BAB 2

LANDASAN TEORI

2.1 Text Preprocessing

Text preprocessing merupakan suatu proses pengubahan bentuk sebuah data menjadi lebih terstruktur sesuai dengan kebutuhannya dalam proses *data mining* dan biasanya akan menjadi nilai numerik (Ashari, et al., 2020). Proses ini merupakan tahap awal dari mendeteksi berita *hoax* sebelum digunakan oleh model. Proses ini akan dibagi menjadi beberapa tahap agar *dataset* yang telah dikumpulkan dapat digunakan pada algoritma yang telah dipilih. Tahapan dari *text preprocessing* adalah:

1. *Case Folding*

   Case folding merupakan proses untuk mengubah semua teks di dalam *dataset* menjadi huruf kecil.

2. *Tokenizing*

   Tokenizing adalah tahap untuk memisahkan atau memecah teks menjadi bagian–bagian kata yang disebut token.

3. *Stopwords Removing*

   Stopwords removing adalah tahap untuk mengambil kata yang dianggap penting dari hasil *tokenizing* atau membuang kata yang dianggap tidak memiliki arti penting dalam proses *text preprocessing*.

2.2 TF-IDF

TF-IDF atau *Term Frequency-Inverse Document Frequency* adalah algoritma yang berfungsi dalam menghitung nilai bobot pada setiap kata yang ada pada dokumen (Ashari, et al., 2020). *Term Frequency* adalah jumlah frekuensi kemunculan sebuah kata dalam suatu dokumen. *Inverse Document Frequency* adalah perhitungan di mana suatu kata tersebut tersebar dalam suatu dokumen. Frekuensi dokumen menunjukan seberapa umum kata tersebut muncul pada sebuah dokumen, sehingga nilai bobot antara suatu kata dan dokumen menjadi tinggi bila frekuensi kata tersebut tinggi pada suatu dokumen (Nurjannah, et al., 2013).

TF-IDF akan berfungsi sebagai metode ekstrasi fitur dari dokumen yang sudah diproses melalui *text preprocessing*. Proses ini bertujuan untuk mengevaluasi seberapa relevan kata pada suatu dokumen dalam kumpulan dokumen. Proses ini digunakan untuk menghitung nilai bobot setiap kata pada dokumen. Bobot tersebut merupakan sebuah nilai pentingnya sebuah kata yang ada pada dokumen dan semakin besar nilai bobotnya maka peran kata tersebut di dalam dokumen menjadi sangat penting.

Proses dimulai dengan menghitung nilai *Term Frequency* dari sebuah kata yang dimiliki oleh suatu dokumen, dengan menghitung jumlah munculnya kata tersebut pada dokumen tersebut. Berikutnya mencari nilai *Inverse Document Frequency* dengan cara menghitung jumlah dokumen yang mengandung kata tersebut. Semakin banyak kata tersebut tersebar pada dokumen lain, semakin kurang pentingnya kata itu pada dokumen tersebut.Berikut rumus dari *TF-IDF*:

$$W_{dt} = tf_d \times idf_t \tag{2.1}$$

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

Keterangan:

W_{dt} = Nilai bobot kata ke-t pada dokumen d

tf_d = Jumlah munculnya kata t pada dokumen d

N = Jumlah dokumen secara keseluruhan

df_t = Jumlah dokumen yang mengandung kata t

d = Dokumen ke-d

t = Kata ke-t

2.3 Multinomial Naïve Bayes

Multinomial Naïve Bayes adalah model yang dikembangkan dari algoritma Bayes yang cocok dalam hal pengklasifikasian teks atau dokumen (Ashari, et al., 2020). Model ini memperhitungkan frekuensi dari setiap kata yang muncul pada suatu dokumen (Rahman, et al., 2017). Untuk memperhitungkan kelas dari suatu dokumen, maka dapat dihitung dengan rumus:

$$C_{map} = \arg max_{c \in C}\ P(c) \prod_{n=1}^{k} P(t_n|c) \tag{2.3}$$

Keterangan:

C_{map} = Maximum a posteriori. Probabilitas suatu dokumen termasuk kelas *c*

$P(c)$ = Probabilitas *prior* dari kelas *c*

$P(t_n|c)$ = Probabilitas kata ke-n dengan diketahui kelas *c*

Probabilitas *prior* kelas *c* ditentukan dengan rumus:

$$P(c) = \frac{N_c}{N} \tag{2.4}$$

Keterangan:

N_c = Jumlah dokumen di kelas *c*

N = Jumlah seluruh dokumen

Probabilitas kata ke-n ditentukan dengan menggunakan teknik *laplace smoothing*:

$$P(t_n|c) = \frac{count(t_n, c) + 1}{count(c) + |V|} \tag{2.5}$$

Keterangan:

$count(t_n, c)$ = Jumlah kata t_n yang ditemukan di seluruh data pelatihan dengan dari kelas *c*

$count(c)$ = Jumlah kata di seluruh data pelatihan dengan kategori *c*

V = Jumlah seluruh kata pada data pelatihan

Rumus *Multinomial* yang digunakan dengan pembobotan kata *TF-IDF* adalah sebagai berikut:

$$P(t_n|c) = \frac{W_{ct} + 1}{(\sum W' \text{€} VW') + B'} \tag{2.6}$$

Keterangan:

W_{ct} = Nilai pembobotan *TF-IDF* atau W dari kata t di kelas *c*

$\sum W' \text{€} VW'$ = Jumlah total W dari keseluruhan kata yang berada di kelas *c*

B' = Jumlah W dari kata unik

2.4 Logistic Regression

Logistic Regression adalah algoritma klasifikasi yang digunakan untuk memprediksi probabilitas dari sebuah variabel kategori yang dependen (Nada, et al., 2019). *Logistic Regression* berfungsi dalam menggambarkan data dan menjelaskan hubungan antara satu variabel biner yang dependen dan variabel independen lainnya. Di mana variabel dependen merupakan data yang memiliki bentuk kategori seperti nilai 1(iya, benar) atau 0 (tidak, salah).

Logistic Regression dapat digunakan untuk memperkirakan *odds ratio* atau yang biasa disebut sebagai probabilitas untuk setiap variabel independennya. Hal ini berguna untuk mengetahui perubahan nilai dari variabel dependen. Hal ini dapat dilihat bila terdapat variabel independen tertentu yang memiliki pengaruh terhadap variabel dependen.

Logistic Regression menggunakan *S-shape curve* yang disebut sebagai *sigmoid*. *Sigmoid* memiliki properti untuk memetakan sebuah nilai dari 0 ke 1, atau bisa disebut sebagai probabilitas. Salah satu fungsi dari *sigmoid* adalah *logistic sigmoid function* yang digunakan untuk *logistic function*. Function ini akan digunakan untuk menghasilkan probabilitas dari 0 ke 1, dengan menghitung *input* dari sebuah fitur. Berikut rumus dari *Logistic Sigmoid Function*:

$$p(x) = \frac{1}{1 + e^{-t}} \tag{2.7}$$

$$t = \theta \times x + b \tag{2.8}$$

Keterangan:

t = Bobot dari sebuah *input* fitur

θ = Parameter model

e = Eksponen

x = Nilai dari fitur

b = Koefisien

2.5 Recursive Feature Elimination

Recursive Feature Elimination adalah metode pemilihan fitur yang mencoba untuk memilih fitur optimal berdasarkan model dan akurasi klasifikasi (Jeon & Oh, 2020). Metode ini akan menyesuaikan dengan model dan menghapus fitur yang lemah sehingga jumlah fitur yang ditentukan tercapai. Tidak semua fitur memiliki kontribusi terhadap variabel prediksi atau model. Sehingga menghapus fitur dapat meningkatkan performa dan mengatasi *overfitting* model (Chen & Jeong, 2008).

Cara kerja *Recursive Feature Elimination* adalah melakukan proses rekursif dan mengurutkan fitur berdasarkan tingkat pengaruh atau pentingnya fitur tersebut dalam proses prediksi. Pertama menggunakan *estimator* atau model untuk dilatih dengan jumlah fitur pertama atau awal. Lalu setelah itu fitur akan dinilai bobotnya dalam mempengaruhi model. Lalu fitur tersebut akan diberikan peringkat dan fitur yang memiliki peringkat yang paling rendah akan dihilangkan dari kumpulan data tersebut. Setelah itu model akan dilatih kembali menggunakan jumlah fitur yang baru. Proses ini dilakukan berulang kali sehingga jumlah fitur yang telah ditentukan tercapai.

Untuk menentukan peringkat setiap fitur di dalam data, diperlukan rumus untuk menghitung bobot *w*. Berikut rumus dalam menghitung bobot fitur:

$$w = \sum_{i=1}^{k} \alpha_i y_i x_i \tag{2.9}$$

α = Hasil klasifikasi model

y = Kelas label

x = Data

w = Bobot

2.6 F1-Score

Confusion Matrix adalah informasi yang berisi tentang performa sistem klasifikasi yang dievaluasi menggunakan data atau metrik dengan menganalisis seberapa baik klasifikasi telah dilakukan pada kelas sebenarnya dan kelas yang diprediksi (Bachtiar, et al., 2020). *F1-Score* merupakan salah satu pengukuran performa dari *confusion matrix* yang berfungsi sebagai pengukur *score* model pada *dataset* yang digunakan untuk mengevaluasi sistem klasifikasi seperti positif atau negatif. *F1-Score* merupakan kombinasi dari model *precision* dan *recall*. Berikut rumus dari *F1-Score*:

$$Precision = \frac{TP}{TP + FP} \tag{2.10}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.11}$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.12}$$

Keterangan:

TP = *True positive*. Jumlah hasil prediksi model yang positif dan benar

FP = *False Positive.* Jumlah hasil prediksi model yang positif namun salah

FN = *False Negative*. Jumlah hasil prediksi model yang negatif namun salah

Precision = Persentase atau jumlah *true positive* yang dibuat dari semua yang diklasifikasikan model sebagai positif

Recall = Persentase atau jumlah *true positive* dari semua target positif

2.7 Hyperparameter Space

Istilah *hyperparameter* digunakan dalam konteks untuk model statistik yang merujuk pada *parameter* yang mengatur bentuk dari distribusi data yang telah diasumsikan sebelumnya (Province, 2015). Berbeda dengan parameter *default* yang biasanya digunakan oleh suatu model, *hyperparameter* menggunakan sejumlah parameter yang memiliki kemungkinan untuk menghasilkan model yang optimal dan sesuai.

Salah satu contoh dari *hyperparameter* adalah *GridSearchCV*. *GridSearchCV* merupakan algoritma yang secara langsung mengarah pada prediksi yang paling akurat dengan menemukan kombinasi yang paling optimal. Proses *GridSearchCV* dimulai dengan menentukan seluruh parameter dari setiap algoritma yang telah ditentukan. Lalu kombinasi dari semua *parameter* yang terdaftar yang disebut sebagai *hyperparameter*, akan diuji kan satu persatu. Lalu dengan semua kombinasi yang telah diuji cobakan maka akan menghasilkan hasil dan parameter yang

berbeda–beda karena hasil dari satu percobaan tidak bergantung pada percobaan lain (Yu & Zhu, 2020). Lalu dari salah satu uji coba tersebut akan menghasilkan hasil dan parameter yang paling optimal.

GridSearchCV memiliki beberapa parameter penting yang digunakan dalam prosesnya. Pertama *estimator* yang digunakan sebagai daftar model atau algoritma yang digunakan untuk *hyperparameter*. Kedua adalah *param_grid* sebagai daftar semua *parameter* dari setiap algoritma yang telah ditentukan di dalam *estimator*. Ketiga adalah *cv* atau *cross-validation* yang digunakan untuk menentukan jumlah *fold* dari setiap kombinasi *parameter*. Dan keempat adalah *scoring* yang digunakan sebagai evaluasi model.