KLASIFIKASI DATA MINING

Disusun Guna Memenuhi Tugas Mata Kuliah Data Mining

Dosen Pengampu :

Asep Muhidin, S.Kom, M.Kom

OLEH :

NAMA : DIAN HARDIANSYAH

NIM : 312110084

KELAS : TI.21.C.4

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PELITA BANGSA
TAHUN AJARAN 2023-2024

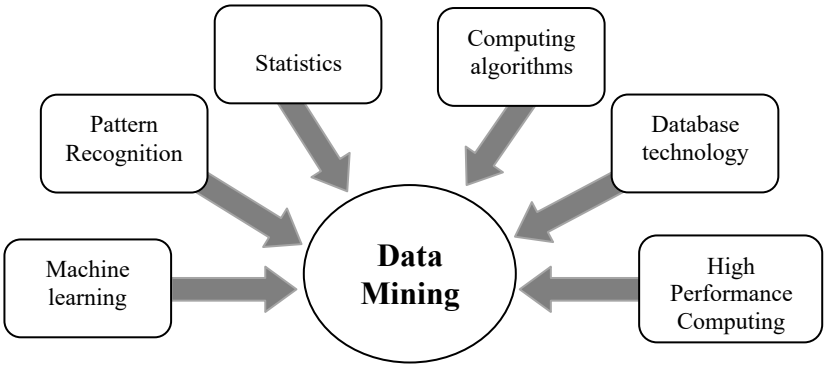
PENDAHULUAN

Kebutuhan informasi yang tinggi kadang tidak sebanding dengan penyajian informasi yang memadai. Informasi yang disajikan sering kali masih harus digali dari data dalam jumlah besar. Salah satu contoh yaitu data yang tumbuh dalam bidang kesehatan. Data kesehatan menyimpan banyak sekali data-data yang terkait dalam lingkungan kesehatan seperti data pasien, data obat, data penyakit, yang sangat penting untuk dapat diolah supaya lebih bermanfaat. Metode tradisional yang biasa digunakan untuk menganalisis data, tidak dapat menangani data dalam jumlah besar. Oleh karena itu data tersebut dapat diolah menjadi pengetahuan menggunakan teknik yang disebut *data mining*. Sebagai bidang ilmu yang relatif baru, *data mining* menjadi pusat perhatian para akademisi maupun praktisi. Beragam penelitian dan pengembangan *data mining* banyak diaplikasikan pada bidang kesehatan. Bab ini memberikan pandangan secara singkat mengenai definisi *data mining*, *dataset*, jenis *dataset*, dan jenis atribut.

A. Definisi *Data Mining*

Data mining dikenal sejak tahun 1990-an, ketika adanya suatu pekerjaan yang memanfaatkan data menjadi suatu hal yang lebih penting dalam berbagai bidang, seperti marketing dan bisnis, sains dan teknik, serta seni dan hiburan. Sebagian ahli menyatakan bahwa *data mining* merupakan suatu langkah untuk menganalisis pengetahuan dalam basis data atau biasa disebut *Knowledge Discovery in Database* (KDD). *Data mining* merupakan proses untuk menemukan pola data dan pengetahuan yang menarik dari kumpulan data yang sangat besar. Sumber data dapat mencakup *database*, *data warehouse*, *web*, *repository*, atau data yang dialirkan ke dalam sistem dinamis (Han, 2006).

Data mining, secara sederhana merupakan suatu langkah ekstraksi untuk mendapatkan informasi penting yang sifatnya implisit dan belum diketahui. Selain itu, *data mining* mempunyai hubungan dengan berbagai bidang diantaranya statistik, *machine learning* (pembelajaran mesin), *pattern recognition*, *computing algorithms*, *database technology*, dan *high performance computing*. Diagram hubungan *data mining* disajikan pada Gambar 1.1.



Gambar 1.1. Diagram hubungan *data mining*

Secara sistematis, langkah utama untuk melakukan *data mining* terdiri dari tiga tahap, yaitu sebagai berikut (Gonunescu, 2011);

1. Eksplorasi atau pemrosesan awal data

   Eksplorasi atau pemrosesan awal data terdiri dari pembersihan data, normalisasi data, transformasi data, penanganan *missing value*, reduksi dimensi, pemilihan subset fitur, dan sebagainya.

2. Membangun model dan validasi

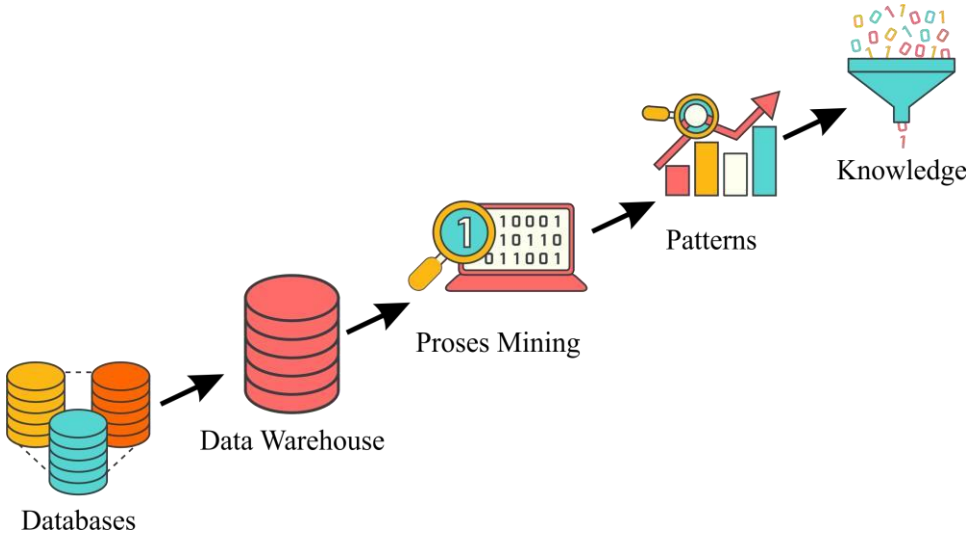
   Membangun model dan validasi, yaitu melakukan analisis dari berbagai model dan memilih model sehingga menghasilkan kinerja yang terbaik.

   Pembangunan model dilakukan menggunakan metode-metode seperti klasifikasi, regresi, analisis *cluster*, dan asosiasi.

3. Penerapan

   Penerapan dilakukan dengan menerapkan model yang dipilih pada data yang baru untuk menghasilkan kinerja yang baik pada masalah yang diinvestigasi.

Tahapan proses *data mining* ada beberapa yang sesuai dengan proses KDD sebagaimana seperti yang digambarkan pada Gambar 1.2.



Gambar 1.2. Proses KDD

B. Teknik *Data mining*

Teknik *data mining* merupakan suatu proses utama yang digunakan saat metode diterapkan untuk menemukan pengetahuan berharga dan tersembunyi dari data. Dengan definisi *data mining*, ada beberapa teknik dan sifat analisa yang dapat digolongkan dalam *data mining* yaitu, sebagai berikut.

C. *Classification* (Klasifikasi)

Klasifikasi dapat didefinisikan sebagai suatu proses yang melakukan pelatihan/pembelajaran terhadap fungsi target *f* yang memetakan setiap vektor (set fitur) *x* ke dalam satu dari sejumlah label kelas *y* yang tersedia. Di dalam klasifikasi diberikan sejumlah *record* yang dinamakan *training* set, yang terdiri dari beberapa atribut, atribut dapat berupa kontinyu ataupun kategoris, salah satu atribut menunjukkan kelas untuk *record.* Model klasifikasi dapat dilihat pada Gambar 2.2.

Gambar 2.2. Blok diagram model klasifikasi

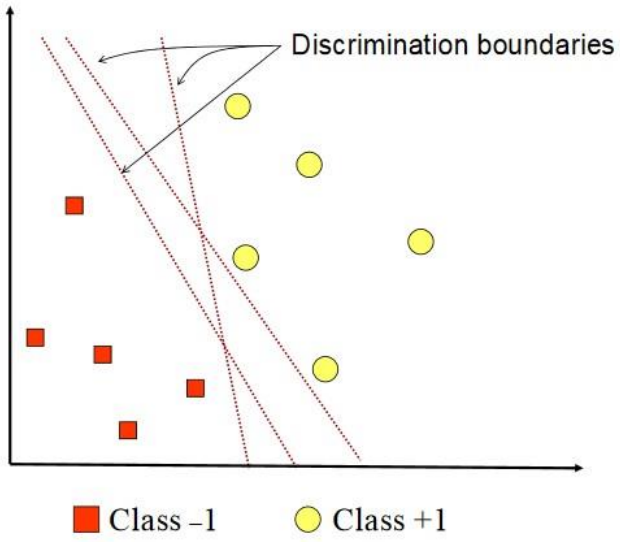
Klasifikasi merupakan teknik yang digunakan untuk menemukan model agar dapat menjelaskan atau membedakan konsep atau kelas data, dengan tujuan untuk dapat memperkirakan kelas dari suatu objek yang labelnya tidak diketahui. Metode klasifikasi yang sering digunakan yaitu, *Support Vector Machine, Multilayer Perceptron, Naive bayes, ID3, Ensemble Methode,* dll.

1. *Support Vector Machine (SVM)*

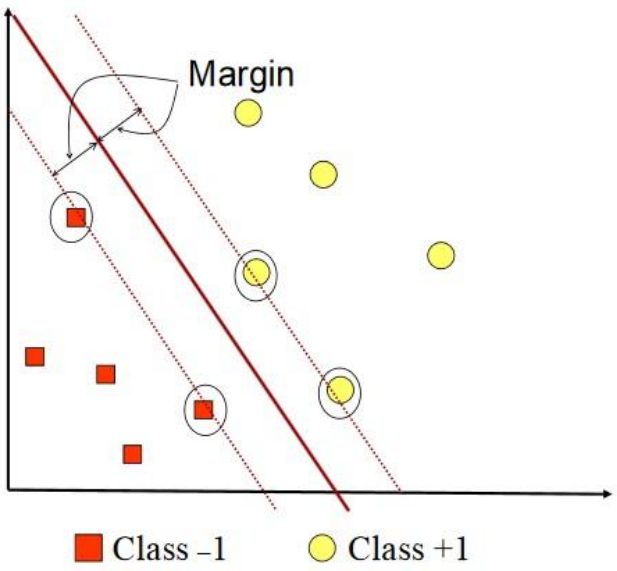
Support Vector Machine (SVM) dikembangkan pada tahun 1992 oleh Vladimir Vapnik bersama dengan kedua rekannya, Bernhard Boser dan Isabelle Guyon. SVM dikembangkan dari teori *structural risk minimalization*. Dengan menggunakan trik kernel untuk memetakan sampel pelatihan dari ruang *input* ke ruang fitur yang berdimensi tinggi (Li *et al.*, 2008: 786). Menurut Han *et al.,* (2012: 408) metode SVM menjadi sebuah metode baru yang menjanjikan untuk pengklasifikasian data, baik data *linear* maupun data *nonlinear*.

Konsep SVM dapat dijelaskan secara sederhana sebagai usaha mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah *class* pada *input space*. Gambar 2.3 memperlihatkan beberapa *pattern* yang merupakan anggota dari dua buah *class,* yaitu: +1 dan –1. *Pattern* yang tergabung pada *class* –1 disimbolkan dengan warna merah (kotak), sedangkan *pattern* pada *class* +1, disimbolkan dengan warna kuning (lingkaran). Problem klasifikasi dapat diterjemahkan dengan usaha menemukan garis *(hyperplane)* yang memisahkan antara kedua kelompok tersebut. Berbagai alternatif garis pemisah *(discrimination boundaries)* ditunjukkan pada Gambar 2.3. *Hyperplane* pemisah terbaik antara kedua *class* dapat ditemukan dengan mengukur *margin hyperplane* tersebut dan mencari titik maksimalnya. *Margin* adalah jarak antara *hyperplane* tersebut dengan *pattern* terdekat dari masing-masing *class*. *Pattern* yang paling dekat ini disebut sebagai *support vector*. Garis solid pada Gambar 2.4 menunjukkan *hyperplane* yang terbaik, yaitu yang terletak tepat pada tengahtengah kedua *class*, sedangkan titik merah dan kuning yang berada

dalam lingkaran hitam adalah *support vector*. Usaha untuk mencari lokasi *hyperplane* ini merupakan inti dari proses pembelajaran pada SVM (Nugroho *et al.,* 2003).



Gambar 2.3. *Discrimination boundaries*



Gambar 2.4. *Hyperplane*

Langkah awal suatu algoritma SVM adalah pendefinisian persamaan suatu *hyperplane* pemisah yang dituliskan dengan persaman berikut.

$$w.X + b = 0$$

Keterangan:

w : bobot vektor, $w = \{w_1, w_2, \ldots, w_n\}$;

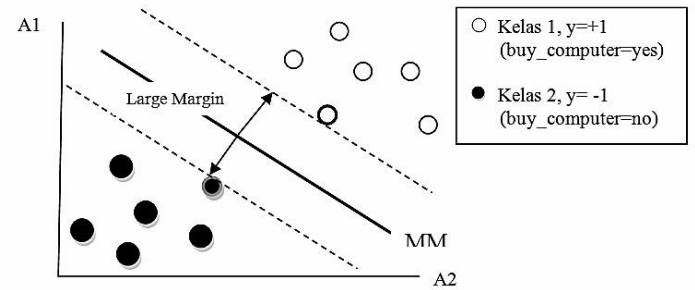
b : skalar yang disebut dengan bias. Jika berdasarkan pada atribut A_1, A_2 dengan permisalan tupel pelatihan

X : (x_1, x_2), x_1 dan x_2 merupakan nilai dari atribut A_1 dan A_2, dan jika b dianggap sebagai suatu bobot tambahan w_0,

maka persamaan suatu *hyperplane* pemisah dapat ditulis ulang seperti pada persamaan berikut.

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

Setelah persamaan dapat didefinisikan, nilai x_1 dan x_2 dapat dimasukkan ke dalam persamaan untuk mencari bobot w_1, w_2, dan w_0 atau b. Grafik pemisahan dua kelas data dengan margin maksimum dapat dilihat pada Gambar 2.5.



Gambar 2.5. Pemisahan dua kelas data dengan margin maksimum

Pada gambar di atas, dijelaskan bahwa SVM menemukan *hyperplane* pemisah maksimum, yaitu *hyperplane* yang mempunyai jarak maksimum antara tupel pelatihan terdekat. *Support vector* ditunjukkan dengan batasan tebal pada titik tupel. Dengan demikian, setiap titik yang letaknya di atas *hyperplane* pemisah memenuhi persamaan berikut.

$$w_0 + w_1 x_1 + w_2 x_2 > 0$$

Sedangkan, titik yang letaknya di bawah *hyperplane* pemisah memenuhi rumus seperti pada persamaan berikut.

$$w_0 + w_1 x_1 + w_2 x_2 < 0$$

Jika dilihat dari dua kondisi di atas, maka didapatkan dua persamaan *hyperplane*, seperti pada persamaan yang ada di bawah ini.

$H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 0$ unttuk y_i=+1

$H_2: w_0 + w_1 x_1 + w_2 x_2 \leq 0$ untuk y_i=-1

Dengan demikian, setiap *tuple* yang berada di atas H_1 memikiki kelas +1, dan setiap *tuple* yang berada di bawah H_2 memiliki kelas -1. Perumusan model SVM menggunakan trik matematika yaitu *lagrangian formulation*. Berdasarkan *lagrangian formulation*, *Maksimum Margin Hyperplane* (MMH) dapat ditulis ulang sebagai suatu batas keputusan (*decision boundary*) dituliskan dengan persamaan berikut.

$$d(X^T) = \sum_{i=1}^{l} y_i a_i \; X_i X^T + b_0$$

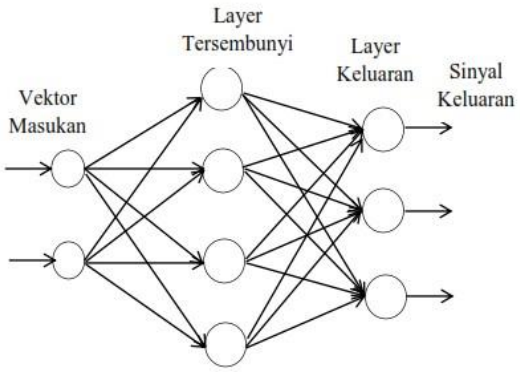
y_i adalah label kelas dari support vector X_i, X^T merupakan suatu tuple test. $a_i$dan b_o adalah parameter numerik yang ditentukan secara otomatis oleh optimalisasi algoritma SVM dan $l$adalah jumlah *support vector*.

2. Multi Layer Perceptron

Multi Layer Perceptron (MLP) merupakan turunan algoritma Artificial Neural Network (ANN) dari perceptron, berupa ANN feed forward dengan satu atau lebih dari hidden layer. Biasanya, jaringan terdiri dari satu layer masukan, setidaknya satu layer neuron komputasi di tengah (hidden layer) dan sebuah layer neuron komputasi keluaran. Sinyal masukan dipropagasikan dengan arah maju pada layer per layer.

Sebenarnya ada satu layer lagi dalam MLP, yaitu layer masukan berupa vektor masukan. Akan tetapi, karena dalam layer ini tidak ada komputasi, yang dilakukan hanya meneruskan sinyal/vektor masukan yang diterima ke

layer di depannya. Contoh arsitektur MLP diberikan pada Gambar 2.6. Pada gambar tersebut ada satu hidden layer dengan empat neuron, dan satu *layer* keluaran dengan tiga *neuron*.



Gambar 2.6. Arsitektur *Multilayer Perceptron*

Setiap layer dalam MLP masing-masing mempunyai fungsi khusus. Vektor masukan berfungsi menerima sinyal/vektor masukan dari luar dan mendistribusikannya ke semua *neuron* dalam *hidden layer*. *Layer* keluaran berfungsi untuk menerima sinyal keluaran atau dengan kata lain stimulus pola dari *hidden layer* dan memunculkan sinyal/nilai/kelas keluaran dari keseluruhan jaringan.

Tahapan pada algoritma *multilayer perceptron*, yaitu sebagai berikut.

a. Langkah 1: Inisialisasi

Inisialisasi semua bobot pada *hidden layer* keluaran, lalu menetapkan fungsi aktivasi yang digunakan untuk setiap *layer*, kemudian tetapkan laju pembelajaran. Untuk inisialisasi semua bobot bisa menggunakan bilangan acak dalam jangkauan [-1,1] atau [-0.5,0.5] selain itu juga bisa menggunakan inisialisasi distribusi seragam dalam jangkauan kecil (tidak ada aturan baku mengenai interval tersebut). Berikut adalah contoh inisialisasi distribusi seragam.

. $\left(-\frac{2.4}{F_i}, +\frac{2.4}{F_i}\right)$, dengan F_i adalah jumlah *neuron* masukan 1 dalam ANN.

b. Langkah 2: Aktivasi

Mengaktivasi jaringan dengan menerapkan inputan $x_1(p)$, $x_2(p)$, ..., $x_n(p)$ dan keluaran yang diharapkan $y_{d1}(p)$, $y_{d2}(p)$, ..., $y_{dn}(p)$

a) Menghitung keluaran yang didapatkan dari *neuron* dalam *hidden layer*, dengan persamaan berikut:

$$v_j(p) = \sum_{i=1}^{r} \delta_k(p).w_{jk}(p)$$

$$y_j(p) = \frac{1}{1 + e^{-v_j(p)}}$$

r adalah jumlah fitur data inputan pada *neuron j* dalam *hidden layer*.

b) Menghitung keluaran yang didapatkan dari *neuron* dalam *layer* keluaran, menggunakan persamaan berikut ini:

$$v_k(p) = \sum_{j=1}^{m} x_j(p).w_{jk}(p)$$

$$y_k(p) = \frac{1}{1 + e^{-v_k(p)}}$$

m adalah jumlah masukan pada *neuron k* dalam *layer* keluaran.

c. Langkah 3: Perbarui bobot

Bobot diperbarui pada saat *error* dirambatkan balik dalam ANN, *error* dikembalikan sesuai dengan arah keluarnya sinyal keluaran.

a) Menghitung *gradien error* untuk *neuron* dalam *layer* keluaran, menggunakan persamaan berikut:

$$e_k(p) = y_{dk}(p) - y_k(p)$$

$$\delta_k(p) = y_k(p)x(1 - y_k(p))xe_k(p)$$

Kemudian menghitung koreksi bobot:

$$\Delta w_{jk}(p) = \pi x y_j(p) + \delta_k(p)$$

Memperbarui bobot pada *neuron layer* keluaran:

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p)$$

b) Hitung *gradien error* untuk *neuron* dalam *hidden layer* menggunakan persamaan berikut ini:

$$\delta_j(p) = y_j(p)x[1 - y_j(p)] + \sum_{k=1}^{i} \delta_k(p).w_{jk}(p)$$

Menghitung koreksi bobot:

$$\Delta w_{ij}(p) = \pi x x_j(p) + \delta_j(p)$$

Memperbarui bobot pada *neuron layer* keluaran:

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

3. *Naive Bayes*

a. Teorema *Bayes*

Bayes merupakan teknik prediksi berbasis probalistik sederhana yang berdasar pada penerapan teorema *bayes* atau aturan *bayes* dengan asumsi independensi (ketidaktergantungan) yang kuat (*naïve*). Dengan kata lain, *Naïve Bayes* adalah model fitur independen (Prasetyo, 2012: 59). Dalam *Bayes* (terutama *Naïve Bayes*), maksud independensi yang kuat pada fitur adalah bahwa sebuah fitur pada sebuah data tidak berkaitan dengan ada atau tidaknya fitur lain dalam data yang sama. Prediksi *Bayes* didasarkan pada teorema *Bayes* dengan formula umum dengan persamaan berikut.

$$P(H|E) = \frac{P(E|H)\,P(H)}{P(E)}$$

Penjelasan formula diatas sebagai berikut:

Parameter	Keterangan
P(H|E)	Probabilitas bebas bersyarat (*conditional probability)* suatu hipotesis H jika diberikan bukti *(Evidence)* E terjadi.
P(E|H)	Probabilitas sebuah bukti E terjadi akan mempengaruhi hipotesis H
P(H)	Probabilitas awal (priori) hipotesis H terjadi tanpa memandang bukti apapun
P(E)	Probabilitas awal (priori) bukti E terjadi tanpa memandang hipotesis/bukti yang lain

Ide dasar dari aturan *Bayes* adalah bahwa hasil dari hipotesis atas peristiwa (H) dapat diperkirakan berdasarkan pada beberapa bukti (E) yang diamati. Ada beberapa hal penting dalam aturan *Bayes* tersebut yaitu,

1) Sebuah probabilitas awal/prior H atau P(H) adalah probabilitas dari suatu hipotesis sebelum bukti diamati.
2) Sebuah probabilitas akhir H atau P(H|E) adalah probabilitas dari suatu hipotesis setelah bukti diamati.

b. *Naïve Bayes* untuk Klasifikasi

Prasetyo (2012: 61) menjelaskan kaitan antara *Naïve Bayes* dengan klasifikasi, korelasi hipotesis dan bukti klasifikasi adalah bahwa hipotesis dalam teorema *bayes* merupakan label kelas yang menjadi target pemetaan dalam klasifikasi, sedangkan bukti merupakan fitur-fitur yang menjadikan masukan dalam model klasifikasi. Jika X adalah vektor masukkan yang berisi fitur dan Y adalah label kelas, *Naïve Bayes* dituliskan dengan P(X|Y). Notasi tersebut berarti probabilitas label kelas Y didapatkan setelah fitur-fitur X diamati. Notasi ini disebut juga probabilitas akhir (*posterior probability*) untuk Y, sedangkan P(Y) disebut probabilitas awal (*prior probability*) Y.

Selama proses pelatihan harus dilakukan pembelajaran probabilitas akhir P(Y|X) pada model untuk setiap kombinasi X dan Y berdasarkan informasi yang didapat dari data latih. Dengan membangun model tersebut, suatu data uji X' dapat diklasifikasikan dengan mencari nilai Y' dengan memaksimalkan P(Y'|X') yang didapat. Formulasi *Naïve Bayes* untuk klasifikasi yaitu pada persamaan berikut.

$$\mathrm{P(Y|X)} = \frac{\mathrm{P(Y)} \prod_{i=1}^{q} \mathrm{P(X}_i\mathrm{|Y)}}{\mathrm{P(X)}}$$

P(X|Y) adalah probabilitas data dengan vektor X pada kelas Y. P(Y) adalah probabilitas awal kelas Y. $\prod_{i=1}^{q} \mathrm{P(X}_i\mathrm{|Y)}$ adalah probabilitas independen kelas Y dari semua fitur dalam vetor X. Nilai P(X) selalu

tepat sehingga dalam perhitungan prediksi nantinya kita tinggal menghitung bagian P(Y) $\prod_{i=1}^{q} \mathrm{P}(\mathrm{X}_i|\mathrm{Y})$ dengan memilih yang terbesar sebagai kelas yang dipilih sebagai hasil prediksi. Sementara probabilitas independen $\prod_{i=1}^{q} \mathrm{P}(\mathrm{X}_i|\mathrm{Y})$ tersebut merupakan pengaruh semua fitur dari data terhadap setiap kelas Y, yang dinotasikan dengan Persamaan berikut.

$$\mathrm{P}(\mathrm{X}|\mathrm{Y}=\mathrm{y}) = \prod_{i=1}^{q} \mathrm{P}(\mathrm{X}_i|\mathrm{Y}=\mathrm{y})$$

Setiap set fitur $\mathrm{X}=\{\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \ldots, \mathrm{x}_\mathrm{q}\}$ terdiri atas *q* atribut (*q* dimensi).

Umumnya, *Bayes* mudah dihitung untuk fitur bertipe ketegoris seperti pada kasus klasifikasi hewan dengan fitur "penutup kulit" dengan nilai {bulu, rambut, cangkang} atau kasus fitur "jenis kelamin" dengan nilai {pria, wanita}. Namun untuk fitur dengan tipe numerik (kontinyu) ada perlakuan khusus sebelum dimasukan dalam *Naïve Bayes*. Caranya yaitu:

1) Melakukan diskretisasi pada setia fitur kontinyu dan mengganti nilai fitur kontinyu tersebut dengan nilai interval diskret. Pendekatan ini dilakukan dengan mentransformasikan fitur kontinyu ke dalam fitur ordinal.
2) Mengasumsi bentuk tertentu dari distribusi probabilitas untuk fitur kontinyu dan memperkirakan parameter distribusi dengan data pelatihan. Distribusi Gaussian biasanya dipilih untuk merepresentasikan probabilitas bersyarat dari fitur kontinyu pada sebuah kelas $\mathrm{P}(\mathrm{X_i}|\mathrm{Y})$, sedangkan distribusi Gaussian dikarakteristikkan dengan dua parameter, yaitu: *mean* (μ) dan *varian* (σ^2).

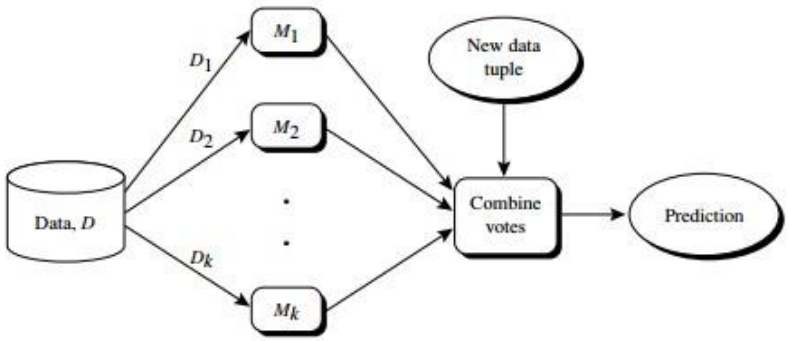
   Untuk setiap kelas Y_j, probabilitas bersyarat kelas Y_j untuk fitur $\mathrm{X_i}$ adalah seperti pada persamaan berikut.

$$\mathrm{P}\left(\mathrm{X_i}=\mathrm{x_i}\middle|\mathrm{Y_j}=\mathrm{y_j}\right) = \frac{1}{\sqrt{2\pi\sigma_{\mathrm{ij}}}} \exp^{-\frac{\left(\mathrm{x_i}-\mu_{\mathrm{ij}}\right)^2}{2\sigma_{\mathrm{ij}}^2}}$$

   Parameter μ_{ij} bisa didapat dari *mean* sampel $\mathrm{X_i}$ $(\bar{x})$ dari semua data latih yang menjadi milik kelas Y_j, sedangkan σ_{ij}^2 dapat diperkirakan dari *varian* sampel (s^2) dari data latih.

4. Ensemble Methode

Ensemble method merupakan salah satu metode yang digunakan untuk meningkatkan akurasi algoritma klasifikasi dengan membangun beberapa *classifier* dari data *training* kemudian pada saat proses klasifikasi metode ini menggunakan *voting/aggregating* dari *classifier-classifer* yang digunakan (Tan, *et al*., 2006: 276). Han *et al.,* (2012: 378) dalam bukunya, menjelaskan bahwa *bagging*, *boosting*, dan *random forest* adalah contoh metode *ensemble*. Metode *ensemble* yaitu menggabungkan serangkaian k model pembelajaran (dasar klasifikasi), seperti:M_1, M_2, ... , M_k. Diberikan sebuah *dataset* D yang digunakan untuk membuat k *data training*, yaitu D_1, D_2, ... , D_k, di mana D_i ($1 \leq i \leq k-1$) digunakan untuk menghasilkan *classifier* M_i . Ilustrasi metode *ensemble* ditunjukkan pada Gambar 2.5.



Gambar 2.5. *Ensemble Method*

Teknik ensemble merupakan teknik yang sukses untuk menangani *dataset* yang tidak seimbang meskipun tidak secara khusus dirancang untuk masalah data yang tidak seimbang. Teknik *bagging* merupakan salah satu teknik *ensemble*. Teknik *ensemble* pada klasifikasi digunakan untuk memisahkan data *training* ke dalam beberapa data *training* baru dengan *random sampling* dan membangun model berbasis data *training* baru (Wahono & Suryana, 2013: 157).

a. Bootstrap Aggregating (Bagging)

Bagging ditemukan oleh Breiman (1996) yang merupakan kepanjangan dari

"*bootstrap aggregating*" dan juga salah satu teknik yang dapat digunakan pada beberapa metode klasifikasi dan regresi untuk mereduksi variansi dari suatu prediktor, dan dengan demikian dapat memperbaiki proses prediksi (Sutton, 2005: 176). Han *et al.,* (2012: 379) dalam bukunya menyatakan bahwa *bagging* adalah salah satu teknik dari *ensemble method* dengan cara memanipulasi data *training*, data *training* diduplikasi sebanyak *d* kali dengan pengembalian (*sampling with replacement*), yang akan menghasilkan sebanyak *d* data *training* yang baru, kemudian dari *d* data *training* tersebut akan dibangun *classifierclassifier* yang disebut sebagai *bagged classifier*. Karena data pada teknik *bagging* dilakukan *sampling with replacement*, ukuran data *bagging* sama dengan data aslinya, tetapi distribusi data dari tiap data *bagging* berbeda, beberapa data dari data *training* bisa saja muncul beberapa kali atau mungkin tidak muncul sama sekali (Han & Kamber, 2001: 390). Hal inilah yang menjadi kunci kenapa *bagging* bisa meningkatkan akurasi karena dengan *sampling with replacement* dapat memperkecil *variance* dari *dataset* (Tan, et al., 2006: 283).

Berdasarkan namanya, maka ada dua tahapan utama dalam analisis ini, yaitu *bootstrap* yang tidak lain adalah pengambilan sampel dari data *learning* yang dimiliki (*resampling*) dan *aggregating* yaitu menggabungkan banyak nilai dugaan menjadi satu nilai dugaan. Pada kasus klasifikasi, mengambil suara terbanyak (*majority vote*) merupakan salah satu proses menggabungkan nilai dugaan dari beberapa pohon menjadi satu dugaan akhir, sedangkan untuk kasus regresi menggunakan rata-rata. Dengan demikian proses pembuatan dugaan secara *bagging* menggunakan pohon menurut Sutton (2005: 178) adalah sebagai berikut:

1) Tahapan *bootstrap*
    a) Tarik sampel acak dengan pemulihan berukuran n dari gugus data *learning*.
    b) Susun pohon terbaik berdasarkan data tersebut.

c) Ulangi langkah a dan b sebanyak *B* kali sehingga diperoleh *B* buah pohon klasifikasi

.

2) Tahapan *aggregating*

Lakukan pendugaan gabungan berdasarkan *B* buah pohon klasifikasi tersebut menggunakan aturan *majority vote* (suara terbanyak).

Berdasarkan namanya, maka ada dua tahapan utama dalam analisis ini, yaitu *bootstrap* yang tidak lain adalah pengambilan sampel dari data *learning* yang dimiliki (*resampling*) dan *aggregating* yaitu menggabungkan banyak nilai dugaan menjadi satu nilai dugaan. Pada kasus klasifikasi, mengambil suara terbanyak (*majority vote*) merupakan salah satu proses menggabungkan nilai dugaan dari beberapa pohon menjadi satu dugaan akhir, sedangkan untuk kasus regresi menggunakan rata-rata. De demikian proses pembuatan dugaan secara *bagging* menggunakan pohon adalah sebagai berikut;

- Mempersiapkan *dataset*. *Dataset* bisa diambil dari *UCI repository of machine learning*.
- Melakukan pembagian *dataset* menjadi data latih dan data uji menggunakan *k-fold cross validation.*
- Kemudian data *training* dibagi lagi oleh *bagging* menjadi *sub dataset (bootstrap)* sebanyak 10 perulangan *(iterations)*.
- Masing-masing *bootstrap* data kemudian diproses dengan metode klasifikasi misalnya metode C4.5.
- Lakukan proses C4.5 pada *sub dataset (bootstrap)* sebanyak 10 perulangan (*iterations*) sehingga diperoleh 10 buah pohon/model acak.
- Lakukan pendugaan gabungan (*aggregating*) berdasarkan *k* buah pohon tersebut. Untuk memilih kasus klasifikasi menggunakan konsep *majority vote* (suara terbanyak).

- Hasil pendugaan gabungan (*aggregating*) berdasarkan *majority vote* (suara terbanyak) digunakan untuk menguji ketepatan klasifikasi menggunakan *confusion matrix*.

b. Adaptive Boosting (Adaboost)

Algoritma *Adaboost* pertama kali diperkenalkan pada tahun 1995 oleh Freund dan Schapire, telah banyak memecahkan berbagai masalah praktis dari algoritma sebelumnya. *Boosting* merupakan salah satu contoh metode *ensemble* (*ensemble methods*) yang menggabungkan suatu urutan model pembelajaran *k* (atau disebut juga *classifier* dasar), M_1, M_2, ..., M_k, dengan tujuan menciptakan model klasifikasi gabungan yang lebih baik. *Ensemble methods* ini mengembalikan hasil prediksi kelas berdasarkan penilaian dari *classifier* dasarnya (Han, et al., 2012). Adapun algoritma *Adaboost* memiliki pondasi teori yang solid, prediksi yang sangat akurat, tingkat kesederhanaan yang tinggi (cukup hanya dengan 10 baris kode), dan penggunaannya yang luas dan sukses (Wu, et al., 2007).

Adapun penggambaran kerja dari algoritma *Adaboost*, yaitu sebagai berikut: misalkan $\mathcal{X}$ dinotasikan sebagai *instance* dan $\mathcal{Y}$ sebagai *set* label kelas. Diasumsikan $\mathcal{Y} = \{-1, +1\}$. Kemudian diberikan algoritma pembelajaran dasar atau lemah (*weak or base learning algorithm*) dan sebuah *training set* $\{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$ di mana $x_i \in \mathcal{X}$ dan $y_i \in \mathcal{Y}$. Kemudian algoritma *Adaboost* bekerja seperti berikut: pertama tiap contoh *training* (*training example*) (x_i, y_i) $(i \in \{1, \dots . m\})$ diberikan bobot yang sama. Denotasikan distribusi bobot pada putaran pembelajaran (*learning round*) ke-*t* sebagai D_t. Dari *training set* dan D_t algoritma *Adaboost* ini menghasilkan suatu *weak* atau *base learner* $h_t : \mathcal{X} \to \mathcal{Y}$ dengan memanggil algoritma pembelajaran dasarnya. Kemudian contoh *training* tersebut digunakan untuk menguji h_t, dan bobot-bobot dari contoh klasifikasi yang salah akan meningkat. Dengan demikian, suatu distribusi bobot yang telah diperbarui D_{t+1} diperoleh. Dari *training set* dan D_{t+1} *Adaboost* menghasilkan *weak learner* lain

dengan memanggil algoritma pembelajaran dasarnya lagi. Proses tersebut diulang sebanyak *T* putaran, dan model akhir diperoleh dengan bobot suara terbanyak (*weighted majority voting*) dari kumpulan *T weak learner*, di mana bobot dari *learner* tersebut ditentukan selama proses pelatihan atau *training*.

Pengembangan metode *Adaboost* memiliki banyak varian turunan antara lain: *Adaboost.M1, Adaboost.M1W, Killback-Leibler Boosting* (*KLBoosting*)*,* dan *Jensen-Shannon Boosting* (*JSBoosting*). *Adaboost.M1* merupakan generalisasi langsung dari *Adaboost* untuk dua kelompok dari masalah multikelas. Sedangkan *Adaboost.M1W* merupakan pengembangan dari *Adaboost.M1* dengan meminimalisasi batas atas pengukuran kinerja yang disebut dengan *guessing error*. Kemudian untuk algoritma *KLBoosting* dan *JSBoosting* digunakan untuk pendeteksian pola atau objek gambar. Implementasi *Adaboost* dalam WEKA menggunakan varian *Adaboost.M1*. Metode *ensemble* atau dikenal dengan metode *classifier combination*, merupakan teknik yang dapat digunakan untuk meningkatkan akurasi dari klasifikasi pada *data mining*. Pada *adaboost* ini *training set* yang digunakan untuk setiap *base classifier* dipilih berdasarkan performasi dari *classifier* sebelumnya. Di dalam *boosting*, sampel yang tidak diprediksikan dengan benar oleh *classifier* di dalam rangkaian akan dipilih lebih sering dibandingkan dengan sampel yang telah diprediksikan dengan benar.

Adapun tahapan-tahapan teknik *ensemble adaboost* dalam pengklasifikasian *dataset* menggunakan algoritma klasifikasi. Misalnya, menggunakan algoritma *Support Vector Machine* (SVM), yaitu sebagai berikut.

a) Mempersiapkan data *training* dan memisalkan $\mathcal{X}$ dinotasikan sebagai *instance* dan $\mathcal{Y}$ sebagai *set* label kelas. Diasumsikan $\mathcal{Y} = \{-1, +1\}$.
b) Memberikan inputan berupa suatu kumpulan *sample* penelitian dengan label pada sebuah *training set* $\{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$ di mana $x_i \in \mathcal{X}$ dan

$y_i \in \mathcal{Y} = \{1, \ldots, k\}$ dan suatu algoritma pembelajaran dasar atau lemah (*weak or base learning algorithm*) yaitu *support vector machine*, dengan jumlah iterasi atau perulangan *T*.

c) Kemudian algoritma *Adaboost* menginisialisasikan bobot pada contoh *training* (*training example*) $D_i^1 = 1/m$, (x_i, y_i) $(i \in \{1, \ldots . m\})$

d) Denotasikan distribusi bobot pada putaran pembelajaran (*learning round*) ke-*t* sebagai D_t dan t=1, ..., *T*. Di mana *T* adalah banyak putaran atau iterasi.

e) Dari *training set* dan D_t algoritma *Adaboost* ini menghasilkan suatu *weak* atau *base learner* $h_t : \mathcal{X} \to \mathcal{Y}$ dengan memanggil algoritma pembelajaran dasarnya.

f) Selanjutnya menghitung kesalahan pelatihannya dengan persamaan berikut:

$$\varepsilon_t = \sum_{i=1}^{m} D_i^t, y_i \neq sh_t(x_i)$$

Jika $\varepsilon_t \geq 0{,}5$, maka set *T*= $t - 1$, batalkan *loop* dan langsung menuju *output.*

g) Menghitung nilai bobot untuk *base classifier*.

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$

h) *Update* distribusi bobot *sample* pelatihan

$$D_i^{t+1} = \frac{D_i^t \exp\{-\alpha_t y_i h_t(x_i)\}}{Z_t}$$

Lakukan perulangan atau iterasi sebanyak t.

i) Dengan demikian, suatu distribusi bobot yang telah diperbarui D_{t+1} diperoleh. Dari *training set* dan D_{t+1} *Adaboost* menghasilkan *weak learner* lain dengan memanggil algoritma pembelajaran dasarnya lagi. Proses tersebut diulang sebanyak *T* putaran, dan model akhir diperoleh dengan bobot suara terbanyak (*weighted majority voting*)

dari kumpulan *T weak learner*, dimana bobot dari *learner* tersebut ditentukan selama proses pelatihan atau *training*. *Output* yang didapatkan dari teknik *ensemble adaboost* ini berupa:

$$f(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$