BAB 2

LANDASAN TEORI

2.1 Text Classification dan Text Pre-processing

Klasifikasi teks atau kategorisasi teks merupakan proses yang secara otomatis menempatkan dokumen teks ke dalam suatu kategori berdasarkan konten yang dimilikinya (Zhang, Huang, & K.L., 2009). Ada beberapa tahapan umum yang dapat dilakukan untuk melakukan proses klasifikasi teks secara otomatis, yaitu *text preprocessing*, *feature extraction* atau *selection, modeling* dengan menggunakan teknik pembelajaran mesin yang sesuai, serta training dan testing pada *classifier* (Dalal & Zaveri, 2011).

Dalam proses *pre-processing* ada berberapa tahap yang dilakukan (Sutami, 2015), di antaranya:

1. Case Folding: Mengubah seluruh huruf dalam teks ke dalam format huruf kecil. Hal ini dilakukan untuk menghilangkan variasi dalam data teks untuk mempermudah proses lanjutan yang akan dilakukan.
2. Convert Number: Mengubah angka menjadi kata atau menghapus angka yang tidak dibutuhkan.
3. Remove Punctuation: Menghapus tanda baca dan spasi jika dianggap tidak relevan terhadap proses lanjutan yang akan dilakukan.
4. Tokenization: Proses pemotongan suatu dokumen menjadi bagian-bagian paling kecil yang menyusunnya. Bagian-bagian kecil ini disebut dengan istilah token.

5. Stopword Removal: proses penghilangan kata yang termasuk ke dalam *list stopwords* dalam suatu bahasa tertentu. Istilah *stopwords* merujuk pada kata-kata yang sering muncul dan tidak mencirikan identitas dari sebuah teks (Dalal & Zaveri, 2011), sebagai contoh kata "adalah", "yaitu", "yang", "dan", serta "atau" pada Bahasa Indonesia.
6. Stemming: Proses pengembalian sebuah kata ke kata dasar. Proses ini dilakukan guna menghilangkan variasi yang dianggap tidak relevan pada proses lanjutan. Proses stemming terbukti dapat meningkatkan performa model pembelajaran mesin pada ranah *text retrieval* dan *text classification* (Sutami, 2015).

2.2 Decision Tree Learning

Istilah *decision tree learning* atau proses pembuatan pohon keputusan, merujuk pada istilah proses pengolahan kumpulan data menjadi sekumpulan aturan-aturan keputusan yang dapat digunakan untuk mengategorikan, ataupun mengelompokkan data. Manfaat utama dari penggunaan algoritma *decision tree* adalah kemampuannya untuk memecah-mecah sekumpulan data berdasarkan ciri yang dimilikinya secara sederhana. Di sisi lain, model *decision tree* dapat dianggap lebih mudah untuk diinterpretasikan oleh manusia dibandingkan dengan model-model lainnya yang lebih kompleks. Menurut (Arviana, 2020), terdapat tiga elemen yang menyusun sebuah *decision tree,* yaitu:

1. *root node* (akar): *top-level node* yang merepresentasikan tujuan akhir dari sebuah *decision tree*.

2. *branches* (ranting): Percabangan yang menggambarkan pilihan-pilihan yang dapat diambil untuk mencapai sebuah keputusan.
3. *leaf node* (daun): Hasil keputusan yang akan diambil.

Untuk membangun sebuah *decision tree*, dilakukan proses *divide and conquer* untuk membagi kelompok data ke kelompok lainnya yang lebih kecil secara *recursive,* dengan menggunakan rumus Entropi dan Gain yang dijelaskan pada bagian di bawah ini (Andriani, 2013),

1. Entropi

$$Entropi\ (S) = \sum_{j=1}^{k} -\ p_j\ log_2\ p_j \tag{2.1}$$

di mana keterangan variabel yang muncul pada rumus di atas dapat dilihat pada bagian berikut,

a. S = Sekumpulan data disertai informasi kelompok data yang ingin dihitung nilai entropinya.
b. K = Informasi kelompok unik yang membagi data.
c. p_j = Nilai probabilitas kemunculan data dengan kelas j dalam S.

2. Gain

$$Gain\ (A) = Entropi\ (S) - \sum_{j=1}^{k} \frac{|S_j|}{|S|} \times Entropi\ (S_j) \tag{2.2}$$

Di mana keterangan variabel yang muncul pada rumus di atas dapat dilihat pada bagian berikut,

a. $Entropi(S)$ = Entropi dari sekumpulan data S.
b. K = Indeks subkelompok yang dibentuk berdasarkan kumpulan data S.

c. S_j = Subkelompok yang dibentuk berdasarkan kumpulan data S.

d. A = Atribut yang dipilih untuk memecah sekumpulan data S ke dalam subkelompok $S_1, \dots, S_k$.

e. $Entropi\ (S_j)$ = Entropi dari subkelompok S_j.

2.3 Term Frequency Inverse Document Frequency

Metode TF-IDF merupakan metode untuk menghitung bobot setiap kata yang paling umum digunakan pada *information retrieval*. Metode ini juga terkenal dikarenakan kesederhanaan implementasi miliknya untuk merepresentasikan sekumpulan kata yang menyusun sebuah teks secara akurat berdasarkan sekumpulan dokumen. Metode ini akan menghitung nilai *Term Frequency* (TF) untuk setiap kata dalam setiap dokumen dan nilai *Inverse Document Frequency* (IDF) berdasarkan informasi kemunculan kata berdasarkan seluruh dokumen masukan (Hidayat, 2016). Rumus TF-IDF yang digunakan untuk menghitung bobot dari setiap *term* dalam sebuah dokumen dapat dilihat seperti pada bagian berikut, (Amelia Rahman, 2017),

$$W_{d,t} = tf_{d,t} \times idf_t = tf_{dt} \times \log\left(\frac{N}{df_t}\right) \quad (2.3)$$

dengan keterangan variabel yang muncul pada rumus dijelaskan pada bagian berikut,

a. $W_{d,t}$ = Bobot *term t* terhadap dokumen *d*

b. $tf_{d,t}$ = Jumlah kemunculan *term t* dalam dokumen *d*

c. N = Jumlah dokumen secara keseluruhan

d. df_t = Jumlah dokumen yang mengandung *term t*

2.4 Random Forest Classifier

Metode Random Forest (RF) adalah pengembangan dari metode *decision tree*, yaitu dengan menerapkan metode *bootstrap aggregating* (*bagging*) dan *random feature selection*. Metode RF (Adnyana, 2015), terdapat sekumpulan *decision tree* yang dilibatkan dalam mengambil sebuah keputusan. Sekumpulan *decision tree* ini dirujuk dengan istilah *forest*. Sebagai metode klasifikasi, metode RF merupakan salah satu bentuk metode *ensemble* yaitu (*bagging* dan *random feature selection)*. Metode *ensemble* bekerja dengan cara mengombinasikan sekumpulan metode klasifikasi guna memperoleh performa klasifikasi yang lebih baik (Han, 2012). Proses pelatihan metode RF) dilakukan dengan melakukan proses *random sampling* terhadap sekumpulan atribut dan individu dalam data latih untuk membangun sekumpulan *decision tree* (Adnyana, 2015). Saat digunakan untuk mengklasifikasikan data masukan, setiap pohon yang menyusun sebuah model hasil latih akan memberikan keputusan terhadap data tersebut dan akan dilakukan proses *majority voting* untuk menentukan keputusan akhir (Han, 2012).

Dibandingkan dengan model yang terbentuk dari *single decision tree* metode RF memiliki beberapa keunggulan, di antaranya, metode RF tidak seperti metode *decision tree* yang rentan terhadap *overfitting*; dan metode RF dapat menghasilkan performa yang lebih baik saat jumlah dataset yang dipelajari berukuran lebih besar. (Islami, 2019).

2.5 F1-Score

Pengukuran kinerja model dalam penelitian akan menggunakan metrik F1-Score. Metode *F1-Score* dihitung berdasarkan nilai Precision dan Recall milik model terkait. Untuk menghitung nilai Precision dan Recall, dibutuhkan nilai Confusion Matrix berdasarkan hasil prediksi dan label aktual milik data. Definisi dari Confusion Matrix, diberikan pada bagian di bawah ini,

Tabel 2. 1 Tabel *Confusion Matrix*

	Positif (actual)	**Negatif (actual)**
Positif (sistem)	TP	FP
Negatif (sistem)	FN	TN

Penjelasan dari definisi Confusion matrix diberikan pada bagian berikut,

1. TP (True Positive) mencerminkan jumlah hasil prediksi positif dari sistem yang sesuai dengan label aktual
2. TN (True Negative) mencerminkan jumlah hasil prediksi negative dari sistem yang sesuai dengan label aktual
3. FP (False Positive) mencerminkan jumlah hasil prediksi kelas positif dengan label aktual negatif.
4. FN (False Negative) mencerminkan jumlah hasil prediksi kelas negatif dengan label aktual negatif.

Kemudian, berdasarkan definisi Confusion Matrix, dapat dihitung nilai Precision dan Recall dengan menggunakan kedua rumus berikut,

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

dan,

$$Recall = \frac{TP}{TP + FN} \tag{2.5}$$

Berdasarkan nilai *precision* dan *recall* milik sebuah model klasifikasi, dapat dihitung nilai F1-Score yang mencerminkan geometric mean dari kedua nilai dengan menggunakan rumusan berikut (Rasyid, Bijaksana,, & &, 2019),

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.4}$$

2.6 Recursive Feature Elimination

Metode *Recursive Feature Elimination* (RFE) pada dasarnya bekerja dengan cara melakukan proses rekursif berdasarkan hasil pemeringkatan fitur berdasarkan nilai kepentingannya terhadap sebuah proses prediksi. Pada setiap iterasinya, setiap fitur akan diukur tingkat kepentingannya; kemudian sekumpulan fitur yang dianggap kurang relevan akan dihilangkan; dan berdasarkan kumpulan fitur yang tersisa akan dilatih model klasifikasi yang baru (Wibawa & Novianti, 2017). Metode RFE merupakan salah satu metode eliminasi fitur secara otomatis. Metode RFE menjadi relevan untuk digunakan ketika jumlah fitur yang digunakan untuk melatih sebuah model memiliki jumlah yang besar sehingga proses eliminasi fitur secara manual sulit untuk dilakukan.