BAB 2
LANDASAN TEORI

Telaah literatur merupakan proses sistematis untuk mengkaji, mengevaluasi, dan merangkum literatur yang relevan tentang suatu topik. Telaah literatur menghasilkan pengetahuan baru tentang topik tersebut, berupa kerangka kerja dan perspektif baru [18].

2.1 Pemilu

Pemilu atau pemilihan umum merupakan aksi penggunaan hak pilih masyarakat dalam berdemokrasi yang bertujuan untuk memilih pejabat kursi pemerintahan. Pemilu menjadi wadah pertarungan pengisian jabatan politik di pemerintahan [19]. Pemilihan umum dilakukan secara langsung oleh rakyat yang menjadi sarana kedaulatan rakyat dengan harapan untuk menghasilkan pemerintahan yang berdasar pada Pancasila dan UUD 1945. Dalam pemilihan umum tersebut, dilakukan juga proses pemilihan Presiden dan Wakil Presiden secara langsung oleh rakyat [20].

2.2 Natural Language Processing

Natural Language Processing atau biasa disingkat NLP merupakan sebuah cabang dari *machine learning* yang berkaitan dengan teks. NLP adalah teknik yang fungsi utamanya adalah untuk menganalisis teks dengan dorongan hipotesis. Tujuannya adalah mencapai pemrosesan bahasa yang mirip dengan bahasa manusia untuk diimplementasikan di berbagai bidang [21].

NLP yang merupakan teknik menganalisis yang dibantu komputer menjadi populer sejak era digitalisasi. Pasalnya, digitalisasi membuat lebih banyak cara untuk melakukan *data harvesting* secara digital. Dengan NLP juga, peneliti lebih mudah mendapatkan hasil dari *dataset* teks sembari mengurangi kompleksitas komputasi yang memberatkan [22].

2.3 Sentiment Analysis

Salah satu implementasi dari NLP adalah *sentiment analysis* [21]. *Sentiment Analysis* bertujuan untuk menghasilkan *output* berupa sentimen atau opini

dari *dataset* yang berupa teks. Dengan begitu, analisis sentimen juga dapat dikategorikan sebagai masalah klasifikasi untuk teks karena hasil dari analisis sentimen adalah klasifikasi sentimen positif, negatif atau netral dari sebuah teks. Ada tiga level dari *sentiment analysis*, yaitu [23]:

1. *Aspect-level sentiment analysis* Di level ini, analisis sentimen fokus pada satu aspek dalam kalimat. Jadi, opini atau sentimen yang dianalisis adalah opini atau sentimen dari sebuah aspek di dalam sebuah kalimat.

2. *Sentence-level sentiment analysis* Di level ini, analisis sentimen dilakukan secara lebih umum, karena opini atau sentimen yang dianalisis berdasarkan dari satu kalimat.

3. *Document-level sentiment analysis* Level ini adalah level paling umum dari analisis sentimen, yaitu analisis sentimen dari seluruh dokumen yang diberikan.

Beberapa algoritma *machine learning* yang banyak digunakan saat melakukan analisis sentimen adalah Naive Bayes, Maximum Entropy, Support Vector Machine, Neural Network, VADER dan masih banyak lagi [8].

2.4 Data Preprocessing

Sebelum proses algoritma dilakukan, data yang dikumpulkan perlu dilakukan *preprocessing* terhadap data tersebut. *Data preprocessing* berperan penting dalam *machine learning* karena dengan *preprocessing* yang baik dan benar, semakin baik performa dan hasil dari algoritma *machine learning*. Berikut adalah beberapa proses *preprocessing* yang akan digunakan dalam penelitian ini

1. *Cleaning*

   Seperti namanya, proses *cleaning* merupakan proses membersihkan teks dari karakter atau simbol selain alfabet [24].

2. *Case Folding*

   Proses *case folding* merupakan proses mengubah seluruh karakter dalam kalimat menjadi huruf kapital ataupun huruf kecil [24].

3. *Tokenization*

   Dalam proses *tokenization*, dilakukan pemenggalan kalimat per kata [24].

4. *Stopword Removal*

   Proses *stopword removal* adalah proses menghilangkan atau menghapus kata-kata yang tidak penting atau tidak memberikan arti dalam kalimat [24].

5. *Lemmatization*

   Lemmatization adalah proses untuk mengubah kata berimbuhan menjadi bentuk dasarnya [25].

6. *Labelling*

   Proses *labelling* merupakan proses memberikan label positif, negatif ataupun netral pada suatu kalimat [24].

Proses-proses di atas termasuk dalam *preprocessing* yang berguna untuk mempersiapkan data untuk proses klasifikasi atau dalam hal ini analisis sentimen. Dataset yang diambil dari internet biasanya memiliki banyak *noise* dan bagian tidak penting lainnya yang tidak berguna untuk klasifikasi. Bagian-bagian tersebutlah yang dieliminasi dalam *preprocessing* [26].

2.5 TextBlob

TextBlob adalah sebuah *library* di python [27]. *Library* ini menyediakan modul *text mining, text analysis* dan juga *text processing* [27]. TextBlob memilki *method* sentiment() yang menghasilkan parameter *polarity* dan juga *subjectivity* [27]. Untuk mengklasifikasikan label dari teks dapat menggunakan parameter *polarity* sebagai berikut:

- polarity >0 : Positif
- polarity <0 : Negatif
- polarity = 0 : Netral

2.6 VADER

VADER atau *Valence Aware Dictionary and sEntiment Reasoner* merupakan alat atau *tools* dalam proses *sentiment analysis* yang berbasis leksikon dan juga *rule-based* [27]. VADER yang menggunakan kombinasi leksikon sentimen cukup populer untuk mengatasi masalah yang berkaitan dengan teks media sosial dan

lainnya [27]. VADER menganalisis sebuah teks untuk melihat jika kata dalam teks tersebut ada di leksikon VADER dan memberi hasil bukan hanya hasil labelnya, namun juga seberapa negatif dan seberapa positif teks tersebut [27]. Adapun *compund score* sebagai salah satu *output* dari VADER yang merupakan sebuah metrik yang menghitung jumlah *lexicon rating* yang dinormalisasi antara -1 hingga 1 [27]. *Compund score* dapat digunakan untuk mengklasifikasikan label antara negatif, positif atau netral [27]. Batasan yang biasa digunakan adalah sebagai berikut:

- *Compund score* >= 0.05 : Positif
- *Compund score* <= -0.05 : Negatif
- *Compund score* >-0.05 dan <0.05 : Netral

2.7 SMOTE

SMOTE atau *Synthetic Minority Over-sampling Technique* merupakan teknik membuat replikasi dari *minority class* dengan tujuan menyeimbangkan jumlah data tanpa membuang sampel yang lebih banyak dari data [28]. Teknik *oversampling* SMOTE ini digunakan ketika data tidak seimbang atau *balance* [29].

2.8 TF-IDF

Ada beberapa metode untuk menentukan bobot kata saat melakukan klasifikasi. Salah satu yang paling populer digunakan adalah TF-IDF (*Term Frequency - Inverse Document Frequency*. TF-IDF berguna untuk mengubah teks menjadi bentuk vektor berdasarkan frekuensi kata yang muncul [30]. TF-IDF dipilih karena kemampuan TF-IDF yang dapat melakukan ektraksi fitur dengan *dataset* berukuran besar dan menjaga efektivitasnya [31]. Garis besar dari TF-IDF berdasar pada kata yang muncul lebih sering dalam dokumen akan menjadi lebih penting karena akan lebih berguna dalam klasifikasi. Maka, seringkali TF-IDF digunakan untuk melakukan ekstraksi kata menjadi vektor [26].

TF-IDF dapat direpresentasikan dengan rumus berikut.

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{2.1}$$

n_{ij} merupakan jumlah munculnya kata t_i di *file* d_j dan $\sum_k n_{k,j}$ adalah jumlah munculnya semua kata di *file* d_j.

$$\text{idf}_i = \log \frac{|D|}{\left|\left[j : t_i \in d_j\right]\right|} \tag{2.2}$$

Yang mana $|D|$ merupakan jumlah *file* dan $\left|\left[j : t_i \in d_j\right]\right|$ merupakan jumlah dokumen dengan kata t_i

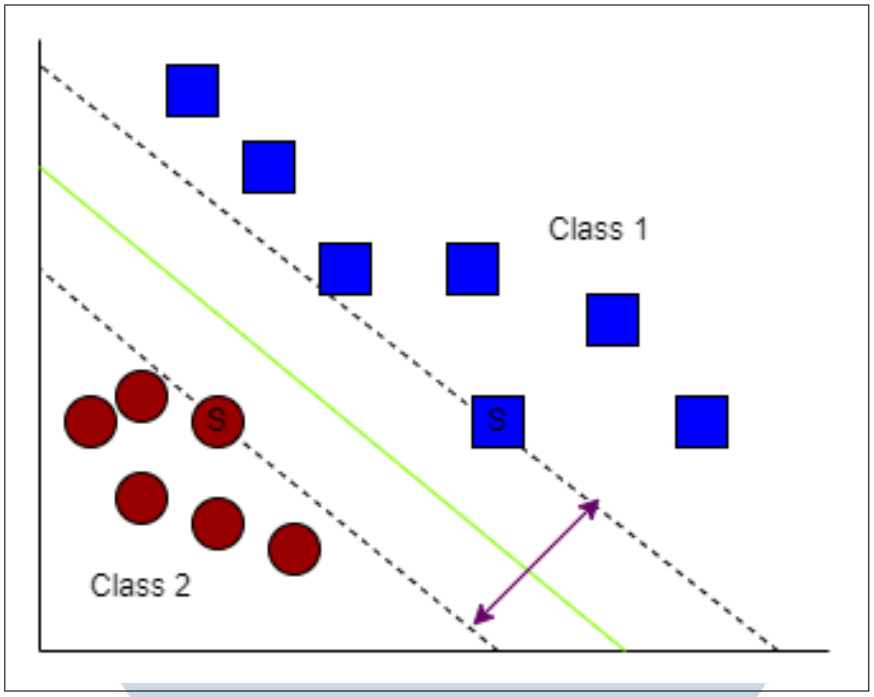
$$\text{tfidf } f_{i,j} = \frac{tf_{i,j} \times idf_i}{\sqrt{\sum_{t_i \in d_j} \left[tf_{i,j} \times idf_i\right]^2}} \tag{2.3}$$

2.9 Support Vector Machine

Support Vector Machine atau SVM merupakan algoritma *machine learning* yang memerlukan data latih atau biasa disebut sebagai *supervised learning algorithm* pertama kali dipublikasikan oleh Vapnik di tahun 1992. Fokus dari algoritma ini adalah untuk mencari *hyperplane* paling optimal untuk memisahkan dua kelas berbeda berdasarkan data latih. *Hyperplane* tersebut dapat digunakan untuk menentukan kelas yang cocok untuk data uji [32]. Dapat dikatakan juga, SVM mencari pemisah antar dua kelas. Jika dalam dua dimensi, pemisah tersebut berupa garis, namun jika dalam tiga dimensi atau lebih disebut sebagai *hyperplane* [33]. *Output* dari SVM sendiri sudah ditentukan, yaitu label sentimen dan *input*-nya adalah *vector*. SVM sendiri sudah terbukti menjadi salah satu algoritma terbaik dan paling populer untuk klasifikasi teks seperti *sentiment analysis* [7].

$$\min_{w,b} \frac{1}{2} w^2 + C \times \sum_{i=1}^{n} \varepsilon_i \text{ st } \forall i : y_i \left(\langle w, x_{ij} \rangle + b\right) \geq 1 - \varepsilon_i \quad \varepsilon \geq 0 \tag{2.4}$$

Dari persamaan 2.4, x_{mn} dan y_{mn} menunjukkan kumpulan dari data latih dan x_{ij} menunjukkan kejadian j dalam waktu i. ε_i sendiri merupakan variabel *slack* yang di dalamnya terdapat kasus yang tidak dapat dipisahkan. Lalu, variabel w adalah kemiringan dari *hyperplane* sebagai pemisah data dengan b sebagai *margin* dan C yang lebih dari 0 adalah *soft margin* yang melakukan penalti untuk *misclassified data* [26].



Gambar 2.1. Contoh grafik SVM

Gambar 2.1 merupakan visualisasi dari algoritma SVM. Terdapat dua kelas dan beberapa data observasi yang dibagi berdasarkan kelas tersebut. Garis hijau dalam gambar tersebut merupakan *hyperplane* yang menjadi pemisah antara kelas 1 dan kelas 2. Bagian di antara dua garis putus-putus yang digambarkan dengan panah berwarna ungu merupakan *margin*, yaitu jarak antara *hyperplane* dengan *support vectors*. *Support vector* yang digambarkan dengan data dengan huruf "S" sendiri merupakan titik data yang paling dekat dengan *hyperplane*.

Algoritma SVM awalnya memang didesain untuk menjadi *binary classifier*, SVM juga dapat digunakan sebagai *multiclass classifier* walaupun performanya tidak sebanding dengan performa sebagai *binary classifier*-nya [34]. Metode *multiclass classifier* yang sudah beredar adalah OvO (*One vs. One*) dan OvR (*One vs. Rest*) [34]. Di *library* dari scikit learn, OvR sudah menjadi *default* untuk *parameter* "decision_function_shape" dan OvO tidak digunakan lagi [35].

2.10 Hyperparameter

Hyperparameter adalah parameter yang diberikan oleh pengguna untuk mengatur atau mengontrol jalannya proses *learning* [36]. Nilai dari *hyperparameter* sangat mempengaruhi performa model *machine learning*, khususnya pada

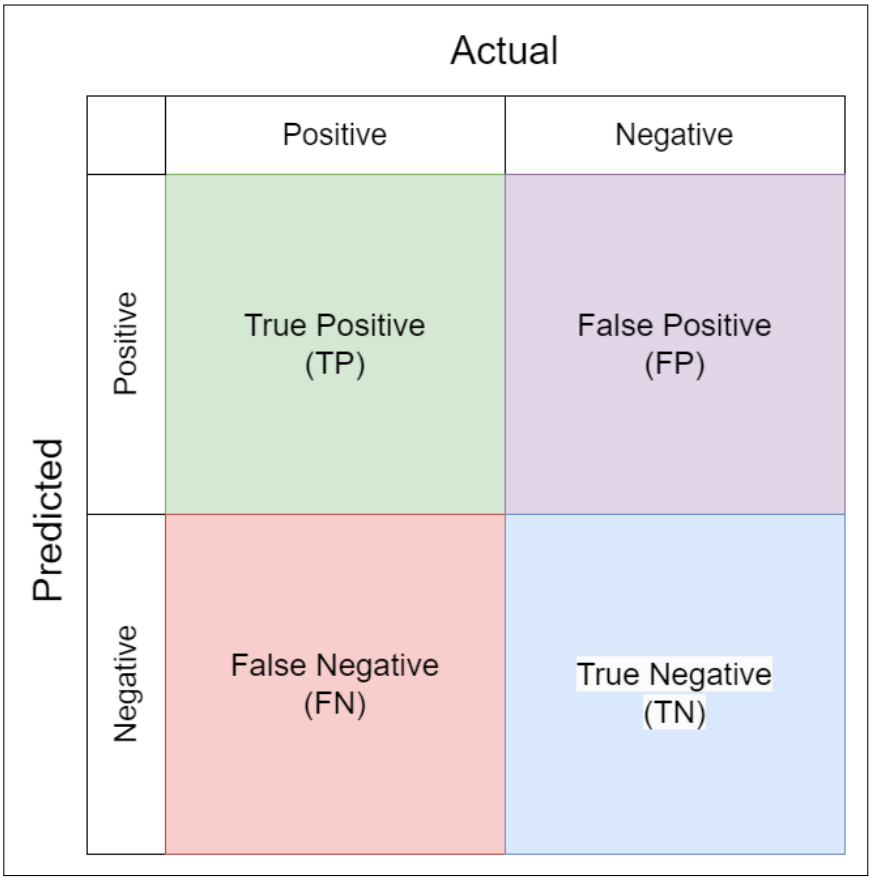
supervised machine learning [37] seperti salah satunya algoritma SVM. Adapun proses mencari *hyperparameter* yang dapat digunakan disebut sebagai *hyperparameter tuning* [38]. Salah satu teknik *hyperparameter tuning* adalah *grid search* [36].

2.11 Grid Search

Grid search merupakan salah satu teknik *hyperpatameter tuning* yang sering digunakan [36]. *Grid search* menentukan *hyperparameter* optimal dengan cara menelusuri setiap konfigurasi optimal dari parameter yang diberikan [36]. Pencarian dengan metode *grid search* memakan waktu komputasi yang cukup lama karena memiliki *high dimensional spaces* [39]. Salah satu *library* yang dapat digunakan untuk implementasi *grid search* adalah GridSearchCV dari scikit-learn.

2.12 Confusion Matrix

Confusion matrix digunakan untuk mengukur atau menghitung akurasi dari sebuah model yang dibuat dengan cara membandingkan hasil prediksi dengan data aktual [40]. *Confusion matrix* sudah menjadi teknik pengujian yang fundamental dalam *machine learning* [40].



Gambar 2.2. Confusion matrix

Gambar 2.2 merupakan visualisasi dari *confusion matrix*. Keadaan *true positive* merupakan kondisi yang terjadi ketika data aktual yang positif diprediksi secara benar, *false positive*, merupakan kondisi yang terjadi ketika data aktual yang positif diprediksi secara salah *false negative*, merupakan kondisi yang terjadi ketika data aktual yang negatif diprediksi secara salah dan *true negative* merupakan kondisi yang terjadi ketika data aktual yang negatif diprediksi secara benar [40].

2.13 Classification Report

Classification Report dari scikit-learn berisikan laporan dari model *machine learning* yang berisi *precision, recall, f1 score* dan juga *accuracy* [41]. *Accuracy* sering kali dianggap yang paling penting dalam *classification report*, namun metrik lain juga tidak kalah pentingnya [42].

2.13.1 Accuracy

Accuracy dalam model *machine learning* mengukur hasil prediksi yang benar atau tepat. Hasil prediksi tersebut termasuk *true positive* dan juga *true negative* [42].

$$\text{Accuracy} = \frac{tp+tn}{tp+tn+fp+fn} \tag{2.5}$$

Adapun persamaan untuk mengukur *accuracy* untuk seluruh kelas dalam model *machine learning multiclass* seperti dalam persamaan berikut [43]:

$$\text{Akurasi Total} = \frac{\sum tp(i)}{\sum \text{Total Prediksi}} \tag{2.6}$$

2.13.2 Recall

Recall atau *sensitivity* merupakan tingkat probabilitas dari model *machine learning* untuk mendeteksi kasus *true positive* [44].

$$\text{Recall} = \frac{tp}{tp+fn} \tag{2.7}$$

2.13.3 Precision

Precision atau presisi merupakan tingkat kemampuan dari model *machine learning* untuk memberi label yang benar terhadap kasus positif [42].

$$\text{Precision} = \frac{tp}{tp+fp} \tag{2.8}$$

2.13.4 F1 Score

Skor F1 merupakan rata-rata dari *precision* dan *recall* yang harmonis. Skor terbaik adalah 1.0 dan skor terburuk adalah 0.0. [45].

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.9}$$

2.14 Information Processing

Pemrosesan informasi atau *information processing* secara umum dibagi menjadi dua, yakni sistematis dan juga heuristik. Pemrosesan informasi secara sistematis menggunakan upaya kognitif yang besar saat menerima informasi untuk mengevaluasi dan menilai validitas pesannya terkait pengambilan kesimpulan. Sedangkan secara heuristik, pemrosesan informasi dilakukan dengan sedikit analisis terhadap pesannya [46].