BAB 2
LANDASAN TEORI

Penelitian berupa imeplementasi algoritma Naïve Bayes untuk analisis sentimen terhadap *rebranding* Twitter menjadi X disusun dengan menggunakan berbagai teori yang relevan. Penelitian ini berfokus pada implementasi algoritma Naïve Bayes dan menggunakan sejumlah teori yang menjadi landasan serta acuan bagi penulis. Adapun teori-teori yang digunakan berperan penting dalam mendukung pelaksanaan penelitian secara keseluruhan. Teori-teori yang digunakan untuk mendukung pelaksanaan penelitian ini sebagai berikut ini.

2.1 Analisis Sentimen

Analisis sentimen adalah pengelompokkan teks pada kalimat atau dokumen untuk menendukan pendapat yang disampaikan dalam kalimat tersebut bersifat positif atau negatif. Tujuan dari dilakukannya analisis sentimen ini adalah untuk menganalisis pendapat, sentimen, evaluasi, sikap, penilaian, dan emosi seseorang terhadap suatu topik, organisasi, layanan, individu, ataupun kegiatan tertentu lainnya [12].

2.2 Twitter

Twitter adalah media sosial berbasis teks pendek atau *microblogging* dan dikembangkan pada tahun 2006 oleh Jack Dorsey [13]. Twitter memiliki beberapa fitur utama, yaitu *tweet, retweet, hashtag, likes, quote tweet, dan reply*. Twitter juga memiliki fitur *trending topics* yang memudahkan penggunanya memperoleh informasi terbaru dan mengikuti topik yang sedang ramai dibicarakan.

2.3 Data Preprocessing

Preprocessing merupakan tahap untuk membersihkan dan menyesuaikan data sebelum dianalisis lebih lanjut. Tujuan utama dari *preprocessing* adalah memastikan bahwa teks yang digunakan tidak mengandung kata yang dapat mengubah makna asli yang dimaksud oleh penulis [14]. Proses *preprocessing* sangat berpengaruh terhadap tingkat akurasi model yang digunakan. Tahap *preprocessing* dilakukan dengan langkah-langkah berikut:

1. *Cleaning*

   Cleaning merupakan proses penghapusan karakter khusus dan tanda baca pada teks yang menjadi data. Tujuan dilakukan *cleaning* adalah untuk mengurangi *noise* pada data yang akan diproses lebih lanjut [6].

2. *Case Folding*

   Case Folding merupakan proses mengubah seluruh karakter pada dokumen menjadi *upper case* atau *lower case* [6]. Untuk penelitian ini, *case folding* yang dilakukan adalah mengubah seluruh karakter menjadi *lower case*.

3. *Tokenization*

   Tokenisasi (*tokenization*) merupakan proses pemisahan setiap kata yang berada pada kalimat untuk mengetahui asal munculnya kata [8].

4. *Slangword Replacement*

   Mengganti kata-kata gaul dan tidak baku menjadi bentuk baku berdasarkan kamus dari *slangword*. Proses mengganti kata gaul dan tidak baku ini dilakukan berdasarkan kamus slangword [15].

5. *Stopword Removal*

   Tahap ini adalah penghapusan kata-kata yang tidak memiliki pengaruh pada sentimen, seperti konjungsi, preposisi, dan artikel [7].

6. *Stemming*

   Stemming adalah proses mengubah kata yang memiliki imbuhan pada data menjadi kata dasar. Proses ini dilakukan dengan menghapus awalan atau akhiran pada suatu kata [16].

2.4 CountVectorizer

CountVectorizer adalah salah metode ekstraksi fitur sederhana yang banyak digunakan dalam pemrosesan teks. Teknik ini mengubah teks menjadi representasi vektor dengan cara menghitung frekuensi kemunculan setiap kata dalam dokumen [17]. Proses ini menghasilkan vektor numerik yang menjadi merepresentasikan jumlah kemunculan masing-masing kata pada setiap dokumen.

2.5 TF-IDF

TF-IDF (*Term Frequency - Inverse Document Frequency*) merupakan suatu metode pembobotan yang menggabungkan konsep *Term Frequency* dan *Document Frequency* [18]. Penggabungan metode ini bertujuan untuk mencari bobot dari suatu kata pada dokumen di setiap kategori dan mencari kata kunci yang mirip dengan kategori yang ada [19]. Rumus 2.1 berikut digunakan untuk menghitung nilai TF-IDF.

$$TF(t,d) - IDF(t) = TF(t,d) \times IDF(t) \tag{2.1}$$

2.5.1 TF (Term Frequency)

Nilai yang digunakan untuk mengukur seberapa sering sebuah kata muncul dalam suatu dokumen. Semakin tinggi frekuensi kemunculan kata, semakin besar nilai TF-nya [20]. Rumus 2.2 berikut ini digunakan untuk menghitung nilai TF

$$TF(t,d) = \frac{t}{d} \tag{2.2}$$

Rumus 2.2 memiliki keterangan sebagai berikut:

- t: jumlah kemunculan kata *"t"* dalam dokumen *"d"*
- d: total kata pada suatu dokumen

2.5.2 IDF (Inverse Documument Frequency)

Nilai yang digunakan untuk mengukur seberapa penting suatu kata dalam keseluruhan dokumen. IDF mengurangi kata-kata sering muncul dalam beberapa dokumen dan meningkatkan bobot kata-kata yang jarang muncul. Semakin jarang sebuah kata muncul dalam dokumen, semakin tinggi nilai IDF-nya [20]. Rumus 2.3 berikut ini digunakan untuk menghitung nilai IDF

$$IDF(t) = \log\left(\frac{N}{DF(t)}\right) \tag{2.3}$$

Rumus 2.3 memiliki keterangan sebagai berikut:

- N: total dokumen yang ada
- *df(t)*: jumlah dokumen dengan kata "*t*"

2.6 Naïve Bayes

Algoritma yang digunakan dalam analisis sentimen ini adalah Naïve Bayes. Algoritma ini merupakan salah satu algoritma dengan pendekatan *supervised learning* [7]. Naïve Bayes melakukan klasifikasi berdasarkan probabilistik dan statistik yang dikenalkan oleh Thomas Bayes untuk melakukan prediksi berdasarkan pengalaman dimasa sebelumnya sehingga dikenal sebagai *Teorema Bayes* Teorema ini digunakan bersamaan dengan Naive sehingga kondisi antar atribut tidak bergantung satu sama lain [21]. Persamaan dari teorema Bayes ini ditunjukkan pada Persamaan 2.4 berikut [22].

$$P(H \mid X) = \frac{P(X \mid H) \cdot P(H)}{P(X)} \tag{2.4}$$

Rumus 2.4 memiliki keterangan sebagai berikut:

- $P(H \mid X)$: Probabilitas hipotesis Y berdasarkan kondisi X (*posteriori probability*)
- $P(X \mid H)$: Probabilitas X berdasarkan hipotesis Y (*likelihood*)
- *P(H)*: Probabilitas hipotesis Y (*prior probability*)
- *P(X)*: Probabilitas X (*evidence*)

A Multinomial Naive Bayes

Multinomial Naive Bayes merupakan salah satu varian dari algoritma Naïve Bayes . Algoritma ini bekerja dengan menghitung frekuensi kemunculan frekuensi kata [9]. Metode dari algoritma ini terdiri dari dua tahap, yaitu tahap pelatihan dan tahap klasifikasi. Proses klasifikasi dilakukan untuk menentukan kelas terbaik dari suatu dokumen. Proses ini dihitung melalui Persamaan 2.5 berikut untuk mendapatkan *maximum a posteriori* [23].

$$c_{\text{map}} = \arg\max_{c \in C} P(c) \prod_{k=1}^{n} P(t_k \mid c) \tag{2.5}$$

Rumus untuk menghitung *prior probability*

$$P(c) = \frac{N_c}{N} \tag{2.6}$$

Diterapkan juga metode *laplace smoothing* dengan menambahkan angka satu. Metode ini diterapkan untuk menghindari probabilitas dari suatu kata memiliki nilai 0 [23]. Penerapan *laplace smoothing* ditunjukkan pada Persamaan 2.7 [22].

$$P(t_k \mid c) = \frac{1 + N_k}{|V| + N'} \tag{2.7}$$

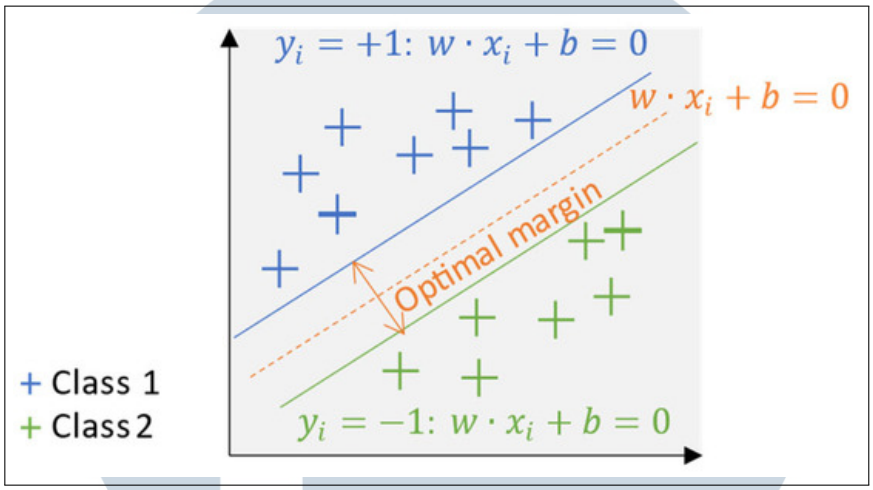
Persamaan 2.5 dan 2.7 memiliki keterangan sebagai berikut:

- *argmax*: nilai *posterior probability* terbesar
- $P(c)$: *prior probability* dokumen pada kelas c
- $P(c|d)$: probabilitas dokumen d termasuk dalam kelas c
- $P(t_k \mid c)$: probabilitas kata t_k pada dokumen dengan kelas c
- N: jumlah total semua kata dari semua dokumen pada kelas c
- N_k: jumlah kemunculan kata t_k dalam semua dokumen yang termasuk kelas c
- $|V|$: jumlah kata unik (ukuran kosakata)

2.7 Support Vector Machine (SVM)

Support Vector Machine (SVM) merupakan algoritma dengan pendekatan *supervised learning* yang digunakan untuk menganalisis data, mengenali pola, serta dapat diterapkan untuk klasifikasi maupun regresi [14]. SVM bekerja dengan mencari sebuah garis *hyperplane* yang berfungsi untuk memisahkan dua kelas data dengan optimal. Algoritma ini memilih *hyperplane* dengan jarak maksimum terhadap titik-titik data terluar dari masing-masing kelas [11]. Titik-titik data terluar yang memengaruhi posisi *hyperplane* disebut sebagai *support vector*, sedangkan jarak antara *hyperplane* dan support vector disebut sebagai *margin*. Besarnya margin menentukan kemampuan model dalam melakukan generalisasi. Semakin

besar margin, maka semakin baik model melakukan generalisasi dan mengenali pola pada data baru [24]. SVM dapat divisualkan seperti Gambar 2.1 yang menunjukkan keterangan setiap garis dan titik.



Gambar 2.1. Grafik SVM dengan *margin, hyperplane,* dan keterangan kelas
Sumber: [25]

Tujuan utama dari SVM adalah mencari *hyperplane* optimal yang dapat memisahkan data dari dua kelas secara maksimal. *Hyperplane* yang optimal adalah garis yang memiliki margin terbesar terhadap titik data dari masing-masing kelas. Rumus 2.8 berikut digunakan untuk menyatakan hyperplane [25].

$$w \cdot x_i + b = 0 \tag{2.8}$$

Rumus tersebut memiliki keterangan w sebagai bobot vektor, x_i sebagai data ke-i, dan b sebagai bias. Pemisahan kelas dapat dilakukan dengan memenuhi syarat yang ditunjukkan pada Pertidaksamaan 2.9 dan 2.10.

$$w \cdot x_i + b \geq +1 \text{ untuk } y_i = +1 \tag{2.9}$$

$$w \cdot x_i + b \leq -1 \text{ untuk } y_i = -1 \tag{2.10}$$

Garis *hyperplane* yang optimal adalah garis yang memiliki margin terbesar. Pencarian garis ini dilakukan dengan memaksimalkan jarak antara hyperplane dan titik data terdekatnya. Pencarian margin maksimum dapat dirumuskan sebagai

quadratic programming problem yang ditunjukkan pada Rumus 2.11 [17].

$$\text{minimize}\,\frac{1}{2}\|w\|^2 \tag{2.11}$$

Rumus 2.11 harus memenuhi syarat kendala yang ditunjukkan pada Pertidaksamaan 2.12

$$y_i(w \cdot x_1 + b) - 1 \geq 0 \tag{2.12}$$

Optimisasi ini dapat diselesaikan menggunakan *Lagrange Multiplier* yang ditunjukkan pada Rumus 2.13 berikut ini.

$$Lp = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i y_i (w \cdot x_i + b) - 1 \tag{2.13}$$

Penjelasan sebelumnya memiliki asumsi bahwa data dari dua kelas dapat dipisahkan secara sempurna oleh *hyperplane*. Namun, tidak jarang ditemui dua kelas pada tidak dapat terpisah secara sempurna yang menyebabkan proses optimisasi tidak dapat diselesaikan karena tidak ada w dan b yang memenuhi Pertidaksamaan 2.12. Untuk mengatasi kesalahan klasifikasi, digunakan pendekatan *soft margin*. Pendekatan ini menambahkan variabel *slack* (ξ_i) pada pertidaksamaan [17]. Penambahan variabel *slack* dilakukan pada Rumus 2.14 dan harus memenuhi syarat kendala pada Pertidaksamaan 2.15.

$$\min_{w \in R, b \in R, \xi \in R} = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i \tag{2.14}$$

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i \tag{2.15}$$

Rumus 2.14 dan Pertidaksamaan 2.15 tersebut memiliki penambahan variabel ξ_i dan parameter C. Variabel ini berfungsi untuk memberikan toleransi terhadap pelanggaran margin pada data yang tidak dapat dipisahkan secara sempurna. Sementara itu parameter C berfungsi untuk mengontrol *trade-off* margin maksimum dan kesalahan klasifikasi. Nilai C yang besar akan memberikan penalti yang lebih tinggi terhadap kesalahan klasifikasi dan membatasi pelanggaran terhadap margin [25].

Klasifikasi yang dilakukan oleh SVM hanya bekerja pada data yang dapat dipisahkan secara linear. Namun, pendekatan *kernel trick* dapat digunakan untuk data yang distribusi kelasnya tidak linear dengan. Trik *kernel* ini berfungsi untuk memetakan data menjadi dimensi yang lebih tinggi supaya data dapat terpisahkan secara linear [25]. Fungsi *kernel* yang dapat digunakan pada SVM ditunjukkan pada Fungsi 2.16-2.19

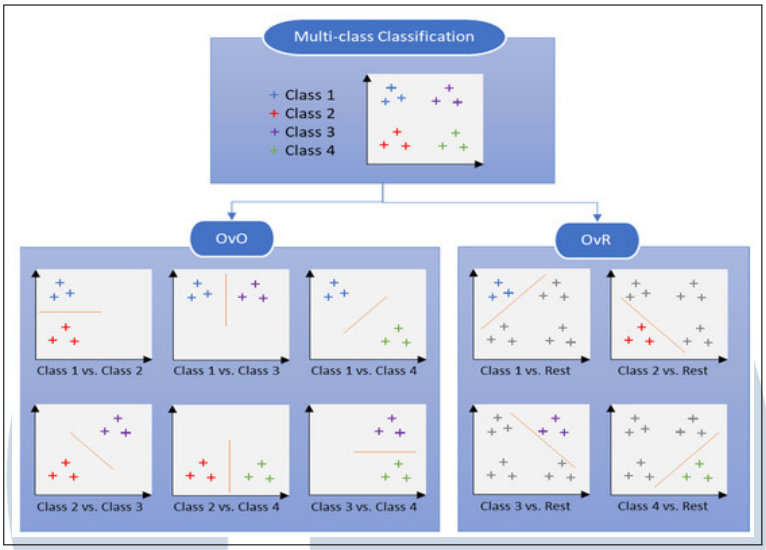
$$\text{Linear kernel: } K(x_i, x_j) = x_i \cdot x_j \tag{2.16}$$

$$\text{Polynomial kernel: } K(x_i, x_j) = (\gamma x_i \cdot x_j + r)^d \tag{2.17}$$

$$\text{RBF kernel: } K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right) \tag{2.18}$$

$$\text{Sigmoid kernel: } K(x_i, x_j) = \tanh\left(\gamma x_i \cdot x_j + r\right) \tag{2.19}$$

Klasifikasi data dengan jumlah kelas lebih dari dua menggunakan SVM dapat dilakukan melalui pendekatan *multi-class classification*. Dua metode yang umum digunakan untuk menangani kasus ini adalah *One-vs-One* (OvO) dan *One-vs-Rest* (OvR). Metode OvO membagi klasifikasi menjadi sejumlah k(k-1)/2 pasangan klasifikasi biner dengan k merupakan jumlah kelas [**?**] Sementara itu, metode OvR membangun k model klasifikasi biner untuk membedakan satu kelas terhadap gabungan kelas lainnya [25]. Pendekatan *multi-class classification* diilustrasikan pada Gambar 2.2 berikut.



Gambar 2.2. Ilustrasi *multi-class classification* SVM

Sumber: [26]

2.8 Logistic Regression

Logistic Regression merupakan salah satu algoritma yang digunakan untuk melakukan analisis hubungan antara variabel independen (fitur) dengan variabel dependen (label) yang bersifat kategorikal [27]. Algoritma ini bertujuan untuk memodelkan probabilitas suatu data termasuk ke dalam suatu kelas tertentu. Logistic Regression bekerja dengan memanfaatkan fungsi sigmoid yang mengubah kombinasi linear dari variabel input menjadi nilai probabilitas, yaitu bernilai antara 0 dan 1. Fungsi ini menghasilkan kurva berbentuk ”S” yang memastikan bahwa output dari model berada dalam rentang 0 dan 1 [28]. Output ini kemudian dibandingkan dengan nilai *threshold* yang digunakan untuk menentukan kelas prediksi data. Rumus 2.20 merupakan fungsi sigmoid yang digunakan untuk menghitung probailitas suatu kelas pada model Logistic Regression [29].

$$P(\mathrm{y}=1 \mid x)=\frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_n x_n)}} \tag{2.20}$$

Rumus 2.20 memiliki keterangan sebagai berikut:

- β_0: *intercept*
- β_1: koefisien regresi untuk fitur x_i
- x = *feature*

Untuk klasifikasi dengan kelas yang lebih dari dua, digunakan algoritma Multinomial Logistic Regression. Algoritma ini dapat melakukan klasifikasi variabel dependen yang lebih dari dua kategori dengan membandingkan beberapa fungsi logit terhadap satu kelas referensi, yaitu $Y = 0$. Rumus 2.21 dan 2.22 digunakan dalam model *Multinomial Logistic Regression* untuk membandingkan logit dari setiap kelas terhadap kelas referensi [30].

$$g_1(x) = \ln\left(\frac{P(Y=1 \mid x)}{P(Y=0 \mid x)}\right) = \beta_{10} + \beta_{11}x_1 + \beta_{12}x_2 + \cdots + \beta_{1p}x_p \tag{2.21}$$

$$g_2(x) = \ln\left(\frac{P(Y=2 \mid x)}{P(Y=0 \mid x)}\right) = \beta_{20} + \beta_{21}x_1 + \beta_{22}x_2 + \cdots + \beta_{2p}x_p \tag{2.22}$$

Rumus untuk menghitung probabilitas setiap kelas pada model *Multinomial Logistic Regression* ditunjukkan pada Rumus 2.23, 2.24, 2.25 berikut [30]:

$$P(y=0 \mid x) = \frac{1}{1 + e^{g_1(x)} + e^{g_2(x)}} \tag{2.23}$$

$$P(y=1 \mid x) = \frac{e^{g_1(x)}}{1 + e^{g_1(x)} + e^{g_2(x)}} \tag{2.24}$$

$$P(y=2 \mid x) = \frac{e^{g_2(x)}}{1 + e^{g_1(x)} + e^{g_2(x)}} \tag{2.25}$$

2.9 SMOTE

SMOTE (*Synthetic Minority Over-sampling Technique*) merupakan salah satu teknik yang digunakan dalam menangani data tidak seimbang (*imbalanced*). SMOTE bekerja dengan melakukan *oversampling* pada kelas minoritas dengan cara menambahkan data buatan untuk menyeimbangkan persebaran data [31]. Penerapan ini akan menghasilkan perseraban data menjadi seimbang pada tiap

kelasnya.

2.10 Hyperparameter Tuning

Hyperparameter Tuning merupakan metode yang digunakan untuk meningkatkan akurasi model dengan menentukan parameter-parameter tertentu sebelum proses pelatihan model. Parameter yang ditentukan sebelum pelatihan model ini disebut sebagai *hyperparameter*. Kombinasi dari berbagai nilai *hyperparameter* dapat menghasilkan model dengan performa terbaik. Proses pencarian kombinasi *hyperparameter* ini dapat dilakukan melalui beberapa metode, seperti *Grid Search, Random Search, dan Bayesian Optimization* [32].

Penelitian ini menggunakan Random Search (RS) untuk menemukan kombinasi hyperparameter yang optimal. Random Search (RS) merupakan salah satu metode optimasi *hyperparameter* yang mencoba setiap kombinasi parameter secara acak dari parameter yang telah ditentukan. Kombinasi dari berbagai parameter ini diharapkan dapat menghasilkan model terbaik dengan tingkat akurasi yang lebih tinggi. [33]

2.11 Ensemble Learning

Ensemble learning adalah teknik penggabungan dua atau lebih *classifier* untuk membangun model dengan performa lebih baik. Berbagai metode ensemble yang dapat digunakan seperti bagging, boosting, voting, dan stacking [34].

Stacking adalah metode yang menggabungkan beberapa model awal yang disebut sebagai *base model* untuk meningkatkan akurasi prediksi. *Base model* merupakan model tunggal yang dilatih menggunakan data latih dan menghasilkan output berupa prediksi dari masing-masing model. Hasil prediksi dari *base model* tersebut digunakan untuk membentuk dataset baru yang menjadi *input* bagi *meta model*. Meta model akan menghasilkan prediksi akhir dengan mempelajari *output* dari *base model* [35].

Base model yang akan digunakan untuk proses *ensemble learning* adalah model Naïve Bayes dan *Support Vector Machine* (SVM) . Kedua model ini digabung untuk proses *ensemble learning* dengan menggunakan stacking. Output dari penggabungan *base model* ini akan menjadi input untuk *meta model* menghasilkan prediksi akhir. *Meta model* yang digunakan untuk proses ini adalah algoritma *Logistic Regression*.

2.12 Cross Validation

Cross Validation (CV) merupakan salah satu metode *resampling* data yang digunakan untuk menilai kemampuan generalisasi model prediktif untuk mencegah *overfitting* [7]. *Cross-validation* dilakukan untuk mengetahui kinerja akhir dari model terhadap data baru. Metode *cross-validation* yang digunakan pada penelitian ini adalah *Stratified K-Fold Cross Validation* (SKCV). SKCV merupakan salah satu varian dari *K-Fold Cross Validation*. Pada K-Fold Cross Validation, data dibagi menjadi *k* lipatan (*fold*). Setiap lipatan secara bergantian digunakan sebagai data uji, sementara lipatan lainnya digunakan untuk pelatihan model. Namun pada K-Fold standar, pembagian kelas dilakukan secara acak sehingga dapat menyebabkan ketidakseimbangan distribusi kelas pada setiap *fold*.

Distribusi kelas yang tidak seimbang dapat meningkatkan risiko terjadinya *overfitting* [36]. Untuk mengatasi distribusi kelas yang tidak seimbang, digunakan SKCV yang memastikan bahwa setiap *fold* memiliki pembagian kelas yang seimbang. Dengan demikian, model yang dihasilkan lebih representatif untuk setiap kelasnya dan memiliki kemampuan generalisasi yang lebih baik terhadap data baru

2.13 Confusion Matrix

Confusion Matrix adalah matriks yang berisi data prediksi dan dibandingkan dengan data aktual untuk evaluasi proses model klasifikasi berupa jumlah data uji yang benar dan salah [11]. *Confusion Matrix* berfungsi untuk mengukur performa klasifikasi. Pada matriks ini terdiri dari sebuah tabel yang memiliki 4 kombinasi berbeda dari nilai prediksi dan nilai aktual. Tabel 2.1 merupakan contoh dari *confusion matrix* yang digunakan untuk mengukur performa model.

Tabel 2.1. Confusion matrix

Aktual	**Prediksi**	
	Positif	Negatif
Positif	TP (*True Positive*)	FN (*False Negative*)
Negatif	FP (*False Positive*)	TN (*True Negative*)

Tabel 2.1 memiliki keterangan sebagai berikut:

- *TP (True Positive)*: data aktual bernilai positif dan model prediksi bernilai positif

- *TN (True Negative)*: data aktual bernilai negatif dan model prediksi bernilai negatif
- *FP (False Positive)*: data aktual bernilai positif dan model prediksi bernilai negatif
- *FN (False Negative)*: data aktual bernilai negatif dan nilai prediksi positif

Berdasarkan nilai-nilai pada matriks, dapat diperoleh nilai *accuracy, precision, recall,* dan *F1-Score* dari model yang telah dikembangkan. Berikut merupakan rumus yang digunakan untuk menghitung setiap nilai yang dibutuhkan untuk mengetahui performa model.

- *Accuracy*

  Akurasi (*Accuracy*) adalah nilai rasio data dari *post* yang berhasil dideteksi dalam pengujian [8]. Rumus untuk menghitung nilai akurasi adalah sebagai berikut:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.26}$$

- *Precision*

  Presisi (*Precision*) adalah nilai ketepatan sistem mengenai informasi sistem untuk menunjukkan data positif dan data negatif yang benar. Rumus untuk menghitung nilai presisi adalah sebagai berikut:

$$Precision = \frac{TP}{TP + FP} \tag{2.27}$$

- *Recall*

  Recall adalah nilai yang digunakan untuk mengukur seberapa banyak dari seluruh data positif yang berhasil diprediksi dengan benar oleh model. Rumus yang digunakan untuk menghitung nilai *recall* adalah sebagai berikut:

$$Recall = \frac{TP}{TP + FN} \tag{2.28}$$

- *F1-Score*

  F1-Score adalah perbandingan rata-rata presisi dan recall yang dibobotkan. Rumus yang digunakan untuk memperoleh *F1-Score* adalah sebagai berikut:

$$\textit{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.29}$$