BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian terdahulu memiliki peran penting dalam memberikan konteks, justifikasi, dan pemahaman mendalam. Selain itu, penelitian terdahulu juga menunjukkan hubungan antara penelitian baru dengan penelitian yang yang sudah ada sebelumnya. Hasil dan temuan dari penelitian terdahulu juga dipertimbangkan sehingga dapat digunakan dalam membantu penulis mengevaluasi kontribusi yang dibawa oleh penelitian ini. Tabel 2. 1 merupakan daftar penelitian terdahulu yang digunakan dalam penelitian ini.

Tabel 2. 1 Tabel Penelitian Terdahulu

Penulis	Judul	Jurnal	Metode	Hasil
D. Kristiyanti, Normah, A. Umam [36]	*Prediction of Indonesia Presidential Election Results for the 2019-2024 Period Using Twitter Sentiment Analysis*	*Proceedings of 2019 5th International Conference on New Media Studies, CONMEDIA 2019* (2019)	PSO-SVM, GA-SVM	Model PSO-SVM berkinjerja paling optimal (Akurasi: 82.85% dan 86.20%)
P. Arunkumar, S. Chandramathi, S. Kannimuthu [40]	*Sentiment Analysis-Based Framework for Assessing Internet Telemedicine Videos*	*International Journal of Data Analysis Techniques and Strategies* (2019)	*Bayes Net*, SVM, C.4.5, KNN, PSO-SVM	Model PSO-SVM bekinerja paling optimal (Akurasi: 80.3%)
P. Windha Mega, Haryoko [41]	*Improved Support Vector Machine* (SVM) *Performance on Go-Jek Service Review Classification Using Particle Swarm Optimization*	*Proceedings - 4th International Conference on Informatics, Multimedia, Cyber and Information System, ICIMCIS* 2022 (2022)	GA-SVM, PSO-SVM	Model PSO-SVM berkinerja paling optimal (Akurasi: 87%)
Warjiyono, S. Aji, Fandhilah, N. Hidayatun, H. Faqih, Liesnaningsih [42]	*The Sentiment Analysis of Fintech Users Using Support Vector Machine and Particle Swarm Optimization Method*	*2019 7th International Conference on Cyber and IT Service Management, CITSM 2019* (2019)	SVM, PSO-SVM	Model PSO-SVM berkinerja paling optimal (Akurasi: 82.33%)
K. Sigit, A. Dewi, G. Windu, Nurmalasari, T. Muhamad,	*Comparison of Classification Methods on Sentiment Analysis of Political Figure*	*IOP Conference Series: Materials Science and Engineering* (2019)	SVM, PSO-SVM	Model PSO-SVM berkinerja paling optimal (Akurasi: 78.40%)

Penulis	Judul	Jurnal	Metode	Hasil
N. Kadinar [43]	*Electability Based on Public Comments on Online News Media Sites*			
D. Sharma, M. Sabharwal [44]	*Sentiment Analysis for Social Media using Svm Classifier of Machine Learning*	*International Journal of Innovative Technology and Exploring Engineering* (2019)	CNN, PSO-SVM	Model PSO-SVM mampu bersaing dengan *deep learning* seperti CNN (Akurasi: 91.91%)
N. Hayatin, G. Marthasari, L. Nuraini [45]	*Optimization of Sentiment Analysis for Indonesian Presidential Election using Naïve Bayes and Particle Swarm Optimization*	Jurnal *Online* Informatika (2020)	*Naive Bayes*, PSO-*Naive Bayes*	Algoritma PSO mampu meningkatkan akurasi pada model Naive Bayes sekitar 4.12% (Akurasi: 90.74%)
S. Ahmad, A. Bakar, M. Yaakub [39]	*Ant Colony Optimization for Text Feature Selection in Sentiment Analysis*	*Intelligent Data Analysis* (2019)	IG-RSAR, IG-GA, ACO-KNN	Model ACO-KNN berkinerja paling optimal (Avg. F-Score 82.7%)
G. Jain, M. Sharma, B. Agarwal [46]	*Optimizing Semantic LSTM for Spam Detection*	*International Journal of Information Technology (Singapore)* (2019)	LSTM, SVM, *Naive Bayes*, *Artificial Neural Network* (ANN), KNN, *Random Forest*	Algoritma LSTM mampu unggul dalam klasifikasi teks pada pendekatan *sentiment analysis* (Akurasi 95.90%)
J. Shobana, M. Murali [38]	*An Efficient Sentiment Analysis Methodology Based on Long Short-Term Memory Networks*	*Complex and Intelligent Systems* (2021)	LSTM, ANN, SVM, APSO-LSTM	Model APSO-LSTM bekerja paling optimal (Akurasi: 97.8%)
A. Alarifi, A. Tolba, Z. Al-Makhadmeh, W. Said [15]	*A Big Data Approach to Sentiment Analysis Using Greedy Feature Selection with Cat Swarm Optimization-Based Long Short-Term Memory Neural Networks*	*Journal of Supercomputing* (2020)	PSO-LSTMN, CSO-LSTMN	CSO-LSTMN berkinerja lebih optimal (Akurasi: 92.56%)
Barik, Kousik Misra, Sanjay Ray, Ajoy Kumar [30]	*LSTM-DGWO-Based Sentiment Analysis Framework for Analyzing Online Customer Reviews*	*Computational Intelligence and Neuroscience (2023)*	LSTM-DGWO (*Differential Grey Wolf*	LSTM-DGWO bekerja optimal (Akurasi: 98.89%)

Penulis	Judul	Jurnal	Metode	Hasil
Bokolo, Anthony			*Optimization*)	
O. Kraaijeveld, J. De Smedt [47]	*The Predictive Power of Public Twitter Sentiment for Forecasting Cryptocurrency Prices*	*Journal of International Financial Markets, Institutions and Money* (2020)	*Sentiment Analysis*	Hasil penelitian menunjukkan bahwa sentimen publik pada media sosial seperti Twitter atau X dapat mempengaruhi volatilitas dari *cryptocurrency*
S. Bouktif, A. Fiaz, A. Ouni, M. Serhani [24]	*Optimal Deep Learning LSTM Model for Electric Load Forecasting Using Feature Selection and Genetic Algorithm: Comparison with Machine Learning Approaches*	*Energies* (2018)	*Deep Learning, Pre-processing, Feature Selection*	Hasil dari penelitian menunjukkan penerapan *pre-processing* pada klasifikasi menggunakan algoritma *deep learning* dapat menghasilkan akurasi yang lebih tinggi

Berdasarkan Tabel 2. 1, penelitian [36] di tahun 2019 membandingkan algoritma *feature selection* pada *sentiment analysis* menggunakan data pada *platform* X. Penelitian tersebut membandingkan beberapa model baik dengan atau tanpa *feature selection* untuk diklasifikasikan berdasarkan sentimennya. *Feature selection* yang digunakan yaitu PSO dan *Generic Algorithms* (GA) dengan klasifikasi menggunakan algoritma SVM. Hasil dari penelitian tersebut menunjukkan bahwa optimasi model klasifikasi SVM dengan algoritma PSO mampu memberikan hasil akurasi dan *Area Under the Curve* (AUC) terbaik, masing-masing 86.20% dan 0.934 mengungguli SVM dan GA-SVM.

Penelitian [40] di tahun 2019 membandingkan beberapa model klasifikasi dalam melakukan *sentiment analysis* antara lain *Bayes Net*, SVM, C4.5, KNN, dan SVM-PSO. Hasil dari klasifikasi algoritma SVM dengan optimasi menggunakan algoritma PSO sebagai *feature selection* mendapatkan hasil terbaik di antara algoritma klasifikasi lainnya, dengan akurasi yang didapat yaitu 80.3%. Namun sangat disayangkan pada penelitian tersebut kurang adil dalam membandingkan, karena algoritma klasifikasi lainnya tidak melalui proses *feature selection*.

Penelitian [41] di tahun 2022 membahas *sentiment analysis* pada *review* suatu aplikasi jasa, dengan membandingkan model yang dioptimasi dengan *feature selection* menggunakan GA dan PSO. Model yang dioptimasi adalah algoritma SVM yang digunakan untuk klasifikasi suatu *review* aplikasi dengan pendekatan *sentiment analysis*. Penelitian tersebut menunjukkan bahwa optimasi dengan algoritma PSO lebih unggul dibandingkan GA dalam meningkatkan kinerja model SVM dengan akurasi yang didapat sekitar 87%.

Penelitian [42] di tahun 2019 membahas mengenai perbedaan klasifikasi dengan dan tanpa *feature selection* pada pendekatan *sentiment analysis*. Algoritma yang digunakan untuk klasifikasi *yaitu* SVM, sedangkan untuk *feature selection* yang digunakan adalah PSO. Hasil yang didapat dari penelitian tersebut yaitu terdapat peningkatan akurasi pada model klasifikasi dengan nilai awal atau hasil tanpa menggunakan *feature selection* yaitu 75.33% menjadi 82.33% menggunakan *feature selection* PSO.

Penelitian [43] di tahun 2019 melakukan eksperimen dengan membandingkan klasifikasi antara algoritma *Naive Bayes* dan SVM dengan dan tanpa *feature selection* PSO. Penelitian ini bertujuan untuk membandingkan kinerja model dalam melakukan *sentiment analysis* pada komentar di portal berita. Hasil yang didapat yaitu terdapat peningkatkan akurasi dan AUC untuk model yang menggunakan *feaure selection*, dimana model terbaik adalah SVM yang disandingkan dengan PSO untuk *feature selection* dengan akurasi 78.40% dan AUC 0.850.

Berbeda halnya dengan penelitian sebelumnya, pada penelitian [44] membandingkan model antara *deep learning* dan *machine learning* pada pendekatan *sentiment analysis*. Algoritma *deep learning* yang digunakan yaitu CNN dan dibandingkan dengan algoritma *machine learning* yaitu SVM. Pada dasarnya *deep learning* tidak memerlukan *feature selection* karena kemampuannya untuk menganalisis data secara *unsupervised* dalam jumlah besar dan dapat mengekstraki pola yang kompleks. Penelitian tersebut membuktikan bahwa dengan menambahkan *feature selection* khususnya pada algoritma *machine learning*

seperti SVM mampu menyaingi kinerja dari *deep learning* CNN dengan akurasi 91.91%.

Penelitian yang membahas *sentiment analysis* menggunakan algoritma SVM untuk klasifikasi sudah banyak dilakukan. Penelitian [45] melakukan klasifikasi sentimen menggunakan algoritma *Naive Bayes*. Penelitian tersebut mencoba untuk mengklasifikasikan sentimen menggunakan Naive Bayes, kemudian membandingkannya dengan menambahkan algoritma PSO sebagai *feature selection*. Hasil yang didapat yaitu terdapat peningkatkan akurasi sebesar 4.12% setelah ditambahkan PSO sebagai *feature selection* menjadi 90.74% pada algoritma *Naive Bayes*.

Penelitian [39] yang membahas *sentiment analysis* melakukan proses membandingkan pada beberapa pasangan model menggunakan ACO dan GA untuk *feature selection* serta KNN, *Information Gain* (IG) dan *Rough Set Attribute Reduction* (RSAR) untuk klasifikasi sentimen. Penelitian tersebut mendapatkan hasil bahwa model ACO-KNN merupakan model terbaik yang mampu untuk mendapatkan subset fitur yang paling optimal dan akurasi yang baik dengan rata-rata *f-score* 82.7%, mengungguli model IG-RSAR dan IG-GA.

Penelitian [46] membandingkan berbagai algoritma klasifikasi pada *data spam*. Algoritma klasifikasi yang digunakan pada penelitian tersebut yaitu LSTM, SVM, *Naive Bayes*, ANN, KNN, dan *Random Forest*. Hasil yang didapat dari penelitian tersebut yaitu pada klasifikasi sentimen menggunakan LSTM mengungguli semua algoritma klasifikasi dengan akurasi yang didapat 95.90% pada *sentiment analysis* menggunakan data yang diambil dari *platform* X.

Penelitian [38] mengatakan bahwa algoritma *deep learning* seperti LSTM mampu memahami pola yang kompleks pada data teks, sehingga dapat digunakan pada *sentiment analysis*. Adapun cara meningkatkan performa algoritma LSTM salah satunya melakukan optimasi parameter dengan *feature selection* berbasis *swarm intelligence* yaitu PSO, tujuannya agar dapat meningkatkan efisiensi pada proses *sentiment analysis*. Model yang diusulkan yaitu optimasi algoritma LSTM dengan APSO yang dibandingkan dengan metode konvensional seperti LSTM,

ANN, dan SVM. Hasil yang didapat dari penelitian ini yaitu model APSO-LSTM mampu mengungguli semua model berdasarkan metriks *accuracy* sebesar 97.8%, *precision* sebesar 87.28%, *recall* sebesar 78.08%, dan *f-measure* sebesar 82.42%.

Penelitian [15] memperkenalkan teknik *big data* dalam penerapan *machine learning* untuk mengevaluasi dalam meningkatkan efisiensi *sentiment analysis*. Penelitian tersebut melakukan optimasi dengan membandingkan *feature selection* berbasis *swarm intelligence* dengan algoritma PSO dan CSO pada klasifikasi sentimen menggunakan algoritma *deep learning* yaitu *Long Short-Term Memory Networks* (LSTMN). Hasil dari penelitian tersebut adalah model CSO-LSTMN lebih unggul dibandingkan PSO-LSTMN, karena model CSO-LSTMN mampu menemukan sorotan fitur lebih dalam, lebih cepat, dan mendapatkan akurasi lebih baik yaitu 92.56%.

Penelitian [30] melakukan optimasi terhadap algoritma LSTM menggunakan algoritma *swarm intelligence* yaitu DGWO. Model LSTM-DGWO dalam melakukan *sentiment analysis* terhadap *customer reviews* dengan jumlah data 50140 dari Kaggle dan mendapatkan akurasi yang tinggi yaitu 98.89%. Keberhasilan optimasi yang dilakukan penelitian tersebut didapatkan dari penyesuaian terhadap *hyper-parameter* algoritma LSTM yaitu *learning rate* dan *batch size* dengan iterasi menggunakan algoritma DGWO dalam *objective function*.

Berdasarkan penelitian [47] pertumbuhan media sosial yang sangat pesat bukan hanya menjadikan media untuk membagikan suatu informasi, tetapi dapat juga dijadikan media untuk melakukan manipusi pada harga *cryptocurrency*. Dilihat dari data historis terdapat korelasi antara sentimen yang beredar pada media sosial dengan volatilitas harga *cryptocurrency*, menyebabkan media sosial menjadikan alat yang sangat berperan dalam kenaikkan atau penurunan harga *cryptocurrency*. Namun untuk melakukan analisis terhadap data dari media sosial diperlukan alat agar proses analisis memberikan hasil yang efektif dan efisien salah satunya menggunakan *sentiment analysis*. Terdapat beberapa cara untuk melakukan *sentiment analysis*, banyak penelitian yang membuktikan bahwa peranan *feature selection* dapat digunakan untuk melakukan optimasi atau meningkatkan akurasi

dari hasil *sentiment analysis*. Tidak hanya meningkatkan kinerja model klasifikasi menggunakan *machine learning* saja yang melalui tahapan *pre-processing* menggunakan *feature selection*, melainkan *deep learning* juga dapat ditingkatkan kinerjanya menggunakan *feature selection* seperti yang dibahas pada penelitian [24]. Oleh karena itu, tidak menutup kemungkinan *feature selection* pada *deep learning* akan memberikan hasil yang lebih akurat dibandingkan hanya mengandalkan kemampuan *deep learning* itu sendiri.

Berdasarkan penelitian terdahulu yang dituliskan pada Tabel 2. 1, mayoritas penenelitian *sentiment analysis* yang dilakukan masih menggunakan *machine learning*. *Machine learning* tersebut dioptimasi menggunakan *feature selection* berbasis *swarm intelligence* dan terbukti dapat meningkatkan kinerja pada model klasifikasi. Terdapat beberapa penelitian yang menggunakan *deep learning* untuk *sentiment analysis*, karena menurut teori algoritma *deep learning* mampu melakukan klasifikasi pada data yang lebih kompleks dan lebih besar. Adapun penelitian yang membuktikan bahwa *feature selection* dapat meningkatkan kinerja algoritma *machine learning* sehingga mampu bersaing dengan algoritma *deep learning*. Banyak dari penelitian terdahulu yang hanya melakukan optimasi menggunakan *feature selection* PSO pada *sentiment analysis.* Mayoritas penelitian terdahulu juga tidak membandingkan algoritma *swarm intelligence* lainnya seperti ACO dan CSO untuk mencari algoritma optimasi terbaik khususnya pada *sentiment analysis* menggunakan algoritma LSTM pada domain *cryptocurrency*. Banyak juga yang tidak memperlihatkan perbandingan model sebelum dan sesudah diterapkan *feature selection*, serta hasil yang diperoleh tidak diuji menggunakan *statistical test* agar suatu model terbukti dapat meningkatkan kinerja model dalam klasifikasi sentimen. Untuk mengisi *gap* pada penelitian terdahulu, penelitian ini melakukan optimasi algoritma *deep learning* yaitu LSTM dengan membandingkannya dengan dan tanpa *feature selection* berbasis *swarm intelligence*, ditambah membandingkan algoritma *swarm intelligence* lainnya seperti ACO dan CSO untuk meningkatkan kinerja LSTM dalam klasifikasi sentimen pada data *cryptocurrency*. Kemudian hasil yang diperoleh diuji menggunakan *statistical test* dengan *Shapiro-Wilk Test* untuk menguji distribusi *accuracy* dan *loss*, serta ANOVA *Test* untuk melihat

bahwa terdapat perbedaan signifikan pada model sebelum dan setelah diterapkan *feature selection* berbasis *swarm intelligence*.

2.2 Tinjauan Teori

2.2.1 *Cryptocurrency*

Cryptocurrency adalah mata uang digital atau aset digital yang dibangun di atas teknologi *blockchain*, yang memungkinkan verifikasi pembayaran dan transaksi lainnya tanpa adanya kustodian terpusat atau pihak ketiga [48]. Penerapan sistem keuangan pada teknologi *blockchain* menghadirkan cara baru untuk transaksi yang lebih cepat, aman, murah, dan transparan. *Cryptocurrency* pertama yaitu Bitcoin diperkenalkan pada tahun 2009 oleh seorang anonim dengan *nickname* Satoshi Nakamoto, yang sampai sekarang masih tidak diketahui identitas asli dari pencipta Bitcoin ini [48]. Adapun sebutan untuk *cryptocurrency* selain Bitcoin yang disebut sabagai *Altcoins*. Banyak orang percaya bahwa Bitcoin dan *Altcoins* merupakan aset masa depan, karena sejumlah kelebihan yang ditawarkan termasuk terdesentralisasi atau tidak terikat dengan pihak manapun termasuk pemerintah dan bank sentral.

Tujuan awal diciptakannya *cryptocurrency* yaitu untuk melakukan transaksi yang terdesentralisasi, namun karena memiliki volatilitas yang tinggi maka sangat tidak cocok untuk dijadikan alat pembayaran [49]. Sebagian besar orang menjadikan *cryptocurrency* sebagai *store of value* atau instumen investasi karena sebagian *cryptocurrency* jumlahnya terbatas seperti Bitcoin yang jumlahnya hanya 21 juta koin. Apalagi Bitcoin merupakan salah satu *cryptocurrency* yang menggunakan mekanisme PoW yang melibatkan proses penambangan dengan mengkonsumsi energi yang masif, menyebabkan jumlah persediaannya akan terus berkurang sehingga nilainya terus meningkat sama halnya dengan emas [4]. Berbeda dengan Bitcoin, Altcoins memiliki tujuan bukan hanya sebagai instrumen investasi melainkan sebagai suatu solusi untuk melengkapi kekurangan dari Bitcoin seperti membuat mekanisme-mekanisme yang lebih ramah lingkungan dan memiliki fungsi yang dapat menyelesaikan permasalahan nyata di dunia.

2.2.2 *Platform* X

Platform X merupakan media sosial yang awalnya dikenal dengan nama Twitter sebelum diganti nama oleh Elon Musk pada Oktober 2022 [50]. Sama halnya dengan media sosial pada umumnya, *platform* X memungkinkan pengguna untuk membagikan konten berupa komentar, gambar, video, atau disebut sebagai *tweets* secara publik dan dapat dilihat atau ditanggapi oleh pengguna lain. Salah satu penyebab *platform* X populer dibandingkan media sosial yang lain karena *platform* ini berfokus pada melindungi kebebasan berbicara [51]. Kebebasan berbicara membuat *platform* ini menjadi salah satu media sosial yang sangat berdampak pada kehidupan sosial di seluruh dunia. Adapun penelitian yang mengatakan bahwa *platform* X merupakan salah satu media sosial yang kaya akan kecerdasan emosional, sehingga dapat mempengaruhi persepsi orang dalam memandang suatu hal termasuk *cryptocurrency* [12].

2.2.3 *Web Scrapping*

Web scrapping atau *crawling* adalah teknik yang digunakan untuk mengumpulkan data secara otomatis dari internet atau suatu situs dengan menggunakan *tools*. Penggunaan *web scrapping* dapat membantu dalam mengekstraksi data dalam jumlah besar kemudian menyimpannya ke dalam *dataset*, sehingga dapat menghemat waktu ketika ingin menganalisis data [52]. Data yang diambil dari suatu situs dapat sangat bermanfaat untuk melakukan analsis pasar, terutama dalam memahami sentimen pasar dan mendapatkan wawasan untuk pengambilan suatu keputusan investasi. Namun data yang diambil menggunakan teknik *web scrapping* biasanya memiliki banyak *noise* dan tidak terstruktur, sehingga menjadi tantangan ketika melakukan *pre-processing* pada *dataset* [16].

2.2.4 *Sentiment Analysis*

Sentiment analysis adalah proses menganalisis dan mengekstraksi informasi dari suatu data [13]. Data yang dimaksud dapat berupa opini, komentar, *review*, atau *tweets* kemudian diklasifikasikan berdasarkan polaritasnya. Klasifikasi data berdasarkan polaritasnya dalam *sentiment*

analysis dapat dibagi menjadi dua jenis yaitu *binary sentiment analysis* dan *multi-class sentiment analysis* [53]. *Binary sentiment analysis* yaitu mengklasifikasikan data berdasarkan dua sentimen yaitu positif dan negatif. *Multi-class sentiment analysis* adalah mengklasifikasikan data berdasarkan tiga sentimen yaitu positif, negatif, dan netral. Oleh karena itu, penggunaan jenis *sentiment analysis* perlu disesuaikan dengan kebutuhan dan tujuan suatu penelitian.

2.2.5 *Deep Learning*

Deep learning merupakan sebuah pengembangan dari *machine learning* yang berfokus pada algoritma yang terinspirasi dari struktur dan fungsi otak [54]. *Deep learning* telah menjadi sangat populer karena kemampuannya yang unik dalam menangani data dalam jumlah besar dan kompleks [55]. Perbedaan antara *machine learning* dengan *deep learning* yaitu pada *machine learning* sering memerlukan intervensi manual untuk mengidentifikasi fitur yang relevan dari data, sedangkan *deep learning* mampu secara otomatis menemukan fitur yang penting untuk melakukan prediksi atau klasifikasi, sehingga dapat meningkatkan efisiensi waktu terutama pada tahapan *modeling* [56]. *Deep learning* juga terbukti dapat meningkatkan akurasi secara signifikan terutama pada data yang berdimensi tinggi dan memiliki kompleksitas tinggi, sehingga penggunaannya sangat sesuai apabila menggunakan data yang berasal dari media sosial yang dikumpulkan dengan teknik *web scrapping* [57].

2.2.6 *Data Pre-processing*

Data pre-processing adalah tahapan yang melibatkan proses pengolahan dan persiapan data mentah sebelum digunakan untuk pelatihan model. Tujuan *data pre-processing* yaitu untuk membuat data lebih sesuai untuk pemrosesan dengan model *machine learning* atau *deep learning*, dimana format data yang digunakan secara langsung dapat mempengaruhi kinerja dan efektivitas model yang akan dilatih [58][59]. Hal tersebut membuat tahapan *data pre-processing* menjadi suatu tahapan yang krusial dan paling memakan waktu karena kompleksitas data terus meningkat seiiring waktu. Terdapat beberapa langkah

yang umum digunakan pada *data pre-processing* khususnya pada data dari *platform* X antara lain [60][61][62]:

1) *Remove URLs, Mention, Hashtag, Special Characters and Numbers*

   Menghilangkan elemen-elemen seperti *URLs* (http/https), *mention* (*@username*), *hashtag* (#), *special characters* ($BTC), dan *numbers* (1, 2, 3) dari data. Penghapusan elemen-elemen tersebut dilakukan karena cenderung tidak memberikan informasi yang berguna dalam klasifikasi teks dan dapat menyebabkan ambiguitas pada model klasifikasi. Langkah *data pre-processing* diharapkan dapat membantu dalam mengurangi *noise*, sehingga teks menjadi fokus pada konten yang relevan.

2) *Remove Punctuation*

   Menghapus seluruh tanda baca dalam teks dan hanya menyisakan kata-kata yang menjadi fokus analisis. Tujuannya yaitu membantu dalam mengurangi kompleksitas teks dan memudahkan pemrosesan lebih lanjut pada tahapan *tokenization*. Tanda baca yang dihapus contohnya titik (.), koma (,), tanda tanya (?), dan sejenisnya.

3) *Convert to Lowercase*

   Convert to lowercase merupakan suatu teknik untuk mengubah keseluruhan data teks menjadi kuruf kecil. Konversi semua data teks ke huruf kecil bertujuan untuk menyederhanakan variasi atau keberagaman kapitalisasi dalam korpus yang dapat menyebabkan masalah selama klasifikasi. Apabila suatu teks tidak dibuat konsisten, akan berakibat pada model yang kurang baik dalam melakukan klasifikasi.

4) *Tokenization*

   Tokenization merupakan suatu proses mengubah teks menjadi unit-unit yang lebih kecil atau disebut sebagai *token* dapat berupa kata, frasa, atau karakter. *Tokenization* memungkinkan suatu model untuk memproses dan menganalisis teks pada tingkat kata termasuk proses pelabelan data berdasarkan sentimen. *Tokenization* juga merupakan salah satu syarat untuk

melakukan *sentiment analysis* agar algoritma *machine learning* atau *deep learning* mampu melakukan klasifikasi teks dengan baik.

5) *Remove Stop Words*

Remove stop words merupakan suatu proses untuk menghapus kata-kata umum yang sering muncul dalam teks seperti "*and*", "*or*", "*in*", dan sejenisnya. Kata-kata tersebut cenderung tidak memberikan atau memiliki makna yang signifikan. Tujuan dengan menghapus kata-kata yang tidak bermakna agar dapat meningkatkan relevansi pada pemrosesan teks.

6) *Lemmatization*

Lemmatization yaitu proses mengubah kata-kata ke bentuk dasar misalnya "*running*", "*ran*, dan "*run*". Semua bahasa khususnya Bahasa Inggris biasanya memiliki beragam bentuk walaupun memiliki arti yang sama. Tujuan diterapkan *lemmatization* yaitu membantu mengurangi variasi kata agar perbedaan bentuk kata tidak mempengaruhi hasil analisis.

7) *Remove Words with Length < 3*

Remove words with length < 3 merupakan proses menghapus kata-kata yang memiliki panjang kurang dari tiga karakter. Kata-kata yang terlalu pendek cenderung kurang informatif atau tidak memiliki makna. Tujuan diterapkannya proses ini yaitu dapat membantu mengurangi kompleksitas pada data khususnya data yang akan di proses dengan algoritma.

2.2.7 *Data Labeling*

Data labeling merupakan proses memberikan label pada *dataset* yang belum terstruktur atau *unsupervised* dengan mengklasifikasikan atau mengidentifikasi elemen data tersebut berdasarkan karakteristik tertentu yang terkandung di dalamnya. Dalam konteks *machine learning* atau *deep learning*, data yang sudah diberikan label atau *supervised* dapat digunakan untuk melatih model agar model tersebut dapat mempelajari pola atau fitur dari data tersebut, sehingga dapat memprediksi atau melakukan klasifikasi pada data baru yang serupa [63]. Proses *data labeling* sangat penting dalam pengembangan sistem

yang menggunakan *machine learning* atau *deep learning* karena secara langsung mempengaruhi kinerja model. Terdapat dua proses *data labeling* yaitu dilakukan yaitu secara manual oleh manusia (anotator) dengan memeriksa bagian demi bagian kemudian menandai setiap elemen data dan secara otomatis dengan menggunakan algoritma yang sudah dilatih sebelumnya untuk menghasilkan label berdasarkan fitur data [64].

Sudah banyak penelitian yang menggunakan teknik pelabelan otomatis dan memberikan hasil yang baik dalam menentukan kategori data, termasuk penelitian yang melakukan *sentiment analysis*. *Data labeling* secara otomatis dapat dilakukan dengan menggunakan beberapa *library data labeling* yang tersedia. Tabel 2. 2 merupakan tabel perbandingan *library* populer untuk *data labeling* berdasarkan kelebihan dan kekurangan.

Tabel 2. 2 Perbandingan *Library Data Labeling*

Library	**Kelebihan**	**Kekurangan**
VADER	- Akurat untuk digunakan pada data yang berasal dari media sosial [65] - Kemudahan penggunaan untuk *labeling* [66]	- Akurasi terbatas untuk teks panjang atau formal [67] - Tidak mampu menangani konteks kompleks [68]
TextBlob	- Kemudahan implementasi *data labeling* [69] - Fleksibilitas dalam tugas *Natural Language Processing* (NLP) [70]	- Akurasi lebih rendah dibandingkan dengan model *deep learning* [69] - Terbatas pada *sentiment analysis* sederhana [71]
BERT	- Akurasi tinggi untuk berbagai tugas NLP [69] - Fleksibilitas dalam tugas NLP [72]	- Memerlukan banyak sumber daya komputasi [67] - Latensi lebih tinggi [73]

Library VADER digunakan pada penelitian ini dikarenakan kemampuannya dalam melakukan *data labeling* pada data yang berasal dari media sosial. Hal tersebut dibuktikan dalam beberapa skenario pengujian bahwa *library* VADER memberikan akurasi yang paling tinggi dibandingkan *TextBlob* dalam *sentiment analysis* [74]. *Library Valence Aware Dictionary and sEntiment Reasoner* (VADER) memiliki kemampuan dalam mengklasifikasikan data teks berdasarkan kamus kata khusus yang dirancang untuk menangani data pada media sosial yang meliputi emotikon, bahasa gaul,

kontraksi, negasi, dan akronim [75]. Kelebihan yang ditawarkan VADER yaitu cepat, peka, dan berkaitan dengan kekuatan emosi terhadap polaritas sentimen sehingga dapat mengidentifikasi apakah suatu data termasuk ke dalam sentimen positif, negatif, atau netral [76]. Berdasarkan penelitian sebelumnya, penggunaan VADER dalam *sentiment analysis* khususnya menggunakan data *tweets* mengenai *cryptocurrency* menunjukkan korelasi jangka pendek pada pergerakkan harga Bitcoin [77].

Umumnya terdapat 3 label untuk *sentiment analysis* yaitu positif, negatif, dan netral. Namun banyak penelitian *sentiment analysis* yang hanya menggunakan 2 label saja yaitu positif dan negatif. Dengan hanya digunakan 2 label untuk *sentiment analysis* dapat menyederhanakan analisis dan mengurangi bias atau kebingungan pada proses *modeling* [78][79]. Selain itu penggunaan 2 label dalam *sentiment analysis* dapat mengurangi kesalahan klasifikasi, sehingga algoritma dapat fokus pada polaritas ekstrim yang jelas sehingga lebih akurat [80].

2.2.8 *Data Splitting*

Data splitting adalah proses membagi data menjadi beberapa bagian, pembagian yang umum dilakukan yaitu membagi data menjadi dua bagian yaitu *data training* dan *data testing*. Tujuan utama *data splitting* yaitu untuk mencapai pemisahan yang representatif secara merata dari *dataset* [81]. *Data training* digunakan untuk melatih model menggunakan algoritma, sedangkan *data testing* digunakan untuk melakukan validasi atas model yang dilatih. Biasanya pembagian data ini menggunakan rasio seperti 90:10, 80:20, atau 70:30, namun terdapat suatu penelitian yang membandingkan hasil kinerja model bahwa rasio 80:20 memberikan hasil kinerja yang paling optimal dengan akurasi yang paling tinggi dibandingkan rasio yang lain khususnya pada bahasan *cryptocurrency* [82].

2.2.9 *Feature Selection*

Data yang diambil dari internet misalnya media sosial umumnya memiliki karakteristik seperti berdimensi tinggi dan banyak *noise* yang ditandai dari

banyak fitur yang tidak relevan dan terdapat redudansi pada data yang digunakan. Hal tersebut menyebabkan menurunnya kinerja pada model algoritma, sehingga akurasi yang didapat kurang maksimal. Salah satu pendekatan yang dapat mengatasi permasalahan tersebut adalah *feature selection*. *Feature selection* merupakan salah satu teknik *pre-processing* yang melibatkan pemilihan subset minum dari fitur yang relevan pada set fitur asli yang besar sehingga dapat meningkatkan efisiensi kinerja algoritma [83]. Terdapat tiga metode *feature selection* untuk algoritma *supervised learning* yaitu metode *filter, wrapper,* dan *hybrid*.

2.2.9.1 Metode *Filter*

Metode *filter* bersifat *univariant,* sehingga interaksi antar prediktor tidak diperhatikan [84]. *Feature selection* pada metode *filter* menggunakan ukuran statistik seperti korelasi, selain itu metode *filter* dilakukan tanpa model prediktif membuat metode ini lebih cepat ketika jumlah fitur sangat banyak [85]. Contoh *feature selection* dengan metode *filter* adalah *Chi-squre*, ANOVA, LDA, *pearson's correlation, gain ratio,* IG, *log likelihood ratio*, dan *relief-f.*

2.2.9.2 Metode *Wrapper*

Metode *wrapper* bersifat *multivariat*, dalam metode ini algoritma *feature selection* keluar sebagai *wrapper* di sekiar algoritma prediktif dan menggunakan model yang sama untuk memilih fitur terbaik. Penggunaan metode *wrapper* pada pendekatan *feature selection* bertujuan agar tetap memberikan kinerja yang baik saat data cenderung *overfitting*. Namun harga komputasi cukup mahal untuk metode *wrapper* karena metode ini memiliki tingkat kompleksitas yang tinggi [85].

2.2.9.3 Metode *Hybrid*

Metode *hybrid* adalah gabungan dari kedua metode yaitu metode *filter* dan metode *wrapper*. Selain itu metode *hybrid* bersifat *multivariant* layaknya metode *wrapper* [86]. Ada 2 teknik yang membuat subset fitur menjadi optimal dalam metode *hybrid* yaitu IG dan *Entropy Weighted*

Generic Algorithm (EWGA). Adapun pendekatan metaheuristik contohnya seperti PSO, ACO dan CSO. Metode *hybrid* pada *feature selection* dirancang untuk mengurangi waktu pencarian, menangani komputasi yang kompleks, dan mengekstraksi *subset* fitur untuk menghasilkan akurasi klasifikasi tertinggi dengan jumlah fitur minimum [77].

2.2.10 Metrik Evaluasi

Metrik evaluasi berperan penting dalam mencapai hasil yang optimal terutama selama *training* model klasifikasi [87]. Terdapat banyak metrik yang dapat digunakan untuk membandingkan antar model satu dengan model yang lain termasuk *accuracy*, *loss*, dan *execution time*. Ketiga metrik tersebut digunakan karena berkaitan erat dengan kinerja algoritma klasifikasi seperti LSTM. Berikut penjelasan dari masing-masing metrik:

1) *Accuracy*

Metrik *accuracy* sangat umum ditemukan pada penelitian yang membahas model klasifikasi. Model yang baik adalah model yang memiliki *accuracy* tinggi, yang menunjukkan suatu model mampu memprediksi data berdasarkan kategorinya [88]. Persamaan *acurracy* dapat dilihat pada persamaan (2.1) [88]. Berdasarkan persamaan (2.1), TP (*True Positive*) merupakan jumlah sampel positif yang diprediksi benar oleh model, TN (*True Negative*) merupakan jumlah sampel negatif yang diprediksi benar oleh model, FP (*False Positive*) merupakan jumlah sampel negatif yang salah diprediksi sebagai positif oleh model, FN (*False Negative*) merupakan jumlah sampel positif yang salah diprediksi sebagai negatif oleh model.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (2.1)$$

Rumus 2. 1 Rumus *Accuracy*

2) *Loss*

Metrik *loss* digunakan untuk mengukur ketidaksamaan antara probabilitas prediksi suatu kategori dan label kategori sebenarnya [88].

Nilai *loss* yang tinggi menandakan bahwa suatu model kurang dapat memprediksi dengan benar terutama pada *data testing*. Dalam *binary classification* atau klasifikasi yang memiliki dua label, kelas *true* mempunyai nilai 1, sedangkan kelas lainnya mempunyai nilai 0. Kelas yang bernilai *true* dilambangkan dengan $p(y = 1|x)$, sedangkan probabilitas prediksi lain dilambangkan dengan $p(y = 0|x)$. Persamaan *loss* untuk *binary class* dapat dilihat pada persamaan (2.2) [88].

$$L(y, p) = -(y \log(p) + (1 - y) \log(1 - p)) \quad (2.2)$$

Rumus 2. 2 Rumus *Loss*

3) Execution Time

Pengukuran kinerja dari model juga dapat dilihat berdasarkan *execution time* atau berapa lama proses *modeling* selesai [89]. Semakin cepat *execution time* dari suatu model, menandakan bahwa kinerja dari model tersebut semakin baik. Persamaan metrik *execution time* dapat dilihat pada persamaan (2.3).

$$Execution\ Time\ =\ End\ Time\ -\ Start\ Time \quad (2.3)$$

Rumus 2. 3 Rumus *Execution Time*

2.2.11 *Statistical Test*

Statistical test adalah metode analisis statistik yang digunakan untuk mengidentifikasi perbedaan yang signifikan secara statistik antara dua atau lebih kelompok data. Metode ini umumnya digunakan untuk mengevaluasi hasil numerik [90]. Tujuan utama dari *statistical test* adalah membuat keputusan tentang populasi berdasarkan sampel data yang diambil dari populasi tersebut. Dalam penelitian ini, *statistical test* digunakan untuk membuktikan adanya perbedaan hasil analisis antara model-model yang dibandingkan berdasarkan efektivitas seperti *accuracy*, *loss*, dan *execution time*. Penggunaan *statistical test* dapat membantu peneliti dalam menghasilkan keputusan yang lebih cerdas mengenai isu yang sedang diteliti [91]. Terdapat dua *statistical test* yang digunakan pada penelitian ini antara lain:

1) *Shapiro-Wilk Test*

Shapiro-Wilk Test digunakan untuk menilai apakah sampel berasal dari distribusi normal (*gaussian*). Penilaian ini sangat penting dalam statistik karena banyak teknik analitis yang mengamsumsikan bahwa data mengikuti distribusi normal. *Test* ini bekerja dengan membandingkan skor data dengan distribusi normal, semakin dekat data dengan distribusi normal, semakin besar kemungkinan asumsi normal dipenuhi. Persamaan (2.4) merupakaan persamaan *Shapiro-Wilk Test* [92].

$$W = \frac{\left(\Sigma_{i=1}^{\eta} = a_i x_{(i)}\right)^2}{\Sigma_{i=1}^{n}(x_i - \bar{x})^2} \quad (2.4)$$

Rumus 2. 4 Rumus *Shapiro-WIlk Test*

Berikut penjelasan dari persamaan (2.4):

- W adalah statistik uji *Shapiro-Wilk*.
- η adalah ukuran sampel.
- $x_{(i)}$ adalah $i-th$ urutan statistik, yaitu sampel data yang diurutkan dari nilai terkecil ke terbesar.
- $\bar{x}$ adalah rata-rata sampel.
- a_i adalah koefisien yang berasal dari nilai ekspektasi urutan statistik dari distribusi normal, yang dihitung berdasarkan ukuran sampel η.

2) *Analysis of Variance (*ANOVA*) Test*

ANOVA *Test* digunakan untuk membandingkan tiga atau lebih kelompok sampel untuk menentukan apakah setidaknya terdapat satu pasangan kelompok memiliki rata-rata yang berbeda signifikan. Tujuan utama dari ANOVA adalah untuk mengetahu variabilitas antar kelompok dibandingkan dengan variabilitas dalam kelompok. Persamaan (2.5) merupakan persamaan ANOVA [93].

$$F = \frac{MS_{between}}{MS_{\omega within}} \quad (2.5)$$

Rumus 2. 5 Rumus ANOVA *Test*

Berikut penjelasan dari persamaan (2.5):

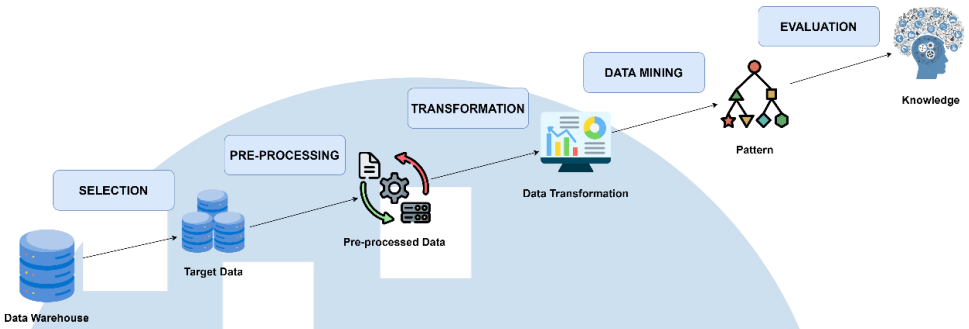
- F adalah statistik uji ANOVA.
- $MS_{between}$ (*Mean Square Between Groups*) adalah varians antar kelompok, dihitung dengan rumus $\frac{SS_{between}}{df_{between}}$, dimana $SS_{between}$ adalah jumlah kuadrant antar kelompok, dan $df_{between}$ adalah derajat kebebasan antar kelompok.
- $MS_{\omega ithin}$ (*Mean Square Within Groups*) adalah varians dalam kelompok, dihitung dengan rumus $\frac{SS_{\omega ithin}}{df_{\omega ithin}}$, dimana $SS_{\omega ithin}$ adalah jumlah kuadrant dalam kelompok, dan $df_{\omega ithin}$ adalah derajat kebebasan dalam kelompok.

Uji statistik *Shapiro-Wilk Test* maupun ANOVA *Test* memiliki peran penting dalam analisis data, terutama membantu peneliti dalam memahami distribusi dan perbedaan data yang dikerjakan.

2.3 *Framework* dan Algoritma

2.3.1 *Knowledge Discovery in Database* (KDD)

KDD merupakan sebuah *framework* pada *data mining* yang dapat dilakukan untuk menemukan pengetahuan yang berguna dan berarti dari data dalam jumlah besar [94]. *Data mining* secara khusus mengacu pada penerapan *machine learning*, sedangkan KDD mencakup proses yang lebih luas dalam mengubah pertanyaan atau masalah menjadi berdasarkan dengan data. Selain itu KDD juga mencakup tugas-tugas seperti pengumpulan data, persiapan, analisis, interpretasi, dan implementasi hasil. *Framework* KDD tidak hanya sebatas pada serangkaian teknik komputasi, namun merupakan pendekatan metodologis untuk berinovasi dan mengeksekusi aplikasi baru berdasarkan wawasan berbasis data [95].



Gambar 2. 1 *Framework* KDD

Sumber: [96]

Gambar 2. 1 merupakan alur pada *framework* KDD yang terdiri dari 5 tahapan utama yaitu *selection*, *pre-processing*, *transformation*, *data mining*, dan *evaluation*. Proses *framework* ini dapat berulang pada setiap langkah sehingga memungkinkan untuk kembali ke langkah sebelumnya apabila hasil yang didapat kurang sesuai [94]. Berikut penjelasan masing-masing tahapan pada KDD [97]:

1) *Selection*

   Tahapan *selection* merupakan tahapan pertama dalam KDD yang bertujuan untuk melakukan proses identifikasi dan seleksi *subset* data yang relevan untuk dianalisis. Pemilihan data ini juga meliputi proses pengumpulan data dari berbagai sumber termasuk media sosial, namun harus melakukan seleksi sesuai dengan tujuan penelitian.

2) *Pre-processing*

   Tahapan *pre-processing* merupakan tahapan kedua dalam KDD yang melakukan proses membersihkan data dari *noise* dan data yang tidak konsisten, termasuk menangani *value* dan *outliners*. Tahapan ini juga merupakan salah satu tahapan yang krusial dalam pendekatan *data mining*, karena data tidak bersih atau tidak siap pakai bila dimasukkan ke model *machine learning* akan mengakibatkan kinerja yang kurang baik pada model.

3) *Transformation*

Tahapan *transformation* merupakan tahapan untuk melakukan transformasi data ke bentuk atau format yang sesuai dengan kebutuhan *data mining* seperti algoritma *machine learning* atau *deep learning*. Tahapan ini meliputi proses normalisasi, agregasi, atau metode lainnya untuk mempersiapkan data. Termasuk menambahkan label ke dalam data pada pendekatan *sentiment analysis*.

4) *Data Mining*

Tahapan *data mining* adalah tahapan yang untuk mengekstaksi pola atau model dari data yang diproses. Tahapan ini juga merupakan tahapan inti dari *framework* KDD yang meliputi proses *classification*, *clustering, association rule learning, regression*. Tidak hanya itu melainkan proses mengeksplorasi dalam memaksimalkan kinerja algoritma juga termasuk dalam proses *data mining*.

5) *Evaluation*

Tahapan *evaluation* atau *interpretation* yaitu proses evaluasi pola dan model yang dihasilkan untuk memastikan pola dan model tersebut bermakna dan berguna sesuai kebutuhan penelitian. Contohnya melakukan visualisasi hasil penelitian, melakukan *testing* pada data baru, atau melakukan *statistical test*. Tahapan ini juga melibatkan proses penafsiran hasil dalam konteks tujuan dan memungkinkan untuk melakukan revisi pada proses jika hasil yang didapat tidak memuaskan.

Keluaran dari *framework* ini yaitu *knowledge* atau wawasan sehingga hasil analisis dapat digunakan untuk mengintegrasikan pengetahuan ke dalam lingkungan operasional. *Knowledge* tersebut didapatkan dengan melibatkan penerapan model prediktif ke dalam sistem produksi atau menggunakan wawasan yang diperoleh untuk membantu dalam proses pengambilan keputusan atau kesimpulan. Berdasarkan keputusan tersebut dapat membantu seseorang untuk melakukan tindakan atau aksi yang berlandasan dengan data.

2.3.2 *Long Short-Term Memory Networks* (LSTM)

Algoritma LSTM adalah salah satu algoritma *deep learning* yang merupakan evolusi dari RNN dengan menambahkan interaksi tambahan per

modul (*cell*) [98]. Algortima LSTM dirancang untuk mengatasi permasalahan pembelajaran dependensi jangka panjang, yang sering kali sulit dilakukan oleh RNN tradisional. Kemampuan LSTM dalam mempelajari fitur secara langsung dari *time-series data* dan menangkap dependensi jangka panjang melalui perilaku berulang dan mekanisme gerbang pada LSTM menjadi salah satu alasan mengapa LSTM sering digunakan pada penelitian yang berkaitan dengan *time-series data* [99]. Adapun penggunaan *objective function* yang berfungsi untuk melatih model LSTM dengan cerdas [100]. Tujuan diterapkannya *objective function* yaitu melakukan penalti pada hasil prediksi yang salah sehingga hasil prediksi menjadi lebih aman [101]. Penggunaan *objective function* terbukti mampu menyesuaikan bobot dalam model LSTM semedikian rupa, sehingga *loss* menjadi minimal dan pembelajaran model menjadi efektif. Persamaan-persamaan untuk operasi dalam algoritma LSTM adalah sebagai berikut [102][103]:

$$f_t = \sigma\left(W_f \cdot [x_t, h_{t-1}] + b_f\right) \quad (2.6)$$

$$i_t = \sigma(W_i \cdot [x_t, h_{t-1}] + b_i) \quad (2.7)$$

$$\widetilde{c_t} = tanh(W_c \cdot [x_t, h_{t-1}] + b_c) \quad (2.8)$$

$$C_t = f_i \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (2.9)$$

$$O_t = \sigma(W_0 \cdot [x_t, h_{t-1}] + b_o) \quad (2.10)$$

$$h_t = O_t \cdot \tanh(c_t) \quad (2.11)$$

Rumus 2. 6 Rumus Algoritma LSTM

Berikut penjelasan dari masing-masing komponen:

- *Forget gate* f_t bertanggung jawab untuk menentukan seberapa banyak informasi dari *cell state* sebelumnya C_{t-1} yang harus dilupakan atau diabaikan. Dilakukan dengan melalui sebuah lapisan *sigmoid*, yang mengeluarkan nilai antara 0 dan 1 untuk setiap angka di *cell state*. Nilai mendekati 0 berarti "lupakan informasi ini" sedangkan 1 berarti "pertahankan informasi ini"

- *Input gate* i_t menentukan informasi baru mana yang akan disimpan di *cell state*. Fungsinya adalah mengontrol pembaruan *cell state* dengan melakukan *filter* informasi yang masuk. Mirip dengan *forget gate, input gate* juga menggunakan lapisan *sigmoid* untuk menentukan bagian mana dari informasi yang masuk penting dan harus disimpan.
- *Candidate cell state* $\tilde{c}_t$ menciptakan sebuah vektor kandidat untuk *cell state* baru dengan menggunakan fungsi aktivasi *tanh*, yang menghasilkan nilai antara -1 dan 1. Komponen ini merupakan versi sementara yang menunjukkan nilai yang mungkin ditambahkan ke *cell state* berdasarkan informasi masuk saat ini.
- *Update cell state* C_t menggabungkan tiga komponen sebelumnya untuk mengupdate *cell state* lama C_{t-1} menjadi *cell state* baru C_t. Hal tersebut dilakukan dengan pertama-tama menggunakan *forget gate* untuk menghapus informasi yang dinilai tidak penting dan kemudian menambahkan informasi baru yang di *filter* oleh *input gate* dari *candidate cell state*.
- *Output gate* O_t menentukan bagian dari *cell state* yang akan keluar dari *unit* LSTM. Fungsinya adalah untuk melakukan *filter* informasi dari *cell state* yang akan digunakan sebagai *output* dengan menggunakan fungsi aktivasi *sigmoid*. *Output* yang dihasilkan kemudian dikalikan dengan fungsi *tanh* dari *cell state* untuk memastikan bahwa nilai *output* berada di antara -1 dan 1.
- *Output* h_t merupakan hasil akhir yang peroleh dari operasi *output gate*. *Output* ini akan digunakan sebagai bagian dari *output* jaringan pada waktu t dan sebagai *input* untuk LSTM pada langkah waktu berikutnya.
- W dan b adalah bobot dan bias yang dipelajari selama proses pelatihan

- σ adalah fungsi aktivasi *sigmoid* yang menghasilkan nilai antara 0 dan 1, digunakan untuk mengontrol proporsi informasi yang dilewatkan melalui *gate*.
- $tanh$ merupakan fungsi aktivasi *hyperbolic tangent* yang menghasilkan nilai antara -1 dan 1, digunakan untuk mengantur nilai di dalam *candicate cell state*.
- $\cdot$ merupakan perkalian elemen demi elemen (*hadamard product*)

Operasi-operasi ini memungkinkan LSTM untuk secara selektif mengingat atau melupakan informasi, yang membuatnya sangat efisien untuk tugas-tugas yang memerlukan pemahaman konteks jangka panjang.

2.3.3 *Swarm Intelligence*

Swarm intelligence adalah bagian dari *Articial Intelligence* (AI) untuk menyelesaikan masalah optimasi yang kompleks dengan menyediakan metode yang cepat dan andal [35]. *Swarm intelligence* dipakai untuk membantu mengoptimalkan kinerja dari *feature selection*, sehingga mampu memilih fitur yang relevan, meningkatkan akurasi, serta menurunkan waktu eksekusi dengan lebih efisien. Berikut penjelasan algoritma *swarm intelligence* yang digunakan dalam penelitian ini.

2.3.3.1 *Particle Swarm Optimization* (PSO)

Algoritma PSO merupakan teknik optimasi yang terinspirasi dari perilaku sosial burung atau ikan saat mencari makanan [104]. Burung tidak tahu dimana lokasi persis suatu makanan berada, maka mereka umumnya mengikuti burung lain yang dianggap dekat dengan sumber makanan. Dalam konteks PSO, setiap burung atau ikan disebut sebagai *particle* dan setiap *particle* memiliki *fitness function* (*square of error*), serta sekelompok *particle* dikenal sebagai *swarm*. Cara kerja PSO yaitu pada setiap iterasi pertama-tama akan menemukan solusi terbaik yang ditemukan dalam *swarm* kemudian disimpan sebagai pbest, setelah itu gbest disimpan sebagai solusi terbaik yang ditemukan dari seluruh iterasi

yang sudah dilakukan. Berikut merupakan persamaan algoritma PSO dalam menemukan pbest dan gbest [105].

$$\begin{cases} \vec{v}_i \Leftarrow \vec{v}_i + \vec{U}(0,\phi_1) \otimes (\vec{p}_i - \vec{x}_i) + \vec{U}(0,\phi_2) \otimes (\vec{p}_g - \vec{x}_i) \\ \vec{x}_i \Leftarrow \vec{x}i + \vec{v}_i \end{cases}, \quad (2.12)$$

Rumus 2. 7 Rumus Algoritma PSO

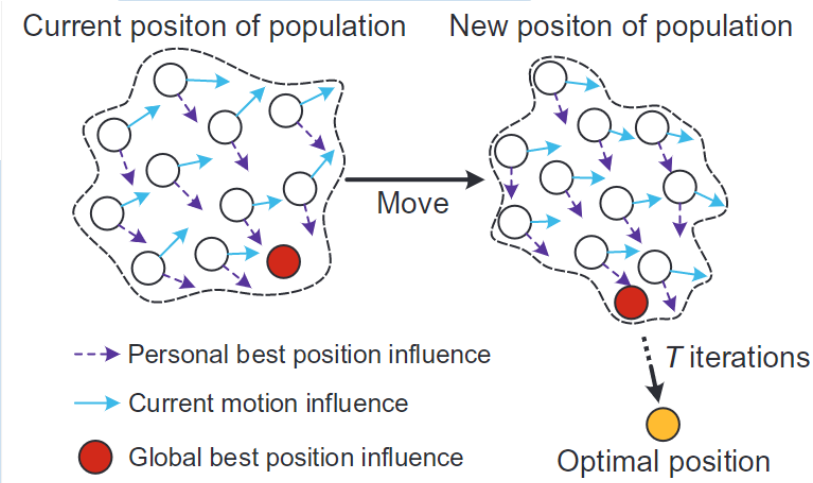
Berdasarkan persamaan (2.12), $\vec{v}_i$ merepresentasikan sebagai *particle's velocity*, $\vec{x}_i$ merepresentasikan sebagai *particle's position*, $\vec{U}(0,\phi_i)$ melambangkan barisan bilangan acak yang terdistribusi secara seragam antara 0 dan ϕ yang baru dihasilkan untuk setiap *particle* pada setiap iterasi, $\otimes$ menunjukkan operasi perkalian elemen secara bersamaan [106]. Untuk memastikan keseimbangan, setiap komponen *velocity vector* $\vec{v}_i$ dibatasi dalam ambang batas *minimum* dan *maxium velocity* yang dinotasikan sebagai $[-V_{max}, +V_{max}]$. *Pseudocode* untuk algoritma PSO dapat dituliskan seperti Tabel 2. 3 [105].

Tabel 2. 3 *Pseudocode* Algoritma PSO

(1) Menyiapkan sebuah *array* partikel dengan koordinat dan kecepatan acak di dalam dimensi 'D'.
(2) Mulai iterasi
(3) Untuk setiap partikel dalam iterasi, tentukan nilai fungsi optimasi yang ditargetkan dalam variabel 'D'.
(4) Evaluasi *fitness* partikel dan bandingkan dengan posisi terbaik yang tercatat (pbest). Jika evaluasi baru lebih baik, perbarui pbest ke ukuran baru ini dan catat koordinat saat ini dari partikel sebagai tempat terbaiknya dalam grid dimensi 'D'.
(5) Kenali partikel yang paling sukses di sekitarnya dan tetapkan indeks partikel ini ke sebuah variabel g.
(6) Modifikasi gerakan dan lokasi setiap partikel menggunakan persamaan (2.12), yang menggabungkan *best positions* yang diidentifikasi oleh partikel individu dan tetangganya.
(7) Lanjutkan iterasi sampai persyaratan tertentu terpenuhi, yang bisa berupa tingkat fitness yang dapat diterima atau batasan jumlah iterasi.
(8) Akhiri *loop* iterasi

Dalam konteks *feature selection*, algoritma PSO dirancang untuk menemukan subset fitur optimal yang dapat meningkatkan kinerja model dengan mengurangi dimensi data sambil mempertahankan atau meningkatkan akurasi klasifikasi [107]. Algoritma PSO merupakan algoritma populer untuk digunakan sebagai optimasi, karena dapat

meningkatkan kinerja algoritma. Untuk mempermudah dalam memahami cara kerja algoritma PSO dapat dilihat ilustrasi Gambar 2. 2.



Gambar 2. 2 Skema Algoritma PSO

Sumber: [108]

2.3.3.2 *Ant Colony Optimization* (ACO)

Algoritma ACO merupakan teknik optimasi yang terinspirasi dari perilaku semut dalam mencari makanan. Saat semut mencari makanan, mereka meninggalkan jalur *pheromone* yang berfungsi sebagai rute untuk memandu mereka kembali ke sarang [104]. Jumlah semut yang melewati jalur tersebut mempengaruhi kepadatan pengendapan dan penguapan *pheromone*. Serta kualitas dan kuantitas makanan yang dibawa semut juga mempengaruhi *pheromone deposition*. Oleh karena itu, semut dapat mengidentifikasi jalur optimal dengan mengikuti jejak dengan *pheromone density* maksimum. Terdapat dua persamaan pada ACO yaitu penemuan jalur optimal dan pembaruan *pheromone* oleh semut ditentukan berdasarkan persamaan (2.13) dan (2.14) masing-masing [109].

$$P\left(\frac{c_i}{s}\right) = \frac{[\tau_i]^{\alpha} \cdot [n(c_i)]^{\beta}}{\Sigma cj \in N(s)[\tau_j]^{\alpha}[n(c_j)]^{\beta}} \quad \forall c_i \in N(s) \tag{2.13}$$

$$\tau_i \leftarrow (1-\rho)\tau_i + \rho \cdot \Sigma\left\{s \leftarrow \frac{S_{upd}}{C_i Es}\right\} w_s \cdot F(s) \quad (2.14)$$

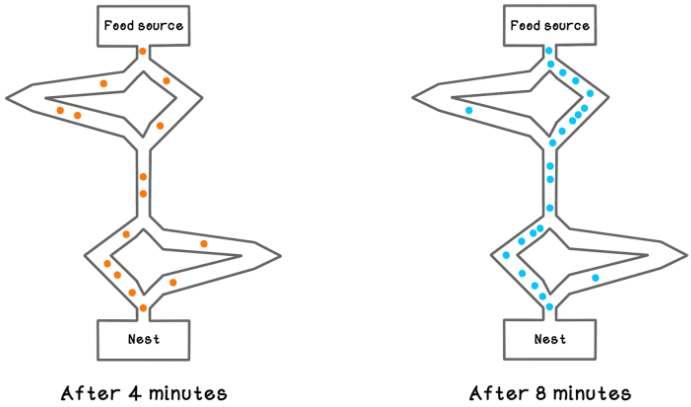
Rumus 2. 8 Rumus Algoritma ACO

Berdasarkan persamaan (2.13) dan (2.14), *pheromone deposition* τ_i yang merepresentasikan pengendapan *pheromone* pada node ith, n adalah *optional weighing function*, c_j mewakiliki setiap solusi yang layak, dan α dan β adalah parameter positif. Adapun pembaruan *pheromone* s_{upd} yang merupakan solusi untuk pembaruan *pheromone*, w_s adalah *weight of solution* dari s, ρ adalah *evaporation constant*, dan $F(s)$ adalah *quality function*. *Pseudocode* untuk algoritma ACO dapat dituliskan seperti Tabel 2. 4 [109].

Tabel 2. 4 *Pseudocode* Algoritma ACO

(1)	Inisialisasi jejak *pheromone*
(2)	*While* (kriteria terminasi belum terpenuhi) *do*
(3)	*For each* semut
(4)	Bangun jalur solusi berdasarkan jejak *pheromone* dan *heuristic information* (2.13)
(5)	Hitung *fitness* dari solusi
(6)	Perbarui jejak *pheromone* lokal (2.14)
(7)	Akhiri iterasi
(8)	Perbaui jejak *global pheromone* berdasarkan solusi terbaik yang ditemukan
(9)	End

Dalam konteks *feature selection*, ACO digunakan untuk menemukan subset fitur yang menghasilkan performa terbaik untuk model dengan meniru cara semut menemukan jalur terpendek dari sarang ke sumber makanan [35]. Semut virtual melakukan iterasi melalui fitur-fitur tersebut, membangun solusi dengan memilih fitur-fitur berdasarkan probabilitas yang dipengaruhi oleh tingkat *pheromone*. Hal ini meningkatkan kemungkinan memilih fitur yang berkontribusi positif terhadap peningkatan akurasi klasifikasi [39]. Gambar 2. 3 merupakan ilustrasi cara kerja algoritma ACO.



Gambar 2. 3 Skema Algoritma ACO

Sumber: [110]

2.3.3.3 *Cat Swarm Optimization* (CSO)

Algoritma CSO merupakan teknik optimasi yang terinspirasi dari perilaku kucing, memanfaatkan dua perilaku spesifik kucing yang dikenal sebagai *seeking mode* dan *tracing mode* [15]. Setiap kucing memiliki posisi yang terdiri dari dimensi M dalam ruang pencarian, dimana setiap dimensi mempunyai *velocity* (kecepatan) masing-masing. *Fitness value* menunjukkan seberapa baik kinerja serangkaian solusi (kucing). Selain itu, terdapat *flag* yang digunakan untuk mengklasifikasikan kucing apakah termasuk ke dalam *seeking mode* atau *tracing mode*. Cara kerja CSO melibatkan penentuan jumlah kucing yang terlibat dalam setiap iterasi dan menjalankannya melalui algoritma. Kucing terbaik pada setiap iterasi disimpan ke dalam memori dan kucing pada iterasi terakhir mewakili solusi terakhir. Algoritma CSO bertujuan untuk menemukan solusi optimal dalam ruang pencarian dengan memanfaatkan perilaku *seeking mode* dan *tracing mode* yang terinspirasi dari kucing. Dalam *seeking mode*, kucing secara acak menjelajahi atau mengamati sekelilingnya untuk menemukan posisi yang lebih baik. Dalam *tracing mode* algoritma CSO, kucing bergerak menuju sasaran dengan *velocity* yang dihitung secara matematis. Algoritma CSO dinyatakan dalam persamaan (2.15), (2.16), (2.17), dan (2.18) meliputi kedua mode kucing tersebut [111].

$$xjd_{new} = (1 + rand \cdot SRD) \cdot xjd_{old} \quad (2.15)$$

$$Pi = \frac{|FS_i - FS_b|}{FS_{max} - FS_{min}}, where\ 0 < i < j \quad (2.16)$$

Rumus 2. 9 Rumus Algoritma CSO dalam *Seeking Mode*

Persamaan (2.15) dan (2.16) merupakan persamaan untuk *seeking mode* pada algoritma CSO. *Seeking mode* pada algoritma CSO meniru perilaku istirahat kucing. Terdapat empat parameter penting dalam mode ini yaitu *Seeking Memory Pool* (SMP), *Seeking Range of Selected Dimension* (SRD), *Count of Dimensions to Change* (CDC), dan *Self-Position Considering* (SPC) yang merupakan nilai yang ditetapkan secara manual. Pada setiap iterasi CSO, pilih secara acak dimensi CDC yang akan dimutasi. Tambahkan atau kurangi nilai acak dalam SRD dari nilai saat ini, ganti posisi lama xjd_{old} dengan posisi baru xjd_{new}, seperti yang ditunjukkan pada persamaan (2.15). Di sini, xjd_{new} mewakili posisi berikutnya, j menunjukkan indeks kucing, d berarti dimensi, dan $rand$ adalah bilangan acak dalam interval 0 dan 1. Berdasarkan probabilitas, pilih salah satu kandidat yang akan ditunjuk jadilah posisi selanjutnya untuk kucing. Titik kandidat dengan *fitness value* (FS) yang lebih tinggi kemungkinan besar untuk terpilih, seperti yang ditunjukkan pada persamaan (2.16). Namun, jika semua *fitness value* sama, atur probabilitas pemilihan masing-masing kandidat poin menjadi 1. Jika tujuannya untuk minimalisasi tetapkan $FS_b = FS_{max}$ sebaliknya, jika sasarannya untuk maksimalisasi tentukan $FS_b = FS_{min}$. Tabel 2. 5 merupakan *pseudocode* algoritma CSO dalam *seeking mode* [112].

Tabel 2. 5 Pseudocode Algoritma CSO dalam *Seeking Mode*

(1) Buat j instansi dari posisi saat ini dari kucing, dimana j = SMP. Jika SPC adalah kondisi yang benar, biarkan j = (SMP − 1), pertahankan posisi saat ini sebagai salah satu opsi di antara kandidat yang munkin.
(2) Untuk setiap instansi, sesuai dengan CDC, secara acak tingkatkan atau turunkan persentase SRD dari nilai-nilai yang ada dan gantikan yang sebelumnya.
(3) Hitung *fitness values* (FS) untuk semua *candidate points*.

(4) Jika tidak semua FS identik, hitung probabilitas pemilihan setiap titik kandidat dengan persamaan (2.16); jika tidak atur semua probabilitas pemilihan setiap titik kandidat menjadi 1.
(5) Pilih secara acak titik untuk berpindah dari titik-titik kandidat, dan ganti posisi kucing k.

$$V_{k,d} = V_{k,d} + r_1 c_1 \left(x_{best,d} - x_{k,d}\right), \quad (2.17)$$

$$where\ d = 1,2,\ldots,M$$

$$V_{k,d} = V_{k,d} + V_{k,d} \quad (2.18)$$

Rumus 2. 10 Rumus Algoritma CSO dalam *Tracing Mode*

Persamaan (2.17) dan (2.18) merupakan persamaan *tracing mode* pada algoritma CSO. Pada iterasi pertama dalam *tracing mode* algoritma CSO, *velocity value* ditetapkan secara acak untuk semua dimensi posisi kucing, Namun *velocity value* perlu diperbarui untuk setiap dimensi sesuai persamaan (2.17) untuk langkah selanjutnya. Jika *velocity* melebihi nilai maksimum dari yang diperbolehkan, maka diatur ke maximum *velocity*. Perbarui posisi kucing berdasarkan persamaan (2.18). Tabel 2. 6 merupakan *pseudocode* algoritma CSO dalam *tracing mode* mengacu pada persamaan di atas [112].

Tabel 2. 6 *Pseudocode* Algoritma CSO dalam *Tracing Mode*

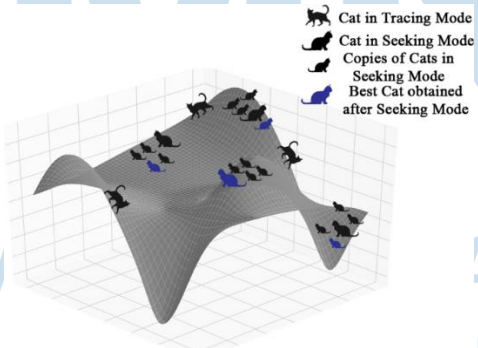
(1) Perbarui *velocities* untuk setiap dimensi $V_{k,d}$ seperti yang ditunjukkan dalam persamaan (2.17).
(2) Pastikan bahwa *velocities* berada dalam batas maksimal *velocities* yang diizinkan. Sesuaikan *velocity* apa pun yang melebihi batas ini kembali ke batas maksimum.
(3) Sesuaikan lokasi kucing k, sesuai dengan persamaan (2.18).

Berdasarkan kedua mode algoritma CSO yaitu *seeking mode* dan *tracing mode*. *Seeking mode* digunakan untuk eksplorasi ruang pencarian dengan cara memvariasikan posisi kucing, sedangkan *tracing mode* digunakan untuk ekploitasi ruang pencarian dengan memfokuskan pencarian pada area yang telah identifikasi sebagai area yang mengandung solusi terbaik. Tabel 2. 7 merupakan *pseudocode* algoritma CSO dari kombinasi kedua mode tersebut [111].

Tabel 2. 7 *Pseudocode* Algoritma CSO

(1) Inisialisasi dengan membuat ‘N’ kucing dalam algoritma. (2) Tempatkan kucing secara acak dalam area pencarian ‘M’-dimensi, menetapkan *velocities* yang berada dalam batas maksimum yang telah ditentukan. Tentukan secara acak jumlah kucing untuk terlibat dalam *tracing mode* berdasarkan *Mixing Ratio* (MR), menempatkan sisanya dalam *seeking mode*. (3) Hitung *fitness* untuk setiap kucing menggunakan *fitness function* yang mengukur kedekatan mereka terhadap tujuan, dan ingat lokasi kucing paling optimal xbest. (4) Pindahkan kucing berdasarkan mode yang ditugaskan: yang dalam *seeking mode* menjalani proses yang berbeda, sementara yang dalam *tracing mode* menyesuaikan *velocity* dan posisi mereka menurut rumus spesifik. (5) Secara selektif mengubah sejumlah kucing kembali ke *tracing mode* sesuai dengan MR, dan sisanya terus dalam *seeking mode*. (6) Periksa jika kondisi akhir algoritma telah terpenuhi; jika ya, hentikan algoritma, jika tidak, ulangi langkah 3 hingga 5 kembali.

Dalam konteks *feature selection*, algoritma CSO beroperasi dengan mengeksplorasi ruang fitur untuk menemukan kombinasi fitur yang optimal. Melalui iterasi antara *seeking mode* dan *tracing mode*, CSO secara adaptif mengeksplorasi dan memanfaatkan informasi dari ruang fitur untuk mengidentifikasi *subset* fitur yang memberikan performa terbaik untuk model yang digunakan. Proses pada algoritma CSO bertujuan untuk menyeimbangkan eksplorasi (mencari kombinasi fitur yang berbeda) dan eksploitasi (mengikuti posisi yang menjanjikan) untuk mendapatkan solusi optimal. Gambar 2. 4 merupakan ilustrasi cara kerja algoritma CSO.



Gambar 2. 4 Skema Algoritma CSO

Sumber: [113]

2.4 *Tools* Penelitian

2.4.1 JupyterLab

JupyterLab merupakan pengembangan dari Jupyter Notebook yang diluncurkan pada tahun 2018. Dibandingkan dengan Jupyter Notebook, JupyterLab menawarkan pengalaman pengguna yang lebih fleksibel dan canggih melalui *User Interface* (UI) yang telah diperbarui [114]. Pengguna juga dapat dengan mudah mengatur dan mengelola lingkungan kerja mereka sesuai preferensi masing-masing. Salah satu keunggulan JupyterLab adalah kemampuannya untuk mengintegrasikan berbagai alat dan fitur yang diperlukan oleh pengguna. Selain menyediakan fitur seperti *text editor*, JupyterLab juga menyediakan *terminal* yang dapat menjalankan perintah skrip yang ditulis. JupyterLab juga menawarkan beragam *extension* yang dapat memperluas fungsionalitasnya sehingga pengguna dapat memperkaya dan menyesuaikan pengalaman pengguna mereka [115].

2.4.2 Python

Python adalah bahasa pemrograman tingkat tinggi yang sering digunakan dalam pengembangan berbagai proyek aplikasi, analisis data, *Artificial Intelligence* (AI), pengembangan *web*, dan banyak bidang lainnya [116]. Python didesain dengan fokus utama pada keterbacaan kode dan sintaks yang sederhana sehingga mudah dipahami dan digunakan oleh para *developer*. Python memiliki berbagai *library* yang sangat luas, sehingga mempermudah dalam pengembangan aplikasi, contohnya *library* seperti NumPy, Pandas, dan MatPlotLib digunakan secara luas dalam analisis data dan komputasi ilmiah. Kelebihan lainnya dari bahasa pemrograman ini yaitu komunitas yang besar dan aktif yang menyediakan berbagai *resources*, dokumentasi, dukungan, sehingga mempermudah *developer* untuk mempelajari dan memahami bahasa Python [117].