BAB II

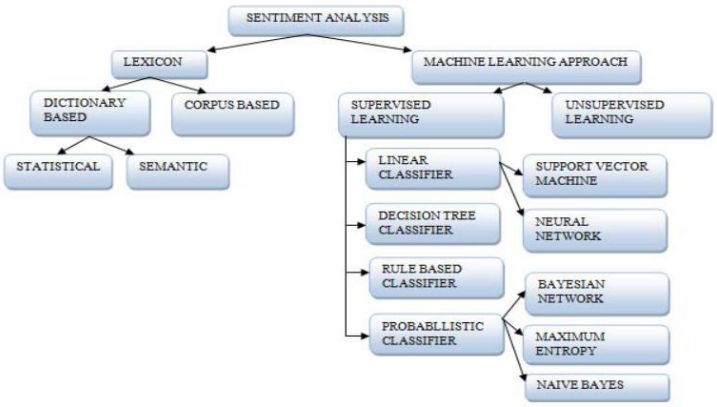
LANDASAN TEORI

2.1. Teori Terkait

2.1.1. Analisis Sentimen (*Sentiment Analisis*)

Analisis sentimen adalah bidang studi yang menganalisis pendapat, sentimen, penilaian, sikap dan emosi seseorang terhadap suatu entitas dan atributnya yang diungkapkan dalam sebuah teks tertulis (Liu, 2015). Entitas yang dimaksud dapat berupa produk, layanan, organisasi, individu, kejadian, isu, atau sebuah topik.

Menurut (Kaur, Mangat dan Nidhi, 2017), analisis sentimen diaplikasikan sebagai: a) Dukungan dalam pengambilan keputusan, seperti "buku apa yang harus dibeli", "hotel mana yang dituju", dan "film apa yang harus ditonton". b) Dalam dunia bisnis digunakan untuk penilaian individu terhadap sebuah produk, Google *Product Search* adalah salah satu ilustrasinya. c) Prediksi dan analisis tren: dengan analisis sentimen, seseorang dapat memprediksi tren pasar dengan melacak penilaian publik. Teknologi fundamental dalam aplikasi *sentiment analysis* dan *opinion-mining* yang sering digunakan saat ini adalah klasifikasi (Pang dan Lee, 2008). (Kaur, Mangat dan Nidhi, 2017) mengemukakan terdapat dua pendekatan yang sering digunakan dalam klasifikasi sentimen, yaitu *Subjective Lexicon* dan *Machine Learning* yang ditunjukkan pada Gambar 2.1.



Gambar 2.1 *Pendekatan Sentiment Classification* (Kaur, Mangat dan Nidhi, 2017)

Subjective Lexicon adalah kumpulan kata yang setiap kata memiliki skor yang menunjukkan sifat teks yang positif, negatif, netral dan objektif. Dalam pendekatan ini, untuk bagian teks tertentu, agregasi skor kata subjektif dilakukan, yaitu skor kata positif, negatif, netral, dan objektif dijumlahkan secara terpisah. Sedangkan pendekatan *machine learning* merupakan klasifikasi otomatis. Klasifikasi dilakukan dengan menggunakan fitur teks yang diekstraksi dari teks. Pendekatan *machine learning* ini terdiri dari 2 jenis, yaitu *supervised* dan *unsupervised*.

(Hikmawan, Pardamean dan Khasanah, 2020) mengemukakan proses klasifikasi *sentiment analysis* terdiri dari pengumpulan data, *preprocessing* data, pemodelan, validasi, dan hasil. Pengumpulan data dapat dilakukan dari berbagai sumber, contohnya adalah *Twitter*. Pengumpulan data dari *Twitter* dilakukan berdasarkan kata kunci tertentu. Tahapan *prepocessing* dilakukan untuk menyiapkan data yang baik dan sesuai untuk digunakan pada tahapan *modeling*. Semakin baik tahapan *preprocessing*, maka nilai akurasi yang didapatkan pada tahap *modeling* akan semakin tinggi. Dalam *sentiment analysis*, tahapan yang

sering digunakan pada *preprocessing* adalah *tokenizing*, *stopword removal*, dan *stemming*. Pada tahapan pemodelan, data digunakan sebagai *data training* untuk sebuah algoritma klasifikasi. Tahap evaluasi dilakukan untuk mengevaluasi performa dari model yang telah dibuat. Biasanya pada tahap ini menghasilkan nilai akurasi dari model yang telah dibuat. Jika akurasi belum mencapai target, model akan dievaluasi kembali untuk mendapatkan akurasi yang lebih baik.

2.1.2. Term Frequency–Inverse Document Frequency (TF-IDF)

Term Frequency–Inverse Document Frequency (TF-IDF) merupakan statistik numerik untuk mencerminkan pentingnya sebuah kata pada dokumen dalam sebuah kumpulan dokumen atau corpus (Rajaraman dan Ullman, 2011). TF-IDF terdiri dari *Term Frequency* (TF) dan *Inverse Document Frequency* (IDF).

Term Frequency (TF) merupakan jumlah frekuensi kemunculan kata dalam sebuah dokumen, di mana jumlah suatu kata yang muncul pada dokumen dibagi dengan jumlah kata yang terdapat pada dokumen, yang dirumuskan sebagai:

$$TF_{t,d} = \frac{f_{t,d}}{jumlah\ kata\ pada\ d} \tag{2.1}$$

di mana $f_{t,j}$ merupakan jumlah kata t yang muncul pada dokumen d.

Inverse Document Frequency (IDF) merupakan ukuran berapa banyak informasi yang diberikan oleh sebuah kata. Berbeda dengan *IDF* tradisional, *IDF sklearn* menggunakan konstanta kesatuan dalam penyebut dan pembilang. (Pedregosa *dkk.*, 2011) yang dirumuskan sebagai:

$$idf(t) = \log \frac{1+n}{1+df(t)} + 1 \tag{2.2}$$

di mana n merupakan jumlah dokumen dalam sebuah korpus dan $df(t)$ merupakan jumlah munculnya kata i pada seluruh dokumen. Sehingga TF-IDF dirumuskan sebagai:

$$TF.IDF_{(i,j,k)} = TF_{ij} \times IDF_i \quad (2.3)$$

2.1.3. *Machine Learning*

Machine Learning adalah bagian dari *Artificial Intelegence* (AI) yang memungkinkan sebuah sistem untuk belajar dari data, bukan melalui pemrograman eksplisit (Hurwitz dan Kirsch, 2018). *Machine Learning* merupakan sebuah sistem yang dapat belajar dengan sendirinya. Sistem tersebut dapat memutuskan sesuatu dengan sendirinya tanpa harus ada campur tangan manusia. Hal ini menyebabkan komputer menjadi semakin pintar karena dapat belajar sendiri dari data yang dimilikinya.

Dalam *machine learning*, data berperan sebagai bahan *input* untuk belajar (*training*) mengenai sesuatu untuk menghasilkan analisis yang benar. Data pada *machine learning* biasanya terbagi menjadi dua bagian, yaitu *data training* dan *data testing*. *Data training* digunakan untuk melatih algoritma *machine learning* sedangkan *data testing* digunakan untuk mengetahui performa dari algoritma yang digunakan. Dalam beberapa kasus terdapat *data validation* yang berperan sebagai bahan evaluasi untuk algoritma apabila hasil kurang maksimal.

Teknik *machine learning* dibagi menjadi beberapa macam, yaitu *supervised learning, unsupervised learning, reinforcement learning,* serta *neural network* dan *deep learning* (Hurwitz dan Kirsch, 2018). Pada *supervised learning,* data memiliki fitur berlabel yang mendefinisikan arti data. Sedangkan pada *unsupervised*

learning, data tidak memiliki label. Teknik *unsupervised learning* memungkinkan mengklasifikasikan data berdasarkan klaster yang ditemukan. *Reinforcement learning* menerima umpan balik dari analisis data sehingga pengguna dipandu ke hasil terbaik. Teknik ini belajar melalui *trial* dan *error* sehingga keputusan yang berhasil akan menghasilkan proses yang diperkuat. *Deep learning* adalah metode *machine learning* khusus yang menggabungkan jaringan saraf dalam lapisan yang berurutan untuk belajar dari data secara berulang yang memungkinkan menyelesaikan permasalahan dari data yang tidak terstruktur.

2.1.4. *Logistic Regression*

Logistic Regression merupakan algoritma *machine learning* yang terinspirasi dari algoritma *Linear Regression* yang memiliki konsep yang sama dengan model regresi lain yaitu menemukan hubungan antara variabel hasil dan satu set variabel independen. Yang membedakan *Logistic Regression* dengan *Linear Regression* adalah variabel hasil dalam *Logistic Regression* bersifat biner atau dikotomi (Hosmer, Lemeshow dan Sturdivant, 2013).

Logistic Regression dapat menangani model yang berisi variabel tunggal maupun variabel ganda. Hal ini yang menjadi kelebihan dari *Logistic Regression*. Untuk menangani kasus variabel ganda, *Logit* dari *Logistic Regression* didefinisikan sebagai (Hosmer, Lemeshow dan Sturdivant, 2013):

$$g(x) = ln\left(\frac{\pi(x)}{1-\pi(x)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p \qquad (2.4)$$

Untuk permasalahan multi-kelas, dapat menggunakan *Multinomial Logistic Regression*. Fungsi *Multinomial Logistic Regression* terdiri dari 2 lapisan

fungsional, yaitu fungsi *linear prediction* yang didefinisikan sebagai (Sharma, 2020):

$$Z_{[k\times1]} = W_{[k\times m]} \cdot X_{[m\times1]} + B_{[k\times1]} \quad (2.5)$$

serta fungsi *softmax* yang didefinisikan sebagai (Goodfellow, Bengio dan Courville, 2016):

$$softmax(Z)_i = \frac{\exp(Z_i)}{\sum_j \exp(Z_i)} \quad (2.6)$$

2.1.5. *Random Forest*

Random Forest merupakan kombinasi beberapa *tree predictor* sehingga setiap pohon bergantung pana nilai vektor acak yang diambil sampelnya secara independen dan dengan distribusi yang sama untuk semua pohon di dalam hutan (Breiman, 2001). Menurut (Singh dan Sarraf, 2020), *random forest* memiliki dua tipe keacakan, yaitu *bagging* dan *bootstrapping* yang merupakan konsep utama *random forest classifier.* Berikut ini merupakan algoritma dari *random forest*:

Input : Jumlah pohon = B, Data latih = N, total fitur, f = subset fitur.

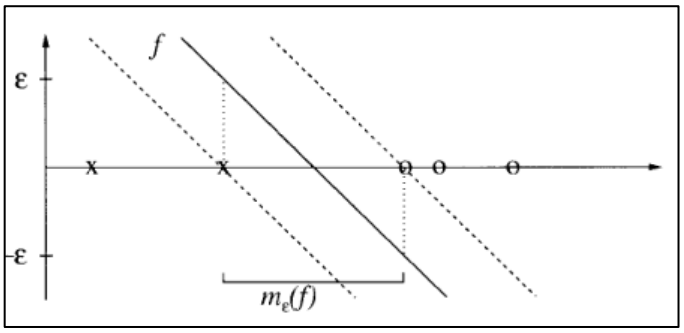
Output : Untuk data masukan, tingkat kelas yang dikantongi

1. Menganalisis setiap pohon pada hutan B:
    b. Pemilihan sampel *bootstrap* S dengan ukuran N dari data latih.
    c. Pembuatan pohon Tb mengulangi langkah – langkah berikut secara rekursif:
        i. Memilih secara acak f dari F.
        ii. Pemilihan nilai F yang terbaik.
        iii. Pemisahan node.

2. Setelah membuat pohon B, contoh pengujian akan diteruskan ke setiap pohon serta akan ada penetapan label kelas berdasarkan suara terbanyak.

2.1.6 *Support Vector Machine*

Support Vector Machine (SVM) adalah suatu teknik untuk melakukan sebuah prediksi, baik klasifikasi maupun regresi yang terinspirasi dari teori pembelajaran statistik (Vapnik, 1999). SVM membangun *hyper-plane* atau set *hyper-plane* dalam ruang dimensi tinggi atau tak terbatas, yang dapat digunakan untuk klasifikasi, regresi atau tugas lainnya (Pedregosa *dkk.*, 2011) . Secara intuitif, pemisahan yang baik dicapai oleh *hyper-plane* yang memiliki jarak terbesar ke titik data pelatihan terdekat dari kelas mana pun (disebut margin fungsional), karena secara umum semakin besar margin, semakin rendah kesalahan generalisasi pengklasifikasi (Pedregosa *dkk.*, 2011).



Gambar 2.2 *Hyper-plane pada SVM* (Schölkopf *dkk.*, 2000)

Dalam kasus klasifikasi linier, SVM membangun persamaan linear dengan memaksimalkan margin $m_\varepsilon(f)$ dan meminimalkan $|w|$ (Schölkopf *dkk.*, 2000). Persamaan linear tersebut didefinisikan sebagai:

$$f(x) = w \cdot x + c \qquad (2.7)$$

Keputusan yang diambil pada saat klasifikasi didapatkan dari fungsi (Gandhi, 2018):

$$f^*(x) = sgn((w \cdot x) - b) \tag{2.8}$$

Tujuan dari proses *training* SVM adalah untuk memaksimalkan margin antara poin data dengan *hyperplane* (Gandhi, 2018). Fungsi kerugian yang dapat memaksimalkan margin adalah *hinge function* yang didefinisikan sebagai (Gandhi, 2018):

$$c(x, y, f(x)) = \begin{cases} 0, & if\ y * f(x) \geq 1 \\ 1 - y * f(x), & else \end{cases} \tag{2.9}$$

atau

$$c\big(x, y, f(x)\big) = (1 - y * f(x))_+ \tag{2.10}$$

Biaya 0 jika nilai prediksi dan nilai aktual sama, jika tidak maka dihitung nilai kerugiannya. Parameter regulasi (λ) ditambahkan pada fungsi biaya dengan tujuan untuk menyeimbangkan maksimalisasi margin dan kerugian, sehingga fungsi biaya menjadi (Gandhi, 2018)ga:

$$min_w \lambda \|w\|^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle_+ \tag{2.11}$$

Kemudian menggunakan *gradient* untuk memperbaharui nilai *weight* :

$$\frac{\delta}{\delta w_k}(1 - y_i \langle x_i, w \rangle_+ = \begin{cases} 0, & if\ y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & else \end{cases} \tag{2.12}$$

Optimasi nilai *weight* pada proses *training* menggunakan fungsi:

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda \mathrm{w}) \tag{2.13}$$

jika terjadi *missclassification.* Serta menggunakan fungsi:

$$w = w - \alpha \cdot (2\lambda \mathrm{w}) \tag{2.14}$$

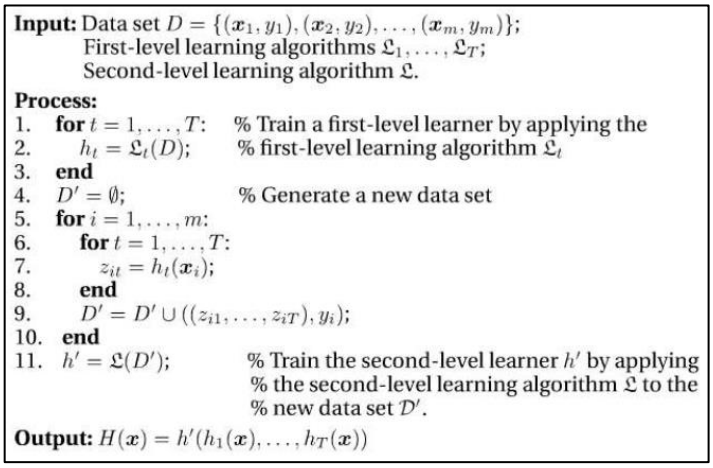
jika tidak terjadi *missclassification.*

2.1.7 *eXtreme Gradient Boosting* (*XGBoost*)

eXtreme Gradient Boosting (XGBoost) merupakan sistem *machine learning* yang dapat diskalakan untuk *tree boosting* yang tersedia sebagai paket *open souce* (Chen dan Guestrin, 2016). Sistem ini telah diakui secara luas dalam sejumlah kompetisi *machine learning*. Contohnya, kompetisi pada situs Kaggle yang diantara 29 solusi yang diterbitkan pada Kaggle selama 2015, 17 solusi menggunakan *XGBoost*. Hal yang menjadi faktor suksesnya *XGBoost* adalah *scalability* dalam berbagai skenario yang dibuktikan dengan sistem yang dibangun 10 kali lebih cepat dibandingkan dengan solusi – solusi yang populer (Chen dan Guestrin, 2016).

2.1.8 *Stacking Ensemble*

Stacking Ensemble merupakan sebuah teknik mengombinasikan beberapa algoritma berdasarkan pembelajaran. *Stacking* adalah prosedur umum di mana pelajar (*learner*) dilatih untuk menggabungkan pelajar individu (*individual learner*) (Zhou, 2012). *Individual learner* disebut dengan *first-level learners*, sedangkan yang mengombinasi disebut dengan *second-level learner* atau *meta-learner*. Proses *stacking* secara sederhana yaitu dataset asli dipelajari oleh *first-level learners* yang menghasilkan dataset baru yang akan digunakan sebagai input untuk *second-level learner*. *Pseudo code* prosedur umum dari *stacking* ditunjukkan pada Gambar 2.3.



Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
First-level learning algorithms $\mathfrak{L}_1, \ldots, \mathfrak{L}_T$;
Second-level learning algorithm $\mathfrak{L}$.

Process:

for $t = 1, \ldots, T$: % Train a first-level learner by applying the
$h_t = \mathfrak{L}_t(D)$; % first-level learning algorithm $\mathfrak{L}_t$
end
$D' = \emptyset$; % Generate a new data set
for $i = 1, \ldots, m$:
for $t = 1, \ldots, T$:
$z_{it} = h_t(\boldsymbol{x}_i)$;
end
$D' = D' \cup ((z_{i1}, \ldots, z_{iT}), y_i)$;
end
$h' = \mathfrak{L}(D')$; % Train the second-level learner h' by applying
% the second-level learning algorithm $\mathfrak{L}$ to the
% new data set D'.

Output: $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), \ldots, h_T(\boldsymbol{x}))$

Gambar 2.3 *Pseudo Code Prosedur Umum Stacking* (Zhou, 2012)

2.1.9 *Cross Validation*

Cross Validation (CV) merupakan suatu teknik validasi dalam *machine learning*. *Cross validation* adalah metode statistik untuk mengevaluasi dan membandingkan algoritme pembelajaran dengan membagi data menjadi dua segmen: satu digunakan untuk mempelajari atau melatih model dan yang lainnya digunakan untuk memvalidasi model (Refaeilzadeh, Tang dan Liu, 2016).

Bentuk dasar dari *cross validation* adalah *k-fold cross-validation*. Dalam *k-fold cross-validation*, data pertama kali dipartisi menjadi k segmen atau lipatan berukuran sama (atau hampir sama). Selanjutnya k iterasi pelatihan dan validasi dilakukan sedemikian rupa sehingga dalam setiap iterasi diadakan lipatan data yang berbeda untuk validasi, sedangkan sisa k - 1 lipatan digunakan untuk pembelajaran.

Salah satu model *k-fold cross validation* yang sering digunakan adalah *Stratified k-fold*. *Stratified k-fold* merupakan variasi dari *k-fold* dengan mengembalikan lipatan yang bertingkat. Setiap set berisi persentase sampel yang sama dari setiap kelas target sebagai set lengkap (Pedregosa *dkk.*, 2011).

2.1.10 *Leaning Curve*

Learning Curve merupakan sebuah kurva yang menunjukkan tentang ketergantungan kinerja generalisasi *learner* pada ukuran set pelatihan (Viering dan Loog, 2021). *Learning Curve* dapat digunakan dalam pemilihan model, untuk memprediksi efek dari banyaknya data dalam pelatihan, dan untuk mengurangi kompleksitas komputasi pelatihan model serta penyetelan hyperparameter (Viering dan Loog, 2021).

2.1.11 *Confusion Matrix*

Confusion matrix merupakan suatu metode yang biasa digunakan untuk mengukur akurasi pada data mining. *Confusion matrix* meringkas kinerja klasifikasi *clasifier* sehubungan dengan beberapa data pengujian (Ting, 2010). Matriks ini merupakan matriks dua dimensi berupa tabel matriks, jika data set hanya terdiri dari dua kelas, kelas yang satu dianggap sebagai positif dan yang lainnya negatif.

Tabel 2.1 *Confusion Matrix*

Klasifikasi yang benar	**Diklasifikasikan sebagai**	
	+	-
+	*True Positive*	*False Negative*
-	*False Positive*	*True Negative*

Keterangan :

1. *True positive* : jumlah *record* positif yang diklasifikasikan sebagai positif,
2. *False positive* : jumlah *record* negatif yang diklasifikasikan sebagai positif,

3. *False negative* : jumlah *record* positif yang diklasifikasikan sebagai negatif,
4. *True negative* : jumlah *record* negatif yang diklasifikasikan sebagai *negatif*.

Dari tabel *confusion matrix* di atas dapat dihitung *accuracy*, *precision, recall*, dan *f1 – score* dengan rumus (Han, Kamber dan Pei, 2012):

$$accuracy = \frac{TP + TN}{P + N} \tag{2.6}$$

$$precision = \frac{TP}{TP + FP} \tag{2.7}$$

$$recall = \frac{TP}{TP + FN} \tag{2.8}$$

$$f1 - score = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.9}$$

2.2. Pekerjaan Terkait

2.2.1. *State of The Art* Bidang Penelitian

Berdasarkan hasil kajian terhadap pekerjaan – pekerjaan terkait terdapat berbagai macam algoritma dan metode yang digunakan untuk klasifikasi sentimen. Dari penelitian – penelitian tersebut terdapat persamaan dan perbedaan dari masing – masing penelitian yang dapat dilihat dari penggunaan metode dan algoritmanya.

(Hikmawan, Pardamean dan Khasanah, 2020) menganalisis dan membandingkan algoritma *Naive Bayers*, *Support Vector Machine*, dan *k-NN* untuk analisis sentimen publik Joko Widodo terhadap wabah COVID-19 dengan menggunakan *Gata Framework* untuk tahap *preprocessing*. Hasilnya, algoritma

yang mendapatkan performa yang baik adalah *Support Vector Machine* dengan nilai *accuracy* 84.58%, *precision* 82.14%, dan *recall* 85.82%.

(Ritonga *dkk.*, 2021) menggunakan algoritma *Naive Bayes* untuk analisis sentimen vaksin COVID-19 di Indonesia. Hasil pengukuran sentimen menunjukkan lebih dari 3.4 ribu *tweet* (56%) memiliki sentimen negatif, lebih dari 2,4 ribu *tweet* (39%) memiliki sentimen positif, dan 301 *tweet* (1%) memiliki sentimen netral.

(Dubey, 2021) melakukan analisis sentimen terhadap vaksin COVID-19 di India yang hasilnya menyimpulkan bahwa untuk vaksin Covaxin memiliki 36% *tweet* positif dan 31% *tweet* negatif, sedangkan vaksin Covishield memiliki 71% *tweet* positif dan 29% *tweet* negatif.

(Sarkar, 2020) menggunakan metode *stacked ensemble* dengan algoritma *Naive Bayes* dan *Support Vector Machine* untuk mendeteksi sentimen polaritas pada *tweet* Bengali. Hasilnya model *stacked ensemble* yang diusulkan mendapatkan akurasi mencapai 56.2% yang melebihi algoritma LSTM yang mendapatkan akurasi 55.27%.

(Giovani *dkk.*, 2020) menggunakan algoritma PSO berbasis *Support Vector Machine* untuk analisis sentimen komentar aplikasi Ruang Guru yang menghasilkan nilai akurasi sebesar 78.55% dan nilai AUC sebesar 0.853.

(Rachman dan Pramana, 2020) melakukan analisis sentimen pro dan kontra masyarakat Indonesia tentang vaksin COVID-19 pada media sosial *Twitter* menggunakan metode *Latent Dirichlet Allocation* (LDA).

(Hayatin, Marthasari dan Nuarini, 2020) melakukan optimasi analisis sentimen terhadap pemilihan presiden tahun 2019 dengan algoritma *Naive Bayes*

dan *Particle Swarm Optimization* (PSO) yang menghasilkan akurasi sebear 90.74% yang meningkat sebesar 4.12% dari algoritma *Naive Bayes* saja.

(Xia, Chen dan Yang, 2021) mengusulkan metode baru dengan *Weighted Classifier Selection* dan *Stacked Ensemble* untuk klasifikasi *multi-label* yang menghasilkan nilai akurasi 0.909 untuk model MLWSE-L1 dan 0.91 untuk model MLWSE-L2.

(Meel *dkk.*, 2020) melakukan analisis konten web untuk berita palsu menggunakan metode *Max Voting* dan *Stacking Ensemble Classifier*. Hasilnya, kombinasi algoritma *Logistic Regression*, *Support Vector Machine,* dan *Random Forest* pada metode *Stacking Ensemble Classifier* mendapatkan performa yang terbaik dengan nilai *accuracy* 98.37%, *precision* 0.97, *recall* 0.99, *f1-score* 0.98, dan AUC 0.9888. Sedangkan pada dataset 2 mendapat nilai *accuracy* 94.32%, *precision* 0.95, *recall* 0.93, *f1-score* 0.94, dan AUC 0.9853. Nilai ini lebih baik daripada menggunakan *Max Voting ensemble* yang mendapatkan *accuracy* sebesar 97.89%, *precision* 0.97, *recall* 0.99, *f1-score* 0.98, dan AUC 0.9683 untuk dataset 1, serta akurasi sebesar 93.85%, *precision* 0.94, *recall* 0.90, *f1-score* 0.92, dan AUC 0.9625 untuk dataset 2.

(Adrian *dkk.*, 2021) melakukan perbandingan algoritma *Random Forest* dan *Support Vector Machine* untuk analisis sentimen PSBB. Hasilnya, nilai akurasi yang didapatkan algoritma *Random Forest* mencapai 58% dengan nilai *precision, recall,* serta *f1-score* masing – masing adalah 35%, 58%, dan 44%. Sedangkan algoritma *Support Vector Machine* yaitu 56% dengan nilai *precision, recall,* serta *f1-score* masing – masing adalah 38%, 56%, dan 44%. Nilai performa kedua

algoritma tersebut sangat kecil yang dikarenakan penggunaan data yang sangat sedikit, yaitu hanya 466 data *tweet*.

(Santoso *dkk.*, 2021) menggunakan algoritma *Logistic Regression* untuk klasifikasi persepsi pengguna *Twitter* terhadap kasus COVID-19 yang menghasilkan nilai akurasi 77% dan *f1-score* 74% untuk *hyperparameter* L2, dan akurasi sebesar 74% dengan *f1-score* 70% untuk *hyperparameter None*.

Pada *state of the art* tersebut belum ada penelitian yang mengimplementasikan metode *stacking ensemble classifier* untuk klasifikasi multi kelas pada studi kasus analisis sentimen vaksin covid-19 di Indonesia. Dalam beberapa penelitian di atas, metode *stacking ensemble classifier* memiliki performa yang lebih baik dari metode atau algoritma yang lain. Pada penelitian ini, metode *stacking ensemble classifier* akan diimplementasikan untuk klasifikasi sentimen multi kelas terhadap vaksin covid-19 di Indonesia pada media sosial *Twitter*.

Deskripsi rinci tentang pekerjaan terkait dapat dilihat dalam Tabel *State of The Art* Penelitian Terkait pada Lampiran 1.

2.2.1. Matriks Penelitian

Matriks penelitian menunjukkan persamaan dan perbedaan penelitian – penelitian sebelumnya dengan penelitian ini. Tabel 3.1 menunjukkan persamaan dan perbedaan penelitian ini dengan penelitian – penelitian sebelumnya yang terdiri dari algoritma atau metode yang digunakan jenis klasifikasi yang dilakukan, serta objek penelitian.

Tabel 3.1 Matriks Penelitian

No.	Penulis/ Tahun	Judul	Ruang Lingkup									
			Algoritma/Metode						Tujuan Klasifikasi		Objek Covid-19	
			Naive Bayes	**RF**	**LR**	**SVM**	**XGB**	***Stacking Ensemble***	***Binary-class***	***Multi-class***	**Kasus**	**Vaksin**
1.	(Hikmawan, Pardamean dan Khasanah, 2020)	Sentimen Analisis Publik Terhadap Joko Widodo Terhadap Wabah Covid-19 Menggunakan	√	√	-	√	-	-	-	√	√	-

		Metode Machine Learning										
2.	(Ritonga *dkk.*, 2021)	*Sentiment analysis of COVID-19 vaccine in Indonesia using Naïve Bayes Algorithm*	√	-	-	-	-	-	-	√	√	-
3.	(Sarkar, 2020)	*A Stacked Ensemble Approach to*	√	-		√	-	√	-	√	-	-

		Bengali Sentiment Analysis										
4.	(Giovani *dkk.*, 2020)	Analisis Sentimen Aplikasi Ruang Guru di Twitter Menggunakan Algoritma Klasifikasi	-	-	-	√	-	-	√	-	-	-
5.	(Hayatin, Marthasari	*Optimization of Sentiment Analysis for*	√	-	-	-	-	-	√	-	-	-

	dan Nuarini, 2020)	*Indonesian Presidential Election using Naïve Bayes and Particle Swarm Optimization*										
6.	(Meel *dkk.*, 2020)	*Web Text Content Credibility Analysis using Max Voting and Stacking*	√	√	√	√		√	√	-	-	-

		Ensemble Classifiers										
7.	(Adrian *dkk.*, 2021)	*Perbandingan Metode Klasifikasi Random Forest dan SVM Pada Analisis Sentimen PSBB*	-	√	-	√	-	-	-	√	-	-
8.	(Santoso *dkk.*, 2021)	*Klasifikasi Persepsi Pengguna*	-	-	√	-	-	-	-	√	√	-

		Twitter Terhadap Kasus Covid-19 Menggunakan Metode Logistic Regression										
9.	(Rama, 2021)	Implementasi *Stacking Ensemble* untuk Klasifikasi Sentimen	-	√	√	√	√	√	-	√	-	√

		Terhadap Topik Vaksin Covid-19 di Indonesia pada Media Sosial *Twitter*.										

2.2.2. Relevansi Penelitian

Relevansi penelitian menunjukkan perbandingan penelitian ini dengan penelitian sebelumnya yang paling relevan. Penelitian ini paling relevan dengan penelitian (Santoso *dkk.*, 2021) dengan beberapa persamaan dan beberapa improvisasi yang dilakukan pada penelitian ini. Tabel 3.2 merupakan rincian relevansi penelitian ini.

Tabel 2.2 Relevansi Penelitian

Peneliti	**(Santoso *dkk.*, 2021)**	**(Rama, 2021)**
Judul	*Klasifikasi Persepsi Pengguna Twitter Terhadap Kasus Covid-19 Menggunakan Metode Logistic Regression*	Implementasi *Stacking Ensemble* untuk Klasifikasi Sentimen Terhadap Topik Vaksin Covid-19 di Indonesia pada Media Sosial *Twitter*.
Masalah Penelitian	Membandingkan algoritma *Logistic Regression* dengan *hyperparameter* L2 dan *None* untuk klasifikasi *multi-class* kasus covid-19 pada media sosial *twitter*	Mengimplementasikan metode *Stacking Ensemble Classifier* untuk klasifikasi *multi-class* vaksin covid-19 di Indonesia pada media sosial *Twitter*.
Objek Penelitian	Kasifikasi *multi-class* pada media sosial *twitter* mengenai kasus covid-19	Klasifikasi *multi-class* sentimen topik vaksin covid-19 di

		Indonesia pada media sosial *twitter*.
Algoritma / Metode	Algoritma *Logistic Regression* dengan *hyperparameter* L2 dan *None*	*Stacking Ensemble Classifier* menggunakan algoritma *Logistic Regression, Support Vector Machine, Random Forest,* dan *XGBoost*
Implementasi	Klasifikasi *multi-class* dilakukan dengan bahasa pemrograman *python.*	Klasifikasi *multi-class* dilakukan dengan bahasa pemrograman *python.*