BAB II
TINJAUAN PUSTAKA

2.1 Kajian Terkait

Untuk penelitian klasifikasi emosi menggunakan lirik lagu menggunakan *Support Vector Machine* sudah ada yang melakukannya, seperti penelitian oleh (Abirawa et al., 2018) dengan berbagai ekspresi emosi (marah, sedih, senang, tenang, dll.) yang memiliki nilai akurasi yang terbaik berada di nilai akurasi 62.50%.

Pada penelitian klasifikasi emosi yang lain, khususnya menggunakan lirik lagu berbahasa Indonesia seperti pada (Salekhah, 2016) menggunakan metode *Multi Class Support Vector Machine* yang menghasilkan nilai akurasi sebesar 36.67% dan dapat ditarik kesimpulan bahwa semakin besar data latih, nilai akurasi akan semakin menurun. Hal ini dapat terjadi karena pada penelitian tersebut tidak menggunakan seleksi fitur, yang mengakibatkan banyak fitur yang tidak relevan dengan proses pengklasifikasian ikut masuk pada proses klasifikasi, sehingga terjadi penurunan akurasi.

Melanjutkan penelitian diatas, pada penelitian oleh (Azizah & Rainarli, 2019) masalah yang ditemukan adalah pada penelitian sebelumnya yaitu data latih yang digunakan berbahasa Inggris sehingga saat diterjemahkan kedalam bahasa indonesia terjadi kesalahan makna kosakata. Solusi yang diberikan adalah mengganti data latih dengan lirik lagu bahasa Indonesia yang telah divalidasi oleh ahli bahasa dan menambahkan seleksi fitur *Information Gain* karena seleksi fitur dapat mendapatkan fitur yang relevan dalam mendeteksi emosi pada lagu sehingga dapat meningkatkan kinerja metode klasifikasi.digunakan data latih lagu bahasa Indonesia yang telah divalidasi oleh ahli bahasa. Hasil pengujian menunjukan bahwa penggunaan *Support Vector Machine* dan seleksi fitur *Information Gain* dengan data uji sebanyak 20 lagu menunjukkan nilai akurasi sebesar 70%, dibandingkan dengan penggunaan *Support Vector Machine* tanpa seleksi fitur yang menunjukkan nilai akurasi hanya sebesar 55%.

Maka untuk meningkatkan akurasi, mengacu penelitian oleh (Que et al., 2020) penelitian ini menggunakan metode *Support Vector Machine* (SVM) dan

penggunaan *Particle Swarm Optimization* (PSO) untuk mengatur parameter SVM agar lebih optimal. Penelitian ini membandingkan metode SVM dan SVM-PSO tujuannya adalah untuk membandingkan hasil sentimen positif dan negatif dan akurasi. Data yang digunakan adalah 1.852 *tweet*, yang dibagi menjadi 1.130 data *training*, dan 722 data *testing* yang menghasilkan akurasi sebesar 95,46% untuk metode SVM dan sebesar 96,04% untuk metode SVM-PSO, menunjukkan peningkatan sebesar 0,58% untuk metode SVM-PSO.

Tabel 2. 1 Rangkuman Kajian Terkait

Peneliti	Judul	Metode	Kesimpulan
(Abirawa et al., 2018)	Klasifikasi Emosi Pada Lirik Lagu Menggunakan Metode *Support Vector Machine*	*Support Vector Machine* (SVM)	Nilai akurasi 62.50%
(Salekhah, 2016)	Implementasi Metode *Multi Class Support Vector Machine* Untuk Klasifikasi Emosi Pada Lirik Lagu Bahasa Indonesia	*Multi Class Support Vector Machine* (SVM)	Nilai akurasi 36.67%
(Azizah & Rainarli, 2019)	*Support Vector Machine* Dan *Information Gain* Untuk Klasifikasi Emosi Pada Lirik Lagu	*Support Vector Machine* (SVM) dan *Information Gain* (IG)	Nilai akurasi SVM 55% Nilai akurasi SVM-IG 70%
(Que et al., 2020)	Analisis Sentimen Transportasi *Online* Menggunakan *Support Vector Machine* Berbasis *Particle Swarm Optimization*	SVM dan SVM-PSO	Nilai akurasi SVM 95.46% Nilai akurasi SVM-PSO 96.04%

2.2 Lirik Lagu

Lirik lagu merupakan bagian penting pada lagu, karena pada lirik dapat menggambarkan emosi dari lagu tersebut. Lirik lagu merupakan suatu penyampaian perasaan seseorang secara tidak langsung tehadap apa yang didengarnya (Sinaga et al., 2019). Lirik lagu dapat dimasukkan ke dalam genre puisi dalam karya sastra, lirik lagu adalah susunan kata sebuah nyanyian atau karya sastra yang berupa curahan perasaan pribadi oleh seseorang (Pentury, 2020). Puisi sendiri memiliki 2 konsep, yaitu puisi lama dan puisi baru. Puisi baru atau lebih dikenal dengan sebutan puisi *modern* ini adalah lawan dari puisi lama. Jika pada puisi lama itu terikat dengan aturan-aturan unsur yang membentuknya, pada puisi *modern* ini

lebih bebas, dalam artian tidak terikat aturan-aturan seperti pada puisi lama, pusi baru atau *modern* ini lebih bebas dalam menggunakan rima-rima, baris tiap baitnya, kata tiap barisnya dan sebagainya. Puisi *modern* ini juga lebih lepas membangun imajinasi atau ide-ide kreatif yang ingin disampaikan oleh si penulis puisi namun tetap memperhatikan etika dan estetika dari sastra puisi itu sendiri (Sulkifli & Marwati, 2016). Jika melihat dari definisi dan strukturnya maka lirik lagu merupakan salah satu bentuk dari puisi baru. Dalam penelitian ini, akan menggunakan lirik lagu sebagai *dataset*. *Dataset* yang digunakan terdiri dari 3 bentuk, yaitu *dataset* perbaris, perbait, dan keseluruhan lagu.

2.3 Pelabelan Data

Dataset yang telah dikumpulkan akan dilabeli secara manual oleh 3 orang, sehingga akan mengurangi kecendrungan opini pribadi. Menurut (Febrianti, 2020) pelabelan manual adalah dengan membaca *dataset* satu persatu, lalu akan ditentukan *dataset* tersebut masuk kedalam kelasnya masing-masing. Kelebihan dari pelabelan dengan cara ini adalah hasil yang didapatkan akan lebih sesuai dengan kenyataan dan tidak memperlukan proses komputasi. Kelemahan dari cara ini adalah dikhawatirkan adanya kecenderungan opini seseorang dan cara ini akan sulit diterapkan pada data yang berjumlah besar. Pada penelitian ini, pelabelan akan dilakukan berdasarkan 5 kelas emosi yaitu *anger* (marah), *fear* (takut), *sadness* (sedih), *love* (cinta). *Dataset* yang digunakan terdiri dari 3 bentuk, yaitu *dataset* perbaris, perbait, dan keseluruhan lagu.

2.4 Preprocessing

Preprocessing merupakan tahapan penting dalam khususnya dalam *Text Classification* (TC). *Preprocessing* adalah tahapan mereduksi sebuah kalimat atau beberapa kata menjadi satu bentuk kata tunggal. Tujuan utama dari *preprocessing* teks adalah untuk mendapatkan suatu bentuk kata tunggal (*root word*) dari sebuah dokumen, dengan menghapus fitur non-informatif seperti (kata sambung, imbuhan, angka dan karakter khusus) (Kadhim, 2018).

2.5 Case Folding

Case folding merupakan proses dalam *text preprocessing* yang dilakukan

untuk menyeragamkan karakter pada data. Proses *case folding* adalah proses mengubah seluruh huruf menjadi huruf kecil. Pada proses ini karakter-karakter ‘A’-‘Z’ yang terdapat pada data diubah kedalam karakter ‘a’-‘z’. Karakter-karakter selain huruf ‘a’ sampai ‘z’ (tanda baca dan angka) akan dihilangkan dari data dan dianggap sebagai delimiter (Benbrahim & Bramer, 2009).

2.6 Cleaning

Data cleaning merupakan proses pembersihan kata dengan menghilangkan delimiter (koma, titik, dan tanda baca lainnya). Pembersihan dilakukan untuk mengurangi *noise* (Luqyana, 2018).

2.7 Normalisasi

Tahapan ini bertujuan untuk mengembalikan bentuk penulisan dari masing masing kata yang sesuai dengan KBBI. Proses ini dilakukan dengan mencocokkan setiap kata pada dokumen data latih maupun data uji dengan kata yang ada pada kamus bahasa tidak baku (Luqyana, 2018).

2.8 Tokenisasi

Tokenisasi adalah proses untuk memotong *document* menjadi pecahan kecil yang dapat berupa bab, sub-bab, paragraf, kalimat, dan kata (token). Pada proses ini akan menghilangkan *whitespace* (Luqyana, 2018).

2.9 Stopword Removal

Stopword adalah salah satu metode yang paling sering digunakan untuk menghapus kata sambung seperti (*a, an, the, from,* pada bahasa Inggris) karena kata tersebut dianggap tidak terlalu penting untuk dijadikan kata kunci pada saat melakukan pencarian. Pada umumnya bahasa di berbagai negara memiliki kata sambung seperti yang ada pada artikel dan preposisi yang hampir selalu muncul pada dokumen teks (Benbrahim & Bramer, 2009).

Setiap kata akan diperiksa apakah masuk ke dalam *stoplist* atau tidak, jika sebuah kata masuk ke dalam *stoplist* maka kata tersebut tidak akan diproses lebih lanjut dan akan dihilangkan. Sebaliknya apabila sebuah kata tidak termasuk ke

dalam *stoplist* maka kata tersebut akan masuk ke proses berikutnya. *Stoplist* yang digunakan diambil dari *library* nltk (Lopes et al., 2019). Kata-kata yang termasuk *stopword* tersebut biasanya berupa kata ganti orang, kata penghubung, dan lain sebagainya.

2.10 Stemming

Stemming adalah proses yang memetakan kata yang berbeda ke satu kata dasar / umum (*stem*), proses ini juga dikenal sebagai penggabungan. Berdasarkan asumsi bahwa istilah-istilah yang memiliki satu *stem* yang sama biasanya akan memiliki arti yang sama, proses ini sering digunakan dalam pencarian informasi sebagai cara untuk meningkatkan kinerja. Selain kemampuannya untuk meningkatkan kinerja pencarian, proses *stemming* yang dilakukan pada waktu pengindeksan, juga akan mengurangi ukuran file indeks (Tala, 2003).

2.11 Rejoin

Rejoin adalah penggabungan kembali token-token yang telah dipecah dalam tahapan tokenisasi, agar kembali menjadi kalimat-kalimat yang utuh (Lorento, 2022).

2.12 Pembobotan TF-IDF

Metode *Term Frequency-Inverse Document Frequency* (TF-IDF) adalah cara pemberian bobot hubungan suatu kata (*term*) terhadap dokumen. Untuk dokumen tunggal tiap kalimat dianggap sebagai dokumen. Metode ini menggabungkan dua konsep untuk perhitungan bobot, yaitu *Term frequency* (TF) merupakan frekuensi kemunculan kata (t) pada kalimat (d). *Document frequency* (DF) adalah banyaknya kalimat dimana suatu kata (t) muncul.

Frekuensi kemunculan kata di dalam dokumen yang diberikan menunjukkan seberapa penting kata itu di dalam dokumen tersebut. Frekuensi dokumen yang mengandung kata tersebut menunjukkan seberapa umum kata tersebut. Bobot kata semakin besar jika sering muncul dalam suatu dokumen dan semakin kecil jika muncul dalam banyak dokumen (Auliaguntary Arif Putra, 2016).

Pada aloritma TF-IDF digunakan rumus untuk menghitung bobot (W) masing masing dokumen terhadap kata kunci dengan rumus yaitu :

$$W_{dt} = tf_{dt} * IDF_t \quad (2.1)$$

Dimana:

d = dokumen ke-d

t = kata ke-t dari kata kunci

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

tf = banyaknya kata yang dicari pada sebuah dokumen

IDF = Inversed Document Frequency

IDF = log (D/df)

D = total dokumen

df = banyak dokumen yang mengandung kata yang dicari

Setelah bobot (W) masing-masing dokumen diketahui, maka dilakukan proses *sorting*/pengurutan dimana semakin besar nilai W, semakin besar tingkat similaritas dokumen tersebut terhadap kata kunci, demikian sebaliknya. *Inverse Document Frequency* memperhatikan kemunculan *term* pada kumpulan dokumen. Pada metode ini, *term* yang dianggap bernilai/berharga adalah *term* yang jarang muncul pada koleksi/ kumpulan dokumen (Amin, 2012). Persamaan IDF adalah sebagai berikut:

$$IDF(t) = log \frac{N}{df(t)} \quad (2.2)$$

Dimana $df(t)$ adalah banyak dokumen yang mengandung *term* t.

TF*IDF merupakan kombinasi metode TF dengan metode IDF. Sehingga

persamaan TF*IDF adalah sebagai berikut:

$$TF * IDF(d,t) * IDF(t) \quad (2.3)$$

Perhitungan bobot *query relevance* merupakan bobot hasil perbandingan kemiripan (similaritas) antara *query* yang dimasukkan oleh *user* terhadap keseluruhan kalimat. Sedangkan bobot similarity kalimat, merupakan bobot hasil perbandingan kemiripan antar kalimat.

2.13 Syinthetic Minority Oversampling Technique (SMOTE)

Data tidak seimbang (*Imbalanced Data*) merupakan suatu keadaan dimana distribusi kelas data tidak seimbang, jumlah kelas data (*instance*) yang satu lebih sedikit atau lebih banyak dibanding dengan jumlah kelas data lainnya. Kelompok kelas data yang lebih sedikit dikenal dengan kelompok minoritas (*minority*), kelompok kelas data yang lainnya disebut dengan kelompok mayoritas (*mayority*) (Siringoringo, 2018). Pada umumnya melakukan klasifikasi pada data yang tidak seimbang akan mengganggu kinerja klasifikasi, karena akan cenderung ke kelompok mayoritas dan mengabaikan kelompok minoritas. Berdasarkan (Google Developer, 2019) data tidak seimbang terbagi menjadi 3 kategori berdasarkan presentasi proporsi kelompok minoritas yaitu *mild, moderate* dan *extreme* seperti yang ditampilkan pada gambar 2.1 dibawah ini.

Degree of imbalance	Proportion of Minority Class
Mild	20-40% of the data set
Moderate	1-20% of the data set
Extreme	<1% of the data set

Gambar 2. 1 *Imbalance Data*

Karena masalah *imbalanced data* tersebut maka dibutuhkan suatu teknik untuk menyeimbangkan *dataset* agar menghasilkan data yang lebih akurat.

Beberapa teknik yang sering digunakan untuk mengatasi permasalahan ini adalah *random oversampling* (ROS) dan *random undersampling* (RUS). ROS membuat salinan/duplikat yang sama persis dari data yang berasal dari kelas minoritas, akan tetapi kekurangan teknik ini dapat menyebabkan *overfitting*. Sedangkan RUS membuang data (yang berasal dari kelas yang bersifat mayoritas) secara acak, kekurangan dari teknik ini dapat menyebabkan *underfitting*. Salah satu teknik yang mirip dengan *oversampling* adalah dengan membuat data sampel sintetik, yaitu *synthetic minority oversampling technique* (SMOTE) (Boyle, 2019), metode ini diusulkan pertama kali pada tahun 2002 oleh Chawla, perbedaannya dengan teknik ROS ada pada sampel yang dihasilkan, tidak diduplikat secara *random* dari sampel yang sudah ada, tetapi sampel tersebut dibuat dengan mempergunakan konsep *nearest neighbour* (Arifiyanti & Wahyuni, 2020). Teknik ini bekerja dengan membuat replikasi dari data minoritas. Replikasi tersebut dikenal dengan data sintetis (*syntetic data*). Metode SMOTE bekerja dengan mencari k-*nearest neighbour* (yaitu ketetanggaan terdekat data sebanyak k) untuk setiap data di kelas minoritas, setelah itu dibuat data sintetis sebanyak prosentase duplikasi yang diinginkan antara data minor dan k-*nearest neighbour* yang dipilih secara acak (Siringoringo, 2018). SMOTE akan digunakan pada penelitian ini dengan fungsi *library imblearn*.

2.14 Metode Support Vector Machine (SVM)

SVM merupakan metode klasifikasi untuk data linear dan nonlinear. Singkatnya, SVM adalah algoritma yang bekerja menggunakan pemetaan nonlinear untuk mengubah data pelatihan asli menjadi dimensi yang lebih tinggi. Dalam dimensi baru ini, ia mencari *hyperplane* yang memisahkan optik linear (yaitu, "batas keputusan" memisahkan *tuple* dari satu kelas dari kelas yang lain). Dengan pemetaan nonlinier yang tepat ke dimensi yang cukup tinggi, data dari dua kelas selalu dapat dipisahkan oleh *hyperplane*. SVM berusaha menemukan *hyperplane* menggunakan vektor dukungan ("esensial" pelatihan *tuples*) dan *margin* (ditentukan oleh vektor dukungan) (Hermanto et al., 2020).

Teknik ini termasuk dalam metode klasifikasi jenis terpandu (*supervised*) karena memiliki target pembelajaran tertentu. Klasifikasi dilakukan dengan

mencari *hyperplane* atau garis pembatas (*decision boundary*) yang memisahkan antara satu kelas dengan kelas lainnya. Dalam konsep ini, SVM berusaha untuk mencari *hyperplane* terbaik diantara fungsi yang tidak terbatas jumlahnya. Fungsi yang tidak terbatas dalam pencarian *hyperplane* di metode Support Vector Machine merupakan sebuah keuntungan, dimana pemrosesan pasti akan selalu bisa dilakukan bagaimanapun data yang dimilikinya (Hermanto et al., 2020).

Tahapan dalam metode *Support Vector Machine* berdasarkan (Tineges et al., 2020) adalah sebagai berikut :

1. Menentukan kata yang sering muncul dari tiap dokumen atau teks yang digunakan.
2. Menentukan inisialisasi awal untuk nilai $\alpha = 0.5$, $\mathrm{C} = 1$,$\lambda = 0.5$, $\gamma = 0.5$, dan $\varepsilon = 0.5$.
3. Menghitung matriks dengan rumus:

$$D_{ij} = y_i y_j (K(\overrightarrow{x_i}.\overrightarrow{x_j}) + \lambda^2) \quad (2.4)$$

Keterangan :

D_{ij} = elemen matriks data ke-ij
y_i = kelas atau label data ke-i
y_j = kelas atau label data ke-j
λ = turunan batas teoritis
$K(\overrightarrow{x_i}.\overrightarrow{x_j})$ = fungsi kernel

4. Untuk data ke – $n = 1,2,3, \ldots, n$ gunakan persamaan berikut ini :

$$\varepsilon_i = \Sigma_{j=1}^{n} D_{ij} \quad (2.5)$$

$$\delta\alpha i = \min\{max[\gamma(1-Ei), -\alpha i], C-\alpha i\} \quad (2.6)$$

$$\alpha i = \alpha i + \delta\alpha i \quad (2.7)$$

Keterangan :

ε_i = nilai error data ke - i
γ = tingkat pembelajaran
$max_{(i)} D_{ij}$ = nilai maksimum diagonal matriks hessian

5. Mencari nilai bias (b) dengan menggunakan persamaan :

$$b = -\frac{1}{2}[\ \mathrm{w} \cdot x^{+} + \mathrm{w} \cdot x^{-}\ ] \quad (2.8)$$

6. Pengujian pada dokumen yang diuji
7. Perhitungan keputusan sebagai berikut:

$$h(x) = \begin{cases} +1, & if\ w \cdot x + b \geq 0 \\ -1, & if\ w \cdot x + b \leq 0 \end{cases} \quad (2.9)$$

8. Jika hasil perhitungan keputusan lebih dari sama dengan 0 maka nilai adalah +1, maka termasuk kelas positif dan jika hasil perhitungan keputusan kurang dari 0 maka nilai nilai -1, maka termasuk kelas negatif. Perhitungan keputusan dengan menggunakan persamaan :

$$h(x) = w \,.\, x + b$$
atau
$$h(x) = \Sigma_{i=1}^{m} a_i y_i K(x, x_i) + b \quad (2.10)$$

Pada awalnya SVM dikembangkan untuk persoalan klasifikasi dua kelas, kemudian dikembangkan kembali untuk klasifikasi multi kelas. Dalam klasifikasi kasus multi kelas, *hyperplane* yang terbentuk adalah lebih dari satu. Salah satu metode pendekatan yang digunakan adalah *One Versus All*. Sesuai dengan namanya, satu lawan semua, metode ini membandingkan satu kelas dengan semua kelas lainnya. Untuk mengklasifikasikan data ke dalam k kelas, Anda harus membangun sejumlah k model SVM *biner*. Setiap model SVM *biner* ke-i dilatih menggunakan keseluruhan data, untuk mencari jawaban apakah sebuah data diklasifikasikan sebagai kelas ke-i atau tidak. Misalnya, Anda memilliki masalah klasifikasi dengan 4 kelas. Anda harus membangun 4 buah SVM *biner*, dimana SVM *biner* pertama dilatih menggunakan semua data latih untuk mengklasifikasikan data ke dalam kelas C, atau bukan, SVM *biner* kedua dilatih menggunakan semua data latih untuk mengklasifikasikan data ke dalam kelas C atau bukan, dan seterusnya (Suyanto, 2019).

Dalam penelitian ini, *multi class* SVM akan menggunakan pendekatan *One Versus All*, karena memiliki nilai akurasi atau performa yang lebih baik dibandingkan dengan pendekatan lainnya (Alita et al., 2020). Untuk fungsi kernel yang digunakan adalah fungsi kernel RBF, karena memiliki performansi yang paling baik dibandingkan kernel linier pada parameter tertentu maupun kernel polinomial (Pratama & Trilaksono, 2015). pemodelan SVM pada penelitian ini akan menggunakan fungsi dari *library sklearn*.

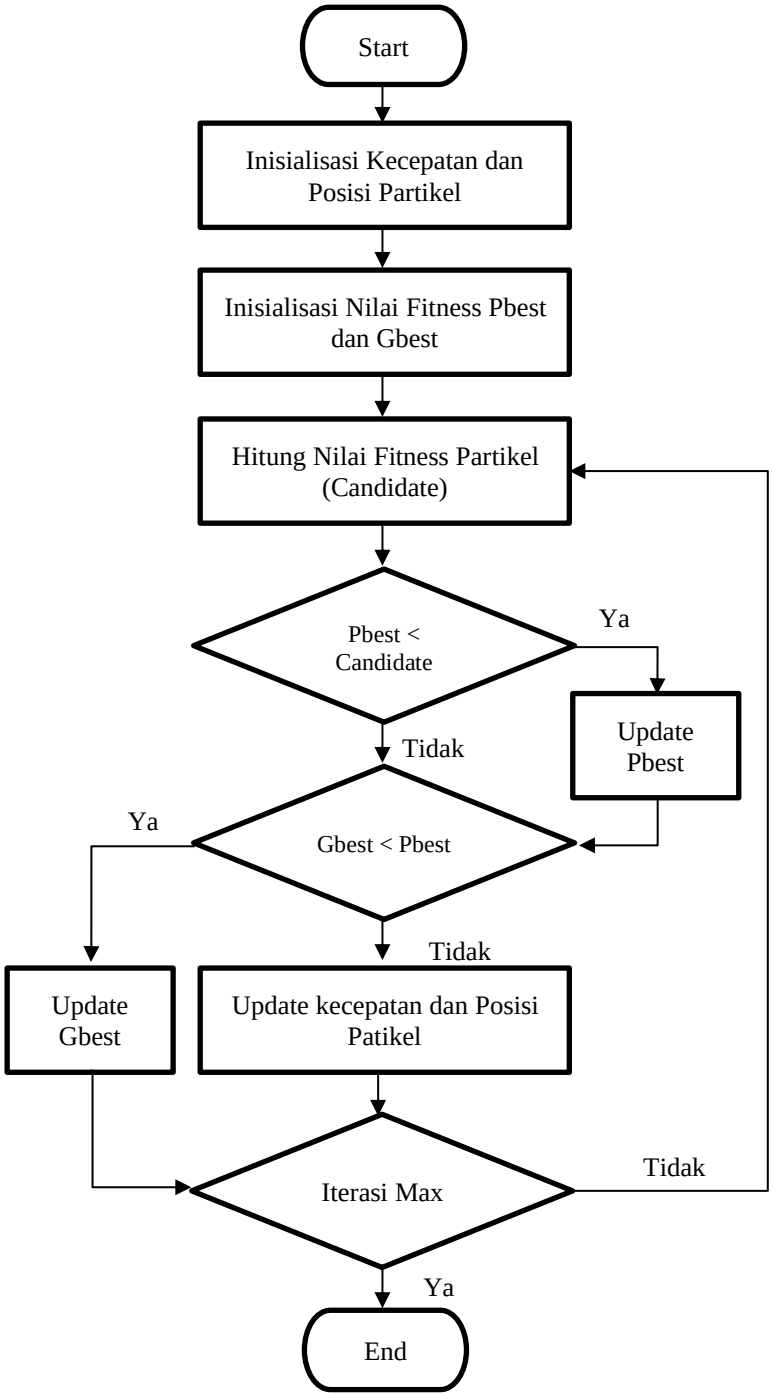
2.15 Particle Swarm Optimization (PSO)

PSO terinspirasi oleh tabiat sekelompok burung yang terbang bergerombol atau ikan yang berenang bergerombol. Ratusan burung atau ratusan ikan dapat bergerak dengan cepat tanpa saling bertabrakan, padahal jarak mereka sangat dekat.

Kelebihan PSO adalah sederhana, mudah diterapkan, dan kecepatan konvergensinya. Data mentah atau dalam kalimat opini dikonversi ke dalam numerik dan TF-IDF digunakan untuk mengubah data opini menjadi numerik (Que et al., 2020).

Pencarian solusi pada algoritma PSO oleh populasi tertentu berdasarkan sejumlah partikel. Populasi dinilai secara acak dan memiliki batasan nilai terkecil dan terbesar. Partikel - partikel melacak solusi dengan melalui ruang pencarian dengan cara beradaptasi terhadap letak terbaiknya *(local best)* dan beradaptasi terhadap letak partikel terbaik pada seluruh kelompok *(global best)* sewaktu melalui *search space* (Que et al., 2020).

Particle Swarm Optimization (PSO) dimulai dengan suatu populasi yang terdiri atas sejumlah partikel (yang menyatakan calon-calon solusi) yang dibangkitkan secara acak. Selanjutnya dilakukan perbaruan posisi dan kecepatan terbang setiap partikel secara iteratif untuk menghasilkan solusi baru yang lebih baik. *Particle Swarm Optimization* (PSO) akan berhenti ketika solusi optimum telah ditemukan atau kondisi tertentu telah tercapai. Dengan konsep ini, *Particle Swarm Optimization* (PSO) mudah diimplementasikan dengan sedikit pengaturan parameter (Saputra et al., 2019).



Gambar 2. 2 *Flowchart* Algoritma PSO

Pada Gambar 2.2 ditampilkan flowchart PSO yang dibangun berdasarkan (Que et al., 2020) dengan skema seperti dibawah ini :

1. Inisialisasi kecepatan dan posisi partikel
2. Inisialisasi nilai *fitness Pbest* (Personal) dan *Gbest* (Global)
3. Menghitung nilai fitness partikel (*Candidate*)

4. Membandingkan nilai fitness candidate dengan nilai Pbest. Jika nilai Pbest lebih kecil, maka nilai fitness candidate akan menjadi nilai Pbest yang baru
5. Membandingkan nilai Pbest dan Gbest. Jika nilai Gbest lebih kecil, maka nilai Pbest akan menjadi nilai Gbest yang baru.
6. Update kecepatan dan posisi partikel untuk iterasi selanjutnya
7. Mulai kembali dari langkah ke-2 hingga mencapai iterasi maksimum.

Berikut rumus dalam meng-*update* kecepatan dan posisi partikel:

$$v_{(i)} = \omega \, . v_{(i)} + c_1 r_1 \, (Pbest - x_{(i)}) + c_2 r_2 \, (Gbest - x_{(i)}) \quad (2.11)$$

$$x_{(i)} = x_{(i)} + v_{(i)} \quad (2.12)$$

Keterangan :

$v_{(i)}$	=	kecepatan partikel saat ini
$x_{(i)}$	=	posisi partikel saat ini
ω	=	berat inertia
c_1, c_2	=	koefisien akselerasi
r_1, r_2	=	bilangan acak dengan rentang 0 hingga 1
$Pbest$	=	posisi *personal best* partikel saat ini
$Gbest$	=	posisi *global best* partikel saat ini

Dalam penelitian ini, PSO akan digunakan sebagai *tuning hyperparameter* untuk mendapatkan *hyperparameter gamma* dan C yang optimal pada SVM (Que et al., 2020). Parameter *w*, c_1, dan c_2 yang akan digunakan masing-masing bernilai 0.9, 0.5, dan 0.3 berdasarkan (Lester James V. Miranda, 2017). Menurut (Piotrowski et al., 2020) jumlah partikel yang optimal bernilai 100, dengan iterasi sebanyak 2 kali sebagai kondisi tercapainya algoritma PSO oleh peneliti.

2.16 K-Fold Cross Validation

K-Fold Cross Validation adalah salah satu metode yang dapat memeriksa *overfitting* pada suatu model. Data yang dibagi menjadi k bagian membolehkan setiap bagian data berhenti memprediksi data lebih cepat ketimbang tidak dibagi terlebih dahulu (Barrow & Crone, 2013). Pada proses ini seluruh *dataset* akan diacak kemudian dibagi menjadi k bagian yang sama besar, di mana akan terdapat bagian yang menjadi data uji, sedangkan sisanya yang menjadi data latih. Contoh *k-fold cross validation* yang sering digunakan adalah *10-fold cross validation*.

Pada *10-Fold Cross Validation*, 1 bagian data akan menjadi data uji dan 9

bagian data lainnya akan menjadi data latih. Contohnya jika pada *dataset* terdapat 1000 data, akan dibagi ke dalam *10-fold*. *Fold* ke-1 akan menjadi data uji sedangkan sisanya menjadi data latih. Setelah training tahap pertama selesai, selanjutnya *training* dilakukan dengan menjadikan fold ke-2 sebagai data uji dan 9 data bagian lainnya menjadi data latih, seperti itu selanjutnya hingga *fold* ke -10 (Rangga et al., 2019).

2.17 Confusion Matrix

Confusion matrix adalah alat yang berguna untuk menganalisis seberapa baik pengklasifikasi dilakukan hingga dapat mengenali tupel dari kelas yang berbeda (Han et al., 2012). Menurut Suyanto (Suyanto, 2019), *confusion matrix* sangat berguna untuk menganalisis kualitas *classifier* dalam mengenali tuple-tuple dari kelas yang ada. Contoh *confusion matrix* untuk klasifikasi ditunjukkan pada tabel 2.2.

Tabel 2. 2 Tabel *Confusion Matrix*

		Kelas Hasil Prediksi		
		Ya	Tidak	Jumlah
Kelas Aktual	Ya	TP	FN	P
	Tidak	FP	TN	N
	Jumlah	P’	N’	P+N

Empat istilah sangat penting untuk memahami semua ukuran evaluasi dalam tabel 1 adalah sebagai berikut:

1. TP atau *True Positives* adalah jumlah tuple positif yang dilabeli dengan benar oleh *classifier*.
2. TN atau *True Negatives* jumlah tuple negatif yang dilabeli dengan benar oleh *classifier*.
3. FP atau *False Positives* adalah jumlah tuple negatif yang salah dilabeli oleh *classifier*.
4. FN atau *False Negatives* adalah jumlah tuple positif yang salah dilabeli oleh *classifier*.

Terdapat sejumlah ukuran yang dapat digunakan untuk menilai atau mengevaluasi model klasifikasi, diantaranya adalah: *accuracy*, *precision*, *recall* dan

f1-score. *Accuracy* menyatakan persentase dari jumlah tuple dalam data uji yang diklasifikasikan dengan benar oleh *classifier*, kemudian *precision* adalah ukuran kepastian, yaitu berapa persentase tuple yang dilabeli sebagai positif adalah benar pada kenyataannya, kemudian *recall* adalah ukuran kelengkapan, yaitu berapa persentase tuple positif yang dilabeli sebagai positif (Suyanto, 2019) dan *f1-score* yaitu *harmonic mean* dari nilai *recall* dan *precision* untuk mengetahui seberapa presisi dan handalnya performa model dalam menglasifikasikan kelas (Alita et al., 2020).

$$Accuracy = \frac{TP + TN}{P + N} \tag{2.12}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.13}$$

$$Recall = \frac{TP}{P} \tag{2.14}$$

$$F1 - Score = \frac{2(Precision\ x\ Recall)}{Precision + Recall} \tag{2.15}$$