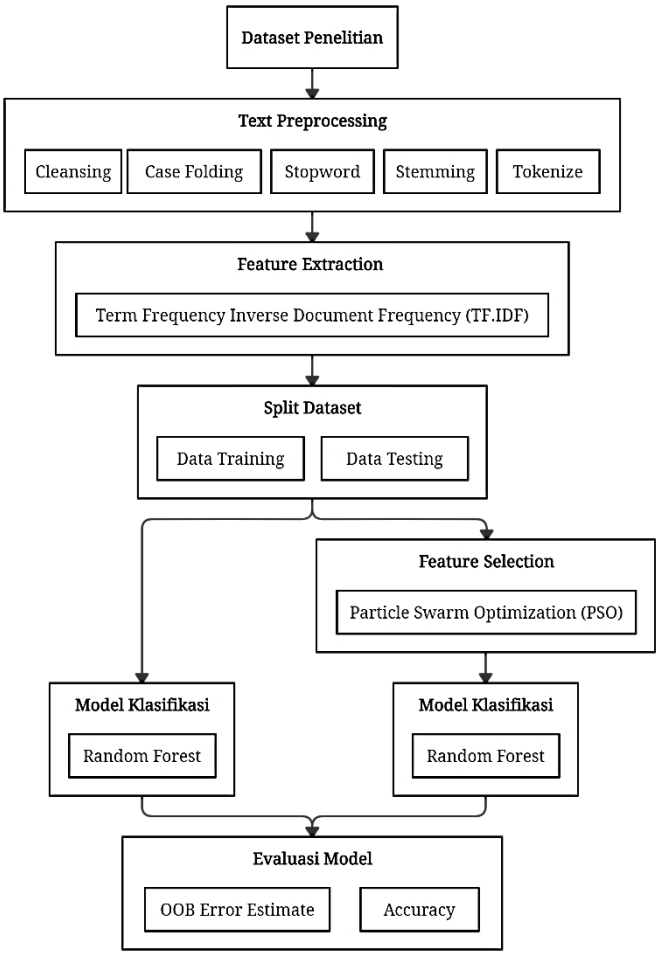
BAB III
METODOLOGI PENELITIAN

Metodologi penelitian menjelaskan bagaimana tahapan-tahapan dalam alur proses sistem Analisis Sentimen Berbasis Feature Selection Particle Swarm Optimizaton Menggunakan Metode Random Forest yang akan dibangun bekerja dimulai dari tahapan pengumpulan data hingga tahapan evaluasi model klasifikasi.



Gambar 3.1 Alur kerja sistem analisis sentimen

3.1 Dataset Penelitian

Alur kerja sistem analisis sentimen dimulai dari tahap pengumpulan data. Pada tahapan ini dilakukan pengumpulan data yang akan digunakan dalam penelitian

diperoleh melalui API Twitter yang menampung data *tweet* penggunanya. Data *tweet* yang diambil merupakan *tweet* yang mengandung kata "bawaslu" dan "#BawasluTegasMinimPelanggar". Setelah data diperoleh akan dilakukan proses pemberian kelas secara manual agar dapat digunakan sebagai dataset dimana kelas negatif direpresentasikan dengan nilai 0 dan sebaliknya kelas positif dengan nilai 1. Setelah data telah dikumpulkan dan diberikan kelas, selanjutnya sistem akan membaca dataset yang diinputkan. Contoh dataset yang telah dilabeli dapat dilihat pada Tabel 3.1.

Tabel 3.1 Contoh dataset yang telah dilabeli

Dokumen	Teks	Kelas
D_1	Bawaslu harus lebih baik kinerjanya.	0
D_2	Aku dukung banget kinerja bawaslu.	1
D_3	Lebih baik bawaslu mengawasi secara langsung.	0
D_4	Dampak kinerja bawaslu langsung kelihatan.	1

3.2 *Text Preprocessing*

Data teks *tweet* pengguna yang berasal dari dataset yang dimasukkan ke dalam sistem akan dilakukan tahap *text preprocessing*. *Text preprocessing* diperlukan pada sistem ini dengan tujuan agar mengurangi kata yang tidak diperlukan, kata yang sering muncul, dan menstandarisasikan kata ke dalam bentuk kata baku. Data teks akan diproses pada bagian awal yaitu *Cleansing* dimana pada bagian ini akan menghapus *noise* yang terdapat pada teks yaitu *emoticon, unicode character*, huruf *non-alphabet, username, hashtag*, alamat url dan menghapus tanda baca. Pada proses ini teks akan melalui bagian *Case Folding* dimana teks dikonversikan menjadi huruf kecil guna untuk menstandarisasikan kata. Normalisasi kata informal dilakukan dikarenakan kebiasaan penggunaan kata tersebut pada komunikasi sehari-hari yang tentunya dapat mempengaruhi kualitas data. Kata-kata informal tersebut dapat disebabkan oleh salah pengetikan, bahasa sehari-hari maupun penyingkatan bentuk kata [25]. Proses *Stopword* dilakukan guna mengurangi kata-kata umum yang sering muncul dan tidak relevan serta menghapus kata yang terdiri

kurang dari 4 huruf seperti kata “di”, “dan”, “aku”, “ini” dan “itu”. Kata-kata yang tersisa akan melalui bagian terakhir *Stemming*, bagian *stemming* ini akan memeriksa kata-kata tersebut dan mengubahnya ke dalam bentuk yang kata dasar. Contoh hasil *text preprocessing* dapat dilihat pada Tabel 3.2.

Tabel 3.2 Contoh hasil *text preprocessing*

Dokumen	Teks	*Text preprocessing*
D_1	Bawaslu harus lebih baik kinerjanya.	{“harus”, “lebih”, “baik”, “kinerja”}
D_2	Aku dukung banget kinerja bawaslu.	{“dukung”, “banget”, “kinerja”}
D_3	Lebih baik bawaslu mengawasi secara langsung.	{“lebih”, “baik”, “awas”, “langsung”}
D_4	Dampak kinerja bawaslu langsung kelihatan.	{“dampak”, “kinerja”, “langsung”, “lihat”}

3.3 *Feature Extraction*

Setelah data teks dilakukan proses *text preprocessing* akan menghasilkan teks yang telah bersih dan memiliki kata-kata baku. *Feature extraction* merupakan proses mendapatkan sekumpulan nilai diskriminatif dan informatif untuk mengubah teks ke nilai numerik (bobot) dan direpresentasikan dalam bentuk vektor agar mengetahui pengaruh kata atau token terhadap dokumen atau kelas [4][6]. Salah satu *feature extraction* yang sering digunakan dalam klasifikasi data teks yaitu *Term Frequency Inverse Document Frequency* (TF.IDF). TF.IDF merupakan metode yang mengintegrasikan model *Term Frequency (TF)* dan model *Inverse Document Frequency (IDF)* yang dapat dilihat pada persamaan (3.1) [24]. *Term Frequency* merupakan model yang menghitung jumlah kemunculan sebuah kata (*term*) dalam satu teks atau dokumen dirumuskan pada persamaan (3.2) dan *Inverse Document Frequency* merupakan model yang menghitung kata (*term*) yang muncul pada total dokumen dirumuskan pada persamaan (3.3).

$$W_{t,d} = W_{tf_{t,d}} \times idf_t \tag{3.1}$$

$$W_{tf_{t,d}} = \begin{cases} 1 + \log_{10} tf_{t,d} & , untuk\ tf_{t,d} > 0 \\ 0 & , untuk\ tf_{t,d} \leq 0 \end{cases} \quad (3.2)$$

$$idf_t = \log_{10} \frac{N}{df_t} \quad (3.3)$$

Keterangan:

$tf_{t,d}$ = Jumlah kemunculan *term* t pada dokumen d

$W_{tf_{t,d}}$ = Bobot *Term Frequency*

N = Jumlah total dokumen d

df_t = Jumlah frekuensi dokumen yang mengandung *term*

idf_t = Bobot *Inverse Document Frequency*

$W_{t,d}$ = Bobot TF-IDF term t pada dokumen d

Berdasarkan hasil *text preprocessing* pada contoh Tabel 3.2 dapat dilakukan pembobotan kata TF-IDF terhadap dokumen D_1 pada kata "kinerja". Kata "kinerja" muncul pada dokumen D_1 sebanyak 1 kali, sehingga $tf_{kinerja,D1}$ = 1. Lalu untuk menghitung bobot TF kata "kinerja" dapat menggunakan rumus persamaan (3.2).

$$W_{tf_{kinerja,D1}} = 1 + \log_{10}(1) = 1$$

Sedangkan bobot IDF dapat dihitung menggunakan rumus persamaan (3.3) dengan total dokumen N = 4 dan frekuensi kemunculan kata "kinerja", $df_{kinerja}$ = 3.

$$idf_{kinerja} = \log_{10}\left(\frac{4}{3}\right) = 0{,}1249$$

Setelah mendapatkan bobot TF dan IDF, bobot kata "kinerja" dapat dihitung menggunakan rumus persamaan (3.1). Pada Tabel 3.3 dapat dilihat perbedaan nilai IDF berbanding terbalik dengan frekuensi kemunculan kata.

$$W_{kinerja,D1} = W_{tf_{kinerja,D1}} \times idf_{kinerja} = 1 \times 0{,}1249 = 0{,}1249$$

Tabel 3.3 Contoh pembobotan kata TF-IDF

Kata (*term*)	TF	IDF	TF.IDF
harus	1	0,6020	0,6020
lebih	1	0,3010	0,3010

Kata (*term*)	TF	IDF	TF.IDF
baik	1	0,3010	0,3010
kinerja	1	0,1249	0,1249

3.4 *Split* Dataset

Setelah didapatkan dataset berupa vektor bobot, dataset akan dilakukan pemisahan terlebih dahulu menjadi data *training* dan data *testing*. Data *training* merupakan data yang akan digunakan mesin selama proses pembelajaran. Sedangkan data *testing* merupakan data yang akan digunakan untuk menguji model yang terbentuk. Pemisahan dataset ini dilakukan dengan ukuran data *training* dan data *testing* dengan masing-masing sebesar 75% dan 25%.

3.5 *Feature Selection*

Feature selection merupakan proses pengoptimalan untuk mengurangi kumpulan fitur asli yang besar agar menonjolkan subset fitur yang relatif kecil dan meningkatkan akurasi klasifikasi dengan cepat dan efektif secara signifikan [2]. Particle Swarm Optimization (PSO) merupakan teknik *feature selection* yang sangat mudah untuk diaplikasikan dengan modifikasi parameter yang sedikit [15]. PSO hanya memilih fitur yang relevan agar akurasi klasifikasi dapat meningkat atau setidaknya memilih fitur yang tidak mengurangi akurasi klasifikasi. PSO akan memilih fitur yang telah diperoleh melalui proses *feature extraction* dimana ukuran fitur merupakan dimensi pencarian *particle*. Posisi *particle* yang berupa vektor biner menggambarkan fitur digunakan jika bernilai 1 dan sebaliknya nilai 0 merupakan fitur yang tidak digunakan [7].

Tabel 3.4 Contoh posisi *particle*

Kata (*term*)	harus	lebih	baik	kinerja
***Particle* 1**	1	0	0	1
***Particle* 2**	0	0	1	1
***Particle* 3**	0	1	0	1
***Particle* 4**	1	0	1	1

3.6 Model Klasifikasi

Sistem analisis sentimen penelitian ini membentuk 2 macam model klasifikasi yang dapat dilihat pada Gambar 3.1, yaitu model pertama menggunakan metode Random Forest tanpa *feature selection* Particle Swarm Optimization dan model kedua mengkombinasikan metode Random Forest dengan Particle Swarm Optimization. Pada model pertama, Random Forest akan dilakukan pembelajaran dengan data *training* yang memiliki fitur asli alias tidak diseleksi. Sedangkan model kedua, Random Forest akan melakukan pembelajaran menggunakan data *training* yang fitur nya dipilih seiring dengan iterasi Particle Swarm Optimization.

3.7 Evaluasi Model

Model klasifikasi Random Forest yang telah terbentuk pada tahapan 3.6 akan dilakukan evaluasi menggunakan OOB *error estimate* dan *Accuracy*. Validasi internal OOB *error* digunakan untuk mendapatkan nilai evaluasi kinerja model klasifikasi yang tidak bias [13][24]. Semakin rendah nilai OOB *error* suatu model maka semakin baik kinerja model tersebut dalam melakukan pengklasifikasian dan berlaku juga sebaliknya. Selain OOB *error*, dilakukan juga evaluasi akurasi model dengan menghitung nilai akurasi prediksi data *testing* berdasarkan *Confusion Matrix*.