# BAB 2
# LANDASAN TEORI

## 2.1 Generative AI

*Generative AI* merupakan sebuah teknologi *machine learning* yang dapat menghasilkan konten berupa teks, gambar, musik dan video yang baru dengan menganalisis data-data terkait dengan konten yang sebelumnya pernah dibuat [7]. Contoh dari *generative AI* yang terkenal seperti Midjourney [29], Stable Diffusion [30], dan DALL-E [31]. *Generative AI* yang disebutkan tadi dapat menerima *input* dari pengguna berupa teks dan mengeluarkan *output* berupa gambar [32].

## 2.2 Analisis Sentimen

Analisis sentimen digunakan untuk mengidentifikasi dan mengkategorikan emosi atau pendapat pengguna terhadap suatu hal [33]. Analisis sentimen ini termasuk ke dalam permasalahan klasifikasi teks dalam *machine learning* [34]. Proses memperoleh informasi dari suatu teks yang bermakna pada analisis sentimen menggunakan teknik NLP dan dapat menentukan sikap penulis yang terbagi menjadi positif, negatif atau netral [12].

Terdapat beberapa permasalahan yang masih sulit untuk dipecahkan oleh analisis sentimen seperti kata-kata gaul, singkatan, *emoticon*, serta sarkasme [35]. Maka dari itu, selain membutuhkan data *training* berupa teks yang telah di-*clean* terlebih dahulu dengan menghilangkan *white space*, tanda petik, *tag HTML*, karakter spesial dan simbol [36], analisis sentimen juga memerlukan pandangan *user* (manusia) untuk menganalisis hasil dari komputasi yang dilakukan [37].

## 2.3 Textblob

Textblob merupakan *library* untuk memproses teks yang tersedia di Python 2 dan Python 3. Textblob menyediakan API dasar untuk melakukan tugas terkait dengan NLP seperti identifikasi *part-of-speech* (POS), mengekstrak kata benda, melakukan analisis sentimen, klasifikasi serta melakukan terjemahan [38]. Textblob akan mengembalikan 2 nilai yaitu:

1. Nilai *polarity* dengan kisaran [-1, 1] yang menunjukkan ukuran emosi dari sebuah teks. Semakin mendekati nilai -1 maka menunjukkan sentimen negatif

dan sebaliknya.

2. Nilai *subjectivity* yang menunjukkan seberapa subjektif suatu kalimat dengan nilai mendekati 0 menunjukkan bahwa kalimat semakin objektif dan sebaliknya jika mendekati angka 1 maka kalimat semakin subjektif.

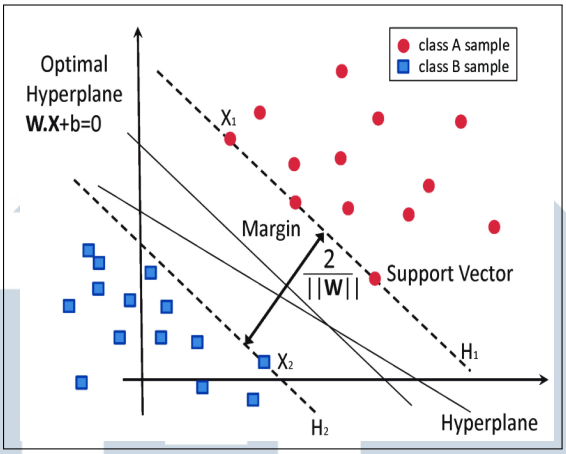
## 2.4 Support Vector Machine

Support Vector Machine (SVM) merupakan algoritma klasifikasi yang menggunakan konsep *hyperplane*, yaitu garis atau bidang yang memisahkan data dengan label yang berbeda. Algoritma ini akan berfokus untuk membuat *hyperplane* yang paling optimal untuk mengkategorikan data *training* yang sudah diberi label (*supervised learning*) [39]. Secara umum, *vector* direpresentasikan sebagai urutan elemen-elemen yang mewakili titik dalam ruang berdimensi-n yang memiliki arah. Misalnya, dalam ruang dua dimensi, sebuah *vector* dapat direpresentasikan sebagai [x, y] di mana x dan y membentuk sebuah titik koordinat. Dalam ruang tiga dimensi, *vector* direpresentasikan sebagai [x, y, z] dan seterusnya [40]. Sementara itu *support vectors* merupakan titik koordinat yang terletak paling dekat dengan *hyperplane* ini, dan mereka sangat penting karena menentukan posisi dan orientasi *hyperplane* [41]. Persamaan *hyperplane* dapat dituliskan ke dalam rumus (2.1) sedangkan persamaan *margin* dapat dilihat pada rumus (2.2).

$$|W.X + b| = 1 \tag{2.1}$$

$$M = 2/||W|| \tag{2.2}$$

Dalam konteks ini, 'W' merupakan vektor bobot yang tegak lurus terhadap *hyperplane*, sedangkan $\frac{2}{\|W\|}$ mengindikasikan *margin* yang merupakan jarak *hyperplane* dari titik awal sepanjang vektor *w*. *Dataset* pelatihan $(x_1, y_1), \ldots, (x_n, y_n)$ mewakili 'n' titik yang digunakan dalam pelatihan model. Di dalam penelitian ini, '$x_n$' mencerminkan kalimat atau aspek yang telah divektorisasi, sedangkan '$y_n$' mencerminkan label sentimen dari kalimat tersebut.




Gambar 2.1. Cara kerja SVM

SVM atau Support Vector Machine telah banyak digunakan dan terbukti efektif untuk analisis sentimen dan *text mining* [42, 43]. SVM memiliki fungsi untuk memecahkan masalah klasifikasi. Metode ini menggunakan *kernel* untuk mengubah data ke dalam ruang dimensi yang lebih tinggi agar dapat mengklasifikasikan data yang tidak dapat dipisahkan secara *linear*. *Kernel* membutuhkan pengaturan parameter yang dapat disesuaikan, yang biasa disebut sebagai *hyperparameter* [44]. Ada beberapa jenis fungsi *kernel* yang umum digunakan, seperti *kernel linear*, *kernel* polinomial, dan *kernel Radial Basis Function* (RBF) [25].

Berikut adalah persamaan matematis untuk *kernel linear* yang dapat dilihat pada persamaan (2.3).

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j \tag{2.3}$$

Berikut adalah persamaan matematis untuk *kernel sigmoid* yang dapat dilihat pada persamaan (2.4).

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + r) \tag{2.4}$$

Berikut adalah persamaan matematis untuk *kernel RBF* yang dapat dilihat pada persamaan (2.5).

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right) \tag{2.5}$$

Berikut adalah persamaan matematis untuk *kernel polinomial* yang dapat

dilihat pada persamaan (2.6).

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d \tag{2.6}$$

## 2.5 SMOTE

Synthetic Minority OverSampling Technique (SMOTE) digunakan untuk menyeimbangkan kumpulan data dengan melakukan sintesis pada data minoritas. Data *training* terdiri dari *minority data points* ($S_{min}$) dan *majority data points* ($S_{maj}$). Untuk setiap $(X_i, Y_i) \in S_{min}$, SMOTE akan membuat sebuah *data point* baru di garis penghubung ($X_i$) dan salah satu *neighbors* terdekat yang dipilih secara acak. [45].

$$X_{new} = X_i + \left(\frac{X_j}{X_i}\right) \times \delta \tag{2.7}$$

Persamaan matematis untuk SMOTE dapat dilihat pada persamaan (2.7). Xi merupakan titik dari label minoritas dan Xj merupakan titik dari label mayoritas, $\delta$ adalah bilangan acak antara 0 dan 1. Bilangan acak ini memastikan agar titik yang dibentuk tidak hanya terletak pada satu tempat saja.

**Algorithm 1** Pseudocode of SMOTE Algorithm (Part 1)

***function*** SMOTE($T$, $N$, $k$)
***Input***: $T$, $N$, $k$ *#minority class examples, Amount of oversampling, #nearest neighbors*
***Output***: $(N/100) \times T$ *synthetic minority class samples*
***Variables***:
*Sample: array for original minority class samples*
*newindex: keeps a count of number of synthetic samples generated, initialized to 0*
*Synthetic: array for synthetic samples*
**if** $N < 100$ **then**
    *Randomize the T minority class samples*
    $T = (N/100) \times T$
    $N = 100$
**end if**

**Algorithm 2** Pseudocode of SMOTE Algorithm (Part 2)

1: $N = \text{int}(N/100)$ *The amount of SMOTE is assumed to be in integral multiples of 100*
2: **for** $i = 1$ **to** $T$ **do**
3: *Compute k nearest neighbors for i, and save the indices in the nn-array*
4: POPULATE($N$, $i$, nn0array)
5: **end for**
6: ***end function***

*Pseudocode* untuk SMOTE dapat dilihat pada Algoritma 1 dan Algoritma 2 [25]. Prosesnya dimulai dengan menetapkan nilai N, yang merupakan total jumlah *oversampling* (dinyatakan sebagai nilai integer). Ini dilakukan dengan mencari nilai yang dapat menghasilkan perbandingan distribusi kelas 1:1 atau menggunakan proses *wrapper* untuk menemukan distribusi kelas. Kemudian, masuk ke dalam perulangan, dimulai dengan memilih secara acak *instance* kelas minoritas dari himpunan data *training*. Setelah itu, *K nearest neighbors* diperoleh, dengan nilai k yang diatur secara *default* menjadi 5.

Langkah terakhir adalah mendapatkan perbedaan antara *feature vector* dari sampel yang sedang diproses dan setiap *neighbor* yang dipilih. Selanjutnya, perbedaan ini dikalikan dengan angka acak antara 0 dan 1; Hasilnya kemudian ditambahkan ke *feature vector* sebelumnya. Selanjutnya, terdapat proses pemilihan *random point* yang ada pada segmen garis fitur yang telah dibuat.

## 2.6 Confusion Matrix

Confusion matrix merupakan matriks yang digunakan untuk mengevaluasi performa dari sebuah model klasifikasi yang telah dibuat. Matriks ini akan membandingkan nilai aktual dengan hasil prediksi dari model *machine learning*. Ukuran dari matriks ini bergantung kepada jumlah target *class* dari label yang ingin diprediksi. Umumnya confusion matrix akan menghasilkan 4 buah nilai yaitu *accuracy*, *precision*, *recall*, dan *f1 score* [46]. Gambaran dari confusion matrix dan persamaan matematis dari keempat nilai tersebut dapat dilihat pada Gambar 2.2. TP merupakan *True Positive*, TN merupakan *True Negative*, TNR merupakan *True Neutral*, FP merupakan *False Positive*, FN merupakan *False Negative*, dan FNR merupakan *False Neutral*.

| Actual | Prediction | | |
|---|---|---|---|
| | Positive | Negative | Neutral |
| Positive | TP | FN | FNR |
| Negative | FP | TN | FNR |
| Neutral | FP | FN | TNR |

Gambar 2.2. Confusion matrix yang digunakan

$$\text{Accuracy} = \frac{\textit{Number of texts correctly classified}}{\textit{Total number of texts}} * 100\% \tag{2.8}$$

$$\text{Precision} = \frac{\textit{The number of texts correctly classified into the category}}{\textit{The number of texts classified into the category}} * 100\% \tag{2.9}$$

$$\text{Recall} = \frac{\textit{The number of texts correctly classified into the category}}{\textit{Total number of texts in the category}} * 100\% \tag{2.10}$$

$$\text{F1 Score} = 2 \times \frac{\textit{Precision} \times \textit{Recall}}{\textit{Precision} + \textit{Recall}} * 100\% \tag{2.11}$$

## 2.7 Pemodelan Topik

Pemodelan topik adalah sebuah algoritma yang sangat populer dan penting dalam *machine learning* dan *Natural Language Processing* (NLP). Metode ini memilih pendekatan untuk mengekstrak topik-topik tersembunyi dari dokumen-dokumen besar [27]. Pemodelan topik bekerja dengan mengamati bagaimana konstruk-konstruk berbeda muncul bersama-sama dalam koleksi teks. Ini sejalan dengan gagasan bahwa kata-kata yang sering muncul bersama-sama memiliki makna yang terkait. Sebagai contoh, kata ”hujan” dan ”payung” sering muncul bersama-sama dalam konteks yang sama [47].

Pemodelan topik berfokus untuk mengidentifikasi topik-topik utama yang ada dalam teks dan bagaimana topik-topik tersebut terhubung satu sama lain. Pemodelan topik menggunakan sebuah korpus teks sebagai *input* dan mencoba untuk mendapatkan dua set distribusi [47]:

1. Distribusi Topik (T): Ini adalah distribusi probabilitas untuk setiap topik dalam korpus teks. Setiap topik memiliki sejumlah bobot yang menunjukkan seberapa sering topik tersebut muncul dalam korpus.

2. Distribusi Dokumen (Z): Ini adalah distribusi probabilitas untuk setiap dokumen dalam korpus terhadap setiap topik. Distribusi ini memberikan representasi yang dapat diinterpretasikan dari dokumen-dokumen dalam korpus.

## 2.8 Latent Dirichlet Allocation

LDA (Latent Dirichlet Allocation) adalah algoritma pemodelan topik yang paling populer dalam aplikasi ekstraksi topik dari kumpulan teks [48]. Asumsi dasar utama dari model LDA adalah bahwa setiap dokumen mengandung beragam topik, dan setiap topik memiliki distribusi probabilitas atas kata-kata. Salah satu keunggulan utama dari model LDA adalah bahwa topik dapat diekstraksi dari kumpulan dokumen tanpa *input knowledge* sebelumnya [49].

Jumlah topik merupakan parameter yang paling penting dalam menghasilkan hasil akhir dari pemodelan topik. Biasanya, jumlah topik bergantung pada pertanyaan penelitian dan tujuan penelitian. Jika nilai k terlalu tinggi, hasilnya dapat menjadi tidak lengkap dalam hal informasi; sedangkan jika nilai k terlalu rendah, dapat terjadi pengelompokan yang berlebihan [48]. Performa dari *topic modelling* dapat diukur menggunakan *coherence value*. *Coherence value* ini memiliki nilai yang berbeda-beda tergantung jumlah topik yang dimasukkan. Semakin tinggi nilai *coherence value* maka semakin baik pula distribusi pemodelan topik dilakukan [50].

$$p(\phi_{1:k}, \theta_{1:M}, X_{1:M}, w_{1:M}) = \prod_{i=1}^{k} p(\phi_i) \prod_{m=1}^{M} p(\theta_m) \prod_{n=1}^{N} p(X_{m,n}|\theta_m) p(w_{m,n}|\phi_{1:k}, x_{m,n}) \tag{2.12}$$

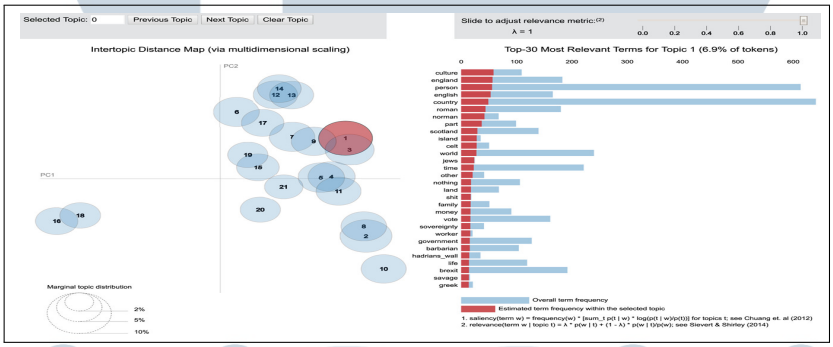
Model LDA menggunakan persamaan (2.12) untuk mewakili secara matematis topik-topik tersembunyi dan distribusi kata-kata yang diamati dalam proses LDA. Persamaan tersebut mengasumsikan bahwa korpus $D$ mengandung $k$ topik dan $M$ dokumen, dengan setiap dokumen memiliki $N$ kata. Notasi $\alpha$ dan $\beta$ adalah *hyperparameter Dirichlet* untuk distribusi topik $\theta$ dalam dokumen dan distribusi kata $\phi$ dalam topik. Dalam persamaan (2.12), $p(\phi_k)$ adalah distribusi kata-kata untuk topik $k$, $p(\theta_m)$ adalah distribusi topik untuk dokumen $m$, dan $x_{m,n}$ adalah topik yang diberikan kepada kata $w_{m,n}$ dalam dokumen $m$ [51].

Dalam persamaan (2.12), $x$ dan $w$ mewakili topik dan kata-kata. Berdasarkan persamaan tersebut, untuk setiap topik $i$, distribusi $p(\phi_i)$ ditentukan,

kemudian untuk setiap dokumen $m$, distribusi $p(\theta_m)$ ditentukan, dan kemudian sebuah topik $x$ dipilih dari distribusi topik $p(\theta_m)$ untuk setiap posisi kata. Akhirnya, sebuah kata $w$ dipilih dari distribusi $p(\phi_k)$. Dengan menggunakan ini, topik-topik yang paling sering muncul dalam korpus dapat diidentifikasi dan ditampilkan [51].

## 2.9 PyLDAvis

PyLDAvis adalah alat visualisasi interaktif yang bisa diakses melalui *library* Python yang digunakan untuk menampilkan topik-topik yang diidentifikasi oleh pendekatan LDA (Latent Dirichlet Allocation). Dengan menggunakan PyLDAvis, pengguna dapat dengan mudah melihat dan memahami topik-topik yang ada dalam suatu korpus teks. Alat ini menyediakan dua tampilan utama yang memberikan pemahaman yang mendalam mengenai topik-topik tersebut [52].




Gambar 2.3. Tampilan PyLDAvis

Pertama, pada panel kiri, PyLDAvis menampilkan topik-topik dalam bentuk lingkaran dua dimensi. Setiap lingkaran mewakili satu topik, dan ukuran lingkaran mencerminkan tingkat prevalensi atau kepentingan topik tersebut dalam korpus. Lingkaran yang lebih besar menunjukkan topik yang lebih dominan atau sering muncul dalam dokumen-dokumen. Sementara itu, pada panel kanan, terdapat diagram batang yang menampilkan istilah-istilah penting dari setiap topik. Diagram ini memberikan gambaran yang lebih detail mengenai makna dan signifikansi setiap topik berdasarkan istilah-istilah yang sering muncul dalam topik tersebut. PyLDAvis juga memiliki *slider* yang memungkinkan pengguna untuk menyesuaikan tingkat relevansi istilah yang ditampilkan, sehingga analisis topik menjadi lebih fleksibel dan sesuai dengan kebutuhan pengguna [53].

## 2.10 Text Pre-processing

Sebelum melakukan proses klasifikasi, data komentar perlu diproses terlebih dahulu agar dimensi dari model ruang vektor menjadi lebih kecil dan menghilangkan berbagai simbol atau karakter yang tidak relevan dan malah membuat proses klasifikasi lebih sulit untuk dilakukan. Dengan melakukan *data pre-processing*, proses klasifikasi akan menjadi lebih cepat dan akurat. Tahap *pre-processing* akan menghilangkan simbol dan karakter spesial seperti emoji, menghilangkan *tag HTML*, menyamakan bentuk kata-kata ke *lowercase*, mengambil token dari suatu kalimat, menghilangkan kata-kata gaul dan mengubah menjadi kata baku, serta melakukan *labeling* sebagai patokan uji performa yang dilakukan [54].

### 2.10.1 Data Cleaning

*Data cleaning* dalam analisis sentimen mengacu pada proses mempersiapkan teks untuk dianalisis dengan menghapus informasi yang tidak relevan. Ini merupakan langkah penting untuk memastikan bahwa model analisis sentimen mampu mengidentifikasi dan menganalisis sentimen pada suatu teks dengan efektif [55]. Pada tahapan *data cleaning* terdapat proses untuk menghapus karakter yang dianggap tidak memberikan kontribusi pada analisis sentimen [56]. Pada tahapan ini juga dilakukan *case folding* pada data, hal ini dilakukan untuk mencegah adanya perhitungan pada kata-kata yang sama namun memiliki huruf kapital yang berbeda [57].

### 2.10.2 Tokenization

*Tokenization* adalah proses membagi suatu kalimat teks menjadi bagian-bagian kecil yang memiliki makna, seperti kata-kata atau pun frasa yang disebut sebagai token. Daftar token ini digunakan sebagai *input* untuk tahapan analisis sentimen dan pemodelan topik nantinya. Setiap kalimat pada komentar dipisahkan menjadi kata-kata atau frasa, yang kemudian akan diproses lebih lanjut dengan melakukan penghapusan terhadap *stop word* dan *stemming*. Dengan demikian, tokenisasi membantu mengatur teks menjadi unit-unit yang lebih ter struktur untuk analisis sentimen yang lebih efektif [58].

### 2.10.3 Removing Stop Word

Pada tahapan ini dilakukan penghapusan terhadap kata-kata umum yang tidak memberikan banyak informasi dalam analisis sentimen, seperti ”*the*”, ”*and*”, ”*is*”, dan sebagainya. *Stop word removal* membantu memfokuskan pada kata-kata yang lebih bermakna dalam mengekspresikan sentimen [59].

### 2.10.4 Lemmatization

*Lemmatization* adalah proses mengubah kata-kata menjadi bentuk dasar atau akar kata sehingga kata-kata yang memiliki makna serupa dapat dihubungkan dengan satu kata yang sama. Misalnya, kata ”*rocks*” akan diubah menjadi ”*rock*”, ”*corpora*” menjadi ”*corpus*”, dan ”*better*” menjadi ”*good*”. Contoh lainnya adalah kata-kata ”*tester*”, ”*testing*”, dan ”*tested*” yang semua nya memiliki akar kata yang sama yaitu ”*test*”. *Lemmatization* membantu dalam menganalisis dan memproses teks dengan lebih efisien karena kata-kata yang memiliki makna serupa dianggap sebagai satu entitas yang sama [60].

### 2.10.5 Labeling

*Labeling* pada data merupakan aspek penting dalam analisis sentimen. Proses ini akan memberikan label pada data teks untuk mencerminkan sentimen yang diungkapkan. Secara tradisional *labeling* melibatkan anotasi manual oleh penilai manusia yang ahli di bidang tersebut, hal ini dapat memakan waktu dan biaya yang cukup banyak ketika menangani data teks dalam jumlah yang besar. Untuk memecahkan masalah tersebut diciptakan lah otomatisasi proses pemberian label data dengan menggunakan berbagai sumber daya terkait dengan *lexicon*, yang terdiri dari kamus atau *dataset* yang sudah diberi label pada kata-kata dan frasa dalam informasi sentimen. Pemberian label pada analisis sentimen dapat dilakukan dengan menggunakan *library* yang memanfaatkan *lexicon* ini seperti VADER dan TextBlob [61].

## 2.11 TF-IDF

TF-IDF (Term Frequency–Inverse Document Frequency) adalah teknik pembobotan yang umum digunakan dalam pengambilan informasi dan *data mining*. TF atau *term frequency* merujuk pada frekuensi kata dalam teks, sedangkan

IDF merujuk pada *inverse* dari indeks frekuensi teks. Pendekatan ini mengukur pentingnya sebuah kata dalam sebuah dokumen relatif terhadap seluruh korpus dokumen. Kata-kata yang sering muncul dalam sebuah dokumen namun jarang muncul dalam korpus secara keseluruhan dianggap lebih penting [55]. Oleh karena itu, teknik ini banyak digunakan dalam ekstraksi kata kunci, perbandingan kesamaan teks, dan klasifikasi topik.

Algoritma TF-IDF digunakan untuk menghitung setiap kata dan nilai bobotnya dalam setiap teks dokumen. Algoritma ini mempertimbangkan probabilitas TF kemunculan kata dalam satu teks dan bobot IDF kata dalam seluruh dokumen. Misalnya, jika terdapat N teks dalam satu dokumen, bobot kata w dalam teks s dihitung sebagai berikut [46]:

$$t_{sw} = \text{tf}w \times \text{idf}w = \text{tf}w \times \log\left(\frac{N}{Nw}\right) \tag{2.13}$$