BAB 2
LANDASAN TEORI

2.1 Analisis Sentimen

Analisis Sentimen adalah studi komputasi tentang pendapat, sikap, dan emosi orang terhadap suatu entitas. Entitas dapat mewakili individu, peristiwa, atau topik. Tujuan utama dalam Analisis Sentimen adalah untuk mengklasifikasikan dokumen opini sebagai pernyataan opini atau sentimen positif atau negatif [10]. Ada tiga level klasifikasi utama dalam Analisis Sentimen yaitu:

1. Level dokumen bertujuan untuk mengklasifikasikan dokumen opini sebagai pernyataan opini atau sentimen positif atau negatif.
2. Level kalimat bertujuan untuk mengklasifikasikan sentimen yang diungkapkan dalam setiap kalimat.
3. Level aspek bertujuan untuk mengklasifikasikan sentimen sehubungan dengan aspek tertentu dari entitas.

2.2 Analisis Sentimen Berbasis Aspek

Analisis sentimen tradisional berfokus pada mengklasifikasikan keseluruhan sentimen yang diungkapkan dalam teks tanpa menentukan tentang apa sentimen tersebut. Ini mungkin tidak cukup jika teks secara bersamaan mengacu pada topik atau entitas yang berbeda (juga dikenal sebagai aspek), kemungkinan mengungkapkan sentimen yang berbeda terhadap aspek yang berbeda. Mengidentifikasi sentimen yang terkait dengan aspek tertentu dalam teks adalah proses yang lebih kompleks yang dikenal sebagai analisis sentimen berbasis aspek [11]. Analisis sentimen aspek adalah variasi dari analisis sentimen yang mengidentifikasi dan mengevaluasi opini atau perasaan yang terkait dengan aspek tertentu dari teks [2]. Terdapat banyak bagian yang bisa didapat dari analisis sentimen berbasis aspek ini yaitu aspek yang dibahas, sentimen di dalamnya, dan penyebab sentimen.

2.3 *Text Mining*

Text mining adalah proses ekstraksi informasi yang berharga dan bermakna dari teks yang tak terstruktur. Hal ini melibatkan penggunaan teknik dan algoritma komputasi untuk mengenali dan mengekstrak pola, hubungan, dan informasi penting dari teks [12]. Dalam *text mining*, teks yang tak terstruktur, seperti dokumen, artikel, *tweet*, atau posting blog, diubah menjadi bentuk terstruktur yang dapat dianalisis oleh komputer. Teknik-teknik seperti pemrosesan bahasa alami, pemodelan statistik, dan pembelajaran mesin digunakan untuk mengidentifikasi kata kunci, tema, sentimen, atau entitas tertentu dalam teks. Tujuan dari *text mining* adalah untuk mendapatkan pemahaman yang lebih dalam dari teks yang terkumpul, menemukan wawasan baru, dan menghasilkan pengetahuan yang berharga. Aplikasi *text mining* dapat meliputi analisis sentimen, klasifikasi teks, ekstraksi informasi, pengelompokan dokumen, deteksi topik, dan banyak lagi. Teknik ini digunakan secara luas dalam berbagai bidang, termasuk bisnis, akademik, pemerintahan, dan penelitian ilmiah untuk menggali pengetahuan dari teks yang ada.

2.4 *Web Scraping*

Web Scraping adalah proses pengambilan sebuah dokumen semi-terstruktur dari internet, umumnya berupa halaman-halaman web dalam bahasa *markup* seperti *HTML* atau *XHTML*, dan menganalisis dokumen tersebut untuk diambil data tertentu dari halaman tersebut untuk digunakan bagi kepentingan lain. *Web scraping* sering dikenal sebagai *screen scraping*. *Web Scraping* tidak dapat dimasukkan dalam bidang data mining karena data *mining* menyiratkan upaya untuk memahami pola semantik atau tren dari sejumlah besar data yang telah diperoleh. Aplikasi *web scraping* (juga disebut *intelligent*, *automated*, atau *autonomous agents*) hanya fokus pada cara memperoleh data melalui pengambilan dan ekstraksi data dengan ukuran data yang bervariasi [13].

2.5 *Preprocessing*

Preprocessing adalah langkah penting dalam analisis teks yang melibatkan serangkaian tahapan untuk membersihkan, mengubah, dan mempersiapkan teks mentah sebelum dianalisis lebih lanjut. Tujuan dari *preprocessing* adalah untuk meningkatkan kualitas data teks, mengurangi kekacauan, dan membuatnya lebih mudah dipahami oleh algoritma komputasi. Berdasarkan hasil penelitian [14], menunjukkan bahwa *preprocessing* memiliki pengaruh yang cukup baik dalam meningkatkan kinerja sistem. Berikut adalah tahapan yang akan dilakukan dalam proses *preprocessing*:

2.5.1 *Cleaning*

Cleaning adalah proses membersihkan data teks dari *noise*, informasi yang tidak relevan, atau format yang tidak diinginkan. Tujuan dari tahap ini adalah untuk membersihkan teks agar lebih mudah diproses dan mencegah gangguan dalam analisis teks.

2.5.2 *Case Folding*

Tahapan *case folding* dalam *preprocessing* adalah proses mengubah semua huruf dalam teks menjadi huruf kecil atau huruf besar. Tujuan dari *case folding* adalah untuk menghilangkan perbedaan dalam penulisan yang disebabkan oleh penggunaan huruf kapital atau huruf kecil yang berbeda. Misalnya “Bisa” menjadi “bisa”.

2.5.3 *Tokenizing*

Tokenizing adalah proses membagi teks menjadi unit-unit yang lebih kecil yang disebut token. Token-token ini biasanya berupa kata, frasa, atau karakter terpisah yang memiliki makna dalam konteks yang diberikan. Misalnya teks awal adalah “Hari ini sangat cerah” menjadi token-token terpisah [“Hari”, “ini”, “sangat”, “cerah”].

2.5.4 *Normalization*

Pada tahapan ini merupakan proses untuk menyesuaikan semua kata non-baku yang ada menjadi kata baku sesuai dengan Kamus Besar Bahasa Indonesia (KBBI). Misalnya "s*abi*" menjadi "bisa".

2.5.5 *Stemming*

Stemming merupakan tahapan proses mengubah kata-kata menjadi bentuk dasar atau kata dasar. Tahapan ini bertujuan untuk menghilangkan awalan dan akhiran dari kata-kata agar hanya menyisakan bentuk dasar kata tersebut. Misalnya adalah kata "perasaan" diubaj menjadi "rasa".

2.5.6 *Stopword Removal*

Tahapan *Stopword Removal* adalah proses menghilangkan kata-kata umum yang dianggap tidak memiliki makna penting atau kontribusi yang signifikan dalam pemrosesan teks. Dalam konteks bahasa Indonesia, beberapa contoh kata-kata yang sering dihapus dalam tahapan ini meliputi kata-kata seperti "dan", "atau", "di", "dari", "ke", "untuk", "adalah", "saya", "kamu", "kami", "mereka", "agar", "tapi", "juga", dan sebagainya. Tujuan utama dari tahapan ini adalah untuk menyaring kata-kata yang tidak memberikan informasi penting atau bermakna dalam teks, sehingga dapat meningkatkan efisiensi pemrosesan dan menganalisis teks lebih fokus pada kata-kata yang memiliki makna yang lebih penting.

2.6 *Term Frequency Inverse Document Frequency (TF-IDF)*

Term Frequency-Inverse Document Frequency (TF-IDF) merupakan sebuah algoritma untuk menunjukkan frekuensi kata yang keluar dalam dokumen [15]. *TF-IDF* sangat berguna dalam berbagai tugas pemrosesan teks, termasuk mesin pencari, analisis teks, klasifikasi teks, dan rekomendasi. Dengan

menggunakan skor *TF-IDF*, kita dapat mengetahui seberapa pentingnya sebuah kata kunci dalam konteks dokumen dan melakukan analisis dan pengambilan keputusan yang lebih akurat berdasarkan bobot kata kunci tersebut.

Secara umum, *TF-IDF* sendiri merupakan bagian dari dua statistik, yaitu *TF (Term Frequency)* dan *IDF (Inverse Document Frequency)*. *Term Frequency (TF)* adalah frekuensi munculnya kata dalam suatu dokumen, sedangkan *Document Frequency (DF)* adalah banyaknya dokumen yang mengandung kata tertentu. [16]. Pada penelitian ini, proses *TF-IDF* akan menggunakan *library* dari *sklearn*, yaitu sebuah *library Python* yang populer untuk pemrosesan data dan pembelajaran mesin. Pada perhitungan *TF-IDF* menggunakan *library sklearn* terdapat perbedaan dalam perumusan nya, bobot *TF-IDF* dihitung berdasarkan rumus persamaan (2.1) berikut:

$$W_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right) + 1 \tag{2.1}$$

Keterangan dari persamaan berikut ini adalah $tf_{i,j}$ = banyaknya kata-i pada dokumen, $N =$ adalah total jumlah dokumen, dan $df_i =$ adalah banyaknya dokumen yang mengandung kata ke-i. Penambahan nilai 1 dalam rumus *TF-IDF sklearn* bertujuan untuk menghindari pembagian dengan nol. Jika sebuah kata kunci tidak muncul dalam dokumen, frekuensi kemunculan *(TF)* akan menjadi nol. Dalam kasus ini, jika nilai *IDF* juga nol, maka hasil perhitungan *TF-IDF* akan menghasilkan *NaN (Not a Number)* atau tidak terdefinisi. Dengan menambahkan nilai 1 pada rumus, dapat memastikan bahwa tidak ada pembagian dengan nol yang terjadi. Sehingga, bahkan jika sebuah kata kunci tidak muncul dalam dokumen, skor *TF-IDF* akan tetap terdefinisi dan memiliki nilai yang berarti.

2.7 *Modified K-Nearest Neighbor (MKNN)*

Modified K-Nearest Neighbor (MKNN) merupakan pengembangan dari metode *KNN* dengan penambahan beberapa proses yaitu, perhitungan nilai

validitas dan perhitungan bobot *(weight voting)*. Langkah-langkah pada algoritma *Modified K-Nearest Neighbor* yaitu [17]:

1. Menentukan nilai *k*
2. Jarak antar data latih menggunakan rumus *Euclidean Distance* pada persamaan (2.2) berikut :

$$d\ (P,Q) = \sqrt{\sum_{i=1}^{n} (P_i - Q_i)^2} \tag{2.2}$$

   Dimana *n* merupakan jumlah data latih, *P* merupakan masukkan data ke-*i* dari data uji, dan *Q* merupakan masukkan data ke-*i* dari data latih Proses perhitungan dilakukan untuk semua data latih. Kemudian hasil perhitungan diurutkan secara *ascending* dengan memilih tetangga terdekat sesuai nilai *k*
3. Validitas data *training*, merupakan proses perhitungan jumlah titik dengan label yang sama pada semua data latih. Setiap data memiliki validitas yang bergantung pada tetangga terdekatnya. Rumus yang digunakan untuk menghitung validitas pada data latih yaitu:

$$Validitas_{(i)} = \frac{1}{k} \sum_{i-1}^{k} S(lbl_{(x)}\ , lbl\ Ni_{(x)}) \tag{2.3}$$

   Keterangan dari persamaan di atas adalah $k =$ jumlah titik terdekat, $lbl\ (x) =$ kelas x dan $lbl\ Ni\ (x) =$ label kelas titik terdekat x, Fungsi dari *S* digunakan menghitung kesamaan antara titik *a* dan data ke- *b* pada tetangga terdekat dengan menggunakan persamaan (2.4) berikut :

$$S\ a,b = \begin{Bmatrix} 1\ a = b \\ 0\ a \neq b \end{Bmatrix} \tag{2.4}$$

   dimana *a* merupakan kelas *a* pada data *training* dan *b* merupakan kelas selain *a* pada data *training.*

4. Jarak antara data uji dengan data latih menggunakan rumus *Euclidean Distance.* Perhitungan dilakukan untuk selurah data latih.
5. *Weight voting* (pembobotan)

   Perhitungan ini menggunakan *k* tetangga terdekat yang merupakan variasi metode *K-Nearest Neighbor*. Selanjutnya dilakukan validitas dari setiap data *training* yang akan dikalikan dengan *weight voting* berdasarkan jarak pada setiap tetangganya. Rumus *weight voting* menggunakan persamaan (2.5) berikut:

$$W_{(i)} = Validitas_{(i)} \times \frac{1}{d_e + \alpha} \tag{2.5}$$

   Keterangan dari persamaan di atas adalah $W_{(i)}$ =merupakan perhitungan weight voting ke-i , $Validitas\ (i) =$ merupakan nilai validitas ke-i, $d_e =$ merupakan jarak *euclidean*, dan $alfa\ (\alpha) =$ merupakan nilai *regulator smoothing* (pemulusan).
6. Menentukan kelas dari data uji dengan memilih bobot terbesar sesuai dengan nilai *k.* Hasil perhitungan *weight voting* yang telah didapatkan, selanjutnya diurutkan secara descending untuk mendapatkan klasifikasi kelas.

Dengan adanya tambahan proses perhitungan nilai validitas dapat melatih sampel yang menyebabkan lebih banyak informasi tentang situasi data pelatihan di ruang fitur dan dapat memperhitungkan nilai stabilitas dan kekokohan data terkait dengan tetangganya, serta dengan proses *weight voting* memiliki efek memberikan kepentingan yang lebih besar pada sampel referensi yang memiliki validitas dan kedekatan yang lebih besar dengan data uji [18].

2.8 Jarak *Euclidean*

Pengukuran jarak pada antar data digunakan untuk menentukan kemiripan antar data. Salah satu cara untuk menghitung jarak skalar jauh dan dekatnya pada

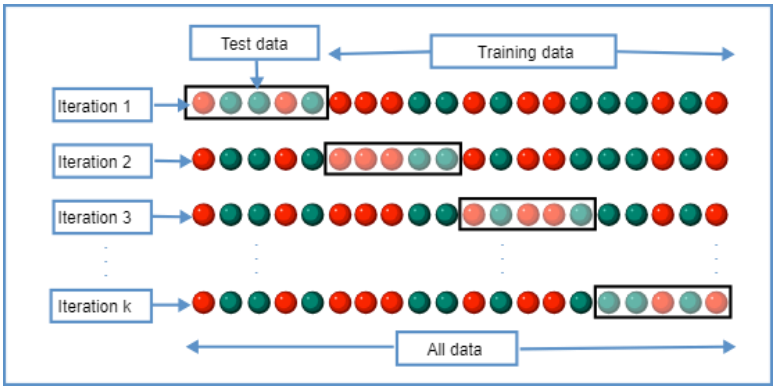
suatu data adalah *Euclidean Distance*. *Euclidean distance* adalah perhitungan jarak dari dua buah titik dalam *Euclidean space* [19]. Jarak *Euclidean* digunakan untuk menghitung jarak antara dua vektor yang berfungsi menguji ukuran yang bisa digunakan sebagai interpretasi kedekatan jarak antara dua objek yang direpresentasikan dalam persamaan. Tingkat kemiripan ditentukan dari nilai terdekat dengan menggunakan rumus persamaan (2.6) berikut:

$$D(a,b) = \sqrt{\sum_{k=1}^{d} (\, a_k \, - \, b_k \,)^2} \tag{2.6}$$

D (a, b) adalah jarak skalar antara *a* dan *b*, *d* adalah ukuran dimensi, *a* adalah ciri pada data *a*, dan *b* adalah ciri data pada data *b*. Semakin besar nilai *euclidean distance* akan menunjukan semakin besar perbedaan data nya, dan sebaliknya jika semakin kecil nilainya berarti makin dekat kemiripan data tersebut.

2.9 *Cross Validation*

Cross validation merupakan salah satu metode dalam melakukan validasi model terbaik. Teknik ini akan menguji keefektifan dari model yang dibentuk dengan melakukan penyusunan ulang (*resampling*) pada data untuk membaginya menjadi dua bagian yaitu data *training* dan data *testing* [20]. Salah satu teknik dari *cross validation* adalah *K-fold cross validation*, yang mana memecah data menjadi *K* bagian set data dengan ukuran yang sama. Secara keseluruhan, *K-fold Cross Validation* membantu membuat proses evaluasi dan pengembangan model lebih stabil, akurat, dan andal. Ini membantu mendapatkan pemahaman yang lebih baik tentang bagaimana model akan berperilaku pada data yang tidak dilihat sebelumnya. Alur kerja *K fold-cross validation* diilustrasikan pada Gambar 2.1.



Gambar 2. 1 Simulasi *K-fold Cross Validation*

2.10 *Confusion Matrix*

Confusion matrix adalah metode praktis yang digunakan untuk menentukan kinerja model pengelompokan data uji dengan nilai sebenarnya yang diketahui [21]. *Confusion matrix* berbentuk seperti tabel yang mencakup matriks dua dimensi, dimana dimensi yang pertama diisi oleh kelas sebenarnya dari suatu objek dan di dimensi yang lain diisi oleh kelas yang dihasilkan oleh klasifikasi. Berikut adalah *confusion matrix* yang biasanya digunakan secara umum:

Tabel 2. 1 *Confusion Matrix*

		Kelas Prediksi	
		Positif	Negatif
Kelas Aktual	Positif	TP	FN
	Negatif	FP	TN

Keterangan pada tabel tersebut adalah *TP* = Jumlah data yang bernilai Positif dan diprediksi benar sebagai Positif, *FP* = Jumlah data yang bernilai Negatif tetapi diprediksi sebagai Positif, *FN* = Jumlah data yang bernilai Positif tetapi diprediksi sebagai Negatif dan *TN* = Jumlah data yang bernilai Negatif dan diprediksi benar sebagai Negatif. *Confusion matrix* merepresentasikan tingkat akurasi dari proses klasifikasi yang telah dilakukan. Tingkat akurasi menunjukkan

proporsi jumlah prediksi benar. Rumus pengukuran akurasi dapat dilihat pada persamaan (2.7) berikut :

$$Accuray = \frac{TP + TN}{TP + FP + TN + FN} \quad (2.7)$$

2.11 *Literature Review*

Bagian penting yang dibutuhkan dalam sebuah penelitian adalah adanya *literature review* yang kemudian digunakan sebagai dasar dari penyusunan laporan penelitian tersebut. *Literature review* juga dianggap penting menjadi landasan mengenai alasan peneliti memutuskan untuk memilih tema maupun judul tertentu. *Literature Review* juga hanya dapat dianggap sebagai pondasi lingkup pekerjaan yang akan dilaporkan. Berikut *Literature Review* yang telah disusun:

2.11.1 *Literature Review* Jurnal 1

Berikut ini hasil *literature review* dari referensi penelitian ke-1 [6]:

Tabel 2. 2 *Literature Review* Jurnal 1

Judul Peneletian	OPTIMASI TEKNIK KLASIFIKASI MODIFIED K NEAREST NEIGHBOR MENGGUNAKAN ALGORITMA GENETIKA
Penulis	Siti Mutrofin, Abidatul Izzah, Arrie Kurniawardhani, dan Mukhamad Masrur
Tahun Terbit	2014
Metode Penelitian	K Nearest Neighbor (KNN), Modified k Nearest Neighbor (MKNN), dan Genetic Modified k Nearest Neighbor (GMKNN)
Hasil Penelitian	Algoritma KNN, MKNN dan GMKNN memiliki kinerja yang sama baiknya, dalam melakukan klasifikasi data Iris dengan hasil akurasi 100%.

2.11.2 *Literature Review* Jurnal 2

Berikut ini hasil *literature review* dari referensi penelitian ke-2 [7]:

Tabel 2. 3 *Literature Review* Jurnal 2

Judul Peneletian	Comparative Analysis of K-Nearest Neighbor and Modified K-Nearest Neighbor Algorithm for Data Classification
Penulis	Okfalisa, Mustakim, Ikbal Gazalba, Nurul Gayatri Indah Reza
Tahun Terbit	2017
Metode Penelitian	Metode K-Nearesta Neighbor dan Modified K-Nearest Neighbor
Hasil Penelitian	Hasil akurasi KNN tertinggi adalah 94,95% dengan akurasi rata-rata selama pengujian adalah 93,94% sedangkan akurasi tertinggi MKNN adalah 99,51% dan akurasi rata-rata selama pengujian adalah 99,20%. sehingga dapat dikatakan kemampuan algoritma MKNN lebih baik dari segi akurasi dengan selisih akurasi sebesar 5-7%.

2.11.3 *Literature Review* Jurnal 3

Berikut ini hasil *literature review* dari referensi penelitian ke-3 [8]:

Tabel 2. 4 *Literature Review* Jurnal 3

Judul Peneletian	Analisis Sentimen Terhadap Ulasan Pengguna MRT Jakarta Menggunakan Information Gain dan Modified K-Nearest Neighbor
Penulis	Adella Ayu Paramitha, Indriati, Yuita Arum Sari
Tahun Terbit	2020

Metode Penelitian	Metode klasifikasi Modified Knearest Neighbor (MKNN) dan seleksi fitur Information Gain
Hasil Penelitian	Berdasarkan hasil evaluasi, didapatkan nilai accuracy sebesar 0,86769 dan f-measure sebesar 0,86265 dengan parameter k=3 dan threshold=25%.

2.11.4 *Literature Review* Jurnal 4

Berikut ini hasil *literature review* dari referensi penelitian ke-4 [10]:

Tabel 2. 5 *Literature Review* Jurnal 4

Judul Peneletian	Analisis Sentimen Terhadap Ulasan Pengguna MRT Jakarta Menggunakan Information Gain dan Modified K-Nearest Neighbor
Penulis	Adella Ayu Paramitha, Indriati, Yuita Arum Sari
Tahun Terbit	2020
Metode Penelitian	Metode klasifikasi Modified Knearest Neighbor (MKNN) dan seleksi fitur Information Gain
Hasil Penelitian	Berdasarkan hasil evaluasi, didapatkan nilai accuracy sebesar 0,86769 dan f-measure sebesar 0,86265 dengan parameter k=3 dan threshold=25%.

2.11.5 *Literature Review* Jurnal 5

Berikut ini hasil *literature review* dari referensi penelitian ke-5 [17]:

Tabel 2. 6 *Literature Review* Jurnal 5

Judul Peneletian	Identifikasi Penyakit Diabetes Mellitus Menggunakan Metode Modified K-Nearest Neighbor (MKNN)
Penulis	Silvia Ikmalia Fernanda, Dian Eka Ratnawati, Putra Pandu Adikara
Tahun Terbit	2017

Metode Penelitian	Menggunakan metode Modified K-Nearest Neighbor (MKNN)
Hasil Penelitian	Berdasarkan pengujian yang telah dilakukan memperoleh hasil akurasi terbaik 93,33% dengan error rate 6,67%. Berdasarkan hasil tersebut, sistem yang menggunakan metode Modified K-Nearest Neighbor (MKNN) dapat diimplementasikan dalam kehidupan sehari-hari.

2.11.6 *Literature Review* Jurnal 6

Berikut ini hasil *literature review* dari referensi penelitian ke-6 [18]:

Tabel 2. 7 *Literature Review* Jurnal 6

Judul Peneletian	MKNN: Modified K-Nearest Neighbor
Penulis	Hamid Parvin, Hosein Alizadeh dan Behrouz Minaei-Bidgoli
Tahun Terbit	2008
Metode Penelitian	Menggunakan KNN dan MKNN
Hasil Penelitian	Evaluasi metode pada lima tugas benchmark: masalah Wine, Isodata, dan tiga Monk mengkonfirmasi klaim ini. Karena outlier biasanya mendapatkan nilai validitas yang rendah, hal itu sangat menghasilkan kekokohan metode MKNN menghadapi outlier.