BAB 2
LANDASAN TEORI

Analisis sentimen yang dilakukan dalam penelitian ini didasarkan pada telaah literatur yang berfungsi sebagai landasan teori. Literatur yang digunakan mencakup berbagai referensi yang relevan, baik dari penelitian sebelumnya seperti jurnal ilmiah dan artikel yang membahas tentang analisis sentimen, beberapa literatur yang digunakan antara lain adalah :

2.1 Analisis Sentimen

Analisis sentimen merupakan suatu proses penggunaan *text analytics* untuk mendapatkan berbagai macam data untuk memperoleh sentimen atau opini dari media internet dan berbagai macam media sosial. Tujuan dari analisis sentimen adalah mendapatkan suatu kesimpulan yang berupa opini dari media tersebut [11]. Analisis sentimen dapat memahami dan membentuk kelompok setiap emosi dalam kategori positif, netral, dan negatif [11].

2.2 Media Sosial X (Twitter)

Media sosial twitter atau yang sekarang dikenal dengan X merupakan sebuah layanan jejarin sosial dan juga *microblog* yang memungkinkan penggunanya berkirim dan membaca pesan yang disebut sebagai *tweet*. Twitter didirikan pada 21 Maret 2006 oleh Jack Dorsey,Noah Glass, Biz Stone, dan Evan Williams. Twitter dirilis publik pada 15 Juli 2006 dan diakuisisi oleh Elon Musk pada tahun 2022 [12].

2.3 Tweet Harvest

Tweet harvest merupakan sebuah *command-line tools* yang digunakan untuk mendapatkan data dari *tweets* yang telah diunggah oleh pengguna X, metode *tweet harvest* mengumpulkan data berdasarkan *keywords* yang menjadi input dan data yang telah dikumpulkan akan dikonversi menjadi file CSV. *Requirement* untuk menggunakan *tweet harvest* dibutuhkan *authorization token* yang didapat melalui akun X dari setiap pengguna [13].

2.4 TF-IDF (Term Frequency - Inverse Document Frequency)

Metode dari *Term Frequency - Inverse Document Frequency* (TF-IDF) merupakan suatu teknik untuk mengolah teks dan pemodelan bahasa alami [14]. Tujuan dari penggunaan metode TF-IDF adalah untuk mengevaluasi seberapa penting kata dalam sebuah dokumen [14]. Rumus yang digunakan TF-IDF untuk kata w_i dalam dokumen d adalah sebagai berikut :

$$\text{T}F\text{-}IDF(w_i,d) = TF(w_i,d) \times IDF(w_i) \tag{2.1}$$

Dalam perhitungannya TF-IDF memperhitungkan dua faktor penting, faktor tersebut antara lain :

2.4.1 Term Frequency (TF)

Term frequency digunakan untuk mengukur intensitas suatu kata muncul dalam sebuah dataset, pendekatan umum untuk menghitung TF adalah menghitung jumlah kemunculan kata dibagi dengan total kata dalam dataset [14]. Rumus yang digunakan untuk menghitung *term frequency* adalah sebagai berikut :

$$\text{T}F(w_i,d) = \frac{f(w_i,d)}{\sum_{i=1}^{n} f(w_i,d)} \tag{2.2}$$

- $TF(w_i,d)$ = Nilai Term Frequency dari kata w_i dalam dokumen d.
- $f(w_i,d)$ = Jumlah kemunculan kata w_i dalam dokumen d.
- $\sum_{i=1}^{n} f(w_i,d)$ = Total jumlah seluruh kata dalam dokumen d.

2.4.2 Inverse Document Frequency (IDF)

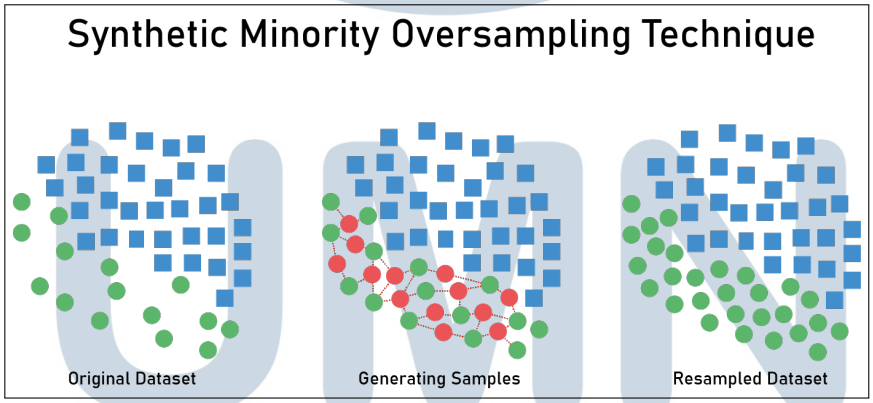
Inverse document frequency digunakan untuk menghitung seberapa penting suatu kata dalam konteks dataset yang lebih besar, kata-kata yang muncul lebih jarang di dataset berpeluang memiliki IDF yang lebih tinggi [14]. Rumus yang digunakan untuk menghitung *inverse document frequency* adalah sebagai berikut :

$$\text{I}DF(w_i) = \log \frac{N}{df(w_i)} \tag{2.3}$$

- N adalah total jumlah dokumen dalam korpus.
- $df(w_i)$ adalah jumlah dokumen yang mengandung kata w_i.

2.5 SMOTE

SMOTE atau *Synthetic Minority Oversampling Technique* adalah suatu teknik yang digunakan untuk menyeimbangkan distribusi data antara kelas minoritas dan mayoritas dengan cara menghasilkan sampel baru dari kelas minoritas hingga jumlahnya setara dengan kelas mayoritas [15]. Penerapan metode *SMOTE* memiliki potensi menyebabkan *overfitting*. Hal ini disebabkan oleh proses duplikasi data pada kelas minoritas, yang bisa mengakibatkan kemunculan data pelatihan yang identik. Proses *SMOTE* dimulai dengan menghitung jarak antar data dalam kelas minoritas, kemudian menentukan persentase *SMOTE* yang diinginkan, memilih jumlah tetangga terdekat (k), dan akhirnya menghasilkan data sintetik menggunakan rumus tertentu [15].



Gambar 2.1. Ilustrasi smote oversampling [16]

2.6 Algoritma Klasifikasi

Klasifikasi merupakan salah satu metode dalam pembelajaran mesin yang digunakan untuk memprediksi label atau kategori dari suatu data berdasarkan fitur yang dimilikinya. Pada penelitian ini, digunakan beberapa algoritma klasifikasi seperti Support Vector Machine (SVM), Multinomial Naive Bayes (MNB), dan

Random Forest (RF), yang masing-masing memiliki pendekatan berbeda dalam membangun model prediktif untuk data teks.

2.6.1 Algoritma Naive Bayes

Naive Bayes merupakan sebuah pengklasifikasian *probabilistic* sederhana yang menghitung probabilitas dengan penjumlahan beberapa frekuensi dan kombinasi *values* dari dataset yang diberikan [17]. Cara kerja dari metode *naive bayes* adalah melakukan *learning* atau pembelajaran terhadap data yang telah diambil, dari data yang telah di *learning* data tersebut akan diklasifikasikan sesuai dengan kateogri yang ada [18]. Kekurangan dari penggunaan algoritma *Naive Bayes* antara lain adalah Keakuratannya tidak bisa diukur menggunakan satu probabilitas saja dan jika probabilitas kondisionalnya bernilai nol, maka probabilitas prediksi juga akan bernilai nol [19].

A Teorema Bayes

Teorema bayes merupakan konsep dasar statistika yang membantu memperbarui hipotesis berdasarkan informasi terbaru [20]. Teorema bayes berguna untuk mengetahui peluang *(oppurtunity)* berdasarkan suatu kejadian bersyarat yang tidak bisa diprediksi [20]. Rumus Teorema Bayes adalah sebagai berikut :

$$\mathrm{P}(C|d) = \frac{P(C) \cdot P(d|C)}{P(d)} \tag{2.4}$$

Di mana :

- $P(C|d)$adalah probabilitas suatu dokumen (d) yang termasuk dalam kelas (C).
- P(C) adala probabilitas awal suatu kelas atau *prior probability*.
- $P(d|C)$ adalah probabilitas dokumen (d) muncul dalam kelas (C) *likelihood*.
- P(d) adalah probabilitas dokumen d secara keseluruhan *(evidence)*.

B Multinomial Naive Bayes

Multinomial *Naive Bayes* merupakan salah satu model dari algoritma *Naive Bayes* yang sering digunakan dalam klasifikasi teks [21]. Algoritma multinomial naive bayes mengasumsikan bahwa setiap dokumen merupakan kumpulan kata-kata

yang memiliki distribusi multinomial [21]. Rumus untuk menghitung probabilitas kelas C diberikan dokumen d pada Multinomial Naive Bayes adalah:

$$\mathrm{P}(C|d) = \frac{P(C)\prod_{i=1}^{n} P(w_i|C)^{f(w_i,d)}}{P(d)} \tag{2.5}$$

Di mana:

- $P(C|d)$: Probabilitas kelas C diberikan dokumen d,
- $P(C)$: Probabilitas prior dari kelas C,
- $P(w_i|C)$: Probabilitas kata w_i muncul dalam kelas C,
- $f(w_i,d)$: Frekuensi kata w_i dalam dokumen d,
- $P(d)$: Probabilitas data dokumen d, yang biasanya menjadi konstanta dalam perhitungan, dan tidak mempengaruhi hasil klasifikasi akhir.

Jika menggunakan TF-IDF, rumus multinomial naive bayes tersebut menjadi:

$$\mathrm{P}(d|C) = \prod_{i=1}^{n} P(w_i|C)^{TF\text{-}IDF(w_i,d)} \tag{2.6}$$

- $TF\text{-}IDF(w_i,d)$: Nilai TF-IDF dari kata w_i dalam dokumen d, yang menggambarkan pentingnya kata tersebut dalam konteks dokumen tersebut.

2.6.2 Chi-Square

Chi-Square adalah metode statistik yang digunakan untuk mengukur hubungan independensi antara dua variabel kategorikal, dalam hal ini antara fitur (kata) dan label kelas [22]. Dalam konteks pemrosesan teks, uji Chi-Square digunakan untuk menilai seberapa besar suatu fitur (term) memiliki korelasi dengan kelas tertentu.

Pengujian berguna dalam proses seleksi fitur karena dapat membantu memilih fitur-fitur yang paling relevan terhadap target klasifikasi. Semakin tinggi nilai suatu fitur terhadap label, maka semakin besar kemungkinan fitur tersebut relevan terhadap kelas tertentu [22].

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{2.7}$$

Dimana:

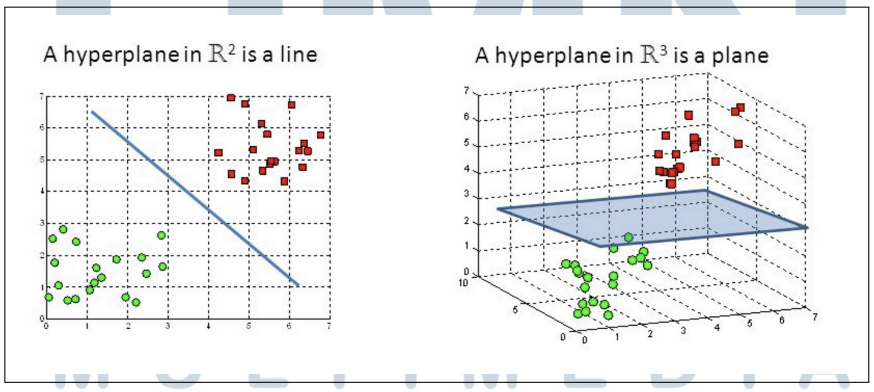
- O_i = nilai observasi (jumlah aktual fitur pada kelas tertentu),
- E_i = nilai ekspektasi (jumlah yang diharapkan jika tidak ada hubungan antara fitur dan kelas).

Pada penelitian ini, metode `SelectKBest` dari Scikit-Learn digunakan dengan fungsi skor `chi2`, untuk memilih fitur-fitur TF-IDF terbaik yang paling berkontribusi dalam klasifikasi sentimen.

2.6.3 Algoritma Support Vector Machine

Algoritma *support vector machine* adalah algoritma dari *machine learning* yang dapat digunakan untuk melakukan klasifikasi dan regresi [23]. SVM memiliki cara kerja berdasarkan *structural risk minimization* yang dirancang untuk memproses data menjadi *hyperplane* [23].

Hyperplane merupakan keputusan yang membedakan dua kelas pada SVM. Titik yang berada di kedua sisi *hyperplane* dapat dikaitkan dengan kelas yang berbeda [24]. Berikut terdapat gambar ilustrasi *hyperplane*.



Gambar 2.2. Ilustrasi svm
[25]

A Prinsip Dasar SVM

Tujuan utama SVM adalah mencari *hyperplane H* yang dapat memisahkan dua kelas data secara optimal. Persamaan *hyperplane* adalah sebagai berikut:

Dimana:

$$\mathrm{H} : \mathbf{w} \cdot \mathbf{x} + b = 0 \tag{2.8}$$

- **w**: Vektor bobot yang tegak lurus terhadap *hyperplane*.
- **x**: Data fitur.
- b: Bias (*intercept*).

Hyperplane optimal adalah hyperplane yang memiliki margin terbesar antara dua kelas data. Margin didefinisikan sebagai jarak antara hyperplane dan data terdekat dari masing-masing kelas [24]. SVM memaksimalkan margin tersebut dengan meminimalkan fungsi objektif berikut:

$$Minimalkan \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.9}$$

Dengan kendala:

$$\mathrm{y}_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \forall i \tag{2.10}$$

Dimana :

- y_i: Label kelas(+1 atau -1)
- x_i: Vektor fitur untuk data ke-i

B Linear Kernel

Linear kernel merupakan jenis kernel sederhana yang memetakan data ke ruang fitur berdimensi lebih tinggi tanpa transformasi nonlinier. Kernel ini hanya menghitung produk dot antara dua vektor fitur [26], dan umum digunakan pada data teks yang memiliki dimensi tinggi.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j \tag{2.11}$$

Pada awal eksperimen, linear kernel dipertimbangkan karena sifat data teks yang tinggi dimensinya (sparse vector dari TF-IDF). Namun, linear kernel memiliki keterbatasan dalam menangani data non-linear dan multi-kelas secara kompleks. Oleh karena itu, perlu dievaluasi kernel alternatif untuk meningkatkan performa klasifikasi.

C Radial Basis Function (RBF) Kernel

Kernel RBF (Radial Basis Function) merupakan jenis kernel nonlinier yang banyak digunakan pada SVM karena kemampuannya dalam menangani data yang tidak dapat dipisahkan secara linear. RBF memetakan data ke ruang fitur berdimensi tak hingga, sehingga memfasilitasi pemisahan antar kelas yang lebih kompleks [27].

Fungsi kernel RBF dinyatakan sebagai:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \tag{2.12}$$

Dimana γ mengontrol jangkauan pengaruh satu data terhadap data lain. Semakin besar nilai γ, maka pengaruhnya lebih lokal.

Penggunaan RBF kernel pada penelitian ini ditentukan berdasarkan hasil proses *hyperparameter tuning* yang menunjukkan bahwa kombinasi parameter terbaik ditemukan pada kernel RBF. Model dengan RBF kernel secara konsisten menunjukkan performa klasifikasi yang lebih tinggi dibandingkan kernel linier pada skenario multi-kelas. Oleh karena itu, RBF digunakan sebagai kernel utama dalam pemodelan akhir.

2.6.4 Decision Tree

Decision tree merupakan salah satu metode yang digunakan dalam penerapan pola melalui sejumlah tahapan dan proses yang berkaitan dengan aktivitas klasifikasi dan prediksi [28]. Dalam prosesnya, *Decision Tree* dimulai dari keseluruhan data yang memuat berbagai kondisi dan informasi lengkap, kemudian berlanjut hingga menghasilkan suatu keputusan akhir [29]. Struktur dari *decision*

tree menyerupai struktur dari sebuah pohon, dengan penjelasan sebagai berikut:

1. *Root Node*

   Root node merupakan *node* atau simpul paling atas pada *decision tree*, *root node* merupakan titik awal pemrosesan data pada dataset.

2. *Internal Node*

   Internal node merupakan *child* dari *root node*, *internal node* memiliki *input* dan *output* pada *node* di *decision tree*, dengan maksimal dua *output*. jumlah *internal node* berdasarkan filtrasi dari *root node*.

3. *Leaf Node*

   Leaf node merupakan *child* dari *internal node*, *leaf node* hanya memiliki *input* dan tidak memiliki *ouput*, *leaf node* merupakan *node* terakhir pada *decision tree*.

Proses *decision tree* dapat dimulai dengan cara mencari nilai *entrophy* dan selanjutnya menghitung nilai *information gain* [30].

1. **Entropy** dari sebuah himpunan data D didefinisikan sebagai [31]:

$$\text{E}ntropy(D) = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{2.13}$$

   di mana:

   - n adalah jumlah kelas,
   - p_i adalah probabilitas kemunculan data dari kelas ke-i dalam himpunan data D.

2. **Information Gain** dari atribut A terhadap data D dihitung dengan [31]:

$$\text{G}ain(D,A) = Entropy(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} \cdot Entropy(D_v) \tag{2.14}$$

   di mana:

   - $Values(A)$ adalah himpunan semua nilai unik dari atribut A,
   - D_v adalah subset dari D untuk nilai v pada atribut A,
   - $|D_v|$ adalah jumlah data pada D_v, dan $|D|$ adalah jumlah data pada D.

2.6.5 Random Forest

Algoritma *Random Forest* merupakan pengembangan dari model algoritma *decision tree*, algoritma *random forest* membangun banyak *decision tree* dan menggabungkannya menjadi sebuah prediksi dengan mengamati voting mayoritas [8].

Langkah-langkah kerja algoritma *Random Forest* dapat dijelaskan sebagai berikut [32]:

1. Algoritma mengambil sejumlah sampel secara acak dari dataset yang tersedia.
2. Untuk setiap sampel yang diambil, dibuat sebuah pohon keputusan (*Decision Tree*) yang menghasilkan prediksi.
3. Dilakukan proses agregasi terhadap seluruh hasil prediksi.
    - Untuk permasalahan klasifikasi, digunakan metode *voting* untuk menentukan kelas yang paling sering muncul.
    - Untuk permasalahan regresi, digunakan nilai rata-rata (*mean*) dari semua prediksi.
4. Hasil prediksi akhir ditentukan berdasarkan kelas dengan suara terbanyak (untuk klasifikasi) atau nilai rata-rata (untuk regresi).

2.7 Hyperparameter Tuning

Hyperparameter merupakan variabel konfigurasi eksternal yang ditentukan untuk mengatur proses pelatihan model machine learning. Berbeda dengan parameter model, *hyperparameter* ditentukan secara manual sebelum pelatihan dimulai. Sementara itu, parameter model sendiri adalah nilai-nilai internal yang dihasilkan secara otomatis oleh model selama proses pembelajaran dan tidak dikendalikan langsung [33].

2.8 Ensemble Learning

Ensemble Learning adalah pendekatan dalam *machine learning* di mana beberapa model, yang sering disebut sebagai ”*weak learners*”, dilatih untuk menyelesaikan permasalahan yang sama, kemudian dikombinasikan untuk menghasilkan prediksi dengan akurasi yang lebih tinggi. Konsep dasarnya

adalah bahwa dengan menggabungkan beberapa model sederhana, dapat diperoleh model yang lebih andal dan berkinerja lebih baik [34]. Tujuan utama dari metode Ensemble Learning adalah mengombinasikan beberapa algoritma *Machine Learning* untuk meningkatkan performa model secara keseluruhan. Terdapat tiga teknik utama dalam *Ensemble Learning*, yaitu Bagging, Boosting, dan Stacking. Dalam penelitian ini, teknik yang digunakan adalah*stacking*.

2.8.1 Stacking

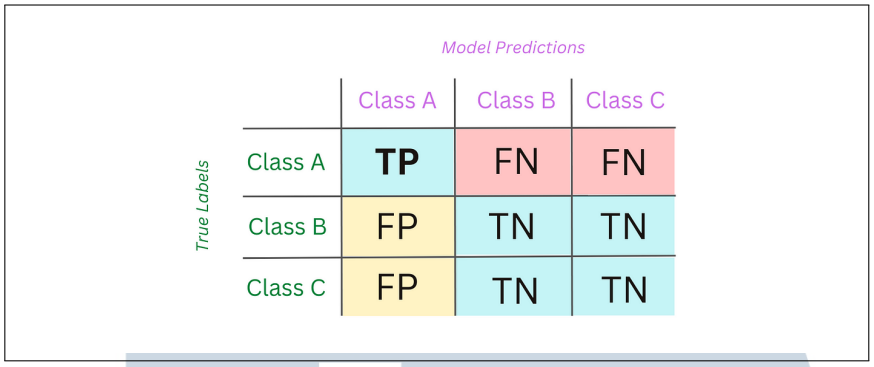
Stacking termasuk dalam salah satu metode *ensemble learning* yang berisi tumpukan dari beberapa hasil klasifikasi algoritma tunggal [35]. Arsitektur *stacking* memiliki dua level pembelajaran uatam, yaitu *level 0 base-learner* dan *level 1 meta-learner* [35].

Cara kerja metode *ensemble learning stacking* [33], yaitu :

1. Melatih beberapa *base learners* (misal: Naive Bayes, SVM, Random Forest) pada data pelatihan asli.
2. Menggunakan *cross-validation* untuk menghasilkan prediksi dari setiap *base learner*, yang kemudian menjadi fitur baru (meta-feature).
3. Melatih *meta-learner* menggunakan meta-feature dan label asli untuk mempelajari kombinasi prediksi terbaik.
4. Pada saat prediksi, *base learners* memberikan prediksi awal, lalu *meta-learner* menghasilkan prediksi akhir.

2.9 Confusion Matirx

Confusion Matrix merupakan tabel yang menyatakan klasifikasi jumlah data pengdujian yang benar dan jumlah data pengujian yang salah [36]. Confusion Matrix merupakan tabel yang memiliki empat kombinasi antara nilai prediksi dan nilai aktual [37]. Kombinasi yang dimiliki confusion matrix adalah True Positif (TP), True Negatif (TN), False Positif (FP), dan False Negatif (FN).



Gambar 2.3. Confusion matrix table

[38]

Confusion matrix memiliki perhitungan yang dapat digunakan untuk menghitung *accuracy, precision, recall dan F-score*.

2.9.1 Accuracy

Accuracy menggambarkan seberapa akurat model dalam melakukan pengklasifikasian terhadap seluruh data. Metrik ini menghitung proporsi prediksi yang benar terhadap total jumlah prediksi yang dilakukan. Semakin tinggi nilai akurasi, semakin baik kinerja model dalam mengenali seluruh kelas secara keseluruhan.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.15}$$

2.9.2 Precision

Precision menggambarkan akurasi antara data yang diminta (positif) dengan hasil prediksi yang diberikan oleh model. Metrik ini menilai seberapa banyak prediksi positif yang benar-benar relevan. Nilai precision yang tinggi menunjukkan bahwa model menghasilkan sedikit kesalahan dalam mengklasifikasikan data negatif sebagai positif.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{2.16}$$

2.9.3 Recall

Recall menggambarkan kemampuan model dalam menemukan kembali data yang relevan dari seluruh data aktual yang positif. Metrik ini penting ketika kesalahan dalam melewatkan data positif harus diminimalkan. Semakin tinggi nilai recall, semakin baik model dalam menangkap seluruh kejadian positif yang sebenarnya.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.17}$$

2.9.4 F1-Score

Score menggambarkan perbandingan rata-rata precision dan recall yang dibobotkan. Accuracy tepat kita gunakan sebagai acuan performansi algoritma jika dataset kita memiliki jumlah data False Negatif dan False Positif yang sangat mendekati (symmetric). Namun jika jumlahnya tidak mendekati, maka sebaiknya kita menggunakan F1 Score sebagai acuan.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.18}$$