BAB 2
LANDASAN TEORI

2.1 Grafologi

Grafologi berasal dari bahasa Yunani, *graph* yang berarti tulisan atau menulis, dan logos yang berarti ilmu. Grafologi adalah cabang dari ilmu psikologi dalam mata kuliah *psikografik* atau *psikodiagnostik.* Grafologi adalah "seni dan ilmu yang mempelajari tentang tulisan tangan". Grafologi adalah seni menilai karakter seseorang dan kepribadian seseorang dengan cara melihat tipe tulisan tangan dan tanda tangan.

Ketepatan seseorang grafologi menganlisis tulisan tangan dan tanda tangan bergantung pada keahlian dan bakat sebagaimana pakar psikologi klinikal. Seorang ahli grafologi memerlukan waktu yang tidak singkat agar benar-benar mahir dalam bidang ini. Grafologi tidak saja bermanfaat untuk mengetahui karakter seseorang untuk kepentingan pribadi, tetapi ilmu ini banyak digunakan dalam memilih dan menilai calon karyawan, seperti yang banyak dilakukan di Amerika Serikat dan Eropa. Di dunia pendidikan, grafologi bisa digunakan untuk mendeteksi *problem* yang sedang dialami peserta didik ataupun dalam mengukur kecerdasan si anak. Bahkan di ranah praktik terapi psikis, grafologi dapat pula dijadikan sebagai salah satu metode terapi/rehabilitasi.

Grafo-*test* sudah digunakan sebagai bagian di *forensic* atau *biometric*. Di Amerika, grafo-*test* digunakan untuk mengetahui kejujuran, kestabilan emosi, kemungkinan bertindak kasar, dan *judgment*. Di Australian *Federal*, *State* & *Territory Police* sebagai bentuk test yang lebih akurat daripada *lie detector*. Di Prancis dan Swiss, banyak perusahaan menggunakan grafo-*test* untuk mencari karakter karyawan yang sesuai dengan kriteria perusahaan. [1]

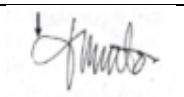
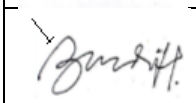
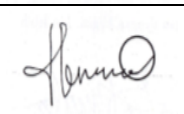
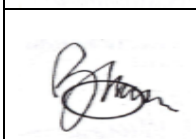
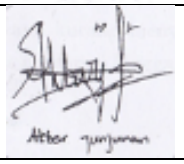
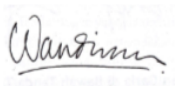
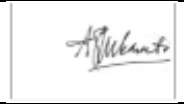
Berikut adalah beberapa manfaat dari grafo-*test*:

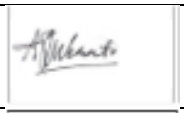
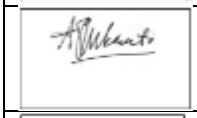
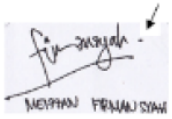
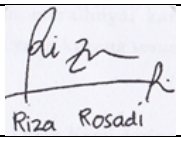
1. Rekrutmen.
2. *Test* kepribadian
3. Pembentukan tim manajemen dan peningkatan performa dari staf yang ada.

4. Konsultasi untuk anak dan dewasa.
5. Bimbingan mengenai bidang pendidikan dan pekerjaan yang cocok dengan kepribadian.
6. Penilaian mengenai kemungkinan penipuan/kecenderungan berbohong.
7. Memberikan gambaran mengenai diri sendiri agar dapat lebih mengenal dan mengembangkan diri. [1]

Fitur-fitur tanda tangan yang terdapat [4] dan sudah diperbaiki oleh grafologi Indonesia diantaranya dapat dilihat pada Tabel 2.1 sebagai berikut:

Tabel 2.1 Fitur - Fitur Tanda Tangan

No	Fitur	Gambar	Ciri	Kepribadian
1	Cangkang		Lengkung tertutup	Ketakutan berlebihan, introvert, tidak memperdulikan sekitar, tidak suka bergaul dan bekerja sama.
2	Awal Kurva		Lengkung mundur	Nyaman akan masa lalu.
			Lengkung tajam	Mampu memformulasi pikiran secara tajam.
			Lengkung lembut	Hati-hati, ramah, diplomatis.
3	Coretan Akhir		Menaik	Terbuka, pandangan ke depan, keinginan maju, percaya diri.
			Menurun	Kurang semangat, berfikir realistis, kurang percaya diri, mudah putus asa.
4	Coretan Ditengah		Adanya coretan	Kurang percaya diri dan mudah depresi.
5	Garis Bawah		Adanya garis bawah	Membutuhkan dukungan membuat keputusan, serta memiliki keandalan dalam memimpin.
6	*Margin* ekstrim		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	Struktur Titik		Ada titik	Pendirian stabil, memiliki rasa curiga, selalu menjaga jarak tidak mudah percaya.
8	Tanda Tangan Terpisah		Ada tanda tangan terpisah	Memiliki pengalaman kurang menyenangkan di masa lalu.
9	Garis Terpisah		Ada garis terpisah	Membatasi keinginannya, tidak berani mengambil resiko,sering patah semangat dan ragu mengambil keputusan.

Penelitian ini menggunakan 4 fitur yaitu awal kurva, coretan akhir, adanya coretan ditengah, garis bawah. 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]

Tabel 2.2 Warna Dan Nilai Penyusun Warna

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 *Preprocessing*

Tahap *preprocessing* merupakan penjelasan tahap awal dalam alur kerja untuk menyiapkan citra sebagai data input yang siap diolah lebih lajut oleh sistem. Proses pada *preprocessing* 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:

$$I = a\ x\ R + b\ x\ G + c\ x\ B$$

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]:

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

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 [4]:

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

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:

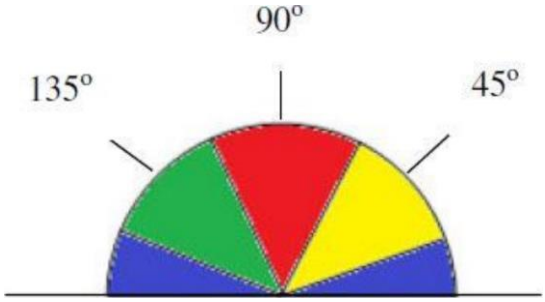
$$Mx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad My = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (2.3)$$

   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).

$$G = \sqrt{{G_x}^2 + {G_y}^2} \quad (2.4)$$

Selanjutnya menentukan tepian yang sebenarnya ini, arah tepian harus ditentukan dan disimpan seperti ditunjukkan dalam Persamaan (2.6) [4]

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



Gambar 2.1 Area Untuk 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 T_1 (ambang bawah) dan T_2 (ambang atas). Semua piksel yang bernilai lebih besar dari T_1 dianggap sebagai piksel tepi, lalu semua piksel yang memiliki nilai lebih besar dari T_2 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 T_1 dan T_2. Nilai yang kurang dari T_1 akan diubah menjadi warna hitam (nilai 0) dan nilai yang lebih besar dari T_2 akan diubah menjadi putih (nilai 255). Sementara nilai yang lebih atau sama dengan T_1 tetapi kurang dari T_2 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]

$$\text{Edge Tracking} = \begin{cases} 0 & ,nilai < T_1 \\ 255 & ,nilai > T_2 \\ 255 & ,nilai \geq T_1 \quad dan \quad nilai \leq T_2 \end{cases}$$

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 penskalaan 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

$$L_o = \frac{lb * lp}{la} \quad (2.8)$$

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 ini:

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

   Dengan x′ = *Adjusted* Data

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

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

   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

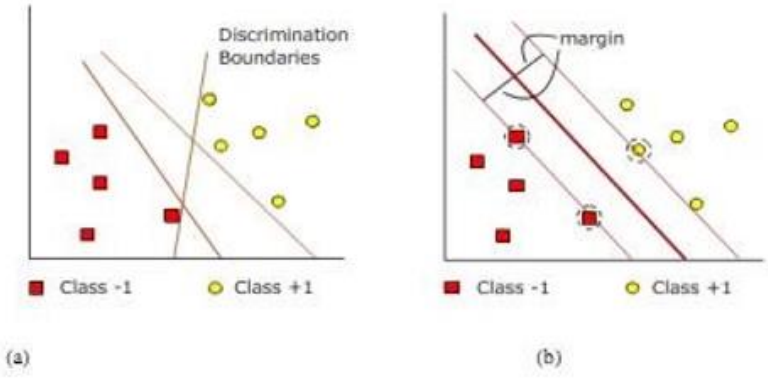
$$PC = X' \times v \tag{2.15}$$

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

$$PCA\ data = PC^T \times X'^T \quad (2.15)$$

2.6 *Support Vector Machine*

Support Vector Machine (SVM) pertama kali diperkenalkan oleh Valdimir Vanpik pada tahun 1992 sebagai rangkaian harmonisasi konsep-konsep unggulan dalam bidang pengenalan pola. *Support Vector Machine* merupakan metode pembelajaran yang digunakan untuk klasifikasi biner, ide dasarnya adalah mencari hyperplane terbaik yang berfungsi sebagai pemisah dua *class* pada *input space* [16].



Gambar 2.2 SVM Berusaha Menemukan Hyperplane Terbaik Yang Memisahkan Dua Kelas -1 Dan +1

Metode *Support Vector Machine* (SVM) banyak digunakan untuk melakukan klasifikasi otomatis. Beberapa penelitian telah menggunakan SVM untuk berbagai penerapan, diantaranya yaitu pada pengolahan citra, analisis medik, ataupun untuk melakukan prediksi. Hal ini berbeda jika dibandingkan dengan metode Neural Network yang mana bisa mendapatkan solusi yang diperoleh dalam minimum lokal. Pada umumnya SVM membagi ruang vector menjadi 2 yaitu positif dan kelas negatif. Hal tersebut tidak menutup kemungkinan untuk menggunakan SVM untuk keperluan membagi menjadi lebih dari 2 kelas [17], gambar (a) menunjukan pola-pola yang merupakan anggota dari dua buah kelas +1 dan -1, pola yang berada pada anggota -1 kemudian disimbolkan dengan warna merah, sedangkan pola pada +1 warna kuning. Masalah pada klasifikasi dapat dilakukan dengan usaha menemukan

garis (*hyperplane*) yang memisahkan kelompok tersebut. Pendefinisian persamaan suatu *hyperplane* pemisah yang dituliskan dengan:

$$w * x_i + b = 0 \quad (2.16)$$

Data x_i yang terbagi dalam dua kelas, yang termasuk kelas -1 (sample negatif) didefinisikan sebagai vektor yang memenuhi pertidaksamaan 2.17. Sedangkan yang termasuk kelas +1 (sample positif) memenuhi persamaan 2.18. Berikut adalah persamaan 2.17 dan 2.18 dibawah ini:

$$w * x_i + b < 0 \text{ untuk } y_i = -1 \quad (2.17)$$

$$w * x_i + b > 0 \text{ untuk } y_i = +1 \quad (2.18)$$

Keterangan:

x_i = data input

y_i = label yang diberikan

w = nilai dari bidang normal

b = posisi bidang relatif terhadap pusat koordinat

Parameter w dan b adalah parameter yang akan dicari nilainya, bila label y_i = -1, maka pembatas menjadi persamaan 2.19. Apabila label data y_i = +1, maka pembatas menjadi persamaan 2.20. Berikut adalah persamaan 2.19 dan 2.20 dibawah ini:

$$w * x_i + b \leq -1 \quad (2.19)$$

$$w * x_i + b \geq +1 \quad (2.20)$$

Margin terbesar dapat dicari dengan cara memaksimalkan jarak antara bidang pembatas kedua kelas dan titik terdekatnya, yaitu $2/|w|$. Hal ini dirumuskan sebagai permasalahan *Quadratic Programming* (QP) *problem* yaitu mecari titik minimal persamaan 2.21 dengan memperhatikan persamaan 2.22 Berikut:

$$min\tau(w) = \frac{1}{2}\|w\|^2 \quad (2.21)$$

$$y_i(w * x_i + b) - 1 \geq 0, (i = 1, \dots, n) \quad (2.22)$$

Permasalahan ini dapat dipecahkan dengan berbagai teknik komputasi. Lebih mudah diselesaikan dengan mengubah persamaan 2.21 Kedalam fungsi *lagrangian* pada persamaan 2.23 dan menyederhanakannya menjadi persamaan 2.14 berikut:

$$L(w,b,a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} a_i \left(y_i\left((w^T x_i + b) - 1\right)\right) \quad (2.23)$$

$$L(w,b,a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} a_i y_i (w^T x_i + b) + \sum_{i=1}^{n} a_i \quad (2.24)$$

Dimana a_i adalah *lagrange multiplier* yang bernilai nol atau positif ($a_i > 0$). Nilai optimal dari persamaan 2.24 dapat dihitung dengan meminimalkan *L* terhadap *w*, *b*, dan *a*. Dapat dilihat pada persamaan 2.26 sampai 2.27 Berikut:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} a_i y_i x_i = 0 \quad (2.25)$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n} a_i y_i = 0 \quad (2.26)$$

$$\frac{\partial L}{\partial a} = \sum_{i=1}^{n} a_i y_i (w^T x_i + b) - \sum_{i=1}^{n} a_i = 0 \quad (2.27)$$

Maka masalah *lagrange* untuk klasifikasi dapat dinyatakan pada persamaan 2.28 berikut:

$$Min\ L(w,b,a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} a_i y_i (w^T x_i + b) - \sum_{i=1}^{n} a_i \quad (2.28)$$

Dengan memperhatikan persamaan 2.29 dan 2.30 berikut:

$$w - \sum_{i=1}^{n} a_i y_i x_i = 0 \quad (2.29)$$

$$\sum_{i=1}^{n} a_i y_i = 0 \quad (2.30)$$

Model persamaan 2.30 diatas merupakan *primal lagrange*. Sedangkan dengan memaksimalkan *L* terhadap a_i, persamaannya menjadi persamaan 2.31 berikut:

$$Max \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1,j=1}^{n} a_i a_j y_i {y_j}^T x_i x_j^T \quad (2.31)$$

Dengan memperhatikan persamaan 2.32 berikut:

$$\sum_{i=1}^{n} a_i y_i = 0, a_i \geq 0 (i, j = 1, \dots, n) \tag{2.32}$$

Nilai *x* didapatkan dari persamaan 2.34 kernel linier untuk *x* berikut:

$$\sum_{i=1,j=1}^{n} x_i x_j^T , (i, j = 1, \dots, n) \tag{2.34}$$

Nilai *y* didapatkan dari persamaan 2.35 kernel linier untuk *y* berikut:

$$\sum_{i=1,j=1}^{n} y_i y_j^T , (i, j = 1, \dots, n) \tag{2.35}$$

Untuk mendapatkan jarak tegak lurus yang optimal dengan mempertimbangkan vektor positif, maka hasil perhitungan drai substitusi nilai *x* dan nilai *y* ke persamaan 2.35 diberi nilai bias = 1. Kemudian cari parameter a_i, dengan terlebih dahulu mencari nilai fungsi setiap abstrak menggunaan persamaan 2.36, lalu mencari nilai a_i, pada persamaan linear menggunakan persamaan 2.37 dengan memperhatikan *i*,*j* = 1,...,*n* berikut:

$$\sum_{i=1,j=1}^{n} a_i S_i^T S_j \tag{2.36}$$

$$\sum_{i=1,j=1}^{n} a_i S_i^T S_j = y_i \tag{2.37}$$

Setelah parameter a_i didapatkan kemudian dimasukkan ke persamaan 2.38 berikut:

$$\hat{\mathrm{W}} = \sum_{i=1}^{n} a_i S_i \tag{2.38}$$

Hasil yang didapatkan menggunakan persamaan 2.38, selanjutnya digunakan persamaan 2.39, untuk mendapatkan nilai *w* dan *b*:

$$y = wx + b \tag{2.39}$$

Sedemikian sehingga didapatkanlah nilai *w* dan *b* atau nilai *hyperplane* untuk mengklasifikasikan kedua kelas.

Berikut ini adalah beberapa fungsi kernel yang umum digunakan yaitu [10]:

a. Kernel linear

$$K(x_i, x) = {x_i}^T x \tag{2.40}$$

b. Polynomial

$$K(x_i, x) = ({x_i}^T x + 1)^d \tag{2.41}$$

c. Radial *basis Function*

$$K(x_i, x) = exp\left(-\|x \quad - x_i\|_2^2 \quad / \sigma^2\right) \tag{2.42}$$

d. Sigmoid *kernel*

$$K(x_i, x) = \tanh[k{x_i}^T x + \theta] \tag{2.43}$$

Hyperplane yang menjadi pemisah terbaik dapat ditemukan dengan mengukur margin *hyperplane* dan mencari titik maksimalnya. *Margin* adalah jarak antara *hyperplane* dengan *pattern* terdekat dari masing-masing kelas. *Pattern* yang paling dekat tersebut disebut *Support Vector*. Untuk pembahasan suatu kasus yang dapat dipisahkan secara linier, maka dalam hal ini fungsi pemisah yang dicari adalah fungsi linier. Fungsi tersebut dapat didefinisikan sebagai:

$$g(x) \coloneqq sqn(f(x)) \tag{2.44}$$

dengan

$$f(x) = WT\ X + b \tag{2.45}$$

Masalah klasifikasi ini dapat dirumuskan berikut: kita akan menemukan set parameter (*w*,*b*) sehingga *f*(*x*) = < *w*,*x*> + *b* = *yi*, untuk semua *i*. Dalam teknik ini kita berusaha menemukan fungsi pemisah (*classfiier*/*hyperplane*) terbaik diantara fungsi yang tidak terbatas jumlahnya untuk memisahkan dua macam objek.

Hyperplane terbaik adalah *hyperplane* yang terletak ditengah-tengah antara dua set objek dari dua kelas. Mencari *hyperplane* terbaik ekuivalen dengan memaksimalkan margin atau jarak antara dua set objek dari kelas yang berbeda. Jika WX_1 + b = +1 adalah *hyperplane* pendukung dari kelas +1 (WX_1 + b = +1) dan WX_2 + b = -1 *hyperplane* pendukung dari kelas -1 (WX_2 + b = -1), margin antara dua kelas dapat dihitung dengan mencari jarak antara kedua *hyperplane-hyperplane* pendukung dari kedua kelas. Secara spesifik *margin* dihitung dengan cara berikut:

$$(WX_1 + b = +1) - (WX_2 + b = -1) = W(X_1 - X_2) = 2 = > \quad (2.46)$$

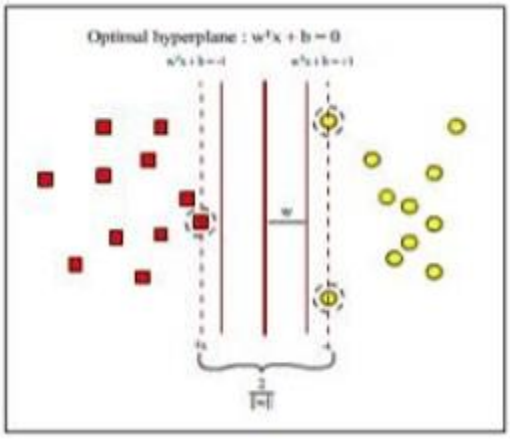
$$LD = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i,j} a_i a_j y_i y_j \, (x_i{}^T x + 1)^d \quad (2.47)$$

$$syarat\ 1\text{: } \sum_{i=1}^{n} a_i y_i = 0$$

$$syarat\ 2\text{: } a_i \geq 0, i = 1,2, \dots, N$$

xi.xj merupakan *dot product* dua data dalam data latih. *Hyperplane* (batas keputusan atau pemisah).

Gambar 2.3 menunjukan bagaimana SVM bekerja untuk menentukan suatu fungsi pemisah dengan margin yang maksimal [16].



Gambar 2.3 Mencari Fungsi Pemisah Yang Optimal Untuk Objek Yang Dapat Dipisahkan Secara Linier

Untuk membuktikan bahwa memaksimalkan margin antara dua set objek akan meningkatkan probabilitas pengelompokkan secara benar dari data testing. Pada dasarnya jumlah fungsi pemisah tidak terbatas jumlahnya, misalnya dari jumlah yang tak terbatas kita ambil dua fungsi, yaitu $f_1(x)$ dan $f_2(x)$. Fungsi f_1

memiliki margin yang lebih besar daripada f_2. Setelah menemukan dua fungsi ini, sekarang suatu data baru masuk dengan keluaran -1. Maka kita harus mengelompokkan apakah data ini ada dalam kelas -1 atau +1 menggunakan fungsi pemisah yang sudah kita temukan. Dengan mengguankan f_1, kita akan kelompokkan data baru ini di kelas -1 yang berarti kita benar mengelompokkannya. Kemudian dengan f_2 kita akan menempatkannya di kelas +1 yang berarti salah. Dari contoh sederhana ini kita lihat bahwa memperbesar *margin* bisa meningkatkan probabilitas pengelompokkan suatu data secara benar.

2.6.1 *Support Vector Machine* Untuk Multi-Kelas

Pada awal dikembangkannya SVM pendekatan ini digunakan untuk klasifikasi dua kelas. Pengembangan ke arah persoalan klasifikasi untuk multi kelas masih menjadi perhatian peneliti. Terdapat dua pendekatan utama kuntuk SVM multi kelas, yang pertama kita dapat menemukan dan menggabungkan beberapa fungsi pemisah persoalan klasifikasi dua kelas untuk menggabungkan beberapa fungsi pemisah persoalan klasifikasi dua kelas untuk menyelesaikan persoalan multi kelas. Kedua, secara langsung menggunakan semua data dari semua kelas dalam satu formulasi persoalan optimasi. Yang termasuk pada pendekatan pertama diama beberapa fungsi untuk problem dua kelas dikembangkan lalu digabung antara lain: satu-lawan-semua (*One-Against-All*, OAA), dan satu-lawan-satu (*One-Against-One*, OAO) . [16]

2.6.1.1 Metode Satu-Lawan-Semua (*One-Against-All*)

Dengan menggunakan metode ini, dibangun k sebuah model SVM biner (k adalah jumlah kelas). Setiap model klasifikasi ke-i dilatih dengan menggunakan keseluruhan data, untuk mencari solusi, terdapat permasalahan klasifikasi dengan 4 buah kelas. Untuk pelatihan digunakan 4 buah SVM biner seperti pada Tabel 2.1 dan penggunaannya dalam mengklasifikasi data baru dapat dilihat pada Gambar 2.4.

$$\begin{aligned}
&\min_{w^i,b^i,\xi^i} \frac{1}{2}\left(w^i\right)^T w^i + C\sum_t \xi_t^i \\
&s.t\ \left(w^i\right)^T \phi(x_t) + b^i \geq 1 - \xi_t^i \rightarrow y_t = i, \\
&\left(w^i\right)^T \phi(x_t) + b^i \geq -1 + \xi_t^i \rightarrow y_t \neq i, \\
&\xi_t^i \geq 0
\end{aligned} \tag{2.48}$$

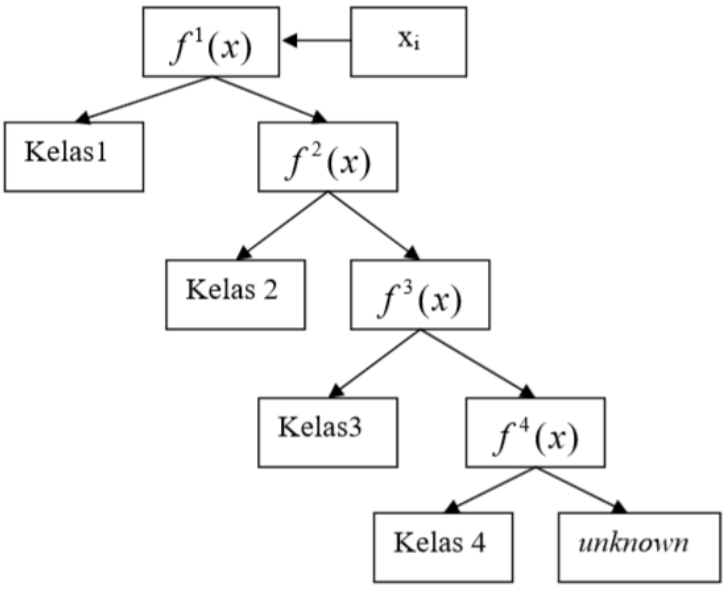
Gambar 2.4 Perhitungan Metode One Vs All

Tabel 2.1 Contoh 4 SVM Biner dengan Metode *One Vs All*

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

Konsep pada OAA yaitu dimisalkan pada kasus lima kelas, kelas 1, 2, 3, dan 4. Bila akan diujikan $\rho^{(1)}$, semua data dalam kelas 1 diberi label +1 dan data dari kelas lainnya diberi label -1. Pada $\rho^{(2)}$, semua data dalam kelas 2 diberi label +1 dan data dari kelas lainnya diberi label -1 dst hingga data terakhir. Kemudian dicari *hyperplane* dengan algoritma SVM du akelas. Maka akan didapat *hyperplane* untuk masing-masing kelas diatas. Kemudian kelas dari suatu data barus *x* ditentukan berdasarkan nilai terbesar dari *hyperplane* [19]:

$$kelas\ x = \arg\max_{l=1\ldots k} \left(\left(w^{(l)}\right)^T . \emptyset(x) + b^{(l)}\right) \tag{2.49}$$



Gambar 2.5 Contoh Klasifikasi dengan Metode *One Vs All*

2.6.1.2 Metode Satu-Lawan-Satu (*One-Against-One*)

Dengan menggunakan metode ini, dibangun persamaan:

$$\frac{k(k-1)}{2} \tag{2.50}$$

Keterangan:

k = jumlah kelas dalam SVM

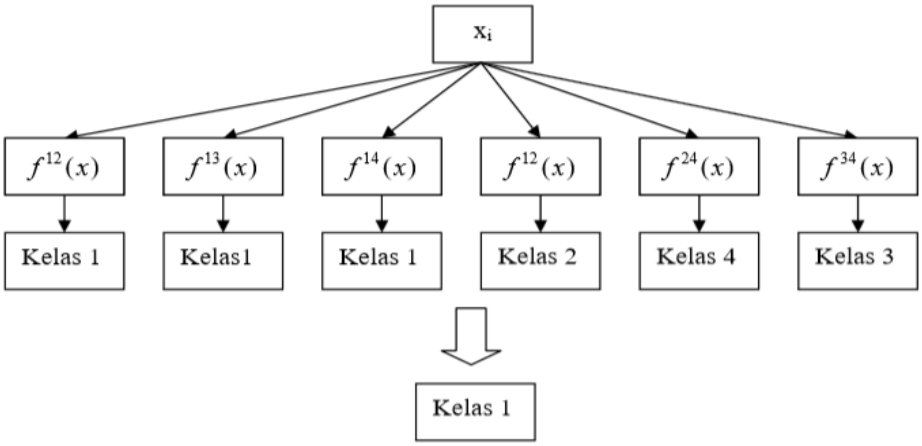
Setiap model klasifikasi dilatih pada data dari dua kelas. Untuk data pelatihan dari kelas ke-i dan kelas ke-j, dilakukan pencarian solusi untuk persoalan optimasi konstrain sebagai berikut:

$$\begin{aligned} &\min_{w^{ij},b^{ij},\xi^{ij}} \frac{1}{2}\left(w^{ij}\right)^T w^{ij} + C\sum_t \xi_t^{ij} \\ &s.t\ \left(w^{ij}\right)^T \phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij} \rightarrow y_t = i, \\ &\left(w^{ij}\right)^T \phi(x_t) + b^{ij} \geq -1 + \xi_t^{ij} \rightarrow y_t = j, \\ &\xi_t^{ij} \geq 0 \end{aligned} \tag{2.51}$$

Terdapat beberapa metode untuk melakukan pengujian setelah keseluruhan k(k-1)/2 model klasifikasi selesai dibangun. Salah satunya adalah metode voting [16].

Tabel 2.2 Contoh 6 SVM Biner dengan Metode One Vs 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	$f^{34}(x) = (w^{34})x + b^{34}$



Gambar 2.6 Contoh Klasifikasi dengan Metode *One Vs One*

Jika data *x* dimasukan kedalam fungsi hasil pelatihan:

$$f(x) = \left(w^{ij}\right)^T \emptyset(x) + b) \tag{2.52}$$

Dan hasilnya menyatakan *x* adalah kelas *i*, maka suara untuk kelas *i* ditambah satu. Kelas dari data *x* akan ditentukan dari jumlah suara terbanyak. Jika terdapat dua buah kelas yang jumlah suaranya sama, maka kelas yang indeksnya lebih kecil dinyatakan sebagai kelas dari data. Jadi pada pendekatan ini terdapat $\frac{k(k-1)}{2}$ buah persamaan *Quadratic Programming* yang masing-masing memiliki 2*n/k* variabel (*n* adalah jumlah data pelatihan). Contohnya, terdapat permasalahan klasifikasi dengan 4 buah kelas. [16]

2.7 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.

Unified Modelling Language (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]**

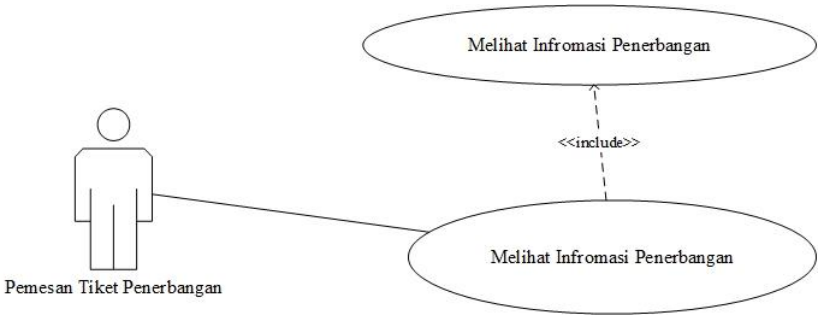
2.7.1 *Use Case* Diagram

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 *actor*dan sistem dimodelkan sebagai *use case*. Contoh diagram *use case* sederhana dapat dilihat pada Gambar 2.7.

Gambar 2.7 *Use Case* Diagram Sederhana

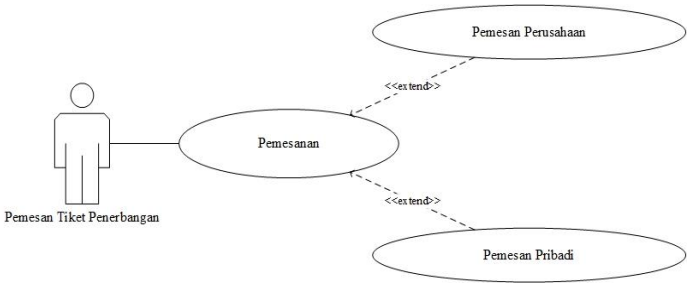
Dalam diagram *use case,* sering ditemukan asosiasi yang bertipe <*include*>. Asosiasi <*include*> ini menujukkan bahwa *use case* tertentu, misalnya "Melihat Informasi Penerbangan" harus dilakukan terlebih dahulu sebelum *use case* lainnya seperti "Memesan Kursi Pesawat" dilakukan. Terdapat asosiasi 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 2.8.



Gambar 2.8 *Use Case* Diagram dengan 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

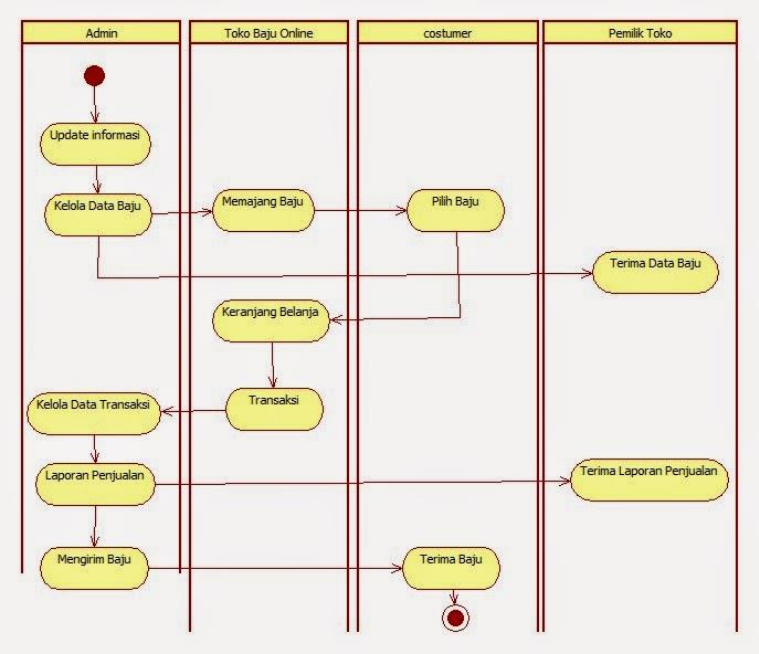
dari kedua *use case* ini harus dilaksanakan. Implementasi asosiasi <*extend*> dalam kasus tersebut pada diagram *use case* dapat dilihat Gambar 2.9.



Gambar 2.9 *Use Case* Diagram Dengan *Asosiasi Extend*

2.7.2 Activity Diagram

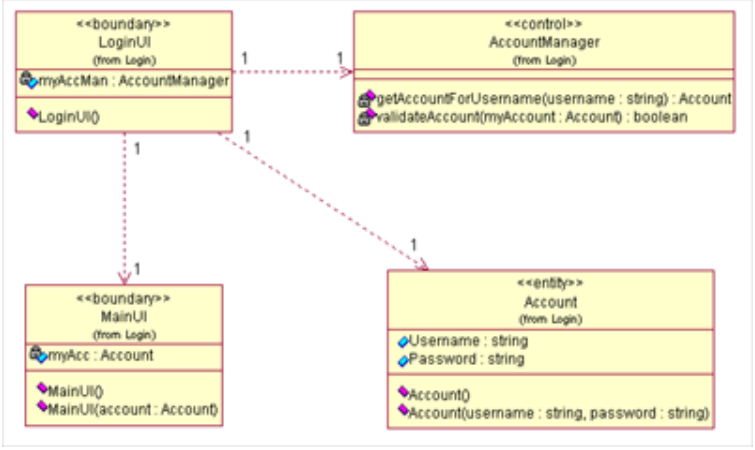
Sebuah *activity* diagram atau diagram aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana actor menggunakan sistem untuk melakukan aktivitas. Sama seperti state, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* 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 2.8.



Gambar 2.10 *Activity* Diagram

2.7.3 *Class* Diagram

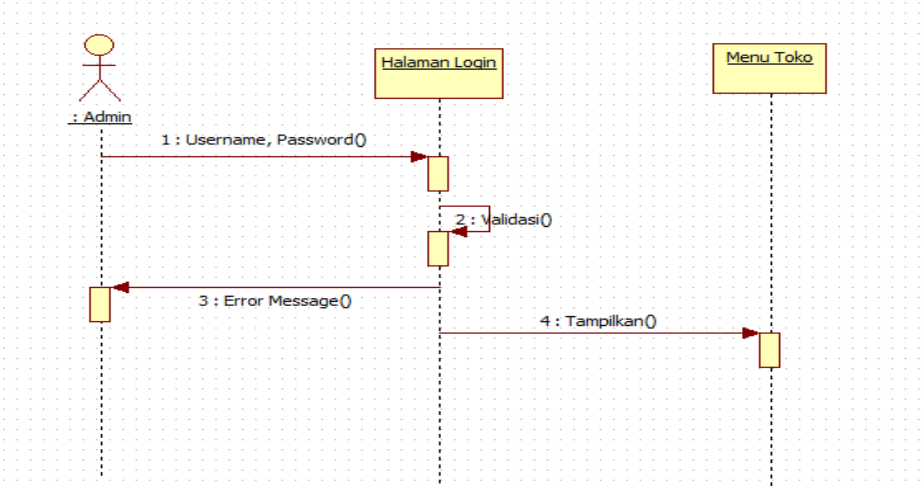
Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. *Class* diagram membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain, class 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 2.11.



Gambar 2.11 *Class* Diagram

2.7.4 *Sequence* Diagram

Sequence 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* diagram. Contoh *Sequence* diagram dapat dilihat pada Gambar 2.12.



Gambar 2.12 *Sequence* Diagram

2.8 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]

2.9 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. [23] Model confusion matrix untuk contoh 2 kelas dapat dilihat pada Tabel 2.3 sebagai berikut:

Tabel 2.3 Model *Confusion Matrix*

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

Keterangan:

TP = jumlah nilai positive yang diklasifikasikan positif.

TN = jumlah nilai negatif yang diklasifikasikan negatif.

FP = jumlah nilai positif yang diklasifikasikan negatif.

FN = jumlah nilai negatif yang diklasifikasikan positif.

Perhitungan untuk mendapatkan akurasi dapat dilihat pada persamaan 2.53 sebagai berikut:

$$\text{Akurasi} = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.53}$$

Perhitungan untuk mengetahui PPV(nilai prediksi positif) dapat dilihat pada persamaan 2.54 sebagai berikut:

$$\text{PPV} = \frac{TP}{TP+FP} \tag{2.54}$$

Perhitungan untuk mengetahui NPV(nilai prediksi negatif) dapat dilihat pada persamaan 2.55 sebagai berikut:

$$\text{NPV} = \frac{TN}{TN+FN} \tag{2.55}$$