BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Beberapa penelitian telah dilakukan eksperimen menggunakan berbagai metode, baik sederhana maupun kompleks dalam melakukan prediksi data *time series*. Penelitian yang dilakukan Pangaribuan, dkk (2023), menerapkan metode ARIMA untuk memprediksi penjualan bisnis rumah properti. Peneliti menggunakan dataset *time series* yang diperoleh dari perusahaan HtAG *Analytics*. Dalam penelitian ini, peneliti memvisualisasikan data terlebih dahulu untuk mengidentifikasi pola, tren, dan anomali pada dataset. Peneliti melakukan uji stasioneritas pada dataset menggunakan metode statistik, seperti *Augmented Dickey-Fuller* (ADF) atau *Kwiatkowski-Phillips-Schmidt-Shin* (KPSS), untuk memastikan bahwa data memiliki rata-rata, variansi, dan autokorelasi yang konstan sepanjang waktu. Ketika data tidak stasioner, peneliti menggunakan teknik *differencing* untuk menghilangkan tren atau komponen non-stasioner dengan cara mengurangi nilai data saat ini dengan nilai sebelumnya. Setelah data menjadi stasioner, peneliti menganalisis grafik *Autocorrelation Function* (ACF) dan *Partial Autocorrelation Function* (PACF) untuk menentukan parameter *p* (*AutoRegressive Order*)*, d* (*Differencing Order*)*,* dan *q* (*Moving Average Order*)*.*

Parameter *p* mengacu pada titik waktu sebelumnnya dalam model *autoregressive* (AR), yaitu hubungan antara nilai data saat ini dengan nilai data sebelumnya. Parameter *d* menunjukkan tingkat *differencing* yang diperlukan untuk membuat data menjadi *stasioner.* Parameter *q* merujuk pada jumlah *lag* dalam model *moving average* (MA), yaitu hubungan antara nilai data saat ini dengan *error* residual dari *lag* sebelumnya. Peneliti menerapkan model ARIMA pada data *time series* dengan parameter yang telah ditentukan dan mengevaluasi performa model menggunakan *Mean Squared Error* (MSE), *Mean Absolute Percentage Error* (MAPE), dan *Root Mean Squared Error* (RMSE). Hasil evaluasi menunjukkan bahwa model ARIMA memiliki nilai MSE sebesar 0,079191, MAPE sebesar 3,4%, dan RMSE sebesar 0,281409. Penelitian ini membuktikan bahwa metode ARIMA

dapat digunakan secara efektif untuk memprediksi data *time series,* terutama dalam konteks penjualan rumah properti. Namun, metode ARIMA masih memiliki keterbatasan, seperti teknik *differencing* yang digunakan hanya efektif pada tren linier, sehingga untuk tren yang lebih kompleks, mungkin diperlukan metode tambahan. Penyesuaian parameter *p, d,* dan *q* juga sangat bergantung pada interpretasi grafik ACF dan PACF, yang sering kali bersifat subyektif dan dapat menyebahkan kesalahan dalam pemodelan. ARIMA juga mengasumsikan bahwa hubungan dalam data bersifat linier dan tidak mempertimbangkan faktor eksternal yang mungkin memengaruhi pergerakan data. Model ini juga cenderung kurang efektif dalam menangkap pola yang kompleks dibandingkan dengan metode berbasis *deep learning*, seperti *Long Short-Term Memory* (LSTM). LSTM memiliki keunggulan dalam mengenali pola jangka panjang dan hubungan non-linier dalam data *time series*, sehingga lebih sesuai untuk menangani data dengan karakteristik yang lebih dinamis. Selain itu, evaluasi model ARIMA hanya dilakukan pada dataset penelitian ini, sehingga penerapan model pada dataset lain memerlukan validasi tambahan. Dengan tahapan yang terstruktur dan evaluasi yang baik, penelitian ini memberikan kontribusi signifikan dalam penggunaan metode ARIMA untuk analisis data *time series*, namun perlu mempertimbangkan alternatif model seperti LSTM untuk menangani data dengan pola yang lebih kompleks.

Penelitian lain yang dilakukan Aji, dkk (2022), mengusulkan model prediksi *Simple Moving Average* (SMA), *Weighted Moving Average* (WMA) dan *Exponential Smoothing* (ES), untuk membandingkan metode peramalan yang efektif. Penelitian tersebut menggunakan dataset dari penjualan obat pada Klinik Doa Sehat selama 3 tahun terakhir. Proses yang dilakukan adalah mengolah data dengan masing-masing metode secara bergantian. Data peramalan yang digunakan dipilih selama pertiga bulan untuk metode SMA, dan 1 tahun untuk WMA dan ES. Metode pertama yang digunakan adalah metode SMA. Prediksi yang dilakukan dengan cara mencari nilai rata-rata dari data *n* periode sebelumnya. Metode kedua yang digunakan adalah metode WMA. Dalam metode kedua dilakukan proses memberikan bobot pada data *n* periode sebelumnya dan dikalikan dengan permintaan dalam periode *n* serta membagi dengan jumlah bobot. Bobot terbesar

diberikan ke data satu periode sebelumnya. Sementara dalam metode ketiga ES dilakukan dengan pembobotan pada titik-titik yang diberi bobot oleh fungsi eksponensial. Selanjutnya dilakukan evaluasi metode menggunakan MSE, MAD, dan MAPE. Hasil eksperimen dalam penelitian tersebut menghasilkan rata-rata akurasi nilai MSE 1749,587 (SMA), 1628,449 (WMA), 1341,137 (ES). MAD 1,040 (SMA), 1028 (WMA), 883 (ES). MAPE 2,63 % (SMA), 2,27% (WMA), 0,90% (ES). Namun, metode SMA, WMA, dan ES tersebut lebih cocok digunakan untuk data dengan pola yang stabil dan tidak mempertimbangkan hubungan jangka panjang antar data.

Penelitian yang dilakukan Selle, dkk (2022), mengevaluasi beberapa metode *deep learning* untuk memprediksi penggunaan listrik. Eksperimen yang dilakukan adalah dengan membandingkan hasil prediksi dari metode LSTM dengan RNN. Hasil dari penelitian tersebut menunjukkan bahwa kinerja metode LSTM lebih baik dan mengungguli kinerja metode lainnya untuk memprediksi penggunaan listrik berikutnya. Hasil RMSE terendah untuk metode LSTM sebesar 49,18, dan RNN sebesar 50,35. Sementara dalam penelitian (Primawati, dkk, 2023), membandingkan metode LSTM dan *Prophet* dalam data *time series* pada produksi susu sapi harian. Hasil yang didapatkan kinerja metode LSTM lebih baik dibandingkan metode *Prophet*. Hasil evaluasi kinerja metode LSTM dengan MSE sebesar 39,293, RMSE sebesar 6,268, MAPE sebesar 9,98%, dan R^2 sebesar 0,3789. *Prophet* dengan nilai MSE sebesar 41,128, RMSE sebesar 6,413, MAPE sebesar 9,99%, dan R^2 sebesar 0,3496.

Penerapan metode LSTM juga telah dilakukan untuk melakukan peramalan deret waktu. Dalam penelitian Sirisha, dkk (2022) melakukan pembangunan model *time series* untuk memprediksi laba saham dan membandingkan performa model yang dihasilkan dari metode ARIMA, SARIMA, dan LSTM. Penelitian tersebut menggunakan dataset penjualan dari *eforexcel.com*. Berdasarkan hasil evaluasinya, metode LSTM menghasilkan performa kinerja yang lebih baik dalam melakukan prediksi dengan akurasi RMSE 3,917264522. ME 0,470382847. MPE 0,004556761. MAE 3,257199791, MAPE 0,029891568, *Corr* 0,840131571. *MinMax Error* 0,178130191. Berdasarkan evaluasi tersebut, rata-rata *accuracy*

metode LSTM dengan MAPA sebesar 97,01%. Sementara untuk metode ARIMA rata-rata akurasi MAPA sebesar 93,84% dan SARIMA 94,38%.

Keunggulan kinerja metode LSTM dalam melakukan prediksi juga telah dibuktikan dalam penelitian Meng, dkk (2021). Dalam penelitiannya, mengusulkan metode Prophet dan LSTM dalam menguji performa untuk melakukan prediksi. Penelitian tersebut menggunakan dataset *time series* penjualan obat selama 3 tahun. Proses dalam penelitian tersebut dilakukan secara bergantian dengan masing-masing metode. Metode pertama yang digunakan adalah metode Prophet. Data yang diperlukan metode Prophet adalah kolom ds (tanggal dan waktu) dan y (jumlah penjualan). Proses yang dilakukan adalah memetakan input y dengan interval (0,1), menggunakan standar deviasi dan hari libur. Menetapkan titik perubahan 0,15 dan interval prediksi 1 tahun ke depan. Proses metode Prophet ini juga menggunakan fungsi validasi silang untuk mengukur kesalahan prediksi menggunakan data historis. Fungsi validasi silang digunakan untuk proses verifikasi dengan menentukan inisial 900 hari, periode 55 hari, dan horizon 110 hari. Sementara dalam metode LSTM dilakukan transformasi data asli secara linear menggunakan standar deviasi dan memetakan nilai data asli antara 0 / 1. Membagi data latih 80% dan data uji 20%. Inisialisasi metode LSTM dengan *random seed* 7, *input layer* 1, *hidden layer* 40 *neuron*, aktivasi fungsi *sigmoid*, iterasi 50 kali, dan *batch size* 80. Berdasarkan hasil eksperimen, kedua metode tersebut menghasilkan kinerja yang baik dalam melakukan prediksi. Secara keseluruhan, eksperimen yang dilakukan metode LSTM menunjukkan kinerja yang lebih baik dibandingkan metode *Prophet*. Hasil perbandingan akurasi kesalahan RMSE untuk kedua metode tersebut adalah 0.047 (LSTM) dan 0.050 *(Prophet).*

Berdasarkan keunggulan model LSTM, masih terdapat kelemahan dalam hal akurasi saat memprediksi data *time series* dengan data yang berbeda, seperti yang ditunjukkan dalam penelitian Alim (2023), pada penelitiannya menghasilkan akurasi dari data saham ANTM dan saham ICBP, dengan akurasi MAE sebesar 0,0079, MAPE sebesar 40,8%, dan RMSE 0,013 pada saham ANTM, sementara dalam data saham ICBP akurasi yang dihasilkan untuk MAE sebesar 0,0075, MAPE sebesar 3,3%, RMSE 0,0122. Penelitian lain yang dilakukan Suharmanto &

Ernawati (2023), menunjukkan bahwa akurasi peramalan yang dihasilkan dengan metode ARFIMA dan LSTM dengan data saham BRI dan Bank IBK, menghasilkan akurasi SMAPE untuk saham BRI paling tinggi sebesar 23,31% dan untuk Bank IBK sebesar 54,09%.

Penelitian mengenai optimasi model *deep neural network*, dalam meningkatkan akurasi peramalan telah dilakukan beberapa peneliti, seperti Reyad, dkk (2023), dalam penelitiannya menerapkan algoritma optimasi untuk melatih model jaringan saraf tiruan. Berdasarkan hasil eksperimen, penerapan algoritma optimasi dan fungsi aktivasi sangat mempengaruhi akurasi model prediksi. Hasil eksperimen menyatakan bahwa algoritma yang digunakan seperti *HN_Adam, AdaBelief, Adam,* SGD, Yogi*, RAdam,* MSVAG menghasilkan akurasi yang cukup baik dengan pemilihan dua fungsi aktivasi ReLU dan *Softmax*. Akurasi yang dihasilkan sebesar 73,20% (*HN_Adam*), 70,08% (*AdaBelief*), 63,79% (*Adam*), 70,23% (SGD), 68,23% (Yogi), 67,62% (*RAdam*), dan 65,99% (MSVAG).

Penerapan teknik optimasi juga telah dibuktikan dalam penelitian Setiawan, dkk (2022), dalam meningkatkan akurasi peramalan. Dalam penelitiannya mengevaluasi metode LSTM dengan menerapkan optimasi RMSProp dan Adam dengan mengevaluasi akurasi menggunakan MSE dan MAPE. Berdasarkan hasil penelitian yang dilakukan prediksi menggunakan teknik optimasi Adam lebih baik dari pada optimasi RMSProp. Hasil akurasi yang dihasilkan pada evaluasi MSE sebesar 491505,1, dan MAPE sebesar 1,739155%.

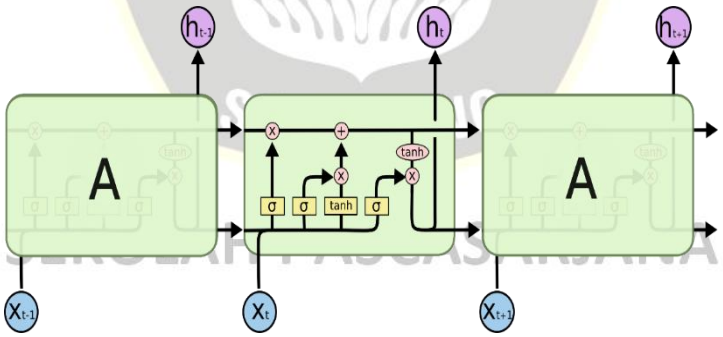
2.2 Dasar Teori

2.2.1 *Long Short-Term Memory*

Long Short-Term Memory (LSTM) adalah modifikasi dari *Recurrent Neural Network* (RNN) dengan menambahkan *memory cell* yang mampu menyimpan informasi dalam waktu lama. LSTM menjadi solusi untuk menghindari masalah ketergantungan jangka panjang pada RNN. LSTM dirancang untuk mengatasi masalah *difusi gradien* pada RNN (Houdt, dkk, 2020). Komponen utama lapisan LSTM disebut dengan sel memori, yang terdiri dari 4 elemen utama, seperti *input gate, neuron* dengan koneksi berulang, *forget gate*, dan *output gate*. *Input* yang

diberikan pada LSTM mengontrol operasi yang dilakukan oleh *gate* pada sel memori (Liang, dkk, 2022), seperti *input gate*, *output gate*, *forget gate* (Septiadi, dkk, 2020).

Arsitektur LSTM memungkinkan penambahan atau penghapusan informasi dari *cell state* melalui proses yang disebut *gates*. *Gates* berfungsi untuk mengatur informasi yang akan diteruskan atau dihentikan. Proses ini menggunakan lapisan *sigmoid* dan operasi perkalian elemen (*pointwise multiplication*). Lapisan *sigmoid* menghasilkan *output* berupa nilai antara 0 dan 1 dalam bentuk matriks atau vektor, yang menunjukkan tingkat pengaruh informasi. Nilai mendekati 1 menunjukkan informasi tersebut akan diteruskan, sementara nilai mendekati 0 menunjukkan informasi tersebut akan diabaikan (Septiadi, dkk, 2020). Dimensi *output* dari lapisan *sigmoid* sama dengan dimensi *hidden state*, sehingga pengolahan informasi dilakukan secara elemen demi elemen. *Cell* pada LSTM mampu menghubungkan informasi sebelumnya dengan informasi selanjutnya, sehingga sangat efektif dalam menyimpan informasi jangka panjang yang diperlukan untuk mengolah data *time series*. Gambaran arsitektur dari *Long Short-Term Memory* (LSTM) ini ditunjukkan dalam Gambar 2.1 (Septiadi, dkk, 2020).

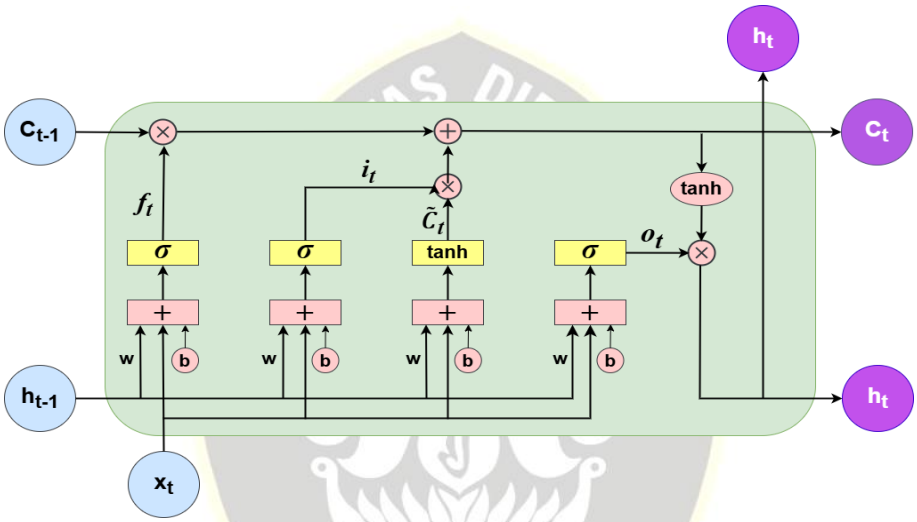


Gambar 2.1 Arsitektur *Long Short-Term Memory* (LSTM)

Berdasarkan Gambar 2.1, model LSTM menyaring informasi melalui struktur gerbang untuk mempertahankan dan memperbarui keadaan *cell* memori. Setiap *cell* memori memiliki tiga lapisan *sigmoid* dan satu lapisan *tanh*, serta lapisan tersembunyi yang terdiri dari *cell* memori. Satu *cell* memori tersusun atas tiga *gates*, yaitu (Chang, dkk, 2019):

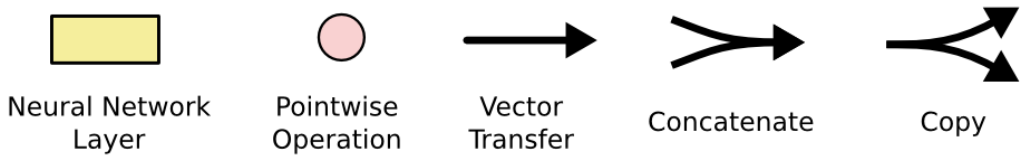
a. *Forget Gate*, merupakan *gate* yang memutuskan informasi yang akan dihapus atau disimpan dari *cell*.
b. *Input Gate*, merupakan *gate* yang memutuskan nilai dari *input* untuk di perbarui pada *state* memori.
c. *Output Gate*, merupakan *gate* yang memutuskan nilai yang akan dihasilkan sesuai dengan *input* dan memori pada *cell*.

Gambar 2.2 merupakan gambaran dari *cell* memori *Long Short-Term Memory* (LSTM) (Subowo, dkk, 2022).



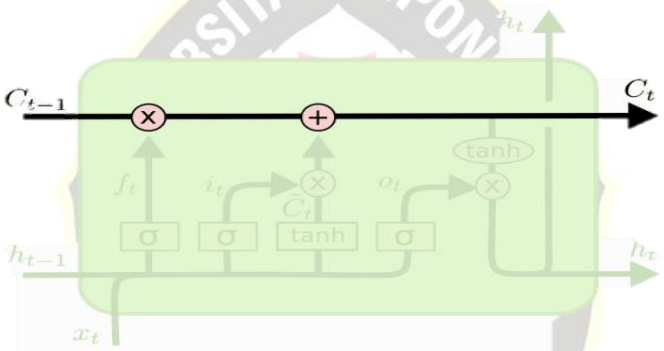
Gambar 2.2 *Cell* Memori *Long Short Memory* (LSTM)

Gambar 2.1 dan Gambar 2.2 menampilkan berbagai garis, simbol, dan warna. Setiap garis merepresentasikan vektor keluaran dari suatu simpul yang menuju *input* simpul lainnya. Lingkaran merah muda menunjukan operasi titik (*Pointwise Operation*), seperti penambahan vector (*Vector Transfer*), sedangkan kotak kuning mewakili lapisan jaringan saraf yang dipelajari (*Neural Network Layer*). Penggabungan garis menunjukkan proses penggabungan (*Concatenate*), sementara percabangan garis (*Copy*) menandakan bahwa kontennya disalin dan dikirim ke lokasi berbeda. Gambar 2.3 menggambarkan komponen yang menjelaskan notasi arsitektur LSTM yang terdapat pada Gambar 2.1 dan Gambar 2.2. (Olah, 2015).



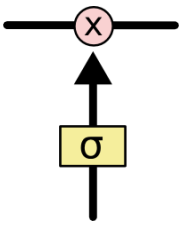
Gambar 2.3 Komponen Notasi Arsitektur *Long Short Memory* (LSTM)

Kunci utama *Long Short Memory* (LSTM) adalah status sel (Ct). *Cell* LSTM memiliki dua keluaran, yaitu *hidden state* (h_t), dan *cell state* (Ct). *Cell state* atau status sel merupakan garis horizontal yang menghubungkan semua lapisan keluaran LSTM, seperti yang ditunjukan pada Gambar 2.4 (Olah, 2015).



Gambar 2.4 *Cell State Long Short-Term Memory*

LSTM memiliki mekanisme untuk menambah dan menghapus data dari status sel, yang disebut dengan *gate*. Fungsi *gate* adalah memeriksa data yang masuk menggunakan lapisan *sigmoid*, yang terdiri dari lapisan jaringan saraf dan operasi *pointwise*, seperti yang ditunjukkan pada Gambar 2.5 (Olah, 2015).

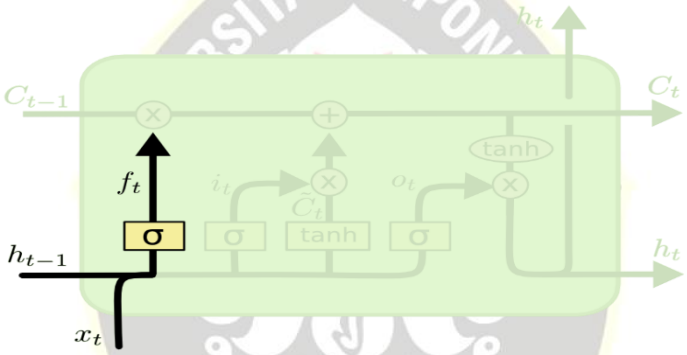


Gambar 2.5 *Sigmoid Layer Long Short-Term Memory*

Berdasarkan ilustrasi gambar-gambar sebelumnya, proses tahap metode LSTM memiliki langkah-langkah perulangan pada arsitektur LSTM, yaitu (Septiadi, dkk, 2020):

a. Memutuskan informasi yang akan dihapus atau dibuang (*Forget gate*)

Langkah pertama adalah memutuskan informasi apa yang akan dibuang dari *state cell* (Status sel). Keputusan ini dibuat oleh lapisan *sigmoid,* yaitu *forget gate layer*. Dalam lapisan *forget gate*, *input* berupa h_{t-1} dan x_t diproses, menghasilkan *output* berupa angka 0 dan 1 pada status sel C_{t-1}. Angka mendekati 1 menunjukkan informasi akan disimpan, sedangkan angka 0 menunjukkan informasi akan dihapus atau dibuang, seperti yang ditunjukkan pada Gambar 2.6 (Septiadi, dkk, 2020).



Gambar 2.6 Langkah *Forget Gate Layer*

Persamaan yang digunakan untuk menghitung *forget gate* dinotasikan dalam persamaan 2.1 (Joseph, dkk, 2022).

$$f_t = \sigma\,(W_f . x_t + U_f . h_{t-1} + b_f) \quad (2.1)$$

Keterangan:

f_t : *Forget gate*

σ : Fungsi aktivasi *sigmoid*

W_f : Nilai bobot untuk *forget gate*

U_f : Nilai bobot berulang untuk *forget gate*

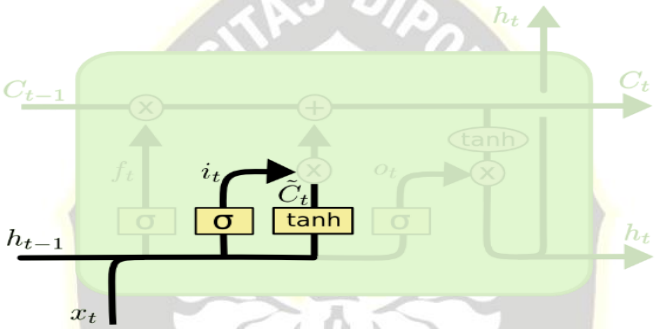
h_{t-1} : Nilai *output* sebelum waktu ke t

x_t : Nilai *input* pada waktu ke t

b_f : Nilai bias pada *forget gate*

b. Menyimpan informasi ke dalam c*ell state* (*Input gate*)

Tahapan ini dilakukan dengan memutuskan informasi yang akan disimpan ke dalam *cell state* C_t. Proses ini terdiri dari 2 bagian yaitu, bagian pertama *sigmoid layer* atau *input gate layer*, yang menentukan nilai yang akan diperbarui, bagian kedua adalah *tanh layer* yang digunakan untuk membuat kandidat dengan nilai baru $\tilde{C}_t$ yang dapat ditambahkan ke dalam *cell state*. Tahap ketiga *output* dari *input gate layer* dan *tanh layer* digabungkan untuk memperbarui *cell state*, seperti yang ditunjukkan pada Gambar 2.7 (Septiadi, dkk, 2020). Persamaan yang digunakan untuk menghitung *input gate* ini dinotasikan dalam Persamaan 2.2 (Joseph, dkk, 2022).



Gambar 2.7 Langkah *Input Gate Layer dan Tanh Layer*

$$i_t = \sigma\,(W_i.x_t + U_i.h_{t-1} + b_i) \tag{2.2}$$

Keterangan:

i_t : *Input gate*

σ : Fungsi *sigmoid*

W_i : Nilai bobot untuk *input gate*

U_i : Nilai bobot berulang untuk *input gate*

h_{t-1} : Nilai *output* sebelum orde ke t

x_t : Nilai *input* pada orde ke t

b_i : Nilai bias pada *input gate*

Terdapat juga persamaan untuk mencari kandidat baru yang akan digunakan, seperti yang dinotasikan dalam persamaan 2.3 (Joseph, dkk, 2022).

$$\tilde{C}_t = \tanh\,(W_c.x_t + U_c.h_{t-1} + b_c) \tag{2.3}$$

Keterangan:

$\tilde{C}_t$: Nilai baru yang dapat ditambahkan ke *cell state*

tanh : Fungsi *tanh*

W_c : Nilai bobot untuk *cell state*

U_c : Nilai bobot berulang untuk *cell state*

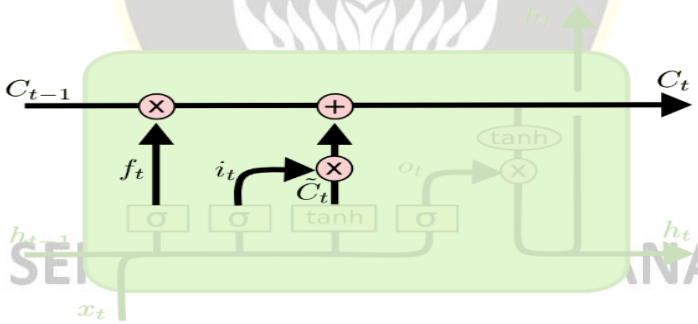
h_{t-1} : Nilai *output* sebelum orde ke t

x_t : Nilai *input* pada orde ke t

b_c : Nilai bias pada status sel (*cell state*)

c. Memperbaharui status sel (*Cell State*)

Langkah selanjutnya adalah memperbaharui *cell state* (status sel) lama (C_{t-1}) menjadi *cell state* baru (C_t). Proses ini dilakukan dengan mengalikan (*) *cell state* lama dengan f_t untuk menghapus data yang ditentukan sebelumnya pada langkah *forget gate layer* atau tahap pertama. Kemudian, *cell state* baru ditambahkan dengan nilai kandidat baru $i_t * \tilde{C}_t$ yang digunakan untuk memperbarui *state*. Proses pembaharuan *cell state* ini dapat dilihat pada Gambar 2.8 (Septiadi, dkk, 2020), dan persamaan *cell state* dinotasikan dalam persamaan 2.4 (Joseph, dkk, 2022).



Gambar 2.8 Langkah update status sel (*cell state*)

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (2.4)$$

Keterangan:

C_t : *Cell state*

f_t : *Forget state*

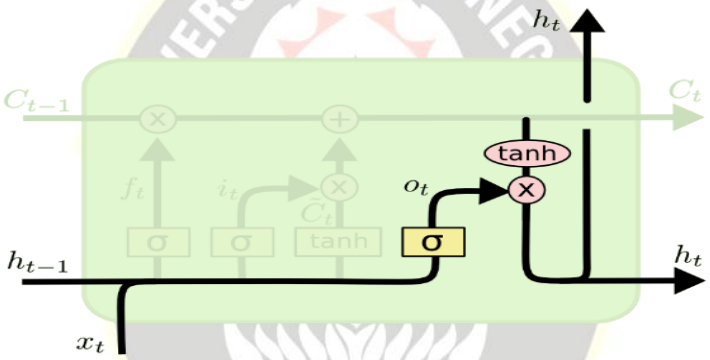
C_{t-1} : *Cell state* sebelum orde ke t

i_t : *Input gate*

$\tilde{C}_t$: Nilai baru yang ditambahkan ke *cell state*

d. Mengkonfirmasi hasil *output* h_t

Langkah terakhir adalah menentukan hasil keluaran (*output*) pada *output gate*. Hasil *output* harus sesuai dengan *cell state* yang telah diolah sebelumnya. Pada langkah pertama, lapisan *sigmoid* menentukan nilai dari *cell state* yang akan menjadi hasil *output*. Langkah kedua, *output* dari *cell state* ditambahkan ke dalam lapisan *tanh* (untuk mengubah nilai dari -1 menjadi 1), kemudian dikalikan dengan *sigmoid gate*, agar *output* yang dihasilkan sesuai dengan yang ditentukan sebelumnya. Proses ini dapat dilihat pada Gambar 2.9, yang menampilkan persamaan untuk menentukan *output* (Septiadi, dkk, 2020).



Gambar 2.9 Langkah menentukan *Output*

Persamaan yang digunakan untuk menghitung nilai dari *output gate* dinotasikan dalam persamaan 2.5 (Joseph, dkk, 2022).

$$o_t = \sigma\,(W_o.x_t + U_o.h_{t-1} + b_o) \quad (2.5)$$

Keterangan:

o_t : *Output gate*

σ : Fungsi *sigmoid*

W_o : Nilai bobot untuk *output gate*

W_o : Nilai bobot berulang untuk *output gate*

h_{t-1} : Nilai *output* sebelum orde ke t

x_t : Nilai *input* pada orde ke t

b_o : Nilai bias pada *output gate*

Sedangkan persamaan nilai *output* orde *t* diuraikan dalam persamaan 2.6 (Joseph, dkk, 2022).

$$h_t = o_t * \tanh(c_t) \quad (2.6)$$

Keterangan:

h_t : Nilai *output* orde ke t

o_t : *Output gate*

tanh : Fungsi *tanh*

c_t : *Cell state*

2.2.2 Arsitektur *Long Short-Term Memory* Dalam *Time Series*

Long Short-Term Memory (LSTM) adalah jenis *Recurrent Neural Network* (RNN) yang sangat efektif dalam menangani masalah *vanishing gradient*, menjadikannya sangat cocok untuk data *time series*. LSTM mampu menangkap pola jangka panjang dalam data, sehingga ideal untuk peramalan data seperti harga saham. LSTM memiliki sel memori yang memungkinkan jaringan untuk menyimpan informasi dalam jangka waktu yang lama. Komponen utama dari LSTM meliputi, *Forget Gate* yang digunakan untuk memutuskan informasi yang akan dibuang dari sel memori. *Input Gate,* digunakan untuk memutuskan informasi yang akan ditambahkan ke sel memori, dan *Output Gate* digunakan untuk memutuskan informasi atau nilai dari sel memori yang akan dikeluarkan sebagai *output*.

Data *time series* adalah kumpulan data yang diperoleh dari serangkaian titik waktu tertentu yang berurutan. Dalam konteks peramalan harga saham, data *time series* umumnya terdiri dari dua komponen utama, yaitu waktu (tanggal pengamatan) dan variabel yang diamati (misalnya, harga penutupan saham). Data *time series* seperti ini dapat memberikan wawasan mengenai pola, tren, dan fluktuasi pasar seiring berjalannya waktu.

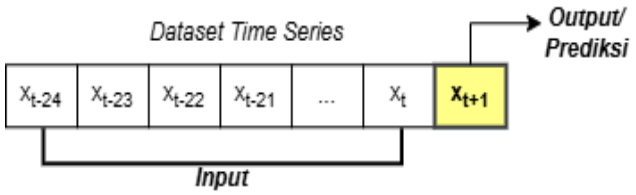
Tabel 2.1 merupakan contoh data *time series* saham ANTM.JK berdasarkan harga penutupan. Dataset ini merupakan contoh representatif dari data *time series*, di mana setiap baris mewakili observasi pada titik waktu tertentu. Dataset diperoleh dari sumber *Yahoo Finance (https://finance.yahoo.com/)*.

Tabel 2.1 Contoh Data *Time Series*

Time	***Close Price***
2/1/2024	1.735
3/1/2024	1.700
…	…
30/1/2024	1.565
31/1/2024	1.550

Pada peramalan data *time series* penyusunan data *input* pada harga saham dilakukan dengan menggunakan harga penutupan yang disusun secara sekuensial menggunakan pendekatan *windowing*. Sebagai contoh, sekuens harga penutupan saham selama 25 hari digunakan untuk memprediksi harga penutupan pada hari berikutnya. Oleh karena itu, setiap input model LSTM memiliki format yang disusun dalam bentuk (*batch size, time steps, features*). *Time steps* merujuk pada panjang *window* atau variabel *lag* (misalnya, 25 hari), sedangkan *features* adalah jumlah fitur atau variabel *input* yang digunakan (misalnya, harga penutupan (1)).

Variabel *lag* dalam konteks ini mengacu pada panjang *window*, yaitu jumlah hari sebelumnya yang digunakan untuk membuat prediksi. Sebagai contoh, jika harga penutupan saham selama 30 hari adalah (100, 101, 102, ..., 129), dan kita menggunakan *window* dengan panjang 25 hari, maka input pertama untuk model adalah harga penutupan selama 25 hari pertama (100, 101, 102, ..., 124), dengan target prediksi pertama adalah harga pada hari ke-26, yaitu 125. Kemudian, *input* kedua adalah data harga penutupan dari hari ke-2 hingga hari ke-26 (101, 102, 103, ..., 125), dan target prediksi kedua adalah harga pada hari ke-27, yaitu 126. Proses ini berlanjut dengan menggeser *window* satu hari ke depan hingga semua data digunakan, seperti yang ditunjukkan pada Gambar 2.10.



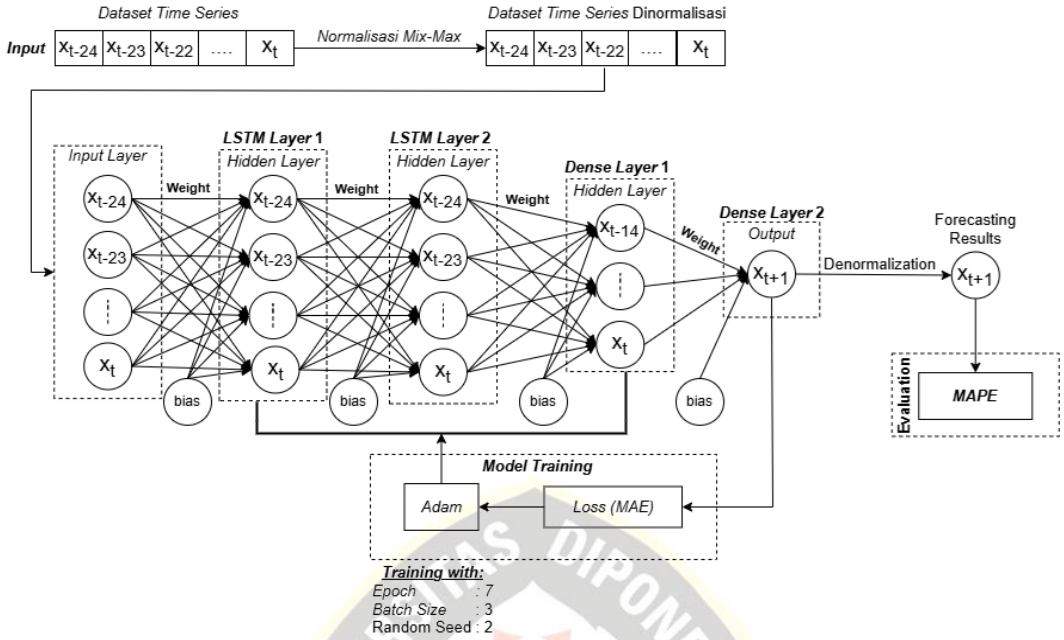
Gambar 2.10 Prediksi *Time Series*

Dalam mengimplementasikan model LSTM untuk peramalan data *time series*, struktur model terdiri dari beberapa lapisan, yaitu lapisan LSTM, lapisan *Dense*, dan parameter model. *Input layer* pada lapisan LSTM mengatur *input* dalam bentuk (*batch size, time steps, features*), yang terdiri dari 1 fitur, yaitu fitur '*Close*' dalam data saham.

Model LSTM pada penelitian ini memiliki 2 lapisan LSTM. Lapisan LSTM pertama menggunakan 25 neuron dengan parameter *return_sequences=True*, yang berfungsi untuk menangkap pola temporal dari seluruh *sekuens input*. Lapisan LSTM kedua, yang juga menggunakan 25 neuron, memiliki parameter *return_sequences=False,* dan berfungsi untuk menghasilkan *output* hanya pada langkah waktu terakhir. Kedua lapisan LSTM ini termasuk dalam *hidden layers*.

Pada lapisan *Dense,* umumnya hanya digunakan 1 *layer* dengan 1 *neuron* sebagai *output layer* untuk menghasilkan hasil prediksi. Namun, pada penelitian ini, ditambahkan 2 lapisan *Dense*. Lapisan *Dense* pertama berfungsi sebagai hidden layer dengan 15 *neuron*, yang menerima *output* dari lapisan LSTM kedua. Lapisan ini mengolah informasi yang telah diproses oleh LSTM dan menghitung *output*nya tanpa pengelolaan memori tambahan. Sementara itu, lapisan *Dense* kedua dengan 1 *neuron* berfungsi untuk menghasilkan output prediksi.

Untuk proses pelatihan model, digunakan parameter model seperti *optimizer Adam* yang berfungsi untuk mempercepat konvergensi model dengan pengaturan *learning rate* adaptif, dan fungsi kerugian (*loss function*) MAE (*Mean Absolute Error*) yang digunakan untuk mengukur kesalahan prediksi model. Proses prediksi dengan model LSTM ini ditunjukkan pada Gambar 2.11.



Gambar 2.11 Arsitektur LSTM Dalam *Time Series*

2.2.3 *Hyperparameter Long Short-Term Memory* (LSTM)

Dalam proses pembuatan model LSTM, diperlukan beberapa *hyperparameter* yang ditentukan di awal dan tetap kontinu selama proses pelatihan. Parameter yang digunakan antara lain jumlah *neuron, epoch, dan batch size. Epoch* adalah parameter yang menentukan berapa kali algoritma *deep learning* bekerja melewati dataset baik secara *forward* maupun *backward. Epoch* juga dapat disebut sebagai jumlah perulangan selama proses pelatihan dan berfungsi untuk memperbarui bobot pada jaringan. Satu *epoch* tercapai ketika algoritma *deep learning* telah melewati seluruh dataset sebanyak satu kali.

Proses pelatihan dalam satu *epoch* dapat berlangsung relatif lama, sehingga pembagian data ke dalam *batch* yang disebut *batch size* sangat diperlukan. *Batch size* adalah jumlah data yang diproses dalam satu perulangan. Pembagian ini bertujuan untuk mempersingkat waktu pelatihan, terutama ketika dataset yang digunakan terlalu besar dan tidak dapat dilatih sekaligus karena keterbatasan memori komputer. Oleh karena itu, data dibagi menjadi ukuran yang lebih kecil dan diproses dalam beberapa *batch*.

Jumlah *neuron* pada setiap *layer* LSTM menentukan kapasitas model untuk belajar dari data. Semakin banyak *neuron* yang digunakan, semakin kompleks model yang dapat dihasilkan. Namun, hal ini juga meningkatkan risiko *overfitting* jika tidak diimbangi dengan regulasi yang tepat. Di sisi lain, jumlah *layer* LSTM yang digunakan mempengaruhi kedalaman model dan kemampuannya untuk menangkap pola temporal yang kompleks dalam data. Meskipun demikian, penggunaan terlalu banyak *layer* dapat menyebabkan masalah *vanishing gradient* dan meningkatkan waktu pelatihan.

Modifikasi *hyperparameter* model merupakan langkah penting dalam meningkatkan performa model. *Hyperparameter* seperti jumlah *neuron* pada *layer* LSTM, jumlah *layer* LSTM, *learning rate, dropout rate,* dan lainnya perlu disesuaikan secara cermat agar dapat menghasilkan model LSTM yang optimal. Teknik-teknik seperti *Grid Search, Random Search, Bayesian Optimization*, dan *Manual Tuning* dapat digunakan untuk menemukan kombinasi *hyperparameter* yang paling efektif. Kombinasi yang tepat dari parameter-parameter ini akan memastikan model LSTM dapat berfungsi secara maksimal dan memberikan akurasi prediksi yang optimal.

2.2.4 Modifikasi LSTM Dalam Meningkatkan Akurasi Prediksi *Time Series*

Model LSTM konvensional sering kali menghadapi tantangan ketika diterapkan pada data baru, terutama dalam hal akurasi prediksi. Beberapa penelitian menunjukkan bahwa prediksi dengan model LSTM konvensional dapat menghasilkan akurasi yang kurang optimal. Contohnya, penelitian yang dilakukan oleh Jung & Kim (2020) menunjukkan bahwa penggunaan model LSTM tanpa penyesuaian *hyperparameter* yang tepat dapat menghasilkan kesalahan prediksi yang tinggi dalam peramalan harga saham. Demikian juga, penelitian yang dilakukan oleh Moghar & Hamiche (2020) menunjukkan bahwa model LSTM memiliki keterbatasan dalam memprediksi data *time series* yang sangat volatil, jika dilakukan tanpa penyesuaian dan optimasi lebih lanjut. Penyebab utama yang menghasilkan akurasi peramalan menjadi kurang bagus adalah saat penentuan jumlah layer LSTM dan *dense, neuron, epoch, batch size,* fungsi aktivasi, dan *optimizer.*

Untuk mengatasi permasalahan tersebut, beberapa modifikasi pada model LSTM dapat dilakukan, seperti penyesuaian *hyperparameter,* penambahan lapisan *dense*, dan penerapan teknik optimasi. Dalam penyesuaian *hyperparameter* model LSTM, dapat dilakukan dengan menentukan *tuning* pada jumlah unit di layer LSTM, jumlah *layer* LSTM, *learning rate, dropout rate*, dan lain-lain. Teknik seperti *Grid Search, Random Search, Bayesian Optimization,* dan *Manual Tuning* dapat digunakan untuk menemukan kombinasi *hyperparameter* yang paling efektif.

Penambahan lapisan *Dense* pada model LSTM juga dapat membantu model menangkap pola non-linear yang kompleks dalam data. Penelitian yang dilakukan Sudriyanto, dkk (2024) menunjukkan bahwa penambahan lapisan *dense* dapat meningkatkan akurasi prediksi model LSTM. Sementara itu, dalam penggunaan *optimizer Adam*, model ini dapat mempercepat proses konvergensi dan meningkatkan akurasi model. Setiawan (2022) menunjukkan bahwa Adam *optimizer,* yang menggabungkan metode momentum dan *adaptive learning rate*, sangat efektif dalam pelatihan model *deep learning*, termasuk LSTM.

Metode LSTM juga memiliki beberapa keterbatasan lain yang signifikan ketika digunakan untuk prediksi data baru. Salah satunya adalah *overfitting*, yang terjadi ketika model terlalu kompleks dengan banyak *layer* dan *neuron*. Hal ini dapat diatasi dengan menyederhanakan arsitektur model, mengurangi jumlah *layer* dan *neuron* untuk mengurangi risiko *overfitting* dan meningkatkan efisiensi model.

Selain itu, parameter *hyperparameter* yang tidak optimal juga dapat menyebabkan masalah seperti *underfitting atau overfitting*. Oleh karena itu, penting untuk melakukan penyesuaian *hyperparameter* dengan hati-hati, seperti dalam menentukan jumlah *epoch, batch size,* dan *learning rate.*

Dengan modifikasi model LSTM yang mencakup penyesuaian *hyperparameter*, penyederhanaan arsitektur, penambahan *layer dense*, dan penggunaan *optimizer* yang tepat, akurasi prediksi dapat ditingkatkan dan keterbatasan-keterbatasan dalam model LSTM dapat diatasi.

2.2.5 Fungsi Aktivasi

Fungsi aktivasi adalah fungsi yang digunakan dalam jaringan saraf untuk mengaktifkan atau menonaktifkan *neuron* serta menghitung jumlah *input* dan *bias*

yang terbobot. Fungsi ini biasanya memanipulasi data yang disajikan melalui pemrosesan *gradien*, seperti *gradient descent,* untuk menghasilkan *output* pada jaringan saraf yang berisi parameter dalam data. Fungsi aktivasi dapat berupa linier atau non-linear, tergantung pada jenis fungsi yang mempresentasikannya, dan juga digunakan untuk mengontrol *output* dari jaringan saraf. Berikut adalah beberapa fungsi aktivasi yang sering digunakan:

a. *Sigmoid Function*

Sigmoid merupakan fungsi aktivasi non-linear yang dipakai pada jaringan saraf (*Neural Network*). Fungsi *sigmoid* digunakan untuk mengubah nilai *input x* menjadi nilai antara 0 hingga 1. Dalam fungsi *sigmoid σ(x)* akan menghasilkan kurva dalam rentang 0-1 pada sumbu *y*. jika *x* adalah bilangan positif sangat besar, nilai e^{-x} akan memiliki nilai yang mendekati 0, dan menghasilkan *output* yang mendekati 1. Sedangkan jika x adalah bilangan *negative* sangat besar, nilai e^{-x} akan memiliki nilai yang besar dan menghasilkan *output* yang mendekati 0. Persamaan yang digunakan dalam menghitung *sigmoid* ditunjukkan pada persamaan 2.7 (Septiadi, dkk, 2020).

$$\sigma(x) = \frac{1}{(1+e^{-x})} \tag{2.7}$$

Keterangan:

σ(*x*) : Nilai *output* fungsi *sigmoid*

x : Nilai *input* fungsi *sigmoid*

e : Konstanta eksponensial

b. Fungsi aktivasi *tanh*

Tanh merupakan fungsi aktivasi alternatif dari *sigmoid*. Fungsi *tanh* memiliki bentuk grafik yang sama seperti fungsi *sigmoid*, namun menghasilkan nilai antara -1 dan 1. Oleh karena itu, fungsi *tanh* dapat dikatakan mirip dengan fungsi *sigmoid*, tetapi rentang nilai yang luas membuatnya lebih efektif untuk pemodelan non-linear yang kompleks. Jika *x* merupakan bilangan negatif sangat besar, maka nilai *tanh* akan mendekati -1, sementara jika nilai *x* merupakan bilangan positif sangat besar, maka nilai *tanh* mendekati 1. Bentuk dari fungsi *tanh* ini dapat dilihat dalam perhitungan dengan persamaan 2.8 (Septiadi, dkk, 2020).

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (2.8)$$

Keterangan:

tanh : Fungsi *tanh*

x : Data *input*

e : Konstanta eksponensial

2.2.6 Optimasi *Adam* (*Adaptive Moment Estimation*)

Optimasi Adam adalah algoritma optimasi berbasis stokastik orde pertama dari fungsi objektif stokastik, yang digunakan untuk memperbaharui bobot jaringan secara berulang berdasarkan data pelatihan (Chang, dkk, 2019). Algoritma ini secara dinamis memperbaharui tingkat pembelajaran untuk setiap parameter dan dianggap efisien secara komputasi dengan kebutuhan memori yang rendah.

Metode Adam menggabungkan optimasi dari RMSProp dan momentum untuk penurunan gradien (Reyad, dkk, 2023). Optimasi Adam sangat cocok digunakan secara langsung pada model apa pun yang melibatkan kumpulan data dan parameter yang besar. Tujuan dari optimasi Adam adalah untuk menghitung *adaptive learning rate* untuk setiap parameter. *Learning rate* ini menentukan seberapa besar perubahan bobot pada jaringan. Berikut ini merupakan persamaan untuk melakukan optimasi Adam dalam model LSTM, yang diilustrasikan dalam persamaan 2.9, 2.10, 2.11, 2.12, 2.13 (Reyad, dkk, 2023).

a. Menghitung gradien g_t pada waktu *t*.

b. Menghitung bias momen pertama.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (2.9)$$

c. Menghitung bias momen kedua.

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (2.10)$$

d. Memperbaiki bias momen pertama.

$$\hat{m}_t = \left(\frac{m_t}{1 - \beta_1^t}\right) \quad (2.11)$$

e. Memperbaiki bias momen kedua.

$$\hat{v}_t = \left(\frac{v_t}{1 - \beta_2^t}\right) \quad (2.12)$$

f. Memperbaiki parameter dengan *learning rate*.

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \quad (2.13)$$

Keterangan:

g_t : Gradien

m_t : Rata-rata Momen pertama

v_t : Rata-rata Momen kedua

$\hat{m}_t, \hat{v}_t$: Koreksi bias momen pertama dan kedua

β_1, β_2 : *Hyperparameter* rata-rata momen pertama dan kedua

θ_t : Parameter model

t : Iterasi saat ini

α : Nilai *learning rate*

ϵ : Nilai epsilon

2.2.7 Peramalan *Time Series*

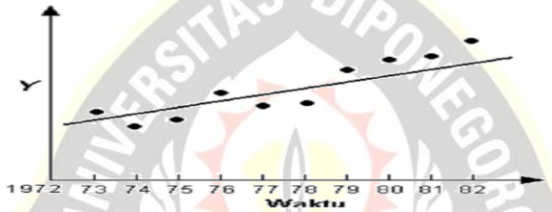
Peramalan adalah proses untuk memperkirakan kebutuhan di masa depan, yang meliputi kebutuhan dalam ukuran kuantitas, kualitas, waktu, dan lokasi, guna memenuhi permintaan barang atau jasa. Kegiatan peramalan dilakukan secara rasional atau berdasarkan analisis data terhadap suatu peristiwa, dengan meramalkan kejadian di masa mendatang menggunakan data historis dan memprediksi data di masa depan dengan model yang sistematis (Nurdini, dkk, 2022).

Peramalan atau prediksi dapat dikelompokkan berdasarkan periode, yaitu peramalan jangka pendek dengan kurun waktu kurang dari 3 bulan, peramalan jangka menengah dengan rentang waktu 4 bulan hingga 3 tahun, dan peramalan jangka panjang dengan waktu lebih dari 3 tahun. Dalam *forecasting*, terdapat metode yang sering digunakan untuk peramalan atau prediksi, yaitu metode kualitatif dan kuantitatif. Peramalan menggunakan metode kualitatif dilakukan dengan teknik *forecasting* yang tidak memerlukan data untuk melakukan peramalan, atau bersifat subjektif. Sementara itu, metode kuantitatif dilakukan secara matematis dengan membutuhkan data berupa angka atau numerik.

Forecasting time series terdiri dari beberapa langkah penting untuk menentukan metode *time series* yang baik, dengan mempertimbangkan jenis pola data yang akan digunakan. Pola data dalam peramalan atau prediksi terdiri dari 4 bagian, antara lain (Nurdini, dkk, 2022):

a. Pola *Trend*

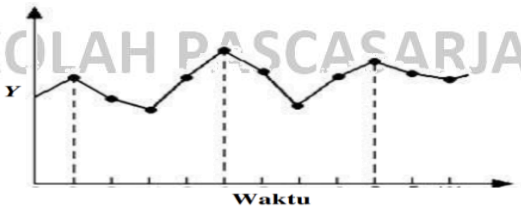
Pola *trend* adalah pola pergerakan data yang menunjukkan kecenderungan pergerakan menurun atau meningkat dalam jangka panjang. Data yang tampak tidak stabil atau fluktuatif dapat diamati dalam periode panjang, dan garis yang ditarik untuk menunjukkan pola tersebut disebut garis *trend*. Contoh pergerakan pola *trend* dalam peramalan *time series* ditunjukkan pada Gambar 2.12 (Nurdini, dkk, 2022).



Gambar 2.12 *Forecasting* Pola Trend

b. Pola *Seasonality* (Musiman)

Pola musiman terjadi apabila data tidak tetap, namun ketidaktepatan tersebut terlihat berulang dalam satu interval periode tertentu. Pola ini disebut pola *seasonality* (musiman) karena terjadi secara berulang pada kuartal tertentu, seperti hari, minggu, bulan, dan tahun. Pergerakan pola *seasonality* (musiman) dalam peramalan *time series* ini ditunjukkan pada Gambar 2.13 (Nurdini, dkk, 2022).

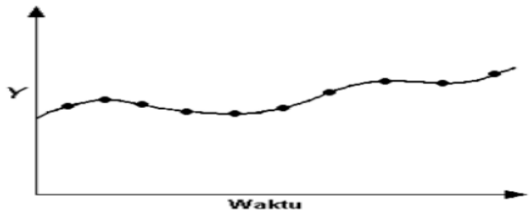


Gambar 2.13 *Forecasting* Pola *Seasonality* (Musiman)

c. Pola *Cycles*

Pola siklis terjadi karena data dipengaruhi oleh ketidaktepatan penjualan dalam periode panjang, seperti yang berhubungan dengan siklus bisnis. Penjualan produk, seperti penjualan mobil dan peralatan lainnya, termasuk dalam pola siklus.

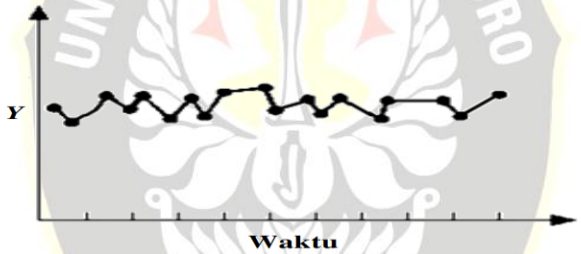
Pergerakan pola siklus dalam peramalan *time series* ditunjukkan pada Gambar 2.14 (Nurdini, dkk, 2022).



Gambar 2.14 *Forecasting* Pola *Cycles*

d. Pola Horizontal

Pola horizontal terjadi ketika nilai data berfluktuasi di sekitar rata-rata yang konstan atau stabil. Pola ini menunjukkan deret data yang bersifat stasioner terhadap rata-ratanya, tanpa adanya pergerakan signifikan berupa peningkatan atau penurunan selama periode tertentu. Pergerakan pola horizontal dalam peramalan *time series* dapat dilihat pada Gambar 2.15 (Nurdini, dkk, 2022).



Gambar 2.15 *Forecasting* Pola Horizontal

Dalam peramalan *time series* terdapat berbagai metode dalam melaksanakan peramalan. Berikut ini adalah beberapa metode *time series* yang sering digunakan, diantaranya:

a. *Weighted Moving Average*

Weighted moving average (WMA) adalah indikator teknikal yang digunakan untuk menganalisis data deret waktu, terutama dalam bidang keuangan dan perdagangan. WMA memberikan bobot yang lebih besar kepada titik data terbaru dibandingkan dengan titik data yang lebih lama. Hal ini dilakukan dengan mengalikan setiap titik data dengan bobot tertentu, kemudian menjumlahkan hasil perkalian tersebut.

Metode ini juga dikenal sebagai metode rata-rata bergerak tertimbang, yang mendahulukan manajemen data untuk menetapkan bobot (*weighted factor*) dari

data yang ada. Penentuan bobot ini bersifat subjektif, tergantung pada pengalaman dan ketentuan analis data. Rumus untuk menghitung WMA ini ditunjukkan dalam persamaan 2.14 (Fitri, dkk, 2022).

$$F_t = (W_{t-1} A_{t-1} + W_{t-2} A_{t-2} + \ldots + W_{t-n} A_{t-n}) / (W_{t-1} + W_{t-2} + \ldots + W_{t-n}) \quad (2.14)$$

Keterangan:

W_t : Bobot yang diberikan untuk periode waktu t (W = 1)

F_t : Peramalan untuk periode mendatang (periode t)

n : Jumlah periode yang dirata-ratakan

A_{t-1} : Jumlah actual periode sebelumnya hingga periode n

b. *Single Exponential Smoothing*

Single Exponential Smoothing (SES) adalah metode peramalan sederhana dan efektif yang digunakan untuk memprediksi nilai masa depan dalam suatu deret data berdasarkan pemberian bobot terhadap nilai-nilai aktual sebelumnya. Maksudnya, nilai aktual terbaru diberi bobot yang lebih besar dibandingkan nilai-nilai yang lebih lama. Metode ini pertama kali dirumuskan pada periode 1950-an oleh Brown dan Holt, yang sedang mengembangkan model peramalan untuk sistem kontrol inventaris (Seong, dkk, 2020).

Dalam metode SES, setiap data diberikan bobot yang dilambangkan dengan α (alpha). Nilai α dapat ditentukan secara bebas untuk mengurangi forecast error. Metode ini memerlukan nilai alpha (α) sebagai parameter pemulusan, di mana bobot lebih tinggi diberikan kepada data yang lebih baru. Pemilihan nilai parameter α yang tepat akan menghasilkan prediksi yang optimal dengan nilai kesalahan (error) terkecil. Nilai konstanta pemulusan harus memenuhi ketentuan $0 < \alpha < 1$ (Hudaningsih, dkk, 2020).

Metode SES ini hanya menghasilkan peramalan satu periode ke depan dan cocok untuk data yang bersifat stasioner. Jika diterapkan pada data dengan tren yang dominan atau konsisten, peramalan yang dihasilkan akan selalu tertinggal di belakang tren. Metode eksponensial ini memberikan bobot lebih tinggi pada observasi terbaru dibandingkan nilai-nilai sebelumnya. Perhitungan SES ditunjukkan dalam persamaan 2.15 (Hudaningsih, dkk, 2020).

$$F_t = F_{t-1} + \alpha (A_{t-1} - F_{t-1}) \quad (2.15)$$

Keterangan:

F_t : Nilai peramalan baru

F_{t-1} : Nilai peramalan sebelumnya

α : Konstanta penghalusan ($0< \alpha<1$)

A_{t-1} : Permintaan aktual periode sebelumnya

c. Model *Facebook Prophet*

Prophet merupakan prosedur yang paling cocok untuk meramalkan data deret waktu dengan beberapa musim data historis dan efek musiman yang kuat. Prophet adalah platform sumber terbuka milik facebook yang terinspirasi dari peramalan deret waktu yang dilakukan Meta (Costa, 2022). Metode ini digunakan untuk menyusun dan meramalkan data *time series*. *Prophet* berfokus pada model aditif, dengan pola *nonlinear* yang sesuai dengan musiman dan efek musim libur yang bersifat harian, mingguan, dan tahunan (Cheng, dkk, 2024).

Prophet menangani data yang hilang dan perubahan dalam praktik serta mampu mengelola *outlier* dengan baik. Metode ini juga membantu dalam mengakumulasikan variabel model eksogen. Model *Prophet* mengandalkan fungsi logistik untuk memodelkan tren, deret *Fourier* untuk memodelkan musiman, dan parameterisasi untuk menerapkan efek hari libur. Metode ini menggunakan model dekomposisi waktu yang unik dan fleksibel, sehingga dapat menangani data *time series* yang menunjukkan pola non-linear karena tren dan musiman. Secara umum, model *Prophet* dirumuskan sebagai penjumlahan dari tiga komponen utama, yaitu tren, musiman, dan hari libur, seperti yang diilustrasikan dalam persamaan 2.16 (Cheng, dkk, 2024).

$$y(t) = g(t) + s(t) + h(t) + \in_t \quad (2.16)$$

Keterangan:

y(t) : Prediksi pada waktu t

g(t) : *Trend (linear/logistic)*

s(t) : Periode Musiman / Perubahan berskala

h(t) : Pengaruh hari libur

$\in_t$: Istilah Kesalahan/*noise*

2.2.8 Peramalan Harga Saham

Peramalan harga saham merupakan upaya untuk memperkirakan harga masa depan saham berdasarkan analisis data historis dan berbagai faktor ekonomi lainnya. Peramalan ini sangat penting bagi investor untuk membuat keputusan investasi yang tepat dan menguntungkan (Tauzani & Douzi, 2021).

Menurut *Efficient Market Hypothesis* (EMH), tidak ada investor yang dapat *consistently outperform the market* melalui prediksi harga saham karena semua informasi yang relevan sudah tercermin dalam harga saham. EMH adalah teori yang menyatakan bahwa harga saham mencerminkan semua informasi yang tersedia (Islam & Nguyen, 2020).

Saham memiliki sentimen yang sangat fluktuatif dan dapat berubah dengan cepat. Sentimen pasar mengacu pada keseluruhan sikap investor terhadap pasar atau sekuritas tertentu. Analisis sentimen dapat melibatkan pengumpulan data dari media sosial, berita, dan laporan keuangan untuk mengukur emosi dan persepsi investor. Data ini kemudian digunakan untuk memprediksi pergerakan harga saham. Sentimen pasar sering kali diukur menggunakan teknik pembelajaran mesin yang menganalisis teks dari berbagai sumber untuk menentukan hasil sentimen tersebut positif, negatif, atau netral (Qiu, dkk, 2020).

2.2.8.1 Metode Peramalan Harga Saham

Dalam peramalan harga saham, metode prediksi sangat bervariasi, mulai dari teknik statistik tradisional hingga model pembelajaran mesin modern. Metode-metode yang digunakan untuk memprediksi harga saham meliputi analisis teknikal, analisis fundamental, metode *machine learning*, dan jaringan saraf tiruan.

a. Analisis Teknikal

Dalam analisis teknikal, pergerakan harga saham diprediksi berdasarkan data historis harga dan *volume* perdagangan. Prinsip dasar dari analisis ini adalah bahwa semua informasi yang tersedia sudah tercermin dalam harga saham, dan pola harga cenderung berulang karena psikologi pasar. Teknik yang sering digunakan dalam analisis teknikal meliputi penggunaan indikator teknikal seperti *moving average, Relative Strength Index (RSI), dan Bollinger Bands* (Shen & Shafiq, 2020).

b. Analisis Fundamental

Analisis fundamental melibatkan penilaian nilai intrinsik saham dengan menganalisis laporan keuangan perusahaan, kondisi ekonomi, dan faktor-faktor lain yang mempengaruhi kinerja perusahaan. Tujuan dari analisis ini adalah untuk menentukan nilai saham tersebut *undervalued* atau *overvalued* dibandingkan dengan harga pasar saat ini. Faktor yang dianalisis meliputi pendapatan, laba, pertumbuhan, dividen, dan kesehatan keuangan secara keseluruhan (Tauzani & Douzi, 2021).

c. Metode *Machine Learning*

Metode pembelajaran mesin telah menjadi alat yang semakin populer dalam prediksi harga saham karena kemampuannya menangkap pola kompleks dalam data. Beberapa teknik yang sering digunakan, seperti regresi *linear*, *support vector machine* (SVM), dan *random forest*. Regresi *linear* digunakan untuk memodelkan hubungan antara variabel independen dan harga saham. S*upport vector machine* (SVM), digunakan untuk klasifikasi dan regresi, serta dapat mengoptimalkan pemilihan fitur dan parameter model untuk prediksi yang lebih akurat. Sementara *random forest*, sebagai metode *ensemble* yang menggabungkan prediksi dari berbagai pohon keputusan untuk meningkatkan akurasi dan mengurangi *overfitting* (Shen & Shafiq, 2020).

d. Metode *Artificial Neural Network*

Jaringan saraf tiruan adalah model komputasi yang terdiri dari kumpulan unit pemrosesan yang disebut *neuron* buatan. *Neuron* buatan ini, mirip dengan *neuron* biologis di otak manusia yang terhubung satu sama lain membentuk jaringan. Setiap *neuron* menerima masukan, melakukan komputasi terhadap masukan tersebut, dan menghasilkan keluaran. Jaringan saraf tiruan terdiri dari banyak *neuron* yang terstruktur dalam lapisan-lapisan. Jaringan ini biasanya memiliki tiga jenis lapisan, yaitu lapisan masukan, lapisan tersembunyi, dan lapisan keluaran.

Lapisan masukan menerima data yang perlu diproses jaringan, sementara lapisan keluaran menghasilkan hasil akhir atau prediksi. Lapisan tersembunyi berada di antara lapisan masukan dan keluaran, yang melakukan pemrosesan internal untuk mempelajari pola dan hubungan dalam data. Setiap *neuron* dalam

suatu lapisan terhubung ke *neuron* lapisan berikutnya melalui koneksi berbobot, di mana bobot ini menentukan seberapa penting masukan dari *neuron* sebelumnya terhadap *neuron* berikutnya.

Jaringan saraf tiruan belajar dengan mengubah bobot berdasarkan data pelatihan yang diberikan. Dalam prediksi *time series,* jenis jaringan saraf tiruan seperti LSTM (*Long Short-Term Memory*) dan GRU (*Gated Recurrent Unit*) sering digunakan. LSTM merupakan tipe dari *Recurrent Neural Network* (RNN) yang efektif dalam menangani data sekuensial seperti harga saham. LSTM dirancang untuk mengatasi masalah *vanishing gradient* dan mampu mengingat informasi jangka panjang dan pendek (Qiu, dkk, 2020). Sementara GRU merupakan varian dari LSTM yang lebih sederhana dan lebih cepat dalam pelatihan, namun tetap mampu memberikan hasil prediksi yang baik (Tauzani & Douzi, 2021).

2.2.9 *Means Absolute Percentage Error* (MAPE)

Mengukur akurasi hasil peramalan dengan menghitung *error* merupakan langkah penting dalam menilai tingkat perbedaan antara hasil peramalan dan permintaan aktual. Beberapa metode digunakan untuk menunjukkan *error* dari suatu teknik peramalan, yang biasanya melibatkan rata-rata dari beberapa fungsi perbedaan antara nilai aktual dan nilai peramalan. Pada penelitian ini, tingkat *error* peramalan dianalisis menggunakan metode *Mean Absolute Percentage Error* (MAPE).

MAPE merupakan besaran kesalahan *relative* yang biasanya lebih bermakna dibandingkan dengan model *mean absolute deviation* (MAD) karena MAPE menyatakan persentase kesalahan hasil peramalan terhadap permintaan sebenarnya selama waktu tertentu, yang memberikan informasi persentase kesalahan terlalu tinggi atau terlalu rendah. MAPE memberikan nilai akurasi dari hasil *forecasting* dengan membandingkannya dengan hasil yang aktual. Untuk menghitung akurasi menggunakan metode ini, dilakukan rumus seperti pada persamaan 2.17 (Pangaribuan, dkk, 2023). Besarnya persentase kesalahan dari akurasi yang dihasilkan MAPE dapat dilihat dalam kriteria nilai MAPE seperti pada Tabel 2.2 (Arkadia, dkk, 2022):

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right| * 100\% \quad (2.17)$$

Keterangan:

MAPE : *Mean Absolute Percentage Error*

$\hat{Y}_i$: Nilai *Forecasting*

Y_i : Nilai aktual

n : Jumlah data yang diuji

Tabel 2.2 Kriteria Nilai MAPE

No	Kriteria	Nilai MAPE
1	Peramalan Sangat Akurat	≤10%
2	Peramalan Akurat	11-20%
3	Peramalan Cukup Akurat	21-50%
4	Peramalan Tidak Akurat	>50%