BAB 2
LANDASAN TEORI

2.1 Tinjauan Teori

Berikut merupakan penjabaran dari teori-teori yang mendasari penelitian ini.

2.1.1 Kesalahan Tik

Kesalahan tik merupakan kesalahan penulisan ejaan yang sering terjadi ketika seseorang sedang mengetik sesuatu [14]. Terjadinya kesalahan tik dapat menyebabkan sebuah informasi tidak tersampaikan dengan baik kepada pembaca dan juga dapat menyebabkan kesalahpahaman akan informasi yang diberikan kepada pembaca.

2.1.2 Portal Berita Daring

Portal Berita Daring merupakan sebuah situs atau halaman web yang berisi berbagai jenis berita, seperti politik, ekonomi, sosial, budaya hingga hiburan yang bersifat *hard news* maupun *soft news* [15].

2.1.3 Tribunnews

Tribunnews.com merupakan portal berita daring nomor satu di Indonesia yang dikelola oleh PT Tribun Digital Online. Tribunnews.com memiliki jaringan yang telah tersebar ke seluruh penjuru Indonesia yang bernama Tribun Network. Adapun *tagline* yang memperkuat Tribunnews.com yaitu ”Mata Lokal Menjangkau Indonesia” yang dimana Tribunnews.com membawa misi *Hyperlocal* yang dimana hal ini berakar dari keyakinan bahwa setiap dari kita adalah warga lokal yang bertugas untuk melestarikan nilai dan perspektif dari setiap daerah ke seluruh Indonesia [9].

2.1.4 Natural Language Processing

Natural Language Processing (NLP) merupakan salah satu cabang ilmu *Artificial Intelligence* yang berfokus pada pengolahan bahasa yang biasa digunakan oleh manusia untuk berkomunikasi satu sama lain [16]. Tujuan dari adanya

NLP adalah untuk memungkinkan sebuah mesin untuk dapat memahami bahasa manusia dan memberikan respon yang sesuai. Teknologi NLP sendiri telah berada sangat dekat dengan kehidupan kita sehari-hari, seperti fitur *autocorrect*, mesin penerjemah, *chatbot* hingga *voice assistant* [17].

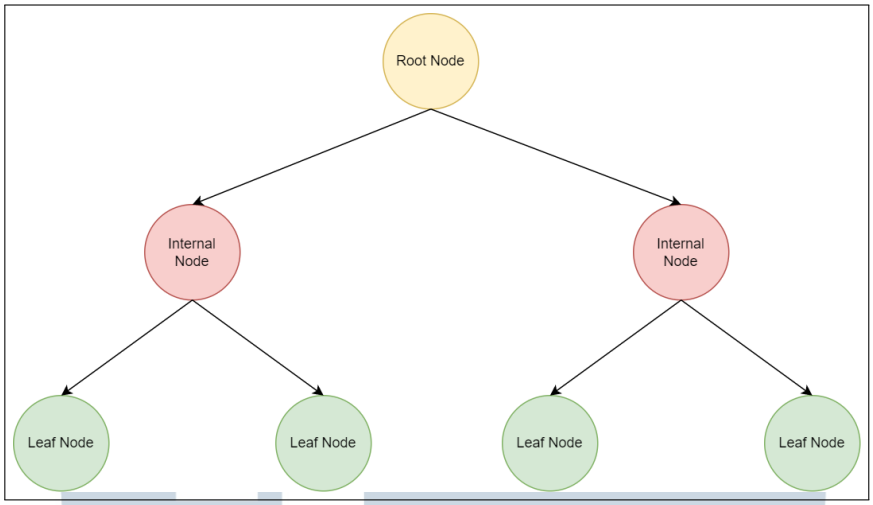
2.1.5 Text Preprocessing

Text Preprocessing merupakan suatu proses yang bertujuan untuk melakukan seleksi pada data teks sehingga nantinya akan lebih terstruktur [18]. Pada *text preprocessing*, terdapat beberapa tahapan yang perlu dilakukan, antara lain sebagai berikut.

1. *Case Folding*, merupakan sebuah proses yang bertujuan untuk mengubah seluruh karakter menjadi huruf kecil dan juga menghilangkan karakter yang tidak termasuk huruf [19].
2. *Tokenizing*, merupakan sebuah proses untuk melakukan pemecahan kalimat menjadi kata [20].
3. *Filtering*, merupakan tahap lanjutan dari *tokenizing* yang dimana tahapan ini digunakan untuk memilih kata-kata yang penting dari hasil *tokenizing* yang telah dilakukan sebelumnya dengan menghapus kata-kata yang tidak memiliki arti atau dapat disebut juga dengan *stopword* [18].
4. *Stemming*, merupakan suatu proses untuk merubah kata-kata yang memiliki imbuhan menjadi kata akar atau kata dasarnya [21].

2.1.6 Decision Tree

Decision tree atau pohon keputusan merupakan salah satu metode untuk melakukan klasifikasi dengan menggunakan representasi dari struktur pohon (*tree*) yang dimana setiap *node* merepresentasikan atributnya, lalu cabangnya merepresentasikan nilai dari atribut dan daunnya merepresentasikan kelasnya [22].



Gambar 2.1. Diagram *decision tree*

Sumber: [22]

Gambar 2.1 merupakan diagram dari *decision tree*. Seperti yang dapat dilihat pada diagram 2.1, *node* pada *decision tree* terbagi menjadi 3 jenis, antara lain sebagai berikut [22].

1. *Root Node* merupakan *node* yang terletak di paling atas yang dimana pada *node* ini tidak memiliki *input* dan memungkinkan untuk tidak memiliki *output* namun memungkinkan juga untuk memiliki *output* lebih dari satu.

2. *Internal Node* merupakan *node* percabangan. *node* ini memiliki satu *input* dan memiliki *output*. Pada *internal node*, dilakukan perhitungan *entropy*. *Entropy* berfungsi untuk mengukur tingkat ketidakpastian atau ketidakmurnian dari atribut [23]. Setelah menentukan *entropy* maka dilakukan perhitungan *gain* dari masing-masing atribut. *Gain* merupakan suatu nilai yang digunakan untuk memilih atribut untuk menghasilkan *node* baru. Berikut merupakan rumus dari perhitungan *entropy* dan *gain* [23].

$$Entropy(S) = \sum_{i=1}^{n} -pi * log_2 * pi \quad (2.1)$$

Keterangan:

S = himpunan kasus

n = jumlah partisi S

Pi = proporsi Si terhadap S

$$Gain(S,A) = Entropy(S) - \sum_{i=1}^{n} \frac{|Si|}{|S|} * Entropy(Si) \quad (2.2)$$

S: himpunan kasus

A: atribut

n : jumlah partisi atribut a

Si : jumlah kasus pada partisi ke-I

S : jumlah kasus dlm S

3. *Leaf Node* atau dapat disebut juga dengan *Terminal Node* merupakan *node* akhir yang dimana pada *node* ini hanya memiliki satu *input* dan memiliki *output* berupa keputusan atau prediksi akhir.

2.1.7 Ensemble Learning

Ensemble learning merupakan sebuah pembelajaran mesin yang dimana suatu data dipelajari dengan menggunakan kombinasi dari beberapa model untuk mendapatkan hasil yang lebih akurat [24]. Pada *ensemble learning* terdapat 3 model, antara lain sebagai berikut [25].

1. *Bagging* merupakan salah satu metode dari *ensemble learning* yang dimana menggunakan satu tipe *base model* dengan melakukan pelatihan secara paralel dan independen pada setiap *base model*, kemudian barulah digabungkan untuk mendapatkan hasil yang terbaik. Algoritma *Random Forest* merupakan salah satu algoritma yang termasuk kedalam model *bagging*.

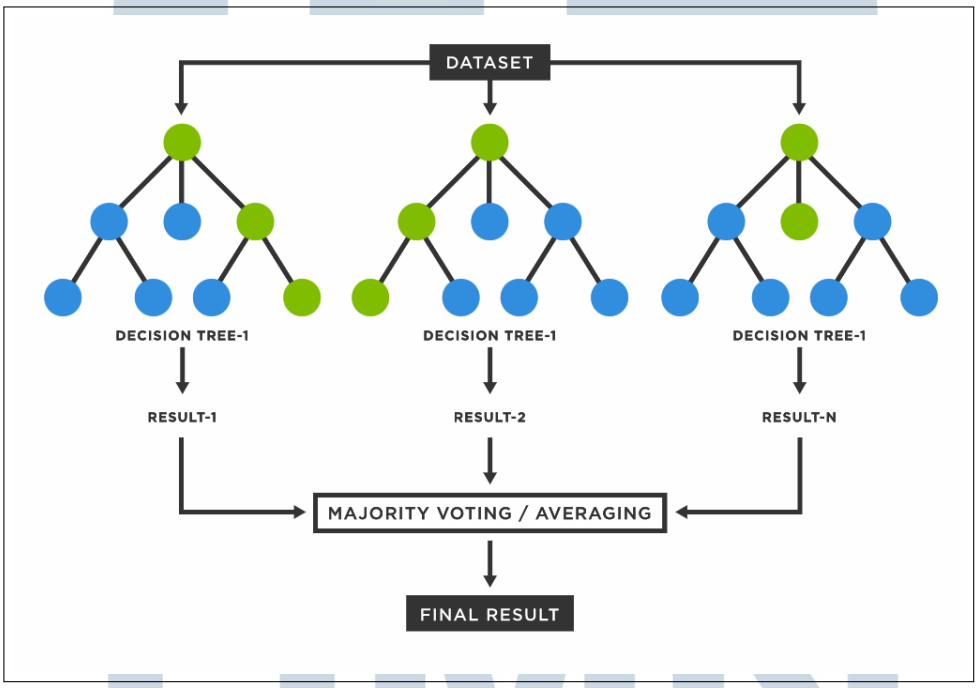
2. *Boosting* merupakan salah satu metode dari *ensemble learning* yang dimana pada metode ini menggunakan satu tipe *base model* yang dimana pelatihan dilakukan secara berurutan dan hasil dari tiap *base model* itu bergantung dari hasil *base model* sebelumnya. Algoritma *Adaptative boosting* (AdaBoost) merupakan salah satu algoritma yang termasuk kedalam model *boosting*.

3. *Stacking* merupakan salah satu metode dari *ensemble learning* yang dimana dalam melakukan pelatihan menggunakan beberapa *base model* yang kemudian dilakukan secara paralel dan independen pada setiap *base model* kemudian menggunakan satu algoritma yang berasal dari pembelajaran lain

untuk menghasilkan *output* dari gabungan setiap *base model*. Algoritma *Blending* merupakan salah satu algoritma yang termasuk kedalam model *stacking*.

2.1.8 Random Forest

Random forest merupakan salah satu algoritma dalam *ensemble learning* yang digunakan untuk melakukan pengklasifikasian data set yang berjumlah besar [26].



Gambar 2.2. Diagram algoritma *random forest*
Sumber: [27]

Gambar 3.5 merupakan diagram dari algoritma *random forest*. Cara kerja dari algoritma *random forest* ini adalah membangun beberapa *decision tree* kemudian menggabungkan keputusan dari setiap pohon yang telah dibangun dan mengambil suara terbanyak dari prediksi setiap pohon sehingga nantinya akan menghasilkan prediksi yang stabil dan akurat. Kumpulan *decision tree* dilatih dengan menggunakan metode *bagging* [26].

2.1.9 Confusion Matrix

merupakan sebuah tabel yang digunakan untuk mengukur kinerja dari *machine learning*. Pada tabel *confusion matrix* terdapat empat variabel, antara lain *True Positive* (TP) merupakan data yang bernilai positif dan diprediksi bahwa benar positif oleh sistem, *False Positive* (FP) merupakan data yang bernilai negatif namun diprediksi positif oleh sistem, *False Negative* (FN) merupakan data yang bernilai positif namun diprediksi negatif oleh sistem, dan *True Negative* (TN) merupakan data yang bernilai negatif dan diprediksi bahwa benar negatif oleh sistem [28]. Tabel 2.1 merupakan bentuk dari tabel *confusion matrix*.

	Predicted Value	
Actual Value	1	0
1	TP	FN
0	FP	TN

Tabel 2.1. Tabel *Confusion Matrix*

Confusion matrix digunakan untuk menghitung *accuracy, precision, recall* dan *F1 score*. Berikut merupakan cara yang digunakan pada *confusion matrix* untuk menghitung keempat hal tersebut [28].

1. *Accuracy* merupakan gambaran dari seberapa akurat sistem yang telah dibuat dalam melakukan klasifikasi dengan benar. Berikut merupakan rumus dari cara mencari nilai *accuracy*.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.3}$$

2. *Precision* merupakan gambaran akurasi dari data yang diminta dengan hasil prediksi yang diberikan oleh sistem. Berikut merupakan rumus dari cara mencari nilai *precision*.

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

3. *Recall* merupakan tingkat dari keberhasilan suatu sistem dalam menemukan kembali sebuah informasi. Berikut merupakan rumus dari cara mencari nilai *recall*.

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

4. *F1 Score* merupakan nilai rata-rata dari *precision* dan *recall*. Berikut merupakan rumus dari cara mencari nilai *F1 Score*

$$F1 = \frac{2 * Recall * Precision}{Recall + Precision} \tag{2.6}$$