Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.202293538
Vol. 9, No. 3, Juni 2022, hlm. 489-500 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 30/E/KPT/2018 e-ISSN: 2528-6579

EVALUASI KINERJA MLLIB APACHE SPARK PADA KLASIFIKASI BERITA PALSU DALAM BAHASA INDONESIA

Antonius Angga Kurniawan*[1], Metty Mustikasari[2]

[1,2]Universitas Gunadarma, Depok
Email: [1]anggaku@staff.gunadarma.ac.id, [2]metty@staff.gunadarma.ac.id
*Penulis Korespondensi

(Naskah masuk: 08 Mei 2020, diterima untuk diterbitkan: 25 Mei 2022)

Abstrak

Machine learning digunakan untuk menganalisis, mengklasifikasikan, atau memprediksi data. Untuk melakukan tugas dari machine learning diperlukan alat bantu dengan kinerja serta lingkungan yang kuat demi mendapatkan akurasi dan efisiensi waktu yang baik. MLlib Apache Spark adalah library machine learning yang memiliki kemampuan dan kecepatan yang sangat baik. Hal ini dikarenakan dalam melakukan pemrosesan data, MLlib berjalan di atas memori. Penelitian ini menggunakan MLlib Apache Spark untuk melakukan klasifikasi berita palsu berbahasa Indonesia dengan jumlah data sebanyak 1786 yang diperoleh dari situs penyedia berita palsu dan fakta, yaitu TurnBackHoax.id. Algoritma klasifikasi yang diterapkan adalah Naïve Bayes, Gradient-Boosted Tree, SVM dan Logistic Regression. Keempat algoritma dipilih karena kemampuannya yang sudah terbukti baik dalam melakukan klasifikasi dan beberapa algoritma yang jarang digunakan namun memiliki kemampuan yang baik juga dalam hal klasifikasi. Tahap pengolahan data diantaranya adalah preprocessing, feature extraction, penerapan algoritma. Evaluasi dilakukan berdasarkan accuracy, test error, f1-score, confusion matrix, dan running time. Hasil menunjukkan bahwa MLlib Apache Spark terbukti memiliki kinerja yang cepat dan baik karena dalam melakukan pemrosesan machine learning, running time tercepat yang didapat adalah 6.46 detik dengan menggunakan algoritma Logistic Regression. Akurasi yang didapat juga cukup baik dengan rata-rata test error dari keempat algoritma hanya 0.180. F1-score yang diperoleh pada keempat algoritma juga cukup baik dengan rata-rata sebesar 0.818. Confusion matrix yang dihasilkan juga baik, karena jumlah prediksi benar jauh lebih banyak dibandingkan dengan jumlah yang salah.

Kata kunci: *Apache Spark, evaluasi, klasifikasi, machine learning, MLlib*

PERFORMANCE EVALUATION OF MLLIB APACHE SPARK ON INDONESIAN LANGUAGE FAKE NEWS CLASSIFICATION

Abstract

Machine learning is used to analyze, classify, or predict data. To do the task of machine learning, we need tools with a strong performance and environment to get good accuracy and time efficiency. MLlib Apache Spark is a machine learning library that has excellent capabilities and speed. This is because in performing data processing, MLlib runs on memory. This research uses MLlib Apache Spark to classify fake news in Indonesian language with 1786 data that were obtained from fake news and fact provider sites, TurnBackHoax.id. The classification algorithm applied was Naïve Bayes, Gradient-Boosted Tree, SVM and Logistic Regression. The four algorithms were chosen because of their proven ability to classify and several algorithms that are rarely used but have good abilities in terms of classification. Data processing stages include preprocessing, feature extraction, and algorithm implementation. Evaluation was done based on accuracy, error test, f1-score, confusion matrix, and running time. The results showed that MLlib Apache Spark was proven to have a fast and good performance because in doing machine learning processing, the fastest running time was 6.46 seconds using the Logistic Regression algorithm. The accuracy obtained was also quite good with an average test error of the four algorithms of only 0.180. F1-scores obtained on the four algorithms were also quite good with an average of 0.818. The result of confusion matrix was also good, because the number of correct predictions was far more than the number of incorrect ones.

Keywords: *Apache Spark, evaluation, classification, machine learning, Mllib*

1. PENDAHULUAN

Big data merupakan sumber data berukuran besar dan beragam. Tiga karakteristik utama dari *big data*, yaitu *Volume*, *Velocity*, dan *Variety* (3Vs). Karakteristik tersebut adalah salah satu tantangan dalam melakukan analisis *big data*. Oleh karena itu,

dibutuhkan kerangka kerja, strategi dan lingkungan *machine learning* yang kuat untuk dapat melakukan analisis dengan baik dan benar (Etaiwi, Biltawi and Naymat, 2017).

Tidak semua alat *machine learning,* seperti R dan Weka mampu menganalisis *volume* data dengan jumlah yang besar dan dapat masuk ke memori utama pada satu komputer (Al-Saqqa, Al-Naymat and Awajan, 2018). Salah satu alat yang digunakan untuk melakukan pemrosesan data dengan skala besar adalah Apache Hadoop. Apache Hadoop dirancang untuk menyelesaikan masalah dengan tujuan analitik, namun Apache Hadoop memiliki kekurangan pada bagian yang cukup penting. Kekurangan pada Apache Hadoop adalah *overhead* yang tinggi pada saat menjalankan setiap prosesnya, ketergantungan antara penyimpanan data dan hasil perhitungan ke *disk*. Oleh karena itu, Apache Hadoop tidak cocok digunakan untuk kasus yang sifatnya iteratif atau latensi rendah (Fu, Sun and Wang, 2017).

Apache Spark adalah salah satu kerangka kerja baru yang digunakan untuk komputasi terdistribusi dan dirancang untuk mengoptimalkan tugas-tugas latensi rendah. Apache Spark menyimpan data, hasil, dan bekerja di dalam memori, sehingga sangat cocok untuk aplikasi iteratif dan *machine learning* (Fu, Sun and Wang, 2017).

Apache Spark memiliki komponen *library open-source* yang digunakan untuk melakukan analisis data skala besar, yaitu *Machine Learning library* (MLlib). MLlib Apache Spark telah dilengkapi dengan serangkaian fungsionalitas untuk berbagai tugas *machine learning*, termasuk regresi, klasifikasi, klustering, dan juga aturan ekstraksi (Assefi *et al.*, 2017).

Penerapan *machine learning* yang efektif telah dipelajari sejak lama di komunitas penelitian, namun studi tentang *machine learning library* untuk *big data* seperti MLlib Apache Spark sejauh ini sangat terbatas (Assefi *et al.*, 2017). Selain itu, sampai saat ini belum banyak penelitian yang dilakukan sehubungan dengan pemrosesan data di dalam memori seperti MLlib Apache Spark (Yasrobi *et al.*, 2017).

MLlib Apache Spark memiliki algoritma *machine learning* dengan kemampuan yang baik, mudah digunakan, kinerja yang tinggi, cepat dalam melakukan pemrosesan data, dan dapat berjalan di banyak *platform* (Meng *et al.*, 2016).

Jamil (2016) melakukan analisis data berdasarkan algoritma *data mining* menggunakan kerangka kerja Weka. Data yang dianalisis adalah data kanker payudara. Atribut yang digunakan adalah *age, tumor-size, menopause, breast, inv-nodes, node-caps, deg-malig, breast-quad, irradiat, class*. Data dianalisis dengan membandingkan algoritma klasifikasi Naïve Bayes, Support Vector Machine (SVM), Decision Tree, KStar, dan Artificial Neural Network (ANN). Perbandingan dilakukan dengan melihat hasil evaluasi yang diperoleh menggunakan alat bantu Weka untuk melakukan pemrosesan data. Evaluasi yang dilakukan adalah *accuracy*, *kappa statistic* dan *test error*. Hasil analisis performa dari algoritma *machine learning* untuk klasifikasi data kanker payudara adalah Naïve Bayes dengan nilai *accuracy* sebesar 71.67%, nilai *kappa statistic* sebesar 0.2857, nilai *test error* sebesar 0.3272. SVM nilai *accuracy* 69.58%, nilai *kappa statistic* sebesar 0.1983, nilai *test error* sebesar 0.3042. Decision Tree nilai *accuracy* 75.52%, nilai *kappa statistic* sebesar 0.2826, nilai *test error* sebesar 0.3676. K-Star nilai *accuracy* 73.52%, nilai *kappa statistic* sebesar 0.2864, nilai *test error* sebesar 0.3354. Hasil tertinggi diperoleh dengan algoritma klasifikasi ANN dengan nilai *accuracy* 73.77%, nilai *kappa statistic* 0.3386, dan nilai *test error* 0.3198 (Jamil, 2016).

Assefi *et al.*, (2017) melakukan penelitian dengan membandingkan alat bantu dalam melakukan pemrosesan data berskala besar, yaitu Apache Spark dan Weka. Assefi *et al.,* membandingkan algoritma klasifikasi, yaitu SVM, Random Forest, Decision Tree, Naïve Bayes, dan algoritma klustering, yaitu K-Means. *Dataset* yang digunakan terdiri dari 6 variasi, di mana 5 *dataset* berasal dari UCI *Machine Learning Repository* dan 1 *dataset* berasal dari US *Government's Bureau of Transportation Research and Innovative Technology Administration* (RITA). Setiap *dataset* dibagi menjadi 75% untuk data *training* dan 25% untuk data *testing.* Hasil perbandingan dilihat berdasarkan evaluasi terhadap *running time* dalam melakukan pemrosesan data dan juga hasil dari Area Under ROC. Hasil yang diperoleh menunjukkan bahwa Apache Spark dengan algoritma klasifikasi dan algoritma klustering yang diujikan mendapatkan total *running time* yang jauh lebih rendah dibandingkan dengan Weka, namun di dalam penelitian tidak dituliskan secara spesifik berapa waktu untuk *running time* baik pada Apache Spark maupun Weka. Area Under ROC yang diperoleh menunjukkan hasil statistik yang beragam. Hasil yang didapat tidak memiliki selisih yang jauh berbeda pada masing-masing algoritma. Hasil berdasarkan beberapa *dataset* yang digunakan pada Apache Spark dan Weka sama-sama pernah memiliki hasil yang lebih baik (Assefi *et al.*, 2017).

Deshai, Sekhar and Venkataramana (2019) dalam penelitiannya mengevaluasi paradigma inti dari *open-source* Apache Spark, *core technology* dan operasi desentralisasi sistem *library* dari Apache Spark terutama MLlib. Pada penelitian dilakukan perbandingan antara Apache Spark, Weka dan Hadoop. Data yang digunakan adalah *Flight dataset*. Perbandingan Apache Spark dan Weka dievaluasi berdasarkan *running time* yang diterapkan untuk *machine learning* menggunakan algoritma klasifikasi, yaitu Random Forest, Decision Tree, Naïve Bayes, dan SVM. Hasil yang didapat Apache Spark memiliki *running time* yang lebih rendah pada semua algoritma yang diuji dibandingkan dengan Weka. Pada penelitian ini tidak dituliskan secara spesifik mengenai berapa waktu *running time* dan

selisih yang diperoleh pada setiap algoritma. Perbandingan antara Apache Spark dan Hadoop dievaluasi berdasarkan *response time* untuk *machine learning, latency,* dan *performance gain* pada GraphX. Hasil yang didapat menunjukkan bahwa Apache Spark selalu unggul di dalam setiap pengujian dibandingkan dengan Hadoop. Penelitian ini tidak menuliskan secara spesifik perbandingan hasil yang didapat, namun hanya ditunjukkan dengan sebuah grafik (Deshai, Sekhar and Venkataramana, 2019).

Juwiantho et al., (2020) dalam penelitiannya melakukan analisis sentimen terhadap *Tweet* berbahasa Indonesia. Metode yang digunakan adalah *classical machine learning* yang sudah banyak digunakan untuk analisis sentimen. Juwiantho et al., berpendapat metode tersebut tidak memperhatikan pentingnya urutan kata pada suatu kalimat. Pada penelitiannya diusulkan metode *deep learning*, yaitu *Deep Convolutional Network* untuk menjawab permasalahan yang ditemukan. Data yang digunakan sebanyak 999 tweet. Data diolah dengan cara manual *labeling* dengan membagi sentimen menjadi 2 sentimen (positif dan tidak positif), 3 sentimen (positif, negatif, dan netral), tokenisasi, *stopword* removal, digunakan juga model Word2Vec untuk mengubah *tweet* menjadi sebuah vektor, dan kemudian diimplementasikan *Deep Convolutional Network* (DCNN). Pada penelitian ini, dibandingkan hasil dari metode DCNN dengan metode *classical machine learning* seperti Naïve Bayes dan SVM. Perbandingan dilihat berdasarkan hasil evaluasi akurasi pada masing-masing metode. Perbandingan metode *classical* dan *deep learning* diperoleh hasil akurasi pada metode *classical* sebesar 70,90% menggunakan Naïve Bayes pada 2 sentimen dan 59,83% menggunakan SVM pada 3 sentimen. Akurasi terbaik pada metode deep learning sebesar 76,40% pada 2 sentimen dan 69,20% pada 3 sentimen. Hasil dari metode *deep learning* lebih baik 5,5% pada 2 sentimen dan 9,37% pada 3 sentimen dibandingkan dengan metode *classical machine learning* (Juwiantho *et al.*, 2020).

Tujuan penelitian ini adalah menyajikan hasil evaluasi kinerja pada MLlib *Classification*. Evaluasi kinerja dilihat dengan menerapkan *machine learning* menggunakan empat algoritma klasifikasi. Pada penelitian sebelumnya, terdapat beberapa algoritma yang sering digunakan untuk *machine learning*, diantaranya adalah Random Forest, Decision Tree, Naïve Bayes dan Support Vector Machine (SVM). Pada penelitian ini digunakan algoritma Naïve Bayes, SVM, Logistic Regression dan Gradient-Boosted Tree (GBT). Naïve Bayes adalah algoritma yang sederhana serta cepat dalam *machine learning* sehingga sering mengungguli metode lainnya, khususnya untuk klasifikasi (Jamil, 2016). SVM adalah algoritma terbaik dan sering digunakan karena tingkat akurasinya yang lebih baik (Tripathy, Agrawal and Rath, 2015). Logistic Regression dan GBT dipilih karena pemakaian dari kedua algoritma ini jarang digunakan di dalam penelitian, namun sebenarnya kedua algoritma tersebut juga bagus digunakan untuk *machine learning* terutama dalam hal klasifikasi teks. Logistic Regression sangat efektif pada data teks dan algoritma yang mendasarinya juga cukup mudah dimengerti (Ganesan and Subotin, 2014). Lebih penting lagi, di dunia NLP (*Natural Language Processing*), secara umum diterima bahwa Logistic Regression adalah algoritma pemula yang bagus untuk klasifikasi terkait teks (Jurafsky and Martin, 2019). GBT merupakan salah satu algoritma umum untuk meningkatkan akurasi algoritma *machine learning*. GBT menggunakan kombinasi dari pengklasifikasi lemah untuk membuat model klasifikasi yang lebih kuat. Algoritma klasifikasi GBT memungkinkan model *decision tree* pertama dilatih berdasarkan *dataset* pelatihan, meningkatkan akurasi model secara iteratif dengan melatih kembali model. Selama setiap pelatihan, tupel akan diputar ulang, sehingga model dapat mengurangi tingkat kesalahannya (De'ath, 2007).

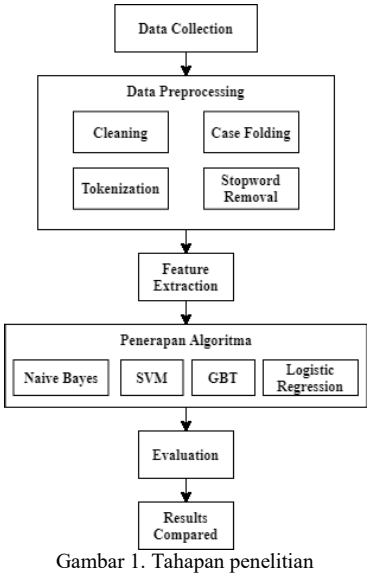
Empat algoritma tersebut digunakan untuk mengklasifikasi berita palsu dalam bahasa Indonesia. Berdasarkan hasil dari penelitian sebelumnya, empat algoritma tersebut memang biasa digunakan untuk melakukan klasifikasi, terutama klasifikasi teks. Belum adanya hasil evaluasi secara spesifik seperti berapa waktu *running time* yang dibutuhkan dalam melakukan pemrosesan data, tingkat akurasi yang masih kurang dari 80%, dan masih minimnya penelitian terkait *machine learning* menggunakan MLlib Apache Spark terutama dalam melakukan klasifikasi teks berbahasa Indonesia. Oleh karena itu, untuk menunjukkan hasil evaluasi yang lebih spesifik, maka algoritma yang digunakan dievaluasi dengan *accuracy*, *test error, f1-score, confusion matrix,* dan *running time* yang dirangkum ke dalam sebuah tabel. Evaluasi *running time* sangat berperan penting untuk membuktikan seberapa cepat MLlib Apache Spark dapat memproses suatu data yang diolah di dalam memori.

Kontribusi yang diharapkan dari penelitian ini yang pertama adalah memberikan pengetahuan tentang kemampuan MLlib Apache Spark terutama dari sisi kecepatan dalam melakukan pemrosesan *machine learning*. Kedua adalah hasil evaluasi pada empat algoritma dapat menjadi bahan pertimbangan untuk memilih algoritma yang tepat dalam melakukan klasifikasi teks.

2. METODE PENELITIAN

Metode yang dilakukan di dalam penelitian secara garis besar ditunjukkan pada Gambar 1. Pada penelitian ini digunakan Apache Spark versi 2.3.0, Python versi 3.7.0, dan Jupyter Notebook.

Tahapan penelitian terdiri dari *data collection, data preprocessing, feature extraction,* penerapan algoritma, *evaluation,* dan *results compared.*



Gambar 1. Tahapan penelitian

2.1. *Data Collection*

Data yang digunakan di dalam penelitian ini diperoleh dan dikumpulkan dari situs penyedia berita fakta atau berita palsu, yaitu Turnbackhoax.id. Situs ini dikelola oleh Masyarakat Anti Hoax Indonesia (MAFINDO). MAFINDO adalah organisasi perkumpulan resmi yang didirikan pada tanggal 19 November 2016. Sumber konten dari situs ini berasal dari Forum Anti Fitnah Hasut dan Hoax (FAFHH) (*Tentang Kami - TurnBackHoax*, 2016).

Setiap artikel yang terdapat di dalam situs tersebut memiliki label yang menandakan apakah artikel tersebut adalah fakta atau berita palsu. Pada artikel juga disertai dengan penjelasan mengenai kebenaran artikel tersebut.

Data yang dikumpulkan sebanyak 1786 artikel. Data diambil berdasarkan artikel berita dari periode Juni 2016 sampai dengan Desember 2018. Data sumber yang telah dikumpulkan disimpan ke dalam dokumen dengan format .csv yang diberi nama hoax-data.csv. Contoh dari data sumber yang akan digunakan ditunjukkan pada tabel 1.

Tabel 1. Contoh data sumber

Class	*Text*
0	Mendiang Abdullah Fithri Setiawan – rahiimahulloh – adalah karyawan TV Muhammadiyah, yang meliput dalam tentang mobil Esemka. Dan kini dia – rahiimahulloh – dibunuh dengan sadis.
0	Beredar pesan siaran atau broadcast … dilakukan lewat situs hingga hari pencoblosan ...
1	Penonton sepi teman Ahok dikasih tiket gratis dan uang Rp ribu supaya mau nonton film si Ahok

Tabel 1 menunjukkan bahwa data sumber terdiri dari dua kolom, yaitu kolom "*class*" dan kolom "*text*". Kolom "*class*" menunjukkan sebuah label apakah artikel termasuk kategori fakta "0" atau kategori berita palsu "1". Kolom "*text*" menunjukkan isi dari artikel.

2.2. *Data Preprocessing*

Pada tahap ini, data yang sudah dimuat kemudian diolah dan dipersiapkan agar lebih mudah dibaca oleh mesin. Pada penelitian ini *pre-processing* terdiri menjadi 4 tahap, yaitu proses *cleaning,* proses *case folding,* proses *tokenization, dan* proses *stopword removal.*

a. Cleaning

Proses ini bertujuan untuk membersihkan elemen-elemen yang dapat mengurangi makna sebenarnya dari suatu teks. Elemen yang dihilangkan adalah tanda baca, angka, spasi ganda, *link* URL, dan *mention* (@). Elemen tersebut ada di dalam data teks, namun tidak relevan dengan topik dari berita palsu. Contoh hasil dari proses *cleaning* ditunjukkan pada tabel 2.

Tabel 2. Contoh hasil *cleaning*

Data Asli	**Hasil Cleaning**
Mendiang Abdullah Fithri Setiawan – rahiimahulloh – adalah karyawan TV Muhammadiyah, yang meliput dalam tentang mobil Esemka. Dan kini dia – rahiimahulloh – dibunuh dengan sadis.	Mendiang Abdullah Fithri Setiawan rahiimahulloh adalah karyawan TV Muhammadiyah yang meliput dalam tentang mobil Esemka Dan kini dia rahiimahulloh dibunuh dengan sadis
Beredar pesan siaran atau broadcast … dilakukan lewat situs https://sidalih3.kpu.go.id hingga hari pencoblosan. ...	Beredar pesan siaran atau broadcast … dilakukan lewat situs hingga hari pencoblosan ...
Penonton sepi teman Ahok dikasih tiket gratis dan uang Rp 50 ribu supaya mau nonton film si Ahok	Penonton sepi teman Ahok dikasih tiket gratis dan uang Rp ribu supaya mau nonton film si Ahok

Tabel 2 menunjukkan hasil dari contoh proses *cleaning*. Dalam tabel terlihat tanda baca seperti titik dan setrip dihilangkan. Kalimat yang mengandung alamat *URL,* dan angka juga ikut dihilangkan. Setelah proses *cleaning* selesai dilakukan, berikutnya adalah melakukan tahapan *case folding.*

b. Case Folding

Seluruh teks yang telah dibersihkan kemudian diubah menjadi menjadi huruf kecil. Hal ini dilakukan karena tidak semua data teks konsisten dalam penggunaan huruf kapital. Contoh hasil dari proses *case folding* ditunjukkan pada tabel 3.

Case folding adalah proses di mana semua kalimat yang memiliki huruf besar akan diubah ke dalam huruf kecil seperti ditunjukkan pada tabel 3. Setelah proses *case folding* selesai dilakukan, berikutnya adalah melakukan tahapan *tokenization.*

Tabel 3. Contoh hasil *case folding*

Hasil *Cleaning*	Hasil *Case Folding*
Mendiang Abdullah Fithri Setiawan rahiimahulloh adalah karyawan TV Muhammadiyah yang meliput dalam tentang mobil Esemka Dan kini dia rahiimahulloh dibunuh dengan sadis	mendiang abdullah fithri setiawan rahiimahulloh adalah karyawan tv muhammadiyah yang meliput dalam tentang mobil esemka dan kini dia rahiimahulloh dibunuh dengan sadis
Beredar pesan siaran atau broadcast … dilakukan lewat situs hingga hari pencoblosan ...	beredar pesan siaran atau broadcast … dilakukan lewat situs hingga hari pencoblosan ...
Penonton sepi teman Ahok dikasih tiket gratis dan uang Rp ribu supaya mau nonton film si Ahok	penonton sepi teman ahok dikasih tiket gratis dan uang rp ribu supaya mau nonton film si ahok

c. *Tokenization*

Secara umum, tahap *tokenization* adalah proses pemisahan kata di dalam teks ke dalam unit kata atau token. Tujuan dari proses ini adalah untuk membuat token *individual* yang mewakili setiap kata dalam pernyataan. Alur dari proses *tokenization* ditunjukkan pada gambar 2.

Gambar 2. Alur dari proses *tokenization*

Proses pertama, mesin menerima *input* kalimat yang akan di*tokenize*. Proses kedua adalah memisahkan kalimat menjadi kata demi kata. Proses ketiga, mengubah kata ke dalam bentuk *array*. Setelah itu didapatkan hasil dari *tokenization*. Contoh hasil dari proses *tokenization* ditunjukkan pada tabel 4.

Proses *tokenization* merubah kalimat menjadi satuan kata yang dibentuk ke dalam sebuah *array* seperti yang ditunjukkan pada tabel 4. Kalimat dipisahkan ke dalam kata satu per satu. Setelah proses *tokenization* selesai dilakukan, berikutnya adalah melakukan tahapan *stopword removal.*

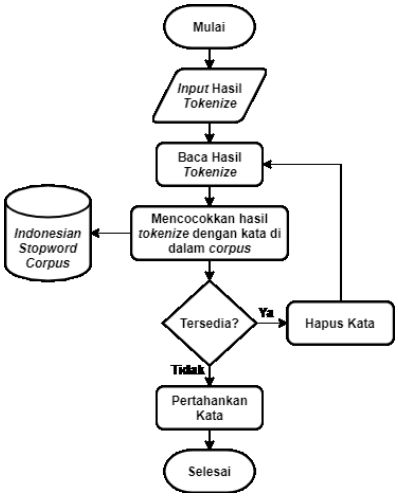
d. *Stopword Removal*

Pada tahap ini, istilah-istilah yang tidak relevan dengan subjek utama dari data yang digunakan akan dihilangkan, meskipun kata-kata tersebut sering muncul dalam teks yang digunakan. Kata-kata tersebut terdiri dari kata penghubung, kata depan, kata penentu, dan kata-kata lain yang sejenis.

Tabel 4. Contoh hasil *tokenization*

Hasil *Case Folding*	Hasil *Tokenization*
mendiang abdullah fithri setiawan rahiimahulloh adalah karyawan tv muhammadiyah yang meliput dalam tentang mobil esemka dan kini dia rahiimahulloh dibunuh dengan sadis	[“mendiang”, ”abdullah”, ”fithri”, ”setiawan”, ”rahiimahulloh”, ”adalah”, ”karyawan”, ”tv”, ”muhammadiyah”, ”yang”, ”meliput”, ”dalam”, ”tentang”, ”mobil”, ”esemka”, ”dan”, ”kini”, ”dia”, ”rahiimahulloh”, ”dibunuh”, ”dengan”, ”sadis”]
beredar pesan siaran atau broadcast … dilakukan lewat situs hingga hari pencoblosan ...	[”beredar”, ”pesan”, ”siaran”, ”atau”, ”broadcast”, … ”dilakukan”, ”lewat”, ”situs”, ”hingga”, ”hari”, ”pencoblosan” ...]
penonton sepi teman ahok dikasih tiket gratis dan uang rp ribu supaya mau nonton film si ahok	[”penonton”, ”sepi”, ”teman”, ”ahok”, ”dikasih”, ”tiket”, ”gratis”, ”dan”, ”uang”, ”rp”, ”ribu”, ”supaya”, ”mau”, ”nonton”, ”film”, ”si”, ”ahok”]

Stopword juga bisa dikostumisasi sesuai dengan kebutuhan datanya (Cios *et al.*, 2007). Daftar *stopwords* dalam bahasa Indonesia yang digunakan pada tahap ini merupakan *dataset* publik yang didapatkan dari Devid Haryalesmana yang diunggah pada repositori Github (Haryalesmana, 2016). Alur dari proses *stopword removal* ditunjukkan pada gambar 3.



Gambar 3. Alur dari proses *stopword removal*

Hasil dari proses *tokenize* digunakan sebagai input. Kemudian, *Stopword remover* akan membaca kata demi kata. Setelah itu, *stopword remover* memeriksa apakah kata tersebut tersedia di dalam *corpus* atau tidak. Jika ya, kata itu akan dihapus, tetapi jika tidak, maka kata itu akan tetap disimpan. Contoh hasil dari proses *stopword removal* ditunjukkan pada tabel 5.

Tabel 5. Contoh hasil *stopword removal*

Hasil *Tokenization*	Hasil *Stopword Removal*
[“mendiang”, ”abdullah”, ”fithri”, ”setiawan”, ”rahiimahulloh”, ”adalah”, ”karyawan”, ”tv”, ”muhammadiyah”, ”yang”, ”meliput”, ”dalam”, ”tentang”, ”mobil”, ”esemka”, ”dan”, ”kini”, ”dia”, ”rahiimahulloh”, ”dibunuh”, ”dengan”, ”sadis”]	[“mendiang”, ”abdullah”, ”fithri”, ”setiawan”, ”rahiimahulloh”, ”karyawan”, ”tv”, ”muhammadiyah”, ”meliput”, ”mobil”, ”esemka”, ”rahiimahulloh”, ”dibunuh”, ”sadis”]
[”beredar”, ”pesan”, ”siaran”, ”atau”, ”broadcast”, … ”dilakukan”, ”lewat”, ”situs”, ”hingga”, ”hari”, ”pencoblosan” ...]	[”beredar”, ”pesan”, ”siaran”, ”broadcast”, … ”situs”, ”pencoblosan” ...]
[”penonton”, ”sepi”, ”teman”, ”ahok”, ”dikasih”, ”tiket”, ”gratis”, ”dan”, ”uang”, ”rp”, ”ribu”, ”supaya”, ”mau”, ”nonton”, ”film”, ”si”, ”ahok”]	[”penonton”, ”sepi”, ”teman”, ”ahok”, ”dikasih”, ”tiket”, ”gratis”, ”uang”, ”rp”, ”ribu”, ”nonton”, ”film”, ”si”, ”ahok”]

Proses *stopword* seperti terlihat pada tabel 5 akan menghilangkan kata-kata seperti “yang”, “dan”, “di”, “dengan”, “saya” dan sebagainya. Dalam hal ini perlu disempurnakan atau dikostumisasi lagi isi dari *corpus* yang digunakan supaya memperoleh hasil yang lebih akurat dalam melakukan *stopword removal*. Setelah proses *stopword removal* selesai dilakukan, berikutnya adalah melakukan tahapan proses *feature extraction.*

2.3. *Feature Extraction*

Pada tahap ini digunakan untuk merubah teks yang sebelumnya sudah dirubah menjadi token-token dan dilakukan *stopword removal* akan dikonversi menjadi sebuah vektor. Untuk menghasilkan vektor tersebut, digunakan *TF-IDF* (*Term Frequency times Inverse Document Frequency*) yang merupakan pengukuran tentang seberapa terkonsentrasi kemunculan suatu kata dalam beberapa dokumen. *TF.IDF* dihitung seperti berikut, misalkan terdapat kumpulan *N* dokumen. Didefinisikan f_{ij} sebagai *frequency* (jumlah kemunculan) suatu istilah (kata) *i* dalam dokumen *j*. Didefinisikan *term frequency* TF_{ij} sebagai bentuk dari persamaan (1).

$$TF_{ij} = \frac{f_{ij}}{max_k f_{kj}} \quad (1)$$

Term frequency pada istilah *i* dalam dokumen *j* adalah f_{ij} yang dinormalisasikan dengan membaginya dengan jumlah maksimum kemunculan setiap istilah (kemungkinan tidak termasuk *stopword*) dalam dokumen yang sama. Karena itu, istilah yang paling sering muncul dalam dokumen *j* mendapatkan nilai *TF* 1, dan istilah lain mendapatkan sedikit bagian dari *term frequency* dalam dokumen tersebut.

Nilai *IDF* untuk suatu istilah didefinisikan misalkan istilah *i* muncul dalam n_i dari *N* dokumen seperti pada persamaan (2).

$$IDF_i = log_2 \left(\frac{N}{n_i}\right) \quad (2)$$

Nilai *TF.IDF* untuk istilah *i* dalam dokumen *j* kemudian didefinisikan sebagai persamaan (3).

$$TF.IDF = TF_{ij} \times IDF_i \quad (3)$$

Istilah dengan nilai *TF.IDF* tertinggi sering kali merupakan kata yang mengkarakterisasikan topik suatu dokumen dengan paling baik (Leskovec, Rajaraman and Ullman, 2014). Contoh hasil dari proses *feature extraction* terlihat pada tabel 6.

Tabel 6. Contoh hasil *feature extraction*

Hasil *Tokenization*	Hasil *Feature Extraction*
[“mendiang”, ”abdullah”, ”fithri”, ”setiawan”, ”rahiimahulloh”, ”karyawan”, ”tv”, ”muhammadiyah”, ”meliput”, ”mobil”, ”esemka”, ”rahiimahulloh”, ”dibunuh”, ”sadis”]	(41149,[23,184,64...)
[”beredar”, ”pesan”, ”siaran”, ”broadcast”, … ”situs”, ”pencoblosan” ...]	(41149,[0,14,59,6...)
[”penonton”, ”sepi”, ”teman”, ”ahok”, ”dikasih”, ”tiket”, ”gratis”, ”uang”, ”rp”, ”ribu”, ”nonton”, ”film”, ”si”, ”ahok”]	(41149,[1,6,8,10,...)

Machine learning tidak bisa melakukan pemrosesan data apabila belum dilakukan proses *feature extraction*. Hasil dari proses *feature extraction* dapat dilihat pada tabel 6, di mana setiap token yang ada diubah menjadi sebuah angka yang biasa disebut dengan kumpulan vektor. Vektor tersebut yang digunakan untuk proses penerapan algoritma atau pemodelan dalam menerapkan *machine learning*.

2.5. Penerapan Algoritma

Data yang sudah diproses sampai tahap *feature extraction* dipisahkan ke dalam dua bagian, yaitu data *train* dan data *test* dengan rasio 0.7:0.3, di mana rasio dari data *train* harus lebih banyak dibandingkan dengan data *test*. Penerapan algoritma dilakukan untuk mendapatkan hasil prediksi dari teks bahwa artikel termasuk ke dalam fakta atau berita palsu. Penelitian ini menggunakan alat bantu MLlib Apache Spark yang mendukung tugas-tugas *machine learning*. Algoritma yang digunakan adalah Naïve Bayes, Gradient-Boosted Tree (GBT), SVM, Logistic Regression.

a. Naïve Bayes

Algoritma Naïve Bayes menggunakan teorema Bayes, di mana probabilitas atau peluang bersyarat dinyatakan sebagai persamaan 4 berikut.

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)} \quad (4)$$

di mana *X* adalah bukti, *H* adalah hipotesis, *P(H|X)* adalah probabilitas bahwa hipotesis *H* benar untuk bukti *X* atau dengan kata lain *P(H|X)* merupakan probabilitas posterior *H* dengan syarat *X*, *P(X|H)* adalah probabilitas bahwa bukti *X* benar untuk hipotesis *H* atau probabilitas posterior *X* dengan syarat *H*, *P(H)* adalah probabilitas prior hipotesis *H*, dan *P(X)* adalah probabilitas prior bukti *X*.

Dalam *data mining*, *X* adalah sebuah tupel atau objek data, *H* adalah hipotesis atau dugaan bahwa tupel *X* adalah kelas *C*. Secara spesifik, dalam masalah klasifikasi dapat dihitung *P(H|X)* sebagai probabilitas bahwa hipotesis *H* benar untuk tupel *X* atau dengan kata lain *P(H|X)* adalah probabillitas bahwa hipotesis *H* benar untuk setiap tupel tidak peduli nilai-nilai atributnya sedangkan *P(X)* adalah probabilitas prior dari tupel *X* (Suyanto, 2017).

Contoh dari hasil penerapan algoritma menggunakan Naïve Bayes terlihat pada gambar 4.

```
predictions_show.show(5)
```

```
+-----+--------------------+----------+
|label|            features|prediction|
+-----+--------------------+----------+
|  0.0|(25291,[0,1,3,5,4...|       0.0|
|  0.0|(25291,[0,1,3,6,7...|       1.0|
|  0.0|(25291,[0,1,3,6,1...|       0.0|
|  0.0|(25291,[0,1,3,9,1...|       0.0|
|  0.0|(25291,[0,1,4,14,...|       0.0|
+-----+--------------------+----------+
only showing top 5 rows
```

Gambar 4. Contoh hasil prediksi algoritma Naïve Bayes

Hasil dari pemrosesan *machine learning* dengan Naïve Bayes ditunjukkan seperti pada gambar 4. Hasil yang ditampilkan hanya 5 baris teratas dengan kolom label sebagai label awal bahwa berita termasuk fakta "0"atau palsu "1", kolom *features* sebagai hasil dari proses ekstraksi berita ke dalam bentuk vektor, dan kolom *prediction* yang digunakan sebagai hasil prediksi berdasarkan label awal. Dari kelima contoh hasil yang ditampilkan menunjukkan bahwa hanya ada 1 baris yang hasil prediksinya tidak sesuai dengan label awal, yaitu pada baris kedua.

b. Gradient-Boosted Tree

Gradient-Boosted Trees (GBTs) adalah klasifikasi populer dan metode regresi menggunakan *ensembles decision tree*. GBT secara iteratif melatih pohon keputusan untuk meminimalkan fungsi kerugian atau kesalahan. Implementasi MLlib Apache Spark mendukung GBT untuk klasifikasi biner dan untuk regresi, serta menggunakan fitur kontinu dan kategoris (*Classification and regression - Spark 2.4.6 Documentation*, no date).

Contoh dari hasil penerapan algoritma menggunakan GBT terlihat pada gambar 5.

```
+----------+------------+--------------------+
|prediction|indexedLabel|            features|
+----------+------------+--------------------+
|       1.0|         0.0|(25291,[0,1,3,5,4...|
|       1.0|         0.0|(25291,[0,1,3,6,7...|
|       1.0|         0.0|(25291,[0,1,3,6,1...|
|       1.0|         0.0|(25291,[0,1,3,6,1...|
|       0.0|         0.0|(25291,[0,1,4,5,1...|
+----------+------------+--------------------+
only showing top 5 rows
```

Gambar 5. Contoh hasil prediksi algoritma GBT

Hasil dari pemrosesan *machine learning* dengan GBT ditunjukkan seperti pada gambar 5. Hasil yang ditampilkan hanya 5 baris teratas dengan kolom indexedLabel sebagai label awal bahwa berita termasuk fakta "0"atau palsu "1", kolom *features* sebagai hasil dari proses ekstraksi berita ke dalam bentuk vektor, dan kolom *prediction* yang digunakan sebagai hasil prediksi berdasarkan label awal. Dari kelima contoh hasil yang ditampilkan menunjukkan bahwa terdapat 4 baris yang hasil prediksinya tidak sesuai dengan label awal, yaitu pada baris pertama sampai keempat.

c. SVM

Algoritma SVM menggunakan contoh pelatihan untuk membuat *hyperplane* yang memisahkan *dataset* ke dalam kelas. Kompleksitas kelas dapat bervariasi, tetapi bentuk paling sederhana dari algoritma SVM hanya memiliki dua label yang dapat dipilih (Sopyła and Drozda, 2015). Tugas utama model klasifikasi SVM adalah untuk menemukan *maximum margin hyperplane* yang mengklasifikasikan kelompok vektor fitur diantara dua kelas (0 atau 1), yang merupakan jarak maksimum antara *hyperplane* dan *x* terdekat dari kedua kelas (Etaiwi, Biltawi and Naymat, 2017).

SVM yang digunakan dalam penelitian ini adalah Linear SVM yang ada pada Spark Machine Learning. Linear SVM mencari *hyperplane* yang secara simetris memisahkan titik data dalam set pelatihan antara kelas yang berbeda. Di sini dikenal sebagai batas keputusan, yang memisahkan ruang menjadi dua bagian, yaitu satu separuh untuk kelas berita fakta "0", dan separuh lainnya untuk kelas berita palsu "1".

Contoh dari hasil penerapan algoritma menggunakan SVM terlihat pada gambar 6.

```
predictions.show(5)
```

```
+-----+--------------------+----------+
|label|            features|prediction|
+-----+--------------------+----------+
|  0.0|(25291,[0,1,4,6,8...|       1.0|
|  0.0|(25291,[0,1,5,9,1...|       0.0|
|  0.0|(25291,[0,1,7,8,1...|       1.0|
|  0.0|(25291,[0,1,9,12,...|       0.0|
|  0.0|(25291,[0,1,11,12...|       0.0|
+-----+--------------------+----------+
only showing top 5 rows
```

Gambar 6. Contoh hasil prediksi algoritma SVM

Hasil dari pemrosesan *machine learning* dengan SVM ditunjukkan seperti pada gambar 6. Hasil yang ditampilkan hanya 5 baris teratas dengan kolom label sebagai label awal bahwa berita termasuk fakta "0"atau palsu "1", kolom *features* sebagai hasil dari proses ekstraksi berita ke dalam bentuk vektor, dan kolom *prediction* yang digunakan sebagai hasil prediksi berdasarkan label awal. Dari kelima contoh hasil yang ditampilkan menunjukkan bahwa terdapat 2 baris yang hasil prediksinya tidak sesuai dengan label awal, yaitu pada baris pertama dan ketiga.

d. Logistic Regression

Dalam *Natural Language Processing (NLP)*, Logistic Regression adalah algoritma *machine learning* yang digunakan untuk klasifikasi, dan juga memiliki hubungan yang sangat dekat dengan *neural network*. Hal ini dikarenkan *neural network* dapat dilihat sebagai serangkaian pengklasifikasi *logistic regression* yang ditumpuk satu sama lain. Logistic Regression dapat digunakan untuk mengklasifikasikan pengamatan menjadi satu dari dua kelas (seperti 'berita fakta' dan 'berita palsu'), atau ke dalam salah satu dari banyak kelas (Jurafsky and Martin, 2019).

Model Logistic Regression membangun model Binary Classification untuk memprediksi suatu berita fakta atau palsu. Logistic Regression menggunakan persamaan sebagai representasi, sangat mirip dengan regresi linier. Nilai *input (x)* digabungkan secara linier menggunakan bobot atau nilai koefisien untuk memprediksi nilai *output (y)*. Perbedaan utama dari regresi linier adalah bahwa nilai *output* yang dimodelkan adalah nilai biner (0 atau 1), bukan nilai numerik. Persamaan (5) menunjukkan persamaan dari Logistic Regression.

$$y = e\hat{}(b0 + b1 * x)/(1 + e\hat{}(b0 + b1 * x)) \quad (5)$$

Di mana *y* adalah *output* yang diprediksi, *b0* adalah istilah bias atau intersep dan *b1* adalah koefisien untuk nilai *input* tunggal *(x)*. Setiap kolom dalam data input memiliki koefisien *b* terkait (nilai riil konstan) yang harus dipelajari dari data pelatihan. Nilai *e* dalam persamaan (5) mewakili kurva berbentuk *S* yang memiliki nilai antara 0 dan 1. Representasi sebenarnya dari model yang disimpan dalam memori atau dalam file adalah koefisien dalam persamaan (nilai beta atau *b*).

Contoh dari hasil penerapan algoritma menggunakan Logistic Regression terlihat pada gambar 7.

Hasil dari pemrosesan *machine learning* dengan Logistic Regression ditunjukkan seperti pada gambar 7. Hasil yang ditampilkan hanya 5 baris teratas dengan kolom label sebagai label awal bahwa berita termasuk fakta "0"atau palsu "1", kolom *features* sebagai hasil dari proses ekstraksi berita ke dalam bentuk vektor, dan kolom *prediction* yang digunakan sebagai hasil prediksi berdasarkan label awal.

```
predictions.show(5)
```

```
+-----+--------------------+----------+
|label|            features|prediction|
+-----+--------------------+----------+
|  0.0|(25291,[0,1,2,3,4...|       1.0|
|  0.0|(25291,[0,1,3,4,1...|       0.0|
|  0.0|(25291,[0,1,3,5,4...|       0.0|
|  0.0|(25291,[0,1,3,6,1...|       0.0|
|  0.0|(25291,[0,1,3,6,1...|       0.0|
+-----+--------------------+----------+
only showing top 5 rows
```

Gambar 7. Contoh hasil prediksi algoritma Logistic Regression

Dari kelima contoh hasil yang ditampilkan menunjukkan bahwa hanya ada 1 baris yang hasil prediksinya tidak sesuai dengan label awal, yaitu pada baris pertama.

2.6. *Evaluation*

Pada tahap ini digunakan sebagai evaluator untuk mengukur efektivitas *classifier* yang dihasilkan ketika diuji dengan data yang tidak terlihat (Hossin and Sulaiman, 2015). *Evaluation metrics* yang digunakan dalam penelitian ini adalah *accuracy, confusion matrix, f1-score, test error*. Evaluasi terhadap *running time* juga dilakukan pada masing-masing algoritma yang digunakan untuk pemrosesan *machine learning*.

a. Accuracy

Pengukuran seberapa sering algoritma klasifikasi yang dibuat berhasil membuat prediksi yang benar. Formula dari pengukuran nilai akurasi terlihat pada persamaan (6) (Hossin and Sulaiman, 2015).

$$Accuracy = \frac{Total\ Prediksi\ Benar}{Total\ Prediksi} \quad (6)$$

Akurasi merupakan rasio antara jumlah prediksi benar dan total jumlah prediksi atau total prediksi benar dibagi dengan total prediksi.

b. Confusion Matrix

Confusion matrix memberikan informasi perbandingan hasil klasifikasi yang dilakukan oleh model dengan hasil klasifikasi sebenarnya. Baris matriks mewakili label *actual*, dan kolom mewakili prediksi. Dengan *confusion matrix,* evaluasi terhadap *machine learning* dapat dihitung seperti nilai *accuracy, test error, f1-score,* dan yang lainnya (Zheng, 2015). Tabel 7 menunjukkan contoh bentuk dari *confusion matrix*.

Bentuk tabel dari *confusion matrix* terdiri dari 2 kolom, yaitu prediksi *y = 0* sebagai fakta, prediksi *y = 1* sebagai berita palsu dan 2 baris, yaitu *actual y = 0* sebagai fakta, *actual y = 1* sebagai berita palsu. Diantara nilai prediksi dan *actual* saling berhubungan

sehingga sebuah *confusion matrix* terdiri dari nilai *True Positive* (TP), *False Positive* (FP), *False Negative* (FN) dan *True Negative* (TN).

Tabel 7. Contoh bentuk dari *confusion matrix*

	Prediksi y=0 (fakta)	**Prediksi y=1 (berita palsu)**
***Actual* y=0 (fakta)**	*True Positive* (TP)	*False Positive* (FP)
***Actual* y=1 (berita palsu)**	*False Negative* (FN)	*True Negative* (TN)

Pada penelitian ini, *True Positive* merupakan data fakta yang diprediksi benar sebagai fakta, *False Positive* merupakan data fakta yang diprediksi salah sebagai berita palsu, *False Negative* merupakan data berita palsu yang diprediksi salah sebagai fakta, *True Negative* merupakan data berita palsu yang diprediksi benar sebagai berita palsu.

c. F1-*Score*

F1-*Score* atau biasa juga dikenal dengan F-*Measure* adalah matriks yang mewakili rata-rata harmonik antara nilai *recall* dan *precision*. Biasanya formula dari F1-*Score* terlihat seperti pada persamaan (7) (Hossin and Sulaiman, 2015).

$$F1Score = \frac{2*precision*recall}{precision+recall} \quad (7)$$

Nilai F1-*Score* diperoleh dengan mengkalikan nilai dari *precision* dan *recall* dikali dengan 2, kemudian hasilnya dibagi dengan nilai dari *precision* ditambah dengan *recall. Precision* digunakan untuk mengukur pola positif yang diprediksi dengan benar dari total pola prediksi dalam kelas positif. *Recall* digunakan untuk mengukur pecahan pola positif yang diklasifikasikan dengan benar (Hossin and Sulaiman, 2015).

d. *Test Error*

Test error digunakan untuk mengukur rasio prediksi yang salah atas jumlah total kejadian yang dievaluasi. Formula dari pengukuran *test error* terlihat pada persamaan (8) (Hossin and Sulaiman, 2015).

$$Test\ Error = \frac{fp+fn}{tp+fp+tn+fn} \quad (8)$$

Untuk mendapatkan nilai *test error,* nilai *fp* ditambah dengan nilai *fn* kemudian dibagi dengan jumlah dari *tp, fp, tn,* dan *fn*, di mana *fp* adalah *false positive, fn* adalah *false negative, tp* adalah *true positive, tn* adalah *true negative.*

e. *Running Time*

Evaluasi ini digunakan untuk mengamati waktu yang dibutuhkan suatu algoritma dalam memproses data *train* dan juga dalam memproses data *test* hingga didapatkan hasil prediksinya.

2.7. *Results Compared*

Setelah didapatkan hasil evaluasi pada masing-masing algoritma, kemudian dilakukan perbandingan yang disajikan ke dalam bentuk tabel. Tabel tersebut berisi hasil evaluasi dari masing-masing algoritma yang dijalankan menggunakan MLlib Apache Spark. Evaluasi yang dilakukan adalah *accuracy, test error, f1-score,* dan *running time*. Hasil dari *confusion matrix* juga dibuat ke dalam tabel pada setiap algoritma yang digunakan.

3. HASIL DAN PEMBAHASAN

Setelah algoritma dilatih menggunakan data *train*, kemudian diuji coba dengan menggunakan data *test,* maka didapatkan hasil evaluasi dari *machine learning* menggunakan MLlib Apache Spark. Hasil dari keempat algoritma yang digunakan, yaitu Naive Bayes (NB), Gradient-Boosted Tree (GBT), Support Vector Machine (SVM), dan Logistic Regression (LR) ditunjukkan pada tabel 8.

Tabel 8. Hasil evaluasi *machine learning*

	Accuracy	***Test Error***	**F1-*Score***	***Running Time***
NB	0.833	0.166	0.834	16.3s
GBT	0.802	0.197	0.802	5m 15s
SVM	0.818	0.181	0.818	8.36s
LR	0.820	0.179	0.819	6.46s

Pada Naive Bayes didapatkan nilai *accuracy* sebesar 0.833, nilai *test error* sebesar 0.166, nilai *f1-score* sebesar 0.834, dan waktu *running time* 16.3 detik.

Pada GBT didapatkan nilai *accuracy* sebesar 0.802, nilai *test error* sebesar 0.197, nilai *f1-score* sebesar 0.802, dan waktu *running time* 5 menit 15 detik.

Pada SVM didapatkan nilai *accuracy* sebesar 0.818, nilai *test error* sebesar 0.181, nilai *f1-score* sebesar 0.818, dan waktu *running time* 8.36 detik.

Pada Logistic Regression didapatkan nilai *accuracy* sebesar 0.820, nilai *test error* sebesar 0.179, nilai *f1-score* sebesar 0.819, dan waktu *running time* 6.46 detik.

Evaluasi yang dihasilkan sudah cukup baik karena untuk parameter *accuracy,* semakin mendekati angka 1 maka akurasi tersebut semakin baik. Parameter *test error* dikatakan baik jika hasil evaluasi yang didapat mendekati angka 0. Parameter f1-*score* dikatakan baik jika hasil evaluasi yang didapat mendekati angka 1. Terakhir adalah parameter *running time*, semakin sedikit *running time* yang dibutuhkan atau semakin mendekati 0, maka hal tersebut semakin baik dalam melakukan pemrosesan data yang besar.

Hasil *confusion matrix* yang diperoleh pada Naive Bayes ditunjukkan pada tabel 9.

Tabel 9. Hasil *confusion matrix* algoritma NB

	Prediksi y=0 (fakta)	**Prediksi y=1 (berita palsu)**
Actual y=0 (fakta)	246	54
Actual y=1 (berita palsu)	34	193

Tabel 9 menunjukkan jumlah data fakta yang diprediksi benar sebagai fakta sebanyak 246, jumlah data fakta yang diprediksi salah sebagai berita palsu sebanyak 54, jumlah data berita palsu yang diprediksi salah sebagai fakta sebanyak 34, kemudian jumlah data berita palsu yang diprediksi benar sebagai berita palsu sebanyak 193.

Hasil *confusion matrix* yang diperoleh pada Gradient-Boosted Tree ditunjukkan pada tabel 10.

Tabel 10. Hasil *confusion matrix* algoritma GBT

	Prediksi y=0 (fakta)	**Prediksi y=1 (berita palsu)**
Actual y=0 (fakta)	252	55
Actual y=1 (berita palsu)	50	175

Tabel 10 menunjukkan jumlah data fakta yang diprediksi benar sebagai fakta sebanyak 252, jumlah data fakta yang diprediksi salah sebagai berita palsu sebanyak 55, jumlah data berita palsu yang diprediksi salah sebagai fakta sebanyak 50, kemudian jumlah data berita palsu yang diprediksi benar sebagai berita palsu sebanyak 175.

Hasil *confusion matrix* yang diperoleh pada SVM ditunjukkan pada tabel 11.

Tabel 11. Hasil *confusion matrix* algoritma SVM

	Prediksi y=0 (fakta)	**Prediksi y=1 (berita palsu)**
Actual y=0 (fakta)	267	46
Actual y=1 (berita palsu)	54	185

Tabel 11 menunjukkan jumlah data fakta yang diprediksi benar sebagai fakta sebanyak 267, jumlah data fakta yang diprediksi salah sebagai berita palsu sebanyak 46, jumlah data berita palsu yang diprediksi salah sebagai fakta sebanyak 54, kemudian jumlah data berita palsu yang diprediksi benar sebagai berita palsu sebanyak 185.

Hasil *confusion matrix* yang diperoleh pada Logistic Regression ditunjukkan pada tabel 12.

Tabel 12. Hasil *confusion matrix* algoritma LR

	Prediksi y=0 (fakta)	**Prediksi y=1 (berita palsu)**
Actual y=0 (fakta)	278	45
Actual y=1 (berita palsu)	58	193

Tabel 12 menunjukkan jumlah data fakta yang diprediksi benar sebagai fakta sebanyak 278, jumlah data fakta yang diprediksi salah sebagai berita palsu sebanyak 45, jumlah data berita palsu yang diprediksi salah sebagai fakta sebanyak 58, kemudian jumlah data berita palsu yang diprediksi benar sebagai berita palsu sebanyak 193.

Dalam melakukan klasifikasi, algoritma Naive Bayes sering memiliki tingkat akurasi yang lebih tinggi dibandingkan dengan algoritma klasifikasi lainnya. Hal ini disebabkan karena metode dari Naive Bayes hanya membutuhkan jumlah data pelatihan (*training data*) yang sedikit atau kecil dalam menentukan estimasi parameter yang diperlukan untuk proses pengklasifikasian. Pada Naive Bayes, berlaku asumsi independensi variabel, maka hanya varians dari suatu variabel dalam sebuah kelas yang dibutuhkan untuk menentukan klasifikasi, bukan keseluruhan dari matriks kovarians (Chauhan, 2018).

Pada algoritma GBT dalam melakukan pemrosesan data memiliki hasil dengan selisih waktu yang berbeda jauh dengan ketiga algoritma lainnya, hal ini disebabkan karena selama setiap pelatihan, tupel akan diputar ulang, sehingga model dapat mengurangi tingkat kesalahan. Namun, karena sifatnya yang iteratif tersebut, sehingga dibutuhkan waktu lebih lama untuk melatih model dan mendapatkan hasil prediksi (De'ath, 2007).

Dari hasil *confusion matrix* yang didapatkan, sebenarnya juga bisa dihitung nilai akurasi sesuai dengan formula dari *accuracy* seperti terlihat pada persamaan 8 yang menunjukkan contoh perhitungan akurasi berdasarkan *confusion matrix* pada Naive Bayes.

$$Accuracy = \frac{246+193}{246+54+34+193} = \frac{439}{527} = 0.833 \qquad (8)$$

Hasil ini membuktikan bahwa *confusion matrix* mempunyai peranan penting untuk menghitung evaluasi-evaluasi lain yang biasa digunakan dalam melakukan *machine learning* seperti f1-*score, precision, recall,* dan sebagainya.

Berdasarkan uraian hasil evaluasi dan pengujian pada masing-masing model menunjukkan bahwa model Naïve Bayes memiliki tingkat akurasi yang lebih baik daripada ketiga model lainnya dalam menentukan berita fakta atau berita palsu. Hal ini dibuktikan dengan skor akurasi dari Naïve Bayes adalah 0.833, kemudian diikuti dengan model Logistic Regression sebesar 0.820, SVM sebesar 0.818, dan GBT sebesar 0.802.

4. KESIMPULAN DAN SARAN

Penerapan MLlib Apache Spark telah berhasil dilakukan untuk pemrosesan *machine learning* menggunakan empat algoritma klasifikasi, yaitu Naive Bayes, Gradient-Boosted Tree, SVM dan Logistic Regression. MLlib terbukti dapat melakukan pemrosesan data dengan jumlah data 1786 dengan

kecepatan yang cukup baik sesuai dengan kelebihannya bekerja di dalam memori. Rata-rata waktu *running time* yang diperoleh dari keempat algoritma adalah 86.53 detik dan dengan *running time* tercepat yang didapat adalah 6.46 detik menggunakan Logistic Regression.

Nilai akurasi yang didapatkan dengan MLlib Apache Spark mampu memberikan hasil yang cukup baik dengan rata-rata *test error* dari keempat algoritma hanya 0.180. Nilai f1-*score* yang diperoleh pada keempat algoritma juga cukup baik dengan rata-rata sebesar 0.818. Selain itu, *confusion matrix* yang dihasilkan juga menunjukkan hasil yang baik karena jumlah *actual* yang diprediksi benar jauh lebih banyak dibandingkan dengan jumlah *actual* yang diprediksi salah.

Berdasarkan hasil yang diperoleh, membuktikan bahwa Mllib Apache Spark memiliki kinerja yang cepat dan keakuratan yang baik untuk melakukan pemrosesan data khususnya *machine learning*.

Untuk penelitian lebih lanjut, diharapkan dapat menggunakan jumlah data dan algoritma yang lebih banyak lagi agar dapat lebih membuktikan bahwa MLlib Apache Spark memiliki kecepatan dan akurasi tinggi untuk memproses data skala besar. Selain itu, perbandingan juga dapat dilakukan dengan membandingkan kinerja dari Mllib Apache Spark dengan alat bantu lainnya yang sebidang dan biasa digunakan untuk pemrosesan *machine learning.*

DAFTAR PUSTAKA

AL-SAQQA, S., AL-NAYMAT, G. & AWAJAN, A. 2018. A large-scale sentiment data classification for online reviews under apache spark. *Procedia Computer Science*. doi: 10.1016/j.procs.2018.10.166.

ALICE ZHENG. 2015. *Evaluating Machine Learning Models - O'Reilly Media*, *Oreilly*.

ASSEFI, M. *Dkk.* 2017. Big data machine learning using apache spark Mllib. *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, pp. 3492–3498. doi: 10.1109/BigData.2017.8258338.

CHAUHAN, G. 2018. *All about Naive Bayes - Towards Data Science*. Available at: https://towardsdatascience.com/all-about-naive-bayes-8e13cef044cf (Accessed: 17 June 2020).

CIOS, K. J. *dkk.* 2007. *Data mining: A knowledge discovery approach*, *Data Mining: A Knowledge Discovery Approach*. doi: 10.1007/978-0-387-36795-8.

CLASSIFICATION & REGRESSION - SPARK 2.4.6 DOCUMENTATION (no date). Available at: https://spark.apache.org/docs/latest/ml-classification-regression.html#gradient-boosted-tree-classifier (Accessed: 16 June 2020).

DE'ATH, G. 2007. Boosted trees for ecological modeling and prediction', *Ecology*. doi: 10.1890/0012-9658(2007)88[243:BTFEMA]2.0.CO;2.

DESHAI, N., SEKHAR, B. V. D. S. & VENKATARAMANA, S. 2019. Mllib: machine learning in apache spark. *International Journal of Recent Technology and Engineering*.

ETAIWI, W., BILTAWI, M. & NAYMAT, G. 2017. Evaluation of classification algorithms for banking customer's behavior under Apache Spark Data Processing System. *Procedia Computer Science*. doi: 10.1016/j.procs.2017.08.280.

FU, J., SUN, J. & WANG, K. 2017. SPARK-A Big Data Processing Platform for Machine Learning', in *Proceedings - 2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration. ICIICII 2016*. doi: 10.1109/ICIICII.2016.0023.

GANESAN, K. & SUBOTIN, M. 2014. A general supervised approach to segmentation of clinical texts', in *Proceedings - 2014 IEEE International Conference on Big Data, IEEE Big Data 2014*. doi: 10.1109/BigData.2014.7004390.

HOSSIN, M. & SULAIMAN, M. N. 2015. A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*. doi: 10.5121/ijdkp.2015.5201.

ID-STOPWORDS/ID.STOPWORDS.02.01.2016.TXT AT MASTER · MASDEVID/ID-STOPWORDS · GITHUB (no date). Available at: https://github.com/masdevid/ID-stopwords/blob/master/id.stopwords.02.01.2016.txt (Accessed: 16 June 2020).

JAMIL, L. S. 2016. Data Analysis Based on Data Mining Algorithms Using Weka Workbench. *International Journal of Engineering Sciences & Research Technology*. doi: 10.5281/zenodo.59630.

JURAFSKY, D. & MARTIN, J. H. 2019. Logistic Regression', in *Speech and Language Processing*. 3rd edn, pp. 75–92.

JUWIANTHO, H. *ET AL.* 2020. Sentiment Analysis Twitter Bahasa Indonesia Berbasis Word2vec Menggunakan Deep Convolutional Neural Network. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*, 7(1), pp. 181–188. doi: 10.25126/jtiik.202071758.

LESKOVEC, J., RAJARAMAN, A. &ULLMAN, J.

D. 2014. *Mining of massive datasets: Second edition*, *Mining of Massive Datasets: Second Edition*. doi: 10.1017/CBO9781139924801.

MENG, X. *ET AL*. 2016. MLlib: Machine learning in Apache Spark. *Journal of Machine Learning Research*.

SOPYŁA, K. & DROZDA, P. 2015. Stochastic gradient descent with Barzilai-Borwein update step for SVM. *Information Sciences*. doi: 10.1016/j.ins.2015.03.073.

SUYANTO., 2017. *Data mining Untuk Klasifikasi dan Klasterisasi Data*, *SpringerReference*. doi: 10.1007/SpringerReference_5414.

TENTANG KAMI – TURNBACKHOAX, 2016. Available at: https://turnbackhoax.id/tentang-kami/ (Accessed: 14 June 2020).

TRIPATHY, A., AGRAWAL, A. & RATH, S. K. 2015. Classification of Sentimental Reviews Using Machine Learning Techniques', in *Procedia Computer Science*. doi: 10.1016/j.procs.2015.07.523.

YASROBI, S. dkk. 2017. Performance Analysis of Sparks Machine Learning Library. *Transactions on Machine Learning and Data Mining*, 10(2).