LAPORAN TUGAS BESAR AKHIR SEMESTER

Struktur Data dan Algoritma

MARKET BASKET ANALYSIS - INTRODUCTION TO DATA MINING

Disusun oleh

Fachri Dhia Fauzan (191524041)
M. Syahid Abdurrahman (191524051)

Program Studi D-IV Teknik Informatika
Jurusan Teknik Komputer dan Informatika
Politeknik Negeri Bandung

2020

1. DESKRIPSI APLIKASI

1.1 Market Basket Analysis

Market basket analysis adalah suatu analisis atas perilaku konsumen secara spesifik dari suatu golongan / kelompok tertentu. Sumber data dari market basket analysis antara lain dapat bersumber dari transaksi kartu kredit, kartu lotre, kupon diskon, panggilan keluhan pelanggan. Market basket analysis umumnya dimanfaatkan sebagai titik awal pencarian pengetahuan dari suatu transaksi data ketika kita tidak mengetahui pola spesifik apa yang kita cari. Kebutuhan market basket analysis berawal dari keakuratan dan manfaat yang dihasilkannya dalam wujud aturan asosiasi (association rules). Yang dimaksud dengan association rules adalah pola-pola keterkaitan data dalam basis data.

Proses market basket analysis dimulai dengan transaksi yang terdiri dari satu/lebih penawaran produk/jasa dan beberapa informasi dasar suatu transaksi. Hasil dari market basket analysis adalah berwujud aturan assosiasi (association rules).

$$support\ (A \Rightarrow B) = P(A \cup B)$$

Gambar 1.1 - Persamaan 1

Persaman 1 menjelaskan bahwa nilai support itemset A terhadap itemset B sebesar probabilitas dari gabungan itemset A dan B. Contoh, pada itemset yang memiliki support seperti berikut :

List Transaksi :

1. Pembelian Barang 1,2,3
2. Pembelian Barang 2,3,4
3. Pembelian barang 1,2,5
4. Pembelian Barang 1,2,4

Perhitungan Support :

(A U B) = 3 (transaksi 1,2,4)

Maka, Support (A => B) adalah Support([1,2]) = 3

$$confidence\ (A \Rightarrow B) = P(B|A) = \frac{support\_count\ (A \cup B)}{support\_count\ (A)}$$

Gambar 1.2 - Persamaan 2

Persamaan 2 menjelaskan bahwa persentase keyakinan (confidence) itemset A terhadap itemset B sebesar probabilitas dari gabungan itemset A dan B dibagi probabilitas itemset A. Pada contoh sebelumnya, dimana :

Perhitungan Support :
(A U B) = 3 (transaksi 1,2,4)
Support (A => B) adalah Support([1,2]) = 3
Support (A) adalah Support([1]) = 3
Support (B) adalah Support([2]) = 4
Perhitungan Confidence :
Confidence (A=>B) adalah :

$$\frac{Support([1,2])}{Support([1]) + Support([2])} = \frac{3}{4+3} = \frac{3}{7}$$

$$improvement = \frac{\mathrm{p}(conditional \text{ and } result)}{\mathrm{p}(conditional) \cdot \mathrm{p}(result)}$$

Gambar 1.3 - Persamaan 3

Persamaan 3 menampilkan perhitungan improvement. Dari persamaan tersebut dijelaskan bahwa besar nilai improvement diperoleh dari probabilititas gabungan penentu (conditional) dan hasil (result) dibagi dengan hasil perkalian antara probabilititas penentu dengan probabilititas hasil.
Ketika improvement lebih besar dari 1 maka hasil aturan menjadi lebih baik dari pada kesempatan acak dan merupakan hasil yang berguna. Dari rumus ini dapat disimpulkan bahwa nilai improvement akan semakin besar bila nilai probabilitas penentu (conditional) dan probabilitas hasil (result) semakin kecil.
Catatan : pada persamaa

1.2 Kelebihan Market Basket Analysis

1.2.1. Hasilnya jelas dan mudah dimengerti sebab hanya merupakan suatu pola "jika-maka". Misalnya: Jika produk A dan B dibeli secara bersamaan, maka kemungkinan produk C turut dibeli.

1.2.2. Market basket analysis sangat berguna untuk undirected data mining, yaitu pencarian awal pola.

1.2.3. Market basket analysis dapat memproses transaksi tanpa harus kehilangan informasi sebab dapat memproses banyak variabel tanpa perlu dirangkum (summarization) terlebih dahulu.

1.2.4. Proses komputasi yang mudah, meskipun jumlah perhitungan akan meningkat pesat bersamaan dengan peningkatan jumlah transaksi dan jumlah items yang berbeda dalam analisis.

1.3 Kekurangan Market Based Analysis

1.3.1. Tingkat pertumbuhan proses secara eksponensial sebagai akibat pertumbuhan ukuran data.

1.3.2. Memiliki keterbatasan untuk atribut data, misalnya hanya berdasarkan tipe produk.

1.3.3. Sulit untuk menentukan items yang akan diolah secara tepat, sebab frekuensi dari items tersebut harus diusahakan seimbang.

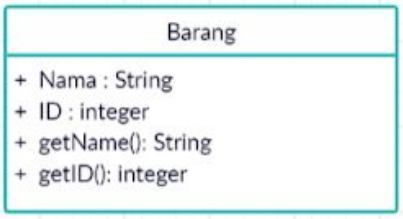
2. DESAIN APLIKASI

2.1. Struktur Data dan Perilakunya

2.1.1. Class Barang

Pada market basket analysis, variable utama tidak jauh dari sebuah barang yang akan dianalisis kecocokannya dengan barang yang lain. Nyatanya, sebuah barang dalam sebuah toko memuat nama barang itu sendiri, dan nomor barang (bisa berupa barcode yang akan discan pada pembelian, ataupun lainnya), untuk harga, dalam MBA (market based analysis) kita tidak memakainya karena kita akan menganalisis keterkaitan barang dengan barang lain, bukan masalah harga. Barang memiliki beberapa *behaviour* yang diperlukan seperti bagaimana kita mendapatkan nama, dan ID/nomor barang nya.

Dari uraian tersebut kita perlu menyimpan dua data yang akan membangun sebuah Barang di MBA. Struktur data ini yang akan dibuat sebagai Class di Bahasa Pemrograman Java seperti berikut.

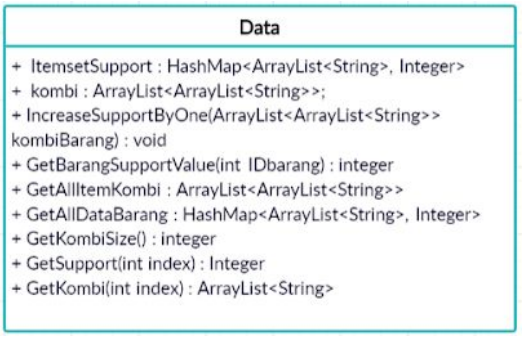


Gambar 2.1 Class Barang

Perlu diperhatikan bahwa nama barang dan ID barang tidak akan pernah berubah, sehingga kita tidak perlu menambahkan behaviour untuk mengganti nama dan ID barang.

2.1.2 Class Data (Data dari Toko)

Hal utama pada MBA merupakan Data. Data dimana ia akan menyimpan beberapa variabel seperti Kombinasi Barang (Tries) beserta Support-nya. Data ini yang akan dipakai dalam prosess perhitungan persamaan yang telah disampaikan (Persamaan 1-3).
Dengan demikian, kita perlu mendefinisikan Barang dengan ID-nya yang berupa String, maupun Integer (kita dapat mengkonversikan antara keduanya), sebagai sebuah Itemset (Kombinasi dari beberapa barang) beserta Supportnya. Untuk memudahkan akses itemset, kami membuat sebuah variable penampung bertype “ArrayList<ArrayList<String>>” sehingga itemset dapat diakses menggunakan index.



Gambar 2.2 Class Data

Pada Class Data, terdapat beberapa Fungsi dan Prosedur yang penting seperti:

a. IncreaseSupportByOne(ArrayList<ArrayList<String>> kombiBarang);
   Prosedur ini akan menambah value support dari ArrayList of ArrayList<String> yang merupakan sebuah kombinasi dari barang yang akan dibeli dan ditambah supportnya sebanyak 1. Tiap-Tiap ArrayList<String> yang diterima akan diubah keberadaannya di Atribut ItemSetSupport dengan mereplace, lalu menambahkan value (Supportnya dengan OldValue+1, yang artinya bertambah satu dari value sebelumnya).
b. GetBarangSupportValue(int IDbarang);
   Fungsi ini akan mengambil value IDbarang yang didapatkan dari method pemanggil, lalu akan mengubahnya sebagai String yang akan dimasukkan ke sebuah ArrayList sehingga kita dapat mencarinya di dalam HashMap ItemSetSupport, Apabila barang tidak ditemukan maka akan mengembalikan nilai 0.

2.1.3 Class Toko

Toko merupakan komponen utama dari MBA, ialah yang menyediakan barang dan transaksi apa saja yang akan kita analisis. Dalam toko, kita memiliki beberapa variable seperti List barang, dan Data pembeliannya, disini, kita menggunakan List barang sebagai barang apa saja yang kita jual, dan Data dengan class Data sebelumnya.

Gambar 2.3 Class Toko

Pada Class Toko, kita akan menempatkan List Barang dan Store Data (Berisi Support) yang akan diproses di Association Rules. Disini, kita memiliki method yang penting seperti :

a. Beli(ArrayList<String> pembelian);
   Fungsi ini akan mengembalikan “ArrayList<ArrayList<String>>”. Yang merupakan Array of kombination dari barang yang diinputkan. Alurnya, input barang yang berupa ArrayList<String> akan diambil tiap tiap Stringnya, lalu akan diambil ID dari nama barang tersebut (Misal Handphone, menjadi IDnya yaitu “1”). Lalu ID tersebut akan dimasukkan kedalam variable addKombi (Kombinasi yang akan ditambahkan Supportnya). Setelah itu, method ini akan membuat sebuah kombinasi dari tiap tiap elemen addKombi (Misal {“1”,”2”} menjadi {1} , {2}, {1,2}). Method ini akan mereturn kombinasi tersebut, sehingga method ini hanya akan digunakan ketika kita memanggil method **IncreaseSupportByOne** dengan memasukkan method **Beli** kedalam argument-nya.
b. searchBarangByID(int ID);
   Fungsi ini akan mengembalikan “String” nama dari barang yang IDnya ia dapatkan dari fungsi pemanggilnya. Jika tidak ditemukan akan mengembalikan null dan mengeluarkan tanda “Barang Tidak ditemukan” atau “List barang Kosong” ketika list barang di toko belum dibuat.

2.1.4 Class Kombi

Class ini merupakan class dari Pak Urip yang kami modifikasi dan tambahkan dari internet (yang juga dimodifikasi terlebih dahulu). Modifikasinya seperti berikut :

a. Sebelum modifikasi, class ini tidak mengembalikan apa apa ketika sebuah kombinasi di "generate", saya menghilangkan kode yang tidak diperlukan lalu mebuat ArrayList<String> sebagai penampungan kombinasi-kombinasi yang telah didapatkan dari proses yang sedemikian rupa, Sehingga dapat digunakan di dalam Association Rules utk menyimpan support dan itemset yang ada.
b. Terdapat fungsi tambahan dari luar yaitu "Combination Easy". dari sumber : https://github.com/mission-peace/interview/blob/master/src/com/interview/recursion/Combination.java. Dengan kode awal :

```
public void combinationEasy(char[] input) {
      List<Character> r = new ArrayList<>();
      Arrays.sort(input);
      combinationEasy(input, 0, r);
   }

   private void combinationEasy(char[] input, int pos, List<Character> r) {

      r.forEach(r1 -> System.out.print(r1 + " "));
      System.out.println();
      for (int i = pos; i < input.length; i++) {
         if (i != pos && input[i] == input[i-1]) {
            continue;
         }
         r.add(input[i]);
         combinationEasy(input, i + 1, r);
         r.remove(r.size() - 1);
      }
   }
```

Kode diatas menggunakan prinsip stack yang ketika dia di 'pop' ia akan dihapus, sehingga kita belum bisa menggunakan hasil dari kombinasi ke proses pembuatan kombinasi pembelian ataupun association rules yang lain, Namun kode ini akan bisa memenuhi kebutuhan saya (menginputkan sebuah array of char yang berupa id dari barang yang akan dikombinasikan dan dibuatkan kombinasinya). Sehingga saya mengubah kodenya menjadi seperti ini :

```
   public ArrayList<ArrayList<String>> getCombiFromInput(char[] input) {
      ArrayList<String> r = new ArrayList<>();
      ArrayList<String> temp = new ArrayList<>();
      ArrayList<ArrayList<String>> hasil = new ArrayList<>();
      Arrays.sort(input);
      return getCombiFromInput(input, 0, r, hasil, temp);
   }

   public ArrayList<ArrayList<String>> getCombiFromInput(char[] input, int pos,
ArrayList<String> r, ArrayList<ArrayList<String>> hasil, ArrayList<String> temp) {
```

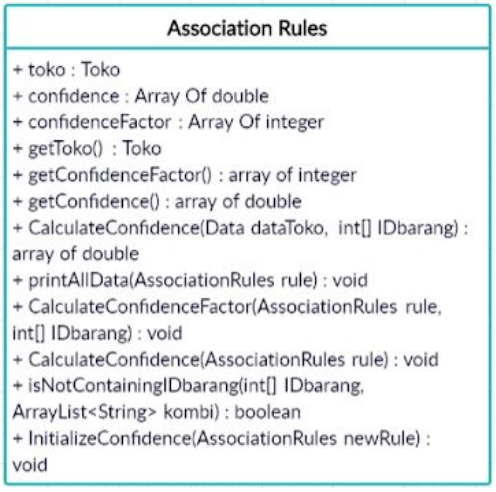
```
        temp = new ArrayList<>(r);
        hasil.add(temp);
        for (int i = pos; i < input.length; i++) {
            if (i != pos && input[i] == input[i-1]) {
                continue;
            }
            r.add(String.valueOf(input[i]));

            getCombiFromInput(input, i + 1, r, hasil,temp);
            r.remove(r.size() - 1);
        }
        return hasil;
    }
```

Kombinasi tersebut saya inputkan kedalam sebuah ArrayList<String> hasil, sehingga output dari fungsi ini bukan void, tapi merupakan ArrayList of ArrayList<String> yang merupakan sebuah kombinasi dari id barang - barang yang "Dibeli" dan akan ditambah Supportnya.

2.1.5 Class Association Rules

Association rules merupakan inti dari MBA, ia ada untuk menghitung "confidence" dari tiap tiap itemset yang ada terhadap barang yang lain sesuai dengan rumus di persamaan 1, 2 maupun 3. Karena itu, dalam class ini terdapat beberapa atribut seperti :

a. Confidence Factor (Array of integer)
b. Confidence (Array of double)
c. toko (Object dari Class Toko yang menampung data support dan kombinasi barang).



Gambar 2.4 Class Association Rules

Pada Class Association Rules, terdapat banyak fungsi penting untuk menghitung confidence ketika seseorang membeli barang. (Misal ia membeli barang 1, lalu akan diberikan confidence ke barang lainnya.)

a. public double[] CalculateConfidence(Data dataToko, int[] IDbarang).
   Main Function yang akan memanggil CalculateConfidenceFactor Untuk mendapatkan faktor dengan contoh (Pada List barang {A,B,C,D}, dan Itemset yang sedang di analisis {A,B} terhadap item C, ia akan mendapatkan confidence factor yaitu Support (A) + Support(B), lalu menjadi $\frac{Support(AnB\ U\ C)}{Support(A)+Support(B)}$.
b. private void CalculateConfidenceFactor(AssociationRules rule, int[] IDbarang)

CalculateConfidenceFactor menghitung Support penyebut (Support(AUB)) yang berada diatas pembagian dari Itemset yang sedang di analisis (Misal {A,B}) terhadap semua itemset lain kecuali item yang berada dalam itemset yang sedang dianalisis.

c. private double[] CalculateConfidence(AssociationRules rule)
   CalculateConfidence ini merupakan method overload (karena memiliki nama yang sama) untuk menghitung Confidence yang sebenarnya dengan membagi Confidence Factor dengan Support Factor dan mengembalikannya ke method pemanggil.

d. isNotContainingIDbarang(int[] IDbarang, ArrayList<String> kombi)
   Method boolean yang akan mengecek apakah ID barang yang didapatkan TIDAK berada dalam kombi, apabila benar akan mengembalikan True, apabila ADA, mengembalikan false.

e. InitializeConfidence(AssociationRules newRule)
   Initialize confidence dipakai dalam awal perhitungan agar semua atribut di association rules (confidence, confidence factor) memiliki nilai 0 bukan trash value ataupun belum terinisialisasi.

f. private int CalculateSupportFactor(AssociationRules rule, int[] IDbarang)
   Method/Fungsi ini menghitung Support($\in$ *IDbarang*) /Tiap tiap anggota IDbarang (ID dari barang yang sedang dianalisis). dengan menjumlahkan support tiap tiap barangnya.

g. public void printAllData(AssociationRules rule){
   Method ini akan menampilkan seluruh data yang ada mencakup suport, confidence, dan itemsetnya.

h. CalculateandGetThreshold();
   Method ini akan menambahkan semua jumlah confidence , lalu dibagi dari rata rata confidence yang lebih dari 0 yang akan menjadi Treshold.

2.2. Desain Proses

2.2.1. Operasi Penambahan Barang dalam Toko.

Membuat sebuah ArrayList baru untuk menampung data nama barang yang berupa String. Lalu meninstansiasikan Class Toko dengan memanggil factory methodnya, menggunakan ArrayList barang tadi sebagai argumen.

```
ArrayList<String> listbrg = new ArrayList<>();
listbrg.add("Komputer");
listbrg.add("Laptop");
listbrg.add("Headset");
listbrg.add("Earphone");
Toko myToko = Toko.CreateToko(listbrg);
```

Dalam potongan kode ini, toko akan memiliki list barang yang diinputkan kedalam ArrayList, lalu ID barang akan mulai dari 0, dari awal List (komputer-earphone)

2.2.2. Operasi “Pembelian” Barang / Penambahan Support Tiap barang yang dibeli.

Membuat sebuah ArrayList baru untuk menampung data nama barang yang akan dibeli(string). Memanggil method IncreaseSupportByOne dengan argumen Beli yang keduanya merupakan method dari Class Toko.

```
ArrayList<String> listpembelian = new ArrayList<>();
listpembelian.add("Komputer");
listpembelian.add("Laptop");
listpembelian.add("Headset");
indomaret.getData().IncreaseSupportByOne(indomaret.Beli(listpembelian));
```

Dengan begitu, “Komputer” , “Laptop”, “Headset” akan dibuatkan Kombinasinya oleh method Beli dengan unsurnya = {1,2,3} menjadi {1}{2}{3}{1,2} dan seterusnya, lalu dikirim ke method IncreaseSupportByOne untuk tiap tiap kombinasinya ditambahkan supportnya. Konsepnya seperti tries namun menggunakan Array of Array of String.

2.2.3. Operasi Perhitungan Confidence terhadap Barang Lain.

Dengan menginstansiasikan Objek dari Class AssociationRules, kita dapat menggunakan factory method untuk membuat perhitungan dengan memberikan jumlah barang dan toko dalam argumen nya, lalu memanggil method Calculate Confidence untuk mendapatkan nilai Confidence.

Contoh kasus : Barang yang akan dianalisa adalah barang dengan ID 1 dan 2. (Komputer dan Laptop)

```
AssociationRules newRule = CreateAssociationRules(listbrg.size(), myToko);
int[] a = {1,2};
double[] conf = newRule.CalculateConfidence(myToko.getData(), a);
```

Setelah memanggil CalculateConfidence, kita akan mendapatkan confidence terhadap tiap tiap barang yang ada di toko dengan tiap index merupakan ID-1, misal index 0 = barang dengan ID 1.

2.2.4. Rekomendasi

Setelah mendapatkan confidence terhadap barang-barang lainnya, Treshold dapat dihitung dengan jumlah nilai confidence yang ada dibagi jumlah confidence aktif (lebih dari 0) sebagai rata rata dan barang yang memenuhi treshold (confidence > treshold) akan direkomendasikan melalui method Recommend(); di class AssociationRules.

3. IMPLEMENTASI

3.1. Class Barang

Attribut dan Behaviour dari Barang yang dijual harus didefinisikan sebagai sebuah class, yang mencakup nama, dan ID, juga behaviour untuk mendapatkan nama dan juga ID barang.

Barang.java

```
package store;
public class Barang {
    private String name;
    private int id;
    public Barang (int id, String name){
        this.name = name;
        this.id = id;
    }
    public int GetID(){
        return this.id;
    }
    public String GetName(){
        return this.name;
    }
}
```

3.2. Class Data

Class data membutuhkan method untuk menambah Support dari sebuah transaksi (pembelian) yang dilakukan, membutuhkan pengambilan data nama barang berdasarkan ID, mendapatkan support barang, dan method untuk mendapatkan atribut-atributnya (method getter).

Data.java

```
package store;
import java.util.ArrayList;
import java.util.HashMap;
import static store.Kombi.generate;
public class Data {
    private HashMap<ArrayList<String>, Integer> ItemsetSupport;
    /* ArrayList string = ListBarang, Integer = support */
    private ArrayList<ArrayList<String>> kombi;
    public Data(int jmlJenisBarang){
        Kombi newkombi = new Kombi();
        kombi = generate(jmlJenisBarang);

        ItemsetSupport = new HashMap<>();
        kombi.forEach((b) -> {
            ItemsetSupport.put(b, 0);
        });
    }
    void IncreaseSupportByOne(ArrayList<ArrayList<String>> barang){
        int oldValue;
        for(ArrayList<String> oneBarang : barang){
            oldValue = ItemsetSupport.get(oneBarang);
```

```
            ItemsetSupport.put(oneBarang, oldValue+1); //Menambah Support dengan 1
        }
    }
    int GetBarangSupportValue(int IDbarang){
        ArrayList<String> brg = new ArrayList<>();
        brg.add(String.valueOf(IDbarang));
        if(ItemsetSupport.get(brg) != null){
        return ItemsetSupport.get(brg);
        }
        else{ return 0;}
    }
    /* Setter-Getter */
    ArrayList<String> GetKombi(int index){
        return kombi.get(index);
    }
    Integer GetSupport(int index){
        return ItemsetSupport.get(kombi.get(index));
    }
    int GetKombiSize(){
        return ItemsetSupport.size();
    }
    ArrayList<ArrayList<String>> GetAllItemKombi(){
        return kombi;
    }
    HashMap<ArrayList<String>, Integer> GetAllDataBarang(){
        return ItemsetSupport;
    }
}
```

3.3. Class Toko

Class toko diperlukan karena kita perlu menampung Data , dan List barang yang ada didalamnya. Toko mencakup Objek dari Class Data dan Array List dari Class Barang.

Toko.java

```
package store;
import java.util.ArrayList;
public class Toko {
    private ArrayList<Barang> BarangList;
    private Data StoreData;
    private Toko(ArrayList<String> listbrg){
        int i=1;
        BarangList = new ArrayList<>();
        Barang newBarang;
        for(String s : listbrg){
            newBarang = new Barang(i, s);
            BarangList.add(newBarang);
            i++;
        }
        StoreData = new Data(GetNumberofBarang());
    }
    public static Toko CreateToko(ArrayList<String> listbrg) {
```

```
        return new Toko(listbrg);
    }
    public Data getData(){
        return StoreData;
    }
    public ArrayList<Barang> GetAllBarang(){
        return BarangList;
    }
    public int GetNumberofBarang(){
        return BarangList.size();
    }
    public String searchBarangByID(int ID){
        if(this.GetAllBarang() == null){System.out.println("Toko belum memiliki stok barang.");}
        for(Barang b : this.GetAllBarang()){
            if(ID == b.GetID()){
                return b.GetName();
            }
        }
        System.out.println("Tidak ada barang yang dicari.");
        return null;
    }
    public ArrayList<ArrayList<String>> Beli(ArrayList<String> pembelian){
        int i = 0;
        char[] addKombi = new char[pembelian.size()];
        ArrayList<ArrayList<String>> newKombi = new ArrayList<>();
        for(String s : pembelian){
            for(Barang b : BarangList){
                if(s.equals(b.GetName())){
                    addKombi[i] = (char) ((char) b.GetID() + '0');
                    i++;
                }
            }
        }
        /* Masukkan Support ke kombi, lalu return (ke increase support, untuk supportnya ditambah
satusatu) */
        Kombi a = new Kombi();
        newKombi = a.getCombiFromInput(addKombi);
        newKombi.remove(0);
        return newKombi;
    }
}
```

3.4. Class AssociationRules

Association Rules merupakan inti dari MBA, ia menghitung confidence yang ia dapatkan dari data-data support yang ada, lalu menghitung treshold untuk kemudian direkomendasikan ketika seseorang membeli sesuatu atau saat kita menganalisis sebuah item/itemset.

AssociationRules.java

```
package store;
import java.util.ArrayList;
import java.util.HashMap;
```

```
public class AssociationRules {
   private Toko toko;
   private double[] confidence;
   private int[] confidenceFactor;
   private double treshold;
   private AssociationRules(int jmlBrg, Toko newToko){
      confidence = new double[jmlBrg];
      confidenceFactor = new int[jmlBrg];
      toko = newToko;
      for(int i = 0; i < jmlBrg; i++){
         confidence[i] = 0;
         confidenceFactor[i] = 0;
      }
   }

   public static AssociationRules CreateAssociationRules(int jmlbrg, Toko newToko) {
      return new AssociationRules(jmlbrg, newToko);
   }

   /* Setter - Getter */
   public Toko getToko(){
      return toko;
   }
   public int[] getConfidenceFactor(){
      return confidenceFactor;
   }

   private double[] getConfidence(){
      return confidence;
   }

   /* Process Method */

   /* Class untuk memproses persamaan, dimana ada 3 persamaan. */
   public double[] CalculateConfidence(Data dataToko, int[] IDbarang){
      InitializeConfidence(this);
      CalculateConfidenceFactor(this, IDbarang);
      int suppFactor  =  CalculateSupportFactor(this, IDbarang);
      CalculateConfidence(this, suppFactor);
      return confidence;
   }

   public void printAllData(){
       for(int i = 0 ; i < this.getToko().getData().GetKombiSize(); i++){
          System.out.println(this.getToko().getData().GetKombi(i) + " Support : " +
this.getToko().getData().GetSupport(i));
       }
   }

   private int CalculateSupportFactor(AssociationRules rule, int[] IDbarang){
```

```
        int supportFactorTemp = 0;
            for(int i = 0; i < IDbarang.length; i++){
                supportFactorTemp += rule.getToko().getData().GetBarangSupportValue(IDbarang[i]);
            }
        return supportFactorTemp;
    }

    private void CalculateConfidenceFactor(AssociationRules rule, int[] IDbarang){
        int[] confidenceFactorTemp = rule.getConfidenceFactor();
        for(ArrayList<String> kombi : rule.getToko().getData().GetAllItemKombi()){
            for(int i = 0; i < 4; i++){
                if( (IsNotMemberOf(i+1,IDbarang)) && (IsContainingIDbarang(IDbarang, i+1, kombi))
){
                    confidenceFactorTemp[i] = confidenceFactorTemp[i] +
rule.getToko().getData().GetAllDataBarang().get(kombi); //menambahkan value support yang
                }
            }
        }
    }
    private boolean IsNotMemberOf(int num, int[] ListID){
        for(int i = 0 ; i < ListID.length; i++){
            if(num == ListID[i]){
                return false;
            }
        }
        return true;
    }
    private double[] CalculateConfidence(AssociationRules rule, int suppFactor){
        int jmlBarang = rule.getConfidenceFactor().length;
        int[] supportList = new int[jmlBarang];
        double[] ConfidenceTemp = rule.getConfidence();
        int support = suppFactor;
        for(int i = 0;i<jmlBarang;i++){
            if(support == 0){
                ConfidenceTemp[i] = 0;
            }else{
                ConfidenceTemp[i] = (double)confidenceFactor[i] / (double)support;
            }
        }

       return ConfidenceTemp;
    }

    private boolean IsContainingIDbarang(int[] IDbarang, int index, ArrayList<String> kombi){

        boolean valid = true;
        if(!kombi.contains(String.valueOf(index))){
            return false;
        }
        for(int i = 0; i < IDbarang.length; i++){
```

```
            if(!kombi.contains(String.valueOf(IDbarang[i]))){
                return false;
            }
        }

        return true;
    }
    public double CalculateandGetThreshold(){
        double total = 0;
        double count = 0;
        for(int i = 0; i < this.confidence.length; i++){
            total += this.confidence[i];
        }
        //Calculate how much item is != 0
        for(int i = 0; i < this.confidence.length; i++){
            if(confidence[i] != 0.0){
                count++;
            }
        }
        total = total / (double) count;
        this.treshold = total;
        return treshold;
    }

    public void Recommend(){
        treshold = CalculateandGetThreshold();
        for(int i = 0; i < this.confidence.length; i++){
            if(confidence[i] > treshold){
                System.out.println("Merekomendasikan Membeli " + this.toko.searchBarangByID(i+1));
            }
        }
    }

    private void InitializeConfidence(AssociationRules newRule){
        int[] confidencefactor = newRule.getConfidenceFactor();
        double[] confidence = newRule.getConfidence();
        int jmlBarang = confidence.length;
        for(int i = 0; i < jmlBarang ; i++){
            confidencefactor[i] = 0;
            confidence[i] = 0.0;
        }
    }
}
```

3.5. Class Main

Class main merupakan class yang akan memanggil method dan objek dari class lain yang akan diproses di main method, ia akan melakukan semua pemanggilan dan pemilihan proses yang akan dilakukan sesuai input. Pada class main, terdapat juga fitur Input dari CSV (Comma separated values)

yang dapat kita jadikan input “Transaksi” yang akan menambahkan Support dari masing-masing barang yang ada.

TestMain.java

```
package store;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Random;
import java.util.Scanner;
import static store.AssociationRules.CreateAssociationRules;
public class TestMain {
   public static <T> ArrayList<T> removeDuplicates(ArrayList<T> list)
   {
      // Create a new ArrayList
      ArrayList<T> newList = new ArrayList<T>();
      // Traverse through the first list
      for (T element : list) {
         // If this element is not present in newList
         // then add it
         if (!newList.contains(element)) {
            newList.add(element);
         }
      }
      // return the new list
      return newList;
   }
   public static void makeRandomTransactionToCSV(int jmlPembelian, Toko toko) throws
IOException{
      // Our example data
      Random rand = new Random();
      int IDbarangRandom;
      int jmlBrgdiRandom;
      int jmlBrgdiToko = toko.GetNumberofBarang();
      FileWriter csvWriter = new FileWriter("ListBarang.csv");
      for (int i=1;i<=jmlPembelian;i++){
         jmlBrgdiRandom = (rand.nextInt(jmlBrgdiToko)%jmlBrgdiToko)+1;
         // Jml barang yg dibeli di random juga karena tidak akan selalu membeli semuanya
sekaligus.

         for(int j=1;j<=jmlBrgdiRandom;j++){
            IDbarangRandom=(rand.nextInt(jmlBrgdiToko)%jmlBrgdiToko)+1; //+1 karena ID
barang dimulai dari 1, agar hasil minimal = 0
            csvWriter.append(toko.searchBarangByID(IDbarangRandom)); //input Nama barang
sesuai ID yang dirandom tadi
            csvWriter.append(",");
```

```
            }
            csvWriter.append("\n");
        }
        csvWriter.flush();
        csvWriter.close();
    }
    public static void inputTransactionFromCSV(ArrayList<String> listpembelian, Toko myToko){
        String line = "";
        String splitBy = ",";
        int i = 0;
        try {
            //parsing a CSV file into BufferedReader class constructor
            BufferedReader br = new BufferedReader(new FileReader("ListBarang.csv"));
            System.out.println("\nHasil:");
            while ((line = br.readLine()) != null) //returns a Boolean value
            {
                i = i + 1;
                String[] employee = line.split(splitBy); // use comma as separator
                for(int j=0;j<employee.length;j++){
                    listpembelian.add(employee[j]);
                }
                listpembelian = removeDuplicates(listpembelian);
                myToko.getData().IncreaseSupportByOne(myToko.Beli(listpembelian));
                listpembelian.clear();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public static int mainMenu(){
        System.out.println("--Market Basket Analysis Simulation--");
        System.out.println("--Made by Kelompok 4--");
        System.out.println();
        System.out.println("Main Menu");
        System.out.println("1. Buat random sample data, lalu hitung supportnya. (defaultnya telah
dibuat, menu ini untuk menambah sample data)");
        System.out.println("2. Input Barang, Lalu Analisa Confidence Ke Barang Lain");
        System.out.println("3. Tampilkan Kombinasi Barang (Itemset) dan Supportnya");
        System.out.println("0. Keluar");
        Scanner input = new Scanner(System.in);
        System.out.print("Pilih : ");
        return input.nextInt();
    }
    public static void mainMenuSwitch(int menu, AssociationRules newRule) throws IOException{
        int temp;
        int i;
        int inputBrgLagi=1;
        int inputBrg;
        ArrayList<Integer> listInputBrg = new ArrayList<Integer>();
        Scanner input = new Scanner(System.in);
```

```
      switch(menu){
         case 1 :
            System.out.print("Banyak Transaksi : ");
            temp = input.nextInt();
            makeRandomTransactionToCSV(temp, newRule.getToko());
            break;
         case 2 :
            System.out.println("List Barang : ");
            i = 1;
            for(Barang b : newRule.getToko().GetAllBarang()){
               System.out.println(i + ". " + b.GetName());
               i++;
            }
            while(inputBrgLagi == 1 && listInputBrg.size() < 4){
               System.out.print("Input No. Barang (barang duplikasi otomatis dihapus, max 3
barang): ");
               inputBrg = input.nextInt();
               listInputBrg.add(inputBrg);
               System.out.print("Input Barang Lagi? (1=Lg/0=Tdk): ");
               inputBrgLagi = input.nextInt();
            }
            listInputBrg = removeDuplicates(listInputBrg);
            int[] arr = new int[listInputBrg.size()];
            int index = 0;
            for (final Integer value: listInputBrg) {
              arr[index] = value;
              System.out.println(arr[index]);
              index++;
            }
            double[] conf = newRule.CalculateConfidence(newRule.getToko().getData(), arr);
            for (i=0;i<conf.length;i++){
               System.out.println(i+1 + ".");
               System.out.println("Confidence Factor to : " +
newRule.getToko().searchBarangByID(i+1) + " is : " + newRule.getConfidenceFactor()[i]);
               System.out.println("Confidence to : " + newRule.getToko().searchBarangByID(i+1) +
" is : " + conf[i]);
            }
            System.out.println();
            System.out.println("Treshold : " + newRule.CalculateandGetThreshold());
            System.out.println("Rekomendasi : ");
            newRule.Recommend();
            break;
         case 3 :
            newRule.printAllData();
            break;
         case 0 :
            break;
      }
   }
   public static void main(String[] args) throws IOException {
```

```
        int menu = 1;
        ArrayList<String> listbrgdiToko = new ArrayList<>();
        ArrayList<String> listpembelian = new ArrayList<>();
        listbrgdiToko.add("laptop");
        listbrgdiToko.add("mouse");
        listbrgdiToko.add("keyboard");
        listbrgdiToko.add("headset");

        /* Buat object toko, dengan list barang yang telah dibuat diatas */
        Toko myToko = Toko.CreateToko(listbrgdiToko);

        /* Generate random transaction ke dalam csv , lalu ambil datanya */
        makeRandomTransactionToCSV(50, myToko);
        inputTransactionFromCSV(listpembelian,myToko);  //Disini, support sudah terhitung.

        /* initialisasi A0ssociation Rules utk menghitung confidence */
        AssociationRules newRule = CreateAssociationRules(listbrgdiToko.size(), myToko);

        while(menu != 0){
           menu = mainMenu();
           mainMenuSwitch(menu, newRule);
        }
        newRule.printAllData();
    }
}
```

Pada TestMain.java, kami mengambil beberapa fitur dari internet yang dimodifikasi, seperti method inputTransactionFromCSV => sumber : http://naura-lab.blogspot.com/2015/02/parsing-file-csv-di-java.html, yang awalnya hanya menginput nilai ID barang (berupa integer), kami buat menjadi String(nama barang) sehingga memudahkan ketika input data untuk menambahkan Supportnya.

4. BATASAN PROGRAM

4.1. Batasan Input

Program MBA kami memiliki batasan dalam input, yaitu :

1. Input barang yang dibeli tidak boleh duplikasi (ArrayList<String> listBarang isinya tidak boleh ada yang sama, misal : “Laptop”,”Keyboard”,”Laptop”) karena proses selanjutnya adalah pembuatan kombinasi dari barang tersebut , dalam pemisalan menjadi 1,2,1 lalu akan ada kombinasi {1,1} yang tidak ada dalam array of kombinasi didalam Class Data. Sehingga solusinya, kami membuat “Remove Duplicate” untk ArrayList yang menampung list barang yang dibeli.
2. Input CSV kami bataskan sekali input menjadi 50 transaksi, karena data yang terlalu banyak dalam satu waktu akan memperlambat kerja program, berhubung karena program ini dibuat untuk kasus sederhana, kami menghindari data yang terlalu banyak.
3. Jumlah barang di toko rekomendasi 4, minimal 3 (karena apabila kurang dari 3 tidak akan akurat) dan apabila terlalu banyak, jumlah data dan kombinasi akan meningkat secara eksponensial dan membuat program menjadi lambat.

4.2. Batasan Fitur

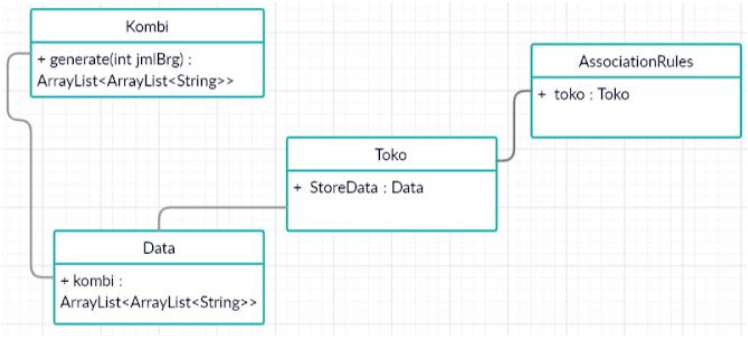
Pada penjelasan Desain Aplikasi (BAB 1), terdapat 3 persamaan di bagian Market Based Analysis. Persamaan satu yaitu support (A ke B) yang kami definisikan sebagai “Confidence Factor” sama dengan Support(AUB). Persamaan kedua merupakan Confidence = ConfidenceFactor dibagi SupportFactor. Program belum bisa membuat persamaan ketiga, karena dirasa belum terlalu membutuhkan persamaan tersebut dan belum bisa menemukan faktor-faktor conditional dan result dari transaksi yang sangat sederhana ini.

5. ALUR PROGRAM MENYELESAIKAN MASALAH

5.1. Membuat Kombinasi

Kombinasi dibuat di Class Kombi, dimana ada beberapa case yaitu :

a. Pembuatan kombinasi dari seluruh barang, dilakukan di method generate (Pada Class Kombi), yang akan disimpan di DataToko (Class Data) dalam Objek Class Toko



Gambar 5.1 Alur pembuatan Kombi dan penampungannya

b. Pembuatan kombinasi dari barang yang dibeli saja. Dilakukan di method getCombiFromInput di Class Kombi yang akan dipanggil oleh method “Beli” di Class Toko yang akan dipanggil oleh method “IncreaseSupportByOne” di Instansi Data dalam Class toko.

5.2. Menghitung Support

Support dihitung dari hasil kombinasi barang yang dibuat oleh case b di **5.1**, barang yang dibeli akan dibuat kombinasinya (Misal : barang 1,2, dan 3 akan menjadi {1},{2}.{3}.{1,2,3},dst). Lalu, akan dikirimkan ke method “IncreaseSupportByOne” yang akan mengecek apakah ada unsur kombinasi dalam data, apabila ada, maka value yang ada akan direplace oleh value awal + 1. (Contoh, menambah value support dari itemset {1,2,3} dengan mengecek kombi di Data (if kombi.contains(itemsetA)), jika benar maka akan ditambah 1 dari value awalnya.

5.3. Menentukan Confidence

Confidence dihitung dengan membagi ConfidenceFactor (Support(AUB)) dengan SupportFactor(Support(A)+Support(B)) yang dilakukan di method CalculateConfidence di Class AssociationRules, dengan rumus :

$$Confidence = \frac{ConfidenceFactor}{SupportFactor}$$

Confidence akan ditampilkan di layar sebagai patokan untuk rekomendasi

5.4. Menentukan Treshold dan Recommendation

Treshold akan dibuat dengan mengambil rata rata nilai Confidence yang lebih dari 0. Sehingga apabila ada Confidence terhadap benda lain yang lebih dari rata rata, ia akan langsung direkomendasikan (ditampilkan dilayar sebagai rekomendasi).

6. TEST PROGRAM

6.1 Menggunakan Data Tertanam di program.

6.1.1 Data Terdefinisi

```
public static void main(String[] args) throws IOException {
    int menu = 1;
    ArrayList<String> listbrgdiToko = new ArrayList<>();
    ArrayList<String> listpembelian = new ArrayList<>();
    listbrgdiToko.add("laptop");
    listbrgdiToko.add("mouse");
    listbrgdiToko.add("keyboard");
    listbrgdiToko.add("headset");

    /* Buat object toko, dengan list barang yang telah dibuat diatas */
    Toko myToko = Toko.CreateToko(listbrgdiToko);


    listpembelian.add("laptop");
    listpembelian.add("mouse");
    listpembelian.add("keyboard");
    myToko.getData().IncreaseSupportByOne(myToko.Beli(listpembelian));
```

Pada gambar terlihat bahwa list barang yang berada ditoko adalah laptop,mouse,keyboard,dan headset. Lalu ada satu transaksi laptop,mouse,keyboard. Hal tersebut akan menambahkan support dari kombinasi barang (itemset) dengan 1.

6.1.2 Output

```
run:
[1] Support : 1
[2] Support : 1
[3] Support : 1
[4] Support : 0
[1, 2] Support : 1
[1, 3] Support : 1
[1, 4] Support : 0
[2, 3] Support : 1
[2, 4] Support : 0
[3, 4] Support : 0
[1, 2, 3] Support : 1
[1, 2, 4] Support : 0
[1, 3, 4] Support : 0
[2, 3, 4] Support : 0
[1, 2, 3, 4] Support : 0
1.
Confidence Factor to : laptop is : 0
Confidence to : laptop is : 0.0
2.
Confidence Factor to : mouse is : 0
Confidence to : mouse is : 0.0
3.
Confidence Factor to : keyboard is : 1
Confidence to : keyboard is : 0.5
4.
Confidence Factor to : headset is : 0
Confidence to : headset is : 0.0

Treshold : 0.5
Rekomendasi :
Merekomendasikan Membeli keyboard
BUILD SUCCESSFUL (total time: 0 seconds)
```

Pada gambar terlihat kombinasi barang / itemset yang telah memiliki support lalu dihitung confidencenya menurut barang {1,2}. {1,2} akan memberikan confidence ke {1,2,3} sebanyak 1 sebagai ConfidenceFactor, dan memberikan support {1} + {2} yaitu 2. Sehingga Confidence = ConfidenceFactor / support, 1/2= 0,5. (hasil valid).

6.2. Menggunakan Data Random yang digenerate kedalam CSV (Comma Separated Value).

6.2.1 Data Terdefinisi :

```
public static void main(String[] args) throws IOException {
    int menu = 1;
    ArrayList<String> listbrgdiToko = new ArrayList<>();
    ArrayList<String> listpembelian = new ArrayList<>();
    listbrgdiToko.add("laptop");
    listbrgdiToko.add("mouse");
    listbrgdiToko.add("keyboard");
    listbrgdiToko.add("headset");

    /* Buat object toko, dengan list barang yang telah dibuat diatas */
    Toko myToko = Toko.CreateToko(listbrgdiToko);

    /* Generate random transaction ke dalam csv , lalu ambil datanya */
    makeRandomTransactionToCSV(50, myToko);
    inputTransactionFromCSV(listpembelian,myToko);  //Disini, support sudah terhitung.

    /* initialisasi Association Rules utk menghitung confidence */
    AssociationRules newRule = CreateAssociationRules(listbrgdiToko.size(), myToko);
```

Dimana method makeRandomTransactionToCSV :

```
public static void makeRandomTransactionToCSV(int jmlPembelian, Toko toko) throws IOException{
    // Our example data

    Random rand = new Random();
    int IDbarangRandom;
    int jmlBrgdiRandom;
    int jmlBrgdiToko = toko.GetNumberofBarang();
    FileWriter csvWriter = new FileWriter("ListBarang.csv");

    for (int i=1;i<=jmlPembelian;i++){
        jmlBrgdiRandom = (rand.nextInt(jmlBrgdiToko)%jmlBrgdiToko)+1;
        // Jml barang yg dibeli di random juga karena tidak akan selalu membeli semuanya sekaligus.

        for(int j=1;j<=jmlBrgdiRandom;j++){
            IDbarangRandom=(rand.nextInt(jmlBrgdiToko)%jmlBrgdiToko)+1; //+1 karena ID barang dimulai dari 1, agar hasil minimal = 0
            csvWriter.append(toko.searchBarangByID(IDbarangRandom)); //input Nama barang sesuai ID yang dirandom tadi
            csvWriter.append(",");
        }
        csvWriter.append("\n");
    }
    csvWriter.flush();
    csvWriter.close();
}
```

dan Method inputTransactionFromCSV :

```
public static void inputTransactionFromCSV(ArrayList<String> listpembelian, Toko myToko){
    String line = "";
    String splitBy = ",";
    int i = 0;
    try {
        //parsing a CSV file into BufferedReader class constructor
        BufferedReader br = new BufferedReader(new FileReader("ListBarang.csv"));
        System.out.println("\nHasil:");
        while ((line = br.readLine()) != null) //returns a Boolean value
        {
            i = i + 1;
            String[] employee = line.split(splitBy); // use comma as separator
            for(int j=0;j<employee.length;j++){
                listpembelian.add(employee[j]);
            }
            listpembelian = removeDuplicates(listpembelian);
            myToko.getData().IncreaseSupportByOne(myToko.Beli(listpembelian));
            listpembelian.clear();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Method pertama akan men-generate kombinasi barang barang sebanyak 50 (terdefinisi di main program) lalu method kedua akan membacanya dan menambahkan support tiap tiap transaksinya (Setelah menghapus transaksi yang duplikat).

6.2.2 Output Program

a. Itemset dan Supportnya.

```
[1] Support : 24
[2] Support : 26
[3] Support : 24
[4] Support : 23
[1, 2] Support : 11
[1, 3] Support : 12
[1, 4] Support : 9
[2, 3] Support : 11
[2, 4] Support : 9
[3, 4] Support : 12
[1, 2, 3] Support : 6
[1, 2, 4] Support : 4
[1, 3, 4] Support : 3
[2, 3, 4] Support : 6
[1, 2, 3, 4] Support : 2
```

b. Confidence, Confidence Factor, Treshold dan rekomendasi terhadap itemset {1.2}

```
1.
Confidence Factor to : laptop is : 0
Confidence to : laptop is : 0.0
2.
Confidence Factor to : mouse is : 0
Confidence to : mouse is : 0.0
3.
Confidence Factor to : keyboard is : 8
Confidence to : keyboard is : 0.16
4.
Confidence Factor to : headset is : 6
Confidence to : headset is : 0.12

Treshold : 0.14
Rekomendasi :
Merekomendasikan Membeli keyboard
```

Perhitungan :

ConfidenceFactor

Itemset {1,2} memiliki ConfidenceFactor yaitu Support({1,2,3}) + Support({1,2,3,4}) utk barang 3 (Keyboard) jadi 6+2 = 8. (valid)

Itemset {1,2} memiliki ConfidenceFactor yaitu Support({1,2,4}) + Support({1,2,3,4}) utk barang 4 (Keyboard) jadi 4+2 = 6. (valid)

SupportFactor

Itemset {1,2} memiliki support factor yaitu 24+26 = 50.

Confidence

Itemset {1,2} terhadap item {3} memiliki confidence :

8/50 = 0.16 (valid)

Itemset {1,2} terhadap item {4} memiliki confidence :

6/50 = 0.12 (valid)

Proses Pengerjaan

Log Book

Tanggal Pengerjaan	Keterangan	Status
2 Juli 2020	Perancangan konsep program, apa saja yang harus dipersiapkan, dan class dasar yang harus dibuat dengan atribut-nya	Selesai
4 Juli 2020	Pembuatan Class Awal, yaitu Class Toko, Class Data, Class Main, dan Class DataAssociation	Belum Selesai
5 Juli 2020	Class Toko Selesai Class Data Selesai	Selesai
7 Juli 2020	Class Data Association telah memiliki fungsi untuk menghitung Support berdasarkan SATU Barang input (yang dibeli).	Selesai
7 Juli 2020	Pembuatan list Barang Random, dengan barang maksimal 4 Barang. dibuat dalam file Excel	Selesai
8 Juli 2020	membuat Class NewExcel yang digunakan untuk menginputkan data data yang telah disimpan pada satu file Excel, sehingga output daripada class tersebut berupa data dari file tersebut	Selesai
11 Juli 2020	Class Data Association telah memiliki fungsi untuk menghitung support lebih dari satu barang input (Misalnya input barang A dan B, Lalu menghitung supportnya ke Barang C dan D).	Selesai

12 Juli 2020	Class Data Association di refaktor agar menjadi lebih clean code	Selesai
16 Juli 2020	Class Data Association dapat menghitung Confidence berdasarkan jumlah support gabungan (AUB) dari support mandiri (A, B).	Belum Selesai
18 Juli 2020	Class Data Association Memiliki fitur perhitungan confidence	Selesai
19 Juli 2020	Class Data Association Memiliki fitur perhitungan Treshold untuk keperluan rekomendasi, menambahkan fitur random data generator untuk sample data otomatis berupa CSV	Selesai.