BAB 2
TINJAUAN PUSTAKA

2.1 Klasifikasi Dokumen

Klasifikasi dokumen adalah proses untuk mengelompokan suatu dokumen. Klasifikasi dokumen merupakan pemberian kategori kepada dokumen yang belum memiliki kategori [7]. Proses klasifikasi dilakukan karena banyaknya jumlah dokumen, untuk mencari suatu dokumen akan lebih mudah jika dikumpulkan secara kelompok sesuai dengan kategori yang sudah ditentukan. Dalam pengklasifikasian terhadap teks pada dokumen terdapat dua proses yang dilakukan [7]. Pertama adalah proses *training* digunakan data latih yang telah diketahui kategorinya untuk melakukan pelatihan pada teks dokumen. Kedua adalah proses *testing* digunakan untuk mengetahui keakuratan pada teks dokumen menggunakan data uji untuk kategorinya [7]. Dokumen-dokumen yang memiliki isi yang sama akan dikelompokan ke dalam kategori yang sama. Sehingga, orang yang melakukan pencarian informasi dapat dengan mudah melewatkan kategori yang tidak diinginkan atau yang tidak menarik.

Pada penelitian ini dokumen yang akan digunakan berdasarkan Kelompok keilmuan Teknik Informatika UNIKOM yaitu:

1. Kelompok Keilmuan A (Sistem Informasi)

   Pada Kelompok Keilmuan A ini mencakup beberapa kajian diantaranya :

   a. Sistem Informasi Geografis (SIG) digunakan untuk perencanaan, pelaksanaan, dan pengendalian berkaitan dengan wilayah geografis yang memaparkan informasi dalam bentuk grafis menggunakan peta.
   b. Sistem Informasi Manajemen (SIM) digunakan untuk sistem perencanaan bagian dari pengendalian internal bisnis meliputi pemanfaatan manusia, dokumen, teknologi, dan prosedur oleh pihak manajerial untuk memecahkan masalah bisnis seperti biaya produk, layanan, atau suatu strategi bisnis.

c. *Supply Chain Management* (SCM) digunakan untuk memaparkan informasi dan arus keuangan antara perusahaan yang berpartisipasi.
d. *Customer Relationship Management* (CRM) digunakan untuk merencanakan, menjadwalkan, dan mengendalikan aktivitas-aktivitas prapenjualan dan pasca penjualan dalam sebuah organisasi. CRM melingkupi semua aspek yang berhubungan dengan calon pelanggan dan pelanggan saat ini.
e. Sistem Informasi Manajemen Proyek digunakan untuk implementasi dari pengetahuan, keterampilan, perangkat dan teknik pada suatu aktifitas proyek untuk memenuhi kebutuhan atau tujuan suatu proyek.
f. Sistem Informasi Eksekutif (EIS) digunakan untuk memaprkan informasi oleh top eksekutif untuk memantau kondisi bisnis perusahaan secara umum atau hasil sebuah proses bisnis secara khusus.

2. Kelompok Keilmuan B (Rekayasa Perangkat Lunak)

   Pada Kelompok Keilmuan B ini mencakup beberapa kajian diantaranya :

   a. Rekayasa Perangkat Lunak digunakan dalam membangun sebuah perangkat lunak. Kajian untuk RPL diantaranya : *Coding Assesment,* pembangunan *Framework, Design Pattern Finding, Code Refinement,* pembangunan *Class Library*, *Format Method, Human Computer Interaction, Software Quality, Software Matrics, Software Testing, Software Migration* dan *Software Refinement.*
   b. *Business Intelligence* digunakan untuk mengumpulkan, menyimpan dan menyediakan akses terhadap data untuk pengembangan bisnis.
   c. Rekayasa Data digunakan untuk membahas model data, implementasi teknologi basis data dan terapannya dalam kehidupan sehari-hari. Kajian untuk Rekayasa Data diantaranya : *Data Mining, Datawarehouse, Clustering Database, Big Data, Web Mining/Text Mining,* Basis Data Terdistribusi, *NoSQL Performance, Data Spatio Temporal, Data Spatial, Query Processing and Optimization* dan Data Master Management

3. Kelompok Keilmuan C (Jaringan Komputer dan Multimedia)

   Pada Kelompok Keilmuan C ini mencakup beberapa kajian diantaranya :

   a. Multimedia digunakan untuk mengolah data dengan media-media tertentu yang menghasilkan nilai informasi yang lebih tinggi. Kajian untuk bidang Multimedia diantaranya : *Voice Tecnology, Video Recognition, Virtual Museum, Virtual Reality* dan *Augmented Reality, Chatbot Multimedia* dan *Game (*Algoritma/Metode*)* .
   b. Keamanan Sistem, kajian untuk bidang ini diantaranya : VPN, Firewell, Proxy.
   c. Teknologi IT, kajian untuk bidang ini diantaranya : QRCode, RFID, NFC, Internet of Things.
   d. Jaringan Komputer dan Internet digunakan untuk membangun atau mengembangkan sistem dan konten yang terdapat dilingkungan jaringan komputer (LAN/WAN). Kajian untuk bidang Jaringan Komputer dan Internet diantaranya : Pembangunan VOIP (*Voice Over Internet protocol*), VPN (*Virtual Private Network*). Cloud Computing Aplikasi, Security dan Arsitektur.

4. Kelompok Keilmuan D (Teknologi Internet dan Mobile)

   Pada Kelompok Keilmuan D ini mencakup beberapa kajian diantaranya :

   a. *E-Learning,*
   b. *Learning Management System (LMS),*
   c. *Internet Technology,*
   d. *Pengembangan E-Commerce,*
   e. *Mobile Technology,*

5. Kelompok Keilmuan E (*Computer Science*)

   Pada Kelompok Keilmuan E ini mencakup beberapa kajian diantaranya :

   a. *Artificial Intelligence*, kajian untuk bidang ini diantaranya : *Perceptual Computing, Machine Learning (Statistical Machine Learning, Machine Learning in Robot),* dan *Optimization Case.*

b. *Information Retrieval*, kajian untuk bidang ini diantaranya :*Indexing* dan *Search Engine.*
c. *Image Recognition*
d. *Voice Recognition*, kajian untuk bidang ini diantaranya : *Speaker Recognition,* dan *Speech Recognition.*
e. Teori Komputasi, kajian untuk bidang ini diantaranya : Teori Kompleksitas Komputasi (Masalah P = NP), Kriptografi, Teori Quantum Computing.
f. *Intelligent Tutoring*, kajian untuk bidang ini diantaranya : *Essay Grading, Question Generation, Programming learning (error-checking, suggestions, peer grading, style analysis).*
g. *Natural Language Processing*, kajian untuk bidang ini diantaranya : *Information Extraction, Question Answering, Sentiment & Social Meaning, Parsing & Tagging, Machine Translation, Dialog & Speech Processing, Multilingual NLP, Linguistic Structure, Summarization.*
h. *Programing Language & Environments*, kajian untuk bidang ini diantaranya : *Automatic Analyzer of Correctness Program, Languages Comparison, Software Support for Languages (Compiler), Program Optimization, Formal Reasoning, Automated Debugging (Finding Bug Fixes, Bugs Detection)* .

2.2 *Preprocessing*

Preprocessing adalah proses untuk mempersiapkan teks menjadi data yang akan diolah ke proses berikutnya. Masukan awal pada proses ini adalah dokumen abstrak tugas akhir. Tujuan dari *Preprocessing* untuk menghilangkan noise dalam suatu teks sehingga dapat menghapus data yang kurang relevan dalam proses mengklasifikasikan dokumen abstrak tugas akhir. *Preprocessing* pada penelitian ini terdiri dari beberapa proses meliputi proses *case folding*, proses *filtering*, proses *tokenizing* dan proses *stopword removal* [8],[9] .

2.2.1 *Case Folding*

Case Folding adalah proses untuk mengubah semua huruf didalam dokumen menjadi huruf kecil (*lower-case*). Tujuan penggunaan *case folding* adalah untuk mengatasi ketidaksamaan penggunaan huruf kapital dan huruf kecil didalam dokumen. Perubahan yang dilakukan meliputi keseluruhan huruf yang ada [8], [9].

2.2.2 *Filtering*

Filtering adalah proses mengambil kata penting dari hasil token. Dengan kata lain filtering merupakan proses dimana teks berupa simbol dan spasi akan dihilangkan. *Filtering* menggunakan algoritma *stoplist* (membuang kata yang kurang penting) atau *wordlist* (menyimpan kata penting) [9].

2.2.3 *Tokenizing*

Tokenizing adalah proses untuk pemisahan setiap kata yang terdapat didalam dokumen. Pada proses *tokenizing* dilakukan penghilangan tanda baca, karakter dan angka selain huruf alfabet. *Tokenizing* dilakukan untuk membuat data lebih terstruktur dan juga memudahkan sistem dalam pengolahan kata. Hasil keluaran akan digunakan sebagai masukan dalam tahap transformasi teks [8], [9].

2.2.4 *Stopword Removal*

Stopword Removal adalah proses untuk menghapus kata yang terdapat dalam stopword list dari token list seperti kami, saya, dan, dari, dll. *Stoplist* berisi sekumpulan kata yang tidak relevan namun sering muncul dalam sebuah dokumen. *Stoplist* berisi sekumpulan *stopwords*. Setiap kata akan diperiksa apakah terdapat *stoplist* atau tidak, jika sebuah kata termasuk kedalam *stoplist* maka kata tersebut tidak akan diproses lebih lanjut dan akan dihilangkan. Sebaliknya jika sebuah kata tidak termasuk kedalam s*toplist* maka kata tersebut akan masuk ke proses berikutnya. Stopword biasanya berisi kata-kata yang sering kali muncul berupa kata sambung, kata depan, kata ganti, kata penghubung. Tujuan dari *Stopword Removal* adalah untuk mengurangi jumlah kata yang akan diproses. Pada penelitian

ini, menggunakan daftar stopword untuk Bahasa Indonesia hasil Penelitian yang dilakukan oleh Fadillah Z Tala [8], [9], [10].

2.3 Pembobotan Kata TF-Idf

Term Frequency – Inverse Document Frequency digunakan untuk menentukan nilai frekuensi sebuah kata di dalam banyaknya dokumen. Perhitungan statistik numerik yang dimaksudkan untuk mencerminkan betapa pentingnya dan seberapa relevannya sebuah kata di dalam sebuah dokumen. Pembobotan diperoleh dari frekuensi jumlah kemunculan sebuah kata yang terdapat di dalam sebuah dokumen, *term frequency* (tf). Sebuah kata atau jumlah kemunculan *term* di dalam koleksi dokumen, *inverse document frequency* (idf). TF-IDF dapat berhasil digunakan dalam penyaringan di berbagai bidang, termasuk *text summarization* dan klasifikasi. Bobot suatu istilah semakin besar jika istilah tersebut sering muncul dalam suatu dokumen dan semakin kecil jika istilah tersebut muncul dalam banyak dokumen [11].

Nilai idf sebuah *term* (kata) dapat dihitung menggunakan persamaan (2.1) berikut:

$$IDF = \log(N/df) \quad (2.1)$$

Untuk menghitung bobot (W) masing-masing dokumen terhadap setiap *term* (kata) dapat menggunakan persamaan (2.2) berikut:

$$W = tf \,.\, IDF \quad (2.2)$$

Dimana:

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

d = *dokumen ke-d*

t = *kata ke-t*

tf = *banyaknya kata yang dicari pada sebuah dokumen*

N = total dokumen

df = *banyak dokumen yang mengandung tiap kata*

Metode ini mampu menghitung bobot setiap *term* yang ada pada dokumen. Namun dalam metode ini tidak ada solusi untuk mengatasi resiko adanya redudansi (kemiripan) *term* yang dihasilkan dalam dokumen.

2.4 *Mutual Information* (MI)

Seleksi fitur dapat digunakan untuk mengklasifikasikan suatu dokumen dengan lebih efisien dan efektif dengan mengurangi fitur maupun mengidentifikasi fitur yang sesuai untuk dipertimbangkan dalam proses pengklasifikasian tersebut. Dalam seleksi fitur, ada dua jenis utama metode seleksi fitur dalam *machine learning* yaitu wrapper dan *filter*. *Wrapper* menggunakan akurasi klasifikasi dari beberapa algoritma sebagai fungsi evaluasinya. Sedangkan metode filter terdiri dari *document frequency*, *mutual information*, *information gain*, dan *chi-square* [12]. Seleksi fitur bertujuan untuk dapat meningkatkan akurasi klasifikasi dengan menghilangkan fitur *noise* [4]. Proses pencarian fitur yang relevan ini digunakan untuk dapat menentukan atribut unik yang dapat meminimalkan penggunaan kata dalam suatu kelompok.

Tahap seleksi fitur *Mutual Information* (MI) bertujuan untuk melakukan seleksi atribut ataupun kata yang akan dimasukan dalam proses klasifikasi sehingga proses klasifikasi yang dilakukan lebih efektif dan informatif dengan mengurangi jumlah data yang dianalisis [9],[13],[14]. Proses pemilihan seleksi fitur *Mutual Information* (MI) merupakan proses pemilihan fitur yang relevan yang berguna untuk membangun model klasifikasi [9],[15]. *Mutual Information* digunakan untuk mengukur jumlah atribut yang berperan dalam membuat klasifikasi benar didalam kelas sehingga akan menghasilkan inputan yang lebih berpengaruh pada proses klasifikasi [9]. Fitur yang tidak relevan adalah fitur yang tidak memberikan informasi berguna tentang data. Metode *filter* menyeleksi fitur yang relevan sebelum berpindah pada fase pembelajaran selanjutnya. Fitur yang terlihat paling signifikan dipilih untuk klasifikasi, sementara sisanya yang lain disisihkan [16]. Seleksi Fitur *Mutual Information* (MI) adalah model seleksi fitur yang menunjukkan berapa banyaknya informasi ada atau tidaknya sebuah kata yang memberikan konstribusi dalam membuat keputusan klasifikasi secara benar atau salah [13].

Salah satu metode yang digunakan untuk melakukan *Filter* adalah *Mutual Information* (MI). Keluaran dari MI merupakan suatu matriks n * n dimana n adalah banyaknya atau jumlah atribut dimulai nilai dalam matriks tersebut

merupakan nilai keterkaitan antar dua atribut [9]. Matriks yang saling memiliki keterkaitan, semakin besar nilai MI maka semakin besar keterkaitan antar sebuah atribut. Pengurutan nilai MI dari besar ke kecil, dimana semakin besar nilai MI maka semakin besar suatu atribut tersebut mempengaruhi suatu kelas [9]. Hasil *Mutual Information* (MI) diurutkan berdasarkan yang terbaik (*ascending*). Yang pertama adalah atribut atau kata dengan nilai *Mutual Information* (MI) yang paling tinggi dan akan dipilih sebagai fitur kata yang informatif untuk digunakan dalam sebuah klasifikasi [14], [15]. Nilai *Mutual Information* (MI) yang berada pada peringkat bawah atau paling kecil pada *term* akan dihapus [16]. Langkah-langkah yang digunakan untuk dalam proses seleksi fitur *Mutual Information* (MI) diantaranya: Pertama, melakukan penghitungan nilai kata/informasi dari setiap kategori dengan MI. Kedua, melakukan pemilihan kata dengan nilai MI tertinggi dari setiap kategori. Ketiga, melakukan penggabungan fitur kata dengan MI tertinggi dari setiap kategori [13], [14],[15].

Mutual Information (MI) merupakan fitur yang sering digunakan untuk memodelkan hubungan antar kata. Bila terdapat kata t dan kategori c, dengan A sebagai banyak dokumen dengan t dan c muncul secara bersamaan, B sebagai banyak dokumen dengan t muncul tanpa c, C sebagai banyak dokumen dengan c muncul tanpa t, N sebagai jumlah seluruh dokumen, P(t∩c) sebagai probabilitas kata t dan muncul pada kategori c, P(t) sebagai probabilitas kata t muncul dan P(c) sebagai probabilitas kategori c dipilih [17]. Maka MI dapat didefinisikan sebagai berikut:

$$\mathrm{MI}(t,c) = \log \frac{P(t \cap c)}{P(t)\,.\,P(c)} \qquad (2.3)$$

Dan dapat juga dihitung dengan:

$$\mathrm{MI}(t,c) \approx \log \frac{A*N}{(A+C)*(A+B)} \qquad (2.4)$$

Dimana :

A : jumlah dokumen dikelas k yang mengandung *term* t

B : jumlah dokumen diluar kelas k yang tidak mengandung *term* t

C : jumlah dokumen dikelas k yang tidak mengandung *term* t

N : total dokumen

2.5 *Support Vector Machine* (SVM)

SVM merupakan suatu teknik untuk menemukan hyperplane yang bisa memisahkan dua set data dari dua kelas yang berbeda. *Hyperplane* adalah garis batas pemisah data antar kelas, sedangkan *margin* adalah jarak antara *hyperplane* dengan data terdekat pada masing-masing kelas. Adapun data terdekat dengan *hyperplane* pada masing-masing kelas inilah yang disebut *support vector*. Karena konsep awal SVM adalah untuk mengatasi masalah klasifikasi dua kelas maka, ada dua metode diusulkan agar SVM bisa digunakan untuk klasifikasi *multi-class* dengan pendekatan mengombinasikan beberapa *binary classifier* atau menggabungkan semua data yang terdiri dari beberapa kelas ke dalam permasalahan optimasi[18].

Dalam SVM pada dasarnya membahas tentang memaksimalkan batas *hyperlane*. Data yang di notasikan dengan (x_i, y_i) dengan $i = 1,2, \dots, N$ dan $x_i = \{x_{i1}, x_{i2}, \dots, x_{iq}\}^T$ merupakan atribut (fitur) set untuk data latih kelas ke-i. Untuk $yi \in \{-1,+1\}$ merupakan label kelas. *Hyperplane* klasifikasi linear SVM yang didefinisikan dengan persamaan (2.5):

$$w \,.\, x_i + b = 0 \tag{2.5}$$

Data x_i yang terbagi ke dalam dua kelas, yang termasuk kelas -1 (sampel negatif) didefinisikan dengan persamaan (2.6):

$$w \,.\, x_i + b \leq -1 \tag{2.6}$$

Sedangkan yang termasuk kelas +1 (sampel positif) didefinisikan dengan persamaan (2.7):

$$w \,.\, x_i + b \geq +1 \tag{2.7}$$

Dimana:

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 data $y_i = -1$, maka akan memenuhi persamaan (2.8):

$$w \,.\, x_a + b = -1 \tag{2.8}$$

Bila label data y_i = +1, maka akan memenuhi persamaan (2.9):

$$w \,.\, x_i + b = +1 \tag{2.9}$$

Margin terbesar dapat dicari dengan cara memaksimalkan jarak antara *hyperlane* dan titik terdekatnya. Sehingga *margin* dapat dihitung dengan mengurangkan persamaan (2.9) dan persamaan (2.10). maka akan didapatkan persamaan (2.10):

$$w \,.\, (x_b - x_a) = 2 \tag{2.10}$$

Margin hyperlane diberikan oleh jarak antara dua *hyperlane* dari dua kelas tersebut. Persamaan (2.10) dapat diuraikan menjadi persamaan (2.11):

$$||w||\ x\ d = 2 \ \text{ atau } d = \frac{2}{||w||} \tag{2.11}$$

Dimana ||w|| adalah vektor bobot w dan d adalah *margin.* Selanjutnya masalah ini diformulasikan ke dalam problem *Quadratic Programming* (QP) dengan meminimalkan invers persamaan (2.11), $\frac{1}{2}||w||^2$, dengan syarat sebagai berikut:

Minimalkan :

$$\frac{1}{2}||w||^2 \tag{2.12}$$

Syarat :

$$y_i\,(w \,.\, x_i + b) \ \geq\ 1\ ,\ i = 1,2,\ \ldots\ ,N \tag{2.13}$$

Optimalisasi ini dapat dipecahkan dengan berbagai teknik komputasi menggunakan metode *Largrange multiplier.* Metode ini dapat dinyatakan dengan persamaan (2.14):

$$L_p = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha \text{iyi}\,(w \,.\, x_i + b) - 1 \tag{2.14}$$

Dimana α_i adalah *lagrange multiplier* yang berkorespodensi dengan x_i . Nilai α_i merupakan nilai nol atau positif ($\alpha_I \geq 0$). Selanjutnya persamaan (2.14) akan diminimalan terhadap w dan b dan di set dengan nilai nol sehingga dapat dilihat syarat pada persamaan (2.15) dan persamaan (2.16):

Syarat 1 :

$$\frac{\partial \text{Lp}}{\partial \text{w}} = 0 \rightarrow w = \sum_{i=1}^{n} \alpha iyi\ \text{x}i \tag{2.15}$$

Syarat 2 :

$$\frac{\partial \mathrm{Lp}}{\partial \mathrm{w}} = 0 \rightarrow w = \sum_{i=1}^{n} \alpha i y i \, \mathrm{x}i \; = 0 \qquad (2.16)$$

Dimana N adalah jumlah data yang menjadi *support vector*. Karena *Lagrange Multiplier* (α) tidak diketahui nilainya, persamaan di atas tidak dapat diselesaikan secara langsung untuk mendapatkan w dan b. Untuk menyelesaikan masalah tersebut, modifikasilah Persamaan (2.14) diatas menjadi kasus memaksimalkan dengan syarat optimal untuk dualitas menggunakan konstrain KKT (*Karush-Kuhn-Tucker*) sebagai berikut:

Syarat 1 :

$$\alpha_i \, [y_i \, (w \, . \, x_i + b) \, -1] = 0 \qquad (2.17)$$

Syarat 2 :

$$\alpha_i \geq 0, \, i = 1,2, \, \ldots \, , N \qquad (2.18)$$

Dengan menerapkan kendala pada persamaan (2.17) dan (2.18), dipastikan bahwa nilai *Lagrange Multiplier* sama banyaknya dengan data latih, tetapi sebenarnya banyak dari data latih yang *Lagrange Multiplier*nya sama dengan nol (karena hanya beberpa saja yang akan menjadi *support vector*) ketika menerapkan syarat pertama. Kendala diatas menyatakan bahwa *Lagrange Multiplie*r α_i harus nol, kecuali untuk data latih x_i yang memenuhi persamaan:

$$y_i \; (w \, . \, x_i + b) \; = 1 \qquad (2.19)$$

Data latih tersebut, dengan $\alpha_i > 0$, terletak pada *hyperplane* b_1 atau b_2, dan disebut *support vector*. Data latih yang tidak terletak di *hyperplane* tersebut mempunyai $\alpha_i = 0$. Persamaan (2.15) dan (2.16) juga menyarankan parameter w dan b, yang mendefinisikan *hyperplane*, hanya tergantung *support vector*.

Masalah optimalisasi di atas masih sulit karena banyaknya parameter: w, b, dan α. Untuk menyederhanakannya, persamaan (2.14) harus ditransformasi ke dalam fungsi *Lagrange Multiplier* itu sendiri (disebut dualitas masalah). Persamaan *Lagrange multiplier* dapat dijabarkan sebagai berikut:

$$L_p = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{n} \alpha \mathrm{i} y \mathrm{i} \, (w \, . \, x_i) \, - b \sum_{i=1}^{n} \alpha \mathrm{i} y \mathrm{i} + \sum_{i=1}^{n} \alpha \mathrm{i} \qquad (2.20)$$

Syarat optimal persamaan (2.18) ada dalam suku ketiga di ruas kanan dalam persamaan (2.20) dan memaksa suku ini menjadi sama dengan 0. Dengan mengganti w

dari syarat persamaan (2.15) dan suku $||w||^2$ $w_i w_j$ maka persamaan diatas akan berubah menjadi dualitas *Lagrange multiplier* berupa L_D dan didapatkan:

Memaksimalkan :

$$L_D = \sum_{i=1}^{n} \alpha i - \frac{1}{2} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j x_i x_j \quad (2.21)$$

Syarat 1 :

$$\sum_{i=1}^{n} \alpha_i y_j = 0 \quad (2.22)$$

Syarat 2 :

$$\alpha_i \geq 0, i = 1,2, \dots, N \quad (2.23)$$

Dengan $\boldsymbol{x_i x_j}$ merupakan *dot-product* dua data dalam data latih. *Hyperplane* (batas keputusan atau pemisah) didapatkan dengan persamaan (2.24):

$$(\sum_{i=1}^{n} \alpha_i y_j \, x_i \, z) + b = 0 \quad (2.24)$$

N adalah jumlah data yang menjadi support vector, x_i merupakan *support vector*, z merupakan data uji yang akan diprediksi kelasnya, dan $x_i \, . \, z$ merupakan *inner-product* antara x_i dan z. Untuk nilai b didapatkan dari Persamaan (2.17) pada *support vector*. Karena α_i dihitung dengan 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 (2.17), biasanya diambil nilai rata – rata dari b yang didapat untuk menjadi parameter *hyperplane*. Untuk mendapatkan b. Persamaan (2.17) dapat disederhanakan menjadi persamaan (2.25):

$$b_i = 1 - y_i \, (w \, . \, x_i) \quad (2.25)$$

2.5.1 *Nonlinear Support Vector Machine*

Nonlinear Support Vector Machine adalah *hyperlane linear* yang hanya bekerja pada data yang dapat dipisahkan secara *linear*. Untuk data yang distribusi kelasnya tidak *linear* biasanya menggunakan pendekatan *kernel* pada fitur data awal set data [18]. *Kernel* dapat didefinisikan sebagai suatu fungsi yang memetakan fitur data dari dimensi awal yang rendah ke fitur baru dengan dimensi yang relatif lebih tinggi. Algoritma pemetaan *kernel* dinyatakan sebagai persamaan (2.26):

$$D^r \rightarrow D^q \quad (2.26)$$
$$x \rightarrow \Phi(x)$$

Dengan Φ merupakan fungsi kernel yang digunakan untuk pemetaan, D merupakan data latih, r merupakan set fitur dalam satu data yang lama dan q merupakan

set fitur yang baru sebagai hasil pemetaan untuk setiap data latih. Sementara x merupakan data latih, dengan $x_1, x_2, \ldots$, N ∈ D^r merupakan fitur – fitur yang akan dipetakan ke fitur berdimensi tinggi q. Jadi set data yang digunakan sebagai pelatihan adalah set data dari dimensi fitur yang lama r ke dimensi baru q, misalnya untuk sampel data N seperti pada persamaan (2.27):

$$((x_1),1,\Phi(x_2), y_2,\ldots,\Phi(x_N), x_n) \in \quad (2.27)$$

Proses pemetaan pada fase ini memerlukan perhitungan *dot-product* dua buah data pada ruang fitur baru. *Dot-product* kedua buah vektor (x_i) dan (x_j) dinotasikan sebagai $(x_i).(x_j)$. Umumnya transformasi Φ tidak diketahui dan sangat sulit dipahami. Oleh karena itu, perhitungan nilai *dot-product* dapat dihitung dengan fungsi kernel K(x_i x_j^T) yang mendefinisikan secara implisit fungsi transformasi φ tersebut. Inilah yang disebut *kernel trick*, yang di formulasikan sebagai pada persamaan (2.28):

$$(x_i\, x_j^T) = (x_i).\,(x_j) \quad (2.28)$$

Dan untuk prediksi pada set data dengan dimensi fitur yang baru diformulasikan terdapat pada persamaan (2.29):

$$(\Phi(x)) = (w.\Phi(z) + b) = sign\,(\textstyle\sum_{i=1}^{n} \alpha_i y_j\ \Phi(x_i).\Phi(z)+b) \quad (2.29)$$

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

2.6 Pemrograman Berorientasi Objek

Pemrograman Berorientasi Objek (*Object Oriented Programming*) adalah cara untuk menangani masalah melalui pengamatan setiap objek didunia nyata dan objek tersebut merupakan entitas tunggal yang memiliki kombinasi struktur data dan fungsi tertentu.

Konsep dasar berorientasi objek diantaranya [19]:

1. Objek (*Object*) adalah konsep atau abstraksi tentang suatu yang memiliki arti bagi aplikasi yang akan dikembangkan. Biasanya objek adalah kata benda, namun objek dalam konteks OOP biasanya menyangkut rumus persamaan kuadrta, liberalism, marxisme, dan sebagainya. Objek dapat dikategorikan dalam beberapa kategori diantaranya: Objek konsep, Objek event, Objek organisasi, Objek manusia, Objek tempat dan sesuatu yang terukur serta objek peralatan.

2. Kelas (*Class*) mencakup sifat-sifat umum yang dimiliki objek-objek. Suatu kelas tunggal dapat digunakan untuk sejumlah objek. Contoh: Suatu Kelas bernama Staf Akademik memiliki objek diantaranya rector, dekan, ketua jurusan, dosen dan lainnya. Contoh lainnya kelas Alat Musik memiliki objek drum, bass, gitar, keyboard.
3. Abstraksi (*Abstraction*) adalah menemukan sesuatu yang berarti untuk dituangkan dalam sistem/perangkat lunak. Abstraksi memfokuskan apa itu objek dan ciri-ciri yang dimiliki oleh objek tersebut (atribut-atributnya)serta apa yang objek itu lakukan (operasi suatu objek). Contoh: Kelas=Poligon berisi Atribut= titik sudut, warna batas, warna pengisian.
4. Atribut (*Attribute*) dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek.
5. Pewarisan (*Inheritance*) adalah mengijinkan suatu objek pada suatu kelas untuk mengakses fungsi atau peubah dalam kelas terdahulu tanpa mendefinisikan kembali sehingga dapat menciptakan kelas baru.
6. Pembungkusan (*Encapsulation*) adalah memfokuskan diri pada implementasi internal suatu objek sehingga meninggalkan aspek eksternal pada suatu objek.
7. Generalisasi/spesialisasi memungkinkan objek-objek berbagi data serat perilaku yang sama pada hierarki pewarisan. Memungkinkan pengurangan ukuran kode dan menyediakan kemungkinan pengembangan sistem/perangkat lunak yang mudah untuk dipelihara.
8. Polimorfisme (*Polymorphism*) adalah kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

2.7 *Unifed Modeling Language* (UML)

UML adalah bahasa standar untuk merancang dan mendokumentasikan perangkat lunak dengan cara berorientasi objek [20]. Ada beberapa diagram yang

digunakan proses pembuatan perangkat lunak berorientasi objek diantaranya, *use case diagram, activity diagram, class diagram* dan *sequence diagram*.

2.7.1 *Use Case Diagram*

Use case diagram adalah pemodelan untuk tingkah laku (*behavior*) pada sistem yang akan dibuat [20]. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. *Use case diagram* digunakan untuk mengetahui fungsi apa saja yang terdapat pada sistem. Terdapat dua hal utama yang diperlukan dalam pembentukan suatu *use case diagram* yaitu aktor dan *use case*.

1. Aktor merupakan orang, benda maupun sistem lain yang berinteraksi dengan sistem yang akan dibangun.
2. *Use Case* merupakan fungsionalitas atau layanan yang disediakan oleh sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

2.7.2 *Class Diagram*

Class Diagram adalah sebuah spesifikasi yang jika diinstansi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class Diagram* digambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti containment, pewaris, asosiasi dan lain-lain [20].

2.7.3 *Sequence Diagram*

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 [20].

2.7.4 *Activity Diagram*

Activity Diagram adalah gambaran *workflow* (aliran kerja) atau aktivitas dari sebuah sistem, proses bisnis atau menu yang ada pada perangkat lunak [20]. Setiap *use case* yang telah dibentuk digambarkan aktivitasnya dalam *activity diagram*, mulai dari peran aktor, peran sistem, dan *decision*.

2.8 Bahasa Pemrograman *Python*

Python merupakan salah satu Bahasa pemrograman yang popular. *Python* banyak digunakan untuk dapat menyelesaikan berbagai bidang persoalan diantaranya komputasi sains, antariksa, ekonomi dan bidang lainnya. *Python* secara default telah terpasang di beberapa sistem operasi diantaranya *Linux, Windows, Linux Mint* dan lainnya. Sisi utama yang membedakan *Python* dengan bahasa lain adalah dalam hal aturan penulisan kode program. Beberapa fitur yang dimiliki *Python* adalah: memiliki kepustakaan yang luas, dalam distribusi *Python* telah disediakan modul-modul siap pakai untuk berbagai keperluan, memiliki tata bahasa yang jernih dan mudah dipelajari, memiliki aturan *layout* kode sumber yang memudahkan pengecekan, pembacaan kembali dan penulisan ulang kode sumber, berorientasi obyek [21].