BAB 2
LANDASAN TEORI

2.1 Whoosh Kereta Cepat

Aplikasi Whoosh - Kereta Cepat adalah aplikasi mobile yang memudahkan pengguna dalam mendapatkan informasi terbaru terkait jadwal dan layanan kereta cepat. Aplikasi ini menawarkan berbagai fitur, seperti pembaruan jadwal secara *real-time*, pemesanan tiket online yang mudah, pemberitahuan perubahan, serta ulasan dari pengguna lainnya [10]. Fitur-fitur ini memungkinkan pengguna untuk merencanakan perjalanan dengan lebih efisien dan nyaman. Selain itu, aplikasi ini menyediakan peta rute dan informasi stasiun yang terperinci untuk membantu navigasi. Aplikasi Whoosh tidak hanya mempermudah pengalaman perjalanan, tetapi juga memberikan tempat bagi pengguna untuk memberikan ulasan dan opini, yang pada akhirnya meningkatkan interaksi serta kualitas layanan kereta cepat secara keseluruhan.

2.2 Analisis Sentimen

Menurut penelitian yang dilakukan oleh Al-Ayyoub dkk [11], analisis sentimen adalah sebuah ranah studi yang terus berkembang, yang berada di titik temu berbagai disiplin ilmu, seperti Penambangan Data (*Data Mining*), Pemrosesan Bahasa Alami (*Natural Language Processing*), dan Pembelajaran Mesin (*Machine Learning*). Tujuan utama dari analisis ini adalah untuk mengidentifikasi dan mengekstrak sentimen atau emosi yang terkandung dalam kalimat berdasarkan konteksnya. Para peneliti dalam studi ini berupaya menentukan apakah sebuah teks memiliki sentimen positif, negatif, atau netral, serta mengukur kekuatan ekspresi tersebut.

Kemudian, menurut Vinodhini dan Chandrasekaran [12], analisis sentimen adalah bidang ilmu yang bersifat interdisipliner, di mana pendekatannya melibatkan penggunaan perspektif dari berbagai disiplin ilmu yang relevan secara bersamaan dan terintegrasi dalam pemecahan masalah.

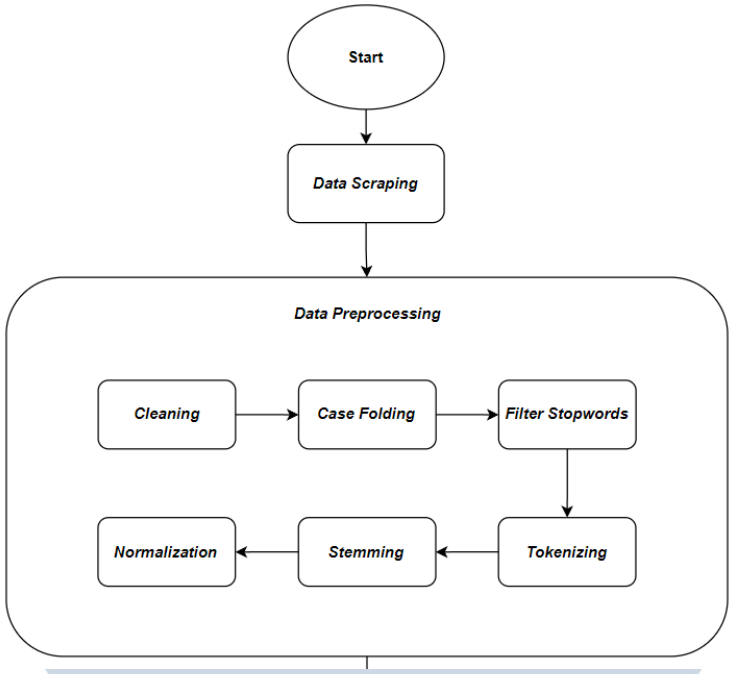
2.3 Text Mining

Menurut A. R. T. Ririd, dkk [13], *text mining* adalah metode untuk menghasilkan informasi baru yang lebih spesifik dari sejumlah dokumen. Proses ini melibatkan ekstraksi inti dari dokumen teks tidak terstruktur dengan menggunakan teknik preprocessing teks seperti pencarian, ekstraksi data, dan kategorisasi. Teknik ini sangat berguna untuk memahami makna dalam dokumen teks yang kompleks. *Text mining* diperlukan untuk menggali informasi tersembunyi atau tidak terstruktur dalam teks, terutama ketika analisis manual tidak efisien. Dengan teknik ini, informasi berharga dapat diekstraksi, pola diidentifikasi, dan pemahaman yang lebih dalam diperoleh dari data teks.

2.4 Text-Preprocessing

Text-Preprocessing adalah tahap awal dalam pemrosesan data, yang berfokus pada transformasi data mentah yang belum terorganisir menjadi bentuk yang lebih terstruktur dan siap untuk dianalisis lebih lanjut [14]. Langkah ini bertujuan untuk menghasilkan format data yang optimal untuk analisis, serta mengurangi kemungkinan kesalahan dalam ekstraksi ciri atau atribut dari teks ulasan. Proses ini berpengaruh besar terhadap hasil yang diperoleh dari algoritma yang digunakan. Dengan mengurangi noise dalam data, algoritma akan berfungsi lebih efektif [15]. Dalam text-preprocessing, terdapat beberapa tahapan yang dapat dilakukan untuk mentransformasi data, antara lain:

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA



Gambar 2.1. Alur Text Preprocessing

a. ***Cleaning*** Proses ini untuk membersihkan dokumen dari kata yang tidak relevan [16]. Tujuannya adalah agar dokumen yang akan diolah menjadi lebih bersih dan relevan.

b. ***Case Folding*** teknik yang digunakan dalam pemrosesan teks untuk menyamakan format huruf, baik menjadi semua huruf kecil atau semua huruf besar, guna memastikan konsistensi dalam analisis data dan menghindari perbedaan interpretasi antara variasi huruf. [17].

c. ***Filter Stopwords*** merupakan langkah dalam analisis teks yang berfungsi untuk menghapus kata-kata umum dan tidak relevan, seperti ’yang,’ ’adalah,’ dan ’atau,’ yang tidak memberikan nilai tambah terhadap pemahaman keseluruhan dokumen. [18].

d. ***Tokenizing*** merupakan tahap dalam pemrosesan bahasa alami yang mencakup pemecahan teks menjadi bagian-bagian lebih kecil yang disebut token [19]. Token-token ini bisa berupa kata, frasa, atau simbol yang lebih kecil, sehingga membantu komputer dalam memahami dan menganalisis struktur serta makna dari teks tersebut.

e. ***Stemming*** proses dalam pemrosesan bahasa alami yang bertujuan untuk mengurangi kata-kata ke bentuk dasarnya dengan menghapus imbuhan atau akhiran yang ada [20]. Misalnya, kata ”berlari,” ”lari,” dan ”pelari” akan direduksi menjadi kata dasar ”lari”.

f. ***Normalization*** proses untuk mengoreksi kesalahan dalam penulisan kata, seperti kesalahan ejaan, sehingga kata-kata yang memiliki makna dasarnya [21].

2.5 Oversampling SMOTE

Ketidakseimbangan kelas dalam masalah klasifikasi sering kali disebabkan oleh distribusi yang tidak merata dari label kelas target, yang menjadi tantangan selama pelatihan model. Dataset yang tidak seimbang cenderung melatih pada kelas mayoritas, menyebabkan prediksi yang bias dan potensi pengabaian kelas minoritas sebagai *noise* atau *outlier*. Masalah ini secara signifikan mempengaruhi kinerja klasifikasi. Oleh karena itu, penyelesaian masalah ketidakseimbangan data merupakan langkah krusial dalam klasifikasi. Penelitian ini menggunakan teknik SMOTE (*Synthetic Minority Over-sampling Technique*), teknik oversampling yang banyak digunakan dalam data mining, yang berfokus pada ”*feature space*”. Menghasilkan instansi sintetis dengan interpolasi *K-nearest neighbors* dari sampel minoritas yang dipilih secara acak [22].

Proses SMOTE dimulai dengan mengidentifikasi kelas minoritas dalam data. Setelah itu, algoritma menentukan *K nearest neighbors* dengan menghitung jarak *Euclidean* antara data minoritas. Data sintetis kemudian dihasilkan di sepanjang garis yang menghubungkan titik-titik minoritas dengan *K nearest neighbors*. Titik acak baru dihasilkan di antara titik-titik ini, menghasilkan sampel tambahan untuk kelas minoritas [23]. Rumus yang digunakan untuk menciptakan data sintetis pada dataset yang tidak seimbang menggunakan metode SMOTE dapat dilihat pada persamaan berikut.

$$X_{\text{synthetic}} = X_i + \lambda \times (X_{nn} - X_i) \tag{2.1}$$

Sedangkan jarak antara *K nearest neighbors* untuk data yang memiliki skala numerik, perhitungan dilakukan dengan menggunakan jarak Euclidean, yang dapat dilihat pada persamaan berikut.

$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2} \tag{2.2}$$

2.6 Pembobotan TF-IDF

Pembobotan TF-IDF (Term Frequency-Inverse Document Frequency) adalah metode yang digunakan untuk menilai seberapa signifikan suatu kata dalam sebuah dokumen atau kumpulan teks. Teknik ini menggabungkan dua konsep utama, yaitu Term Frequency (TF) dan Inverse Document Frequency (IDF) [24]. Term Frequency (TF) menghitung seberapa sering suatu kata muncul dalam dokumen tertentu, sehingga memberikan bobot lebih pada kata-kata yang sering ditemukan, yang menunjukkan relevansi yang lebih tinggi dalam konteks dokumen tersebut. Sebaliknya, Inverse Document Frequency (IDF) mengukur frekuensi kemunculan kata tersebut di seluruh dokumen dalam korpus yang lebih besar. Tujuan dari prinsip ini adalah memberikan bobot lebih kepada kata-kata yang jarang muncul, karena kata-kata tersebut umumnya memiliki peran yang lebih penting dalam membedakan atau mengidentifikasi dokumen [25]. Rumus untuk menghitung TF-IDF dapat dinyatakan sebagai berikut:

$$\text{TF-IDF} = \text{TF} \times \text{IDF} \tag{2.3}$$

Untuk menghitung TF-IDF, perlu dilakukan perhitungan dua matriks: satu untuk mengukur seberapa sering sebuah kata muncul dalam semua dokumen (IDF), dan satu lagi untuk mengukur seberapa sering kata tersebut muncul dalam setiap dokumen (TF).

$$\text{TF}(t,d) = \frac{\text{Jumlah kali kemunculan term } t \text{ dalam dokumen } d}{\text{Total jumlah term dalam dokumen } d} \tag{2.4}$$

$$\text{IDF}(t,D) = \log\left(\frac{N}{\text{df}(t)}\right) \tag{2.5}$$

Keterangan:

t : Term atau kata kunci di seluruh korpus.

D : Seluruh koleksi dokumen atau korpus.

N : Jumlah total dokumen dalam korpus D.

$df(t)$: Jumlah dokumen dalam korpus D yang mengandung term t.

2.7 Machine Learning

Machine learning adalah salah satu sub-bidang dari kecerdasan buatan yang memungkinkan komputer untuk mengembangkan kemampuan belajar dan beroperasi dengan cara yang menyerupai tindakan manusia. Dengan *machine learning*, komputer dapat mengembangkan kemampuannya melalui pengalaman dan membuat keputusan secara otomatis tanpa perlu diprogram ulang secara spesifik oleh manusia. Ini memungkinkan sistem untuk menyesuaikan diri dengan perubahan secara dinamis. Proses *machine learning* melibatkan analisis *big data* untuk mengidentifikasi pola-pola tertentu [26]. Data ini digunakan untuk melatih algoritma, sehingga mesin dapat memberikan analisis yang lebih akurat. Dalam *machine learning*, ada dua jenis data yang digunakan *training data* untuk melatih algoritma, dan *testing data* untuk mengevaluasi kinerja algoritma ketika menghadapi data baru yang belum pernah dilihat sebelumnya.

Dalam konteks teknik pembelajaran, machine learning dibagi menjadi beberapa kategori, termasuk supervised learning dan *unsupervised learning* [27]. *Supervised learning* memanfaatkan data yang sudah dilabeli untuk mengajarkan algoritma bagaimana membuat prediksi. Sementara itu, *unsupervised learning* bekerja dengan data yang tidak dilabeli, berusaha mengelompokkan data berdasarkan pola atau karakteristik yang ditemukan selama proses analisis. Dalam bidang *machine leanring*, terdapat berbagai metode klasifikasi yang termasuk dalam konsep *supervised learning*. Konsep ini digunakan untuk memprediksi kategori data berdasarkan model yang dibentuk dari data latihan yang telah diberi label kelas. Dalam konteks penelitian ini, fokus diberikan pada penggunaan algoritma SVM sebagai algoritma klasifikasi.

2.8 Support Vector Machine (SVM)

Support Vector Machine (SVM) merupakan sebuah algoritma yang digunakan dalam prediksi untuk klasifikasi dan regresi, termasuk dalam pembelajaran supervisi (*supervised learning*). Pertama kali diperkenalkan oleh Vapnik pada tahun 1992 [28]. SVM bertujuan untuk menemukan *hyperplane* pemisah yang optimal dengan margin terbesar untuk memisahkan dua kelas data secara efisien. Algoritma ini sangat efektif pada dataset berdimensi tinggi dengan menggunakan teknik *kernel* untuk mentransformasi ruang fitur [29]. SVM bekerja dengan memanfaatkan titik-titik data yang krusial(*support vector*) untuk membentuk model, sehingga mengurangi kompleksitas namun tetap menjaga akurasi. Akurasi model SVM sangat dipengaruhi oleh pemilihan fungsi *kernel* dan penyesuaian *parameter*. SVM dibagi menjadi SVM *Linear* untuk data yang dapat dipisahkan secara *linear* dengan *hyperplane*, dan SVM *Non-Linear* yang menggunakan *kernel* untuk menangani data yang tidak dapat dipisahkan secara *linear* [30].

Awalnya digunakan untuk mengklasifikasi data numerik, SVM telah terbukti cepat dan efektif dalam menangani masalah data teks. Data teks cocok untuk klasifikasi dengan algoritma SVM karena sifatnya yang berdimensi tinggi, di mana banyak fitur yang tidak relevan tetapi sering berkorelasi dan umumnya terstruktur dalam kategori yang terpisah secara *linear*. Inti dari prinsip SVM adalah menemukan *hyperplane* yang paling efektif untuk memisahkan dua kelas dalam ruang input. Hyperplane pemisah yang optimal ini ditentukan dengan mengevaluasi margin, yaitu jarak antara *hyperplane* dan titik data terdekat dari setiap kelas, yang dikenal sebagai support vector. Proses pelatihan SVM berfokus pada penentuan posisi terbaik dari *hyperplane* tersebut. Berikut adalah persamaan model SVM yang digunakan untuk menghasilkan *hyperplane*:

$$y = \mathbf{w} \cdot \mathbf{x} + b \tag{2.6}$$

Pertidaksamaan ini menunjukkan bahwa untuk sebuah titik data, kurang dari atau sama dengan -1, maka titik data tersebut dianggap sebagai sampel negatif (kelas -1).

$$\mathbf{w} \cdot \mathbf{x} + b \leq -1 \tag{2.7}$$

Sedangkan pertidaksamaan ini menunjukkan bahwa untuk sebuah titik data, besar dari atau sama dengan 1, maka titik data tersebut dianggap sebagai sampel

positif (kelas 1).

$$\mathbf{w} \cdot \mathbf{x} + b \geq 1 \tag{2.8}$$

Jika $\mathbf{w} \cdot \mathbf{x} + b = +1$ merupakan *hyperplane* pendukung untuk kelas positif, sedangkan $\mathbf{w} \cdot \mathbf{x} + b = -1$ merupakan *hyperplane* pendukung untuk kelas negatif, maka nilai margin (d) merupakan *hyperplane* pendukung untuk kelas negatif, maka nilai margin.

$$d = \frac{(1-b) - (-1-b)}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \tag{2.9}$$

Keterangan:

x : vektor fitur input

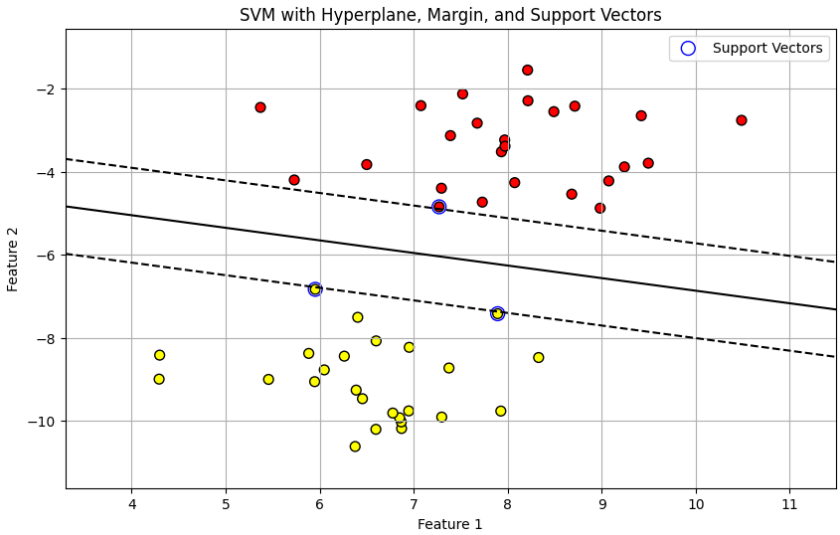
w : vektor bobot(weights) yang memiliki dimensi yang sama dengan x

b : nilai bias atau error

y : nilai target dari himpunan data x, prediksi (+1 atau -1)

Pada dasarnya, SVM merupakan sebuah pengklasifikasi *linear*. Namun, SVM dapat dikembangkan menjadi pengklasifikasi *non-linear* menggunakan konsep *kernel trick* pada ruang berdimensi lebih tinggi. Gambar 2.2 menunjukkan bagaimana algoritma ini memisahkan dua kelas data menggunakan *hyperplane*. Dalam grafik, titik-titik data dari dua kelas ditampilkan, dengan *hyperplane* sebagai garis hitam yang membagi ruang fitur menjadi dua bagian, masing-masing mewakili satu kelas. Di sekitar *hyperplane*, terdapat margin, yang merupakan jarak maksimum antara *hyperplane* dan titik data terdekat dari setiap kelas, ditandai dengan garis putus-putus. *Support vectors*, yang merupakan titik data yang paling dekat dengan *hyperplane*, ditandai dengan lingkaran biru, dan titik-titik ini sangat penting karena mereka menentukan posisi *hyperplane*. SVM berusaha untuk memaksimalkan *margin* ini untuk mencapai klasifikasi yang lebih baik dan generalisasi pada data yang belum pernah dilihat sebelumnya.

UNIVERSITAS MULTIMEDIA NUSANTARA



Gambar 2.2. Illustrasi SVM

Terdapat beberapa fungsi *kernel* yang sering digunakan dalam SVM, di antaranya:

a. *Kernel Linear*

Kernel linear merupakan salah satu jenis fungsi kernel yang paling sederhana dan umum digunakan untuk dataset yang dapat dipisahkan dengan cara linear. Metode ini terbukti sangat efisien, terutama ketika jumlah fitur dalam dataset sangat banyak, karena pemetaan ke ruang dimensi yang lebih tinggi tidak memberikan peningkatan kinerja yang signifikan, khususnya dalam konteks klasifikasi teks.

$$k(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i} \cdot \mathbf{x_j} \tag{2.10}$$

b. *Kernel Polynomial*

Kernel polinomial adalah sebuah fungsi yang melibatkan variabel dan koefisien yang membentuk polinomial, dan biasanya digunakan ketika data pelatihan telah dinormalisasi.

$$k(\mathbf{x_i}, \mathbf{x_j}) = ((\mathbf{x_i} \cdot \mathbf{x_j}) + 1)^d \tag{2.11}$$

c. *Kernel Radial Bias Function* (RBF)

Kernel RBF, atau dikenal juga sebagai Gaussian, digunakan untuk menangani masalah yang tidak dapat diselesaikan secara linear. Selama prosesnya, RBF memiliki keunggulan dalam tingkat akurasi pelatihan dan prediksi yang tinggi. Kernel RBF memiliki dua parameter utama: *Gamma* dan *Cost* [31]. Paramter *Cost* atau C berfungsi sebagai alat untuk mengoptimalkan SVM agar dapat menghindari kesalahan dalam mengklasifikasikan setiap sampel dalam dataset pelatihan. Sementara itu *Gamma* menentukan seberapah jauh pengaruh satu sampel dataset pelatihan, dengan nilai rendah dan tinggi memiliki efek yang berbeda pada hasil.

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2) \tag{2.12}$$

Keterangan:

K : fungsi kernel
$\mathbf{x_i}, \mathbf{x_j}$: vektor dari data set
γ : nilai bias atau error
exp : basis dari logaritma alami
d : pangkat polynomial

Dalam penelitian [32], langkah-langkah yang digunakan dalam metode SVM adalah sebagai berikut:

1. **Menghitung nilai TF-IDF**: menilai relevansi setiap kata dalam dokumen dengan memberikan bobot berdasarkan frekuensi kemunculan kata tersebut serta seberapa jarang kata itu muncul dalam seluruh koleksi dokumen..
2. **Mengkalkulasi matriks kernel**: Penghitungan matriks kernel dilakukan untuk menunjukkan hubungan antara data. Beberapa fungsi kernel yang umum digunakan meliputi linear, polynomial, dan radial basis function (RBF). Dengan menggunakan fungsi-fungsi ini, SVM dapat menangani data yang bersifat non-linear dengan cara memproyeksikan data ke ruang berdimensi lebih tinggi.
3. **Memperbarui nilai** *y*: Nilai *y* inisial diperbarui dengan membagi nilai tersebut dengan nilai maksimum dari matriks kernel. Ini membantu dalam

skala data agar lebih mudah dalam proses optimisasi.

4. **Menghitung nilai *Lagrange Multiplier* ($\delta\alpha_i$)**: *Lagrange dual Objective* dihitung dengan rumus:

$$\text{maximize} \quad L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{2.13}$$

5. **Menghitung nilai *error rate* (E_i)**: Jika diperlukan, nilai *error rate* dapat dihitung sebagai:

$$E_i = \sum_{j=1}^{n} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{2.14}$$

   Di mana E_i adalah kesalahan untuk data ke-i. Namun, langkah ini tidak selalu diperlukan tergantung pada implementasi SVM yang digunakan.

6. **Menghitung nilai bias (b)**: Bias dihitung menggunakan support vector (titik untuk mana $\alpha_i > 0$):

$$b = y_k - \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k) \tag{2.15}$$

   Di mana $\mathbf{x}_k$ adalah support vector yang dipilih.

7. **Menghitung vektor bobot (w)**: Vektor bobot dihitung dengan rumus:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{2.16}$$

   Keterangan:

   $\mathbf{w}$ = Vektor bobot
   α_i = Koefisien untuk data ke-i
   y_i = Kelas data ke-i
   $\mathbf{x}_i$ = Data ke-i

8. **Fungsi klasifikasi**: Setelah memperoleh nilai bias b, langkah selanjutnya adalah menghitung kernel dan bobot dari data uji. Ini digunakan dalam fungsi klasifikasi untuk menentukan kelas dari data uji tersebut. Nilai $f(x)$ dapat

dihitung dengan persamaan berikut:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b\right) \tag{2.17}$$

Pemilihan algoritma SVM didasarkan pada penelitian sebelumnya yang mengungkapkan keunggulannya dalam analisis sentimen. SVM terkenal kuat dalam menangani pemisahan kelas yang kompleks dan efektif dalam memodelkan data berdimensi tinggi. Studi terdahulu, seperti yang dilaporkan dalam penelitian [33], menunjukkan bahwa SVM memberikan hasil klasifikasi yang baik dalam analisis sentimen. Selain itu, SVM terbukti mampu mengatasi tantangan pemodelan dengan dataset yang relatif kecil, seperti yang diamati dalam penelitian sebelumnya [34]. Oleh karena itu, SVM dipilih sebagai algoritma utama dalam penelitian ini.

Penelitian ini menggunakan SVM dengan dua jenis kernel: linear dan non-linear. SVM linear menggunakan kernel linear yang efektif dalam klasifikasi teks, namun memiliki keterbatasan dalam menangani data yang mengandung pola kompleks seperti makna implisit atau hubungan antar-kata yang rumit. Oleh karena itu, penelitian ini juga memasukkan SVM non-linear dengan kernel polynomial dan RBF untuk mengeksplorasi perbedaan performa model. Tujuan utamanya adalah mencari model yang paling optimal dalam mengatasi tantangan klasifikasi teks yang kompleks, termasuk memahami makna yang tersembunyi dan relasi yang kompleks antar-kata dalam teks.

2.9 K-Fold Cross Validation

Cross Validation merupakan teknik *resampling* yang umum digunakan untuk memvalidasi model. Teknik ini cocok digunakan pada kasus di mana jumlah sampel dataset terbatas. Metode Validasi k-fold melibatkan pembagian data (D) menjadi k subset (D1, D2, ..., Dk) dengan ukuran yang sama. Selama setiap iterasi, satu subset digunakan untuk pengujian sementara k subset lainnya digabungkan untuk pelatihan. Proses ini diulang sebanyak k kali, dengan setiap iterasi menggantikan subset sebagai set pengujian, untuk memastikan pelatihan dan pengujian yang komprehensif di seluruh dataset. Akurasi klasifikasi kemudian dihitung rata-ratanya dari semua iterasi k. Pilihan umum untuk nilai k termasuk 3, 5, 10, dan 20, tergantung pada kebutuhan spesifik dari penelitian [35]. Nilai akurasi dari *Cross validation* dapat dilakukan menggunakan persamaan sebagai

berikut [36].

$$\text{Akurasi} = \left(\frac{\text{Jumlah klasifikasi benar}}{\text{Total data uji}} \right) \times 100\% \tag{2.18}$$

2.10 Labeling

Tahap *Labeling* bertujuan untuk menetapkan kategori pada data, yaitu kategori positif dan negatif [37]. Pada langkah ini, pengelabelan dilakukan menggunakan *library BERT*, yang merupakan model representasi bahasa yang telah dilatih sebelumnya. BERT memanfaatkan teknik *deep learning* untuk mempelajari hubungan kontekstual antara kata-kata dalam teks. Model BERT telah menunjukkan keunggulan yang signifikan dalam berbagai tugas pemrosesan bahasa alami (NLP), termasuk dalam klasifikasi sentimen [38]. Dalam penelitian ini, karena ulasan data mayoritas menggunakan bahasa Indonesia dan *library BERT*, khususnya model *bert-base-multilingual-uncased-sentiment*, tidak mendukung penggunaan bahasa Indonesia secara langsung, maka ulasan akan diterjemahkan dari Bahasa Indonesia ke Bahasa Inggris terlebih dahulu sebelum dilakukan pelabelan.

2.11 Confusion matrix

Confusion matrix merupakan alat yang digunakan untuk mengevaluasi perbandingan antara hasil klasifikasi dan data yang sebenarnya. Matriks ini menyajikan akurasi dalam bentuk persentase, serta berfungsi sebagai dasar untuk menilai kinerja algoritma klasifikasi [39]. Tabel 2.1 di bawah ini menunjukkan tabel *confusion matrix*.

Tabel 2.1. Confusion Matrix

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Confusion matrix dan evaluasinya dapat dilakukan menggunakan empat parameter: TP, FN, FP, dan TN.

a. TP (*True Positive*) adalah jumlah data yang sebenarnya benar dan terdeteksi oleh model sebagai benar.

b. FN (*False Negative*) adalah jumlah data yang sebenarnya salah tetapi terdeteksi oleh model sebagai benar.

c. FP (*False Positive*) adalah jumlah data yang sebenarnya benar tetapi terdeteksi oleh model sebagai salah.

d. TN (*True Negative*) adalah jumlah data yang sebenarnya salah dan terdeteksi oleh model sebagai salah.

Setelah mendapatkan hasil dari *confusion matrix*, langkah berikutnya adalah menggunakan rumus-rumus untuk menghitung akurasi, *precision*, *recall*, dan *F-1 Score*.

Akurasi mengukur seberapa dekat prediksi dengan nilai sebenarnyam biasanya dipaparkan dalam bentuk persentase.

$$\text{Akurasi} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.19}$$

Precision menggambarkan sejauh mana data yang diminta cocok dengan hasil prediksi yang diberikan oleh model:

$$\text{Presisi} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.20}$$

Recall mengindikasikan seberapa baik model dapat mengenali kembali informasi yang diminta:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.21}$$

F-1 Score merupakan nilai penggabungan antara *precision* dan *recall*, disebut juga sebagai nilai rata-rata atau *mean*:

$$\text{F-1 Score} = 2 \times \frac{\text{Presisi} \times \text{Recall}}{\text{Presisi} + \text{Recall}} \tag{2.22}$$

UNIVERSITAS MULTIMEDIA NUSANTARA