BAB 2
LANDASAN TEORI

2.1 Emosi

Emosi adalah gambaran keadaan seseorang yang pada umumnya disebabkan oleh kejadian penting. Emosi meliputi keadaan mental sadar, gangguan jasmani pada beberapa organ tubuh, dan pengenalan ekspresi pada wajah. Emosi merupakan suatu aspek psikis yang berkaitan dengan perasaan dan merasakan. Emosi pada diri seseorang erat kaitannya dengan suatu keadaan psikis tertentu yang distimulasi.

Sejumlah penelitian tentang emosi telah dilakukan sehingga ada kesepakatan tentang emosi dasar [8]:

1. Senang sebagai berhasil atau bergerak menuju selesainya peran yang bernilai tujuan
2. Sedih digambarkan sebagai kegagalan atau kerugian tujuan
3. Marah ganjalan atau frustasi dari peran atau tujuan yang dirasakan orang lain
4. Takut ancaman fisik atau sosial untuk diri sendiri
5. Jijik menggambarkan penghapusan atau jarak dari seseorang, atau menolak ide untuk diri sendiri.

2.2 Preprocessing

Preprocessing adalah tahapan untuk mempersiapkan teks menjadi data yang akan diolah di tahapan berikutnya. Teks yang akan dilakukan pada proses ini terdapat beberapa karakteristik, berdimensi tinggi, *noise* dan terdapat struktur yang tidak baik [9]. Adapun tahapan *preprocessing* yang akan dilakukan pada penelitian ini yaitu *case folding, convert negation, tokenizing, filtering*, dan *stopword removal* . Berikut adalah gambaran tahapan *preprocessing* yang dapat dilihat pada Gambar 2.1 :

Gambar 2.1 Tahapan *Preprocessing*

1. **Case Folding**

Case folding merupakan proses yang dilakukan untuk menyeragamkan karakter pada teks [13]. Tidak semua dokumen/teks konsisten dalam penggunaan huruf. Oleh karena itu tahapan ini mengubah semua huruf dalam teks menjadi huruf kecil (*lowercase*).

2. **Convert Negation**

Convert Negation merupakan proses konversi kata-kata negasi yang terdapat pada suatu kalimat, karena kata negasi mempunyai pengaruh dalam merubah nilai emosi pada sebuah kalimat [10]. Dalam bahasa Indonesia kata negasi mempunyai pengaruh dalam merubah nilai emosi. Perbedaan arti “tak cinta” dan “tidak mudah” dapat menempatkan lirik lagu pada emosi yang berbeda. Contoh kata negasi dalam bahasa Indonesia adalah “tak”, “tidak”, dan “gak”.

3. **Tokenizing**

Tokenizing adalah proses mengubah teks menjadi ukuran *token*. Tahapan ini yaitu penguraian lirik lagu yang berupa kalimat – kalimat menjadi kata - kata. Proses *tokenizing* dilakukan dengan cara memisahkan kata berdasarkan spasi yang terletak diantara dua kata [13].

4. **Filtering**

Filtering adalah tahap pemilihan kata – kata penting dari hasil *token*, yaitu kata apa saja yang akan mewakili isi dari dokumen. *Wordlist* adalah kata – kata deskriptif (penting) yang harus disimpan dan tidak dibuang dengan pendekatan *bag-of-word*. Pada tahap ini karakter selain huruf dihilangkan dan dianggap delimiter [10].

5. **Stopword Removal**

Stopword Removal adalah proses untuk menghilangkan kata yang tidak relevan pada sebuah dokumen teks dengan cara membandingkan dengan *stoplist* yang ada.

Stoplist berisi sekumpulan kata yang tidak relevan namun sering muncul dalam sebuah dokumen. *Stoplist* berisi sekumpulan *stopwords*.

Setiap kata akan diperiksa apakah terdapat *stoplist* atau tidak, jika sebuah kata termasuk kedalam *stoplist* maka kata tersebut tidak akan diproses lebih lanjut dan akan dihilangkan. Sebaliknya jika sebuah kata tidak termasuk kedalam s*toplist* maka kata tersebut akan masuk ke proses berikutnya.

Stoplist yang digunakan diambil dari penelitian Fadillah Z. Tala [11]. Jumlah *stopword* yang terdapat pada penelitian tersebut sebanyak 756 kata. Kata – kata yang termasuk kedalam *stoplist* biasanya berupa kata ganti orang, kata penghubung dan lain sebagainya.

2.3 Pembobotan TF-IDF

Frequency – Inverse Document Frequency digunakan untuk menentukan bobot sebuah kata di dalam banyaknya dokumen. Perhitungan statistik numerik yang dimaksudkan untuk mencerminkan seberapa pentingnya dan seberapa relevannya sebuah kata di dalam sebuah dokumen . Bobot suatu *term* akan emakin besar jika istilah tersebut sering muncul dalam suatu dokumen dan semakin kecil jika istilah tersebut muncul dalam banyak dokumen [12].

Pembobotan dapat diperoleh berdasarkan jumlah kemunculan suatu kata (*term*) dalam koleksi dokumen *term frequency* (*tf*) dan jumlah kemunculan *term* dalam koleksi dokumen *inverse document frequency* (*idf*). Nilai idf_t sebuah term dapat dihitung menggunakan persamaan 2.1 :

$$idf_t = \log(\frac{N}{df}) \quad (2.1)$$

Dengan *N* adalah jumlah dokumen dan df_i adalah kemunculan *term* terhadap *N*. Adapun persamaan yang digunakan untuk menghitung bobot (*W*) masing – masing dokumen yaitu menggunakan persamaan (2.2) :

$$W_t = tf_{dt} * idf_t \quad (2.2)$$

Dimana :

W = bobot dokumen ke-*d* terhadap kata ke-*t*

d = dokumen ke-*d*

t = *term* ke-*t*

tf = banyaknya *term* pada sebuah dokumen

N = total dokumen

df = banyak dokumen yang mengandung *term t*

2.4 Seleksi Fitur

Tahap seleksi fitur bertujuan untuk mengurangi fitur yang redudan dan tidak relevan. Fitur yang tidak relevan adalah fitur yang tidak memberikan informasi berguna tentang data. Fitur redundan adalah fitur yang tidak memberikan informasi lebih banyak daripada fitur yang saat ini dipilih [13].

Algoritma seleksi fitur terbagi ke dalam tiga pendekatan yaitu pendekatan *filter*, pendekatan *wrapper*, dan pendekatan *embedded* [13]. Pendekatan *Filter* mengevaluasi
kualitas dari fitur yang diseleksi secara independen dari algoritma klasifikasi. Pendekatan *Wrapper* membutuhkan penerapan dari algoritma klasifikasi untuk mengevaluasi kualitas klasifikasi. Pendakatan *Embedded* menerapkan seleksi fitur selama pembelajaran dari parameter-parameter yang optimal.

Metode *filter* menyeleksi fitur yang relevan sebelum berpindah pada fase pembelajaran selanjutnya. Fitur yang terlihat paling signifikan dipilih untuk klasifikasi, sementara sisanya yang lain disisihkan. Salah satu metode yang digunakan untuk melakukan *Filter* adalah *Information Gain.*

2.4.1 Information Gain

Information Gain adalah salah satu algoritma seleksi fitur yang digunakan untuk memilih fitur terbaik. Nilai *Information Gain* yang didapat akan digunakan untuk menyeleksi fitur menggunakan *threshold* sehingga menghasilkan fitur terbaik [5]. Nilai *Information Gain* dapat didefinisikan dengan persamaan (2.3) sebagai berikut [13] :

$$IG(t) = -\frac{A+C}{N} log\left(\frac{A+C}{N}\right) + \frac{A}{N} log\left(\frac{A}{A+B}\right) + \frac{C}{N} log\left(\frac{C}{C+D}\right) \quad (2.3)$$

Dimana :

A : jumlah dokumen dikelas k yang mengandung term t

B : jumlah dokumen diluar kelas k yang mengandung term t

C : jumlah dokumen dikelas k yang tidak mengandung term t

D : jumlah dokumen diluar kelas k yang tidak mengandung term t

Setelah nilai *Information Gain* didapat, maka dilakukan langkah selanjutnya adalah menentukan nilai *threshold*. Nilai *Information Gain* yang paling kecil dapat dijadikan sebagai nilai *threshold* dengan akurasi terbaik [14]. Apabila nilai *Information Gain threshold* bernilai kecil, mengakibatkan informasi penting terlalu banyak sehingga dapat menimbulkan *noise* karena masih luasnya lingkup fitur penting [5].

2.5 Support Vector Machine

Support Vector Machine adalah algoritma *supervised learning* yang digunakan untuk analisis klasifikasi dan regresi. Algoritma *Support Vector Machine* dapat beroperasi bahkan dalam set fitur yang besar dengan mengukur batas pemisahan data [15]. *Support Vector Machine* merupakan *binary classifier* yang membagi data menjadi dua kelas yang disebut *hyperlane*. *Hyperlane* ini tepat berada di tengah – tengah kedua *class. Margin* adalah jarak antara *hyperlane* tersebut dengan data terdekat dari masing – masing *class*. Data yang paling dekat dengan ini disebut *support vector* [16].

Ide dasar SVM adalah memaksimalkan batas *hyperlane*. Data yang tersedia dinotasikan dengan $(x_i y_i)$ dengan $i = 1,2, \dots, N$ dan $x_i = (x_{i1}, x_{i2}, \dots, x_{iq})^T$ merupakan atribut (fitur) set untuk data latih ke-i. Untuk $y_i \epsilon \{-1, +1\}$ merupakan label *class*. *Hyperlane* klasifikasi linear SVM dapat dinyatakan dengan persamaan (2.4) :

$$w.x_i + b = 0 \quad (2.4)$$

Bila data x_i dengan label data y_i = -1 (sampel negatif) dinyatakan dengan persamaan (2.5):

$$w.x_i + b \leq -1 \quad (2.5)$$

Sedangkan data x_i dengan label data y_i - +1 (sampel positif) dinyatakan dengan persamaan (2.6) sebagai berikut :

$$w.x_i + b \geq +1 \quad (2.6)$$

Dengan w dan b adalah parameter model. Bila label data y_i = -1 maka akan memenuhi persamaan (2.7) :

$$w.x_a + b = -1 \quad (2.7)$$

Bila label data y_i = +1 maka akan memenuhi persamaan (2.8) :

$$w.x_a + b = +1 \quad (2.8)$$

Margin terbesar dapat ditemukan dengan memaksimalkan jarak antara *hyperlane* dan titik terdekatnya. Dengan demikian *margin* dapat dihitung dengan mengurangkan persamaan (2.8) dan persamaan (2.9). maka akan didapatkan persamaan (2.9) sebagai berikut :

$$\text{w}.(x_b - x_a) = 2 \quad (2.9)$$

Margin hyperlane diberikan oleh jarak antara dua hyperlane dari dua kelas tersebut. Persamaan (2.9) dapat diringkas menjadi persamaan (2.10) sebagai berikut :

$$||w||x\ d = 2 \text{ atau } d = \frac{2}{||w||} \quad (2.10)$$

Dimana ||w|| adalah vektor bobot w dan d adalah *margin.* Selanjutnya masalah ini diformulasikan ke dala problem *Quadratic Programming* (QP) dengan meminimalkan invers persamaan (2.10), $\frac{1}{2}||w||^2$, dengan syarat sebagai berikut :

Minimalkan :

$$\frac{1}{2}||w||^2 \quad (2.11)$$

Syarat :

$$y_i(w.x_i + b) \geq 1\ , i = 1,2, \dots, N \quad (2.12)$$

Optimalisasi ini dapat dipecahkan dengan berbagai teknik komputasi menggunakan metode *Largrangian multiplier.* Metode ini dapat dinyatakan dengan persamaan (2.13) sebagai berikut :

$$L_p = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i y_i (w.x_i + b) - 1 \quad (2.13)$$

Dimana α_i adalah lagrange multiplier yang berkorespodensi dengan x_i . Nilai α_i merupakan nilai nol atau positif ($\alpha_i \geq 0$). Selanjutnya persamaan (2.13) akan diminimalan terhadap w dan b dan di set dengan nilai nol sehingga dapat dilihat syarat pada persamaan (2.14) dan persamaan (2.15) berikut :

Syarat 1:

$$\frac{\partial L_p}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{N} \alpha_i y_i x_i \quad (2.14)$$

Syarat 2:

$$\frac{\partial L_p}{\partial b} = 0 \rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 \quad (2.15)$$

Dimana N adalah jumlah data yang menjadi *support vector*. Karena *Lagrange Multiplier* (α) tidak diketahui nilainya, persamaan di atas tidak dapat diselesaikan secara langsung untuk mendapatkan w dan b. Untuk menyelesaikan masalah tersebut, modifikasilah Persamaan (2.13) diatas menjadi kasus memaksimalkan dengan syarat optimal untuk dualitas menggunakan konstrain KKT (*Karush-Kuhn-Tucker*) sebagai berikut :

Syarat 1 :

$$\alpha_i[y_i(w.x_i + b) - 1] = 0 \quad (2.16)$$

Syarat 2 :

$$\alpha_i \geq 0, \quad i = 1,2,\dots,N \quad (2.17)$$

Dengan menerapkan kendala pada persamaan (2.16) dan (2.17), dipastikan bahwa nilai *Lagrange Multiplier* sama banyaknya dengan data latih, tetapi sebenarnya banyak dari data latih yang *Lagrange Multiplier*-nya sama dengan nol (karena hanya beberpa saja yang akan menjadi *support vector*) ketika menerapkan syarat pertama. Kendala diatas menyatakan bahwa *Lagrange Multiplier* α_i harus nol, kecuali untuk data latih x_i yang memenuhi persamaan:

$$y_i(w.x_i + b) = 1 \quad (2.18)$$

Data latih tersebut, dengan $\alpha_i > 0$, terletak pada *hyperplane* b_1 atau b_2, dan disebut *support vector*. Data latih yang tidak terletak di *hyperplane* tersebut mempunyai $\alpha_i = 0$. Persamaan (2.14) dan (2.15) juga menyarankan parameter w dan b, yang mendefinisikan *hyperplane*, hanya tergantung *support vector*.

Masalah optimalisasi di atas masih sulit karena banyaknya parameter: w, b, dan α. Untuk menyederhanakannya, persamaan (2.13) harus ditransformasi ke dalam fungsi *Lagrange Multiplier* itu sendiri (disebut dualitas masalah).

Persamaan *Lagrange multiplier* dapat dijabarkan sebagai berikut :

$$L_P = \frac{1}{2}||w||^2 - \sum_{i=1}^{N} \alpha_i\, y_i(w.x_i - b \sum_{i=1}^{N} \alpha_i y_i + \sum_{i=1}^{N} \alpha_i \quad (2.19)$$

Syarat optimal persamaan (2.17) ada dalam suku ketiga di ruas kanan dalam persamaan (2.19) dan memaksa suku ini menjadi sama dengan 0. Dengan

mengganti w dari syarat persamaan (2.14) dan suku $||w||^2 = w_i w_j$, maka persamaan diatas akan berubah menjadi dualitas Lagrange multiplier berupa L_D dan didapatkan :

Maksimalkan :

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

Syarat 1:

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad (2.21)$$

Syarat 2:

$$\alpha_i \geq 0, i = 1,2, \dots, N \quad (2.22)$$

Dengan $\boldsymbol{x_i x_j}$ merupakan *dot-product* dua data dalam data latih. *Hyperplane* (batas keputusan atau pemisah) didapatkan dengan persamaan (2.23) berikut :

$$(\sum_{i=1}^{N} \alpha_i y_i x_i . z) + b = 0 \quad (2.23)$$

N adalah jumlah data yang menjadi support vector, x_i merupakan *support vector*, z merupakan data uji yang akan diprediksi kelasnya, dan$x_i . z$ merupakan *inner-product* antara x_i dan z. Untuk nilai b didapatkan dari Persamaan (2.16) pada *support vector*. Karena α_i dihitung dengan metode numerik dan mempunyai eror numerik, nilai yang dihitung untuk b bisa jadi tidak sama. Hal ini disebabkan oleh *support vector* yang digunakan dalam Persamaan (2.16), biasanya diambil nilai rata – rata dari b yang didapat untuk menjadi parameter hyperplane. Untuk mendapatkan b. Persamaan (2.16) dapat disederhanakan menjadi persamaan (2.24) berikut.

$$b_i = 1 - y_i(w . x_i) \quad (2.24)$$

.

2.5.1 Nonliniear SVM

SVM adalah *hyperlane linear* yang hanya bekerja pada data yang dapat dipisahkan secara linear. Untuk data yang distribusi kelasnya tidak linear biasanya menggunakan pendekatan kernel pada fitur data awal set data [14]. Kernel dapat didefinisikan sebagai suatu fungsi yang memetakan fitur data dari dimensi awal yang rendah ke fitur baru dengan dimensi yang relatif lebih tinggi. Algoritma pemetaan kernel dinyatakan sebagai persamaan (2.25) sebagai berikut :

$$\Phi: D^r \rightarrow D^q \qquad (2.25)$$
$$x \rightarrow \Phi(x)$$

Dengan Φ merupakan fungsi kernel yang digunakan untuk pemetaan, D merupakan data latih, r merupakan set fitur dalam satu data yang lama dan q merupakan set fitur yang baru sebagai hasil pemetaan untuk setiap data latih. Sementara x merupakan data latih, dengan $x_1, x_{2,\dots,} x_n \in D^r$ merupakan fitur – fitur yang akan dipetakan ke fitur berdimensi tinggi q. Jadi set data yang digunakan sebagai pelatihan adalah set data dari dimensi fitur yang lama r ke dimensi baru q, misalnya untuk sampel data N seperti pada persamaan (2.26) :

$$(\Phi(x_1), y_1, \Phi(x_2), y_2, \dots., \Phi(x_N), y_N) \in D^q \qquad (2.26)$$

Proses pemetaan pada fase ini memerlukan perhitungan *dot-product* dua buah data pada ruang fitur baru. *Dot-product* kedua buah vektor (x_i) dan (x_j) dinotasikan sebagai $\Phi(x_i).\Phi(x_j)$. pada umumnya transformasi Φ tidak diketahui dan sangat sulit dipahami. Oleh karena itu, perhitungan nilai *dot-product* dapat dihitung dengan fungsi kernel K($x_i x_j^T$) yang mendefinisikan secara implisit fungsi transformasi φ tersebut. Inilah yang disebut *kernel trick*, yang di formulasikan sebagai pada persamaan (2.27) :

$$K(x_i, x_j) = \Phi(x_i).\Phi(x_j) \qquad (2.27)$$

Dengan *kerner trick* ini, maka hanya perlu mengetahui wujud dari fungsi nonlinier Φ. Pada umumnya, terdapat empat jenis fungsi kernel yang dapat digunakan yaitu [16]

Tabel 2.1 Fungsi Kernel

Nama Kernel	Definisi Fungsi
Linear	$K(x, x_k) = x.x_k^T$
Polynomial	$K(x, x_k) = (x.x_k^T + 1)^d$
Gassian RBF	$K(x, x_k) = exp\left(\frac{-\|x - x_k\|^2}{2.\sigma^2}\right)$
Sigmoid	$K(x, x_k) = \tanh[K\, x_k^T.x + \theta]$

Dan untuk prediksi pada set data dengan dimensi fitur yang baru diformulasikan terdapat pada persamaan (2.28) sebagai berikut :

$$f\big(\Phi(x)\big) = sign(w.\Phi(z) + b) = sign\ (\sum_{i=1}^{N} \alpha_i y_i \Phi(x_i).\Phi(z) + b)$$

(2.28)

Dimana N adalah jumlah data yang menjadi *support vector*, x_i adalah *support vector* dan z adalah data uji yang akan dilakukan prediksi.

2.5.2 Multiclass SVM

SVM saat pertama kali diperkenalkan oleh Vapnik, hanya dapat mengklasifikasikan data ke dalam dua buah kelas (klasifikasi biner). Namun, untuk penelitian lebih lanjut untuk mengembangkan SVM sehingga bisa mengklasifikasi data yang memiliki lebih dari dua kelas, terus dilakukan [17].

Dalam penelitian ini, peneliti menggunakan metode *One Against One*. Dengan menggukanan metode ini, dibangun $\frac{k(k-1)}{2}$ buah model klasifikasi biner, dimana k adalah jumlah kelas. Metode ini harus membangun sejumlah model SVM biner yang membandingkan satu kelas dengan kelas lainnya [17]. Setiap model klasifikasi dilatih pada data dari dua kelas [18]. Untuk data pelatihan dari kelas ke-i dan kelas ke-j, dilakukan pencarian solusi untuk persoalan opimasi konstrain sebagai berikut [18] :

$$\begin{aligned}
&\min_{w^{ij},b^{ij},\xi^{ij}} \frac{1}{2}\left(w^{ij}\right)^T w^{ij} + C\sum_t \xi_t^{ij} \\
&s.t\ \left(w^{ij}\right)^T \phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij} \rightarrow y_t = i, \\
&\left(w^{ij}\right)^T \phi(x_t) + b^{ij} \geq -1 + \xi_t^{ij} \rightarrow y_t = j, \\
&\xi_t^{ij} \geq 0
\end{aligned} \tag{2.29}$$

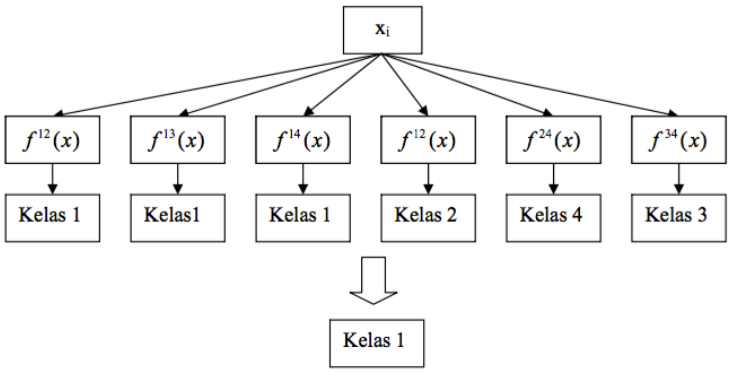
Terdapat beberapa metode untuk melakukan pengujian setelah keseluruhan k(k −1)/ 2 model klasifikasi selesai dibangun. Berikut ini adalah tabel contoh hipotesis untuk SVM *One Against One* untuk 4 kelas :

Tabel 2.2* contoh 4 kelas SVM dengan *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	$f^{34}(x) = (w^{34}x) + b^{34}$

Dibawah ini adalah contoh hasil klasifikasi SVM dengan metode *One Against One* :



Gambar 2.2* contoh klasifikasi dengan metode *One Against One

Jika data x dimasukkan ke dalam fungsi hasil pelatihan $(f(x) = (w)^T \Phi(x) + b)$ dan hasilnya menyatakan 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) [18] .

2.6 Pengujian kinerja klasifikasi

Matrix confussion merupakan tabel yang mencatat hasil kerja klasifikasi. Pada tabel berikut merupakan contoh *matrix confussion* yang melakukan klasifikasi masalah empat kelas [16] . Setiap set F_{ij} dalam matriks menyatakan jumlah record/data dari kelas i yang hasil prediksinya masuk ke kelas j. Misalnya sel F_{11} adalah jumlah data dalam kelas 1 yang secara benar dipetakan ke kelas 1. Dan F_{12} adalah jumlah data dalam kelas 1 yang dipetakan ke kelas 2.

Tabel 2.3 *Matrix Confussion*

F_{ij}		Kelas prediksi(j)			
		Kelas 1	Kelas 2	Kelas 3	Kelas 4
Kelas asli (i)	Kelas 1	F_{11}	F_{12}	F_{13}	F_{14}
	Kelas 2	F_{21}	F_{22}	F_{23}	F_{24}
	Kelas 3	F_{31}	F_{32}	F_{33}	F_{34}
	Kelas 4	F_{41}	F_{42}	F_{43}	F_{44}

Ketepatan klasifikasi dapat dilihat dari akurasi klasifikasi. Akurasi klasifikasi menunjukkan performasi model klasifikasi secara keseluruhan, dimana semakin tinggi akurasi klasifikasi hal ini semakin baik performansi model klasifikasi. Persamaan (2.30) digunakan untuk menghitung akurasi :

$$Akurasi = \frac{jumlah\ data\ yang\ diprediksi\ secara\ benar}{jumlah\ prediksi\ yang\ dilakukan} \quad (2.30)$$

2.7 Pemrograman Berorientasi Objek

Pemrograman berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis, serta didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas yang meliputi, rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek. Konsep dasar berorientasi objek diantaranya [22]:

1. Kelas (*Class*) adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dari kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas semula dapat diwariskan ke kelas yang baru.
2. Objek *(Object)* adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, strutur, status,

atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

3. Metode *(Method)* adalah operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada terstruktur. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri
4. Atribut (*Attribute*) dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.
5. Abstraksi (*Abstraction*) merupakan prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.
6. Enkapsulasi (*Encapsulation*) adalah pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerja.
7. Pewarisan (*Inheritance*) adalah mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.
8. Antarmuka (*Interface*) sangat mirip dengan kelas, tetapi tanpa atribut kelas dan tanpa memiliki metode yang dideklarasikan. Antarmuka biasanya digunakan agar kelas lain tidak langsung mengakses ke suatu kelas.
9. *Reusability* merupakan pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut
10. Generalisasi dan Spesialisasi menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil dan motor

11. Komunikasi antar objek dilakukan lewat pesan (*message*) yang dikirim dan satu objek ke objek lainnya.
12. Polimorfisme (*Polymorphism*) adalah kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.
13. *Package* adalah sebuah kontainer atau kemasan yang daoat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda

2.8 Unified Modeling Language (UML)

UML (*Unified Modeling Language*) adalah notasi yang lengkap untuk membuat visualisasi model suatu sistem. Sistem berisi informasi dan fungsi, tetapi secara normal digunakan untuk memodelkan sistem komputer. Di dalam pemodelan objek guna menyajikan sistem yang berorientasi objek kepada orang lain, akan sangat sulit di lakukan dalam bentuk kode bahasa pemrograman [19].

UML disebut sebagai bahasa pemodelan bukan metode. Bahasa pemodelan (sebagian besar grafik) merupakan notasi model yang digunakan untuk mendesain secara cepat. Bahasa pemodelan merupakan bagian terpenting dari metode. UML merupakan bahasa standar untuk penulisan blueprint software yang digunakan untuk visualisasi, spesifikasi, pembentukan dan pendokumentasian alat-alat dari sistem perangkat lunak. UML biasanya disajikan dalam diagram atau gambar yang meliputi class beserta atribut dan operasinya, serta hubungan antar kelas. UML terdiri dari banyak diagram diantaranya *use case diagram, activity diagram, class diagram* dan *sequence diagram*.

2.8.1 Use case diagram

Dalam konteks UML , tahap konseptualisasi dilakukan dengan pembuatan use case diagram yang sesungguhnya merupakan deskripsi peringkat tinggi bagaimana perangkat lunak (aplikasi) akan digunakan oleh penggunanya.

Use case diagram merupakan deskripsi lengkap tentang interaksi yang terjadi antara para aktor dengan sistem. Dalam hal ini, setiap objek yang berinteraksi dengan sistem merupakan aktor untuk sistem, sementara *use case*

merupakan deskripsi lengkap tentang bagaimana sistem berperilaku kepada aktornya. Aktor dalam *use case diagram* digambarkan sebagai ikon yang berbentuk manusia, sementara *use case* digambarkan sebagai *elips* yang berisi nama *use case* yang bersangkutan. Untuk mempermudah pemahaman, aktor biasanya dituliskan sebagai kata benda, sementara *use case* biasanya dituliskan sebagai kata kerja.

2.8.2 Activity Diagram

Activity diagram pada dasarnya menggambarkan skenario secara grafis. Serta *activity diagram* cukup serupa dengan diagram alur (*flow chart*), yang membedakan mungkin hanya *swimlane* yang menunjukkan suatu *state* berada pada objek/kelas tertentu. Keunggulan dari *activity diagram* adalah bahwa diagram tersebut lebih mudah dipahami dibanding skenario. Selain itu, dengan menggunakan *activity diagram*, kita dapat melihat dibagian manakah sistem dari suatu skenario akan berjalan.

2.8.3 Class Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment* , pewarisan, asosiasi, dan lain-lain.

2.8.4 Sequence Diagram

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan *sequence diagram* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya *sequence diagram* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada *sequence diagram* sehingga semakin banyak *use case* yang didefinisikan maka *sequence diagram* yang harus dibuat juga semakin banyak. Penomoran pesan berdasarkan urutan iteraksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu.

2.9 Python

Python adalah bahasa pemrograman komputer, sama seperti bahasa pemrograman lain, misalnya C, C++, Pascal, Java, PHP, Perl dan lain - lain. Sebagai bahasa pemrograman, Python tentu memiliki dialek, kosakata atau kata kunci, dan aturan tersendiri yang jelas berbeda dengan bahasa pemrograman lainnya [20].

Bahasa pemrograman Python disusun di akhir tahun 1980-an dan implementasinya baru dimulai pada Desember 1989 oleh Guido Van Rossum di Centrum Wiskunde & Information (CWI), sebuah pusat riset di bidang matematika dan sains, Amsterdam – Belanda; sebagai suksesor atau pengganti dari bahasa pemrograman pendahulunya, bahasa pemrograman ABC, yang juga dikembangkan di CWI oleh Leo Geurts, Lambert Meertens, dan Steven Pemberton [20]. Secara umum, para programmer banyak yang menjatuhkan pilihannya ke bahasa Python karena alasan – alasan berikut :

1. Python memiliki konsep design yang bagus dan sederhana, yang berfokus pada kemudahan dalam penggunaan. Kode Python dirancang untuk mudah dibaca, dipelajari, digunakan ulang, dan dirawat. Selain itu, Python juga mendukung pemrograman berorientasi objek dan pemrograman fungisional.
2. Python dapat meningkatkan produktivitas dan menghemat waktu bagi para programmer. Untuk memperoleh hasil program yang sama, kode Python jauh lebih sedikit dibandingkan dengan kode yang ditulis menggunakan bahasa – bahasa pemrograman lain.
3. Program yang ditulis menggunakan Python dapat dijalankan di hamper semua sistem operasi (Linux, Windows, Mac, dan lain – lain), termasuk untuk perangkat – perangkat mobile
4. Python bersifat gratis atau bebas (free) dan open-source, meskipun digunakan untuk kepentingan komersil.

2.9.1 Python 2.7

Sejak Python 2.6, komunitas Python menyertakan modul _ future_ ke dalam Python 2. Dengan menggunakan modul ini, para pengguna Python 2 masih dapat

menikmati fitur – fitur yang ada terdapat di dalam Python 3. Python 2 dilengkapi dengan tool bernama 2to3, yang digunakan untuk mengkonversi kode Python 2 ke Python 3 [19].

2.9.2 Python 3

Python 3 dikembangkan dengan tujuan jangka panjang. Fasilitas – fasilitas baru yang saat ini ditambahkan ke dalama bahasa Python hanya diimplementasikan ke dalam Python 3 dengan harapan para programmer yang masih menggunakan Python 2 secara bertahap bisa beralih ke Python 3.