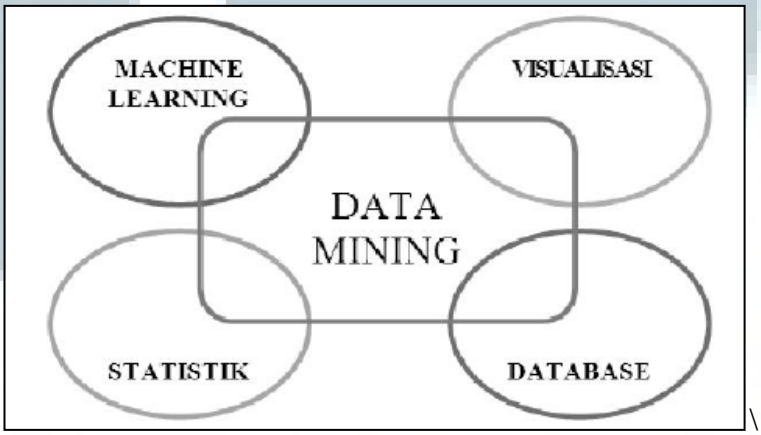
BAB II

TINJAUAN PUSTAKA

2.1 Data Mining

Data mining adalah suatu istilah yang digunakan untuk menguraikan penemuan pengetahuan di dalam *database* (Turban, Aronson, & Liang, 2007). Menurut Kusrini & Luthfi (2009), *Data mining* adalah proses yang menggunakan teknik statistik, matematika, kecerdasan buatan, dan *machine learning* untuk mengekstraksi dan mengidentifikasi informasi yang bermanfaat dan pengetahuan yang terkait dari berbagai *database* besar.



Gambar 2.1 *Data Mining* (Sumber: Gunadi & Sensuse, 2012)

Larose dalam bukunya yang berjudul *"Discovering Knowledge in Data: An Introduction to Data Mining*" (2005), *data mining* dibagi menjadi beberapa kelompok berdasarkan tugas yang dapat dilakukan, yaitu :

a. Deskripsi

Terkadang peneliti dan analis secara sederhana ingin mencoba mencari cara untuk menggambarkan pola dan kecenderungan yang terdapat dalam data.

Deskripsi dari pola kecenderungan sering memberikan kemungkinan penjelasan untuk suatu pola atau kecenderungan.

b. Klasifikasi

Dalam klasifikasi, terdapat target variabel kategori. Sebagai contoh, penggolongan pendapatan dapat dipisahkan dalam tiga kategori, yaitu pendapatan tinggi, pendapatan sedang, dan pendapatan rendah.

c. Estimasi

Estimasi hampir sama dengan klasifikasi, kecuali variabel target estimasi lebih ke arah numerik dari pada ke arah kategori. Model dibangun menggunakan baris data (*record*) lengkap yang menyediakan nilai dari variabel target sebagai nilai prediksi. Selanjutnya, pada peninjauan berikutnya estimasi nilai dari variabel target dibuat berdasarkan nilai variabel prediksi.

d. Prediksi

Prediksi hampir sama dengan klasifikasi dan estimasi, kecuali bahwa dalam prediksi nilai dari hasil akan ada di masa mendatang. Beberapa metode dan teknik yang digunakan dalam klasifikasi dan estimasi dapat pula digunakan (untuk keadaan yang tepat) untuk prediksi.

e. Pengklasteran (*Clustering*)

Pengklasteran merupakan pengelompokkan *record*, pengamatan, atau memperhatikan dan membentuk kelas obyek-obyek yang memiliki kemiripan. Klaster adalah kumpulan *record* yang memiliki kemiripan satu dengan yang lainnya dan memiliki ketidakmiripan *record* dalam klaster yang lain. Berbeda dengan klasifikasi, pada pengklasteran tidak ada variabel target. Pengklasteran tidak melakukan klasifikasi, mengestimasi, atau memprediksi nilai dari variabel

target, akan tetapi, algoritma pengklasteran mencoba untuk melakukan pembagian terhadap keseluruhan data menjadi kelompok-kelompok yang memiliki kemiripan (homogen), yang mana kemiripan *record* dalam satu kelompok akan bernilai maksimal, sedangkan kemiripan dengan *record* dalam kelompok lain akan bernilai minimal.

f. Asosiasi

Tugas asosiasi dalam *data mining* adalah untuk menemukan atribut yang muncul dalam satu waktu.

2.2 Machine Learning

Machine Learning adalah bagian dari *Artificial Intelligence*. Dalam buku *Machine Learning*, Tom Mitchell, 1997 mendefinisikan *machine learning* berbicara tentang bagaimana membangun sebuah program komputer yang dapat belajar dari pengalaman. Hasil pembelajaran dapat diperoleh dari beberapa cara, diantaranya melalui perhitungan menggunakan rumus-rumus sesuai dengan tujuan algoritma. Dalam penelitian ini, algoritma yang digunakan bertujuan untuk mencari *association rules*.

Recommendation System (RS) adalah salah satu bentuk *machine learning* yang dapat dikembangkan dengan beberapa algoritma. Dengan menggunakan algoritma yang efektif, maka hasil akan lebih akurat.

2.3 Recommendation System

Recommendation System atau *Recommendation Agent (RA)* adalah agen perangkat lunak yang memiliki informasi penting mengenai produk pilihan konsumen, baik secara eksplisit maupun implisit, dan memberikan rekomendasi berdasarkan informasi tersebut (Xiao & Izak, 2007). Secara eksplisit artinya *input*

yang sengaja dibuat oleh pelanggan dengan tujuan memberi informasi kepada aplikasi rekomendasi sebagai acuannya, seperti *rating user* terhadap suatu *item*, atau komentar *user* terhadap suatu *item*. Secara implisit artinya barang spesifik yang sedang dilihat pelanggan atau yang berada di riwayat pembelian.

Recommendation Agent (RA) dapat meningkatkan kualitas hasil keputusan atas pemilihan suatu produk. *Recommendation Agent (RA)* dapat mengurangi informasi yang terlalu banyak serta pilihan yang terlalu luas bagi konsumen. *Recommendation Agent (RA)* banyak dimanfaatkan di berbagai bidang, di antaranya pendidikan, manajemen organisasi, dan *e-commerce*. Dalam konteks *e-commerce*, *Recommendation Agent (RA)* dapat dibedakan dalam dua sudut pandang, yaitu *product brokering* dan *merchant brokering* (Spiekermann, 2001). *Product brokering* bertujuan untuk menemukan produk yang paling cocok, sedangkan *merchant brokering* bertujuan menemukan penjual yang paling cocok. Yang diimplentasikan pada *website* ini adalah *product brokering*.

Menurut Xiao & Izak (2007) dalam *E-commerce Product Recommendation Agents: Use, Characteristics, and Impact*, *Recommendation Agent (RA)* memiliki karakteristik sebagai berikut.

a. Jenis-jenis *Recommendation Agent*

Berdasarkan sumber informasi yang digunakan untuk menentukan hasil rekomendasi, terdapat dua jenis *Recommendation Agent (RA)*, yaitu *content filtering* dan *collaborative filtering*.

- *Content filtering*, memberikan rekomendasi berdasarkan atribut produk yang disukai oleh konsumen.

- *Collaborative filtering*, memberikan rekomendasi berdasarkan informasi tentang pengguna dan atau menggunakan informasi pengguna lain yang memiliki kesamaan profil
- *Hybrid RA*, memberikan rekomendasi dengan menggabungkan metode *content filtering* dan *collaborative filtering*.

Berdasarkan sifat atribut yang dimiliki oleh produk:

- *Compensatory*, memungkinkan terjadinya pertukaran atribut. Atribut yang kurang diinginkan akan diimbangi oleh atribut yang diinginkan dari sebuah produk. Semua atribut diperhitungkan untuk mendapatkan hasil rekomendasi.
- *Non-compensatory*, tidak memungkinkan terjadinya pertukaran atau penggantian atribut. Jika suatu atribut tidak diinginkan, maka nilainya tidak dapat digantikan oleh atribut yang lain.

Berdasarkan cara memperoleh preferensi:

- *Feature-based*, menanyakan secara spesifik fitur dari produk yang diinginkan pengguna.
- *Needs-based*, menanyakan tujuan pengguna membeli produk yang diinginkan dan bagaimana pengguna akan menggunakan produk tersebut.
- *Hybrid RA*, menggabungkan metode *feature-based* dan *needs-based*.

b. *Input Recommendation Agent (RA)*

Untuk dapat menghasilkan suatu rekomendasi, diperlukan metode untuk mendapatkan informasi pilihan pengguna bahkan saat preferensi pengguna diperbaharui. Akan lebih maksimal jika pengguna dapat menentukan sejauh mana *RA* akan memberikan rekomendasi kepada pengguna.

c. Proses *Recommendation Agent (RA)*

Saat *RA* mencari rekomendasi yang sesuai dari *database*, perlu ditampilkan progres pencarian, sehingga pengguna tahu bahwa hasil rekomendasi akan ditampilkan dalam waktu yang dapat diprediksi. Hal ini juga berkaitan dengan algoritma apa yang digunakan saat melakukan *scan* atau pencarian pada *database*.

d. *Output Recommendation Agent (RA)*

Hasil yang ditampilkan oleh *RA* dapat dinilai dari isi dan format. Berbagai pilihan yang harus diputuskan untuk menentukan isi *RA* adalah apakah produk akan ditampilkan berdasarkan skor atau prediksi *rating*, ditampilkan bersama rincian produk atau tidak. Dalam pemilihan format juga dapat dipertimbangkan apakah produk akan ditampilkan secara berurut, apakah pengguna bisa melihat jumlah *item* yang direkomendasikan.

Karakteristik *Recommendation Agent (RA)* menunjukkan bahwa *Recommendation Agent (RA)* termasuk *Decision Support System (DSS)* (Grenci & Todd, 2002) berdasarkan tiga elemen penting DSS yang dikemukakan oleh Mallach (2000):

- DSS adalah sistem informasi
- DSS digunakan untuk membuat keputusan
- DSS digunakan sebagai pendukung, bukan untuk menggantikan manusia.

2.4 Algoritma Apriori

Algoritma Apriori adalah suatu algoritma dasar yang diusulkan oleh Agrawal & Srikant pada tahun 1994 untuk penentuan *frequent itemset* untuk aturan asosiasi boolean (Han, Kamber, & Pei, 2002).

Bentuk algoritma dari metode Apriori yang dikemukakan Agrawal & Srikant dalam *Fast Algorithms for Mining Association Rules* (1994) dapat dituliskan sebagai berikut:

L1 = {frequent itemset with one element}
for(k=2; $L_{k-1} \neq \emptyset$; k++)
{
C_k = Apriori-gen(L_{k-1}); //pembuatan kandidat baru
for all transaction t
{
C'_t = subset(C_k, t); //kandidat yang tampil pada t
for all candidates c ε C'_t do c.count++;
}
L_k = { c ε C't | c.count ≥ minsup}
}
Return $U_k L_k$*;*

Keterangan:

L: himpunan *frequent itemset*

C : himpunan kandidat *itemset*

c : kandidat *itemset*

t : transaksi

minsup : *minimum support*

Beberapa parameter untuk mencari *association rule* menggunakan algoritma Apriori adalah *minimum support* dan *minimum confidence*. Kedua parameter ini akan menjadi acuan ditemukannya *frequent* baru yang memiliki *strong rule*. Minimum *support* merupakan ukuran yang menunjukkan seberapa besar tingkat dominasi atau persentase suatu kombinasi *item* dari keseluruhan transaksi (Kusrini & Luthfi, 2009). Minimum *confidence* merupakan ukuran atau nilai kepastian untuk menunjukkan kuatnya hubungan antar *item* dalam sebuah Apriori. Misalnya, seberapa sering *item* B dibeli jika orang membeli *item* A. Pola yang nantinya terbentuk disebut *interesting rule* atau *strong rule*. *Interesting rule* atau

strong rule ditemukan bila pola memenuhi kedua nilai minimum parameter yang sudah ditentukan.

Metodologi dasar analisis asosiasi terbagi menjadi dua tahap:

a. Analisa Pola Frekuensi Tinggi

Tahap ini mencari kombinasi *item* yang memenuhi syarat minimum dari nilai *support* dalam *database*. Nilai *support item* diperoleh dengan persamaan berikut.

$$Support\ (A) = \frac{\Sigma\ Transaksi\ yang\ mengandung\ A}{\Sigma\ Transaksi} \quad \text{... Rumus 2.1}$$

Nilai *support* dari 2 item diperoleh dari persamaan berikut.

$$Support(A, B) = P\ (A\ \cap B) = \frac{\Sigma\ Transaksi\ mengandung\ A\ dan\ B}{\Sigma\ Transaksi} \quad \text{... Rumus 2.2}$$

b. Pembentukan Aturan Asosiatif

Setelah semua pola frekuensi tinggi ditemukan, barulah dicari aturan asosiatif yang memenuhi syarat minimum untuk *confidence* dengan menghitung *confidence* aturan asosiatif A → B. Nilai tersebut diperoleh dari persamaan berikut.

$$Confidence = P(B|A) = \frac{\Sigma\ Transaksi\ Mengandung\ A\ dan\ B}{\Sigma\ Transaksi\ Mengandung\ A} \quad \text{... Rumus 2.3}$$

Proses utama yang dilakukan dalam algoritma Apriori untuk menemukan *frequent itemset* adalah sebagai berikut (Erwin, 2009).

1. *Join* (Penggabungan)

   Proses ini membuat kombinasi semua *item* sampai tidak terbentuk kombinasi lagi.

2. *Prune* (Pemangkasan)

   Pada proses ini *item* yang telah dikombinasikan, dipangkas menggunakan minimum *support* yang telah ditentukan oleh *user*.

Algoritma Apriori dibagi menjadi beberapa tahap yang disebut iterasi. Tiap iterasi menghasilkan pola frekuensi tinggi dengan panjang satu. Di iterasi pertama ini, *support* dari setiap *item* dihitung dengan melihat *database*. Setelah *support* dari tiap *item* didapat, *item* yang memiliki *support* di atas minimum *support* dipilih sebagai pola frekuensi tinggi dengan panjang 1 atau disebut juga 1-*itemset*. Pada iterasi kedua menghasilkan 2-*itemset*, artinya tiap set memiliki dua *item*. Di tahap ini dilakukan pembuatan kandidat 2-*itemset* dari kombinasi semua 1-*itemset* yang telah diperoleh sebelumnya. Untuk tiap kandidat 2-*itemset* dihitung masing-masing *support*. Kandidat 2-*itemset* yang memiliki nilai di atas *minimum support* ditetapkan sebagai 2-*itemset* yang juga merupakan pola frekuensi tinggi dengan panjang 2 *item*. Beberapa tahap atau narasi Apriori tersebut disimpulkan sebagai berikut (Setiawati, 2009).

1. Pembentukan kandidat *itemset*, k-*itemset* dibentuk dari kombinasi(k-1)-*itemset* yang didapat dari iterasi sebelumnya.
2. Penghitungan *support* dari tiap kandidat k-*itemset*. *Support* dari tiap kandidat k-*itemset* didapat dengan melakukan *scan database* untuk menghitung jumlah transaksi yang memuat semua *item* di dalam kandidat k-*itemset* tersebut.
3. Tetapkan pola frekuensi tinggi. Pola frekuensi tinggi yang memuat k item atau k-itemset ditetapkan dari kandidat k-itemset yang *support*- nya lebih besar dari minimum *support*.

4. Bila tidak didapat pola frekuensi tinggi baru maka seluruh proses dihentikan. Bila tidak, maka k ditambah satu dan kembali ke bagian 1.

Dari langkah-langkah di atas, terlihat bahwa untuk penghitungan pada tiap k-*itemset*, perlu dilakukan *scan database*.

2.5 Algoritma Improved Apriori Shankar Bargadiya

Shankar dan Bargadiya melakukan improvisasi terhadap algoritma Apriori yang sudah ada sebelumnya. Algoritma ini mengatasi kelemahan-kelemaham utama yang ada pada Apriori, yaitu *scan database* yang berulang-ulang sehingga mempengaruhi kecepatan proses, dan banyaknya *frequent itemset* yang berulang-ulang atau redudansi yang menambah kompleksitas.

Algoritma ini hanya memerlukan dua kali *scan database*, sehingga mengurangi kompleksitas penghitungan *itemset* dan kuantitas kandidat *set* serta menghemat memori. Untuk mendapat hasil tersebut, yang dilakukan pertama kali adalah menemukan *frequent* 1-*itemset* dari *database* kemudian menginisialisasi *count=0*. Inisialisasi ini dimasukkan ke dalam *Global power set*. Ketika *scan database* dilakukan untuk menghitung *itemset*, *itemset* yang tidak memenuhi *minimum support* akan dihapus. Langkah ini akan mengurangi generasi berlebihan dari kandidat itemset yang akan dihasilkan. Setelah itu, mencari *Local power set*, yaitu hasil penghitungan *item* yang tersisa dari *Global power set*. Langkah ini mengurangi *scan database* berulang-ulang (Shankar & Bargadiya, A New Improved Apriori Algorithm For Association Rules Mining, 2013).

Input yang dibutuhkan dari penggunaan algoritma ini adalah:

1. *Database* D dengan format (Tid, itemset), di mana *Tid* adalah transaksi id.
2. Minimum *support*.

Ouput yang dihasilkan adalah *frequent itemset* dari *database*.

Berikut adalah langkah-langkah kerja algoritma Apriori yang telah diimprovisasi oleh Shankar & Bargadiya (2013).

1. L1 = menemukan *frequent* 1-*itemset*.
2. Membuat *Global power set*, yaitu L1 dengan inisialisasi *count* = 0;
3. Lakukan scan *database* hingga transaksi terakhir.
   a. Baca *itemset* dari transaksi dan hapus *item* yang tidak ada di L1.
   b. Mencocokkan *Global power set* dan *Local power set*. Jika cocok, maka *count Global power set* ditambah satu.
4. *Scan Global power set* dan perhatikan *count* setiap *item*. Jika di bawah *minimum support*, maka *item* tersebut dihapus dari *Global power set*.
5. *Item* yang tersisa adalah *frequent itemsets*.

Langkah 4 dan 5 adalah tahap *prune* (pemangkasan). Dengan menggunakan algoritma ini, tidak diperlukan tahap *join* (penggabungan).

Berikut adalah contoh gambaran cara kerja algoritma Apriori Shankar Bargadiya dengan *minimum support* = 3.

Tabel 2.1 Daftar Transaksi

Tid (ID Transaksi)	Itemset
1	Item3, Item4
2	Item1, Item2, Item3, Item4
3	Item1, Item3
4	Item2, Item3
5	Item1, Item2, Item3
6	Item2
7	Item1, Item3
8	Item1, Item3

Tabel 2.2 Daftar Transaksi (lanjutan)

Tid (ID Transaksi)	Itemset
9	Item1, Item2
10	Item2

Langkah 1: menemukan *frequent* 1-itemset dari *database*.

Tabel 2.3 *Frequent 1-itemset*

Items	Frequency of item
Item1	6
Item2	6
Item3	7

Item4 dihapus karena memiliki *count* = 2 lebih kecil dari *minimum support* yang telah ditentukan.

Langkah 2: Langkah ini akan membuat *Global power set* dengan inisialisasi *count*=0.

Tabel 2.4 Inisialisasi *Global Power Set*

Candidate 1 itemset	Item1	Item2	Item3
1-Count	0	0	0
Candidate 2 itemset	Item1, Item2	Item1, Item3	Item2, Item3
2-Count	0	0	0
Candidate 3 itemset	Item1, Item2, Item3		
3-Count	0		

Langkah 3: Melakukan *scan* terhadap transaksi yang ada di *database*.

Untuk transaksi dengan Tid=1 terdapat Item4 yang telah dihapus, sehingga pada transaksi tersebut hanya terdapat satu *item*, yaitu Item3. Jika setelah penghapusan masih tersisa lebih dari satu *item*, maka Tid tersebut dicocokkan dengan *Global power set*.

Setelah *scan database* selesai dilakukan, maka *Global power set* akan tersimpan dengan data sebagai berikut.

Tabel 2.5 Statistik kandidat *itemset* dan frekuensi muncul *item*

Candidate 1 itemset	Item1	Item2	Item3
1-Count	6	6	7
Candidate 2 itemset	Item1, Item2	Item1, Item3	Item2, Item3
2-Count	3	5	3
Candidate 3 itemset	Item1, Item2, Item3		
3-Count	2		

Langkah 4: Berdasarkan hasil pada Tabel 2.4 dan *minimum support* yang telah ditetapkan, maka kandidat *itemset* yang tidak memenuhi syarat dihapus.

Tabel 2.6 Pemangkasan kandidat itemset

Candidate 1 itemset	Item1	Item2	Item3
1-Count	6	6	7
Candidate 2 itemset	Item1, Item2	Item1, Item3	Item2, Item3
2-Count	3	5	3
Candidate 3 itemset	**Item1, Item2, Item3**		
3-Count	**2**		

Langkah 5: Setelah pemangkasan selesai, hasil *frequent itemset* ditampilkan pada Tabel 6.

Tabel 2.7 *Frequent itemset*

Candidate 1 itemset	Item1	Item2	Item3
1-Count	6	6	7
Candidate 2 itemset	Item1, Item2	Item1, Item3	Item2, Item3
2-Count	3	5	3

2.6 CMS Opencart

Opencart adalah perangkat lunak *Content Management System* (CMS) berbasis *opensource* yang khusus dibuat untuk kebutuhan *e-commerce* atau pembuatan toko *online* (Bowo, 2013). Opencart dikembangkan oleh Daniel Kerr dan mulai diluncurkan pada bulan Oktober 2008. Sampai saat ini, Opencart sudah mencapai versi 1.5.6.4 yang dirilis pada tanggal 23 April 2014.