.6 | 0.83 |
| 13 | 15, 1 | 0.4 | 0.6 | 0.67 |
| 14 | 9, 15, 1 | 0.4 | 0.5 | 0.8 |
| 15 | 9, 15 | 0.5 | 0.6 | 0.83 |

Berdasarkan Tabel 3.15, semua *frequent itemset* terpilih untuk dibangkitkan menjadi *rule* karena mempunyai nilai *confidence* yang lebih besar dari nilai *minimum confidence* yang sebelumnya telah ditentukan.

Selanjutnya *rule* yang didapatkan dilakukan pengujian kekuatan *rule* nya dengan menggunakan *lift ratio*. Nilai *lift ratio* didapatkan dari perbandingan antara *confidence* dengan *benchmark confidence*. Nilai dari *benchmark confidence* tersebut didapatkan dari perbandingan antara jumlah *frekuensi* data yang mengandung *consequent* (akibat) dengan total data spesifikasi. Kekuatan *rule* dikatakan baik bila memiliki nilai *lift ratio* lebih besar dari 1.

* 9 → 26

Tabel 3.16 *Lift ratio* dari *rule*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | ***Rule*** | ***Confidence*** | ***Benchmark Confidence*** | ***Lift Ratio*** |
| 1 | 9 → 26 | 0.42 | 0.25 | 1.68 |
| 2 | 15 → 26 | 0.42 | 0.25 | 1.68 |
| 3 | 9, 15 → 26 | 0.5 | 0.25 | 2 |
| 4 | 17→ 34 | 0.67 | 0.2 | 3.35 |
| 5 | 1 → 18 | 0.91 | 0.5 | 1.82 |
| 6 | 9 → 18 | 0.75 | 0.5 | 1.5 |
| 7 | 15 → 18 | 0.58 | 0.5 | 1.16 |
| 8 | 9, 1 → 18 | 0.9 | 0.5 | 1.8 |
| 9 | 15, 1 → 18 | 0.88 | 0.5 | 1.76 |
| 10 | 9, 15 → 18 | 0.7 | 0.5 | 1.4 |
| 11 | 9, 15, 1 → 18 | 0.88 | 0.5 | 1.76 |
| 12 | 9 → 1 | 0.83 | 0.55 | 1.51 |
| 13 | 15 → 1 | 0.67 | 0.55 | 1.22 |
| 14 | 9, 15 → 1 | 0.8 | 0.55 | 1.45 |
| 15 | 9 → 15 | 0.83 | 0.6 | 1.38 |

Berdasarkan hasil perhitungan *lift ratio* yang telah dilakukan, dapat dilihat bahwa semua *rule* memiliki kekuatan yang baik karena nilai *lift ratio* semua *rule* lebih besar dari 1. Keterangan *Rule* yang terbentuk beserta nilai *lift* *ratio*-nya dapat dilihat dalam Tabel 3.17.

Tabel 3.17 Keterangan *rule* dan nilai lift ratio

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | ***Rule*** | **Keterangan** | ***Lift Ratio*** |
| 1 | 9 → 26 | Jika memiliki spesifikasi 512 MB RAM maka memiliki spesifikasi GPU Adreno 200. | 1.68 |
| 2 | 15 → 26 | Jika memiliki spesifikasi Core CPU Single Core maka memiliki spesifikasi GPU Adreno 200. | 1.68 |
| 3 | 9, 15 → 26 | Jika memiliki spesifikasi 512 MB RAM dan Core CPU Single Core maka memiliki spesifikasi GPU Adreno 200. | 2 |
| 4 | 17→ 34 | Jika memiliki spesifikasi Core CPU Dual Core maka memiliki spesifikasi GPU Adreno 203. | 3.35 |
| 5 | 1 → 18 | Jika memiliki spesifikasi Memory Internal 4 GB maka memiliki spesifikasi Clock Speed CPU 1 Ghz. | 1.82 |
| 6 | 9 → 18 | Jika memiliki spesifikasi 512 MB RAM maka memiliki spesifikasi Clock Speed CPU 1 Ghz. | 1.5 |
| 7 | 15 → 18 | Jika memiliki spesifikasi Core CPU Single Core maka memiliki spesifikasi Clock Speed CPU 1 Ghz. | 1.16 |
| 8 | 9, 1 → 18 | Jika memiliki spesifikasi 512 MB RAM dan Memory Internal 4 GB maka memiliki spesifikasi Clock Speed CPU 1 Ghz. | 1.8 |
| 9 | 15, 1 → 18 | Jika memiliki spesifikasi Core CPU Single Core dan Memory Internal 4 GB maka memiliki spesifikasi Clock Speed CPU 1 Ghz. | 1.76 |
| 10 | 9, 15 → 18 | Jika memiliki spesifikasi 512 MB RAM dan Core CPU Single Core maka memiliki spesifikasi Clock Speed CPU 1 Ghz. | 1.4 |
| 11 | 9, 15, 1 → 18 | Jika memiliki spesifikasi 512 MB RAM dan Core CPU Single Core dan Memory Internal 4 GB maka memiliki spesifikasi Clock Speed CPU 1 Ghz. | 1.76 |
| 12 | 9 → 1 | Jika memiliki spesifikasi 512 MB RAM maka memiliki spesifikasi Memory Internal 4 GB. | 1.51 |
| 13 | 15 → 1 | Jika memiliki spesifikasi Core CPU Single Core maka memiliki spesifikasi Memory Internal 4 GB. | 1.22 |
| 14 | 9, 15 → 1 | Jika memiliki spesifikasi 512 MB RAM dan Core CPU Single Core maka memiliki spesifikasi Memory Internal 4 GB. | 1.45 |
| 15 | 9 → 15 | Jika memiliki spesifikasi 512 MB RAM maka memiliki spesifikasi Core CPU Single Core. | 1.38 |

1. **Rancangan Antar Muka**

Rancangan antar muka secara umum terdiri dari tampilan awal dijalankannya program. Tampilan antar muka secara umum dapat dilihat pada Gambar 3.19.



Gambar 3.19 Rancangan antar muka secara umum

1. **Rancangan Uji Coba**

Uji coba sistem dibagi menjadi 2 pengujian. Pengujian pertama untuk mengetahui pengaruh *minimum support* dan *minimum confidence* terhadap jumlah *rule* yang dihasilkan. Pengujian kedua bertujuan untuk mengetahui pengaruh *minimum support* dan *minimum confidence* terhadap rata-rata *lift ratio*. Tabel pengujian dapat dilihat pada Tabel 3.18 dan Tabel 3.19.

Tabel 3.18 Tabel pengujian pengaruh *minimum support* dan *minimum confidence* terhadap jumlah *rule*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Minimum Support* (%)** | **Jumlah *Rule*** | | | | | | | | | |
| ***Minimum Confidence* (%)** | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

Tabel 3.19 Tabel pengujian pengaruh *minimum support* dan *minimum confidence* terhadap rata-rata *lift ratio*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Minimum Support* (%)** | **Rata-Rata *Lift Ratio*** | | | | | | | | | |
| ***Minimum Confidence* (%)** | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |