BAB 2
LANDASAN TEORI

2.1. Ekstraksi Informasi

Dalam perkembangan yang begitu pesat dari infomasi sekarang, banyaknya informasi pada dokumen tidak terstruktur atau bebas yang menyebabkan sulitnya mendapatkan informasi yang di butuhkan secara langsung. Hal tersebut menghasilkan kebutuhan yang terus meningkat akan metode yang efektif dan efisien dalam menganalisa informasi dari dokumen teks bebas untuk menemukan informasi yang di butuhkan secara cepat, berdasarkan hal tersebut munculnya teknologi Ekstraksi Informasi menjadi salah satu solusi permasalahannya

Ekstraksi Informasi mengacu pada ekstraksi otomatis informasi terstruktur seperti entitas, hubungan antara entitas, dan atribut yang menggambarkan entitas dari sumber yang tidak terstruktur [1]. Berbagai penelitian dan algoritma telah banyak di kembangakan dalam ekstraksi informasi, beberapa algoritma dalam melakukan ekstraksi informasi seperti algoritma SVM (Support Vector Machine), LVQ (Learning Vector Quantization), KNN (K-Nearest Neigbour), Navie Bayes, rule-based dan masih banyak lagi. Setiap penelitian yang dilakukan memiliki hasil yang berfariasi dalam menggunakan algoritma dan kasus yang berbeda, hal itu menyebabkan terus berkembangnya teknologi dalam ekstraksi informasi. Untuk pengerjaan Tugas Akhir ini, metode yang akan di gunakan adalah SVM (Support Vector Machine) dengan penambahan fitur information gain dengan harapan memiliki akurasi yang lebih baik lagi.

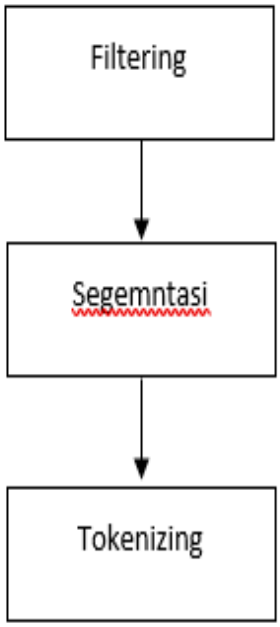
2.2. Dokumen Karya Tulis Ilmiah

Karya tulis ilmiah merupakan bagian yang sangat penting dalam dunia pendidikan dan penelitian. Karya tulis ilmiah merupakan hasil dari berbagai macam riset yang telah dilakukan guna menambah pengetahuan baru bagi dunia. Adapun, yang dimaksud dengan kaidah-kaidah keilmuan adalah bahwa karya ilmiah menggunakan metode ilmiah di dalam membahas permasalahan , menyajikan

kajiannya menggunakan bahasa baku dan tata tulis ilmiah, serta menggunakan prinsipprinsip keilmuan yang lain seperti: bersifat objektif, logis, empiris (berdasarkan fakta), sistematis, lugas, jelas, dan konsisten [8].

2.3. Text Preprocessing

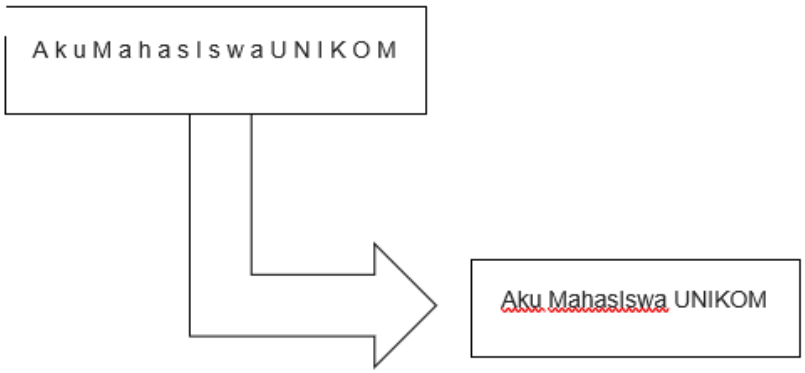
Dalam proses ekstraksi informasi dengan sumber informasi yang tidak terstruktur memberikan kesulitan dalam proses komputerisasi secara otomatis, maka dari itu di butuhkannya suatu proses yang akan merubah dokumen yang memiliki format tidak terstruktur menjadi terstruktur. Proses ini di biasa di sebut *Text Preprocessing* [9]. Dalam prosesnya mengubah dokumen tidak terstruktur menjadi terstruktur dengan memberikan data sebuah nilai-nilai numeric. Setelah data menjadi terstruktur dan memiliki nilai-nilai numeric maka data dapat di olah lebih lanjut. Beberapa proses yang akan digunakan dalam *Text Preprocessing* adalah *Filtering*, *segmentasi* dan *filtering*. Berikut gambaran tahapan *Preprocessing* yang dapat dilihat pada gambar 2.1 :



Gambar 2.1 Tahapan Preprocessing

2.3.1. Filtering

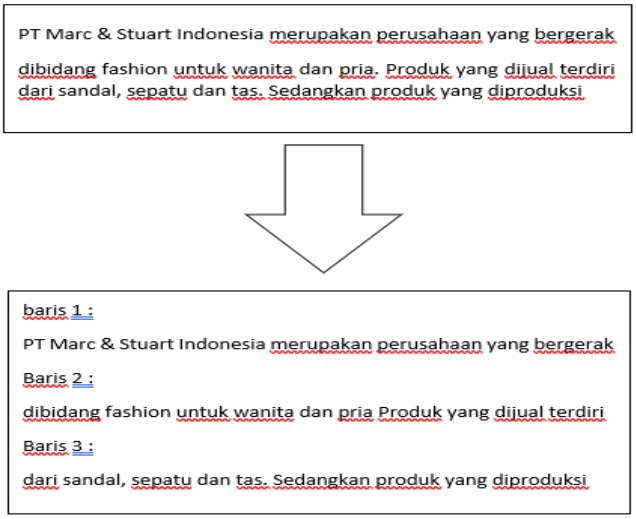
Filtering adalah tahapan membuang tanda baca yang tidak dibutuhkan pada teks dokumen sehingga data menjadi lebih optimal, tanda baca yang akan di buang seperti spasi (“ ”) , (?), (!) dan tanda baca titik (“.”). Proses filtering ini bertujuan untuk menghilangkan karakter-karakter yang dianggap tidak berguna dan tidak diperlukan dalam tahapan selanjutnya. Proses filtering dapat dilihat pada gambar 2.2.



Gambar 2.2 Ilustrasi Filtering

2.3.2. Segmentasi

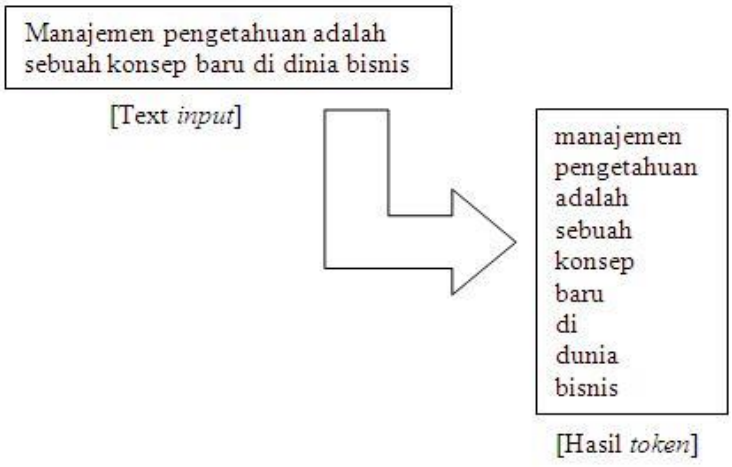
Segmentasi data adalah proses yang digunakan untuk menghapus baris kosong dan menghasilkan data segmen per baris. Berikut blok diagram segmentasi data Ilustrasi dari *segmentasi* dapat di lihat pada gambar 2.3.



Gambar 2.3 Ilustrasi Segmentasi

2.3.3. Tokenizing

Tokenizing merupakan teknik dalam memecah kalimat menjadi satuan terkecilnya yaitu kata, huruf, atau symbol [10]. Dalam peroses ini setiap kalimat, paragraf atau dokumen akan di bagi mejadi token-token tertentu, sebagai contoh tokenisasi dari kalimat "Aku Mahasiswa UNIKOM" akan menghasilkan 3 token, yaitu : "Aku", "Mahasiswa", "UNIKOM". Ilustrasi dari *tokenizing* dapat di lihat pada gambar 2.4.



Gambar 2.4 Ilustrasi Tokenizing

2.4. Ekstraksi Fitur

Ekstraksi Fitur merupakan proses untuk mencari nilai - nilai fitur yang terkandung dalam dokumen [11]. Fitur dapat diartikan sebagai ciri dari setiap data yang dikenali oleh sistem sehingga menghasilkan nilai fitur, untuk setiap kalimat dalam dokumen, skor kalimat dihitung berdasarkan fitur ekstraksi. Fitur-fitur yang akan digunakan berjumlah 14 fitur, untuk 13 fitur merujuk pada penelitian yang dilakukan oleh Firdamdam [3] dan Aditya [4], sedangan untuk fitur LINE merupakan fitur yang dibuat oleh penulis. Dalam penjelasan fitur-fitur yang akan digunakan dalam penelitian ini dapat dilihat pada **Tabel 2.1**.

Tabel 2.1 Keterangan Ekstraksi Fitur

No	Nama Fitur	Keterangan
1	**INITCAPS**	Mengenali setiap token yang hurufnya diawali dengan kapital.
2	**ALLCAPS**	Mengenali setiap token yang semua hurufnya kapital.
3	**CONTAINSDIGIT**	Mengenali setiap token yang mengandung angka.
4	**ALLDIGIT**	Mengenali setiap token yang semuanya angka
5	**CONTAINSDOTS**	Mengenali setiap token yang mengandung titik.
6	**LOWERCASE**	Mengenali setiap token yang semuanya huruf kecil.
7	**PUNCTUATION**	Mengenali setiap token yang mengandung tanda tertentu seperti titik, koma, titik dua, titik koma, tanda kurung, dan tanda seru.
8	**EIGHTDIGIT**	Fitur tambahan pada penelitian ini, fitur ini dikhususkan untuk mengenali token yang memiliki digit dengan panjang 8 digit.
9	**WORD**	Fitur tambahan pada penelitian ini, fitur ini dikhususkan untuk memberikan bobot pada token

		untuk kelas JENIS_PENELITI AN dan KALIMAT_PENG AJUAN. 10
10	**LINE_START**	Mengenali posisi token pada indeks array awal.
11	**LINE_IN**	Mengenali posisi token pada indeks array tengah.
12	**LINE_END**	Mengenali posisi token pada indeks array akhir
13	**LINE**	Mengenali posisi setiap token
14	**YEAR**	Mengenali ciri token tahun.

2.5. Feature Selection

Feature Selection adalah proses pemilihan fitur yang relevan, atau subset calon fitur. Kriteria evaluasi yang digunakan untuk mendapatkan bagian fitur optimal [7]. Hal ini bertujuan untuk memilih subset kecil dari fitur yang relevan dari yang asli menurut kriteria evaluasi relevansi tertentu, yang biasanya menyebabkan kinerja learning yang lebih baik (misalnya, akurasi yang lebih tinggi untuk klasifikasi), biaya komputasi yang lebih rendah, dan model interpretability yang lebih baik. Penerapan metode fiture selection digunakan untuk mengurangi dimensi dari set fitur dengan menghapus fitur yang tidak relevan.

Feature selection memiliki sejumlah keunggulan seperti ukuran dataset yang lebih kecil, menyusutnya ruang pencarian, dan kebutuhan komputasi ynag lebih rendah. Tujuannya adalah pengurangan ukuran dimensi untuk menghasilkan peningkatan akurasi klasifikasi [7]. Metode untuk fiture subset selection untuk klasifikasi dokumen teks menggunakan fungsi evaluasi yang diterapkan untuk satu kata. Scoring dari kata-kata individu (Fitur individu terbaik) dapat dilakukan menggunakan beberapa tindakan, salah satunya information gain (IG). Metode fiture-scoring ini, peringkat fitur dan skor ditentukan secara independent.

2.5.1. Information Gain

Information gain merupakan ekspektasi dari pengurangan entropi yang dihasilkan dari partisi objek dataset berdasarkan fitur tertentu [7].
Entropi dari koleksi label benda S didefinisikan pada Persamaan (2.1) :

$$Entropy\ (\mathrm{S}) = \sum_{i=1}^{c} -p_i log_2 p_i \qquad (2.1)$$

Dimana Pi adalah proporsi S milik kelas i. Berdasarkan ini, persamaan matrik IG seperti berikut.

Persamaan proporsi P (2.2) :

$$P(E) = \frac{x}{n} \qquad (2.2)$$

Keterangan :

$P = proporsi\ sample$

$E = suatu\ kondisi\ setiap\ sample$

$x = banyaknya\ produk\ pada\ setiap\ sample$

$n = jumlah\ seluruh\ sample$

Persamaan *IG* (2.3) :

$$IG(S.A) = Entropy\ (\mathrm{S}) - \sum cevalues(A) \frac{|S_v|}{|S|} Entropy(S_v) \qquad (2.3)$$

Dimana nilai-nilai (A) adalah himpunan nilai untuk fitur A, S himpunan contoh training, Sv himpunan objek training set dimana A memiliki v.

2.6. Support Vector Machine (SVM)

Support Vector Machine (SVM) adalah salah satu metode yang dapat digunakan dalam ekstraksi informasi. Support Vector Machine (SVM) dikembangkan oleh Boser, Guyon, dan Vapnik, pertama kali diperkenalkan pada tahun 1992 di *Annual Workshop on Computational Learning Theory*. Pemahan

dalam cara kerja SVM adalah memisahkan dua buah kelas yang terpisah secara linier dengan membuat sebuah garis pemisah yang disebut hyperplane [12]. Dalam SVM dikenal istilah margin, yaitu jarak antara garis hyperplane dengan data yang paling dekat yang disebut dengan support vector [4]. Usaha untuk mencari lokasi *hyperplane* ini merupakan inti dari proses pelatihan pada svm [12].

Support Vector Machine (SVM) memiliki beberapa tahap dalam pengerjaannya, pada tahap awal yaitu pendefinisan persamaan suatu *hyperplane* pemisah. Hyperplane adalah sebuah garis lurus atau bidang mendatar yang memisahkan kelas-kelas.

Persamaan (2.4) Hyperplane :

$$f(x) = \vec{w} \,.\, \vec{x} + b \tag{2.4}$$

dimana w merupakan suatu bobot vektor, yaitu $\{w_{1,} w_{2,} \dots, w_n\}$ n adalah jumlah atribut dan b merupakan suatu skalar yang disebut dengan bias. Jika berdasarkan pada atribut A1, A2 dengan permisalan tupel pelatihan X = $(x_{1,} x_2)$, x1 dan x2 merupakan nilai dari atribut A1 dan A2.

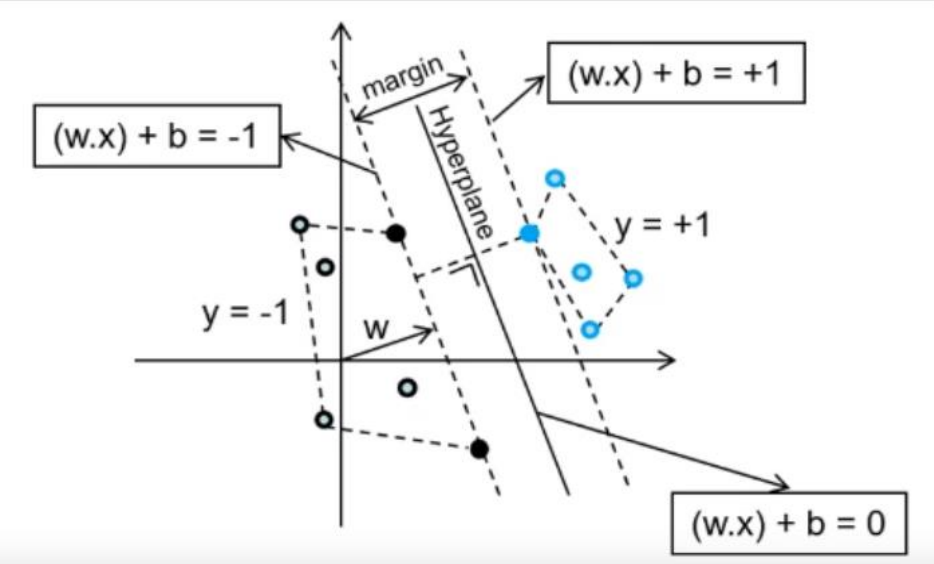
Sehingga diperoleh persamaan -1 (sample negatif) memenuhi pertidaksamaan (2.5)

$$\vec{w} \,.\, \vec{x} + b \leq -1 \tag{2.5}$$

Dan kelas +1 pattern yang memenuhi pertidaksamaan (2.6) :

$$\vec{w} \,.\, \vec{x} + b \geq +1 \tag{2.6}$$

Pada Gambar 1.5 merupakan ilustrasi Hyperplane .



Gambar 2.5 Ilustrasi Hyperplane

Margin terbesar dapat dicari dengan cara memaksimalkan jarak antar bidang pembatas kedua kelas dan titik terdekatnya, yaitu 2/|w|. Hal ini dirumuskan sebagai permasalahan quadratic programming (QP) problem yaitu mencari titik minimal persamaan (2.7) dengan memperhatikan persamaan (2.8) berikut:

$$\min\ \tau(w) = \frac{1}{2}||w||^2 \tag{2.7}$$

$$y_i\ (w * x_i + b) - 1 \geq 0, (i = 1, .., n) \tag{2.8}$$

Permasalahan ini ini dapat dipecahkan dengan berbagi teknik komputasi. Lebih mudah diselesaikan dengan mengubah persamaan (2.7) ke dalam fungsi *Lagrangian* pada persamaan (2.9) berikut:

$$L\,p = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i y_i (w^\tau x_i + b) - 1 \tag{2.9}$$

α_i adalah *Lagrange Multiplier* yang berkorespondensi dengan x_i. Nilai α_i adalah nol atau positif:

Untuk meminimalkan *Lagrangian*, Persamaan (2.9) harus diturunkan terhadap w dan b, dan diset dengan nilai nol untuk syarat optimasi di atas:

Syarat 1:

$$\frac{\partial Lp}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{N} \alpha_i y_i x_i \quad (2.10)$$

Syarat 2:

$$\frac{\partial Lp}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{N} \alpha_i y_i = 0 \quad (2.11)$$

N adalah jumlah data yang menjadi *support vector*.

Karena *Lagrange Multiplier* (α) tidak diketahui nilainya, persamaan di atas tidak dapat diselesaikan secara langsung untuk mendapatkan w dan b. Untuk menyelesaikan masalah tersebut, modifikasi Persamaan 2.9 diatas menjadi kasus memaksimalkan dengan syarat optimal untuk dualitasnya menggunakan konstrain Karush-Kuhn-Tucker (KKT) sebagai berikut:

Syarat 1:

$$\alpha_i [y_i(\boldsymbol{w}.\boldsymbol{x_i} + b) - 1] = 0 \quad (2.12)$$

Syarat 2:

$$\alpha_i > 0, i = 1,2, \dots, N \quad (2.13)$$

Dengan menerapkan konstrain pada Persamaan (2.12) dan (2.13) maka dipastikan bahwa nilai *Lagrange Multiplier* sama banyaknya dengan data latih, meskipun sebenarnya banyak dari data latih yang *Lagrange Multiplier* sama dengan nol (karena hanya beberpa saja yang akan menjadi *support vector*) ketika menerapkan syarat pertama. Konstrain diatas menyatakan bahwa *Lagrange Multiplier* α_i harus nol kecuali untuk data latih x_i yang memenuhi persamaan:

$$y_i(\boldsymbol{w}.\boldsymbol{x_i} + b) = 0 \quad (2.14)$$

Data latih tersebut, dengan $\alpha_i > 0$, terletak pada *hyperplane* b_{i1} atau b_{i2}, dan disebut *support vector*. Data latih yang tidak terletak di *hyperplane* tersebut

mempunyai $\alpha_i = 0$,. Persamaan 2.16 dan 2.17 juga menyarankan parameter *w* dan b yang mendefinisikan *hyperplane* hanya tergantung *support vector*.

Masalah optimasi di atas masih sulit diselesaikan karena banyaknya parameter (w, b dan α_i). Untuk menyederhanakannya, persamaan optimasi 2.9 diatas harus ditransformasi ke dalam fungsi *Lagrange Multiplier* itu sendiri (disebut dualitas masalah).

Persamaan *Lagrange Multiplier* 2.15 dapat dijabarkan menjadi:

$$Lp = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i y_i (\boldsymbol{w}.\boldsymbol{x_i}) - b\sum_{i=1}^{N} \alpha_i y_i + \sum_{i=1}^{N} \alpha_i \quad (2.15)$$

Syarat optimal (2.11) ada dalam suku ketiga di ruas kanan dalam persamaan (2.15), dan memaksa suku ini menjadi sama dengan nol. Dengan mengganti *w* dari syarat (2.10), dan suku $\|w\|^2 = \boldsymbol{w_i}.\boldsymbol{w_j}$, maka persamaan di atas akan berubah menjadi dualitas *Lagrange Multiplier* berupa Ld dan didapatkan:

Maksimalkan:

$$Ld = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j}^{N} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i}.\boldsymbol{x_j} \quad (2.16)$$

$\boldsymbol{x_i}.\boldsymbol{x_j}$ merupakan *dot-product* dua data dalam data latih.

Syarat 1:

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad (2.17)$$

Syarat 2:

$$\alpha_i > 0,\ i = 1,2,\ldots,N \quad (2.18)$$

Untuk set data yang besar, masalah dualitas optimasi tersebut (2.16, 2.17, 2.18) dapat diselesaikan dengan metode numerik seperti *Quadratic Programming*. Sekali α_i didapatkan, persamaan (2.10) dan (2.11) bisa digunakan untuk mendapatkan solusi layak untuk *w* dan b.

Hyperplane (batas keputusan) didapatkan dengan formula:

$$\left(\sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x_i}.\boldsymbol{z}\right) + b = 0 \tag{2.19}$$

N adalah jumlah data yang menjadi *support vector*, x_i merupakan *support vector*, *z* merupakan data uji yang akan diprediksi kelasnya, dan $\boldsymbol{x_i}.\boldsymbol{z}$ merupakan *inner-product* antara x_i dan z. Untuk nilai *b* didapatkan dari persamaan (2.12) pada *support vector*. Karena α_i dihitung dengan teknik metode numerik dan mempunyai *error* numerik, nilai yang dihitung untuk *b* bisa jadi tidak sama. Hal ini disebabkan oleh *support vector* yang digunakan dalam persamaan (2.12), biasanya diambil nilai rata-rata dari *b* yang didapat untuk menjadi parameter *hyperplane*. Untuk persamaan (2.12) dalam mendapat dapat *b* dapat disederhanakan menjadi:

$$b_i = 1 - y_i(\boldsymbol{w}.\boldsymbol{x_i}) \tag{2.20}$$

Penjelasan di atas berdasarkan asumsi bahwa kedua kelas dapat terpisah secara sempurna oleh *hyperplane*. Akan tetapi, pada umumnya kedua kelas tersebut tidak dapat terpisah secara sempurna. Hal ini menyebabkan proses optimalisasi tidak dapat diselesaikan karena tidak ada *w* dan *b* yang memenuhi pertidaksamaan 2.8. Untuk itu pertidaksamaan tersebut dimodifikasi dengan memasukan variabel slack ξ_i ($\xi_i \geq 0$), Menjadi:

$$y_i(\boldsymbol{w}.\boldsymbol{x_i} + b) \geq 1 - \xi_i \tag{2.21}$$

Demikian juga untuk masalah persamaan(2.7):

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \quad (2.22)$$

Parameter *C* berguna untuk mengontrol *trade-off* antara margin dan *error* klasifikasi. Semakin besar nilai *C* maka semakin besar pula pelanggaran yang dikenakan untuk tiap klasifikasi [13].

Metode untuk mengoptimalisasi *hyperplane* SVM umumnya dipakai untuk menyelesaikan *Quadratic Programming* dengan konstrain yang ditetapkan. Beberapa pilihan metode yang bisa digunakan adalah *chunking* (Vapnik, 1982), metode dekomposisi (Osuna *et al*, 1997), dan *Sequential Minimal Optimization* (SMO)(Plat, 1999).

SVM sebenarnya adalah *hyperplane* linear yang hanya bekerja pada data yang dapat dipisahkan secara linear [13]. Untuk data yang distribusi kelasnya tidak linear biasanya digunakan pendekatan kernel pada fitur data dari awal set data. Kernel dapat di definisikan sebagai suatu fungsi yang memetakan fitur data dari dimensi awal (rendah) ke fitur lain yang berdimensi lain yang lebih tinggi (bahkan jauh lebih tinggi) [13].

Berikut ini adalah beberapa fungsi kernel yang umum digunakan yaitu:

a. Kernel linier

   K (xi,x) = xiTx

b. Polynomial

   K (xi,x) = (γ.xiTx+r)p ,γ > 0

c. Radial basis function (RBF)

   K(xi,x)=exp(-γ | xi-x | 2),γ>0

d. Sigmoid kernel

   K(xi,x)= tanh (γxiT+r).

Keterangan:

x adalah pasangan dua data dari semua bagian data latih. Parameter $\gamma > 0$, merupakan konstanta. $\|x_i - x\ \|^2$merupakan kuadrat jarak antara vektor x_i dan x .

2.6.1. SVM Multiclass Dengan "one-againts-all"

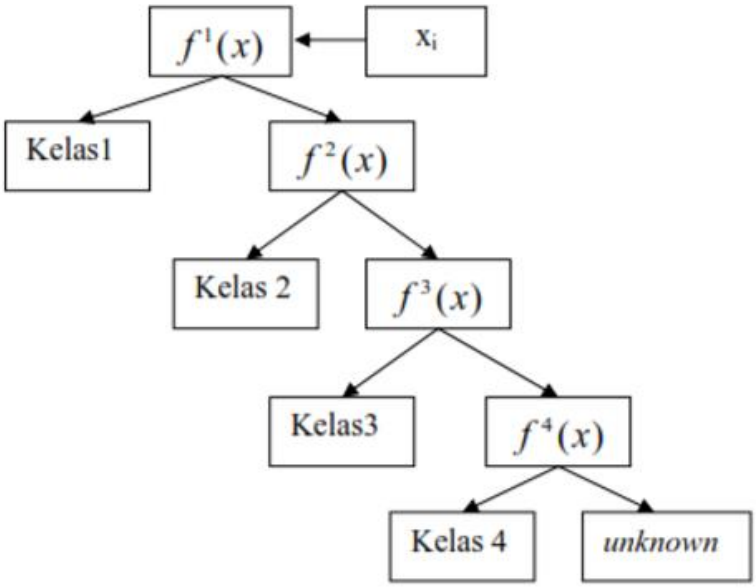
SVM saat pertama kali diperkenalkan oleh Vapnik, hanya dapat mengklasifikasikan data ke dalam dua kelas (klasifikasi biner). SVM hanya dapat melakukan klasifikasi biner (dua kelas), sementara masalah di dunia nyata umumnya mempunyai banyak kelas seperti pengenalan karakter, pengenalan wajah, atau diagnosis pasien, di mana data masukan terbagi menjadi lebih dari dua kelas [13]. Namun, penelitian lebih lanjut untuk mengembangkan SVM sehingga bisa mengklasifikasi data yang memiliki lebih dari dua kelas, terus dilakukan. Ada 3 Pendekatan SVM Multikelas yaitu *one-against-all* (OAA), *one-against-one* (OAO) dan *error correcting output code* (ECOC) (Tan *et al*, 2005). Tetapi yang akan dijelaskan pada penelitian ini adalah SVM Multikelas *one-against-all* (OAA).

Dengan metode ini, dibangun k buah model SVM biner (k adalah jumlah kelas). Setiap model klasifikasi ke-i dilatih dengan menggunakan keseluruhan data, untuk mencari solusi permasalahan [10]. Contohnya, terdapat permalasahan klasifikasi dengan 4 buah kelas. Untuk pelatihan digunakan 4 buah SVM biner dapat dilihat pada Tabel 2.2.

Tabel 2.2 Contoh 4 SVM biner dengan metode one-against-all

Yi = +1	Yi = -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	$f^4(x) = (w^4)x + b^4$

Untuk Ilustrais dari metode One-Againt-All dapat dilihat pada gambar 2.6.



Gambar 2.6 Klasifikasi Dengan Metode One-Against-All

Penelitian lebih lanjut untuk mengembangkan SVM sehingga dapat melakukan klasifikasi lebih dari dua kelas yaitu multi class SVM. Dalam klasifikasi kasus multikelas SVM, *hyperplane* yang terbentuk adalah lebih dari satu. Yang umum digunakan untuk mengimplementasikan multikelas SVM adalah pendekatan metode *One Against All*(OAA).

Konsep pada OAA yaitu dimisalkan pada kasus lima kelas, kelas 1, 2, 3, 4 dan 5. Bila akan diujikan ρ(1), semua data dalam kelas 1 diberi label +1 dan data dari kelas 2, 3, 4 dan 5 diberi label -1. Pada ρ(2), semua data dalam kelas 2 diberi label +1 dan data dari kelas 1, 3, 4 dan 5 diberi label -1. Pada ρ(3), semua data dalam kelas 3 diberi label +1 dan data dari kelas 1, 2, 3 dan 4 diberi label -1. Pada ρ(4), semua data dalam kelas 4 diberi label +1 dan data dari kelas 1, 2, 3, 4 diberi label -1. Begitu juga untuk ρ(5), semua data dalam kelas 5 diberi label +1 dan data dari kelas 1, 2, 3 dan 4 diberi label -1. Kemudian dicari *hyperplane* dengan algoritma SVM dua kelas. Maka akan didapat *hyperplane* untuk masing-masing kelas di atas. Kemudian kelas dari suatu data baru x ditentukan berdasarkan nilai terbesar dari *hyperplane*.

$$kelas\ x = \max_{\ell=1....k} \left(\left(w^{(\ell)} \right)^T . \Phi(x) + b^{(\ell)} \right) \qquad (2.23)$$

2.7. Metode Waterfall

Metode penelitian yang diterapkan pada penelitian ini adalah dengan pengembangan metode waterfall. Model ini mengusulkan sebuah pendekatan pada kepada pengembangan software yang sistematik dan skuensial. Menurut [14] Metode Waterfal memiliki tahapan tahapan sebagai berikut :

1. Requirements analysis and definition

   Layanan sistem, kendala, dan tujuan ditetapkan oleh hasil konsultasi dengan pengguna yang kemudian didefinisikan secara rinci dan berfungsi sebagai spesifikasi sistem.

2. System and software design

   Tahapan perancangan sistem mengalokasikan kebutuhan-kebutuhan sistem baik perangkat keras maupun perangkat lunak dengan membentuk arsitektur sistem secara keseluruhan. Perancangan perangkat lunak melibatkan identifikasi dan penggambaran abstraksi sistem dasar perangkat lunak dan hubungannya.

3. Implementation and unit testing

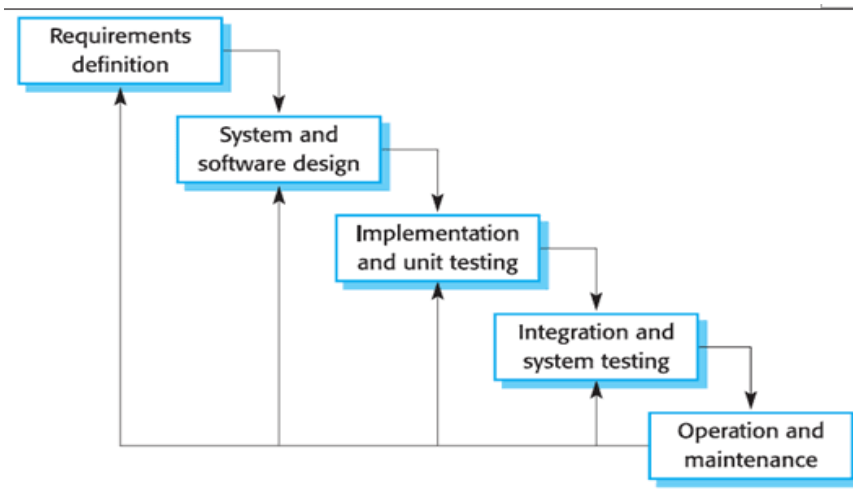
   Pada tahap ini, perancangan perangkat lunak direalisasikan sebagai serangkaian program atau unit program. Pengujian melibatkan verifikasi bahwa setiap unit memenuhi spesifikasinya.

4. Integration and system testing

   Unit-unit individu program atau program digabung dan diuji sebagai sebuah sistem lengkap untuk memastikan apakah sesuai dengan kebutuhan perangkat lunak atau tidak. Setelah pengujian, perangkat lunak dapat dikirimkan ke customer Operation and maintenance Biasanya (walaupun tidak selalu), tahapan ini merupakan tahapan yang paling panjang. Sistem dipasang dan digunakan secara nyata.

5. Operation and Maintenance

melibatkan pembetulan kesalahan yang tidak ditemukan pada tahapan-tahapan sebelumnya, meningkatkan implementasi dari unit sistem, dan meningkatkan layanan sistem sebagai kebutuhan baru



Gambar 2.7 Ilustrasi Metode Waterfall

Gambar 2.7 adalah bagan metode waterfall yang merupakan metode pengembangan system yang digunkan pada penelitian ini.

2.8. Unified Modeling Language (UML)

Unified Modelling Language (UML) adalah sebuah pemodelan standar dalam industri untuk visualisasi, merancang, dan mendokumentasikan sistem piranti lunak. Dengan menggunakan UML dapat membuat model untuk semua jenis aplikasi piranti lunak, aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi, dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. UML memuat diagram-diagram pemodelan sistem yang terdiri dari Use Case Diagram (diagram kasus), Class Diagram (diagram kelas), Activity Diagram (diagram aktivitas), Sequence Diagram (diagram urutan) [15].

2.8.1. Usecase Diagram

Use Case Diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use Case Diagram* adalah diagram yang menggambarkan kebutuhan sistem dari sudut pandang *user*, yang memperlihatkan hubungan-hubungan yang terjadi antara Aktor dengan *use case* dalam sistem [15].

2.8.2. Skenario Use Case

Skenario use case menjelaskan masing-masing use case yang terdapat pada Use Case Diagram. Penjelasan tersebut berkaitan dengan reaksi atau tanggapan dari sistem terhadap suatu aksi yang diberikan oleh aktor. Setiap use case memiliki skenario normal dan skenario alternatif [15].

2.8.3. Activity Diagram

Activity Diagram menggambarkan alir aktivitas pengguna dengan sistem, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. Activity Diagram tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar sub-sistem) secara eksak, tetapi lebih menggambarkan proses dan jalur-jalur aktivitas dari level atas secara umum [15].

2.8.4. Sequence Diagram

Sequence Diagram menggambarkan interaksi antar obyek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence Diagram* terdiri dari antara dimensi vertikal (waktu) dan dimensi horizontal (obyek-obyek yang terkait). *Sequence Diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah langkah yang dilakukan sebagai *respond* dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang me-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan [15].

2.8.5. Class Diagram

Class Diagram adalah sebuah spesifikasi yang jika diinstansi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut atau properti) suatu sistem, serta menawarkan layanan untuk memanipulasi keadaan tersebut (metoda

atau fungsi). Class Diagram digambarkan struktur dan deskripsi Class, package dan objek beserta hubungan satu sama lain seperti containment, pewaris, asosiasi dan lain-lain [15].

2.9. MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: database management system) atau DBMS yang multithread, multi-user, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

2.10. PHP: Hypertext Preprocessor

PHP adalah *bahasa pemrograman script server-side yang didesain untuk pengembangan web. Selain itu, PHP juga bisa digunakan sebagai bahasa pemrograman umum.* PHP disebut bahasa pemrograman **server side** karena PHP diproses pada komputer server. PHP dapat digunakan dengan gratis dan bersifat *Open Source*. PHP dirilis dalam lisensi *PHP License***,** sedikit berbeda dengan lisensi *GNU General Public License (GPL)* yang biasa digunakan untuk proyek *Open Source. U*ntuk pembuatan web, kode PHP biasanya di sisipkan kedalam dokumen HTML. Karena fitur inilah PHP disebut juga sebagai Scripting Language atau bahasa pemrograman script

2.11. Black Box Testing

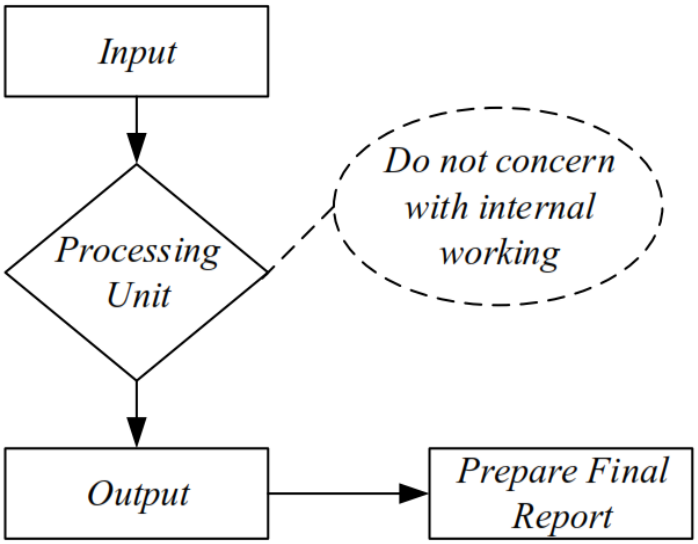
Black Box Testing merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, *tester* dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program [16]. Dalam black box testing cenderung untuk menemukan hal-hal berikut.

a. Fungsi yang tidak benar atau tidak ada.
b. Kesalahan antarmuka (*interface errors*).
c. Kesalahan pada struktur data dan akses basis data.

d. Kesalahan performa (*performance errors*).

e. Kesalahan inisialisasi dan terminasi.

Berikut pada Gambar 2.7 gambaran dari *black box testing*, dimana *user* menyatakan masukan dan keluaran pada setiap unit proses dan didapatkan hasil dari setiap proses tersebut apakah sesuai dengan yang dinyatakan atau tidak.



Gambar 2.8 Gambaran Pengujian Black Box [16].

2.12. Confusion Matrix

Confusion matrix merupakan sebuah metode perhitungan yang digunakan untuk mencari keakuratan pada hasil klasifikasi [17]. Pada dasarnya *confusion matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan klasifikasi yang seharusnya.

Pada pengukuran kinerja menggunakan *Confusion Matrix*, terdapat empat istilah sebagai representasi hasil proses klasifikasi. Keempat istilah tersebut adalah *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Nilai TP merupakan data positif yang terdeteksi dengan benar, nilai TN merupakan jumlah data negatif yang terdeteksi benar, nilai FP merupakan data negatif namun terdeteksi sebagai data positif, dan FN merupakan kebalikan dari

nilai TP namun terdeteksi sebagai data negatif. Berikut contoh *confusion matrix* dapat dilihat pada Tabel 2.3 berikut.

Tabel 2.3 Aturan Nilai Dalam Confusion Matrix

Kelas		**Hasil Klasifikasi**	
		P	**N**
Target	T	TP	TN
	F	FP	FN

Keterangan:

a. *True Positive* (TP), merupakan jumlah dokumen dari kelas 1 yang benar diklasifikasikan sebagai kelas 1.
b. *False Positive* (FP), merupakan jumlah dokumen dari kelas 0 yang salah diklasifikasikan sebagai kelas 1.
c. *False Negative* (FN), merupakan jumlah dokumen dari kelas 1 yang salah diklasifikasikan sebagai kelas 0.
d. *True Negative* (TN), merupakan jumlah dokumen dari kelas 0 yang benar diklasifikasikan sebagai kelas 0.

Untuk menghitung akurasi digunakan perhitungan *Overall Accuracy* dengan membagi jumlah hasil klasifikasi yang bernilai benar (Jumlah dari diagonal di *confusion matrix*) dengan total dari seluruh baris dan kolom dari *confusion matrix*, atau dapat dilihat pada rumus 2.24.

$$Accuracy = \frac{Keseluruhan\ data\ terklasifikasi\ benar}{Keseluruhan\ testing\ data} * 100\% \qquad (2.24)$$

$$Error\ Rate = \frac{Keseluruhan\ data\ tidak\ klasifikasi\ benar}{Keseluruhan\ testing\ data} * 100\% \qquad (2.25)$$

Kemudian rumus untuk menghitung *precision, recall*, dan *f1-score* jika jumlah kelas lebih dari 1 adalah menggunakan rumus rata-ratanya. Formula yang digunakan untuk klasifikasi multikelas adalah sebagai berikut.

$$Precision = \frac{\sum_{i=1}^{l} \frac{TP_i}{TP_i + FP_i}}{l} * 100\% \quad (2.26)$$

$$Recall = \frac{\sum_{i=1}^{l} \frac{TP_i}{TP_i + FN_i}}{l} * 100\% \quad (2.27)$$

$$F1 - Score = \frac{\sum_{i=1}^{l} \frac{2\ x\ precision_i\ x\ recall_i}{precision_i\ x\ recall_i}}{l} * 100\% \quad (2.28)$$

Dengan keterangan tambahan :

l = Jumlah Kelas.