BAB 2
TINJAUAN PUSTAKA

2.1 *Text Mining*

Text mining, yang juga disebut sebagai Teks *Data Mining* (TDM) atau *Knowledge Discovery in Text* (KDT), secara khusus dikembangkan untuk proses ekstraksi informasi dari dokumen-dokumen teks tak terstruktur (*unstructured*) [7]. *Text Mining* mencoba untuk mengekstrak informasi yang berguna dari sumber data melalui identifikasi dan eksplorasi dari suatu pola menarik dan sumber datanya adalah dokumen.

2.1.1. *Text* Pre-Processing

Pre-processing merupakan langkah-langkah dalam mengolah data mentah yang berikutnya akan dimasukan ke dalam sistem klasifikasi [1]. Proses ini bertujuan untuk meningkatkan kualitas data, meningkatkan efisiensi dalam proses penambangan data [1]. Tahapan dalam *preprocessing text* adalah sebagai berikut:

a) *Case Folding*

Tidak semua dokumen teks hanya menggunakan huruf kapital. Oleh karena itu, *Case folding* digunakan untuk melakukan penyeragaman karakter dengan cara melakukan pengubahan karakter menjadi huruf kecil(*lowercase)* atau huruf kapital(*uppercase*) [8]. Contoh proses *case folding* adalah sebagai berikut:

Teks *input:*

Program Studi Teknik Informatik merupakan salah satu jurusan di UNIKOM

Hasil :

program **s**tudi teknik informatik merupakan salah satu jurusan di unikom

b) *Filtering*

Tahap *filtering* adalah tahap yang mengacu pada proses untuk menentukan istilah yang akan mempresentasikan atau menggambarkan isi dokumen dan membedakan dokumen satu dan yang lainnya pada suatu koleksi. Jadi *filtering* merupakan proses dimana teks selain karakter ‘a’ sampai ‘z’ dan spasi akan dihilangkan. Tahap-tahapan yang dilakukan yaitu:

1. Membaca data masukan yang akan diproses
2. Melakukan pengecekan apakah ada huruf selain ‘a’ sampai ‘z’ dan ‘ ‘ (spasi) jika tidak ditemukan maka algoritma berakhir. Jika ada
3. Hapus karakter yang bukan termasuk ‘a’ sampai ‘z’ dan ‘ ‘(spasi).
4. Algoritma selesai.

Contoh dari tari tahap *Filtering* adalah sebagai berikut:

Teks Input:

nama saya afdhalul ihsan, dan umur saya 21 tahun

Hasil:

nama saya afdhalul ihsan umur saya tahun

c) *Tokenizing*

Tokenization adalah proses memecah teks yang berupa kalimat, paragraf atau dokumen menjadi kata, frasa, simbol, atau elemen yang bermakna lain yang disebut sebagai token. Di dalam data dokumen, proses ini cukup sederhana karena‘ *spasi*(‘ ‘) akan dianggap sebagai delimeter yangmemisahkan kata-kata [1]. Contoh proses *tokenizing* adalah sebagai berikut :

Teks input :

nama saya adalah afdhalul Ihsan

Hasil :

nama saya adalah

afdhalul ihsan

d) *Stopword Removal*

Tahapan ini adalah proses pengambilan kata-kata penting dan pembuangan kata-kata yang dianggap tidak penting. Proses pembuangan kata tidak penting dapat diterapkan pada stoplist (membuang kata yang kurang penting) atau wordlist (menyimpan kata penting). *Stopword* yang biasanya sering digunakan adalah kata hubung dan kata ganti orang. Contoh *stopword* adalah "yang", "dan", "di", "dari", dan sebagainya [8].

2.2 *Machine Learning*

Istilah *machine learning* diperkenalkan oleh Arthur Samuel pada tahun 1959 melalui jurnal yang berjudul "*Some Studies in Machine Learning Using the Game of Checkers*" [9], di mana komputer diajari (*learning*) bermain catur dengan tujuan komputer dapat bermain lebih baik dari dirinya. Secara sederhana *machine learning* dapat diartikan sebagai sebuah program komputer yang mampu belajar untuk mencapai kriteria/ peforma tertentu [9]. Dalam proses pemenuhan kriteria/ peforma tersebut, *machine learning* menggunakan *data training* atau pengalaman dimasa lalu (*past experience*).

Saat ini sudah sangat banyak algoritma *machine learning* yang dikembangkan oleh para peneliti, setiap algoritma dibuat untuk berbagai macam tujuan. Tujuan tersebut muncul dari banyak nya persoalan rill yang terjadi dan memerlukan dukungan *machine learning* [9]. Secara umum algoritma *machine learning* dapat deklompokkan menjadi beberapa kelompok yaitu *supervised learning, unsupervised learning*, dan *reinforcement learning.Supervised Learning.*

2.3 Klasifikasi Dokumen

Klasifikasi dokumen adalah bidang penelitian dalam perolehan informasi yang mengembangkan metode untuk menentukan atau mengkategorikan suatu dokumen ke dalam satu atau lebih kelompok yang telah dikenal sebelumnya secara otomatis berdasarkan isi dokumen [10]. Klasifikasi dokumen bertujuan untuk mengelompokkan dokumen yang tidak terstruktur ke dalam kelompok-kelompok yang menggambarkan isi dari dokumen [10].

2.4 *Feature Extraction*

Feature Extraction merupakan tahapan untuk mengubah teks atau dokumen yang sebelumnya masih berbentuk kata kedalam bentuk yang lebih representative, salah satunya vector [1]. Tujuannya agar teks atau dokumen dapat lebih mudah diklasifikasikan ke dalam kategori masing-masing. *Term weighting/ TF-IDF* adalah salah satu metode ekstraksi fitur yang sering digunakan [11]. Metode ini menggabungkan dua konsep dalam perhitungan bobot yaitu, frekuensi kemunculan sebuah kata di dalam sebuah dokumen dan inverse dari frekuensi dokumen yang mengandung kata tersebut[10]. Frekuensi dokumen yang mengandung kata tersebut menunjukkan seberapa umum kata tersebut. Bobot hubungan antara sebuah kata dan sebuah dokumen akan tinggi apabila frekuensi kata tersebUt tinggi didalam dokumen dan frekuensi keseluruhan dokumen yang mengandung kata tersebut yang rendah pada kumpulan dokumen[10]. Rumus umum untuk pembobotan *TF-IDF*:

$$W_t = tf_{i,j} * idf \tag{2.1}$$

Term frequency (*TF*) adalah faktor yang menentukan bobot term (kata) pada suatu dokumen berdasarkan jumlah kemunculannya dalam dokumen tersebut. Nilai jumlah kemunculan term (kata) diperhitungkan dalam pemberian nilai bobot terhadap suatu kata. Semakin besar jumlah kemunculan suatu term (kata) dalam suatu dokumen, semakin besar pula bobotnya dalam dokumen atau akan memberikan nilai kesesuain yang semakin besar. Nilai *Term frequency* pada suatu dokumen juga dapat diartikan nilai dari kemunculan term (kata) dalam suatu

dokumen dibagi dengan jumlah kata dalam seluruh dokumen. Untuk menghitung nilai *term frequency* dapat digunakan persamaan 2.2 [12].

$$tf = \frac{\text{n}}{\sum nk} \tag{2.2}$$

Inverse Document Frequency (*IDF*) adalah pengurangan dominansi *term* (kata) yang sering muncul di berbagai dokumen. Hal ini diperlukan karena *term* (kata) yang sering muncul dalam berbagai dokumen dapat dianggap sebagai *term* umum, sehingga nilai pada *term* tersebut dapat dianggap tidak penting. Sebaliknya faktor kejarangan munculan suatu *term* (kata) dapat dainggp sebagai suatu kata yang penting. Pembobotan pada suatu *term* (kata) akan memperhitungkan nilai dari *inverse dokumen.* Untuk menghitung nilai *IDF* pada suatu term dapat menggunakan persammaan 2.3.

$$\boldsymbol{idf = log \frac{n}{df}} \tag{2.3}$$

Dimana :

idf = *inverse document frequency*

n = jumlah dokumen

df = jumlah dokumen yang memiliki kata (*term*)

2.5 *Supervised Learning*

Supervised learning adalah suatu metode dalam machine learning yang akan memprediksi output dari input yang diberikan [9]. Pada metode ini mesin akan mempelajari mapping function antara input dengan output dari dataset yang telah disiapkan. Dataset tersebut berisi data-data (yang akan menjadi ciri pada label tertentu) yang sudah diberi label.

Permasalahan-permasalahan yang dapat diselesaikan dengan *supervised learning* dapat dikategorikan menjadi dua jenis [9]:

1. *Classification*

   Classification bertujuan untuk memprediksi *output* dari *input*, dimana output variable berbentuk kategori-kategori. Contoh: pria/wanita, sakit/sehat dan lain-lain.

2. *Regression*

   Regression bertujuan untuk memprediksi *output dari* input, dimana output variable berbentuk nilai aktual. Contoh: tinggi badan seseorang, curah hujan dan lain-lain.

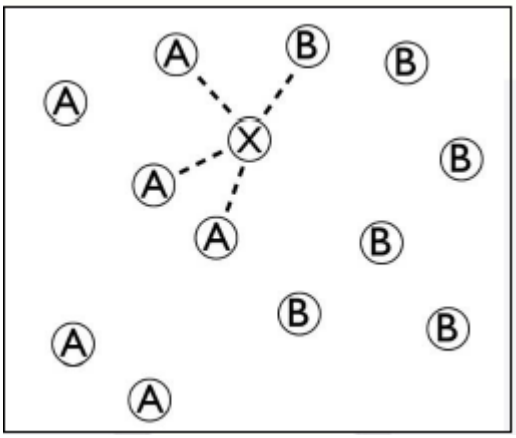
Banyak metode *supervised learning*, antara lain: *decision tree*, *naive bayes, artificial neural network, support vector machine, linear regression, logistic regression, k-Nearest neighbor* dan lain-lain

2.6 Algoritma *K-Nearest neighbor*

K-Nearest Neighbor (k-NN) adalah salah satu algoritma yang digunakan untuk melakukan klasifikasi. Algoritma k-NN termasuk metode supervised learning yang berarti adanya data latih dan terdapat variabel atau atribut yang ditargetkan sehingga tujuan dari algoritma ini adalah memetakan suatu data ke data yang sudah ada [8]. Algoritma *k-Nearest neighbor* adalah metode machine learning yang sangat mudah diimplementasikan. *K-Nearest neighbor* merupakan teknik klasifikasi yang melakukan prediksi secara tegas pada data uji berdasarkan banyaknya K tetangga terdekeat. Pada algoritma *k-nearest neighbor,* data berdimensi N, dapat dihitung jarak dari data tersebut ke data lainnya. Nilai dari jarak yang dihitung tersebut yang akan digunakan sebagai nilai kedekatan atau ketidak miripan antara data latih dan data uji. Hal yang penting dalam algoritma ini adalah pemilihan nilai k, jika k sangat kecil maka akan meniptakan *noise*. Sebaliknya, jika terlalu besar dapat menciptakan N dengan banyak kelas yang harus diklasifikasikan [8]. Pengukuran jarak antara data latih dan data uji dalam algoritma *k-nearest neighbor* dapat dhitung dengan menggunakan metode *euclidean distance* berikut :

$$d(\mathrm{x,y}) = \sqrt{\sum_{i=1}^{n}(xi - yi)^2} \qquad (2.4)$$

Dimana x dan y merupakan titik pada ruang vektor n dimensi sedangkan xi dan yi merupakan besaran skalar untuk dimensi ke i dalam ruang vektor n dimensi.



Gambar 2.1 Sebaran Data dalam Algoritma *k-Nearest neighbor*

Gambar 2.1 menunjukkan sebaran data dalam algoritma k-NN yang terdiri dari dua kelas yaitu kelas A dan B [8]. Dalam hal ini, sebuah data uji akan diklasifikan berdasarkan data latih [8]. Sebagai contoh, k yang digunakan adalah 4. Maka, data uji akan mencari 4 data latih yang berdekatan dengan data uji. 4 data tersebut adalah 1 titik menuju kelas B dan 3 titik menuju kelas A. Karena jumlah titik kelas A lebih dari titik kelas B, maka data uji akan diklasifikasikan sebagai kelas A.

2.7 *Particle swarm optimization*

Particle swarm optimization (PSO) merupakan algoritma evolusi yang mirip dengan algoritme genetika dengan memanfaatkan fungsi fitness yang digunakan untuk mengevaluasi kualitas suatu solusi permasalahan. Dalam PSO setiap solusi ada dalam ruang pencarian dipandang sebagai sebagai partikel yang mana setiap partikel memiliki nilai fitness untuk dioptimalkan, dan memiliki kecepatan untuk perpindahan partikel [13]. PSO memiliki beberapa kelebihan yaitu sedikit parameter, mudah diterapkan, konvergensi yang cepat, dan sederhana sehingga PSO banyak diterapkan pada optimasi fungsi, optimasi metode konvensional dan klasifikasi pola [13]. Kata partikel menunjukkan misalnya seekor burung dalam kawanan burung. Setiap individu atau partikel berperilaku dengan menggunakan kecerdasannya(inteligence) sendiri dan juga dipengaruhi perilaku

kelompok kolektifnya. Dengan demikian, jika suatu partikel atau seekor burung menemukan jalan yang tepat atau pendek maka sisa kelompok yang lain juga akan segera mengikuti jalan tersebut meskipun lokasi mereka jauh dari kelompok tersebut [14].

Dalam PSO sebuah partikel sama dengan sebuah individu. Sederhananya, partikel-partikel terbang melalui sebuah ruang pencarian multi dimensi dimana posisi dari setiap partikel diubah menurut pengalaman mereka sendiri dan tetangga mereka. Jika $X_i(t)$ merupakan posisi partikel dalam ruang pencarian di waktu t, maka posisi partikel berubah dengan menambah kecepatan $V_i(t)$ kedalam posisi saat ini dengan cara:

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (2.5)$$

2.7.1 Keuntungan dan Kerugian Algoritma Particle swarm optimization

Menurut Bai(2010) keuntungan dari algoritma *particle swarm optimization* adalah sebagai berikut :

1. PSO berdasar pada kecerdasan (intelligence). Ini dapat diterapkan ke dalam kedua penggunaan dalam bidang teknik dan riset ilmiah.
2. PSO tidak punya overlap dan kalkulasi mutasi. Pencarian dapat dilakukan oleh kecepatan dari partikel. Selama pengembangan beberapa generasi, kebanyakan hanya partikel yang optimis yang dapat mengirim informasi kepartikel yang lain, dan kecepatan dari pencarian adalah sangat cepat.
3. Perhitungan didalam Algoritma PSO sangat sederhana, menggunakan kemampuan optimisasi yang lebih besar dan dapat diselesaikan dengan mudah.
4. PSO memakai kode/jumlah yang riil, dan itu diputuskan langsung dengan solusi, dan jumlah dimensi tetap sama dengan solusi yang ada.

Sedangkan kekurangan dari algoritma *particle swarm optimization* adalah sebagai berikut :

1. Metode mudah mendapatkan optimal parsial (sebagian), yang mana menyebabkan semakin sedikit ketepatannya untuk peraturan tentang arah dan kecepatan.

2. Metode tidak bisa berkembang dari permasalahan sistem yang tidak terkoordinir, seperti solusi dalam bidang energi dan peraturan yang tidak menentu didalam bidang energy.

2.7.2 Global Best *PSO*

Untuk *global best PSO*, tetangga setiap partikel adalah seluruh swarm. Jaringan sosial yang digunakan oleh *global best PSO* menggambarkan topologi bintang. Untuk topologi tetangga bintang, komponen sosial dari pembaruan kecepatan partikel menggambarkan informasi yang diperoleh dari seluruh partikel dalam swarm [14]. Dalam kasus ini, informasi sosial merupakan posisi terbaik yang ditemukan swarm yang di anotasikan sebagai *gb*(t).

Untuk *global best PSO* , kecepatan partikel *i* dihitung dengan menggunakan persamaan:

$$V_i(t+1) = V_{ij}(t) + c_1 r_1 j(t)[y_{ij}(t) - X_{ij}(t)] + c_2 r_2 j(t)[gb(t) - X_{ij}(t)] \qquad (2.6)$$

Dimana $V_{ij}(t)$ merupakan kecepatan partikel *i* pada dimensi J = 1….n dalam waktu ke t, X_{ij} merupakan posisi partikel *i* dalam dimensi J dalam waktu ke t, c1 dan c2 merupakan konstanta akselarasi yang digunakan untuk memberi skala kontribusi komponen kognitif dan sosilal, dan r1 dan r2 merupakan nilai acak dengan *range* [0,1].

Posisi terbaik individu (y_i) berhubungan dengan partikel *i*, merupakan posisi terbaik partikel yang telah dikunjugi sejak waktu pertama. Dengan memperhitungkan masalah maksimasi, posisi terbaik individu dalam waktu selanjutnya, t+1 adalah

$$Y_i(t+1) = \begin{cases} yi(t) & jika\ f(xi(t+1) \geq f(yi(t)) \\ Xi(t) & jika\ f(xi(t+1) < f(yi(t)) \end{cases}$$

Dimana $F: \longrightarrow R^{nx}$ R merupakan fungsi objektif yang dipetakan dalam bilangan real. Seperti pada algoritma evolusi, fungsi objektif mengukur seberapa

dekat solusi dengan solusi optimum. Contohnya fungsi objektif mengukur performansi atau kualitas partikel.

2.7.3 Inisialisasi Partikel

Diasumsikan bahwa sebuah partikel untuk setiap dimensinya harus berada dalam domain yang didefinisikan oleh dua vektor, X_{min} dan X_{max} yang mewakili batas atas dan batas bawah di setiap dimensi. Metode inisialisasi yang efisien untuk posisi partikel adalah:

$$X(0) = Xmin + r_j(Xmax - Xmin) \quad (2.7)$$

Dimana $r_j \sim U(0,1)$ merupakan nilai acak dengan range [0,1]. Kecepatan awal dapat diinisialisasikan menjadi nol.

2.7.4 Kondisi Berhenti

Kondisi berhenti merupakan kriteria yang digunakan untuk mengakhiri proses iterasi pencarian dalam *particle swarm optimization.* Ketika memilih kriteria untuk mengakhiri proses iterasi pencarian, ada 2 aspek penting yang harus diperhatikan yaitu:

1. Kondisi berhenti harus tidak menyebabkan hasil pencarian dalam *particle swarm optimization* konvergen dini karena solusi yang suboptimal akan diperoleh.
2. Kondisi berhenti harus menjaga *oversampling* fungsi objektif, kompleksitas komputasi proses harus meningkat secara signifikan.

Beberapa kondisi berhenti yang dapat dipakai menurut Engelbrecht adalah:

1. Berhenti ketika jumlah iterasi sudah mencapai jumlah iterasi maksimum yang diperbolehkan
2. Berhenti ketika solusi yang diterima ditemukan
3. Berhenti ketika tidak ada perkembangan setelah beberapa iterasi

2.7.5 Proses Algoritma Particle swarm optimization

Menurut Chen & Shih (2013) untuk memulai algoritma PSO, kecepatan awal (velocity) dan posisi awal (position) ditentukan secara random. Kemudian proses pengembangannya sebagai berikut:

1. Asumsikan bahwa ukuran kelompok atau kawanan (jumlah partikel) adalah N. Kecepatan dan posisi awal pada tiap partikel dalam N dimensi ditentukan secara random (acak)
2. Hitung kecepatan dari semua partikel. Semua partikel bergerak menuju titik optimal dengan suatu kecepatan. Awalnya semua kecepatan dari partikel diasumsikan sama dengan nol, set iterasi i = 1.
3. Nilai fitness setiap partikel ditaksir menurut fungsi sasaran (objective function) yang ditetapkan. Jika nilai fitness setiap partikel pada lokasi saat ini lebih baik dari Pbest, maka Pbest diatur untuk posisi saat ini
4. Nilai fitness partikel dibandingkan dengan Gbest. Jika Gbest yang terbaik maka Gbest yang diupdate.
5. Persamaan (2.5) dan (2.6) ditunjukkan di untuk memperbaharui (update) kecepatan (velocity) dan posisi (position) setiap partikel.
6. Cek apakah solusi yang sekarang sudah konvergen. Jika posisi semua partikel menuju ke satu nilai yang sama, maka ini disebut konvergen. Jika belum konvergen maka langkah 2 diulang dengan memperbarui iterasi i = i + 1, dengan cara menghitung nilai baru dari Pbest,j dan Gbest. Proses iterasi ini dilanjutkan sampai semua partikel menuju ke satu titik solusi yang sama. Biasanya akan ditentukan dengan kriteria penghentian (stopping criteria), misalnya jumlah selisih solusi sekarang dengan solusi sebelumnya sudah sangat kecil.

2.8 Seleksi Fitur

Seleksi fitur adalah proses mengurangi jumlah fitur dengan menghilangkan fitur yang tidak relevan atau dengan memilih fitur yang paling relevan/ informatif untuk meningkatkan proses klasifikasi. Manfaat dari seleksi fitur adalah mengurangi jumlah data yang dibutuhkan untuk learning process, meningkatkan waktu komputasi dan meningkatkan kecepatan dan akurasi klasifikasi.

Seleksi fitur memilih subset fitur dari fitur keseluruhan untuk meningkatkan performa klasifikasi. Seleksi fitur dapat mengurangi kompleksitas dari model klasifikasi tanpa mengurangi akurasi dari classifier [15]. Seleksi fitur adalah dengan memilih m fitur dari n fitur keseluruhan atau fitur asli dari dataset (m < n)

[15]. Beberapa metode dalam seleksi fitur adalah, *filter base*, *wrapper base* dan *hybrid*.

a) Metode *filter*

Metode filter menerapkan beberapa analisis statistik pada satu set fitur untuk menyelesaikan permasalahan seleksi fitur tanpa menggunakan model pembelajaran (learning algorithm). Diantaranya chi-square test, information gain dan correlation coefficient [16] Oleh karena itu, metode ini biasanya tidak membutuhkan waktu komputasi yang banyak. [16]

b) Metode *Wrapper*

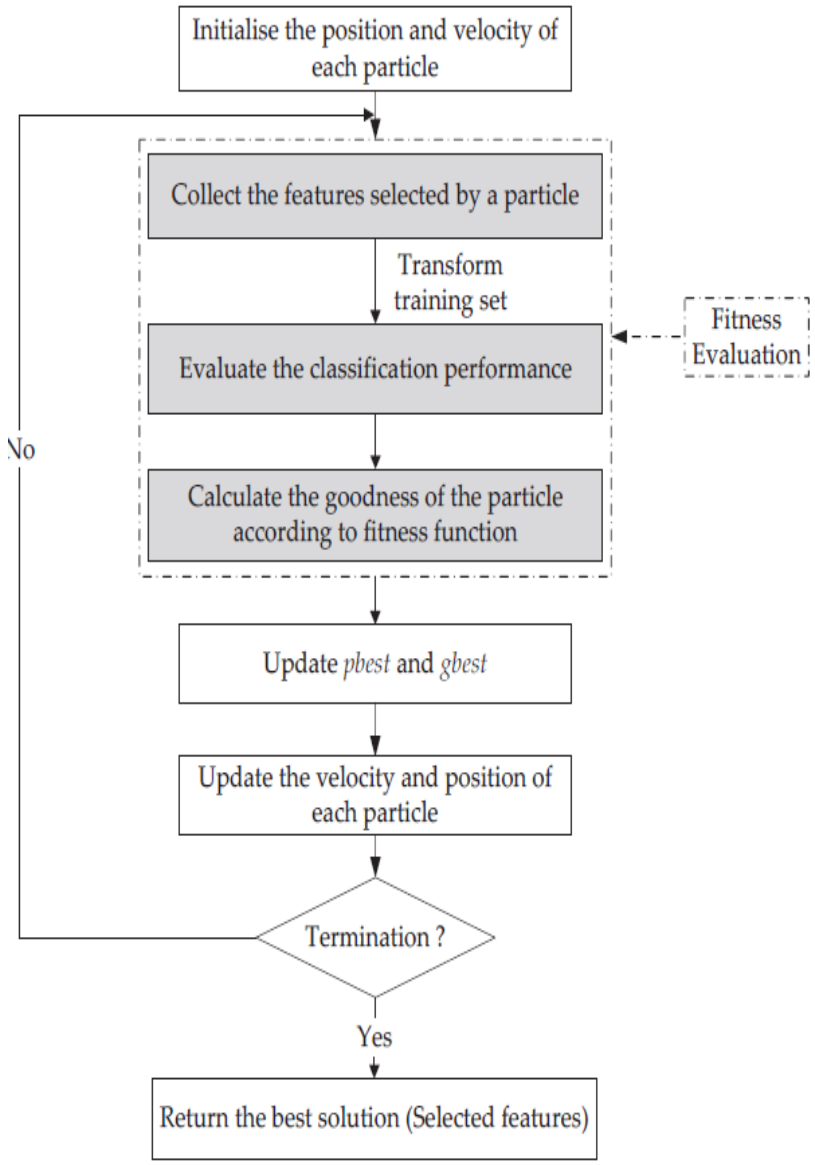
Metode wrapper menggunakan algoritma pembelajaran untuk melakukan evaluasi terhadap subset fitur melalui proses pencarian. Dengan kata lain, metode wrapper adalah proses pencarian berulang dimana hasil dari algortima pembelajaran pada setiap literasi digunakan untuk memandu proses pencarian [16].

c) Metode *Hybrid*

Metode hybrid adalah kombinasi dari metode filter dan wrapper [16]. Metode hybrid berfokus kepada penggabungan filter dan wrapper untuk mencapai performa terbaik yaitu dengan menggunakan algoritma pembelajaran tertentu dengan waktu kompleksitas mirip dengan metode filter.

2.9 Seleksi Fitur dengan *Particle swarm optimization*

Masalah utama pada klasifikasi dokumen adalah pada dataset yang ada memiliki fitur yang besar, fitur yang tidak relevan dan redudansi fitur yang menyebabkan noise pada proses klasifikasi dan ini merupakan tantangan dalam proses klasifikasi [17]. Seleksi fitur dengan *particle swarm optimization* merupakan proses seleksi fitur dengan metode seleksi wrapper based [15].Proses dari seleksi fitur berbasis *particle swarm optimization* dapat dilihat pada gambar 2.2 [15].



Gambar 2.2 Proses Seleksi Fitur *Particle swarm optimization*

Proses Seleksi Fitur dengan PSO adalah sebagai berikut:

1. Inisialisasi posisi dan velocity partikel
2. Kumpulkan fitur yang sudah dipilih secara acak oleh partikel
3. Evaluasi fitur yang sudah dipilih dengan metode klasifikasi
4. Hitung nilai fitnesi tiap partikel, nilai fitness adalah error rate pada setiap partikel setelah di evaluasi menggunakan metode klasifikasi
5. Update pbest dan gbest berdasarkan iterasi yang berjalan saat ini

6. Hitung velocity baru tiap partikel dan lakukan update posisi berdasarkan nilai velocity yang didapat
7. Cek kondisi berhenti, apakah sudah mencapai kondisi berhenti atau sudah mencapai iterasi maksimum
8. Jika iterasi maksimun dan solusi yang ada belum sesuai dengan kondisi berhenti maka lakukan kembali proses seleksi fitur pso degan posisi partikel yang baru
9. Jika kondisi berhenti sudah terpenuhi dan iterasi sudah mencapai iterasi maksimum hentikan proses.

2.10 *Confussion Matriks*

Pada tahap evalusai bertujuan untuk mengetahui tingkat akurasi dari hasil penggunaan metode klasifikasi yang digunakan dengan cara menghitung jumlah data uji yang kelas prediksinya diprediksi dengan benar. Adapun untuk mengetahui kinerja dari sistem akan dievaluasi dengan menggunakan *confussion matriks. Confussion matriks* adalah matriks yang berisi data aktual dan data prediksi yang telah diklasifikasikan oleh sistem.

Tabel 2.1 *Confussion Matriks*

Prediksi		Aktual/Fakta	
		1	0
Kelas Sebenarnya	1	TP	TN
	0	FP	FN

Keterangan :

1. *True Positive* (TP) merupakan jumlah dokumen dari kelas 1 yang benar diklasifikasikan sebagai kelas 1.
2. *False Positive* (FP) merupakan jumlah dokumen dari kelas 0 yang salah diklasifikasikan sebagai kelas 1.
3. *False Negative* (FN) merupakan jumlah dokumen dari kelas 1 yang salah diklasifikasikan sebagai kelas 0.

4. *True Negative* (TN) merupakan jumlah kelas dari kelas 0 yang benar di klasifikasikan sebagai kelas 0.

Kemudian akan dihitung akurasinya, akurasi merupakan perhitungan antara jumlah prediksi benar dan dibagikan dengan jumlah prediksi, untuk menghitung akurasi digunakan rumus sebagai berikut [18]:

$$accuracy = \frac{Jumlah\ prediksi\ benar}{jumlah\ seluruh\ prediksi\ yang\ dilakukan} \quad (2.8)$$

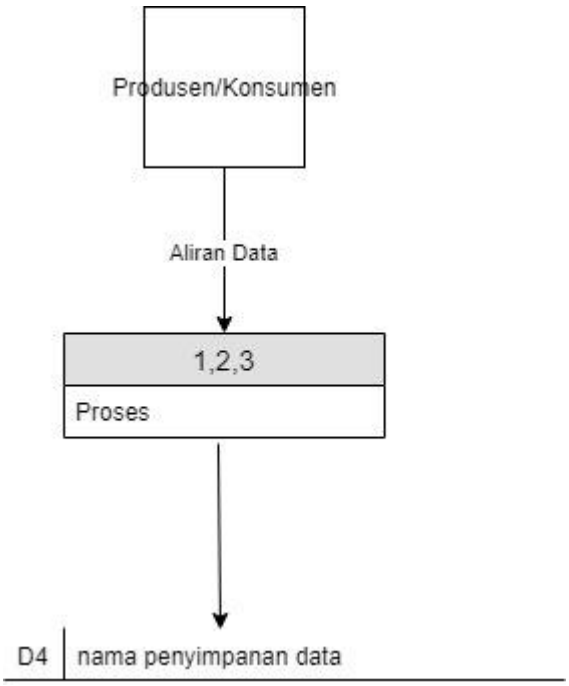
2.11 Pemrograman Terstruktur

Pemrograman terstruktur adalah konsep atau paradigma atau sudut pandang pemrograman yang membagi program berdasarkan fungsi-fungsi atau prosedur yang dibutuhkan program komputer [19]. Modul-modul (pembagian program) biasanya dibuat dengan mengelompokkan fungsi-fungsi dan prosedur-prosedur yang diperlukan sebuah proses tertentu.

Fungsi-fungsi dan prosedur-prosedur ditulis secara sekuensial atau terurut dari atas kebawah sesuai dengan kebergantungan antar fungsi atau prosedur. Pemodulan pada pemrograman terstruktur lebih berfokus bagaimana memodelkan data dan fungsi-fungsi atau prosedur-prosedur yag harus dibuat. Jenis paradigma pemrograman yang digunakan dapat dideteksi dari bahasa pemrograman apa yang digunakan untuk membuat program, baru setelah itu ditentukan paradigma pemrograman apa yang akan digunakan.

2.11.1. *DFD (*Data Flow Diagram)

Data flow diagram awalnya dikembangkan oleh Chris Gane dan Trish Sarson pada tahun 1979 yang termasuk dalam *Structured System Analysis and Design Methodology* (SSADM). Sistem yang dikembangkan ini berbasis pada dekomposisi fungsional dari sebuah sistem. Berikut adalah contoh DFD yang dikembangkan oleh Chris Gane dan Trish Sarson dapat dilihat pada gambar 2.3.



Gambar 2.3 Contoh DFD yang dikembangkan Chris Gane dan Trish Sarson

Edward Yourdon dan Tom Demarco memperkenalkan metode yang lain pada tahun 1980-an dimana mengubah persegi dengan lingkaran untuk menotasikan [19]. DFD Edward Yourdon dan Tom Demarco populer digunakan sebagai model analisis sistem perangkat lunak untuk sistem perangkat lunak yang dibangun secara terstruktur.

Informasi yang ada didalam perangkat lunak dimodifikasi dengan beberapa transformasi yang dibutuhkan. *Data flow diagram* (DFD) adalah repesentasi grafik yang menggambarkan aliran informasi dan trasnformasi informasi yang diaplikasikan sebagai data yang mengalir dari masukan (input) dan keluaran (output) [19].

DFD dapat digunakan untuk merepesentasikan sebuah sistem perangkat lunak pada beberapa level abstraksi. DFD dapat dibagi menjadi beberapa level yang lebih detail untuk merepesentasikan aliran informasi atau fungsi yang lebih detail. DFD menyediakan mekanisme untuk pemodelan fungsional ataupun pemodelan aliran informasi. Oleh karena itu DFD lebih sesuai digunakan untuk memodelkan fungsi-fungsi perangkat lunak yang akan diimplementasikan menggunakan pemrograman terstruktur, karena pemrograman terstruktur membagi-bagi bagian

programnya dengan fungsi-fungsi dan prosedur-prosedur. Berikut ini adalah tahapan – tahapan perancangan menggunakan DFD:

1. Membuat DFD level 0 atau sering disebut dengan diagram konteks. DFD level 0 menggambarkan sistem yang akan dibuat sebagai suatu entitas tunggal yang berinterkasi dengan orang maupun sistem lain. DFD level 0 digunakan untuk menggambarkan interaksi antara sistem yang akan dikembangkan dengan entitas luar [19].
2. Membuat DFD level 1 digunakan untuk menggambarkan modul-modul yang ada dalam sistem yang akan dikembangkan. DFD level 1 merupakan hasil breakdown DFD level 0 yang sebelumnya sudah dibuat.
3. Membuat DFD level 2 modul-modul pada DFD level 1 dapat di breakdown menjadi lebih detail tergantung pada tingkat kedetailan modul tersebut. Apabila modul tersebut sudah cukup detail dab rinci maka modul tersebut sudah tidak perlu untuk di breakdown lagi. Untuk sebuah sistem jumlah DFD level 2 sama dengan jumlah modul pada DFD level 1 yang di breakdown.
4. Membuat DFD level 3,4 dan seterusnya merupakan breakdown dari modul DFD pada level diatasnya. Breakdown pada level 3,4,5 dan seterusnya aturannya sama seperti pada DFD level 1 atau level 2 [19].

Pada satu diagram DFD sebaiknya jumlah modul tidak boleh lebih dari 20 buah. Jika lebih dari 20 buah modul diagram akan terlihat rumit dan susah untuk dibaca sehingga menyebabkan sistem yang akan dibangun menjadi rumit [19].

2.12 Python

Python adalah bahasa pemrograman interpretatif, interaktif dan *object oriented* (OO) yang diciptakan oleh Guido van Rossum pada tahun 1991 dan dikembangkan oleh Python Software Foundation dari versi 2.1 sampai saat ini [20]. Python mencakup modul, *exceptions*, *dynamic typing*, *very high level dynamic data types*, dan *classes*. Python memiliki *syntax* yang sangat bersih dan memiliki *interfaces* ke *banyak system calls* dan *libraries*, serta ke berbagai *windows systems*, dan *extensible* di C atau C++. Selain itu, python bisa digunakan sebagai *extension language* untuk aplikasi yang membutuhkan *interface* yang dapat diprogram.

Python bersifat *protable* dan bisa berjalan diberbagai varian Unix, Mac dan Windows 2000 sampai yang terbaru.

Python adalah *high-level general-purpose programming language* yang bisa sdapat diterapkan ke banyak kelas masalah yang berbeda-beda [20]. Bahasa ini dilengkapi dengan *standard library* besar yang mencakup bidang-bidang seperti pemrosesan string (ekspresi reguler, Unicode, penghitungan perbedaan antara file), protokol Internet (HTTP, FTP, SMTP, XML-RPC, POP, IMAP, pemrograman CGI), rekayasa perangkat lunak (pengujian unit, pencatatan, pembuatan profil, penguraian kode Python), dan *operating system interfaces* (*system calls*, *filesystems*, soket TCP / IP. Selain itu, Berbagai *extension* pihak ketiga juga banyak tersedia. Python telah berhasil digunakan di ribuan aplikasi bisnis di seluruh dunia, berikut ini beberapa aplikasi yang menggunakan python [21], antara lain youtube.com, Industrial Light & Magic, Google, Journyx, IronPort, EVE Online, HomeGain, Thawte Consulting, University of Maryland, EZTrip.com, RealEstateAgent.com dan Firaxis Games. Ada beberapa *Library* yang umum digunakan dengan *python*, antara lain: TensorFlow, Keras dan lain-lain.