BAB 2
LANDASAN TEORI

2.1 Analisis Sentimen

Analisis sentimen adalah salah satu analisis kunci yang saat ini digunakan dengan tujuan mengklasifikasikan sentimen dan opini yang dihasilkan oleh manusia dalam teks [14]. Sistem analisis sentimen dapat dilakukan melalui beberapa tahap.

1. *Data-crawling*, dilakukan dengan memberikan kata kunci selama jangka waktu tertentu. Setelah data diperoleh, dilakukan prosedur pelabelan untuk menilai sentimen. Tingkat preprocessing berikut diatur untuk pemilihan dan modifikasi data[15].

2. *Pre-processing*, tahap ini berguna untuk mempersiapkan data ke tahapan selanjutnya sehingga bisa meningkatkan performa model dalam proses klasifikasi, tahap ini dibagi menjadi beberapa sub-proses yang pada umumnya adalah *cleansing, case folding, tokenization, filtering* dan *stemming*[16].

3. *Feature selection*, tahap ini digunakan untuk identifikasi fitur-fitur untuk peningkatan performa pada klasifikasi yang akan dilakukan [17].

4. *Method Classifiers*, implementasikan metode atau model klasifikasi sesuai dengan masalah yang telah ditemukan.

2.2 TF-IDF

TF-IDF adalah salah satu metode untuk pemobobotan istilah atau kata. Secara khusus, metode ini digunakan untuk mengekstraksi kata-kata inti (yaitu, kata kunci) dari dokumen, menghitung derajat kesamaan antar dokumen, menentukan peringkat pencarian, dan sebagainya.

TF (*Term Frequency*) dalam TF-IDF berarti kemunculan kata-kata tertentu dalam dokumen. Kata-kata dengan nilai TF yang tinggi memiliki arti penting dalam dokumen. Nilai TF dapat dihitung dengan rumus 2.1 dengan nilai tf_t adalah jumlah kemunculan dari istilah t. [18].

$$tf_t = 1 + log(tf_t) \tag{2.1}$$

Di sisi lain, DF (*Document Frequency*) menyiratkan berapa kali kata tertentu muncul dalam kumpulan dokumen. Ini menghitung kemunculan kata di banyak dokumen, tidak hanya di satu dokumen. Kata-kata dengan nilai DF tinggi tidak memiliki arti penting karena biasanya muncul di semua dokumen. Oleh karena itu, IDF (*Inverse Document Frequency*) yang merupakan kebalikan dari DF digunakan untuk mengukur pentingnya kata-kata dalam semua dokumen. Nilai IDF yang tinggi berarti kata-kata yang jarang ditemukan di seluruh dokumen, sehingga meningkatkan tingkat kepentingannya[19]. Untuk menghitung nilai IDF dengan Persamaan 2.2

$$idf_t = log(\frac{D}{df_t}) \tag{2.2}$$

Dimana, nilai D adalah jumlah dokumen dan df_t adalah jumlah dokumen yang terdapat isitlah t [18]. Setelah perhitungan TF dan IDF dilakukan, dapat menghitung nilai TF-IDF demgan menggunakan Persamaan 2.3, dimana nilai $W_{t,d}$ adalah bobot istilah t pada dokumen d.

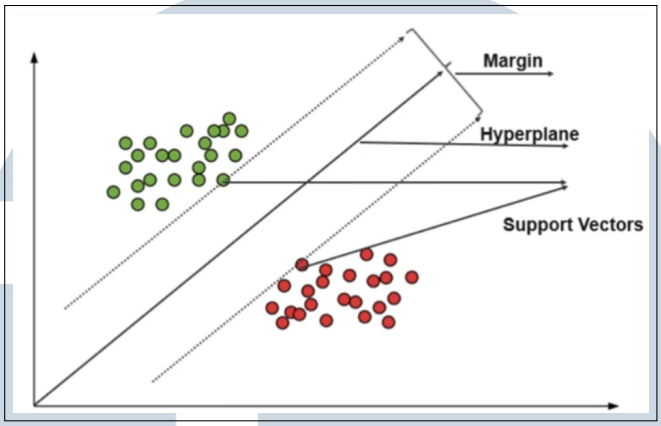
$$W_{t,d} = tf_t \times idf_t \tag{2.3}$$

2.3 Support Vector Machine

Support Vector Machine merupakan metode pembelajaran mesin yang sederhana dan fleksible dimana dapat diaplikasikan untuk mengatasi berbagai masalah klasifikasi, SVM secara khusus menghasilkan kinerja prediktif yang seimbang, bahkan dalam studi di mana ukuran sampel yang terbatas [20]. Metode SVM bekerja dengan meletakkan data-data cluster, lalu ditarik garis hyperplane atau garis pemisah (*decision boundary*) dari data cluster tersebut.

Garis pemisah pinggiran maksimum yang terletak di suatu ruang dan mengklasifikasikan data yang dipisahkan oleh batas non-linier atau linear yang dapat dibangun dengan menemukan sekumpulan hyperplane yang memisahkan dua atau lebih kelas titik data. Setelah konstruksi hyperplanes, SVM menemukan jarak antara kelas masukan dan elemen masukan terhadap garis disebut dengan vektor pendukung. Dari sekumpulan sampel training tertentu yang diberi label positif atau negatif, hyperplane membagi sampel training positif atau negatif, sehingga jarak antara margin dan hyperplane menjadi maksimal. Jika tidak ada hyperplane yang dapat membagi sampel positif atau negatif, SVM akan memilih hyperplane yang

membagi sampel sedekat mungkin, sambil tetap memaksimalkan jarak ke contoh terdekat yang terpecah secara ketat [21]. Ilustrasi dapat dilihat pada Gambar 2.1.



Gambar 2.1. Support Vector Machine
Sumber: [22]

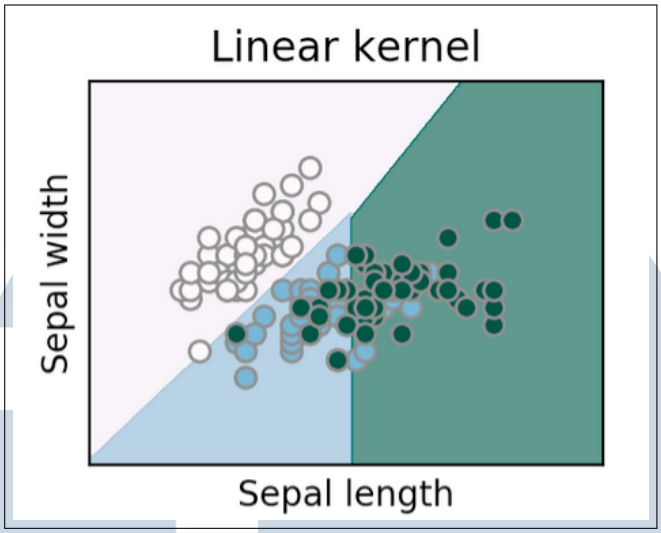
Garis ini memiliki margin atau jarak atar kelas terbesar menggunakan beberapa kernel yang diklasifikasikan antara lain, Linear, Radial Basic Function (RBF), dan Polynomial digunakan sebagai penentu performa dari model SVM. [23]

2.3.1 Linear Kernel

Merupakan kernel paling simpel, tanpa menggunakan nilai gamma (γ), dengan menggunakan Persamaan 2.4.

$$k(x_i.x_j) = x_i^T.x_j \tag{2.4}$$

Dimana nilai $k(x_i.x_j)$ merupakan nilai kernel, x_i adalah nilai data *training* dan x_j adalah nilai dari data *test*. Kernel ini cocok digunakan jika dataset yang diuji besar, namun dengan keakuratan yang cukup lemah. [24]. Ilustrasi dari kernel linear dapat dilihat pada Gambar 2.2



Gambar 2.2. Kernel Linear

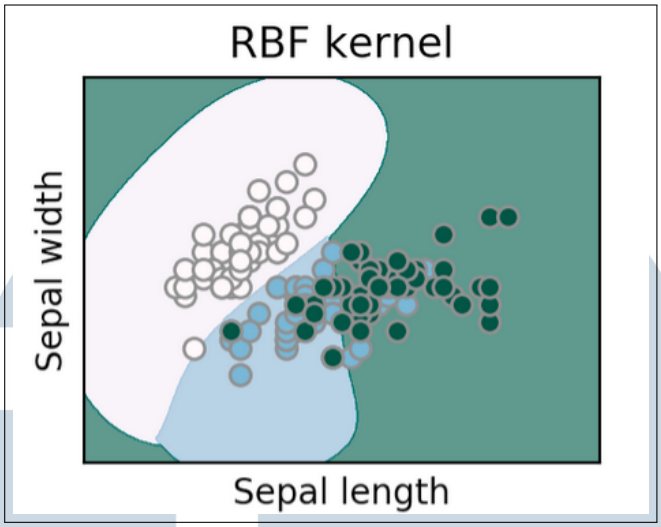
Sumber: [22]

2.3.2 Radial Basic Function Kernel

Merupakan kernel non-linear, menggunakan parameter nilai gamma ($\gamma > 0$) sebagai penentu fleksibilitas dari kernel ini, dapat dijabarkan dengan Persamaan 2.5.

$$k(x_i.x_j) = exp(-\gamma.||x_i.x_j||^2) \tag{2.5}$$

Kernel ini cocok digunakan untuk data yang tidak dapat dipecahkan secara linear dengan tingkat akurasi dan presisi yang tinggi [25]. Dapat dilihat pada Gambar 2.3.



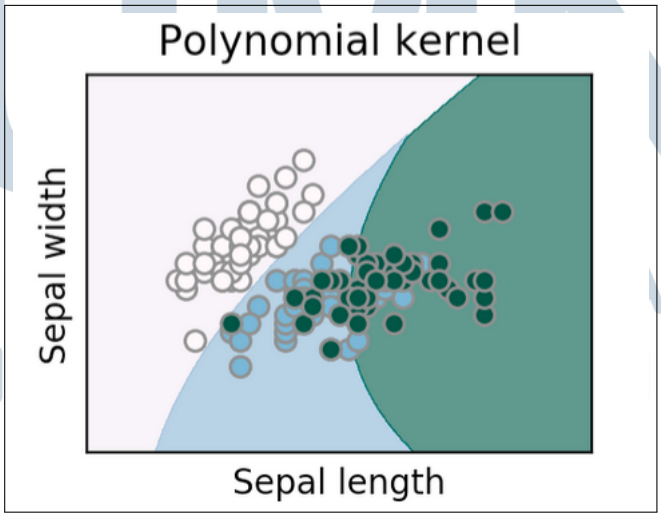
Gambar 2.3. Kernel Radial

Sumber: [22]

2.3.3 Polynomial Kernel

Merupakan kernel non-linear, menggunakan parameter nilai nilai gamma $(\gamma > 0)$ dan nilai d sebagai koefisien *degree* penenetu pinalti untuk fleksibilitas, dapat dijabarkan dengan Persamaan 2.6 dengan ilustrasi pada Gambar 2.4.

$$k(x_i.x_j) = \gamma(x_i^T.x_j + r)^d \tag{2.6}$$



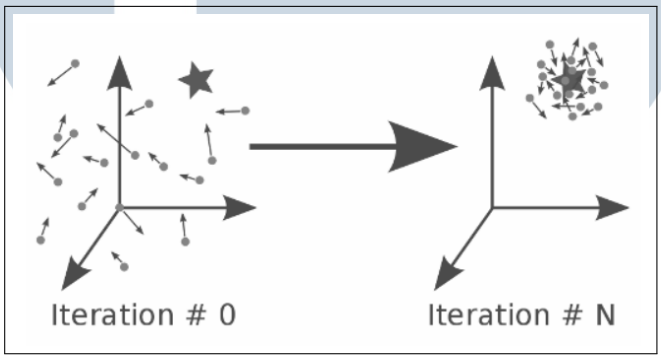
Gambar 2.4. Kernel Polynomial

Sumber: [22]

Nilai *gamma* yang besar akan menghitung juga data *training* yang jauh dari *decision boundary* namun akan menyebabkan nilai akurasi yang kecil, dan nilai *c* adalah parameter bebas, nilai *r* merupakan bias, parameter *gamma* dan *r* memiliki hubungan yang kuat [26].

2.4 Particle Swarm Optimization

PSO adalah suatu metode optimasi paling sederhana untuk memodifikasi beberapa parameter. Optimasi pada PSO dapat dilakukan dengan cara menyeleksi atribut (attribute selection) dan feature selection, serta meningkatkan bobot atribut (attribute weight) pada semua atribut atau variabel yang digunakan [27], ilustrasi algoritma PSO dapat dilihat pada Gambar 2.5



Gambar 2.5. Ilustrasi PSO

Sumber: [28]

Pada awalnya, partikel ditempatkan pada posisi menggunakan Persamaan 2.7 dan 2.8, lalu melakukan pencarian nilai optimal dari fungsi objektif tertentu melalui eksplorasi dan eksploitasi. Nilai fitness dari fungsi objektif pada posisi tersebut juga disimpan yang dihitung dengan Persamaan 2.9. [25].

$$x_0^i = x_{min} + rand(0,1)(x_{max} - x_{min}) \tag{2.7}$$

$$v_0^i = v_{min} + rand(0,1)(v_{max} - v_{min}) \tag{2.8}$$

$$F_{k+1}^i = f(X_{k+1}^i) \tag{2.9}$$

dimana:

x = posisi partikel.

v = kecepatan partikel.

i = indeks partikel.

$f(x)$ merupakan fungsi optimasi.

Setiap partikel akan memiliki nilai *pbest* (*personal best*) dan *gbest* (*global best*). Nilai *pbest* ini merupakan posisi partikel yang terbaik selama iterasi yang dilakukan ($F_{k+1}^{i} < F_{k}^{i}$), sedangkan *gbest* adalah nilai posisi partikel yang paling mendekati dengan target ($F_{k+1}^{b_1} < F_{k}^{b}$). Pergerakan partikel dalam kawanan bergantung pada tiga faktor yaitu *pbest*, *gbest* dan kecepatan (*velocity*)[12]. Rumus kecepatan partikel dapat dihitung dengan Persamaan 2.10

$$v_{k+1}^{i} = w \cdot v_{k}^{i} + c_1 \cdot rand(pbest_i - x_{k}^{i}) + c_2 \cdot rand(gbest_i - x_{k}^{i}) \quad (2.10)$$

Dimana:

$pbest_i$ = personal best partikel i.

w = bobot inertia (biasanya diantara 0.9-0.4).

$gbest_i$ = global best partikel i.

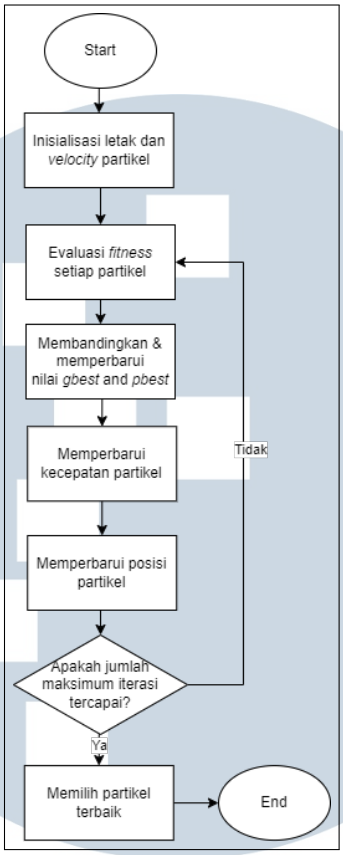
c_1 = learning factor personal (biasanya diantara 0-1)

c_2 = learning factor global (biasanya diantara 0-1)

Nilai parameter c_1 dan c_2 harus diinisialisasi terlebih dahulu, selanjutnya menghitung kecepatan menggunakan rumus 2.7, setelah itu evalusi nilai *fitness* menggunakan Persamaan 2.9, simpan nilai *pbest* dan nilai *gbest*, jika kriteria tidak terpenuhi evaluasi kecepatan partikel menggunakkan Persamaan 2.10 dan memperbarui posisi partikel dengan menggunakan Persamaan 2.11.

$$x_{k+1}^{i} = x_{k}^{i} + v_{k+1}^{i} \quad (2.11)$$

Iterasi dilakukan dengan terus memperbarui nilai *fitness* sehingga nilai *gbest* dan nilai *pbest* memenuhi kriteria yang dibutuhkan. Langkah tersebut dapat dilihat dengan Gambar 2.5.



Gambar 2.6. Flowchart PSO

Sumber: [25]

2.5 Confusion Matrix

Confusion Matrix memberikan informasi komparatif antara hasil klasifikasi yang sebenarnya atau hasil yang sebenarnya dan prediksi. Secara umum, matriks ini memiliki Struktur 2X2 untuk klasifikasi biner.[29] yang digambarkan pada Tabel 2.1

Tabel 2.1. Tabel *Confusion Matrix*

		Predicted	
		Positive	Negative
Actual	Positive	True Positive (TP)	False positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Dimana:

True Positive, jika tes prediksi sama benar dengan aktual.

False positive, jika tes prediksi memiliki nilai salah berbeda dengan nilai benar aktual.

False Negative, jika tes prediksi salah namun aktual benar.

True Negative, jika tes prediksi sama salah dengan aktual.

Matriks ini juga menjadi dasar perhitungan akurasi, presisi, *recall*, dan nilai skor f1 dari tes. Nilai akurasi menunjukkan prediksi yang benar hasil dengan kondisi sebenarnya dengan Persamaan 2.12. Presisi menunjukkan keakuratan hasil tes dengan Persamaan 2.13. *Recall* menunjukkan nilai untuk mengukur proporsi hasil nilai yang benar diidentifikasi, nilainya dapat dihitung dengan Persamaan 2.14. F1-Score adalah hasil dari presisi dan *recall*, dapat dihitung dengan Persamaan 2.15.

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \tag{2.12}$$

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

$$Recall = \frac{(TP)}{(TP+FN)} \tag{2.14}$$

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

2.6 TextBlob

TextBlob adalah paket pemrosesan teks untuk Python 2 dan 3. Ini menyediakan API dasar untuk melakukan tugas pemrosesan bahasa alami (NLP) umum seperti penandaan *part-of-speech*, ekstraksi frasa kata benda, analisis sentimen, klasifikasi, dan terjemahan[30]. Untuk kalimat masukan tertentu, ia juga mengembalikan dua properti. Polaritas Polaritas menentukan polaritas emosi yang direpresentasikan dalam pernyataan yang sedang dipertimbangkan. Kisarannya adalah $[-1,1]$, dimana -1 menunjukkan sentimen negatif, dan 1 menunjukkan sentimen baik. Subjektivitas Subjektivitas digunakan untuk menentukan keadaan pribadi pembicara, seperti emosi, keyakinan, dan pendapat.