PERBANDINGAN METODE *NAÏVE BAYES CLASSIFIER* DENGAN *K-NEAREST NEIGHBOR* (KNN) UNTUK KLASIFIKASI KATEGORI ABSTRAK SKRIPSI

SKRIPSI

Oleh :
MOHAMAD ALIF IRFAN ANSHORI
NIM. 16650029

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2020

PERBANDINGAN METODE *NAÏVE BAYES CLASSIFIER* DENGAN *K-NEAREST NEIGHBOR* (KNN) UNTUK KLASIFIKASI KATEGORI ABSTRAK SKRIPSI

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
MOHAMAD ALIF IRFAN ANSHORI
NIM. 16650029

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2020

HALAMAN PERSETUJUAN

PERBANDINGAN METODE *NAÏVE BAYES CLASSIFIER* DENGAN *K-NEAREST NEIGHBOR* (KNN) UNTUK KLASIFIKASI KATEGORI ABSTRAK SKRIPSI

SKRIPSI

Oleh :
MOHAMAD ALIF IRFAN ANSHORI
NIM. 16650029

Telah Diperiksa dan Disetujui untuk Diuji

Tanggal : 17 Mei 2020

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Cahyo Crysdian	**Ainatul Madhiyah, M.Cs**
NIP. 19740424 200901 1 008	**NIDT. 2010030102147**

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

PERBANDINGAN METODE *NAÏVE BAYES CLASSIFIER* DENGAN *K-NEAREST NEIGHBOR* (KNN) UNTUK KLASIFIKASI KATEGORI ABSTRAK SKRIPSI

SKRIPSI

Oleh :
MOHAMAD ALIF IRFAN ANSHORI
NIM. 16650029

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima sebagai Salah Satu Persyaratan
untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 6 Juni 2020

Susunan Dewan Penguji :			**Tanda Tangan**
Penguji Utama	:	Irwan Budi Santoso, M.Kom NIP. 19770103 201101 1 004	( )
Ketua Penguji	:	Fajar Rohman Hariri, M.Kom NIP. 19890515 201801 1 001	( )
Sekertaris Penguji	:	Dr. Cahyo Crysdian NIP. 19740424 200901 1 008	( )
Anggota Penguji	:	Ainatul Mardhiyah, M.Cs NIDT. 2010030102147	( )

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini,

Nama	: Mohamad Alif Irfan Anshori
NIM	: 16650029
Jurusan	: Teknik Informatika
Fakultas	: Sains dan Teknologi
Judul Skripsi	: Perbandingan Metode *Naïve Bayes Classifier* dengan *K-Nearest Neighbor* (KNN) untuk Klasifikasi Kategori Abstrak Skripsi

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambilan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka. Apabila di kemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan saya tersebut.

Malang, 8 Mei 2020
Yang membuat pernyataan,

Mohamad Alif Irfan Anshori
NIM.16650029

KATA PENGANTAR

Assalamu'alaikum Warohmatullaahi Wabarakaatuh

Segala puji bagi *Ilahi Rabbi* Allah SWT, satu-satunya *dzat* yang selalu memberikan rahmat serta kekuatan sehingga penulis mampu menyelesaikan skripsi dengan judul "Perbandingan Metode *Naïve Bayes Classifier* dengan *K-Nearest Neighbor* (KNN) untuk Klasifikasi Kategori Abstrak Skripsi" dengan lancar dan baik. Shalawat sekaligus salam penghormatan kepada junjungan kita, nabi *akhirruz zaman*, Nabi Muhamad SAW yang telah menuntun umatnya dari zaman yang gelap, *jahil*, menuju zaman yang terang benderang, Islam *rahmatan lil-'aalamin*.

Selama proses pengerjaan skripsi ini, penulis mendapatkan banyak bantuan serta bimbingan dari berbagai pihak. Oleh karena itu, dengan hati yang lapang penulis mengucapkan terima kasih serta doa yang tulus kepada:

1. Dr. Cahyo Crysdian, selaku Ketua Jurusan Teknik Informatika sekaligus selaku dosen pembimbing I yang telah dengan sabar membimbing penulis, memberikan masukan, saran dan juga arahan hingga akhir.
2. Ainatul Mardhiyah, M.Cs, selaku dosen pembimbing II yang telah teliti membimbing penulis untuk dapat mencapai hasil skripsi yang lebih baik.
3. Irwan Budi Santoso, M.T dan Fajar Rohman Hariri, M.Kom selaku dosen penguji dengan sikap profesional telah menguji seluruh proses ujian skripsi penulis mulai dari seminar proposal hingga sidang skripsi dengan lancar.
4. Seluruh jajaran staf dan dosen jurusan Teknik Informatika yang secara langsung maupun tidak langsung terlibat dalam proses pengerjaan skripsi.

5. Ibu, bapak dan adik menjadi manusia yang luar biasa memberikan dukungan dan semangat tanpa lelah kepada penulis supaya terus berusaha menyelesaikan skripsi tepat waktu.
6. Seluruh sahabat, teman-teman se-angkatan, kakak tingkat, adik tingkat dan seluruh teman seperjuangan yang secara langsung maupun tidak langsung memberikan bantuan dalam proses pengerjaan skripsi.
7. Para peneliti yang telah meneliti algoritma Naive Bayes Classifier dan K-Nearest Neighbor yang menjadi acuan penulis menyelesaikan skripsi ini. Serta seluruh pihak yang tidak dapat penulis sebutkan satu-persatu.

Peneliti menyadari bahwa penelitian ini masih jauh dari kata sempurna. Maka dari itu penulis membuka kesempatan selebar-lebarnya untuk setiap saran dan kritik yang membangun. Terlepas dari segala kekurangan skripsi ini, peneliti berharap ada manfaat yang dapat diambil oleh kita semua. *Aamin ya rabbal 'aalamin.*

Wassalamu'alaikum Warohmatullaahi Wabarakaatuh

Malang, 18 Mei 2020

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

ABSTRAK

Anshori, Mohamad A.I. 2020. **Perbandingan Metode *Naïve Bayes Classifier* Dengan *K-Nearest Neighbor* (KNN) untuk Klasifikasi Kategori Abstrak Skripsi**. Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Dr. Cahyo Crysdian, (II) Ainatul Mardhiyah, M.Cs.

Kata kunci: Klasifikasi dokumen, *Naïve Bayes*, *K-Nearest Neighbor* (KNN)

Klasifikasi merupakan salah satu substansi *Data Mining* yang sering digunakan untuk mengelompokkan data berdasarkan kategori atau variabel tertentu. Diantara metode klasifikasi yang sering digunakan untuk mengklasifikasikan data berupa dokumen adalah *Naïve Bayes* dan *K-Nearest Neighbor* (KNN). Penelitian ini membahas secara rinci perbandingan kedua metode tersebut dengan mengukur akurasi dan lama waktu proses dengan 4 skenario uji coba. Dari hasil penelitian, *Naïve Bayes* mengungguli KNN dengan rata-rata hasil akurasi *Naïve Bayes* sebesar 76.81% dan KNN bernilai K 3, 6, 9 berturut-turut sebesar 72,47%, 71,12%, 71,56%. Namun jika hanya mengambil akurasi tertinggi, KNN unggul dengan perbandingan *Naïve Bayes* sebesar 72.22% dengan KNN sebesar 77.78%. Dengan catatan, selisih masih bisa bertambah berbanding lurus dengan jumlah perbendaharaan dokumen latih. Rata-rata waktu proses yang dibutuhkan *Naïve Bayes* sebanyak 0.31 detik berbanding KNN sebanyak 1.09 detik.

ABSTRACT

Anshori, Mohamad A.I. 2020. **Naïve Bayes Classifier Method Comparison with K-Nearest Neighbor (KNN) for Thesis Abstract Category Classification**. Thesis. Informatics Engineering Department of Science and Technology Faculty Islamic State University Maulana Malik Ibrahim Malang. Supervisor: (I) Dr. Cahyo Crysdian, (II) Ainatul Mardhiyah, M.Cs.

Keywords: Document classification, Naïve Bayes, K-Nearest Neighbor (KNN)

Classification is one of the substance of Data Mining which is often used to group data based on certain categories or variables. Among the classification methods that are often used to classify data in the form of documents are Naïve Bayes and K-Nearest Neighbor (KNN). This study discusses in detail the comparison of the two methods by measuring the accuracy and length of processing time with 4 trial scenarios. From the results of the study, Naïve Bayes outperformed KNN with an average accuracy of 76.81% Naïve Bayes and KNN K 3, 6, 9 respectively 72.47%, 71.12%, 71.56%. But if it only takes the highest accuracy, KNN is superior with the Naïve Bayes ratio of 72.22% with KNN of 77.78%. With notes, the difference can still increase directly proportional to the amount of treasury training documents. The average processing time required by Naïve Bayes is 0.31 seconds compared to KNN as much as 1.09 seconds.

مستخلص البحث

انصاري, محمد اليف ارفان. 2020. مقارنة طريقة Naïve Bayes Classifier واقرب جار K-Nearest Neighbor (KNN) لتصنيف اطروحة الملخصت. قسم المعلوماتية. كلية العلوم والتكنولوجيا. جامعة الإسلامية الحكومية مولانا مالك إبراهيم مالانج.

المشرف : (I) جحيو كريسديان الدكتور (II) عينة المرْضيّاة الماجيستير

الكلمات الرئيسية : تصنف الوثيقة, Naive Bayes Classifier, K-Nearest Neighbor

التصنيف هو أحد عناصر استخراج البيانات الذي يُستخدم غالبًا لتجميع البيانات بناءً على فئات أو متغيرات معينة. بين طرق التصنيف التي غالبًا تُستخدمه لتصنيف البيانات في شكل مستندات هي Naïve Bayes و K-Nearest Neighbor (KNN). يناقش هذا البحث بالتفصيل مقارنة الطريقتين عن طريق قياس دقة وطول مدة المعالجة مع ٤ سيناريوهات تجريبية. من نتائج البحث، تفوقت Naïve Bayes على KNN بمتوسط دقة ٧٦,٨١٪ Naïve Bayes و KNN K٣،٦،٩ على التوالي ٧٢,٤٧٪ ، ٧١,١٢٪ ، ٧١,٥٦٪. ولكن إذا كانت فقط تأخذ أعلى دقة، فإن KNN تتفوق مع نسبة Naïve Bayes ٧٢,٢٢٪ مع KNN بنسبة ٧٧,٧٨٪. مع الملاحظة، لا يزال الفرق يمكن أن يتناسب طرديا مع كمية وثائق التدريب على الخزانة. معدل وقت المعالجة الذي تتطلبه Naïve Bayes هو ٠,٣١ ثانية مقارنة بـــ KNN بقدر ١,٠٩ ثانية.

BAB I
PENDAHULULAN

1.1. Latar Belakang

Pada era modern sekarang ini, data adalah salah satu aset penting dalam kehidupan. Seiring dengan pesatnya perkembangan teknologi, kebutuhan akan data juga meningkat. Pada tahun 2025, diprediksi seluruh data yang dihasilkan manusia akan mencapai 175ZB (IDC, 2018). Data berukuran besar yang sudah disimpan jarang digunakan secara optimal karena manusia seringkali tidak memiliki waktu dan kemampuan yang cukup untuk mengelolanya (Sitanggang, 2017). Data bervolume besar seperti data teks, jauh melampaui kapasitas pengolahan manusia yang sangat terbatas (Toyota, 2012). Untuk mengatasi hal tersebut, diperlukan otomasi sehingga kumpulan data-data tersebut dapat dikelompokkan secara otomatis sesuai dengan kelasnya.

Data mining sangat sesuai untuk diterapkan pada data berukuran besar (Turban, 2015). Metode data mining yang akan diterapkan dalam penelitian ini adalah klasifikasi. Klasifikasi dokumen adalah proses melabeli dokumen sesuai dengan kategori yang dimilikinya (Hariri, 2015). Adapun beberapa penelitian mengenai klasifikasi dokumen yang telah dilakukan sebelumnya adalah penelitian Klasifikasi Dokumen Teks Menggunakan Algoritma *Naïve Bayes* dengan Bahasa Pemograman Java oleh Silfia Andini pada tahun 2013. Menggunakan artikel berita dengan metode *Naïve Bayes*, penelitian ini bisa membantu *user* dalam memilih dan mengkategorikan dokumen. Kesimpulan pada penelitian ini didapatkan bahwa metode *Naïve Bayes* cukup baik dalam mengklasifikasikan dokumen, namun tidak dijabarkan berapa persen akurasinya (Andini, 2013). Penelitian tentang klasifikasi teks juga pernah dilakukan oleh Aji Suprapto untuk mengklasifikasikan opini pada tahun 2017 dengan judul Sistem Klasifikasi Opini pengguna Maskapai Penerbangan di Indonesia pada Jejaring Sosial Twitter menggunakan Metode *K-Nearest Neighbor*. Penelitian ini menghasilkan akurasi sebesar 86,674% untuk sentimen positif dan 93,345% untuk sentimen negatif, namun karena data yang diuji merupakan media bebas dan tidak terikat, maka data yang diuji pun menjadi kurang konsisten kaidah penulisannya.

Sehingga diperlukan koleksi kamus lanjutan untuk mengurangi ambiguitas dan meningkatkan konsistensi (Suprapto, 2017).

Studi kasus yang digunakan pada penelitian ini adalah data abstrak tugas akhir mahasiswa Universitas Airlangga dengan subjek *Transportation and Communications* dengan total 297 data abstrak pada *website repository* Unair yang diakses pada 10 Juni 2019. Ada 3 bidang minat yang telah diklasifikasikan sebelumnya, yaitu *Transportation and communications* (131), *Transportation geography. Trade routes* (23) dan *Automotive transportation Including trucking, bus lines, and taxicab service* (56). Setiap dokumen abstrak yang ada dibawah satu bidang minat memiliki kelas yang berbeda dari bidang minat yang lain. Dalam penelitian ini, jika ada dokumen yang beririsan dengan bidang minat yang lain, maka dokumen tersebut tidak akan digunakan sebagai dokumen uji maupun dokumen latih untuk menghindari ambiguitas.

Penelitian ini berfokus untuk mengukur performa algoritma metode *Naïve Bayes* dan *K-Nearest Neighbor*. Adapun parameter pengujiannya meliputi akurasi dan waktu proses yang diperlukan untuk mengklasifikasikan dokumen abstrak tugas akhir skripsi. Untuk *Stemming* dokumen, penelitian ini menggunakan algoritma Nazief & Adriani karena algoritma ini memiliki akurasi yang lebih baik dibandingkan algoritma porter (Agusta, 2009). Berbeda dengan penelitian sebelumnya, penelitian ini menggunakan dokumen teks abstrak tugas akhir berbahasa Indonesia dan mengklasifikasikan berdasarkan bidang minat penelitian. Objek penelitian merupakan karya ilmiah yang terikat sehingga memiliki konsistensi pada kaidah penulisannya. Uji coba pada penelitian ini menggunakan beberapa skenario, dari jumlah dokumen uji dan dokumen latih, dan faktor-faktor penentu lainnya untuk mendapatkan hasil yang bervariasi antara kedua metode yang sudah disebutkan diatas.

Diharapkan *output* dari penelitian ini dapat bermanfaat untuk berbagai pihak, khususnya sebagai referensi untuk penelitian sejenis yang akan datang untuk mengetahui mana metode yang paling baik untuk digunakan sebagai klasifikator dokumen. Hal ini untuk mengamalkan perintah Allah dalam potongan Q.S. Al-Baqarah/2:42, yang berbunyi:

وَلَا تَلْبِسُواْ ٱلْحَقَّ بِٱلْبَـٰطِلِ وَتَكْتُمُواْ ٱلْحَقَّ وَأَنتُمْ تَعْلَمُونَ

“Janganlah kalian campur-adukkan antara kebenaran dan kebatilan, dan kalian sembunyikan yang benar padahal kamu mengetahuinya”.

1.2. Pernyataan Masalah

Berdasarkan latar belakang masalah yang telah dipaparkan diatas, pertanyaan masalah pada penelitian ini adalah sebagai berikut.

1. Seberapa akurat hasil klasifikasi menggunakan metode *Naïve Bayes Classifier* berbanding *K-Nearest Neighbor*?
2. Seberapa lama waktu yang diperlukan oleh proses klasifikasi metode *Naïve Bayes Classifier* berbanding *K-Nearest Neighbor*?

1.3. Tujuan Penelitian

Adapun tujuan yang ingin dicapai dari penelitian yang akan dilakukan ini adalah sebagai berikut:

1. Untuk mengukur seberapa akurat hasil klasifikasi metode *Naïve Bayes Classifier* dan *K-Nearest Neighbor*.
2. Untuk mengukur seberapa lama waktu yang diperlukan dalam proses klasifikasi antara metode *Naïve Bayes Classifier* dan *K-Nearest Neighbor*.

1.4. Manfaat Penelitian

Penelitian ini akan menghasilkan *output* data performa dan akurasi antara metode *Naïve Bayes Classifier* berbanding *K-Nearest Neighbor* untuk mengklasifikasikan dokumen uji berupa dokumen berbasis teks. Adapun pihak-pihak yang dapat memanfaatkan *output* dari penelitian ini adalah:

a. Peneliti *Data Mining/Text Mining* untuk tujuan rujukan penelitian;
b. Komunitas *Data Science* untuk tujuan diskusi; dan pihak-pihak lain yang berkaitan.

1.5. Batasan Masalah

Adapun batasan masalah dari penelitian ini adalah sebagai berikut.

1. Abstrak yang dianalisis adalah dokumen berbahasa Indonesia.
2. Kelas yang ditentukan berdasarkan konsentrasi yang telah terdefinisi pada *website* repository Unair dengan Subjek *Transportation and Communication* yang diakses pada tanggal 10 Juni 2019. Diantara kelas kategorinya adalah HE1-9990 *Transportation and communications,* HE323-328 *Transportation geography. Trade routes,* HE5601-5725 *Automotive transportation Including trucking, bus lines, and taxicab service*.

BAB II
TINJAUAN PUSTAKA

2.1. *Data Mining*

Data mining merupakan proses yang bertujuan menemukan pengetahuan yang menarik dari data dalam jumlah besar (Han, 2000). Pada prakteknya, dalam aplikasi, tahapan *data mining* dapat dibagi menjadi enam proses utama: *business understanding*, *data understanding*, *data preparation*, *modeling*, *evaluation* dan *deployment* (Zhao, 2013). *Data Mining* adalah bidang multi-disiplin yang didalamnya termasuk *machine learning*, *statistics*, *database*, *artificial intelligence*, *information retrieval* dan *visualization*. *Data mining* juga disebut sebagai *knowledge discovery in databases* (KDD) karena kemampuannya memperoleh pola atau informasi yang berguna dari kumpulan data mentah, misalnya gambar, web, *database*, teks, dan lain-lain. Tugas *data mining* yang paling umum adalah klasifikasi (*supervised learning)*, pengelompokan (*unsupervised learning/clustering*), *Isequential pattern mining*, dan *association rule mining* (Liu, 2007).

Dari kumpulan data mentah berukuran besar yang tidak memiliki informasi utuh, *data mining* yang mencakup algoritma inti memungkinkan seseorang untuk memperoleh wawasan dan pengetahuan. *Data mining* tergolong bidang interdisipliner dan bagian dari proses penemuan pengetahuan yang lebih besar, yang mencakup tugas pembersihan data, reduksi data, pra-pengolahan seperti ekstraksi data, fitur konstruksi, dan pencampuran data, dan serta *hypothesis confirmation and generation, pattern and model interpretation* dan sebagainya (Zaki, 2014).

2.2. *Text Mining*

Text mining dideskripsikan secara umum merupakan proses pengetahuan intensif dimana pengguna menggunakan seperangkat alat analisis untuk berinteraksi dengan koleksi dokumen dari waktu ke waktu (Fieldman, 2007). Dengan teknik yang sama dengan *Data Mining*, misi *Text Mining* adalah untuk

mendapatkan intisari informasi yang bermanfaat dari sumber data dengan mengidentifikasi dan mengeksplorasi pola yang menarik. Pola yang menarik merangkum sebuah informasi yang tidak ditemukan pada data mentah dalam *database*. Namun, disimpulkan dalam rangkaian data teks dokumen yang tidak terstruktur.

Penelitian di bidang *Text Mining* berhubungan dengan masalah yang terkait dengan pemodelan pola, pencarian informasi atau ekstraksi, pengelompokkan, klasifikasi, dan representasi teks. Dalam hal ini, pemilihan karakteristik domain dan prosedur penelitian menjadi sangat penting. *Text Mining* adalah proses mengeksplorasi dan menganalisis sejumlah besar data teks tidak terstruktur yang dibantu oleh perangkat lunak yang dapat mengidentifikasi konsep, pola, topik, kata kunci, dan atribut lainnya dalam data. Ini juga dikenal sebagai analisis teks, meskipun beberapa orang menarik perbedaan antara dua istilah; dalam pandangan itu, analitik teks adalah aplikasi yang diaktifkan oleh penggunaan teknik *Text Mining* untuk memilah-milah *dataset* (Hotho, 2005).

Text Mining telah menjadi lebih praktis bagi para ilmuwan data dan pengguna lain karena pengembangan *platform* data besar dan algoritma pembelajaran mendalam yang dapat menganalisis kumpulan data yang tidak terstruktur secara besar-besaran. Langkah awal sebelum suatu data teks dianalisis menggunakan metode-metode dalam *Text Mining* adalah melakukan *pre-processing* teks. Selanjutnya, setelah didapatkan data yang siap diolah, analisis *text mining* dapat dilakukan (Fieldman, 2007).

2.3. *Text Preprocessing*

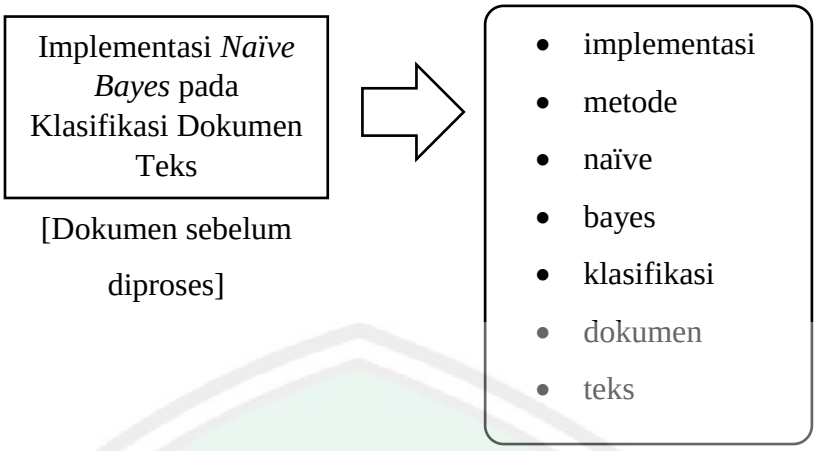
Struktur data yang baik dapat secara otomatis memfasilitasi proses perhitungan. Dalam *text mining*, informasi yang akan diekstraksi berisi informasi-informasi yang ambigu secara struktural. Oleh karena itu, perlu untuk mengubah formulir menjadi data terstruktur sesuai dengan kebutuhan proses *data mining*, yang biasanya merupakan nilai numerik. Proses ini umumnya disebut dengan "*Text Preprocessing"* (Fieldman, 2007).

Text preprocessing adalah tahapan awal dari *text mining* yang bertujuan untuk mempersiapkan teks menjadi data yang akan mengalami pengolahan pada tahapan

berikutnya. Pada penelitian ini, tahap ini meliputi beberapa proses yaitu *Tokenizing, Case Folding, Stopword Removal* dan *Stemming.* Tahapan *pre processing* ini dilakukan agar dalam klasifikasi dapat diproses dengan baik. Tahapan dalam *preprocessing text* adalah sebagai berikut:

a. *Case Folding*, merupakan proses untuk mengubah semua karakter pada teks menjadi huruf kecil. Karakter yang diproses hanya huruf "a" hingga "z" dan selain karakter tersebut akan dihilangkan seperti tanda baca titik (.), koma (,), dan angka (Weiss, 2010).
b. *Tokenizing*, merupakan proses memecah yang semula berupa kalimat menjadi kata-kata atau memutus urutan string menjadi potongan-potongan seperti kata-kata berdasarkan tiap kata yang menyusunnya (Weiss, 2010).
c. *Stopwords*, merupakan kosakata yang bukan merupakan kata unik atau ciri pada suatu dokumen atau tidak menyampaikan pesan apapun secara signifikan pada teks atau kalimat (Tala, 2003). Kosakata yang dimaksudkan adalah kata penghubung dan kata keterangan yang bukan merupakan kata unik misalnya "sebuah", "oleh", "pada", dan sebagainya.
d. *Stemming*, yakni proses untuk mendapatkan kata dasar dengan cara menghilangkan awalan, akhiran, sisipan, dan *confixes* (kombinasi dari awalan dan akhiran) (Adriani, 2007). Algoritma *Stemming* yang digunakan adalah Nazief Adriani.

Beberapa contoh tindakan yang dapat dilakukan pada tahap ini, mulai dari tindakan yang bersifat kompleks seperti *partofspeech* (pos) *tagging*, *parse tree*, hingga tindakan yang bersifat sederhana seperti proses parsing sederhana terhadap teks, yaitu memecah suatu kalimat menjadi sekumpulan kata. Contoh dari tahap ini seperti pada Gambar 2.1 berikut.



Gambar 2.1. Contoh *Text Preprocessing*

2.4. *Stemming* Nazief Adriani

Stemming adalah proses pengubahan kata dari kata berimbuhan menjadi kata dasarnya. Proses *Stemming* pada teks berbahasa Indonesia meliputi identifikasi dan pengapusan *suffix*, *prefix*, dan *confix* sehingga dapat menjadi bentuk kata dasar (Adriani, 2007). Algoritma *Stemming* Nazief Adriani didasarkan pada aturan leksikal komprehensif yang merangkum apakah afiks dapat mencakup prefiks, sufiks, dan konfiks (kombinasi prefiks dan sufiks yang juga disebut sebagai *circumfixes*). Algoritma Stemming Nazief Adriani memiliki tahapan sebagai berikut (Asian, 2007).

1) Cari kata yang akan diproses dalam kamus kata dasar, jika ditemukan maka diasumsikan kata adalah *root word* dan algoritma ini berhenti.
2) Hapus *inflectional suffixes* (“-lah”, “-kah”, “tah”, atau “-pun”) dan dilanjutkan dengan penghapusan *possessive pronouns* (“-ku”, “-mu” atau “-nya”) jika ada.
3) Hapus *derivational suffixes* (“-i”, “-an” atau “-kan”) seperti contoh kata “membelikan” yang akan menjadi kata “membeli”. Jika sampai tahap ini masih belum ditemukan dalam kamus kata dasar, maka dilanjutkan dalam penghapusan *prefix* di tahap selanjutnya.
4) Hapus *derivational prefixes* (“be-“, “di-“, “ke-“, “me-“, “pe-“, “se-“, atau “te-“).

   a. Proses ini berhenti jika:

- Kata awal dalam langkah 3 mempunyai gabungan awalan dan imbuhan yang tidak diijinkan dalam Tabel 2.1

Tabel 2.1 Gabungan awalan dan akhiran yang tidak diizinkan

Awalan	Akhiran
“ber-“	“-i”
“di-“	“-an”
“ke-“	“-i”, -kan”
“me-“	“-an”
“ter-“	“-an”
“per-“	“-an”

- Awalan yang dihilangkan sama dengan awalan yang dihilangkan sebelumnya.
- Awalan telah dihilangkan sebanyak 3 kali.

b. Identifikasi tipe awalan dan disambiguitasnya. Awalan ini dibagi menjadi 2 tipe:
   - Plain: awalan (“di-“, “ke-“ atau “se-“) dapat dihilangkan secara langsung.
   - Complex: awalan (“be-“, “te-“, “me-“ atau “pe-“) harus dianalisis ambiguitasnya mennggunakan aturan pada Tabel 2.2 karena memiliki varian yang berbeda. Awalan “me-“ bias menajdi “mem-“, “men-“, “meng-“ atau “meny-“ tergantung pada huruf di awal.

Tabel 2.2. Aturan prefiks dan sufiks pada algoritma Nazief dan Adriani

Rule	*Construct*	*Return*
1	berV…	ber-V… | be-rV…
2	berCAP…	ber-CAP… where C!=’r’ and P!=’er’
3	berCAerV…	ber-CAerV… where C!=’r’
4	belajar…	bel-ajar
5	beC_1erC_2	beC_1erC_2... where C_1!={‘r’|’I’}
6	terV…	ter-V… | te-rV…
7	terCerV…	ter-CerV… where C!=’r’
8	terCP…	ter-CP… where C!=’r’ and P!=’er’
9	teC_1erC_2	teC_1erC_2... where C_1!=’r’
10	me{l|r|w|y}V…	me-{l|r|w|y}V…
11	mem{b|f|v}…	mem-{b|f|v}…
12	mempe{r|l}…	mem-pe…
13	mem{rV|V}…	me-m{rV|V}… |me-p|{rV|V}…
14	men{c|d|j|z}…	men-{c|d|j|z}…
15	menV…	me-nV…|meng-kV…
16	meng{g|h|q}…	meng-{g|h|q}…
17	mengV…	meng-V…|meng…kV…
18	menyV…	meny-sV…
19	mempV…	mem-pV… where V!=’e’
20	pe{w|y}V…	pe-{w|y}V…
21	perV…	per-V…|pe-rV…
22	perCAP…	per-CAP… where C!=’r’ and P!=’er’
23	perCAerV…	per-CAerV… where C!=’r’
24	pem{b|f|v}…	pem-{b|f|v}…
25	pem{rV|V}…	pe-m{rV|V}… |pe-p|{rV|V}…
26	pen{c|d|j|z}…	pen-{c|d|j|z}…
27	penV…	pe-nV…|pe-tV…
28	peng{g|h|q}…	peng-{g|h|q}…
29	pengV…	peng-V…|peng…kV…
30	penyV…	peny-sV…
31	pelV…	pe-lV…Exception: for “pelajar” return ajar
32	peCerV…	per-erV… where C!={r|w|y|l|m|n}
33	peCP…	pe-CP… where C!={r|w|y|l|m|n} and P!=’er’

c. Jika dalam kamus masih tidak ditemukan, maka ulangi langkah 4 ini secara berulang sampai menemukan akar katanya atau sampai melanggar kondisi 4a. Jika kondisi 4a terjadi, penghapusan awalan rekursif tidak berhasil

karena tidak ada awalan yang berlaku untuk dihapus dan dilanjutkan pada tahap selanjutnya.

5) Jika semua proses di atas gagal, maka algoritma mengembalikan kata awalnya.

2.5. *Naïve Bayes Classifier*

Algoritma *Naïve Bayes Classifier* merupakan algoritma yang digunakan untuk mencari nilai *probabilities* tertinggi untuk mengklasifikasi data uji pada kategori yang paling tepat (Fieldman, 2007). Sebuah keuntungan dari *Naïve Bayes Classifier* adalah bahwa dia memerlukan sejumlah kecil data pelatihan untuk mengestimasi parameter (sarana dan varian dari variabel) yang diperlukan untuk klasifikasi. Karena variabel bebas diasumsikan, hanya varian dari variabel-variabel untuk setiap kebutuhan kelas yang akan ditentukan dan tidak seluruh matriks kovarian.

Dalam algoritma *Naïve Bayes Classifier*, setiap dokumen direpresentasikan dengan pasangan atribut "x1, x2, x3,…xn" dimana x1 adalah kata pertama, x2 adalah kata kedua dan seterusnya. Sedangkan V adalah himpunan kategori. Pada saat klasifikasi algoritma akan mencari probablititas tertinggi dari semua kategori dokumen yang diujikan (VMAP), persamaannya adalah sebagai berikut:

$$V_{MAP} = \underset{V\ j\ e\ V}{arg\ max} \frac{P(x1, x2, x3, \dots, xn|V_j)P(V_j)}{P(x1, x2, x3, \dots, xn)} \quad \text{.....(2.1)}$$

Untuk P(x1, x2, x3,…xn) nilainya konstan untuk semua kategori (Vj) sehingga persamaan dapat ditulis sebagai berikut:

$$V_{MAP} = \underset{V\ j\ e\ V}{arg\ max}\ P(x1, x2, x3, \dots, xn|V_j)P(V_j) \quad \text{.....(2.2)}$$

Persamaan di atas dapat disederhanakan menjadi sebagai berikut:

$$V_{MAP} = \underset{V\ j\ e\ V}{arg\ max} \prod_{i=1}^{n} P(xi|V_j)P(V_j) \quad \text{.....(2.3)}$$

Adapun ringkasan Algoritma dibagi menjadi 2 (Andini, 2013):

1. Proses Pelatihan: Input adalah dokumen-dokumen contoh yang telah diketahui kategorinya.
   a. Kosakata: Himpunan semua kata yang unik dari dokumen-dokumen contoh
   b. Untuk setiap kategori V_j lakukan:

1) $Docs_j$: Himpunan dokumen-dokumen yang berada pada kateogori V_j.
2) Hitung $P(V_j)$ dengan persamaan 2.3
3) Untuk setiap kata W_k pada kosakata, lakukan perhitungan $P(W_k \mid V_j)$

2. Proses Klasifikasi: Input adalah dokumen yang belum diketahui kategorinya. Kemudian hasilkan Vmap sesuai dengan persamaan 2.3 dengan menggunakan $P(V_j)$ dan $P(W_k \mid V_j)$ yang telah diperoleh dari pelatihan.

2.6. *Laplace Correction*

Laplace Correction atau *Laplace Smoothing* merupakan metode yang banyak digunakan, sekaligus *smoothing* yang disebut sebagai *default* dan tertua yang pernah di implementasikan pada *Naïve Bayes Classifier* (C & J, 2012). Laplace *smoothing* juga disebut dengan *Add-one smoothing* karena metode *smoothing* ini menambahkan angka 1 pada setiap frekuensi token yang didapat. Rumus *Naïve Bayes*sian dengan *Laplace Smoothing*, dapat dihitung menggunakan formula sebagai berikut :

$$P\ (t|c) = \frac{T_{ct\prime} + 1}{\sum_{t\prime \in} V\ T_{ct\prime} + |V|} \qquad \text{.....(2.4)}$$

Dimana *P(t|c)* adalah peluang munculnya sebuah kata dalam sebuah kelas, dan |V| merupakan jumlah kata unik pada semua kelas.

2.7. *K-Nearest Neighbor*

K-Nearest Neighbor (KNN) merupakan salah satu algoritma pembelajaran mesin sederhana. Hal ini hanya didasarkan pada gagasan bahwa suatu objek yang dekat satu sama lain juga akan memiliki karakteristik yang mirip. Ini berarti jika kita mengetahui ciri-ciri dari salah satu objek, maka kita juga dapat memprediksi objek lain berdasarkan tetangga terdekatnya. KNN adalah improvisasi lanjutan dari teknik klasifikasi *Nearest Neighbor*. Hal ini didasarkan pada gagasan bahwa setiap contoh baru dapat diklasifikasikan oleh suara mayoritas dari K tetangga, dimana K adalah bilangan bulat positif, dan biasanya dengan jumlah kecil (Khamis, 2014). Algoritma klasifikasi KNN memprediksi kategori tes sampel sesuai dengan sampel pelatihan K yang merupakan tetangga terdekat dengan sampel uji, dan memasukkan ke dalam kategori yang memiliki kategori *probabilities* terbesar (Suguna, 2010).

Tujuan dari algoritma KNN adalah untuk mengklasifikasikan objek baru berdasarkan atribut dan sampel data. Hasil sampel uji baru diklasifikasikan menurut sebagian besar kategori di KNN. Dalam proses klasifikasi, KNN tidak menggunakan model apapun untuk dicocokkan, tetapi hanya didasarkan pada memori. Algoritma KKN menggunakan klasifikasi ketetanggaan sebagai nilai prediksi sampel uji yang baru (Krisdandi, 2013).

Menurut Kurniawan (2012), algoritma *K-Nearest Neighbor* dijabarkan dengan tahapan berikut.

1. Menentukan nilai K (jumlah tetangga terdekat);
2. Menghitung jarak dokumen yang akan dievaluasi dengan semua dokumen latih;
3. Mengurutkan secara *descending* jarak dokumen latih yang terbentuk hingga nilai K.
4. Mengganti variabel dokumen dengan kelasnya;
5. Mencari modus dari kelas yang muncul dan menetapkannya sebagai kelas pemenang.

Metode klasifikasi KNN memiliki beberapa tahap, yang pertama nilai K yang merupakan jumlah tetangga terdekat yang akan menentukan kueri baru masuk ke kelas mana ditentukan. Tahap kedua, K tetangga terdekat dicari dengan cara menghitung jarak titik kueri dengan titik *training*. Tahap ketiga, setelah mengetahui jarak masing-masing titik training dengan titik kueri, kemudian lihat nilai yang paling kecil. Tahap keempat ambil K nilai terkecil selanjutnya lihat kelasnya. Kelas yang paling banyak merupakan kelas dari kueri baru (Pramesti, 2013).

2.8. *Feature Weighting* WIDF

Feature weighting adalah proses penetapan nilai pada setiap *feature* berdasarkan korelasi dari masing-masing fitur dan dampaknya pada hasil klasifikasi. Proses ini akan menghasilkan nilai bobot sebagai indikator untuk menentukan pentingnya setiap kata dalam dokumen (Purnomo, 2010).

Metode WIDF merupakan peningkatan dari metode IDF. Kelemahan dari metode tersebut adalah semua dokumen yang berisikan rangkaian kata tertentu disamakan dengan komputasi biner. Dengan meningkatkan frekuensi dan fungsi

pengumpulan dokumen, metode WIDF telah membuktikan kinerjanya, yang dapat disimpulkan lebih baik dari *feature weighting* TF dan TF-IDF (Purnomo, 2010). Adapun formula metode WIDF ditunjukkan pada persamaan berikut:

$$WIDF\ (d,t) = \frac{TF(\mathrm{d},\mathrm{t})}{\sum_{i \epsilon D} TF(i,t)} \quad \text{.....(2.5)}$$

2.9. *Cosine Similarity*

Cosine Similarity adalah metode yang menghitung kesamaan antara dua vektor n-dimensi dengan menemukan kosinus dari sudut antara dua vektor n-dimensi, dan biasanya digunakan untuk membandingkan dokumen dalam *text mining* (Zhiqiang, 2009). *Cosine Similarity* digunakan untuk menghitung korelasi kueri dengan dokumen. Menentukan korelasi kueri dengan dokumen untuk mengukur kemiripan antara vektor dokumen dengan vektor kueri. Semakin mirip vektor kueri dengan vektor dokumen, semakin dokumen dapat dilihat berdasarkan kueri.

Adapun formula yang digunakan untuk menghitung *Cosine Similarity* adalah sebagai berikut. (Ye, 2014)

$$cosSim(X, d_j = \frac{\sum_{i=1}^{m} \mathrm{x}_i . d_{ji}}{\sqrt{\sum_{i=1}^{m} x^2{}_i} \cdot \sqrt{\sum_{i=1}^{m} d^2{}_{ji}}} \quad \text{.....(2.6)}$$

X = dokumen uji

dj = dokumen sampel

xi = nilai bobot *term i* pada dokumen

dji = nilai bobot *term i* pada dokumen sampel

2.10. Klasifikasi Dokumen

Penelitian mengenai klasifikasi dokumen telah dilakukan sebelumnya yaitu penelitian Klasifikasi Dokumen Teks Menggunakan Algoritma *Naïve Bayes* dengan Bahasa Pemograman Java oleh Silfia Andini pada tahun 2013. Menggunakan artikel berita sebagai objek, dengan metode *Naïve Bayes* bisa membantu user dalam memilih dan mengkategorikan dokumen. Pada penelitian ini didapatkan bahwa metode *Naïve Bayes* cukup baik dalam mengklasifikasikan dokumen, namun tidak dijabarkan berapa persen akurasinya (Andini, 2013).

Penelitian serupa juga pernah dilakukan oleh Jeevanandam Jotheeswaran untuk mengklasifikasikan opini pada tahun 2013 dengan judul *Opinion Mining Using Decision Tree Based Feature Selection Through Manhattan Hierarchical Cluster Measure,* penelitian ini meneliti metode seleksi fitur PCA untuk klasifikasi opini dengan menggunakan IMDb *dataset*. Klasifikasi dengan menggunakan ekstraksi fitur menggunakan PCA akurasinya lebih baik sekitar 5%, namun masih perlu adanya penelitian dan skenario untuk metode PCA (Jotheeswaran, 2013).

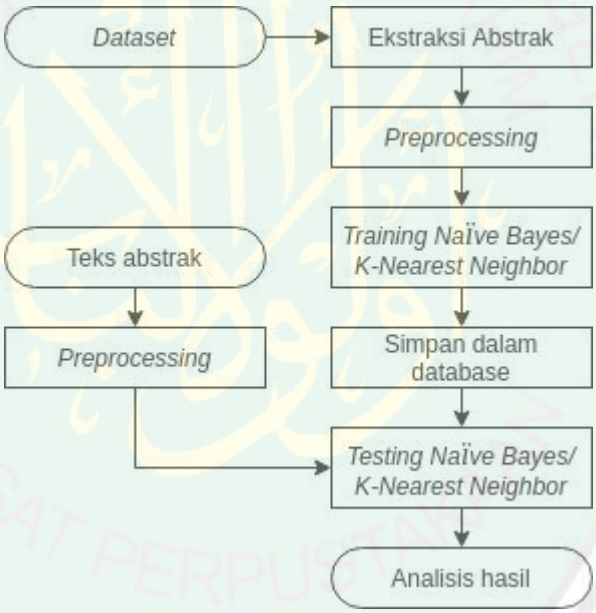
Aji Suprapto menulis penelitian untuk mengklasifikasikan opini pada tahun 2017 dengan judul Sistem Klasifikasi Opini pengguna Maskapai Penerbangan di Indonesia pada Jejaring Sosial Twitter menggunakan Metode *K-Nearest Neighbor*, penelitian ini menghasilkan akurasi sebesar 86,674% untuk sentimen positif dan 93,345% untuk sentimen negatif, namun karena data yang diuji merupakan media bebas dan tidak terikat, maka data yang diuji pun menjadi kurang konsisten. Sehingga diperlukan koleksi kamus lanjutan untuk meningkatkan akurasi dan konsistensi (Suprapto, 2017).

Penelitian tentang klasifikasi teks lainnya juga pernah dilakukan oleh Hariri, dkk. dengan judul *Learning Vector Quantization* untuk Klasifikasi Abstrak Tesis. Penelitian ini menghasilkan kesimpulan akurasi yang dihasilkan oleh metode LVQ dalam mengklasifikasi dokumen abstrak tugas akhir mahasiswa Jurusan Teknik Informatika Universitas Trunojoyo Madura mencapai 90% untuk keseluruhan data. Dengan berhasil mengenali abstrak tugas akhir bidang minat SI RPL dan CAI dengan akurasi 100%. Namun, untuk bidang minat multimedia hanya menghasilkan akurasi sebesar 70%. Akurasi terbaik sebesar 90% didapatkan dengan kondisi: Reduksi dimensi dengan parameter 20%, perbandingan data testing dan training sebesar 75% data testing dan 25% data training, learning rate antara 0,1-0,5.

BAB III
DESAIN DAN IMPLEMENTASI SISTEM

3.1. Desain Sistem

Penelitian ini memiliki dua tahap inti, yakni tahap membangun model terbaik menggunakan kombinasi algoritma *Naïve Bayes* dan KNN, dimana kedua tahap sebelumnya didukung oleh beberapa metode lainnya, diantaranya: *preprocessing*, TF-IDF, WIDF, *Cosine Similarity*. Adapun langkah-langkah utama pada penelitian ini meliputi pengumpulan data, distribusi kelas *dataset*, *input* teks abstrak uji, *Text Preprocessing*, klasifikasi oleh *Naïve Bayes* dan KNN, kemudian menghasilkan *output* berupa kelas dari teks abstrak yang diuji. Desain sistem pada penelitian ini yang dimuat pada Gambar 3.1



Gambar 3.1. Desain Sistem

3.1.1. *Dataset*

Sumber data yang digunakan pada penelitian ini adalah data primer yang dikumpulkan sendiri di *Website Repository* Unair (http://repository.unair.ac.id/view/subjects/HE.html) yang diakses pada tanggal 10 Juni 2019, berupa karya ilmiah abstrak skripsi mahasiswa Universitas Airlangga dengan konsentrasi subjek "*Transportation and Communication*". Data yang

diambil berupa kategori skripsi dan teks abstrak skripsi tiap kategori. Tiap kategori diambil sebanyak 20 teks abstrak skripsi untuk meningkatkan akurasi dan mengurangi ambiguitas karena jumlah tiap skripsi kategori yang tidak sama. Struktur *dataset* yang diambil dapat dilihat pada Gambar 3.2.

Gambar 3.2. Struktur *Dataset*

Ada tiga kategori yang ditentukan berdasarkan data yang telah terdefinisi pada *website repository* Unair. Daftar lengkapnya dapat dilihat pada Tabel 3.1.

Tabel 3.1. Kelas-kelas klasifikasi

ID	Kelas
HE1-9990	HE1-9990 *Transportation and communications*
HE323-328	HE323-328 *Transportation geography. Trade routes*
HE5601-5725	HE5601-5725 *Automotive transportation Including trucking, bus lines, and taxicab service*

3.1.2. *Text Preprocessing*

Sebelum diklasifikasikan, setiap teks abstrak akan melalui tahap *Text Preprocessing*. *Text processing* bertujuan untuk mengubah data dari tidak terstruktur menjadi terstruktur, serta mengurangi ambiguitas kata yang diproses.

Adapun diantara tahapan *Text Preprocessing* pada penelitian ini adalah *Case Folding*, *Tokenizing*, *Stopword Removal* dan *Stemming*. Alur proses *preprocessing* dapat dilihat pada Gambar 3.3.

Gambar 3.3. *Block Diagram Text Preprocessing*

1) *Case Folding*

Case Folding adalah proses mengonversi karakter menjadi bentuk standar, dimana semua karakter akan dikonversi ke huruf kecil. Misalnya, kata "Klasifikasi" berada di awal kalimat, ia akan memiliki huruf kapital sebagai karakter pertama. Bila dibandingkan dengan kata yang sama tetapi tidak berada di awal kalimat, tidak menggunakan huruf kapital, sistem akan dianggap sebagai *string* berbeda karena *string* memiliki sifat *case sensitive*. Maka perlu dilakukan *Case Folding* sehingga semua kata yang nampak sama tetapi memiliki struktur karakter yang berbeda akan dianggap sama lagi karena bentuknya telah diubah ke bentuk yang serupa.

Hampir semua bahasa pemrograman memiliki *String Function* untuk mengubah seluruh teks mentah menjadi huruf kecil. Contohnya, pada bahasa pemrograman Javascript dapat menggunakan *function String.toLowerCase()*. Contoh hasil keluaran *Case Folding* dapat dilihat pada Gambar 3.4.

[Sebelum *Case Folding*] [Hasil *Case Folding*]

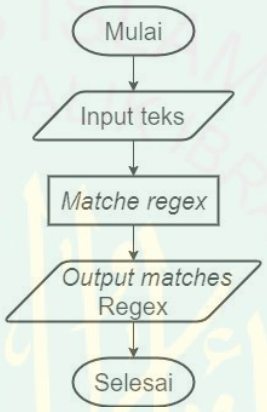
Gambar 3.4. Contoh keluaran *Case Folding*

2) *Tokenizing*

Tokenizing adalah proses memecah kalimat menjadi satu kata yang independen. Topik yang terbentuk dari banyak kalimat akan dipecah menjadi kumpulan kata-kata tunggal dan menghilangkan tanda baca dan spasi. Proses

ini dilakukan untuk mempermudah menghitung frekuensi kemunculan kata selama proses klasifikasi data.

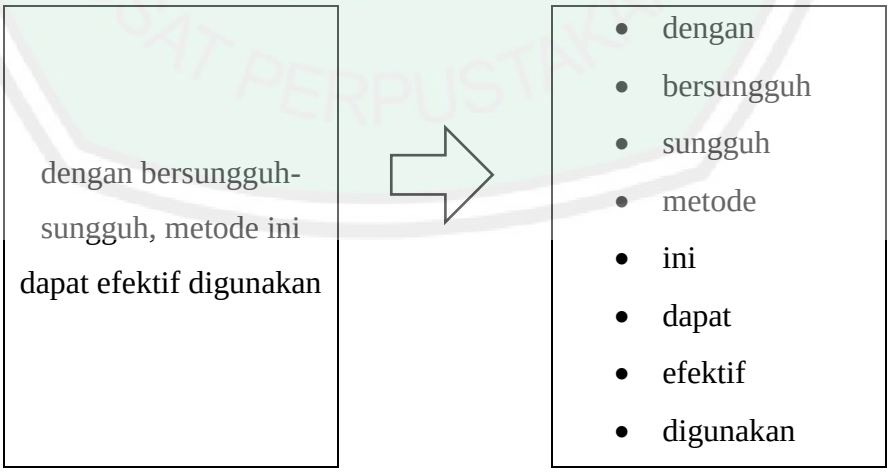
Untuk melakukan *Tokenizing*, dapat menggunakan *Function String Replace* dengan bantuan *Regular* Expression untuk menghapus simbol dan tanda baca menjadi *whitespace* (spasi) kemudian menggunakan mengonversi kumpulan *string* yang dipisahkan spasi menjadi *array* kumpulan kata. Proses lengkapnya dapat dilihat *flowchart* pada Gambar 3.5. Adapun pola RegEx yang digunakan dapat dilihat pada Gambar 3.6 beserta contoh pada Gambar 3.7.



Gambar 3.5. *Flowchart Tokenizing*

```
([A-Z])(\.[A-Z])+\.?|\w+(-\w+)*|\$?\d+(\.\d+)?%?|\.\.\.|[][.,;'"?():-_`]
```

Gambar 3.6. Pola *Regular Expression*

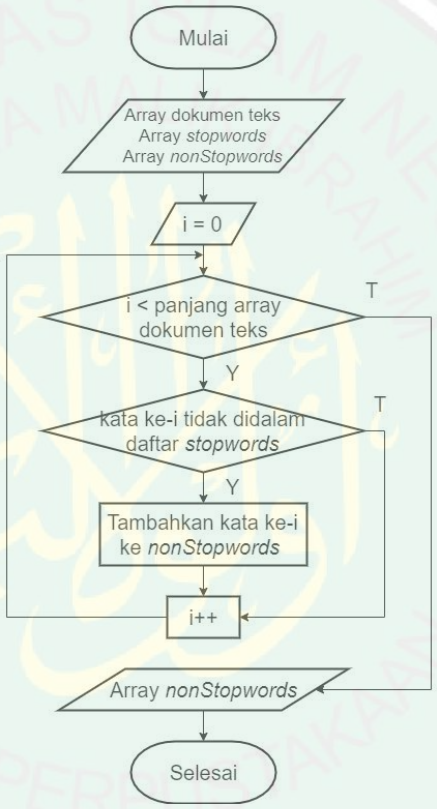


[Sebelum *Tokenizing*] [Hasil *Tokenizing*]

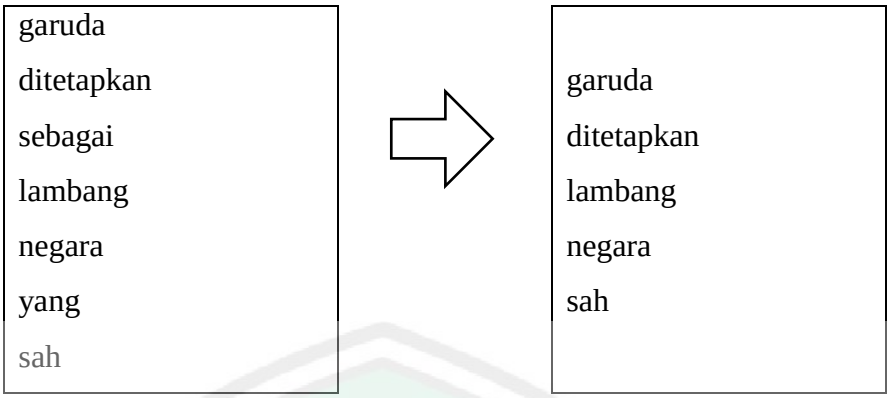
Gambar 3.7. Contoh keluaran *Tokenizing*

3) *Stopwords Removal*

Stopwords Removal adalah proses untuk membuang kata-kata yang dianggap tidak memiliki makna. Biasanya berupa kata sambung, atau kata-kata umum yang tidak bernilai. Kumpulan *stopwords* diambil dari penelitian Tala F.Z (2013) yang berjudul "*A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*". *Flowchart* proses penghapusan *stopwords* dijelaskan pada Gambar 3.8 dan *output* dari penghapusan stopwords ditunjukkan pada Gambar 3.9.



Gambar 3.8. *Flowchart Stopwords Removal*

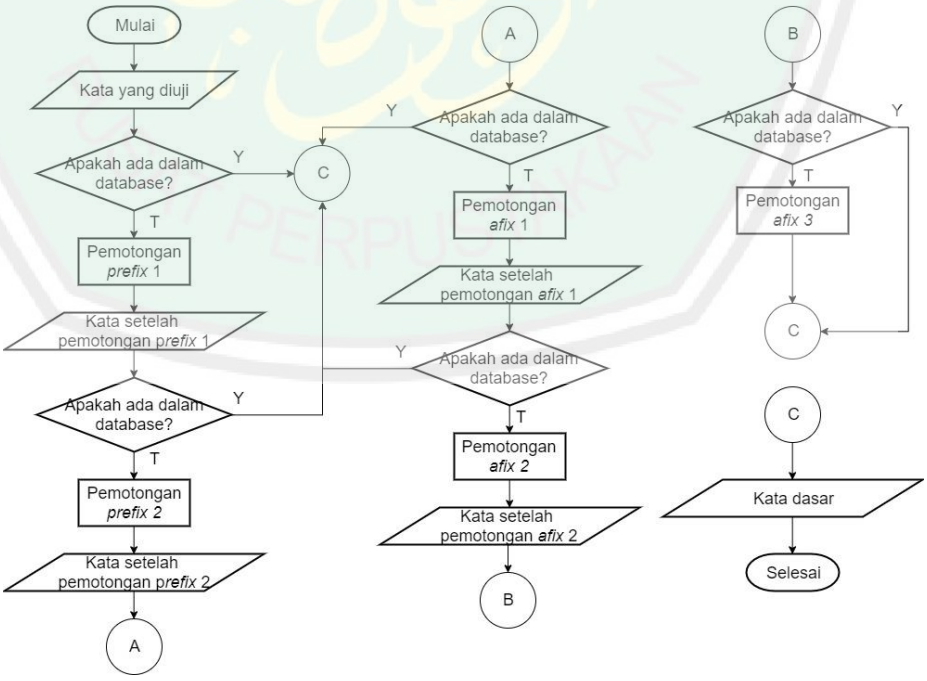


[Sebelum *stopwords removal*] [Hasil *stopwords removal*]

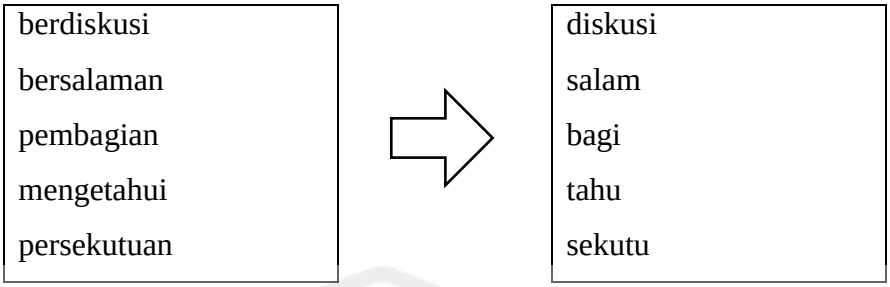
Gambar 3.9. Contoh keluaran *Stopwords Removal*

4) *Stemming*

Stemming adalah proses untuk menstandarisasi kata baku yang memiliki imbuhan menjadi kata dasar. Sama dengan *Case Folding*, tujuan *Stemming* adalah untuk mengurangi ambiguitas dan kesalahan perhitungan *term frequent* karena perbedaan kuantitas kata. Metode *Stemming* yang digunakan adalah Algoritma Nazief & Adriani. Proses detil *Stemming* dapat dilihat pada Gambar 3.10 dan contoh keluaran *Stemming* dapat dilihat pada Gambar 3.11.



Gambar 3.10. *Flowchart Stemming*

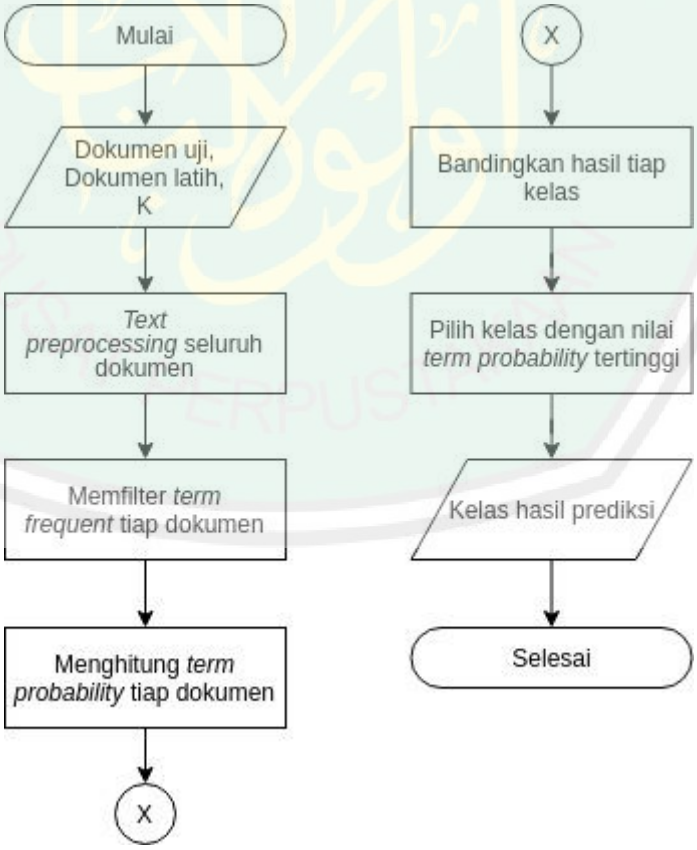


[Sebelum *Stemming*] [Hasil *Stemming*]

Gambar 3.11. Contoh keluaran *Stemming*

3.1.3. *Naïve Bayes Classifier*

Pada Metode *Naïve Bayes Classifier* dilakukan proses pengambilan *term frequent* pada setiap dokumen dengan mengasumsikan setiap dokumen tidak memiliki rangkaian kata yang unik yang hanya terjadi pada kelas tertentu. *Flowchart* proses klasifikasi *Naïve Bayes Classifier* dapat dilihat pada Gambar 3.12.



Gambar 3.12. *Flowchart* proses Klasifikasi *Naïve Bayes Classifier*

Langkah pertama adalah melakukan pemfilteran *term* unik pada seluruh dokumen. Kemudian, setiap dokumen dihitung *term frequent*-nya untuk mengetahui jumlah kata yang muncul. Pada Tabel 3.2 berikut adalah contoh dokumen latih beserta kelasnya.

Tabel 3.2. Contoh dokumen latih

Dokumen	Teks	Kelas
D1	saya suka film	P
D2	saya benci film	N
D3	film hebat film bagus akting buruk	P
D4	akting buruk	N
D5	akting bagus film bagus	P

Dari data Tabel 3.2 dibuat sebuah model *Prior Probabilities* dengan mengacu pada persamaan:

$$P(c) = \frac{\mathrm{Nc}}{\mathrm{N}} \qquad \text{.....(3.1)}$$

Maka *Prior Probabilities* untuk Tabel 3.2 ditunjukan pada Tabel 3.3 dengan mengacu pada Nc adalah jumlah kelas yang muncul dan N adalah jumlah seluruh kelas. Kemudian, dilakukan pemfilteran kata unik dan dihitung *term frequent* pada setiap kelas sehingga ditunjukan pada Tabel 3.4.

Tabel 3.3. Contoh hasil perhitungan *Prior Probabilities* pada setiap kelas

Atribut kelas	**P(*class*)**
P	3/5
N	2/5

Tabel 3.4. Contoh *term* unik pada seluruh dokumen

Term	**Kelas P**	**Kelas N**
saya	1	1
suka	1	0
film	4	1

benci	0	1
hebat	1	0
bagus	2	0
akting	2	1
buruk	1	1
Total	**12**	**5**

Seperti diketahui pada Tabel 3.4, kata unik berjumlah 8. Kemudian, setelah mengetahui jumlah *term frequency*, dilakukan perhitungan *probabilities term* setiap kelas dengan persamaan sebagai berikut:

$$P(Wk|Class) = \frac{nk + 1}{n + |Vocabulary|} \quad \text{.....(3.2)}$$

Dimana W_k adalah *term* yang diuji, n_k adalah *term frequency* ditambah angka 1 untuk memenuhi variabel pada Metode *Laplace Correction*, *n* adalah jumlah seluruh *term* pada kelas yang diuji, dan |*Vocabulary*| adalah jumlah seluruh kata unik. Berikut adalah contoh perhitungan *probabilities* untuk *term* "akting" pada kelas P.

$$P(akting|P) = \frac{2 + 1}{12 + 8} = 0.15$$

Perhitungan lengkap *probabilities* semua *term* ditunjukan pada Tabel 3.5.

Tabel 3.5. Contoh perhitungan *probabilities* setiap *term*

Term	**Positif**	**Negatif**
saya	0.1	0.015
suka	0.1	0
film	0.25	0.015
benci	0.5	0.015
hebat	0.1	0
bagus	0.15	0
akting	0.15	0.015
buruk	0.1	0.015

Data *probabilities* tiap *term* pada Tabel 3.5 akan disimpan di *database* sebagai acuan proses pengujian klasifikasi. Selanjutnya, alur proses *testing* kurang lebih sama dengan *training*. Contoh dokumen uji ditunjukan pada Tabel 3.6.

Tabel 3.6. Contoh dokumen uji

Teks	Kelas
saya benci akting buruk	?

Perhitungan *probabilities* pada bagian $\prod_{1 \le k \le nd} P\ (t_k \mid c)$, persamaan tersebut dihitung terlebih dahulu untuk memudahkan kalkulasi selanjutnya. Untuk hasilnya terdapat Tabel 3.7.

Tabel 3.7. Contoh hasil nilai *probabilities* tiap *term* pada dokumen uji

Term	TF dokumen uji	TF Kelas P	TF Kelas N
Saya	1	0.1	0.015
Benci	1	0.1	0.015
Akting	1	0.1	0.015
Buruk	1	0.1	0.015

Dari Tabel 3.7 didapatkan nilai *probabilities* setiap *term*. Untuk mendapatkan total nilai, *term probabilities* dikalikan dengan *term probabilities* yang lain pada kelas yang sama kemudian dikalikan lagi dengan *Prior Probabilities*-nya. Nilai yang paling besar diantara kelas tersebut adalah kelas pemenang. Contoh perhitungan total nilai klasifikasi pada kelas P:

$$V_j(\text{P}) = p(\text{P}) \cdot p(\text{saya}|\text{P}) \cdot p(\text{benci}|\text{P}) \cdot p(\text{akting}|\text{P}) \cdot p(\text{buruk}|\text{P})$$
$$= 0.6 \cdot 0.1 \cdot 0.1 \cdot 0.1 \cdot 0.1 = 6 \times 10^{-5}$$

Adapun hasil total nilai klasifikasi seluruh kelas ditunjukan pada Tabel 3.8.

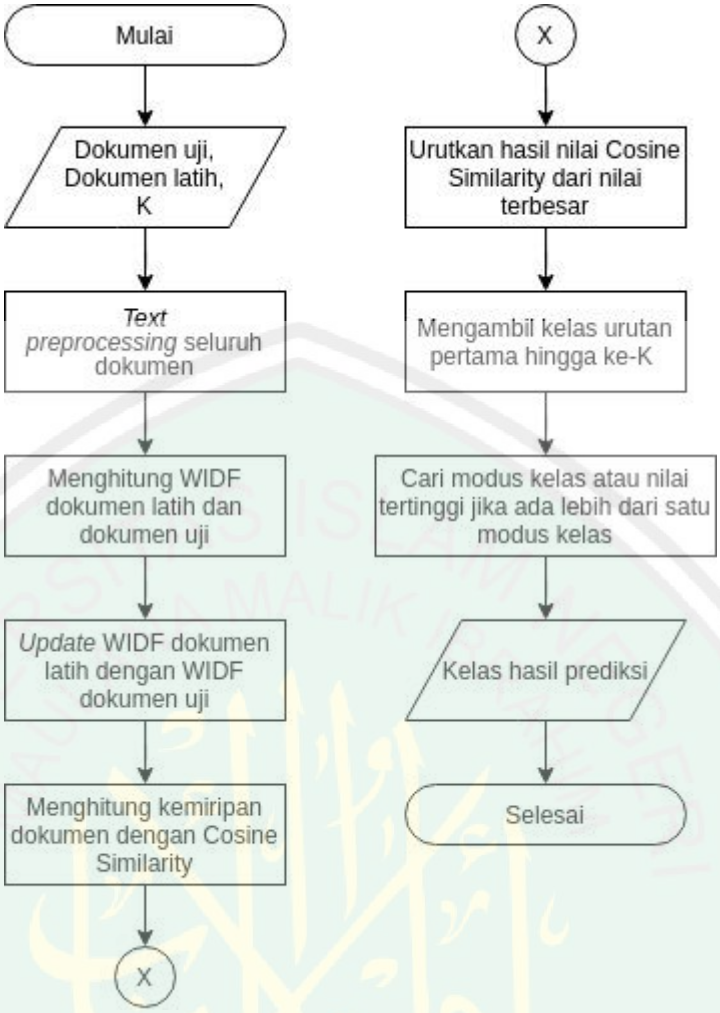
Tabel 3.8. Contoh total nilai klasifikasi

Kelas	Nilai
P	6×10^{-5}
N	2.02×10^{-8}

Dan hasil perhitungan pada Tabel 3.8 di atas diketahui bahwa kelas N memiliki nilai yang lebih besar dari kelas P. Oleh karena itu, dokumen uji termasuk kelas N.

3.1.4. *K-Nearest Neighbor*

Dalam proses klasifikasi dengan metode *K-Nearest Neighbor*, dilakukan pengenalan pola dengan menggunakan *dataset* abstrak yang telah diklasifikasikan sebelumnya sesuai dengan kelasnya. Kemudian, model *dataset* tersebut akan digunakan sebagai pembanding terhadap dokumen abstrak yang belum terklasifikasi. Modus kelas dokumen latih yang paling mirip dengan dokumen uji akan dipilih sebagai kelas pemenang. *Flowchart* proses klasifikasi KNN ditunjukan pada Gambar 3.13.



Gambar 3.13. *Flowchart* proses Klasifikasi KNN

Adapun tahap pertama proses klasifikasi pada metode ini adalah menentukan parameter K. Parameter K adalah jumlah dokumen yang akan diambil dari seluruh dokumen latih yang paling mirip dari dokumen uji. Pada penelitian ini, akan dilakukan iterasi untuk mendapatkan nilai K terbaik dengan akurasi terbaik.

Setelah menentukan nilai K, tahap selanjutnya adalah menghitung *similarity* (jarak) tiap dokumen dengan dokumen yang lain. Untuk itu, KNN membutuhkan bantuan metode lain, yakni *Cosine Similarity*. *Cosine Similarity* digunakan untuk mengubah teks dokumen abstrak yang luas menjadi sebuah nilai linier yang dapat dibandingkan dengan nilai linier yang lain. Untuk menghitung tingkat kemiripan, *Cosine Similarity* membutuhkan informasi data bobot *term frequent* tiap dokumen. Metode WIDF digunakan untuk mengumpulkan informasi tersebut. Metode WIDF

dicontohkan pada Tabel 3.9, dimana *d* (kolom) adalah dokumen abstrak dan *t* (baris) adalah *term* yang dicari dengan contoh 5 dokumen.

Tabel 3.9. Contoh koleksi data

	d_1	d_2	d_3	d_4	d_5
...	…	…	…	…	…
t_x	2	5	3	2	4
t_y	3	2	3	2	3
…	…	…	…	…	…

Angka yang ada pada Tabel 3.9 merupakan data *term frequent* (*t*) pada dokumen (*d*). Pada contoh kasus d_2, dapat dihitung nilai bobotnya sesuai dengan rumus (2.5) dengan rincian sebagai berikut:

$$WIDF\ (d2) = \frac{5}{2+5+3+2+4} = 0.3125$$

Bobot dari t_x dan d_2 dihitung dengan membagi jumlah t_x pada d_2 dengan jumlah keseluruhan t_x pada semua dokumen. Dengan kata lain, WIDF merupakan normalisasi *form frequency* dari semua dokumen. Dari perhitungan di atas, didapatkan hasil pembobotan pada Tabel 3.10.

Tabel 3.10. Contoh bobot WIDF dokumen latih

	d_1	d_2	d_3	d_4	d_5
...	…	…	…	…	…
t_x	0.125	0.3125	0.1875	0.125	0.25
t_y	0.2308	0.1538	0.2308	0.1538	0.2308
…	…	…	…	…	…

Berdasarkan data sesungguhnya, nilai yang terdapat pada Tabel 3.10 bisa saja bernilai 0 jika kata yang terdapat pada dokumen uji tidak ada dalam teks abstrak dokumen latih. Selanjutnya, nilai bobot tiap dokumen didapatkan dari jumlah seluruh bobot tiap *term* pada dokumen tersebut. Contoh perhitungan bobot pada

dokumen 2 dibawah ini dan pada tabel 3.11 ditunjukan keseluruhan nilai bobot dokumen beserta kelasnya.

$$W\ (d2) = 0.125 + 0.2308 = 0.02885$$

Tabel 3.11. Contoh bobot dokumen latih

Dokumen	**Bobot**	**Kelas**
D1	0.02885	HE1-9990 *Transportation and communications*
D2	0.4663	HE323-328 *Transportation geography. Trade routes*
D3	0.4183	HE1-9990 *Transportation and communications*
D4	0.2788	HE5601-5725 *Automotive transportation Including trucking, bus lines, and taxicab service*
D5	0.4808	HE1-9990 *Transportation and communications*

Selanjutnya, dihitung nilai bobot *term frequent* dokumen uji yang dimasukan. Dokumen uji melalui tahapan yang sama dengan dokumen latih sehingga didapatkan bobot WIDF pada Tabel 3.12.

Tabel 3.12. Contoh bobot WIDF dokumen uji

	W(*x,t*)
…	…
t_x	0
t_y	0.13
…	…

Dari Tabel 3.12, dapat diketahui bahwa nilai bobot dokumen uji adalah 0.13. Selanjutnya, bobot masing-masing *term* dalam dokumen latih dikali dengan bobot masing-masing *term* dalam dokumen uji sehingga didapatkan nilai bobot dokumen. Contoh perhitungan untuk mendapatkan bobot d_1 adalah sebagai berikut:

$$W\ (d1, t) = (0.125 \cdot 0) + (0.2308 \cdot 0.13) = 0.03$$

Sehingga untuk hasil perhitungan seluruh dokumen ditunjukan pada Tabel 3.13.

Tabel 3.13. Contoh bobot baru dokumen

	W(*d1*,*t*)	**W(*d2*,*t*)**	**W(*d3*,*t*)**	**W(*d4*,*t*)**	**W(*d5*,*t*)**
	0	0	0	0	0
	0.03	0.01	0.03	0.01	0.03
Total	**0.03**	**0.01**	**0.03**	**0.01**	**0.03**

Bobot pada Tabel 3.13 digunakan untuk menghitung *Cosine Similarity* dengan rumus (2.6) untuk mendapatkan angka kemiripan dokumen latih dengan dokumen uji. Adapun contoh perhitungan pada *d*2 adalah sebagai berikut:

$$CosSim\ (x, d2) = \frac{0.01}{\sqrt{0.13} \times \sqrt{0.4663}} = 0.04$$

Nilai bobot baru *d*2 dibagi dengan hasil dari akar kuadrat bobot dokumen uji kali akar kuadrat bobot dokumen *d*2. Kemudian, nilai kesamaan dokumen latih dengan dokumen uji yang ditunjukan pada Tabel 3.14.

Tabel 3.14. Contoh nilai kemiripan dokumen

Dataset	**Nilai Kemiripan**
*d*1	0.49
*d*2	0.04
*d*3	0.12
*d*4	0.05
*d*5	0.12

Langkah berikutnya adalah pengurutan hasil nilai kemiripan dokumen pada Tabel 3.14, mulai dari yang paling tinggi, hingga yang paling rendah seperti yang ditunjukan pada Tabel 3.15. Kemudian, diambil dokumen teratas berdasarkan jumlah K. Misalnya pada contoh ini K = 3, maka 3 dokumen teratas ditunjukan pada Tabel 3.16.

Tabel 3.15. Contoh pengurutan nilai kemiripan dokumen

Dataset	**Nilai Kemiripan Data**
*d*1	0.49
*d*3	0.12

*d*5	0.12
*d*4	0.05
*d*2	0.04

Tabel 3.16. Contoh hasil pembatasan ketetanggaan

Nomor	***Dataset***	**Nilai Kemiripan Data**	**Kelas**
1	*d*1	0.49	HE1-9990 Transportation and communications
2	*d*3	0.12	HE1-9990 Transportation and communications
3	*d*5	0.12	HE1-9990 Transportation and communications

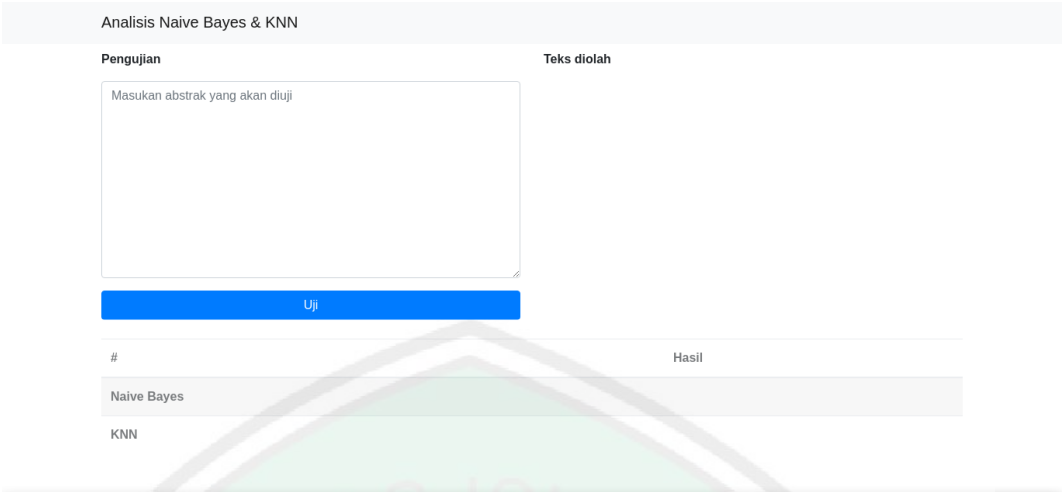
Dengan parameter k = 3, dapat dilihat bahwa hasil menunjukan semua kelas dokumen memiliki kelas yang sama, yaitu HE1-9990 *Transportation and communications* yang sekaligus menjadi hasil prediksi metode klasifikasi KNN. Namun pada penggunaan pada data yang sesungguhnya, tidak selalu menghasilkan kelas yang seragam. Jika hal itu terjadi, maka dipilih kelas dengan modus terbanyak.

3.2. Implementasi Sistem

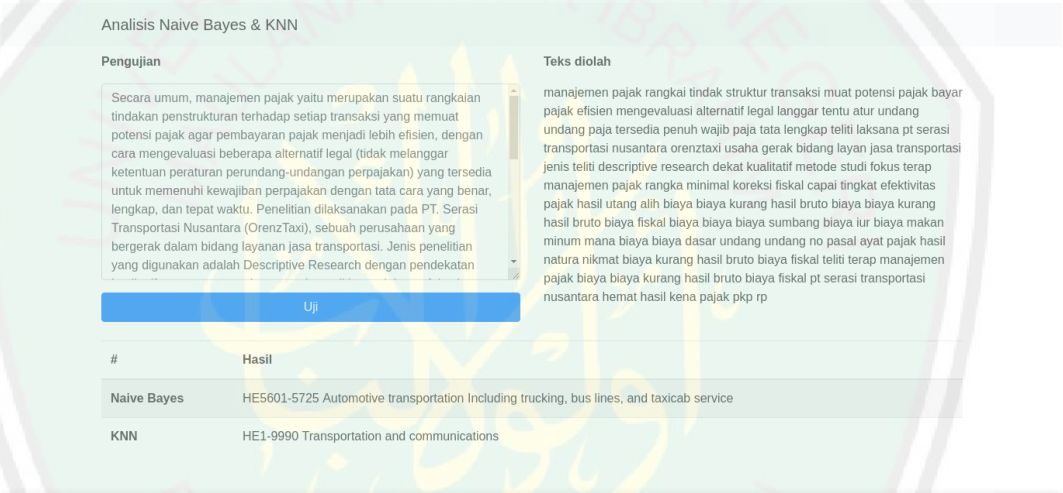
Kedua metode yang diteliti pada penelitian ini diimplementasikan pada aplikasi ber-*platform* Web dengan menggunakan bahasa Javascript dengan V8-*Engine* Node.js dan menggunakan MySQL sebagai RDBMS.

3.2.1. Tampilan Aplikasi

Interface yang digunakan bersifat *generic template* sehingga satu tampilan dapat digunakan untuk berbagai kebutuhan. Tampilan aplikasi secara lengkap dijelaskan pada Gambar 3.14 dan 3.15 disertai dengan *source code* pada Gambar 3.16.



Gambar 3.14. Tampilan aplikasi sebelum dilakukan pengujian



Gambar 3.15. Tampilan aplikasi setelah dilakukan pengujian

```
<% include header %>
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container">
    <a class="navbar-brand" href="#">Analisis Naive Bayes &amp;amp; KNN</a>
  </div>
</nav>
<div class="container mt-2">
  <div class="row">
    <div class="col-md">
      <p class="font-weight-bold">Pengujian</p>
      <form action="/classify" method="POST">
        <textarea
          name="abstract"
          rows="10"
          class="form-control"
          placeholder="Masukan abstrak yang akan diuji"
          >
<%= original %></textarea
          >
          <div class="text-right mt-3">
            <button type="submit" class="d-block btn btn-primary">Uji</button>
          </div>
        </form>
      </div>
      <div class="col-md">
        <p class="font-weight-bold">Teks diolah</p>
        <p>
          <%= stemmed %>
        </p>
      </div>
    </div>
  </div>
  <div class="container mt-4">


    <table class="table table-striped">
      <thead>
        <tr>
          <th scope="col">#</th>
          <th scope="col">Hasil</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <th scope="row">Naive Bayes</th>
          <td><%= naiveBayes %></td>
        </tr>
        <tr>
          <th scope="row">KNN</th>
          <td><%= knn %></td>
        </tr>
      </tbody>
    </table>
  </div>
  <% include footer %>
```

Gambar 3.16. *Source code* tampilan aplikasi

3.2.2. Proses *Preprocessing* Dokumen

Pada setiap dokumen yang akan diklasifikasikan, setiap teks abstrak akan harus terlebih dahulu dilakukan *preprocessing*. *Preprocessing* bertujuan untuk mengubah data dari tidak terstruktur menjadi terstruktur, serta mengurangi ambiguitas kata agar data tersebut lebih mudah diproses. Adapun diantara tahapan *Text Preprocessing* pada penelitian ini adalah *Case Folding*, *Tokenizing*, *Stopword Removal* dan *Stemming*.

1) *Case Folding*

Case Folding adalah proses mengonversi karakter menjadi bentuk standar, dimana semua karakter akan diubah ke huruf kecil (bukan kapital). Fungsi untuk melakukan *case folding* dapat dilihat pada Gambar 3.17.

```
const caseFolding = text => {
  return text.toString().toLowerCase()
}
```

Gambar 3.17. *Source code Case Folding*

2) *Tokenizing*

Tokenizing adalah proses memecah kalimat menjadi satu kata yang independen. Topik yang terbentuk dari banyak kalimat akan dipecah menjadi kumpulan kata-kata tunggal serta menghilangkan tanda baca dan spasi. Proses ini dilakukan untuk mempermudah menghitung frekuensi kemunculan kata selama proses klasifikasi data. Fungsi yang digunakan pada tahap ini dapat dilihat pada Gambar 3.18.

```
const tokenizing = (text) => {
  var sent = this.parseHtmlEnteties(text)
  sent = sent.toLowerCase()
  sent = sent.replace(/(www\.|https?|s?ftp)\S+/g, '')
  sent = sent.replace(/\S+@\S+/g, '')
  sent = sent.replace(/(@|#)\S+/g, '')
  sent = sent.replace(/&.*;/g, '')
  sent = sent.replace(/[^a-z\s]/g, ' ')
  sent = sent.replace(/\s+/g, ' ')
  sent = sent
    .trim()
    .replace(/ /g, '')
    .replace(/<[^\/>][^>]*><\/[^>]+>/g, '')
  return sent.split(/\s+/)
}
```

Gambar 3.18. *Source code Tokenizing*

3) *Stopword Removal*

Stopwords removal adalah proses untuk membuang kata-kata yang dianggap tidak memiliki makna. Biasanya berupa kata sambung, atau kata-kata umum yang tidak bernilai. Fungsi pada tahap ini dapat dilihat pada Gambar 3.19.

```
const stopwordsRemoval = (tokenizedText) => {
  const stopwords = fs
    .readFileSync(__dirname + '/resources/stopword_list_tala.txt', 'utf-8')
    .split(/\s/gm)
  return tokenizedText.filter((word) => !stopwords.includes(word))
}
```

Gambar 3.19. *Source code Stopword Removal*

4) *Stemming*

Stemming adalah proses untuk menstandarisasi kata baku yang memiliki imbuhan menjadi kata dasar. Metode *Stemming* yang digunakan pada penelitian ini adalah Algoritma Nazief & Adriani. Fungsi untuk tahap ini dapat dilihat pada Gambar 3.20.

```
this.stem = function(word){
    word = word.toLowerCase();

    var rootFound    = false
    var originalWord = word
    var particle,possesive,suffix

    if (word.length < 3) {
        return word
    }
    if (find(word)) {
        return word
    }
    if (/^(be.+lah|be.+an|me.+i|di.+i|pe.+i|ter.+i)$/.test(word)) { //ok
        // Remove prefix
        funcret = removePrefixes(word)
        rootFound = funcret[0]
        word = funcret[1]
        if (rootFound) {
            return word
        }
        // Remove particle
        funcret = removeParticle(word)
        particle = funcret[0]
        word = funcret[1]
        if (find(word)) {
            return word
        }

        // Remove possesive
        funcret = removePossesive(word)
        possesive = funcret[0]
```

```
        word = funcret[1]
        if (rootFound) {
            return word
        }
        // Remove particle
        funcret = removeParticle(word)
        particle = funcret[0]
        word = funcret[1]
        if (find(word)) {
            return word
        }

        // Remove possesive
        funcret = removePossesive(word)
        possesive = funcret[0]
        word = funcret[1]
        if (find(word)) {
            return word
        }

        // Remove suffix
        funcret = removeSuffix(word)
        suffix = funcret[0]
        word = funcret[1]
        if (find(word)) {
            return word
        }
    } else {

        // Remove particle
        funcret = removeParticle(word)
        particle = funcret[0]
        word = funcret[1]
        if (find(word)) {
            return word
        }

        // Remove possesive
        funcret = removePossesive(word)
        possesive = funcret[0]
        word = funcret[1]
        if (find(word)) {
            return word
        }

        // Remove suffix
        funcret = removeSuffix(word)
        suffix = funcret[0]
        word = funcret[1]
        if (find(word)) {
            return word
        }
        // Remove prefix
        funcret = removePrefixes(word)
        rootFound = funcret[0]
        word = funcret[1]
        if (rootFound) {
            return word
        }
    }
        // If no root found, do loopPengembalianAkhiran
        removedSuffixes = ["", suffix, possesive, particle]
        if (suffix == "kan") {
            removedSuffixes = ["", "k", "an", possesive, particle]
        }
        funcret = loopPengembalianAkhiran(originalWord, removedSuffixes)
        rootFound = funcret[0]
        word = funcret[1]
        if (rootFound) {
            return word
        }

        // When EVERYTHING failed, return original word
        return originalWord
    }
```

Gambar 3.20. *Source code Stemming*

3.2.3. Pengolahan Data Latih

Baik *Naïve Bayes* maupun KNN merupakan metode *supervised learning* yang berarti memerlukan data latih sebagai acuan. Untuk itu, seluruh dokumen latih di-*Text Preprocessing*-kan kemudian disimpan dalam *database* supaya proses

klasifikasi lebih cepat dan penghitungan waktu proses hanya fokus untuk klasifikasi saja. Berikut *source code* untuk mengolah data latih ditampilkan pada Gambar 3.21.

```
const preprocessAllDocs = async (req, res) => {
  const [data] = await connection.query('SELECT id, text FROM abstracts')
  const processed = data.map((el) => {
    const processed = textPreprocessing(el.text)
    return { ...el, processed }
  })
  await connection.query('TRUNCATE processed')
  await connection.query(
    `INSERT INTO processed(abstract_id, text) VALUES ${'(?,?),'.repeat(
      processed.length - 1
    )} (?,?)`,
    processed.reduce((prev, { processed, id }) => {
      return prev.concat([id, processed])
    }, [])
  )
}
```

Gambar 3.21. *Source code* untuk mengolah data latih

3.2.4. Implementasi Metode *Naïve Bayes*

Langkah pertama adalah melakukan pemfilteran *term* unik pada seluruh dokumen. Kemudian, setiap dokumen dihitung *term frequent*-nya untuk mengetahui jumlah kata yang muncul. Fungsi untuk menghitung *term frequent* dapat dilihat pada Gambar 3.22.

```
const wordEachSubject = {}
const trainingDocs = getTrainingDocs()

// populate each document and class into variable
trainingDocs.forEach((el) => {
  const text = el.text.split(/\s/gm)
  wordEachSubject[el.className] = {
    text,
    length: text.length,
  }
})

// collect term frequent of document testing
const valuePerSubject = Object.keys(wordEachSubject).map
  const subjWord = {}
  wordEachSubject[className].text.forEach((el) => {
    if (!subjWord[className]) subjWord[className] = 0
    subjWord[className] += docTestProcessed.filter(
      (inner) => el === inner
    ).length
  })
```

Gambar 3.22. *Source code* untuk memperoleh *Term Frequent*

Dari data TF, selanjutnya dilakukan perhitungan untuk mendapatkan *term probabilities*. Kemudian, di-akumulasi-kan dengan mengalikan seluruh *term*

probabilities per kelas. Karena jumlah dokumen setiap kelas sama, maka *Prior Probabilities* bisa diabaikan karena selalu memiliki nilai yang sama. Fungsi untuk menghitung *term probabilities* dapat dilihat pada Gambar 3.23.

```
// multiply all factors
value: Object.values(subjWord).reduce((prev, curr) => {
  const calc =
    (curr + 1) /
    (wordEachSubject[className].length + allUniqueString.length)

  return calc * prev
}, 1),
```

Gambar 3.23. *Source code* untuk menghitung *Term Probabilities* per kelas

Dari hasil penghitungan diatas, kemudian dibandingkan hasil *score*-nya tiap kelas. Kelas pemenang adalah kelas dengan *score* tertinggi. Fungsi untuk memperoleh kelas pemenang ditampilkan pada Gambar 3.24.

```
const classRes = valuePerSubject.reduce(
  (prev, el) => ({ ...prev, ...el }),
  {}
)

// sort to the best score
return Object.keys(classRes)
  .map((el) => ({ class: el, value: classRes[el] }))
  .sort((a, b) => b.value - a.value)[0].class
```

Gambar 3.24. *Source code* untuk memperoleh hasil kelas pemenang

3.2.5. Implementasi Metode *K-Nearest Neigbor*

Langkah pertama yang dilakukan adalah memfilter seluruh kata unik dari data training. Kemudian, dilakukan pembobotan untuk menghitung *term frequent* pada tiap kelas dilanjutkan dengan menghitung WIDF dokumen latih. Fungsi untuk langkah ini ditampilkan pada Gambar 3.25.

```
// get all unique words from training docs
const allUniqueString = await getAllUniqueWords()

const allAbstracts = {}
const trainingDocs = getTrainingDocs()

// populate and map training docs into variable
trainingDocs.forEach((el) => {
  const text = el.text.split(/\s/gm)
  allAbstracts[el.id] = {
    text,
    length: text.length,
    className: el.className,
  }
})

// populate initial weight (bobot)
const weight = {}
Object.keys(allAbstracts).forEach((abstractId) => {
  // get weight from specified string from unique string
  allUniqueString.forEach((string) => {
    if (!weight[abstractId]) weight[abstractId] = {}
    weight[abstractId][string] = allAbstracts[abstractId].text.filter(
      (inner) => inner === string
    ).length
    // weight = raw tf from all unique words each train docs
  })
})
```

Gambar 3.25. *Source code* penghitungan TF dan WIDF dokumen latih

Setelah menghitung bobot WIDF dokumen latih, selanjutnya dilakukan penghitungan bobot WIDF. Fungsi ini dapat dilihat pada Gambar 3.26.

```
// calculating document testing's weight
const testDocWeight = {}
allUniqueString.forEach((string) => {
  // store weight value of specified word in result var
  const result = {}
  Object.keys(weight).forEach((abstractId) => {
    result[abstractId] = weight[abstractId][string]
  })

  // iterate each doc to find weight
  Object.keys(result).forEach((abstractId) => {
    const divider = Object.keys(result).reduce(
      (prev, curr) => prev + result[curr],
      0
    )
    weight[abstractId][string] =
      Number(result[abstractId]) / Number(divider)
    // weight = weight(bobot) each word

    // init weight testing docs
    const tf = docTestProcessed.filter((inner) => inner === string).length
    testDocWeight[string] = tf / (Number(divider) + tf)
    // testDocWeight = weight document testing
  })
})
```

Gambar 3.26. *Source code* penghitungan TF dan WIDF dokumen uji

Langkah selanjutnya adalah pembobotan kemiripan dokumen dengan metode *Cosine Similarity*. Fungsi yang menjalankan metode *Cosine Similarity* dapat dilihat pada Gambar 3.27.

```
// collect weight before update
const totalDocWeightBeforeUpdate = {}
Object.keys(weight).forEach((abstractId) => {
  totalDocWeightBeforeUpdate[abstractId] = Object.values(
    weight[abstractId]
  ).reduce((prev, curr) => prev + curr, 0)
})

// multiply traindoc weight with testdoc
const trainDocAfterUpdate = {}
Object.keys(weight).forEach((abstractId) => {
  trainDocAfterUpdate[abstractId] = {}
  Object.keys(weight[abstractId]).forEach((string) => {
    trainDocAfterUpdate[abstractId][string] =
      weight[abstractId][string] * testDocWeight[string]
  })
})

const totalDocAfterUpdate = {}
Object.keys(trainDocAfterUpdate).forEach((className) => {
  totalDocAfterUpdate[className] = Object.values(
    trainDocAfterUpdate[className]
  ).reduce((prev, curr) => prev + curr, 0)
})

const totalTestDocWeight = Object.values(testDocWeight).reduce(
  (prev, curr) => prev + curr,
  0
)

Object.keys(weight).forEach((abstractId) => {
  // consine similarity formula
  weight[abstractId] =
    totalDocAfterUpdate[abstractId] /
    (Math.sqrt(totalTestDocWeight) *
      Math.sqrt(totalDocWeightBeforeUpdate[abstractId]))
})
```

Gambar 3.27. *Source code Cosine Similarity*

Setelah bobot untuk perhitungan *cosine similarity* telah didapatkan, langkah berikutnya adalah mengurutkan hasil perhitungan kemiripan dokumen mulai dari yang paling inggi, hingga yang paling rendah. Kemudian, diambil dokumen teratas berdasarkan jumlah K yang telah ditentukan di awal. Dari dokumen-dokumen tersebut, dipilih dokumen dengan kelas yang frekuensi memiliki kemunculan paling tinggi atau jika jumlah kemunculan sama, maka diambil dengan *score* tertinggi. Fungsi pengambilan hasil dokumen berdasarkan K dan pemilihan kelas pemenang secara berturut-turut ditampilkan pada Gambar 3.28 dan 3.29.

```
// pick the top K result score
const resClassList = Object.keys(weight)
  .sort((a, b) => weight[b] - weight[a])
  .map((el) => ({
    class: allAbstracts[el].className.trim(),
    value: weight[el],
  }))
  .slice(0, KNN_K) // KNN_K is respective of K
```

Gambar 3.28. *Source code* pengambilan dokumen teratas berdasarkan nilai K

```
// sort the most exist class
const modes = []
resClassList.forEach((el) => {
  const found = modes.find((inner) => inner.class === el.class)
  if (!found) modes.push({ class: el.class, value: 1 })
  else found.value++
})

// return the most exist class
return modes.sort((a, b) => b.value - a.value)[0].class.trim()
}
```

Gambar 3.29. *Source code* pemilihan kelas pemenang

BAB IV
UJI COBA DAN PEMBAHASAN

4.1. Skenario Uji Coba

Penelitian ini melakukan beberapa skenario uji coba untuk mendapatkan hasil akurasi yang bervariasi pada kedua metode. Dari seluruh *dataset* yang ada dibagi menjadi dua jenis, yakni dokumen latih dan dokumen uji dengan menggunakan prosentase yang bervariasi. Uji coba ini dilakukan pada setiap iterasi pada skenario yang dijelaskan pada Tabel 4.1.

Tabel 4.1. Skenario uji coba

Iterasi	Jumlah Dokumen latih	Jumlah Dokumen uji	Keterangan
1	100%	100%	Keseluruhan *dataset* akan menjadi dokumen latih dan dokumen uji
2	75%	25%	75% *dataset* akan menjadi dokumen latih dan 25% sisanya akan menjadi dokumen uji
3	50%	50%	50% *dataset* akan menjadi dokumen latih dan 50% sisanya akan menjadi dokumen uji
4	25%	75%	25% *dataset* akan menjadi dokumen latih dan 75% sisanya akan menjadi dokumen uji

Pada metode KNN, ada penambahan beberapa sub-iterasi uji coba untuk menentukan nilai K terbaik dalam mengklasifikasikan dokumen. Nilai K ditentukan didapat dari jumlah kelas yang terdapat pada *dataset*, kemudian dikalikan tiga kali berturut-turut. Selengkapnya dijelaskan pada Tabel 4.2.

Tabel 4.2. Sub-iterasi pengujian pada Metode KNN

Sub-iterasi	K	Keterangan
1	3	Jumlah K sama dengan jumlah kelas pada *dataset*
2	6	Jumlah K sama dengan jumlah kelas pada *dataset* dikali 2
3	9	Jumlah K sama dengan jumlah kelas pada *dataset* dikali 3

Setelah didapati hasil uji coba pada setiap iterasi dan sub-iterasi skenario, kemudian dihitung tingkat akurasi hasil uji pada masing-masing metode, dengan formula akurasi sebagai berikut.

$$\frac{TP + TN}{TP + TN + FP + FN} \times 100\% \quad \text{.....(4.1)}$$

Dimana:

TP = *True Positive*

TN = *True Negative*

FP = *False Positive*

FN = *False Negative*

Selain akurasi, juga dilakukan pengujian waktu yang dibutuhkan pada setiap proses klasifikasi. Kemudian, diambil rata-ratanya pada setiap metode per iterasi. Uji coba kecepatan proses dilakukan untuk mengetahui metode mana yang paling efisien dan menghemat waktu. Uji coba dilakukan menggunakan perangkat dengan spesifikasi yang dijelaskan pada Tabel 4.3.

Tabel 4.3. Spesifikasi perangkat pengujian

Brand	*Notebook* Acer E5-475G
Processor	Intel Core i5-7200U 2.5GHz
RAM	8GB DDR4
OS	Manjaro Linux 18.04 LTS

4.2. Hasil Uji Coba

4.2.1. Akurasi

Uji coba sistem pada penelitian ini dilakukan sesuai dengan skenario yang telah dipaparkan pada sub-bab 4.1. Hasil uji coba setiap skenario dikelompokkan dalam bentuk tabel. Hasil akurasi uji coba skenario pertama hingga skenario ke-empat dapat dilihat pada Tabel 4.4. Hasil uji secara detil secara berturut-turut dapat dilihat pada Lampiran 1 - 4.

Tabel 4.4. Hasil akurasi tiap skenario

Skenario	**Akurasi**			
	Naïve Bayes	**KNN**		
		K = 3	**K = 6**	**K = 9**
Skenario 1 (100%-100%)	92.92%	88.89%	82.83%	80.8%
Skenario 2 (75%-25%)	70.37%	74.07%	66.67%	77.78%
Skenario 3 (50%-50%)	71.72%	61.62%	69.7%	59.6%
Skenario 4 (25%-75%)	72.22%	65.28%	65.28%	68.06%

4.2.2. Kecepatan Proses

Uji coba kecepatan proses menggunakan skenario yang sama seperti pengujian akurasi. Uji coba ini dilakukan dengan menghitung kecepatan waktu saat proses klasifikasi dijalankan hingga sistem selesai memberikan hasil *output*. Rata-rata kecepatan metode *Naïve Bayes* dan KNN ditunjukkan pada Tabel 4.8.

Tabel 4.8. Rata-rata kecepatan proses *Naïve Bayes* dan KNN

Skenario	Rata-rata waktu pemrosesan (s)			
	Naïve Bayes	KNN		
		K = 3	K = 6	K = 9
Skenario 1	0,25	1,3	1,33	1,6
Skenario 2	0,21	1,04	1,19	1,05
Skenario 3	0,34	0,93	0,913	0,948
Skenario 4	0,445	0,93	0,917	1

4.3. Pembahasan

Setelah dilakukan uji coba terhadap Metode *Naïve Bayes* dan KNN, dapat disimpulkan secara garis besar bahwa kedua metode tersebut tidak ada yang jauh mengungguli diatas yang lain. Penjelasan untuk hasil uji coba akan dibahas secara mendetail setiap bagiannya pada paragraf selanjutnya.

Naïve Bayes dan KNN adalah metode *supervised learning*, yang berarti memerlukan data latih sebagai acuan uji coba. Jika dilihat dari parameter dan objek yang diuji, *Naïve Bayes* dan KNN sama-sama menggunakan *term frequent* yang sebelumnya harus melalui tahap *Text Preprocessing* supaya dapat diproses. Dengan memperhatikan *dataset* abstrak skripsi yang ditulis berdasarkan kaidah-kaidah penulisan ilmiah yang baku, maka tingkat *error* dan ambiguitas kata dapat diabaikan sehingga tidak perlu dilakukan *mapping* ulang.

Jika dilihat pada hasil uji coba, skenario pertama memberikan hasil yang paling tinggi diantara yang lain karena seluruh dokumen yang diujikan juga digunakan sebagai dokumen latih. Tentu hal ini tidak akan terjadi pada aplikasi sesungguhnya. Oleh karena itu, pada pembahasan kali ini difokuskan pada hasil uji coba skenario kedua hingga keempat yang menggunakan data latih dan data uji terpisah, tidak saling beririsan. Pada sisi performa, dapat dilihat Metode *Naïve Bayes* memiliki nilai akurasi yang konsisten pada ketiga skenario uji coba (selain skenario pertama) yakni kisaran 70.72% - 72.22%. Hal ini disebabkan oleh sifat "naif" dari *Naïve Bayes* itu sendiri. Setiap *term* dianggap berdiri sendiri, tidak terikat, dan tidak memiliki ketergantungan dengan *term* yang lain. Jika terdapat satu *term* pada dokumen uji tidak terdapat pada data latih, maka *term* tersebut diabaikan, atau bernilai 1 sesuai dengan kaidah *Laplace Correction*. Hal inilah yang menyebabkan banyaknya data latih tidak mempengaruhi nilai akurasi. Sedangkan pada Metode KNN, nilai akurasi lebih bervariasi pada tiap skenario dan tiap nilai K. Dapat dilihat bahwa uji coba dengan skenario kedua memberikan hasil paling optimal daripada skenario ketiga dan kempat, yakni nilai paling rendah 66.67% dan dua yang lain secara berturut-turut 74.97% dan 77.78%. Hal ini karena KNN memiliki sifat yang berlawanan dengan *Naïve Bayes*. KNN menilai sebuah dokumen berdasarkan bobot kemiripan dengan dokumen lain. Sebuah kalimat utuh tidak dapat dibandingkan dengan kalimat lain yang memiliki variasi kata yang sama. Jika pada *Naïve Bayes*

pemilihan kelas pemenang langsung dipilih berdasarkan nilai tertinggi hasil klasifikasi, KNN memilih kelas pemenang berdasarkan modus kelas dari nilai K. Hal inilah yang membuat banyaknya data latih menjadi penting pada KNN. Karena semakin banyak data latih, semakin banyak pula perbendaharaan dokumen yang dimiliki. Sehingga kelas dengan bobot kemiripan tertinggi berasal dari kelas yang sebenarnya. Hal ini pula yang menyebabkan waktu proses yang dibutuhkan KNN secara signifikan lebih lama (hampir dua kali lipat) dari *Naïve Bayes*. Masalah yang muncul dari Metode KNN adalah menentukan nilai K. Jika nilai K yang diberikan tepat, maka hasil klasifikasi akan memberikan akurasi terbaik yang lebih tinggi dari nilai akurasi *Naïve Bayes*. Sedangkan jika nilai K kurang tepat, maka akan memberikan hasil yang kurang, dibawah nilai akurasi *Naïve Bayes*.

Kemudian, setelah mengetahui kekurangan dan kelebihan masing-masing, pemilihan metode yang tepat yang sesuai dengan kondisi dan kebutuhan menjadi penting untuk dilakukan. Dalam beberapa ayat Al-Qur'an sendiri ada anjuran agar (kaum muslimin) mendahulukan *mashlahat* yang lebih besar atau mendahulukan *mafsadah* yang lebih kecil serta ada larangan dari (melakukan) sesuatu yang *mafsadah* (keburukan)-nya lebih dominan dari pada *mashlahat*-nya. Allah berfirman dalam QS. Al-Baqarah/2:42, yang berbunyi:

وَلَا تَلْبِسُوا۟ ٱلْحَقَّ بِٱلْبَـٰطِلِ وَتَكْتُمُوا۟ ٱلْحَقَّ وَأَنتُمْ تَعْلَمُونَ

Artinya: "Janganlah kalian campur-adukkan antara kebenaran dan kebatilan, dan kalian sembunyikan yang benar padahal kamu mengetahuinya".

Zubdatut Tafsir Min Fathil Qadir / Syaikh Dr. Muhammad Sulaiman Al Asyqar, mudarris tafsir Universitas Islam Madinah mengatakan bahwa QS. Al-Baqarah/2:42 memiliki tafsir dua poin utama sebagai berikut.

1. Ada dua jenis manusia yang berhubungan dengan ayat ini, yakni orang yang mengamalkan apa-apa yang diperintahkan oleh ayat dan mereka adalah ahli ilmu. Yaitu mereka dari kalangan ulama dan da'i-da'i penyebar da'wah. Yang kedua adalah orang-orang yang menutupi kebenaran dengan kebathilan. Yaitu mereka tidak membedakan antara satu perkara dengan perkara lainnya padahal mereka mengerti tentang itu. Maka mereka itu adalah da'i-da'i yang membawa kepada neraka jahannam. Karena sesungguhnya manusia tidak akan mampu

berdiri diatas agamanya tanpa peran ulama sebagai pembimbing, maka pilihlah untuk dirimu dari kedua golongan ini.

2. Para da'i yang menyesatkan umat mereka memiliki dua cara. Yang pertama yaitu mencampur adukkan yang *haq* dengan yang *bathil*, dalam hal ini diisyaratkan oleh firman Allah : وَلَا تَلْبِسُوا۟ ٱلْحَقَّ بِٱلْبَـٰطِلِ . Dan cara yang kedua adalah dengan memerangi kebenaran dan menyembunyikannya. Allah menjelaskan cara ini dengan ayat-Nya : وَتَكْتُمُوا۟ ٱلْحَقَّ .

Dari tafsir diatas dapat disimpulkan bahwa sebagai muslim yang baik, sudah sepatutnya penulis memberikan informasi yang benar dan jelas dari penelitian yang sudah dilakukan. Menyampaikan mana yang baik dan yang buruk. Oleh karena itu, penulis memberikan penjelasan secara gamblang mengenai kelebihan, kekurangan, dan faktor-faktor yang mempengaruhi hasil akurasi dalam konteks penelitian ini adalah perbandingan Metode *Naïve Bayes* dengan KNN. Sehingga pembaca yang akan menggunakan penelitian ini sebagai referensi tidak merasa rugi atau salah dalam mengambil acuan.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil pembahasan perbandingan akurasi dan performa Metode *Naïve Bayes* dengan *K-Nearest Neighbor* (KNN) secara umum diambil dari semua skenario pengujian, maka *Naïve Bayes* mengungguli KNN dengan rata-rata hasil akurasi dari semua skenario *Naïve Bayes* sebesar 76.81% dan KNN bernilai K 3, 6, 9 berturut-turut sebesar 72,47%, 71,12%, 71,56%. Namun, jika dengan mengabaikan hasil uji coba skenario pertama karena memiliki data yang beririsan antara dokumen latih dan dokumen uji dan jika hanya mengambil nilai akurasi tertinggi dari kedua metode, KNN memiliki akurasi lebih baik daripada *Naïve Bayes*. KNN memberikan akurasi (paling tinggi) sebesar 77.78% berbanding dengan akurasi *Naïve Bayes* (paling tinggi) sebesar 72.22%. Selisih ~5% lebih tinggi KNN daripada *Naïve Bayes*. Dengan catatan, selisih bisa bertambah berbanding lurus dengan jumlah perbendaharaan dokumen latih. Namun, perlu dilakukan uji coba berulang kali untuk menentukan nilai K yang paling baik.

Waktu proses yang dibutuhkan kedua metode terlampau cukup jauh. Yakni rata-rata hampir 3x lipat (0,31 detik berbanding 1,09 detik) lebih lama KNN daripada *Naïve Bayes*. Namun, untuk rentang waktu tersebut masih bisa dimaklumi dan diwajarkan.

5.2. Saran

Penulis menyadari bahwa masih banyak kekurangan dan kelemahan pada penelitian ini sehingga perlu dilakukan pengembangan lagi untuk mendapatkan hasil yang lebih akurat. Beberapa saran penulis untuk penelitian selanjutnya adalah sebagai berikut.

1. *Dataset* yang digunakan pada penelitian ini terbatas, akan lebih presisi jika menggunakan sumber data yang lebih banyak lagi.
2. Objek kategori skripsi pada penelitian ini memiliki tingkat kemiripan yang sangat tinggi, sehingga akurasi yang dihasilkan tidak terlalu tinggi. Hasil-

akurasi akan lebih tinggi jika objek penelitian memiliki tingkat kemiripan rendah.

3. Dapat ditambahkan dengan metode lain sebagai perbandingan supaya hasil penelitian menjadi semakin luas dan bervariasi.

DAFTAR PUSTAKA

Adriani, M. A. (2007). Stemming Indonesian : A Confi x-Stripping Approach. *ACM Transactions on Asian Language Information Processing (TALIP), 6(4),*, 1–33.

Agusta, L. (2009). *Perbandingan Algoritma Stemming Porter dengan Algoritma Nazief & Adriani untuk Stemming Dokumen Teks Bahasa Indonesia.* Bali: Konferensi Nasional Sistem dan Informatika.

Andini, S. (2013). *Klasifikasi Dokument Teks Menggunakan Algoritma Naïve Bayes dengan.* Jurnal Teknologi Informasi & Pendidikan.

Asian, J. (2007). Effective Techniques for Indonesian Text Retrieval. *School of Computer Science and Information Technology, RMIT University*.

Fieldman, R. &. (2007). *The Text Mining Handbook Advanced Approoaches in Analyzing Unstructured Data.* New York: Cambridge University Press.

Han, J. &. (2000). *Data Mining*: Concepts and Techniques. *Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.*

Hariri, U. A. (2015). *Learning Vector Quantization untuk Klasifikasi.* Yogyakarta: STMIK AMIKOM.

Hotho, A. N. (2005). A Brief Survey of *Text Mining*. *Computational Linguistics and Language Technology*.

IDC. (2018). *The Digitization of the World.* Framingham.

Jotheeswaran, J. &. (2013). Opinion Mining Using Decision Tree Based *Feature* Selection Through Manhattan Hierarchical Cluster Measure. *Journal of Theoretical and Applied Information Technology, Vol 58, No 1*, 72-80.

Khamis, H. K. (2014). Application of k-*Nearest Neighbor* Classification in Medical *Data Mining*. *International Journal of Information and Communication Technology Research, Vol. 4, No, 4*.

Krisdandi, N. H. (2013). Algoritma *K-Nearest Neighbor* dalam Klasifikasi Data Hasil Produksi Kelapa Sawit pada PT. Minamas Kecamatan Parindu. *Buletin Ilmiah Math. Stat. dan terapannya (Bimaster).*

Kurniawan, H. (2012). Sistem Penentuan Kualitas Air pada Depot Air Minum menggunakan Metode *K-Nearest Neighbor*. *Program Studi Teknik*

Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sultan syarif Kasim Riau, Pekanbaru.

Liu, B. (2007). Web *Data Mining. ACM Computing Classification, Springer Berlin Heidelberg. ISBN-10 3-540-37881-2*.

Pramesti, R. (2013). Identifikasi Karakter Plat Nomor Kendaraan Menggunakan Ekstraksi Fitur ICZ dan ZCZ dengan Metode Klasifikasi KNN. *Scientific Repository of Bogor Agricultural University*.

Purnomo, J. F. (2010). Analisis Perbandingan Beberapa Metode Pembobotan Kata terhadap Performansi Kategorisasi Teks. *Teknik Informatika, Fakultas Teknik Informatika, Universitas Telkom*.

Sitanggang, I. S. (2017). *Clustering Menggunakan Self-Organizing Maps.* Jurnal Ilmiah Ilmu Komputer.

Suguna, N. d. (2010). An Improved k-*Nearest Neighbor* Classification Using Genetic Algorithm. *International Journal of Computer Science Issues, Vol. 7, Issue 4, No 2*.

Suprapto, A. (2017). *Sistem Klasifikasi Opini pengguna Maskapai Penerbangan di Indonesia pada Jejaring Sosial Twitter menggunakan Metode K-Nearest Neighbor*. Malang.

Tala, F. Z. (2003). A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia. *M.Sc. Thesis. Master of Logic Project. Institute for Logic, Language and Computation. Universiteit van Amsterdam, The Netherlands*.

Toyota, T. N. (2012). *Visualization of the Internet News Based on Efficient SelfOrganizing Map Using Restricted Region Search and Dimensionality Reduction.* Journal of Advanced Computational Intelligence and Intelligent Informatics.

Turban, E. (2015). *Decision Support System and Intelligent System.* Yogyakarta: Andi.

Weiss, S. M. (2010). Text mining: Predictive Methods for Analyzing Unstructured Information. *New York: Springer*.

Ye, J. (2014). Vector Similarity Measures of Simplified Neutroshopic Sets and Their Application in Multicriteria Decision Making. *Internasional Journal of Fuzzy Systems Volume 16*.

Zaki, M. J. (2014). *Data Mining and Analysis*: *Fundamental Concepts and Algorithms*. *Cambridge University Press, New York. ISBN 978-0-521-76633-3*.

Zhao, Y. (2013). R and *Data Mining*: Examples and Case Studies. *Elsevier Publisher Inc.*

Zhiqiang, L. W. (2009). Measuring Semantic Similarity between Words Using Wikipedia. *IEEE. 251-255*.

LAMPIRAN

Lampiran 1. Hasil uji coba skenario pertama.

ID Abstrak	Kelas Sesungguhnya	Kelas Prediksi			
		Naïve Bayes	KNN		
			K = 3	K = 6	K= 9
2	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
3	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
4	HE5601-5725	HE5601-5725	HE323-328	HE5601-5725	HE323-328
5	HE5601-5725	HE5601-5725	HE323-328	HE5601-5725	HE5601-5725
6	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
7	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE323-328
8	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
9	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
10	HE5601-5725	HE5601-5725	HE5601-5725	HE323-328	HE323-328
11	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
12	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
13	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
14	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE323-328
15	HE5601-5725	HE323-328	HE323-328	HE323-328	HE323-328
16	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
17	HE5601-5725	HE5601-5725	HE5601-5725	HE323-328	HE5601-5725
18	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
19	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE1-9990
20	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
21	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
22	HE5601-5725	HE5601-5725	HE5601-5725	HE323-328	HE323-328
23	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
24	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
25	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
26	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
27	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
28	HE323-328	HE323-328	HE323-328	HE5601-5725	HE5601-5725
29	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
30	HE323-328	HE323-328	HE5601-5725	HE5601-5725	HE323-328
31	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
32	HE323-328	HE323-328	HE5601-5725	HE5601-5725	HE5601-5725
33	HE323-328	HE5601-5725	HE323-328	HE323-328	HE323-328
34	HE323-328	HE323-328	HE5601-5725	HE5601-5725	HE5601-5725
35	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328

36	HE323-328	HE323-328	HE5601-5725	HE5601-5725	HE323-328
37	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
38	HE323-328	HE323-328	HE323-328	HE323-328	HE5601-5725
39	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
40	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
41	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
42	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
43	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
44	HE323-328	HE323-328	HE323-328	HE5601-5725	HE1-9990
45	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
46	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
47	HE1-9990	HE1-9990	HE323-328	HE1-9990	HE1-9990
48	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
49	HE1-9990	HE1-9990	HE1-9990	HE323-328	HE323-328
50	HE1-9990	HE323-328	HE323-328	HE323-328	HE323-328
51	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
52	HE1-9990	HE323-328	HE323-328	HE323-328	HE5601-5725
53	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
54	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
55	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
56	HE1-9990	HE1-9990	HE1-9990	HE323-328	HE323-328
57	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
58	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
59	HE1-9990	HE323-328	HE1-9990	HE323-328	HE323-328
60	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
61	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
62	HE1-9990	HE5601-5725	HE1-9990	HE1-9990	HE1-9990
63	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
64	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
65	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE5601-5725
66	HE1-9990	HE323-328	HE5601-5725	HE5601-5725	HE5601-5725
67	HE1-9990	HE1-9990	HE1-9990	HE5601-5725	HE1-9990
TP		**59**	**55**	**49**	**47**
TN		**125**	**121**	**115**	**113**
FP		**7**	**11**	**17**	**19**
FN		**7**	**11**	**17**	**19**
Akurasi (%)		**92,92929293**	**88,88888889**	**82,82828283**	**80,80808081**

Ket: Nama kelas diwakili ID kelas

Lampiran 2. Hasil uji coba skenario kedua.

ID Abstrak	Kelas Sesungguhnya	Kelas Prediksi			
		Naïve Bayes	KNN		
			K = 3	K = 6	K= 9
18	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
19	HE5601-5725	HE5601-5725	HE5601-5725	HE1-9990	HE5601-5725
20	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
21	HE5601-5725	HE323-328	HE5601-5725	HE5601-5725	HE5601-5725
22	HE5601-5725	HE5601-5725	HE323-328	HE323-328	HE5601-5725
23	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
40	HE323-328	HE5601-5725	HE323-328	HE5601-5725	HE5601-5725
41	HE323-328	HE323-328	HE1-9990	HE1-9990	HE1-9990
42	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
43	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
44	HE323-328	HE5601-5725	HE1-9990	HE1-9990	HE1-9990
45	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
62	HE1-9990	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
63	HE1-9990	HE5601-5725	HE1-9990	HE5601-5725	HE323-328
64	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
65	HE1-9990	HE5601-5725	HE5601-5725	HE1-9990	HE1-9990
66	HE1-9990	HE323-328	HE5601-5725	HE5601-5725	HE1-9990
67	HE1-9990	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
TP		**10**	**11**	**9**	**12**
TN		**28**	**29**	**27**	**30**
FP		**8**	**7**	**9**	**6**
FN		**8**	**7**	**9**	**6**
Akurasi (%)		**70,37037037**	**74,07407407**	**66,66666667**	**77,77777778**

Ket: Nama kelas diwakili ID kelas

Lampiran 3. Hasil uji coba skenario ketiga

ID Abstrak	Kelas Sesungguhnya	Kelas Prediksi			
		Naïve Bayes	KNN		
			K = 3	K = 6	K= 9
13	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
14	HE5601-5725	HE5601-5725	HE1-9990	HE1-9990	HE323-328
15	HE5601-5725	HE323-328	HE323-328	HE5601-5725	HE323-328
16	HE5601-5725	HE323-328	HE5601-5725	HE5601-5725	HE5601-5725
17	HE5601-5725	HE5601-5725	HE1-9990	HE5601-5725	HE5601-5725
18	HE5601-5725	HE323-328	HE1-9990	HE5601-5725	HE5601-5725
19	HE5601-5725	HE5601-5725	HE1-9990	HE1-9990	HE1-9990
20	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
21	HE5601-5725	HE323-328	HE323-328	HE5601-5725	HE5601-5725
22	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
23	HE5601-5725	HE323-328	HE5601-5725	HE5601-5725	HE5601-5725
35	HE323-328	HE323-328	HE323-328	HE1-9990	HE1-9990
36	HE323-328	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
37	HE323-328	HE323-328	HE1-9990	HE1-9990	HE323-328
38	HE323-328	HE323-328	HE1-9990	HE323-328	HE5601-5725
39	HE323-328	HE5601-5725	HE1-9990	HE1-9990	HE1-9990
40	HE323-328	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
41	HE323-328	HE323-328	HE1-9990	HE1-9990	HE1-9990
42	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
43	HE323-328	HE323-328	HE5601-5725	HE323-328	HE323-328
44	HE323-328	HE323-328	HE1-9990	HE1-9990	HE5601-5725
45	HE323-328	HE323-328	HE323-328	HE323-328	HE5601-5725
57	HE1-9990	HE1-9990	HE1-9990	HE5601-5725	HE5601-5725
58	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE5601-5725
59	HE1-9990	HE323-328	HE1-9990	HE323-328	HE323-328
60	HE1-9990	HE1-9990	HE323-328	HE5601-5725	HE5601-5725
61	HE1-9990	HE5601-5725	HE1-9990	HE1-9990	HE1-9990
62	HE1-9990	HE5601-5725	HE323-328	HE323-328	HE5601-5725
63	HE1-9990	HE5601-5725	HE1-9990	HE323-328	HE5601-5725
64	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
65	HE1-9990	HE1-9990	HE5601-5725	HE1-9990	HE5601-5725
66	HE1-9990	HE323-328	HE5601-5725	HE1-9990	HE5601-5725
67	HE1-9990	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
TP		**19**	**14**	**18**	**13**
TN		**52**	**47**	**51**	**46**
FP		**14**	**19**	**15**	**20**
FN		**14**	**19**	**15**	**20**
Akurasi (%)		**71,71717172**	**61,61616162**	**69,6969697**	**59,5959596**

Ket: Nama kelas diwakili ID kelas

Lampiran 4. Hasil uji coba skenario keempat

ID Abstrak	Kelas Sesungguhnya	Kelas Prediksi			
		Naïve Bayes	KNN		
			K = 3	K = 6	K= 9
8	HE5601-5725	HE5601-5725	HE5601-5725	HE323-328	HE323-328
9	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
10	HE5601-5725	HE1-9990	HE1-9990	HE1-9990	HE5601-5725
11	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
12	HE5601-5725	HE1-9990	HE1-9990	HE323-328	HE5601-5725
13	HE5601-5725	HE323-328	HE5601-5725	HE5601-5725	HE5601-5725
14	HE5601-5725	HE5601-5725	HE1-9990	HE323-328	HE323-328
15	HE5601-5725	HE323-328	HE5601-5725	HE5601-5725	HE5601-5725
16	HE5601-5725	HE323-328	HE5601-5725	HE5601-5725	HE5601-5725
17	HE5601-5725	HE323-328	HE323-328	HE323-328	HE5601-5725
18	HE5601-5725	HE323-328	HE1-9990	HE1-9990	HE5601-5725
19	HE5601-5725	HE5601-5725	HE1-9990	HE5601-5725	HE1-9990
20	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
21	HE5601-5725	HE323-328	HE1-9990	HE5601-5725	HE5601-5725
22	HE5601-5725	HE323-328	HE5601-5725	HE5601-5725	HE5601-5725
23	HE5601-5725	HE323-328	HE5601-5725	HE323-328	HE323-328
30	HE323-328	HE323-328	HE5601-5725	HE5601-5725	HE323-328
31	HE323-328	HE323-328	HE5601-5725	HE323-328	HE323-328
32	HE323-328	HE323-328	HE5601-5725	HE1-9990	HE1-9990
33	HE323-328	HE5601-5725	HE5601-5725	HE1-9990	HE1-9990
34	HE323-328	HE323-328	HE5601-5725	HE1-9990	HE5601-5725
35	HE323-328	HE323-328	HE323-328	HE5601-5725	HE323-328
36	HE323-328	HE323-328	HE5601-5725	HE5601-5725	HE5601-5725
37	HE323-328	HE323-328	HE1-9990	HE1-9990	HE1-9990
38	HE323-328	HE323-328	HE323-328	HE323-328	HE5601-5725
39	HE323-328	HE5601-5725	HE5601-5725	HE5601-5725	HE5601-5725
40	HE323-328	HE323-328	HE323-328	HE5601-5725	HE5601-5725
41	HE323-328	HE323-328	HE1-9990	HE1-9990	HE1-9990
42	HE323-328	HE323-328	HE1-9990	HE323-328	HE323-328
43	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
44	HE323-328	HE323-328	HE1-9990	HE1-9990	HE1-9990
45	HE323-328	HE323-328	HE323-328	HE323-328	HE323-328
52	HE1-9990	HE323-328	HE5601-5725	HE323-328	HE323-328
53	HE1-9990	HE323-328	HE1-9990	HE1-9990	HE323-328
54	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
55	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
56	HE1-9990	HE323-328	HE1-9990	HE1-9990	HE1-9990
57	HE1-9990	HE1-9990	HE5601-5725	HE5601-5725	HE323-328

58	HE1-9990	HE1-9990	HE1-9990	HE5601-5725	HE5601-5725
59	HE1-9990	HE1-9990	HE1-9990	HE323-328	HE323-328
60	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
61	HE1-9990	HE323-328	HE323-328	HE323-328	HE323-328
62	HE1-9990	HE5601-5725	HE1-9990	HE1-9990	HE1-9990
63	HE1-9990	HE5601-5725	HE323-328	HE1-9990	HE323-328
64	HE1-9990	HE1-9990	HE1-9990	HE1-9990	HE1-9990
65	HE1-9990	HE1-9990	HE5601-5725	HE1-9990	HE1-9990
66	HE1-9990	HE5601-5725	HE5601-5725	HE5601-5725	HE323-328
67	HE1-9990	HE323-328	HE323-328	HE5601-5725	HE5601-5725
TP		**28**	**23**	**23**	**25**
TN		**76**	**71**	**71**	**73**
FP		**20**	**25**	**25**	**23**
FN		**20**	**25**	**25**	**23**
Akurasi (%)		**72,22222222**	**65,27777778**	**65,27777778**	**68,05555556**

Ket: Nama kelas diwakili ID kelas