# BAB II

# LANDASAN TEORI

## 2.1 Penelitian Terdahulu

Penelitian ini mengacu pada beberapa studi terdahulu yang berkaitan dengan prediksi saham berbasis *deep learning*. Tabel 2.1 menyajikan rangkuman penelitian-penelitian terkait yang menjadi dasar acuan dalam penelitian ini.

Tabel 2. 1 Penelitian Terdahulu

| No | Nama Jurnal | Judul Jurnal, Penulis | Metode | Hasil |
|---|---|---|---|---|
| 1 | Procedia Computer Science 252 (2025) 893–904 [6] | "*Stock market time series forecasting using comparative machine learning algorithms*",<br><br>Gupta S., Nachappa S., Paramanandham N. | LSTM, GARCH, XGBoost, AdaBoost, Random Forest, SVM | LSTM adalah model paling akurat dengan akurasi 93.54%, diikuti oleh GARCH dengan akurasi 91.28%. LSTM unggul dalam menangkap dependensi jangka panjang dan pola kompleks dalam data time series, sementara GARCH efektif dalam memodelkan volatilitas. Model lain seperti XGBoost, AdaBoost, Random Forest, dan SVM menunjukkan akurasi yang lebih rendah, dengan SVM menjadi yang paling tidak akurat. |
| 2 | Procedia Computer Science, Vol. 234, 2024, pp. 204-212 [8] | "*Long Short-Term Memory and Gated Recurrent Unit for Stock Price Prediction*",<br><br>Rahmadeyan A., Mustakim | LSTM dan GRU | Studi ini membandingkan kinerja LSTM dan GRU dalam memprediksi harga saham Bank Rakyat Indonesia (BRI). Hasil menunjukkan bahwa GRU lebih unggul dengan nilai MSE 4958.9168, RMSE 70.4195, dan MAPE 1.1699%, dibandingkan LSTM yang memiliki nilai MSE 4997.4083, RMSE 70.6923, dan |

| No | Nama Jurnal | Judul Jurnal, Penulis | Metode | Hasil |
|---|---|---|---|---|
| | | | | MAPE 1.1834%. Optimizer RMSprop terbukti paling efektif dalam meningkatkan kinerja model. GRU diprediksi akan mengalami penurunan harga saham pada Januari 2023. Kesimpulannya, GRU lebih efektif dalam memprediksi harga saham BRI dibandingkan LSTM, dan hasil prediksi ini dapat digunakan untuk evaluasi strategi bisnis perusahaan. |
| 3 | International Journal of Accounting, Management and Economics Research (IJAMER), Vol. 2, No.1, 2024, pp. 111-125 [11] | *"Optimizing Banking Stock Price Prediction: Deep Learning Based Approach"*,<br><br>Setiawan D., Wira T., Suryawijaya E., Anugrah M., Pearl J., Chasanah A. | LSTM dan GRU | GRU menunjukkan performa lebih baik dibanding LSTM pada 3 dari 4 saham. MSE GRU terendah 0.000037 (BBNI), RMSE 0.0061, dan MAPE 9.19 juta. LSTM memiliki RMSE terendah 0.0104 pada BRIS. |
| 4 | Jurnal Matematika, Statistika dan Komputasi, Vol. 21, No. 1, September 2024, pp. 321-333 [12] | "*Long Short-Term Memory and Gated Recurrent Unit Modeling for Stock Price Forecasting*",<br><br>Khairunisa N., Hendikawati P. | LSTM dan GRU | Studi ini membandingkan kinerja LSTM dan GRU dalam memprediksi harga saham PT Mayora Indah Tbk. Hasil menunjukkan bahwa GRU lebih unggul dengan RMSE 34,4233 dan MAPE 1,27%, dibandingkan LSTM yang memiliki RMSE 35,3775 dan MAPE 1,28%. Arsitektur terbaik GRU menggunakan learning rate 0,001, 50 neuron, dan 150 epoch, sementara LSTM optimal dengan learning |

| No | Nama Jurnal | Judul Jurnal, Penulis | Metode | Hasil |
|---|---|---|---|---|
| | | | | rate 0,001, 100 neuron, dan 150 epoch. Kesimpulannya, GRU lebih efektif dalam prediksi harga saham dibandingkan LSTM. |
| 5 | Applied Computer Science, vol. 19, no. 1, pp. 82–94 [5] | "*Predicting Banking Stock Prices Using RNN, LSTM, and GRU Approach*",<br><br>Satria D. | ARIMA Box-Jenkins, RNN, LSTM, GRU | Studi ini membandingkan ARIMA Box-Jenkins, RNN, LSTM, dan GRU untuk prediksi harga saham empat bank besar di Indonesia. ARIMA tidak cocok karena melanggar asumsi white noise. GRU mencatat performa terbaik dengan RMSE terendah (0.140 untuk BCA) dan MAPE terendah (26.36% untuk BNI), serta lebih sederhana daripada LSTM. Kesimpulannya, GRU adalah model terbaik untuk prediksi harga saham bank di Indonesia. |
| 6 | Akdeniz İİBF Dergisi 2024, 24 (1) 1-13 [13] | "*Comparative Analysis of Deep Learning Models for Silver Price Prediction: CNN, LSTM, GRU and Hybrid Approach*",<br><br>Emre GÜR Y. | CNN, LSTM, GRU, Hybrid CNN-LSTM-GRU | Studi ini membandingkan model CNN, LSTM, GRU, dan hybrid CNN-LSTM-GRU untuk prediksi harga perak. Model hybrid CNN-LSTM-GRU paling unggul dengan RMSE 0.1089, MAE 1.4789%, dan $R^2$ 0.9913. Model hybrid ini lebih akurat dalam memprediksi harga perak dibandingkan model tunggal. |
| 7 | Engineering, Technology & Applied | *"Harnessing Deep Learning and Technical* | LSTM, GRU, RNN | LSTM memberikan akurasi terbaik ($R^2$: 0.96, MAE: rendah), disusul GRU dan RNN. LSTM |

| No | Nama Jurnal | Judul Jurnal, Penulis | Metode | Hasil |
|---|---|---|---|---|
| | Science Research Vol. 15, No. 1, 2025, 20348-20357 [14] | *Indicators for Enhanced Stock Predictions of Blue-Chip Stocks on the Indonesia Stock Exchange (IDX)",*<br><br>Dwiandiyanta B., Hartanto R., Ferdiana R. | | mampu mengikuti pola harga saham dengan lebih presisi. |
| 8 | KILAT (Jurnal Ilmu dan Teknologi), Vol. 12, No. 1, April 2023, pp. 64-78 [15] | "*Stock Price Prediction Using Machine Learning With Long Short Therm Memory Method (LSTM)*",<br><br>Juan Sydti, Syariful Alam, Irsan Jaelani | LSTM | Studi ini membandingkan kinerja LSTM dalam memprediksi harga saham PT Aneka Tambang Tbk (ANTM) dan PT Merdeka Copper Gold Tbk (MDKA). Hasil menunjukkan bahwa LSTM menghasilkan nilai MAE 48.25 dan MAPE 2.17% untuk saham ANTM, serta MAE 117.83 dan MAPE 3.08% untuk saham MDKA. Model LSTM optimal dengan batch size 16, 50 neuron, dan 50 epoch. Kesimpulannya, LSTM efektif dalam memprediksi harga saham, dengan performa lebih baik pada saham ANTM dibandingkan MDKA. |
| 9 | Jurnal JIFORTY, Vol. 1, No. 1, Juni 2020, pp. 1-8 [16] | "*Prediksi Data Time Series Saham Bank BRI Dengan Mesin Belajar LSTM (Long Short-Term Memory)*",<br><br>Adhitio Satyo Bayangkari Karno | LSTM | Studi ini menggunakan metode LSTM untuk memprediksi data time series saham Bank BRI (BBRI). Hasil menunjukkan bahwa model LSTM dengan 4 hidden layer (masing-masing 50 node) dan dropout 0.2 berhasil mencapai nilai RMSE |

| No | Nama Jurnal | Judul Jurnal, Penulis | Metode | Hasil |
|---|---|---|---|---|
| | | | | 227.47, menunjukkan akurasi yang baik. Visualisasi grafik prediksi juga menunjukkan kemiripan dengan data aktual. Kesimpulannya, LSTM efektif untuk prediksi saham, meskipun pemilihan jumlah epoch perlu hati-hati untuk optimasi waktu dan akurasi. Penelitian lanjutan disarankan untuk membandingkan LSTM dengan metode lain seperti RNN atau GRU. |
| 10 | UJMC (Unisda Journal of Mathematics and Computer Science), vol. 6, no. 01, pp. 9-18, Jun. 2020 [17] | "*Implementasi Long Short-Term Memory Pada Harga Saham Perusahaan Perkebunan Di Indonesia*"<br><br>Rahmadi Yotenka, Fazano Fikri El Huda | LSTM | Studi ini menggunakan metode LSTM untuk memprediksi harga saham perusahaan perkebunan di Indonesia, yaitu SSMS, LSIP, dan SIMP. Hasil menunjukkan bahwa model LSTM terbaik untuk SSMS menggunakan optimizer RMSProp dengan 70 hidden neuron menghasilkan RMSE 21.328, untuk LSIP menggunakan optimizer Adam dengan 80 hidden neuron menghasilkan RMSE 33.097, dan untuk SIMP menggunakan optimizer Adamax dengan 100 hidden neuron menghasilkan RMSE 8.337. Prediksi menunjukkan kenaikan harga saham SSMS, LSIP, dan SIMP pada 25 Juli 2019. Kesimpulannya, LSTM efektif dalam |

| No | Nama Jurnal | Judul Jurnal, Penulis | Metode | Hasil |
|---|---|---|---|---|
| | | | | memprediksi harga saham perkebunan dengan akurasi yang baik. |
| 11 | Engineering, Technology & Applied Science Research, Vol. 15, No. 1, 2025 [18] | "*Optimization of Stock Predictions on Indonesia Stock Exchange: A New Hybrid Deep Learning Method*",<br><br>Dwiandiyanta B., Hartanto R., Ferdiana R. | Hybrid GRU-RNN, LSTM, GRU, RNN | Model *hybrid* menunjukkan kinerja yang jauh lebih unggul dibandingkan model tunggal. Untuk saham BBNI, model ini mencapai $R^2$ 0.9886, membuktikan efektivitas penggabungan data historis dan teknikal secara terpisah. |

Berdasarkan Tabel 2.1, telah dilakukan analisis terhadap sepuluh artikel jurnal yang membahas prediksi harga saham menggunakan berbagai pendekatan algoritma *machine learning* dan *deep learning*, khususnya model *time series*. Mayoritas penelitian mengangkat metode LSTM, GRU, dan RNN, serta membandingkan kinerjanya dengan metode lain seperti ARIMA, CNN, dan gabungan model *hybrid*.

Beberapa artikel seperti pada [8], [11], [12], [5], dan [14] secara eksplisit membandingkan performa model LSTM, GRU, dan RNN dalam konteks prediksi harga saham di Indonesia. Evaluasi model pada penelitian-penelitian ini mengacu pada metrik umum seperti RMSE, MAE, dan MAPE. Hasil menunjukkan bahwa meskipun LSTM banyak digunakan, GRU sering kali menunjukkan performa lebih unggul, terutama dalam hal kecepatan pelatihan dan stabilitas error.

Pada artikel [8], [11], [12], dan [5], GRU terbukti memiliki kinerja terbaik dibandingkan LSTM dan RNN dalam memprediksi harga saham, dengan nilai error yang lebih rendah dan kemampuan yang lebih baik dalam menangkap pola data. Penelitian [13] juga menyertakan model *hybrid* CNN-LSTM-GRU dan menunjukkan bahwa gabungan model ini memberikan akurasi prediksi tertinggi dalam studi harga komoditas perak. Sejalan dengan itu, penelitian [18] memperkenalkan model *hybrid* GRU-RNN yang secara spesifik memisahkan pemrosesan data historis dan indikator teknikal, terbukti mencapai akurasi tinggi

pada saham-saham di Bursa Efek Indonesia. Di sisi lain, penelitian [6] dan [14] menunjukkan keunggulan LSTM dibandingkan model lainnya.

Sementara itu, jurnal [15], [16], dan [17] secara khusus fokus pada penggunaan LSTM saja, menunjukkan bahwa meskipun bukan selalu yang paling unggul, LSTM tetap memiliki performa yang sangat baik untuk prediksi harga saham, terutama bila dioptimasi dengan parameter seperti jumlah *neuron*, *epoch*, dan jenis *optimizer*.

Perbedaan utama penelitian ini dibandingkan dengan penelitian-penelitian sebelumnya terletak pada objek dan ruang lingkup studi. Penelitian ini berfokus pada tiga saham perbankan *blue-chip* Indonesia yaitu BBCA, BBRI, dan BMRI, serta melakukan perbandingan menyeluruh antara tiga model *deep learning* yaitu LSTM, GRU, dan RNN. Selain itu, penelitian ini juga mencakup proses *tuning hyperparameter* menggunakan Optuna untuk masing-masing model secara eksplisit sebagai bagian dari proses optimalisasi performa prediksi.

Penelitian ini juga menghasilkan prediksi harga saham untuk 30 hari ke depan (*forecasting*) sebagai bentuk penerapan nyata model yang dikembangkan. Hasil prediksi kemudian dibandingkan dengan data aktual guna menilai seberapa baik model mampu merepresentasikan dinamika pasar yang kompleks. Pendekatan ini memberikan *insight* tambahan yang aplikatif bagi investor dan pengambil keputusan di sektor keuangan.

Dengan demikian, kontribusi utama dari penelitian ini adalah memberikan pemahaman mendalam terhadap efektivitas ketiga model tersebut dalam konteks saham perbankan Indonesia, serta menawarkan insight baru mengenai performa aktual model dalam skenario dunia nyata berdasarkan metrik evaluasi yang relevan.

## 2.2 Teori Penelitian

### 2.2.1 Saham dan Pasar Modal

Pasar modal merupakan sistem keuangan terorganisir yang memfasilitasi pertemuan antara pihak yang membutuhkan dana (emiten) dan pihak yang memiliki kelebihan dana (investor). Di dalam pasar modal, berbagai instrumen keuangan

diperjualbelikan, salah satunya adalah saham. Saham adalah surat berharga yang mencerminkan kepemilikan atas suatu perusahaan. Dengan membeli saham, investor mendapatkan hak atas sebagian kepemilikan dan keuntungan perusahaan, serta potensi capital gain dari kenaikan harga saham di pasar [19].

Harga saham bersifat dinamis dan dipengaruhi oleh berbagai faktor, baik internal maupun eksternal. Faktor internal meliputi kinerja keuangan perusahaan, aksi korporasi, serta laporan tahunan. Sementara itu, faktor eksternal meliputi kondisi makroekonomi, kebijakan moneter, tingkat suku bunga, inflasi, serta peristiwa politik [20]. Karena kompleksitas faktor-faktor tersebut, harga saham bersifat fluktuatif dan sulit diprediksi secara deterministik.

Dalam konteks Indonesia, saham perbankan seperti PT Bank Central Asia Tbk, PT Bank Rakyat Indonesia Tbk, dan PT Bank Mandiri Tbk termasuk dalam kategori saham *blue-chip* yang memiliki kapitalisasi pasar besar, tingkat likuiditas tinggi, dan reputasi fundamental perusahaan yang kuat [21]. Ketiga saham tersebut merupakan konstituen utama dalam Indeks LQ45 dan sering dijadikan acuan dalam portofolio investasi jangka panjang.

### 2.2.2 *Machine Learning* dan *Deep Learning*

*Machine Learning* (ML) merupakan cabang dari kecerdasan buatan (*Artificial Intelligence*) yang memungkinkan komputer belajar dari data tanpa harus diprogram secara eksplisit. ML memanfaatkan algoritma statistik untuk menemukan pola tersembunyi dari data historis, dan menggunakannya untuk membuat prediksi atau keputusan terhadap data baru [22]. Dalam konteks prediksi harga saham, ML memungkinkan model mengenali pola fluktuasi harga berdasarkan data historis.

*Machine learning* terbagi menjadi tiga kategori utama, yaitu *supervised learning*, *unsupervised learning*, dan *reinforcement learning*. *Supervised learning* bekerja dengan melatih model menggunakan data berlabel, sehingga model dapat mempelajari hubungan antara input dan output untuk melakukan prediksi. Sebaliknya, *unsupervised learning* digunakan pada data yang tidak memiliki label, dengan tujuan menemukan pola tersembunyi atau struktur dari data tersebut.

Sementara itu, *reinforcement learning* memungkinkan sistem belajar membuat keputusan terbaik melalui proses coba-coba, dengan memanfaatkan umpan balik dari tindakan sebelumnya untuk meningkatkan performa di masa depan [23].

Salah satu pengembangan dari *machine learning* adalah *deep learning*, yaitu teknik pembelajaran mesin yang menggunakan jaringan saraf tiruan dengan banyak lapisan (*deep neural networks*). *Deep learning* unggul dalam menangani data tidak terstruktur dan besar seperti teks dan time series. Arsitektur *deep learning* seperti RNN, LSTM, dan GRU banyak digunakan dalam prediksi saham karena kemampuannya dalam mengingat dan memproses data sekuensial [24].

### *2.2.3 Time Series Forecasting*

*Time series forecasting* adalah teknik untuk memprediksi nilai masa depan berdasarkan pola yang terdapat dalam data historis yang tersusun berurutan menurut waktu. Metode ini melibatkan analisis data historis untuk mengidentifikasi pola-pola seperti tren, variasi musiman, dan siklus. Ciri khas data deret waktu adalah adanya ketergantungan temporal antar data, sehingga informasi dari masa lalu dapat membantu memperkirakan kondisi di masa depan [25].

Secara teori, sebuah data *time series* dapat diuraikan menjadi beberapa komponen utama yang membentuk polanya. Komponen pertama adalah tren (*trend*), yang menunjukkan pergerakan atau arah data dalam jangka panjang, apakah cenderung meningkat, menurun, atau konstan [26]. Komponen kedua adalah musiman (*seasonality*), yaitu fluktuasi periodik yang terjadi secara teratur dalam interval waktu tertentu (misalnya, harian, mingguan, atau tahunan) [26]. Selanjutnya, terdapat komponen siklus (*cycle*), yang merupakan pola naik-turun dengan durasi yang tidak tetap dan biasanya berlangsung lebih dari satu tahun, sering kali terkait dengan kondisi ekonomi [26]. Terakhir, terdapat komponen acak atau tidak teratur (*irregular/noise*), yang merupakan fluktuasi yang tidak terduga dan tidak mengikuti pola tertentu [27].

Dalam konteks pasar saham, data deret waktu biasanya mencakup harga pembukaan, harga penutupan, harga tertinggi, harga terendah, dan volume

transaksi. Variabel-variabel ini sering digunakan untuk memahami perilaku pasar dan membangun model prediksi.

Pendekatan klasik yang umum digunakan untuk peramalan deret waktu adalah model statistik seperti ARIMA. Meskipun efektif untuk data yang bersifat linear dan stasioner, model ini memiliki keterbatasan dalam menangani pola non-linear dan kompleksitas pasar saham [28].

Oleh karena itu, metode berbasis *deep learning* seperti RNN, LSTM, dan GRU semakin banyak digunakan. Model-model ini mampu menangkap pola non-linear dan ketergantungan jangka panjang dalam data *time series*, sehingga memberikan prediksi yang lebih akurat pada harga saham [28].

### 2.2.4 *Overfitting* dan *Underfitting*

Dalam proses pelatihan model *machine learning*, tujuan utamanya adalah untuk menciptakan model yang tidak hanya akurat pada data yang telah dilatihkan, tetapi juga mampu melakukan generalisasi dengan baik pada data baru yang belum pernah dilihat sebelumnya. Dua masalah umum yang dapat menghalangi tercapainya tujuan ini adalah *overfitting* dan *underfitting* [29].

*Overfitting* adalah kondisi di mana model menjadi terlalu kompleks dan menghafal data latih, termasuk detail minor dan *noise* yang tidak relevan [29]. Akibatnya, model menunjukkan performa yang sangat baik pada data latih, namun kinerjanya akan sangat buruk ketika dihadapkan pada data baru. Model yang *overfit* gagal menangkap pola umum yang mendasari data, sehingga tidak dapat diandalkan untuk membuat prediksi di dunia nyata.

*Underfitting* adalah kondisi sebaliknya, di mana model terlalu sederhana untuk dapat menangkap pola atau struktur yang ada di dalam data, menghasilkan performa buruk baik pada data latih maupun data baru [29]. Ini biasanya terjadi ketika kapasitas model tidak cukup untuk kompleksitas data yang dihadapi.

Salah satu cara paling efektif untuk mendiagnosis *overfitting* dan *underfitting* adalah melalui analisis visual kurva pembelajaran (*learning curves*), khususnya kurva *loss* (*loss curve*) yang diamati selama proses pelatihan. Kurva ini memetakan

nilai *error* (kerugian) model pada setiap *epoch*, yang terbagi menjadi dua komponen utama: *Training Loss*, yang merupakan *error* model pada data latih, dan *Validation Loss*, yang merupakan *error* model pada data validasi yang tidak digunakan dalam pelatihan [29].

Interaksi antara kedua kurva inilah yang dapat mengindikasikan kondisi model. Sebuah model dapat dikatakan telah mencapai kondisi yang baik (*good fit*) apabila kedua kurva sama-sama menurun dan konvergen pada nilai *error* yang rendah dan stabil. Sebaliknya, *overfitting* terdeteksi ketika kurva *training loss* terus menurun, namun kurva *validation loss* berhenti menurun dan mulai bergerak datar atau bahkan naik kembali, yang ditandai dengan melebarnya celah di antara kedua kurva. Sementara itu, kondisi *underfitting* diidentifikasi jika kedua kurva gagal menunjukkan penurunan yang signifikan dan sama-sama stagnan pada level *error* yang tinggi [29].

### 2.2.5 *Hyperparameter*

Dalam pengembangan model *machine learning* maupun *deep learning*, *hyperparameter* merupakan elemen penting yang harus ditentukan sebelum proses pelatihan model dilakukan. *Hyperparameter* berbeda dengan parameter internal model seperti bobot dan bias yang diperoleh melalui proses pembelajaran. Sebaliknya, *hyperparameter* merupakan nilai-nilai yang ditentukan secara eksplisit oleh peneliti atau praktisi untuk mengatur bagaimana model akan belajar dari data [30]. Beberapa contoh *hyperparameter* yang umum digunakan dalam neural network antara lain adalah *learning rate*, jumlah unit pada *layer* tersembunyi, jumlah *epoch*, *batch size*, *dropout rate*, dan jumlah *layer* dalam arsitektur jaringan [10].

*Learning rate* adalah *hyperparameter* krusial yang mengontrol ukuran langkah penyesuaian bobot model selama proses optimasi berbasis gradien, seperti Adam. Nilai *learning rate* yang terlalu tinggi dapat menyebabkan model melompati solusi optimal dan gagal konvergen, sementara nilai yang terlalu rendah dapat mengakibatkan konvergensi yang sangat lambat atau model terjebak dalam lokal

minimum yang suboptimal [30]. Oleh karena itu, pemilihan *learning rate* yang tepat sangat penting untuk efisiensi pelatihan dan pencapaian kinerja model yang baik.

Sementara itu, jumlah unit (*neuron*) dalam lapisan tersembunyi (*hidden layer*) sebuah jaringan saraf secara langsung menentukan kapasitas representasi dan kompleksitas model dalam mempelajari pola data. Jumlah unit yang tidak memadai dapat menyebabkan *underfitting*, di mana model gagal menangkap pola esensial dalam data. Sebaliknya, jumlah unit yang terlalu banyak dapat mengakibatkan *overfitting* model mempelajari *noise* alih-alih pola umum sehingga kinerjanya buruk pada data baru serta meningkatkan biaya komputasi secara signifikan [10]. Penentuan jumlah unit yang optimal seringkali bergantung pada kompleksitas data dan merupakan bagian penting dari proses *tuning*.

*Dropout rate* mengontrol teknik regularisasi *dropout* yang efektif untuk mengurangi risiko *overfitting* dalam jaringan saraf. Mekanisme kerjanya adalah dengan menonaktifkan sebagian neuron secara acak pada lapisan tertentu selama setiap iterasi pelatihan. Hal ini memaksa jaringan untuk mempelajari fitur yang lebih robust dan tidak terlalu bergantung pada neuron individual, sehingga meningkatkan kemampuan generalisasinya terhadap data yang belum pernah dilihat sebelumnya [10]. Jumlah *layer* juga menentukan kedalaman arsitektur dan kapasitas model. *Batch size* memengaruhi stabilitas dan kecepatan pelatihan, sementara jumlah *epoch* menentukan berapa kali seluruh dataset digunakan dalam proses pelatihan [10].

Pemilihan kombinasi *hyperparameter* yang optimal merupakan aspek krusial dalam meningkatkan performa model pembelajaran mesin. Namun, proses pencarian *hyperparameter* terbaik sering kali menjadi tantangan tersendiri, terutama ketika ruang pencariannya luas dan terdiri dari banyak parameter dengan berbagai kemungkinan nilai. Metode pencarian seperti *manual tuning*, *grid search*, dan *random search* memang sering digunakan, tetapi masing-masing memiliki keterbatasan, terutama dalam hal efisiensi waktu dan komputasi. Seiring berkembangnya kebutuhan akan pencarian yang lebih sistematis dan efisien, pendekatan otomatis seperti *Bayesian Optimization* mulai banyak digunakan [31].

Pendekatan otomatis seperti *Optuna* juga mulai banyak digunakan karena efisiensi dan kemampuannya dalam menemukan kombinasi optimal secara sistematis [32].

### 2.2.6 *Optuna*

*Optuna* merupakan salah satu *automated hyperparameter optimization framework* berbasis Python yang bersifat *open-source* dan dirancang untuk efisien serta fleksibel dalam mencari kombinasi *hyperparameter* terbaik dalam model *machine learning* [33]. *Framework* ini mendukung proses optimasi berbasis *define-by-run*, yaitu pendekatan dinamis di mana ruang pencarian parameter ditentukan secara fleksibel pada saat runtime [33]. Hal ini memberikan keleluasaan dalam merancang eksperimen pencarian parameter yang kompleks maupun yang bersifat kondisional.

Dalam konteks *deep learning*, pemilihan *hyperparameter* seperti jumlah *unit* pada *layer*, tingkat *dropout*, dan nilai *learning rate* memiliki pengaruh yang sangat besar terhadap performa model [10]. Melakukan pencarian *hyperparameter* secara manual atau menggunakan metode *grid search* tidak selalu efisien, terutama ketika ruang pencarian sangat luas. Di sinilah *Optuna* memainkan peran penting dengan mengadopsi pendekatan *Bayesian optimization* dan algoritma *Tree-structured Parzen Estimator (TPE)* yang memungkinkan proses pencarian yang lebih adaptif dan efisien dibandingkan pendekatan ekshaustif [32].

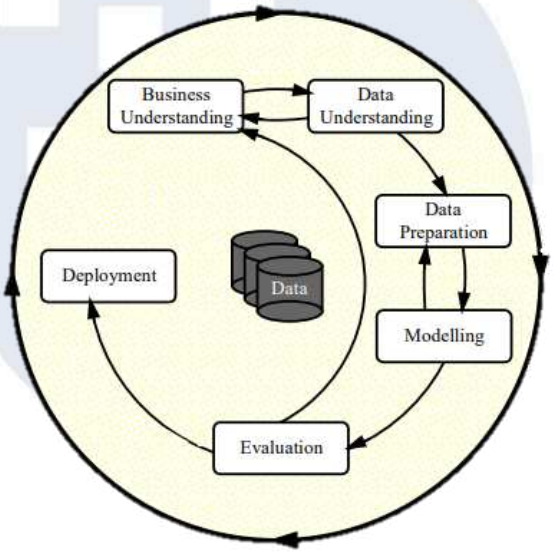
*Optuna* bekerja melalui konsep *study* dan *trial*. Sebuah *study* merepresentasikan seluruh proses optimasi, sedangkan *trial* adalah satu percobaan evaluasi kombinasi *hyperparameter* [32]. Dalam setiap *trial*, *Optuna* akan mencoba satu set *hyperparameter*, melatih model, dan mengevaluasi performanya menggunakan metrik tertentu (misalnya *val_loss*). Berdasarkan hasil *trial* sebelumnya, *Optuna* akan memperbarui strategi pencarian untuk *trial* selanjutnya guna mempercepat konvergensi ke *parameter* terbaik [32].

## 2.3 *Framework*, Algoritma, dan Metode Evaluasi

### 2.3.1 *Framework* CRISP-DM

CRISP-DM *(Cross-Industry Standard Process for Data Mining*) adalah sebuah model proses komprehensif yang dirancang untuk pelaksanaan

proyek *data mining* [34]. Metodologi ini bersifat fleksibel dan dapat diterapkan di berbagai sektor industri maupun penelitian akademik, termasuk dalam konteks analisis dan prediksi harga saham [35]. Relevansi dan keandalan CRISP-DM dalam lingkup analisis finansial telah divalidasi secara luas, di mana metodologi ini juga menjadi landasan bagi berbagai penelitian serupa yang berfokus pada peramalan pasar saham dengan *machine learning* [36], [37], [38]. Gambar 2.1 menunjukkan enam tahap dalam CRISP-DM, yaitu:




Gambar 2. 1 *Framework* CRISP-DM
Sumber: [34]

a. *Business Understanding*

Tahap ini bertujuan untuk memahami tujuan proyek dari sisi bisnis atau domain aplikasi. Fokus utamanya adalah merumuskan permasalahan dan tujuan secara jelas agar analisis data yang dilakukan selaras dengan kebutuhan pengguna akhir [34].

b. *Data Understanding*

Pada tahap ini dilakukan pengumpulan dan eksplorasi data awal untuk memahami karakteristik dasar data. Aktivitasnya meliputi pemeriksaan kualitas data, identifikasi nilai yang hilang atau tidak konsisten, serta pengenalan terhadap pola atau tren awal yang mungkin muncul [34].

c. *Data Preparation*

Tahap ini bertujuan untuk mengubah data mentah menjadi format yang sesuai untuk proses pemodelan. Proses ini mencakup seleksi fitur, pembersihan data dari nilai yang hilang atau *outlier*, serta transformasi seperti normalisasi dan pembentukan urutan data dalam bentuk *time series*. Tahapan ini sangat penting agar model dapat mempelajari pola dengan optimal [34].

d. *Modeling*

Pada tahap ini, algoritma *data mining* atau *machine learning* diterapkan untuk membangun model prediksi atau klasifikasi. Pemilihan algoritma disesuaikan dengan jenis data dan tujuan analisis, serta dilakukan proses *tuning* parameter agar model bekerja secara optimal [34].

e. *Evaluation*

Setelah model dibuat, tahap evaluasi dilakukan untuk mengukur seberapa baik kinerja model dalam menjawab kebutuhan yang telah didefinisikan di tahap pertama [34]. Evaluasi menggunakan metrik statistik contohnya seperti MAE (*Mean Absolute Error*), RMSE (*Root Mean Squared Error*), MAPE (*Mean Absolute Percentage Error*), dan $R^2$ (*R-squared*).

f. *Deployment*

Tahap deployment merupakan proses akhir di mana hasil analisis dan model yang telah dibangun disampaikan dalam bentuk yang dapat digunakan. Dalam konteks penelitian, tahap ini umumnya berupa penyajian prediksi, visualisasi performa model, serta rekomendasi yang mendukung pengambilan keputusan oleh pihak terkait [34].

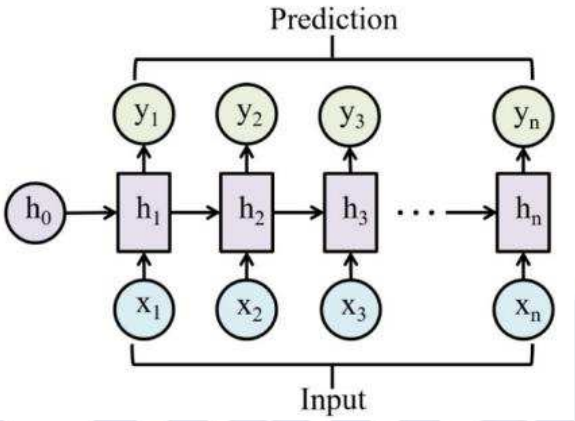
Metodologi CRISP-DM dipilih karena mampu memberikan alur kerja yang terstruktur dan dapat diadaptasi untuk berbagai jenis permasalahan data, termasuk prediksi harga saham. Dalam penelitian ini, kerangka CRISP-DM digunakan sebagai pedoman dalam merancang proses prediksi

harga saham perbankan (BBCA, BBRI, dan BMRI) menggunakan algoritma *deep learning* seperti LSTM, GRU, dan RNN.

### 2.3.2 Algoritma

#### 2.3.2.1 *Recurrent Neural Network* (RNN)

*Recurrent Neural Network* (RNN) adalah jenis jaringan saraf tiruan yang dirancang khusus untuk menangani data berurutan atau deret waktu [39]. Berbeda dengan jaringan saraf *feedforward*, RNN memiliki koneksi umpan balik yang memungkinkan informasi dari langkah sebelumnya digunakan dalam pemrosesan langkah saat ini. Struktur ini memberikan RNN kemampuan untuk mengingat informasi sebelumnya dalam urutan data, menjadikannya sangat efektif dalam menangani tugas-tugas seperti pemrosesan bahasa alami (NLP), pengenalan suara, dan analisis deret waktu [40].




Gambar 2. 2 Arsitektur Dasar RNN
Sumber: [41]

Gambar 2.2 menunjukkan arsitektur dasar dari sebuah RNN, di mana informasi dapat berputar dalam sebuah *loop*. Pada setiap langkah waktu $t$, RNN menerima input $\boldsymbol{x_t}$ dan memperbarui *hidden state* $\boldsymbol{h_t}$ yang menyimpan informasi dari langkah-langkah sebelumnya. Proses ini secara matematis dapat dijelaskan dengan rumus berikut [41]:

$$\boldsymbol{h_t} = \sigma_h(\boldsymbol{W_{xh}x_t} + \boldsymbol{W_{hh}h_{t-1}} + \boldsymbol{b_h}) \quad (2.1)$$

Rumus 2. 1 *Hidden State* RNN

Dalam rumus 2.1, *hidden state* baru ($\boldsymbol{h}_t$) yang merupakan "memori" jaringan, dihitung berdasarkan kombinasi dari input saat ini ($\boldsymbol{x}_t$) dan memori dari langkah sebelumnya ($\boldsymbol{h}_{t-1}$). Proses ini diatur oleh matriks bobot ($\boldsymbol{W}_{xh}$ dan $\boldsymbol{W}_{hh}$) dan bias ($\boldsymbol{b}_h$) yang nilainya dipelajari selama pelatihan, kemudian hasilnya dilewatkan melalui fungsi aktivasi ($\sigma_h$) agar model mampu mempelajari pola yang kompleks.

Setelah *hidden state* diperbarui, model kemudian menghasilkan output untuk langkah waktu tersebut. Output ($\boldsymbol{y}_t$) pada setiap langkah waktu, $t$, dihasilkan oleh rumus berikut [41]:

$$\boldsymbol{y}_t = \sigma_y(\boldsymbol{W}_{hy}\boldsymbol{h}_t + \boldsymbol{b}_y) \tag{2.2}$$

Rumus 2. 2 *Output* RNN

Rumus 2.2 digunakan untuk menghasilkan nilai prediksi ($\boldsymbol{y}_t$) berdasarkan *hidden state* saat ini ($\boldsymbol{h}_t$). Transformasi ini diatur oleh matriks bobot ($\boldsymbol{W}_{hy}$) dan bias ($\boldsymbol{b}_y$), yang hasilnya kemudian dilewatkan melalui fungsi aktivasi output ($\sigma_y$) yang sesuai dengan jenis masalahnya.

Namun, RNN standar menghadapi tantangan signifikan yang dikenal sebagai masalah *vanishing gradient* [42]. Masalah ini terjadi ketika sinyal kesalahan yang dibawa ke belakang selama proses pelatihan menjadi sangat kecil, sehingga mempersulit pembelajaran pada urutan yang panjang. Akibatnya, RNN kesulitan dalam menangkap dependensi jangka panjang dalam data sekuensial [43].

Untuk mengatasi keterbatasan ini, arsitektur jaringan yang lebih kompleks seperti LSTM dan GRU dikembangkan. LSTM memperkenalkan mekanisme gerbang (*gates*) yang mengatur aliran informasi dan memungkinkan jaringan untuk mempertahankan informasi relevan dalam jangka waktu yang lebih panjang. Sementara itu, GRU menyederhanakan struktur LSTM dengan menggabungkan beberapa gerbang, namun tetap efektif dalam menangani masalah *vanishing gradient* [39].

Dengan demikian, RNN dan variannya seperti LSTM dan GRU telah menjadi fondasi penting dalam pengembangan model-model *deep learning* yang mampu memahami dan memproses data berurutan secara efektif.

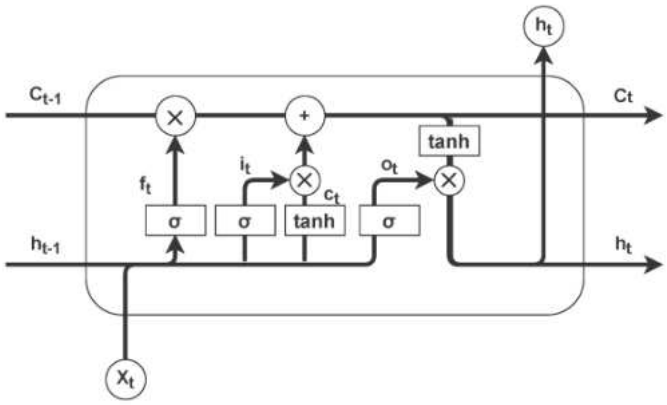
Dalam konteks penelitian ini, *hyperparameter* yang dioptimasi memiliki kaitan erat dengan arsitektur dan proses pelatihan RNN:

- Jumlah Unit: *Hyperparameter* ini secara langsung menentukan dimensionalitas dari *hidden state* ($\boldsymbol{h}_t$) dalam setiap sel RNN. Jumlah unit yang lebih besar mengindikasikan kapasitas model yang lebih tinggi untuk menyimpan dan memproses informasi dari *sequence* data historis, namun juga meningkatkan kompleksitas komputasi dan risiko *overfitting*.
- *Dropout Rate*: Meskipun *dropout* tidak secara inheren menjadi bagian dari unit RNN dasar, lapisan *dropout* ditambahkan setelah lapisan RNN dalam arsitektur model penelitian ini. *Dropout rate* mengatur proporsi neuron pada output lapisan RNN yang dinonaktifkan secara acak selama proses pelatihan. Hal ini bertujuan untuk mencegah *overfitting* dengan mengurangi ketergantungan antar neuron dan mendorong pembelajaran representasi yang lebih robust.
- *Learning Rate*: *Hyperparameter* ini mengontrol besaran penyesuaian bobot model selama proses optimasi (misalnya, menggunakan *optimizer* Adam). *Learning rate* memengaruhi semua bobot yang dapat dipelajari dalam jaringan RNN, termasuk bobot untuk input, bobot *recurrent*, dan bias, sehingga sangat krusial untuk mencapai konvergensi yang stabil dan optimal.

#### 2.3.2.2 *Long Short-Term Memory* (LSTM)

*Long Short-Term Memory* (LSTM) adalah salah satu jenis jaringan saraf tiruan yang termasuk dalam keluarga RNN, yang secara khusus dirancang untuk mengatasi keterbatasan utama dari RNN konvensional, yaitu *vanishing gradient* [44]. Permasalahan ini sering terjadi saat pelatihan jaringan pada data sekuensial yang panjang, di mana gradien yang sangat

kecil menyebabkan hilangnya kemampuan jaringan untuk belajar dari data di masa lalu. LSTM mengatasi masalah ini dengan memperkenalkan struktur yang lebih kompleks, yaitu sel memori (*memory cell*), yang memungkinkan informasi untuk disimpan dan diakses dalam jangka waktu panjang [44].




Gambar 2. 3 Arsitektur Jaringan LSTM
Sumber: [41]

Gambar 2.3 mengilustrasikan arsitektur internal sebuah unit LSTM. Struktur ini memiliki sebuah cell state ($\boldsymbol{C_t}$) yang berfungsi seperti "sabuk konveyor", membawa informasi relevan sepanjang urutan data. Aliran informasi pada *cell state* ini diatur oleh tiga gerbang (*gates*) yang berfungsi sebagai penjaga gerbang: *input gate, forget gate,* dan *output gate*. Proses komputasi di dalam LSTM dijelaskan oleh serangkaian rumus berikut [41]:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \quad (2.3)$$

Rumus 2. 3 Rumus *Input Gate*

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_{xf}\boldsymbol{x}_t + \boldsymbol{W}_{hf}\boldsymbol{h}_{f-1} + \boldsymbol{b}_f) \quad (2.4)$$

Rumus 2. 4 Rumus *Forget Gate*

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_{xo}\boldsymbol{x}_t + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \quad (2.5)$$

Rumus 2. 5 Rumus *Output Gate*

$$\mathbf{g}_t = \tanh(\boldsymbol{W}_{xg}\boldsymbol{x}_t + \boldsymbol{W}_{hg}\boldsymbol{h}_{t-1} + \boldsymbol{b}_g) \quad (2.6)$$

Rumus 2. 6 Rumus *Cell Input*

$$\mathbf{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t \quad (2.7)$$

Rumus 2. 7 Rumus *Cell State*

$$\mathbf{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \quad (2.8)$$

Rumus 2. 8 Rumus *Hidden State*

Dalam serangkaian rumus 2.3 hingga 2.8, aliran informasi diatur secara selektif. Pertama, *input gate* ($\boldsymbol{i}_t$) bersama dengan *cell input* ($\mathbf{g}_t$) memutuskan informasi baru apa yang relevan untuk disimpan. Selanjutnya, *forget gate* ($\boldsymbol{f}_t$) menentukan informasi mana dari *cell state* sebelumnya ($\boldsymbol{c}_{t-1}$) yang akan dilupakan. *Cell state* baru ($\boldsymbol{c}_t$) kemudian diperbarui dengan menggabungkan informasi yang tersisa dari masa lalu dan informasi baru yang telah dipilih. Terakhir, *output gate* ($\boldsymbol{o}_t$) menyaring *cell state* yang telah diperbarui untuk menghasilkan *hidden state* ($\boldsymbol{h}_t$), yang akan diteruskan ke langkah waktu berikutnya dan juga dapat digunakan sebagai output prediksi [41].

Dengan struktur ini, LSTM mampu menyimpan informasi penting selama ratusan langkah waktu, tanpa mengalami degradasi performa yang signifikan. Kemampuan ini menjadikan LSTM sangat efektif dalam aplikasi-aplikasi yang bergantung pada urutan data dan konteks jangka panjang, seperti pemrosesan bahasa alami, pengenalan suara, dan yang relevan dalam penelitian ini prediksi harga saham. Dalam konteks pasar saham, pola harga historis sering kali tidak langsung atau linear, dan bisa dipengaruhi oleh kejadian masa lalu yang tidak terjadi dalam jangka pendek. LSTM menawarkan solusi dengan kemampuannya mengenali dan mengingat pola jangka panjang yang tersembunyi dalam data deret waktu (*time series*), seperti fluktuasi mingguan, bulanan, hingga tahunan.

Kinerja dan arsitektur model LSTM yang diterapkan dalam penelitian ini sangat dipengaruhi oleh beberapa *hyperparameter* kunci yang dioptimasi, yaitu:

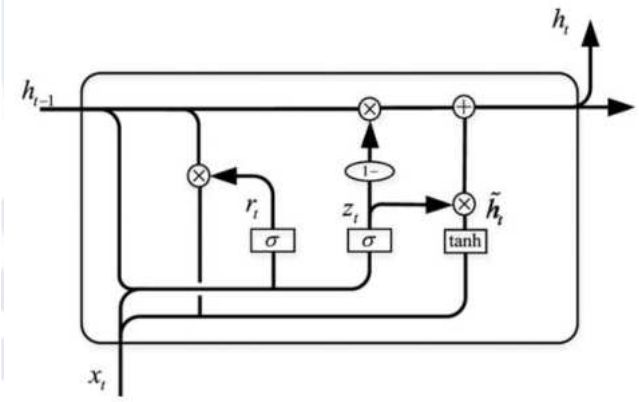
- Jumlah Unit: Merujuk pada dimensionalitas dari *hidden state* ($\boldsymbol{h}_t$) dan *cell state* ($\boldsymbol{c}_t$) di dalam setiap unit LSTM. Jumlah unit yang lebih besar meningkatkan kapasitas LSTM untuk memproses dan mengingat informasi yang lebih kompleks dari data deret waktu.

- *Dropout Rate*: Dalam arsitektur model penelitian ini, lapisan *dropout* ditempatkan setelah lapisan LSTM. *Hyperparameter dropout rate* menentukan fraksi dari output unit LSTM yang secara acak diabaikan selama pelatihan, yang bertujuan untuk mencegah *overfitting* dan meningkatkan kemampuan generalisasi model.

- *Learning Rate*: Mengatur besarnya pembaruan bobot selama proses pelatihan. Ini memengaruhi semua bobot yang dapat dipelajari dalam unit LSTM, termasuk bobot-bobot pada ketiga gerbang (*forget, input, output*) dan bobot yang terlibat dalam pembaruan sel memori, sehingga memainkan peran kunci dalam konvergensi model ke solusi optimal.

#### 2.3.2.3 *Gated Recurrent Unit* (GRU)

*Gated Recurrent Unit* (GRU) merupakan varian dari arsitektur RNN yang dikembangkan untuk menyederhanakan struktur LSTM tanpa mengorbankan kemampuan dalam menangkap ketergantungan jangka panjang pada data sekuensial. GRU diperkenalkan oleh Cho et al. pada tahun 2014 sebagai solusi yang lebih efisien dalam hal komputasi, dengan struktur yang lebih ringan namun tetap efektif dalam mengatasi permasalahan *vanishing gradient* yang juga dialami oleh RNN tradisional [45].




Gambar 2. 4 Arsitektur jaringan GRU
Sumber: [41]

Gambar 2.4 menunjukkan arsitektur jaringan GRU, yang dimana GRU menyederhanakan arsitektur LSTM dengan hanya menggunakan dua

gerbang: *reset gate* dan *update gate*. GRU juga menggabungkan *cell state* dan *hidden state* menjadi satu, sehingga jumlah parameternya lebih sedikit. Proses pembaruan *hidden state* pada GRU dapat dirumuskan sebagai berikut:

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_{xz}\boldsymbol{x}_t + \boldsymbol{W}_{hz}\boldsymbol{h}_{t-1} + \boldsymbol{b}_z) \quad (2.9)$$

Rumus 2. 9 Rumus *Update Gate*

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_{xr}\boldsymbol{x}_t + \boldsymbol{W}_{hr}\boldsymbol{h}_{t-1} + \boldsymbol{b}_r) \quad (2.10)$$

Rumus 2. 10 Rumus *Reset Gate*

$$\boldsymbol{h'}_t = \tanh(\boldsymbol{W}_{xh}\boldsymbol{x}_t + \boldsymbol{r}_t \odot (\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1}) + \boldsymbol{b}_h) \quad (2.11)$$

Rumus 2. 11 *Candidate Hidden State*

$$\mathbf{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \boldsymbol{h'}_t \quad (2.12)$$

Rumus 2. 12 *Current Hidden State*

Dalam serangkaian Rumus 2.9 hingga 2.12, aliran informasi diatur secara efisien. Pertama, reset gate ($\boldsymbol{r}_t$) menentukan seberapa banyak informasi dari masa lalu ($\boldsymbol{h}_{t-1}$) yang harus diabaikan saat menghitung kandidat *hidden state* ($\boldsymbol{h'}_t$). Selanjutnya, update gate ($\boldsymbol{z}_t$) berfungsi sebagai penyeimbang, di mana ia mengatur proporsi antara memori lama ($\boldsymbol{h}_{t-1}$) yang dipertahankan dan kandidat baru ($\boldsymbol{h'}_t$) yang akan ditambahkan untuk membentuk *hidden state* akhir ($\mathbf{h}_t$) yang baru.

Salah satu keunggulan GRU terletak pada efisiensinya dalam pelatihan. Karena arsitekturnya yang lebih ringkas, GRU memerlukan lebih sedikit sumber daya komputasi dan waktu pelatihan dibandingkan LSTM [46]. Hal ini membuat GRU menjadi pilihan yang sangat menarik, khususnya dalam penelitian atau aplikasi dengan keterbatasan waktu atau perangkat keras. Dalam berbagai studi, GRU bahkan menunjukkan performa yang setara atau lebih baik dibandingkan LSTM, terutama ketika digunakan pada dataset dengan ukuran kecil hingga sedang.

Dalam konteks prediksi harga saham, GRU menawarkan keunggulan dalam hal kecepatan pelatihan dan kesederhanaan struktur, yang berguna dalam eksperimen dan iterasi model yang cepat. Meskipun tidak memiliki

sel memori eksplisit seperti LSTM, kemampuan GRU dalam mempertahankan informasi historis yang relevan masih sangat baik, sehingga tetap mampu mengenali pola dalam data deret waktu seperti fluktuasi harga saham. Pada penelitian ini, GRU diimplementasikan dan dibandingkan secara langsung dengan LSTM dan RNN, untuk menilai efektivitas relatif masing-masing model dalam melakukan prediksi harga saham emiten perbankan Indonesia.

Dengan membandingkan performa GRU dengan model-model lain, penelitian ini juga berupaya memberikan wawasan mengenai trade-off antara akurasi prediksi dan efisiensi komputasi, yang penting dalam aplikasi dunia nyata, khususnya ketika pengambilan keputusan harus dilakukan secara cepat dan dengan keterbatasan sumber daya.

Sama halnya dengan model lainnya, efektivitas GRU dalam penelitian ini juga sangat bergantung pada konfigurasi *hyperparameter* spesifik yang ditentukan melalui proses optimasi, meliputi:

- Jumlah Unit: *Hyperparameter* ini mendefinisikan ukuran atau dimensionalitas dari *hidden state* ($\mathbf{h}_t$) dalam setiap unit GRU, yang secara langsung memengaruhi kapasitas model dalam memproses dan menyimpan informasi dari *sequence* data.
- *Dropout Rate*: Serupa dengan LSTM, lapisan *dropout* yang diterapkan setelah lapisan GRU dalam penelitian ini berfungsi untuk mencegah *overfitting*. *Dropout rate* yang dioptimasi akan menentukan seberapa besar regularisasi yang diterapkan.
- *Learning Rate*: Mengendalikan laju pembaruan bobot-bobot internal GRU, termasuk bobot pada *reset gate* dan *update gate*, serta bobot yang terlibat dalam perhitungan *hidden state*. Pemilihan *learning rate* yang tepat sangat penting untuk efektivitas dan efisiensi proses pelatihan model GRU.

### 2.3.3 Metode Evaluasi

Dalam penelitian ini, evaluasi performa model prediksi harga saham dilakukan menggunakan empat metrik evaluasi, yaitu *Mean Absolute Error* (MAE), *Root Mean Squared Error* (RMSE), *Mean Absolute Percentage Error* (MAPE), dan *R-squared* ($R^2$). Masing-masing metrik memiliki karakteristik dan fungsi evaluasi yang berbeda.

#### 2.3.3.1 *Mean Absolute Error (*MAE*)*

MAE mengukur rata-rata nilai absolut selisih antara nilai aktual dan nilai prediksi. MAE memberikan gambaran seberapa besar rata-rata kesalahan prediksi tanpa memperhatikan arah kesalahan (positif atau negatif).

$$MAE = \frac{1}{m}\sum_{i=1}^{m} |X_i - Y_i| \tag{2.13}$$

Rumus 2. 13 Rumus *Mean Absolute Error*

MAE mudah diinterpretasikan karena berada dalam satuan yang sama dengan data asli dan tidak memberikan penalti besar pada kesalahan besar. Namun, MAE tidak sensitif terhadap outlier [47]. Dalam konteks finansial, nilai MAE dapat diinterpretasikan secara langsung sebagai rata-rata selisih absolut antara harga prediksi dengan harga aktual dalam satuan mata uang, memberikan gambaran praktis mengenai besaran rata-rata kesalahan prediksi.

#### 2.3.3.2 *Root Mean Squared Error (*RMSE*)*

RMSE adalah akar kuadrat dari rata-rata kuadrat selisih antara nilai aktual dan prediksi. RMSE memberikan penalti yang lebih besar pada kesalahan besar karena menggunakan kuadrat selisih.

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(X_i - Y_i)^2} \tag{2.14}$$

Rumus 2. 14 Rumus *Root Mean Squared Error*

RMSE berguna untuk menilai seberapa jauh prediksi model menyimpang dari nilai aktual secara umum, dan lebih sensitif

terhadap *outlier* dibanding MAE [47]. Serupa dengan MAE, nilai RMSE juga berada dalam satuan yang sama dengan data asli. Dalam aplikasi prediksi harga saham, RMSE mengukur besaran tipikal dari kesalahan prediksi dan dapat digunakan untuk mengestimasi potensi risiko finansial dari ketidakakuratan model.

#### 2.3.3.3 *Mean Absolute Percentage Error (*MAPE*)*

MAPE mengukur rata-rata persentase kesalahan absolut antara nilai aktual dan prediksi, sehingga memberikan gambaran kesalahan relatif terhadap nilai aktual.

$$MAPE = \frac{1}{m}\sum_{i=1}^{m}\left|\frac{X_i - Y_i}{Y_i}\right| \tag{2.15}$$

Rumus 2. 15 Rumus *Mean Absolute Percentage Error*

MAPE mudah dipahami karena dinyatakan dalam persen, tetapi memiliki kelemahan jika nilai aktual mendekati nol karena dapat menghasilkan nilai yang sangat besar atau tak terdefinisi [47]. Dalam konteks prediksi harga saham, MAPE berguna untuk menilai seberapa besar kesalahan prediksi dalam proporsi terhadap nilai harga aktual, sehingga memudahkan perbandingan kinerja antar saham atau model yang memiliki skala nilai yang berbeda.

#### 2.3.3.4 *R-squared (*$R^2$*)*

$R^2$ mengukur proporsi variansi data aktual yang dapat dijelaskan oleh model prediksi. Nilai $R^2$ berkisar antara 0 hingga 1, di mana nilai yang lebih tinggi menunjukkan model yang lebih baik dalam menjelaskan variabilitas data.

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(X_i - Y_i)^2}{\sum_{i=1}^{m}(\bar{Y} - Y_i)^2} \tag{2.16}$$

Rumus 2. 16 Rumus *R-squared*

$R^2$ memberikan informasi yang lebih komprehensif dibanding MAE, MAPE, dan RMSE karena memperhitungkan variabilitas data dan kesalahan prediksi secara keseluruhan [47]. Dalam aplikasi

keuangan, $R^2$ menunjukkan seberapa baik model mengikuti pergerakan tren harga saham aktual. Semakin tinggi nilai $R^2$, semakin besar proporsi pergerakan harga saham yang berhasil diprediksi dengan baik oleh model.

## 2.4 Alat Penelitian

### 2.4.1 *Google Colaboratory*

*Google Colaboraroty* adalah *platform cloud* berbasis *Jupyter Notebook* yang disediakan oleh Google, memungkinkan pengguna menjalankan kode Python secara langsung di browser. Fitur unggulan dari Colab meliputi [48]:

- Mendukung GPU dan TPU secara gratis, mempermudah pelatihan model *deep learning*.
- Integrasi penuh dengan *Google Drive* untuk manajemen file.
- Mendukung instalasi pustaka Python secara dinamis dan mendukung eksekusi *real-time*.

Google Colab menjadi pilihan ideal untuk penelitian ini karena memfasilitasi eksperimen yang cepat dan efisien tanpa memerlukan spesifikasi perangkat keras lokal yang tinggi.

### 2.4.2 TensorFlow

TensorFlow adalah *library open-source* yang dikembangkan oleh Google untuk keperluan komputasi numerik dan *deep learning* [49]. TensorFlow menyediakan API untuk membangun dan melatih berbagai jenis jaringan saraf, termasuk RNN, LSTM, dan GRU. Fitur utama TensorFlow :

- Mendukung training model berbasis CPU maupun GPU.
- API tinggi (tf.keras) memudahkan pembuatan arsitektur model yang kompleks.
- Kompatibel dengan Google Colab dan ekosistem Python.

Dalam penelitian ini, TensorFlow digunakan sebagai backend utama dalam membangun model *deep learning* yang akan dibandingkan.

### 2.4.3 Python

Python adalah bahasa pemrograman tingkat tinggi yang sangat populer di bidang *data science* dan *machine learning* [50]. Keunggulan Python meliputi:

- Sintaks yang mudah dipahami dan ditulis.
- Dukungan *library* yang luas seperti pandas untuk analisis data, matplotlib dan seaborn untuk visualisasi, scikit-learn untuk *preprocessing* dan evaluasi model.
- Integrasi mulus dengan TensorFlow, Keras, dan platform cloud seperti Colab.

Python menjadi bahasa utama dalam penelitian ini karena komunitasnya yang besar dan dokumentasi yang lengkap, yang mendukung proses eksplorasi dan pengembangan model prediktif secara efisien.