BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian ini menggunakan beberapa penelitian terdahulu yang berkaitan dengan penggunaan algoritma LSTM penggunaan aktivasi dan optimasi dalam melakukan prediksi harga komoditas emas dan perak. Pada tabel 2.1 merupakan beberapa penelitian terdahulu yang digunakan.

Tabel 2. 1 Tabel penelitian terdahulu

Judul Artikel	Penerbit	Tahun	Penulis	Algoritma	Hasil
How can we predict transportation stock prices using artificial intelligence? Findings from experiments with Long Short-Term Memory based algorithms [23]	*International Journal of Information Management Data Insights*	2024	Dinar A. Kristiyanti, Willibrordus B. N. Pramudya, Samuel A. Sanjaya	LSTM dengan kombinasi fungsi aktivasi (Linear, ReLU, Sigmoid, Tanh) dan optimizers (Adam, AdaGrad, Nadam, RMSProp, AdaDelta, SGD, Adamax)	Model LSTM dengan ReLU *activation* dan Adam *optimizer* terbukti menghasilkan akurasi terbaik dalam prediksi harga saham transportasi, dengan R^2 mencapai 96%, MAE sebesar 0.00929, dan MAPE hanya 0.0642.
Development of stock price prediction system using Flask framework and LSTM algorithm [24]	*Journal of Infrastructure, Policy and Development*	2023	Kefas Bagastio, Raymond S. Oetama, Arief Ramadhan	LSTM	Prediksi harga saham bank besar di Indonesia dengan LSTM; semua model mencapai MAPE < 10% (*Highly Accurate*) dengan MAPE terendah di terapkan pada bank BMRI.JK
Gold Price Prediction Using the ARIMA and LSTM Models [18]	Sinkron: Jurnal dan Penelitian Teknik Informatika	2023	Yudha R. Madhika, Kusrini, Tonny Hidayat	ARIMA, LSTM	LSTM unggul (RMSE: 8.124, MAPE: 0.023) dibanding ARIMA dalam memprediksi harga emas.

Judul Artikel	Penerbit	Tahun	Penulis	Algoritma	Hasil
Comparative Analysis of Deep Learning Models for Silver Price Prediction: CNN, LSTM, GRU and Hybrid Approach[21]	*Akdeniz İİBF Dergisi*	2024	Yunus Emre GÜR	CNN, LSTM, GRU, CNN-LSTM-GRU (hybrid)	Model hybrid CNN-LSTM-GRU memberikan hasil terbaik dalam prediksi harga perak harian (XAG/TRY) dengan akurasi tinggi. Hasil uji data: RMSE: 0.1089, MAE: 1.4789%, MAPE: 0.0170, R^2: 0.9913, MASE: 0.9846, SMAPE: 0.9678. Jauh lebih unggul dibanding model tunggal.
A Novel Bitcoin and Gold Prices Prediction Method Using an LSTM-P Neural Network Model[25]	*Computational Intelligence and Neuroscience*	2022	Xinchen Zhang, Linghao Zhang, Qincheng Zhou, Xu Jin	LSTM-P (optimized LSTM + wavelet transform)	Model LSTM-P (Long Short-Term Memory Plus) menghasilkan prediksi harga emas dan Bitcoin yang lebih akurat dibanding LSTM biasa. MAPE turun dari 6.08% (LSTM) menjadi 4.81%, R^2 naik dari 0.7973 menjadi 0.8862.
Comparison of commodity prices by using machine learning models in the COVID-19 era[22]	*Turkish Journal of Engineering*	2023	Sena Alparslan, Tamer Uçar	LSTM, SVR, Random Forest	LSTM memberikan hasil terbaik untuk prediksi harga perak selama COVID-19 (MAPE: 0.0173), sedangkan SVR unggul untuk harga emas selama COVID-19 (MAPE: 0.0149). Sebaliknya, LSTM paling akurat di periode pra-COVID-19 untuk kedua komoditas. Pada periode pra-COVID-19, algoritma LSTM memberikan hasil prediksi terbaik untuk harga emas dengan nilai MAPE sebesar 0.0182 dan

Judul Artikel	Penerbit	Tahun	Penulis	Algoritma	Hasil
					untuk harga perak, LSTM juga menjadi model paling akurat dengan MAPE sebesar 0.0216.
Deep learning systems for forecasting the prices of crude oil and precious metals [26]	*Foroutan and Lahmiri Financial Innovation*	2024	Parisa Foroutan, Salim Lahmiri	16 models: LSTM, BiLSTM, GRU, CNN, TCN, BiGRU, hybrids, LightGBM, etc.	Model BiGRU memberikan hasil prediksi terbaik untuk harga emas, dengan nilai MAE sebesar 15.188 menggunakan *input sequence* 30 hari. Sementara itu, untuk harga perak, model TCN terbukti paling akurat dengan MAE terendah yaitu 0.346. Hasil ini menunjukkan bahwa pendekatan *deep learning* seperti BiGRU dan TCN sangat efektif dalam memprediksi harga logam untuk *time frame* harian.
Gold Price Forecasting Using LSTM, Bi-LSTM and GRU [15]	*European Journal of Science and Technology*	2021	Mustafa Yurtsever	LSTM, Bi-LSTM, GRU	Model LSTM memberikan hasil terbaik dengan nilai MAPE: 3.48, RMSE: 61.728, dan MAE: 48.85. Model ini lebih unggul dibanding Bi-LSTM dan GRU dalam memprediksi harga emas.
Forecasting Gold Price in Rupiah using Multivariate Analysis with LSTM and GRU Neural Networks [19]	*Advances in Science, Technology and Engineering Systems Journal*	2021	Sebastianus B. Primananda, Sani M. Isa	LSTM dan GRU	Untuk periode < 3 tahun, GRU lebih akurat (MAPE serendah 1.30%). Untuk periode > 3 tahun, LSTM lebih unggul (MAPE serendah 0.72%). Grid search meningkatkan akurasi signifikan.
Gold price prediction by a CNN-Bi-LSTM model along with automatic	PLOS ONE	2024	Amirhossein Amini, Robab Kalantari	CNN-Bi-LSTM, CNN-LSTM,	Model CNN-Bi-LSTM memberikan hasil prediksi harga emas terbaik dengan nilai R^2: 0.95,

Judul Artikel	Penerbit	Tahun	Penulis	Algoritma	Hasil
parameter tuning [20]				ConvLSTM, LSTM, CNN	RMSE: 34.87, dan RMAE: 5.15.
Prediksi harga emas menggunakan metode LSTM dan GRU [27]	JITET (Jurnal Informatika dan Teknik Elektro Terapan)	2023	Nanda Kurnia Agusmawati, Fitwatul Khoiriyah, Abu Tholib	LSTM, GRU	Model LSTM dengan batch 32 dan epoch 400 menghasilkan MAE: 0.0389, RMSE: 0.0475, MAPE: 5.2047%, dan lebih baik dari GRU (MAPE: 6.0688%).
How good are different machine and deep learning models in forecasting the future price of metals? Full sample versus sub-sample [28]	*Resources Policy*	2024	Anu Varshini, Parthajit Kayal, Moinak Maiti	Stacked LSTM, ConvLSTM, Bi-LSTM, SVR, XGBoost, GRU	Model Stacked LSTM memberikan hasil paling akurat untuk logam emas dan aluminium, dengan nilai MAPE masing-masing sebesar 0,74% dan 1,59% pada data *full sample* dengan *input* 60 hari. Sementara itu, model Bidirectional LSTM (Bi-LSTM) unggul dalam memprediksi harga perak, platinum, dan paladium, dengan MAPE rendah yaitu 1,52%, 1,57%, dan 2,58%.
A Comprehensive Overview and Comparative Analysis on Deep Learning Models		2024	Farhad Mortezapour Shiri, Thinagaran Perumal, Norwati Mustapha, Raihani Mohamed	CNN, RNN LSTM, Bi-LSTM, GRU, dan TCN	LSTM menunjukkan kinerja yang jauh lebih unggul dibanding RNN, dengan akurasi meningkat dari 59,03 % menjadi 87,53 % pada dataset IMDB dan dari 93,17 % menjadi 93,29 % pada dataset ARAS .

Dari arikel jurnal penelitian terdahulu pada tabel 2.1 membuktikan bahwa penggunaan algoritma Long Short-Term Memory (LSTM) terbukti efektif dan relevan dalam berbagai penelitian prediksi harga emas dan perak. LSTM memiliki kemampuan untuk memproses data *time-series* dan mengelola data yang besar, sehingga dapat digunakan untuk mengikuti pergerak pasar yang begitu cepat. Dalam berbagai studi, baik untuk prediksi

harga logam mulia maupun saham, LSTM menunjukkan hasil yang konsisten dan akurat dibandingkan metode statistik atau algoritma *machine learning* lainnya. Dapat dibuktikan sebagai berikut Untuk periode > 3 tahun, LSTM lebih unggul (MAPE serendah 0.72%). *Grid search* meningkatkan akurasi signifikan [19].

Selain arsitektur vanilla LSTM, sejumlah penelitian mengembangkan varian seperti Stacked LSTM dan Bi-LSTM[28]. Pengaruh konfigurasi pelatihan seperti epoch, batch size, serta pembagian data *training-testing* juga menjadi faktor penting dalam meningkatkan akurasi model. Penerapan *tuning* LSTM dengan *batch size* 32 dan epoch 400 menghasilkan MAE 0.0389 dan MAPE 5.2047%, lebih baik dibandingkan GRU[27]. Setiap arsitektur memiliki karakteristik masing-masing dalam menangkap pola data yang lebih kompleks. Penggunaan *optimizer* seperti AdaDelta, AdaGrad, Adam, Nadam, RMSProp, dan SGD, serta fungsi *activation* seperti Linear, ReLU, Sigmoid, dan Tanh, turut diuji untuk meningkatkan efisiensi pelatihan dan akurasi prediksi. Kombinasi LSTM dengan ReLU *activation* dan Adam *optimizer* menghasilkan R^2 hingga 96%, MAE 0.00929, dan MAPE 0.0642, menjadikannya konfigurasi terbaik untuk prediksi harga saham transportasi [23].

Berdasarkan hasil beberapa penelitian terdahulu, algoritma LSTM terbukti lebih unggul dibandingkan CNN, ANN, dan GRU dalam menangani data sekuensial seperti time-series. LSTM memiliki kemampuan yang kuat dalam mengenali pola jangka panjang dan mengatasi permasalahan vanishing gradient, sehingga menghasilkan akurasi prediksi yang lebih tinggi [9]. Hal ini membuat LSTM sangat efektif dan relevan untuk digunakan dalam prediksi harga atau tren yang bersifat temporal, termasuk prediksi harga komoditas seperti emas dan perak.

Dari jurnal penelitian sebelumnya memiliki beberapa kelemahan dalam melakukan penelitian, keterbatasan dalam penggunaan *dataset* dengan menggunakan periode waktu yang singkat. Hal tersebut tentunya mempengaruhi hasil prediksi seperti pada, performa model LSTM yang optimal pada data sebelum pandemi belum tentu sama saat menghadapi kondisi pandemi atau krisis ekonomi [22]. Kemudian pada penggunaan model LSTM dengan menambahkan varian *hybrid* yang kompleks membutuhkan komponen sumber daya komputasi yang mendukung karena dalam memproses membutuhkan waktu yang lama dan dinilai kurang efisien [26].

2.2 Tinjauan Teori

2.2.1 Invesasi

Investasi merupakan suatu tindakan meletakan uang pada suatu aset, dengan harapan aset tersebut menghasilkan keuntungan secara jangka panjang. Biasanya investasi bertujuan untuk mendapatkan keuntungan yang signifikan di masa depan. Dalam berinvestasi jangka Panjang diperlukan beberapa hal yang perlu dipertimbangkan antara lain kebijakan dividen, kebijakan utang, profitabilitas perusahaan, risiko investasi, dan lainnya [29]. Dalam melakukan investasi pastinya dibutuhkan manajemen resiko agar terhindar dari kerugian yang besar maka dari itu diperlukannya pengelola resiko dengan cara melakukan identifikasi, evaluasi, dan mengatasi resiko [30].

2.2.2 Emas

Emas adalah salah satu komoditas paling berharga dan dicari di dunia. Emas sering dijadikan sebagai aset investasi karena dianggap sebagai aset *safe heaven* atau nilai yang cenderung naik dan stabil, Maka dari itu emas dinilai berperan penting terutama pada aspek ekonomi dan keuangan. Emas telah lama dianggap sebagai perlindungan terhadap inflasi, mata uang yang melemah, dan ketidakpastian ekonomi[31]. Emas dapat di beli dengan kontrak berjangka. Kontrak berjangka emas adalah salah satu bentuk pengembangan bisnis dan menjadi bagian dari instrumen derivatif. Kontrak berjangka emas awalnya diperdagangkan dengan motif lindung nilai, tetapi sekarang juga diperdagangkan dengan motif spekulatif untuk mendapatkan keuntungan dari fluktuasi harga.

2.2.3 Perak

Komoditas perak adalah istilah yang digunakan untuk mengacu pada perak yang merupakan salah satu komoditas logam mulia yang diperdagangkan di pasar komoditas. Komoditas perak sering disebut sebagai "emas putih" atau dibandingkan dengan emas karena memiliki beberapa kesamaan dalam konteks investasi dan penggunaan. Sebagai komoditas, perak adalah barang fisik yang memiliki nilai ekonomi dan diperdagangkan di berbagai bursa komoditas di seluruh dunia [32].

2.2.4 Kontrak berjangka

Kontrak berjangka, juga dikenal sebagai kontrak *futures*, merupakan perjanjian standar antara kedua belah pihak untuk melakukan perdagangan menjual dan membeli suatu aset atau komoditas di masa yang akan mendatang dengan harapan menghasilkan keuntungan dari nilai jual atau beli sesuai dengan yang diperkirakan [32]. Kontrak ini digunakan di pasar berjangka untuk tujuan lindung nilai *hedging* atau spekulasi. Biasanya pembelian kontrak berjangka menggunakan

leverage. Leverage adalah penggunaan dana pinjaman yang disediakan oleh bank atau broker untuk meningkatkan keuntungan dalam *investasi* [33].

2.2.5 Data *time series*

Data *time series* merupakan kumpulan pengamatan data dan pengukuran yang diambil dari waktu ke waktu, biasanya data harga atau observasi berurutan dari suatu variabel pada interval waktu tertentu. Dengan menggunakan data *time series* digunakan pada saat untuk melakukan analisis tren atau prediksi suatu harga [34]. Penggunaan data *time series* bisanya digunakan untuk melakukan analisis ekonomi, keuangan, dan lainnya. Sehingga sesuai untuk diterapkan pada penelitian ini.

2.2.6 *Deep learning*

Deep learning adalah salah bagian dari dari *machine learning* yang menggunakan arsitektur jaringan saraf tiruan *neural network* yang terdiri dari banyak lapisan (layer) untuk memproses data [35]. Secara umum, struktur dalam *deep learning* terdiri dari tiga jenis lapisan utama, yaitu *input layer, hidden layer*, dan *output layer. Input layer* berfungsi sebagai pintu masuk data ke dalam jaringan saraf. *Hidden layer* berada di antara *input* dan *output* merupakan tempat berlangsungnya proses transformasi atau pemrosesan data. Sementara itu, *output* layer menghasilkan pemrosesan yang dilakukan pada *hidden* layer, yang kemudian ditampilkan sebagai prediksi atau hasil model [36]. *Deep learning* sendiri telah berhasil diterapkan pada berbagai bidang seperti pengenalan suara, pengenalan wajah, pengenalan gambar, pengenalan tulisan tangan, dan pengenalan objek lainnya.

2.2.7 *Performance Metrics*

Dalam pengujian dan evaluasi model prediksi, khususnya yang berbasis *time series* seperti harga komoditas, diperlukan ukuran kuantitatif yang dapat menunjukkan seberapa baik model dalam memprediksi nilai dibandingkan dengan data aktual. Ukuran ini disebut sebagai *performance metrics*, yang digunakan untuk mengukur tingkat kesalahan atau ketepatan dari model[37] [38]. Beberapa metrik evaluasi yang umum digunakan adalah sebagai berikut:

1. *Mean squared error* (MSE)
   MSE mengukur rata-rata kuadrat dari selisih antara nilai aktual dan nilai prediksi. Metrik ini memberikan penalti lebih besar untuk kesalahan besar karena sifat kuadratnya. Semakin kecil

nilai MSE, maka semakin baik performa model. Berikut merupakan rumusnya (2.1)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

(2. 1)[23]

2. *Root mean squared error* (RMSE)
   RMSE merupakan akar dari MSE, dan memberikan satuan yang sama dengan data aslinya sehingga lebih mudah diinterpretasikan. RMSE sensitif terhadap outlier dan cocok digunakan untuk model yang memprioritaskan kesalahan besar. Berikut merupakan rumusnya (2.2)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

(2. 2) [23]

3. *Mean absolute error* (MAE)
   MAE mengukur rata-rata nilai absolut dari perbedaan antara nilai aktual dan prediksi. Metrik ini lebih toleran terhadap outlier dibandingkan MSE. MAE memberikan gambaran seberapa besar kesalahan prediksi secara umum dalam satuan yang sama dengan data. Berikut merupakan rumusnya (2.3)

$$MAE = \frac{\sum_{n=1}^{N}|\hat{r}_n - r_n|}{n}$$

(2. 3)[23]

4. *Mean absolute percentage error* (MAPE)
   MAPE merupakan perbedaan antara nilai prediksi dengan nilai *actual* yang dihitung berdasarkan absolut rata-rata, di nyatakan dalam bentuk persentase dari nilai sebenarnya. MAPE ideal digunakan saat nilai aktual tidak mendekati nol, karena bisa menghasilkan nilai ekstrem jika pembagi mendekati nol. Berikut merupakan rumusnya (2.4)

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{yl - y}{yi} \right|$$

(2. 4)[23]

5. R^2 / *R-squared*

   R^2 mengukur seberapa besar variasi pada data aktual yang dapat dijelaskan oleh model prediksi. Nilai R^2 berkisar dari 0 hingga 1 (atau negatif jika model sangat buruk). Semakin mendekati 1, maka semakin baik model dalam menjelaskan variabilitas data. Berikut merupakan rumusnya (2.5)

$$\text{R – Squared / } R^2 = 1 - \frac{\sum_{i=1}^{N} (X_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (\bar{y}_i - Y_i)^2}$$

(2. 5)[23]

6. *Resisual error*

   Residual error adalah selisih antara nilai aktual dan nilai prediksi (*residual* = aktual - prediksi). Analisis *residual* penting untuk mengevaluasi distribusi kesalahan model, mengidentifikasi *autokorelasi, heteroskedastisitas*, dan stabilitas model.

7. *Kolmogorov-Smirnov Test*

   Kolmogorov–Smirnov Test atau K–S *Test* merupakan salah satu uji statistik yang dapat digunakan untuk menilai apakah dua data sampel berasal dari distribusi yang serupa sama. Uji K–S menghitung perbedaan maksimum antara fungsi distribusi kumulatif empiris (ECDF) dari masing-masing sampel (2.6):

$$Fx(x) = \frac{1}{n_1} \sum_{i=1}^{n_1} 1\,[x_i \leq x],\ F_Y(y) = \frac{1}{n^2} \sum_{i=1}^{n_2} 1\,|y_i \leq y|$$

(2. 6)*[33]*

   statistik uji K–S, yaitu D, adalah perbedaan maksimum absolut antara ECDF kedua sampel (2.7):

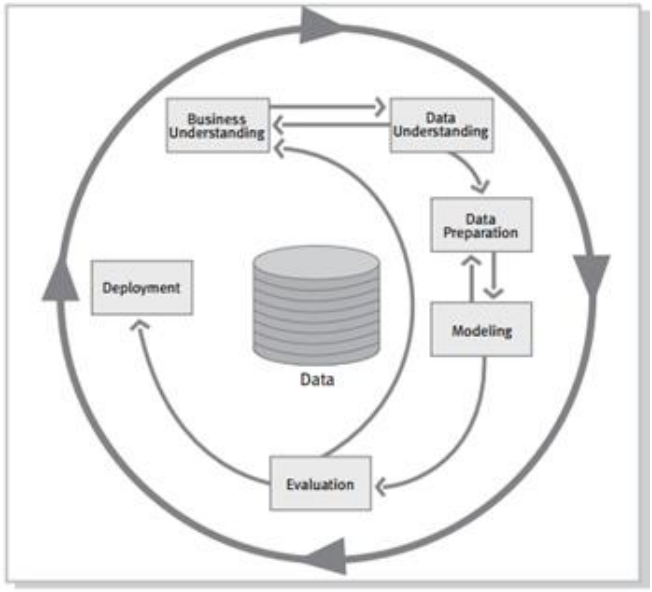
$$D = sup|Fx(x) - F_Y(x)|$$

(2. 7) [39]

2.3 Kerangka dan Algoritma Penelitian

2.3.1 CRISP-DM

CRISP-DM *framework Cross-Industry Standard Process for Data Mining* merupakan teknik metodologi yang digunakan dalam proses data *mining* dan analisis data. Penerapan CRISP-DM yaitu bertujuan untuk membantu dalam melakukan analisis data dalam melakukan proses data *mining* secara sistematis dan efektif [40]. CRISP-DM membantu dalam merancang, melaksanakan, dan mengevaluasi analisis data secara efektif, serta membantu untuk memastikan bahwa hasil analisis data dapat diaplikasikan dalam konteks bisnis yang sesuai [41]. Berikut merupakan enam tahap dalam metodologi CRISP-DP, seperti *business understanding, data understanding, data preparation, modeling, evaluation,* dan *deployment*. Berikut merupakan Langkah-langkah dalam mengimplementasikan CRISP-DM, yaitu:



Gambar 2. 1 CRISP-DM [41]

1. *Business Understanding*
   Business understanding adalah tahap awal dalam pengembangan model data *mining*. Pada tahap ini, dilakukan pemahaman terhadap masalah bisnis yang ingin dipecahkan dan tujuan dari pengembangan model data *mining*. Pada tahap ini, juga dilakukan identifikasi terhadap data yang tersedia dan sumber data yang dapat digunakan.
2. *Data Understanding*
   Data understanding adalah tahap untuk memahami data yang tersedia dan mempersiapkan data untuk tahap

selanjutnya. Pada data *understanding*, dilakukan analisis data yang akan digunakan selama penelitian ini berlangsung, seperti statistik deskriptif, visualisasi data, dan identifikasi *outlier*.

3. *Data Preparation*
   Data preparation adalah tahap untuk mempersiapkan data agar siap digunakan pada tahap pemodelan. Pada tahap ini, dilakukan pembersihan data, penggabungan data, penghilangan data yang tidak relevan, dan pengisian data yang hilang.
4. *Modeling*
   Modeling adalah tahap untuk membangun model data *mining*. Pada tahap ini, dilakukan pemilihan algoritma yang tepat, pengaturan parameter algoritma, dan pembangunan model.
5. *Evaluation*
   Evaluation adalah tahap untuk mengevaluasi model yang telah dibangun. Pada tahap ini, dilakukan pengujian model terhadap data yang belum pernah dilihat sebelumnya, dan dilakukan evaluasi terhadap performa model.
6. *Deployment*
   Deployment adalah tahap untuk mengimplementasikan model yang telah dibangun ke dalam lingkungan produksi. Pada tahap ini, dilakukan integrasi model ke dalam sistem yang ada, dan dilakukan pemantauan terhadap performa model.

2.3.2 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) adalah suatu tipe dari struktur jaringan saraf tiruan yang diterapkan untuk mengolah informasi berurutan, seperti teks, suara, dan video dalam data berurutan atau *sequential*. LSTM dikembangkan untuk mengatasi masalah *vanishing gradient* pada jaringan saraf tiruan biasa *feedforward neural network* yang mengalami kesulitan dalam memproses data berurutan yang panjang. Dengan menggunakan LSTM dapat mengingat informasi yang berguna dan menghilangkan informasi yang kurang relevan. LSTM terdiri dari tiga gerbang utama yaitu *gate* berupa *forget gate, input gate*, *cell state update*, dan *output gate*, serta sel memori yang digunakan untuk menyimpan informasi jangka panjang. Berikut merupakan penjelasan dari *gate* tersebut, yaitu:

1. *Forget gate*, merupakan tahap pertama yaitu dengan menggunakan fungsi sigmoid atau yang biasa dikenal sebagai f_t *forget gate* dilakukan untuk mengidentifikasi data yang perlu dihapus dan disimpan pada *memory cell*.

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

(2. 8) [42]

Di mana:

f_t = nilai dari *forget gate*
σ = fungsi sigmoid
W_f = bobot untuk nilai *input*
h_{t-1} = nilai *output* dari waktu ke $t-1$
x_t = nilai *input* pada waktu ke t
b_i = bias pada *input gate*

2. *Input gate*, pada tahap *input gate* data yang sudah dipilih kemudian disimpan ke *cell state*. Pada *input gate* terdapat dua bagian yaitu *gate* dan tanh. Pada *input gate* i_t berguna untuk menentukan nilai mana yang perlu diperbaharui dengan menggunakan fungsi dari sigmoid dan tanh $\tilde{C}_t$ yang bertujuan untuk mendapatkan nilai baru dari fungsi tanh.

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

(2. 9) [42]

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_C$$

(2. 10) [42]

Di mana:

i_t = nilai dari *input gate*
σ = fungsi sigmoid
W_i = bobot untuk nilai *input* pada waktu ke t
h_{t-1} = nilai *output* dari waktu ke $t-1$
x_t = nilai *input* pada waktu ke t
b_i = bias pada *input gate*
$\tilde{C}_t$ = nilai kandidat *cell state*
$tanh$ = fungsi hiperbolik tangent
W_c = bobot untuk nilai *input cell* ke c
b_c = bias pada *cell* ke c

3. *Cell state update*, merupakan tahap ketiga di mana berfungsi untuk memperbaharui *cell state* yang lama

menjadi *cell state* yang baru. Pembaruan *call state* ini berlangsung dengan cara mengalihkan persamaan antara (2.8) dengan *cell state* yang sebelumnya, kemudian *call state* tersebut akan ditambahkan dengan persamaan (2.9) dan (2.10) sehingga menghasilkan *cell state* baru.

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

(2. 11) [11]

Di mana:

$\tilde{C}_t$ = nilai kandidat *cell state*
f_i = nilai *forget gate*
C_{t-1} = nilai *memory cell state* pada *cell* sebelum
i_t = nilai dari *input gate*
$\tilde{C}_t$ = nilai kandidat *cell state*

4. *Output gate*, merupakan lapisan terakhir di mana untuk menentukan hasil akhir *output*. Penggunaan lapisan sigmoid untuk menentukan bagian dari *cell state* yang akan dijadikan hasil *output* (2.12). Kemudian dari hasil *output* tersebut dilalui ke dalam tanh layer dan akan dikalikan dengan sigmoid untuk menjaga stabilitas akurasi (2.13).

$$o_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_o$$

(2. 12) [11]

$$h_t = o_t \cdot \tanh(C_t)$$

(2. 13) [11]

Di mana:

o_t = nilai dari *output gate*
σ = fungsi sigmoid
W_i = bobot untuk nilai *input* pada waktu ke t
h_{t-1} = nilai *output* dari waktu ke $t-1$
x_t = nilai *input* pada waktu ke t
b_o = bias pada *output gate*
h_t = *output final*
o_t = nilai dari *output gate*
$tanh$ = fungsi hiperbolik tangent
C_t = nilai *memory cell state*

Pengembangan model LSTM, menghasilkan beberapa variasi arsitektur, yaitu Vanilla LSTM, Stacked LSTM, dan Bidirectional LSTM atau Bi-LSTM. Berikut merupakan penjelasan dari ketiga arsitektur tersebut, yaitu:

1. Vanilla LSTM adalah versi standar dari LSTM yang hanya memiliki satu lapis tersembunyi dan dalam melakukan pemrosesan data hanya satu arah[42]. Berikut merupakan gambarnya:
   Forget gate:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f \quad (2.\ 14)\ [42]$$

   Input gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i \quad (2.\ 15)\ [42]$$

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_C \quad (2.\ 16)\ [42]$$

   Call state update:

$$C_t = f_i \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (2.\ 17)\ [11]$$

   Output gate:

$$o_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_o \quad (2.\ 18)\ [11]$$

$$h_t = o_t \cdot \tanh(C_t) \quad (2.\ 19)\ [11]$$

   Di mana:
   f_t = nilai dari *forget gate*
   σ = fungsi sigmoid
   W_f = bobot untuk nilai *input*
   h_t = *output final*
   h_{t-1} = nilai *output* dari waktu ke $t-1$
   x_t = nilai *input* pada waktu ke t
   b_f = bias pada *forget gate*
   i_t = nilai dari *input gate*
   W_i = bobot untuk nilai *input* pada waktu ke t
   b_i = bias pada *input gate*

C_t = nilai *memory cell state*
$\tilde{C}_t$ = nilai kandidat *cell state*
C_{t-1} = nilai *memory cell state* pada *cell* sebelum
$tanh$ = fungsi hiperbolik tangent
W_c = bobot untuk nilai *input cell* ke c
b_c = bias pada *cell* ke c
f_i = nilai *forget gate*
o_t = nilai dari *output gate*
b_o = bias pada *output gate*

2. Stacked LSTM adalah arsitektur yang terdiri dari beberapa lapisan (layer) LSTM yang ditumpuk secara vertikal, di mana output dari satu lapisan menjadi *input* bagi lapisan berikutnya. Tujuannya adalah untuk membentuk representasi fitur yang lebih abstrak dan kompleks, mirip seperti pada *deep learning* pada umumnya[11]. Berikut merupakan gambarnya:

$$h_t^{(2)} = LSTM^{(2)}\left(h_t^{(1)}, h_{t-1}^{(2)}, C_{t-1}^{(2)}\right) \quad (2.\ 20)\ [11]$$

Di mana
$h_t^{(2)}$ = output dari LSTM layer ke-1
$h_t^{(2)}$ = hidden state dari LSTM layer ke-2

3. Bi-LSTM adalah arsitektur LSTM yang memproses *input squens* dalam dua arah. Yaitu dari masa lalu ke masa kini dan dari masa depan ke masa kini, dari kedua data yang berjalan secara paralel tersebut, kemudian hasilnya digabungkan untuk menghasilkan hasil akhir [15]. Berikut merupakan gambarnya:
Maju:

$$\overrightarrow{h_t} = LSTM_f\left(x_t, \vec{h}_{t-1}, \vec{C}_{t-1}\right) \quad (2.\ 21)\ [43]$$

Mundur:

$$\overleftarrow{h_t} = LSTM_f\left(x_t, \overleftarrow{h}_{t-1}, \overleftarrow{C}_{t-1}\right) \quad (2.\ 22)\ [43]$$

Hasil akhir:

$$h_t = \left[\vec{h}_t; \overleftarrow{h}_t\right] \quad (2.\ 23)\ [43]$$

Di mana:

$\overrightarrow{h_t}$ = *output* dari arah maju

$\overleftarrow{h_t}$ = *output* dari arah mundur

$[\cdot;\cdot]$ = penggabungan

2.3.3 *Activation functions*

Activation merupakan komponen penting dalam melakukan pemodelan di mana dengan adanya *activation*, jaringan saraf tiruan dapat mempelajari hubungan yang kompleks pada suatu data dan memperkenalkan sifat tidak linier ke dalam suatu model [16]. Dengan adanya *activation* dapat memaksimalkan hasil pemodelan pada algoritma LSTM. Berikut merupakan beberapa fungsi *activation* yang digunakan, yaitu Linear, ReLU, Tanh, dan Sigmoid. Berikut merupakan rumus yang digunakan untuk melakukan *optimizer techniques*, yaitu:

Linear merupakan *activation* paling sederhana, di mana hasil *output* yang diberikan yaitu proporsional terhadap *input* nya. Penggunaan linear cocok digunakan pada *layer output regresi*, namun penggunaan linear tidak cocok pada *hidden layer*. Pada (2.24) merupakan rumus dari penggunaan linear.

$$f(x_i) = kx_i$$

(2. 24) [23]

ReLU atau Rectified Linear Unit merupakan fungsi *activation* yang paling sering digunakan karena kesederhanaan dan efisiensinya dalam menyelesaikan masalah *vanishing gradient* [44]. Penggunaan ReLU sanggatlah cepat dalam melakukan proses pelatihan model sehingga cocok untuk digunakan pada DNN dan CNN. Pada (2.25) merupakan rumus dari penggunaan ReLU.

$$ReLU(x) = max(x, 0)$$

(2. 25) [23]

Tanh merupakan fungsi *activation* tidak liner yang sering digunakan pada jaringan saraf tiruan, terlebih digunakan pada arsitektur yang perlu sebuah nilai output yang memili nilai 0 di sekitarnya [44]. Pada (2.26) merupakan rumus dari penggunaan Tanh.

$$\text{Tanh}\ (x) = \frac{1-e^{-x}}{1+e^{-x}}$$

(2. 26) [23]

Sigmoid merupakah fungsi dari *activation* yang paling sering digunakan untuk jaringan saraf awal dan pada layer *output* yang memiliki tugas untuk klasifikasi kode biner[44]. Fungsi dari penggunaan sigmoid tersebut tercermin pada rumus (2.27) dengan mengubah nilai *imput* ke dalam rentang 1 dan 0.

$$f(x) = \frac{1}{1 + e^{-x}}$$

(2. 27) [23]

2.3.4 *Optimizer techniques*

Dalam melakukan analisis diperlukan proses model *deep learning*, di mana dibutuhkan sebuah algoritma optimasi untuk mengurangi *loss function* dengan cara memperbaiki jaringan secara bertahap berdasarkan alurnya[45]. Pada penelitian ini menggunakan enam *optimizer*, yaitu Adam, AdaGrad atau adaptive gradients, Nadam, RMSProp atau root mean square propagation, AdaDelta, dan SGD atau stochastic gradient descent[23]. Berikut merupakan penjelasan dari keenam *optimizer*, yaitu:

Adam, merupakan algoritma optimasi yang dikembangkan dengan cara menggabungkan dari dua pendekatan yaitu RMSProp dengan Momentum[46]. Dalam prosesnya Adam menyimpan rata-rata dari nilai gradien pertama dan gradien kedua sehingga hasil yang ada dapat menyesuaikan *learning rate* lebih adaptif pada ketiga parameter. Sehingga penggunaan Adam cocok untuk data skala besar. Pada (2.28) merupakan rumus dari penggunaan Adam.

$$\theta_{t+1} = \theta_t - \frac{n}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t$$

(2. 28) [23]

AdaGrad (Adaptive Gradients), merupakan algoritma *optimizer* yang dibuat untuk menyesuaikan *learning rate* secara otomatis selama periode pelatihan. Adegrad berfungsi untuk menghitung *learning rate* yang lebih spesifik kepada setiap parameter berdasarkan frekuensinya[47]. Pada (2.29) merupakan rumus dari penggunaan AdaGrad.

$$w_t = w_{t-1} - \lambda_t^1 \frac{\partial \nabla}{\partial w_{t-1}}$$

(2. 29) [23]

Nadam, merupakan *optimizer* kembangan dari Adam di mana menghubungkan dengan konsep *Nesterov Momentum,* dengan

adanya penggabungan tersebut dapat mempercepat konversi dengan melakukan pertimbangan masa depan pada saat menghitung gradien [17]. Pada (2.30) merupakan rumus dari penggunaan Adam.

$$W_t^i = W_{t-1}^i - \frac{\eta}{\sqrt{v_t + \varepsilon}} \cdot \widetilde{m}_t$$

(2. 30) [23]

RMSProp (*Root Mean Square Propagation*), merupakan *optimizer* yang adaptif, penggunaan RMSProp dikembangkan untuk mengatasi kelemahan dari Adagrad yang dalam melakukan pengujiannya *learning rate* yang hasilnya terus mengecil selama pengulangan. RMSProp bekerja dengan cara membagi *learning rate* dengan nilai rata-rata kuadrat dari gradien sebelumnya, sehingga dapat menyelesaikan masalah pada *vanishing gradient* [48]. Pada (2.31) merupakan rumus dari penggunaan RMSProp.

$$G = \nabla_w C(w_t)$$

(2. 31) [23]

AdaDelta, merupakan pengembangan dari Adagrad, dengan menggunakan mekanisme baru yang membatasi pertumbuhan kumulatif dari pembobotan gradien. AdaDelta sendiri telah menggunakan rasio dari rata-rata gradien terhadap nilai rata-rata dari perubahan suatu parameter[47]. Pada (2.32) merupakan rumus dari penggunaan Adam.

$$\acute{g}_t = \frac{\sqrt{u_{t-1}} + \varepsilon}{\sqrt{s_t}} \acute{g}_t$$

(2. 32) [23]

SGD (*Stochastic Gradient Descent*), adalah *optimizer* yang berguna untuk menghitung gradien dari fungsi yang kurang terhadap setiap sampel yang dipilih secara acak dan melakukan pembaruan *value*. SGD termasuk ke dalam *optimizer* yang lambat dan sering terjebak pada *local minima* [48]. *Local minima* adalah titik di mana fungsinya lebih kecil atau sama di sekitar titiknya. Pada (2.33) merupakan rumus dari penggunaan SGD.

$$W_{i+1} = W_i - \eta \frac{\partial L}{\partial W_i}$$

(2. 33) [23]

2.4 Alat Penelitian

2.4.1 Python

Python adalah bahasa yang paling populer pada saat ini karena memiliki fokus pada kemudahan penggunaan dan pembacaan kode yang mudah dipahami. *Python* sering digunakan pada pengembangan *website*, kecerdasan buatan, analisis data, dan masih banyak lainnya. *Python* sering digunakan dalam melakukan analisis data karena sintaksisnya yang sederhana dan *library* yang disediakan banyak seperti *NumPy, pandas,* dan *Matplotlib*[49]. *Python* dan *Jupyter Notebook* sering digunakan untuk melakukan mengelola, analisis, dan visualisasi data.

2.4.2 *Jupiter notebook*

Jupyter Notebook adalah aplikasi yang digunakan untuk mengembangkan interaktif *open-source* yang sangat populer untuk pemrograman dan analisis data. Aplikasi ini mendukung berbagai macam bahasa pemrograman seperti R, *Python*, dan lainnya. Dengan menggunakan *Jupyter notebook* memungkinkan untuk pengguna dalam merancang dan berbagi dokumen yang berisi kode, teks naratif, hasil eksekusi kode, dan visualisasi dalam satu tempat [50]. *Jupyter notebook* sering digunakan sebagai alat analisis dalam berbagai bidang seperti di dunia pemrograman, ilmu data, meta data, dan penelitian ilmiah.