BAB 2
LANDASAN TEORI

2.1 Grafologi

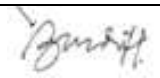
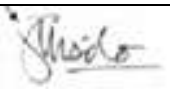
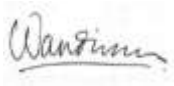
Grafologi berasal dari bahasa Yunani. Grafologi terdiri dari dua kata yaitu ‘graph’ artinya tulisan, dan ‘logos’ yang berarti ilmu. Menurut Kamus Besar Bahasa Indonesia,Grafologi adalah ilmu tentang aksara atau sistem tulisan, ilmu surat tangan,ilmu tentang hubungan antara watak dan tulisan tangan (rajah).

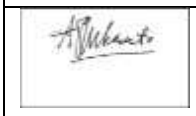
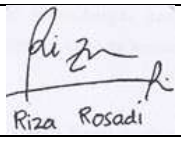
Grafologi merupakan seni dan ilmu yang mempelajari tentang tulisan tangan.Tulisan merupakan salah satu hasil karya manusia yang unik.Itulah sebabnya,antara satu manusia dengan manusia lainnya tidak memiliki tulisan yang sama. Grafologi tulisan tangan terbagi menjadi dua , yang pertama tulisan tangan berbentuk paraparagraph yang kedua tulisan tangan berbentuk tanda tangan [1]

Dalam grafologi tulisan tangan berbentuk tanda tangan dapat melakukan prediksi kepribadian, menurut Margaret Gullan-Whur disebutkan ada beberapa aspek yang mesti dicermati dalam menganalisis tanda tangan, mulai dari fitur, penempatan ukuran keseimbangan,koreksi dan dekorasi (unsur tambahan), pada kasus ini akan digunakan fitur tanda tangan.

Fitur-fitur tanda tangan yang terdapat pada [2] dapat dilihat pada tabel 2.1 berikut :

Tabel 2. 1 Fitur - fitur Tanda Tangan

No	Gambar	Fitur	Kepribadian
1		Awal Kurva Lengkung tertutup	Ketakutan berlebihan, introvert,tidak memperdulikan sekitar,tidak suka bergaul dan bekerja sama.
2		Awal Kurva Lengkung mundur	Nyaman akan masa lalu.
		Awal Kurva Lengkung tajam	Mampu memformulasi pikiran secara tajam.
		Awal Kurva Lengkung lembut	Hati-hati,ramah, diplomatis.
3		Coretan Akhir Menaik	Terbuka, pandangan ke depan, keinginan maju, percaya diri.
		Coretan Akhir Menurun	Kurang semangat, berfikir realistis, kurang percaya diri,mudah putus asa.
4		Ada Coretan Tengah	Kurang percaya diri dan mudah depresi.
5		Adanya garis bawah	Membutuhkan dukungan Membuat keputusan, serta memiliki keandalan dalam memimpin.
6		Cenderung ke kanan	Ceroboh,kurang perhatian

		Cenderung ke kiri	Takut gagal,takut pada orang lain,kurang percaya diri,pesimis
		Cenderung di atas	Respek pada diri sendiri,mencerminkan pribadi bahagia.
		Cenderung ke bawah	Depresi,pemalu,merasa asing
7		Ada titik	Pendirian stabil,memiliki rasa curiga,selalu menjaga jarak tidak mudah percaya.
8		Ada tanda tangan terpisah	Memiliki pengalaman kurang menyenangkan di masa lalu.
9		Ada garis terpisah	Membatasi keinginannya,tidak berani mengambil resiko,seringpatah semangat dan ragu mengambil keputusan.

Penelitian ini menggunakan 7 fitur yaitu lengkung mundur, lengkung tajam,lengkung lembut , adanya coretan ditengah,adanya garis bawah, coretan akhir menaik dan coretan akhir menurun. Terdiri dari 10 kelas lengkung mundur, lengkung tajam, lengkung lembut, coretan akhir menaik, coretan akhir menurun, tidak adanya coretan akhir, adanya coretan ditengah, tidak adanya ditengah, adanya garis bawah, tidak adanya garis bawah.

2.2 Citra Digital

Secara umum, istilah pengolahan citra digital menyatakan "pemrosesan gambar berdimensi-dua melalui komputer digital. Foto adalah contoh gambar berdimensi dua yang dapat diolah dengan mudah. Setiap foto dalam bentuk citra digital (misalnya berasal dari kamera digital) dapat diolah melalui perangkat tertentu. Sebagai contoh, apabila hasil bidikan kamera terlihat agak gelap, citra dapat diolah menjadi lebih terang dimungkinkan pula untuk memisahkan foto orang dari latar belakangnya. Gambaran tersebut menunjukkan hal sederhana yang dapat dilakukan melalui pengolahan citra digital. Tentu saja banyak hal pelik lain yang dapat dilakukan melalui pengolahan citra digital. [11]

2.3 Jenis Citra

Jenis citra dibagi menjadi tiga jenis citra yang umum digunakan untuk pemroresan citra. Ketiga citra tersebut yaitu citra berwarna, citra berskala keabuan, dan citra biner.

2.3.1 Citra Berwarna

Citra berwarna, atau biasa dinamakan Citra RGB, merupakan jenis citra yang menyajikan warna dalam bentuk komponen R(merah), G(hijau), dan B(biru). Tiap komponen warna menggunakan 8 bit (nilainya berkisar antara 0 sampai dengan 255). Dengan demikian, kemungkinan warna yang dapat disajikan mencapai 255 x 255 x 255 atau 16.581.375 warna. Tabel 2.2 menunjukkan contoh warna R, G dan B. [11]

Warna	R	G	B
Merah	255	0	0
Hijau	0	255	0
Biru	0	0	255
Hitam	0	0	0
Putih	255	255	255
Kuning	0	255	255

2.3.2 Citra Berskala Keabuan

Citra berskala keabuan menangani gradasi warna hitam dan putih, yang menghasilkan efek warna abu-abu. Pada jenis gambar ini, warna dinyatakan dengan intensitas. Dalam hal ini, intensitas berkisar antara 0 sampai dengan 255. Nilai 0 menyatakan hitam dan nilai 255 menyatakan putih. [11]

2.3.3 Citra Biner

Citra biner adalah citra dengan setiap piksel hanya dinyatakan dengan sebuah nilai dari dua kemungkinan yaitu nilai 0 dan 1. Nilai 0 menyatakan warna hitam dan nilai 1 menyatakan warna putih. Citra jenis ini banyak dipakai dalam pemroresan citra, misalnya untuk kepentingan memperoleh tepi objek. [11]

2.4 Praproses

Tahap praproses merupakan penjelasan tahap awal dalam alur kerja untuk menyiapkan citra sebagai data input yang siap diolah lebih lajut oleh sistem. Proses pada pra-proses yang digunakan pada penelitian ini sebagai berikut:

2.4.1 *Grayscale*

Citra grayscale atau citra berskala keabuan. Berguna untuk merubah citra berwarna menjadi citra berskala keabuan. Secara umum perubahan citra berwarna menjadi citra keabuan dengan menggunakan rumus:

$$\boldsymbol{I = a * R + b * G + c * B}$$

Dimana (R) adalah nilai pada warna merah, (G) adalah nilai pada warna hijau, dan (B) adalah ilai pada warna biru. Sementara untuk (a, b, c) adalah nilai mutlak, nilai mutlak yang biasa dipakai untuk (a, b, c) adalah:

$$a = 0.2989$$

$$b = 0.5870$$

$$c = 0.1141$$

Sehingga didapatkan rumus untuk merubah citra berwarna menjadi citra berskala keabuan adalah sebagai berikut [11].

$$I = 0.2989 * R + 0.5870 * G + 0.1141 * B \quad (2.1)$$

2.4.2 Deteksi Tepi Canny

Deteksi tepi berfungsi untuk memperoleh tepi objek. Operator Canny dikemukakan oleh John F. Canny pada tahun 1986, terkenal sebagai operator deteksi tepi yang paling optimal. Algoritma ini memberikan tingkat kesalahan yang rendah, melokalisasi titik-titik tepi (jarak piksel-piksel tepi yang ditemukan deteksi dan tepi yang sesungguhnya sangat pendek), dan hanya memberikan satu tanggapan untuk satu tepi. [11]

Terdapat langkah-langkah yang digunakan untuk melakukan deteksi tepi canny. Langkah-langkah tersebut sebagai berikut.

1. *Image Smoothing*

   Merupakan sebuah proses untuk mengaburkan gambar untuk menghilangkan derau. Pada proses ini dapat dilakukan dengan menggunakan filter gausian dapat dilihat pada persamaan 2.2 sebagai berikut [12].

$$\frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \quad (2.2)$$

2. *Finding Gradient*

Merupakan sebuah proses untuk mendapatkan kekuatan tepi. Tepian harus ditandai pada gambar memiliki gradien yang besar. Operator yang digunakan untuk menentukan gradien dapat menerapkan dengan operator sobel. Operator sobel dapat dilihat pada persamaan 2.3 sebagai berikut.

$M_{x=}$ | -1 | 0 | 1 |
|---|---|---|
| -2 | 0 | 2 |
| -1 | 0 | 1 |

$M_y =$ | 1 | 2 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| -1 | -2 | -1 |

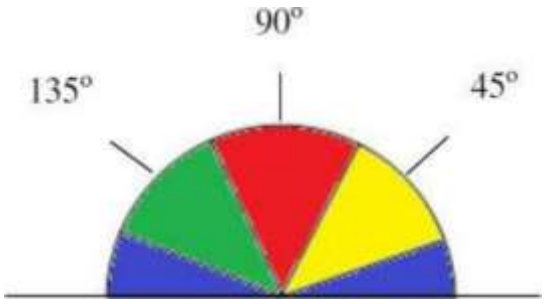
(2.3)

Hasil dari Gausian filter tahap pertama deteksi tepi canny dikonvolusi kembali dengan filter operator sobel Mx sehingga menghasilkan nilai Gx, sedangkan untuk mengetahui nilai Gy adalah hasil konvolusi gausifan filter dikonvolusi kembali dengan filter operator sobel My. Magnitudo gradien (juga dikenal sebagai kekuatan tepi) dapat ditentukan sebagai jarak Euclidean yang diukur mengukur dengan menerapkan hukum Pythagoras seperti yang ditunjukkan dalam Persamaan (2.4).

$$G = \sqrt{G_{x^2} + G_{y^2}} \tag{2.4}$$

Selanjutnya menentukan tepian yang sebenarnya ini, arah tepian harus ditentukan dan disimpan seperti ditunjukkan dalam Persamaan (2.6) [12]

$$\theta = arctan\left(\frac{|G_x|}{|G_y|}\right) \quad (2.6)$$



Gambar 2. 1 Area Konversi Arah Tepi

Aturan konversi yang berlaku sebagai berikut:

a. Semua arah tepi yang berkisar antara 0 dan 22,5 serta 157,5 dan 180 derajat (warna biru) diubah menjadi 0 derajat.

b. Semua arah tepi yang berkisar antara 22,5 dan 67,5 derajat (warna kuning) diubah menjadi 45 derajat.

c. Semua arah tepi yang berkisar antara 67,5 dan 112,5 derajat (warna merah) diubah menjadi 90 derajat.

d. Semua arah tepi yang berkisar antara 112,5 dan 157,5 derajat (warna hijau) diubah menjadi 135 derajat. [11]

3. *Non-Maximum Suppresion*

Merupakan proses yang digunakan untuk melakukan penghilangan non- maximum. Penghilangan non-maximum dilakukan di sepanjang tepi pada arah tepi dan menghilangkan piksel-piksel (piksel diatur menjadi 0) yang tidak dianggap sebagai tepi. [11]

4. *Double Thresholding*

Merupakan proses tepian yang berpotensi ditentukan oleh thresholding. Pada proses ini menggunakan dua ambang T1 (ambang bawah) dan T2 (ambang atas). Semua piksel yang bernilai lebih besar dari T1 dianggap sebagai piksel tepi, lalu semua piksel yang memiliki nilai lebih besar dari T2 juga dianggap sebagai piksel tepi. [11]

5. *Edge Tracking by Hysteresis*

Merupakan proses tepian final ditentukan dengan menekan semua sisi yang tidak terhubung dengan tepian yang sangat kuat. Pengambangan hysteresis dilakukan dengan melibatkan dua ambang T1 dan T2. Nilai yang kurang dari T1 akan diubah menjadi warna hitam (nilai 0) dan nilai yang lebih besar dari T2 akan diubah menjadi putih (nilai 255). Sementara nilai yang lebih atau sama dengan T1 tetapi kurang dari T2 akan diberi nilai 128 dan yang menyatakan nilai abu-abu atau belum jelas, akan dijadikan 0 atau 255, apabila kondisi seperti itu terpenuhi, angka 128 diubah menjadi 255. [11]

2.4.3 Segmentasi Objek

Dalam pengolahan citra, terkadang kita menginginkan pengolahan hanya pada obyek tertentu. Oleh sebab itu, perlu dilakukan proses segmentasi citra yang bertujuan untuk memisahkan antara objek (foreground) dengan background. Pada umumnya keluaran hasil segmentasi citra adalah berupa citra biner di mana objek (foreground) yang dikehendaki berwarna putih (1), sedangkan background yang ingin dihilangkan berwarna hitam (0). Sama halnya pada proses perbaikan kualitas citra, proses segmentasi citra juga bersifat eksperimental, subjektif, dan bergantung pada tujuan yang hendak dicapai. [11]

2.4.4 Resize

Resize atau penskalaan adalah sebuah operasi geometri yang digunakan untuk memperbesar atau memperkecil ukuran dari sebuah citra sesuai dengan ukuran yang dibutuhkan. Pada penskalaan apabila variabel penskalaannya bernilai lebih besar dari 1, maka ukuran citra akan diperbesar, namun apabila variabel penskalaannya bernilai lebih kecil dari 1 maka ukuran citra akan diperkecil.

Proses skala dapat dilakukan dengan rumus [13]:

$$x = \frac{pb * pp}{pa} \tag{2.7}$$

Keterangan:

x = Nilai piksel baris baru

pb = Ukuran panjang matriks baru

pp = Posisi piksel baris

pa = Ukuran panjang matriks lama

$$\text{y} = \frac{lb * lp}{la} \tag{2.8}$$

Keterangan:

y = Nilai piksel kolom baru

lb = Ukuran lebar matriks baru

lp = Posisi piksel kolom

la = ukuran lebar matriks lama

2.5 Ekstraksi Ciri

Ekstraksi ciri adalah proses pengukuran terhadap data yang telah di normalisasi untuk membentuk sebuah nilai fitur. Nilai fitur digunakan oleh pengklasifikasi untuk mengenali unit masukan dengan unit target keluaran dan memudahkan

pengklasifikasian karena nilai ini mudah untuk dibedakan. Pada penelitian ini, metode yang digunakan untuk ekstraksi ciri adalah metode Principal Component Analysis.

2.5.1 *Principal Component Analysis*

Principal Componen Analysis (PCA) adalah sebuah cara untuk mengidentifikasi pola pada data dan kemudian mengekspresikan data tersebut ke bentuk yang lain untuk menunjukkan perbedaan dan persamaan antar pola. Tujuan dari PCA adalah untuk mereduksi dimensi yang besar dari ruang data (observed variables) menjadi dimensi yang lebih kecil dari ruang fitur (independent variables), yang dibutuhkan untuk mendeskripsikan data lebih sederhana. *Principal Component Analysis* menggunakan vektor-vektor yang disebut dengan eigenvector dan nilai-nilai yang disebut dengan eigenvalue untuk mendapatkan fitur yang paling signifikan pada dataset. Prinsip dasar dari algoritma *Principal Component Analysis* adalah mengurangi satu set data namun tetap mempertahankan sebanyak mungkin variasi dalam set data tersebut.

Secara matematis *Principal Component Analysis* mentransformasikan sebuah variabel yang berkolerasi ke dalam bentuk yang bebas tidak berkolerasi. [14] Berikut adalah tahap dalam algoritma PCA:

1. Hitung rata-rata seluruh sampel data diperoleh dengan menggunakan persamaan:

$$\bar{x}_j = \frac{\sum_{ij=1}^{n} x_{ij}}{n} \tag{2.9}$$

2. *Adjusted* Data (data yang telah disesuaikan) adalah hasil pengurangan dari setiap data dengan rata-rata setiap data yang diperoleh dengan rumusan berikut:

$$Adjusted\ data = x_{ij} - \bar{x}_j \tag{2.10}$$

Dengan x’ = *Adjusted* Data

3. Hitung matrik kovarian (c) dihitung dengan menggunakan persamaan berikut [15]:

$$\boldsymbol{c} = \frac{\mathbf{1}}{\boldsymbol{M}-\mathbf{1}}\boldsymbol{x'^{T}}.\boldsymbol{x'} \tag{2.11}$$

Dimana x’ adalah *Adjusted* Data dan x’^{T} adalah *transpose* dari matrik x

4. Hitung nilai eigen dan vector eigen dari matrik kovarian dihitung dengan menggunakan persamaan karakteristik berikut ini:

$$Cv = \lambda Iv \tag{2.12}$$

$$Cv - \lambda Iv = 0$$

$$(C - \lambda I)v = 0 \tag{2.13}$$

$$|C - \lambda I| = 0$$

Dimana c adalah matrik kovarian, I adalah matrik Identitas, λ adalah nilai eigen dan v adalah vector eigen.

5. Hitung nilai eigen yang terbesar yang berkorespondensi terhadap nilai vector eigen yang terbesar dipilih menjadi Principal Component. Vektor eigen yang disusun dari yang terbesar ke yang terkecil dipilih menjadi vektor fitur.

$$v = (eig_1, eig_2, eig_3 \ldots eig_n) \tag{2.14}$$

6. Untuk mencari *Principal Component* dengan x’ sebagai rata-rata

$$PC = X' \times v \tag{2.15}$$

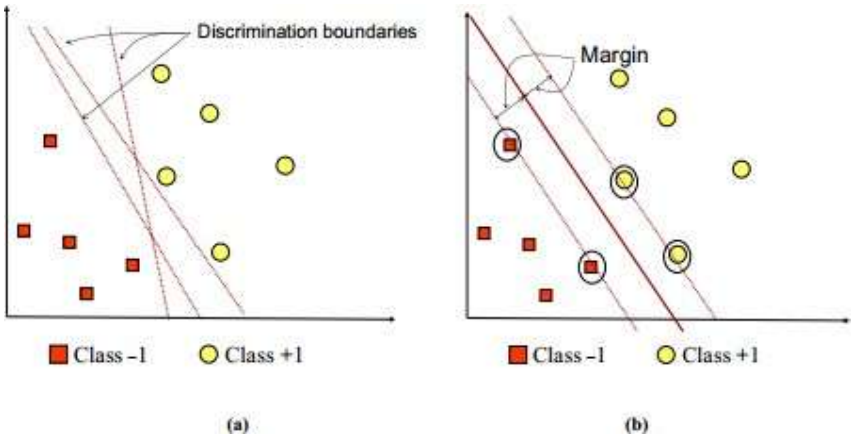
7. Langkah berikutnya untuk melakukan transformasi data untuk menghasilkan data PCA.

$$PCA\ data = PC^{T} \times X'^{T} \tag{2.16}$$

2.6 *Support Vector Machine (SVM)*

Support Vector Machine (SVM) diperkenalkan oleh Vapnik pada tahun 1992 sebagai suatu teknik klasifikasi yang efisien untuk masalah nonlinier. SVM berbeda dengan teknik klasifikasi di era 1980-an, seperti decision tree dan ANN, yang secara konsep kurang begitu jelas dan seringkali terjebak pada optimum lokal. SVM memiliki konsep yang jauh lebih matang, lebih jelas secara matematis, dibanding teknik – teknik klasifikasi sebelum era 1990-an. SVM berusaha menemukan hyperplane dengan memaksimalkan jarak antar kelas. Dengan cara ini, SVM dapat menjamin kemampuan generalisasi yang tinggi untuk data – data yang akan datang. [2]

Menurut penelitian yang dilakukan peneliti [2] konsep SVM dapat dijelaskan secara sederhana sebagai usaha mencari hyperplane terbaik yang berfungsi sebagai pemisah dua buah class pada input space. Gambar 2.2-a memperlihatkan beberapa pattern yang merupakan anggota dari dua buah class : +1 dan –1. Pattern yang tergabung pada class –1 disimbolkan dengan warna merah (kotak), sedangkan pattern pada class +1, disimbolkan dengan warna kuning(lingkaran).



Gambar 2. 2 SVM Mencari Hyperline Terbaik

Masalah klasifikasi dapat diterjemahkan dengan usaha menemukan garis (*hyperplane*) yang memisahkan antara kedua kelompok tersebut. Berbagai alternatif

garis pemisah (*discrimination boundaries*) ditunjukkan pada Gambar 2.2-a. *Hyperplane* pemisah terbaik antara kedua *class* dapat ditemukan dengan mengukur *margin hyperplane* tsb. dan mencari titik maksimalnya. *Margin* adalah jarak antara *hyperplane* tersebut dengan pattern terdekat dari masing-masing *class*. *Pattern* yang paling dekat ini disebut sebagai *support vector*.

Garis solid pada Gambar 2.2-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. Data yang tersedia dinotasikan sebagai $\vec{x_i} \in \Re^d$ sedangkan label masing-masing dinotasikan yi ∈ {-1,+1}untuk i = 1,2,...n ,yang mana n adalah banyaknya data. Diasumsikan kedua *class* –1 dan +1 dapat terpisah secara sempurna oleh *hyperplane* berdimensi *d* , yang didefinisikan sebagai berikut:

$$\vec{w} \,.\, x + b = 0 \tag{2.17}$$

Pattern $\vec{x_i}$ yang termasuk *class* -1 (sampel negatif) dapat dirumuskan sebagai:

$$\vec{w} \,.\, x + b \leq -1 \tag{2.18}$$

Sedangkan *pattern* $\vec{x_i}$ yang termasuk *class* +1 (sampel positif) adalah sebagai berikut:

$$\vec{w} \,.\, x + b \geq +1 \tag{2.19}$$

Margin terbesar dapat ditemukan dengan memaksimalkan nilai jarak antara *hyperplane* dan titik terdekatnya, yaitu $1 \,/\, \|\vec{w}\|$. Hal ini dapat dirumuskan sebagai *Quadratic Programming (QP) problem*, yaitu mencari titik minimal persamaan (2.20), dengan memperhatikan *constraint* persamaan (2.21)

$$\boldsymbol{min\tau}(\boldsymbol{w}) = \frac{\mathbf{1}}{\mathbf{2}}||\boldsymbol{w}||^{\mathbf{2}} \tag{2.20}$$

$$\boldsymbol{y_i}(\boldsymbol{w} * \boldsymbol{x_i} + \boldsymbol{b}) - \mathbf{1} \geq \mathbf{0}, (\boldsymbol{i} = \mathbf{1}, \ldots, \boldsymbol{n}) \tag{2.21}$$

Problem ini dapat dipecahkan dengan berbagai teknik komputasi, di antaranya *Lagrange Multiplier*

$$L(w,b,a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} a_i y_i (w^T x_i + b) + \sum_{i=1}^{n} a_i \quad (2.22)$$

α_i adalah *Langrange multipliers*, yang bernilai nol atau positif ($\alpha_i \geq 0$). Nilai optimal dari persamaan (2.22) dapat dihitung dengan meminimalkan L terhadap $\vec{w}$ dan b, dan memaksimalkan L terhadap α_i. Dengan memperhatikan sifat bahwa pada titik optimal *gradient* L = 0, persamaan (2.22) dapat dimodifikasi sebagai maksimalisasi *problem* yang hanya mengandung α_i, sebagaimana persamaan (2.23) dibawah.

Maximize :

$$Max \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1,j=1}^{n} a_i a_j y_i y_j{}^T x_i x_j^T \quad (2.23)$$

Dimana $\alpha i \geq 0$ (=1,2,….l) dan $\Sigma \alpha i yi = 0 li = 1$. Dari hasil perhitungan ini diperoleh αi yang kebanyakan bernilai positif. Data yang berkorelasi dengan αi yang positif inilah yang disebut *support vector*.

Untuk mendapatkan nilai αi, langkah pertama adalah mengubah setiap *feature* menjadi nilai vektor (*support vector*) = xy. Kemudian vektor akan dimasukkan kedalam persamaan *kernel trick phi* φ yaitu sebagai berikut

$$\boldsymbol{\varphi}\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{bmatrix} = \begin{Bmatrix} \sqrt{\boldsymbol{Xn^2 + Yn^2}} > \boldsymbol{2}, \begin{bmatrix}\boldsymbol{4 - x +} & \boldsymbol{|\, x - y\,|}\\ \boldsymbol{4 - x +} & \boldsymbol{|\, x - y\,|}\end{bmatrix} \\ \sqrt{\boldsymbol{Xn^2 + Yn^2}} \leq \boldsymbol{2}, \boldsymbol{maka}\ \begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{bmatrix} \end{Bmatrix} \quad (2.24)$$

Kemudian untuk mencari nilai αi didapatkan dari persamaan sebagai berikut.

$$\sum_{i=1,j=1}^{n} a_i S_i^T S_j \quad (2.25)$$

$$\sum_{i=1,j=1}^{n} a_i S_i^T S_j = y_i \quad (2.26)$$

Setelah parameter a_i didapatkan kemudian dimasukkan ke persamaan 2.27 berikut.

$$\sum_{i=1}^{n} a_i S_i \tag{2.27}$$

Terakhir akan dicari nilai w dan b untuk menemukan *hyperplane* sebagai patokan proses klasifikasi dengan persamaan sebagai berikut.

$$\boldsymbol{y = wx + b} \tag{2.28}$$

Nilai s_i dari persamaan 2.27 merupakan nilai *support vector* yang telah dihitung sebelumnya. Dengan demikian proses klasifikasi selesai dengan memperhatikan *hyperplane* nya.

2.7 *Reduced Support Vector Machine*

Terdapat dua masalah besar dalam klasifikasi data besar yang non linier, yang pertama kesulitan komputasi dalam memecahkan masalah optimasi tanpa kendala dan melibatkan fungsi kernel yang membutuhkan memori sangat besar. Pada umumnya komputer mengalami out of memory bahkan sebelum dimulai proses pencarian solusi. Yang kedua kesulitan dalam menggunakan formula yang tidak terlihat, untuk bidang pemisah . untuk menyelesaikan masalah tersebut muncul sebuah ide menyelesakan bidang pemisah non linier pada data besar dengan hanya menggunakan sebagian karakteristik dari data. Hal inilah yang mengawali ide dasar dari RSVM. Dengan formulasi data penuh (full set) $A \in R^{mxp}$ dengan square kernel $K(X_i, X'_j) \in R^{mxm}$ dimodifikasi sedemikian hingga reduced dataset $\bar{A} \in R^{\bar{m}xp}$dengan diagonal matriks $\bar{D}$ dan matriks kernel $K\left(X_i, \overline{X'}_j\right) \in R^{mxm}$.Selanjutnya algoritma tersebut diselesaikan dengan smoothing technique. Hasil modifikasi tersebut dirumuskan dengan menggantikan X' dengan $\overline{X'}$. [9]

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 :

$$min_{w,b,\xi} \ \ \frac{c}{2}\xi'\xi + \frac{1}{2}\,(w'w + b^2) \tag{2.29}$$

dengan kendala

$$D\,(Aw + eb) + \ \xi \ \geq e, \xi \ \geq \ 0 \tag{2.30}$$

Solusi problem adalah

$$\xi \ = \big(e - D(Aw + eb)\big) \tag{2.31}$$

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

$$min_{w,b} \ \ \frac{c}{2}||(e - D(Aw - eb)||_2^2 + \frac{1}{2}\,(w'w + b^2) \tag{2.32}$$

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:

$$p\,(x, a) = x + \frac{1}{a}\log(1 + \ \varepsilon^{-ax}), a > 0 \tag{2.33}$$

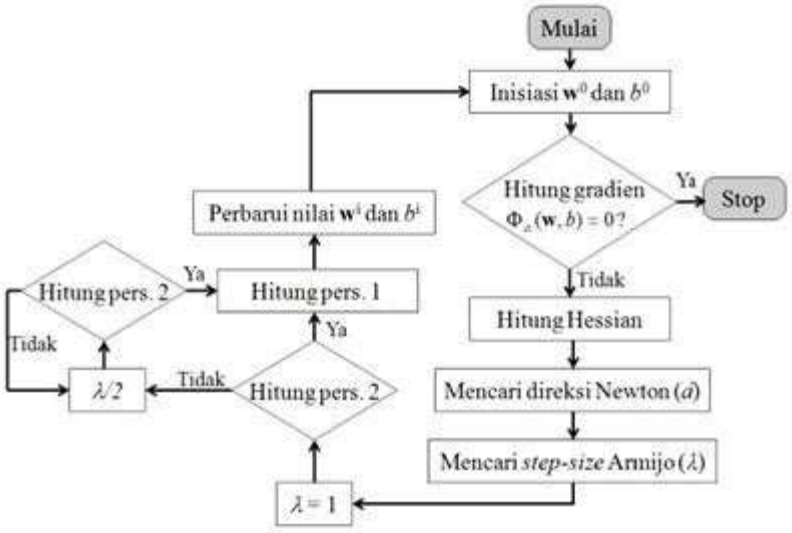
dimana α adalah parameter smoothing. Dengan menggantikan fungsi plus dengan p (x,a) maka diperoleh model SSVM sebagai berikut:

$$min_{(w,b)\in Rp+1}\phi_\alpha(w, b) = min_{(w,b)\in Rp+1} \ \ \tfrac{c}{2}||p(e - D(Aw - eb)||_2^2 + \tfrac{1}{2}\,(w'w + b^2) \tag{2.34}$$

Secara umum, problem optimasi dapat ditulis sebagai berikut:

$$\Psi_\alpha\,(w^i, y^i) \ \ = \tfrac{v}{2}\big\|p(e - D)(Aw - \gamma)\alpha \ \ \big\|_2^2 + \tfrac{1}{2}(W^TW + \gamma^2) \tag{2.35}$$

Yang diselesaikan dengan iterasi Newton Armijo (Gambar 2.3) dan $K(X_{i,}X'_j)$ merupakan fungsi kernel yang dalam penelitian ini digunakan kernel Gaussian atau bisa dirumuskan berikut $K(X_{i,}X'_j) = exp\ (-y(\|X_{i,}X'_j\|^2)$ dengan parameter kernel y.



Gambar 2. 3 Alur Algoritma Newton Armijo

Menghitung Gradien dengan persamaan (2.36) kemudian cek apakah nilai gradiaen = 0? Menggunakan persamaan (2.37) , jika nilai gradien lebih dari 0 , maka harus menghitung nilai hessian dengan persamaan (2.38) berikut.

$$\lim_{a\to\infty} \nabla\Psi_\alpha\ (w,y) \quad = \begin{bmatrix} w - vA^T D\big(\ e - D(Aw - ey)\big) \\ y + ve^T D\big(\ e - D(Aw - ey)\big) \end{bmatrix} \tag{2.36}$$

$$\left\| \lim_{a\to\infty} \nabla\Psi_\alpha\ (w,y) \quad \right\|_2^2 \tag{2.37}$$

$$S_\infty(x) = \lim_{a\to\infty} \left(\frac{1}{1+\ \varepsilon^{-ax}}\right) \quad = \frac{1 + sign(x)}{2} \tag{2.38}$$

Menentukan nilai direksi newton

$$d^i = \nabla^2\Psi_\alpha\ (w^i, y^i)^{-1}(-\Psi_\alpha\ (w^i, y^i) \tag{2.39}$$

Persamaan 1 :

$$\Psi_\alpha\,(w^i, y^i) \quad - \Psi_\alpha(w^i, y^i) + \lambda_i d^i \;\geq\; -\delta\lambda\nabla\Psi_\alpha(w^i, y^i)\; d^i \tag{2.40}$$

Persamaan 2 :

$$\Psi_\alpha(w^i, y^i) + \lambda_i d^i \tag{2.41}$$

Saat iterasi pada algoritma Newton-Armijo berhenti, diperoleh nilai w dan b yang konvergen. Dengan demikian fungsi pemisah yang diperoleh untuk kasus klasifikasi linier adalah :

$$f(x) = sign(g(x)) \tag{2.42}$$

Sedangkan fungsi permisah untuk kasus klasifikasi nonlinier adalah sebagai berikut:

$$F(x) \;=\; sign\,(\text{w'x} \;+\; \gamma\,) \;=\; sign\,(\,\text{u'}D'K(\bar{W}'\bar{W}) \;+\; \gamma) \tag{2.43}$$

Setelah menyelesaikan persamaan SSVM (2.35) yang telah dimodifikasi dan diselesaikan dengan algoritma *Newton-Armijo* dimana W' digantikan oleh $\bar{W}'$ dengan $\bar{D} \subset D$. Maka diperoleh model RSVM sebagai berikut.

$$\Psi_\alpha\,(w^i, y^i) \;=\frac{v}{2}\left\|p(e-D)(Aw-\gamma)\alpha \quad \right\|_2^2 + \frac{1}{2}(\bar{W}^T\bar{W} + \gamma^2) \tag{2.44}$$

2.8 UML

Salah satu model perancangan berorientasi objek yang saat ini paling sering digunakan di seluruh dunia adalah Unified Modelling Language (UML). UML sendiri adalah bahasa pemodelan untuk sistem atau perangkat lunak yang memiliki paradigma berorientasi objek.

UML adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan

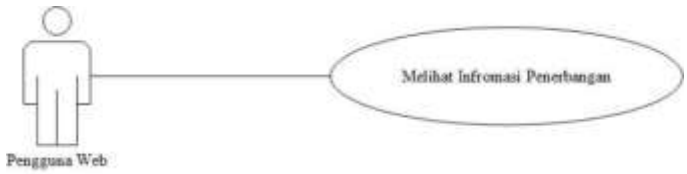
menggunakan UML, perancang atau pemodel dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. UML lebih cocok diterapakan pada piranti berorientasi objek seperti C++, Java, C#, dan sebagainya. Tetapi UML juga tetap dapat digunakan untuk modeling aplikasi prosedural semisal VB atau C.

UML versi 2.0 mencakup 13 macam diagram dan perangkat yang berfungsi untuk menggambarkan sistem informasi berorientasi objek dengan sangat lengkap dan rinci. Meski demikan, tidak selalu ke-13 diagram dan perangkat tersebut digunakan saat para pengembang berupaya mengemabangkan perangkat lunak berorientasi objek. Beberapa diantara digaram UML yang digunakan adalah Use Case Diagram, Activity Diagram, Sequence Diagram, Class Diagram, dan Collaboration Diagram. [20]

2.8.1 *Use Case Diagram*

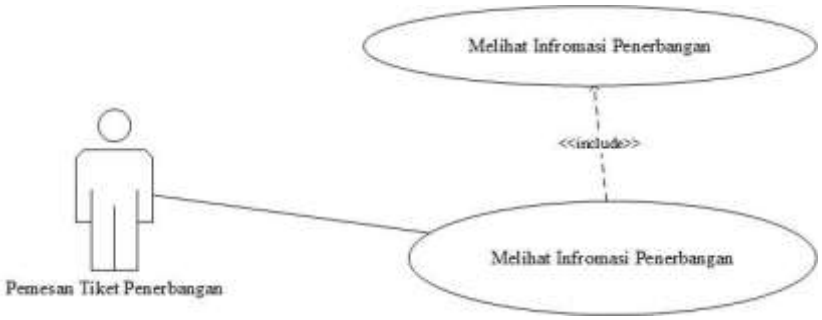
Use case diagram pada dasarnya digunakan untuk mendeskripsikan bagaimana entitas eksternal akan menggunakan sistem atau perangkat lunak. Entitas eskternal itu dapat berupa manusia atau sistem yang lain. Dalam diagram use case, entitas eksternal ini sering dinamakan sebagai actor. Deskripsi diagram use case ini lebih menekankan pada sistem dari sudut pandang penggunanya dan juga menekankan pada interaksi yang terjadi di antara pengguna dengan sistem. Use case sangat membantu pengembang sistem untuk lebih jauh mendefinisikan ruang lingkup sistem serta batasan-batasannya.

Actor pada dasarnya adalah segala sesuatu yang berada di luar sistem atau perangkat lunak yang sedang dikembangkan. Setiap interaksi yang terjadi di antara actordan sistem dimodelkan sebagai use case. Contoh diagram use case sederhana dapat dilihat pada Gambar 2.4.



Gambar 2. 4 Use case diagram

Dalam diagram use case, sering ditemukan asosiasi yang bertipe <include>. Asosiasi <include> ini menujukkan bahwa use case tertentu, misalnya "Melihat Informasi Penerbangan" harus dilakukan terlebih dahulu sebelum use case lainnya seperti "Memesan Kursi Pesawat" dilakukan. Terdapat asosiasi yang terjadi di antara use case "Melihat Informasi Penerbangan" dan use case "Memesan Kursi Pesawat", sedangkan pemesan tiket penerbangan tidak dapat menggunkan use case "Memesan Kursi Pesawat" secara mandiri dari use case "Melihat Informasi Penerbangan". Implementasi asosiasi <include> dalam kasus tersebut pada diagram use case dapat dilihat Gambar 2.5.

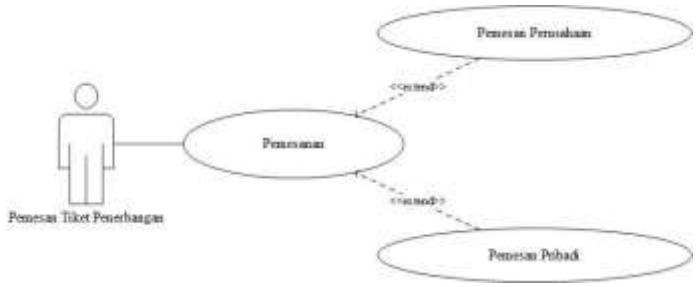


Gambar 2. 5 Asosiasi Include

Selain asosiasi <include> dalam penggambaran juga dikenal asosiasi yang bertipe <extend>. Asosiasi <extend> ini hampir mirip dengan asosiasi bertipe

<include>, hanya saja use case yang diasosiasikan sebagai <extend> tidak wajib dilaksanakan. Misalnya actor "Pemesan Tiket Penerbangan" dapat melakukan use case "Pemesanan" dengan terlebih dahulu melaksanakan use case "Pemesanan Perusahaan" atau "Pemesanan Pribadi", tetapi tidak keduanya. Artinya, salah satu dari kedua use

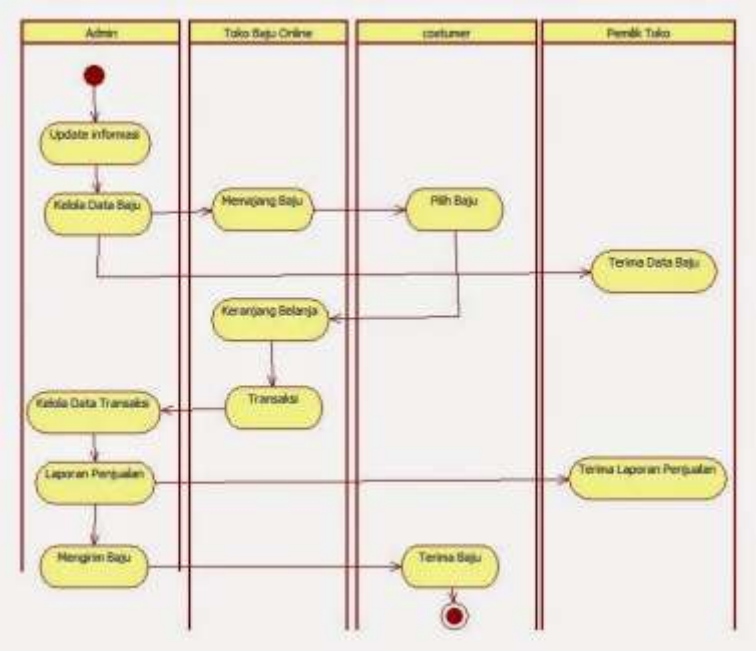
case ini harus dilaksanakan. Implementasi asosiasi <extend> dalam kasus tersebut pada diagram use case dapat dilihat Gambar 2.6.



Gambar 2. 6 Asosiasi Extend

2.8.2 *Activity Diagram*

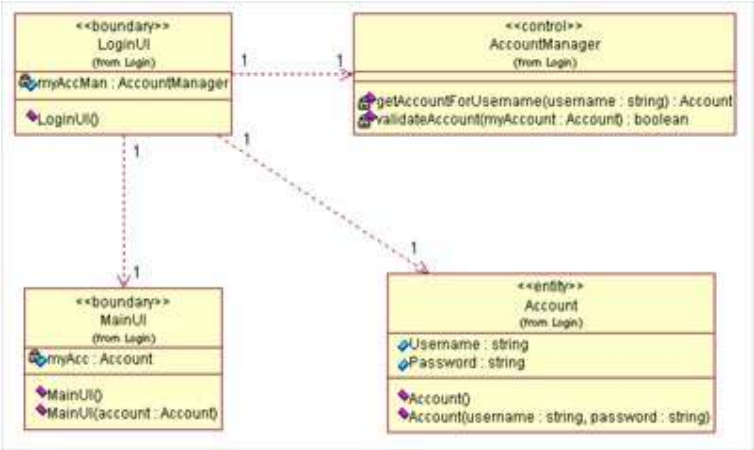
Sebuah activity diagram atau diagram aktivitas dapat direalisasikan oleh satu use case atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara use case menggambarkan bagaimana actor menggunakan sistem untuk melakukan aktivitas. Sama seperti state, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas.Digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses- proses paralel (fork dan join) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. Activity diagram dapat dibagi menjadi beberapa object swimlane untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu. Contoh dari activity diagram dapat dilihat pada Gambar 2.7.



Gambar 2. 7 Activity Diagram

2.8.3 *Class Diagram*

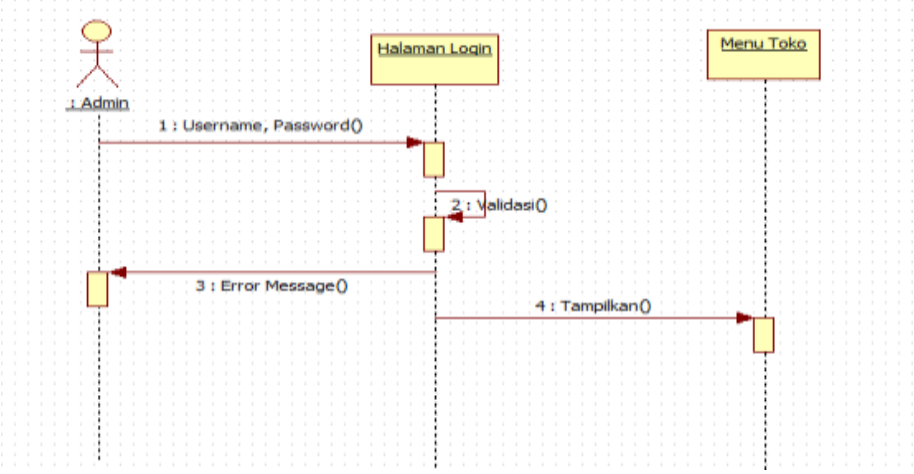
Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. Class diagram membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain, class diagram berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat. Class memiliki tiga area pokok, yaitu nama (dan stereotype), atribut, dan metoda. Contoh class digram dapat dilihat pada Gambar 2.8.



Gambar 2. 8 Class Diagram

2.8.4 *Sequence Diagram*

Squence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya Sequence diagram adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan use case



Gambar 2. 9 Sequence Diagram

diagram. Contoh Sequence diagram dapat dilihat pada Gambar 2.9.

2.9 *Python*

Python adalah bahasa pemrograman komputer, sama layaknya seperti bahasa pemrograman lain misal C, C++, Pascal, Java, PHP, Perl, dan lain-lain. Sebagai Bahasa pembrograman, Python tentu memiliki dialek, kosakata atau kata kunci (*keyword* dan aturan tersendiri yang jelas berbeda dengan bahasa pemrograman lainnya. Bahasa pemrograman Python disusun di akhir tahun 1980-an dan implementasinya baru mulai pada Desember 1989 oleh Guido Van Rossum di Centrum Wiskunde & Informatica (CWI), sebuah pusat riset di bidang matematika dan sains, Amsterdam – Belanda. Sebagai suksesor atas pengganti dari bahasa pemrograman pendahulunya, bahasa pemrograman ABC, yang juga dikembangkan di CWI oleh Leo Guerts, Lambert Mertens, dan Steven Pemberton. [21]

2.10 Pengujian *Confusion Matrix*

Confusion matrix melakukan pengujian untuk memperkirakan obyek yang benar dan salah [22]. Urutan pengujian ditabulasikan dalam confusion matrix dimana kelas yang diprediksi ditampilkan di bagian atas matriks dan kelas yang diamati dibagian kiri. Setiap sel berisi angka yang menunjukan berapa banyak kasus yang sebenernya dari kelas yang diamati untuk diprediksi.Model confusion matrix untuk contoh 2 kelas dapat dilihat pada Tabel 2.2 sebagai berikut:

Tabel 2. 2 Model Confusion Matrix

		Nilai Prediksi	
		Positif	Negatif
Nilai Aktual	Positif	TP	FN
	Negatif	FP	TN

Keterangan:

TP = jumlah nilai positive yang diklasifikasikan positif.

TN = jumlah nilai negatif yang diklasifikasikan negatif.

FP = jumlah nilai positif yang diklasifikasikan positif.

FN = jumlah nilai negatif yang diklasifikasikan negatif.

Perhitungan untuk mendapatkan akurasi dapat dilihat pada persamaan 2.45 berikut:

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.45}$$

Perhitungan untuk mengetahui PPV(nilai prediksi positif) dapat dilihat pada persamaan 2.46 berikut:

$$PPV = \frac{TP}{TP + FP} \tag{2.46}$$

Perhitungan untuk mengetahui NPV(nilai prediksi negatif) dapat dilihat pada persamaan 2.47 berikut:

$$NPV = \frac{TN}{TN + FN} \tag{2.47}$$