BAB 2
LANDASAN TEORI

2.1 Text Mining

Text mining merupakan teknik kecerdasan buatan yang mengubah data tidak terstruktur menjadi data terstruktur menggunakan NLP untuk meningkatkan analisis menggunakan algoritma *machine learning* [17]. NLP (*Natural Language Processing*) adalah subbidang kecerdasan buatan yang berfokus dalam memungkinkan komputer untuk memahami dan memproses bahasa manusia [18]. *Text mining* lebih berfokus pada proses yang menghasilkan informasi berkualitas tinggi dari teks [19]. *Text mining* memiliki potensi yang sangat besar sebagai alat untuk mengambil dan menganalisis data berskala besar dan kompleks [20]. Salah satu teknik terpenting dalam *text mining* adalah analisis sentimen, karena bisa mengekstrak pendapat tersirat dari data tekstual [21].

2.2 Analisis Sentimen

Analisis sentimen adalah proses mengumpulkan dan menganalisis pendapat seseorang tentang suatu topik tertentu [7]. Analisis sentimen dapat menghasilkan *actionable knowledge* dari pendapat publik tentang suatu entitas [9]. *Actionable knowledge* adalah informasi yang telah dievaluasi kebenarannya dan dapat diaplikasikan untuk masalah tertentu [22].

Analisis sentimen memberikan opini atau label pada teks, di mana label tersebut paling sering menunjukkan polaritas [23]. Analisis sentimen membuat kamus sentimen yang terdiri dari kata-kata emosional beserta polaritas yang menunjukkan tingkat kepositifan dan kenegatifan kata-kata yang ada, dan mengukur emosi menggunakan kamus sentimen tersebut [24]. Analisis sentimen menjadi penting karena mengubah data menjadi informasi yang berharga. Hasil analisis yang didapatkan bisa digunakan oleh individu, perusahaan, pemerintah, atau pihak lainnya untuk hal-hal bermanfaat, seperti membuat analisis penjualan dan pemasaran, *review* produk, *feedback*, serta kebijakan dan layanan masyarakat [25].

2.3 Scraping

Scraping adalah proses mengumpulkan data online dari media sosial dan *website* lain dalam bentuk teks yang tidak terstruktur [26]. *Snscrape library* merupakan alat untuk melakukan *scraping*, yang digunakan untuk melakukan pemindaian pada *social networking services*, seperti Twitter [27]. Terdapat beberapa metode yang dapat digunakan untuk mengambil data dari Twitter, namun *snscrape* merupakan metode terbaik karena dapat mengambil data pada tanggal tertentu dan dengan jumlah data yang tidak terbatas [28].

2.4 Remove duplicate

Setelah melakukan *data scraping*, masih terdapat beberapa *tweet* yang sama, yang salah satu penyebabnya adalah karena ada fitur *repost* atau *retweet* [25]. *Tweet* yang sama akan mempengaruhi hasil perhitungan frekuensi kemunculan suatu kata, sehingga perlu dilakukan proses *remove duplicate* untuk mendapatkan hasil perhitungan yang lebih baik [29].

2.5 Foreign Word Translation

Foreign word translation dilakukan untuk mengatasi masalah *multilingual* pada dataset [30]. Ada dua skenario yang digunakan untuk melakukan *translation*, yaitu langsung menerjemahkan data ke Bahasa Indonesia, dan menerjemahkan data ke Bahasa Inggris terlebih dahulu, lalu diterjemahkan kembali ke Bahasa Indonesia [30].

2.6 Text Preprocessing

Karena data yang tidak terstruktur dan memiliki banyak *noise*, diperlukan *text preprocessing* untuk membuat data lebih terstruktur dan menghapus *noise* [31]. *Text preprocessing* dilakukan untuk mengelola teks sebelum membangun model NLP menggunakan *machine learning* [31]. Terdapat berbagai macam metode *text preprocessing*, seperti konversi huruf besar menjadi huruf kecil, koreksi kata-kata umum yang salah eja, penghapusan tag HTML, penghapusan tanda baca, pengurangan label emotikon, pengurangan karakter yang direplikasi, penghapusan stopword, stemming, *lemmatization*, dan terjemahan kata-kata slang ke frasa dengan arti yang sama. [32].

1. *Case folding*.

   Dalam *text preprocessing*, *case folding* bertujuan untuk mengubah semua huruf pada dokumen teks menjadi huruf kecil [33]. Semua huruf diubah menjadi huruf kecil untuk mencegah *case sensitivity* [34].

2. *Data cleaning*.

   Data cleaning mengacu pada langkah-langkah yang diambil untuk membakukan data serta menghapus teks dan karakter yang tidak relevan [35]. Pada tahap ini teks yang mengandung unsur http, link, url, hashtag, mention, tanda baca, dan karakter bukan alfabet akan diganti dengan spasi [36].

3. *Tokenization*

   Tokenization adalah proses pemisahan teks menjadi unit yang lebih kecil yang disebut token [37]. Pada tahap ini dilakukan proses dekomposisi yang bertujuan untuk memisahkan tweet menjadi kata-kata tersendiri [36].

4. *Remove stopwords*

   Stopwords adalah kata yang paling umum dalam suatu bahasa, yang dianggap tidak perlu dan tidak berguna dalam *text mining* [34]. *Remove stopwords* bertujuan untuk menghilangkan kata-kata yang tidak penting [33].

5. *Normalization*

   Pada tahap ini dilakukan normalisasi bahasa terhadap kata tidak baku [38]. *Normalization* bertujuan untuk mengubah kata tidak baku menjadi baku sesuai dengan Kamus Besar Bahasa Indonesia (KBBI) [39].

6. *Stemming*.

   Stemming mengacu pada proses menghilangkan awalan dan akhiran dari kata [40]. Tujuan dari *stemming* adalah untuk mereduksi kata dalam teks menjadi bentuk dasar dengan menghilangkan awalan dan akhiran kata sesuai dengan beberapa aturan gramatikal [34].

2.7 Labeling

Labeling data dilakukan secara otomatis menggunakan kamus *InSet (Indonesia Sentiment Lexicon)*, yang berisi kumpulan kata beserta bobotnya dan berfungsi untuk menentukan nilai polaritas [41]. Nilai polaritas didapatkan melalui penjumlahan setiap bobot kata yang ada dalam teks, kemudian hasil perhitungan

tersebut akan dijadikan label [41]. Jika nilai polaritas lebih besar dari nol, sentimen akan tergolong ke dalam kelas positif, sedangkan jika nilai polaritas sama dengan nol, sentimen akan tergolong ke dalam kelas netral, dan jika nilai polaritas lebih kecil dari nol, maka sentimen akan tergolong ke dalam kelas negatif [42].

2.8 CountVectorizer

CountVectorizer adalah metode dasar pengukuran kata dengan menghitung jumlah kemunculan setiap kata dalam dokumen, sehingga metode ini bisa disebut juga dengan metode *raw count* [43]. *CountVectorizer* dapat mengubah fitur teks menjadi sebuah representasi vektor [44].

2.9 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF adalah metode pembobotan kata atau istilah yang memberikan bobot berbeda pada setiap istilah dalam dokumen berdasarkan *term frequency* per dokumen dan *term frequency* di semua dokumen [45]. *TF-IDF* telah banyak digunakan di bidang pencarian informasi dan *text mining* untuk mengevaluasi hubungan setiap kata dalam kumpulan dokumen [46]. Rumus perhitungan *TF-IDF* adalah sebagai berikut.

$$TF_{t,d} = \frac{n_{t,d}}{\sum_k n_{t,d}}, \tag{2.1}$$

di mana $TF_{t,d}$ merupakan *term frequency*, $n_{t,d}$ menunjukkan jumlah kemunculan kata, dan $\sum_k n_{t,d}$ menunjukkan jumlah kemunculan kata dalam dokumen.

$$IDF_t = log\frac{N}{df_t}, \tag{2.2}$$

di mana IDF_t merupakan *inverse document frequency*, N menunjukkan jumlah dokumen, dan df_t menunjukkan jumlah dokumen yang mengandung term t.

$$TF-IDF_{t,d} = TF_{t,d} * IDF_t, \tag{2.3}$$

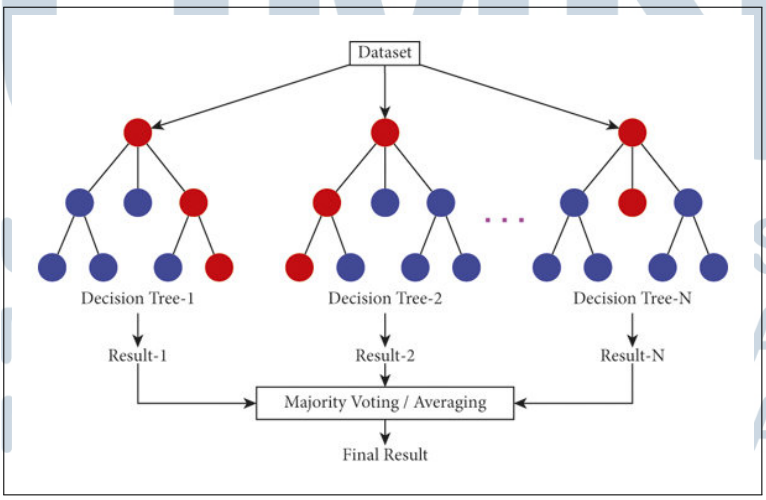
di mana $TF-IDF_{t,d}$ merupakan *term frequency-inverse document frequency*, $TF_{t,d}$ menunjukkan *term frequency*, dan IDF_t menunjukkan *inverse document frequency*.

2.10 Synthetic Minority Oversampling Technique (SMOTE)

Synthetic Minority Oversampling Technique (SMOTE) adalah teknik menyeimbangkan dataset dengan membuat data sintetik [47]. *SMOTE* merupakan teknik oversampling di mana titik baru disintesis di antara titik-titik yang sudah ada [48].

2.11 Random Forest Classifier

Random Forest Classifier merupakan algoritma *ensemble* yang menerapkan teknik *bagging* untuk membangun beberapa *decision tree* menggunakan *bootstrapped samples* [49]. *Ensemble learning* merupakan model *machine learning* yang melatih banyak *learner* untuk memecahkan masalah yang sama [50]. *Bagging* merupakan teknik yang membangun beberapa model homogen dari subsampel yang berbeda untuk mendapatkan prediksi yang lebih akurat daripada model individualnya [51]. Kelebihan dari *Random Forest Classifier* adalah bisa mengidentifikasi data penting dengan cepat dari *dataset* besar, menangani *missing values*, dan mengurangi *overfitting* [13]. *Random Forest* mengurangi tingkat *overfitting* dengan menggabungkan beberapa evaluator overfit (*decision tree*) untuk membentuk algoritma *ensemble learning* [14]. Dengan menggunakan hasil *voting* dari setiap *decision tree*, kategori dari sampel yang akan diuji ditentukan berdasarkan mayoritas, dan hasil akhirnya ditentukan oleh kategori dengan suara tertinggi [14].



Gambar 2.1. *Random Forest*

Sumber: [52]

2.12 Hyperparameter Tuning

Hyperparameter tuning merupakan proses menemukan nilai *hyperparameter* terbaik dari suatu algoritma pembelajaran untuk menghasilkan model terbaik [53]. Dengan melakukan *hyperparameter tuning*, *performance* dari *decision tree* dan *random forest* dapat meningkat dengan signifikan [54].

2.13 Confusion Matrix

Confusion matrix adalah alat yang bisa menunjukkan kinerja dari *classifier* dengan mudah dan efektif serta memiliki keuntungan untuk menginterpretasikan hasilnya dengan mudah [55]. *Confusion matrix* menunjukkan distribusi prediksi pada semua kelas dalam satu tampilan ringkas [56]. Gambar 2.2 merupakan *confusion matrix* yang digunakan pada *multiclass classification*.

		PREDICTED classification				
	Classes	a	b	c	d	Total
ACTUAL classification	a	6	0	1	2	9
	b	3	9	1	1	14
	c	1	0	10	2	13
	d	1	2	1	12	16
	Total	11	11	13	17	**52**

Gambar 2.2. *Multiclass Confusion Matrix*

Sumber: [57]

Beberapa *performance measures* seperti akurasi, presisi, *recall*, dan *f1-score* dapat ditentukan dari informasi yang terkandung dalam *confusion matrix* [58].

$$Accuracy = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} \sum_{j=1}^{n} N_{ij}} \tag{2.4}$$

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{2.5}$$

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \tag{2.6}$$

$$f1 - score_i = 2 * \frac{Recall_i * Precision_i}{Recall_i + Precision_i} \tag{2.7}$$

Keterangan:

N_{ij} = Jumlah sampel yang termasuk dalam kelas C_i tetapi diklasifikasikan sebagai kelas C_j

TP_i = *True Positive* pada kelas i

FP_i = *False Positive* pada kelas i

FN_i = *False Negative* pada kelas i

Pada kasus *multiclass*, *f1-score* harus melibatkan semua kelas. Untuk mendapatkan *macro f1-score*, diperlukan *macro recall* dan *macro precision* terlebih dahulu, sedangkan untuk mendapatkan *micro f1-score*, diperlukan *micro recall* dan *micro precision* [57].

$$Recall(macro) = \frac{1}{n} \sum_{i=1}^{n} Recall_i \tag{2.8}$$

$$Precision(macro) = \frac{1}{n} \sum_{i=1}^{n} Precision_i \tag{2.9}$$

$$f1 - score(macro) = 2 * \frac{Recall(macro) * Precision(macro)}{Recall(macro) + Precision(macro)} \tag{2.10}$$

$$Recall(micro) = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} [TP_i + FN_i]} \tag{2.11}$$

$$Precision(micro) = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} [TP_i + FP_i]} \tag{2.12}$$

$$f1 - score(micro) = 2 * \frac{Recall(micro) * Precision(micro)}{Recall(micro) + Precision(micro)} \tag{2.13}$$