BAB II
DASAR TEORI

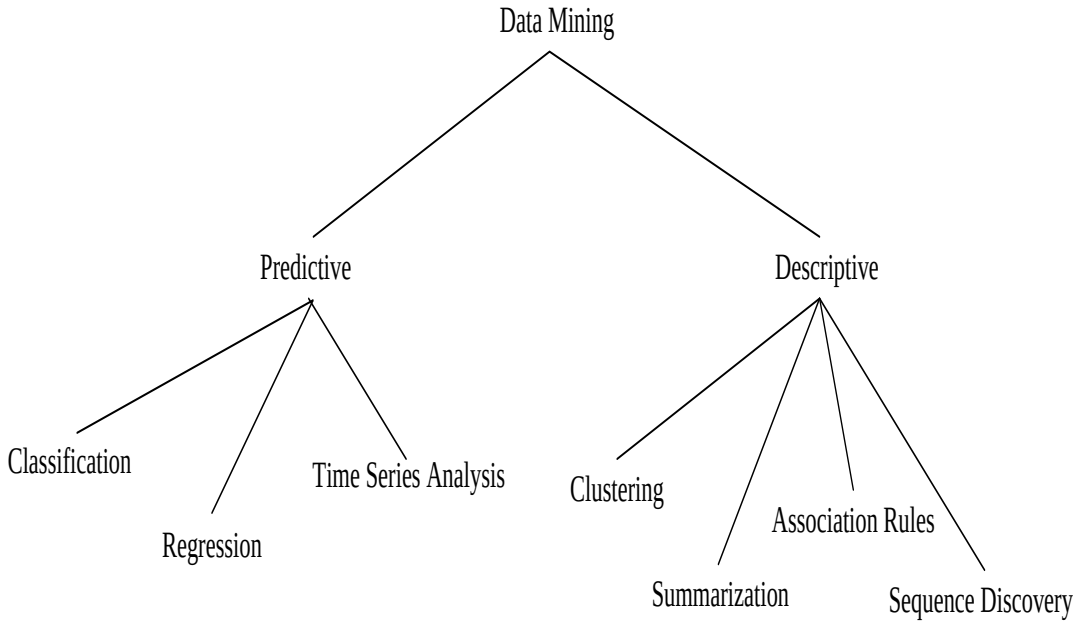
Bab ini menjelaskan tentang dasar teori yang digunakan dalam penyusunan tugas akhir ini, yaitu tentang pengertian *data mining*, pengelompokan *data mining*, metode klasifikasi dan penjelasan algoritma yang digunakan, yaitu *k-Nearest Neighborhood*.

2.1 Data Mining

Data mining merupakan proses menemukan pengetahuan yang menarik dari data yang berjumlah besar yang disimpan di dalam *database*, gudang data atau repositori informasi [1]. *Data mining* juga disebut sebagai serangkaian proses untuk menggali nilai tambah berupa pengetahuan yang selama ini tidak diketahui secara manual dari suatu kumpulan data (Pramudiono, 2007).

Secara umum, *data mining* dapat melakukan dua hal yaitu memberikan kesempatan untuk menemukan informasi menarik yang tidak terduga dan juga bisa menangani data berskala besar. Dalam menemukan informasi yang menarik ini, ciri khas *data mining* adalah kemampuan pencarian secara hampir otomatis, karena dalam banyak teknik *data mining* ada beberapa parameter yang masih harus ditentukan secara manual atau semi manual. *Data mining* juga dapat memanfaatkan pengalaman atau bahkan kesalahan di masa lalu untuk meningkatkan kualitas dari model maupun hasil analisanya, salah satunya dengan kemampuan pembelajaran yang dimiliki beberapa teknik *data mining* seperti klasifikasi.

2.1.1 Pengelompokan *Data Mining*



Gambar 2.1 Pengelompokan *Data Mining*

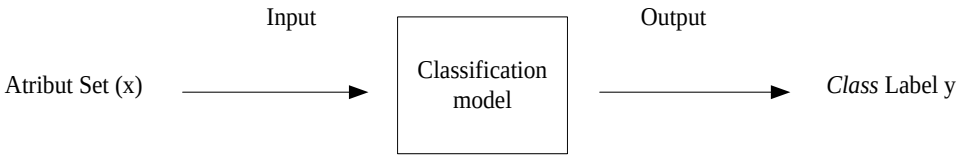
Secara umum, *data mining* terbagi dalam 2 sifat, yaitu:

1. *Predictive*: menghasilkan modal berdasarkan sekumpulan data yang dapat digunakan untuk memperkirakan nilai data yang lain. Metode-metode yang termasuk *predictive data mining* adalah:
   a. *Classification*: pembagian data kedalam beberapa kelompok yang telah ditentukan sebelumya.
   b. *Regression*: memetakan data ke suatu *prediction variable.*
   c. *Time Series Analysis*: pengamatan perubahan nilai atribut dari waktu ke waktu.
2. *Descriptive*: mengidentifikasikan pola atau hubungan dalam data untuk menghasilkan informasi baru. Metode yang termasuk dalam *descriptive data mining* adalah:
   a. *Clustering*: identifikasi hubungan antara data yang satu dengan lainnya.
   b. *Association Rules*: identifikasi hubungan antara data yang satu dengan lainnya.

c. *Summarization*: pemetaan data ke dalam subset dengan deskripsi sederhana.
d. *Sequence Discovery*: identifikasi pola sekuensial dalam data.

2.2 Klasifikasi

Klasifikasi adalah tugas pembelajaran sebuah fungsi target f yang memetakan setiap himpunan atribut x ke salah satu label *class* y yang telah didefinisikan sebelumnya [1]. Klasifikasi dapat juga diartikan suatu proses untuk menemukan suatu model atau fungsi yang menggambarkan dan membedakan kelas data atau konsep dengan tujuan dapat menggunakan model untuk memprediksi kelas objek yang label *class*-nya tidak diketahui.



Gambar 2.2 Model Klasifikasi

Data input untuk klasifikasi adalah isi dari *record*. Setiap *record* dikenal sebagai *instance* atau contoh, yang ditentukan oleh sebuah *tuple* (x, y), dimana x adalah himpunan atribut dan y adalah atribut tertentu, yang dinyatakan sebagai label *class* (juga dikenal sebagai kategori atau atribut target).

Pendekatan umum yang digunakan dalam masalah klasifikasi adalah pertama, *training data* berisi *record* yang mempunyai label *class* yang telah diketahui. *Training data* digunakan untuk membangun model klasifikasi yang kemudian diaplikasikan ke *testing data* yang berisi *record-record* dengan label *class* yang tidak diketahui.

2.3 Algoritma *Nearest Neighborhood*

Algoritma *Nearest Neighborhood* adalah pendekatan untuk mencari kasus dengan menghitung kedekatan antara kasus baru (*testing data*) dengan kasus lama

(*training data*), yaitu berdasarkan pada pencocokan bobot dari sejumlah fitur yang ada.

Jenis algoritma *Nearest Neighborhood* ada 2, yaitu:

1. 1-NN, yaitu pengklasifikasikan dilakukan terhadap 1 *labeled data* terdekat.
2. *k*-NN, yaitu pengklasifikasikan dilakukan terhadap *k labeled data* terdekat dengan *k*>1.

Penulis akan menggunakan algoritma *k-Nearest Neighborhood* dalam tugas akhir ini.

2.3.1 Algoritma *k-Nearest Neighborhood* (*k*-NN)

k-Nearest Neighborhood (*k*-NN) adalah suatu metode yang menggunakan algoritma *supervised* dimana hasil dari *query instance* yang baru diklasifikasikan berdasarkan mayoritas dari label *class* pada *k*-NN. Tujuan dari algoritma *k*-NN adalah mengklasifikasikan objek baru berdasarkan atribut dan *training data*.

Algoritma *k*-NN bekerja berdasarkan jarak terpendek dari *query instance* ke *training data* untuk menentukan *k*-NN-nya. Salah satu cara untuk menghitung jarak dekat atau jauhnya tetangga menggunakan metode *euclidian distance.*

Ecludian Distance sering digunakan untuk menghitung jarak. *Euclidian Distance* berfungsi menguji ukuran yang bisa digunakan sebagai interpretasi kedekatan jarak antara dua obyek, di bawah ini merupakan rumus *Ecludian Distance*:

$$\left(\sum_{k=1}^{m}\left(x_{ik}-x_{jk}\right)^2\right)^{1/2} \quad (2.1)$$

Dimana,

X_{ik} = nilai X pada *training data*

X_{jk} = nilai X pada *testing data*

m = batas jumlah banyaknya data

Jika hasil nilai dari rumus di atas besar maka akan semakin jauh tingkat keserupaan antara kedua objek dan sebaliknya jika hasil nilainya semakin kecil

maka akan semakin dekat tingkat keserupaan antar objek tersebut. Objek yang dimaksud adalah *training data* dan *testing data*.

Dalam algoritma ini, nilai *k* yang terbaik itu tergantung pada jumlah data. Ukuran nilai *k* yang besar belum tentu menjadi nilai *k* yang terbaik begitupun juga sebaliknya.

Langkah-langkah untuk menghitung algoritma *k*-NN:

1. Menentukan nilai *k*.
2. Menghitung kuadrat jarak *euclid* (*query instance*) masing-masing objek terhadap *training data* yang diberikan.
3. Kemudian mengurutkan objek-objek tersebut ke dalam kelompok yang mempunyai jarak *euclid* terkecil.
4. Mengumpulkan label *class* Y (klasifikasi *Nearest Neighborhood*).
5. Dengan menggunakan kategori *Nearest Neighborhood* yang paling mayoritas maka dapat dipredeksikan nilai *query instance* yang telah dihitung.

2.4 Diagram Konteks

Diagram konteks adalah kasus khusus DFD (bagian dari DFD yang berfungsi memetakan model lingkungan), yang dipresentasikan dengan lingkaran tunggal yang mewakili keseluruhan sistem [6]. Diagram konteks merupakan salah satu alat bantu dalam melakukan analisis terstruktur. Diagram konteks ini menggambarkan suatu sistem secara garis besarnya atau keseluruhannya saja. Dalam diagram konteks juga digambarkan entitas eksternal yang merupakan *brainware* yang menghasilkan data yang akan diolah oleh sistem maupun tujuan dari informasi yang dihasilkan oleh sistem.

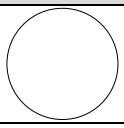
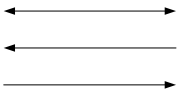
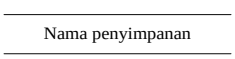
2.4.1 DFD (*Data Flow Diagram*)

DFD (*Data Flow Diagram*) merupakan model yang menggambarkan sistem sebagai jaringan kerja antar fungsi yang berhubungan satu sama lain dengan aliran

dan penyimpanan data [6]. DFD adalah penjelasan lebih rinci dari diagram konteks dan proses fungsional yang ada dalam sistem. DFD menjelaskan tentang aliran masuk, aliran keluar, proses serta penyuntingan file yang digunakan.

Ada 4 komponen dalam model DFD ini, yaitu:

Tabel 2.1 Komponen dalam DFD

Simbol	Keterangan
	**Proses**, menunjukkan transformasi dari masukan menjadi keluaran.
	**Aliran,** komponen ini dipresentasikan dengan menggunakan panah yang menuju kea tau dari proses. Digunakan untuk menggambarkan gerakan paket data atau informasi dari satu bagian ke bagian lain dari sistem dimana penyimpanan mewakili lokasi penyimpanan data.
	**Penyimpanan**, komponen ini digunakan untuk memodelkan kumpulan data atau paket data.
	**Terminator**, komponen berikut ini dipresentasikan menggunakan persegi panjang, yang mewakili entity luar dimana sistem berkomunikasi.

Keuntungan menggunakan DFD adalah memudahkan pengguna yang kurang atau menguasai bidang komputer untuk mengerti sistem yang akan dikerjakan atau dikembangkan.

2.5 *Software* Pendukung

2.5.1 *Visual Basic* 6.0

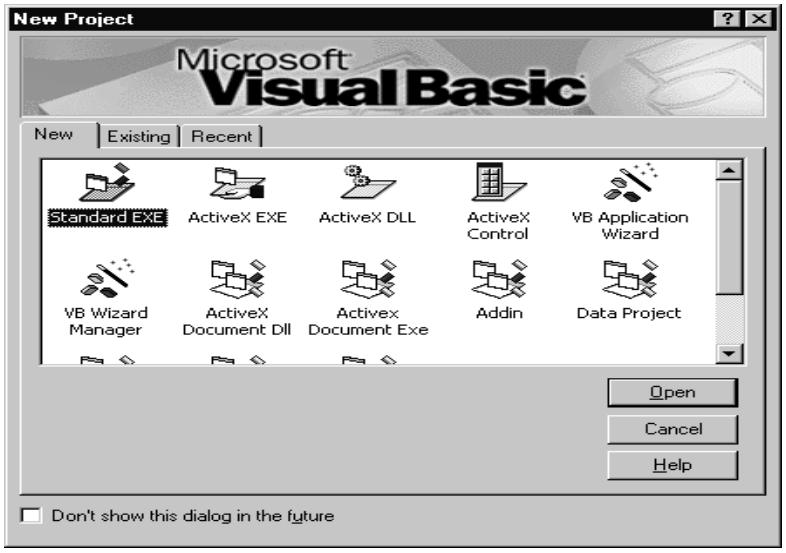
Visual basic 6.0 selain disebut sebagai bahasa pemrograman, juga sering disebut sebagai sarana untuk menghasilkan program-program aplikasi berbasiskan *Windows*. *Visual basic* pada dasarnya adalah bahasa pemrograman komputer. Bahasa pemrograman adalah perintah-perintah atau instruksi-instruksi yang dimengerti oleh komputer untuk melakukan tugas-tugas tertentu. *Visual basic* yang dikembangkan oleh *Microsoft* sejak tahun 1991 merupakan pengembangan

dari pendahulunya yaitu bahasa pemrograman BASIC (*Baginners All-purpose Symbolic Instruction Code*) yang dikembangkan pada era 1950-an. Beberapa kemampuan atau manfaat dari *visual basic* diantaranya:

1. Untuk membuat program aplikasi seperti *Windows*.
2. Untuk membuat objek-objek pembantu program seperti misalnya: kontrol *active*X, *file help*, aplikasi internet, dan sebagainya.
3. Menguji program (*debugging*) dan menghasilkan program EXE yang bersifat *executable*, atau dapat langsung dijalankan.

Langkah awal dari belajar *Visual Basic* adalah mengenal IDE (*Integrated Developement Environment*) *Visual Basic* yang merupakan lingkungan pengembangan terpadu bagi programmer dalam mengembangkan aplikasinya. Dengan menggunakan IDE programmer dapat membuat *user interface*, melakukan koding, melakukan *testing* dan *debuging* serta menkompilasi program menjadi *executable*. Penguasaan yang baik akan IDE akan sangat membantu programmer dalam mengefektifkan tugas-tugasnya sehingga dapat bekerja dengan efisien.

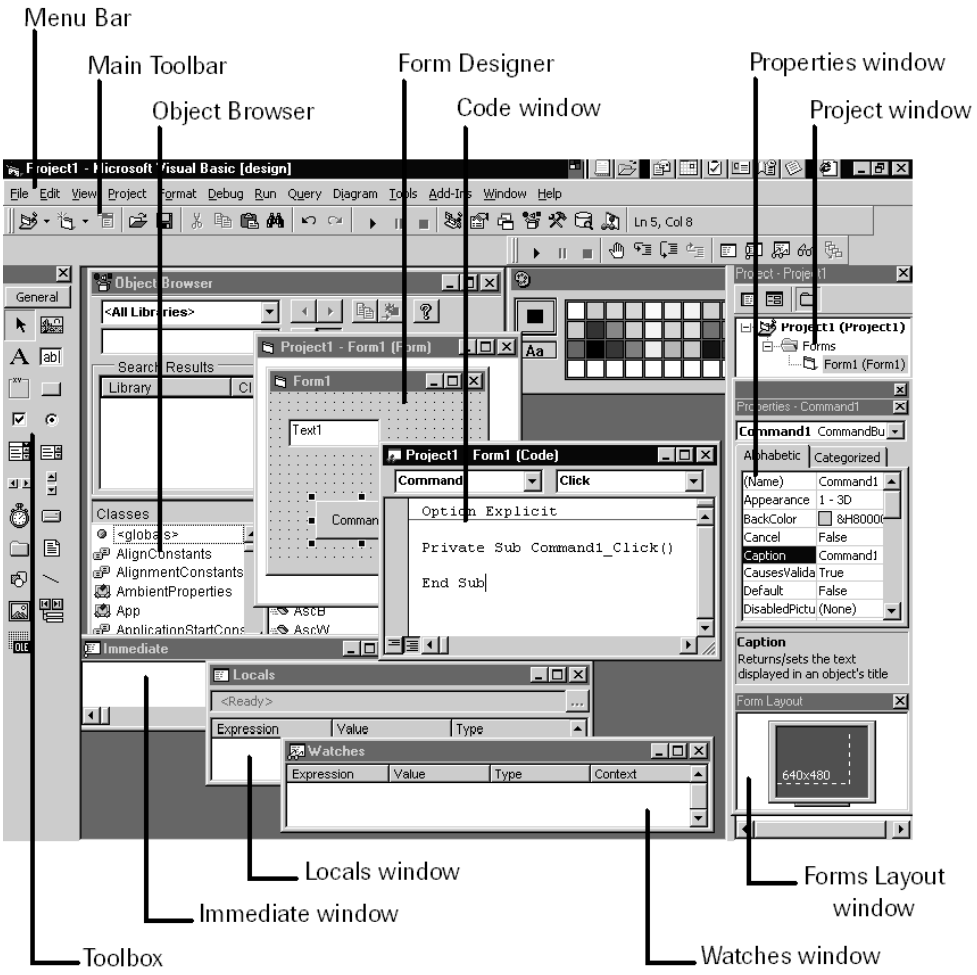
1. Memilih jenis Project



Gambar 2.3 Jenis Project

2. Jendela IDE

IDE *Visual Basic* 6.0 menggunakan model MDI (*Multiple Document Interface*). Berikut ini adalah gambar yang menunjukan bagian-bagian dan nama-nama jendela yang dapat tampil pada IDE *Visual Basic*.



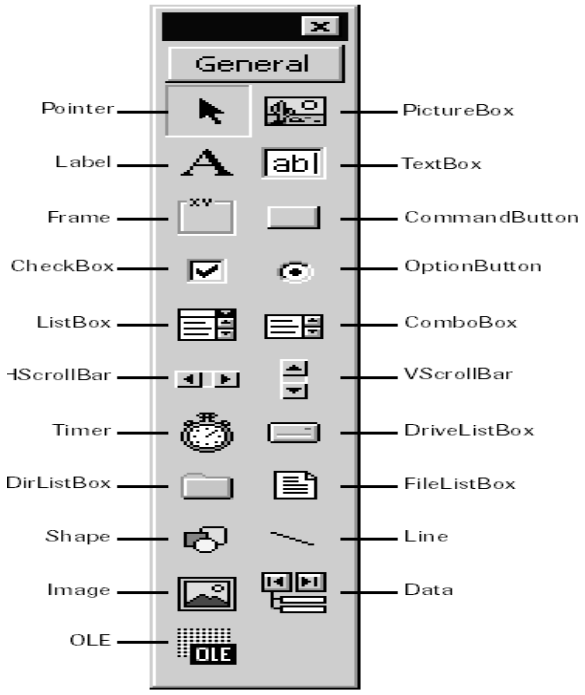
Gambar 2.4 Jendela IDE

Adapun jendela-jendela yang biasa digunakan adalah sebagai berikut:

- *Menu Bar*, digunakan untuk memilih tugas-tugas tertentu seperti menyimpan *project*, membuka *project*, dll
- *Main Toolbar*, digunakan untuk melakukan tugas-tugas tertentu dengan cepat.

- *Jendela Project*, jendela ini berisi gambaran dari semua modul yang terdapat dalam aplikasi anda. Anda dapat menggunakan *icon Toggle Folders* untuk menampilkan modul-modul dalam jendela tersebut secara di group atau berurut berdasarkan nama. Anda dapat menggunakan Ctrl+R untuk menampilkan jendela *project*, ataupun *menggunakan icon Project Explorer*.
- *Jendela Form Designer*, jendela ini merupakan tempat anda untuk merancang *user interface* dari aplikasi anda. Jadi jendela ini menyerupai kanvas bagi seorang pelukis.
- *Jendela Toolbox*, jendela ini berisi komponen-komponen yang dapat anda gunakan untuk mengembangkan *user interface*.
- *Jendela Code*, merupakan tempat bagi anda untuk menulis koding. Anda dapat menampilkan jendela ini dengan menggunakan kombinasi Shift-F7.
- *Jendela Properties*, merupakan daftar properti-properti objek yang sedang terpilih. Sebagai contohnya anda dapat mengubah warna tulisan (*foreground*) dan warna latar belakang (*background*). Anda dapat menggunakan F4 untuk menampilkan jendela properti.
- *Jendela Color Palette*, adalah fasilitas cepat untuk mengubah warna suatu objek.
- *Jendela Form Layout*, akan menunjukan bagaimana form bersangkutan ditampilkan ketika *runtime*.

3. Toolbox



Gambar 2.5 Toolbox

Adapun secara garis besar fungsi dari masing-masing *intrinsic* kontrol tersebut adalah sebagai berikut:

- *Pointer* bukan merupakan suatu kontrol; gunakan icon ini ketika anda ingin memilih kontrol yang sudah berada pada *form*.
- *PictureBox* adalah kontrol yang digunakan untuk menampilkan image dengan format: BMP, DIB (*bitmap*), ICO (*icon*), CUR (*cursor*), WMF (*metafile*), EMF (*enhanced metafile*), GIF, dan JPEG.
- *Label* adalah kontrol yang digunakan untuk menampilkan teks yang tidak dapat diperbaiki oleh pemakai.
- *TextBox* adalah kontrol yang mengandung string yang dapat diperbaiki oleh pemakai, dapat berupa satu baris tunggal, atau banyak baris.
- *Frame* adalah kontrol yang digunakan sebagai kontainer bagi kontrol lainnya.

- *CommandButton* merupakan kontrol hampir ditemukan pada setiap form, dan digunakan untuk membangkitkan *event* proses tertentu ketika pemakai melakukan klik padanya.
- *CheckBox* digunakan untuk pilihan yang isinya bernilai *yes/no*, *true/false*.
- *OptionButton* sering digunakan lebih dari satu sebagai pilihan terhadap beberapa option yang hanya dapat dipilih satu.
- *ListBox* mengandung sejumlah item, dan user dapat memilih lebih dari satu (bergantung pada *property MultiSelect*).
- *ComboBox* merupakan konbinasi dari *TextBox* dan suatu *ListBox* dimana pemasukkan data dapat dilakukan dengan pengetikkan maupun pemilihan.
- *HScrollBar* dan *VScrollBar* digunakan untuk membentuk *scrollbar* berdiri sendiri.
- *Timer* digunakan untuk proses *background* yang diaktifkan berdasarkan interval waktu tertentu. Merupakan kontrol non-visual.
- *DriveListBox*, *DirListBox*, dan *FileListBox* sering digunakan untuk membentuk dialog box yang berkaitan dengan file.
- *Shape* dan *Line* digunakan untuk menampilkan bentuk seperti garis, persegi, bulatan, oval.
- *Image* berfungsi menyerupai *image box*, tetapi tidak dapat digunakan sebagai kontainer bagi kontrol lainnya. Sesuatu yang perlu diketahui bahwa kontrol *image* menggunakan *resource* yang lebih kecil dibandingkan dengan *PictureBox*
- *Data* digunakan untuk *data binding*
- *OLE* dapat digunakan sebagai tempat bagi program eksternal seperti *Microsoft Excel*, *Word*, dll.

2.5.2 *Microsoft Access* 2007

Microsoft Access adalah suatu program *Microsoft Office* pengolahan *database*. Para pengguna *Microsoft Access* umumnya menggunakan program hanya untuk membuat tabel *database*. Padahal, di dalam *Microsoft Access* tersedia

fasilitas yang bisa digunakan untuk membuat laporan dalam bentuk tabel, *query*, maupun *report database* secara maksimal.

Jenis-jenis *Query*:

1. *Select queries.*

   Query ini menampilkan ekstrak data atau seluruh data. Data yang ditampilkan bias dipilih sesuai kriteria tertentu. Selain itu, bisa menampilkan *field* hasil perhitungan dengan formula tertentu.

2. *Action queries.*

   Query ini digunakan untuk melakukan *delete, update, insert* data, dan *create* tabel baru.

3. *Cross tab queries.*

   Query ini digunakan untuk mengkakulasi data untuk keperluan analisis. *Crosstab queries* menghitung sebuah *count, average, sum*, atau tipe total lainnya yang dikelompokan ke dalam dua jenis informasi.