BAB 2
LANDASAN TEORI

2.1. Metode Penelitian

Metode penelitian atau metode ilmiah adalah prosedur atau langkah-langkah daam mendapatkan pengetahuan ilmiah atau ilmu. Jadi metode penelitian adalah cara sistematis untuk menyusun ilmu pengetahuan. Sedangkan teknik penelitian adalah cara untuk melaksanakan metode penelitian. Metode penelitian bisanya mengacu pada bentuk-bentuk penelitian [12].

1. **Metode Eksperimen (Mengujicobakan)**, adalah penelitian untuk menguji apakah variabel-variabel eksperimen efektif atau tidak. Untuk menguji efektif tidaknya harus digunakan variabel kontrol. Penelitian eksprimen biasanya dilakukan untuk bidang yang bersifat eksak. Sedangkan untuk bidang sosial biasanya digunakan metode survey eksplanatory, metode deskriptif, dan historis [12].
2. **Metode Verifikasi (Pengujian)**, adalah menguji seberapa jauh tujuan yang sudah digariskan itu tercapai atau sesuai atau cocok dengan harapan atau teori yang sudah baku. Tujuan dari penelitian verifikasi adalah untuk menguji teori-teori yang sudah ada guna menyusun teori baru dan menciptakan pengetahuan-pengetahuan baru [12].
3. **Metode Deskriptif (mendeskripsikan),** yaitu metode yang digunakan untuk mencari unsur-unsur, ciri-ciri, sifat-sifat suatu fenomena. Metode ini mulai mengumpulkan data, menganalisis data dan menginterprestasikannya. Metode deskriptif dalam pelaksanaannya dilakukan melalui : teknik survey, studi kasus (bedakan dengan suatu kasus), studi komparatif, studi tentang waktu dan gerak, analisis tingkah laku, dan analisis dokumenter [12].
4. **Metode Historis (Merekonstruksi),** yaitu suatu metode penelitian yang meneliti sesuatu yang terjadi di masa lampau. Dalam penerapannya, metode ini dapat dilakukan dengan suatu bentuk studi yang bersifat komparatif-historis, yudiris, dan bibliografik. Penelitian historis bertujuan untuk menemukan generalisasi dan membuat rekonstruksi masa lampau, dengan

cara mengumpulkan, mengevaluasi, memverifikasi serta mensintesiskan bukti-bukti untuk menegakkan fakta-fakta dan bukti guna memperoleh kesimpulan yang kuat [12].

Pada penelitian ini, metode penelitian yang digunakan adalah metode deskriptif.

2.2. Sistem Tanya Jawab

Sistem Tanya Jawab *(Question Answering System)* merupakan sebuah sistem pada komputer yang dapat menjawab pertanyaan menggunakan bahasa alami yang biasa dipakai manusia secara otomatis. Jawaban yang diperoleh dari sistem berisi informasi yang berasal dari sebuah sumber basis data [1]. Sebuah sistem tanya jawab, diharapkan dapat membantu orang untuk mendapatkan informasi lebih mengenai hal tertentu tanpa harus bertanya kepada orang lain [2].

Tujuan utama dari sistem jawab otomatis adalah untuk mencari jawaban yang tepat menggunakan bahasa alami yang ditentukan oleh manusia. Saat ini sistem tanya jawab telah dikembangkan dalam berbagai bidang seperti pada asisten pribadi, pelayanan konsumen yang diterapkan pada pelayanan jasa guna untuk mendapatkan informasi dan memenuhi kebutuhan konsumen.

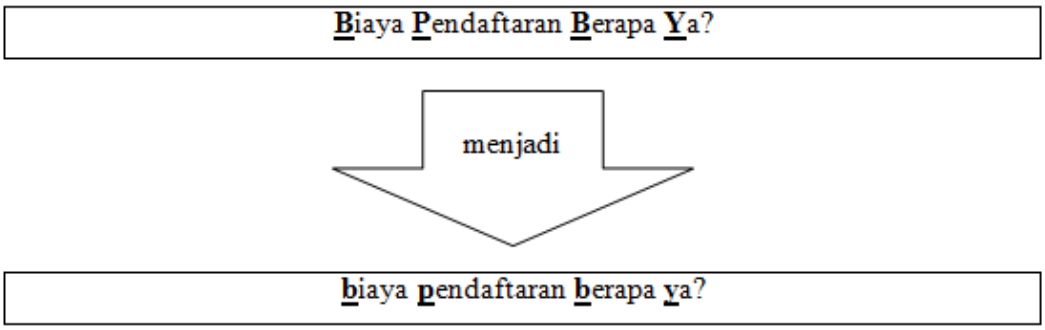
2.3. Preprocessing

Preprocessing adalah proses pengolahan data *raw* sebelum diolah pada tahap selanjutnya. Pada penelitian ini data yang digunakan adalah data pertanyaan berupa *text* dari hasil wawancara dari kuesioner yang tidak terstruktur. Oleh karena itu, proses ini sangat dibutuhkan karena akan digunakan untuk mengubah data yang tidak terstruktur menjadi terstruktur (normalisasi) agar dapat diproses pada tahap selanjutnya. Pada penelitian ini tahapan dalam *preprocessing* terdiri dari *casefolding*, *cleansing* (penghilangan simbol), *tokenizing* dan *remove stopword*.

2.3.1. *Case Folding*

Case Folding adalah proses menyeragamkan semua teks ke dalam case sama menjadi huruf kecil (*lowercase*) atau huruf besar (*uppercase)* [12]. Pada

proses ini, tahap *casefolding* menyeragamkan huruf dalam teks menjadi huruf kecil (*lowercase*). contoh dari tahap case folding ditampilkan pada gambar 2.1. berikut ini:



Gambar 2.1 Contoh *CaseFolding*

Sebagai contoh pada gambar diatas, pada huruf yang digaris bawahi dan ditebalkan terdeteksi memiliki huruf besar diawal kalimat sehingga diseragamkan menjadi huruf kecil menjadi "biaya pendaftaran berapa ya?".

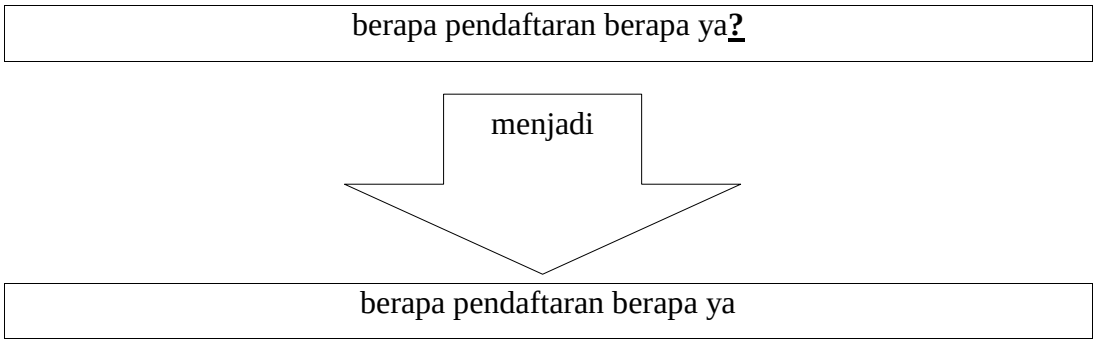
2.3.2. *Cleansing*

Cleansing merupakan proses membersihkan kata-kata yang tidak diperlukan untuk mengurangi noise. Kata yang dihilangkan adalah url, hashtag (#), username (@username), dan email. Selain itu juga tanda baca seperti titik(.), koma(,), dan tanda baca lainnya akan dihilangkan. [12]. Pada penelitian ini, proses *cleansing* untuk menghilangkan simbol seperti tanda baca dan *emoticon* dalam teks. Simbol yang akan dihilangkan ditampilkan pada tabel 2.1.

Tabel 2.1 Daftar tabel tanda baca dan *emoticon*

Karakter	Keterangan
?	Tanda tanya
!	Tanda Seru
"	Tanda Kutip
☺	Ekspresi Senyum
☹	Ekspresi sedih
😐	Ekspresi Datar
.	titik

,	Koma
;	Titik koma

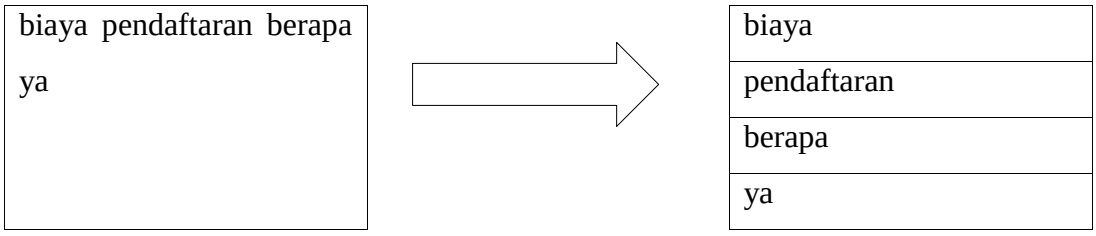


Gambar 2.2 Contoh Penghilangan Simbol

Pada Gambar 2.2 terdeteksi simbol tanda baca "?" atau tanda tanya yang masuk dalam daftar tabel 2.1 pada kalimat "biaya pendaftaran berapa ya?", maka pada proses ini simbol di hilangkan menjadi "biaya pendaftaran berapa ya".

2.3.3. *Tokenizing*

Tokenizing adalah proses untuk memisahkan kata dalam teks menjadi beberapa urutan kata yang dipisahkan melalui spasi atau karakter lainnya [12]. Pada penelitian, spasi digunakan untuk memisahkan kata dalam teks. Berikut merupakan contoh proses *tokenizing* seperti yang ditampilkan pada gambar 2.3.



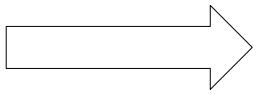
Gambar 2.3 Contoh Proses *Tokenizing*

2.3.4. *Remove Stopword*

Remove Stopword merupakan proses penghilangan kata tidak penting yang terdapat dalam *stoplist*. Kata-kata yang dihilangkan tersebut menunjukan kurang

relevansinya dengan teks dan didefinisikan dalam *stopword list* [12]. Contoh dari proses *remove stopword* ditampilkan pada gambar 2.4.

biaya		biaya
pendaftaran		pendaftaran
berapa		berapa
ya		



Gambar 2.4 Proses *Remove Stopword*

2.3.5. TF-IDF

Metode TF-IDF merupakan metode untuk menghitung bobot setiap kata yang paling umum digunakan pada *information retrieval*. Metode ini juga terkenal efisien, mudah dan memiliki hasil yang akurat [13]. Metode ini akan menghitung bobot setiap *term* dalam sebuah dokumen dengan rumus:

$$IDF_{t_i} = log(\frac{D}{df_i}) \quad (2.1)$$

Dimana:

D : total keseluruhan dokumen,

df_i : banyaknya dokumen yang mengandung kata ke –i yang dicari

Kemudian algoritma yang digunakan untuk menghitung bobot (w) *term* dari masing-masing dokumen terhadap term menggunakan persamaan [13].

$$w_i = tf_i * IDF_{t_i} \quad (2.2)$$

Dimana :

tf_i: *term* frekuensi/ frekuensi kata

W_i: bobot dokumen ke-d terhadap term ke-t

2.4. *Support Vector Machine (*SVM*)*

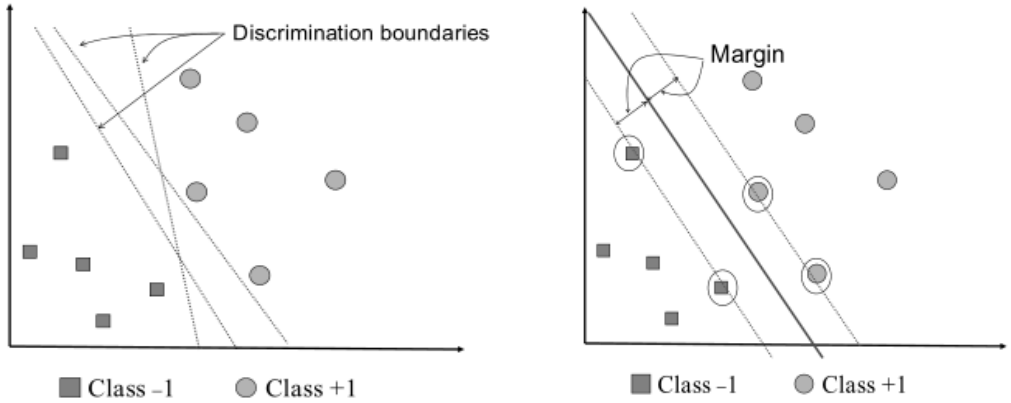
Support Vector Machine (SVM) adalah sistem pembelajaran yang menggunakan ruang hipotesis berupa fungsi-fungsi *linier* dalam sebuah ruang fitur (*feature space*) berdimensi tinggi, dilatih dengan algoritma pembelajaran yang didasarkan pada teori optimasi dengan mengimplementasikan *learning bias* yang berasal dari teori pembelajaran statistik. Teori yang mendasari SVM sendiri sudah berkembang sejak 1960-an, tetapi baru diperkenalkan oleh Vapnik, Boser dan Guyon pada tahun 1992 dan sejak itu SVM berkembang dengan pesat.

SVM adalah salah satu teknik yang relatif baru dibandingkan dengan teknik lain, tetapi memiliki performansi yang lebih baik di berbagai bidang aplikasi seperti *bioinformatics*, pengenalan tulisan tangan, klasifikasi teks dan lain sebagainya [14].

2.4.1. Konsep SVM

Ide dasar SVM adalah memaksimalkan hyperplane (*maximal margin hyperplane*), seperti diilustrasikan pada gambar 2.5. Pada gambar 2.5.(a) ada sejumlah pilihan hyperplane yang mungkin untuk set data, dan 2.5. (b) merupakan hyperplane dengan margin yang paling maksimal. Meskipun 2.5. (a) sebenarnya juga bisa menggunakan hyperplane sembarang, hyperplane dengan margin yang maksimal akan memberikan generalisasi yang lebih banyak pada metode klasifikasi [15].

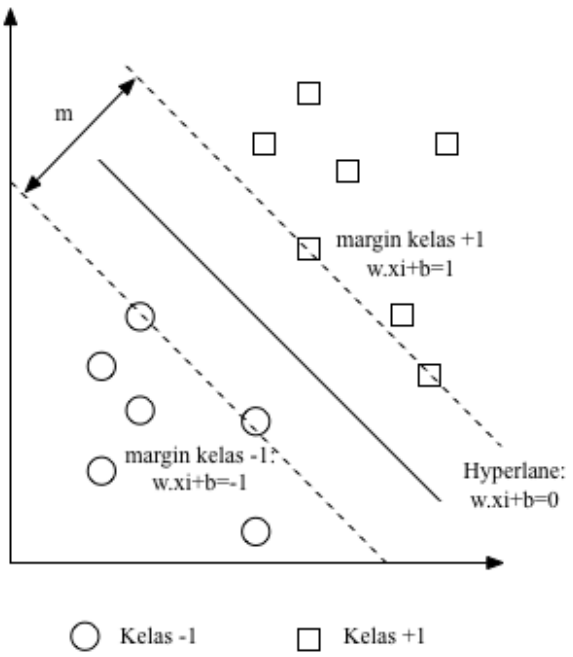
Konsep klasifikasi dengan SVM dapat dijelaskan secara sederhana sebagai usaha mencari hyperplane terbaik yang berfungsi sebagai pemisah dua buah kelas data pada ruang input. Gambar 2.5 memperlihatkan beberapa pola yang merupakan anggota dari dua buah class : +1 dan –1. data yang tergabung pada kelas –1 disimbolkan dengan warna merah (kotak), sedangkan pattern pada class +1, disimbolkan dengan warna kuning(lingkaran) [15].



a. Decision Boundary yang mungkin **b. Decision boundary dengan margin maksimal**

Gambar 2.5 Decision boundary yang mungkin untuk set data [15]

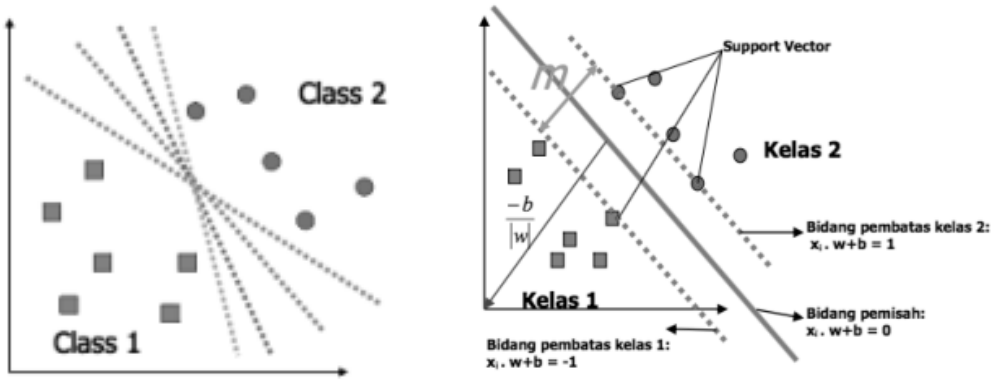
Hyperplane pemisah terbaik antara kedua kelas dapat ditemukan dengan mengukur *margin hyperplane* tsb. dan mencari titik maksimalnya. *Hyperplane* adalah garis batas pemisah data antar kelas. *Margin* adalah jarak antara *hyperplane* tersebut dengan *pattern* terdekat dari masing-masing kelas. *Pattern* yang paling dekat ini disebut sebagai *support vector*. Garis solid pada gambar 2.5 (b) menunjukkan hyperplane yang terbaik, yaitu yang terletak tepat pada tengah-tengah 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 [15].



Gambar 2.6 Margin Hyperplane [15]

2.4.2. SVM pada *Linearly Separable Data*

Linearly separable data merupakan data yang dapat dipisahkan secara *linier*. Misalkan $\{x_1, x_2, \ldots, x_n\}$ adalah dataset dan $yi \in \{+1, -1\}$ adalah label kelas dari data x_i. Pada gambar 2.7 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 [14].



Gambar 2.7 Alternatif bidang pemisah (kiri) dan bidang pemisah terbaik dengan margin (m) terbesar(kanan) [14]

Adapun data yang berada pada bidang pembatas ini disebut *support vector*. Dalam contoh di atas, dua kelas dapat dipisahkan oleh sepasang bidang pembatas yang sejajar. Bidang pembatas pertama membatasi kelas pertama sedangkan bidang batas kedua membatasi kelas kedua [14].

Setiap data latih dinyatakan oleh (x_i, y_i), dimana *i*=1,2,...,N, dan x_i= $\{x_{i1}, x_{i2}, .., x_{iq}\}^T$ merupakan atribut (fitur) set untuk data latih ke-*i*, $y_i \in$ {-1,+1} menyatakan label kelas. *Hyperplane* klasifikasi linear SVM, seperti pada gambar 2.6., dinotasikan dengan

$$\boldsymbol{w}.\boldsymbol{x_i} + b = 0 \qquad (2.3)$$

w dan b adalah parameter model. $\boldsymbol{w}.\boldsymbol{x_i}$ merupakan inner-product dalam antara w dan xi. Data xi yang masuk ke dalam kelas ke -1 adalah data yang memenuhi pertidaksamaan berikut:

$$\boldsymbol{w}.\boldsymbol{x_i} + b \leq -1 \qquad (2.4)$$

Sementara data xi yang masuk ke dalam kelas +1 adalah data yang memenuhi pertidaksamaan berikut:

$$\boldsymbol{w}.\boldsymbol{x_i} + b \geq +1 \qquad (2.5)$$

Sesuai dengan gambar 2.6, jika ada data dalam kelas -1 (misalnya, xn) yang bertempat di hyperplane, persamaan 2.3 akan terpenuhi. Untuk data kelas -1 dengan

$$\boldsymbol{w}.\boldsymbol{x}_a + b = 0 \quad (2.6)$$

Sementara kelas +1 (misal xb) akan memenuhi persamaan

$$\boldsymbol{w}.\boldsymbol{x}_b + b = 0 \quad (2.7)$$

$x_b - x_a$ adalah vektor paralel di posisi hyperplane dan diarahkan dari x_a ke x_b. Karena inner-product dalam bernilai nol, arah w harus tegak lurus terhadap hyperplane (bidang pemisah 2.6).

Dengan memberikan label -1 untuk kelas 1 dan +1 untuk kelas 2, prediksi semua data uji akan menggunakan formula

y= + 1 , jika $w.z + b > 0$

y= - 1 , jika $w.z + b < 0$ (2.8)

Sesuai dengan gambar 2.6. , hyperplane untuk kelas -1 (garis-garis putus) adalah data support vector yang memenuhi persamaan

$$w.x_a + b = -1 \quad (2.9)$$

Sementara hyperplane kelas +1 (garis putus-putus) memenuhi persyaratan

$$w.x_b + b = +1 \quad (2.10)$$

Dengan demikian, margin dapat dihitung dengan mengurangi persamaan 2.9 dengan 2.10 didapatkan

$$w.(x_b - x_a) = 2 \quad (2.11)$$

Margin hyperplane diberikan oleh jarak antara dua hyperplane dari dua kelas tersebut. Notasi di atasi diringkas menjadi.

$\|W\| \times d = 2$ atau d=$\frac{2}{\|w\|}$ (2.12)

Klasifikasi kelas data pada SVM pada persamaan 2.5 dan 2.6 dapat digabungkan dengan notasi

$$y_i(w.x_i + b) \geq 1, i = 1,2,..\mathrm{N} \quad (2.13)$$

Margin optimal dihitung dengan memaksimalkan jarak antara hyperplane dan data terdekat. Jarak ini dirumuskan dengan persamaan 2.12 ($\|w\|$ adalah

vektor bobot w). Selanjutnya, masalah ini diformulasikan ke dalam quadratic programming (QP) problem, dengan meminimalkan invers persamaan 2.12, $\frac{1}{2}\|w\|^2$, dibawah kendala (syarat), seperti berikut:

Minimalkan:

$$\frac{1}{2}\|w\|^2 \quad (2.14)$$

Syarat :

$$y_i(w.x_i + b) \geq 1,\ i = 1, 2, .., N \quad (2.15)$$

Optimalisasi ini dapat diselesaikan dengan Lagrange Multiplier.

$$Lp = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} a_i y_i (w.x_i + b) - 1 \quad (2.16)$$

a_i adalah Lagrange Multiplier yang berkorespondensi dengan x_i. Nilai a_i adalah nol atau positif. Untuk meminimalkan Lagrangian, Persamaan 2.16 harus diturunkan pada *w* dan *b*, dan diset dengan nilai nol untuk syarat optimalisasi diatas [15].

Syarat 1:

$$\frac{\partial Lp}{\partial w} = 0 \rightarrow \mathrm{w} = \sum_{i-1}^{N} a_i y_i x_i \quad (2.17)$$

Syarat 2:

$$\frac{\partial Lp}{\partial b} = 0 \rightarrow \sum_{i-1}^{N} a_i y_i = 0 \quad (2.18)$$

Vektor *w* sering kali bernilai besar (mungkin tak terhingga), tetapi nilai α_i terhingga. Untuk itu, formula lagrangian diubah kedalam *dual problem* LD. Dengan mensubtitusikan persamaan 2.17 ke Lp diperoleh *dual problem* LD dengan konstrain berbeda.

$$L_D(\alpha) = \sum_{i=1,j=1}^{n} a_i a_j y_i y_j x_i x_j \quad (2.19)$$

$$\min_{w,b} L_p = \max_{a} L_D$$

Jadi persoalan pencarian bidang pemisah terbaik dapat dirumuskan sebagai berikut.

$$\max_{\alpha} L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i x_j \quad (2.20)$$

$$s.t. \sum_{j=1}^{n} \alpha_i y_i = 0, \alpha_i \geq 0 \quad (2.21)$$

Dengan demikian, dapat diperoleh nilai α_i yang nantinya digunakan untuk menemukan w. Terdapat nilai α_i untuk setiap data pelatihan. Data pelatihan yang memiliki nilai $\alpha_i > 0$ adalah *support vector* sedangkan sisanya memiliki nilai$\alpha_i = 0$. Dengan demikian fungsi keputusan yang dihasilkan hanya dipengaruhi oleh *support vector*.

Formula pencarian bidang pemisah terbaik ini adalah pemasalahan *quadratic programming*, sehingga nilai maksimum global dari α_i selalu dapat ditemukan. Setelah solusi pemasalahan *quadratic programming* ditemukan (nilai α_i), maka kelas dari data pengujian x dapat ditentukan berdasarkan nilai dari fungsi keputusan.

$$f(x_d) = \sum_{i=1}^{n} \alpha_i y_i x_i . x_d + b \quad (2.22)$$

x_i adalah support vector, n adalah jumlah support vector dan x_d adalah data yang akan diklasifikasikan.

2.4.3. SVM pada *Nonlinearly Separable Data*

Untuk mengklasifikasikan data yang tidak dapat dipisahkan secara linier formula SVM harus dimodifikasi karena tidak akan ada solusi yang ditemukan. Oleh karena itu, kedua bidang pembatas (2.1) harus diubah sehingga lebih fleksibel (untuk kondisi tertentu) dengan penambahan variabel $\xi_i (\xi_i \geq 0, \forall_i : \xi_i = 0$ jika x_i diklasifikasikan dengan benar) menjadi $x_i . w + b \geq 1 - \xi$ untuk kelas 1 dan $x_i . w + b \leq -1 + \xi_i$ untuk kelas 2.

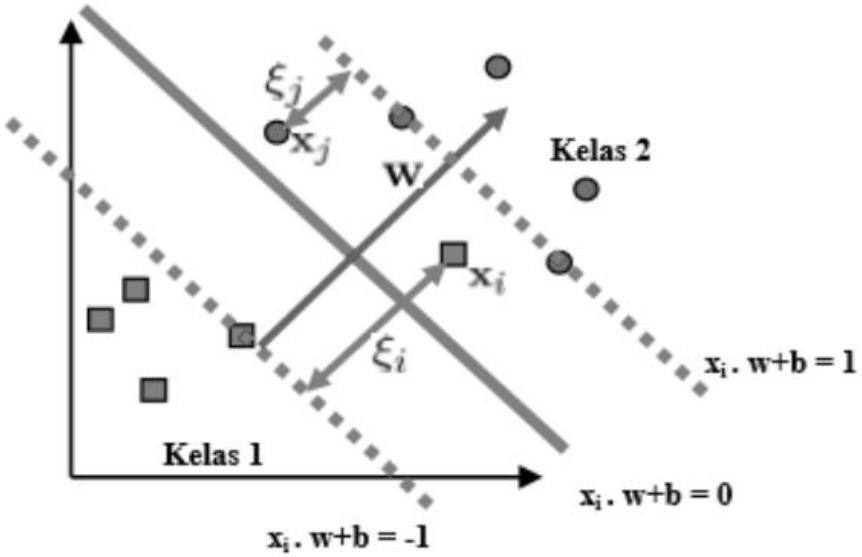
Pencarian bidang pemisah terbaik dengan penambahan variabel ξ_i sering disebut *soft margin hyperplane.* Dengan demikian formula pencarian bidang pemisah terbaik berubah menjadi.

$$\min \frac{1}{2}\|w\|^2 + C(\sum_{i=1}^{N} \xi_i) \tag{2.23}$$

$$s.t.\, y_i(w.x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

C adalah parameter yang menentukan besar penalti akibat kesalahan dalam klasifikasi data dan nilainya ditentukan oleh pengguna. Bentuk persoalan (2.23) memenuhi prinsip SRM, dimana meminimumkan $\frac{1}{2}\|w\|^2$ ekivalen dengan meminimumkan dimensi VC dan meminimumkan $C(\sum_{i=1}^{N} \xi_i)$ berarti meminimumkan error pada data pelatihan, nilai ai adalah $0 \geq a_i \geq C$. *Instance* yang memiliki nilai $a_i = C$ disebut *bounded support vector*.



Gambar 2.8 Soft margin hyperplane

Selanjutnya, bentuk primal problem sebelumnya diubah menjadi.

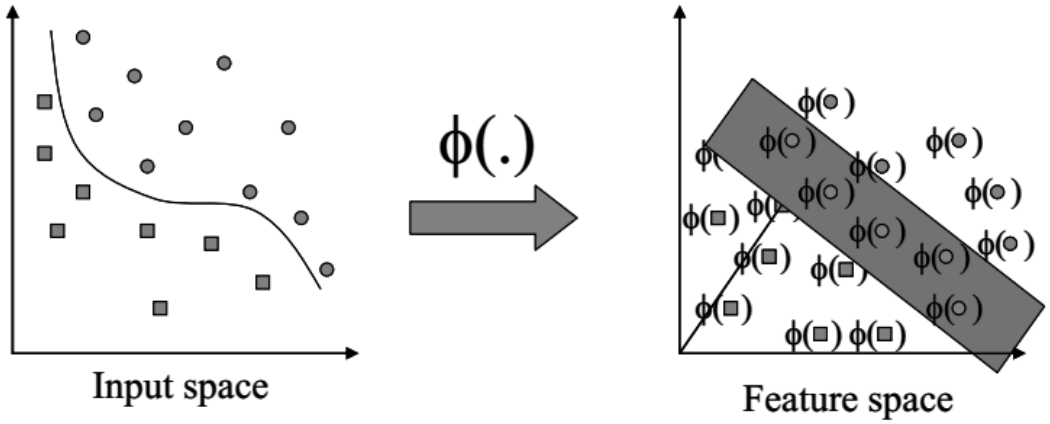
$$\min_{w,b} Lp(w,b,a) = \frac{1}{2}\|w\|^2 + C(\sum_{i=1}^{n}\xi_i) - \sum_{i=1}^{n} a_i\{y_i(x_i.w+b) - 1 + \xi_1\} - \sum_{i=1}^{n}\mu_i\xi_i$$

(2.24)

Pengubahan Lp ke dalam dual problem, menghasilkan formula yang sama dengan persamaan $w = \sum_{i=1}^{n} a_i y_i x_i$ sehingga pencarian bidang pemisah terbaik dilakukan dengan cara yang hampir sama dengan kasus dimana data dapat dipisahkan secara linear, tetapi rentang nilai ai adalah $0 \leq a_i \geq C$. Instance yang memiliki nilai a_i = C disebut *bounded support vector*.

Metode lain untuk mengklasifikasikan data yang tidak dapat dipisahkan secara linier adalah dengan mentransformasikan data ke dalam dimensi ruang fitur (*feature space*) sehingga dapat dipisahkan secara linier pada *feature space.*

Caranya, data dipetakan dengan menggunakan fungsi pemetaan (transformasi)$x_k \rightarrow \boldsymbol{\phi}(\boldsymbol{x_k})$ dalam *feature space* sehingga terdapat bidang pemisah yang dapat memisahkan data sesuai dengan kelasnya (gambar 2.9).



Gambar 2.9 Contoh transformasi data yang tidak dapat dipisahkan secara linear

Dengan mengubah *input space* ke *feature space* $x_k \rightarrow \boldsymbol{\phi}(\boldsymbol{x_k})$**,** maka nilai $w = \sum_{i=1}^{ns} a_i y_i \phi(x_i)$ dan fungsi hasil pembelajaran yang dihasilkan adalah.

$$f(x_d) = \sum_{i=1}^{ns} a_i y_i \phi(x_i)\phi(x_d) + b \qquad (2.25)$$

Feature space dalam prakteknya biasanya memiliki dimensi yang lebih tinggi dari vektor input (*input space*). Hal ini mengakibatkan komputasi pada feature space mungkin sangat besar, karena ada kemungkinan *feature space* dapat memiliki jumlah *feature* yang tidak terhingga. Selain itu, sulit mengetahui fungsi transformasi yang tepat. Untuk mengatasi masalah ini, pada SVM digunakan "*kernel trick*". Dari persamaan (2.24) dapat dilihat terdapat dot product $\boldsymbol{\phi}(x_i)$ $\boldsymbol{\phi}(x_{d_i})$. Jika terdapat sebuah fungsi kernel K sehingga $K(x_i, x_d)$=$\boldsymbol{\phi}(x_i).\boldsymbol{\phi}(x_d)$, maka fungsi transformasi $\boldsymbol{\phi}(x_k)$ tidak perlu diketahui secara persis. Dengan demikian fungsi yang dihasilkan dari pelatihan adalah

$$f(x_d) = \text{sign}(w.x_i + b) = \text{sign}(\sum_{i=1}^{n} a_i y_i K(x_i, x_d) + b) \qquad (2.26)$$

n adalah jumlah data yang menjadi support vector, x_i adalah support vector, dan x_d adalah data uji yang akan dilakukan prediksi.

Fungsi kernel yang umum digunakan adalah sebagai berikut [14]:

Fungsi Kernel		
Linear	$K(x_i, x_j) = x_i^T x$	2.27
Polynomial Kernel	$K(x_i, x) = (\gamma . x_i^T x + r)^p, \gamma > 0$	2.28
Radial Basis Function (RBF)	$K(x_i, x) = \exp(-\gamma \|x_i - x\|^2), \gamma > 0$	2.29
Sigmoid Kernel	$K(x_i, x) = \tanh(\gamma x_i^T x + r)$	2.30

2.4.4. *Multiclass Support Vector Machine* (SVM)

SVM hanya dapat melakukan klasifikassi biner (dua kelas). Sementara masalah di dunia nyata umumnya mempunyai banyak kelas seperti pengenalan karakter, pengenalan wajah, atau diagnosis pasien, dimana data masukan terbagi menjadi lebih dari dua kelas. Ada 3 pendekatan SVM Multikelas dua diantaranya yaitu *one-against-all* dan *one-against-one*. Untuk sejumlah label kelas dalam set data dinotasikan $Y = \{y_1, y_2, y_3, \ldots, y_k\}$ [15].

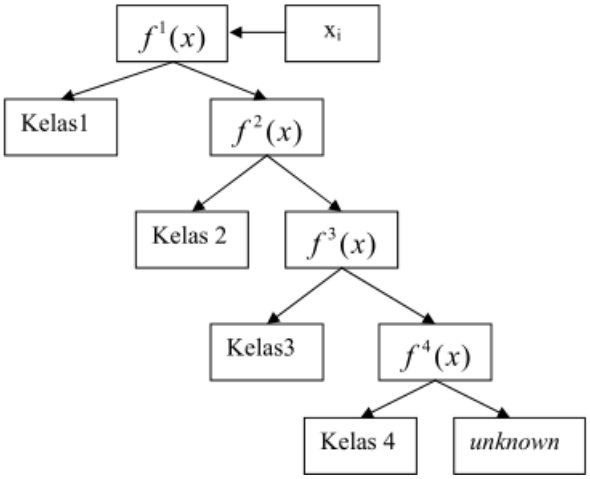
2.4.4.1. Metode satu lawan semua (*One Against All*)

Pendekatan *one-against-all* (OAA) melakukan dekomposisi masalah multikelas menjadi K masalah biner. Untuk setiap $y_i \in Y$, masalah biner dibuat di mana semua vektor yang dimiliki kelas y_i dipandang sebagai sample positif, sedangkan lainnya dipandang sebagai sampel negatif. klasifikasi SVM Biner sebanyak K kemudian dibentuk untuk memisahkan vektor kelas y_i dari yang lainnya [15].

Contohnya, terdapat permasalan klasifikasi dengan 4 buah kelas. Untuk pelatihan digunakan 4 buah SVM biner seperti pada tabel 2.6 dan penggunaannya dalam mengklasifikasi data baru dapat dilihat pada gambar 2.10 [14].

Tabel 2.6. Contoh 4 SVM biner dengan metode *One Against All*

$y_i = 1$	$y_i = -1$	Hipotesis
Kelas 1	Bukan Kelas 1	$f^1(x) = (w^1)x + b^1$
Kelas 2	Bukan Kelas 2	$f^2(x) = (w^2)x + b^2$
Kelas 3	Bukan Kelas 3	$f^3(x) = (w^3)x + b^1$
Kelas 4	Bukan Kelas 4	$f^4(x) = (w^4)x + b^4$



Gambar 2.10 Contoh Klasifikasi dengan metode *One Against All*

2.4.4.2. Metode satu lawan satu (*One Against One*)

Pendekatan *one-against-one* (OAO) membentuk klasifikator biner sebanyak K(K-1)/2 (K merupakan jumlah kelas), dimana setiap klasifikator digunakan untuk membedakan diantara pasangan kelas (y_i, y_j). Vektor yang tidak menjadi anggota kelas y_i ataupun y_j diabaikan ketika pembentukan klasifikator biner (y_i, y_j) [15].

Terdapat beberapa metode untuk melakukan pengujian setelah keseluruhan $\frac{k(k-1)}{2}$model klasifikasi selesai dibangun. Salah satunya dengan metode *voting* [14].

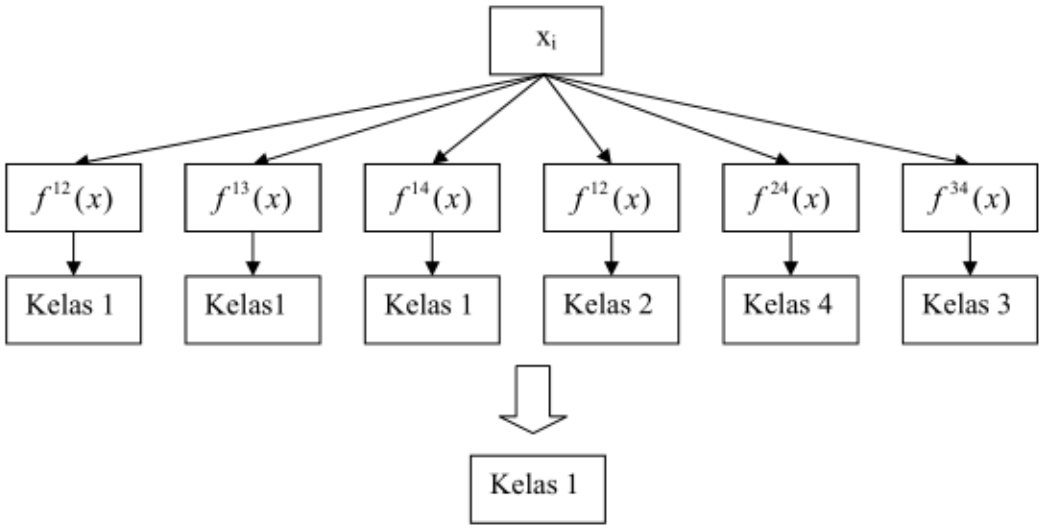
Tabel 2.2 Contoh 4 SVM biner dengan metode *One Against One*

$y_i = 1$	$y_i = -1$	Hipotesis
Kelas 1	Kelas 2	$f^{12}(x) = (w^{12})x + b^{12}$
Kelas 1	Kelas 3	$f^{13}(x) = (w^{13})x + b^{13}$
Kelas 1	Kelas 4	$f^{14}(x) = (w^{14})x + b^{14}$
Kelas 2	Kelas 3	$f^{23}(x) = (w^{23})x + b^{23}$
Kelas 2	Kelas 4	$f^{24}(x) = (w^{24})x + b^{24}$

Kelas 3	Kelas 4	$f^{34}(x) = (w^{34})x + b^{34}$

Jika data x dimasukkan ke dalam fungsi hasil pelatihan $f(x) = (w^{ij})^T \boldsymbol{\phi}(\boldsymbol{x}) + \boldsymbol{b})$ dan hasilnya menyatakan x adalah kelas i, maka suara untuk kelas i ditambah satu. Kelas dari data x akan ditentukan dari jumlah suara terbanyak. Jika terdapat dua buah kelas yang jumlah suaranya sama, maka kelas yang indeksnya lebih kecil dinyatakan sebagai kelas dari data. Jadi pada pendekatan ini terdapat $\frac{k(k-1)}{2}$ buah permasalahan *quadratic programming* yang masing- masing memiliki $\frac{2n}{k}$ variabel (n adalah jumlah data pelatihan) [14].

Contohnya, terdapat permasalahan klasifikasi dengan 4 buah kelas. Oleh karena itu, digunakan 6 buah SVM biner seperti pada tabel 2.2 dan contoh penggunaanya dalam memprediksi kelas data baru dapat dilihat pada gambar 2.11 [14].



Gambar 2.11 Contoh klasifikasi dengan metode *One Against One*

Pada penelitian ini, fungsi kernel yang akan digunakan adalah kernel *Radial Basis Function* (RBF) pada persamaan 2.29 dan SVM yang digunakan *Multiclass* SVM *One Against One* (OAO)/*One Vs One* (OVO). Proses klasifikasi SVM menggunakan library LIBSVM PHP yang dibuat oleh ian barber.

2.5. *Cosine Similarity*

Cosine similarity digunakan untuk menghitung pendekatan relevansi *query* terhadap dokumen. Semakin besar nilai kesamaan vektor *query* dengan vektor dokumen maka *query* tersebut dipandang semakin relevan dengan dokumen. Cosinus dari 0^0 adalah 1, dan kurang dari 1 untuk setiap sudut lainnya. Dengan demikian dua vektor dengan orientasi yang sama memiliki kesamaan cosinus dari 1, dua vektor pada 90^0 memiliki kesamaan 0. Cosine similarity terutama digunakan dalam ruang positif, dimana hasilnya dibatasi (0,1). Cosine similarity kemudian memberi tolak ukur seberapa mirip dua dokumen [16]. Pada penelitian ini, metode *cosine similarity* digunakan untuk pencarian jawaban pada pengujian data, sehingga didapatkan persamaan.

$$CS(q_{testing}, q_i) = \frac{\sum_{i=1}^{n} W_{t,q_{testing}} \times W_{t,q_i}}{\sqrt{\sum_{j=1}^{n} W_{t,q_{testing}}^2} \sqrt{\sum_{i=1}^{n} W_{t,q_i}^2}} \qquad (2.31)$$

Dimana :

$W_{t,q_{testing}}$ = bobot term t pada *question* $q_{testing}$.

W_{t,q_i} = bobot term t pada *question* q_i.

2.6. *Black-Box Testing*

Pengujian ini fokus kepada persyaratan fungsional perangkat lunak. Pengujian ini memungkinkan pelaku RPL mendapatkan serangkaian kondisi *input* yang memenuhi persyaratan fungsional suatu program.

Pengujian ini berusaha menemukan kesalahan dengan kategori sebagai berikut:

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan antarmuka
3. Kesalahan struktur data atau akses basisdata eksternal
4. Kesalahan kinerja
5. Kesalahan inisialisasi atau terminasi

Dengan mengaplikasikan teknik pengujian ini, penguji membuat serangkaian

kasus uji yang:

1. Mengurangi jumlah kasus uji tambahan yang harus dirancang untuk mencapai pengujian yang benar.
2. Memberi tahu mengenai ada atau tidaknya kesalahan [17].

2.6. Akurasi

Untuk menghitung akurasi dari SVM dilakukan perhitungan dengan rumus berikut [15].

$$Akurasi = \frac{Jumlah\ Jawaban\ Relevan}{Total\ Jawaban} * 100\% \quad (2.32)$$

Pada penelitian ini, menggunakan perhitungan akurasi untuk mengetahui tingkat akurasi dari ketepatan jawaban pengujian data pada sistem tanya jawab.

2.7. Pemodelan Sistem

Pemodelan sistem merupakan proses pengembangan model abstrak dari sistem, dengan tiap model memperlihatkan pandangan berbeda dari sistem. Adapun yang digunakan dalam pemodelan sistem pada penelitian tugas akhir ini adalah sebagai berikut.

2.7.1. Basis Data

Basis Data terdiri dari 2 kata, yaitu Basis dan Data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang/berkumpul. Sedangkan Data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa konsep, keadaan, dan sebagainya yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya [18].

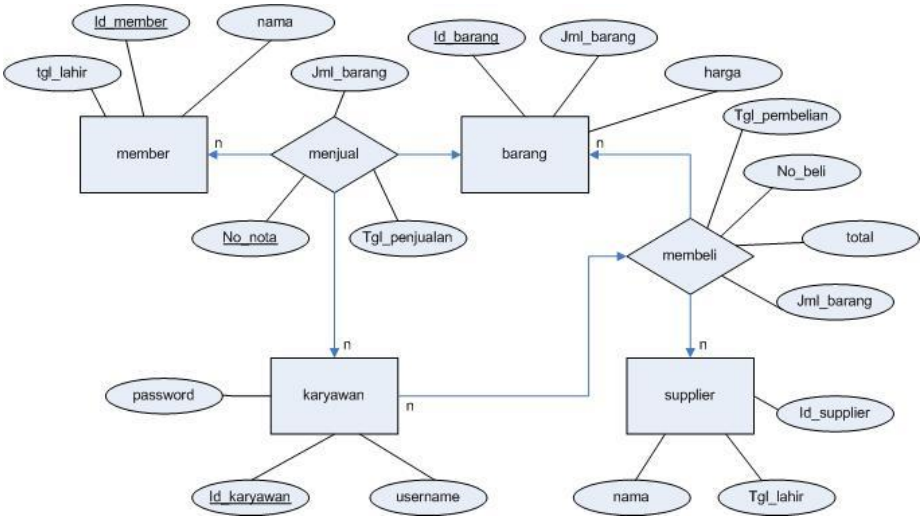
Basis data dibangun oleh beberapa komponen diantarnya ada enam komponen pokok antara lain ialah.

1. Perangkat keras (*Hardware*).
2. Sistem operasi (*Operating System*).
3. Basis data(*Database*).
4. Sistem (Aplikasi/Perangkat Lunak) Pengelola Basis Data (DBMS)

5. Pemakai (User).
6. Aplikasi (perangkat lunak) lain (bersifat opsional) [18].

2.7.2. *Entity Relationship Diagram* (ERD)

ERD merupakan suatu model jaringan yang menggunakan susunan data yang disimpan pada sistem secara abstrak. ERD juga menggambarkan hubungan antara satu entitas yang memiliki sejumlah atribut dengan entitas lain dalam suatu sistem yang terintegrasi. ERD digunakan oleh perancang sistem untuk memodelkan data yang nantinya akan dikembangkan menjadi basis data. ERD ini juga merupakan model konseptual yang dapat mendeskripsikan hubungan antara file yang digunakan untuk memodelkan struktur data serta hubungan antar data [19]. Pada penelitian ini, ERD digunakan untuk memodelkan data yang akan dibangun pada sistem ini. Berikut contoh dari ERD dapat dilihat pada gambar.



Gambar 2.12 Contoh ERD Swalayan

2.7.3. *Flowchart*

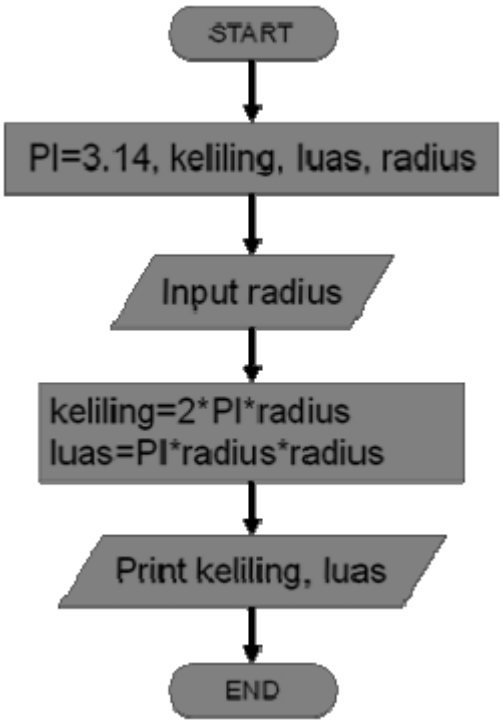
Flowchart adalah cara penulisan algoritma dengan menggunakan notasi grafis. Flowchart merupakan gambar atau bagan yang memperlihatkan urutan atau langkah-langkah dari suatu program dan hubungan antar proses beserta pernyataannya. Gambaran ini dinyatakandengan simbol. Dengan demikian setiap simbol menggambarkan proses tertentu. Sedangkan antara proses digambarkan dengan garis penghubung. Dengan menggunakan *flowchart* akan memudahkan

kita untuk melakukanpengecekan bagian-bagian yang terlupakan dalam analisis masalah [20].

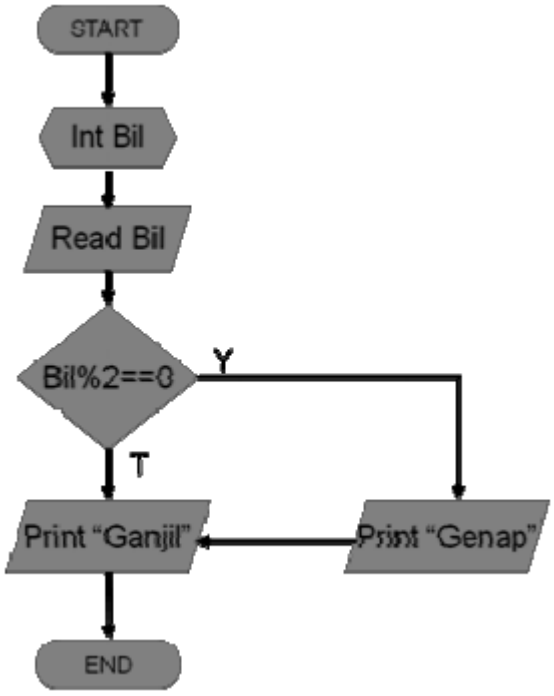
Disamping itu *flowchart* juga berguna sebagai fasilitas untuk berkomunikasi antara pemrogram yang bekerja dalam tim suatu proyek. *Flowchart* menolong analis dan programmer untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian [20].

Pada dasarnya terdapat berbagai macam *flowchart*, diantaranya yaitu *Flowchart* Sistem (*System Flowchart*), *Flowchart Paperwork* / *Flowchart* Dokumen (*Document* Flowchart), *Flowchart* Skematik (*Schematic Flowchart*), *Flowchart Program* (*Program Flowchart*), *Flowchart* Proses (*Process Flowchart*). Untuk keperluan pembuatan program maka digunakan *Flowchart Program* [20].

Flowchart program menggambarkan urutan instruksi yang digambarkan dengan simbol tertentu untuk memecahkan masalah dalam suatu program. Dalam *flowchart* program mengandung keterangan yang lebih rinci tentang bagaimana setiap langkah program atau prosedur seharusnya dilaksanakan. *Flowchart* ini menunjukkan setiap langkah program atau prosedur dalam urutan yang tepat saat terjadi.*Programmer* menggunakan *flowchart program* untuk menggambarkan urutan instruksi dari program komputer.Analis Sistem menggunakan *flowchart program* untuk menggambarkan urutan tugas-tugas pekerjaan dalam suatu prosedur atau operasi [20].



Gambar 2.12 Contoh flowchart sistem [20]



Gambar 2.13 Contoh *Flowchart* program [20]

2.7.4. UML

Unified Modeling Language adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek(OO) [21]. Berikut adalah beberapa model yang akan digunakan dalam perancangan aplikasi data Mining untuk menggambarkan sistem dalam UML:

2.7.4.1. *Use Case Diagram*

Use Case Diagram adalah rangkaian atau uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor. *Use case* digunakan untuk membentuk tingkah laku benda dalam sebuah model serta direalisasikan oleh sebuah kolaborasi. Diagram *use case* menggambarkan fungsionalitas yang diharapkan dari sebuah sistem [22].

Berikut beberapa elemen-elemen diagram use case.

1. Sistem

   Sistem menyatakan batasan sistem dalam relasi dengan actor-actor yang menggunakannya (di luar sistem) dan fitur-fitur yang harus disediakan (dalam sistem). Sistem digambarkan dengan segiempat yang membatasi semua *use case* dalam sistem terhadap pihak mana sistem akan berinteraksi. Sistem disertai label yang menyebutkan nama dari sistem [22].

2. *Actor*

   Actor atau aktor dapat berupa manusia, sistem, atau device yang memiliki peranan dalam keberhasilan operasi dari sistem. Umumnya untuk orang digambarkan dengan icon dengan sosok lengkap seperti dengan kepala, badan tangan dan kaki. Untuk sistem,umumya digambarkan dengan segiempat disertai notasi “<<actor>>” diatas label nama [22].

3. *Use case*

   Use case mengindetifikasi fitur kunci dari sistem. Tanpa fitur ini, sistem tidak akan memenuhi permintaan *user/actor*. Setiap *use case* mengekpesikan *goal* (tujuan) dari sistem yang harus dicapai. Diberi nama sesuai dengan *goal*-nya

dan digambarkan dengan elips (dengan nama didalamnya). Fokusnya tetap pada *goal*, bukan cara implementasinya walaupun use case berimplikasi pada prosesnya nanti [22].

4. *Assosiation*

   Mengindetifikasikan interaksi antara setiap actor tertentu dengan setiap use case tertentu. Digambarkan dengan garis antara actor terhadap use case yang bersangkutan. Asosiasi bisa berarah (garis dengan anak panah) [22].

5. *Stereotape*

   Stereotype digunakan untuk menyediakan informasi lebih banyak mengenai peranan dari elemen tanpa menyebutkan implementasinya.

   Terutama untuk menggambarkan [22] :

   1. *Use case dependency*
   2. *Class*
   3. *Package-package*
   4. *Classifier*

   Sebagai contoh pada kasus ATM yaitu mencakup perilaku terhadap kejadian khusus (extend) : PIN Salah [22].

6. *Dependency*

   Dependency (<<include>>)

   1. Mengindetifikasi hubungan antar dua use case dimana yang satu memanggil yang lain.
   2. Jika pada beberapa use case terdapat bagian yang memiliki aktivitas yang sama maka bagian aktivitas tersebut biasanya dijadikan user case tersendiri dengan relasi dependensi setiap use case semula ke use case yang baru ini sehingga memudahkan pemeliharaan.
   3. Digambarkan dengan garis putus-putus bermata panah dengan notasi <<include>> pada garis.
   4. Arah mata panah sesuai dengan arah pemanggilan.

Dependensi <<extend>>

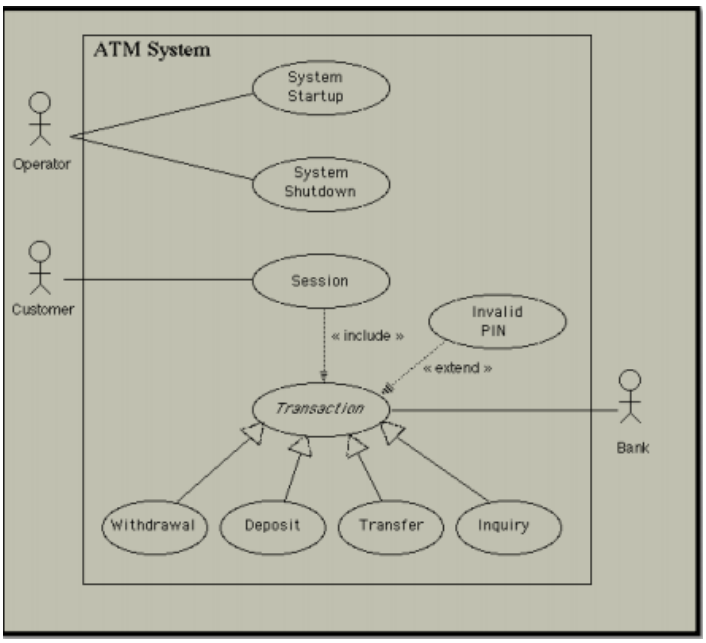
1. Jika pemanggilan memerlukan adanya kondisi tertentu maka berlaku dependensi <<extend>>.

2. Digambarkan serupa dengan dependensi <<include>> kecuali arah panah berlawanan [22].

7. *Generalization*

Mendefinisikan relasi antara dua actor atau dua *use case*. Salah satu nya meng-*inherit* dan menambahkan atau *override* sifat dari yang lainnya. Penggambaran menggunakan garis bermata panah kosong dari yang meng-*inherit* mengarah ke yang di-*inherit*.

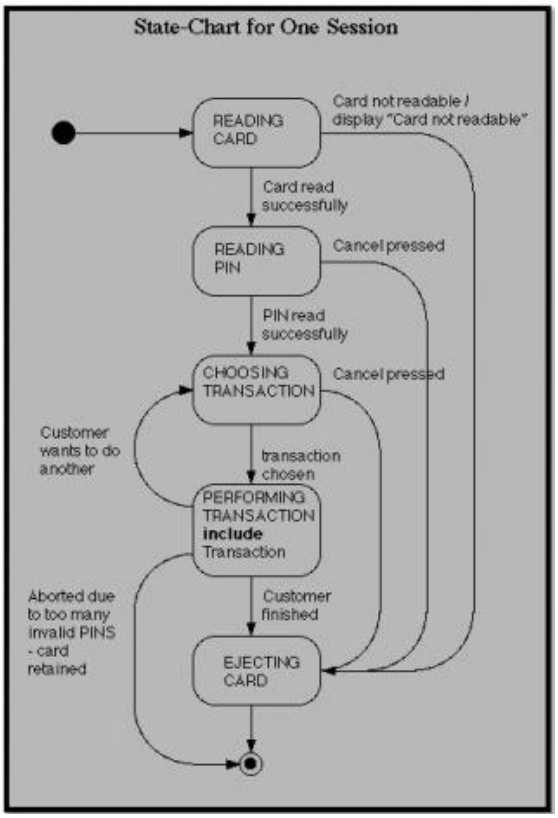
Contoh Use case dapat dilihat pada gambar 2.14



Gambar 2.14 Contoh *Use Case Diagram* pada kasus ATM [23]

2.7.3.2. *Activity Diagram*

Pada dasarnya *Activity Diagram* sering digunakan oleh flowchart. Diagram ini berhubungan dengan diagram Statechart. Diagram Statechart berfokus pada obyek yang dalam suatu proses (atau proses menjadi suatu obyek), *Activity Diagram* berfokus pada aktifitas-aktifitas yang terjadi yang terkait dalam suatu proses tunggal. Jadi dengan kata lain, diagram ini menunjukkan bagaimana aktifitas-aktifitas tersebut bergantung satu sama lain [21].



Gambar 2.15 Contoh Activity Diagram pada kasus ATM [23]

2.7.3.3. *Class Diagram*

Class Diagram memberikan pandangan secara luas dari suatu sistem dengan menunjukan kelas-kelasnya dan hubungan mereka. *Class Diagram* bersifat statis; menggambarkan hubungan apa yang terjadi bukan apa yang terjadi jika mereka berhubungan [21]. *Class Diagram* mempunyai 3 macam relationalships (hubungan), sebagai berikut :

1. *Association*

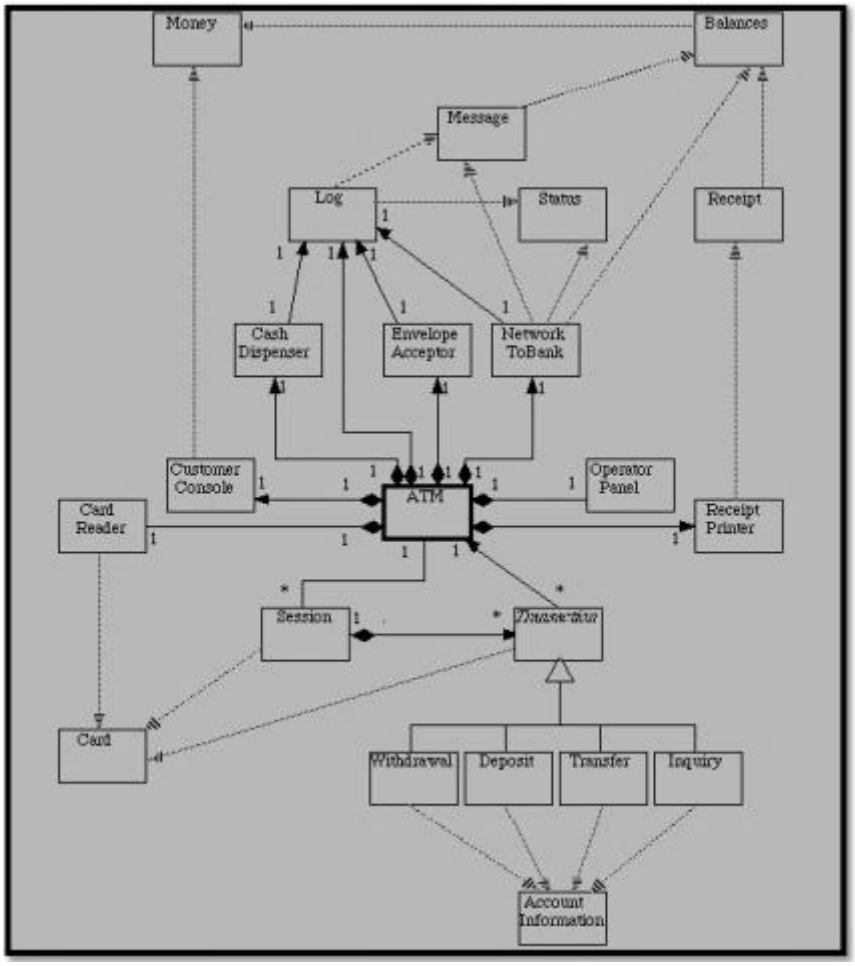
   Suatu hubungan antara bagian dari dua kelas. Terjadi association antara dua kelas jika salah satu bagian dari kelas mengetahui yang lainnya dalam melakukan suatu kegiatan. Di dalam diagram, sebuah *association* adalah penghubung yang menghubungkan dua kelas [21].

2. *Aggregation*

   Suatu association dimana salah satu kelasnya merupakan bagian dari suatu kumpulan. Aggregation memiliki titik pusat yang mencakup keseluruhan bagian. Sebagai contoh : *OrderDetail* merupakan kumpulan dari *Order* [21].

3. *Generalization*

   Suatu hubungan turunan dengan mengasumsikan satu kelas merupakan suatu *superClass* (kelas super) dari kelas yang lain. *Generalization* memiliki tingkatan yang berpusat pada *superClass* [21].



Gambar 2.16 Contoh Class Diagram pada kasus ATM [23]

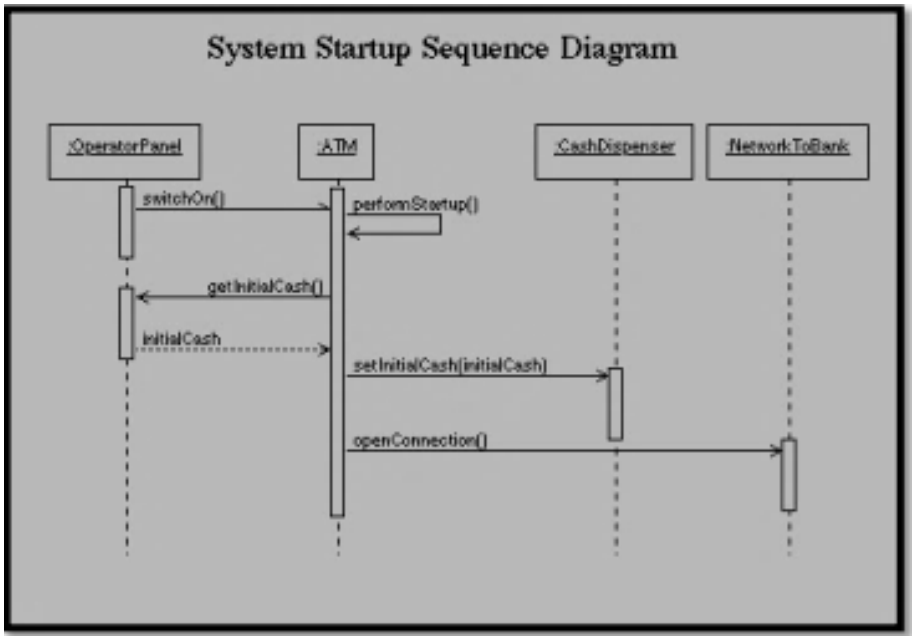
2.7.3.4. *Sequence Diagram*

Sequence diagram merupakan salah satu diagram interaksi dengan suatu objek dalam urutan waktu. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara objek juga interaksi antar objek yang terjadi pada titik

tertentu dalam eksekusi sistem. Dalam UML, objek pada diagram sequence digambarkan dengan segi empat, yang berisi nama dari objek yang digarisbawahi. Terdapat tiga (3) cara untuk menamai objek yaitu, nama objek, nama objek dan *class* serta nama *class.* Misalnya [22] :

BRI	Nama Objek
BRI : Bank	Nama Objek dan Nama Class
:Bank	Nama Kelas

Pada diagram sequence, setiap objek hanya memiliki garis yang digambarkan pada garis putus-putus kebawah. Pesan antar objek digambarkan dengan anak panah dari objek yang mengirimkan pesan ke objek yang menerima pesan [22].



Gambar 2.17 Contoh *Sequence Diagram* pada kasus ATM [23]

2.8. Bahasa Pemrograman

Bahasa pemrograman adalah notasi yang digunakan untuk menulis program (komputer) [24]. Komputer mengerjakan transformasi data berdasarkan kumpulan

perintah program yang telah dibuat oleh program. Kumpulan perintah ini harus dimengerti oleh komputer, berstruktur tertentu (syntax), dan bermakna. Bahasa pemrograman merupakan notasi untuk memberikan secara tepat program komputer.

Pada penelitian bahasa pemrograman yang akan digunakan adalah PHP, dikarenakan bahasa PHP lebih powerfull dalam pemrosesan.

2.8.3. PHP *(Hypertext Preprocessor)*

PHP adalah Bahasa *server-side –scripting* yang menyatu dengan HTML untuk membuat halaman web yang dinamis. Karena PHP merupakan serverside scripting maka sintaks dan perintah-perintah PHP akan diesksekusi diserver kemudian hasilnya akan dikirimkan ke browser dengan format HTML. Banyak sintaks di dalamnya yang mirip dengan bahasa C, Java dan Perl. Tujuan dari bahasa ini adalah membantu para pengembang web untuk membuat web dinamis dengan cepat [25].

Ketika mengunjungi web berbasis PHP, web server akan memproses code-code PHP. Beberapa perintah atau code dari PHP tersebut selanjutnya ada yang diterjemahkan ke dalam HTML dan beberapa ada yang disembunyikan (misalnya proses kalkulasi dan operasi). Setelah diterjemahkan ke dalam HTML, web server akan mengirim kembali ke web browser pengunjung tersebut [25].

2.8.4. HTML

HTML merupakan singkatan dari *hypertext markup language*, yang merupakan program penulisan informasi pada sebuah *homepage*. Penulisan HTML dapat dilakukan menggunakan alat bantu seperti *Notepad* yang terdapat pada *windows* atau *simple text machintosh* [26].

Selain itu juga dapat digunakan editor HTML seperti *Macromedia Dreamweaver* dan *Microsoft Frontpage* yang dapat memudahkan dalam menulis HTML dan memungkinkan dokumen HTML yang dibuat dapat diakses oleh berbagai jenis *Browser*. HTML berupa kode-kode tag yang memberikan instruksi pada *Web Browser* untuk memberikan tampilan sesuai yang diinginkan.

2.8.5. CSS

Cascading Style Sheet (CSS) atau yang bisa disingkat dengan CSS, merupakan suatu dokumen yang digunakan untuk melakuakan pengaturan halaman Web yang ditulis dengan HTML atau XHTML. Penggunaan CSS tidak memerlukan perangkat lunak tertentu karena CSS merupakan *script* yang telah *embedded* dengan HTML. CSS digunakan oleh *Web Designer* untuk menentukan warna, jenis, huruf, tata letak, dan berbagai aspek tampilan dokumen. CSS digunakan terutama untuk memisahkan antara isi dokumen (yang ditulis dengan HTML atau bahasa markup lainnya) dengan presentasi dokumen (yang ditulis dengan CSS). Pemisah ini ditujukan agar dapat memisahkan aksesibilitas isi, memberikan lebih banyak keleluasaan dan kontrol terhadap tampilan, dan mengurangi kompleksitas.

2.8.6. Javascript

Javascript merupakan *cross-platform* yang dikembangkan oleh Netscape dan pertama kali digunakan dalam *browser* Netscape. Javascript dibuat agar mudah diintegrasikan kedalam program dan aplikasi lain, misalnya *browser*. Sebagian besar *browser* saat ini sudah mendukung javascript. Oleh karena itu, *script* dari javascript biasanya dimasukkan kedalam suatu HTML dan dieksekusi di *browser* [26].

2.9. DBMS

Database Management System (DBMS) adalah program aplikasi yang dibuat dan bekerja dalam satu sistem. DBMS didesain untuk membantu dalam hal pemeliharaan dan utilitas kumpulan data dalam jumlah besar. DBMS dapat menjadi alternatif penggunaan secara khusus untuk aplikasi, misalnya penyimpanan data dalam *field* dan menulis kode aplikasi yang spesifik untuk pengaturannya [18].

2.9.3. MySQL

MySQL merupakan salah satu dari sekian banyak Database Server. Database Server merupakan sebuah perangkat lunak bertugas untuk menyediakan

tempat penyimpanan basis data, serta melayani permintaan informasi dari basis data tersebut. Beberapa jenis *database* server lain diantaranya, PostgreSQL, Microsoft SQL Server, Oracle, dan banyak lagi.

Perintah-perintah yang dijalankan dalam MySQL menggunakan bentuk SQL (*Structured Query Language*). SQL adalah sebuah bahasa komputer yang digunakan untuk menyimpan, memanipulasi, dan mengambil data yang tersimpan dalam basis data relasional [18].

Beberapa keunggulan atau keistimewaaan MySQL diantaranya :

1. Portabilitas

   MySQL dapat berjalan stabil pada berbagai sistem operasi, seperti Windows, Linux, FreeBSD, Mac Os Server, dan masih banyak lagi.

2. *Open Source*

   MySQL didistribusikan secara *open source*, di bawah lisensi GPL sehingga dapat digunakan secara cuma-cuma.

3. Multiuser

   MySQL dapat digunakan oleh beberapa pengguna dalam waktu bersamaan tanpa mengalami masalah konflik

4. Performance Tuning

   MySQL mampu memproses lebih banyak *query* SQL per satuan waktu.

5. Jenis Kolom

   MySQL memiliki tipe kolom yang sangat kompleks, seperti integer, float, double, char, text, date, datetime, dan lain lain.

2.10. Software Pendukung

Berikut beberapa software pendukung dalam pembuatan program dalam penelitian ini.

2.10.3. XAMPP

XAMPP adalah perangkat lunak bebas (*free software*), yang mendukung untuk banyak sistem operasi, yang merupakan kompilasi dari beberapa program. Fungsi XAMPP sendiri adalah sebagai server yang berdiri sendiri (localhost), yang terdiri beberapa program antara lain : Apache HTTP Server, MySQL database, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama XAMPP sendiri merupakan singkatan dari X (empat sistem operasi apapun), Apache, MySQL, PHP dan Perl. Program ini tersedia dalam GNU General Public License dan bebas, merupakan web server yang mudah untuk digunakan yang dapat menampilkan halaman web yang dinamis. Untuk mendapatkanya XAMPP anda dapat mendownload langsung dari web resminya [25].

Pada penelitian ini XAMPP berguna sebagai web server dan database server.

2.10.4. Web Browser

Web browser adalah suatu program perangkat lunak yang digunakan menjelajahi internet atau untuk mencari informasi diinternet. Dengan *Web browser*, setiap orang untuk menjelajah dan menemukan berbagai informasi di dunia maya dengan mudah.

Contoh *web browser* yang digunakan saat ini :

1. Internet Explorer
2. Mozila Firefox
3. Safari
4. Chrome
5. Opera

Pada penelitian ini web browser yang dipilih untuk diuji adalah *chrome* untuk menampilkan dan memproses program *prototype* sistem tanya jawab.