BAB II
METODE PENELITIAN

2.1 Objek Penelitian

SIREKAP 2024 adalah aplikasi yang disiapkan oleh KPU untuk melakukan pencatatan dan pendokumentasian hasil penghitungan suara di TPS. Aplikasi SIREKAP 2024 telah tersedia di *Google Play Store* dengan total unduhan mencapai satu juta lebih dan delapan ribuan ulasan di awal bulan februari 2024. Setiap ulasan di aplikasi tersebut sangat bermacam-macam dan memiliki *rating* dari 1 sampai 5 bintang dengan jumlah ulasan yang berbeda pada setiap *rating*nya. Dari delapan ribuan ulasan tersebut akan diambil dan menjadi objek penelitian ini.

2.2 Alat dan Bahan

Alat yang akan digunakan dalam penelitian ini terdiri dari perangkat keras dan perangkat lunak. Perangkat keras terdiri dari Laptop Acer NITRO V15 dengan spesifikasi prosesor 13th Gen Intel(R) Core(TM) i5-13420H 2.10 GHz, kapasitas memori 16GB, dan hardisk SSD 512GB. Sedangkan perangkat lunak yang akan digunakan berupa sistem operasi *Microsoft Windows* 11, *Visual Studio Code* versi 1.87.2, *Jupyter Notebook*

versi 2024.2.0, *Python* versi 3.12.1, dan Library *Python*. Library *Python* yang akan digunakan dapat dilihat pada Tabel 2.1.

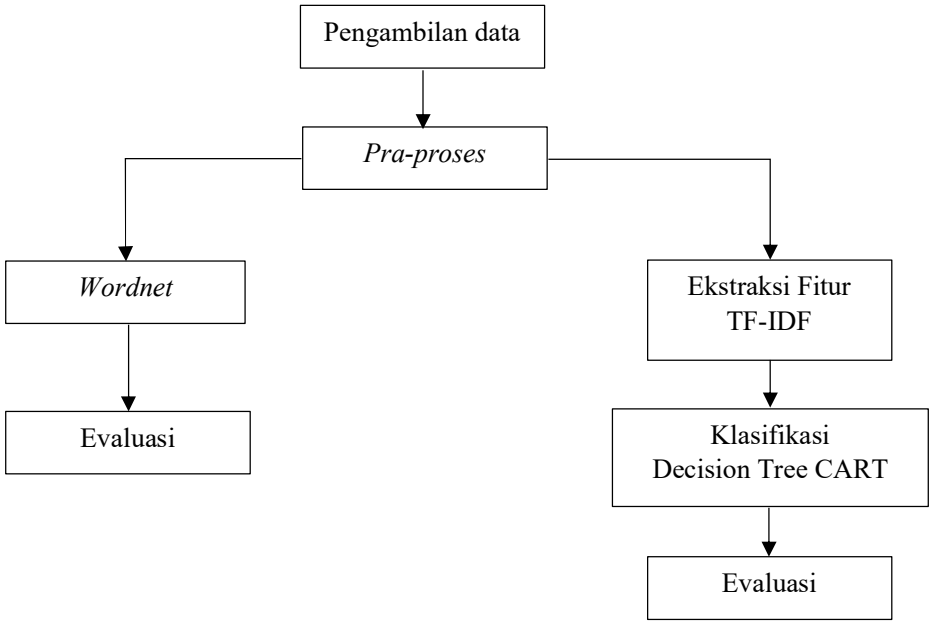
Tabel 2.1 Library *Python*

Library *Python*	Versi	Keterangan
Pandas	2.2.0	untuk mengelola dan menganalisis data
Numpy	1.26.3	pustaka *Python* yang dirancang khusus untuk manipulasi data numerik
Matplotlib	3.8.2	pustaka plot populer dengan *Python* yang digunakan untuk membuat visualisasi dan grafik berkualitas tinggi
NLTK (*Natural Language Toolkit*)	3.8.1	untuk membangun program *Python* agar dapat bekerja dengan data bahasa manusia
Scikit-learn	1.4.0	pustaka *Machine Learning* yang menyediakan berbagai algoritma *Machine Learning*
TextBlob	0.18.0	untuk menganalisa sentimen teks, untuk mengetahui apakah sebuah teks bersifat positif, negatif, atau netral pada sentimen analisis
Sastrawi	1.0.1	mengubah kata berimbuhan ke bentuk dasarnya atau yang biasa disebut *root*
Deep-translator	1.11.4	untuk menerjemahkan berbagai bahasa dengan cara sederhana menggunakan banyak penerjemah
Wordcloud	1.9.3	mengvisualisasikan data kedalam bentuk gambar
Google-play-scraper	1.2.6	API untuk mengekstraksi data informasi aplikasi dan ulasan aplikasi dari *Google Play Store* dengan lebih mudah tanpa ketergantungan eksternal

Adapun bahan yang akan digunakan dalam penelitian ini adalah data yang terdiri dari *rating* dan ulasan yang diambil dari aplikasi SIREKAP 2024 di *Google Playstore* pada tanggal 6 Februari 2024 jam 22:00 WITA.

2.3 Prosedur Penelitian

Tahap penelitian yang akan digunakan untuk membandingkan hasil evaluasi dua tahapan analisis sentimen pada aplikasi SIREKAP 2024 dapat ditunjukkan pada Gambar 2.1.



Gambar 2.1 Alur Penelitian

Tahap awal yang akan dilakukan dalam penelitian ini yaitu pengambilan data. Setelah itu, data akan melalui tahap *Pra-proses*. Kemudian, data dari tahapan *Pra-proses* akan digunakan untuk klasifikasi dan analisis ulasan yakni *Wordnet* dan *Decision Tree* CART. Dalam tahapan *Wordnet*, tahapan setelah *Pra-proses* akan diklasifikasikan ke dalam kelas *rating*. Lalu dalam tahapan *Decision Tree* CART, tahapan setelah *Pra-proses* akan melalui tahapan ekstraksi fitur TF-IDF terlebih dahulu sebelum diklasifikasikan menggunakan model *Decision Tree* CART. Terakhir, masing-masing model akan di evaluasi dan di analisis perbandingan dari hasil kedua model tersebut.

2.3.1 Pengambilan Data

Data yang akan digunakan dalam penelitian ini merupakan data ulasan pada aplikasi SIREKAP 2024 di *Google Play Store*[1]. Dataset diambil pada tanggal 6 februari 2024 menggunakan software *Visual Studio Code* dengan library *Python* yakni *Numpy*, *Pandas*, dan *Google-play-scraper*. Library ini bertujuan untuk mengambil data ulasan dari aplikasi SIREKAP 2024 di *Google Play Store*.

Dalam mengambil data ulasan terdapat tahapan yang akan dilakukan. Tahap pertama yaitu mengimpor library yang akan digunakan ke dalam *Visual Studio Code*. Selanjutnya, memasukan link

[1] https://play.google.com/store/apps/details?id=id.go.kpu.sirekap2024

id dari aplikasi SIREKAP 2024 untuk mengambil data ulasan dari aplikasi tersebut. Setelah mengambil data ulasan, langkah selanjutnya membuat *dataframe* (df) untuk menampilkan seluruh data dalam bentuk tabel dan menampilkan jumlah data yang terambil. Kemudian df akan diatur untuk filter data yang diperlukan pada penelitian ini. Selanjutnya, data diekspor dengan format CSV dan menggunakan separator '|' sebagai pembatas pada data tersebut.

2.3.2 *Pra-proses* Data

Pra-proses data merupakan langkah di mana data diubah, digabungkan, atau disesuaikan agar dapat dimengerti oleh sistem dengan optimal (Cholis & Ulinnuha, 2023). Dataset yang didapat dari tahap pengambilan data merupakan data mentah dan perlu dilakukan *Pra-proses* data. Pada tahapan penelitian ini, menggunakan beberapa tahapan dalam *Pra-proses* data sebagai berikut.

1. *Lower Case*

Lower Case akan merubah data ulasan berupa teks menjadi bentuk huruf kecil untuk membantu mesin dalam menghindari karakter berbeda yang seharusnya sama.

2. *Remove Unecessary Character*

Remove Unecessary Character akan menghapus atribut spesifik dan semua karakter non-alfanumerik atau karakter yang tidak diperlukan pada ulasan seperti simbol, emoji, karakter yang berulang, tanda baca, dan karakter spesial lainnya.

3. *Spell Checker*

Spell Checker akan digunakan untuk memastikan konsistensi tingkat tinggi dalam teks yang digunakan dalam penelitian dengan menstandarkan kata-kata yang berlebihan dan tidak baku ke dalam satu konteks. *Spell Checker* dapat membantu tidak hanya dalam menemukan dan memperbaiki kesalahan ejaan tetapi juga dalam menghilangkan variasi dan redundansi kata, yang dapat berdampak pada interpretasi dan analisis temuan penelitian.

Pada tahap ini, standarisasi akan menggunakan *Kamus Bahasa Indonesia* (Bahasa, 2008) untuk mendeteksi kata yang tidak baku. Sebelum dilakukan *Spell Checker*, akan dibuat kamus bernama "kamus_tidak_baku" dalam format *Comma Separated Value* (CSV) untuk standarisasi kata-kata tidak jelas dan berlebih seperti kata "ancurrrrrrrr" yang distandarisasi menjadi "hancur", kata "lemot" menjadi "lambat" dan sebagainya.

Tahap pertama dalam membuat kamus yang akan digunakan dalam penelitian ini adalah mengkonversi Kamus Bahasa Indonesia dari bentuk *Portable Document Format* (PDF), menjadi *Text File* (TXT) dan diimpor ke dalam *Python*. Selanjutnya, karakter non-alfanumerik yang terdapat pada kamus akan dihapus dan teks dipecah menjadi kata-kata. Kata-kata tersebut kemudian difilter untuk menghapus kata yang mengandung numerik dan mengambil kata yang berjumlah lebih dari sama dengan 4 huruf. Hal ini dilakukan karena kata-kata yang hanya terdiri dari 1 – 3 huruf masih ada yang memiliki konteks yang kurang jelas. Setelah itu, hasilnya akan diekspor dengan nama "4 list lebih kata KBBI.txt"

Selanjutnya, "4 list lebih kata KBBI.txt" ini akan dijadikan acuan untuk mendeteksi kata tidak baku atau kata yang tidak terdeteksi oleh acuan tersebut di dalam data ulasan. Kata yang tidak tidak baku kemudian di ekspor kedalam format CSV dan setiap kata akan diubah secara manual dengan cara menambahkan kolom baru yang berisi kata baku dari kata tidak baku tersebut. Setelah semua

kata-kata yang tidak baku telah dibakukan, hasilnya akan diekspor dengan nama "kamus_tidak_baku.csv" dan siap digunakan untuk proses *Spell Checker*.

4. *Stemming*

Stemming merupakan tahap untuk mengubah kata imbuhan menjadi kata dasar menggunakan library *Sastrawi* seperti 'mencobanya' yang berasal dari kata dasar 'coba', 'dipakai' yang memiliki kata dasar 'pakai', dan 'menjalankan' menjadi 'jalan'. *Stemming* pada penelitian ini akan menggunakan Sastrawi.Stemmer.StemmerFactory yang terdapat pada library *Sastrawi*.

5. Hapus Data Kosong

Tahap akhir dari *Pra-proses* data adalah hapus data kosong. Hapus data kosong dilakukan untuk membantu menghilangkan bias yang mungkin disebabkan oleh nilai yang hilang, sehingga analisis sentimen diharpkan dapat memberikan hasil yang lebih akurat dan dapat diandalkan. Proses awal hapus data kosong yaitu mengecek nilai yang memilki kosong yang terdapat pada hasil *Stemming* dengan menggantinya dengan '*NaN'* menggunakan fungsi *'replace'*. Selanjutnya, '*NaN'* dihapus menggunakan fungsi '*df.dropna'*. Hasilnya akan digunakan untuk proses selanjutnya.

2.3.3 *Wordnet*

1. *Wordnet*

Wordnet adalah basis data leksikal yang menggambarkan hubungan semantik antar kata, yang digunakan untuk mengukur polaritas dan subjektivitas dalam suatu teks. Nilai polaritasnya adalah desimal dalam rentang [-1, 1], sementara nilai subjektivitasnya adalah desimal dalam rentang [0, 1]. Nilai 0 mencerminkan objektivitas yang tinggi, dan nilai 1 menunjukkan subjektivitas yang tinggi. (Fjærli & Larsen, 2022).

Pada penelitian ini, data akan diterjemahkan terlebih dahulu ke dalam bahasa inggris menggunakan API *Google Translate* yang terdapat pada library *Deep-translator* karena *Wordnet* hanya dapat memproses teks dalam bahasa inggris. Implementasi kode pada tahap ini dapat dilihat pada gambar 2.2. Selanjutnya, *Wordnet* menggunakan library *TextBlob* untuk mencari nilai polaritas kata pada data ulasan.

Nilai polaritas yang diperoleh akan diberikan kelas yang dibedakan menjadi lima sentimen berdasarkan *rating* bintang satu sampai lima dari ulasan aplikasi SIREKAP 2024. Lima kelas akan dinyatakan sebagai tingkat ulasan pengguna terhadap aplikasi tersebut dengan rentang nilai kurang dari -0,5 sampai lebih dari 0,5 seperti pada tabel 2.2. implementasi kode pada tahap penggunaan *TextBlob* dan pengklasifikasiannya dapat dilihat pada gambar 2.3.

Tabel 2.2 Pelabelan Kelas *Wordnet*

Values	Peringkat
<(-0,5)	*1 Star*
(-0,5) – (-0,2)	*2 Star*
(-0,1) – 0,1	*3 Star*
0,2 – 0,5	*4 Star*
>0,5	*5 Star*

```python
from deep_translator import GoogleTranslator
import time

def translate_komentar(text):
    text_translated = GoogleTranslator(source='id', target='en').translate(text)
    return text_translated

```

Gambar 2.2 Penggunaan *Deep-translator*

```python
from textblob import TextBlob

def scoring_sentiment(text):
    analysis = TextBlob(text)
    sentiment_score = analysis.sentiment.polarity
    # print("Sentiment polarity:", sentiment_score)
    score_wordnet = 0

    if sentiment_score > 0.5:
        score_wordnet = '5 star'
    elif 0.2 <= sentiment_score <= 0.5:
        score_wordnet = '4 star'
    elif -0.2 < sentiment_score < 0.2:
        score_wordnet = '3 star'
    elif -0.5 <= sentiment_score <= -0.2:
        score_wordnet = '2 star'
    else:
        score_wordnet = '1 star'

    return score_wordnet, sentiment_score
```

Gambar 2.3 Penggunaan *TextBlob* dan Klasifikasi *Wordnet*

Hasil dari implementasi diatas akan memperoleh nilai polaritas dari setiap teks ulasan. Nilai polaritas kemudian diklasifikasikan dalam lima kelas sentimen berdasarkan rentang nilai yang telah ditentukan. Setiap ulasan akan diberikan skor yang sesuai berdasarkan nilai polaritas yang didapatkan. Proses ini memberikan gambaran tentang tingkat kepuasan pengguna terhadap aplikasi SIREKAP 2024, yang selanjutnya akan digunakan sebagai data untuk evaluasi menggunakan *F1-Score*.

2. Evaluasi *F1-Score Wordnet*

Evaluasi pada penelitian ini menggunakan *F1-Score*. untuk mendapatkan *F1-Score* menggunakan *Confusion Matrix*. *Confusion Matrix* merupakan metode pengujian untuk klasifikasi yang bekerja dengan membandingkan hasil prediksi dengan hasil aktual dari dataset (Kurniawati & Arianto, 2019). Contoh Pengujian dengan *Confusion Matrix* dapat dilihat pada Gambar 2.4.

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Gambar 2.4 *Confusion Matrix*

True Positif (TP) adalah jumlah data positif yang diklasifikasikan sebagai nilai positif dan *True Negatif* (TN) adalah jumlah data negatif yang diklasifikasikan sebagai nilai negatif, sedangkan *False Positif* (FP) merupakan jumlah data positif yang diklasifikasikan sebagai nilai negatif dan *False Negatif* (FN) merupakan jumlah data negatif yang diklasifikasikan sebagai nilai positif.

Namun gambar 2.4 hanya dapat digunakan untuk klasifikasi biner yakni klasifikasi dengan dua kelas. Untuk mengklasfikasi data yang lebih dari dua kelas, memerlukan *Confusion Matrix* dengan klasifikasi *Multiclass*. klasifikasi multi class merupakan klasifikasi lebih dari dua kelas dimana setiap sampel dibebankan untuk satu label (Husin, 2023). Contoh *Confusion Matrix* dengan *Multiclass* yang akan digunakan dalam penelitian ini dapat dilihat pada Gambar 2.5.

		Predicted				
		1 Star	2 Star	3 Star	4 Star	5 Star
Actual	1 Star	TP	FN	FN	FN	FN
	2 Star	FP	TN	TN	TN	TN
	3 Star	FP	TN	TN	TN	TN
	4 Star	FP	TN	TN	TN	TN
	5 Star	FP	TN	TN	TN	TN

Gambar 2.5 *Confusion Matrix Multiclass* Pada Kelas 1 *Star*

Dari nilai yang terdapat pada gambar 2.5, yakni TP, FP, TN, dan FN, akan digunakan untuk menghitung nilai *Precision, Recall, dan F1-Score*. *Precision* merupakan persentase kejadian serangan yang diklasifikasikan dengan benar sebagai serangan, atau nilai prediksi positif, Sedangkan *Recall* merupakan efektivitas suatu model dalam mengidentifikasi suatu serangan (Bagui & Li, 2021). Nilai *Precision* setiap kelas akan diperoleh dari persamaan (1) dan nilai *Recall* setiap kelas akan diperoleh dari persamaan (2).

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Setelah nilai *Precision* dan *Recall* pada setiap kelas diperoleh, selanjutnya akan dicari nilai *F1-Score* pada setiap kelas. *F1-Score* merupakan nilai tengah yang diperoleh dari nilai *Precision* dan *Recall*. *F1-Score* berfungsi untuk menguji model pada data yang tidak seimbang (Rizki et al., 2023). nilai *Precision* dan *Recall* setiap kelas yang telah diperoleh akan digunakan untuk mencari nilai *F1-Score* pada masing-masing kelas menggunakan persamaan (3).

$$F1\ Score = 2 \times \frac{Recall \times Precission}{Recall + Precission} \tag{3}$$

Nilai *F1-Score* pada setiap kelas yang diperoleh dari persamaan (3) akan digunakan untuk mencari rata-rata dari nilai keseluruhan *F1-Score* menggunakan *Average Macro*. *F1-Score Average Macro* menjadi pilihan terbaik jika dataset tidak seimbang dan tanpa mempertimbangkan proporsi kelasnya karena memperlakukan semua kelas secara setara (Siregar et al., 2023). *F1-Score Average Macro* akan menggunakan persamaan (4).

$$Macro\ F1\ Score = \frac{\sum F1\ Score}{n} \tag{4}$$

dimana:

- $\sum$ F1 Score : jumlah seluruh nilai *F1-Score* per kelas
- n : jumlah kelas

Nilai yang di hasilkan dari evaluasi *F1-Score Average Macro* ini akan menjadi hasil evaluasi dari model *Wordnet* dalam klasifikasi sentimen dari ulasan aplikasi SIREKAP 2024. Evaluasi pada penelitian ini akan menggunakan sklearn.metrics di library *Scikit-learn*.

2.3.4 Klasifikasi *Decision Tree* CART

1. TF-IDF

Ekstraksi fitur merupakan suatu teknik untuk mengenali ciri-ciri yang berkaitan dengan klasifikasi. Terdapat dua jenis ekstraksi fitur, yakni melalui pendekatan *Machine Learning* dan metode *lexicon based* (Keerthi Kumar et al., 2019). Salah satu ekstraksi fitur *Machine Learning* yang sering digunakan yakni TF-IDF. TF-IDF adalah ekstraksi fitur yang bekerja dimana kata pada kalimat umum akan dihitung dengan bobot pada setiap kata (Pratomo et al., 2021). Tingkat kepentingan kata dalam dokumen atau kumpulan kata tertentu diukur dengan TF-IDF sebagai ukuran statistik.

TF-IDF menghitung dua faktor penting yakni *Term Frequency* (TF) dan *Inverse Document Frequency* (IDF). TF adalah metode untuk menilai seberapa sering frekuensi kata yang muncul. sedangkan IDF adalah metode untuk menilai seberapa penting kata dalam konteks koleksi dokumen yang lebih besar dengan membagi jumlah total dokumen dalam koleksi dan jumlah dokumen yang mengandung kata tersebut. Hasilnya kemudian diambil logaritma untuk meratakan skala (Septiani & Isabela, 2022).

Pada penelitian ini, ekstraksi fitur TF-IDF akan menggunakan sklearn.feature_extraction yang terdapat pada library *Scikit-learn*. Untuk menghitung ekstraksi TF-IDF digunakan persamaan (5).

$$B_{xy} = tf(x,y) \times idf(x) \tag{5}$$

dimana:

- B (x, y) : bobot term ke-x pada dokumen y
- tf (x,y) : frekuensi kemunculan term ke-x pada dokumen y

Kemudian untuk nilai IDF, didapatkan melalui persamaan berikut.

$$idf(x) = \log\frac{1+n}{1+dfx} + 1 \tag{6}$$

dimana:

- idf(x) : nilai idf term x
- dfx : jumlah dokumen dalam kumpulan dokumen yang mengandung term x
- n : jumlah dokumen

Setelah nilai TF-IDF dihasilkan, data akan di normalisasi untuk mengatur skala nilai data dalam rentang 0 sampai 1. Penelitian ini akan menggunakan *Euclidean norm* atau *L2-norm* dengan persamaan (7).

$$v_{norm} = \frac{v}{||v||_2} = \frac{v}{\sqrt{v_1^2 + v_2^2 + v_3^2 + \cdots + v_n^2}} \tag{7}$$

dimana:

- v : nilai tf-idf yang akan di normalisasi
- $||v||_2$: jumlah kuadrat nilai tf-idf dari semua dokumen

Berikut beberapa parameter ekstraksi fitur TF-IDF dalam library *Scikit-learn* yang akan digunakan pada penelitian ini.

a. ngram_range: Parameter yang akan digunakan untuk menentukan rentang panjang n-gram atau nilai n selama proses ekstraksi fitur, pengguna dapat mengatur nilai minimum dan maximum dari n-gram sesuai keinginan. Nilai yang akan digunakan dalam penelitian ini (1,1) dengan parameter ngram_range = (1,1), artinya hanya menghitung unigram.
b. norm: Parameter yang akan digunakan untuk menentukan jenis normalisasi pada hasil vektor. Nilai yang akan digunakan l2, menghitung jumlah kuadrat dari setiap elemen yang terletak pada vektor, mengambil akar kuadrat dari hasil penjumlahan, dan kemudian membagi nilai setiap elemen yang terletak pada vektor dengan nilai akar kuadrat tersebut akan membuat dokumen menjadi normal.

2. *Decision Tree* CART

Decision Tree melakukan klasifikasi dimana nilai dan atribut digambarkan berdasarkan *node* yang ada (Nurfauzan & Maharani, 2021). Algoritma *Decision Tree* yang dibuat untuk masalah klasifikasi dan regresi diwakili oleh metode CART. CART adalah sistem pengambilan keputusan berbasis pohon biner yang bercabang menjadi dua *node* anak di setiap persimpangan (Romadhonia et al., 2023).

CART digunakan untuk membagi objek ke dalam dua atau lebih kelompok. Dalam sebuah dataset yang memiliki p variabel bebas dan satu variabel terikat, jika variabel terikat tersebut merupakan kategori, CART akan membentuk pohon klasifikasi. Sebaliknya, jika variabel terikat tersebut bersifat kontinu atau numerik, CART akan membentuk pohon regresi (Bayu Setiawan & Sulistyo Nugroho, 2023). Pohon keputusan dengan algoritma CART dibangun melalui perhitungan indeks Gini untuk setiap kelas (Suryani et al., 2022), menggunakan rumus yang diberikan dalam persamaan 8.

$$Gini\ j(t) = 1 - \sum_{i=1}^{c} p_i^2\ (j|t) \tag{8}$$

dimana:

- c : jumlah kelas
- P_i (j|t) : probabilitas dalam *node* t yang termasuk kelas j

Dalam persamaan 8, *node* t dibagi menjadi dua subset, yakni *L* dan *R*, dengan ukuran masing-masing N_L dan N_R. Setelah itu, indeks Gini total untuk pembagian ini dihitung menggunakan subset tersebut, dan hasilnya dapat ditemukan dalam persamaan 9.

$$Gini\ split\ (t) = \frac{N_L}{N}\ giniL + \frac{N_R}{N}\ giniR \tag{9}$$

dimana:

- N_L : banyaknya data dalam subset N_L
- N_R : banyaknya data dalam subset N_R
- N : banyaknya data dalam sebuah variabel
- giniL : nilai indeks gini dari subset L untuk setiap variabel
- giniR : nilai indeks gini dari subset R untuk setiap variabel

Apabila jumlah sampel dalam suatu kelas adalah 5 atau kurang, *node* tersebut akan dijadikan sebagai *node* terminal (Bayu Setiawan & Sulistyo Nugroho, 2023). Penentuan label untuk *node* terminal ini dilakukan sesuai persamaan 10, dengan mempertimbangkan jumlah terbanyak.

$$P(j_0|t) = max_j P(j|t) + max_j \frac{m_j(t)}{m(t)} \quad (10)$$

Tahapan ini akan dilakukan proses klasifikasi dengan *Decision Tree* dengan algoritma CART menggunakan *DecisionTreeClassifier* yang ada pada library *Scikit-learn*. Beberapa parameter yang dibutuhkan adalah sebagai berikut.

a. *Criterion*: merupakan parameter yang digunakan untuk menentukan metode pengukuran kualitas split pada setiap *node* pohon keputusan. Dalam penelitian ini digunakan "*gini*" yang mengukur ketidakmurnian (impurity) dari kelas di dalam *node*.
b. *Max Depth*: merupakan parameter untuk menentukan kedalaman maximum dari pohon keputusan. Kedalaman maksimum akan diatur menjadi "*None*" agar Pohon akan terus tumbuh hingga setiap *leaf* adalah murni atau berisi kurang dari '*min_samples_split*'.
c. *Min Samples Split*: merupakan parameter untuk menentukan jumlah minimum sampel yang diperlukan untuk melakukan split pada *node* internal. Pada penelitian ini akan menggunakan 2 sampel pada suatu *node* untuk melakukan split.
d. *Min Samples Leaf*: merupakan parameter untuk menentukan jumlah minimun sampel yang diperlukan pada *leaf*. Penelitian ini akan menggunakan 1 sampel pada suatu *leaf*.
e. *Max Features*: merupakan parameter untuk menentukan jumlah fitur yang dipertimbangkan untuk melakukan split pada setiap *node*. Penelitian ini akan menggunakan semua fitur yang tersedia (*None*).
f. *Class Weight*: merupakan parameter untuk menentukan bobot yang diberikan pada setiap kelas. Untuk memulai penelitian ini, tidak ada penyesuaian bobot kelas yang dilakukan (*None*).

Dalam penggunaan *DecisionTreeClassifier*, yang pertama adalah mengimpor modul dari *Scikit-learn*. Modul ini termasuk *DecisionTreeClassifier* untuk membangun pohon keputusan dan parameter yang telah disebutkan diatas (Source code terlampir pada lampiran 27). Setelah mengimpor modul yang diperlukan, Selanjutnya adalah mengevaluasi menggunakan *F1-Score* dengan teknik *K-fold Cross-Validation* dengan k=10.

3. Evaluasi *F1-Score* CART

Evaluasi pada tahapan ini menggunakan *F1-Score* yang sama metode pencariannya dengan metode evaluasi *F1-Score Wordnet*. Namun, evaluasi *F1-Score* pada tahapan ini menggunakan *K-fold Cross-Validation*. *K-fold Cross-Validation* merupakan teknik validasi dimana kumpulan data independen akan digeneralisasi dengan statistik analisis untuk dinilai bagaimana hasilnya (Sari et al., 2023). Teknik statistik yang disebut *K-fold Cross-Validation* membagi data menjadi data train dan data test untuk menilai dan membandingkan algoritma pembelajaran (Nurnawati et al., 2023).

Teknik *K-fold Cross-Validation* berguna untuk mengevaluasi performa dari proses suatu algoritma dengan cara data sampel yang digunakan akan dibagi secara random dan kemudian akan dikelompokkan sebanyak nilai K yang digunakan. Salah satu kelebihan dari menggunakan *K-Fold Cross-Validation* dalam pengujian adalah kemampuannya untuk mengungkap model dengan akurasi tertinggi. Ini terjadi karena data sampel dibagi secara acak ke dalam K bagian, memungkinkan untuk melihat variasi model yang paling unggul dalam komposisi yang optimal (Nurainun et al., 2023). *K fold Cross Validation* pada penelitian ini akan menggunakan *Kfold* yang terdapat pada library *Scikit-learn*. Penggunaan *K-fold Cross-Validation* dengan nilai k=10 seperti Gambar 2.6.

Fold 1	Test	Train	Train	Train	Train	Train	Train	Train	Train	Train
Fold 2	Train	Test	Train	Train	Train	Train	Train	Train	Train	Train
Fold 3	Train	Train	Test	Train	Train	Train	Train	Train	Train	Train
Fold 4	Train	Train	Train	Test	Train	Train	Train	Train	Train	Train
Fold 5	Train	Train	Train	Train	Test	Train	Train	Train	Train	Train
Fold 6	Train	Train	Train	Train	Train	Test	Train	Train	Train	Train
Fold 7	Train	Train	Train	Train	Train	Train	Test	Train	Train	Train
Fold 8	Train	Train	Train	Train	Train	Train	Train	Test	Train	Train
Fold 9	Train	Train	Train	Train	Train	Train	Train	Train	Test	Train
Fold 10	Train	Train	Train	Train	Train	Train	Train	Train	Train	Test

Gambar 2.6 *K-fold Cross-Validation* K=10

Evaluasi *F1-Score* tahapan ini akan menggunakan sklearn.metrics yang terdpat pada library *Scikit-learn*, dengan parameter untuk setiap *fold* dan perhitungan rata-rata dijelaskan dibawah ini.

a. y_target: nilai sebenarnya dari sebenarnya dari target atau label dari klasifikasi
b. y_pred: nilai yang telah diprediksi oleh model klasifikasi
c. f1: fungsi untuk menilai *F1-Score* dari setiap *fold*