JURNAL DASI ISSN: 1411-3201
Vol. 10 No. 1 Maret 2009

PENERAPAN DATA MINING ALGORITMA ASOSIASI UNTUK MENINGKATKAN PENJUALAN

Emha Taufiq Luthfi
STMIK AMIKOM Yogyakarta
E-mail : emha_tl@yahoo.com

Abstraksi

Strategi penjualan yang tepat merupakan hal yang sangat penting dalam bisnis untuk dapat meningkatkan nilai penjualan. Banyak faktor yang dapat mempengaruhi tingkat penjualan suatu barang. Dalam tulisan ini dibahas penggunaan data mining algoritma asosiasi untuk menyusun sebuah sistem yang memiliki kemampuan melihat pola penjualan barang yang selanjutnya dapat digunakan untuk menyusun strategi penjualan baru.

***Kata Kunci** : strategi penjualan, data mining, asosiasi*

Pendahuluan

Ketersediaan detail informasi transaksi pelanggan mendorong pengembangan teknik yang secara otomatis mencari hubungan antara item dalam data di database. Sebagai contoh data didapat dari scanner bar-code di supermarket. Database penjualan menyimpan jumlah record transaksi penjualan yang sangat besar. Setiap record memberikan daftar item barang yang dibeli oleh pelanggan dalam satu transaksi. Manager mungkin akan tertarik untuk mengetahui jika beberapa kelompok item barang secara konsisten dibeli secara bersama. Manager dapat menggunakan data tersebut dalam pengaturan layout toko untuk meletakkan item barang secara optimal dengan keterkaitan satu dengan lainnya, dapat pula digunakan dalam promosi, atau dalam design katalog dan untuk mengidentifikasi segmen pelanggan berdasar pola pembelian.

Sebagai contoh, jika pembeli membeli susu, bagaimana kemungkinan mereka juga akan membeli roti (roti seperti apa) dalam waktu yang sama di supermarket? Informasi seperti itu akan membantu meningkatkan penjualan dengan membantu retailer melakukan pemasaran yang selektif dan merencanakan layout mereka. Sebagai contoh, meletakan susu dan roti dengan posisi dekat kemungkinan akan meningkatkan penjualan item tersebut bersama dalam sebuah penjualan.

Proses untuk menguraikan penemuan pengetahuan di dalam database seperti yang dilakukan untuk melihat keterkaitan penjualan antar item diatas merupakan suatu konsep yang di sebut dengan data mining. Data mining merupakan konsep utama dalam *business intelligence* (BI) serta *online analytical processing* (OLAP). Terdapat banyak algoritma data mining yang jika dapat diterapkan dalam proses bisnis, akan memberikan nilai positif bagi peningkatan kinerja proses bisnis tersebut yang berujung pada peningkatan keuntungan dari bisnis tersebut.

Algoritma asosiasi merupakan suatu bentuk algoritma dalam data mining yang memberikan informasi hubungan antar item data di database. Algoritma tersebut dapat dimanfaatkan secara luas dalam proses bisnis diantaranya dalam proses penjualan. Data mining algoritma asosiasi dapat membantu dalam proses penjualan dengan memberikan hubungan antar data penjualan yang dilakukan pelanggan sehingga akan didapat pola pembelian pelanggan. Pebisnis dapat memanfaatkan informasi tersebut untuk mengambil tindakan bisnis yang sesuai.

Data mining menurut Turban adalah proses yang menggunakan teknik statistik, matematika, kecerdasan buatan dan machine learning untuk mengekstraksi dan mengidentifikasi informasi yang bermanfaat dan pengetahuan yang terakit dari berbagai database besar. Menurut Gatner Group data mining adalah suatu proses menemukan hubungan yang berarti, pola dan kecenderungan dengan memeriksa dalam sekumpulan besar data yang tersimpan dalam

penyimpanan, dengan menggunakan teknik pengenalan pola seperti teknik statistik dan matematika (Larose,2005).

Cross-Industry Standard Process for Data Mining (CRISP-DM) yang dikembangkan tahun 1996 oleh analis dari beberapa industri seperti DaimlerChrysler, SPSS dan NCR. CRISP DM menyediakan standar proses data mining sebagai strategi pemecahan masalah secara umum dari bisnis atau unit penelitian dalam 6 fasa CRISP-DM (Larose, 2005) :

1. Fasa Pemahaman Bisnis (*Business Understanding Phase*)
2. Fasa Pemahaman Data (*Data Understanding Phase*)
3. Fasa Pengolahan Data (*Data Preparation Phase*)
4. Fasa Pemodelan (*Modeling Phase*)
5. Fasa Evaluasi (*Evaluation Phase*)
6. Fasa Penyebaran (*Deployment Phase*)

Analisis asosiasi atau association rule mining adalah teknik data mining untuk menemukan aturan assosiatif antara suatu kombinasi item. Misal I = {i1, i2, i3, …, im} merupakan himpunan items. Dan D merupakan himpunan database transaksi yang setiap transaksi T merupakan himpunan item yang T C I. Setiap transaksi diassosiasikan dengan identifier yang disebut TID. Misal A dan B adalah himpunan items dalam transaksi. Transaksi T dikatakan mengandung A jika dan hanya jika A C T. Aturan asosiasi merupakan implikasi dengan bentuk A → B, yang mana A C I, B C I, dan A ∩ B = ∅. Aturan A → B berada dalam himpunan transaksi D dengan support S, yang mana S merupakan persentase dari transaksi di D yang mengandung A ∪ B (keduanya A dan B). Hal tersebut merupakan probabilitas P(A ∪ B). Aturan A → B memiliki confidence X dalam himpunan transaksi D. Jika C merupakan persentase dari transaksi dalam D mengandung A dan juga mengandung B. Maka ini merupakan probabilitas kondisional, P(B | A), maka :

Support (A → B) = P(A ∪ B)

Confidence (A → B) = P(B|A)

Aturan yang memenuhi minimum threshold support (min_sup) dan minimum threshold confidence (min_conf) disebut *strong*. Berdasar konvensi, nilai support dan confidence lebih sering antara nilai 0% dan 100% dibanding 0 sampai dengan 1.0.

Himpunan item disebut sebagai itemset. Itemset yang mengandung k items merupakan k-itemset. Himpunan {kopi, gula} merupakan 2-itemset. Kecenderungan kemunculan itemset dalam sejumlah transaksi disebut frequency, support count atau count dari itemset. Aturan asosiasi dilaksanakan dalam 2 langkah proses, yaitu :

1. Temukan semua frequent itemset; Berdasar definisi, masing-masing dari itemset aka muncul sedikitnya dengan frequency sebesar di berikan dalam minimum support count.
2. Munculkan strong association rule dari frequent itemset; Berdasar definisi, aturan ini harus memenuhi minimum support dan minimum confidence.

Pembahasan

Pengembangan Aplikasi meliputi:

1. Perancangan

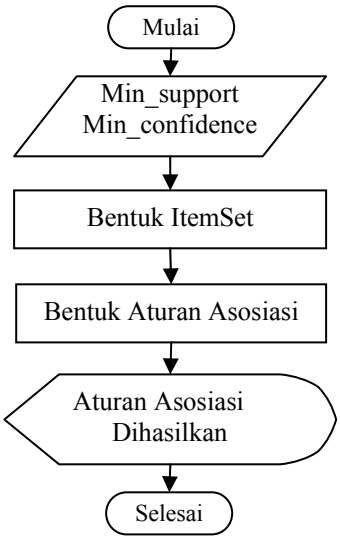
a. Tabel Data

Dalam percobaan ini digunakan data sekunder sebagai contoh data transaksi dengan struktur dan data sebagai berikut :

Tabel 1 Tabel Transaksi Penjualan

ID	NoTransaksi	KodeBarang	NamaBarang
1	001	1	A
2	001	3	C
3	001	4	D
4	002	2	B
5	002	3	C
6	002	5	E
7	003	1	A
8	003	2	B
9	003	3	C
10	003	5	E
11	004	2	B
12	004	5	E

b. FlowChart Program Analisis Asosiasi



c. Pengkodean

```
package asosiasi;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
```

```
public class AsosiasiGUI extends javax.swing.JFrame {

    private String driverDB =
            "sun.jdbc.odbc.JdbcOdbcDriver";
    private String url = "jdbc:odbc:Driver={Microsoft " +
            "Access Driver (*.mdb, *.accdb)};" +
            "DBQ=C:\\Documents and Settings\\" +
            "luthfi\\My Documents\\NetBeansProjects\\" +
                "myApriori\\" +
            "src\\apriori\\myAprioriDataTest1.accdb";

    private String namaTabel = "[Penjualan]";
    private String jumlahKolom = "jumlah";
    private String kolomUtamaTransaksi = "[NoTransaksi]";
    private String kolomDetailBarangTransaksi =
                                "[NamaBarang]";
    private String kolomUrutanTransaksi = "ID";

    private double support;
    private double confidence;
    private boolean apriori;
    private Connection conn;
    private int totalTransaksi = 0;
    private HashMap perhitunganSupport;

    /** Creates new form AsosiasiGUI */
    public AsosiasiGUI() {
        initComponents();
        // bentuk koneksi ke database, dan ambil data
        try {
            Class.forName(driverDB).newInstance();
            // buat koneksi
            conn = DriverManager.getConnection(url);
```

```
        // tampilkan data
        String sql = "Select * from " + namaTabel;
        ResultSet rsDataPenjualan = execSQL(sql);
        //ambil meta data
        ResultSetMetaData meta =
              rsDataPenjualan.getMetaData();
        for (int i=1;i<=meta.getColumnCount();i++) {
          txtData.append(meta.getColumnName(i)+"\t");
        }
        txtData.append("\n");
        while(rsDataPenjualan.next()) {
          for(int i=1;i<=meta.getColumnCount();i++){
 txtData.append(rsDataPenjualan.getString(i) + "\t");
          }
          txtData.append("\n");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    setSize(541, 300);
}

// bagian utama dari proses
public void mulai() throws Exception {
    // inisialisasi
    AsosiasiGUI dm = init();

    // tentukan itemset
    List[] itemSet = dm.cariSemuaItemSet();

    List semuaNItemSet = itemSet[0];
    List semuaItemSet = itemSet[1];

    Map[] isiAturan =
```

```
            dm.cariAturan(semuaNItemSet, semuaItemSet);

        Map aturan = isiAturan[0];
        Map support_aturan = isiAturan[1];
        Map confidence_aturan = isiAturan[2];

        //print out aturan
        Iterator it = aturan.keySet().iterator();
        txtHasil.append("Aturan:\t\t\tSupport:\tConfidence:\n");
txtHasil.append("------\t\t\t--------\t----------\n");

    while (it.hasNext()) {
        Object key = it.next();
        List listNilai = (List)aturan.get(key);
        List listSupport = (List)support_aturan.get(key);
        List listConfidence =
                    (List)confidence_aturan.get(key);

        for (int x = 0; x < listNilai.size(); x++) {

txtHasil.append( key + "->" + listNilai.get(x) + "\t\t" +
((Float)(listSupport.get(x))).floatValue()*100 + "%\t" +
((Float)(listConfidence.get(x))).floatValue()*100+"%\n");
            }
        }
    }

    private void tambahAturan (List nilai, List support,
        List confidence, List semuaItemSet,
        List subSet, List sisa, float hitungKeseluruhan,
        float hitungSubSet) {

        if (apriori) {
            List cekList = new ArrayList((List)subSet);
```

```
            cekList.addAll(sisa);
            if (semuaItemSet.contains(cekList)) {
               nilai.add(sisa);
               support.add(new
  Float((float)hitungKeseluruhan/(float)totalTransaksi));
               confidence.add(new
  Float((float)hitungKeseluruhan/(float)hitungSubSet));
            }
         }


      else {
            nilai.add(sisa);
            support.add(new
Float((float)hitungKeseluruhan/(float)totalTransaksi));
            confidence.add(new
Float((float)hitungKeseluruhan/(float)hitungSubSet));
         }
      }


    public AsosiasiGUI (double support, double confident,
         boolean aprioriAll) throws Exception
    {
       // tentukan driver koneksi
       Class.forName(driverDB).newInstance();

       // buat koneksi
       conn = DriverManager.getConnection(url);
       perhitunganSupport = new HashMap();
       this.support = support;
       this.confidence = confident;
       this.apriori = aprioriAll;
    }
```

```
public AsosiasiGUI init () throws Exception
{
   // tentukan minimun confidence
   confidence = 0.0;

   // tentukan minimum support
   support = 0.0;

   apriori = false;
   if (txtSupport.getText().trim().equals("")) {
      txtSupport.setText("0");
   } else {
     int tempSupport =
      Integer.parseInt(txtSupport.getText().trim());
      if (tempSupport > 0) {
          support = (double)(tempSupport)/100;
      }
   }

   if (txtConfident.getText().trim().equals("")) {
      txtConfident.setText("0");
   } else {
    int tempSupport =
    Integer.parseInt(txtConfident.getText().trim());
      if (tempSupport > 0) {
          support = (double)(tempSupport)/100;
      }
   }
   apriori = true;

   return new AsosiasiGUI(support, confidence,
                          apriori);
```

```
    }

    public int getTotal () throws Exception {
        if (totalTransaksi == 0) {
            String sql = "select count(X.A) from " +
    "(select   distinct("+kolomUtamaTransaksi+") as A " +
    "from "+namaTabel+") X";

            ResultSet rs = execSQL(sql);
            if (rs.next())
                totalTransaksi = rs.getInt(1);
        }
        return  totalTransaksi;
    }

    public String getSQL (int nomorItem, List sebelum) {

String idTransaksi = "a0."+kolomDetailBarangTransaksi+"";
        String tabel = " "+namaTabel+" a0";
        String aprioriFreq = "";
        String skipDupTransaksi = "";
        String idObyek = "";
        String itemSebelum = "";
        String itemDicari = " ";

        if (sebelum != null) {
            StringBuffer sb = new StringBuffer(100);
            for (int i = 0; i < sebelum.size(); i++) {
                ArrayList itemSet =
                        (ArrayList)sebelum.get(i);
                for (int j = 0; j < itemSet.size(); j++){
                    String item =
                            itemSet.get(j).toString();
                    if (sb.toString().indexOf(item)== -1)
```

```
                sb.append(",").append("'"+item+"'");
            }
          }
          itemSebelum = sb.substring(1);
        }
        // bentuk SQL
        if ( itemSebelum.length() > 0 ) {
          itemDicari += " and a0."+
              kolomDetailBarangTransaksi+" in (" +
              itemSebelum + ") ";
        }
        for (int i = 1; i < nomorItem; i++) {
          idTransaksi += ", a" + i +
              "."+kolomDetailBarangTransaksi+"";
          tabel += ", "+namaTabel+" a" + i;
          idObyek += " and a" + (i - 1) +
              "."+kolomUtamaTransaksi+" = a" + i +
              "."+kolomUtamaTransaksi+"";

          if ( itemSebelum.length() > 0 )
            itemDicari += " and a" + i +
"."+kolomDetailBarangTransaksi+" in ("+itemSebelum+") ";

          if (apriori)
            aprioriFreq += " and a" + (i - 1) +
"."+kolomUrutanTransaksi+" < a" + i +
"."+kolomUrutanTransaksi+"";
          else
            skipDupTransaksi += " and a" + (i - 1) +
"."+kolomDetailBarangTransaksi+" < a" + i +
"."+kolomDetailBarangTransaksi+"";
        }

        // SQL itemset awal
```

```
    if (aprioriFreq.equals("") && idObyek.equals("")){
        return  "select " + idTransaksi + ", " +
"count(a0."+kolomUtamaTransaksi+") as jumlah from " +
            tabel + " group by " + idTransaksi;
      }


      else {
        idObyek = idObyek.substring(4);
        return  "select " + idTransaksi
+ ", count(a0."+kolomUtamaTransaksi+") as jumlah  from "+
tabel + " where " + idObyek + skipDupTransaksi +
aprioriFreq + itemDicari + " group by " + idTransaksi;
      }
   }

   public List cariItemSetDanJumlah (int no_i,
          List itemsSetAkhir) throws Exception {

      ArrayList itemSet = new ArrayList();
      // bentuk sql untuk kombinasi no_i itemSet
      String sql = getSQL(no_i, itemsSetAkhir);

      PreparedStatement stmt =
            conn.prepareStatement(sql);
      ResultSet rs = stmt.executeQuery();

      while (rs.next()) {
        // Ambil distinct transaksi
        ArrayList item = new ArrayList();
        for (int i = 0; i < no_i; i++) {
          String idTransaksi = new
              String(rs.getString(i + 1));
          item.add(idTransaksi);
```

```
        }

        int count = rs.getInt(jumlahKolom);
        if (count >= getTotal()*support) {
           itemSet.add(item);
           perhitunganSupport.put(item,
                    new Integer(count));
        }
     }
     rs.close();
     stmt.close();
     return  itemSet;
}


//Tentukan semua itemset yang ada untuk support yg diberikan
public List[] cariSemuaItemSet () throws Exception {
     ArrayList semuaItemSet1 = new ArrayList();
     ArrayList semuaItemSet2 = new ArrayList();

// Tentukan semua frequency itemset berdasar support
     List itemsSetAkhir = null;
     int i = 0;
     while(true) {
        i++;
        List itemSet1 = cariItemSetDanJumlah(i,
                              itemsSetAkhir);
        if (itemSet1.size() > 0) {
           semuaItemSet1.add(itemSet1);
           semuaItemSet2.addAll(itemSet1);
        }
        else
           break;
        itemsSetAkhir = itemSet1;
```

```
    }
    List[] returnList =  {semuaItemSet1,
                          semuaItemSet2};
    return  returnList;
}

/**
 * Cari semua aturan berdasar support dan confidence diberikan
 * @param semuaItemSet1
       list berisi itemsSet berdasar N <br>
 *      misal : {{{1,2},{2,3}},{{1,2,3},{2,3,4}}}<br>
 *
 * @param semuaItemSet2
       list berisi itemSet tanpa batas<br>
 *      misal : { {1,2}, {2,3}, {1,2,3},{2,3,4} }<br>
 *
 * @return  A Map array.<br>
 *          item pertama berisi aturan.<br>
 *          item kedua berisi support.<br>
 *          item ketiga berisi confidence.<br>
 */


public Map[] cariAturan (List semuaItemSet1,
              List semualItemSet2) {

    HashMap aturan = new HashMap();
    HashMap support_aturan = new HashMap();
    HashMap confidence_aturan = new HashMap();

    // untuk setiap itemset di semuaItemSet1
    for (int j = 0; j < semuaItemSet1.size(); j++) {
       List nItemSet1 = (List)semuaItemSet1.get(j);
       // untuk semua item dilevel tsb
```

```
          for (int k = 0; k < nItemSet1.size(); k++) {
             List itemSet1 =
               new ArrayList((List)nItemSet1.get(k));

             // temukan semua subset dari semua item
       for (int l = 0; l < semualItemSet2.size(); l++) {
                List subSetItemSet1 =
            new ArrayList((List)semualItemSet2.get(l));
            List sisa = new ArrayList((List)itemSet1);
                sisa.removeAll(subSetItemSet1);

           if (itemSet1.containsAll(subSetItemSet1) &&
              itemSet1.containsAll(sisa)&&
              sisa.size() > 0) {

             int hitungKeseluruhan =
((Integer)perhitunganSupport.get(itemSet1)).intValue();
             int hitungSubSet =
((Integer)perhitunganSupport.get(subSetItemSet1)).intValue();

            // Cek apakah confidence memenuhi
            if ((float)hitungKeseluruhan/(float)hitungSubSet >=
confidence) {
    ArrayList nilai =
        (ArrayList)aturan.get(subSetItemSet1);
    ArrayList supports =
        (ArrayList)support_aturan.get(subSetItemSet1);
                    ArrayList confidences =
(ArrayList)confidence_aturan.get(subSetItemSet1);


         if (nilai != null) {
           if (!nilai.contains(sisa)) {
```

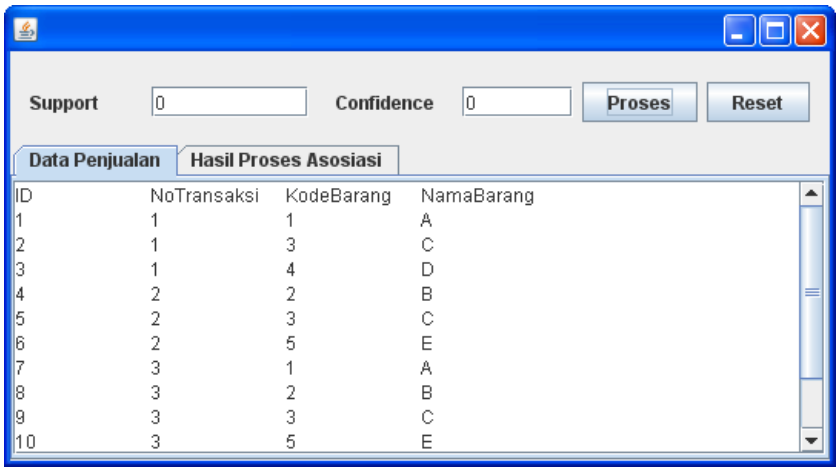
```
tambahAturan(nilai, supports, confidences,
            semualItemSet2, subSetItemSet1, sisa,
            hitungKeseluruhan, hitungSubSet);
              }
           } else {
                  nilai = new ArrayList();
              supports = new ArrayList();
              confidences = new ArrayList();
tambahAturan(nilai, supports, confidences,
            semualItemSet2, subSetItemSet1, sisa,
            hitungKeseluruhan, hitungSubSet);

aturan.put(subSetItemSet1, nilai);

support_aturan.put(subSetItemSet1, supports);

confidence_aturan.put(subSetItemSet1, confidences);
                        }
                     }
                  }
               }
            }
         }
         Map[] returnMap =  {
            aturan, support_aturan, confidence_aturan
         };
         return  returnMap;
      }

      private void btnProsesActionPerformed
              (java.awt.event.ActionEvent evt) {

         try {
            mulai();
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private ResultSet execSQL (String sql)
            throws Exception {

            PreparedStatement stmt =
                    conn.prepareStatement(sql);
        return  stmt.executeQuery();
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new AsosiasiGUI().setVisible(true);
            }
        });
    }
}
```

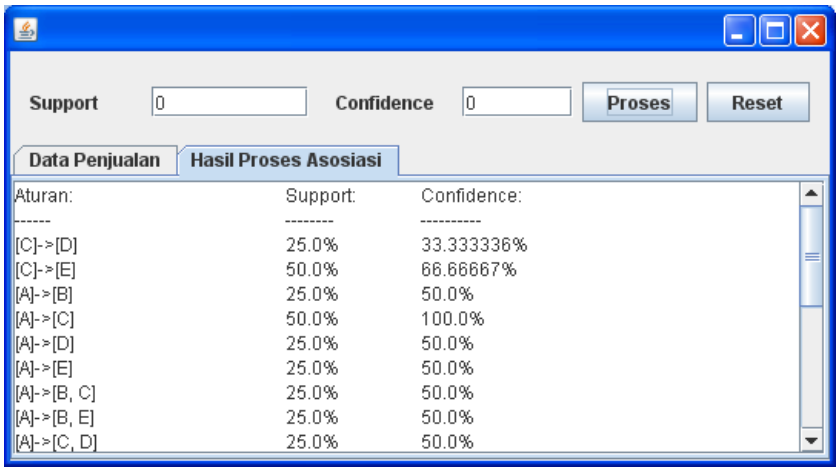
2. Hasil

Dengan menggunakan data transaksi dalam tabel 1, maka program dibangun seperti tampak dalam gambar 1 menampilkan data transaksi keseluruhan. Selanjutnya aplikasi meminta masukan minimum support dan minimum confidence diharapkan.

 ISSN: 1411-3201



Gambar 1 Data Transaksi



Gambar 2 Data Hasil Proses Asosiasi

Gambar 2 memberikan hasil proses analisis asosiasi untuk data transaksi diberikan dengan minimum support 0 % dan confidence 0%. Dengan menggunakan hasil analisis tersebut pemilik usaha dapat menyusun strategi penjualan berkaitan dengan hubungan kemunculan barang secara bersama dalam suatu transaksi.

Penutup

Pemanfaatan data mining dengan analisis algoritma asosiasi dapat membantu pemilik usaha untuk menemukan keterkaitan atau pola kemunculan barang dalam transaksi penjualan, yang pada akhirnya dapat digunakan untuk menyusun strategi penjualan. Penggunaan data mining dalam menggali data terbukti dapat bermanfaat dalam dunia bisnis secara khusus terbukti untuk kasus penjualan barang

Daftar Pustaka

Gordon S. Linoff, 2004, *Data Mining Techniques : For Marketing, Sales, and Customer Relationship Management,* Wiley Publishing, Indianapolis

Kusrini dan Luthfi, E.T, 2009, *Algoritma Data Mining*, Andi Offset, Yogyakarta

Tang, Z dan MacLennan, J, 2005, *Data Mining with SQlServer 2005*, Wiley Publishing, Indianapolis