MODEL PEMBELAJARAN DAN LAPORAN AKHIR
PROJECT-BASED LEARNING
MATA KULIAH MACHINE LEARNING
KELAS B

"PERBANDINGAN KINERJA ALGORITMA KLASIFIKASI UNTUK KATEGORISASI BERITA DETIKCOM DENGAN MENGGUNAKAN METODE DECISION TREE, SVM, DAN NAIVE BAYES"

DISUSUN OLEH KELOMPOK "III" :

1. ALAY MIRZA SAFIRA (21083010039)
2. FIQIH PAVITA A. (21083010042)
3. HAJJAR AYU CAHYANI (21083010044)
4. MUHAMAD HARIS HARTANTO (21083010045)

DOSEN PENGAMPU:

Dr. Eng. Ir. ANGGRAINI PUSPITA SARI, S.T., M.T.

AVIOLLA TERZA DAMALIANA, S.Si., M.Stat.

PROGRAM STUDI SAINS DATA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

JAWA TIMUR

2023

SURAT PERNYATAAN

Kami yang bertanda tangan di bawah ini:

1. NAMA : Alya Mirza Safira
   NPM : 21083010039
2. NAMA : Fiqih Pavita Andharluana
   NPM : 21083010042
3. NAMA : Hajjar Ayu Cahyani
   NPM : 21083010044
4. NAMA : Muhamad Haris Hartanto
   NPM : 21083010045

Dengan ini menyatakan bahwa hasil pekerjaan yang kami serahkan sebagai bagian dari penilaian mata kuliah Machine Learning adalah benar-benar karya orisinal kami, bukan milik orang lain, dan tidak pernah digunakan dalam penilaian tugas yang lain dalam mata kuliah apapun, baik secara keseluruhan ataupun sebagian, di Universitas Pembangunan Nasional "Veteran" Jawa Timur ataupun di institusi lainnya. Apabila di kemudian hari terbukti bahwa kami melakukan kecurangan maka kami bersedia menerima sanksi sesuai dengan aturan yang berlaku di Universitas Pembangunan Nasional "Veteran" Jawa Timur.

Surabaya, 15 Juni 2023




Alya Mirza S. Fiqih Pavita A. Hajjar Ayu C. Muhamad Haris H.

ABSTRAK

Di Indonesia media online menjadi salah satu media yang paling banyak digunakan masyarakat untuk mendapatkan informasi berita. Platform-platform berita online memungkinkan kita untuk mengakses berita dalam hitungan detik dari berbagai sumber di seluruh dunia. Dengan hanya menggunakan smartphone dan koneksi internet, kita dapat mengakses berita dari mana saja dan kapan saja. Oleh karena itu penelitian ini dilakukan untuk melakukan kategorisasi berita secara otomatis dengan menggunakan algoritma klasifikasi. Algoritma klasifikasi yang digunakan antara lain Decision Tree, Support vector machine dan naïve bayes. Berdasarkan hasil analisis klasifikasi, didapatkan bahwa SVM memiliki akurasi tertinggi pada pembagian data 90:10 dengan akurasi 86.7%. Naïve bayes memiliki akurasi tertinggi 74.9% dan Decision tree memiliki akurasi tertinggi 72.4%.

Kata kunci: Decision tree, Naive bayes, SVM, berita

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Teknologi Informasi saat ini telah berkembang pesat dari waktu ke waktu seiring berjalannya zaman. Di era globalisasi ini manusia memang tidak terlepas dalam membutuhkan segala informasi dan komunikasi yang berasal dari berbagai media untuk dapat mengembangkan kemampuan di bidang teknologi yang saat ini berkembang pesat. Tak khayal teknologi informasi juga saat ini telah memberikan banyak sekali manfaat bagi banyak orang. Di Indonesia sendiri media *online* menjadi salah satu media yang paling banyak digunakan masyarakat untuk mendapatkan informasi berita (Asiyah & Fithriasari, 2018). Media *online* telah menghadirkan kemajuan yang luar biasa dalam hal komunikasi dan pertukaran informasi. Platform-platform berita *online* memungkinkan kita untuk mengakses berita dalam hitungan detik dari berbagai sumber di seluruh dunia. Dengan hanya menggunakan *smartphone* dan koneksi internet, kita dapat mengakses berita dari mana saja dan kapan saja.

Berita merupakan laporan yang berisi informasi mengenai suatu opini, peristiwa, situasi, kondisi, interpretasi yang penting, menarik, masih baru, dan harus secepatnya disampaikan kepada khalayak ramai (Fauziah, Maududie, & Nuritha, 2018). Berita dapat disampaikan melalui media berkala seperti surat kabar, radio, televisi, atau media *online* internet. Di era perkembangan teknologi ini, berita dapat diakses dengan mudah pada laman portal berita *online* melalui media internet yang disajikan dalam beberapa *website*, salah satunya yaitu Detikcom. Pada umumnya, berita yang disampaikan dalam *website* terdiri dari beberapa kategori diantaranya, travel, edukasi, otomotif, olahraga, dan kategori lainnya.

Pengklasifikasian berita dilakukan oleh pengelola dari portal berita *online* yang dibantu dengan editor konten berita. Namun, untuk saat ini masih banyak editor yang mengklasifikasi berita ke dalam kategori-kategori secara manual. Prosesnya, sebelum berita diunggah editor atau pengelola harus mengetahui isi seluruh beritanya untuk kemudian diklasifikasikan ke kategori yang paling tepat. Jika jumlah artikel semakin banyak, maka pengelola berita akan kesulitan karena harus membaca isi berita satu per satu secara keseluruhan dengan teliti hingga dapat mengklasifikasikan berita ke kategori yang paling tepat. Hal tersebut akan semakin sulit jika terdapat banyak dokumen dan kategori yang sangat beragam. Oleh karena itu, perlu adanya sistem yang secara otomatis dapat mengelompokkan berita sesuai dengan kategori berita menggunakan *text mining*.

Text mining merupakan salah satu cabang ilmu *data mining* yang menganalisis data berupa dokumen teks. *Text mining* adalah satu langkah dari analisis teks yang dilakukan secara otomatis oleh komputer untuk menggali informasi yang berkualitas dari suatu rangkaian teks yang terangkum dalam sebuah dokumen. Ide awal pembuatan text mining adalah untuk menemukan pola-pola informasi yang dapat digali dari suatu teks yang tidak terstruktur (Hamzah, 2020). Sebelum suatu data teks dianalisis menggunakan metode *text mining* perlu dilakukan *preprocessing*, diantaranya adalah *tokenizing*, *filtering* (*stop word removal*), *case folding*, dan *stemming*. Setelah dilakukan

preprocessing, langkah selanjutnya adalah mengklasifikasikan berita sesuai kategorinya. Klasifikasi data teks bisa dilakukan dengan beberapa metode, diantaranya K-Nearest neighbors (KNN), Support Vector Machine (SVM), dan Decision Tree.

Penelitian sebelumnya yang berkaitan adalah oleh Sudianto (2022) tentang klasifikasi topik berita menggunakan metode SVM dan MLP , dari analisis tersebut dihasilkan akurasi metode SVM sebesar 74%. Selain itu, Riri Nada (2018) meneliti tentang perbandingan kinerja metode Naïve Bayes dan KNN untuk klasifikasi artikel bahasa indonesia menghasilkan ketepatan akurasi sebesar 70% dari metode Naïve Bayes. Terdapat pula penelitian oleh Lusiana Efrizoni (2022) tentang komparasi ekstraksi dalam klasifikasi teks multilabel menggunakan algoritma *machine learning* yang menghasilkan akurasi kurang dari 50% dengan metode Decision Tree. Oleh karena itu, pada penelitian ini dilakukan pengujian untuk mengembangkan dan membandingkan ketiga performa model klasifikasi, yaitu model Decision Tree, SVM, dan Naïve Bayes terhadap data teks berita.

1.2. Permasalahan

Belum tersedia penelitian yang membandingkan performa ketiga algoritma klasifikasi yang meliputi Decision Tree, Support Vector Machine (SVM), dan Naïve Bayes secara langsung dengan menggunakan *dataset* yang sama.

1.3. Tujuan

Membandingkan kinerja algoritma klasifikasi Decision Tree, Support Vector Machine (SVM), dan Naïve Bayes dalam *kategorisasi* berita.

1.4. Manfaat

Mengetahui dengan baik mengenai kinerja algoritma decision tree, Support Vector Machine, dan naive bayes dalam kategorisasi berita detikcom.

BAB II
TINJAUAN PUSTAKA

2.1. Teori Penunjang

2.1.1. Detikcom

Detikcom merupakan media digital *online* terbesar dan terpopuler di Indonesia yang didirikan oleh Budiono Darsono pada tanggal 9 Juli 1998. Detikcom ini didirikan untuk menghadirkan berita terkini dan terupdate di Indonesia dengan menyajikan semua informasi berita tersebut dengan konsep breaking news. Detikcom menyajikan berita dalam bentuk teks, gambar, dan video, serta menyediakan fitur-fitur lain seperti forum diskusi, jadwal TV, dan informasi cuaca. Selain berita, Detik.com juga menyediakan konten-konten lain seperti artikel opini, tips, dan informasi lifestyle. Mereka memiliki banyak pembaca setia dan menjadi sumber berita utama bagi banyak orang di Indonesia yang ingin mendapatkan informasi terbaru (Lestari, 2022).

2.1.2. *Text Preprocessing*

Text processing merupakan proses pembersihan teks dengan cara menghilangkan atau membuang kata-kata yang tidak dibutuhkan dan dilakukan restrukturisasi teks dengan menghilangkan karakter selain huruf dan mengubah semua huruf pada teks menjadi huruf kecil (*case folding*), memisahkan teks menjadi setiap kata (*tokenizing*), menghilangkan imbuhan pada setiap kata (*stemming*), dan melakukan proses penghilangan kata yang tidak relevan pada dokumen (*stopword*) (Nuzul Hikmah, 2018)

- *Tokenizing*

  Tokenizing adalah proses pemotongan string berdasarkan setiap kata yang menyusunnya.
- *Case Folding*

  Case Folding adalah proses yang dilakukan untuk menyeragamkan karakter string pada data dan dilakukan proses pengubahan seluruh huruf (A-Z) menjadi huruf kecil dan karakter-karakter lain selain huruf ‘a’ sampai ‘z’ (tanda baca dan angka) akan dihilangkan dari data dan dianggap sebagai delimiter.
- *Stopword Removal*

  Stopword Removal adalah tahap menghilangkan kata-kata yang tidak memberikan informasi penting.
- *Stemming*

  Stemming adalah proses untuk memperkecil jumlah indeks yang berbeda dari suatu dokumen dan juga untuk melakukan pengelompokan kata-kata lain yang memiliki kata dasar dan arti yang serupa namun memiliki bentuk yang berbeda karena mendapat imbuhan yang berbeda.

2.1.3. Term Frequency-Inverse Document Frequency (TF- IDF)

TF-IDF merupakan suatu cara untuk memberikan bobot hubungan suatu kata (*term*) terhadap dokumen. Metode ini menggabungkan dua konsep untuk perhitungan bobot yaitu, frekuensi kemunculan sebuah kata di dalam sebuah dokumen tertentu dan

inverse document frequency yang mengandung kata tersebut (Karter D. Putung, Arie Lumenta, Agustinus Jacobus, 2016). Metode pembobotan TF-IDF disini dapat digunakan untuk mengubah data yang berbentuk teks menjadi numerik dengan menentukan hubungan masing-masing kata (*term*) pada suatu dokumen dengan cara memberikan bobot pada masing-masing term tersebut (Nuzul Hikmah, 2018).

TF-IDF menggabungkan dua konsep yaitu frekuensi kemunculan sebuah term pada suatu dokumen dan inverse frekuensi dokumen yang mengandung kata tersebut. Term yang lebih sering muncul pada dokumen menjadi lebih penting karena dapat mengindikasikan topik dari dokumen. Frekuensi term *i* dalam dokumen *j* didefinisikan sebagai berikut:

$$tf_{ij} = f_{ij}$$

Dimana f_{ij} adalah jumlah kemunculan term *i* pada dokumen *j*.

Inverse document frequency digunakan untuk menunjukkan *discriminative power* dari term *i*. Secara umum term yang muncul di berbagai dokumen kurang mengindikasikan untuk topik tertentu. Rumus dari *inverse document frequency* didefinisikan sebagai berikut:

$$idf_i = \log\left(\frac{N}{df_i}\right)$$

Dimana df_i adalah frekuensi dokumen dari term *i* dan dapat diartikan juga sebagai jumlah dokumen yang mengandung term *i*. *Log* digunakan untuk meredam efek relatif terhadap tf_{ij}.

Weight (bobot) W_{ij} dihitung menggunakan pengukuran TF-IDF yang didefinisikan sebagai berikut:

$$W_{ij} = tf_{ij} \times idf_i$$

Bobot paling tinggi diberikan kepada term yang sering kali muncul pada dokumen j tetapi jarang muncul dalam dokumen lain.

2.1.4. Klasifikasi

Klasifikasi merupakan metode *supervised learning* yang melakukan proses untuk menemukan model yang dapat membedakan kelas data dengan tujuan agar dapat digunakan untuk memprediksi kelas dari data yang belum diketahui label kelasnya (Kusrini et al, 2009). model tersebut dapat berupa aturan jika-maka, pohon keputusan, atau rumus matematis. Klasifikasi dalam *machine learning* memiliki 2 tahapan proses, yaitu (Cady, 2017):

- Pelatihan model
  Proses sekelompok titik data dan label yang bersesuaian yang mencoba untuk mempelajari pola bagaimana titik-titik data tersebut dipetakan ke label yang sesuai.
- Prediksi
  Setelah dilatih, pengklasifikasi (*classifier*) bertindak sebagai fungsi yang mengambil titik data tambahan dan melakukan prediksi kelas atau label. Prediksi tersebut dapat berupa label tertentu atau nilai kontinu yang dapat dilihat sebagai skor kepercayaan untuk label tertentu.

Klasifikasi yang menghasilkan dua label kelas disebut sebagai klasifikasi biner, sedangkan klasifikasi yang menghasilkan 3 atau lebih label kelas disebut dengan klasifikasi banyak kelas (*multiple class classification*).

2.1.5. Decision Tree

Decision Tree merupakan model klasifikasi yang memiliki bentuk struktur pohon dimana setiap *node* atau simpul dari pohon diasumsikan dengan atribut dari data, dalam hal ini adalah variabel data. Struktur pohon tersebut dapat digunakan untuk membuat keputusan atau prediksi berdasarkan aturan dan kondisi yang terdapat pada setiap *node* (Latifah et al., 2019). Pada Decision Tree bagian bawah dari setiap *node* yang ada dalam pohon atau leaf berisi nilai yang merupakan karakteristik dari label kelas yang diwakili oleh data yang jatuh ke daun tersebut (Aguilar-Chinea et al., 2019).

2.1.6. Support Vector Machine (SVM)

Support Vector Machine merupakan sistem pembelajaran dengan menggunakan ruang menggunakan ruang hipotesis yang berupa fungsi-fungsi linear di dalam sebuah fituryang memiliki dimensi tinggi dan dilatih menggunakan algoritma pembelajaran yang berdasarkan teori optimasi. Metode ini menggunakan hipotesis berupa fungsi –fungsi linier dalam sebuah ruang fitur yang berdimensi tinggi, dengan mengimplementasikan learning bias yang berasal dari teori pembelajaran statistik.Tingkat akurasi pada model yang akan dihasilkan oleh proses peralihan dengan SVM sangat bergantung terhadap fungsi kernel dan parameter yang digunakan. Data pada suatu dataset diberikan variabel xi, sedangkan untuk kelas pada dataset diberikan variabel yi. Metode SVM membagi dataset menjadi 2 kelas. Kelas pertama yang dipisah oleh hyperplane bernilai 1, sedangkan kelas lainnya bernilai -1 (Monika & Furqon, 2018).

$$X_i W + b \geq 1 \quad untuk\ Y_i = \quad 1$$
$$X_i W + b \leq 1 \quad untuk\ Y_i = -1$$

Keterangan:
X_i : data ke-*i*
W : nilai bobot *support vector machine* yang tegak lurus dengan *hyperplane*
b : nilai bias
Y_i : kelas data ke-*i*

2.1.7. Naïve Bayes

Naïve Bayes merupakan pengklasifikasian dengan metode probabilitas dan statistik. Metode klasifikasi ini didasarkan pada teorema Bayes yang memprediksi peluang di masa depan berdasarkan kejadian yang telah terjadi (Fitria, Muslim, & Azis, 2018). Klasifikasi Naïve Bayes diasumsikan bahwa keberadaan suatu fitur dalam suatu kelas tidak bergantung pada keberadaan fitur lainnya pada kelas tersebut. Bentuk umum dari teorema Bayes adalah sebagai berikut (Bustomi, 2014):

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

Keterangan:
X : Data dengan label kelas yang belum diketahui
H : Hipotesis data *X* merupakan suatu label kelas spesifik
P(*H*|*X*) : Probabilitas hipotesis *H* berdasarkan kondisi *X* (*posterior probability*)
P(*H*) : Probabilitas hipotesis *H* (*prior probability*)
P(*X*|*H*) : Probabilitas *X* berdasarkan kondisi pada hipotesis *X*
P(*X*) : Probabilitas *X*

2.2. Penelitian Terkait

2.2.1. Klasifikasi Berita *Online* dengan menggunakan Pembobotan TF-IDF dan Cosine Similarity (Bening Herwijayanti, 2018)

Penelitian yang dilakukan Bening Herwijayanti (2018) yang berjudul "Klasifikasi Berita Online dengan menggunakan Pembobotan TF-IDF dan Cosine Similarity" merupakan penelitian mengenai klasifikasi berita online menggunakan algoritma single pass clustering, dimana data yang akan digunakan berasal dari website berita online yaitu kompas.com. menggunakan pembobotan Term Frequency Inverse Document Frequency (Tf-idf) dan Cosine Similarity. Berdasarkan penelitian tersebut dapat disimpulkan bahwa hasil rata-rata pengujian pada 4 percobaan adalah 91.25% dan akurasi tertinggi terdapat pada pengujian ke 3 dengan akurasi 100%. Hal tersebut dapat terjadi karena hasil pada kategori data hasil sistem sesuai dengan data asli. (Herwijayanti et al., 2018)

2.2.2. Klasifikasi Berita Online Menggunakan Metode Support Vector Machine dan K-Nearest Neighbor (Siti Nur Aisyah, 2016)

Penelitian yang dilakukan Siti Nur Aisyah (2016) yang berjudul "Klasifikasi Berita Online Menggunakan Metode Support Vector Machine dan K- Nearest Neighbor" merupakan penelitian mengenai mengelompokkan berita sesuai dengan kategori berita dengan menggunakan text mining. Dalam penelitian ini, metode yang digunakan dalam klasifikasi adalah SVM dan KNN. Berdasarkan penelitian tersebut didapatkan hasil bahwa performa terbaik menggunakan SVM dengan nilai akurasi 93.2%. sedangkan menggunakan K-nearest neighbor dengan menggunakan 2-NN pada data testing dan word vector sebesar 3784 mendapatkan nilai akurasi hanya 60%. Maka metode terbaik untuk melakukan klasifikasi berita adalah dengan SVM. (Asiyah & Fithriasari, 2016)

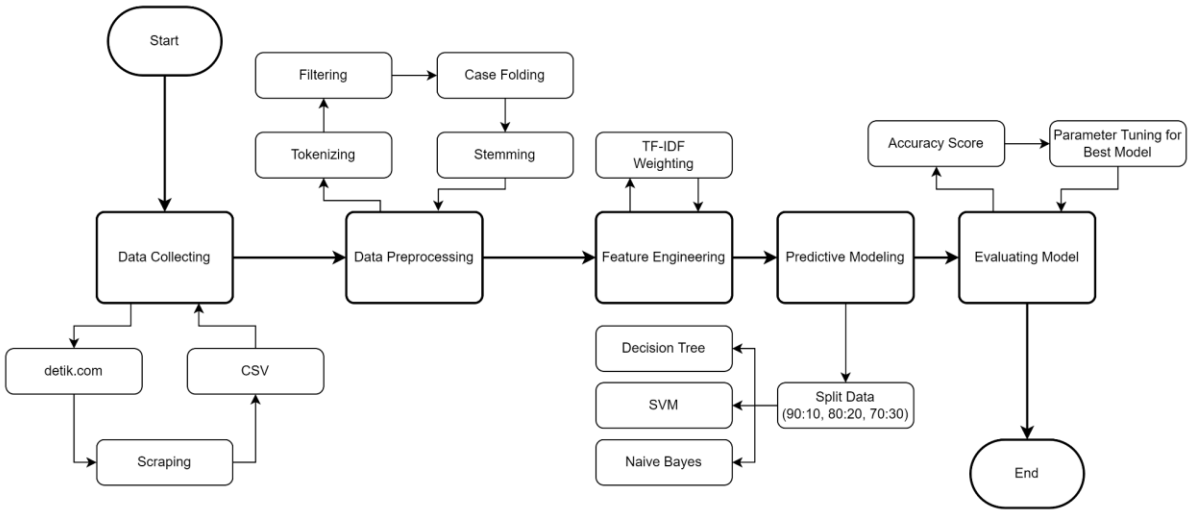
2.2.3. Klasifikasi Berita Indonesia Menggunakan Metode Naive Bayesian Classification dan Support Vector Machine dengan Confix Stripping Stemmer (Dio Ariadi, 2015)

Penelitian yang dilakukan Dio Ariadi (2015) yang berjudul "Klasifikasi Berita Indonesia Menggunakan Metode Naïve Bayesian Classification dan Support Vector Machine dengan Confix Stripping Stemmer" merupakan penelitian mengenai klasifikasi teks. klasifikasi yang digunakan adalah Naive Bayes Classifier (NBC) yang secara umum sering digunakan dalam data teks dan Support Vector Machine (SVM). Sumber data yang digunakan dalam penelitian tersebut adalah artikel berita pada koran online kompas.com.Berdasarkan penelitian didapatkan performa terbaik menggunakan metode

Support vector machine dengan nilai akurasi sebesar 88,1% dan menggunakan metode Naïve bayes classifier mendapatkan nilai akurasi sebesar 82,2%.

BAB III
METODOLOGI PENELITIAN

3.1. Desain Sistem Secara Umum



Gambar 1. Desain Sistem yang Digunakan

3.2. Prosedur Kerja

3.2.1. Instalasi dan Pengimporan *Module* atau *Library*

Tahap pertama yang dilakukan adalah menginstal dan mengimpor *module*/*library* yang diperlukan untuk melakukan semua proses dalam penelitian ini. *Module*/*library* yang digunakan, antara lain:

- pandas
- RegEx
- requests
- BeautifulSoup
- time
- Natural Language Toolkit (NLTK)
- PySastrawi

3.2.2. *Data Collecting*

Proses pengumpulan data melibatkan pengambilan informasi atau fakta yang berkaitan dengan tujuan spesifik dalam penelitian atau analisis. Tahap pengumpulan data sangat penting dalam proses penelitian karena data yang tidak akurat dan tidak mewakili dengan baik dapat menghasilkan temuan yang tidak valid. Data yang digunakan pada penelitian ini diperoleh dari portal berita *online* Detikcom.

Proses pengumpulan data dilakukan dengan menerapkan teknik *web scraping* untuk mengekstrak konten berita yang terdapat pada situs Detikcom. Situs tersebut dapat menyediakan hingga 10.000 berita per kategori berita dalam satu kali pencarian yang dilakukan. Proses pengumpulan data dilakukan dengan mengekstrak 1.000 konten berita dari setiap kategori berita. Kategori berita yang dipilih meliputi detikfinance, detiksport, detikhealth, detikfood, detikoto, detiktravel, dan detikedu. Setelah itu, konten

berita yang berhasil diekstrak disimpan dalam berkas yang berekstensi CSV.

3.2.3. *Data Preprocessing*

Data Preprocessing adalah tahap yang diperlukan untuk mempersiapkan data awal sehingga dapat digunakan secara efektif dalam analisis. Langkah-langkah dalam proses ini disesuaikan dengan kebutuhan tertentu dan bertujuan untuk mengubah data mentah menjadi data yang siap digunakan. Pada penelitian ini, data yang digunakan berupa data teks sehingga diterapkan *text preprocessing* untuk mempersiapkan data. *Text Preprocessing* yang dilakukan meliputi *tokenizing*, *filtering* (*stop word removal*), *case folding*, dan *stemming*.

3.2.3.1. *Tokenizing*

Tokenizing merupakan tahapan yang melibatkan pembagian teks menjadi bagian-bagian kecil yang disebut token. Bagian kecil tersebut berupa kata yang menyusun suatu teks atau kalimat. Setiap token mewakili unit terkecil yang memiliki makna yang relevan dalam konteks yang sedang dianalisis. Proses *tokenizing* membantu dalam mengurangi dimensi data dan mempermudah proses selanjutnya seperti vektorisasi atau pembentukan model statistik.

3.2.3.2. *Filtering*

Filtering atau *stop word removal* adalah langkah penting dalam *text preprocessing* yang bertujuan untuk mengeliminasi kata-kata umum dan tidak signifikan dalam analisis teks. Kata-kata semacam itu disebut *stop word*, contohnya adalah "adalah", "yaitu", "dan", "atau", "yang", dan lain-lain.

Stop word removal dilakukan agar analisis teks lebih berkualitas dengan berfokus pada kata-kata yang memiliki makna penting dan relevan. *Stop word* umumnya tidak memberikan informasi khusus dan sering muncul secara berulang dalam teks yang berbeda. Dengan menghapus *stop word*, kita dapat lebih memusatkan perhatian pada kata-kata kunci yang memiliki dampak yang lebih signifikan dalam pemrosesan dan analisis data.

Tahap *Filtering* dilakukan dengan membandingkan setiap token kata dalam data dengan daftar kata yang termasuk ke dalam *stop word*. Apabila terdapat token kata yang sama, maka token kata tersebut akan dieliminasi dari data. Hasil dari tahap ini berupa data teks yang telah di*filter* sehingga dapat meningkatkan kualitas analisis teks secara keseluruhan.

3.2.3.3. *Case Folding*

Case folding merupakan tahapan yang melibatkan konversi semua karakter dalam teks menjadi huruf kecil dan mengeliminasi karakter selain alfabet. Tujuan utamanya adalah untuk memastikan konsistensi dalam analisis teks tanpa memperhatikan perbedaan antara huruf besar dan huruf kecil. Dengan menggunakan huruf kecil untuk semua karakter, perbedaan huruf besar dan huruf kecil tidak lagi menjadi faktor yang

mempengaruhi hasil analisis. Ini memungkinkan proses analisis teks yang lebih akurat dan konsisten.

3.2.3.4. *Stemming*

Stemming adalah tahapan dalam *text preprocessing* yang bertujuan untuk mengubah kata-kata menjadi bentuk dasar atau akar kata. Hal ini dilakukan untuk mengurangi variasi kata yang memiliki akar kata yang sama, sehingga mempermudah analisis teks. Tujuan dari *stemming* adalah untuk menyederhanakan kata-kata dalam dokumen sehingga kata-kata yang memiliki akar yang sama dapat diidentifikasi sebagai entitas yang sama. Misalnya, kata "berlari", "berlari-lari", dan "berlari-lah" akan diubah menjadi kata dasar "lari" setelah melalui proses stemming.

Penerapan *stemming* melibatkan penggunaan algoritma atau aturan linguistik yang telah ditentukan sebelumnya. Beberapa algoritma *stemming* yang umum digunakan adalah algoritma Porter, algoritma Snowball, dan algoritma Lancaster. Setiap algoritma memiliki pendekatan yang berbeda dalam mengidentifikasi akar kata. Meskipun *stemming* tidak selalu menghasilkan akar kata yang sempurna dalam setiap kasus, ini masih merupakan langkah yang berguna dalam mempersiapkan data teks untuk analisis lebih lanjut, seperti pemrosesan teks, klasifikasi, atau pembentukan model bahasa.

3.2.4. *Feature Engineering*

Feature engineering adalah proses transformasi data mentah menjadi fitur yang dapat digunakan untuk membangun model prediktif. Dalam konteks data teks, *feature engineering* sering melibatkan representasi teks dalam bentuk numerik yang dapat dimasukkan ke dalam algoritma pembelajaran mesin.

Dalam penelitian ini, digunakan metode TF-IDF (*Term Frequency-Inverse Document Frequency*) yang merupakan salah satu teknik yang dapat digunakan untuk *feature engineering* dengan data teks. TF-IDF bekerja dengan memberikan bobot pada kata-kata dalam dokumen berdasarkan pentingnya. Pentingnya suatu kata ditentukan oleh dua faktor: frekuensinya dalam dokumen (*Term Frequency*) dan kelangkaannya di seluruh dokumen dalam korpus (*Inverse Document Frequency*).

Term Frequency (TF) mengukur seberapa sering suatu kata muncul dalam dokumen. Ini dihitung dengan membagi jumlah kali kata muncul dalam dokumen dengan jumlah total kata dalam dokumen. Sedangkan *Inverse Document Frequency* (IDF) mengukur seberapa langka suatu kata di seluruh dokumen dalam korpus. Ini dihitung dengan mengambil logaritma dari rasio jumlah total dokumen dengan jumlah dokumen yang mengandung kata tersebut. Dengan demikian, bobot TF-IDF untuk suatu kata dihitung dengan mengalikan nilai TF dan IDF-nya. Bobot ini merepresentasikan pentingnya kata tersebut dalam konteks dokumen dan seluruh korpus.

3.2.5. Pembuatan Model Klasifikasi

Pembuatan model klasifikasi adalah proses pembuatan model yang dapat digunakan untuk mengklasifikasikan data. Model ini dibuat dengan menggunakan algoritma klasifikasi tertentu. Tujuan dari pembuatan model klasifikasi ini adalah untuk

membuat model yang dapat mengklasifikasikan data dengan akurat. Model ini dapat digunakan untuk memprediksi label kelas dari data baru yang belum diketahui label kelasnya. Proses pembuatan model klasifikasi mulai dari pemilihan algoritma klasifikasi yang sesuai dengan data yang akan diklasifikasikan, yang mana pada penelitian ini digunakan algoritma Decision Tree, Support Vector Machine (SVM), dan Naïve Bayes. Setelah memilih algoritma klasifikasi yang sesuai, langkah selanjutnya adalah melatih model dengan menggunakan data latih. Data latih adalah data yang sudah diketahui label kelasnya dan digunakan untuk melatih model agar dapat mengenali pola dalam data. Proses pelatihan ini dilakukan dengan memberikan data latih kepada model dan menyesuaikan parameter model agar dapat mengklasifikasikan data latih dengan benar. Setelah melatih model, langkah selanjutnya adalah menguji kinerja model dengan menggunakan data uji. Data uji adalah data yang tidak digunakan dalam proses pelatihan dan digunakan untuk mengevaluasi kinerja model. Proses pengujian ini dilakukan dengan memberikan data uji pada model dan membandingkan hasil prediksi model dengan label kelas yang sebenarnya dari data uji.

3.2.6. Evaluasi Performa Model Klasifikasi

Evaluasi performa model adalah proses untuk mengukur seberapa baik model dapat memprediksi atau mengklasifikasikan data yang belum diketahui sebelumnya. Evaluasi performa model penting untuk mengetahui kekuatan dan kelemahan model, serta untuk membandingkan dan memilih model yang paling sesuai dengan tujuan dan data yang digunakan. Pada penelitian ini digunakan metrik akurasi untuk menghitung persentase kasus yang diprediksi dengan benar oleh model. Metrik ini sesuai untuk masalah klasifikasi yang membedakan data menjadi beberapa kelas atau kategori. Untuk menginterpretasikan nilai akurasi, dapat dilakukan dengan membandingkannya dengan nilai acuan atau *baseline*. Nilai acuan dapat berupa nilai akurasi dari model sebelumnya, nilai akurasi dari model lain yang menggunakan data yang sama, atau nilai akurasi yang diharapkan dari tujuan penelitian. Nilai akurasi yang tinggi menunjukkan bahwa model klasifikasi dapat memprediksi label dengan tepat, sedangkan nilai akurasi yang rendah menunjukkan bahwa model klasifikasi sering salah dalam memprediksi label.

BAB IV: HASIL DAN PEMBAHASAN

4.1. *Data Collecting* (Pengumpulan Data)

Pada penelitian ini, proses pengumpulan data dilakukan dengan menerapkan teknik *web scraping* untuk mengekstrak konten berita yang terdapat pada situs Detikcom. Data yang dikumpulkan berupa kumpulan konten berita yang terdiri dari judul, kategori berita, tanggal dan waktu publikasi, serta tautan berita. Berita yang berhasil diekstrak merupakan berita yang telah dipublikasikan pada tanggal 15 Maret 2023 hingga 31 Mei 2023. Proses pengumpulan data dilakukan dengan mengekstrak sebanyak 1.000 konten berita per kategori berita sehingga didapatkan total 7.000 konten berita.

Gambar 2 merupakan potongan kode program yang digunakan untuk melakukan *scraping* konten berita. Digunakan pandas untuk menyimpan hasil ekstraksi konten berita agar terstruktur seperti tabel sehingga dapat disimpan dalam berkas dengan ekstensi Comma Separated Value (CSV). Sementara itu, modul scrape_detikcom merupakan modul yang berisi *predefined function* untuk melakukan *scraping* konten berita pada situs Detikcom. Kode program dari modul scrape_detikcom ditampilkan pada Lampiran 1.

Pada modul yang telah dibuat hanya dirancang untuk melakukan *scraping* konten berita pada satu kategori berita saja sehingga diperlukan tambahan kode program untuk dapat melakukan *scraping* konten berita pada tujuh kategori. Hal tersebut dilakukan dengan membuat *dictionary* cat_id yang menyimpan site_id dari setiap kategori berita yang akan diekstrak. Setelah itu, proses *scraping* dilakukan dengan menerapkan perulangan *for* terhadap elemen dari *dictionary* cat_id. Hasil *scraping* dari setiap kategori berita kemudian disimpan dalam *DataFrame* df_cat dan ditambahkan ke dalam *DataFrame* df_all. Setelah *scraping* berhasil dilakukan, *DataFrame* df_all kemudian dikonversi dan disimpan menjadi berkas dengan ekstensi CSV.

```
# impor modul
import pandas as pd
from func import scrape_detikcom as dtk

# pencarian topik berita
query = '+'

# inisialisasi site_id kategori berita
cat_id = {'finance': 29, 'sport': 21, 'health': 55, 'food':35, 'oto': 44,
          'travel': 66, 'edu': 117}

df_all = pd.DataFrame(columns=['title', 'category', 'publish_date',
                               'article_url', 'content'])
for cat in cat_id:
    print(f"Kategori: {cat}")
    df_cat = dtk.detikcom_articles(query,
                                   cat_id[cat],
                                   1000,
                                   '15/03/2023',
                                   '31/05/2023',
                                   False)
    df_all = pd.concat([df_all, df_cat], ignore_index=True)
```

```
    print()
```

Gambar 2. Potongan Kode Program Proses *Scraping* Konten Berita

Berdasarkan kode program yang terdapat pada Gambar 3, dilakukan proses ekstraksi sebanyak 1.000 konten berita per kategori. Proses ekstraksi ini berlaku untuk konten berita yang dipublikasikan dalam rentang waktu mulai dari tanggal 15 Maret 2023 hingga 31 Mei 2023. Hasil dari proses tersebut menghasilkan total 7.000 konten berita yang terdiri dari berbagai kategori berita yang ada.

	title	category	publish_date	article_url	content
0	Pasar Sawit Dihambat, RI & Malaysia Langsung S...	detikFinance	31 Mei 2023 23:55 WIB	https://finance.detik.com/industri/d-6749907/p...	NaN
1	Nggak Nyangka! Negara Tetangga Ini Pesaing RI ...	detikFinance	31 Mei 2023 23:06 WIB	https://finance.detik.com/industri/d-6749887/n...	NaN
2	Cuan Ratusan Juta Rupiah dari Bisnis Makanan Beku	detikFinance	31 Mei 2023 23:00 WIB	https://finance.detik.com/solusiukm/d-6749882/...	NaN
3	Genjot Kendaraan Listrik, RI Jaring Produsen M...	detikFinance	31 Mei 2023 22:21 WIB	https://finance.detik.com/industri/d-6749860/g...	NaN
4	Insentif buat Bus Listrik Ada, Cek di Sini Pen...	detikFinance	31 Mei 2023 21:17 WIB	https://finance.detik.com/industri/d-6749780/i...	NaN
...	...	...	...	...	...
6995	Siap-siap UTBK 2023, Intip Tata Tertib UTBK & ...	detikEdu	01 Mei 2023 09:00 WIB	https://www.detik.com/edu/seleksi-masuk-pt/d-6...	NaN
6996	Beasiswa ke Jepang MEXT Scholarship 2024 Jenja...	detikEdu	01 Mei 2023 08:00 WIB	https://www.detik.com/edu/beasiswa/d-6696920/b...	NaN
6997	Kurang Waktu Bermain Mandiri Tingkatkan Ganggu...	detikEdu	01 Mei 2023 07:00 WIB	https://www.detik.com/edu/detikpedia/d-6695590...	NaN
6998	IPB Masih Buka Program S1 Beasiswa Utusan Daer...	detikEdu	01 Mei 2023 06:00 WIB	https://www.detik.com/edu/beasiswa/d-6696919/i...	NaN
6999	Mumi Mesir Kuno Ini Ungkap Anemia dan Thalasse...	detikEdu	30 Apr 2023 20:00 WIB	https://www.detik.com/edu/detikpedia/d-6697392...	NaN

7000 rows × 5 columns

Gambar 3. Potongan DataFrame Konten Berita

Gambar 3 merupakan potongan *DataFrame* konten berita yang berhasil diekstrak. Perlu diperhatikan bahwa dalam konteks penelitian ini, fokus klasifikasi hanya dilakukan berdasarkan judul berita saja. Oleh karena itu, isi berita tidak diekstrak dan tidak digunakan dalam proses klasifikasi. Tujuan dari pembatasan ini adalah untuk menyederhanakan proses analisis dan fokus pada informasi yang terkandung dalam judul berita, yang dianggap dapat memberikan indikasi yang cukup untuk melakukan klasifikasi dengan akurasi yang memadai.

Setelah data mentah berhasil diperoleh dan dipersiapkan, langkah berikutnya adalah memuat data ke dalam Jupyter Notebook menggunakan kode program yang ditampilkan pada Gambar 4.

```
# impor modul
import pandas as pd

df = pd.read_csv('data/2023_0531-0315_1000_False_detikcom.csv')
df
```

Gambar 4. Potongan Kode Program Pemuatan Data

4.2. *Data Preprocessing*

Proses *data preprocessing* dilakukan dengan menerapkan konsep *text preprocessing* yang terdiri dari serangkaian tahapan yang meliputi *tokenizing*, *filtering*, *case folding*, dan *stemming*. Namun, sebelum melakukan serangkaian tahapan tersebut, dilakukan pemeriksaan dan transformasi nilai data pada *dataset*. Pemeriksaan yang dilakukan meliputi pemeriksaan data duplikat, *missing value*, dan jumlah data. Kode program yang digunakan untuk melakukan pemeriksaan ditampilkan pada Gambar 5.

```
# Memeriksa data duplikat
df.duplicated().sum()

# memeriksa missing value
df.isnull().sum()

# memeriksa jumlah nilai unik pada setiap kategori
df['category'].value_counts()
```

Gambar 5. Potongan Kode Program Pemeriksaan Data

Berdasarkan pemeriksaan data yang telah dilakukan, ditemukan bahwa tidak terdapat nilai yang hilang (*missing value*) dalam *dataset* yang digunakan. Hal ini menunjukkan bahwa data yang diekstrak sudah lengkap dan tidak ada informasi yang hilang pada atribut-atribut yang diamati. Dengan demikian, nilai unik pada tiap kategori berjumlah 1000.

Sementara itu, transformasi nilai data yang dilakukan adalah menghapus *prefix* "detik" pada nilai kategori berita. Hal ini dilakukan untuk menyederhanakan nilai kategori berita agar menjadi lebih umum dan tidak terlalu panjang. Proses transformasi dilakukan dengan menggunakan *method* map() dari *library* pandas dengan parameter berupa *dictionary* yang berisikan nilai kategori dan tujuan transformasinya. Penerapan dari proses konversi ini dapat dilihat pada contoh kode program dan hasil transformasi yang ditampilkan pada Gambar 6 dan Tabel 1.

```
cat_dict = {'detikFinance': 'finance',
            'detikSport': 'sport',
            'detikHealth': 'health',
            'detikFood': 'food',
            'detikOto': 'automotive',
            'detikTravel': 'travel',
            'detikEdu': 'education'}
df['category'] = df['category'].map(cat_dict)
```

Gambar 6. Potongan Kode Program Konversi Nilai Kategori Berita

Tabel 1. Contoh Hasil Konversi Tipe Data Label

Kategori Berita	Hasil Transformasi
detikFinance	finance
detikSport	sport
detikHealth	health
detikFood	food
detikOto	automotive
detikTravel	travel
detikEdu	edu

4.2.1. *Tokenizing*

Pada tahap ini digunakan bantuan *library* NLTK (Natural Language Toolkit) dengan modul RegexpTokenizer untuk melakukan tokenisasi teks. Parameter (\w+) yang digunakan dalam RegexpTokenizer akan mencocokkan pola kata dalam suatu *string*. Setiap karakter dalam pola tersebut ekuivalen dengan A-Z, a-z, 0-9, dan simbol underscore (_). Dengan demikian, setiap kata yang ditokenisasi akan terdiri dari karakter *alphanumeric* dan underscore, sementara tanda baca atau karakter lainnya akan langsung dieliminasi.

Selain itu, pada tahap ini juga dilakukan konversi huruf kapital menjadi huruf kecil menggunakan *method* lower(). Hal ini dilakukan agar semua token kata memiliki format huruf kecil yang seragam sebelum dilanjutkan ke tahap *filtering* (*stop word removal*). Kode program yang digunakan pada tahap ini dapat dilihat pada Gambar 7.

```
from func import text_cleansing as tc

title_list = df['title'].values.tolist()
title_list = tc.tokenizing(title_list)
```

Gambar 7. Potongan Kode Program Tahap *Tokenizing*

Pada kode program yang ditampilkan pada Gambar 7, dilakukan pengimporan package func yang berisi modul text_cleansing. Di dalam modul tersebut terdapat *predefined function* yang digunakan untuk melakukan seluruh tahapan *text preprocessing* pada penelitian ini, termasuk tahap *tokenizing*. Seluruh *predefined function* yang terdapat dalam modul text_cleansing ditampilkan pada Lampiran 2. Sebagai contoh, hasil dari tahap *tokenizing* ditampilkan dalam Tabel 2.

Tabel 2. Contoh Hasil Tahap Tokenizing

Judul berita	**Hasil *Tokenizing***
Terungkap! Biang Kerok Harga Telur yang Makin Mahal	[terungkap, biang, kerok, harga, telur, yang, makin, mahal]
Menteri Kelautan Ungkap Alasan Ekspor Pasir Laut Dibuka Lagi	[menteri, kelautan, ungkap, alasan, ekspor, pasir, laut, dibuka, lagi]
The Body Shop Buka-bukaan Soal PHK 146 Karyawan	[the, body, shop, buka, bukaan, soal, phk, 146, karyawan]

4.2.2. *Filtering* (*stop word removal*)

Pada tahap ini, digunakan bantuan *library* PySastrawi dengan modul StopwordRemoverFactory untuk melakukan penghapusan kata-kata yang termasuk ke dalam *stop word*. Proses ini melibatkan ekstraksi kata-kata yang termasuk dalam *stop word* pada modul StopwordRemoverFactory dan pencocokan antara kata-kata dalam data dengan daftar kata *stop word* tersebut. Jika pada data terdapat token kata yang sama dengan kata dalam *stop word*, maka token kata tersebut akan dihapus dari data. Kode program yang digunakan pada tahap ini dapat dilihat pada Gambar 8. Sebagai contoh, hasil dari tahap *filtering* ditampilkan dalam Tabel 3.

```
title_list = tc.stopword_removal(title_list)
```

Gambar 8. Potongan Kode Program Tahap *Filtering*

Tabel 3. Contoh Hasil Tahap *Filtering*

Hasil *Tokenizing*	**Hasil *Filtering***
[terungkap, biang, kerok, harga, telur, yang, makin, mahal]	[terungkap, biang, kerok, harga, telur, mahal]
[menteri, kelautan, ungkap, alasan, ekspor, pasir, laut, dibuka, lagi]	[menteri, kelautan, alasan, ekspor, pasir, laut, dibuka]
[the, body, shop, buka, bukaan, soal, phk, 146, karyawan]	[the, body, shop, buka, bukaan, phk, 146, karyawan]

4.2.3. *Case Folding*

Pada tahap ini digunakan *method* isalpha() untuk menghilangkan karakter selain huruf (a-z) dari token kata. *Method* isalpha() akan mengembalikan nilai *True* jika *string* yang dikenakan *method* tersebut hanya terdiri dari karakter huruf "A-Z" atau "a-z". Dengan mengaplikasikan *method* isalpha(), token kata yang mengandung karakter selain huruf akan dieliminasi dari data. Kode program yang digunakan pada tahap ini dapat dilihat pada Gambar 9. Sebagai contoh, hasil dari tahap *case folding* ditampilkan dalam Tabel 4.

```
title_list = tc.case_folding(title_list)
```

Gambar 9. Potongan Kode Program Tahap *Case Folding*

Tabel 4. Contoh Hasil Tahap Case Folding

Hasil *Filtering*	**Hasil *Case Folding***
[terungkap, biang, kerok, harga, telur, mahal]	[terungkap, biang, kerok, harga, telur, mahal]
[menteri, kelautan, alasan, ekspor, pasir, laut, dibuka]	[menteri, kelautan, alasan, ekspor, pasir, laut, dibuka]
[the, body, shop, buka, bukaan, phk, 146, karyawan]	[the, body, shop, buka, bukaan, phk, karyawan]

4.2.4. *Stemming*

Pada tahap ini digunakan bantuan *library* PySastrawi dengan modul StemmerFactory. Tahap *stemming* bertujuan untuk mengubah setiap kata dalam dokumen menjadi bentuk kata dasar. Dalam proses *stemming*, kata dasar yang dihasilkan tidak memperhatikan kaidah kebahasaan yang benar. Setiap kata dalam token kata akan diubah dengan menghapus bagian akhiran atau awalan yang mempengaruhi makna kata tersebut.

Namun, perlu diperhatikan bahwa proses *stemming* tidak memperhatikan kaidah kebahasaan yang benar. Hal ini berbeda dengan proses *lemmatization*, di mana kata akan diubah menjadi kata dasar sesuai dengan kaidah kebahasaan yang benar. Dalam proses *lemmatization*, kata yang dihasilkan akan menjadi bentuk kata dasar yang sesuai dengan

kaidah tata bahasa yang berlaku. Kode program yang digunakan pada tahap ini dapat dilihat pada Gambar 10. Sebagai contoh, hasil dari tahap *stemming* ditampilkan dalam Tabel 5.

```
title_list = tc.stemming(title_list)
title_list = tc.clean_doc(title_list)
```

Gambar 10. Potongan Kode Program Tahap *Stemming*

Tabel 5. Contoh Hasil Tahap *Stemming*

Hasil *Case Folding*	**Hasil *Stemming***
[terungkap, biang, kerok, harga, telur, mahal]	[ungkap, biang, kerok, harga, telur, mahal]
[menteri, kelautan, alasan, ekspor, pasir, laut, dibuka]	[menteri, laut, alas, ekspor, pasir, laut, buka]
[the, body, shop, buka, bukaan, phk, karyawan]	[the, body, shop, buka, buka, phk, karyawan]

4.3. Feature Engineering

Setelah melalui tahap *text preprocessing* dan didapatkan data teks bersih, langkah selanjutnya adalah melakukan pembobotan kata dengan menggunakan algoritma Term Frequency-Inverse Document Frequency (TF-IDF). Proses dimulai dengan menghitung jumlah frekuensi kemunculan kata (*term frequency*) dalam setiap dokumen (token kata judul berita). Setelah itu, dilakukan penghitungan nilai Inverse Document Frequency (IDF) dan didapatkan bobot setiap kata dengan mengalikan nilai TF dengan IDF.
Secara umum, proses ini terdiri dari tiga tahapan, antara lain:

1. Mendapatkan *term frequency* (TF) atau frekuensi kemunculan kata
   Pada tahap ini, dihitung kata yang muncul pada setiap dokumen (judul berita), sesuai dengan rumus berikut:

$$tf_{ij} = f_{ij}$$

   Contoh hasil dari tahap ini ditampilkan pada Gambar 8.

	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	...	D6990	D6991	D6992	D6993	D6994	D6995	D6996	D6997	D6998	D6999
aa	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
aaji	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
aal	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
aare	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
aba	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
zonk	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
zoo	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
zs	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
zulhas	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
zuppa	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

7505 rows × 7000 columns

Gambar 11. Potongan DataFrame Hasil Penghitungan TF

2. Menghitung Nilai Inverse Document Frequency (IDF)
   Nilai IDF didapatkan melalui perhitungan dengan menggunakan rumus berikut:

$$idf_i = \log\left(\frac{N}{df_i}\right)$$

Contoh hasil dari tahap ini ditampilkan pada Gambar 9.

	IDF
aa	3.845
aaji	3.845
aal	3.544
aare	3.845
aba	3.845
...	...
zonk	2.766
zoo	3.845
zs	3.845
zulhas	2.942
zuppa	3.368

7505 rows × 1 columns

Gambar 12. Potongan DataFrame Hasil Penghitungan IDF

3. Menghitung nilai bobot setiap kata
   Nilai bobot kata didapatkan melalui perhitungan dengan rumus:

$$W_{ij} = tf_{ij} \times idf_i$$

Contoh hasil dari tahap ini ditampilkan pada Gambar 10.

	aa	aaji	aal	aare	aba	abad	abah	abai	abal	abang	...	zohri	zombie	zona	zonasi	zone	zonk	zoo	zs	zulhas	zuppa
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6999	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

7000 rows × 7505 columns

Gambar 13. Potongan DataFrame Hasil Penghitungan Bobot Kata

Setelah proses pembobotan kata berhasil dilakukan, selanjutnya dilakukan penggabungan kolom judul dan kategori terhadap DataFrame bobot kata yang ditampilkan pada Gambar 11. Selain itu, seluruh kode program yang digunakan pada proses pembobotan kata ditampilkan pada Lampiran 3 dan Lampiran 4.

	title	category	aa	aaji	aal	aare	aba	abad	abah	abai	...	zohri	zombie	zona	zonasi	zone	zonk	zoo	zs	zulhas	zuppa
0	Pasar Sawit Dihambat, RI & Malaysia Langsung S...	finance	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	Nggak Nyangka! Negara Tetangga Ini Pesaing RI ...	finance	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	Cuan Ratusan Juta Rupiah dari Bisnis Makanan Beku	finance	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	Genjot Kendaraan Listrik, RI Jaring Produsen M...	finance	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	Insentif buat Bus Listrik Ada, Cek di Sini Pen...	finance	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6995	Siap-siap UTBK 2023, Intip Tata Tertib UTBK & ...	education	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6996	Beasiswa ke Jepang MEXT Scholarship 2024 Jenja...	education	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6997	Kurang Waktu Bermain Mandiri Tingkatkan Ganggu...	education	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6998	IPB Masih Buka Program S1 Beasiswa Utusan Daer...	education	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6999	Mumi Mesir Kuno Ini Ungkap Anemia dan Thalasse...	education	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

7000 rows × 7507 columns

Gambar 14. Potongan DataFrame Hasil *Feature Engineering*

4.4. Pembuatan Model Klasifikasi

Pada penelitian ini, dilakukan pengujian untuk membandingkan performa model klasifikasi yang terdiri dari algoritma Decision Tree, Support Vector Machine (SVM), dan Naïve Bayes. Terdapat tiga skema pengujian yang dilakukan dengan menerapkan proporsi pembagian data latih dan data uji sebagai berikut:

- 90% data latih dan 10% data uji,
- 80% data latih dan 20% data uji,
- 70% data latih dan 30% data uji.

Pembagian data latih dan data uji dilakukan dengan menggunakan bantuan *function* train_test_split yang terdapat pada modul sklearn.model_selection dari *library* scikit-learn. Kode program yang digunakan dalam proses pembagian data ditampilkan pada Gambar 12.

```
from sklearn.model_selection import train_test_split

X = df.drop(['category', 'title'], axis=1)
y = df['category']

# split data 90:10
X_train_1, X_test_1, y_train_1, y_test_1 = train_test_split(X, y,
test_size=0.1, random_state=42)

# split data 80:20
X_train_2, X_test_2, y_train_2, y_test_2 = train_test_split(X, y,
test_size=0.2, random_state=42)

# split data 70:30
X_train_3, X_test_3, y_train_3, y_test_3 = train_test_split(X, y,
test_size=0.3, random_state=42)
```

Gambar 15. Potongan Kode Program Pembagian Data

Setelah proses pembagian data latih dan uji selesai, langkah selanjutnya adalah membuat objek model klasifikasi. Pada pembuatan objek model klasifikasi ini, digunakan parameter *default* dari setiap model klasifikasi yang disediakan oleh *library* scikit-learn. Parameter *default* ini telah ditetapkan oleh *library* untuk memberikan hasil yang optimal dalam banyak kasus penggunaan. Setelah objek model klasifikasi berhasil dibuat, selanjutnya dilakukan pelatihan pada seluruh model yang telah dibuat sesuai dengan skema pengujian yang telah ditetapkan sebelumnya.

4.5. Evaluasi Performa Model Klasifikasi

Dilakukan evaluasi performa model klasifikasi pada data uji dengan menggunakan metrik *accuracy* (akurasi). Metrik tersebut menggambarkan seberapa akurat model

dapat mengklasifikasikan dengan benar yang diukur dengan rasio prediksi benar dengan keseluruhan data. Ketiga model yang telah dilatih sebelumnya, digunakan untuk melakukan prediksi pada data uji.

Selanjutnya, dilakukan penghitungan nilai akurasi pada hasil prediksi dengan menggunakan bantuan *function* classification_report yang terdapat dalam modul sklearn.metrics dari *library* scikit-learn. Penggunaan *function* tersebut akan memberikan hasil penghitungan dari berbagai metrik evaluasi, salah satunya adalah nilai akurasi. Seluruh proses pengujian yang terdiri dari pelatihan dan prediksi model dibuat dengan kode program yang ditulis menggunakan pendekatan *functional*. Kode program proses pengujian model klasifikasi ditampilkan pada Gambar 13.

```
# membuat predefined function untuk melakukan training dan testing
def train_test_model(model, xtrain, xtest, ytrain):
    # melakukan training model
    start = time.time()
    model.fit(xtrain, ytrain)
    end = time.time()

    train_time = end - start

    # melakukan testing model
    y_pred = model.predict(xtest)

    # mengembalikan hasil testing
    return y_pred, train_time

# membuat predefined function untuk melakukan evaluasi model
def evaluate_model(ytest, ypred):
    report = classification_report(ytest, ypred, output_dict=True)
    accuracy = report['accuracy']

    return accuracy

# membuat predefined function untuk melakukan training, testing, dan
  evaluasi model
def model_report(models, xtrain, xtest, ytrain, ytest):
    eval_list = []

    for model in models:
        # training dan testing model
        y_pred, train_time = train_test_model(model, xtrain, xtest,
                                              ytrain)
        # evaluasi model
        accuracy = evaluate_model(ytest, y_pred)
        model_name = model.__class__.__name__
        eval_dict = {'Model': model_name,
                     'Accuracy': round(accuracy, 3),
                     'Training Time (s)': round(train_time, 3)
                     }
        eval_list.append(eval_dict)
    eval_df = pd.DataFrame(eval_list)
    eval_df = eval_df.set_index('Model')
    eval_df.index.name = 'Metrik'
    eval_df = eval_df.T

    return eval_df

accuracy_comparison_1 = model_report([tree, svm, nb], X_train_1,
```

```
                                        X_test_1, y_train_1, y_test_1)
accuracy_comparison_2 = model_report([tree, svm, nb], X_train_2,
                                        X_test_2, y_train_2, y_test_2)
accuracy_comparison_3 = model_report([tree, svm, nb], X_train_3,
                                        X_test_3, y_train_3, y_test_3)
```

Gambar 16. Potongan Kode Program Pengujian Model Klasifikasi

Berdasarkan pengujian dilakukan, diperoleh perbandingan performa antara antara model Decision Tree, SVM, dan Naïve bayes dengan menggunakan proporsi pembagian data uji dan data latih yang bervariasi disajikan pada Tabel 6. Dalam tabel tersebut diberikan gambaran mengenai akurasi dan durasi pelatihan yang dihasilkan oleh masing-masing model klasifikasi pada setiap skema pembagian data.

Tabel 6. Hasil Evaluasi Model Klasifikasi

Data Split	**Model**					
	Decision Tree		**SVM**		**Naive Bayes**	
	Accuracy	***Training Time* (*s*)**	***Accuracy***	***Training Time* (*s*)**	***Accuracy***	***Training Time* (*s*)**
90:10	77.9%	10.9	86.0%	161.5	73.7%	0.9
80:20	73.9%	8.0	84.4%	127.8	73.6%	0.8
70:30	74.4%	7.0	84.3%	99.3	74.2%	0.7

Berdasarkan nilai yang tertera pada Tabel 6, diketahui bahwa model SVM menunjukkan akurasi terbaik dalam semua pengujian yang dilakukan. Oleh karena itu, dilakukan pengujian lanjutan dengan menggunakan model SVM dengan memvariasikan parameter kernel yang digunakan. Variasi kernel yang diterapkan meliputi kernel *linear*, *polynomial*, dan *radial basis function*. Hasil dari pengujian ini kemudian disajikan dalam Tabel 7.

Tabel 7. Hasil Evaluasi Model SVM

Data Split	**Support Vector Machine (SVM)**					
	Linear		**Polynomial**		**RBF**	
	Accuracy	***Training Time* (*s*)**	***Accuracy***	***Training Time* (*s*)**	***Accuracy***	***Training Time* (*s*)**
90:10	82.6%	89.5	68.7%	161.5	86.0%	153.8
80:20	82.1%	79.5	62.8%	129.0	84.4%	127.2
70:30	82.5%	72.0	59.9	99.4	84.3%	95.0

BAB V
PENUTUP

5.1. Kesimpulan

Berdasarkan hasil pengolahan dataset kategorisasi berita yang di dapat dari situs detik.com. Pengujian model terhadap *dataset* dilakukan melalui tahap *data collecting*, *data preprocessing*, *feature engineering*, pembuatan model, dan evaluasi model. Model klasifikasi yang digunakan adalah Decision Tree, Support Vector Machine (SVM), dan Naïve Bayes dengan rasio pembagian data *training* dan data *testing* sebesar 90:10, 80:20, dan 70:30. Didapatkan hasil bahwa model klasifikasi terbaik untuk mengkategorisasi berita adalah model Support Vector Machine dengan *splitting* data 90:10 dengan akurasi 86%. Akurasi terbaik pada model Naïve Bayes sebesar 74.9% dan akurasi terbaik pada Decision Tree sebesar 72.4%.

DAFTAR PUSTAKA

Aguilar-Chinea, R. M., Rodriguez, I. C., Expósito, C., Melian-Batista, B., & Moreno-Vega, J. M. (2019). Using a decision tree algorithm to predict the robustness of a transshipment schedule. Procedia Computer Science, 149, 529–536. https://doi.org/10.1016/j.procs.2019.01.172

Ariadi, D., & Fithriasari, K. (2015). Klasifikasi Berita Indonesia Menggunakan Metode Naive Bayesian Classification dan Support Vector Machine dengan Confix Stripping Stemmer. JURNAL SAINS DAN SENI ITS Vol. 4, No.2, 4(2), 248–253.

Asiyah, S. N., & Fithriasari, K. (2016). Klasifikasi Berita Online Menggunakan Metode Support Vector Machine dan K- Nearest Neighbor. Jurnal Sains Dan Seni ITS, 5(2), 317–322.

Asiyah, S. N., & Fithriasari, K. (2018). Klasifikasi Berita Online Menggunakan Metode. JURNAL SAINS DAN SENI ITS, 5(2).

Ayu Fitria, Muslim, M., & Huzain Azis. (2018). Analisis Kinerja Sistem Klasifikasi Skripsi menggunakan Metode Naïve Bayes Classifier. Prosiding SAKTI (Seminar Ilmu Komputer Dan Teknologi Informasi), 3(2), 102–106.

Bustomi. (2014). PENERAPAN ALGORITMA NAIVE BAYES UNTUK MENGKLASIFIKASI DATA NASABAH ASURANSI. PENERAPAN ALGORITMA NAIVE BAYES UNTUK MENGKLASIFIKASI DATA NASABAH ASURANSI , 8(1).

Cady, F. (2017). The Data Science Handbook. New Jersey: Wiley.

Devita, R. N., Herwanto, H. W., & Wibawa, A. P. (2018). Perbandingan Kinerja Metode Naive Bayes dan K-Nearest Neighbor untuk Klasifikasi Artikel Berbahasa indonesia. Jurnal Teknologi Informasi Dan Ilmu Komputer, 5(4), 427. https://doi.org/10.25126/jtiik.201854773

Efrizoni, L., Defit, S., Tajuddin, M., & Anggrawan, A. (2022). Komparasi Ekstraksi Fitur dalam Klasifikasi Teks Multilabel Menggunakan Algoritma Machine Learning. MATRIK : Jurnal Manajemen, Teknik Informatika Dan Rekayasa Komputer, 21(3), 653–666. https://doi.org/10.30812/matrik.v21i3.1851

Fauziah, D. A., Maududie, A., & Nuritha, I. (2018). Klasifikasi Berita Politik Menggunakan Algoritma K-nearst Neighbor. BERKALA SAINSTEK, VI(2), 106-114.

Hamzah, A. (2020). Klasifikasi Teks dengan Naïve Bayes Classifier. Prosiding Seminar Nasional.

Herwijayanti, B., Ratnawati, D. E., & Muflikhah, L. (2018). Klasifikasi Berita Online dengan menggunakan Pembobotan TF-IDF dan Cosine Similarity. Pengembangan Teknologi Informasi Dan Ilmu Komputer, 2(1), 306–312. https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/796

Hikmah, N. (2018). Pemanfaatan Text Mining dalam Pencarian Ayat AlQuran menggunakan TF-IDF dan Cosine Similarity. Jurnal Antartika, 8(1), 13–22.

Kusrini, Emha taufiq luthfi, & Universitas Amikom. (2009). Algoritma Data Mining. Yogyakarta: Penerbit Andi.

Latifah, R., Wulandari, E. S., & Kreshna, P. E. (2019). Model Decision Tree untuk Prediksi Jadwal Kerja menggunakan Scikit-Learn. Prosiding Semnastek, 0(0).

Lestari, Y. A. (2022, September 28). Perjalanan Detik.com sebagai Salah Satu Pelopor Media Online di Indonesia. Retrieved June 15, 2023, from KOMPASIANA website: https://www.kompasiana.com/yemimaanugrahlestari/63339e8708a8b577fa6b3da2/perjalanan-detikcom-sebagai-salah-satu-pelopor-media-online-di-indonesia

Parapat, I., Furqon, M., & Sutrisno. (2018). Penerapan Metode Support Vector Machine (SVM) Pada Klasifikasi Penyimpangan Tumbuh Kembang Anak. Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, 2(10), 3163–3169.

Sudianto, S., Sripamuji, A. D., Ramadhanti, I. R., Amalia, R. R., Saputra, J., & Prihatnowo, B. (2022). Penerapan Algoritma Support Vector Machine dan Multi-Layer Perceptron pada Klasifikasi Topik Berita. Jurnal Nasional Pendidikan Teknik Informatika : JANAPATI, 11(2), 84–91.

Wibowo, A., & Wajhillah, R. (2022). Information Retrieval Pemetaan Peta Jalan Penelitian Perguruan Tinggi Berbasis Dokumen Publikasi Ilmiah Dosen. Jurnal Larik: Ladang Artikel Ilmu Komputer, 2(2), 49–56. https://doi.org/10.31294/larik.v2i2.1816

LAMPIRAN

Lampiran 1. Kode Program Modul scrape_detikcom

```
import re
import requests
import pandas as pd
from bs4 import BeautifulSoup as bs

def detikcom_url(query, category_id, date_start, date_end, pages_num=1):
    url =
f"https://www.detik.com/search/searchall?query={query}&siteid={category_i
d}&sortby=time&fromdatex={date_start}&todatex={date_end}&page={pages_num}
"
    response = requests.get(url)
    page = bs(response.content, "html.parser")

    return page

def detikcom_search_results(query, category_id, date_start, date_end):
    page = detikcom_url(query, category_id, date_start, date_end)

    results = page.find("div", {"class": "search-result"})
    results_num = results.find("span", {"class": "fl text"}).get_text()
    num = int(re.findall(r'\d+', results_num)[0])
    print(f"Jumlah artikel Detikcom yang ditemukan: {num-1}")

    return num-1

def detikcom_article_need(num_result, number_need):
    pages = 999
    return number_need, pages

def detikcom_get_content(article_url):
    response = requests.get(article_url)
    content_page = bs(response.content, "html.parser")

    content_list = []
    multiple_page = content_page.find("div", {"class":
"detail__multiple"})
    if multiple_page:
        multiple_page = [x.get("href") for x in
multiple_page.find_all("a")][:-1]
        for page in multiple_page:
            response = requests.get(page)
            content_page = bs(response.content, "html.parser")
            content_list.extend([p.get_text() for p in
content_page.find_all("p") if not
p.get_text().startswith('\n\n\n\nHalaman\n\n') and
                                 p.get_text() not in ['',

'[Gambas:Instagram]',
                                                      '[Gambas:Video
20detik]',

'\r\nADVERTISEMENT\r\n',
                                                      '\r\n
ADVERTISEMENT\r\n',
                                                      '\r\n
ADVERTISEMENT\r\n  ',
```

```
                                                    '\r\n
ADVERTISEMENT\r\n  ',
                                                    '\r\n
ADVERTISEMENT\r\n',
                                                    '\r\n        SCROLL
TO RESUME CONTENT\r\n  ',
                                                    'Selengkapnya di
halaman selanjutnya.',
                                                    '\n\t\t\t\t\tAyo
share cerita pengalaman dan upload photo album travelingmu di
sini.\n\n\t\t\t\t\t\t\t\t\t\t\tSilakan Daftar atau Masuk\n']])
    else:
        content_list.extend([p.get_text() for p in
content_page.find_all("p")
                             if p.get_text() not in ['',
                                                    '[Gambas:Instagram]',
                                                    '[Gambas:Video
20detik]',

'\r\nADVERTISEMENT\r\n',
                                                    '\r\n
ADVERTISEMENT\r\n',
                                                    '\r\n
ADVERTISEMENT\r\n  ',
                                                    '\r\n
ADVERTISEMENT\r\n  ',
                                                    '\r\n
ADVERTISEMENT\r\n',
                                                    '\r\n        SCROLL
TO RESUME CONTENT\r\n  ',
                                                    '\n\t\t\t\t\tAyo
share cerita pengalaman dan upload photo album travelingmu di
sini.\n\n\t\t\t\t\t\t\t\t\t\t\tSilakan Daftar atau Masuk\n']])

    content_list = '\n\n'.join(content_list)

    return content_list

def detikcom_advertorial_check(article_url):
    response = requests.get(article_url)
    content_page = bs(response.content, "html.parser")

    author = content_page.find("meta", {"content": "Advertorial"})

    if author:
        return True
    else:
        return False

def detikcom_articles(query, category_id, article_need, from_date,
to_date, article_content=False):
    results_num = detikcom_search_results(query, category_id, from_date,
to_date)
    article_lists = []

    if results_num == 0:
        print(f'Tidak ditemukan artikel mengenai "{query}"')
        article_lists = None
    else:
        number_need, pages = detikcom_article_need(results_num,
article_need)
```

```
        print(f"Mengambil {number_need} artikel terbaru...")

        for i in range(1, pages+1):
            page = detikcom_url(query, category_id, from_date, to_date,
i)
            articles = page.find_all("article")

            for article in articles:
                title = article.find("h2", {"class": "title"}).get_text()
                category = article.find("span", {"class":
"category"}).get_text()
                publish_date = article.find("span", {"class":
"date"}).get_text().split(",")[1]
                article_url = article.find("a")["href"]

                if article_content:
                    content = detikcom_get_content(article_url)
                    if any(title in existing_article["title"] for
existing_article in article_lists) or any(content in
existing_article["content"] for existing_article in article_lists):
                        continue
                    else:
                        article_lists.append({
                            "title": title,
                            "category": category,
                            "publish_date": publish_date,
                            "article_url": article_url,
                            "content": content
                        })
                else:
                    if any(title in existing_article["title"] for
existing_article in article_lists):
                        continue
                    else:
                        article_lists.append({
                            "title": title,
                            "category": category,
                            "publish_date": publish_date,
                            "article_url": article_url
                        })

                if len(article_lists) == number_need:
                    break

            if len(article_lists) == number_need:
                break

    data = pd.DataFrame(article_lists)
    print(f"Selesai mengambil {len(article_lists)} artikel.")

    return data
```

Lampiran 2. Kode Program Modul text_cleansing

```
from nltk.tokenize import RegexpTokenizer
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from Sastrawi.StopWordRemover.StopWordRemoverFactory import
StopWordRemoverFactory

def tokenizing(doc):
    tokenizer = RegexpTokenizer(r'\w+')
```

```
    tokenizing_doc = []

    for x in doc:
        lowerdoc = x.lower()
        proses_token = tokenizer.tokenize(lowerdoc)
        tokenizing_doc.append(proses_token)

    return tokenizing_doc

def stopword_removal(doc):
    stopwords_factory = StopWordRemoverFactory()
    stopwords = stopwords_factory.get_stop_words()
    sw_doc = []
    tuple_stopword = tuple(stopwords)

    i = 0
    for a in doc:
        sw_doc.append([])
        for b in a:
            if b not in tuple_stopword:
                sw_doc[i].append(b)
        i += 1

    return sw_doc

def case_folding(doc):
    case_folding_doc = []

    i = 0
    for a in doc:
        case_folding_doc.append([])
        for b in a:
            if b.isalpha():
                case_folding_doc[i].append(b)
        i += 1

    return case_folding_doc

def stemming(doc):
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    stemmed_doc = []

    i = 0
    for a in doc:
        stemmed_doc.append([])
        for b in a:
            stemmed_doc[i].append(stemmer.stem(b))
        i += 1

    return stemmed_doc

def clean_doc(doc):
    readydoc = []

    i = 0
    for x in doc:
        readydoc.append((' '.join(map(str, doc[i]))))
        i += 1

    return readydoc
```

Lampiran 3. Kode Program Modul tf_idf

```
from nltk.tokenize import RegexpTokenizer
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from Sastrawi.StopWordRemover.StopWordRemoverFactory import
StopWordRemoverFactory

def tokenizing(doc):
    tokenizer = RegexpTokenizer(r'\w+')
    tokenizing_doc = []

    for x in doc:
        lowerdoc = x.lower()
        proses_token = tokenizer.tokenize(lowerdoc)
        tokenizing_doc.append(proses_token)

    return tokenizing_doc

def stopword_removal(doc):
    stopwords_factory = StopWordRemoverFactory()
    stopwords = stopwords_factory.get_stop_words()
    sw_doc = []
    tuple_stopword = tuple(stopwords)

    i = 0
    for a in doc:
        sw_doc.append([])
        for b in a:
            if b not in tuple_stopword:
                sw_doc[i].append(b)
        i += 1

    return sw_doc

def case_folding(doc):
    case_folding_doc = []

    i = 0
    for a in doc:
        case_folding_doc.append([])
        for b in a:
            if b.isalpha():
                case_folding_doc[i].append(b)
        i += 1

    return case_folding_doc

def stemming(doc):
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    stemmed_doc = []

    i = 0
    for a in doc:
        stemmed_doc.append([])
        for b in a:
            stemmed_doc[i].append(stemmer.stem(b))
        i += 1

    return stemmed_doc
```

```
def clean_doc(doc):
    readydoc = []

    i = 0
    for x in doc:
        readydoc.append((' '.join(map(str, doc[i]))))
        i += 1

    return readydoc
```

Lampiran 4. Kode Program Proses Pembobotan Kata

```
from func import tf_idf as tfidf

# penghitungan tf
tf = tfidf.get_tf(title_list, tfidf.get_list_word(title_list))
df_tf = tfidf.get_df_tf_wqt(tf)

# penghitungan idf
idf = tfidf.get_idf(tf, tfidf.get_list_word(title_list))
df_idf = tfidf.get_df_idf(idf)

# penghitungan bobot kata
wqt = tfidf.get_wqt(tf, idf)
df_wqt = tfidf.get_df_tf_wqt(wqt).T
df_wqt.reset_index(inplace=True)
df_wqt.drop('index', axis=1, inplace=True)

# penggabungan kolom
df_clean = pd.concat([df['title'], df['category'], df_wqt], axis=1)
```