TUGAS AKHIR – KS184822

# KLASIFIKASI *TWEET* TERHADAP LAYANAN *CUSTOMER CARE* XL AXIATA DENGAN METODE *NAÏVE BAYES* DAN *RANDOM FOREST*

GEDE NARENDRA SAGUNA
NRP 062115 4000 0126

Dosen Pembimbing
Dr. Dra. Kartika Fithriasari, M.Si.

PROGRAM STUDI SARJANA
DEPARTEMEN STATISTIKA
FAKULTAS MATEMATIKA, KOMPUTASI, DAN SAINS DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2019

**TUGAS AKHIR – KS184822**

# KLASIFIKASI *TWEET* TERHADAP LAYANAN *CUSTOMER CARE* XL AXIATA DENGAN METODE *NAÏVE BAYES* DAN *RANDOM FOREST*

**GEDE NARENDRA SAGUNA**
**NRP 062115 4000 0126**

**Dosen Pembimbing**
**Dr. Dra. Kartika Fithriasari, M.Si.**

**PROGRAM STUDI SARJANA**
**DEPARTEMEN STATISTIKA**
**FAKULTAS MATEMATIKA, KOMPUTASI, DAN SAINS DATA**
**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**
**SURABAYA 2019**

**FINAL PROJECT – KS184822**

# TWEETS CLASSIFICATION OF XL AXIATA CUSTOMER CARE USING NAÏVE BAYES AND RANDOM FOREST

**GEDE NARENDRA SAGUNA**
**NRP 062115 4000 0126**

**Supervisor**
**Dr. Dra. Kartika Fithriasari, M.Si.**

**UNDERGRADUATE PROGRAMME**
**DEPARTMENT OF STATISTICS**
**FACULTY OF MATHEMATICS, COMPUTING, AND DATA SCIENCE**
**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**
**SURABAYA 2019**

# LEMBAR PENGESAHAN

# KLASIFIKASI *TWEET* TERHADAP LAYANAN *CUSTOMER CARE* XL AXIATA DENGAN METODE *NAÏVE BAYES* DAN *RANDOM FOREST*

**TUGAS AKHIR**
Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Statistika
pada
Program Studi Sarjana Departemen Statistika
Fakultas Matematika, Komputasi, dan Sains Data
Institut Teknologi Sepuluh Nopember

Oleh :
**Gede Narendra Saguna**
NRP. 062115 4000 0126

Disetujui oleh Pembimbing:
**Dr. Dra. Kartika Fithriasari, M.Si.** ( )
NIP. 19691212 199303 2 002

Mengetahui,
Kepala Departemen Statistika

**Dr. Suhartono**
NIP. 19710929 199512 1 001

SURABAYA, JULI 2019

*(Halaman ini sengaja dikosongkan)*

# KLASIFIKASI *TWEET* TERHADAP LAYANAN *CUSTOMER CARE* XL AXIATA DENGAN METODE *NAÏVE BAYES* DAN *RANDOM FOREST*


| | | |
|---|---|---|
| **Nama Mahasiswa** | **:** | **Gede Narendra Saguna** |
| **NRP** | **:** | **062115 4000 0126** |
| **Departemen** | **:** | **Statistika** |
| **Dosen Pembimbing** | **:** | **Dr. Dra. Kartika Fithriasari, M.Si.** |



## Abstrak

*Customer service merupakan layanan yang wajib dimiliki oleh setiap usaha. Layanan ini diharapkan cepat dan tepat dalam mengatasi permasalahan konsumen. Salah satu media yang dapat digunakan adalah Twitter. Dalam dunia telekomunikasi, perusahaan yang menerapkan hal ini adalah XL Axiata melalui akun @myXLCare. Pada akun tersebut pengguna XL dapat menyampaikan permasalah terkait produk XL yang digunakan. Tanpa adanya batasan waktu dan tempat membuat jumlah aduan dan pertanyaan yang disampaikan menjadi sangat banyak. Hal ini membutuhkan sesuatu yang cepat untuk mengetahui kategori dari aduan atau pertanyaan tersebut. Klasifikasi tweet merupakan metode yang dapat dilakukan untuk mengetahui kategori aduan secara cepat dari sebuah tweet. Metode yang digunakan dalam klasifikasi antara lain adalah Naïve Bayes dan Random Forest. Pada data tweet terdapat kasus imbalance yaitu ketidakseimbangan jumlah tweet antar kategori sehingga dibutuhkan metode untuk menangani hal ini. Synthetic Minority Oversampling Technique digunakan untuk mengatasi kasus imbalance pada data. Model Random Forest memiliki performa klasifikasi yang lebih baik dari pada Naïve Bayes. Penerapan SMOTE mampu meningkatkan performa klasifikasi Naïve Bayes dengan cukup tinggi. Random forest mampu menangani kasus imbalance dengan baik meskipun tanpa SMOTE.*



***Kata kunci:*** ***Naïve Bayes, Random Forest, SMOTE, Text Mining, Twitter***

*(Halaman ini sengaja dikosongkan)*

# TWEETS CLASSIFICATION OF XL AXIATA CUSTOMER CARE USING NAÏVE BAYES AND RANDOM FOREST


| | | |
|---|---|---|
| **Name** | **:** | **Gede Narendra Saguna** |
| **Student Number** | **:** | **062115 4000 0126** |
| **Department** | **:** | **Statistics** |
| **Supervisor** | **:** | **Dr. Dra. Kartika Fithriasari, M.Si.** |



## Abstract

*Customer service is a service that must be owned by every business. This service is expected to be fast and appropriate in overcoming consumer problems. One of the media that can be used is Twitter. Example of companies that implement this is XL Axiata through the @myXLCare account. Through this account, XL users can submit problems related to XL products used. Without the limitation of time and place, the number of complaints and questions submitted becomes very large. This requires something fast to find out the categories related to the complaint or question. Tweets classification is a method that can find out the category of the complaint quickly. The methods used in the classification are Naïve Bayes and Random Forest. In the tweet data, there are cases of imbalance in the number of tweets between categories so that a method is needed to handle this. Synthetic Minority Oversampling Technique is used to handle an imbalance case in data. The Random Forest model has a better classification performance than Naïve Bayes. The application of SMOTE can improve Naïve Bayes classification performance quite high. The Random Forest is able to handle imbalance data well even without SMOTE.*



***Keywords: Naïve Bayes, Random Forest, SMOTE, Text Mining, Twitter***

*(Halaman ini sengaja dikosongkan)*

# KATA PENGANTAR

Puji syukur penulis panjatkan kepada Ida Sang Hyang Widhi Wasa, Tuhan Yang Maha Esa atas limpahan rahmat-Nya sehingga penulis dapat menyelesaikan laporan Tugas Akhir yang berjudul "**Klasifikasi *Tweet* terhadap Layanan *Customer Care* XL Axiata dengan Metode *Naïve Bayes* dan *Random Forest***" dengan lancar.

Penulis menyadari bahwa Tugas Akhir ini dapat terselesaikan tidak terlepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Kedua orang tua, atas segala doa, nasihat, kasih sayang, dan dukungan yang diberikan kepada penulis demi kesuksesan dan kebahagiaan penulis.
2. Dr. Suhartono selaku Ketua Departemen Statistika dan Santi Wulan Purnami, M.Si., Ph.D. selaku Ketua Program Studi Sarjana yang telah memberikan fasilitas, sarana, dan prasarana.
3. Santi Wulan Purnami, M.Si., Ph.D. selaku dosen yang menjadi dosen wali selama masa studi yang telah banyak memberikan saran dan arahan dalam proses belajar di Departemen Statistika.
4. Dr. Dra. Kartika Fithriasari, M.Si. selaku dosen pembimbing yang telah meluangkan waktu dan dengan sangat sabar memberikan bimbingan, saran, dukungan serta motivasi selama penyusunan Tugas Akhir.
5. Dr.Irhamah, S.Si., M.Si. dan Pratnya Paramitha Oktaviana S.Si., M.Si. selaku dosen penguji yang selalu sabar dalam memberikan masukan dan saran dalam penyelesaian Tugas Akhir.
6. Seluruh dosen Statistika ITS yang telah memberikan ilmu dan pengetahuan yang tak ternilai harganya, serta segenap karyawan Departemen Statistika ITS.

7. Teman-teman Statistika ITS ∑26 angkatan 2015, yang selalu memberikan dukungan kepada penulis selama ini.
8. Semua teman, relasi dan berbagai pihak yang tidak bisa penulis sebutkan namanya satu persatu yang telah membantu dalam penulisan laporan ini.

Besar harapan penulis untuk mendapatkan kritik dan saran yang membangun sehingga Tugas Akhir ini dapat memberikan manfaat bagi semua pihak yang terkait.

Surabaya, Juli 2019

Penulis

# DAFTAR ISI

Halaman

# DAFTAR GAMBAR

Halaman

*(Halaman ini sengaja dikosongkan)*

# DAFTAR TABEL

Halaman

# DAFTAR LAMPIRAN

Halaman

*(Halaman ini sengaja dikosongkan)*

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Telekomunikasi adalah komunikasi jarak jauh melalui kawat (telegraf, telepon) dan radio (KBBI). Telekomunikasi merupakan teknologi yang menghilangkan jarak antar benua, antar negara, dan antar manusia (Huurdeman, 2003). Telekomunikasi berasal dari kata komunikasi, dalam Bahasa Latin *communication* yang berarti proses terjadinya pertukaran informasi yang mencakup kebutuhan manusia akan kontak langsung dan saling pengertian. Kata telekomunikasi diberi imbuhan *tele* berarti jarak, dibuat oleh Edouard Estaunie dalam buku yang berjudul *Traite pratique de telecommunication electrique*. Pada buku ini beliau mendefinisikan telekomunikasi sebagai pertukaran informasi sinyal listrik (Estaunie, 1904). *International Telecommunication Union* (ITU) pada tahun 1932 mendefinisikan telekomunikasi sebagai semua komunikasi melalui telegraf atau telepon berupa tanda, sinyal, tulisan, gambar, dan suara dalam bentuk apapun melalui kabel, radio, atau sistem lainnya, atau proses sinyal elektrik dan visual. Saat ini, ITU mendefinisikan telekomunikasi sebagai transmisi, emisi, atau penerimaan tanda, sinyal, tulisan, gambar, dan suara, atau kecerdasan apapun melalui kabel, radio, visual, atau sistem elektromagnetik lainnya.

Indonesia merupakan salah satu negara tempat perkembangan telekomukasi yang cepat. Berdasarkan data ITU, pada tahun 2000 pelanggan telepon seluler di Indonesia berjumlah sekitar 3,6 juta dan pada tahun 2012 pengguna telepon seluler menjadi 280 juta. Data tersebut menunjukkan perkembangan yang sangat signifikan. Perkembangan telekomunikasi di Indonesia diawali munculnya teknologi AMPS (*Advance Mobile Phone System*) pada tahun 1985 yang merupakan cikal bakal CDMA saat ini. Teknologi AMPS diusung 4 operator yaitu PT. Elektrindo Nusantara, PT. Centralindo Panca Sakti, PT. Telekomindo Prima Sakti, dan PT. Telkom. Satelit Palapa Indonesia (Satelindo)

kemudian muncul menjadi operator GSM pertama di Indonesia dengan awal kepemilikan saham oleh PT. Telkom, PT. Indosat, dan PT. Bimagraha Telekomindo. Kemudian pada 26 Mei 1995 didirikan Telkomsel sebagai operator GSM kedua di Indonesia. Selanjutnya berdirilah PT. XL Axiata sebagai operator GSM nasional ketiga. Selain berkembanganya operator telekomunikasi, perkembangan terhadap layanan dan jaringan juga terjadi. Diawali PT. Indosat Multi Media Mobile yang menjadi pelopor layanan GPRS (*General Packet Radio Service*) dan MMS (*Multimedia Messaging Service)* di Indonesia. Telkomsel juga menyajikan layanan serupa yaitu 2G (*Second Generation*). Selanjutnya juga berkembang layanan 3G hingga saat ini adanya layanan 4G (Ariansyah, 2014).

Seiring dengan perkembangan teknologi dalam bidang telekomunikasi, tentunya terdapat banyak permasalahan dan evaluasi terhadap sistem maupun kulaitas layanan yang diberikan penyedia layanan telekomunikasi terhadap pengguna layanan. Semakin meningkatnya teknologi dan kulitas jaringan, pengguna layanan tentunya mengharapkan pengalaman penggunaan layanan dengan lebih baik. Tidak sedikit ditemukan permasalahan-permasalahan yang justru membuat pengalaman dalam menggunakan layanan telekomunikasi kurang baik. Permasalahan seperti ini wajib dijadikan fokus untuk penyedia layanan demi menciptakan kualitas yang baik. Dalam hal ini, operator seluler biasanya menyediakan ruang untuk memberikan pertanyaan, kritik maupun saran dalam bentuk layanan *customer service*. Layanan ini biasanya dapat digunakan dengan mendatangi kantor terkait atau melakukan panggilan langsung kepada operator seluler terkait dengan menekan nomor atau kode tertentu. Melakukan hal seperti ini cukup membutuhkan waktu terlebih jika permasalahan yang dimiliki sangat mendesak dan harus segera diselesaikan. Dengan perkembangan internet dan media sosial yang sangat cepat, semakin banyak media yang dapat digunakan untuk menyampaikan pertanyaan, kritik, dan saran terhadap suatu layanan. Hal ini akan membuat penyampaian kritik dan saran lebih

menghamat waktu dan dapat terselesaikan dengan cepat. Salah satu media yang digunakan untuk menyampaikan kritik dan saran adalah *twitter*. *Twitter* memungkinkan pengguna layanan memberikan pertanyaan, kritik, dan saran tanpa harus mendatangi langsung kantor dari penyedia layanan. Pengguna layanan seluler hanya perlu memiliki akun dan menulis keluhan terhadap operator seluler yang digunakan.

Mudahnya cara memberikan pertanyaan, kritik dan saran terhadap operator membuat jumlah kritik dan saran tersebut sangat banyak. Dalam satu minggu jumlah aduan yang tersampaikan hingga ribuan. Dengan jumlah yang sangat banyak tentunya tidak mudah untuk melakukan pengelompokkan sehingga pertanyaan dapat dijawab dengan mudah dan efisien. Misalkan saja terdapat permasalahan yang sama ditanyakan berulang kali tentu lebih baik jika pertanyaan tersebut dikelompokkan dan dijawab dengan penjelasan yang sama. Untuk membuat pengelompokkan kata atau klasifikasi secara otomatis dapat dilakukan dengan pendekatan *text mining*. *Text Mining* merupakan proses ketika pengguna berinteraksi dengan kumpulan dokumen dari waktu ke waktu dengan menggunakan kumpulan analisis. *Text Mining* berguna untuk mengekstrak informasi dari data melalui identifikasi dan eksplorasi dari pola yang menarik. Dalam kasus *text mining*, sumber datanya adalah kumpulan dokumen dan pola yang menarik ditemukan pada teks yang tidak terstruktur dalam suatu dokumen (Feldman dan Sanger, 2007). Istilah *text mining* biasanya digunakan untuk menunjukkan sistem apapun yang menganalisis teks bahasa dalam jumlah besar dan mendeteksi pola leksikal atau linguistik dengan tujuan mengekstraksi tujuan yang mungkin berguna (Kumar, 2013). Menurut Prayoginingsih dan Kusumawardani (2018), pendekatan *text mining* dapat digunakan untuk menglasifikasikan data *twitter* pelanggan berdasarkan kategori myTelkomsel. Pada penelitian tersebut dilakukan klasifikasi teks dengan menggunakan metode *Support Vector Machine* dengan menggunakan kernel linier dan kernel RBF (*Radial Basis Function*). Menurut Vidya (2015), pendekatan *text*

*mining* dapat digunakan untuk menglasifikasikan sentimen dari operator seluler yaitu PT. Telkomsel Tbk, PT. XL Axiata Tbk, dan PT. Indosat Tbk. Klasifikasi dilakukan menggunakan *Naïve Bayes, Support Vector Machine,* dan *Decision Tree*.

Pada penelitian ini akan digunakan metode klasifikasi yaitu *Naïve Bayes* dan *Random Forest*. Metode *Naïve Bayes* dipilih karena merupakan metode yang popular untuk klasifikasi (Hastie, 2006) dan merupakan metode yang baik untuk klasifikasi dokumen karena sangat cepat dan cukup akurat (Witten, 2011). *Random Forest* digunakan karena memiliki beberapa keuntungan dan sering digunakan dalam kasus klasifikasi. *Random forest* pernah digunakan untuk melakukan klasifikasi terhadap data bidikmisi di Jawa Timur (Iriawan dkk, 2018). *Random Forest* tidak mengalami *overfit* atau *robust* terhadap *overfit* (Breiman, 2001). *Random Forest* sangat *user-friendly* dan mudah diaplikasikan karena hanya mempunyai dua parameter yaitu *number of trees* dan *number of variable* pada setiap *node* (Liaw dan Wiener, 2002). *Random forest* menghasilkan hasil yang lebih baik daripada CART pada kasus klasifikasi teks (Xin, 2016).

Ketersediaan data pada *twitter* tidak bisa ditentukan jumlahnya. Setiap orang memiliki permasalahan yang berbeda-beda yang menyebabkan jumlah *tweet* untuk masing-masing masalah atau kategori juga berbeda. Satu jenis permasalahan bisa terdiri dari jumlah *tweet* yang sangat banyak dan yang lainnya sangat sedikit. Kasus ini disebut dengan *imbalance*. Dengan adanya data yang mengalami *imbalance* akan menyebabkan kurang baiknya performa klasifikasi terutama untuk kelas atau kategori minoritas. Untuk mengatasi hal ini dapat dilakukan *sampling* terhadap data minoritas. Salah satu metode *sampling* yang sering digunakan adalah *Synthetic Minority Oversampling Technique* (SMOTE). SMOTE dilakukan dengan melakukan *over-sampling* pada kelas minoritas dengan membuat sampel sintetik (Chawla dkk, 2002). Prediksi kebangkrutan suatu perusahaan di Spanyol pernah dilakukan untuk mengetahui dampak dari penggunaan SMOTE terhadap data *imbalance.* Pada penelitian

tersebut diperoleh bahwa SMOTE mampu meningkatkan nilai G-mean dan nilai AUC untuk beberapa metode klasifikasi (Sisodia dan Verma, 2018).

Hasil dari klasifikasi dapat digunakan sebagai solusi untuk mengelompokkan data dalam jumlah besar dan untuk mengetahui jenis serta jumlah permasalahan yang muncul dalam bidang telekomunikasi khususnya pada XL Axiata secara *real time* dari waktu ke waktu sebagai bahan evaluasi untuk operator seluler.

## 1.2 Rumusan Masalah

Permasalahan yang terjadi dalam sistem layanan aduan pelanggan (*customer service*) adalah banyaknya *tweet* seputar aduan dan pertanyaan yang disampaikan oleh pengguna layanan terhadap penyedia layanan memerlukan pengelompokkan yang cepat dan tepat. Selain itu penggunaan *Twitter* juga membuat proses penyampaian tersebut menjadi bebas dan rentan akan penggunaan kata yang kurang baik sehingga diperlukan suatu cara untuk dapat menemukan inti dari pesan yang disampaikan. Untuk membuat kemudahan dalam mengetahui informasi apa yang diberikan secara cepat dan tepat akan dilakukan analisis klasifikasi. Metode yang digunakan antara lain *Naïve Bayes* dan *Random forest*. Untuk menangani kasus *imbalance* dilakukan *oversampling* dengan menggunakan *Synthetic Minority Oversampling Technique* (SMOTE). Penggunaan model tersebut akan dibandingkan untuk mengetahui model terbaik.

## 1.3 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut.

1. Mengetahui karakteristik data *tweet* terhadap Layanan *Customer Care* XL Axiata melalui akun @myXLCare.
2. Mengetahui model klasifikasi terbaik untuk data *tweet* terhadap Layanan *Customer Care* XL Axiata melalui akun @myXLCare.

## 1.4 Manfaat Penelitian

Manfaat dari penelitian ini dapat dibagi menjadi dua sisi, yakni bagi operator seluler dan peneliti.

1. Bagi Operator Seluler
   Manfaat yang ingin diberikan melalui penelitian ini adalah mengklasifikasikan pertanyaan dan aduan terhadap operator seluler sehingga mudah dalam mengelompokkan banyaknya pertanyaan yang masuk. Selain itu juga memberikan informasi secara *realtime* terhadap jumlah dan jenis pertanyaan maupun aduan yang ditujukkan kepada operator seluler sebagai bahan evaluasi dalam meningkatkan kualitas layanan.
2. Bagi Peneliti
   Manfaat yang diperoleh bagi peneliti adalah dapat terlibat dalam menangani permasalahan dalam dunia telekomunikasi. Selain itu, dapat memberikan penyelesaian secara statistik dalam permasalahan telekomunikasi.

## 1.5 Batasan Masalah

Batasan masalah yang digunakan dalam penelitian ini yaitu data yang digunakan dalam penelitian ini adalah data yang diperoleh dari *Twitter API* dengan pencarian yang ditujukkan terhadap operator seluler XL Axiata melalui pencarian "@myXLCare" pada tanggal 8 Februari 2019 – 14 Maret 2019. Metode klasifikasi yang digunakan adalah *Random Forest* dan *Naïve Bayes*.

# BAB II
# TINJAUAN PUSTAKA

Bab ini menjelaskan mengenai *text mining*, klasifikasi Naïve Bayes, *Random Forest*, evaluasi model, *k-fold cross validation*, dan *Synthetic Minority Oversampling Technique* (SMOTE).

## 2.1 *Text Mining*

*Text Mining* merupakan proses ketika pengguna berinteraksi dengan kumpulan dokumen dari waktu ke waktu dengan menggunakan kumpulan analisis. *Text Mining* berguna untuk mengekstrak informasi dari data melalui identifikasi dan eksplorasi dari pola yang menarik. Dalam kasus *text mining*, sumber datanya adalah kumpulan dokumen dan pola yang menarik ditemukan pada teks yang tidak terstruktur dalam suatu dokumen (Feldman dan Sanger, 2007). *Text Mining* didefinisikan sebagai ekstrasi non trivia dari informasi yang tersembunyi, sebelumnya belum diketahui, dan berguna dari data berupa teks yang sangat banyak (Waegel, 2006). *Text Mining* biasanya berupa proses tentang struktur penginputan teks, mencari pola dari data teks yang telah terstruktur, dan evaluasi final serta interpretasi dari output (Kumar, 2013). *Text Mining* merupakan cara memperoleh informasi dari sekumpulan dokumen yang tidak terstruktur.

## 2.2 Text Preprocessing

*Text preprocessing* adalah sebuah proses yang penting dari NLP (*Natural Language Processing*) karena karakter, kata, dan kalimat yang diidentifikasi pada tahap ini adalah unit dasar yang diteruskan ke semua tahap pemrosesan lebih lanjut . Tahapan pra-proses dalam *text mining* merupakan tahapan yang penting dalam menggali informasi dari sebuah dokumen. Operasi *text mining* yang efektif didasarkan pada metodologi pemrosesan data yang canggih (Feldman dan Sanger, 2007). Penggunaan *text preprocessing* yang tepat dapat meningkatkan akurasi pada kasus klasifikasi. Menurut Hidayatullah (2016) penggunaan *stopword removal* dapat meningkatkan akurasi terhadap kasus klasifikasi *tweet* berbahasa Indonesia. Selain itu penggunaan *tokenization* dan

*feature selection* dapat meningkatkan hasil klasifikasi pada kasus *Twitter sentiment analysis* dengan menggunakan metode *machine learning* (Krouska, dkk, 2016).

Dalam melakukan *text preprocessing* terdapat beberapa tahap yang dapat dilakukan. Tahapan-tahapan *text preprocessing* secara umum adalah *removing symbols, removing numbers, removing ASCII string, punctuation, tokenization, case folding, stemming,* dan *stopword removal*.

1. *Removing symbol, number, ASCII strings, and punctuation*, merupakan proses penghapusan simbol, nomor, dan tanda baca lainnya dalam *tweet*. *Tweet* mengandung banyak sekali simbol dan tanda baca. Simbol dan tanda baca yang dihapus adalah seperti "# $ % & \ ‘ ( ) * + , - . / : ; < = > ? @ [ \ \ ] ^ _ ` { | } ~ (Jannah, dkk, 2018).
2. *Tokenization*, merupakan proses pemisahan kalimat menjadi kata, frase, symbol, dan elemen lainnya yang memiliki arti atau disebut token. Dalam *tokenization* terdapat metode yang disebut *N-gram*. *N-gram* merupakan potongan n-karakter dari sebuah kalimat (Cavnar, 1994). *N-gram* merupakan metode paling sederhana untuk menetapkan urutan kata atau probabilitas kata yang akan muncul setelah satu kata (Daniel dan James, 2014). *N-gram* biasa digunakan dalam pemrosesan bahasa dan perkataan. Contoh penggunaan *n-gram* adalah misalkan dari kalimat "gangguan internet sinyal jelek" dapat ditunjukkan dengan *2-gram* yaitu "gangguan internet", "internet sinyal", dan "sinyal jelek".
3. *Case folding,* merupakan proses untuk mengubah kata ke dalam format yang sama, dalam hal ini yaitu menjadi format *lowercase* atau *uppercase* (Hidayatullah, 2016). Pada penilitian ini akan *case folding* akan dilakukan dengan mengubah ke format *lowercase*.
4. *Stemming,* merupakan proses untuk menemukan kata dasar dari sebuah kata (Tala, 2003). Sistem kerja tahap *stemming* ini adalah menghilangkan awalan, akhiran, sisipan, dan

*confixes* (kombinasi dari awalan dan akhiran) (Ariadi & Fithriasari, 2015). *Stemming* digunakan agar suatu kata sesuai dengan kaidah Bahasa Indonesia yang benar. Tanpa melakukan *stemming* akan terdapat banyak kata berbeda yang bermakna hampir sama. Hal ini dikarenakan dalam Bahasa Indonesia terdapat banyak sisipan kata.

5. *Stopword removal*, merupakan proses menghapus kata-kata umum dan sering muncul tetapi tidak memiliki pengaruh yang signifikan terhadap makna dari sebuah kalimat. Penggunaan *stopword removal* berguna untuk mengurangi korpus atau jumlah kata tanpa mengurangi informasi dalam kalimat (Krouska dkk, 2016). Beberapa contoh *stopword* dalam Bahasa Indonesia adalah penggunaan kata "dan", "atau", "yang", "itu", dan lainnya.

### 2.2.1 *Confix Striping Stemmer*

*Confix Striping Stemmer* atau disebut CS merupakan pendekatan untuk melakukan *stemming* terhadap Bahasa Indonesia. Berkut merupakan urutan penggunaan afiks, dengan tanda kurung siku berarti bahwa afiks tersebut optional (Adriani dkk, 2007).

[[[DP+]DP+]DP+] kata dasar [[+DS][+PP][+P]]

dengan DP (*Derivational Prefixes*) merupakan awalan, DS (*Derivational Suffixes*) merupakan akhiran, PP (*Possessive Pronouns*) merupakan kata ganti kepunyaan, dan P (*Particles*) merupakan partikel.

Penggunaan urutan tersebut digunakan berdasarkan peraturan dasar tentang Bahasa Indonesia yaitu sebagai berikut.

1. Kata yang mengandung kurang dari 3 karakter tidak bias menganduk afiks. Sehingga, tidak ada *stemming* yang dilakukan pada kata tersebut.
2. Afiks tidak pernah berulang

Langkah-langkah dalam melakukan *stemming* dengan CS adalah sebagai berikut (Adriani dkk, 2007).

1. Pada awal pemrosesan dan pada setiap langkah, dilakukan pemeriksaan kata pada kamus kata dasar. Jika kata tersebut ditemukan, maka dianggap sebagai kata dasar dan seluruh proses dihentikan.
2. Menghilangkan *inflectional suffixes* yang dimulai dari *inflec-tional particle* (’-kah’, ’-lah’, ’-tah’, ’-pun’) dan dilanjutkan menghilangkan *possessive pronoun* (’-ku’, ’-mu’, ’-nya’). Contohnya kata ”bajumulah” akan dipotong menjadi ”baju-mu” dan kemudian ”baju”, dimana kata ini sudah merupakan kata dasar sehingga proses berhenti.
3. Menghilangkan *derivational suffixes* (’-i’, ’-kan’, ’-an’). Contohnya kata ”membelikan” akan dipotong menjadi ”mem-beli”, namun karena kata ini bukanlah kata dasar maka proses dilanjutkan ke langkah selanjutnya.
4. Menghilangkan *derivational prefixes* (’be-’, ’di-’, ’ke-’, ’se-’, ’me-’, ’te-’, ’pe-’).
   a. Proses berhenti jika:
      — Awalan yang teridentifikasi berpasangan dengan akhiran terlarang yang telah dihilangkan pada langkah 3.
      — Awalan yang dideteksi saat ini sama dengan awalan yang telah dihilangkan sebelumnya.
      — Tiga awalan telah dihilangkan.
   b. Identifikasi tipe awalan kemudian hilangkan. Awalan terdiri dari dua tipe berikut ini.
      — Standar (’di-’, ’ke-’, ’se-’) dapat dihilangkan langsung dari kata.
      — Kompleks (’be-’, ’te-’, ’me-’, ’pe-’) dapat bermorfologi sesuai kata dasar yang mengikutinya (dapat mengubah bentuk asli kata dasar).
   c. Mencari kata yang telah dihilangkan awalannya dalam kamus kata dasar. Apabila pencarian tidak ditemukan maka langkah 4 diulang kembali, sedangkan apabila ditemukan maka keseluruhan proses dihentikan.

5. Apabila hingga langkah 4 kata dasar masih belum ditemukan, maka dilakukan proses *recoding*, yaitu menambah atau meng-ganti huruf awal dari kata yang terpenggal pada proses *stem-ming*. Contohnya kata ”menangkap” dihilangkan awalan ”me-” sehingga tersisa ”nangkap”. Kata ”nangkap” bukanlah kata dasar yang valid sehingga dilakukan *recoding* menjadi kata ”tangkap”.
6. Apabila semua langkah tidak berhasil, maka *input* kata diang-gap sebagai kata dasar dan algoritma akan mengembalikan kata seperti semula.

### 2.2.2 *Term Weighting*

Analisis terhadap teks dengan klasifikasi tidak dapat dilakukan secara langsung dengan bentuk data yang ada yaitu berupa teks. Perlu dilakukan tahap pra-proses untuk mengubah dokumen ke dalam bentuk yang data dianalisis. Dokumen biasanya diubah ke dalam bentuk *feature vector*. *Feature* dalam hal ini adalah kumpulan kata-kata yang terdapat dalam dokumen Metode yang paling sederhana adalah dengan cara biner yaitu bobot *feature* bernilai satu ketika suatu kata terdapat dalam dokumen dan bernilai nol untuk sebaliknya (Feldman dan Sanger, 2007). Cara yang lebih kompleks adalah dengan menggunakan *Term Frequency-Inverse Document Frequency* (TF-IDF).

*Term Frequency* (TF) merupakan pendekatan paling sederhana untuk menghitung bobot suatu *term* dimana bobot sama dengan jumlah kemunculan sebuah *term* dalam suatu dokumen. Ketika hanya menggunakan *term frequency* akan terdapat permasalahan yaitu suatu *term* akan memiliki tingkat kepentingan yang sama. Misalkan terdapat *term* yang muncul dalam semua dokumen dengan *term* yang hanya muncul dalam beberapa dokumen. Dengan permasalahan ini maka diperoleh cara untuk menurunkan bobot *term* yang memiliki frekuensi tinggi pada keseluruhan dokumen. Cara ini disebut *inverse document frequency* (IDF) dengan ruus sebagai berikut (Manning dkk, 2007).

$$\text{idf}_t = \log \frac{N}{\text{df}_t} \tag{2.1}$$

dengan,

$N$ : jumlah dokumen

$\text{df}_t$ : jumlah dokumen yang mengandung kata ke-$t$, $t = 1, 2, \ldots, T$

T : jumlah kata pada keseluruhan dokumen

Dalam hal ini $N$ merupakan jumlah kumpulan dokumen, dalam hal ini jumlah keseluruhan *tweet* dan $\text{df}_t$ merupakan jumlah dokumen atau jumlah *tweet* yang mengandung *term t*. Kedua cara tersebut dikombinasikan untuk memperoleh bobot masing-masing *term* pada setiap *tweet*. *Tf-idf* (*term frequency-inverse document frequency*) untuk *term t* pada dokumen *d* dirumuskan sebagai berikut (Manning dkk, 2007).

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t \tag{2.2}$$

dengan,

$\text{tf}_{t,d}$ : jumlah kata ke-$t$ yang muncul pada dokumen ke-$d$, dengan $d = 1, 2, \ldots, N$

$\text{idf}_t$ : *inverse document frequency* untuk kata ke-$t$

## 2.3 *Text Classification*

Klasifikasi teks telah dipelajari secara luas di berbagai komunitas seperti *data mining*, *database*, *machine learning* dan *information retrieval*, dan digunakan dalam sejumlah besar aplikasi di berbagai domain seperti pemrosesan gambar, diagnosis medis, organisasi dokumen, dll. Klasifikasi teks bertujuan untuk menetapkan kelas yang telah ditentukan untuk dokumen teks (Mitchell, 1997).

## 2.4 Naïve Bayes

Klasifikasi *Bayesian* merupakan klasifikasi statistika digunakan untuk memprediksi probabilitas keanggotaan terhadap suatu kelas (Han, 2012). Salah satu metode klasifikasi *Bayesian* adalah *Naïve Bayes*. Klasifikasi *Naïve Bayes* memiliki asumsi bahwa pengaruh nilai atribut pada kelas yang diberikan tidak tergantung pada nilai atribut lainnya. Teorema *Bayesian* adalah sebagai berikut.

$$P(H \mid X) = \frac{P(X \mid H)P(H)}{P(X)} \tag{2.3}$$

dengan,

*P(H|X)* : probabilitas posterior suatu kejadian *H* pada kondisi *X*
*P(H)* : *prior probability* terhadap kejadian *H*
*P(X|H)* : probabilitas *X* terhadap kondisi *H*

*Naïve Bayes* yang digunakan pada klasifikasi teks adalah model *multinomial Naïve Bayes*. Probabilitas dokumen *d* berada pada kelas *c* dihitung sebagai berikut.

$$P(c \mid d) = P(c)\prod_{t=1}^{T} P(w_t \mid c) \tag{2.4}$$

dengan,

*P(c|d)* : probabilitas dokumen *d* berada pada kelas *c*
$P(w_t|c)$ : probabilitas dari kata-kata ($w_t$) muncul dalam dokumen pada kelas *c*
*P(c)* : probabilitas dari kelas *c*
$w_t$ : $\langle w_1, w_2, ..., w_T \rangle$, kata ke *t*

Pada kasus klasifikasi teks tujuannya adalah mencari kelas terbaik dari dokumen. Oleh karena itu, setelah dilakukan estimasi *P(c|d)* yang menunjukkan probabilitas dokumen *d* berada pada kelas *c* maka dilakukan pemilihan kelas terbaik dengan *maximum a-posteriori* (MAP) atau $c_{map}$.

$$c_{map} = \arg\max_{c_j \in \mathbb{C}} P(c_j \mid d) = \arg\max_{c_j \in \mathbb{C}} \hat{P}(c_j)\prod_{t=1}^{T} \hat{P}(\mathrm{w}_t \mid c_j) \tag{2.5}$$

$$c_{map} = \arg\max_{c_j \in \mathbb{C}} \left[ \log \hat{P}(c_j) + \prod_{t=1}^{T} \log \hat{P}(\mathrm{w}_t \mid c_j) \right] \tag{2.6}$$

Estimasi parameter $\hat{P}(c_j)$ adalah sebagai berikut.

$$\hat{P}(c_j) = \frac{N_j}{N} \tag{2.7}$$

dengan,

$c_j$ : kelas ke-*j*, $j = 1, 2, \ldots, L$

L : jumlah kelas
$N_j$ : jumlah dokumen pada kelas $c_j$
$N$ : total dokumen

Estimasi parameter $\hat{P}(w_t \mid c_j)$ adalah sebagai berikut.

$$\hat{P}(w_t \mid c_j) = \frac{\mathrm{tf}_{t,j}}{\sum_{t=1}^{T} \mathrm{tf}_{t,j}} \tag{2.8}$$

$$\hat{P}(w_t \mid c_j) = \frac{\mathrm{tf}_{t,j} + 1}{\sum_{t=1}^{T} \left(\mathrm{tf}_{t,j} + 1\right)} = \frac{\mathrm{tf}_{t,j} + 1}{\sum_{t=1}^{T} \left(\mathrm{tf}_{t,j}\right) + \mathrm{T}} \tag{2.9}$$

dengan,
$\mathrm{tf}_{t,j}$ : jumlah kemunculan kata ke-*t* pada kelas ke-*j* dalam data, termasuk kemunculan berganda suatu kata dalam dokumen
T : jumlah kata pada keseluruhan dokumen

## 2.5 *Random Forest*

*Random forest* merupakan metode klasifiaksi yang berdasarakan oleh pohon keputusan atau *decision tree*. Dalam *random forest* terdapat banyak pohon keputusan yang terbentuk sehingga disebut dengan *forest*. Masing-masing pohon keputusan dibuat dengan pemilihan variabel secara random untuk melakukan *split* pada masing-masing cabangnya. Dengan kata lain, setiap pohon tergantung pada nilai vektor acak yang diambil secara independen dan berdistribusi sama untuk semua pohon yang ada pada *random forest*. Pada pengklasifikasian masing-masing pohon dipilih yang terbaik untuk menjadi hasilnya (Breiman, 2001).

*Random forest* adalah modifikasi substansi dari metode *bagging* yang membangung banyak pohon yang tidak saling berkorelasi, dan kemudian pohon tersebut di rata-rata (Breiman, 2001). Dengan adanya pemilihan variabel secara acak dalam mebangun setiap pohonnya dan dengan banyaknya jumlah pohon yang dibangun pada *random forest* maka ini memungkinkan untuk membuat metode *random forest* bisa menghindari terjadinya

*overfitting* pada pohon keputusan. Langkah-langkah *random forest* adalah sebagai berikut.

1. Untuk b = 1 hingga B dengan B merupakan *ntree* atau jumlah pohon yang akan dibentuk pada *random forest* :
   a. Tentukan sampel *Z* dari *N* data *training*.
   b. Buat pohon $D_b$ dengan data *bootstrapped* dengan melakukan perulangan sesuai langkah untuk setiap *terminal node*.
      i. Pilih *m* variabel secara acak diantara *T* variabel.
      ii. Pilih variabel terbaik sebagai *split point.*
      iii. Pecah *node* menjadi 2 *node* baru.
2. *Output* dari kumpulan pohon $\{D_b\}_1^B$.

*Random forest* dapat digunakan pada kasus klasifikasi dan regresi. Ketika digunakan untuk klasifikasi, *random forest* mendapatkan nilai pilihan kelas untuk masing-masing pohon dan kemudian mengklasifikasikan menggunakan pilihan mayoritas. Dalam kasus regresi, prediksi dari setiap pohon pada target $x$ secara sederhana di rata-rata.

$$\textit{Classification :}\ \hat{K}_{rf}^{B}(x) = \textit{majority vote}\ \left\{K_b(x)\right\}_1^B \qquad (2.10)$$

dengan,

$\hat{K}_{rf}^{B}(x)$ : prediksi *random forest* untuk kasus klasifikasi

$B$ : jumlah pohon yang terbentuk pada *random forest*

$K_b(x)$ : prediksi kelas untuk pohon ke-*b*

*Majority vote* merupakan cara untuk menentukan kelas dengan menggunakan pilihan mayoritas terhadap kelas yang muncul. Kelas yang muncul lebih banyak akan terpilih menjadi kelas prediksi.

Ketika sampel *bootstrap* diambil dari data, terdapat beberapa pengamatan yang tidak termasuk ke dalam sampel *bootstrap*. Hal ini dinamakan "*out-of-bag data*" dan berguna untuk mengestimasi *generalization error* dan *variable importance*. Penggunaan *out-of-bag error* adalah untuk melakukan validasi terhadap pohon yang terbentuk. Hal ini kurang lebih sama dengan penggunaan *K-fold cross validation* (Hastie, 2009).

Dalam menentukan variabel terbaik untuk melakukan *split point* digunakan nilai *gini impurity*. Untuk menghitung *gini impurity* dilakukan dengan cara sebagai berikut.

$$Gini(N) = 1 - \sum_{j=1}^{L} P_j^2 \tag{2.11}$$

$$P_j = \frac{N_j}{N} \tag{2.12}$$

dengan,
$P_j$ : probabilitas kelas ke-*j*
$N_j$ : jumlah dokumen pada kelas ke-*j*
$N$ : jumlah keseluruhan dokumen

Ketika terjadi *binary split* maka untuk menghitung nilai *gini impurity* adalah dengan menghitung pembobotan pada masing-masing partisi. Misalkan terjadi *binary split* pada variabel A yang membagi $N$ ke dalam $N_1$ dan $N_2$ maka untuk menghitung *gini impurity* D adalah sebagai berikut.

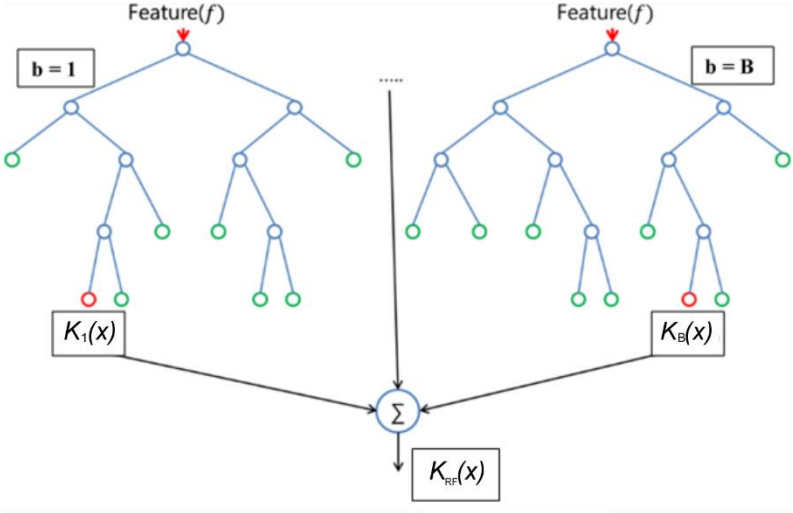
$$Gini_A(N) = \frac{|N_1|}{|N|} Gini(N_1) + \frac{|N_2|}{|N|} Gini(N_2) \tag{2.13}$$

Salah satu cara menentukan variabel terbaik untuk *splitting* adalah berdasarkan selisih atau reduksi kemurnian *gini index* yang dapat dihitung dengan persamaan sebagai berikut.

$$\Delta Gini(A) = Gini(N) - Gini_A(N) \tag{2.14}$$

Semakin tinggi reduksi atau selisih kemurnian *gini index* maka variabel tersebut semakin baik untuk menjadi variabel *splitting*. Dengan kata lain semakin rendah *gini impurity* maka semakin baik variabel digunakan untuk variabel *splitting*.

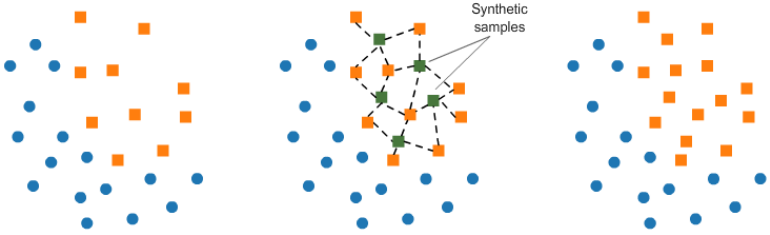
Saat melakukan *random forest*, ukuran contoh peubah penjelas (*m*) yang digunakan sangat mempengaruhi korelasi dan kekuatan tersebut. Meningkatkan m akan meningkatkan keduanya, begitu juga sebaliknya. Penggunaan *m* yang tepat akan menghasilkan *random forest* dengan korelasi antar pohon cukup kecil namun kekuatan setiap pohon cukup besar (Breiman 2001). Algoritma *random forest* ditunjukkan oleh Gambar 2.1.




**Gambar 2.1** Algoritma *Random Forest*
(Sumber: analyticsvidhya.com)

## 2.6 *Synthetic Minority Over-sampling Technique* (SMOTE)

SMOTE (Synthetic Minority Oversampling Technique) merupakan salah satu metode yang dapat digunakan untuk menangani data *imbalance* (Chawla dkk, 2002). SMOTE dilakukan dengan menambah jumlah sampel pada kelas minor agar setara dengan kelas mayor dengan cara membangkitkan data sintetik berdasarkan tetangga terdekat *k-nearest neighbour* dimana tetangga terdekat dipilih berdasarkan jarak *euclidean* antara kedua data (Chawla dkk, 2002).




**Gambar 2.2** Ilustrasi Prosedur SMOTE
(Sumber: Kaggle.com)

Misalkan diberikan data dengan *r* variabel yaitu $\mathbf{x}^T = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_r]$ dan $\mathbf{z}^T = [\mathrm{z}_1, \mathrm{z}_2, \ldots, \mathrm{z}_r]$ maka jarak euclidean *d*(x,z) secara umum sebagai berikut:

$$d(\mathrm{x},\mathrm{z}) = \sqrt{(\mathrm{x}_1 - \mathrm{z}_1)^2 + (\mathrm{x}_2 - \mathrm{z}_2)^2 + \ldots + (\mathrm{x}_r - \mathrm{z}_r)^2} \quad (2.15)$$

Pembangkitan data synthetic dilakukan dengan menggunakan persamaan berikut:

$$\mathbf{x}_{syn} = \mathbf{x}_i + (\mathbf{x}_{knn} - \mathbf{x}_i)\gamma \quad (2.16)$$

Keterangan:

$\mathbf{x}_{syn}$ : data *synthetic*

$\mathbf{x}_i$ : data ke-i dari kelas minor

$\mathbf{x}_{knn}$ : data dengan jarak terdekat dari data yang akan direplikasi

$\gamma$ : bilangan random antara 0 – 1

Algortima SMOTE menurut Nithes V. Chawla (2002) ditulis dalam *pseudocode* sebagai berikut.

**Algoritma** *SMOTE* (T, N, k)
**Input :** T (jumlah sampel minoritas); N (persentase SMOTE); *k*-tetangga terdekat
**Output :** (*N*/100) * T sampel sintetis kelas minoritas

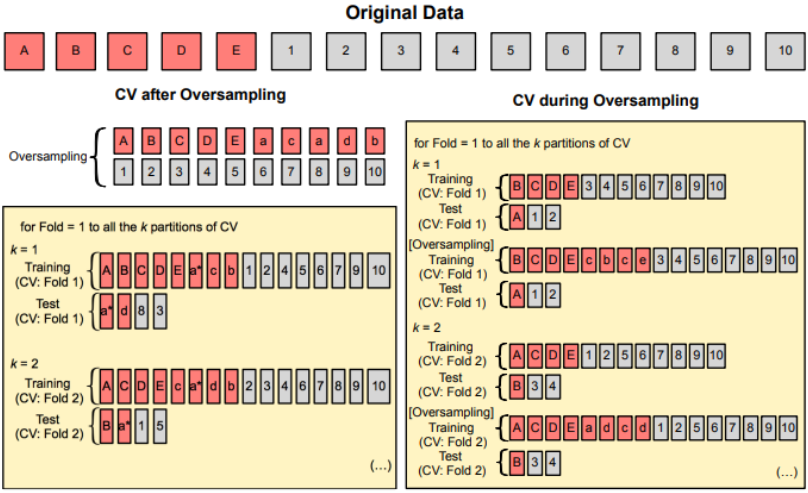
1. (* *Apabila N kurang dari 100%, randomisasi sampel kelas minoritas akan di SMOTE*)
2. if *N* < 100
3. **then** Randomisasi *T* sampel minoritas
4. T = (*N*/100) * T
5. N = 100
6. **endif**
7. N = (*int*)(*N*/100)(* *jumlah SMOTE diasumsikan integer dari perhitungan 100*)
8. *k* = jumlah dari tetangga terdekat
9. *numattrs* = jumlah atribut
10. *Sampel* [ ][ ] = array dari sampel kelas minoritas awal
11. *newindex* = jumlah dari sampel sintetis yang dibangkitkan, diawali dengan 0
12. *Synthetic*[ ][ ] = array dari sampel sintetis

(* *menghitung k tetangga terdekat untuk setiap sampel kelas minoritas*)
13. **for** *i* to *T*
14. Hitung k tetangga yang berdekatan pada *i* dan simpan indexnya ke *nnarray*
15. Populate(N, *i*, *nnarray*)
16. **endfor**

*Populate (N, i, narray) (* fungsi untuk membangkitkan sampel sintetis. *)*
17. **while** *N* <> 0

18. Pilih nomor random antar 1 dan *k*, sebut saja nn, langkah ini memilih satu dari *k* tetangga yang berdekatan dengan *i*.
19. **for** *aatr* 1 to *numattrs*
20. Hitung: *dif = Sampel(nnarray[nn][aatr]-Sampel[i][aatr]*
21. Hitung: *gap* = nomor acak antara 0 sampai dengan 1
22. *Sintetis [newindex][aatr]=Sampel[i][aatr]+gap*dif*
23. **endfor**
24. *newindex++*
25. *N = N* - 1
26. **endwhile**
27. **return**(**End of populate**)
End of Pseudo-Code

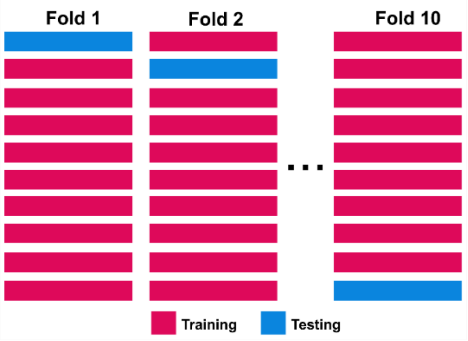
Pada prosedur *k-fold cross validation*, SMOTE dilakukan dengan memasukkan SMOTE ke dalam *k-fold cross validation* dimana *oversampling* dilakukan pada masing-masing data *training* untuk setiap *fold*. Apabila SMOTE dilakukan pada sebelum prosedur *k-fold cross validation* maka akan terjadi *overoptimistic* yaitu hasil klasifikasi yang terlalu baik karena adanya kemungkinan terdapat data dengan pola yang sama antara data *training* dan *testing* pada prosedur *k-fold cross validation* yang dilakukan (Santos dkk, 2018).




**Gambar 2.3** Ilustrasi Prosedur k-*fold cross validation* dengan SMOTE. (Sumber: Santos dkk, 2018)

## 2.7 *K-Fold Cross Validation*

*K-Fold Cross Validation* dilakukan dengan membagi data ke dalam *K* bagian dengan masing-masing bagian terdiri dari jumlah yang sama. Model dibuat dengan *K-1* bagian dari data dan akurasi dihitung dengan melakukan penyesuaian terhadap prediksi dari data ke K. Hal ini dilakukan sebanyak $k = 1, 2, .., K$ dan mengandung estimasi K akurasi (Hastie, dkk, 2009).



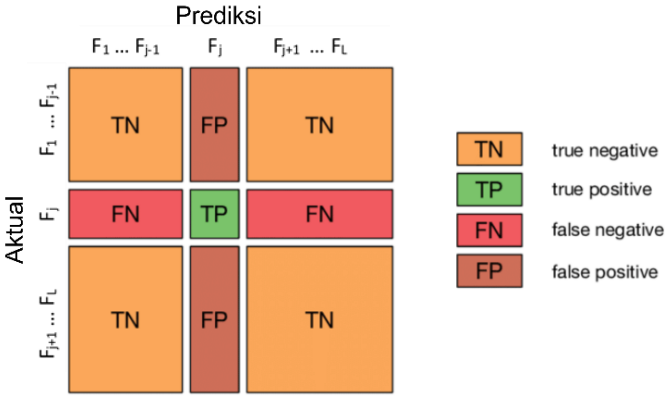

**Gambar 2.4** Pembagian Data untuk 5-*Fold Cross Validation*

Rata-rata evaluasi model pada keseluruhan *fold* dilakukan untuk mengetahui kebaikan model secara keseluruhan. *K-fold* digunakan untuk melihat kestabilan model dalam melakukan klasifikasi untuk data yang beragam. Dengan *K-fold* akan diperoleh kombinasi data *training* dan data *testing* yang berbeda-beda.

## 2.8 Evaluasi Model

*Confusion matrix* digunakan untuk mengetahui seberapa baik sebuah model dalam membuat prediksi klasifikasi. Beberapa ukuran dalam mengevaluasi model adalah akurasi, sensitifitas, dan spesifisitas. Dalam evaluasi model dikenal beberapa istilah sebagai berikut (Han, 2012).

- *True positives* (TP): menunjukkan *tuple* positif yang benar di klasifikasi.
- *True negatives* (TN): menunjukkan *tuple* negatif yang benar di klasifikasi.
- *False positives* (FP): menunjukkan *tuple* positif yang salah di klasifikasi.
- *False negatives* (FN): menunjukkan *tuple* negatif yang salah di klasifikasi.




**Gambar 2.5** *Multiclass Confusion Matrix*
(Sumber: Kruger, 2016)

dengan,

$F_{xy}$ : Jumlah prediksi untuk kelas aktual $x$ dan kelas prediksi $y$, $x = 1, 2, \ldots, L$, $y = 1, 2, \ldots, L$

$TP$ : $F_{jj}$

$TN$ : $\sum_{x=1}^{L\setminus\{j\}} \sum_{y=1}^{L\setminus\{j\}} F_{xy}$

$FP$ : $\sum_{x=1}^{L\setminus\{j\}} F_{xj}$

$FN$ : $\sum_{x=1}^{L\setminus\{j\}} F_{jx}$

Untuk permasalahan dalam klasifikasi yang melibatkan klasifikasi *multiclass*, ukuran performa yang biasa digunakan adalah akurasi, *precision*, *recall* dan *f1 score* (Flach, 2012).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.17}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.18}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.19}$$

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.20}$$

Selain itu, untuk mengetahui performa klasifikasi terhadap data *imbalace* juga digunakan nilai *Area Under Curve* (AUC). AUC merupakan metode untuk menghitung area dibawah kurva *Receiver Operating Characteristic* (ROC) (Fawcett, 2005).

$$\text{AUC} = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right) \tag{2.21}$$

Nilai AUC berada diantara 0.5 hingga 1 (Fawcett, 2005). Semakin mendekati nilai 1 maka AUC baik. Kriteria kebaikan AUC ditunjukkan oleh Tabel 2.1 (Bekkar dkk, 2013).

**Tabel 2.1** Kriteria Nilai AUC

| Nilai AUC | Keterangan |
|---|---|
| 0.5 – 0.6 | Kurang |
| 0.6 – 0.7 | Cukup |
| 0.7 – 0.8 | Baik |
| 0.8 – 0.9 | Sangat Baik |
| 0.9 – 1.0 | Sempurna |

Dalam klasifikasi *multiclass* akan terdapat performa klasifikasi untuk masing-masing kategori yang dimiliki sehingga untuk menghitung satu performa klasifikasi yang menggabungkan semua kategori dilakukan perhitungan kembali. Metode yang dapat digunakan untuk melakukan hal ini adalah *macro-averaging* dan *micro-averaging*. *Macro-averaging* memberikan bobot yang sama untuk setiap kategori dan *micro-averaging* memberikan bobot yang sama untuk masing-masing dokumen (Yang, 2000).

$$\text{Accuracy}_{macro} = \frac{1}{L}\sum_{j=1}^{L}\text{Accuracy}(TP_j, FP_j, TN_j, FN_j) \quad (2.22)$$

$$\text{Precision}_{macro} = \frac{1}{L}\sum_{j=1}^{L}\text{Precision}(TP_j, FP_j, TN_j, FN_j) \quad (2.23)$$

$$\text{Recall}_{macro} = \frac{1}{L}\sum_{j=1}^{L}\text{Recall}(TP_j, FP_j, TN_j, FN_j) \quad (2.24)$$

$$\text{F1 Score}_{macro} = \frac{1}{L}\sum_{j=1}^{L}\text{F1 Score}(TP_j, FP_j, TN_j, FN_j) \quad (2.25)$$

$$\text{AUC}_{macro} = \frac{1}{L}\sum_{j=1}^{L}\text{AUC}(TP_j, FP_j, TN_j, FN_j) \quad (2.26)$$

dengan,

$j$ : 1, 2, …, $L$

$L$ : jumlah kelas

*Micro-average* menghasilkan pengukuran yang efektif untuk kelas besar. Untuk mendapatkan efektifitas pengukuran pada kelas kecil digunakan hasil perhitungan *macro-average* (Manning dkk, 2007).

## 2.9 XL Axiata

Sejarah berdirinya perseroan ini dimulai dengan didirikannya PT Grahametropolitan Lestari yang bergerak di bidang perdagangan dan jasa umum pada tanggal 6 Oktober 1989, tahun 1995 mengubah nama menjadi PT Excelcomindo Pratama dengan kegiatan utama usahanya sebagai penyelenggara jasa telepon dasar. Pada saat awal operasinya (8 Oktober 1996), XL menyediakan jasa telepon dasar menggunakan teknologi GSM 900. Pada tahun 2006, XL memperoleh Izin Penyelenggaraan Seluler untuk teknologi 3G dan meluncurkannya secara komersial pada bulan September 2006. September 2005 merupakan suatu tonggak penting untuk Perseroan (XL), pegembangan seluruh aspek bisnisnya menjadikan XL sebagai suatu perusahaan publik dan tercatat di Bursa Efek Jakarta (sekarang Bursa Efek Indonesia). XL merupakan penyedia layanan telekomunikasi seluler dengan cakupan jaringan yang luas di seluruh wilayah Indonesia bagi pelanggan ritel dan menyediakan solusi bisnis bagi pelanggan korporat. Cakupan Layanan XL antara lain percakapan, data dan layanan nilai tambah lainnya (*value added services*). (Budiyanto, 2010).

**Gambar 2.6** XL Axiata
(Sumber: xl.co.id)

# BAB III
# METODOLOGI PENELITIAN

## 3.1 Sumber Data

Pada penelitian ini sumber data yang akan digunakan merupakan data sekunder yang diperoleh dari *Twitter API* dengan pencarian "@myXLCare" pada tanggal 8 Februari 2019 – 14 Maret 2019. Jumlah data yang digunakan adalah sebanyak 7631 *tweet*. Variabel yang digunakan adalah kata-kata yang diperoleh dari keseluruhan *tweet* tersebut yaitu frekuensi kata *j* yang muncul pada *tweet* ke *i* yang dilambangkan dengan $f_{ij}$ pada Tabel 3.1. Frekuensi kata berskala rasio.

## 3.2 Struktur Data

Struktur data yang digunakan dalam penelitian ini ditunjukkan oleh Tabel 3.1.

**Tabel 3.1** Struktur Data

| **Tweet ke** | $w_1$ | $w_2$ | … | $w_T$ |
|---|---|---|---|---|
| 1 | $tf_{1,1}$ | $tf_{1,2}$ | … | $tf_{1,T}$ |
| 2 | $tf_{2,1}$ | $tf_{2,2}$ | … | $tf_{2,T}$ |
| 3 | $tf_{3,1}$ | $tf_{3,2}$ | … | $tf_{3,T}$ |
| ⋮ | ⋮ | ⋮ | ⋱ | ⋮ |
| $N$ | $tf_{N,1}$ | $tf_{N,2}$ | .. | $tf_{N,T}$ |

keterangan :

$w_t$ : kata ke-$t$, $t = 1, 2, \ldots, T$

$tf_{d,t}$ : frekuensi kata ke-$t$ muncul pada *tweet* ke-$d$, $d = 1, 2, \ldots, N$

$T$ : jumlah kata

$N$ : jumlah *tweet*

## 3.3 Langkah Analisis

Langkah analisis yang akan dilakukan pada penelitian ini yaitu sebagai berikut.

1. Mengumpulkan data, data yang digunakan adalah data Twitter terhadap XL Axiata melalui pencarian

"@myXLCare". Laporan berbentuk data teks yang tidak terstuktur sehingga diperlukan pembersihan terlebih dahulu.

2. *Text preprocessing*, dilakukan untuk membersihkan data *tweet* yang sangat banyak mengandung informasi yang kurang signifikan dalam penelitian.. Adapun langkah-langkah yang dilakukan dalam *text prepocessing* secara umum adalah sebagai berikut.
   a. *Data cleaning*, pembersihan data yang dilakukan pada penelitian ini terdiri dari beberapa tahap diantaranya yaitu sebagai berikut.
      — *Removing symbol, number, ASCII strings, and punctuation,* merupakan proses penghapusan simbol, nomor, dan tanda baca lainnya dalam *tweet* seperti "# $ % & \ ' ( ) * + , - . / : ; < = > ? @ [ \ \ ] ^ _ ` { | } ~.
      — *Tokenization*, dilakukan untuk memecah kalimat *tweet* menjadi kata, frase, symbol, dan elemen lainnya yang memiliki arti. Contoh dari kalimat "gangguan internet sinyal jelek" dapat ditunjukkan dengan yaitu "gangguan", "internet", "sinyal", dan "jelek".
      — *Case folding,* dilakukan untuk mengubah format kata dalam *tweet* ke dalam format *lowercase*.
      — *Stemming*, dilakukan dengan mencari kata dasar dari setiap kata yang diperoleh pada *tweet*. *Stemming* dilakukan dengan bantuan *library* Sastrawi pada python
      — *Stopword removal*, menghapus kata-kata umum dalam *tweet* yang tidak memiliki arti yang penting seperti kata "dan", "atau", "yang", "itu", dan sebagainya.

Contoh data yang melalui proses *data cleaning* dapat dilihat pada Tabel 3.2

**Tabel 3.2** Tahapan *Data Cleaning*

| ***Text Preprocessing*** | **Struktur Kalimat** |
|---|---|
| Kalimat awal | Apakah xl sedang trouble? kenapa lemot kali yaa |
| *Removing symbol, puntuantion* | Apakah xl sedang trouble kenapa lemot kali yaa |
| *Tokenization* | “Apakah”, “xl”, “sedang”, “trouble”, “kenapa”, “lemot”, “kali”, “yaa” |
| *Case folding* | “apakah”, “xl”, “sedang”, “trouble”, “kenapa”, “lemot”, “kali”, “yaa” |
| *Stemming* | “apakah”, “xl”, “sedang”, “trouble”, “kenapa”, “lemot”, “kali”, “ya” |
| *Stopword removal* | “xl”, “trouble”, “lemot” |

b. Menentukan kategori masing-masing *tweet* berdasarkan website XL Axiata.
c. Melakukan pembobotan dengan metode TF-IDF, proses ini dilakukan untuk memperoleh bobot dari masing-masing kata dalam *tweet*. Langkah ini merupakan langkah untuk mengubah *teks* menjadi skala numerik untuk analisis *cluster* dan klasifikasi. Tabel 3.3 menunjukkan contoh *Document-Term Matrix* yang digunakan agar dapat diketahui frekuensi kata yang muncul pada setiap dokumen.

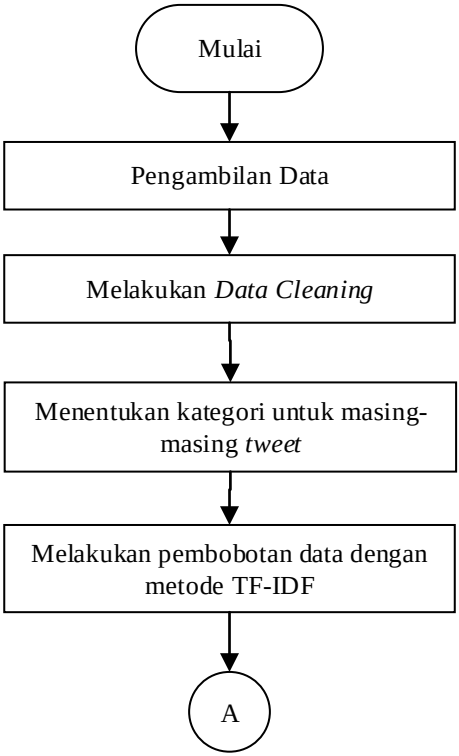
**Tabel 3.3** Contoh Document-Term Matrix

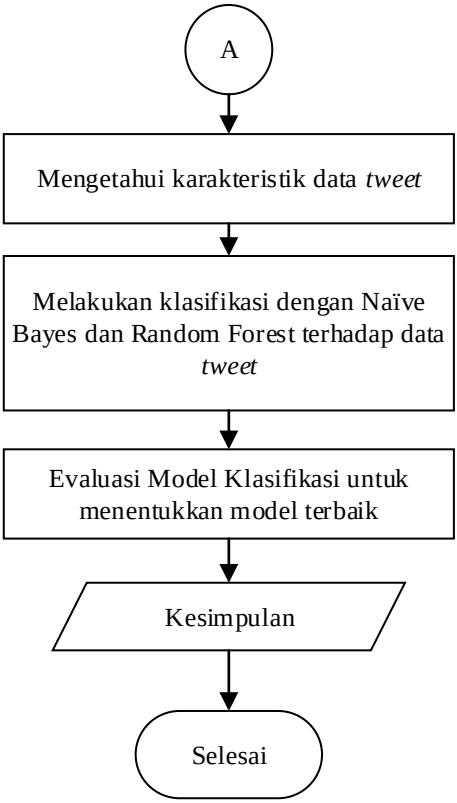
| ***Tweet*** | **internet** | **…** | **jaringan** | **…** | **parah** | **…** | **sinyal** |
|---|---|---|---|---|---|---|---|
| internet lemot | 1 | … | 0 | … | 0 | … | 0 |
| sinyal tidak stabil | 0 | … | 0 | … | 0 | … | 1 |
| sinyal parah | 0 | … | 0 | … | 1 | … | 1 |
| … | … | … | … | … | … | … | … |
| jaringan hilang | 0 | … | 1 | … | 0 | … | 0 |

3. Mengetahui karakteristik data *tweet* terhadap layanan *customer service* XL Axiata melalui akun @myXLCare.
4. Melakukan klasifikasi dengan *Naïve Bayes* dan *Random Forest*. Klasifikasi dilakukan terhadap kategori masing-masing *tweet*. Hasil analisis dibandingkan untuk menentukan model klasifikasi terbaik.
5. Interpretasi dan menarik kesimpulan.

## 3.4 Diagram Alir Penelitian

Berikut merupakan diagram alir yang dilakukan pada penelitian.






**Gambar 3.1** Diagram Alir Penelitian

*(Halaman ini sengaja dikosongkan)*

# BAB IV
# ANALISIS DAN PEMBAHASAN

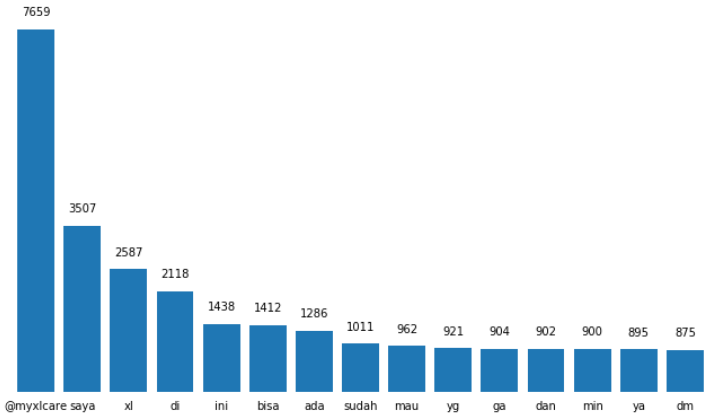
Pada penelitian ini dilakukan klasifikasi *tweet* yang ditujukan terhadap akun layanan *customer care* XL Axiata melalui akun *twitter* @myXLCare. Klasifikasi akan dilakukan dengan metode *Naïve Bayes* dan *Random Forest*. Kategori *tweet* terdiri dari 8 kategori dengan persebaran *tweet* tiap kategori yang tidak seimbang (*imbalance*). *Synthetic Minority Oversampling Technique* (SMOTE) dilakukan untuk mengatasi keadaan *imbalance*. Selanjutnya akan dilakukan perbandingan hasil klasifikasi dari setiap metode menggunakan nilai *precision*, *recall*, *f-1 score*, dan *Area Under Curve* (AUC).

## 4.1 *Preprocessing* dan Karakteristik Data

Hasil *crawling* data dengan Twitter API memperoleh 7631 *tweet* yang ditujukan terhadap akun layanan *customer care* XL Axiata melalui *twitter* @myXLCare. Dari data *tweet* tersebut akan dilihat karakteristik data dan dilakukan *preprocessing* sebelum dilakukan analisis klasifikasi. Tahapan *preprocessing* dilakukan untuk membersihkan data dari *noise* yang dapat menyebabkan bias pada klasifikasi. Karakteristik data dilihat sebelum dan sesudah *preprocessing* untuk mengetahui apa saja informasi yang sebenarnya terkandung dalam *tweet*.

### 4.1.1 Karakteristik Data sebelum *Preprocessing*

Karakteristik data perlu diketahui untuk menemukan informasi-informasi awal terkait data yang akan diolah. Hal ini penting dilakukan untuk mengetahui langkah-langkah apa saja yang akan dilakukan selanjutnya terkait dengan pemrosesan data. Karakteristik data dapat diketahui dengan melihat frekuensi kemunculan kata pada *tweet*. Frekuensi kata yang tinggi berarti kata tersebut sering ditulis oleh pengguna Twitter yang melakukan *tweet* terhadap @myXLCare. Frekuensi kemunculan kata pada *tweet* akan ditunjukkan dalam diagram batang pada Gambar 4.1.




**Gambar 4.1** 15 Kata dengan Frekuensi Kemunculan Tertinggi

Gambar 4.1 menunjukkan bahwa 15 kata dengan frekuensi tertinggi adalah "@myXLCare", "xl", "di" "ini", "bisa", "ada", "sudah", "mau", "yg", "ga", "dan", "min", "ya", dan "dm". Kata dengan frekuensi paling tinggi adalah "@myXLCare" yaitu sebanyak 7659 kata. Kata ini memiliki frekuensi kemunculan yang sangat tinggi dibandingkan kata-kata selanjutnya. Hal ini terjadi karena kata "@myXLCare" merupakan kata yang harus dituliskan untuk melakukan komunikasi dengan layanan *customer service* XL Axiata di Twitter. Selain itu, frekuensi kata "@myXLCare" lebih banyak dibandingkan jumlah *tweet* keseluruhan yang berarti satu pengguna kemungkinan menuliskan dua kata "@myXLCare" atau lebih pada *tweet* mereka.

### 4.1.2 Preprocessing Data *Tweet*

Karakterisik data pada Gambar 4.1 menunjukkan masih terdapat kata-kata yang tidak terlalu memiliki arti penting pada kasus klasifikasi ini, seperti kata "@myXLCare" yang wajib dituliskan, kata "di", "ini", "sudah", "mau", "yg", dan lain-lain. Kata-kata ini jika dibiarkan untuk proses selanjutnya tidak akan memberikan hasil klasifikasi yang maksimal sehingga perlu dilakukan *preprocessing* untuk membersihkan kumpulan kata-kata yang ada. Tahapan *preprocessing* yang dilakukan antara lain *data*

*cleaning* (menghapus link, menghapus tanda *retweet*, menghapus baris *enter*, menghapus tanda baca, menghapus nomor yang tidak berarti), mengubah kata menjadi *lowercase*, memperbaiki ejaan kata, mencari persamaan kata, menghilangkan *stopwords*, dan melakukan *stemming* (menemukan kata dasar tiap kata). Contoh tahapan *preprocessing* dapat dilihat pada Tabel 4.1.

**Tabel 4.1** Tahapan *Preprocessing* pada Kalimat *Tweet*

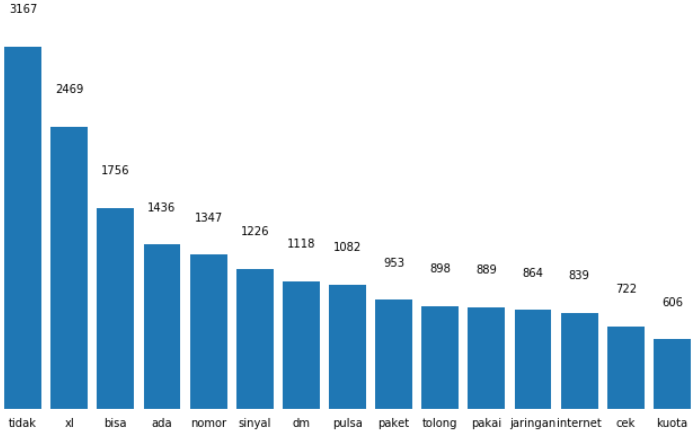
| ***Preprocessing*** | **Kalimat** | **Keterangan** |
|---|---|---|
| Kalimat awal | @mypmicky @myXLCare Di pulogadung juga sama ini jaringannya pdhl tulisannya LTE dan udah bikin laporan sebulan ga ada perubahan. Ku lelaaaahhh | - |
| Menghilangkan *hastag*, *mention*, *link*, tanda *retweet*, dan *emoticon* | Di pulogadung juga sama ini jaringannya pdhl tulisannya LTE dan udah bikin laporan sebulan ga ada perubahan. Ku lelaaaahhh | Menghilangkan @mypmicky dan @myXLCare |
| Menghilangkan *punctuation* | Di pulogadung juga sama ini jaringannya pdhl tulisannya LTE dan udah bikin laporan sebulan ga ada perubahan Ku lelaaaahhh | Menghilangkan tanda baca titik (.) |
| *Lowercase* | di pulogadung juga sama ini jaringannya pdhl tulisannya lte dan udah bikin laporan sebulan ga ada perubahan ku lelaaaahhh | Mengubah kalimat ke dalam *lowercase* |
| Persamaan kata | di pulogadung juga sama ini jaringannya pdhl tulisannya lte dan udah bikin laporan sebulan tidak ada perubahan ku lelah | Mengganti lelaaaahhh menjadi lelah |
| *Stemming* | di pulogadung juga sama ini jaringan pdhl tulis lte dan udah bikin lapor sebulan tidak ada ubah ku lelah | Mencari kata dasar untuk kata 'jaringannya', 'tulisannya', 'laporan', 'perubahan' |

**Tabel 4.1** Tahapan *Preprocessing* pada Kalimat *Tweet* (Lanjutan)

| | | |
|---|---|---|
| *Stopword* | pulogadung jaringan tulis lte lapor tidak ada ubah lelah | menghilangkan kata 'di', 'juga', 'sama', 'ini', 'pdhl', 'dan', 'udah', 'bikin', 'sebulan', 'ku' |

### 4.1.3 Karakteristik Data setelah *Preprocessing*

Tahapan *preprocessing* menghilangkan kata dan karakter yang tidak berarti. Jumlah *tweet* setelah dilakukan *preprocessing* menjadi 7569 *tweet*. Hal ini dapat terjadi karena terdapat *tweet* yang mengandung kata-kata tidak berarti. Setelah kata-kata tersebut hilang maka akan tersisa kata-kata yang untuk mengetahui karakteristik *tweet* yang ditujukan ke @myXLCare. Frekuensi kemunculan kata-kata tersebut dapat ditunjukkan dengan diagram batang pada Gambar 4.2.




**Gambar 4.2** 15 Kata dengan Frekuensi Kemunculan Tertinggi setelah *Preprocessing*

Gambar 4.2 menunjukkan bahwa kata dengan frekuensi tertinggi adalah “tidak”, dan selanjutnya diikuti oleh kata “xl”, “bisa”, “ada”. Kata “tidak” termasuk dalam kata yang memiliki makna dalam klasifikasi ini terutama jika kata “tidak” diikuti dengan kata “ada” ataupun “bisa”. Hal ini mengindikasikan tidak

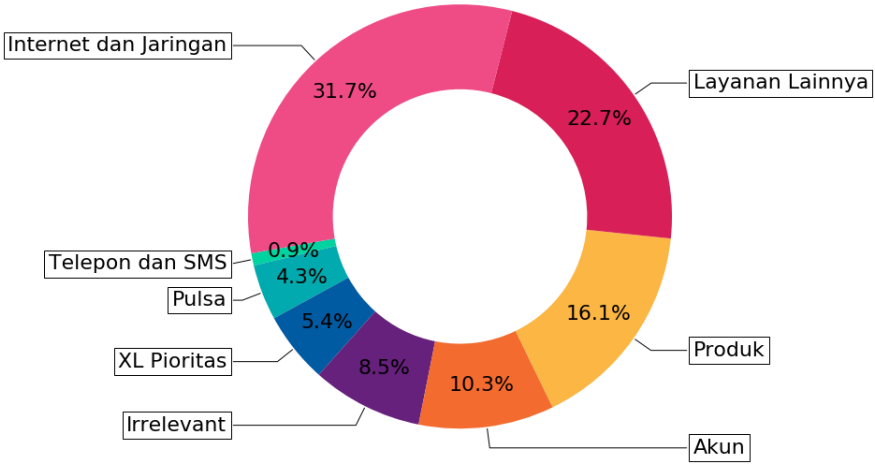
ada atau tidak bisanya suatu layanan dari XL Axiata. Kata "xl" masih termasuk ke dalam kata yang memiliki frekuensi tinggi karena sebagian besar *tweet* membahas informasi terkait XL Axiata. Setelah itu terdapat kata "nomor", "sinyal", "pulsa", "paket", "pakai", "jaringan", "internet", dan "kuota". Kata-kata ini menunjukkan bahwa *tweet* ini membahas jenis layanan dan produk yang dimiliki oleh XL Axiata. Kemudian terdapat kata "dm", "tolong", dan "cek". Kata ini dapat menunjukkan bahwa *tweet* mengandung pembahasan terkait layanan *customer service* XL Axiata seperti meminta tolong untuk menyelesaikan permasalah terkait layanyan XL, pengiriman *direct message* (dm), pengecekan atas permasalahan maupun layanan.

### 4.1.4 Kategori *Tweet*

Selanjutnya *tweet* akan dikategorikan ke dalam kategori yang telah ditetapkan oleh XL Axiata melalui *website* https://www.xl.co.id/id/bantuan. Terdapat delapan kategori diantaranya Pulsa, Akun, Internet dan Pengaturan, Produk, Prioritas, Telepon dan SMS, Prabayar, dan Lain-lain. Berdasarkan penjelasan kategori yang terdapat di halaman *website* tersebut kategori Produk dan Prabayar memiliki kesamaan topik, sehingga kategori Prabayar digabung ke dalam satu kategori yaitu kategori Produk. Dalam kategori Lain-lain membahas tentang layanan atau permasahalan lain terkait XL Axiata seperti *customer service*, penipuan, dan lain-lain. Kategori ini selanjutnya disebut kategori Layanan Lainnya. Pada *tweet* juga ditemukan *tweet* yang tidak termasuk ke dalam kategori manapun. *Tweet* ini berisikan topik yang kurang relevan terhadap layanan XL Axiata. Kategori Irrelevant akan dibuat untuk mengategorikan *tweet* seperti ini. Pada akhirnya, terdapat delapan kategori dalam *tweet* yang ditujukan kepada @myXLCare yaitu:

(1) Pulsa
(2) Akun
(3) Internet dan Jaringan
(4) Produk
(5) Prioritas
(6) Telepon dan SMS
(7) Layanan Lainnya
(8) Irrelevant

Persebaran *tweet* terhadap kategori dapat digambarkan dengan diagram lingkaran pada Gambar 4.3.




**Gambar 4.3** Persentase *Tweet* pada setiap Kategori

Gambar 4.3. menunjukkan bahwa kategori dengan jumlah *tweet* terbanyak adalah kategori Internet dan Jaringan yaitu sebesar 31.7%. Hal ini berarti *tweet* yang banyak disampaikan terhadap @myXLCare adalah tentang permasalahan internet dan jaringan. Selanjutnya kategori dengan jumlah *tweet* paling sedikit adalah kategori Telepon dan SMS yaitu sebesar 0.9%. Data menunjukkan bahwa pada era ini permasalahan internet dan jaringan jauh lebih tinggi dibandingkan permasalahan telepon dan sms. Hal ini sangat mungkin terjadi mengingat penggunaan internet sangat penting saat ini. Selain itu juga dapat diketahui bahwa penggunaan layanan telepon dan sms sudah relatif stabil atau tidak memiliki banyak masalah dibandingkan penggunaan internet yang masih mengalami banyak permasalahan seperti permasalahan kecepatan internet.

### 4.1.5 *Term Weighting*

Transfomasi kata dilakukan dengan mengubah data teks menjadi data numerik. Trasformasi kata dilakukan agar data *tweet* dapat dianalisis lebih lanjut pada analisis klasifikasi. Kumpulan

kata disajikan dalam bentuk frekuensi kemunculan masing-masing kata pada *tweet*. Setiap kata dilakukan pembobotan untuk membedakan nilai frekuensinya dengan kata yang lain.

Hal yang dilakukan pertama kali adalah perhitungan kemunculan kata pada masing-masing *tweet*. Kemunculan kata pada *tweet* ditunjukkan dengan *Document-Term Matrix*. *Document-Term Matrix* untuk data ini ditunjukkan oleh Tabel 4.2.

**Tabel 4.2** *Document-Term Ma*trix

| ***Tweet ke*** | **Kata** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **2g** | … | **lambat** | … | **xl** | … | **youtube** |
| 1 | 0 | … | 0 | … | 0 | … | 0 |
| 2 | 0 | … | 1 | … | 0 | … | 0 |
| ⋮ | ⋮ | ⋱ | ⋮ | ⋱ | ⋮ | ⋱ | ⋮ |
| 2592 | 0 | … | 2 | … | 2 | … | 0 |
| ⋮ | ⋮ | ⋱ | ⋮ | ⋱ | ⋮ | ⋱ | ⋮ |
| 7568 | 0 | … | 0 | … | 1 | … | 0 |
| 7569 | 0 | … | 0 | … | 1 | … | 0 |

Tabel 4.2 menunjukkan frekuensi kemunculan kata pada masing-masing *tweet*. Misalnya, pada *tweet* kedua kemunculan kata "lambat" sebanyak satu kali. Pada *tweet* ke-3166 kemunculan kata "lambat" dan kata "xl" sebanyak satu kali dan seterusnya. Kata-kata ini menjadi variabel penelitian dan frekuensi kemunculan kata menjadi nilai masing-masing variabel. Jumlah kata yang digunakan sebagai variabel adalah sebanyak 507 kata.

Setelah mendapatkan *Document–Term Matrix*, akan dilakukan pembobotan dengan metode *Term Frequency-Inverse Document Frequency* (TF-IDF). Pembobotan dilakukan untuk mendapatkan tingkat kepentingan suatu kata. TF-IDF dilakukangan dengan menghitung nilai *Term Frequency* (TF), yaitu frekuensi kemunculan kata pada *tweet*. Selanjutnya menghitung nilai *Document Frequency* (DF) dan *Inverse Document Frequency* (IDF). *Document Frequency* yaitu jumlah *tweet* yang mengandung kata ke-*i*. *Inverse Document Matrix* (IDF)

dapat dihitung dengan persamaan (2.1). Tabel 4.3 menampilkan contoh perhitungan *Document Frequency* dan *Inverse Document Frequency*.

**Tabel 4.3** Contoh Perhitungan DF dan IDF

| **Kata** | **Tweet ke** | | | | | | **DF** | **IDF** |
|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | ⋯ | **3166** | ⋯ | **7568** | | |
| 2g | 0 | 0 | ⋯ | 0 | ⋯ | 0 | 28 | $\log\left(\frac{7569}{28}\right) = 2,431$ |
| ⋮ | ⋮ | ⋮ | ⋱ | ⋮ | ⋱ | ⋮ | ⋮ | ⋮ |
| lambat | 0 | 1 | ⋯ | 1 | ⋯ | 0 | 520 | $\log\left(\frac{7569}{520}\right) = 1,163$ |
| ⋮ | ⋮ | ⋮ | ⋱ | ⋮ | ⋱ | ⋮ | ⋮ | ⋮ |
| xl | 0 | 0 | ⋯ | 1 | ⋯ | 1 | 2204 | $\log\left(\frac{7569}{2204}\right) = 0,535$ |
| ⋮ | ⋮ | ⋮ | ⋱ | ⋮ | ⋱ | ⋮ | ⋮ | ⋮ |
| youtube | 0 | 0 | ⋯ | 0 | ⋯ | 0 | 106 | $\log\left(\frac{7569}{106}\right) = 1,853$ |

Tabel 4.3 menunjukkan contoh nilai DF dan IDF dari masing-masing kata. Kata “2g” memiliki nilai DF sebesar 28 yang berarti kata “2g” muncul dalam 28 *tweet*. Semakin tinggi nilai DF makan nilai IDF akan semakin kecil. Kata “xl” muncul dalam 2204 *tweet* atau memiliki nilai DF sebesar 2204 dan memiliki nilai IDF sebesar 0,535. Nilai ini lebih kecil dibandingkan nilai IDF “2g”. Bobot kata “xl” akan lebih kecil karena terdapat pada lebih banyak *tweet* sedangkan bobot kata “2g” memiliki bobot yang lebih besar karena terdapat pada lebih sedikit dokumen.

Selanjutnya dilakukan perhitungan TF-IDF dengan menggunakan persamaan (2.2), yaitu dengan mengalikan nilai TF setiap kata-*i* pada masing-masing *tweet* dengan nilai IDF dari kata ke-*i*. Tabel 4.4 menunjukkan hasil perhitungan TF-IDF.

**Tabel 4.4** Contoh Perhitungan TF- IDF

| *Tweet ke* | Kata | | | | | | |
|---|---|---|---|---|---|---|---|
| | **2g** | … | **lambat** | … | **xl** | … | **youtube** |
| 1 | 0 | … | 0 | … | 0 | … | 0 |
| 2 | 0 | … | 1,163 | … | 0 | … | 0 |
| ⋮ | ⋮ | ⋱ | ⋮ | ⋱ | ⋮ | ⋱ | ⋮ |
| 2592 | 0 | … | 2,326 | … | 1,070 | … | 0 |
| ⋮ | ⋮ | ⋱ | ⋮ | ⋱ | ⋮ | ⋱ | ⋮ |
| 7568 | 0 | … | 0 | … | 0,535 | … | 0 |
| 7569 | 0 | … | 0 | … | 0,535 | … | 0 |

Tabel 4.4 menunjukkan hasil perhitungan TF-IDF pada setiap kata untuk setiap *tweet*. Nilai TF-IDF kata "lambat" pada *tweet* 2 sebesar 1,163 diperoleh dari nilai TF sebesar 1 dan nilai IDF sebesar 1,163, sehingga TF-IDF diperoleh sebesar 1 × 1,163 = 1,163 sedangkan untuk *tweet* 2592 diperoleh nilai TF sebesar 2 dan nilai IDF kata "lambat"yaitu 1,163 sehingga 2 × 1,163 = 2,326. Perhitungan nilai TF-IDF untuk kata lainnya juga dilakukan dengan cara tersebut. Pada analisis klasifikasi nilai TF-IDF akan digunakan sebagai variabel prediktor dengan variabel respon yaitu kategori *tweet*. Struktur data setelah dilakukan *preprocessing* dan siap dilakukan analisis klasifikasi ditunjukkan pada Tabel 4.5.

**Tabel 4.5** Struktur Data setelah *Preprocessing*

| **Tweet ke** | $w_1$ | $w_2$ | …. | $w_T$ | $c$ |
|---|---|---|---|---|---|
| 1 | $tf_{1,1}$ | $tf_{1,2}$ | … | $tf_{1,T}$ | $c_1$ |
| 2 | $tf_{2,1}$ | $tf_{2,2}$ | … | $tf_{2,T}$ | $c_2$ |
| 3 | $tf_{3,1}$ | $tf_{3,2}$ | … | $tf_{3,T}$ | $c_3$ |
| ⋮ | ⋮ | ⋮ | ⋱ | ⋮ | ⋮ |
| $N$ | $tf_{N,1}$ | $tf_{N,2}$ | .. | $tf_{N,T}$ | $c_N$ |

keterangan :

$w_t$ : kata ke-$t$, $t = 1, 2, \ldots, T$

$tf_{d,t}$ : frekuensi kata ke-$t$ muncul pada *tweet* ke-$d$, $d = 1, 2, \ldots, N$

$T$ : jumlah kata

*N* : jumlah *tweet*
*c* : kategori *tweet*

## 4.2 Klasifikasi Menggunakan Naïve Bayes

Metode klasifikasi yang pertama digunakan adalah Naïve Bayes. *Naïve Bayes* merupakan metode klasifikasi dengan memprediksi probabilitas keanggotaan terhadap suatu kelas. Dalam hal ini memprediksi probabilitas keanggotaan *tweet* terhadap kategori. Sebelum dilakukan klasifikasi data akan dibagi ke dalam data *training* dan data *testing*. Pembagian data akan dilakukan dengan metode *K-Fold* dengan 10-*Fold*. *K-Fold* dilakukan untuk mengetahui kestabilan model klasifikasi dalam melakukan klasifikasi terhadap jenis data yang beragam. Data *traning* digunakan untuk membentuk model dan data *testing* digunakan untuk melakukan prediksi probabilitas keanggotaan terhadap masing-masing kategori. Untuk mengetahui kebaikan model prediksi akan dilakukan perhitungan nilai *Accuracy*, *Presicion*, *Recall*, *F1 Score*, dan AUC.

### 4.2.1 Klasifikasi *Naïve Bayes* dengan Data Awal (Tanpa SMOTE)

Klasifikasi *Naïve Bayes* dengan menggunakan data awal adalah klasifikasi menggunakan data yang diperoleh setelah hasil *preprocessing* dan pembobotan TF-IDF tanpa dilakukan SMOTE. Data dibagi ke dalam 10-*fold cross validation* untuk mengetahui kestabilan model dalam melakukan klasifikasi. Dari 10-*fold* tersebut selanjutnya dilakukan perhitungan rata-rata untuk mengetahui kebaikan model secara umum dalam melakukan klasifikasi. Dalam menunjukkan proses terbentuknya model klasifikasi *naïve bayes*, digunakan *Fold* terbaik dari 10-*fold* yang ada. Perhitungan model klasifikasi *Naïve Bayes* dilakukan berdasarkan persamaan (2.4). Berdasarkan persamaan tersebut diperoleh model klasifikasi untuk masing-masing kategori. Model klasifikasi *Naïve Bayes* dengan data awal ditunjukkan pada Tabel 4.5.

**Tabel 4.6** Model *Naïve Bayes* terhadap Kategori *Tweet*

| Kategori | Model |
|---|---|
| 1 | $0,0426 \times 0,0008^{f(1)} \times 0,0012^{f(2)} \times \ldots \times 0,0015^{f(507)}$ |
| 2 | $0,1033 \times 0,0008^{f(1)} \times 0,0010^{f(2)} \times \ldots \times 0,0013^{f(507)}$ |
| 3 | $0,3173 \times 0,0020^{f(1)} \times 0,0076^{f(2)} \times \ldots \times 0,0019^{f(507)}$ |
| 4 | $0,1614 \times 0,0006^{f(1)} \times 0,0010^{f(2)} \times \ldots \times 0,0077^{f(507)}$ |
| 5 | $0,0543 \times 0,0010^{f(1)} \times 0,0026^{f(2)} \times \ldots \times 0,0015^{f(507)}$ |
| 6 | $0,0093 \times 0,0015^{f(1)} \times 0,0015^{f(2)} \times \ldots \times 0,0018^{f(507)}$ |
| 7 | $0,2269 \times 0,0007^{f(1)} \times 0,0007^{f(2)} \times \ldots \times 0,0003^{f(507)}$ |
| 8 | $0,0846 \times 0,0007^{f(1)} \times 0,0011^{f(2)} \times \ldots \times 0,0007^{f(507)}$ |

Tabel 4.6 menunjukkan 8 model yang terbentuk dengan klasifikasi Naïve Bayes. Klasifikasi *tweet* dilakukan terhadap data *testing* dengan menghitung probabilitas *tweet* pada masing-masing kategori dengan mengganti nilai $f(1)$ hingga $f(507)$, $f(i)$ adalah frekuensi kata ke-$i$ pada sebuah *tweet*. Model yang terbentuk pada Tabel 4.6 dapat digunakan untuk menghitung nilai probabilitas *tweet* ke dalam 8 kategori. Probabilitas terhadap kategori menunjukkan kecenderungan *tweet* diprediksi ke dalam kategori tersebut. Probabilitas keanggotaan *tweet* terhadap kategori dijelaskan pada tabel 4.7.

**Tabel 4.7** Probabilitas Keanggotaan *Tweet* pada Kategori

| ***Tweet ke*** | **Kategori** | | | | | | **Hasil** |
|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | | **8** | |
| 1 | 0,0043 | 0,0107 | 0,9064 | 0,0172 | … | 0,0176 | 3 |
| 2 | 0,0217 | 0,4733 | 0,2571 | 0,0657 | … | 0,0396 | 2 |
| 3 | 0,0057 | 0,0118 | 0,8824 | 0,013 | … | 0,0233 | 3 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋱ | ⋮ | ⋮ |
| 756 | 0,0385 | 0,2878 | 0,0991 | 0,3219 | … | 0,0663 | 4 |

Tabel 4.7 menunjukkan probabilitas klasifikasi *tweet* terhadap kategori *tweet*. Semakin tinggi probabilitas maka semakin tinggi kecenderungan *tweet* masuk ke dalam kategori tersebut. Misalkan pada *tweet* pertama, probabilitas pada kategori 1 adalah sebesar 0,0043, probabilitas pada kategori 2 adalah 0,0107, probabilitas pada kategori 3 adalah 0,9064, probabilitas pada kategori 4 adalah 0,0172, dan seterusnya. Probabilitas paling tinggi adalah pada kategori 3, sehingga *tweet* pertama diklasifikasikan pada kategori 3. Cara yang sama dilakukan untuk *tweet* kedua hingga *tweet* ke-756 sehingga diperoleh prediksi kategori untuk *tweet* kedua adalah kategori 2, *tweet* ketiga diprediksi termasuk dalam kategori 2, dan seterusnya.

Setelah memperoleh hasil prediksi untuk *tweet* pertama hingga *tweet* ke-756 akan dilakukan validasi kategori dengan membandingkan kategori aktual dengan kategori prediksi. Validasi dilakukan untuk mengetahui berapa jumlah data *tweet* dengan kategori yang terprediksi benar dan berapa jumlah data *tweet* dengan kategori yang terprediksi salah. Nantinya hal ini berguna untuk mengetahui performa dari model klasifikasi. Hasil validasi kategori dapat dijelaskan dengan *confusion matrix*. *Confusion matrix* terdiri dari kategori aktual dan kategori prediksi. *Confusion matrix* untuk model *Naïve Bayes* ditunjukkan pada Tabel 4.8.

**Tabel 4.8** *Confusion Matrix* Klasifikasi Naïve Bayes

| **Kategori Aktual** | **Kategori Prediksi** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 15 | 1 | 0 | 15 | 0 | 0 | 1 | 0 |
| 2 | 0 | 56 | 5 | 13 | 0 | 0 | 4 | 0 |
| 3 | 0 | 4 | 219 | 6 | 1 | 0 | 8 | 2 |
| 4 | 0 | 1 | 10 | 103 | 1 | 0 | 5 | 2 |
| 5 | 0 | 1 | 11 | 3 | 25 | 0 | 1 | 0 |
| 6 | 0 | 4 | 3 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 15 | 3 | 0 | 0 | 152 | 2 |
| 8 | 1 | 1 | 28 | 5 | 0 | 0 | 19 | 10 |

Berdasarkan *confusion matrix* pada Tabel 4.8, pada kategori 1 dengan jumlah *tweet* 32 terdapat 15 *tweet* yang diklasifikasikan dengan benar. Pada kategori 2 dengan jumlah *tweet* 78 terdapat 56 *tweet* yang diklasifikasikan dengan benar. Pada kategori 3 dengan jumlah *tweet* 240 terdapat 219 *tweet* yang diklasifikasikan dengan benar. Pada kategori 4 dengan jumlah *tweet* 122 terdapat 103 *tweet* yang diklasifikasikan dengan benar. Pada kategori 5 dengan jumlah *tweet* 41 terdapat 25 *tweet* yang diklasifikasikan dengan benar. Pada kategori 6 dengan jumlah *tweet* 7 tidak terdapat *tweet* yang diklasifikasikan dengan benar. Pada kategori 7 dengan jumlah *tweet* 172 terdapat 152 *tweet* yang diklasifikasikan dengan benar. Pada kategori 8 dengan jumlah *tweet* 64 terdapat 10 *tweet* yang diklasifikasikan dengan benar.

Jumlah prediksi benar dan prediksi salah pada *confusion matrix* selanjutnya digunakan untuk menghitung performa model klasifikasi. Performa model klasifikasi diketahui dengan evaluasi model. Dalam evaluasi model terdapat beberapa nilai yang sering digunakan diantaranya adalah nilai *precision*, *recall*, *f1 score*, dan nilai AUC. Khusus untuk *f1 score* dan nilai AUC digunakan untuk mengetahui performa model klasifikasi dalam kasus data *imbalance*. Evaluasi model ditunjukkan pada Tabel 4.9.

**Tabel 4.9** Evaluasi Model Klasifikasi *Naïve Bayes*

| **Kategori** | ***Accuracy*** | ***Precision*** | ***Recall*** | ***F1 Score*** | **AUC** |
|---|---|---|---|---|---|
| 1 | 0,77 | 0,94 | 0,47 | 0,63 | 0,73 |
| 2 | 0,77 | 0,82 | 0,72 | 0,77 | 0,85 |
| 3 | 0,77 | 0,75 | 0,91 | 0,82 | 0,89 |
| 4 | 0,77 | 0,70 | 0,84 | 0,76 | 0,89 |
| 5 | 0,77 | 0,93 | 0,61 | 0,74 | 0,80 |
| 6 | 0,77 | 0,00 | 0,00 | 0,00 | 0,50 |
| 7 | 0,77 | 0,80 | 0,88 | 0,84 | 0,91 |
| 8 | 0,77 | 0,63 | 0,16 | 0,25 | 0,57 |
| Rata-rata | 0,77 | 0,70 | 0,57 | 0,60 | 0,76 |

Tabel 4.9 menunjukkan nilai rata-rata *accuracy* adalah 0,77, rata-rata *precision* adalah 0,70, rata-rata *recall* adalah 0,57, rata-rata *f1 score* adalah 0,60, dan rata-rata AUC adalah 0,76. Nilai rata-rata dihitung dengan *macro-average* sesuai dengan persamaan (2.22) hingga (2.26). Nilai *precision* menjelaskan persentase kategori prediksi yang sesuai dengan kategori aktual dari total prediksi terhadap suatu kategori. *Precision* sebesar 0,70 berarti dari total prediksi terhadap suatu kategori, sebanyak 70% sesuai dengan kategori aktual. Nilai *precision* terbesar adalah pada kategori 1 yaitu sebesar 0,94 . Nilai terendah adalah pada kategori 6 yaitu nol yang berarti bahwa dari hasil prediksi kategori tidak ada yang sesuai dengan kategori aktual. Nilai *recall* menjelaskan persentase kategori aktual yang terprediksi dengan benar. *Recall* sebesar 0,57 berarti bahwa dari total kategori aktual sebanyak 57% dapat diprediksi dengan benar. Nilai *recall* terbesar adalah pada kategori 3 yaitu sebesar 0,91. Nilai terendah adalah pada kategori 6 yaitu nol yang berarti bahwa dari total kategori aktual tidak ada yang terprediksi dengan benar. *F1 score* menjelaskan keseimbangan nilai *precision* dan *recall*. *F1 score* sebesar 0,60 menunjukkan kemungkinan nilai *precision* dan *recall* yang rendah atau terjadi ketidakseimbangan nilai *precision* dan *recall*. Pada kasus ini terjadi ketidakseimbangan antara nilai *precision* dan *recall* dengan nilai *recall* yang lebih kecil dari nilai *precision*. Ketidakseimbangan terbesar terdapat pada kategori 8 yaitu nilai *precision* sebesar 0,63 dan nilai *recall* hanya 0,16. Nilai AUC menunjukkan kemampuan model dalam mengklasifikasikan kasus data *imbalance* yaitu ketepatan klasifikasi pada kategori positif dan kategori negatif. AUC rata-rata sebesar 0,76 menunjukkan model baik dalam mengklasifikasikan data *imbalance*.

Validasi terhadap model dilakukan untuk mengetahui performa model klasifikasi Naïve Bayes. Validasi dilakukan dengan 10-*fold cross validation*. Data dibagi ke dalam 10-*fold* data *training testing*. Performa model dievaluasi dengan nilai *Accuracy*, *Presicion*, *Recall*, *F1 Score*, dan AUC. Hasil evaluasi model klasifikasi *Naïve Bayes* ditunjukkan pada Tabel 4.10.

**Tabel 4.10** Evaluasi Klasifikasi *Naïve Bayes* dengan 10-*Fold*

| *Fold* | *Accuracy* | *Precision* | *Recall* | *F1 Score* | AUC |
|---|---|---|---|---|---|
| 1 | 0,76 | 0,70 | 0,55 | 0,58 | 0,75 |
| 2 | 0,73 | 0,63 | 0,52 | 0,55 | 0,73 |
| 3 | 0,74 | 0,66 | 0,54 | 0,57 | 0,74 |
| 4 | 0,72 | 0,65 | 0,51 | 0,54 | 0,73 |
| 5 | 0,74 | 0,66 | 0,54 | 0,57 | 0,74 |
| 6 | 0,74 | 0,65 | 0,53 | 0,56 | 0,74 |
| 7 | 0,77 | 0,70 | 0,57 | 0,60 | 0,76 |
| 8 | 0,74 | 0,66 | 0,55 | 0,58 | 0,75 |
| 9 | 0,75 | 0,68 | 0,53 | 0,56 | 0,74 |
| 10 | 0,71 | 0,64 | 0,52 | 0,54 | 0,73 |
| Rata-rata | 0,74 | 0,66 | 0,54 | 0,56 | 0,74 |

Tabel 4.10 menunjukkan nilai evaluasi model pada masing-masing *fold*. Persebaran kategori *tweet* tidak seimbang atau *imbalance*, sehingga untuk melakukan evaluasi model digunakan nilai *f1 Score* dan AUC. Rata-rata evaluasi model menunjukkan *f1 Score* sebesar 0,56 dan nilai AUC sebesar 0,74. *F1 Score* tersebut berarti bahwa nilai *precision* dan *recall* cenderung kecil atau tidak terdapat keseimbangan nilai antara *precision* dan *recall* dimana *precision* lebih besar daripada *recall*. Sedangkan nilai AUC menunjukkan model baik dalam mengklasifikasikan *tweet* meskipun dalam kasus *imbalance*. Berdasarkan *f1 score* dan nilai AUC, *fold* ke-7 merupakan model dengan performa klasifikasi terbaik.

### 4.2.1 Klasifikasi *Naïve Bayes* dengan SMOTE

Klasifikasi *Naïve Bayes* dilakukan dengan menggunakan SMOTE untuk menangani kasus data *imbalance*. Data yang mengalami *imbalance* akan menyebabkan *f1 score* dan nilai AUC kurang maksimal yang terjadi karena kesalahan klasifikasi pada kelas minoritas. SMOTE dilakukan dengan membuat variabel sintetis pada kategori minoritas sehingga jumlah data pada kategori minoritas menjadi sama dengan kategori mayoritas yang

menyebabkan kemungkinan kesalahan klasifikasi untuk masing-masing kelas menjadi sama besar. Tabel 4.11 menunjukkan perbandingan jumlah data sebelum dan sesudah dilakukan SMOTE pada data awal.

**Tabel 4.11** Perbandingan Jumlah Data Awal dan Data SMOTE untuk Model *Naïve Bayes*

| Jenis Data | Kategori | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| Data Awal | 290 | 704 | 2162 | 1100 | 370 | 64 | 1546 | 577 |
| Data SMOTE | 2162 | 2162 | 2162 | 2162 | 2162 | 2162 | 2162 | 2162 |

Setelah jumlah data pada masing-masing kategori sama, data dibagi ke dalam 10-*fold*. Kemudian data *training* digunakan untuk membuat model klasifikasi *Naïve Bayes* berdasarkan persamaan (2.4). *Fold* terbaik dipilih untuk menunjukkan proses perhitungan model *Naïve Bayes*. Model klasifikasi *Naïve Bayes* dengan data SMOTE ditunjukkan pada Tabel 4.12.

**Tabel 4.12** Model *Naïve Bayes* SMOTE terhadap Kategori *Tweet*

| Kategori | Model |
|---|---|
| 1 | $0{,}1250 \times 0{,}0002^{f(1)} \times 0{,}0008^{f(2)} \times \ldots \times 0{,}0010^{f(507)}$ |
| 2 | $0{,}1250 \times 0{,}0006^{f(1)} \times 0{,}0008^{f(2)} \times \ldots \times 0{,}0014^{f(507)}$ |
| 3 | $0{,}1250 \times 0{,}0020^{f(1)} \times 0{,}0076^{f(2)} \times \ldots \times 0{,}0019^{f(507)}$ |
| 4 | $0{,}1250 \times 0{,}0003^{f(1)} \times 0{,}0007^{f(2)} \times \ldots \times 0{,}0086^{f(507)}$ |
| 5 | $0{,}1250 \times 0{,}0004^{f(1)} \times 0{,}0022^{f(2)} \times \ldots \times 0.0011^{f(507)}$ |
| 6 | $0{,}1250 \times 0{,}0002^{f(1)} \times 0{,}0002^{f(2)} \times \ldots \times 0{,}0011^{f(507)}$ |
| 7 | $0{,}1250 \times 0{,}0007^{f(1)} \times 0{,}0007^{f(2)} \times \ldots \times 0{,}0002^{f(507)}$ |
| 8 | $0{,}1250 \times 0{,}0003^{f(1)} \times 0{,}0007^{f(2)} \times \ldots \times 0{,}0003^{f(507)}$ |

Tabel 4.12 menunjukkan 8 model yang terbentuk dengan klasifikasi *Naïve Bayes* pada data SMOTE. Selanjutnya klasifikasi dilakukan terhadap data *testing*. Klasifikasi *tweet* dilakukan dengan menghitung probabilitas *tweet* pada masing-masing

kategori dengan mengganti nilai *f*(1) hingga *f*(507), *f*(*i*) adalah frekuensi kata ke-*i* pada sebuah *tweet*. Model yang terbentuk pada Tabel 4.12 dapat digunakan untuk menghitung nilai probabilitas *tweet* ke dalam 8 kategori. Probabilitas keanggotaan *tweet* pada data *testing* terhadap kategori dijelaskan pada tabel 4.13.

**Tabel 4.13** Probabilitas Keanggotaan *Tweet* pada Kategori (SMOTE)

| ***Tweet ke*** | **Kategori** | | | | | | **Hasil** |
|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | | **8** | |
| 1 | 0,0020 | 0,0138 | 0,8854 | 0,0160 | … | 0,0159 | 3 |
| 2 | 0,0155 | 0,6050 | 0,1012 | 0,0326 | … | 0,0233 | 2 |
| 3 | 0,0058 | 0,0111 | 0,6958 | 0,0101 | … | 0,0312 | 3 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋱ | ⋮ | ⋮ |
| 756 | 0,0307 | 0,2333 | 0,0313 | 0,2125 | … | 0,0369 | 6 |

Tabel 4.13 menunjukkan probabilitas klasifikasi *tweet* pada data *testing* terhadap kategori *tweet* pada data SMOTE. Pada *tweet* pertama, probabilitas pada kategori 1 adalah sebesar 0,0020, probabilitas pada kategori 2 adalah 0,0138, probabilitas pada kategori 3 adalah 0,8854, probabilitas pada kategori 4 adalah 0,0160, dan seterusnya. Probabilitas paling tinggi adalah pada kategori 3, sehingga *tweet* pertama diklasifikasikan pada Kategori 3. Cara yang sama dilakukan untuk *tweet* kedua hingga *tweet* ke-756. Akan terdapat beberapa perbedaan klasifikasi kategori pada data *testing* dengan menggunakan model *Naïve Bayes* Smote dan *Naïve Bayes* pada data awal, salah satunya adalah pada *tweet* ke-756. Dengan menggunakan model *Naïve Bayes* pada data awal, klasifikasi untuk *tweet* ke-756 adalah kategori Produk (4) sedangkan pada model *Naïve Bayes* SMOTE, klasifikasi untuk *tweet* ke-756 adalah kategori Akun (2).

Setelah memperoleh hasil prediksi untuk *tweet* pertama hingga *tweet* ke-756 akan dilakukan validasi kategori dengan membandingkan kategori aktual dengan kategori prediksi. Hasil validasi kategori dapat dijelaskan dengan *confusion matrix*. *Confusion matrix* ditunjukkan pada Tabel 4.14.

**Tabel 4.14** *Confusion Matrix* Klasifikasi *Naïve Bayes* SMOTE

| Kategori Aktual | Kategori Prediksi | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 27 | 2 | 0 | 2 | 0 | 0 | 0 | 1 |
| 2 | 5 | 57 | 1 | 7 | 0 | 5 | 3 | 0 |
| 3 | 2 | 4 | 200 | 9 | 6 | 4 | 6 | 9 |
| 4 | 10 | 6 | 4 | 92 | 2 | 1 | 1 | 6 |
| 5 | 1 | 1 | 4 | 1 | 33 | 0 | 1 | 0 |
| 6 | 0 | 0 | 0 | 0 | 1 | 6 | 0 | 0 |
| 7 | 3 | 0 | 7 | 1 | 5 | 7 | 134 | 15 |
| 8 | 3 | 2 | 12 | 3 | 1 | 3 | 10 | 30 |

Berdasarkan *confusion matrix* pada Tabel 4.14, pada kategori 1 dengan jumlah *tweet* 32 terdapat 27 *tweet* yang diklasifikasikan dengan benar. Pada kategori 2 dengan jumlah *tweet* 78 terdapat 57 *tweet* yang diklasifikasikan dengan benar. Pada kategori 3 dengan jumlah *tweet* 240 terdapat 200 *tweet* yang diklasifikasikan dengan benar. Pada kategori 4 dengan jumlah *tweet* 122 terdapat 92 *tweet* yang diklasifikasikan dengan benar. Pada kategori 5 dengan jumlah *tweet* 41 terdapat 33 *tweet* yang diklasifikasikan dengan benar. Pada kategori 6 dengan jumlah *tweet* 7, terdapat 6 *tweet* yang diklasifikasikan dengan benar. Pada kategori 7 dengan jumlah *tweet* 172 terdapat 134 *tweet* yang diklasifikasikan dengan benar. Pada kategori 8 dengan jumlah *tweet* 64 terdapat 30 *tweet* yang diklasifikasikan dengan benar. Model klasifikasi *naïve bayes* menghasilkan peningkatan jumlah klasifikasi benar untuk Pulsa, Akun, Prioritas, Telepon dan SMS, dan kategori Irrelevant. Sedangkan kategori lainnya mengalami penurusan klasifikasi benar.

Untuk mengetahui performa model klasifikasi dilakukan evaluasi model. Evaluasi model dilakukan dengan menghitung nilai *precision*, *recall*, *f1 score*, dan nilai AUC. Selanjutnya untuk memperoleh rata-rata dari evaluasi model digunakan *macro-*

*average* dengan persamaan (2.22) hingga (2.26). Evaluasi model untuk data *testing* ditunjukkan pada Tabel 4.15.

**Tabel 4.15** Evaluasi Model Klasifikasi *Naïve Bayes* SMOTE

| Kategori | *Accuracy* | *Precision* | *Recall* | *F1 Score* | AUC |
|---|---|---|---|---|---|
| 1 | 0,77 | 0,53 | 0,84 | 0,65 | 0,90 |
| 2 | 0,77 | 0,79 | 0,73 | 0,76 | 0,85 |
| 3 | 0,77 | 0,88 | 0,83 | 0,85 | 0,89 |
| 4 | 0,77 | 0,80 | 0,75 | 0,78 | 0,86 |
| 5 | 0,77 | 0,69 | 0,80 | 0,74 | 0,89 |
| 6 | 0,77 | 0,23 | 0,86 | 0,36 | 0,92 |
| 7 | 0,77 | 0,86 | 0,78 | 0,82 | 0,87 |
| 8 | 0,77 | 0,49 | 0,47 | 0,48 | 0,71 |
| Rata-rata | 0,77 | 0,66 | 0,76 | 0,68 | 0,86 |

Tabel 4.15 menunjukkan nilai rata-rata *accuracy* sebesar 0,77, rata-rata *precision* adalah 0.66, rata-rata *recall* adalah 0,76, rata-rata *f1 score* adalah 0,68, dan rata-rata AUC adalah 0,86. *Precision* sebesar 0,66 berarti dari total prediksi terhadap suatu kategori, sebanyak 66% sesuai dengan kategori aktual. Nilai *precision* terbesar adalah pada kategori 3 yaitu sebesar 0,88 . Nilai terendah adalah pada kategori 6 yaitu 0,23. *Recall* sebesar 0,76 berarti bahwa dari total kategori aktual sebanyak 76% dapat diprediksi dengan benar. Nilai *recall* terbesar adalah pada kategori 6 yaitu sebesar 0,86. Nilai terendah adalah pada kategori 8 yaitu 0,47. *F1 score* sebesar 0,68 menunjukkan kemungkinan nilai *precision* dan *recall* yang rendah atau terjadi ketidakseimbangan nilai *precision* dan *recall*. Pada kasus ini terjadi ketidakseimbangan antara nilai *precision* dan *recall* dengan nilai *precision* yang lebih kecil dari nilai *recall*. Ketidakseimbangan terbesar terdapat pada kategori 6 yaitu nilai *precision* sebesar 23% dan nilai *recall* 86%. Nilai AUC rata-rata sebesar 86% menunjukkan model sangat baik dalam mengklasifikasikan data *imbalance*.

Validasi kembali dilakukan terhadap model dilakukan untuk mengetahui performa model klasifikasi *Naïve Bayes* dengan data SMOTE. Validasi dilakukan dengan 10-*fold cross validation*. Data dibagi ke dalam 10-*fold* data *training testing*. Performa model dievaluasi dengan nilai *Accuracy*, *Presicion*, *Recall*, *F1 Score*, dan AUC. Hasil evaluasi model klasifikasi *Naïve Bayes* dengan SMOTE ditunjukkan pada Tabel 4.16.

**Tabel 4.16** Evaluasi Klasifikasi *Naïve Bayes* SMOTE dengan 10-*Fold*

| *Fold* | *Accuracy* | *Precision* | *Recall* | *F1 Score* | AUC |
|---|---|---|---|---|---|
| 1 | 0,75 | 0,65 | 0,71 | 0,67 | 0,84 |
| 2 | 0,75 | 0,63 | 0,72 | 0,65 | 0,84 |
| 3 | 0,74 | 0,64 | 0,72 | 0,65 | 0,84 |
| 4 | 0,74 | 0,65 | 0,73 | 0,66 | 0,85 |
| 5 | 0,72 | 0,61 | 0,73 | 0,63 | 0,84 |
| 6 | 0,75 | 0,64 | 0,73 | 0,67 | 0,85 |
| 7 | 0,77 | 0,66 | 0,76 | 0,68 | 0,86 |
| 8 | 0,73 | 0,62 | 0,71 | 0,65 | 0,83 |
| 9 | 0,75 | 0,65 | 0,72 | 0,66 | 0,84 |
| 10 | 0,72 | 0,62 | 0,70 | 0,63 | 0,83 |
| Rata-rata | 0,74 | 0,64 | 0,72 | 0,66 | 0,84 |

Tabel 4.16 menunjukkan nilai evaluasi model 10-*fold* dengan menggunakan data SMOTE. Evaluasi model dilakukan menggunakan nilai *F1 Score* dan AUC. Rata-rata evaluasi model menunjukkan *F1 Score* sebesar 0,66 dan nilai AUC sebesar 0,84. Nilai *F1 Score* tersebut berarti bahwa pada model klasifikasi nilai *precision* dan *recall* cenderung kecil atau tidak seimbangnya nilai *precision* dan *recall*. Pada kasus ini terjadi kurangnya keseimbangan nilai *presicion* dan *recall* dengan nilai *precision* 0,64 lebih kecil dari nilai *recall* 0,72. Sedangkan nilai AUC menunjukkan model sudah lebih baik dalam mengklasifikasikan *tweet*. Metode SMOTE membuat data tidak mengalami kasus *imbalance* dan mengalami kenaikan *f1 score* serta nilai AUC. Berdasarkan *f1 score* dan nilai AUC, *fold* ke-7 merupakan model

dengan performa terbaik yang memiliki nilai *f1 score* sebesar 0,68 dan nilai AUC sebesar 0,86.

Jika dibandingkan dengan model *Naïve Bayes* pada data awal, terdapat beberapa perbedaan hasil evaluasi model. Perbandingan evaluasi model dilakukan dengan membandingkan rata-rata evaluasi model pada 10-*fold cross validation* antara model *Naïve Bayes* pada data awal dengan *Naïve Bayes* pada data SMOTE. Tabel 4.17 menunjukkan hasil perbandingan evaluasi model.

**Tabel 4.17** Perbandingan Evaluasi Model Klasifikasi *Naïve Bayes*

| Evaluasi Model | *Accuracy* | *Precision* | *Recall* | *F1 Score* | AUC |
|---|---|---|---|---|---|
| Data awal | 0,74 | 0,66 | 0,54 | 0,56 | 0,74 |
| Data SMOTE | 0,74 | 0,64 | 0,72 | 0,66 | 0,84 |

Tabel 4.17 menunjukkan perbedaan hasil klasifikasi antara model *Naïve Bayes* dengan data awal dan model *Naïve Bayes* dengan data SMOTE. Terjadi peningkatan nilai *recall* sebesar 0,18, peningkatan *f1 score* sebesar 0,10, dan peningkatan AUC sebesar 0,10. Pada nilai *precision* justru terjadi penurunan sebesar 0,02. Nilai *accuracy* tetap sama yaitu 0,74. Berdasarkan data tersebut disimpulkan model *Naïve Bayes* dengan SMOTE memiliki performa klasifikasi yang lebih baik karena mampu meningkatkan *f1 score* dan nilai AUC.

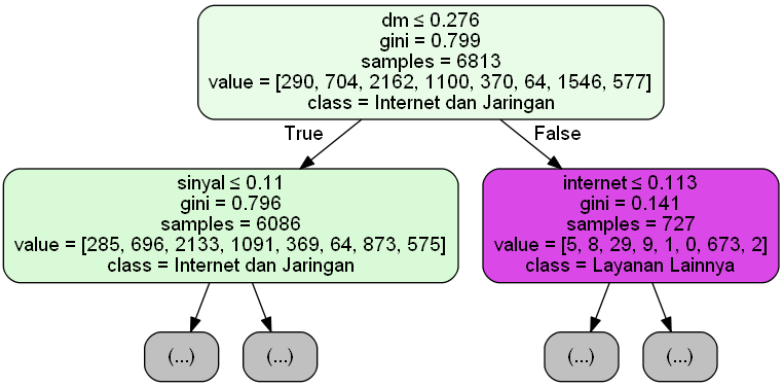
## 4.3 Klasifikasi Menggunakan *Random Forest*

Metode klasifikasi yang kedua adalah *Random Forest*. *Random Forest* merupakan metode klasifikasi berdasarkan pohon keputusan. Pembentukan pohon keputusan dilakukan dengan membuat *split node* yang memisahkan kategori berdasarkan nilai *gini impurity*. Dalam *Random Forest* dibentuk sejumlah pohon keputusan dan hasil klasifikasi kategori diperoleh berdasarkan *vote* terbanyak dari pohon keputusan. Sebelum dilakukan klasifikasi data akan dibagi ke dalam data *training* dan data *testing*. Data *traning* digunakan untuk membentuk model dan data *testing* digunakan untuk melakukan prediksi kategori. Pembagian data akan dilakukan dengan metode *K-Fold* dengan 10-*Fold*. *K-Fold*

*cross-validation* dilakukan untuk mengetahui kestabilan evaluasi model. Hasil evaluasi model untuk 10-*fold* kemudian di rata-rata untuk mengetahui keseluruhan evaluasi kebaikan model. Untuk mengetahui kebaikan model prediksi akan dilakukan perhitungan nilai *Accuracy*, *Presicion*, *Recall*, *F1 Score*, dan AUC.

### 4.3.1 Klasifikasi *Random Forest* dengan Data awal (Tanpa SMOTE)

Klasifikasi *Random Forest* dilakukan dengan membentuk *n* pohon keputusan. Pembentukan pohon keputusan dilakukan dengan memisahkan (*split*) kategori pada kumpulan *tweet* menjadi sehomogen mungkin dengan menggunakan variabel yang ada. Variabel dalam hal ini kata dalam *tweet* digunakan dalam memisahkan (*split*) *tweet* ke dalam kategori dengan melakukan *binary split* berdasarkan frekuensi masing-masing kata yang diperoleh pada tahap *term weighting*. Pemisahan ini dilakukan secara berulang dengan variabel-variabel yang ada sampai diperoleh kelompok *tweet* yang sehomogen mungkin untuk masing-masing kategori. Setelah proses ini maka terbentuk 1 pohon keputusan. Dalam *random forest*, pembentukan pohon keputusan dilakukan hingga terbentuk sebanyak B pohon keputusan. Salah satu pohon keputusan yang terbentuk pada model *Random Forest* ditunjukkan oleh Gambar 4.4.




**Gambar 4.4** Pohon Keputusan pada *Random Forest*

Pemilihan variabel *splitting* dilakukan dengan menghitung nilai *gini impurity*. Variabel yang memiliki selisih *gini impurity* yang tinggi dengan *gini impurity* sebelumnya terpilih menjadi variabel *splitting*. Dengan kata lain semakin rendah *gini impurity* maka variabel tersebut terpilih sebagai *splitting* variabel. Gambar 4.4 menunjukkan variabel yang menjadi variabel *splitting* pertama adalah variabel dm. Hal ini berarti variabel dm memiliki selisih *gini impurity* paling tinggi.

**Tabel 4.18** Perhitungan *Gini Impurity*

| x | *Subset* | Kategori | | | | | | | | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| Data awal (D) | - | 290 | 704 | 2162 | 1100 | 370 | 64 | 1546 | 577 | 6813 |
| dm ($x_1$) | $x_1 \leq 0{,}276$ | 285 | 696 | 2133 | 1091 | 369 | 64 | 873 | 575 | 6086 |
| | $x_1 > 0{,}276$ | 5 | 8 | 29 | 9 | 1 | 0 | 673 | 2 | 727 |
| xl_ prioritas ($x_2$) | $x_2 \leq 0{,}152$ | 0 | 1 | 0 | 0 | 175 | 0 | 0 | 0 | 176 |
| | $x_2 > 0{,}152$ | 290 | 703 | 2162 | 1100 | 195 | 64 | 1546 | 577 | 6637 |

Pada Tabel 4.18 terdapat data yang digunakan untuk menghitung *gini impurity*. *Gini impurity* dihitung dengan persamaan (2.11). Pada ilustrasi ini akan dibandingkan dua variabel untuk menentukan variabel *splitting* terbaik. Dua variabel tersebut adalah variabel dm ($x_1$) dan xl_prioritas ($x_2$). Variabel dm akan memisahkan *tweet* ke dalam dua bagian (*binary split*) yaitu subset $x_1 \leq 0{,}276$ dan $x_1 > 0{,}276$, sedangkan variabel xl_prioritas terjadi *binary split* yaitu untuk subset $x_2 \leq 0{,}152$ dan $x_2 > 0{,}152$. Berikut merupakan perhitungan *gini impurity* untuk masing-masing kasus.

$$\text{Gini (D)} = 1 - \left(\frac{290}{6813}\right)^2 - \left(\frac{704}{6813}\right)^2 - \ldots - \left(\frac{577}{6813}\right)^2 = 0,799 \qquad (4.1)$$

$$\text{Gini } (x_1 \leq 0{,}276) = 1 - \left(\frac{285}{6086}\right)^2 - \left(\frac{696}{6086}\right)^2 - \ldots - \left(\frac{575}{6086}\right)^2 = 0,796 \qquad (4.2)$$

$$\text{Gini}\ (x_1 > 0{,}276) = 1 - \left(\frac{5}{727}\right)^2 - \left(\frac{8}{727}\right)^2 - \ldots - \left(\frac{2}{727}\right)^2 = 0{,}141 \quad (4.3)$$

$$\text{Gini}\ (x_2 \le 0{,}152) = 1 - \left(\frac{0}{176}\right)^2 - \left(\frac{1}{176}\right)^2 - \ldots - \left(\frac{0}{176}\right)^2 = 0{,}011 \quad (4.4)$$

$$\text{Gini}\ (x_2 > 0{,}152) = 1 - \left(\frac{290}{6637}\right)^2 - \left(\frac{703}{6637}\right)^2 - \ldots - \left(\frac{577}{6637}\right)^2 = 0{,}791 \quad (4.5)$$

Berdasarkan perhitungan diperoleh nilai *gini impurity* untuk data awal (D) adalah sebesar 0,799. Pada variabel dm ($x_1$) dan xl_prioritas ($x_2$) terjadi *binary split* sehingga untuk menghitung *gini impurity* pada $x_1$ dan $x_2$ digunakan persamaan (2.13).

$$\text{Gini}\ (x_1) = \left(\frac{6086}{6813} \times 0{,}796\right) + \left(\frac{727}{6813} \times 0{,}141\right) = 0{,}727 \quad (4.6)$$

$$\text{Gini}\ (x_2) = \left(\frac{176}{6813} \times 0{,}011\right) + \left(\frac{6637}{6813} \times 0{,}791\right) = 0{,}770 \quad (4.7)$$

Setelah memperoleh nilai *gini impurity* pada masing-masing variabel maka akan ditentukan variabel terbaik yang menjadi variabel *splitting*. Variabel terbaik ditentukan berdasarkan selisih *gini impurity* variabel dengan *gini impurity* pada data awal. Variabel yang memberikan selisih terbesar akan terpilih menjadi variabel *splitting*. Selisih *gini impurity* dihitung dengan persamaan (2.14).

$$\Delta Gini(x_1) = 0{,}799 - 0{,}727 = 0{,}073 \quad (4.8)$$

$$\Delta Gini(x_2) = 0{,}799 - 0{,}770 = 0{,}029 \quad (4.9)$$

Perhitungan menunjukkan bahwa selisih *gini impurity* untuk variabel dm ($x_1$) lebih tinggi daripada variabel xl_prioritas ($x_2$) sehingga variabel dm terpilih menjadi variabel *splitting* pertama. Kemudian dilakukan lagi perhitungan dengan cara yang sama untuk pemilihan variabel *splitting* selanjutnya hingga terbentuk suatu pohon keputusan. Dalam *Random Forest* hal ini dilakukan hingga terbentuk *n* pohon keputusan. Jumlah pohon keputusan ditentukan oleh peneliti berdasarkan performa model terbaik. Jumlah pohon merupakan salah satu *hyperparameter* dalam *Random Forest* yang ditentukan oleh peneliti untuk mendapatkan performa model terbaik.

Pada *Random Forest* terdapat beberapa *hyperparameter* yang dapat diatur antara lain jumlah pohon keputusan yang dibentuk dan jumlah variabel yang dipertimbangkan dalam menentukan *split point*. Untuk mengetahui parameter terbaik pada model *Random Forest* dilakukan percobaan dengan menggunakan jumlah pohon sebanyak 100 dan 500 serta jumlah variabel yang dipertimbangkan dalam penentuan *splitting* yaitu 10, 23, dan 100 variabel. Evaluasi model yang digunakan untuk menentukan *hyperparameter* terbaik adalah *f1 score* dan nilai AUC. Parameter yang memiliki *f1 score* dan nilai AUC terbaik akan digunakan untuk membentuk model *random forest.* Tabel 4.19 menunjukkan hasil penentuan parameter terbaik.

**Tabel 4.19** Penentuan *Hyperparameter Random Forest*

| ***Tree*** | ***Feature*** | ***F1 Score*** | **AUC** |
|---|---|---|---|
| 100 | 10 | 0,7242 | 0,8372 |
| 100 | 23 | 0,7174 | 0,8379 |
| 100 | 100 | 0,7272 | 0,8424 |
| 500 | 10 | 0,7313 | 0,8423 |
| 500 | 23 | 0,7241 | 0,8418 |
| 500 | 100 | 0,7348 | 0,8471 |

Berdasarkan tabel 4.19 diketahui bahwa parameter terbaik adalah dengan menggunakan jumlah pohon sebanyak 500 dengan variabel yang dipertimbangkan pada *splitting* adalah 100 variabel. Pada klasifikasi *Random Forest*, data terlebih dahulu dibagi dengan 10-*fold cross validation*. Data *training* digunakan untuk membuat model klasifikasi dan data *testing* digunakan untuk memprediksi kategori. *Fold* terbaik digunakan untuk menunjukkan proses pembentukan model *random forest*. Model *Random Forest* dibuat berdasarkan parameter terbaik. Pohon keputusan yang terbentuk pada model klasifikasi *Random Forest* ditunjukkan pada Lampiran 5-7. Setelah model terbentuk dilakukan prediksi terhadap data *testing*. Hasil prediksi akan divalidasi dengan data aktual. Validasi prediksi kategori dijelaskan melalui *confusion matrix* pada Tabel 4.20.

**Tabel 4.20** *Confusion Matrix* Klasifikasi *Random Forest*

| Kategori Aktual | Kategori Prediksi | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 27 | 1 | 0 | 3 | 0 | 0 | 1 | 0 |
| 2 | 6 | 53 | 5 | 9 | 0 | 0 | 3 | 2 |
| 3 | 1 | 3 | 223 | 8 | 0 | 0 | 4 | 1 |
| 4 | 4 | 8 | 3 | 98 | 0 | 0 | 7 | 2 |
| 5 | 0 | 2 | 0 | 0 | 36 | 1 | 2 | 0 |
| 6 | 0 | 1 | 0 | 0 | 0 | 4 | 1 | 1 |
| 7 | 1 | 8 | 13 | 3 | 1 | 0 | 129 | 17 |
| 8 | 1 | 3 | 3 | 3 | 0 | 0 | 9 | 45 |

Berdasarkan *confusion matrix* pada Tabel 4.20, kategori 1 terdapat 27 *tweet* yang diklasifikasikan dengan benar dari total 32 *tweet*. Pada kategori 2 terdapat 53 *tweet* yang diklasifikasikan dengan benar dari total 78 *tweet*. Pada kategori 3 terdapat 223 *tweet* yang diklasifikasikan dengan benar dari total 240 *tweet*. Pada kategori 4 terdapat 98 *tweet* yang diklasifikasikan dengan benar dari total 122 *tweet*. Pada kategori 5 terdapat 36 *tweet* yang diklasifikasikan dengan benar dari total 41 *tweet*. Pada kategori 6 terdapat 4 *tweet* yang diklasifikasikan dengan benar dari total 7 *tweet*. Pada kategori 7 terdapat 129 *tweet* yang diklasifikasikan dengan benar dari total 172 *tweet*. Pada kategori 8 dengan jumlah *tweet* 64 terdapat 45 *tweet* yang diklasifikasikan dengan benar.

Selanjutnya performa model klasifikasi diketahui dengan kebaikan model. Kebaikan model dilakukan dengan menghitung evaluasi model. Evaluasi model yang digunakan adalah nilai *precision*, *recall*, *f1 score*, dan nilai AUC. Dalam kasus *multiclass* dimana terdapat lebih dari 2 kategori maka untuk mendapatkat nilai evaluasi model secara keseluruhan dilakukan pembobotan atau rata-rata. Rata-rata evaluasi model untuk keseluruhan kategori digunakan perhitungan *macro-average* sesuai dengan persamaan (2.22) hingga (2.26). Evaluasi model untuk model klasifikasi *Random Forest* ditunjukkan pada Tabel 4.21.

**Tabel 4.21** Evaluasi Model Klasifikasi *Random Forest*

| Kategori | *Accuracy* | *Precision* | *Recall* | *F1 Score* | AUC |
|---|---|---|---|---|---|
| 1 | 0,81 | 0,67 | 0,84 | 0,75 | 0,91 |
| 2 | 0,81 | 0,67 | 0,68 | 0,67 | 0,82 |
| 3 | 0,81 | 0,90 | 0,93 | 0,91 | 0,94 |
| 4 | 0,81 | 0,79 | 0,80 | 0,80 | 0,88 |
| 5 | 0,81 | 0,97 | 0,89 | 0,92 | 0,94 |
| 6 | 0,81 | 0,80 | 0,57 | 0,67 | 0,78 |
| 7 | 0,81 | 0,83 | 0,75 | 0,79 | 0,85 |
| 8 | 0,81 | 0,66 | 0,70 | 0,68 | 0,83 |
| Rata-rata | 0,81 | 0,79 | 0,77 | 0,77 | 0,87 |

Tabel 4.21 menunjukkan nilai rata-rata *accuracy* adalah 0,81, rata-rata *precision* adalah 0,79, rata-rata *recall* adalah 0,77, rata-rata *f1 score* adalah 0,77, dan rata-rata AUC adalah 0,87. *Precision* sebesar 0,79 berarti dari total prediksi terhadap suatu kategori, sebanyak 79% sesuai dengan kategori aktual. Nilai *precision* terbesar adalah pada kategori 5 yaitu sebesar 0,97 . Nilai terendah adalah pada kategori 8 yaitu sebesar 0,66. *Recall* sebesar 0,77 berarti bahwa dari total kategori aktual sebanyak 77% dapat diprediksi dengan benar. Nilai *recall* terbesar adalah pada kategori 3 yaitu sebesar 0,93. Nilai terendah adalah pada kategori 6 yaitu 0,57. *F1 score* sebesar 0,77 menunjukkan model cukup baik jika ditinjau berdasarkan keseimbangan *precision* dan *recall*. AUC rata-rata sebesar 0,87 menunjukkan model sudah sangat baik dalam mengklasifikasikan data *imbalance*.

Validasi terhadap model dilakukan untuk mengetahui performa model klasifikasi *Random Forest* secara rata-rata. Validasi dilakukan dengan 10-*fold cross validation*. Data dibagi ke dalam 10-*fold* data *training testing*. Performa model dievaluasi dengan nilai *Accuracy*, *Presicion*, *Recall*, *F1 Score*, dan AUC. Hasil evaluasi model klasifikasi *Random Forest* ditunjukkan pada Tabel 4.22.

**Tabel 4.22** Evaluasi Klasifikasi *Random Forest* dengan 10-*Fold*

| *Fold* | *Accuracy* | *Precision* | *Recall* | *F1 Score* | AUC |
|---|---|---|---|---|---|
| 1 | 0,82 | 0,76 | 0,72 | 0,73 | 0,84 |
| 2 | 0,80 | 0,71 | 0,69 | 0,69 | 0,83 |
| 3 | 0,81 | 0,79 | 0,77 | 0,77 | 0,87 |
| 4 | 0,80 | 0,74 | 0,70 | 0,71 | 0,83 |
| 5 | 0,79 | 0,74 | 0,70 | 0,71 | 0,83 |
| 6 | 0,81 | 0,80 | 0,74 | 0,76 | 0,86 |
| 7 | 0,83 | 0,78 | 0,76 | 0,77 | 0,87 |
| 8 | 0,81 | 0,75 | 0,72 | 0,73 | 0,84 |
| 9 | 0,81 | 0,75 | 0,70 | 0,72 | 0,84 |
| 10 | 0,81 | 0,78 | 0,74 | 0,75 | 0,86 |
| Rata-rata | 0,81 | 0,76 | 0,72 | 0,73 | 0,85 |

Tabel 4.22 menunjukkan nilai evaluasi model dengan menggunakan 10-*fold*. Dalam kasus data *imbalance* digunakan *f1 score* dan nilai AUC untuk mengetahui performa model. Rata-rata evaluasi model menunjukkan *f1 Score* sebesar 0,73 dan nilai AUC sebesar 0,85. *F1 Score* tersebut berarti bahwa model sudah cukup baik jika ditinjau berdasarkan keseimbangan nilai *precision* dan *recall*. Sedangkan nilai AUC menunjukkan model baik dalam mengklasifikasikan *tweet* meskipun dalam kasus *imbalance*. Berdasarkan *f1 score* dan nilai AUC, *fold* ke-3 merupakan model dengan performa klasifikasi terbaik.

### 4.3.2 Klasifikasi *Random Forest* dengan SMOTE

SMOTE dilakukan untuk mengatasi permasalahan *imbalance* pada data. SMOTE dilakukan terhadap data *training*. SMOTE dilakukan dengan membuat variabel sintetis pada kategori minoritas sehingga jumlah data pada kategori minoritas menjadi sama dengan kategori mayoritas. Ketika kategori minoritas dan mayoritas telah seimbang maka probabilitas hasil ketepatan klasifikasi untuk masing-masing kategori menjadi sama. Tabel 4.23 menunjukkan perbandingan jumlah data sebelum dan sesudah dilakukan SMOTE pada data awal.

**Tabel 4.23** Perbandingan Jumlah Data Awal dan Data SMOTE untuk Model *Random Forest*

| Jenis Data | Kategori | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| Data Awal | 290 | 704 | 2162 | 1100 | 370 | 64 | 1546 | 577 |
| Data SMOTE | 2162 | 2162 | 2162 | 2162 | 2162 | 2162 | 2162 | 2162 |

Setelah dilakukan SMOTE dan kategori pada data *tweet* menjadi seimbang selanjutnya dibentuk model klasifikasi dengan *Random Forest*. Untuk mengetahui parameter terbaik pada model *Random Forest* ini kembali dilakukan percobaan dengan menggunakan jumlah pohon sebanyak 100 dan 500 serta jumlah variabel yang dipertimbangkan dalam penentuan *splitting* yaitu 10, 23, dan 100 variabel. Evaluasi model yang digunakan sama dengan sebelumnya yaitu *f1 score* dan nilai AUC. Tabel 4.24 menunjukkan hasil penentuan parameter terbaik.

**Tabel 4.24** Penentuan *Hyperparameter Random Forest* dengan SMOTE

| Tree | Feature | F1 Score | AUC |
|---|---|---|---|
| 100 | 10 | 0,7499 | 0,8673 |
| 100 | 23 | 0,7485 | 0,8688 |
| 100 | 100 | 0,7395 | 0,8668 |
| 500 | 10 | 0,7560 | 0,8700 |
| 500 | 23 | 0,7517 | 0,8707 |
| 500 | 100 | 0,7448 | 0,8690 |

Berdasarkan tabel 4.24 ditunjukkan bahwa parameter terbaik untuk model *Random Forest* dengan SMOTE adalah jumlah pohon sebanyak 500 dan variabel yang dipertimbangkan pada *splitting* adalah 10 variabel. Pohon keputusan yang terbentuk pada *Random Forest* dengan SMOTE ditunjukan pada Lampiran 8-10. Selajutnya model *Random Forest* dibentuk dengan menggunakan data *training*. Setelah model terbentuk dilakukan prediksi terhadap data *testing*. Proses pembentukan model *random forest* ditunjukkan menggunakan *fold-3* pada 10-*fold cross validation.* Hasil prediksi akan divalidasi dengan data aktual.

Validasi prediksi kategori dijelaskan melalui *confusion matrix* pada Tabel 4.25.

**Tabel 4.25** *Confusion Matrix* Klasifikasi *Random Forest* dengan SMOTE

| **Kategori Aktual** | **Kategori Prediksi** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 29 | 2 | 0 | 1 | 0 | 0 | 0 | 0 |
| 2 | 4 | 59 | 3 | 7 | 0 | 0 | 2 | 3 |
| 3 | 2 | 5 | 221 | 6 | 1 | 0 | 1 | 4 |
| 4 | 6 | 8 | 2 | 96 | 2 | 1 | 4 | 3 |
| 5 | 0 | 2 | 0 | 0 | 38 | 1 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 5 | 1 | 1 |
| 7 | 2 | 5 | 11 | 3 | 2 | 0 | 129 | 20 |
| 8 | 1 | 2 | 4 | 3 | 0 | 1 | 8 | 45 |

Berdasarkan *confusion matrix* pada Tabel 4.25, kategori 1 terdapat 29 *tweet* yang diklasifikasikan dengan benar dari total 32 *tweet*. Pada kategori 2 terdapat 59 *tweet* yang diklasifikasikan dengan benar dari total 78 *tweet*. Pada kategori 3 terdapat 221 *tweet* yang diklasifikasikan dengan benar dari total 240 *tweet*. Pada kategori 4 terdapat 96 *tweet* yang diklasifikasikan dengan benar dari total 122 *tweet*. Pada kategori 5 terdapat 38 *tweet* yang diklasifikasikan dengan benar dari total 41 *tweet*. Pada kategori 6 terdapat 5 *tweet* yang diklasifikasikan dengan benar dari total 7 *tweet*. Pada kategori 7 terdapat 129 *tweet* yang diklasifikasikan dengan benar dari total 172 *tweet*. Pada kategori 8 terdapat 45 *tweet* yang diklasifikasikan dengan benar dari total 64 *tweet*.

Selanjutnya dilakukan perhitungan evaluasi model berdasarkan jumlah prediksi benar dan perdiksi salah yang terdapat pada Tabel 4.25. Evaluasi model dilakukan untuk mengetahui performa model klasifikasi. Evaluasi model yang digunakan adalah nilai *precision*, *recall*, *f1 score*, dan nilai AUC. Rata-rata nilai evaluasi model dihitung dengan menggunakan metode *macro-average* dengan persamaaan (2.22) hingga (2.26). Evaluasi model ditunjukkan pada Tabel 4.26.

**Tabel 4.26** Evaluasi Model Klasifikasi *Random Forest* dengan SMOTE

| Kategori | *Accuracy* | *Precision* | *Recall* | *F1 Score* | AUC |
|---|---|---|---|---|---|
| 1 | 0,87 | 0,66 | 0,91 | 0,76 | 0,94 |
| 2 | 0,87 | 0,71 | 0,76 | 0,73 | 0,86 |
| 3 | 0,87 | 0,92 | 0,92 | 0,92 | 0,94 |
| 4 | 0,87 | 0,83 | 0,79 | 0,81 | 0,88 |
| 5 | 0,87 | 0,88 | 0,93 | 0,90 | 0,96 |
| 6 | 0,87 | 0,62 | 0,71 | 0,67 | 0,85 |
| 7 | 0,87 | 0,89 | 0,75 | 0,81 | 0,86 |
| 8 | 0,87 | 0,59 | 0,70 | 0,64 | 0,83 |
| Rata-rata | 0,87 | 0,76 | 0,81 | 0,78 | 0,89 |

Tabel 4.26 menunjukkan nilai rata-rata *accuracy* adalah 0,87, rata-rata *precision* adalah 0,76, rata-rata *recall* adalah 0,81, rata-rata *f1 score* adalah 0,78, dan rata-rata AUC adalah 0,89. *Precision* sebesar 0,76 berarti dari total prediksi terhadap suatu kategori, sebanyak 76% sesuai dengan kategori aktual. Nilai *precision* terbesar adalah pada kategori 3 yaitu sebesar 0,92. Nilai terendah adalah pada kategori 8 yaitu sebesar 0,59. *Recall* sebesar 0,81 berarti bahwa dari total kategori aktual sebanyak 81% dapat diprediksi dengan benar. Nilai *recall* terbesar adalah pada kategori 5 yaitu sebesar 0,93. Nilai terendah adalah pada kategori 8 yaitu 0,70. *F1 score* sebesar 0,78 menunjukkan model cukup baik jika ditinjau berdasarkan keseimbangan *precision* dan *recall*. AUC rata-rata sebesar 0,89 menunjukkan model sudah sangat baik dalam mengklasifikasikan data *imbalance*.

Selanjutnya dilakukan validasi terhadap model untuk mengetahui performa model klasifikasi *Random Forest* secara rata-rata. Validasi dilakukan dengan 10-*fold cross validation*. Data dibagi ke dalam 10-*fold* data *training testing*. Performa model dievaluasi dengan nilai *Accuracy*, *Presicion*, *Recall*, *F1 Score*, dan AUC. Hasil evaluasi model klasifikasi *Random Forest* ditunjukkan pada Tabel 4.27.

**Tabel 4.27** Evaluasi Klasifikasi *Random Forest* SMOTE dengan 10-*Fold*

| *Fold* | *Accuracy* | *Precision* | *Recall* | *F1 Score* | AUC |
|---|---|---|---|---|---|
| 1 | 0,83 | 0,79 | 0,77 | 0,77 | 0,87 |
| 2 | 0,81 | 0,71 | 0,71 | 0,70 | 0,84 |
| 3 | 0,82 | 0,76 | 0,81 | 0,78 | 0,89 |
| 4 | 0,81 | 0,73 | 0,76 | 0,74 | 0,86 |
| 5 | 0,81 | 0,75 | 0,75 | 0,75 | 0,86 |
| 6 | 0,82 | 0,79 | 0,79 | 0,79 | 0,88 |
| 7 | 0,85 | 0,77 | 0,79 | 0,78 | 0,88 |
| 8 | 0,82 | 0,76 | 0,76 | 0,75 | 0,87 |
| 9 | 0,82 | 0,76 | 0,76 | 0,76 | 0,87 |
| 10 | 0,81 | 0,74 | 0,76 | 0,74 | 0,86 |
| Rata-rata | 0,82 | 0,76 | 0,77 | 0,76 | 0,87 |

Tabel 4.27 menunjukkan nilai evaluasi model dengan menggunakan 10-*fold*. Rata-rata evaluasi model menunjukkan *f1 Score* sebesar 0,76 dan nilai AUC sebesar 0,87. *F1 Score* tersebut berarti bahwa model sudah cukup baik jika ditinjau berdasarkan keseimbangan nilai *precision* dan *recall*. Sedangkan nilai AUC menunjukkan model sudah sangat baik dalam menglasifikasikan *tweet* meskipun dalam kasus *imbalance*. Berdasarkan *f1 score* dan nilai AUC, *fold* ke-3 merupakan model dengan performa klasifikasi terbaik.

Perbandingan dengan model sebelumnya dilakukan untuk mengetahui performa penggunaan SMOTE pada klasifikasi *Random Forest*. Perbandingan dilakukan berdasarkan rata-rata nilai evaluasi model pada 10-*fold cross validation* antara data awal dan data dengan SMOTE. Tabel 4.28 menunjukkan perbandingan evaluasi model antara model *Random Forest* dengan data awal dan model *Random Forest* dengan SMOTE.

**Tabel 4.28** Perbandingan Evaluasi Model Klasifikasi *Random Forest*

| Evaluasi Model | *Accuracy* | *Precision* | *Recall* | *F1 Score* | AUC |
|---|---|---|---|---|---|
| Data awal | 0,81 | 0,76 | 0,72 | 0,73 | 0,85 |
| Data SMOTE | 0,82 | 0,76 | 0,77 | 0,76 | 0,87 |

Tabel 4.28 menunjukkan nilai *accuracy*, *precision*, *recall*, *f1 score*, dan AUC mengalami peningkatan. Berdasarkan Tabel 4.27 tidak terdapat perbedaan yang signifikan antara model *Random Forest* dengan data awal dan model *Random Forest* dengan SMOTE. Nilai *accuracy* hanya mengalami kenaikkan sebesar 0,10, *f1 score* mengalami kenaikan 0,30, dan nilai AUC hanya mengalami kenaikan sebesar 0,20. Kenaikan yang cukup tinggi adalah pada nilai *recall* yaitu sebesar 0,50 sedangkan pada nilai *precision* tetap sama. Dari data tersebut terlihat bahwa penggunaan SMOTE tidak terlalu mengubah performa klasifikasi *Random Forest* karena dengan data awal saja model *Random Forest* sudah mampu untuk menglasifikasikan dengan baik bahkan pada kasus *imbalance* yang dapat dilihat pada nilai *f1 score* dan AUC yang cukup baik.

## 4.4 Evaluasi Model Klasifikasi Terbaik

Dalam menentukan model klasifikasi terbaik dilakukan dengan membandingkan performa model melalui nilai evaluasi model. Pada kasus *imbalance* penggunaan *f1 score* dan nilai AUC merupakan cara yang tepat untuk menunjukkan keseimbangan performa klasifikasi terhadap kelas mayor dan kelas minor. Model yang dibandingkan antara lain model Naïve Bayes, *Naïve Bayes* dengan SMOTE, *Random Forest*, dan *Random Forest* dengan SMOTE. Tabel 4.29 menunjukkan nilai *f1 score* dan nilai AUC untuk masing-masing metode.

**Tabel 4.29** Evaluasi Model Klasifikasi Terbaik

| **Metode Klasifikasi** | ***F1 Score*** | **AUC** |
|---|---|---|
| Naïve Bayes | 0,56 | 0,74 |
| *Naïve Bayes* SMOTE | 0,66 | 0,84 |
| *Random Forest* | 0,73 | 0,85 |
| *Random Forest* SMOTE | 0,76 | 0,87 |

Berdasarkan Tabel 4.29 diketahui bahwa metode klasifikasi terbaik dilihat dari *f1 score* dan AUC adalah metode *Random Forest* SMOTE dengan nilai *f1 score* yaitu 0,76 dan nilai AUC yaitu 0.87. Selain itu diketahui bahwa dengan menggunakan

SMOTE untuk menangani kasus *imbalance* dapat meningkatkan performa klasifikasi khususnya untuk metode Naïve Bayes. Pada *Random Forest*, SMOTE memberikan kenaikan terhadap *f1 score* dan nilai AUC tapi tidak menunjukkan peningkatan yang tinggi.

# BAB V
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Berdasarkan analisis dan pembahasan yang telah dilakukan pada bab 4, maka diperoleh kesimpulan sebagai berikut.

1. Dalam data *tweet* yang ditunjukkan terhadap layanan *customer care* XL Axiata pada akun Twitter @myXLCare terdapat 8 kategori pembahasan diantaranya kategori pulsa, akun, internet dan jaringan, produk, XL Prioritas, telepon dan sms, layanan lain yang terkait dengan penipuan dan respon *customer service*, dan *tweet* yang tidak membahas terkait layanan XL. Kata "nomor", "sinyal", "pulsa", "paket", "jaringan", "internet", dan "kuota" memiliki frekuensi kemunculan yang tinggi. Hal ini menunjukkan permasalahan dan pertanyaan terkait layanan-layanan tersebut tinggi. Dapat dilihat bahwa sebagian besar kata-kata tersebut berhubungan dengan internet yang menunjukkan bahwa permasalahan internet memang masih tinggi. Kata "dm" juga memiliki frekuensi tinggi yang menunjukkan bahwa permasalahan atau informasi banyak juga yang disampaikan via *direct message* pada Twitter. Terdapat juga kata "tolong" dengan frekuensi tinggi yang berarti bahwa pada data *tweet* ini tidak hanya berisi seputar pertanyaan tentang layanan tetapi juga permintaan bantuan untuk menyelesaikan permasalahan.
2. Pada data *tweet* yang ditunjukkan terhadap layanan *customer care* XL Axiata pada akun Twitter @myXLCare terdapat kasus *imbalance* data. Hal ini terjadi karena terdapat beberapa kategori yang memiliki sedikit *tweet*. Untuk mengatasi hal ini dilakukan *oversampling* dengan metode *Synthetic Minority Oversampling Technique* (SMOTE). Klasifikasi *tweet* dilakukan terhadap data awal dan data dengan *oversampling* SMOTE. Model klasifikasi terbaik yang diperoleh adalah dengan menggunakan metode

*Random Forest* yang menghasilkan *f1 score* sebesar 0.76 dan nilai AUC sebesar 0.87. Penggunaan SMOTE mampu meningkatkan performa model *Naïve Bayes* dengan cukup tinggi sedangkan untuk model *Random Forest* penggunaan SMOTE hanya meningkatkan sedikit performa klasifikasi. Tanpa menggunakan SMOTE, *Random Forest* sudah menghasilkan performa model klasifikasi yang baik.

## 5.2 Saran

Berdasarkan kesimpulan yang diperoleh, dapat dirumuskan saran sebagai pertimbangan penelitian selanjutnya adalah sebagai berikut.

1. Dalam melakukan klasifikasi dokumen seperti *tweet*, kalimat, dll, tahap *preprocessing* menjadi sangat penting. Memilih variabel yang tepat sangat menentukan hasil performa model klasifikasi. Selain itu juga dapat dilakukan *feature selection* sebelum melakukan klasifikasi karena variabel pada data berupa kata-kata yang sangat banyak.
2. Permasalahan internet masih menjadi permasalahan yang sering muncul dalam dunia telekomunikasi. Dengan berkembanganya kecepatan dan teknologi internet, penyedia layanan sebaiknya mampu menjaga kualitas. Selain itu pemasalahan respon yang tanggap pada layanan *customer service* melalui aplikasi Twitter juga perlu ditingkatkan.

## DAFTAR PUSTAKA


Abualigah, L. M., Khader, A. T., & Betar, M. A. A. (2016). *Unsupervised Feature Selection Technique Based on Genetic Algorithm for Improving the Text Clustering*. Jordan: IEEE.

Adriani, M., Asian, J., Nazief, B., Tahaghoghi, S. M., & Williams, H. E. (2007). Stemming Indonesian: A Confix-Stripping Approach. *ACM Transactions on Asian Language Information Processing (TALIP).* 6(4). 1-33.

Allahyari, M., Pouriyeh, S.A., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B., & Kochut, K.J. (2017). *A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques*. New York: Cornell University.

Analytics Vidhya. (2015). *Tuning the parameters of your Random Forest model*. https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/. Diakses pada 20 Februari 2019.

Ariadi, D., & Fithriasari, K. (2015). Klasifikasi berita Indonesia Menggunakan Bayesian Classification dan Support Vector Machine dengan Confix Stripping Stemmer. *Jurnal Sains dan Seni ITS*. 4(2). 248-253.

Ariansyah, Kasmad. (2014). *Proyeksi Jumlah Pelanggan Telepon Bergerak Seluler di Indonesia*. Jakarta: Puslitbang Sumber Daya dan Perangkat Pos dan Informatika.

Bekkar, M., Djemaa, D. K., & Alitouche, D. A. (2013). Evaluation Measure for Models Assument over Imbalanced Data Sets. *Journal of Internation Engineering and Applications.* 3. 27-36.

Bholowalia, P., & Kumar, A. (2014). EBK-Means: A Clustering Technique based on Elbow Method and K-Means in WSN. *International Journal of Computer Applications*. 105(9). 17-24.

Breiman, L. (2001). Random Forest. *Machine Learning*. 45. 5-32.

Budiyanto, S. (2010). *Analisis Jumlah Operator Selular Indonesia dengan Chaos Teori*. Jakarta: Universitas Indonesia.

Castellà, Q., & Sutton, C. (2014). *Word Storms: Multiples of Word Clouds for Visual Comparison of Documents*. New York: Cornell University.

Cavnar, W., & Trenkle, J. (1994). N-Gram-Based Text Categorization. *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*. 1. 161-175.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*. 16. 321-357.

Daniel, J., & James, H. M. (2014). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. New Jersey: Prentice Hall.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*. 27(8). 861-874.

Feldman, R., & Sanger, J. (2007). *The Text Mining Handbook : Advanced Approaches in Analyzing Unsrtructed Data*. New York: Cambridge University Press.

Flach, P. (2012). *Machine Learning: The Art and Science of Algorithms that Makes Sense of Data*. New York: Cambridge University Press.

Han, J. (2012). *Data mining: Concepts and techniques, third edition (3rd ed.)*. Waltham, Mass.: Morgan Kaufmann Publishers.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction (2nd ed.)*. New York: Springer.

Hidayatullah, F., & Maarif, M. (2017). Pre-processing Tasks in Indonesian Twitter Messages. *Journal of Physics: Conference Series*. 801(1). 012072.

Huurdeman, Anton A. (2005). *The Worldwide History of Telecommunications*. New Jersey: John Wiley & Sons, Inc.

Iriawan, N., Fithriasari, K., Ulama, B. S. S., Suryaningtyas, W., & Pangastuti, S. S. (2018). On The Comparison: Random Forest, SMOTE-Bagging, and Bernoulli Mixture to Classify Bidikmisi Dataset in East Java. *2018 International Conference on Computer Engineering, Network and Intelligent Multimedia (CENIM)*. 1. 137-141.

ITU-D ICT Statistics. (2014). *Mobile-cellular telephone subscriptions*. http://www.itu.int/en/ITUD/ Statistics/Documents/statistics/2014/Mobile_cellular_2000-2013.xls. Diakses pada 20 Februari 2019.

Jannah, S. Z., Fithriasari, K., & Usagawa, T. (2018). *Clustering and Visualizing Surabaya Citizen Aspirations by Using Text Mining*, Surabaya: Institut Teknologi Sepuluh Nopember.

Johnson, R. A., & Wichern, D. W. (2007). *Applied multivariate statistical analysis (6th ed.)*. Upper Saddle River, N.J.: Pearson Prentice Hall.

Kaggle. (2017). *Resampling strategies for imbalanced datasets*. https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets. Diakses pada 18 Februari 2019.

Kamus Besar Bahasa Indonesia. https://kbbi.kemdikbud.go.id/. Diakses pada 18 Februari 2019.

Krouska, A., Troussas, C., & Virvou, M. (2016). The effect of preprocessing techniques on Twitter sentiment analysis. *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)*. 1. 1-5.

Krüger, Frank. (2016). *Activity, Context, and Plan Recognition with Computational Causal Behaviour Models*. Germany: University of Rostock.

Kumar, Lokesh. (2013). Text Mining: Concepts, Process And Applications. *Journal of Global Research in Computer Science*. 4. 36-39.

Kusumawardani, R. P., & Prayoginingsih, S. (2018). Klasifikasi Data Twitter Pelanggan Berdasarkan Kategori myTelkomsel Menggunakan Metode Support Vector Machine (SVM) Studi Kasus: Telekomunikasi Selular. *Jurnal Sisfo*. 7. 83-98.

Liaw, A., & Wiener, M. (2002). Classification and Regression by RandomForest. *R News*. 2(3). 18-22.

MacQueen, J. (1967). Some Methods for Classification and Analysis of Multivariate Observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. 1. 281-297.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. New York: Cambridge University Press.

Mitchell, T. M. (1997). *Machine Learning*. Boston: McGraw-Hill.

Prabowo, D. A. (2016). TF-IDF Enhanced Genetic Algorithm untuk Extractive Automatic Text Summarization. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*. 3(3). 208-215.

Santos, M. S., Soares, J. P., Abreu, P. H., Araujo, H., & Santos, J. (2018). Cross-Validation for Imbalanced Datasets: Avoiding Overoptimistic and Overfitting Approaches. *IEEE Computational Intelligence Magazine*. 13(4). 59-76.

Sisodia, D. S., & Verma, U. (2018). The Impact of Data Re-Sampling on Learning Performance of Class Imbalanced Bankruptcy Prediction Models. *International Journal on Electrical Engineering and Informatics*. 10(3). 433-446.

Sonagara, D., & Badheka, S. (2014). Comparison of Basic Clustering Algorithms. *International Journal of Computer Science and Mobile Computing*. 3(10). 58-61.

Srivastava, A., & Sahami, M. (2009). *Text Mining Classification, Clustering, and Application.* USA: Taylor and Francis Group, LLC.

Tala, F. Z. (2003). *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia.* Amsterdam: Universiteit van Amsterdam.

Twitter. (2019). *Twitter Support*. http://support.twitter.com/. Diakses pada 18 Februari 2019.

Vidya, N. A., Fanany, M. I., & Budi, I. (2015). Twitter Sentiment to Analyze Net Brand Reputation of Mobile Phone Providers. *Procedia Computer Science*. 72. 83-98.

Waegel, D. (2006) *The Development of Text-Mining Tools and Algorithms*. Pennsylvania: Ursinus College.

Witten, I.H., Frank, E., & Hall, M.A., (2011). *Data Mining Practical Machine Learning Tools and Technique Third Edition*. New York: Morgan Kaufmann.

Xin, Luo. (2016). *A New Text Classifier Based on Random Forests*. China: Atlantis Press.

Yang, Y. (2000). An Evaluation of Statistical Approaches to Text Categorization. *Journal of Information Retrieval*. 1(1-2). 69–90.

*(Halaman ini sengaja dikosongkan)*

# LAMPIRAN

**Lampiran 1.** Data *Tweet* terhadap Akun @myXLCare.

| no | kalimat_tweet |
|---|---|
| 1 | halo @myXLCare udah 5 hari sinyal di sentul city jelek, sering 3G dan bahkan Edge. LTE pun cuma dapet 1 bar dan useless, tiap hujan sinyal drop. tolong perbaikannya, terima kasih. |
| 2 | @myXLCare Bagaimana caranya? Kalo akses Twitter saja lemot <U+0001F621> |
| 3 | @myXLCare Signal segitu akses google, Facebook, messenger, Twitter dan game online lemot Katanya udah di perbaiki kok gak perubahan? https://t.co/Ow3EhXKEkY |
| 4 | Hai @myXLCare @XLAxiata_ID sinyal di daerah karet kuningan, Setiabudi Barat, Jakarta Selatan dinaikin dong speednya. Masa bagus speednya cuma di jam 00.00 - 09.00 doang? |
| 5 | @myXLCare Kenapa saya mw aktifin waze ga bisa |
| 7 | @myXLCare Tidak apa-apa Maya, itu bukan salahmu. Tak perlulah itu maaf-maaf segala. Andai saja xl memiliki paket semurah indosat tentu pasti aku akan berpaling padamu |
| 8 | @myXLCare Maya, tolong bantu agar @IndosatCare dapat menyelesaikan pengaduan saya. Terimakasih |
| 7891 | @myXLCare @ask_AXIS apakah axis dan xl ada yang berbeda? aoakah dari kecepatannya atau dari apa nya yang bisa berbeda? |

**Lampiran 2.** Syntax Karakteristik Data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
from collections import Counter
```

```
df = pd.read_excel("df_label_fix.xlsx")
text=df['kalimat_tweet']
text.head(10)

mpl.rcParams['figure.figsize']=(12.0,6.0)
mpl.rcParams['font.size']=12
mpl.rcParams['savefig.dpi']=100
mpl.rcParams['figure.subplot.bottom']=.1

kata = [s.lower().split() for s in text if s]
noline_ = [sublist for l in kata for sublist in l]

counts1 = dict(Counter(noline_).most_common(15))
labels1, values1 = zip(*counts1.items())

indSort1 = np.argsort(values1)[::-1]
labels1 = np.array(labels1)[indSort1]
values1 = np.array(values1)[indSort1]
indexes1 = np.arange(len(labels1))

mybar = plt.bar(indexes1, values1)
# get rid of the frame
for spine in plt.gca().spines.values():
    spine.set_visible(False)
# remove all the ticks and directly label each bar with
respective value
plt.tick_params(top='off', bottom='off', left='off', right='off',
labelleft='off', labelbottom='on')
# direct label each bar with Y axis values
for bari in mybar:
    height = bari.get_height()
    plt.gca().text(bari.get_x() + bari.get_width()/2,
bari.get_height() + 300, str(int(height)), ha='center',
color='black', fontsize=10)
# add labels
```

```
plt.xticks(indexes1, labels1)
plt.show()

df1 = pd.read_csv("TFIDF 2105.csv")
text1 = df1['kalimat_tweet']
df1.head
text1.head()

kata1 = [s.lower().split() for s in text1 if s]
noline_1 = [sublist for l in kata1 for sublist in l]

counts2 = dict(Counter(noline_1).most_common(15))
labels2, values2 = zip(*counts2.items())

indSort2 = np.argsort(values2)[::-1]
labels2 = np.array(labels2)[indSort2]
values2 = np.array(values2)[indSort2]
indexes2 = np.arange(len(labels2))

values2

mybar = plt.bar(indexes2, values2)
# get rid of the frame
for spine in plt.gca().spines.values():
    spine.set_visible(False)
# remove all the ticks and directly label each bar with
respective value
plt.tick_params(top='off', bottom='off', left='off', right='off',
labelleft='off', labelbottom='on')
# direct label each bar with Y axis values
for bari in mybar:
    height = bari.get_height()
    plt.gca().text(bari.get_x() + bari.get_width()/2,
bari.get_height() + 300, str(int(height)), ha='center',
color='black', fontsize=10)
```

```
# add labels
plt.xticks(indexes2, labels2)
plt.show()

label = df1.label
counts3 = dict(Counter(label))
labels3, values3 = zip(*counts3.items())
values3

indSort3 = np.argsort(labels3)
labels3 = np.array(labels3)[indSort3]
values3 = np.array(values3)[indSort3]
values3.sum()

label_text=["Pulsa", "Akun", "Internet dan Jaringan",
"Produk", "XL Pioritas", "Telepon dan SMS", "Layanan
Lainnya", "Irrelevant"]
newlabel = []
for item in labels3:
    item = label_text[item-1]
    newlabel.append(item)
newlabel=np.array(newlabel)

mpl.rcParams['figure.figsize']=(13.0,7.5)
mpl.rcParams['font.size']=14
mpl.rcParams['savefig.dpi']=100
mpl.rcParams['figure.subplot.bottom']=.1

colorset= ['#00D3A0', '#00aaae', '#005BA2', '#66217C',
'#F36B2E', '#FBB644', '#D81F58', '#EF4C85']

fig, ax = plt.subplots()

wedges, texts, a = ax.pie(values3,
wedgeprops=dict(width=0.4), colors=colorset,
```

```
                    startangle=90, autopct='%1.1f%%',
pctdistance=0.8,
                    textprops=dict(color="w"))

bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k")
kw = dict(arrowprops=dict(arrowstyle="-"),
        bbox=bbox_props, zorder=0, va="center")

for i, p in enumerate(wedges):
   ang = (p.theta2 - p.theta1)/2. + p.theta1
   y = np.sin(np.deg2rad(ang))
   x = np.cos(np.deg2rad(ang))
   horizontalalignment = {-1: "right", 1: "left"}[int(np.sign(x))]
   connectionstyle = "angle,angleA=0,angleB={}".format(ang)
   kw["arrowprops"].update({"connectionstyle":
connectionstyle})
   ax.annotate(newlabel[i], xy=(x, y), xytext=(1.3*np.sign(x),
1.2*y),
             horizontalalignment=horizontalalignment, **kw)

ax.axis('equal')
plt.tight_layout()
plt.show()
```

**Lampiran 3.** Syntax *Preprocessing*.

```
import numpy as np
import pandas as pd
import re
import string
from Sastrawi.Stemmer.StemmerFactory import
StemmerFactory
from collections import OrderedDict
from collections import Counter
```

```
#remove link
df_link = []
for line in text:
result = re.sub(r"http\S+", " ", line)
df_link.append(result)

#remove RT
df_RT = []
for line in df_link:
result = re.sub(r"RT", " ", line)
df_RT.append(result)

#remove enter
df_enter = []
for line in df_RT:
result =  re.sub("\n"," ",line)
df_enter.append(result)

#remove mention
df_mention = []
for line in df_enter:
result = re.sub(r"@\S+", " ", line)
df_mention.append(result)

#remove hastag
df_hastag = []
for line in df_mention :
result = re.sub(r"#\S+","",line)
df_hastag.append(result)

#remove emoticon
df_emot = []
for line in df_hastag :
result = re.sub(r'<.*?>',"",line)
df_emot.append(result)
```

```
#remove punctuation
df_punct = []
for line in df_emot :
result = re.sub(r"[^\w\s]"," ",line)
df_punct.append(result)

#remove space
df_space = []
for line in df_punct :
result = re.sub(r'\s+',' ',line)
df_space.append(result)

#lowercase
df_lower = []
for line in df_space :
result = line.lower()
df_lower.append(result)
kata={"no service":"tidak ada layanan", "mw":"mau",
"signal":"sinyal", "terimakasih":"terima kasih", " makasih":"
terima kasih", " yaa ":" ya ", "gaosah":"tidak usah", " enggak":"
tidak", " engga":" tidak", " nggak":" tidak"," ngga":" tidak",
"udh":"udah", "nomer":"nomor", "telpon":"telepon", "combo
vip":"combo-vip", "combo lite":"combo-lite", "combo
prima":"combo-prima", "yg":" yang ", "dgn":"dengan",
"ngk":"tidak", "nggk":"tidak", "dlu":" dulu", " tp":" tapi",
"tx":"terima kasih", "komplen":"komplain"," ujan ":" hujan ", "
no ":" nomor ", " kl ":" kalo ", "spt":"seperti", "tlp":"telepon",
"qta":"kita", "lewot":"lambat", " lg":" lagi", "bgt":"banget", "
tnya":" tanya", "extra combo":"xtra-combo",
"extracombo":"xtra-combo", "xtra combo":"xtra-combo",
"extra kuota":"xtra-kuota", "xtra kuota":"xtra-kuota",
"xlprioritas":"xl-prioritas", "xlprio":"xl-prioritas", "xl
prioritas":"xl-prioritas", "xl prio":"xl-prioritas",
```

"xl pass":"xl-pass", "xlpass":"xl-pass", "xl home":"xl-home", "xlhome":"xl-home", "extra":"xtra", "ekstra kuota":"xtra-kuota", "xl care":"xlcare", "combo sosial":"combo-social", "combo social":"combo-social", "sya":"saya", "izzi":"izi", "xlgo":"xl go", "xl go izi":"xl-go-izi", "xl izi go":"xl-go-izi", "xl go":"xl-go-izi", "xl izi":"xl-go-izi", "go izi":"xl-go-izi", "izi go":"xl-go-izi", "super ngebut":"super-ngebut", "pop up":"pop-up", "dikit2":"sedikit", "tpi":"tapi", "psti":"pasti", "mlm":"malam", "psti":"pasti", "skrg":"sekarang", "help":"tolong", " jaring ":" jaringan ", "sidah":"sudah", "nmer":"nomor", "intrntn":"internet", "internetan":"internet", "topup":"top-up", "top up":"top-up", " lemot":" lambat", "tdk":"tidak", "kabwor":"kabur", "snyl":"sinyal", "brhnti":"berhenti","brlangganan":"berlangganan", "intrntn":"internet", "aktipin":"aktif", " ty":" terima kasih", "program2":"program", "aktifin":"aktif", "lemoooootttt":"lambat", "nelpon":"telepon", "sticker":"stiker", "telp ":"telepon ","sinyale":"sinyal","sinyall":"sinyal", "jaringn":"jaringan", "jringan":"jaringan", "pket":"paket", "pkt":"paket", "apliksi":"aplikasi", "dwnload":"download", "donlot":"download", "quota":"kuota", " ilang":" hilang", "hilng":"hilang", " ancur ":" hancur ", " kep ":" kepulauan ", "hpnya":"hp", "pnya":"punya", "service":"servis", " ga ":" tidak ", "tq":"terima kasih", " gak":" tidak", "hhilang":"hilang", " enek ":" enak ", "hangustidak":"hangus tidak", "kmarin":"kemarin", " karu ":" kartu ", "prtama":"pertama", "tifak":"tidak", " sel ":" telkomsel ", " cs ":" customer service ", "intrnet":"internet", "jringnn":"jaringan", "bgus":"bagus", " krg ":" kurang ", "disc ":"diskon ", "discount":"diskon", "plsa":"pulsa", "pnjls":"penjelasan", "tlg":"tolong", "tdi":"tadi", " ngadu":" mengadu", "matoin":"matikan", " ngeh ":" sadar ", " prio ":" prioritas ", "trimakasih":"terima kasih", "maph":"maaf", " lola ":" lambat ", "lelaaaahhh":"lelah", " km ":" kamar ", "trims":"terima kasih", "telfon":"telepon", " lemotttt ":" lambat ",

"parahhhh":"parah", " nmor":" nomor", " nmr":" nomor", "
gabisa":" tidak bisa", " nanya":" tanya","pake ":"pakai ",
"sampe ":"sampai ", " tau ":" tahu ", "thanks":"terima kasih", "
abis":" habis", "bikin":"buat", "eror":"error",
"kemaren":"kemarin","lelet":"lambat",
"telpan":"telepon","oprator":"operator",
"internetnya":"internet", "my xl":"myxl", "please":"tolong", "
thx":" terima kasih", "cust ":"customer ","servis":"service", "
cs":" customer service", "tgl":"tanggal", "aktifasi":"aktivasi", "
apps":" aplikasi"," app":" aplikasi", "bales":"balas",
"balikin":"kembalikan", "bapuk":"jelek","bls":"balas",
"bosen":"bosan", "capek":"lelah","centre":"center",
"cepet":"cepat", "check":"cek", "complain":"komplain",
"connection":"koneksi", "konek ":"connect ", "dibales":"balas",
"dibenerin":"perbaikin","benerin":"perbaikin",
"direspon":"respon", "direstart":"restart",
"ditindaklanjuti":"tindak lanjut","dmnya":"dm", "duit":"uang",
"feb ":"februari ", "hape":"hp", "hilang2an":"hilang",
"history":"histori", "hot rod":"hotrod", "hotroad":"hotrod", "hot
road":"hotrod", " ig ":" instagram ", "log in":"login", "sign
in":"sign-in", "sign up":"sign-up", "inet":"internet",
"informasi":"info", "info":"informasi", "jelasin":"jelaskan",
"sosial media":"sosmed", "kab ":"kabupaten ", " kasi ":" kasih
", "kec ":"kecamatan ", "pengecekan":"cek",
"kenceng":"kencang", "kesel":"kesal",
"kmrn":"kemarin","konter":"counter", "kouta":"kuota",
"legends":"legend", "maap":"maaf", "maen":"main",
"maketin":"paket", "paketin":"paket", "make":"pakai",
"malem":"malam", "males":"malas", "matiin":"mati", "
motong":" potong", "muter2":"putar2", "nelfon":"telepon",
"ngasih":"kasih", "ngelag":"lag", " ngeluh":" mengeluh", "
ngirim":" mengirim", " ngisi":" mengisi", "non
aktif":"nonaktif", "notif ":"notifikasi ", " nunggu":" tunggu",
"nyalain":"nyala", " nyedot":" sedot", " nyuruh":" suruh", "org
":"orang ", "pasca bayar":"pascabayar", "pra bayar":"prabayar",

```
"screen shoot":"screenshot", "semalem":"semalam",
"servicenya":"service", "silahkan":"silakan", "sim card":"sim-
card", "simcard":"sim-card", "prirotas":"prioritas",
"priorital":"prioritas", "intrrnet":"internet", " rb ":" ribu", "regis
":"registrasi ", "regist ":"registrasi ", "register":"registrasi",
"respond":"respon", "responnya":"respon", "screen
shot":"screenshot", "screen shoot":"screenshot",
"semalem":"semalam", "servicenya":"service",
"silahkan":"silakan", "sim card":"sim-card", "simcard":"sim-
card", "speednya":"speed", "temen":"teman", "tks":"terima
kasih", "tlong":"tolong", "twit ":"tweet ", " wa ":" whatsapp ",
"xlku":"xl", "xlnya":"xl", " ml ":" mobile legend ",
"priorotas":"prioritas", "priorital":"prioritas",
"intrrnet":"internet", "prior ":"prioritas ", "priority":"prioritas",
"pls":"pulsa", "trobel":"trouble", "reg2":"register",
"jarigan":"jaringan", "nonaktifkan":"nonaktif",
"prioritasnya":"prioritas", " nternetnya":" internet",
"lamabat":"lambat", "mlempem":"lambat", "barar":"bayar",
"mandet":"lambat", " hub ":" telepon ", "4gnya":"4g", "2g
":"two_generation ", "3g ":"three_generation ", "4g
":"four_generation ", "5g ":"five_generation "}

def replace_all(text, dic):
for i, j in dic.items():
text = text.replace(i, j)
return text

dic = OrderedDict(kata)
df_sinonim = []
for line in df_lower:
result = replace_all(line, dic)
df_sinonim.append(result)

#remove number
df_number = []
```

```
for line in df_sinonim:
result = re.sub(r"\d+", "", line)
df_number.append(result)

#stopwords
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
stopWords = set(stopwords.words('indonesian'))

stopword = open("idn stopwords.txt","r").read()
stopword = set(stopword.split())
not_stopword = {'tidak', 'bisa', 'ada', 'kurang'}
new_stopword = set([word for word in stopword if not word in
not_stopword])

#Stemming
factory = StemmerFactory()
stemmer = factory.create_stemmer()
data_stemmed = map(lambda x: stemmer.stem(x), df_stop)
df_stemmed = list(data_stemmed)

vectorizer = TfidfVectorizer(min_df=1)
vec = vectorizer.fit_transform(df_final)
TFIDF = pd.DataFrame(vec.toarray(), columns =
vectorizer.get_feature_names())
TFIDF['label'] = df['true_label']
TFIDF['kalimat_tweet']=df_final
TFIDF['sum_features'] = TFIDF.drop(['label'],
axis=1).sum(axis=1)
TFIDF = TFIDF.loc[TFIDF['sum_features'] !=
0].drop(['sum_features'], axis=1)
#TFIDF.to_csv('TFIDF 2105.csv')
```

**Lampiran 4.** Syntax Klasifikasi.

```
import pandas as pd
import numpy as np
import collections
from statistics import mean
from sklearn import naive_bayes, model_selection, datasets,
metrics
from sklearn.metrics import confusion_matrix
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import precision_recall_fscore_support
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold,
GridSearchCV
from imblearn.over_sampling import SMOTE
from sklearn.metrics import precision_score, recall_score,
f1_score, roc_auc_score, make_scorer
from sklearn.preprocessing import LabelBinarizer
from sklearn.tree import export_graphviz
import warnings
warnings.filterwarnings("ignore")

data_DTM = pd.read_csv('DTM 2105.csv')
data_TFIDF = pd.read_csv('TFIDF 2105.csv')

label_DTM = data_DTM['label']
feature_DTM = data_DTM.loc[:, "2g":"youtube"]
label_TFIDF = data_TFIDF['label']
feature_TFIDF = data_TFIDF.loc[:, "2g":"youtube"]

feature_TFIDF

def multiclass_roc_auc_score(y_test, pred, average=None):
```

```
    lb = LabelBinarizer()
    lb.fit(y_test)
    y_test = lb.transform(y_test)
    pred = lb.transform(pred)
    return roc_auc_score(y_test, pred, average=average)
mb1 = MultinomialNB()
model_nb = []
pred_nb = []
pred_proba = []
prior_prob_nb = []
con_prob = []
x_train_nb = []
x_test_nb = []
y_train_nb = []
y_test_nb = []
skf = StratifiedKFold(n_splits=10, shuffle=True,
random_state=1)
skf.get_n_splits(feature_TFIDF, label_TFIDF)
for train, test in skf.split(feature_TFIDF, label_TFIDF):
    x_train, x_test = feature_TFIDF.iloc[train],
feature_TFIDF.iloc[test]
    y_train, y_test = label_TFIDF.iloc[train],
label_TFIDF.iloc[test]
    x_train_nb.append(x_train)
    x_test_nb.append(x_test)
    y_train_nb.append(y_train)
    y_test_nb.append(y_test)
    model = mb1.fit(x_train, y_train)
    model_nb.append(model)
    feature_prob = mb1.feature_log_prob_
    con_prob.append(feature_prob)
    prior_prob = mb1.class_log_prior_
    prior_prob_nb.append(prior_prob)
    pred = mb1.predict(x_test)
    pred_nb.append(pred)
```

```
    pred_prb = mb1.predict_proba(x_test)
    pred_proba.append(pred_prb)
    print("Klasifikasi Naive Bayes : ")
    print (confusion_matrix(y_test, pred))
    print("Akurasi Score :
{:.2f}".format(metrics.accuracy_score(y_test, pred)))
    print("Precision Score :
{:.2f}".format(precision_score(y_test, pred, average='macro')))
    print("Recall Score : {:.2f}".format(recall_score(y_test,
pred, average='macro')))
    print("F1 Score : {:.2f}".format(f1_score(y_test, pred,
average='macro')))
    print("AUC Score : ", multiclass_roc_auc_score(y_test,
pred, average='macro'))
    print ("------------------------------------------------------")
conf_matrix = confusion_matrix(y_test_nb[6], pred_nb[6])
conf_matrix
np.savetxt("conf matrix nb.csv", conf_matrix, delimiter=",")
prior_prob_nb[6]
con_prob[6]
np.savetxt("conprob vld.csv", con_prob[6], delimiter=",")
pred_proba[6]
np.savetxt("nbprob vld.csv", pred_proba_smote[6],
delimiter=",")
best_data_nb = x_train_nb[6]
best_data_nb['label']=y_train_nb[6]
best_data_nb.to_csv('data nb.csv')
Akurasi_Score = metrics.accuracy_score(y_test_nb[6],
pred_nb[6])
Precision_Score = precision_score(y_test_nb[6], pred_nb[6],
average=None)
Recall_Score = recall_score(y_test_nb[6], pred_nb[6],
average=None)
F1_Score = f1_score(y_test_nb[6], pred_nb[6], average=None)
```

```
AUC_Score = multiclass_roc_auc_score(y_test_nb[6],
pred_nb[6])
eval_nb = pd.DataFrame(list(zip(Precision_Score,
Recall_Score, F1_Score, AUC_Score)),
                columns=["Precision", "Recall", "F1 Score",
"AUC"])
eval_nb
eval_nb.to_csv('eval nb.csv')

smote = SMOTE('not majority', random_state=10)
model_nb_smote = []
pred_nb_smote = []
pred_proba_smote = []
prior_prob_smote = []
con_prob_smote = []
x_train_nb_smote = []
x_test_nb_smote = []
y_train_nb_smote = []
y_test_nb_smote = []
skf = StratifiedKFold(n_splits=10, shuffle=True,
random_state=1)
skf.get_n_splits(feature_TFIDF, label_TFIDF)
for train, test in skf.split(feature_TFIDF, label_TFIDF):
    x_train, x_test = feature_TFIDF.iloc[train],
feature_TFIDF.iloc[test]
    y_train, y_test = label_TFIDF.iloc[train],
label_TFIDF.iloc[test]
    x_sm, y_sm = smote.fit_sample(x_train, y_train)
    x_train_nb_smote.append(x_sm)
    x_test_nb_smote.append(x_test)
    y_train_nb_smote.append(y_sm)
    y_test_nb_smote.append(y_test)
    model = mb1.fit(x_sm, y_sm)
    model_nb_smote.append(model)
    prior_prob = mb1.class_log_prior_
```

```
    prior_prob_smote.append(prior_prob)
    feature_prob = mb1.feature_log_prob_
    con_prob_smote.append(feature_prob)
    pred = mb1.predict(x_test)
    pred_nb_smote.append(pred)
    pred_prb = mb1.predict_proba(x_test)
    pred_proba_smote.append(pred_prb)
    print("Klasifikasi Naive Bayes : ")
    print (confusion_matrix(y_test, pred))
    print("Akurasi Score :
{:.2f}".format(metrics.accuracy_score(y_test, pred)))
    print("Precision Score :
{:.2f}".format(precision_score(y_test, pred, average='macro')))
    print("Recall Score : {:.2f}".format(recall_score(y_test,
pred, average='macro')))
    print("F1 Score : {:.2f}".format(f1_score(y_test, pred,
average='macro')))
    print("AUC Score : ", multiclass_roc_auc_score(y_test,
pred, average='macro'))
    print ("------------------------------------------------------")
conf_matrix_smote = confusion_matrix(y_test_nb_smote[6],
pred_nb_smote[6])
conf_matrix_smote
np.savetxt("conf matrix nb smote.csv", conf_matrix_smote,
delimiter=",")
con_prob_smote[6]
np.savetxt("conprob_smote vld.csv", con_prob_smote[6],
delimiter=",")
prior_prob_smote[6]
np.savetxt("prior_smote vld.csv", prior_prob_smote[6],
delimiter=",")
pred_proba_smote[6]
np.savetxt("nbprob smote vld.csv", pred_proba_smote[6],
delimiter=",")
```

```
Akurasi_Score = metrics.accuracy_score(y_test_nb_smote[6],
pred_nb_smote[6])
Precision_Score = precision_score(y_test_nb_smote[6],
pred_nb_smote[6], average=None)
Recall_Score = recall_score(y_test_nb_smote[6],
pred_nb_smote[6], average=None)
F1_Score = f1_score(y_test_nb_smote[6], pred_nb_smote[6],
average=None)
AUC_Score = multiclass_roc_auc_score(y_test_nb_smote[6],
pred_nb_smote[6])
eval_nb_smote = pd.DataFrame(list(zip(Precision_Score,
Recall_Score, F1_Score, AUC_Score)),
                  columns=["Precision", "Recall", "F1 Score",
"AUC"])
eval_nb_smote
eval_nb_smote.to_csv('eval nb smote.csv')

ntree = (100, 500)
mtry = (10, 23, 100)
skf = StratifiedKFold(n_splits=10, shuffle=True,
random_state=1)
skf.get_n_splits(feature_TFIDF, label_TFIDF)
for x in ntree:
   for y in mtry:
      fscore_list = []
      auc_list = []
      rf1 = RandomForestClassifier(n_estimators = x,
max_features = y, random_state=1)
      for train, test in skf.split(feature_TFIDF, label_TFIDF):
         x_train, x_test = feature_TFIDF.iloc[train],
feature_TFIDF.iloc[test]
         y_train, y_test = label_TFIDF.iloc[train],
label_TFIDF.iloc[test]
         model = rf1.fit(x_train, y_train)
         pred = rf1.predict(x_test)
```

```
            fscore = f1_score(y_test, pred, average='macro')
            fscore_list.append(fscore)
            auc = multiclass_roc_auc_score(y_test, pred,
average='macro')
            auc_list.append(auc)
        avg_fscore = mean(fscore_list)
        avg_auc = mean(auc_list)
        print('tree :', x, 'var :', y, 'F1-Score :', avg_fscore, 'AUC :',
avg_auc)

rf_model = []
model_rf = []
pred_rf = []
x_train_rf = []
x_test_rf = []
y_train_rf = []
y_test_rf = []
skf = StratifiedKFold(n_splits=10, shuffle=True,
random_state=1)
skf.get_n_splits(feature_TFIDF, label_TFIDF)
for train, test in skf.split(feature_TFIDF, label_TFIDF):
    rf = RandomForestClassifier(n_estimators = 500,
max_features = 100, oob_score=True, random_state=1)
    rf_model.append(rf)
    x_train, x_test = feature_TFIDF.iloc[train],
feature_TFIDF.iloc[test]
    y_train, y_test = label_TFIDF.iloc[train],
label_TFIDF.iloc[test]
    x_train_rf.append(x_train)
    x_test_rf.append(x_test)
    y_train_rf.append(y_train)
    y_test_rf.append(y_test)
    model = rf.fit(x_train, y_train)
    model_rf.append(model)
    pred = rf.predict(x_test)
```

```
    pred_rf.append(pred)
    print("Klasifikasi Random Forest : ")
    print (confusion_matrix(y_test, pred))
    print("Akurasi Score :
{:.2f}".format(metrics.accuracy_score(y_test, pred)))
    print("Precision Score :
{:.2f}".format(precision_score(y_test, pred, average='macro')))
    print("Recall Score : {:.2f}".format(recall_score(y_test,
pred, average='macro')))
    print("F1 Score : {:.2f}".format(f1_score(y_test, pred,
average='macro')))
    print("AUC Score : ", multiclass_roc_auc_score(y_test,
pred, average='macro'))
    print ("----------------------------------------------------")

conf_matrix_rf = confusion_matrix(y_test_rf[2], pred_rf[2])
conf_matrix_rf
np.savetxt("conf matrix rf.csv", conf_matrix_rf, delimiter=",")
Akurasi_Score = metrics.accuracy_score(y_test_rf[2],
pred_rf[2])
Precision_Score = precision_score(y_test_rf[2], pred_rf[2],
average=None)
Recall_Score = recall_score(y_test_rf[2], pred_rf[2],
average=None)
F1_Score = f1_score(y_test_rf[2], pred_rf[2], average=None)
AUC_Score = multiclass_roc_auc_score(y_test_rf[2],
pred_rf[2])
eval_rf = pd.DataFrame(list(zip(Precision_Score,
Recall_Score, F1_Score, AUC_Score)),
                       columns=["Precision", "Recall", "F1 Score",
"AUC"])
eval_rf
eval_rf.to_csv('eval rf.csv')
from sklearn.externals.six import StringIO
from IPython.display import Image
```

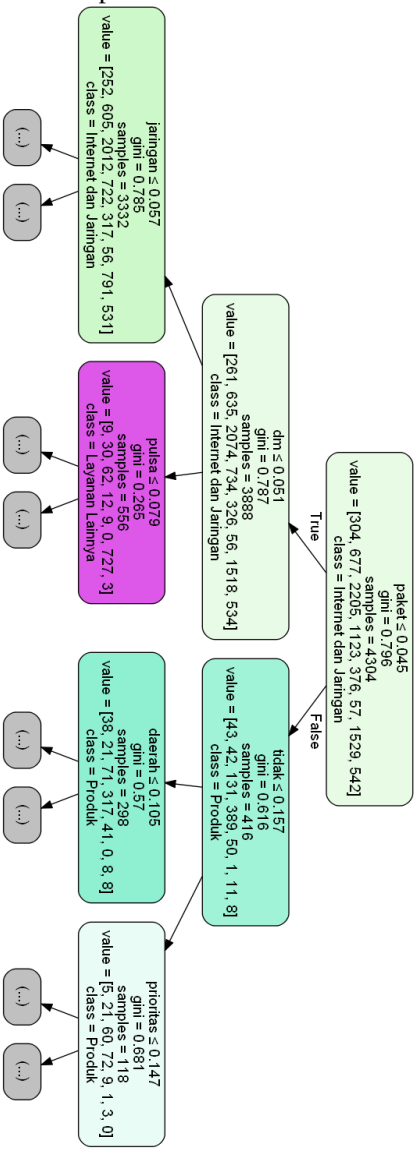
```
from sklearn.tree import export_graphviz
import pydotplus
target_names=["Pulsa", "Akun", "Internet dan Jaringan",
"Produk", "XL Pioritas",
          "Telepon dan SMS", "Layanan Lainnya", "Irrelevant"]
tree = rf_model[2].estimators_[499]
dot_data = StringIO()
export_graphviz(tree, max_depth=2, out_file=dot_data,
filled=True, rounded=True,
              special_characters=True, feature_names =
x_train_rf[2].columns.values,
              class_names=target_names, precision = 3)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
dtree=DecisionTreeClassifier(random_state=1)
dtree.fit(x_train_rf[2],y_train_rf[2])
train_sample_rf = x_train_rf[2].copy()
train_sample_rf['label'] = y_train_rf[2].copy()
train_sample_rf.to_csv('sample rf.csv')
dot_data = StringIO()
export_graphviz(dtree, out_file=dot_data,
              filled=True, rounded=True,
              special_characters=True, max_depth=2,
              feature_names = x_train_rf[2].columns.values,
              class_names=target_names)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
ntree = (100, 500)
mtry = (10, 23, 100)
smote = SMOTE('not majority', random_state=2)
skf = StratifiedKFold(n_splits=10, shuffle=True,
random_state=1)
skf.get_n_splits(feature_TFIDF, label_TFIDF)
for x in ntree:
   for y in mtry:
```

```
        fscore_list = []
        auc_list = []
        rf1 = RandomForestClassifier(n_estimators = x,
max_features = y, random_state=10)
        for train, test in skf.split(feature_TFIDF, label_TFIDF):
            x_train, x_test = feature_TFIDF.iloc[train],
feature_TFIDF.iloc[test]
            y_train, y_test = label_TFIDF.iloc[train],
label_TFIDF.iloc[test]
            x_sm, y_sm = smote.fit_sample(x_train, y_train)
            model = rf1.fit(x_sm, y_sm)
            pred = rf1.predict(x_test)
            fscore = f1_score(y_test, pred, average='macro')
            fscore_list.append(fscore)
            auc = multiclass_roc_auc_score(y_test, pred,
average='macro')
            auc_list.append(auc)
        avg_fscore = mean(fscore_list)
        avg_auc = mean(auc_list)
        print('tree :', x, 'var :', y, 'F1-Score :', avg_fscore, 'AUC :',
avg_auc)
smote = SMOTE('not majority', random_state=2)
rf_model_sm = []
model_rf_sm = []
pred_rf_sm = []
x_train_rf_sm = []
x_test_rf_sm = []
y_train_rf_sm = []
y_test_rf_sm = []
skf = StratifiedKFold(n_splits=10, shuffle=True,
random_state=1)
skf.get_n_splits(feature_TFIDF, label_TFIDF)
for train, test in skf.split(feature_TFIDF, label_TFIDF):
    rf = RandomForestClassifier(n_estimators = 500,
max_features = 10, oob_score=True, random_state=10)
```

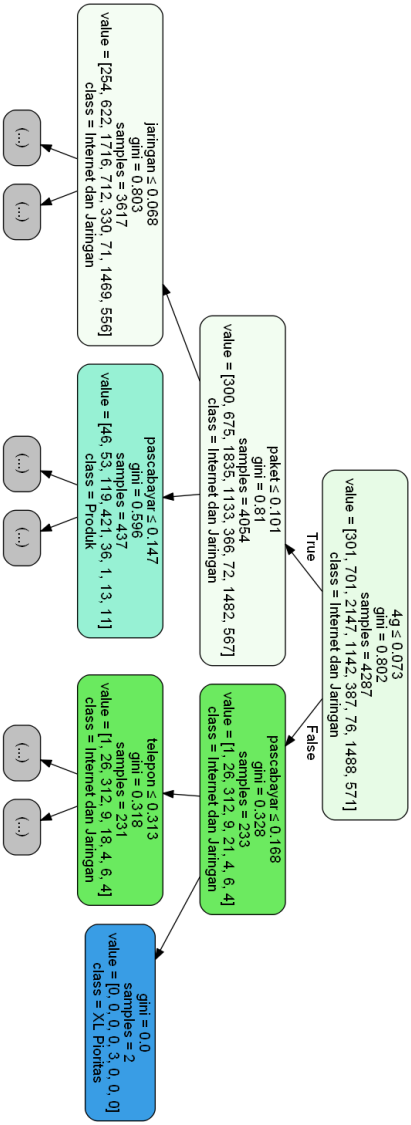
```
    rf_model_sm.append(rf)
    x_train, x_test = feature_TFIDF.iloc[train],
feature_TFIDF.iloc[test]
    y_train, y_test = label_TFIDF.iloc[train],
label_TFIDF.iloc[test]
    x_sm, y_sm = smote.fit_sample(x_train, y_train)
    x_train_rf_sm.append(x_sm)
    x_test_rf_sm.append(x_test)
    y_train_rf_sm.append(y_sm)
    y_test_rf_sm.append(y_test)
    model = rf.fit(x_sm, y_sm)
    model_rf_sm.append(model)
    pred = rf.predict(x_test)
    pred_rf_sm.append(pred)
    print("Klasifikasi Random Forest : ")
    print (confusion_matrix(y_test, pred))
    print("Akurasi Score :
{:.2f}".format(metrics.accuracy_score(y_test, pred)))
    print("Precision Score :
{:.2f}".format(precision_score(y_test, pred, average='macro')))
    print("Recall Score : {:.2f}".format(recall_score(y_test,
pred, average='macro')))
    print("F1 Score : {:.2f}".format(f1_score(y_test, pred,
average='macro')))
    print("AUC Score : ", multiclass_roc_auc_score(y_test,
pred, average='macro'))
    print ("------------------------------------------------------")

conf_matrix_rf_sm = confusion_matrix(y_test_rf_sm[2],
pred_rf_sm[2])
conf_matrix_rf_sm
np.savetxt("conf matrix rf sm.csv", conf_matrix_rf_sm,
delimiter=",")
Akurasi_Score = metrics.accuracy_score(y_test_rf_sm[2],
pred_rf_sm[2])
```

```
Precision_Score = precision_score(y_test_rf_sm[2],
pred_rf_sm[2], average=None)
Recall_Score = recall_score(y_test_rf_sm[2], pred_rf_sm[2],
average=None)
F1_Score = f1_score(y_test_rf_sm[2], pred_rf_sm[2],
average=None)
AUC_Score = multiclass_roc_auc_score(y_test_rf_sm[2],
pred_rf_sm[2], average=None)
eval_rf_sm = pd.DataFrame(list(zip(Precision_Score,
Recall_Score, F1_Score, AUC_Score)),
                  columns=["Precision", "Recall", "F1 Score",
"AUC"])
eval_rf_sm
eval_rf_sm.to_csv('eval rf sm.csv')
target_names=["Pulsa", "Akun", "Internet dan Jaringan",
"Produk", "XL Pioritas",
          "Telepon dan SMS", "Layanan Lainnya", "Irrelevant"]
tree = rf_model_sm[2].estimators_[499]
dot_data = StringIO()
export_graphviz(tree, max_depth=2, out_file=dot_data,
filled=True, rounded=True,
              special_characters=True, feature_names =
x_train_rf[2].columns.values,
              class_names=target_names, precision = 3)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

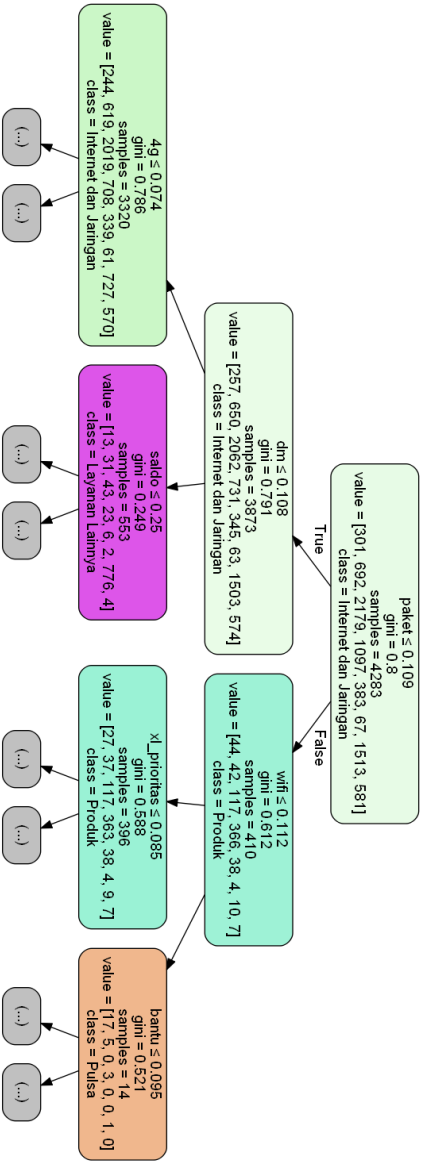
**Lampiran 5.** Pohon Keputusan ke-1 *Random Forest*



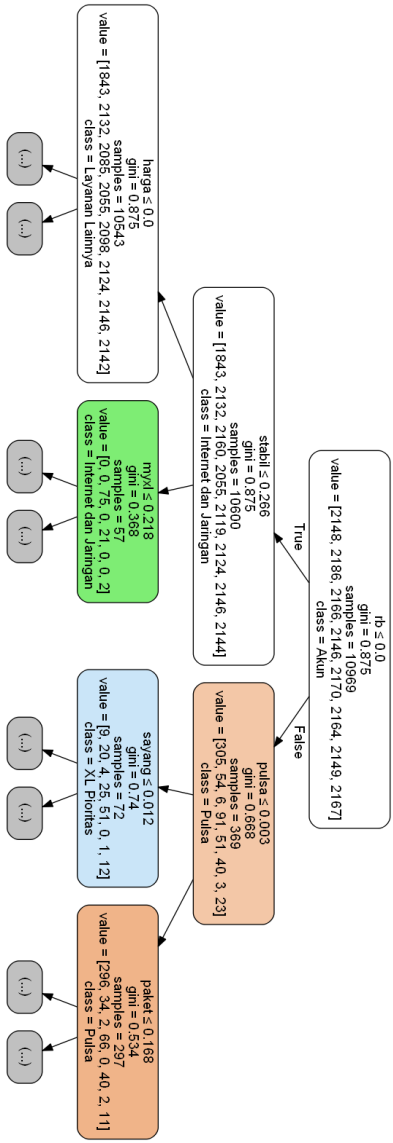
**Lampiran 6.** Pohon Keputusan ke-2 *Random Forest*



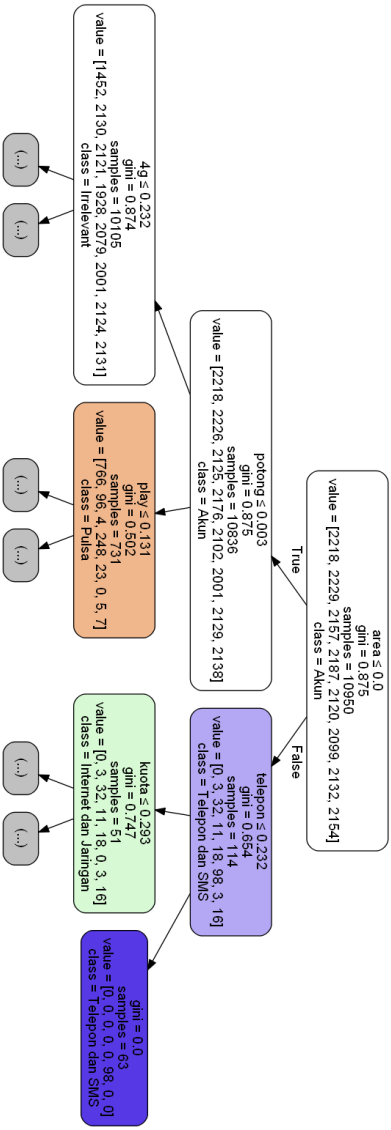
**Lampiran 7.** Pohon Keputusan ke-500 *Random Forest*



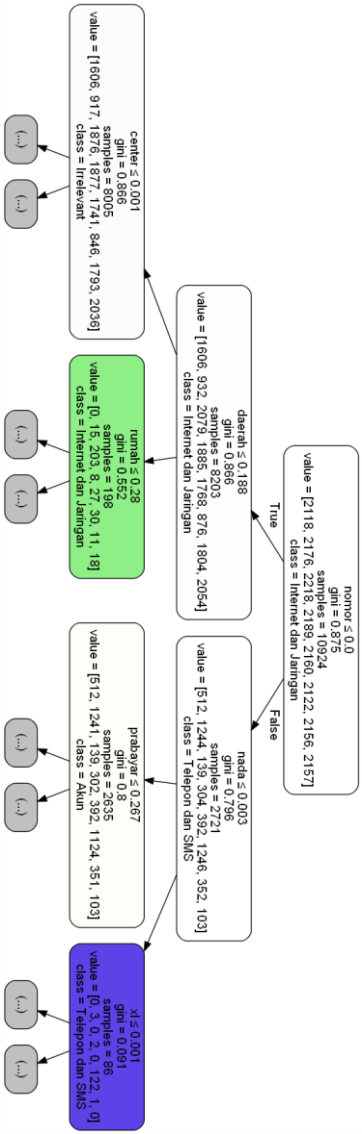
**Lampiran 8.** Pohon Keputusan ke-1 *Random Forest* dengan SMOTE



**Lampiran 9.** Pohon Keputusan ke-2 *Random Forest* dengan SMOTE



**Lampiran 10.** Pohon Keputusan ke-500 *Random Forest* dengan SMOTE



**Lampiran 11.** Surat Pernyataan Data

## SURAT PERNYATAAN

Saya yang bertanda tangan di bawah ini, mahasiswa Departemen Statistika FMKSD ITS:

Nama : Gede Narendra Saguna

NRP : 06211540000126

menyatakan bahwa data yang digunakan dalam Tugas Akhir/ Thesis ini merupakan data sekunder yang diambil dari ~~penelitian / buku/ Tugas Akhir/ Thesis/~~ publikasi lainnya yaitu:

Sumber : Twitter API (*Application Program Interface*)

Keterangan : Data *tweet* dengan *keyword* "@myXLCare"

Surat Pernyataan ini dibuat dengan sebenarnya. Apabila terdapat pemalsuan data maka saya siap menerima sanksi sesuai aturan yang berlaku.

Mengetahui
Pembimbing Tugas Akhir

Surabaya, Juli 2019

Dr. Dra. Kartika Fithriasari, M.Si.
NIP. 19691212 199303 2 002

Gede Narendra Saguna
NRP. 062115 4000 0126

*(coret yang tidak perlu)

# BIODATA PENULIS

Penulis lahir di Denpasar, 18 November 1997 dengan nama lengkap Gede Narendra Saguna, biasa dipanggil Narendra. Penulis menempuh pendidikan formal di SDN 1 Kerobokan Kaja, SMPN 3 Mengwi, dan SMAN 1 Denpasar. Kemudian penulis diterima sebagai mahasiswa Departemen Statistika ITS pada tahun 2015. Selama masa perkuliahan, penulis aktif di beberapa organisasi mahasiswa diantaranya Himpunan Mahasiswa Statistika ITS (HIMASTA-ITS) sebagai Staff Departemen Kesejahteraan Mahasiswa pada periode 2016/2017 dan Ketua Departemen Kesejahteraan Mahasiswa pada periode 2017/2018, Badan Eksekutif Mahasiswa (BEM-ITS) sebagai Staff Kementrian Advokasi dan Kesejahteraan Mahasiswa pada periode 2016/2018. Penulis pernah melaksanakan kerja praktik di Telkomsel, Jakarta. Selain itu, Penulis pernah mengikuti Studi Ekskursi ke Dalian University of Technology, China. Bagi pembaca yang ingin berdiskusi, memberikan saran, dan kritik mengenai Tugas Akhir ini dapat disampaikan melalui email saguna.narendra.gede@gmail.com.