BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Lingkungan Implementasi

Implementasi perangkat lunak ini adalah rekomendasi pembelian pada *web Commerce* dengan menggunakan metode Matrix and Interestingness based Association Rule Mining (MIbARM), adapun yang akan dijelaskan dalam subbab ini meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan rekomendasi pembelian ini adalah Netbook Asus EeePC dengan spesifikasi sebagai berikut:

1. Prosesor AMD Dual Core C-50 1GHz
2. Memori 2 Gb
3. Harddisk 320 Gb
4. LCD 12''

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan program rekomendasi pembelian dan uji coba adalah:

1. Sistem operasi Ubuntu 11.10 AMD 64bit
2. Editor Bluefish 2.0.3
3. XAMPP dengan Apache 2.2.21 , PHP 5.3.8 dan MySQL 5.0.8
4. Emulator Opera Mobile 11.10 dan Mozilla Firefox 12.0

4.2 Implementasi Program Rekomendasi MIbARM

Untuk menghasilkan rekomendasi pembelian maka dilakukan implementasi sesuai dengan perancangan sistem yang telah dijelaskan sebelumnya, berikut merupakan implementasi tersebut:

4.2.1 Pembentukan Frequent Itemset

Untuk mendapatkan kandidat *rule* maka dilakukan pembentukan *frequent itemset* terlebih dahulu, fungsi-fungsi yang digunakan adalah sebagai berikut:

4.2.1.1 Implementasi Pembentukan Transaction Matrix

Semua data yang diperlukan seperti data transaksi dan item di *query* terlebih dahulu, kemudian diubah menjadi Transaction Matrix (*T*) sesuai dengan persamaan 2.1, fungsi ini dapat dilihat pada tabel 4.1 dan *source code* 4.1:

Tabel 4.1 Fungsi Pembentukan Transaction Matrix

Nama Fungsi	Deskripsi
transactionMatrix()	Fungsi untuk mengubah data transaksi dalam database menjadi bentuk Transaction Matrix *(T)*

```
1   function  transactionMatrix($jumlah_transaksi)
    {
2     switch($jumlah_transaksi) {
3           case 500:
4   $limit=500;
5    break;
6   case 1000:
7   $limit=1000;
8   break;
9       }
10  $get_goods= 'select id,Food from goods';
11  $run_get_goods= mysql_query($get_goods) or die
    (mysql_error());
12  $data_goods=array();
13  $data_foods=array();
14  $j=0;
15  while($data=mysql_fetch_array($run_get_goods))
    {
16        $data_goods[$j]=$data['id'];
17        $data_foods[$j]=$data['Food'];

18        $j++;
19        }
20  $data_foods2=array();
21  for($i=0; $i<count($data_foods); $i++) {
22        for($j=0; $j<count($data_foods); $j++) {
23        if(isset($data_foods[$i])              &&
    isset($data_foods[$j])) {
24              if($i <> $j) {
25
```

```
    if($data_foods[$i]==$data_foods[$j]) {
26              unset($data_foods[$j]);
27          }
28              }
29          }
30      }
31  }
32  $data_foods_reindex=array_values($data_foods);
33  $data_foods=$data_foods_reindex;
34  for($i=0; $i<count($data_foods); $i++) {
35        for($j=0; $j<count($data_foods); $j++) {
                  if(isset($data_foods[$i])     &&
36  isset($data_foods[$j])) {
37                      if($i <> $j) {
38  if($data_foods[$i]==$data_foods[$j]) {
39  unset($data_foods[$j]);
40                                  }
41                              }
42                  }
43              }
44          }
45  $data_foods_reindex=array_values($data_foods);
46  $data_foods=$data_foods_reindex;
47  for($i=0; $i<count($data_foods); $i++) {
48        for($j=0; $j<count($data_foods); $j++) {
              if(isset($data_foods[$i])         &&
49  isset($data_foods[$j])) {
50                  if($i <> $j) {
51  if($data_foods[$i]==$data_foods[$j]) {
52                                  unset($data_foods[$j]);
53                              }
54                  }
55              }
56          }
57      }
58  $data_foods_reindex=array_values($data_foods);
59  $data_foods=$data_foods_reindex;
60  $get_receipts=  'select  receipts.ReceiptNumber
    from receipts limit '.$limit.'';
61  $run_get_receipts=   mysql_query($get_receipts)
```

```
    or die (mysql_error());
62  $data_receipts=array();
63  $i=0;
64  while($data2=mysql_fetch_array($run_get_receipts)){
65
    $data_receipts[$i]=$data2['ReceiptNumber'];
66              $i++;
67             }
68  $count_r=count($data_receipts);
69  $C_TM=array();
70  for($i=0; $i<$count_r; $i++){
71        $detail= 'select item from items where Receipt='.$data_receipts[$i].'';
72        $run_detail= mysql_query($detail) or die (mysql_error());
73        $j=0;
74
    while($result=mysql_fetch_array($run_detail)){
75                  $get_goods=  'select  Food  from goods where Id='.$result['item'].'';
                    $run_get_goods=
76  mysql_query($get_goods)          or          die (mysql_error());
77
    $data=mysql_fetch_array($run_get_goods);
78                  $C_TM[$i][$j]=$data['Food'];
79                  $j++;
80             }
81       }
82  $rows_i=count($data_receipts);
83  $cols_j=count($data_foods);
84  $rows = count($C_TM,0);
85  for($i=0; $i<$rows; $i++) {
86       for($j=0; $j<count($C_TM[$i]); $j++) {
87            for($k=0; $k<$cols_j; $k++) {
88
    if($C_TM[$i][$j]==$data_foods[$k]) {
89                     $C_TM[$i][$j]=$k;
90                     }
91                }
92          }
93     }
```

```
94  $new=array();
95  for($i=0; $i<$rows_i; $i++) {
96       for($j=0; $j<$cols_j; $j++) {
97            $new[$i][$j]=$j;
98            }
99       }
100 $rows = count($new,0);
101 $cols = (count($new,1)/count($new,0))-1;
102 $cols_C_TM=(count($C_TM,1)/count($C_TM,0))-1;
103 for($i=0; $i<$rows; $i++) {
104       for($j=0; $j<$cols; $j++) {
105            for($k=0; $k<$cols; $k++) {
106                 if      (isset($C_TM[$i][$k])){
107                      if($j==$C_TM[$i][$k]) {
108                           $new[$i][$j]=1000;
109                           }
110            }
111       }
112   }
113 }
114 for($i=0; $i<$rows; $i++) {
115            for($j=0; $j<$cols; $j++) {
116                 if($new[$i][$j]==1000) {
117                   $new[$i][$j]=1;
118                 }else {
119                           $new[$i][$j]=0;
120                          }
121                 }
122        }
123 $return=array();
124 $return[0]=$new;
125 $return[1]=$data_foods;
126  return $return;
127 }
```

Source Code 4.1 Fungsi Transaction Matrix

4.2.1.2 Implementasi Pembentukan 1-frequent itemset

Setelah terbentuk Transaction Matrix *(T)*, langkah selanjutnya adalah pembentukan *1-frequent itemset*, fungsi ini dapat dilihat pada table 4.2 dan *source code* 4.2:

Tabel 4.2 Fungsi Pembentukan Frequent 1-Itemset

Nama Fungsi	Deskripsi
oneItemset()	Fungsi utama pembentukan *frequent 1-itemset.*
sumJ()	Fungsi untuk menghitung Absolute Support *(SupA)* setiap kandidat *itemset*, sesuai persaman 2.3.
supJ()	Fungsi untuk menghitung Support *(Sup)* setiap kandidat *itemset*, sesuai persamaan 2.4
remove()	Fungsi untuk menghilangkan kandidat *itemset* yang tidak memenuhi Support minimal, sesuai persamaan 2.5.
sumI()	Fungsi untuk menghitung jumlah pembelian item setiap baris dalam matriks (*T*).
removeI()	Fungsi untuk menghilangkan baris dalam matriks (*T*) yang jumlahnya kurang dari 2.

```
1  function oneItemset($c,$setToArray,$k_index){
2  $sup_min=$setToArray[0];
3  $conf_min=$setToArray[1];
4  $i_min=$setToArray[2];
5  $jumlah_transaksi=$setToArray[3];
6  $rows = count($c,0);
7  $cols = (count($c,1)/count($c,0))-1;
8  $jumlah_c=$rows;
9  $totalT=$rows;
10 $sum_j=sumJ($c,$rows,$cols);
11 $c1=removeJ($c,$sum_j,$sup_min,$totalT,$rows,$
   cols);
12 $sum_j_f=array_filter(sumJ($c1,$rows,$cols));
13 $sup_f=supJ($sum_j_f,$totalT,$rows,$cols);
14 $L1=$c1;
15 $sum_akhir=sumI($c1,$rows);
16 $c1_final=removeI($sum_akhir,$c1,$rows,$k_inde
   x);
```

```
17  $c1_final_reindex=array_values($c1_final);
18  $c1_final=$c1_final_reindex;
19  $hasil=array();
20  $hasil[0]=$sum_j_f;
21  $hasil[1]=$c1_final;
22  return $hasil;
23  }
```

Source Code 4.2 Fungsi Pembentukan Frequent 1-Itemset

4.2.1.3 Implementasi Pembentukan 2-Frequent Itemset

Setelah didapatkan *frequent 1-itemset*, selanjutnya dibentuk kandidat *2-itemset* untuk mendapatkan *frequent 2-itemset*. Beberapa fungsi utama yang digunakan dapat dilihat pada table 4.3 dan *source code* 4.3 dan *source code* 4.4.

Tabel 4.3 Fungsi Pembentukan Frequent 2-Itemset

Nama Fungsi	Deskripsi
hermiteMatrix()	Fungsi untuk membentuk Hermite *matrix (H)* untuk membentuk kandidat *2-Itemset*, sesuai dengan persamaan 2.6.
gMatrix()	Fungsi untuk membentuk matriks *(G)* untuk mendapatkan *frequent 2-itemset*, sesuai dengan persaman 2.7.

```
    function
1   hermiteMatrix($c1_final,$cols,$M1,$sum_j_final)
    {
2   $t=$c1_final;
3   $t2=$c1_final;
4   $h=array();
5   for($j=0; $j< $cols; $j++) {
6      for($l=0; $l<$cols; $l++) {
7         $hasil=0;
8         for($k=0; $k<$M1; $k++) {
              if(isset($t2[$k][$j])) {
9
10                if(isset($t[$k][$l])) {
11                  if ($l <> $j){
```

```
                    $hasil    +=($t[$k][$j])    *
($t2[$k][$l]);
                  }
                }
              else {
                  $hasil = null;
                }
              }
            }
          if ($j==$l) {
             $hasil =$sum_j_final[$j];
          }
     $h[$j][$l]=$hasil;
     }
   }
for($i=0; $i<$cols; $i++) {
     $c_h=array_sum($h[$i]);
      if($c_h == 0) {
         unset($h[$i]);
        }
     }
return $h;
}
```

Source Code 4.3 Fungsi Pembentukan Matriks H

```
function gMatrix($h,$rows,$cols,$sup_min) {
$rows_h=count($h,0);
$g=array();
$total_t=$rows;
$sup_h=array();
$sup_L2=array();
$L2=array();
for($i=0; $i<$cols; $i++){
   for($j=0; $j<$cols; $j++){
     if(isset($h[$i][$j])){
       $sup_h[$i][$j]= $h[$i][$j] /$total_t;
       if($i==$j) {
          $g[$i][$j]=0;
          } else {
                  if($sup_h[$i][$j] < $sup_min)
{
                     $g[$i][$j]=0;
```

```
17                          }else {
18                                  $g[$i][$j]=1;
19  $L2[$i][$j]=$g[$i][$j];
20  $sup_L2[$i][$j]=$h[$i][$j];
21                          }
22                      }
23                  }
24              }
25          }
26  $hasil=array();
27  $hasil[0]=$g;
28  $hasil[1]=$L2;
29  $hasil[2]=$sup_L2;
30  return $hasil;
31  }
```

Source Code 4.4 Fungsi Pembentukan Matrik G

4.2.1.4 Implementasi Pembentukan (k>2)-Frequent Itemset

Beberapa fungsi yang digunakan untuk pembentukan kandidat Frequent (k>2)-Itemset dapat dilihat pada tabel 4.4.

Tabel 4.4 Fungsi Pembentukan k>2 Itemset

Nama Fungsi	Deskripsi
makeCk()	Fungsi utama pembentukan kandidat *(k>2)-itemset*, *source code* 4.5.
W(),WFill()	Fungsi pembentukan matriks *w* sesuai dengan persamaan 2.10, *source code* 4.6.
FW()	Fungsi pembentukan matriks *F(w)* sesuai dengan persamaan 2.11, *source code* 4.7.
V()	Fungsi pembentukan matriks *v* untuk mendapatkan *frequent K>2 itemset*, sesuai dengan persamaan 2.13, *source Code* 4.8.
rebuildT()	Fungsi pembentukan matriks *T* setelah *frequent itemset*

Nama Fungsi	Deskripsi
	terbentuk, *source code* 4.9.
Given_value()	Fungsi pembentukan kandidat *itemset* atau matriks U^k yang akan digunakan pada iterasi berikutnya, sesuai dengan persamaan 2.9, *source code* 4.10.

```
1  function makeCk($k,$totalT,$sup_min,$tk_transpose,$cols,$N,$M,$L,$L_values) {
2  $result=array();
3  $w=W($tk_transpose,$cols,$N,$M,$L,$L_values);
4  $w_values=WFill($tk_transpose,$cols,$N,$M,$L,$L_values);
5  $count_NW = count($w_values,0);
6  $count_MW = (count($w_values,1)/count($w_values,0))-1
7  $FW=FW($k,$cols, $N, $M,$w,$w_values);
8  $v=V($k,$cols, $N, $M,$w,$w_values);
9  $v_fix=filterV($cols, $N, $v, $w,$w_values);
10 $jumlah_ck=count($v_fix);
11 $V_final=VFilterBySup($cols,$totalT, $count_NW, $v_fix, $sup_min);
12 $C_L=Candidat($cols, $count_NW, $V_final,$w);
13 $C_L_supA=CandidatValues($cols,$count_NW, $V_final,$w);
14 $L_final=reindexL($cols,$count_NW,$C_L);
15 $L_final_supA=reindexL_supA($cols,$count_NW,$C_L_supA);
16 $L_final_reindex=array_values($L_final);
17 $L_final=$L_final_reindex;
18 $L_final_supA_reindex=array_values($L_final_supA);
19 $L_final_supA=$L_final_supA_reindex;
20 $result[0]=$L_final;
21 $result[1]=$L_final_supA;
22 $result[2]=$jumlah_ck;
```

```
return $result;
}
```

Source Code 4.5 Fungsi Pembentukan Frequent K>2 Itemset

```
function  W($Ttranspose,$cols,$N,$M,  $Larray,
$Larray2) {
$count_NL = count($Larray2,0);
$count_ML                                    =
(count($Larray2,1)/count($Larray2,0))-1;
$w=array();
for($a=0; $a<$count_NL; $a++) {
    for($i=0; $i<$cols; $i++) {
        if(isset($Ttranspose[$i])) {
          $w[$a][$i]= givenExp($Larray[$a],$i);
          }
        }
}
return $w;
}
Function        WFill($Ttranspose,$cols,$N,$M,
$Larray
,$Larray2) {
$count_NL = count($Larray2,0);
$count_ML                                    =
(count($Larray2,1)/count($Larray2,0))-1;
$w_values=array();
for($a=0; $a<$count_NL; $a++) {
    for($i=0; $i<$cols; $i++) {
    $jumlah=0;
    for($j=0; $j<$N; $j++) {
       if(isset($Ttranspose[$i][$j])) {
          $jumlah    +=$Ttranspose[$i][$j]    *
$Larray2[$a][$j];

$jumlah.'='.$Ttranspose[$i][$j].'*'.$Larray2[$
a][$j].'</br>';
        }
     }
  if(isset($Ttranspose[$i])) {
    $w_values[$a][$i]=$jumlah;
    }
   }
```

```
 }
return $w_values;
}
```

Source Code 6 Fungsi Pembentukan Matriks *W*

```
function FW($k,$cols, $N, $M,$w,$w_values) {
$count_NW = count($w_values,0);
$count_MW =
 (count($w_values,1)/count($w_values,0))-1;
$FW=array();
for($i=0; $i<$count_NW; $i++) {
    for($j=0; $j<$cols; $j++) {
       if(isset($w_values[$i][$j])) {
        $explode1= explode(',', $w[$i][$j]);
        $count_explode1=count($explode1);
        $cek=array_unique($explode1);
         $count_cek=count($cek);
        if($w_values[$i][$j] >= $k ) {
           if($count_explode1==$count_cek) {
             $FW[$i][$j]=1;
              }else {
                  $FW[$i][$j]=0;
                   }
            }else {
                 $FW[$i][$j]=0;
                 }
          }
       }
    }
return $FW;
}
```

Source Code 4.7 Pembentukan Matriks *F(W)*

```
function V($k,$cols, $N, $M,$w,$w_values) {
$count_NW = count($w_values,0);
$count_MW                                    =
(count($w_values,1)/count($w_values,0))-1;
$v=array();
for($i=0; $i<$count_NW; $i++) {
    for($j=0; $j<$cols; $j++) {
       if(isset($w_values[$i][$j])) {
         $explode1= explode(',', $w[$i][$j]);
         $count_explode1=count($explode1);
         $cek=array_unique($explode1);
```

```
                    $count_cek=count($cek);
            if($w_values[$i][$j]>=$k) {
11
    if($count_explode1==$count_cek) {
                    $v[$i][$j]=$w_values[$i][$j];
12
13                  }
14              }
15          }
16      }
17  }
18  return $v;
19  }
```

Source Code 4.8 Fungsi Pembentukan Matriks *v*

```
    function
1   rebuildT($T,$M,$cols,$totalT,$min_length) {

2   $sum_j=sumJ($T,$M,$cols);
3   $sum_i=sumI($T,$M);
4   $sup_final=supJ($sum_j,$totalT,$M,$cols);
5   $T_final=removeI($sum_i,$T,$M,$min_length);
6   $T_final_reindex=array_values($T_final);
7   $T_final=$T_final_reindex;
8   return $T_final;
9   }
```

Source Code 4.9 Fungsi Pembentukan Kembali Matrik *T.*

```
 1  function Given_value($Larray,$Tarray, $M) {
 2  $L_filled=array();
 3  for($a=0; $a< count($Larray); $a++) {
 4      $explode1= explode(',', $Larray[$a]);
 5      for($k=0; $k<$M; $k++) {
            $L_filled[$a][$k]=
 6  cek_isset($explode1,$Tarray,$k);
 7          }
 8      }
```

```
 9  return $L_filled;
10  }
```

Source Code 4.10 Fungsi Pembentukan Kandidat Itemset.

4.2.2 Implementasi Pembentukan Rule

Setelah didapatkan semua kemungkinan *frequent itemset*, kemudian dilakukan pembentukan *rule* berdasarkan Confidence dan Interestingness dari setiap kandidat *rule*, fungsi ini dapat dilihat pada tabel 4.5.

Tabel 4.5 Fungsi Pembentukan Rule

Nama Fungsi	Deskripsi
I()	Fungsi Perhitungan Interestingness Rule sesuai dengan persamaan 2.14. *Source code* 4.11.
Conf()	Fungsi Perhitungan Confidence *rule* sesuai dengan persamaan 2.16. *Source code* 4.12.

```
1   function I($b,$supALL,$totalT,$sum_j_f) {
2   $I=array();
3   $count_b=count($b,0);
4   for($i=0; $i<$count_b; $i++) {
5       $explode=explode(',',$b[$i]);
6       $count_explode=count($explode);
7       if($count_explode==2) {
          $interestingness=$supALL[$i] -
8   (($sum_j_f[$explode[0]]*
    $sum_j_f[$explode[1]])/ $totalT);
9               $I[$i]=round($interestingness,2);
10              }else {
11               for($j=0; $j<$count_b; $j++){
12                   $explode2=explode(',',$b[$j]);
13
    $count_explode2=count($explode2);
14                  $cek=0;
15                  for($k=0; $k<($count_explode-1);
    $k++) {
16                      for($l=0;
```

```
    $l<$count_explode2; $l++) {
                            if($explode[$k]==
17  $explode2[$l]{
18                              $cek +=1;
19                              }else {
20                                      $cek +=0;
21                                      }
22                               }
23                          }
24                  if($cek==$count_explode2) {

    $interestingness=$supALL[$i]      -
25  (( $supALL[$j] *
    $sum_j_f[$explode[$count_explode-1]])/
    $totalT);
                        $I[$i]=
26  round($interestingness,2);

27                  }
28              }
29          }
30      }
31  return $I;
32  }
```

Source Code 4.11 Fungsi Perhitungan Interestingness

```
 1  function conf($b,$supALL,$totalT,$sum_j_f) {
 2  $conf=array();
 3  $count_b=count($b,0);
 4  for($i=0; $i<$count_b; $i++) {
 5     $explode=explode(',',$b[$i]);
 6     $count_explode=count($explode);
 7     if($count_explode==2) {
 8        $confidence=$supALL[$i] /
    $sum_j_f[$explode[0]];
 9        $conf[$i]=round($confidence,2);
10       }
11      if($count_explode>2) {
12        for($j=0; $j<$count_b; $j++) {
13           $explode2=explode(',',$b[$j]);
14           $count_explode2=count($explode2);
15           $cek=0;
16           for($k=0; $k<($count_explode-1); $k++) {
```

```
17                for($l=0; $l<$count_explode2; $l++) {
18                  if($explode[$k]==$explode2[$l]) {
19                    $cek +=1;
20                    }else {
21                          $cek +=0;
22                          }
23                    }
24                  }
25                if($cek==$count_explode2) {
26                  $supa=$supALL[$j];
27                  }
28              if(isset($supa)) {
29                $confidence=$supALL[$i]/$supa;
30                $conf[$i]=round($confidence,2);
31                }
32              }
33           }
34       }
35   return $conf;
36   }
```

Source Code 4.12 Perhitungan Confidence

4.2.3 Implementasi Evaluasi Rule

Untuk melakukan evaluasi terhadap *rule* yang terbentuk digunakan Lift Ratio, fungsi ini dapat dilihat pada tabel 4.6.

Tabel 4.6 Fungsi Lift Ratio

Nama Fungsi	Deskripsi
Lift()	Menghitung nilai Lift Ratio, sesuai dengan persamaan 2.14, *source code* 4.13.

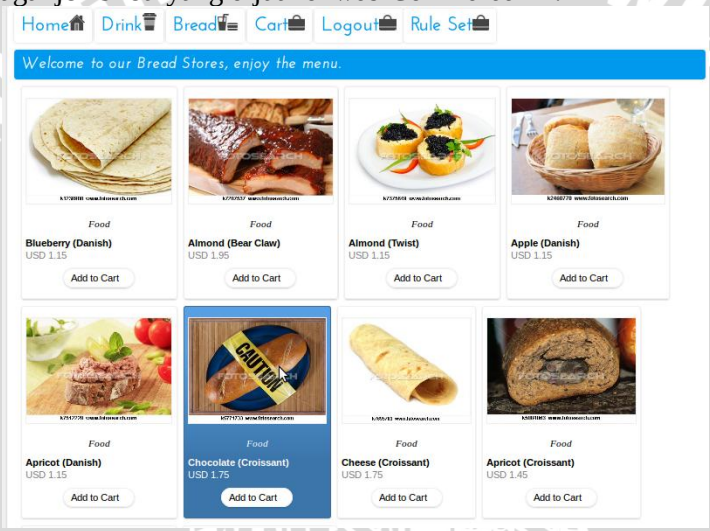
```
1    function
     Lift($b,$Conf,$supALL,$totalT,$sum_j_f) {
2    $lift=array();
3    $count_b=count($b);
4    for($i=0; $i<$count_b; $i++) {
5       if(isset($b[$i])) {
6         $explode= explode(',', $b[$i]);
7         $count_explode=count($explode);
8         $mode=end($explode);
9         $lift_e=
     $Conf[$i]/($sum_j_f[$mode]/$totalT);
```

```
10          $lift[$i]=round($lift_e,2);
11          }
12      }
13  return $lift;
14  }
```

Source Code 4.13 Perhitungan Lift Ratio

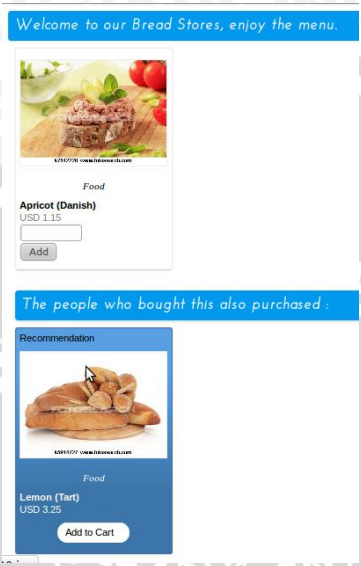
4.2.4 Implementasi Desain Antar Muka

Perancangan desain web kemudian diimplementasikan seperti yang terlihat pada gambar 4.1, dimana pembeli dapat melihat berbagai jenis roti yang dijual di *web Commerce* ini.



Gambar 4.1Tampilan Halaman Utama

Pada Gambar 4.2 merupakan tampilan rekomendasi produk kepada pembeli pada *web Commerce.*



Gambar 4.2 Tampilan Halaman Rekomendasi

Untuk implementasi perancangan halaman pengaturan rekomendasi pembelian dapat dilihat pada gambar

Gambar 4.3 Tampilan Halaman Pengaturan Rekomendasi

4.3 Uji Coba Program

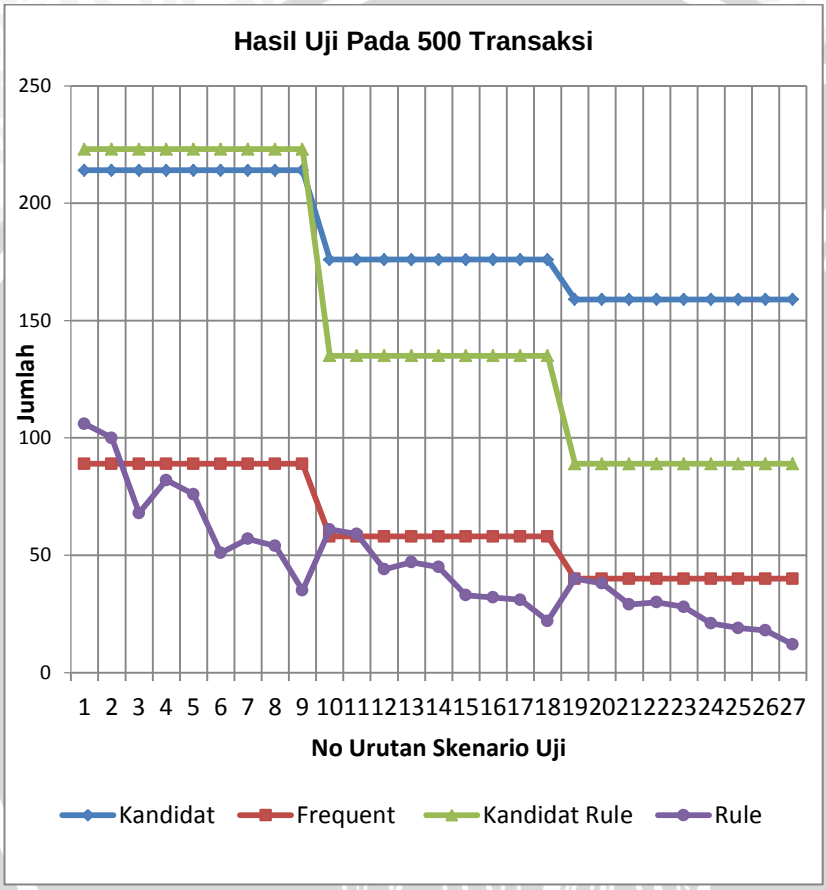
4.3.1 Skenario 500 Transaksi

Pada skenario ini digunakan untuk menguji metode MIbARM menggunakan data transaksi berjumlah 500 Transaksi pembelian, untuk hasil pengujian ini dapat dilihat padat tabel 4.7 dan untuk hasil yang lebih rinci dapat dilihat pada lampiran hasil uji coba pada 500 transaksi.

Tabel 4.7 Hasil Uji Coba Pada 500 Transaksi

N0	S	uji	Sup min	Conf min	Imin	Jumlah Kandidat Itemset	Jumlah Frequent Itemset	Jumlah Kandidat Rule	Jumlah Rule
1	1	1	0.02	0.2	0.5	214	89	223	106
2		2			1	214	89	223	100
3		3			5	214	89	223	68
4		4		0.4	0.5	214	89	223	82
5		5			1	214	89	223	76
6		6			5	214	89	223	51
7		7		0.6	0.5	214	89	223	57
8		8			1	214	89	223	54
9		9			5	214	89	223	35
10	2	1	0.03	0.2	0.5	176	58	135	61
11		2			1	176	58	135	59
12		3			5	176	58	135	44
13		4		0.4	0.5	176	58	135	47
14		5			1	176	58	135	45
15		6			5	176	58	135	33
16		7		0.6	0.5	176	58	135	32
17		8			1	176	58	135	31
18		9			5	176	58	135	22
19	3	1	0.04	0.2	0.5	159	40	89	40
20		2			1	159	40	89	38
21		3			5	159	40	89	29
22		4		0.4	0.5	159	40	89	30
23		5			1	159	40	89	28
24		6			5	159	40	89	21
25		7		0.6	0.5	159	40	89	19
26		8			1	159	40	89	18
27		9			5	159	40	89	12

Line Graph sesuai dengan tabel 4.7 dapat dilihat pada gambar 4.3, dimana sumbu x adalah no urutan skenario uji coba, dan sumbu y adalah skalar nilai dari 0 sampai 250.



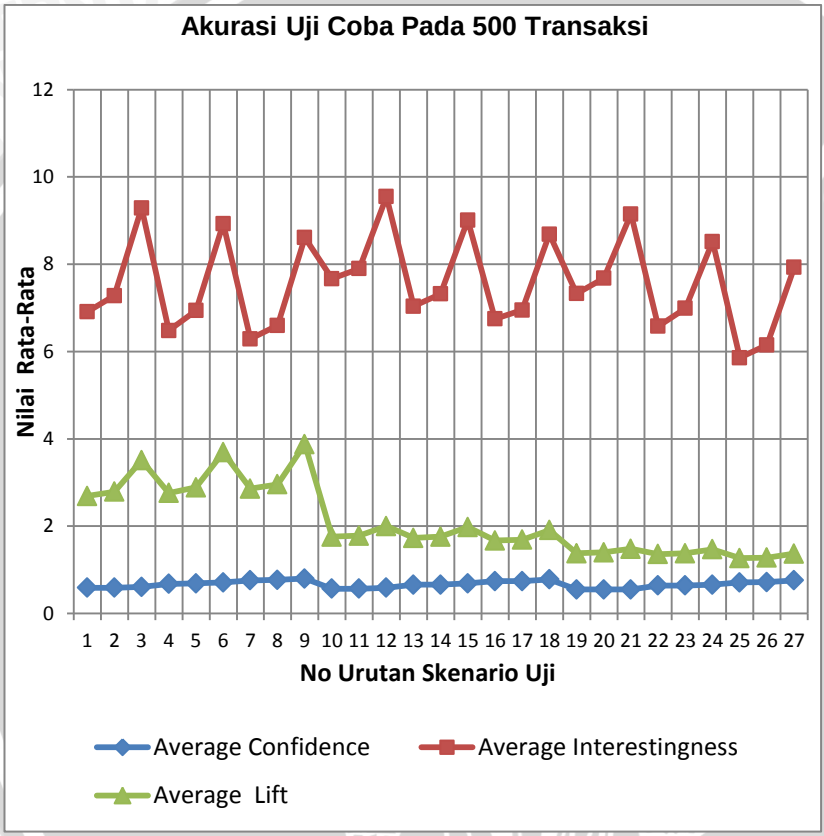
Gambar 4.4 Line Graph Uji Coba Pada 500 Transaksi

Seperti yang terlihat pada gambar 4.3, dapat diketahui bahwa semakin besar minimum Support yang diberikan maka semakin kecil kandidat f*requent itemset* yang terbentuk, sedangkan semakin besar nilai minimum Confidence dan minimum Interestingness maka semakin kecil jumlah *rule* yang dihasilkan. Tabel Akurasi *rule* hasil Pembentukan ini dapat dilihat pada tabel 4.8.

Tabel 4.8 Tabel Akurasi Rule hasil Pembentukan

N0	S	uji	Sup min	Conf min	Imin	Average Confidence	Average Interestingness	Average Lift
1	1	1	0.02	0.2	0.5	0.59	6.91	2.69
2		2			1	0.59	7.28	2.79
3		3			5	0.61	9.29	3.51
4		4		0.4	0.5	0.68	6.48	2.76
5		5			1	0.69	6.94	2.89
6		6			5	0.71	8.93	3.69
7		7		0.6	0.5	0.76	6.29	2.86
8		8			1	0.77	6.6	2.96
9		9			5	0.8	8.61	3.88
10	2	1	0.03	0.2	0.5	0.57	7.67	1.76
11		2			1	0.57	7.9	1.78
12		3			5	0.59	9.55	2
13		4		0.4	0.5	0.66	7.04	1.73
14		5			1	0.66	7.32	1.76
15		6			5	0.69	9.01	1.98
16		7		0.6	0.5	0.74	6.75	1.67
17		8			1	0.74	6.95	1.69
18		9			5	0.78	8.69	1.91
19	3	1	0.04	0.2	0.5	0.55	7.33	1.38
20		2			1	0.55	7.68	1.4
21		3			5	0.55	9.15	1.48
22		4		0.4	0.5	0.64	6.58	1.36
23		5			1	0.64	6.99	1.38
24		6			5	0.66	8.52	1.47
25		7		0.6	0.5	0.71	5.86	1.27
26		8			1	0.72	6.15	1.28
27		9			5	0.76	7.93	1.37

Line Graph sesuai dengan tabel 4.8 dapat dilihat pada gambar 4.4, dimana sumbu x adalah no urutan skenario uji coba dan sumbu y adalah skalar nilai dari 0 sampai 12.



Gambar 4.5 Line Graph Akurasi Hasil Uji Pada 500 Transaksi

Sesuai dengan Gambar 4.4, dapat diketahui bahwa nilai rata-rata Interestingness pada *rule* akan semakin meningkat seiring meningkatnya nilai minimum Interestingness dan minimum Confidence yang diberikan sesuai dengan skenario pada tabel 4.8.

Untuk rata-rata akurasi *rule* dengan Lift Ratio diketahui bahwa

semakin meningkat seiring dengan meningkatkan nilai Interestingness dan Confidence.

Nilai akurasi Lift Ratio tertinggi adalah pada saat skenario 1 uji coba 9, dengan nilai Support minimum 0.02, minimum Confidence 0.6 dan minimum Interestingness 5 dengan jumlah rule 35.

4.3.2 Skenario 1000 Transaksi

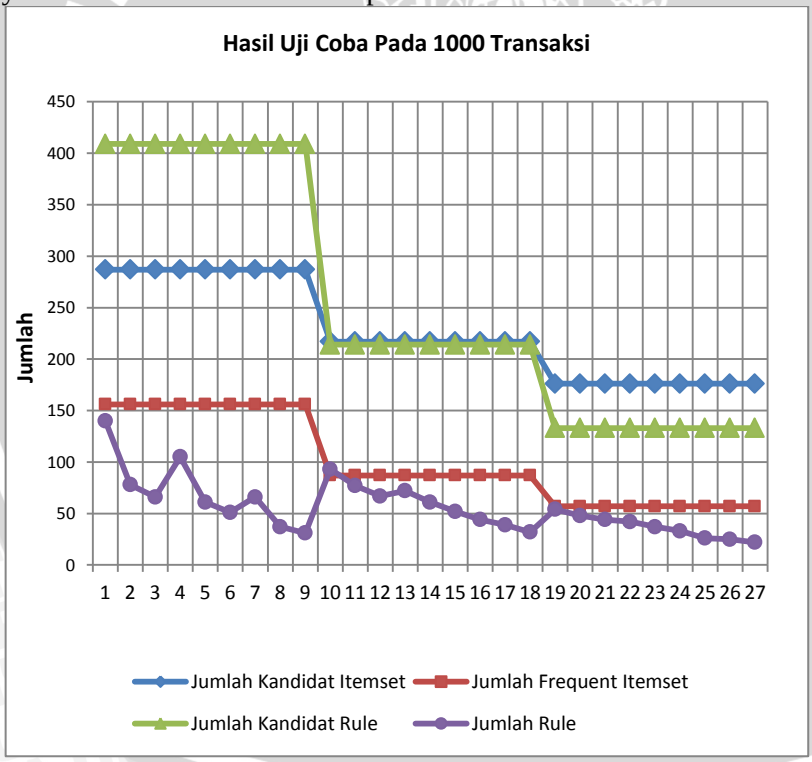
Pada skenario ini digunakan untuk menguji metode MIbARM menggunakan data transaksi berjumlah 1000 Transaksi pembelian, untuk hasil pengujian ini dapat dilihat padat tabel 4.9.

Tabel 4.9 Hasil Uji Coba Pada 1000 Transaksi

N0	S	uji	Sup min	Conf min	Imin	Jumlah Kandidat Itemset	Jumlah Frequent Itemset	Jumlah Kandidat Rule	Jumlah Rule
1	1	1	0.01	0.2	1	287	156	409	140
2		2			5	287	156	409	78
3		3			10	287	156	409	66
4		4		0.4	1	287	156	409	105
5		5			5	287	156	409	61
6		6			10	287	156	409	51
7		7		0.6	1	287	156	409	66
8		8			5	287	156	409	37
9		9			10	287	156	409	31
10	2	1	0.02	0.2	1	217	87	214	93
11		2			5	217	87	214	77
12		3			10	217	87	214	67
13		4		0.4	1	217	87	214	72
14		5			5	217	87	214	61
15		6			10	217	87	214	52
16		7		0.6	1	217	87	214	44
17		8			5	217	87	214	39
18		9			10	217	87	214	32
19	3	1	0.03	0.2	1	176	57	133	54
20		2			5	176	57	133	48

N0	S	uji	Sup min	Conf min	Imin	Jumlah Kandidat Itemset	Jumlah Frequent Itemset	Jumlah Kandidat Rule	Jumlah Rule
21		3			10	176	57	133	44
22		4			1	176	57	133	42
23		5		0.4	5	176	57	133	37
24		6			10	176	57	133	33
25		7			1	176	57	133	26
26		8		0.6	5	176	57	133	25
27		9			10	176	57	133	22

Line Graph sesuai dengan tabel 4.9 dapat dilihat pada gambar 4.5, dimana sumbu x adalah no urutan skenario uji coba, dan sumbu y adalah skalar nilai dari 0 sampai 450.



Gambar 4.6 Line Graph Hasil Uji Coba pada 1000 Transaksi

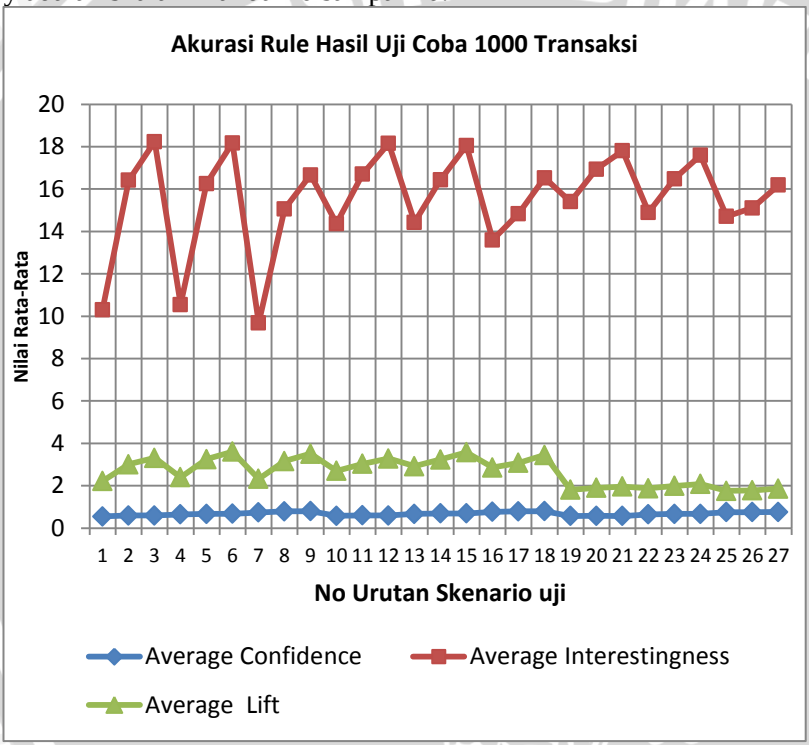
Seperti yang terlihat pada gambar 4.3, dapat diketahui bahwa semakin besar minimum Support yang diberikan maka semakin kecil kandidat *frequent itemset* yang terbentuk, sedangkan semakin besar nilai minimum Confidence dan minimum Interestingness maka semakin kecil jumlah *rule* yang dihasilkan. Tabel akurasi *rule* hasil Pembentukan pada 1000 transaksi dapat dilihat pada tabel 4.10.

Tabel 4.10 Tabel akurasi Rule Pada Uji 1000 Transaksi

N0	S	uji	Sup min	Conf min	Imin	Average Confidence	Average Interestingness	Average Lift
1	1	1	0.01	0.2	1	0.55	10.31	2.22
2		2			5	0.59	16.42	3.01
3		3			10	0.59	18.24	3.31
4		4		0.4	1	0.65	10.55	2.4
5		5			5	0.67	16.26	3.25
6		6			10	0.68	18.18	3.61
7		7		0.6	1	0.74	9.68	2.32
8		8			5	0.79	15.07	3.16
9		9			10	0.8	16.67	3.51
10	2	1	0.02	0.2	1	0.58	14.37	2.7
11		2			5	0.6	16.71	3.03
12		3			10	0.59	18.16	3.28
13		4		0.4	1	0.67	14.43	2.92
14		5			5	0.69	16.44	3.24
15		6			10	0.69	18.06	3.58
16		7		0.6	1	0.77	13.6	2.86
17		8			5	0.79	14.84	3.08
18		9			10	0.8	16.53	3.45
19	3	1	0.03	0.2	1	0.57	15.41	1.81
20		2			5	0.58	16.94	1.9
21		3			10	0.57	17.81	1.96
22		4		0.4	1	0.65	14.9	1.88
23		5			5	0.67	16.49	1.99

N0	S	uji	Sup min	Conf min	Imin	Average Confidence	Average Interestingness	Average Lift
24		6			10	0.67	17.6	2.08
25		7			1	0.75	14.71	1.76
26		8		0.6	5	0.75	15.11	1.78
27		9			10	0.76	16.19	1.86

Line Graph sesuai dengan tabel 4.10 dapat dilihat pada gambar 4.6, dimana sumbu x adalah no urutan skenario uji coba, dan sumbu y adalah skalar nilai dari 0 sampai 20.



Gambar 4.7 Line Graph Akurasi Hasil Uji Pada 1000 Transaksi

Dari hasil uji ini *rule* yang mempunyai rata-rata Lift Ratio tertinggi adalah pada saat diberikan kondisi sesuai dengan skenario 1 uji coba 6 dengan minimum Support *0.01*, minimum Confidence 0.6 dan minimum Interestingnes 10 dengan jumlah *Rule* 51.

4.4 Analisis Hasil

Dari hasil uji coba yang telah dilakukan dilakukan sebelumnya kemudian dilakukan analisis lebih lanjut untuk mengetahui hasil yang didapatkan.

4.4.1 Analisis Pembentukan Frequent Itemset Menggunakan Metode MIbARM

Setelah dilakukan implementasi dan ujicoba, dalam proses pembentukan *frequent itemset*, dapat diketahui bahwa algoritma MIbARM dapat digunakan untuk menghindari terjadinya perulangan kandidat *frequent itemset* seperti terlihat pada contoh tabel 4.11.

Tabel 4.11 Contoh Perulangan Kandidat Itemset

Kandidat Frequent	Kandidat Perulangan
1,2	(2,1)
1,2,3	(1,3,2), (2,3,1),(3,1,2),(2,1,3), (3,2,1)
1,2,3,4	(1,2,4,3), (1,3,2,4), (1,3,4,2), (1,4,2,3), (1,4,3,2), (2,1,3,4), (2,1,4,3), (2,3,1,4), (2,3,4,1), (2,4,1,3), (2,4,3,1), (3,1,2,4), (3,1,4,2), (3,2,1,4), (3,2,4,1), (3,4,1,2), (3,4,2,1), (4,1,2,3), (4,1,3,2), (4,2,1,3), (4,2,3,1), (4,3,1,2)

Metode yang membantu dalam proses iterasi pembentukan kandidat itemset seperti penggunaan matriks *T,* jumlah transaksi dalam matriks T akan selalu berkurang karena ada pemangkasan baris transaksi disetiap iterasi pembentukan kandidat *itemset*, sehingga kandidat-kandidat yang tidak memenuhi kondisi dalam setiap iterasi akan dibuang.

Pada Iterasi pembentukan kandidat *2-Itemset*, digunakan matriks Hermite *(H)* yang merupakan kombinasi dari *item* dalam transaksi, dimana setelah dibentuk kandidat item, kemudian dicari jumlah kemunculan kandidat tersebut *(SupA)*, matriks G digunakan untuk menyimpan nilai dari kandidat bila lebih besar dari minimum Support maka akan bernilai 1, jika kurang maka bernilai 0 dan bila kombinasi tersebut merupakan *item* yang sama maka juga bernilai 0, sehingga dalam pembentukan *frequent itemset* cukup mengambil kombinasi *item* dari matriks G yang bernilai satu sehingga metode ini

juga membantu dalam mengurangi terjadinya perulangan kandidat *frequent itemset*.

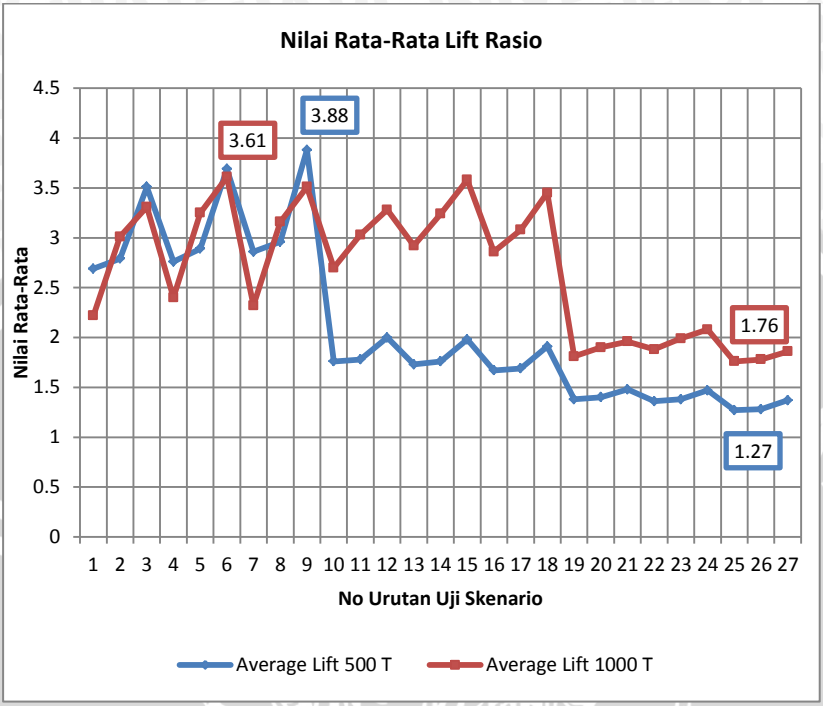
Pada iterasi pembentukan kandidat *frequent* yang lebih dari *2-itemset* metode MIbARM membentuk kandidat yang berasal dari *frequent itemset* sebelumnya, sehingga tidak perlu mencari kandidat dari awal, dengan mengkombinasikan kandidat sebelumnya dengan *item* dalam Transaction matrix *(T)* yang telah dipangkas pada pembentukan *frequent itemset* sebelumnya, maka metode ini juga membantu mengurangi terjadinya perulangan kandidat *frequent itemset*.

4.4.2 Analisis Pembentukan Rule

Hasil uji coba pada 500 dan 1000 data transaksi didapatkan *rule* yang memiliki rata-rata nilai Confidence 0.67 menunjukkan bahwa *rule* yang dihasilkan mempunyai peluang kemunculan lebih dari 50%. Rata-rata nilai Interestingness hasil uji coba pada 500 dan 1000 data transaksi 7.57 dan 15.54, ini menunjukkan bahwa *rule* yang dihasilkan mempunyai nilai Interestingness lebih besar dari 0.

Dari rata-rata nilai Lift Ratio hasil uji pada 500 transaksi dan 1000 transaksi pada gambar 4.5 diketahui pada saat pembentukan *rule* dengan menggunakan Confidence dan Interestingness, nilai Lift Ratio dari setiap *rule* yang dihasilkan lebih besar dari 1 seperti yang terlihat pada gambar 4.7, ini menunjukkan bahwa kejadian pada setiap *rule* Antecendent mempunyai efek positif terhadap kejadian di setiap *rule* Consequent, sehingga semakin tinggi nilai Lift pada *rule* semakin tinggi keterkaitan antara *rule* Antecendent dan *rule* Consequent. Semakin tinggi nilai Confidence yang dimiliki oleh *rule* maka semakin tinggi nilai Lift Ratio *rule* tersebut, demikian juga semakin tinggi nilai Confidence maka semakin tinggi nilai Interestingness rule tersebut. Semakin besar jumkah transaksi semakin tinggi juga nilai Interestingness Rule.

Rata-rata Lift Ratio tertinggi pada uji 500 transaksi adalah 3.88 pada saat nomor urutan uji skenario ke 9 dan terendah 1.37 pada nomor urutan uji skenario coba ke 25, dan rata-rata Lift Ratio tertinggi pada saat uji 1000 transaksi adalah 3.61 pada nomor urutan uji skenario ke 6 dan terendah 1.76 pada nomor urutan uji skenario ke 25. Rata-rata keseluruhan Lift Ratio pada hasil uji 500 dan 1000 Transaksi adalah 2.1 dan 2.67.



Gambar 4.8 Rata-Rata Lift Ratio Rule Hasil Uji Coba Berdasarkan Tabel 4.8 dan Tabel 4.10