BAB 2
LANDASAN TEORI

2.1 Kepribadian

Secara umum kepribadian adalah keseluruhan sikap, ekspresi, perasaan, ciri khas dan juga prilaku seseorang. Sikap perasaan ekspresi dan temperament tersebut akan terwujud dalam tindakan seseorang kalau dihadapkan kepada situasi tertentu. Setiap orang memiliki kecenderungan prilaku yang baku atau berlaku terus menerus secara konsisten dalam menghadapi situasi yang sedang dihadapi, sehingga jadi ciri khas pribadinya[1].

2.2 Pengolahan Citra

Pengolahan citra merupakan proses pengolahan dan analisis citra yang banyak melibatkan persepsi visual dengan bantuan komputer[6]. Dalam hal ini mengolah informasi yang terdapat pada suatu gambar untuk keperluan pengenalan objek secara otomatis. Ada berbagai teknik pengolahan citra tergantung kebutuhan dan keluaran yang diinginkan.

2.3 Cropping

Croping adalah memotong satu bagian dari citra sehingga diperoleh citra yang berukuran lebih kecil. Proses ini bertujuan untuk memisahkan antara objek sketsa pohon dengan background. Pemisahan tersebut dilakukan dengan cara mencari pixel-pixel terluar dari setiap sisi (atas, bawah, kiri, kanan). Pixel-pixel terluar itulah yang akan menjadi batas pemotongan, sehingga didapat citra segi empat yang siap diproses lebih lanjut[7].

2.4 Resize

Resize adalah proses yang digunakan untuk mengubah ukuran citra digital dalam piksel, baik menjadi lebih kecil atau lebih besar dari ukuran sebenarnya [8]. Metode yang digunakan adalah bukan metode khusus melainkan menggunakan perbandingan ukuran dari setiap citra dengan ukuran citra yang diinginkan[9]. Berikut rumus yang digunakan dalam proses *resize:*

1. Menghitung nilai koordinat baris

$$x = \frac{pb \times pp}{pa} \tag{2.1}$$

Keterangan :

x : Nilai piksel baris baru

pb : Ukuran panjang matriks baru

pp : Posisi piksel baris

pa : Ukuran panjang matriks lama

2. Menghitung nilai koordinat kolom

$$y = \frac{lb \times pp}{la} \tag{2.2}$$

Keterangan :

y : Nilai piksel kolom baru

lb : Ukuran lebar matriks baru

lp : Posisi piksel kolom

la : Ukuran lebar matriks lama

2.5 Grayscale

Grayscale adalah citra yang hanya memiliki 1 buah kanal sehingga citra yang ditampilkan terdiri atas warna abu-abu, bervariasi pada warna hitam pada bagian yang intensitas terlemah dan warna putih pada intensitas terkuat [10]. Rumus yang digunakan untuk mengubah citra RGB menjadi citra *grayscale* yaitu "*luma*" atau "*luminance*"[9]. Berikut adalah rumus untuk mengubah citra menjadi *grayscale*: .

$$y' = 0.299R' + 0.587G' + 0.114B' \tag{2.3}$$

Keterangan :

y' : Citra *grayscale*

R' : Komponen merah dari citra RGB

G' : Komponen hijau dari citra RGB

B' : Komponen biru dari citra RGB

2.6 *Thresholding*

Thresholding yaitu mengubah citra grayscale menjadi citra berwarna hitam putih dengan nilai ambang yang sudah ditentukan dengan persamaan *local threshold*.

$$T = \frac{Max+Min}{2} \tag{2.4}$$

Setelah nilai *threshold* sudah didapatkan, selanjutnya masuk ke persamaan dibawah ini[7].

$$f(x,y) = \begin{cases} 255, \ img(x,y) \geq \mathrm{T(x,y)} \\ 0 \ , img(x,y) < \mathrm{T(x,y)} \end{cases} \tag{2.5}$$

Keterangan :

f : fungsi yang menghasilkan nilai 0 atau 255

img : nilai *grayscale* citra

2.7 Binerisasi

Dalam pengolahan citra digital, proses binerisasi adalah mengubah citra grayscale menjadi citra biner, artinya mengubah warna tiap-tiap piksel pada citra bernilai 0 dan 255 ke dalam piksel bernilai 0 dan 1. Sehingga citra hanya berwarna hitam dan putih. Pada proses binerisasi, menggunakan nilai ambang (*threshold*) untuk menentukan nilai *grayscale* tertentu yang diubah menjadi piksel bernilai 0 atau 1. Proses thresholding akan memproses citra yang memiliki nilai dengan intensitas kurang dari nilai ambang (T) akan di set menjadi 0 dan yang lebih dari atau sama dengan nilai ambang (T) akan di set menjadi 1. Operasi nilai ambang (thresholding) dilakukan dengan persamaan 2.2 [11].

$$g(x,y) = \begin{cases} 1 \ if \ f(x,y) \ \geq \ \mathrm{T} \\ 0 \ if \ f(x,y) < \mathrm{T} \end{cases} \tag{2.6}$$

Keterangan :

$g(x,y)$: Nilai biner

$f(x,y)$: Nilai citra *grayscale* suatu piksel

T : Nilai ambang threshold

2.8 Ekstraksi Fitur Zoning

Ekstraksi Ciri *Zoning* merupakan salah satu metode ekstaksi ciri dimana inti dari metode ini adalah citra dibagi pada sejumlah zona yang sama untuk dikenali

ciri dari setiap citra yang selanjutnya menghasilkan sebuah nilai untuk diproses pada tahapan klasifikasi [12]. Berikut merupakan tahapan dalam proses ekstraksi ciri Zoning (Image Centroid Zone).

1. Hitung centroid dari citra masukkan menggunakan persamaan berikut.

$$x_c = \frac{(i_1.p_1+i_2.p_2+\cdots+i_n.p_n)}{(p_1+p_2+\cdots+p_n)} \quad (2.7)$$

$$y_c = \frac{(j_1.p_1+j_2.p_2+\cdots+j_n.p_n)}{(p_1+p_2+\cdots+p_n)} \quad (2.8)$$

Keterangan :

x_c : *centroid* koordinat *x*

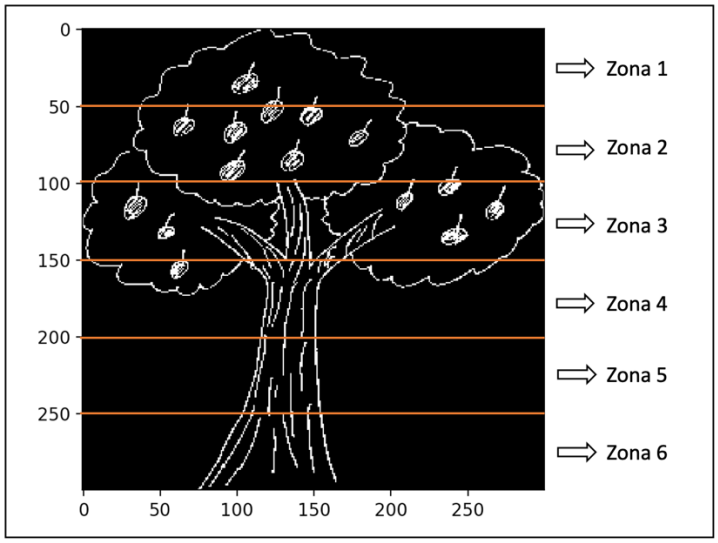
y_c : *centroid* koordinat *y*

x_n : koordinat *x* dari piksel ke-*n*

y_n : koordinat *y* dari piksel ke-*n*

y_c : nilai piksel ke-*n*

2. Bagi citra masukkan ke dalam *n* zona yang sama. Contoh pembagian zona pada citra biner dapat dilihat pada gambar 2.1



Gambar 2.1 Pembagian Zona Citra Biner

3. Hitung jarak antar centroid citra dengan masing-masing pixel yang ada dalam zona.

$$jarak = \sqrt{(x_p - x_c)^2 + (y_p - y_c)^2} \quad (2.9)$$

Keterangan:

Jarak = jarak antara koordinat centroid (*x,y*) dengan koordinat objek

x_p = koordinat x objek

y_p = koordinat y objek

x_c = *centroid* koordinat *x*

y_c = *centroid* koordinat *y*

4. Ulangi langkah ke 3 untuk setiap piksel yang ada di zona
5. Hitung rata-rata jarak antara titik-titik tersebut.
6. Ulangi langkah-langkah tersebut untuk keseluruhan zona,
7. Hasilnya adalah *n* fitur yang akan digunakan dalam klasifikasi dan pengenalan.

2.9 Smooth Support Vector Machine (SSVM)

SSVM adalah pengembangan SVM dengan menggunakan teknik *smoothing*. Metode ini pertama kali diperkenalkan oleh Lee pada tahun 2001[4][13]. SVM memanfaatkan optimasi dengan quadratic programming, sehingga untuk data berdimensi tinggi dan data jumlah besar SVM menjadi kurang efisien. Oleh karena itu dikembangkan smoothing technique yang menggantikan plus function SVM dengan integral dari fungsi sigmoid neural network yang selanjutnya dikenal dengan *Smooth Support Vector Machine* (SSVM).

Keterangan :

w^b : Seukuran Inputan

b^0 : Angka

D : Matrik Diagonal berukuran (m x m) dengan nilai [1, -1] ‘

A : Matrik berukuran (m x n)

m : Banyaknya data

n : fitur

w : vektor normal berukuran (n x 1)

e : vektor berukuran (m x 1)

γ : Parameter penentu lokasi bidang pemisah terhadap titik asal

v : Parameter positif yang menyeimbangkan bobot dari training error dan margin maximation term

Diberikan masalah klasifikasi dari *n* objek dalam ruang dimensi R^p sehingga susunan data berupa matriks *A* berukuran *n x p* dan keanggotaan tiap titik terhadap

kelas {+1} atau {-1} yang didefinisikan pada diagonal matriks D berukuran ,problem optimasinya adalah

$$min_{w,b,\xi}\ \frac{c}{2} + \xi'\xi + \frac{1}{2}(w'w + b^2) \tag{2.10}$$

Dengan kendala

$$D(Aw + eb) + \xi \geq e, \xi \geq 0 \tag{2.11}$$

Solusi problem adalah

$$\xi = (e - D(Aw + eb)) \tag{2.12}$$

Dimana ξ adalah variabel *slack* yang mengukur kesalahan klasifikasi. Kemudian dilakukan substitusi dan konversi, sehingga persamaan dapat ditulis sebagai berikut:

$$min_{w,b}\ \frac{c}{2}||\big(e - D(Aw + eb)\big)||_2^2\ + \frac{1}{2}(w'w + b^2) \tag{2.13}$$

Fungsi objektif dalam persamaan tidak memiliki turunan kedua. Teknik *smoothing* yang diusulkan dilakukan dengan mengganti fungsi plus dengan p(x,a) yaitu integral dari fungsi sigmoid *neural network* atau dapat dituliskan sebagai berikut:

$$p(x, a) = x\frac{1}{a}\log(1 + \varepsilon^{-ax}),\ a > 0 \tag{2.14}$$

dimana α adalah parameter *smoothing*. Dengan menggantikan fungsi plus dengan p (x,a) maka diperoleh model SSVM sebagai berikut:

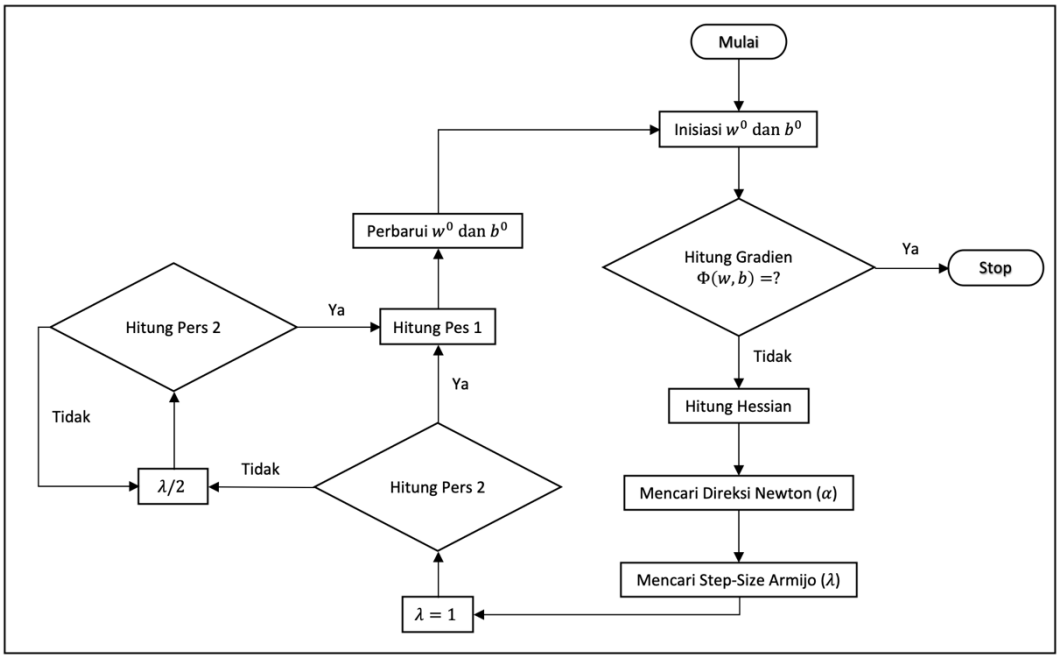
$$min_{(w,b)\epsilon Rp+1}\phi a(w, b)$$
$$= min_{(w,b)\epsilon Rp+1}\frac{c}{2}||p\big(e - D(Aw - eb)\big)||_2^2\ + \frac{1}{2}(w'w + b^2) \tag{2.15}$$

Secara umum, problem optimasi SSVM dapat ditulis sebagai berikut:

$min_{(w,b)\epsilon Rp+1}\phi a(w, b)$

$$= min_{(w,b)\epsilon Rp+1}\frac{c}{2}||p\big(K\big(x_i x_j\big)Dw - D(Aw - eb), a\big)||_2^2\ + \frac{1}{2}(w'w + b^2) \tag{2.16}$$

Yang diselesaikan dengan iterasi Newton Armijo (Gambar 2.2) dan *K(Xi,Xj)* merupakan fungsi kernel yang dalam penelitian ini digunakan kernel Gaussian atau bisa dirumuskan berikut *K(Xi,Xj) = exp (-y(*||Xi – Xj||^2) dengan parameter kernel *y*.



Gambar 2.2 Diagram Alir Algoritma Newton-Armijo

Persamaan 1 :

$$\phi_a(w_i b_i) - \phi_a\big((w_i b_i) + (\lambda_i d_i)\big) \geq -\delta\lambda_i \nabla\phi_a(w_i b_i) d_i \tag{2.17}$$

Persamaan 2 :

$$w_{i+1} b_{i+1} = (w_i b_i) + (\lambda_i d_i) \tag{2.18}$$

Perumusan program linier SVM 1-norm adalah salah satu cara untuk memilih atribut (*feature selection*) diantara varian-varian norm SVM, problem linier tersebut adalah sebagai berikut:

$$\min_{(w,b,s,\xi)\in R^{(2p)+1+n}} Ce'\xi + e's \tag{2.19}$$

Dengan kendala

$$\begin{aligned} D(Aw + eb) + \xi &\geq e \\ -s \leq w &\leq s \\ \xi &\geq 0 \end{aligned} \tag{2.20}$$

Solusi dari w mampu menghasilkan model yang parsimoni dan bersifat *sparsity*. Jika nilai dari elemen vektor wp = 0, maka variabel *p* tidak berkontribusi dalam penentuan kelas. Kontribusi atribut atau variabel prediktor dapat dinilai dari besarnya nilai w*i* untuk masing-masing atribut, dengan *l*=1,2, ..., *p*.

Support Vector Machine (SVM) adalah suatu sistem pembelajaran yang menggunakan ruang hipotesis dari suatu fungsi linear dalam suatu ruang dimensi

berfitur tinggi yang dikembangkan oleh Boser, Guyon, Vapnik, dan pertama kali dipresentasikan pada tahun 1992 di *Annual Workshop on Computational Learning Theory*.

Pada pendekatan *smoothing* yang digagas oleh Lee dan Mangasarian [4], kuadrat 2-*norm* dari vektor variabel *slack* ξ diminimalkan dengan bobot v / 2 menggantikan 1-*norm* dari vektor variabel *slack* ξ sehingga hasil modifikasi dari fungsi objektif SVM linier standar adalah sebagai berikut.

$$\min_{(w,b,s,\xi)\in R^{(2p)+1+m}} \frac{v}{2}\xi'\xi + \frac{1}{2}(w^T w + \gamma^2)$$

$$s.t\ D(Aw - ey) + \xi \geq e, \xi \geq 0 \tag{2.21}$$

Kendala pada Persamaan (2.19) dapat ditulis menjadi.

$$\xi = (e - D(Aw - ey))_+ \tag{2.22}$$

di mana dalam hal ini $x_+ = \max\{0,\ x\}$. Bila Persamaan (2.21) disubtistusikan ke dalam persamaan (2.22) maka diperoleh fungsi objektif bebas kendala, yaitu.

$$\min_{(w,y)\in R^{n+1}} \frac{v}{2}||(e - D(Aw - ey))_+||_2^2 + \frac{1}{2}(w^T w + \gamma^2) \tag{2.23}$$

Bila dibandingkan dengan fungsi objektif (2.22), fungsi objektif (2.23) mereduksi dimensi dari yang semula $n + 1 + m$ menjadi $n + 1$. Namun fungsi objektif ini tidak memiliki turunan kedua sehingga metode optimasi konvensional tidak bisa digunakan. Oleh karena itu, Lee dan Mangasarian mengusulkan *smoothing technique*, yaitu sebuah teknik di mana *plus function* x_+ pada fungsi objektif (2.23) didekati dengan sebuah fungsi *smooth-p* yang merupakan integral dari fungsi *signoid neural network* $1 + \exp(-\alpha x))^{-1}$ atau dapat ditulis sebagai berikut.

$$p(xa) = x + \frac{1}{a} ln(1 + \exp(-ax)) \tag{2.24}$$

dengan $\alpha > 0$ adalah *smooth parameter* yang mengontrol kedekatan kurva $p(x,\ \alpha)$ terhadap kurva *plus function* x_+. Bila *plus function* x_+, pada Persamaan (2.22) diganti dengan $p(x, \alpha)$ maka diperoleh formulasi SSVM berikut.

$$\min_{(w,\gamma,\xi)\in R^{n+1}}$$

$$= \min_{(w,y)\in R^{n+1}} \frac{v}{2}\left\|p\big(e - D(Aw - ey)\big)a\right\|_2^2 + \frac{1}{2}(w^T w + \gamma^2) \tag{2.25}$$

Fungsi objektif SSVM (2.22) memiliki turunan kedua sehingga problem optimasi dapat dilakukan dengan menerapkan algoritma konvergen kuadrat Newton dengan tahapan Armijo yang membuat algoritma tersebut konvergen secara global. Algoritma *Newton-Armijo* dimulai dengan menetapkan nilai awal ($w^0, \gamma^0 \in R^{n+1}$). Proses akan berhenti apabila gradien fungsi objektif SSVM, yaitu $\nabla\Psi_\alpha(w^i, \gamma^i) = 0$. Selain itu, (w^{i+1}, γ^{i+1}) akan dihitung dengan langkah–langkah berikut.

1. *Newton Direction*

   Menentukan *direction* $d^i \in R^{n+1}$ dengan menetapkan linierisasi sama dengan nol dari $\nabla\Psi_\alpha(w, \gamma)$ di sekitar (w^i, γ^i) yang memberikan $n+1$ persamaan linier dengan $n+1$ variabel.

   $$\nabla^2\Psi_\alpha(w^i, \gamma^i)d^i = -\nabla\Psi_\alpha(w^i, \gamma^i)' \tag{2.26}$$

2. *Armijo Stepsize*

   Memilih sebuah *stepsize* $\lambda_i \in R$ sedemikian sehingga

   $$(w^{i+1}, \gamma^{i+1}) = (w^i, \gamma^i) + \lambda_i d^i \tag{2.27}$$

   dengan $\delta \in (0, \frac{1}{2})$.

   Saat $\Psi_\alpha(w^i, \gamma^i) = 0$, iterasi pada Algoritma *Newton-Armijo* berhenti diperoleh nilai w dan γ yang konvergen. Pada proses optimasi ini, matriks Hessian diberikan oleh Lee [4],

   $$\lim_{\alpha\to\infty} \nabla^2\Psi_\alpha(w, y) = \begin{bmatrix} \frac{\partial^2\Psi_a(x,\gamma)}{\partial^2 w^2} & \frac{\partial^2\Psi_a(x,\gamma)}{\partial w\partial\gamma} \\ \frac{\partial^2\Psi_a(x,\gamma)}{\partial^2\partial w} & \frac{\partial^2\Psi_a(x,\gamma)}{\partial^2\gamma^2} \end{bmatrix}$$
   $$= I + v\begin{bmatrix} H_{1,1} & H_{1,2} \\ H_{2,1} & H_{2,2} \end{bmatrix} \tag{2.28}$$

   Dimana

   $$H_{1,1} = A^T diag\left(S_\infty(e - D(Aw - e\gamma))\right)A$$

   $$H_{1,2} = -A^T diag\left(S_\infty(e - D(Aw - e\gamma))\right)e$$

   $$H_{2,1} = -e^T diag\left(S_\infty(e - D(Aw - e\gamma))\right)A$$

   $$H_{2,2} = Ae^T diag\left(S_\infty(e - D(Aw - e\gamma))\right)e$$

   Dan

$$S_{\infty}(x) = \lim_{\alpha \to \infty} \frac{1}{1+\varepsilon^{-ax}} = \frac{1+sign(x)}{2} \tag{2.29}$$

Sementara itu, gradien dari matriks Hessian memiliki formula sebagai berikut.

$$\lim_{\alpha \to \infty} \nabla^2 \Psi_{\alpha}(w, y) = \begin{bmatrix} w - vA^T D\big(e - D(Aw - ey)\big) \\ y + ve^T D\big(e - D(Aw - ey)\big) \end{bmatrix} \tag{2.30}$$

Nilai w dan γ optimum kemudian membentuk *classifier* dari SSVM Linier yang memiliki formulasi.

$$f(x) = sign(g(x)) \tag{2.31}$$

Dengan

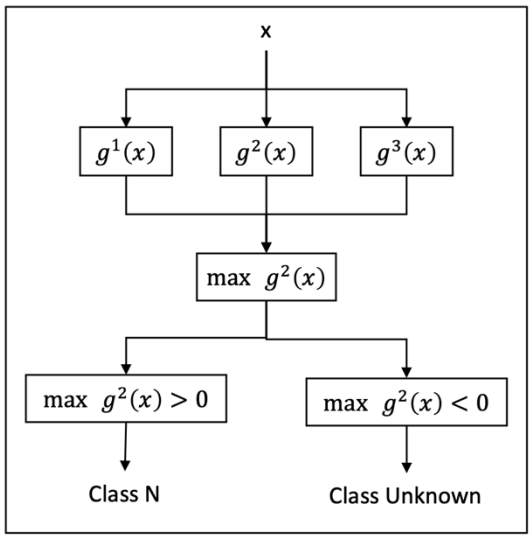
$$g(x) = x^T w - \gamma \tag{2.32}$$

2.10 Multiclass One-Againts-All

Multiclass merupakan gabungan dari beberapa SSVM biner atau menggabungkan semua data yang terdiri dari beberapa kelas kedalam sebuah bentuk permasalahan optimasi[14]. Metode *One-Againts-All* akan di bagun sejumlah c SSVM biner, dengan c merupakan jumlah *class*[15]. Untuk lebih jelasnya perhatikan ilustrasi pada Tabel 2.1 dan Gambar 2.3.

Tabel 2.1 Kombinasi Biner

$y_i = 1$	$y_i = -1$	Hyperplane
Class_1	Bukan class_1	$g^1(x) = x^T.w^1 - y^1$
Class_2	Bukan class_2	$g^2(x) = x^T.w^2 - y^2$
Class_3	Bukan class_3	$g^3(x) = x^T.w^3 - y^3$



Gambar 2.3 Klasifikasi Metode One-Againts-All

2.11 *Confusion Matriks*

Confusion Matrix merupakan sebuah tabel yang menggambar jumlah data *testing* yang terklasifikasi dengan benar dan jumlah data *testing* yang terklasifikasi dengan salah. *Confusion Matrix* juga adalah matriks yang didalamnya terdapat jumlah dari *True Positive, True Negative, False Positive,* dan *False Negative*[16]. TP dan TN merupakan hasil klasifikasi yang benar sedangkan FP dan FN merupakan hasil klasifikasi yang salah. Gambar 2.4 adalah confusion matrix dalam kasus multiclass dengan kelas 1 sebagai positif dan kelas 2 hingga kelas.

		Predicted class	
		P	*N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Gambar 2.4 Confusion Matriks

1. Akurasi merupakan persentase dari total hasil prediksi yang bernilai true (benar). Berikut adalah persamaan untuk menghitung nilai akurasi pada suatu kelas.

$$Akurasi_y = \frac{TP}{TP + TN} \text{ x } 100\% \quad (2.33)$$

Dimana y: kelas yang diprediksi

2. *Precision* adalah nilai ketepatan dari hasil suatu model. Persamaannya menggunakan perbandingan antara hasil *true positive* dengan total data dengan label *positive*. Adapun untuk perhitungan *precision* menggunakan persamaan berikut.

$$Precision = \frac{TP}{TP + FP} \quad (2.34)$$

3. *Recall* adalah nilai kelengkapan dari sebuah model. Persamaan *recall* menggunakan perbandingan antara *true positive* terhadap total contoh yang benar-benar *positive*. Adapun untuk perhitungan *recall* menggunakan persamaan berikut [16]

$$Recall = \frac{TP}{TP + FN} \quad (2.35)$$

4. *F-measure* merupakan perhitungan untuk mencari nilai *harmonic mean* dari *precision* dan *recall,* Adapun untuk perhitungan *recall* menggunakan persamaan berikut [16].

$$F1\ Score = \frac{2 * precision * recall}{precision + recall} \quad (2.36)$$

Jika dalam suatu pengujian jumlah kelas yang diklasifikasikan memiliki lebih dari dua kelas, maka rumus untuk menghitung *presicion*, dan *recall* adalah sebagai berikut [16]

$$Precision_\mu = \frac{\sum_i^L \frac{TP_i}{TP_i + FP_i}}{L} \ X\ 100\% \quad (2.37)$$

$$Recall_\mu = \frac{\sum_i^L \frac{TP_i}{TP_i + FN_i}}{L} \ X\ 100\% \quad (2.38)$$

$$F1 - score_{\mu} = \frac{\sum_{i}^{L} \frac{2 * precision * recall}{precision + recall}}{L} \; X \; 100\% \quad (2.39)$$

Keterangan

L : Jumlah Keseluruhan Data Yang Diuji/Jumlah Kelas

Jumlah kelas yang digunakan pada penelitian ini lebih dari dua. Maka untuk mencari rata-rata dari setiap *classifier* menggunakan persamaan sebagai berikut:

$$Akurasi_{\mu} = \frac{\sum_{i=1}^{n} x_i}{n} \; x \; 100\% \quad (2.40)$$

Keterangan

x_i: Nilai akurasi dari *classifier* ke - i

n: Jumlah total yang data *testing*

2.12 Bahasa Pemrograman Python

Python adalah Bahasa pemrograman bersifat umum. Python diciptakan pada tahun 1990 oleh Guido van Rossum. Bahasa level tinggi, stabil, dinamis, orientasi objek dan cross *platform*. Bahasa pemrograman Python saat ini dikembangkan dan dikelola oleh suatu tim relawan dengan nama *Python Software Foundation*[17]. Pada penelitian ini menggunakan Bahasa pemrograman python dalam pembangunan aplikasi yang berbasis web.

2.12.1 Jupyter Notebook

Jupyter Notebook (file yang berekstensi *ipynb*) adalah dokumen yang dihasilkan oleh *Jupyter Notebook App* yang berisikan kode komputer dan *rich text element* seperti paragraf, persamaan matematik, gambar dan tautan (*links*). *Jupyter Notebook* dikenal sebelumnya sebagai *IPython Notebook* dan dalam waktu dekat akan berevolusi menjadi *Jupyter Lab*[18]. Pada penelitian ini jupyter notebook digunakan untuk membangun rancangan code program berbentuk *console*.

2.12.2 Jetbrains Pycharm

Pycharm merupakan IDE yang cukup populer dikalangan developer Python. PyCharm sendiri memiliki dua versi yaitu Professional Edition dan Community Edition. PyCharm Professional Edition merupakan versi berbayar dari PyCharm dan Community Edition merupakan versi gratis yang tersedia bagi komunitas python dengan lisensi Apache 2. Pada penelitian ini *Pycharm* digunakan sebagai *tool* dalam membangun aplikasi[19]

2.12.3 PyQt

PyQt adalah sebuah *library* pada *python* untuk membangun *Graphic User Interface* (GUI) dengan lebih mudah. Di dalam PyQt sudah terdapat paket bernama QtDesigner untuk mempermudah pembangunan GUI hanya dengan *drag & drop* desain saja. PyQt sendiri sudah memiliki lebih dari 35 ekstensi modul yang siap digunakan dan mampu membuat *Python* sebagai bahasa alternatif dari C++ [20]

2.12.4 OpenCV

OpenCV adalah sebuah *library* yang mengkhususkan dirinya untuk pembangunan *computer vision* dan *machine learning*. OpenCV sendiri sudah memiliki sangat banyak algoritma yang teroptimasi dengan jumlah lebih dari 2500. *Library* OpenCV sendiri tidak hanya ada untuk bahasa pemrograman *Python* saja, melainkan ada pula untuk bahasa lain seperti C++, Java, dan MATLAB Interfaces dan sudah sangat *support* penggunaannya pada operasi sistem Windows, Linux, Mac OS dan Android [21]

2.12.5 Numpy

Numpy adalah sebuah *library* pada *Python* yang berguna untuk melakukan perhitungan *scientific*. Di dalamnya terdapat paket-paket untuk melakukan operasi terhadap objek array yang berupa matriks dengan N-dimensi, aljabar linear, dan banyak [22]

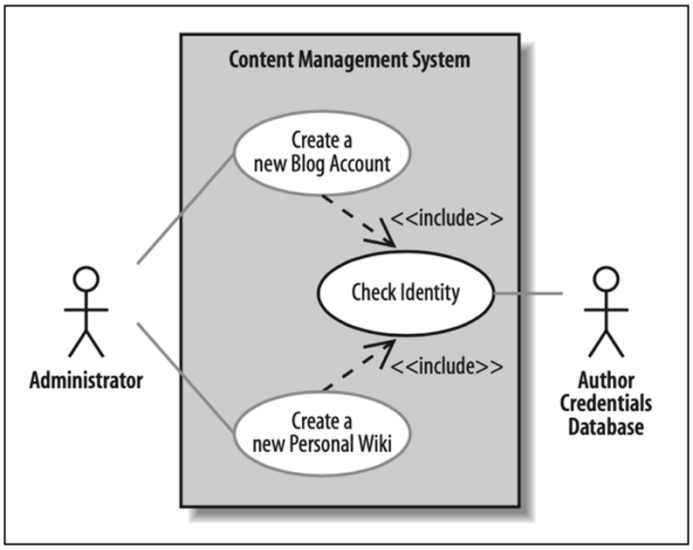
2.13 Unified Modeling Language (UML)

Unified Modeling Language (UML) merupakan notasi yang lengkap untuk membuat visualisasi model suatu sistem. UML disebut sebagai Bahasa pemodelan bukan metode. UML biasanya disajikan dalam bentuk diagram atau gambar yang meliputi *class* beserta atribut dan operasinya, serta hubungan antara kelas. UML

terdiri dari banyak diagram diantaranya *use case diagram, use case scenario, activity diagram, class diagram,* dan *sequence diagram[23].*

2.13.1 Use Case Diagram

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 berprilaku kepada aktornya[23]. Aktor dalam *use case diagram* digambarkan debagai ikon yang berbentuk manusia yang biasanya dituliskan sebagai kata benda, sementara *use case* digambarkan sebagai ikon yang berbentuk elips yang berisi nama *use case* yang bersangkutan, biasanya dituliskan kata kerja untuk mempermudah pemahaman. Berikut contoh *use case diagram* pada Gambar 2.5.



Gambar 2.5 Use Case Diagram

2.13.2 Use Case Scenario

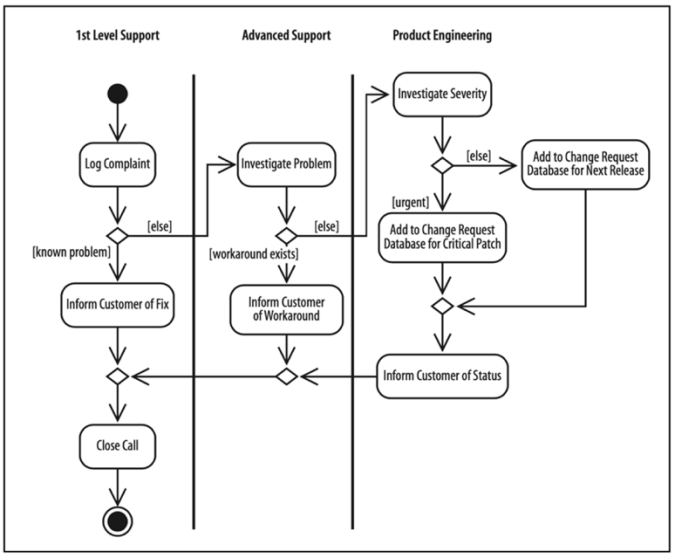
Use case scenario merupakan penjelasan secara tekstual dari sekumpula skenario interaksi. Setiap skenario yang ada akan mendeskripsikan urutan aksi atau langkah yang dilakukan aktor ketika berinteraksi dengan sistem, baik dalam kondisi berhasil atau gagal[23]. Pada penelitian ini digunakan format yang diperkenalkan oleh Rules Miles [20] dengan bagian yang terlibat sebagai berikut :

1. Nama Use Case (*Use case name*)
2. Persyaratan Terkait (*Related Requirements*), kondisi spesifik yang harus terpenuhi sebelum use-case dieksekusi oleh aktor.

3. Tujuan (*Goal in Context*), penggunaan *use case* dan menjelaskan mengapa *use case* ini menjadi penting digunakan.
4. Komdisi Awal (*Precondition*), kondisi awal sebelum *use cse* dieksekusi.
5. Kondisi Akhir Berhasil (*Successful End Condition*), kondisi ketika *use cse* berhasil dieksekusi.
6. Kondisi Akhir Gagal (*Failed End Condition*), kondisi ketika *use case* gagal dieksekusi.
7. Aktor Utama (*Primary Actors*), Aktor utama yang menggunakan *use case*.
8. Aktor Sekunder (*Secondary Actors*), Aktor lainnta atau sekunder yang mengguanan *use case*.
9. Pemicu (*Trigger*), pemicu oelh aktor sehingga *use case* dieksekusi.
10. Alur Utama (*Main Flow*), penjelasan terkait alur utama dari *use case* apabila berjalan secara normal.
11. Ekstensi (*Ekstensions*), yaitu jalur alternatif dari interaksi yang terjadi antara aktor dari sistem yang mencakup percabangan (pilihan) maupun skenario yang gagal sehingga tujuan aktor tidak terpenuhi.

2.13.3 Activity Diagram

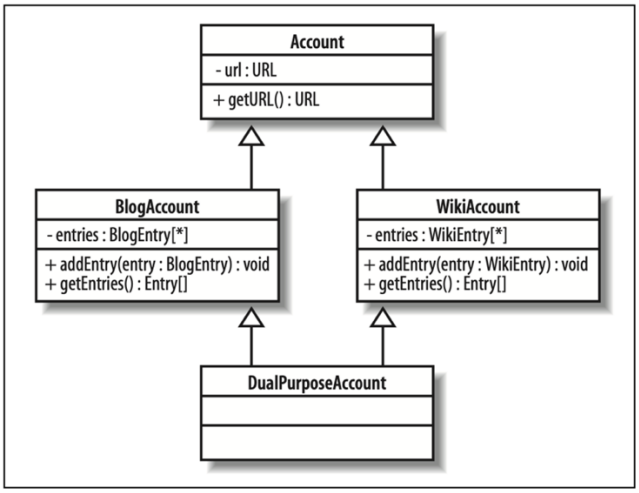
Activity diagram adalah diagram *flowchart* yang diperluas yang menunjukkan aliran kendali satu aktivitas ke aktivitas lain berdasarkan use case diagram[23]. Berikut contoh dari *activity diagram* pada Gambar 2.6.



Gambar 2.6 Activity Diagram

2.13.4 Class Diagram

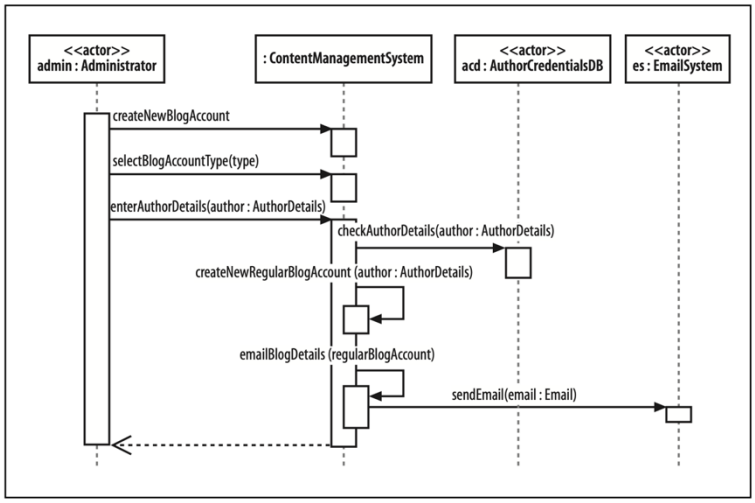
Class diagram merupakan inti dari setiap sistem berorientasi objek, oleh karena itu kaitannya sangat erat dengan *use case diagram*. *Class diagram* bertujuan untuk menjelaskan berbagai jenis objek dan hubungannya dengan kelas lainnya yang dimiliki oleh sistem. Hal inilah yang menjadikan *class diagram* memiliki dua macam informasi, yaitu informasi tentang kondisi awal objek dan bagaimana ia berprilaku dalam lingkungannya[23]. Berikut contoh dari *class diagram* pada Gambar 2.7.



Gambar 2.7 Class Diagram

2.13.5 Sequence Diagram

S*equence diagram* atau dikenal juga sebagai diagram interaksi bertujuan untuk memodelkan interaksi secara *runtime* antara bagian-bagian tertentu pada sistem yang membentuk alur perpindahan sebuah objek. Diagram ini akan menyampaikan urutan dari setiap interaksi pada suatu proses atau bagian-bagian sistem. Dengan menampilkan apa yang menjadi pemicu dari sebuah proses dan menunjukan informasi lain berupa peristiwa dalam suatu interaksi[23]. Berikut contoh dari *sequence diagram* pada Gambar 2.8.



Gambar 2.8 Sequence Diagram

2.14 Pengujian Sistem

Pengujian sistem adalah proses pemeriksaan atau evaluasi komponen sistem untuk memverifikasi apakah sistem memenuhi kebutuhan antara hasil yang diterapkan dengan hasil yang terjadi[24].

Pengujian seharusnya meliputi tiga konsep berikut .

1. Demonstrasi validasi perangkat lunak pada masing-masing tahap diskusi pengembangan sistem.
2. Penentuan validitas sistem akhir dikaitkan dengan kebutuhan pemakai.
3. Pemeriksaanprilakusistemdenganmengeksekusisistempadadatasample pengujian.

2.14.1 Pengujian Black Box

Konsep *black box* digunakan untuk mempresentasikan sistem yang cara kerja di dalamnya tidak tersedia untuk diinspeksi. Dalam *black box,* item-item yang diuji dianggap “gelap” karena logiknya tidak diketahui, yang diketahui hanya apa yang masuk dana pa yang keluar dari *black box[24]*.