BAB II
TINJAUAN PUSTAKA

2.1. Prediksi Kelulusan Mahasiswa

Perkiraan keberhasilan studi mahasiswa berdasarkan atribut yang disediakan, sehingga menghasilkan *output* "lulus tepat waktu" atau "Lulus Terlambat" (Ahmad, 2014). Indeks prestasi sering digunakan sebagai indikator penilaian di Perguruan Tinggi dan banyak Perguruan Tinggi memberi standar minimum yang sulit diperoleh Mahasiswa. Dimana prestasi belajar adalah hasil perubahaan pada diri pembelajaran yang meliputi aspek kognitif, afektif dan psikomotor yang merupakan bukti suatu usaha yang dapat dicapai dalam belajar. Dengan memprediksi kelulusan berdasarkan atribut yang digunakan memberikan hal positif dalam meningkatkan atau mempertahankan kinerja mahasiswa selama masa studi di Perguruan Tinggi (Oyelade, 2010).

2.2. Prediksi

Prediksi adalah suatu proses untuk memperkirakan nilai hasil secara sistematis tentang susuatu yang paling mungkin terjadi dimasa yang akan datang berdasarkan informasi dimasa lalu dan yang sekarang dimiliki sehingga mengasilkan nilai hasil prediksi yang akurat (Nachrowi, D. N., dan Usman, H., 2004). Maka prediksi dapat dibagi menjadi dua bagian yaitu prediksi kualitatif dan prediksi kuantitatif (Sumayang, L., 2003):

2.2.1. Prediksi Kualitatif

Prediksi kualitatif didasarkan atas data kualitatif pada masa lalu. Metode kualitatif digunakan jika data masa lalu dari variabel yang akan diprediksi tidak ada, tidak cukup atau kurang dipercaya. Hasil prediksi yang dibuat sangat tergantung pada individu yang menyusunnya. Hal ini penting karena hasil prediksi tersebut ditentukan berdasarkan pemikiran yang bersifat *judgement* atau opini, pengetahuan dan pengalaman dari penyusunnya. Oleh karena itu metode kualitatif ini disebut juga *judgemental, sudjective, intuitive.*

2.2.2. Prediksi Kuantitatif

Prediksi kuantitatif didasarkan atas data kuantitatif pada masa lalu. Hasil prediksi yang dibuat sangat tergantung pada metode yang dipergunakan dalam prediksi tersebut. Dengan metode yang berbeda akan diperoleh hasil prediksi yang berbeda. Hal yang perlu diperhatikan dari penggunaan metode tersebut adalah baik tidaknya metode yang digunakan dan sangat ditentukan dari penyimpangan antara hasil prediksi dengan kenyataan yang terjadi. Metode yang baik adalah metode yang memberikan nilai-nilai perbedaan atau penyimpangan yang mungkin. Prediksi kuantitatif hanya dapat digunakan apabila terdapat tiga kondisi sebagai berikut:

a. Adanya informasi tentang keadaan yang lain.
b. Informasi tersebut dapat dikuantifikasikan dalam bentuk data.
c. Dapat diasumsikan bahwa pola yang lalu akan berkelanjutan pada masa yang akan datang.

Metode kuantitatif dikelompokkan pada dua jenis, yaitu:

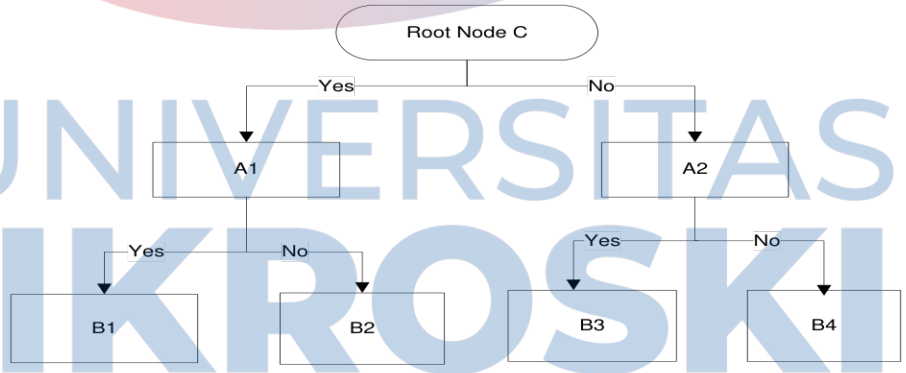
a. Metode serial waktu (Time Series) merupakan metode yang digunakan untuk menganalisis serangkaian data yang merupakan fungsi dari waktu. Mengasumsikan bahwa beberapa pola atau kombinasi pola selalu berulang sepanjang waktu dan pola dasar dapat diidentifikasikan semata mata atas dasar data historis dari serial tersebut.
b. Metode eksplanatori mengasumsikan bahwa nilai suatu varibael merupakan fungsi dari satu atau beberapa variabel lain. Metode ini disebut juga dengan metode kausal mengasumsikan adanya hubungan antar variabel bebas (independen) dengan variabel tak bebas (dependen) yang dipengaruhinya atau dalam bentuk yang lain antara input dan output dari suatu sistem.

2.3. Pohon Keputusan

Salah satu metode data mining yang umumnya digunakan adalah pohon keputusan. Metode pohon keputusan mengubah fakta yang sangat besar menjadi pohon keputusan yang mepresentasikan *rule*. Pohon keputusan merupakan pohon data yang terdiri atas simpul keputusan dan simpul daun yang terstruktur. Simpul

tersebut memiliki kelas-kelas dan simpul keputusan menguji beberapa atribut sampai diperoleh atribut yang terpilih sebagai pemilah. Setiap cabang pengujian tersebut menghasilkan simpul anak (*child node*) dibawah simpul induknya (*parent Node*). Pembentukan pohon keputusan menggunakan prinsip membagi sampel pelatihan menjadi beberapa sub-himpunan yang berbeda. Proses pada pohon keputusan adalah mengubah bentuk data (tabel) menjadi pohon keputusan, mengubah model pohon keputusan menjadi aturan (Jayanti, 2008).

Manfaat utama dari penggunaan pohon keputusan adalah kemampuannya untuk *mem-break down* proses pengambilan keputusan yang kompleks menjadi lebih sederhana sehingga pengambilan keputusan akan lebih menginterpretasikan solusi permasalahan. Pohon keputusan juga berguna untuk mengekplorasi data, yaitu menemukan hubungan tersembunyi antara sejumlah calon variabel *input* dengan sebuah variabel target. *Decision tree* bergantung pada aturan *if- then*, tetapi tidak membutuhkan parameter dan metrik. Struktur sederhana dalam penafsiran dapat memecahkan masalah atribut *multi-type* dan juga dapat mengelola nilai nilai yang hilang (data *noise*).



Gambar 2.1 Struktur *decision tree* (Pramudiono, 2008)

Bagian awal dari pohon keputusan ini adalah titik akar (*root*), sedangkan setiap cabang dari pohon keputusan merupakan pembagian berdasarkan hasil uji dan titik akhir (*leaf*) merupakan pembagian kelas yang dihasilkan. Terdapat 3 tipe simpul pada pohon keputusan, yaitu:

a. Simpul akar (*root*), merupakan *node* paling atas, dimana pada *node* ini tidak memiliki cabang yang masuk dan memiliki cabang yang lebih dari satu,

terkadang tidak memiliki cabang sama sekali. Simpul ini biasanya berupa atribut yang paling memiliki pengaruh terbesar pada suatu kelas tertentu.

b. Simpul *Internal (Node),* merupakan *node* percabangan, pada *node* ini hanya terdapat satu cabang yang masuk dan memiliki lebih dari satu cabang yang keluar.

c. Simpul daun (*leaf*), merupakan *node* akhir, pada *node* ini hanya memiliki satu cabang yang masuk dan tidak memiliki cabang sama sekali dan sekaligus menandai bahwa simpul tersebut merupakan label kelas.

Tahapan awal adalah pengujian simpul akar, jika pada pengujian simpul akar menghasilkan sesuatu, maka proses pengujian juga dilakukan pada setiap cabang berdasarkan hasil dari pengujian. Hal ini berlaku juga untuk simpul internal dimana suatu kondisi pengujian baru akan diterapkan pada simpul daun. Pada umumnya proses dari sistem pohon keputusan adalah mengadopsi strategi pencarian *top-down* untuk solusi ruang pencariannya. Pada proses mengklasifikasikan sampel yang tidak diketahui, nilai atribut akan diuji pada pohon keputusan dengan cara melacak jalur dari titik akar sampai titik akhir, kemudian akan diprediksi kelas yang ditempati sampel baru. Berikut ini langkah-langkah konstruksi pohon keputusan menggunakan Algoritma C4.5 berdasarkan (Ruggieri, 2002):

1. Misalkan T adalah himpunan kasus-kasus yang akan dibuat simpul dimana kasus-kasus tersebut memiliki kelas dan atribut-atribut. Frekuensi terboboti freq (Cj,T) diperoleh dari perhitungan T dan kelas yang dihasilkan adalah Cj, untuk setiap j {1,2,3,...,n}.
2. Jika semua kasus berada dalam kelas Cj yang sama maka simpul yang dihasilkan adalah simpul daun yang dilabeli dengan kelas Cj sebagai kelas terbanyak. Kesalahan klasifikasi pada simpul daun merupakan kasus-kasus dalam T yang berbeda kelas dengan kelas Cj.
3. Jika T berisi kasus yang memiliki dua atau lebih kelas maka dapat dihitung information gain dari setiap atribut tersebut. Untuk atribut diskret, information gain disesuaikan dengan pembagi dalam T dengan nilai atribut yang sudah diketahui sebelumnya. Untuk atribut kontinu, information gain disesuaikan

dengan pembagi T ke dalam dua irisan (biner) yang dilabeli kasus dengan nilai atribut kurang dari atau sama dengan nilai ambang batas ($A \leq v$) dan nilai atribut dengan nilai atribut lebih besar dari nilai ambang batas ($A > v$).

4. Atribut dengan nilai information Gain tertinggi terpilih sebagai pemilah dalam simpul tersebut.
5. Simpul keputusan memiliki cabang sebanyak s yaitu T1,….,Ts dimana s = 2 untuk atribut kontinu dan s = h untuk atribut diskret dengan nilai h yang sudah diketahui.
6. Untuk setiap i = {1,2,...,s}, jika Ti tidak memiliki cabang lagi maka simpul tersebut secara langsung menjadi simpul daun yang diberi label kelas terbanyak di bawah simpul induknya dan kesalahan klasifikasi bernilai 0.
7. Apabila Ti memiliki cabang lagi maka pemilahan diproses kembali menggunakan kasus-kasus dalam Ti. Catatan khusus untuk kasus-kasus dengan nilai yang hilang pada atribut terpilih tersebut dilakukan proses pemilihan pemilah pada setiap simpul anaknya dengan pembobotan banyak kasus yang diketahui dibagi dengan banyak kasus pada simpul tersebut.
8. Kesalahan klasifikasi simpul dihitung dari penjumlahan dari kesalahan-kesalahan simpul anak yang dibandingkan dengan simpul induknya.

2.3.1. Algoritma C4.5

Algoritma C4.5 merupakan bagian dari kelompok algoritma *decision tree* yang digunakan untuk membentuk pohon keputusan dari data. Algoritma C4.5 merupakan pengembangan dari algoritma ID3 (Iterative Dichotomiser 3) yang juga merupakan algoritma untuk membangun sebuah pohon keputusan. Algoritma C4.5 secara rekursif mengunjungi tiap simpul keputusan, memilih percabangan optimal, sampai tidak ada cabang lagi yang mungkin dihasilkan. Algoritma C4.5 memiliki keunggulan dibandingkan dengan ID3 yaitu mampu mengatasi nilai yang hilang (*missing value*), mengatasi data bertipe kontinu dan melakukan pemangkasan pohon (*pruning trees*). Dalam kontruksi pohon keputusan data terbagi menjadi sampel pelatihan (*training sample*) dan sampel pengujian (*testing sample*). Sampel pelatihan digunakan untuk mengkontruksikan pohon keputusan dan sampel

pengujian digunakan untuk menguji hasil kontruksi pohon (witten et al, 2011). Pohon keputusan dianggap sebagai salah satu pendekatan yang paling populer, dalam klasifikasi pohon keputusan terdiri dari sebuah *node* yang membentuk akar (David dan Seng, 2010). Pada dasarnya konsep dari algoritma C4.5 adalah mengubah data menjadi pohon keputusan dan aturan-aturan keputusan (*rule*). Secara umum, untuk membangun sebuah pohon keputusan dengan Algoritma C4.5 adalah sebagai berikut:

1. Input dataset mahasiswa
2. Hitung jumlah data, sebelum menghitung jumlah data maka lakukan *Pre-Processing* data. Selanjutnya dilakukan penjumlahan data sesuai atribut yang digunakan.
3. Lakukan Perhitungan *entropy* dan gain untuk mencari *node*
   Untuk *gain* tertinggi akan dijadikan *node* selanjutnya.
4. Buat cabang untuk tiap-tiap anggota dari *node*.
5. Jika ada anggota *node* yang memiliki nilai *entropy* lebih besar dari nol, ulangi lagi proses dari awal dengan *node* sebagai syarat sampai semua anggota dari *node* bernilai nol.
6. Periksa nilai entropy dari anggota *node* ada yang bernilai nol. Jika ada, tentukan daun yang terbentuk. Jika seluruh nilai *entropy* anggota *node* adalah nol, maka proses pun berhenti

Node adalah atribut yang mempunyai nilai *gain* tertinggi dari atribut-aribut yang ada. Untuk menghitung nilai *gain* suatu atribut. Rumus *gain* seperti dibawah ini (David, 2014):

$$\text{Gain (S, A)} = \textit{Entropy}\ (S) - \sum_{i=1}^{n} \frac{|Si|}{|S|} * \text{Entropy (S)} \qquad (1)$$

Keterangan :

A : Atribut
S : Himpunan kasus
n : Jumlah partisi atribut A
|Si| : Jumlah kasus pada partisi ke i
|S| : Jumlah kasus dalam S

Rumus mencari *entropy* yang digunakan dalam menentukan seberapa informatif sebuah masukan atribut untuk mengkasilkan sebuah atribut (David dan Seng, 2014):

$$\text{Entropy (S)} = \sum_{i=1}^{n} - pi * log_2 pi \quad (2)$$

Keterangan:

S : Himpunan Kasus

n : Jumlah partisi S

pi : Proposisi dari Si terhadap S

2.3.2. Kelebihan Pohon Keputusan

Adapun yang menjadi kelebihan dari pohon keputusan adalah sebagai berikut:

1. Daerah pengambilan keputusan yang sebelumnya kompleks dan sangat global, dapat diubah menjadi lebih simpel dan spesifik.
2. Eliminasi perhitungan-perhitungan yang tidak diperlukan, karena ketika menggunakan metode pohon keputusan maka sample diuji hanya berdasarkan kriteria atau kelas tertentu.
3. Fleksibel untuk memilih fitur dari internal *node* yang berbeda, fitur yang terpilih akan membedakan suatu kriteria dibandingkan kriteria yang lain dalam *node* yang sama. Kefleksibelan metode pohon keputusan ini meningkatkan kualitas keputusan yang dihasilkan jika dibandingkan ketika menggunakan metode penghitungan satu tahap yang lebih konvensional.
4. Dalam analisis multivarian, dengan kriteria dan kelas yang jumlahnya sangat banyak, seorang penguji biasanya perlu untuk mengestimasikan baik itu distribusi dimensi tinggi ataupun parameter tertentu dari distribusi kelas tersebut. Metode pohon keputusan dapat menghindari munculnya permasalahan ini dengan menggunakan kriteria yang jumlahnya lebih sedikit pada setiap *node* internal tanpa banyak mengurangi kualitas keputusan yang dihasilkan.

2.4. *Supervised Learning*

Supervised Learning adalah teknik pembelajaran mesin dengan membuat suatu fungsi dari data latihan. Data latihan terdiri dari pasangan nilai input (biasa dalam bentuk vektor), dan *output* yang diharapkan untuk *input* yang bersangkutan. Tugas dari mesin *supervised learning* adalah memprediksi nilai fungsi untuk semua nilai input yang mungkin, setelah mengalami sejumlah data latihan.

Untuk mencapai tujuan ini, mesin harus dapat melakukan proses generalisasi dari data latihan yang diberikan, untuk memprediksikan nilai *output* dari input yang telah pernah diberikan sebelumnya dengan cara yang masuk akal.

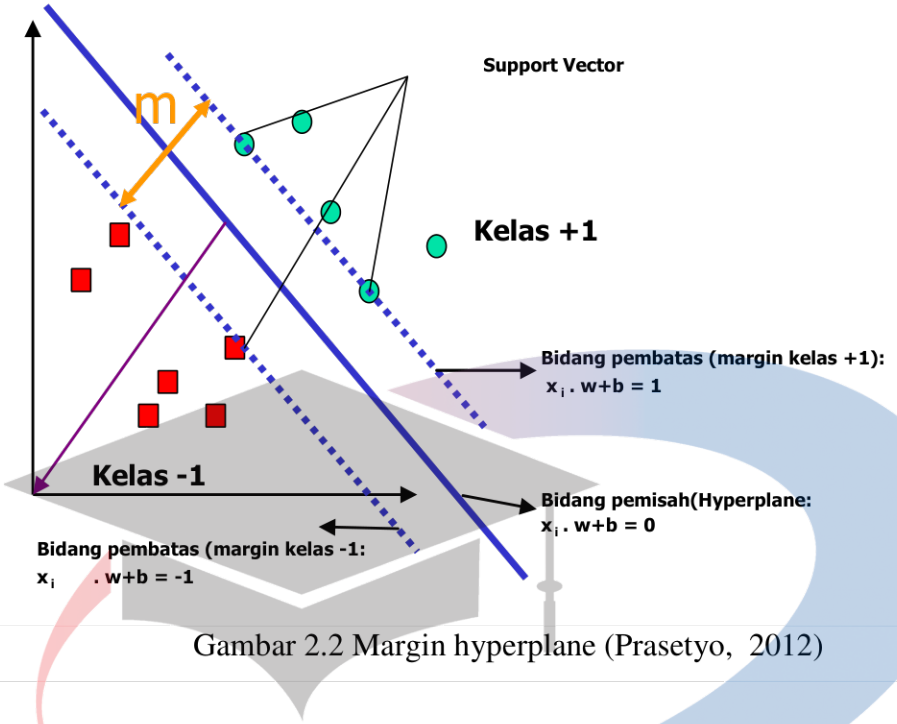
2.5. *Support Vector Machine (SVM)*

Support Vector Machine (SVM) adalah suatu teknik untuk melakukan prediksi, baik dalam kasus klasifikasi maupun regresi (Santosa, 2007). *Support Vector Machine* (SVM) memiliki prinsip dasar linier *classifier* yaitu kasus klasifikasi yang secara linier dapat dipisahkan, namun *Support Vector Machine* (SVM) telah dikembangkan agar dapat bekerja pada problem non-linier dengan memasukkan konsep kernel pada ruang kerja berdimensi tinggi. Pada ruang berdimensi tinggi, akan *hyperplane* yang dapat memaksimalkan jarak (margin) antara kelas data. Metode *Support Vector Machine* (SVM) berakar dari teori pembelajaran statistik yang dapat memberikan hasil yang lebih baik dari pada metode lain.

Banyak teknik data mining atau *machine learning* yang dikembangkan dengan asumsi keliniearan, sehingga algoritma yang dihasilkan terbatas untuk kasus-kasus yang linier (Santosa, 2007*). Support Vector Machine* (SVM) dapat bekerja pada data non-linier dengan menggunakan pendekatan kernel pada fitur awal himpunan data. Fungsi kernel yang digunakan untuk memetakan dimensi awal (dimensi yang lebih rendah) himpunan data ke dimensi baru (dimensi yang relatif lebih tinggi). Pada awalnya *Support Vector Machine* (SVM) dikembangkan untuk persoalan klasifikasi dua kelas, kemudian dikembangkan kembali untuk klasifikasi multikelas (Hsu dan Lin, 2002). Dalam klasifikasi kasus multikelas, *hyperplane* yang terbentuk adalah labih dari satu. Salah satu pendekatan yang

digunakan adalah satu lawan semua (SLA). Metode SLA untuk kasus klasifikasi *k*-kelas, menentukan *k hyperplane* dimana *k* adalah banyak kelas dan ρ adalah *hyperplane.* Dalam metode ini $\rho^{(\iota)}$ diujikan dengan data dari kelas ι dengan label +1, dan semua data dari kelas lain dengan label -1.

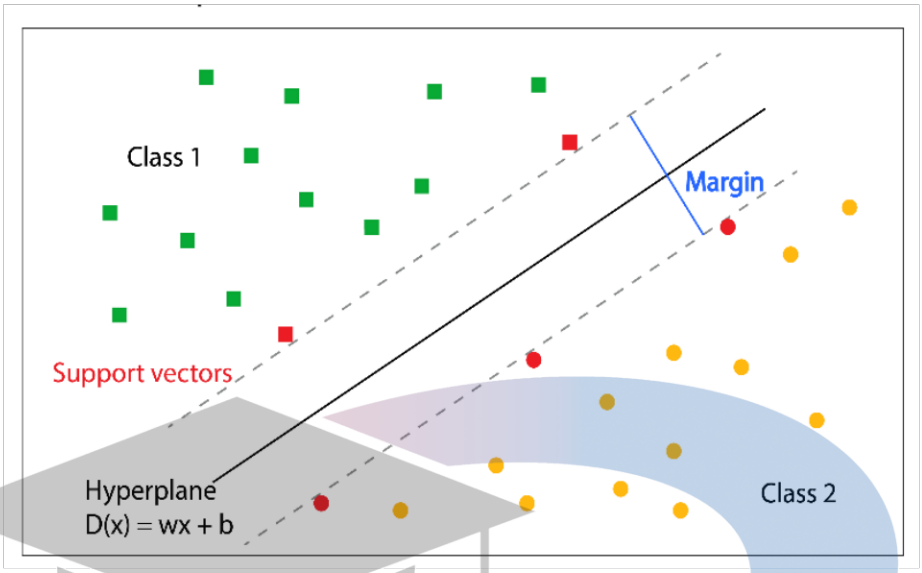
Konsep *Support Vector Machine* (SVM) adalah pencarian *hyperplane* terbaik yang berfungsi sebagai pemisah data dari dua kelas pada *input space. Hyperplane* pemisah terbaik adalah *hyperplane* yang terletak di tengah di antara dua set obyek dari dua kelas. *Hyperplane* terbaik dapat dicari dengan memaksimalkan *margin* atau jarak dari dua set objek dari dua kelas yang berbeda. Pada*). Support Vector Machine* (SVM) pemisahan yang baik dicapai oleh *hyperplane* yang memiliki jarak terbesar ke titik data *training* terdekat dari setiap kelas (margin fungsional), karena pada umumnya semakin besar margin semakin rendah error generalisasi dari pemilah. *Support Vector Machine* (SVM) pada prinsipnya adalah klasifikasi linear, tetapi *Support Vector Machine* (SVM) memiliki keunggulan dalam klasifikasi untuk problem non-linier. Dapat diasumsikan bahwa kedua belah kelas dapat terpisah secara sempurna oleh *hyperplane (linear separable).* Akan tetapi, pada umumnya dua belah kelas pada *input space* tidak dapat terpisah secara sempurna (non linear separable). Untuk mengatasi masalah ini, *Support Vector Machine* (SVM) dirumuskan ulang dengan memperkenalkan metode *Margin Soft*.



Gambar 2.2 Margin hyperplane (Prasetyo, 2012)

2.5.1. Klasifikasi Data Linear Separable

Data Linear *Separable* merupakan data yang dapat dipisahkan secara linear. Misalkan $\{x_1 \dots \dots x_n\}$ adalah dataset {+1, -1} adalah label dari kelas dari data kelas x_n. Pada Gambar.3 dapat dilihat berbagai alternatif bidang pemisah yang dapat memisahkan semua data set sesuai dengan kelasnya. Namun, bidang pemisah terbaik tidak hanya dapat memisahkan data tetapi juga memiliki margin paling besar.



Gambar 2.3 SVM *linear separable* (Prasetyo 2012)

Data latih dinyatakan $(y_i\, x_i)$, dimana i= 1, 2, ..., N, dan $x_i=\{xi1\, x_{i2}\, \ldots, x_{iq})^T$ merupakan atribut (fitur) set untuk data latih ke-*i,* yi € {-1, +1} menyatakan label kelas. *Hyperplane* klasifikasi linier *Support Vector Machine* (SVM) dinotasikan (Prasetyo, 2012):

$$w.\, x_i + b = 0 \quad (3)$$

Keterangan:

w dan b adalah parameter model.

w. x_i merupakan inner-product dalam antara w dan x_i.

jika x_i masuk kedalam -1 maka memenuhi pertidaksamaan sebagai berikut:

$$w.\, x_i + b \leq -1 \quad (4)$$

jika x_i masuk kedalam +1 maka memenuhi pertidaksamaan sebagai berikut:

$$w.\, x_i + b \geq +1 \quad (5)$$

jika data dalam kelas -1 (x_a) terdapat di *hyperplane* maka persamaan akan terpenuhi untuk dinotasikan dengan seperti berikut:

$$w.x_a + b = 0 \quad (6)$$

data kelas +1 (x_b) akan memenuhi persamaan sebagai berikut:

$$w.x_b + b = 0 \quad (7)$$

Dengan mengurangi persamaan sebagai berikut:

$$w.(x_b - x_a) = 0 \quad (8)$$

$x_b - x_a$ merupakan vektor paralel di posisi *hyperplane* dan diarahkan dari x_a ke x_b, maka arah w tegak lurus terhadap *hyperplane* saat *inner product* bernilai nol.

Formula untuk memberikan label -1 untuk kelas pertama, dan +1 untuk kelas kedua seperti berikut:

$$Y = \begin{cases} +1^{\text{jika } w.z+b>0} \\ -1_{\text{jika } w.z+b<0} \end{cases} \quad (9)$$

Hyperplane untuk kelas -1 (garis putus putus) adalah data pada *support vector* yang memenuhi persamaan:

$$w.x_b + b = -1 \quad (10)$$

hyperplane untuk kelas +1 (garis putus putus) memenuhi persamaan:

$$w.x_b + b = +1 \quad (11)$$

Margin dapat dihitung seperti berikut ini:

$$w.(x_b - x_a) = 2 \quad (12)$$

Untuk mencari margin (jarak) *hyperplane* terbaik yang terletak di tengah tengah dua bidang pembatas kelas sama dengan memaksimalkan margin atau jarak antara dua set objek kelas yang berbeda. Maka margin dapat dihitung dengan (Prasetyo, 2010):

$$\|w\| \times d = 2 \text{ atau } d = \frac{2}{\|w\|} \quad (13)$$

2.5.2. Klasifikasi Data Linear *Non-Separable*

Untuk menyelesaikan problem non-linear *Support Vector Machine* (SVM) dimodifikasi dengan memasukkan fungsi kernel. Dalam non-linear *Support Vector Machine* (SVM), pertama-tama data $\vec{x}$ dipetakan oleh fungsi $\Phi(\vec{x})$ ke ruang vektor yang berdimensi lebih tinggi. *Hyperplane* yang memisahkan kedua kelas tersebut dapat dikontruksikan. Selanjutnya bahwa fungsi Φ memetakan tiap data pada input *space* tersebut ke ruang vektor baru yang berdimensi lebih tinggi (dimensi 3), sehingga kedua kelas dapat dipisahkan secara linear oleh sebuah *hyperplane*. Notasi matematika dari *mapping* ini adalah sebagai berikut:

$$\Phi : \acute{R}^d \rightarrow \acute{R}^d \quad d < q \tag{14}$$

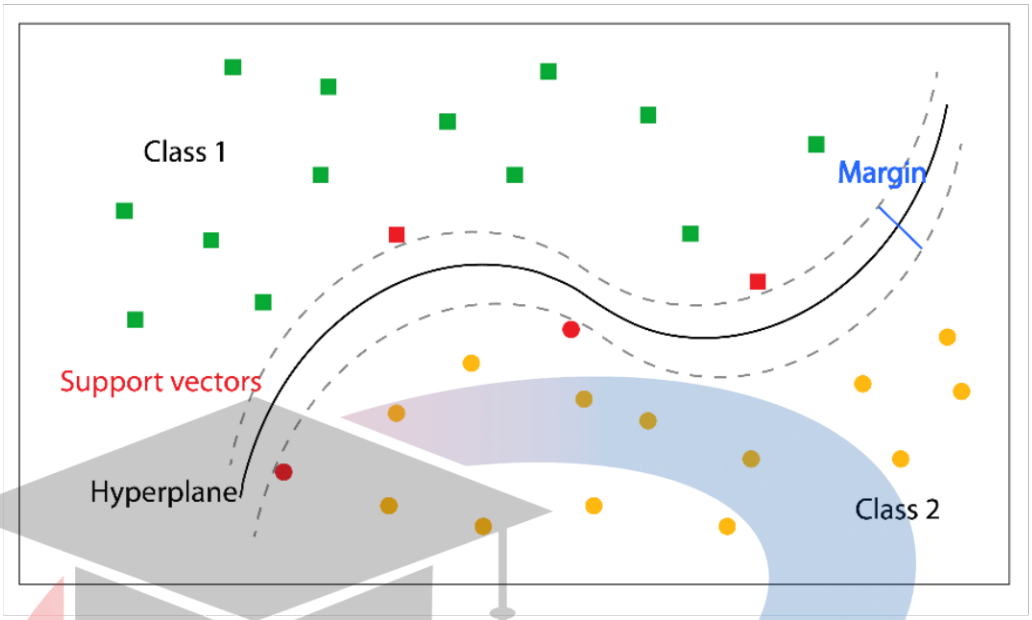
Selanjutnya proses pembelajaran pada *Support Vector Machine* (SVM) dalam menemukan titik-titik *support vector*, hanya bergantung pada *dot product* (perkalian titik) dari data yang sudah ditransformasikan pada ruang baru yang berdimensi lebih tinggi, yaitu $\Phi(\overrightarrow{x_i}).\Phi(\overrightarrow{x_j})$. Karena umumnya transformasi Φ ini tidak diketahui, dan sangat sulit untuk difahami secara mudah, maka perhitungan *dot product* (perkalian titik) dapat digantikan dengan fungsi kernel $K(\overrightarrow{x_i}\ \overrightarrow{x_j})$ yang mendefinisikan secara implisit transformasi Φ. Hal ini disebut sebagai Kernel Trick, yang dirumuskan:

$$K(\overrightarrow{x_i}\ \overrightarrow{x_j}) = \Phi(\overrightarrow{x_i}).\Phi(\overrightarrow{x_j}) \tag{15}$$

Dan prediksi pada set data dengan dimensi fitur yang baru diformulasikan dengan

$$f(\Phi(\vec{x})) = \text{sign}(\vec{w} \cdot \Phi(\vec{x}) + b) = \text{sign}\left(\sum_{i=1,}^{n} xi^{\epsilon sv}\ \alpha i\ yi\right) \tag{16}$$

Support Vector pada persamaan di atas dimaksudkan dengan *subset* dari training set yang terpilih sebagai *support vector*, dengan kata lain data $\overrightarrow{xi}$ yang berkorespondensi pada $\alpha i \geq 0$.



Gambar 2.4 SVM Non-Linear Separable (Prasetyo, 2012)

2.5.3. Metode Kernel

Menurut (Karatzouglou *et al*, 2004) ada beberapa fungsi kernel yang sering digunakan dalam literatur *Support Vector Machine* (SVM) antara lain sebagai berikut:

1. Kernel Linear adalah kernel yang palin sederhana dari semua fungsi kernel. Kernel ini biasa digunakan dalam kasus klasifikasi teks.

$$x^T x$$

2. Kernel Radial Basis Gausian adalah kernel yang umum digunakan untuk data yang sudah *valid* (*avaliable*) dan merupakan *default* dalam *tools Support Vector Machine* (SVM).

$$\exp(-\frac{1}{2}\sigma^2||x - x_i||^2)$$

3. Kernel Polynomial adalah kernel yang sering digunakan untuk klasifikasi gambar.

$$(x^T x_i + 1)^p$$

4. Kernel Tangent *Hyperbolic* (sigmoid) adalah kernel yang digunakan pada *neural network*.

$$\tan h(\beta x^T x_i \beta_1, \text{ dimana } \beta, \beta_1 \epsilon \text{ Ŕ}$$

Penggunaan fungsi kernel akan menentukan *feature space* di mana fungsi klasifier akan dicari. Sepanjang fungsi kernelnya *legistimate*, *Support Vector Machine* (SVM) akan beroperasi secara benar meskipun tidak tau seperti apa map yang digunakan, sehingga lebih mudah menemukan fungsi kernel dari pada mencari map seperti apa yang tepat untuk melakukan mapping dari *input space* ke *feature space*. Pada penerapan metoda kernel, tidak perlu tau *map* apa yang digunakan untuk satu per satu data, tetapi lebih penting mengetahui bahwa *dot product* (perkalian titik) dua titik di *feaure space* bisa digantikan oleh fungsi kernel.

2.5.4. Karakteristik *Support Vector Machine*

Beberapa karakteristik klasifikasi *Support Vector Machine* (SVM) yang dapat diringkas menurut Prasetyo, 2012:

1. *Support Vector Machine* (SVM) sebenarnya bisa dikatakan sebagai teknik kalsifikasi yang *semi-eager learner* karena selain memerlukan proses pelatihan, *Support Vector Machine* (SVM) juga menyimpan sebagian kecil data latih untuk digunakan kembali pada saat proses prediksi. Sebagian data yang masih disimpan ini adalah *Suppor Vector.*
2. Untuk parameter yang sama yang digunakan dalam klasifikasi, *Support Vector Machine* (SVM) memberikan model klasifikasi yang solusinya adalah *global optimal.* Hal ini berarti *Support Vector Machine* (SVM) selalu memberikan model yang sama dan solusi dengan margin maksimal.
3. Proses pelatihan yang dilakukan oleh *Support Vector Machine* (SVM) memberikan kinerja yang lebih baik.
4. *Support Vector Machine* (SVM) membutuhkan komputasi pelatihan dan prediksi yang rumit karena dimensi data yag digunakan dalam proses pelatihan dan prediksi lebih besar dari pada dimensi yang sesungguhnya. Hal ini bertentangan dengan metode lain yang umumnya mengurangi dimensi untuk memberikan kinerja yang lebih cepat dan akurasi yang lebih baik.
5. Untuk dataset berjumlah besar, *Support Vector Machine* (SVM) membutuhkan memori yang sangat besar untuk alokasi matrik kernel yang

digunakan. Misalnya, data latih dengan ukuran 1.000 data dan 10 kolom fitur akan berubah menjadi matrik kernel berukuran 1.000 x 1.000. Metode pelatihan *Support Vector Machine* (SVM) yang membutuhkan memori besar adalah *chunking* (Vapnik, 1982) dan dekomposisi (Osuna, 1997), sedangkan *sequential minimal optimatzation* (SMO) (Platt, 1999) dikembangkan untuk mempercepat proses pelatihan dan mengurangi penggunaan memori.

6. Penggunaan matriks kernel mempunyai keuntungan lain, yaiut kinerja dataset dengan dimensi besar tetapi jumlah datanya sedikit akan lebih cepat karena ukuran data pada dimensi baru berkurang banyak. Misalnya data latih berukuran 10 data 1.000 kolom fitur akan berukuran menjadi matriks kernek berukuran 10x10 saja.

2.5.5. Kelebihan *Support Vector Machine* (SVM)

Menurut Nugroho (2003) mengatakan bahwa *Support Vector Machine* (SVM) memiliki beberapa keuntungan seperti:

1. *Generalisasi*

   Generalisasi didefinisikan sebagai kemampuan suatu metode untuk mengklasifikasikan suatu *pattern*, yang tidak termasuk data yang dipakai dalam *fase* pembelajaran metode itu.

2. *Curse of dimensionality*

   Curse of dimensionality didefinisikan sebagai masalah yang dihadapi suatu metode *pettern recognition* dalam mengestimasikan parameter dikarenakan jumlah sampel data yang *relative* lebih sedikit dibandingkan dengan dimensional ruang *vector* data tersebut. Semakin tinggi dimensi dari ruang vektor informasi yang diolah, membawa konsekuensi dibutuhkan jumlah data dalam proses pembelajara (Cortes & Vapnik, 1995) membuktikan bahwa generalisasi yang diperolah oleh *Support Vector Machine* (SVM) tidak dipengaruhi oleh dimensi dari input *vektor.* Hal ini merupakan alasan mengapa *Support Vector Machine* (SVM) merupakan salah satu metode yang tepat dipakai untuk memecahkan masalah berdimensi tinggi, dalam keterbatasan sampel data yang ada.

3. *Feasibility*

   SVM dapat diimplementasikan *relative* mudah, karena proses penentuan *support vector* dapat dirumuskan dalam QP *problem*. Dengan demikian jika memiliki *library* untuk menyelesaikan QP *problem*, dengan sendirinya *Support Vector Machine* (SVM) dapat diimplementasikan dengan mudah.

2.6. Konsep Klasifikasi

Klasifikasi merupakan kegiatan menilai suatu objek data untuk dimasukkan ke dalam kelas tertentu dari sejumlah kelas yang tersedia. Dalam klasifikai ada dua pekerjaan utama yang dilakukan (Prasetyo, 2012), yaitu:

1. Pembangunan model sebagai prototipe untuk disimpan sebagai memori.
2. Penggunaan model tersebut untuk melakukan pengenalan/klasifikasi/ prediksi pada suatu objek lain agar diketahui peletakan kelas objek data dalam model yang sudah disimpan.

Contoh aplikasi yang sering ditemui adalah pengklasifikasian jenis hewan yang mempunyai banyak atribut. Dengan atribut tersebut, jika ada hewan baru, kelas hewannya bisa langsung diketahui. Contoh lain adalah bagaimana melakukan diagnosis penyakit kulit kanker melanoma (Amaliyah et al, 2011), yaitu dengan melakukan pembangunan model berdasarkan latih yang ada, kemudian menggunakan model tersebut untuk mengidentifikasi penyakit pasien baru sehingga diketahui apakah pasien tersebut mederita kanker atau tidak.

2.6.1. Model

Model dalam klasifikasi mempunyai arti yang sama dengan kotak hitam, dimana ada suatu model yang menerima masukan, kemudian mampu melakukan pemikiran terhadap masukan tersebut, dan memberikan jawaban sebagai keluaran hasil pemikirannya. Kerangka kerja (*framework*) klasifikasi ditunjukkan pada Gambar 4.4. Pada gambar tersebut disediakan sejumlah data latih (x,y) untuk digunakan sebagai data pembangunan model. Model tersebut kemudian dipakai

untuk memprediksi kelas dari data uji (x.?) sehinggan diketahui kelas y yang sesungguhnya.

Kerangka kerja dalam model meliputi dua langkah proses, yaitu induksi dan deduksi. Induksi merupakan langkah untuk membangun model klasifikasi dari data latih yang diberikan, disebut juga proses pelatihan, sedangkan dedukasi merupakan langkah untuk menerapkan model tersebut pada data uji sehingga kelas yang sesungguhnya dari data uji dapat diketahui dan disebut juga proses prediksi (Prasetyo, 2012).

2.6.2. Pengukuran Kinerja Klasifikasi

Sebuah sistem yang melakukan klasifikasi diharapkan dapat melakukan klasifikasi semua set data dengan benar, tetapi kinerja suatu sistem tidak bisa 100% benar, sehingga sistem klasifikasi juga harus diukur kinerjanya. Pengukuran dapat dilakukan dengan matriks konfusi (*confusion matrix*). Matriks konfusi merupakan tabel pencatat hasil kerja klasifikasi. Tabel 2.1 merupakan contoh matriks konfusi yang melakukan klasifikasi biner (dua kelas), hanya ada dua kelas yaitu kelas 0 dan 1. Setiap sel *Fij* dalam matrik menyatakan jumlah rekord/data dari kelas *i* yang hasil prediksinya masuk ke kelas *j* . Misalnya, sel *F11* adalah jumlah data dalam kelas 1 yang secara benar dipetakan ke kelas 1, dan *F10* adalah data dalam kelas 1 yang dipetakan secara salah ke kelas 0.

Tabel 2.1 Matriks *confusi* 1

Fij		Kelas Hasil Prediksi (*j*)	
		Kelas=+1	Kelas= -1
Kelas ali (*i*)	Kelas =1	F11	F10
	Kelas=0	F01	F00

Berdasarkan isi matriks konfusi, kita dapat mengetahui jumlah data dari masing-masing kelas yang diprediksi secara benar, yaitu (F11 + F00), dan data yang diklasifikasikan secara salah, yaitu (F10+F01). Kuantitas matriks konfungsi dapat diringkas menjadi dua nilai, yaitu akurasi dan laju error. Dengan mengatahui

jumlah data yang diklasifikasikan secara benar, kita dapat megetahui akurasi dan hasil prediksi, dan dengan mengetahui jumlah data yang diklasifikasikan secara salah, dapat diketahui laju *error* dan prediksi yang dilakukan. Dua kuantitas ini digunakan sebagai metrik kinerja klasifikasi.

Untuk menghitung akurasi digunakan formula (Prasetyo, 2012):

$$\text{Akurasi} = \frac{Jumlah\ data\ yang\ diprediksi\ benar}{jumlah\ prediksi\ yang\ dilakukan} = \frac{f_{11}+f_{00}}{f_{11}+f_{10}+f_{01}+f_{00}} \quad (17)$$

Untuk menghitung laju *error* (kesalahan prediksi) digunakan formula (Prasetyo, 2012):

$$\text{Laju Eror} = \frac{Jumlah\ data\ yang\ diprediksi\ secara\ salah}{Jumlah\ prediksi\ yang\ dilakukan} = \frac{f_{10}+f_{00}}{f_{11}+f_{10}+f_{01}+f_{00}} \quad (18)$$

Semua algoritma klasifikasi berusaha membentuk model yang mempunyai akurasi tinggi (laju *error* yang rendah). Model yang dibangun dapat memprediksi dengan benar pada semua data yang menjadi data latihnya, tetapi ketika model berhadapan dengan data uji, maka kinerja dari sebuah algoritma klasifikasi ditentukan.

2.6.3. K-*Fold Cross Validation*

Cross-validasi adalah metode *statistic* yang mengevaluasi dan membandingkan algoritma pembelajaran dengan membagi data menjadi dua yaitu data *training* dan data *testing*. Bentuk dari *cross validation* adalah k-fold *cross validation* (Payam R., Lie Tang dan Huan Liu. 2008). Metode K-*fold cross validation* yang sering dipakai adalah 3K-*fold cross validation* dan 5K-*fold cross validation*. Dalam *cross-validation*, tentukan nilai *fold* atau partisi untuk data. Prinsip dari *k-fold cross validation* adalah membagi tiap kelompok data menjadi *k* bagian kelompok data yang selanjutnya data tersebut secara bergantian akan digunakan untuk *training* dan *testing* sejumlah *k* pengujian.

Tabel 2.2 Pengujian k-*fold validation.*

Data *Testing*	**Data *Training***
Data ke 1	Data ke 2 dan data ke 3
Data ke 2	Data ke 1 dan data ke 3
Data ke 3	Data ke 1 dan data ke 2

Misalkan untuk 3 *K-fold cross validation* data dibagi menjadi 3 bagian. Setiap bagian akan digunakan untuk *training* dan *testing* secara bergantian. Dua dari tiga data digunakan untuk *training* dan satu dari tiga data untuk *testing* yang dilakukan secara berulang sebanyak tiga kali sampai semua bagian digunakan untuk *testing* (Ian H. Witten, Frank Eibe, Mark A. Hall. , 2011). Jika bagian data pertama dan kedua digunakan untuk training maka bagian data ketiga digunakan untuk *testing*. Jika data bagian pertama dan ketiga digunakan untuk *training*, data kedua digunakan untuk *testing*. Jika data bagian kedua dan ketiga untuk *training* maka data bagian pertama untuk *testing*.