

Short Course 1

Mastering Time Series with Python

Sabtu, 24 Mei 2025
09.00 - 16.00 WIB

Hotel Bigland, Bogor



Dr. Yenni Angraini, M.Si.

Dosen Prodi Statistika dan Sains Data
Sekolah Sains Data, Matematika, dan Informatika
IPB University

Forecasting in Time Series

*Yenni Angraini
Adelia Putri Pangestika*

About Us



Dr. Yenni Angraini, S.Si., M.Si.

Pendidikan:

- S1 Statistika – IPB University
- S2 Statistika – IPB University
- S3 Statistika – IPB University

Lain-Lain:

- Sekretaris Prodi Sarjana Statistika dan Sains Data – IPB University
- Penulis Buku: Analisis Data Deret Waktu dengan Python Pendekatan Box-Jenkins dan *Machine Learning*

Kontak: y_angraini@apps.ipb.ac.id, WA: 0812-859-2300



Adelia Putri Pangestika, S.Stat.

Pendidikan:

- S1 Statistika dan Sains Data – IPB University
- On going S2 Statistika dan Sains Data – IPB University

Kontak: adelia_pangestika@apps.ipb.ac.id, WA: 0815-4006-7691

Analisis Data Deret Waktu dengan Python

Pendekatan *Box-Jenkins* dan *Machine Learning*

Buku ini memberikan kontribusi dalam memperkaya wawasan pembaca mengenai pemodelan peramalan data deret waktu dengan menggunakan metode klasik *Box-Jenkins* dan metode modern berbasis *Machine Learning*. Selain itu, pembaca akan mendapatkan pemahaman yang seimbang antara teori dan aplikasi praktis. Pendekatan ini memungkinkan pembaca tidak hanya memahami konsep dasar pemodelan peramalan data deret waktu, tetapi juga mampu mengaplikasikannya pada berbagai kasus nyata. (Prof. Dr. Hizir Sofyan, Direktur Sekolah Pascasarjana Universitas Syiah Kuala, Banda Aceh)

Buku "Analisis Data Deret Waktu dengan Python: Pendekatan *Box-Jenkins* dan *Machine Learning*" memberikan panduan komprehensif dalam membangun model prediktif untuk analisis data deret waktu. Buku ini mencakup pendekatan konvensional seperti metode *Box-Jenkins* yang fokus pada model linier (misalnya ARIMA), serta mengeksplorasi kekuatan *machine learning* untuk menangani model non-linear, termasuk penggunaan *neural network*. Kombinasi pendekatan ini memberikan perspektif menyeluruh dalam memahami dinamika deret waktu, mulai dari metode konvensional hingga teknologi modern berbasis pembelajaran mesin. Buku ini sangat direkomendasikan bagi pembaca yang ingin menguasai analisis deret waktu secara holistik. (Prof. Dr. Anang Kurnia, M.Si, Ketua Ikatan Statistisi Indonesia)

Peramalan merupakan kebutuhan yang sangat diperlukan baik untuk pengambilan kebijakan maupun dunia bisnis, sehingga dibutuhkan metode yang tepat agar memberikan daya ramal yang tinggi. Buku ini memberikan pemahaman yang sangat komprehensif tentang metode peramalan yang mencakup pendekatan konvensional dan pendekatan kekinian (*machine learning*). (Prof. Dr. Setiawan, M.S, Guru Besar Statistika Institut Teknologi Sepuluh Nopember, Surabaya)

Buku ini menghadirkan panduan lengkap untuk menganalisis data deret waktu menggunakan software Python, mulai dari konsep dasar hingga aplikasi praktis. Penjelasannya yang sistematis, dilengkapi dengan contoh kode dan studi kasus, membuatnya mudah diikuti oleh pembaca dari berbagai tingkat keahlian. Sumber referensi yang ideal bagi mahasiswa, peneliti, dan praktisi data modern. (Dr. Anna Islamiyatyi, M.Si, Ketua Departemen Statistika Universitas Hasanuddin, Makassar)



PT Penerbit IPB Press

Jalan Taman Kencana No. 3, Bogor 16128

Telp. 0251-8355 158 E-mail: ipbpress@apps.ipb.ac.id

[Facebook](#) [Twitter](#) [LinkedIn](#) Penerbit IPB Press [ipbpress.official](#) [ipbpress.com](#)



Khairil Anwar Notodiputro
Yenni Angraini
Laily Nissa Atul Mualifah



Analisis Data Deret Waktu dengan Python

Pendekatan *Box-Jenkins* dan *Machine Learning*

Agenda

Overview konsep forecasting :

- Apa itu Forecasting?
- Metode Forecasting
- Apa itu data time series?
- Komponen data time series
- Evaluasi hasil forecasting
- Tahapan umum proses forecasting

Time Series data & Metode Smoothing :

- Karakteristik data time series
- Kestasioneran Data Time Series
- Autokorelasi dan Cross-correlation
- Dekomposisi data time series
- Metode Smoothing untuk Forecasting

Pemodelan Box-Jenkins - ARIMA & SARIMA :

- Tahapan Pemodelan Box-Jenkins
- Pemodelan ARIMA
- Pemodelan SARIMA
- Pemodelan time series lainnya

Machine Learning & Deep Learning untuk Time Series:

- Pengantar ML dan DL
- DL untuk Time Series

Overview konsep forecasting

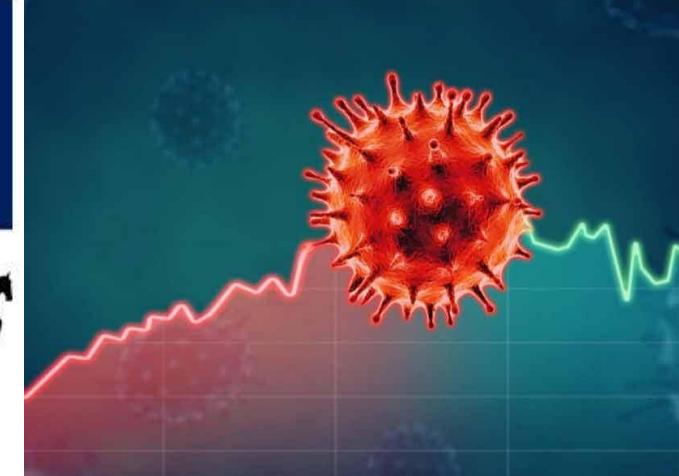
Forecasting

FUTURE



Use this data
to predict the
future

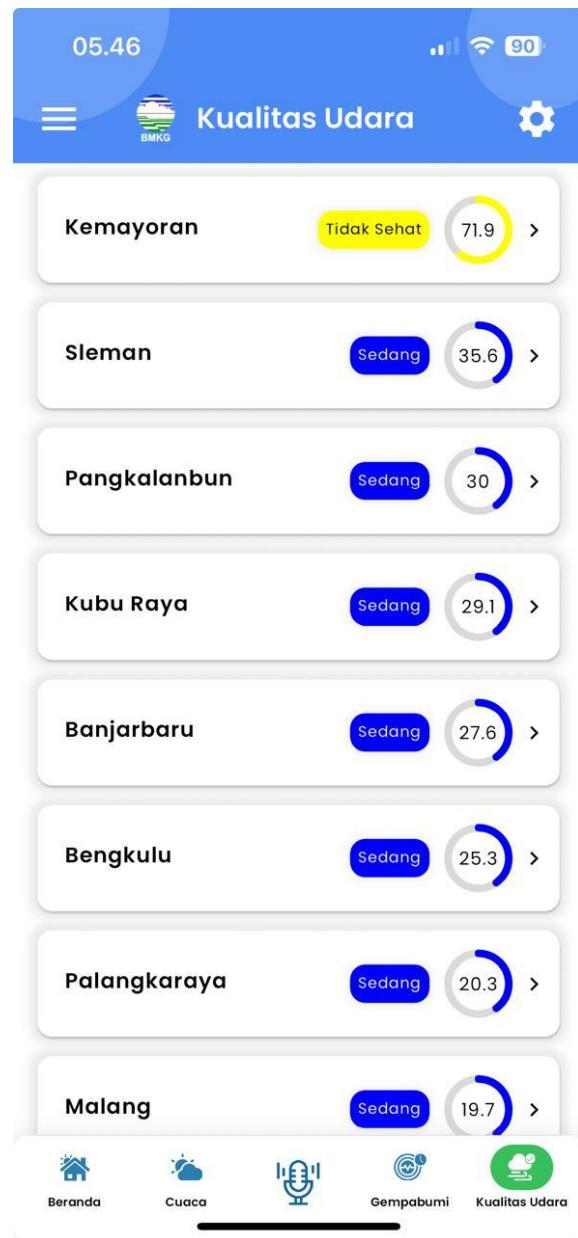
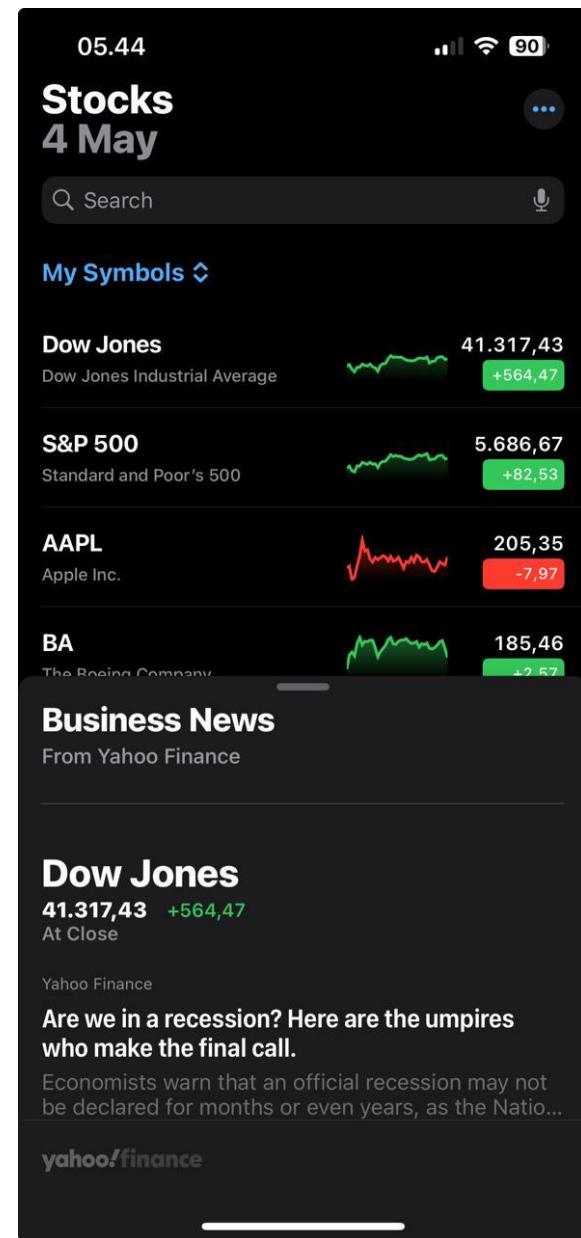
Gather & analyze
data about the
past



Suatu proses melakukan prediksi
kejadian di masa depan berdasarkan
data historis di waktu sebelumnya

Sifat utama hasil forecasting

- ✓ selalu memuat kesalahan (error)
- ✓ selalu berubah seiring dengan perubahan data yang dimiliki
- ✓ semakin jauh periode waktu peramalan maka semakin rendah keandalannya



Contoh Forecasting

Permintaan Produk di Toko Online (E-commerce)

Masalah: Sebuah toko online ingin memprediksi permintaan bulanan produk tertentu agar stok selalu tersedia tanpa overstock.

Data: Penjualan bulanan 3 tahun terakhir.

Tujuan:

Memprediksi permintaan bulan berikutnya.

Menghindari kehabisan stok (stockout) dan penumpukan barang.

Peramalan Konsumsi Listrik

Masalah: PLN ingin memprediksi beban listrik harian di wilayah tertentu untuk perencanaan kapasitas dan pemadaman bergilir.

Data: Konsumsi listrik harian selama 2 tahun.

Komponen khas: Tren naik + musiman harian dan mingguan (weekday vs weekend).

Prediksi Jumlah Wisatawan

Masalah: Dinas pariwisata ingin merencanakan promosi berdasarkan prediksi jumlah wisatawan tiap bulan.

Data: Data kunjungan wisatawan bulanan selama 7 tahun.

Karakteristik: Musiman tinggi (libur nasional, akhir tahun), tren jangka panjang.

Prediksi Curah Hujan Bulanan untuk Sektor Pertanian

Masalah: Petani dan dinas pertanian butuh prediksi curah hujan untuk penjadwalan tanam.

Data: Curah hujan bulanan selama 10 tahun.

Tujuan: Memprediksi curah hujan untuk 3 bulan ke depan di musim tanam.

Forecasting Method

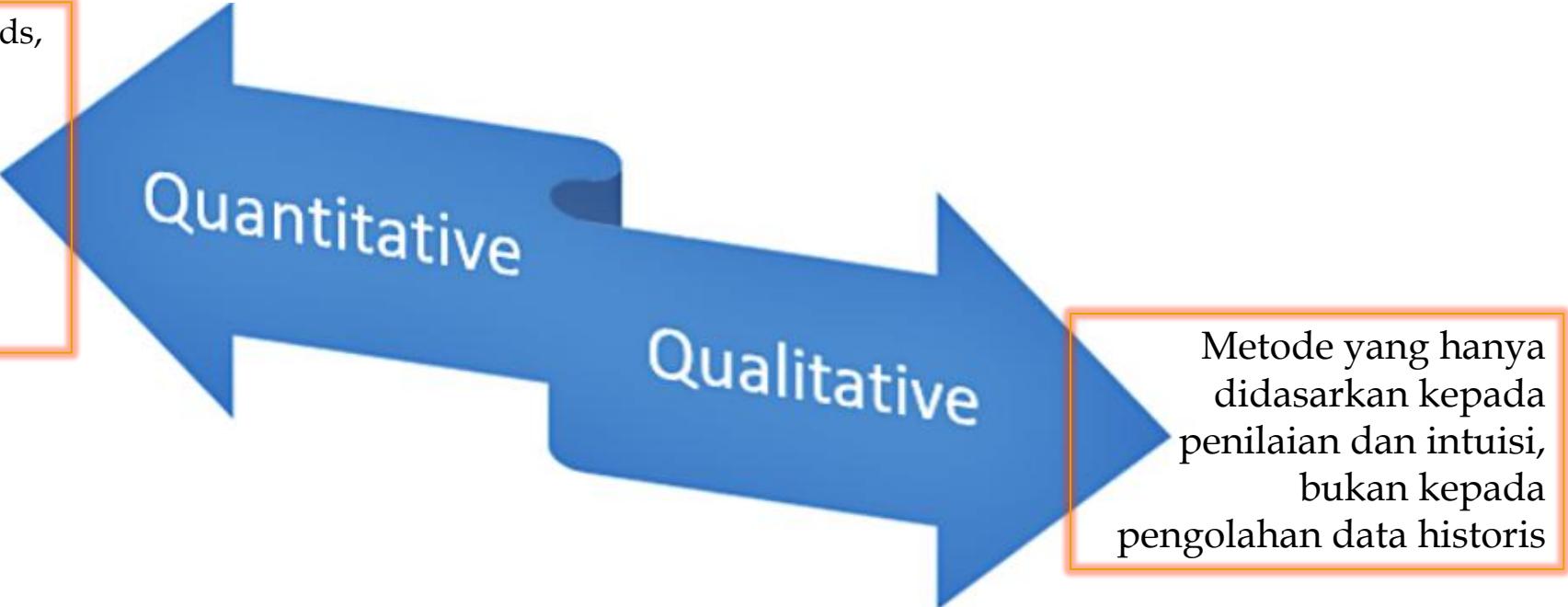
- Classical models (smoothing methods, Time series models, Regression models, etc)
- Machine Learning (Decision Tree, Random Forest, SVR, XGBoost)
- Deep Learning (MLP, CNN, LSTM, GRU, Transformer)
- Hybrid

Smoothing methods

- Moving Average
- Exponential smoothing
- Metode Winter

Time series Models

- ARMA Models
- ARIMA Models
- SARIMA Models
- ARCH-GARCH
- Intervention Models
- Transfer Function Models
- Vector autoregression (VAR)
- dll



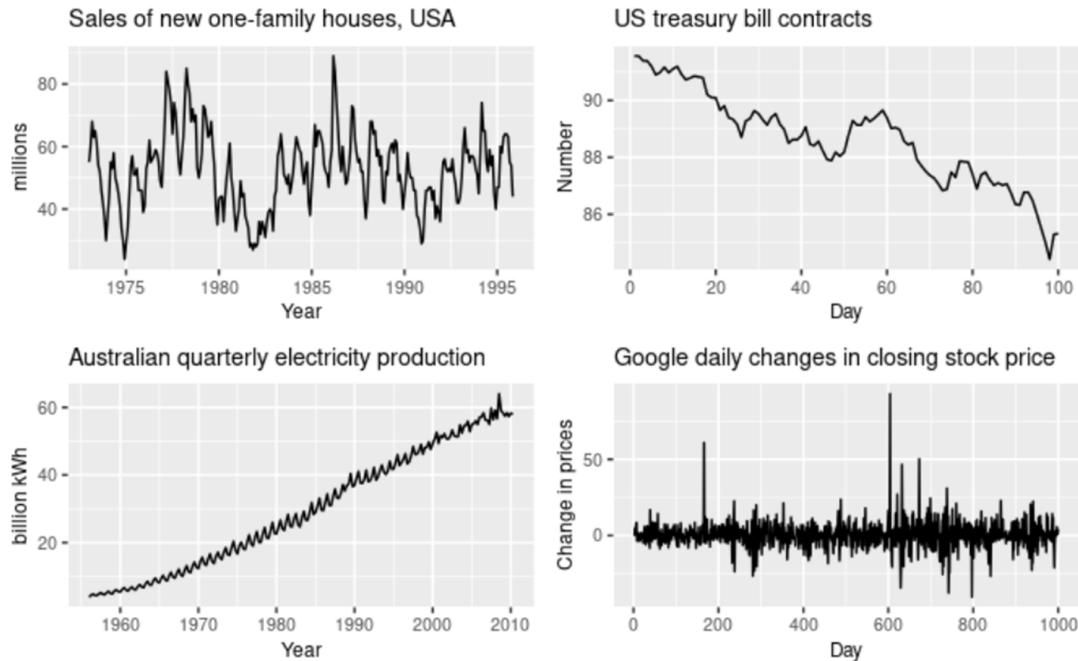
What is Time Series data ?



Analisis Data Time Series untuk Forecasting

- Data time series adalah barisan hasil pengamatan yang dilakukan berdasarkan urutan waktu dengan interval yang seragam
- Analisis forecasting mengasumsikan bahwa prediksi untuk masa depan dapat didasarkan pada perilaku data di masa lampau
 - Model statistika dan machine learning/deep learning
- Analisis time series sering diawali dengan melakukan plot data dan secara visual mengamati perilaku perubahan dari waktu ke waktu
- Satu atau lebih pola dapat saja terlihat secara visual seperti:
 - Trend
 - Variasi musiman (seasonal)
 - Siklus

Komponen data time series



- Trend: perubahan jangka panjang naik atau turun pada data. Pergeseran populasi, perubahan pendapatan, ataupun perubahan budaya sangat mungkin relevan dengan kejadian trend.
- Seasonality: variasi regular jangka pendek yang berulang yang umumnya terkait dengan kalender dan perulangan periode waktu tertentu.
- Cycles: variasi seperti gelombang yang terjadi dalam jangka waktu yang panjang (lebih dari satu tahun). Dapat terjadi karena adanya efek politik, efek kebijakan ekonomi, dan lain-lain.
- Variasi tak teratur: dapat terjadi akibat kejadian-kejadian luarbiasa seperti cuaca ekstrim, pandemi dll. Dalam beberapa hal, data seperti ini sering dikeluarkan dari analisis
- Variasi acak: variasi yang tidak dapat dijelaskan dan dipandang sebagai dampak dari hal-hal lain yang tidak dimasukkan dalam analisis

Evaluasi hasil forecasting

Beberapa ukuran yang dapat dipakai untuk penilaian seberapa baik metode mengepas data:

- Mean Absolute Error (MAE)

$$MAE = \frac{1}{n} \sum_{t=1}^n |y_t - \hat{y}_t|$$

- Mean Squared Error (MSE)

$$MSE = \frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2$$

- Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{1}{n} \sum_{t=1}^n \left| \frac{y_t - \hat{y}_t}{y_t} \right|$$

Mengapa perlu evaluasi hasil peramalan?

- Mengetahui **akurasi** metode yang digunakan.
- Membandingkan **beberapa metode** forecasting.
- Menentukan model terbaik untuk **pengambilan keputusan**

Tahapan umum forecasting

- Menentukan tujuan peramalan
- Menentukan time horizon, selalu ingat bahwa semakin panjang maka semakin rendah akurasinya
- Kumpulkan data, lakukan cleaning dan analisis yang sesuai. Proses ini seringkali melelahkan dan membutuhkan effort yang cukup besar.
- Pilih metode forecast yang sesuai.
- Lakukan forecast
- Lakukan pemantauan terhadap hasil forecasting untuk melihat kinerjanya

Pemodelan Forecasting untuk Variabel Y

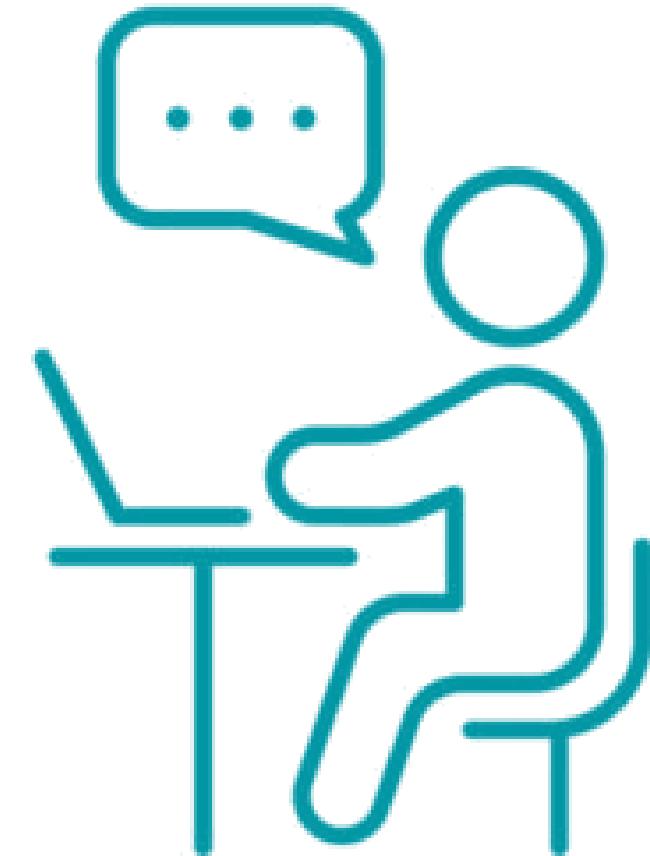
- Tipe 1: Bekerja hanya dengan variabel Y
 - Metode Smoothing
 - Box-Jenkins
- Tipe 2: Memanfaatkan variabel lain (X)
 - Model Fungsi Transfer
 - ARDL (Autoregressive Distributed Lag)
 - VAR
 - dll

Hands-On Session #1

- Belajar membuat plot time series dg Python
- Mengenal pola data time series



<https://github.com/yenniangraini/pelatihanTS/>

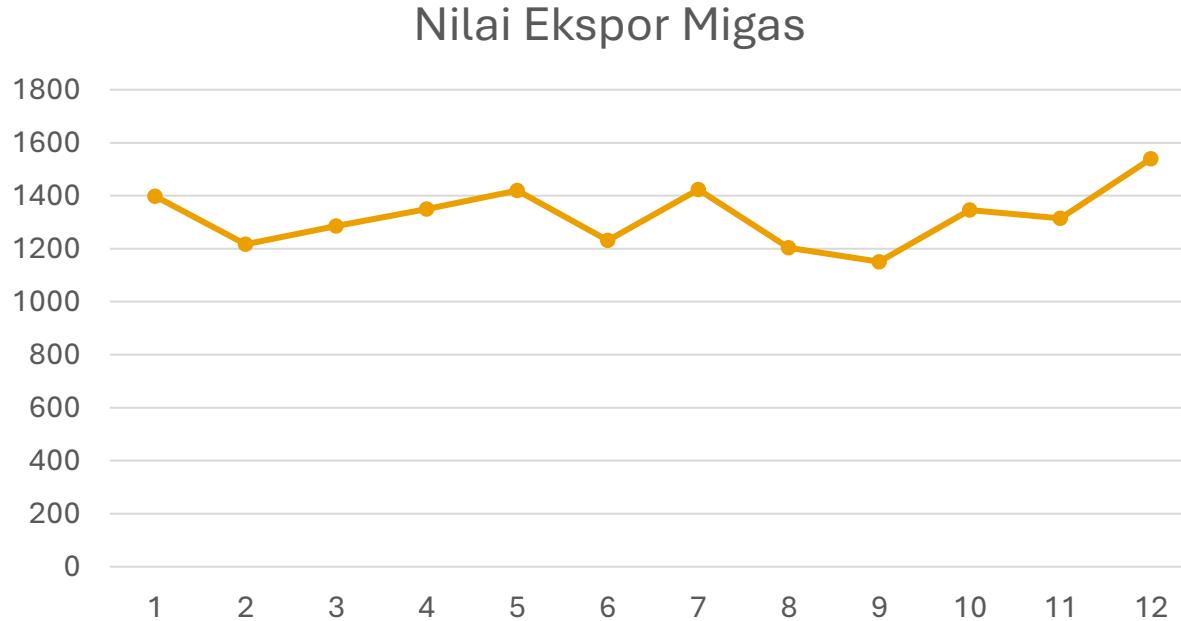


Time Series data & Metode Smoothing

Time Series Data

Bulan	t	Nilai Ekspor Migas Y_t
Januari	1	1397.6
Februari	2	1216.9
Maret	3	1285.2
April	4	1350
Mei	5	1419.1
Juni	6	1231.2
Juli	7	1422.9
Agustus	8	1203.6
September	9	1150.9
Oktober	10	1345.4
November	11	1314.4
Desember	12	1539.4

- Y_t = nilai variabel pada waktu ke-t
- Umumnya divisualisasikan dalam bentuk time series plot
 - Perubahan/tren dari waktu ke waktu
 - Keberadaan siklus atau musiman
 - Fluktiasi/volatilitas



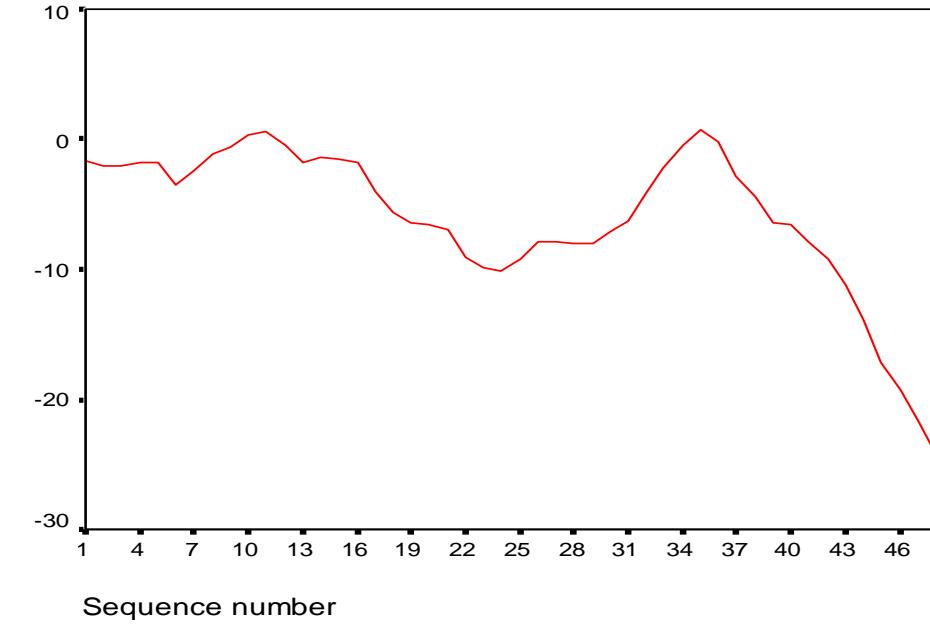
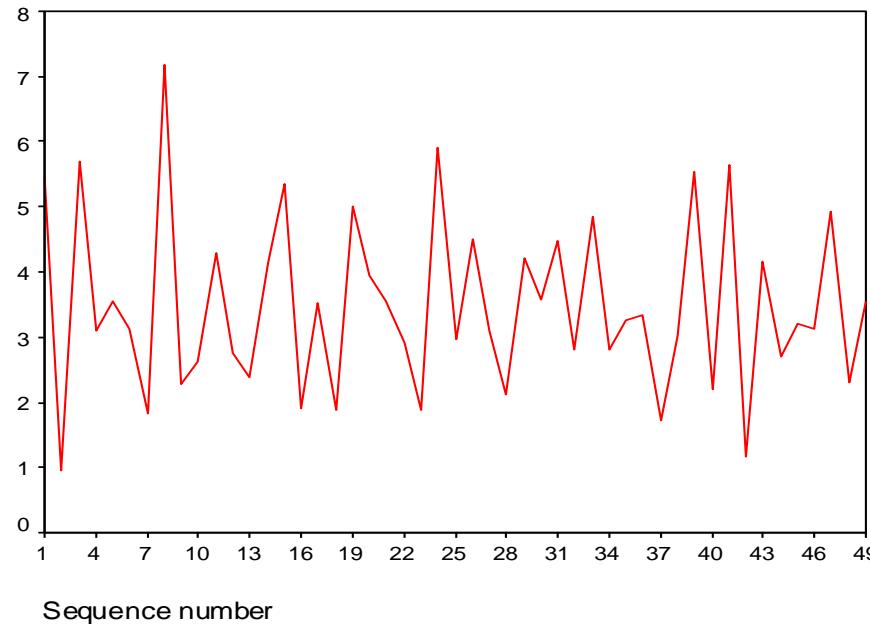
Variabel Beda Kala (lag-time variable)

- Nilai variabel pada waktu sebelumnya
- Y_{t-1} adalah nilai variabel pada satu satuan/langkah waktu sebelumnya
- Y_{t-2} adalah nilai variabel pada dua satuan/langkah waktu sebelumnya

Bulan	t	Nilai Ekspor Migas Y_t	Y_{t-1}	Y_{t-2}
Januari	1	1397.6		
Februari	2	1216.9	1397.6	
Maret	3	1285.2	1216.9	1397.6
April	4	1350	1285.2	1216.9
Mei	5	1419.1	1350	1285.2
Juni	6	1231.2	1419.1	1350
Juli	7	1422.9	1231.2	1419.1
Agustus	8	1203.6	1422.9	1231.2
September	9	1150.9	1203.6	1422.9
Oktober	10	1345.4	1150.9	1203.6
November	11	1314.4	1345.4	1150.9
Desember	12	1539.4	1314.4	1345.4

Karakteristik data time series

- Secara garis besar, data deret waktu dibedakan menjadi dua:
 - stasioner dan
 - tidak stasioner
- Dikatakan stasioner apabila data deret waktu memiliki nilai tengah (rataan) dan ragam (fluktuasi) yang konstan dari waktu ke waktu



Konsep Kestasioneran Data

- Suatu data deret waktu Y_t dikatakan stasioner jika sebaran peluangnya tidak bergantung pada waktu t
- Definisi yang lebih relax:
 - $E(Y_t) = \mu \rightarrow$ nilai harapan konstan
 - $Var(Y_t) = \sigma^2 \rightarrow$ ragam konstan/homogen
 - $cov(Y_t, Y_{t+h}) = cov(Y_s, Y_{s+h}) \rightarrow$ korelasi antar waktu hanya bergantung pada jarak/selisih waktu
- Uji kestasioneran data: ADF test (Augmented Dickey-Fuller test)

Augmented Dickey-Fuller Test

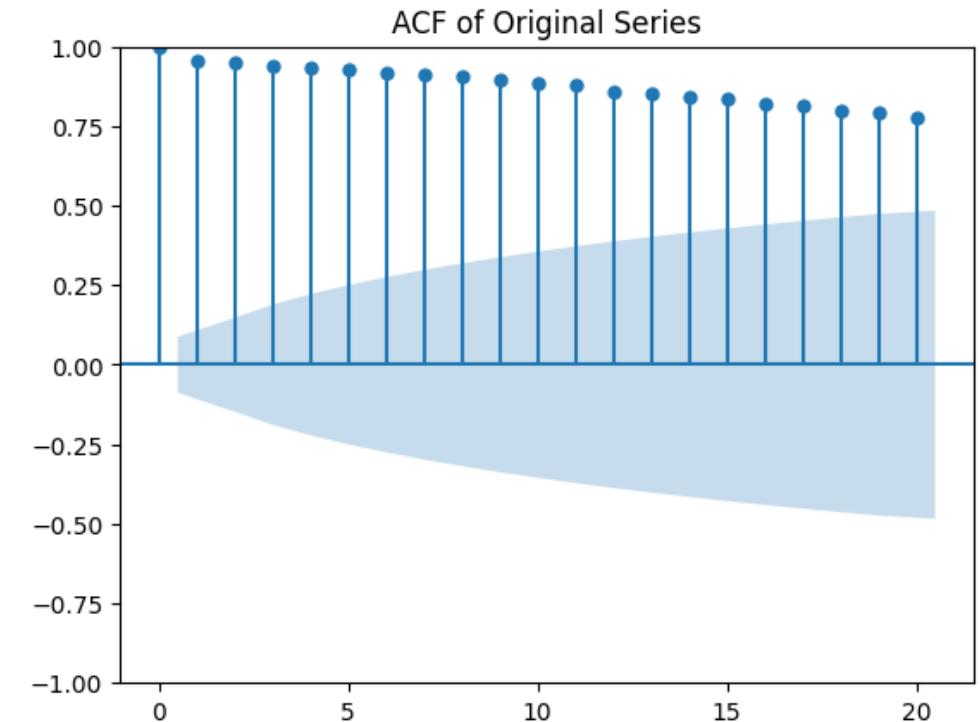
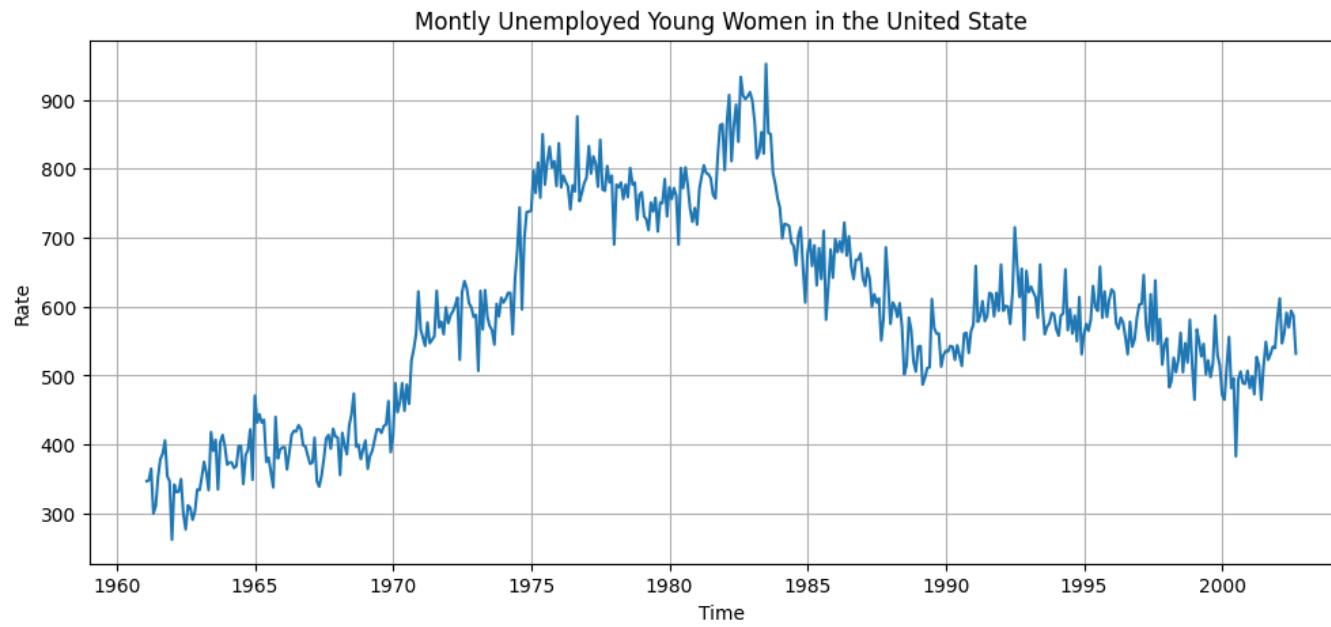
$$y_t = c + \beta t + \alpha y_{t-1} + \phi_1 \Delta Y_{t-1} + \phi_2 \Delta Y_{t-2} \dots + \phi_p \Delta Y_{t-p} + e_t$$

$H_0: \alpha = 1$ → series tidak stasioner

$H_1: \alpha < 1$ → series stasioner

- P-value besar pada hasil tes mengindikasikan H_0 diterima dan series dinyatakan tidak stasioner
- Treatment terhadap data yang tak stasioner → differencing: $y_t - y_{t-1}$

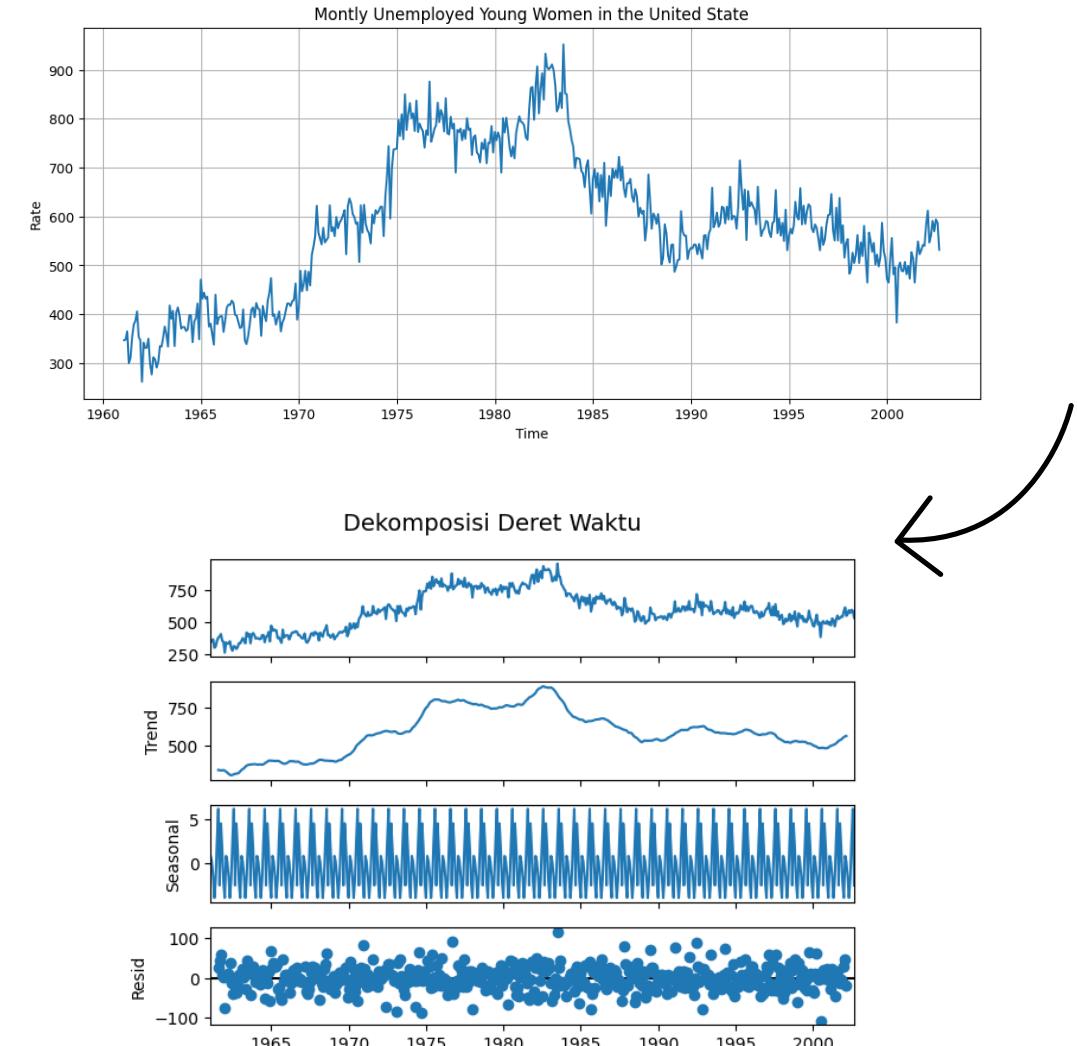
Pemeriksaan Kestasioneran Data



ADF Statistic: -2.044
p-value: 0.268

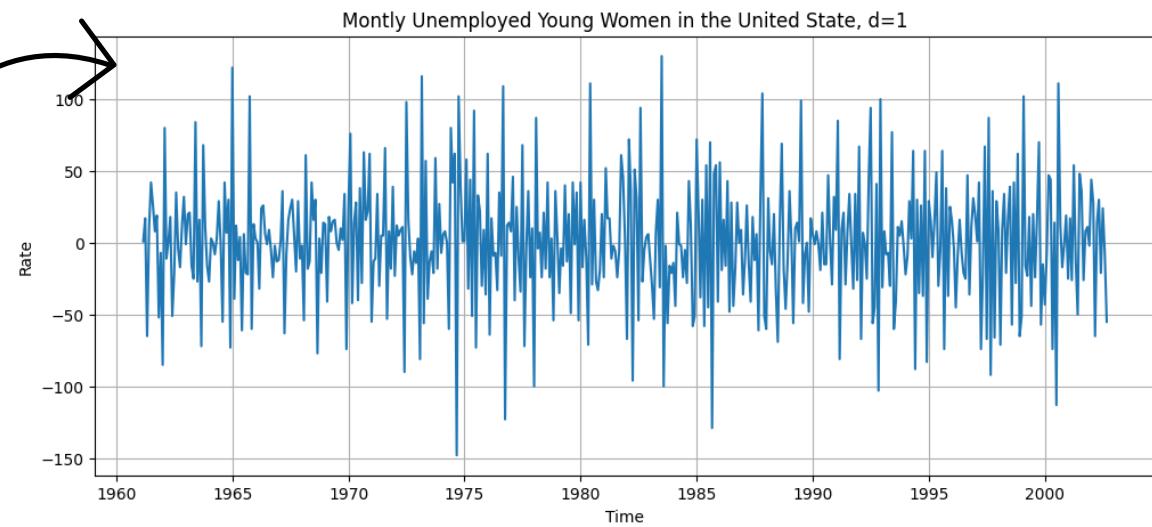
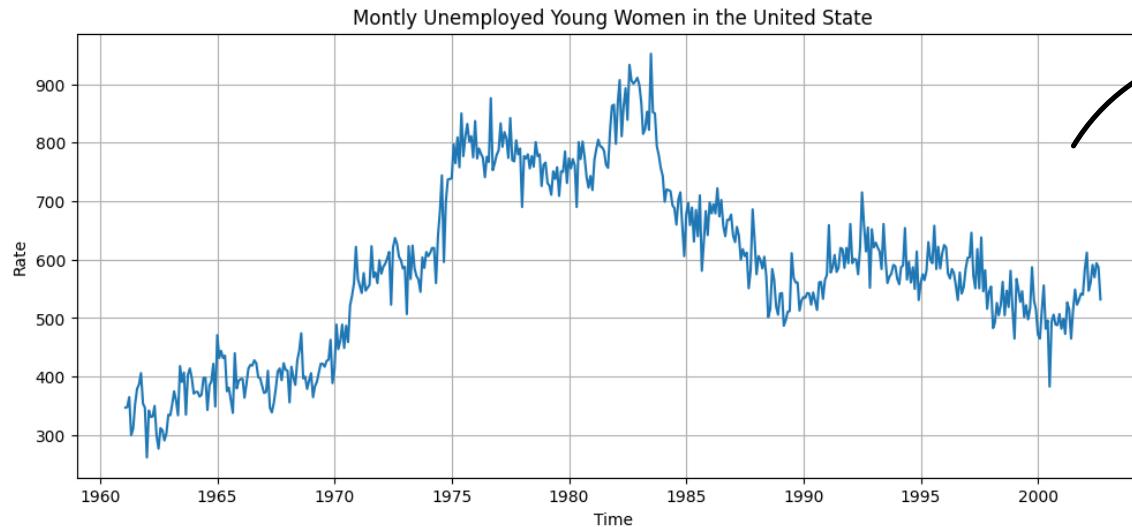
Dekomposisi Data Time Series

- Data pada umumnya tersusun dari beberapa komponen: tren, musiman, noise
- Dekomposisi bertujuan untuk menguraikan data menjadi 3 komponen
- Berguna untuk menentukan model apa yang sesuai untuk memodelkan data



Penanganan Ketakstasioneran Data

$$\text{Differencing} = Y_t - Y_{t-1}$$

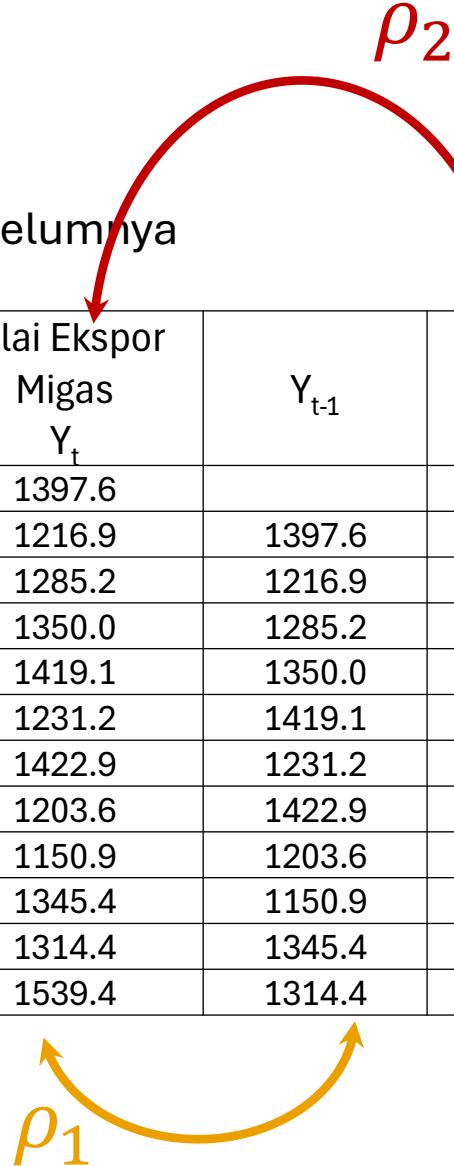


Autocorrelation

- Korelasi antara suatu variabel dengan variabel yang sama pada waktu sebelumnya

$$\rho_1 = \text{cor}(Y_t, Y_{t-1})$$

$$\rho_2 = \text{cor}(Y_t, Y_{t-2})$$



The diagram illustrates autocorrelation with red and yellow curved arrows. A red arrow points from the value at time t to the value at time $t-2$, labeled ρ_2 . A yellow arrow points from the value at time t to the value at time $t-1$, labeled ρ_1 .

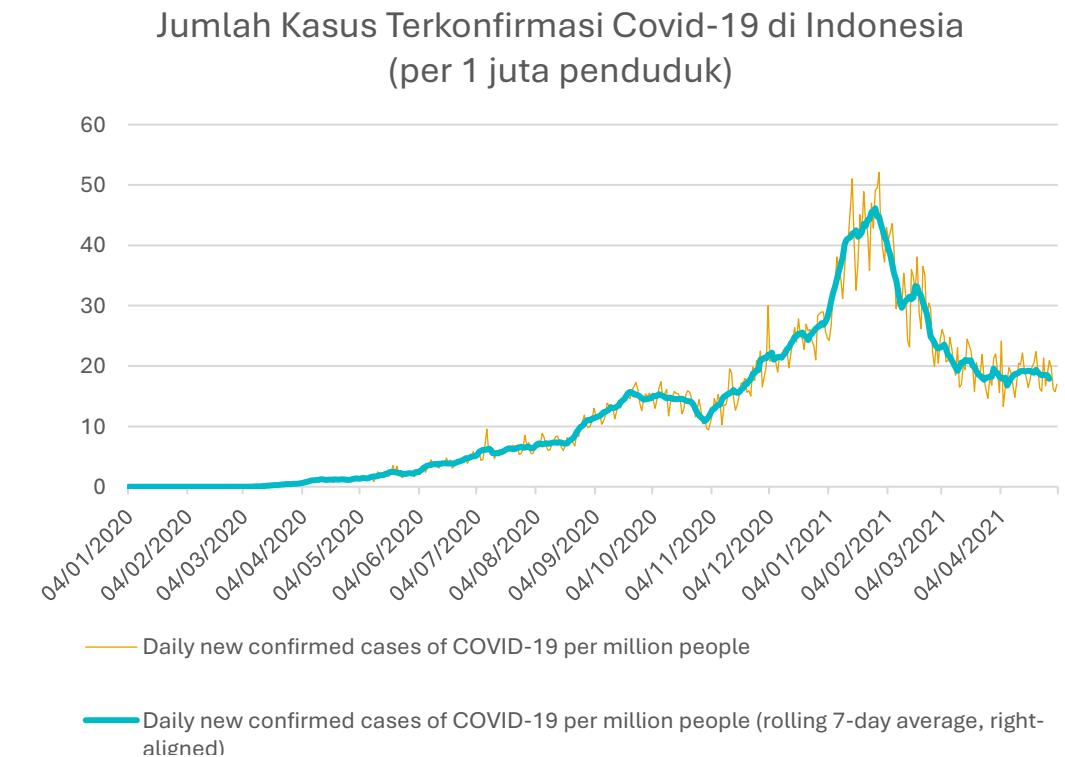
Bulan	t	Nilai Ekspor Migas Y_t	Y_{t-1}	Y_{t-2}
Januari	1	1397.6		
Februari	2	1216.9	1397.6	
Maret	3	1285.2	1216.9	1397.6
April	4	1350.0	1285.2	1216.9
Mei	5	1419.1	1350.0	1285.2
Juni	6	1231.2	1419.1	1350.0
Juli	7	1422.9	1231.2	1419.1
Agustus	8	1203.6	1422.9	1231.2
September	9	1150.9	1203.6	1422.9
Oktober	10	1345.4	1150.9	1203.6
November	11	1314.4	1345.4	1150.9
Desember	12	1539.4	1314.4	1345.4

Cross Correlation

- Korelasi antar dua variabel (data time series) yang berbeda waktu
- Y_t dengan X_{t-1}
- Y_t dengan X_{t-2}

Metode Smoothing untuk Forecasting

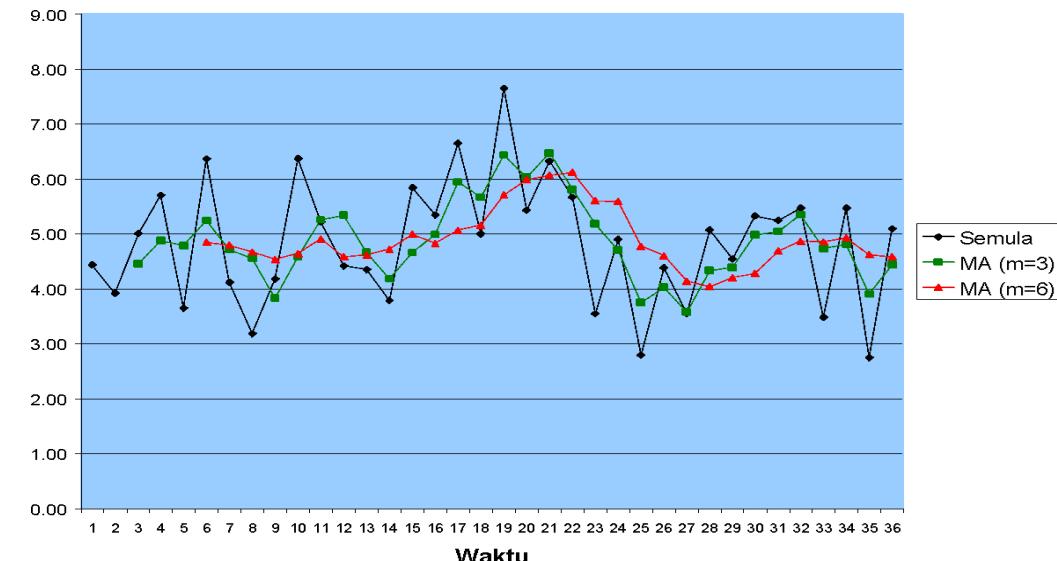
- Prinsip dasar: pengenalan pola data dengan menghaluskan/memuluskan variasi lokal.
- Prinsip pemulusan umumnya berupa rata-rata.
- Beberapa metode pemulusan hanya cocok untuk pola data tertentu.
- Nilai hasil pemulusan digunakan sebagai nilai peramalan



Single Moving Average

- Ide: data pada suatu periode dipengaruhi oleh data beberapa periode sebelumnya
- Cocok untuk pola data konstan/stasioner
- Prinsip dasar:
 - Data *smoothing* pada periode ke- t merupakan rata-rata dari m buah data dari data periode ke- t hingga ke- $(t-m+1)$
 - Data *smoothing* pada periode ke- t berperan sebagai nilai *forecasting* pada periode ke- $t+1$
 - $F_t = S_{t-1}$ dan $F_{n,h} = S_n$

Periode (t)	Data (X_t)	Smoothing (S_t)	Forecasting (F_t)
1	5	-	-
2	7	-	-
3	6	6	-
4	4	5.6	6
5	5	5	5.6
6	6	5	5
7	8	6.3	5
8	7	7	6.3
9	8	7.6	7
10	7	7.3	7.6
11			7.3
12			7.3



Double Moving Average

- Mirip dengan *single moving average*
- Cocok untuk data yang berpola tren
- Proses penghalusan dengan rata-rata dilakukan dua kali

- Tahap I: $S_{1,t} = \frac{1}{m} \sum_{i=t-m+1}^t X_i$

- Tahap II: $S_{2,t} = \frac{1}{m} \sum_{i=t-m+1}^t S_{1,i}$

- Forecasting dilakukan dengan formula

$$F_{2,t,t+h} = A_t + B_t(h)$$

dengan

$$A_t = 2S_{1,t} - S_{2,t}$$

$$B_t = \frac{2}{m-1} (S_{1,t} - S_{2,t})$$

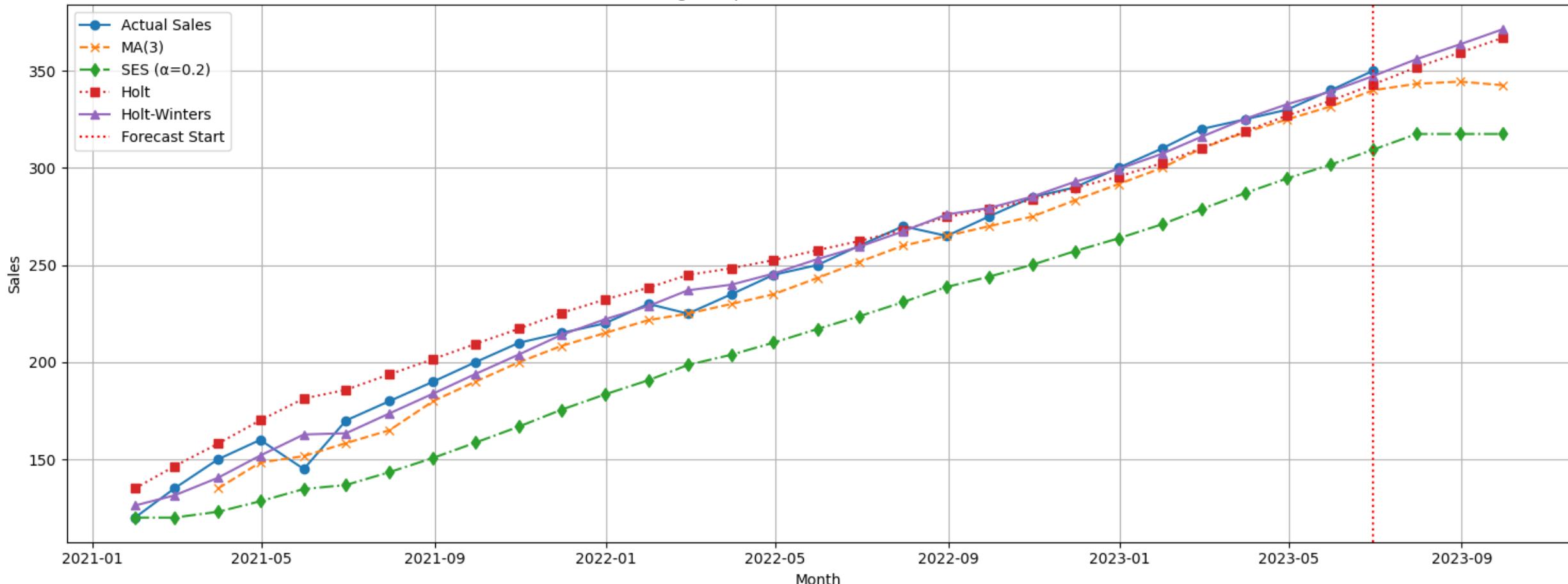
Ilustrasi DMA dengan $m=3$

t	X_t	$S_{1,t}$	$S_{2,t}$	A_t	B_t	$F_{2,t}$
1	12.50					
2	11.80					
3	12.85	12.38				
4	13.95	12.87				
5	13.30	13.37	12.87	13.87	0.50	
6	13.95	13.73	13.32	14.14	0.41	14.37
7	15.00	14.08	13.73	14.43	0.35	14.55
8	16.20	15.05	14.29	15.81	0.76	14.78
9	16.10	15.77	14.97	16.57	0.80	16.57
10						17.37
11						18.17
12						18.97

Metode Smoothing Lain

- Single Exponential Smoothing – cocok untuk data berpola konstan
- Double Exponential Smoothing – cocok untuk data berpola tren
- Holt-Winters Exponential Smoothing – cocok untuk data berpola musiman

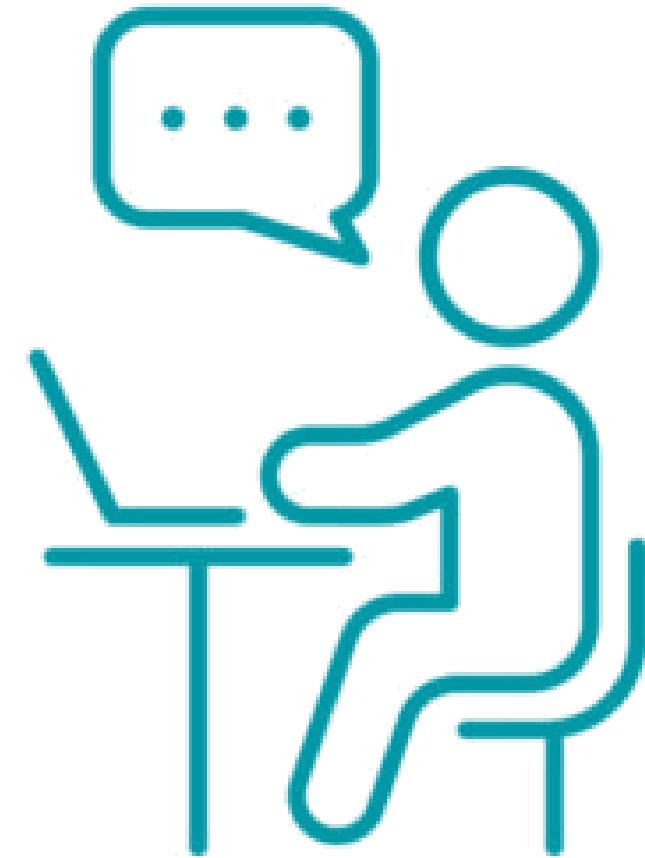
Forecasting Comparison: MA, SES, Holt, Holt-Winters



	MA(3)	SES (0.2)	Holt	Holt-Winters
MAPE	3.66%	14.54%	4.40%	2.26%
MAE	8.21	34.73	9.09	4.52
MSE	79.37	1248.04	129.71	37.37

Hands-On Session #2

- Stasioner data
- Smoothing



Pemodelan Box-Jenkins - ARIMA & SARIMA

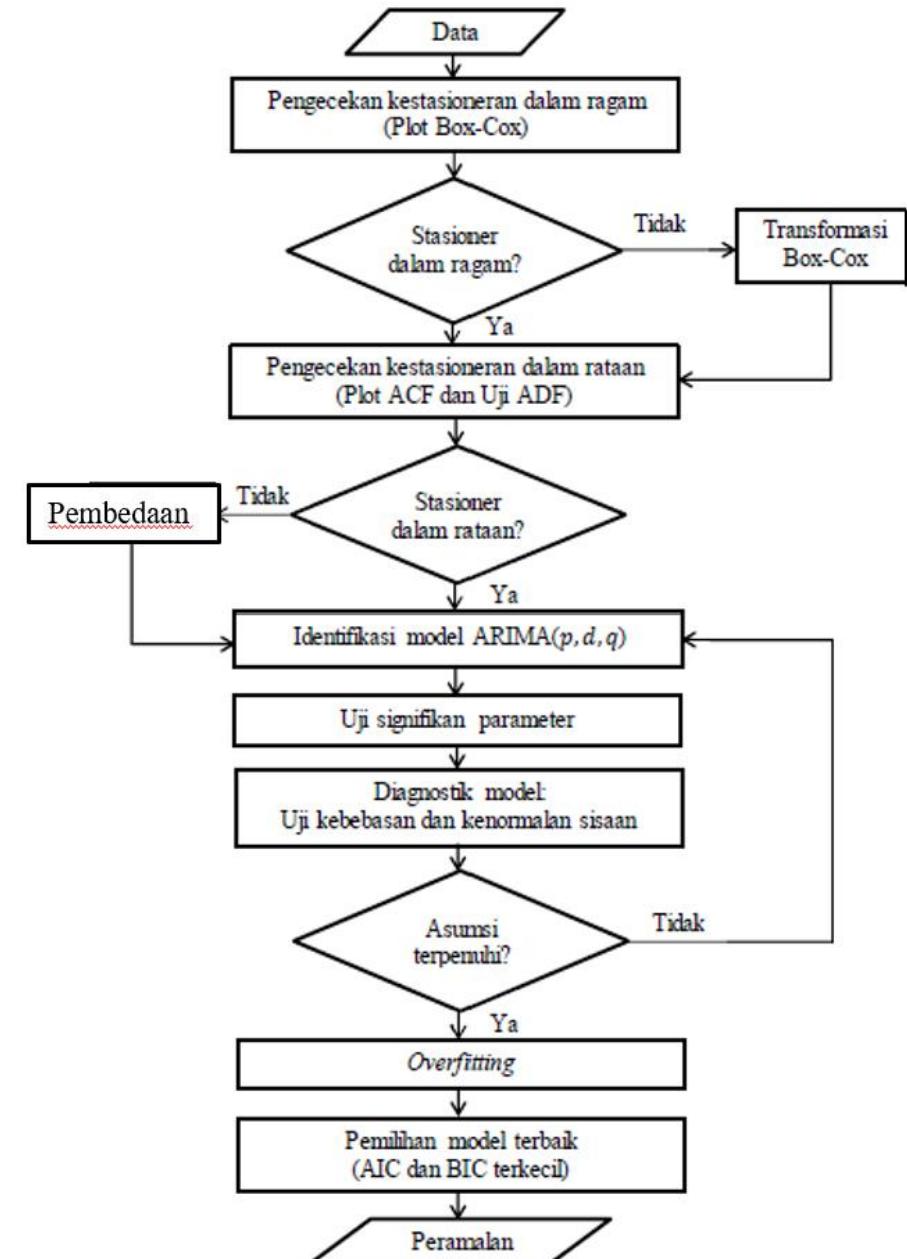
Beberapa Istilah dalam pemodelan time series

- Proses Stokastik (Stochastic Processes)
- Stasioneritas (Stationarity), $E(Y_t) = E(Y_{t-k})$ dan $Var(Y_t) = Var(Y_{t-k})$
- Koragam diri (Autocovariance), $\gamma_k = Cov(Y_t, Y_{t-k})$
- Korelasi diri (Autocorrelation), $\rho_k = Corr(Y_t, Y_{t-k}) = \frac{\gamma_k}{\gamma_0}$

$$\left. \begin{array}{ll} \gamma_0 = Var(Y_t) & \rho_0 = 1 \\ \gamma_k = \gamma_{-k} & \rho_k = \rho_{-k} \\ |\gamma_k| \leq \gamma_0 & |\rho_k| \leq 1 \end{array} \right\}$$

Tahapan Pemodelan Box-Jenkins

- Pengecekan kestasioneran data (plot time series, ACF, Uji ADF)
- Identifikasi Model (ACF, PACF dan EACF)
- Pendugaan Parameter
- Diagnostik model (Analisis sisaan dan *Overfitting*)
- Pemilihan Model terbaik
- **Forecasting**



Model Box-Jenkins

- **Model Autoregressive (AR)**
Berdasarkan data masa lalu
- **Model Moving Average (MA)**
Berdasarkan deviasi acak saat ini dan data masa lalu
- **Model campuran antara AR dan MA (ARMA)**
- **Model ARIMA**
- **Model SARIMA**

Model Box-Jenkins

Model Autoregressive (AR)

- Model AR mempunyai ordo yang besarnya dinotasikan dengan huruf “p”, sehingga dinotasikan dengan AR(p)
- Model AR mengasumsikan tiap observasi dibentuk oleh rata-rata tertimbang pengamatan masa lalu, p periode ke belakang dan deviasi periode sekarang
- Penulisan model :

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + e_t$$

Model Moving Average (MA)

- Model MA mempunyai ordo yang besarnya dinotasikan dengan huruf “q”, sehingga dinotasikan dengan MA(q)
- Model MA mengasumsikan tiap observasi dibentuk oleh rata-rata q periode ke belakang
- Penulisan Model :

$$Y_t = e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} - \cdots - \theta_q e_{t-q}$$

Model Box-Jenkins

Model ARMA

- Adakalanya proses acak yang stasioner mempunyai dua karakteristik AR dan MA
- Proses acak seperti ini perlu didekati dengan model campuran antara autoregressive dan moving average → disebut ARMA(p,q)
- Penulisan model :

$$Y_t = \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + e_t - \theta_1 e_{t-1} - \cdots - \theta_q e_{t-q}$$

$$\phi_p(B)Y_t = \theta_q(B)e_t$$

$$\phi_p(B) = 1 - \phi_1 B - \cdots - \phi_p B^p, \theta_q(B) = 1 - \theta_1 B - \cdots - \theta_q B^q$$

Model ARIMA

- Model ARIMA (Autoregressive Integrated Moving Average) merupakan pengembangan dari model ARMA yang digunakan untuk menganalisis data deret waktu yang **tidak stasioner**.
- Ketidakstasioneran ini biasanya diatasi dengan melakukan proses differencing (pendiferensialan) terhadap data, sehingga data menjadi stasioner.
- Penulisan Model :

$$\phi_p(B)(1 - B)^d Y_t = \theta_q(B)e_t$$

$$\phi_p(B) = 1 - \phi_1 B - \cdots - \phi_p B^p, \theta_q(B) = 1 - \theta_1 B - \cdots - \theta_q B^q$$

Model Box-Jenkins

Seasonal ARIMA

- Model **Seasonal ARIMA (SARIMA)** atau disebut juga **ARIMA musiman**, merupakan perluasan dari model ARIMA yang digunakan untuk menganalisis data deret waktu **yang menunjukkan pola musiman (seasonality)**.
- SARIMA digunakan ketika data menunjukkan **pola berulang pada interval waktu tertentu**, dan menggabungkan komponen musiman dan non-musiman dalam satu model.

$$\Phi_P(B^s)\phi_p(B)(1 - B)^d(1 - B^s)^D Y_t = \theta_q(B)\Theta_Q(B^s)e_t$$

$$\phi_p(B) = 1 - \phi_1 B - \dots - \phi_p B^p, \theta_q(B) = 1 - \theta_1 B - \dots - \theta_q B^q,$$

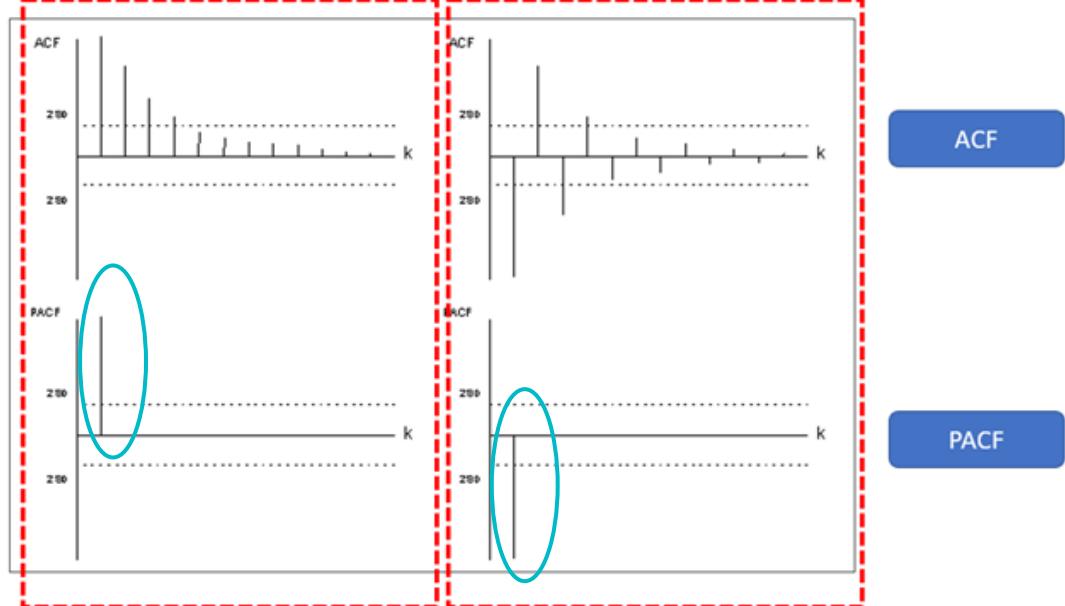
$$\Phi_P(B^s) = 1 - \Phi_1 B^s - \Phi_2 B^{2s} - \dots - \Phi_P B^{Ps}, \Theta_Q(B^s) = 1 - \Theta_1 B^s - \Theta_2 B^{2s} - \dots - \Theta_Q B^{Qs}$$

$\emptyset_p(B)$: $(1 - \emptyset_1 B - \dots - \emptyset_p B^p)$, operator AR(p)
$\theta_q(B)$: $(1 - \theta_1 B - \dots - \theta_q B^q)$, operator MA(q)
$\Phi_P(B^s)$: $(1 - \Phi_1 B^s - \dots - \Phi_P B^{Ps})$, operator Seasonal AR(P)
$\Theta_Q(B^s)$: $(1 - \Theta_1 B^s - \dots - \Theta_Q B^{Qs})$, operator Seasonal MA(Q)
$(1 - B)^d$: pembedaan nonmusiman
$(1 - B^s)^D$: pembedaan musiman
Y_t	: data pada waktu ke- t
(p, d, q)	: ordo bagian nonmusiman dari model
(P, D, Q)	: ordo bagian musiman dari model
s	: jumlah periode per musim
e_t	: sisaan pada waktu ke- t
B	: operator penggeser mundur (<i>backshift</i>)

Identifikasi Model Box-Jenkins

No	Kemungkinan Plot ACF dan PACF	Model Tentatif ARMA
1.	ACF nyata pada lag ke-1,2,...,q dan terpotong pada lag-q (<i>cuts off</i>) PACF menurun cepat membentuk pola eksponensial atau sinus (<i>dies down</i>)	MA(q)
2.	ACF <i>dies down</i> PACF nyata pada lag -p dan <i>cuts off</i> setelah lag ke-p	AR(p)
3.	ACF nyata pada lag -q dan <i>cuts off</i> setelah lag ke-q PACF nyata pada lag -p dan <i>cuts off</i> setelah lag ke-p	MA(q) jika ACF <i>cuts off</i> lebih tajam, AR(p) jika PACF <i>cuts off</i> lebih tajam
4.	Tidak ada autokorelasi yang nyata pada plot ACF dan PACF	ARMA(0,0)
5.	ACF <i>dies down</i> PACF <i>dies down</i>	ARMA(p,q)
6.	ACF nyata pada lag ke-S,2S,...,QS dan <i>cuts off</i> setelah lag-QS PACF <i>dies down</i>	MA(Q)
7.	ACF <i>dies down</i> PACF nyata pada lag -S, 2S,...PS dan <i>cuts off</i> setelah lag ke-PS	AR(P)
8.	ACF nyata pada lag ke-S,2S,...,QS dan <i>cuts off</i> setelah lag-QS PACF nyata pada lag -S, 2S,...PS dan <i>cuts off</i> setelah lag ke-PS	MA(q) jika ACF <i>cuts off</i> lebih tajam, AR(p) jika PACF <i>cuts off</i> lebih tajam
9.	Tidak ada autokorelasi yang nyata pada level musiman dalam plot ACF dan PACF	ARMA(0,0)
10.	ACF <i>dies down</i> pada level musiman PACF <i>dies down</i> pada level musiman	ARMA(P,Q)

ACF and PACF for AR (1)

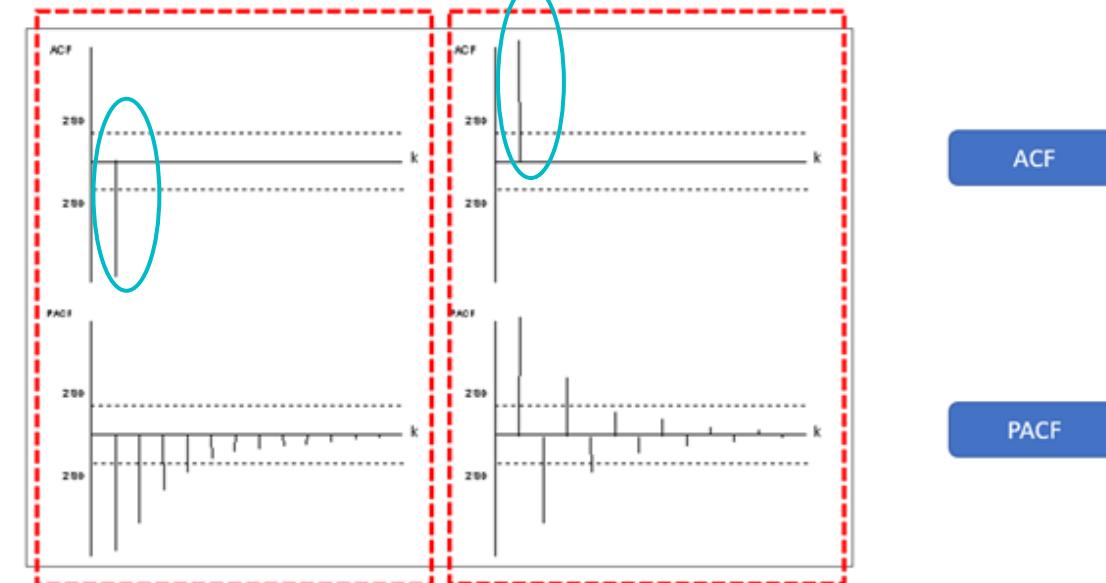


ACF cuts off pada lag 1, PACF
dies down

Identifikasi Model

ACF dies down, PACF cuts off
pada lag 1

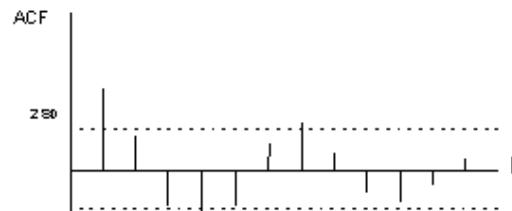
ACF and PACF for MA (1)



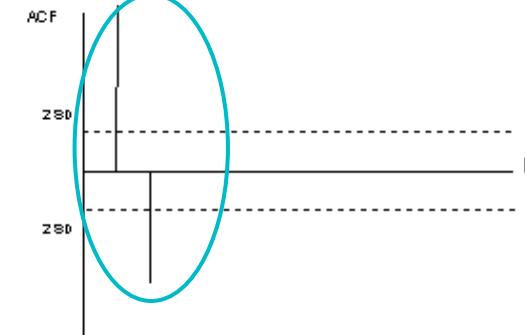
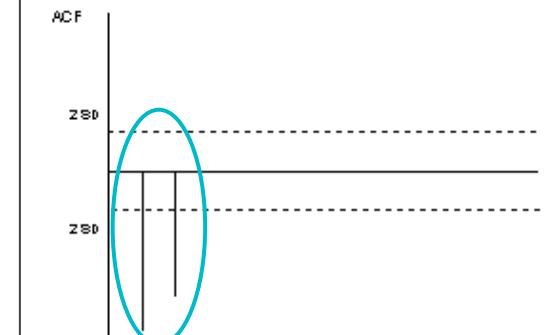
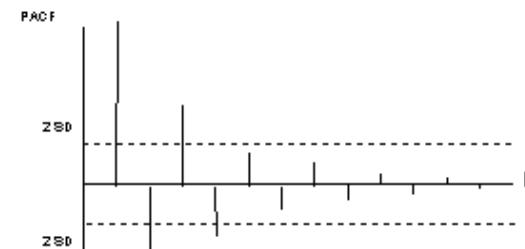
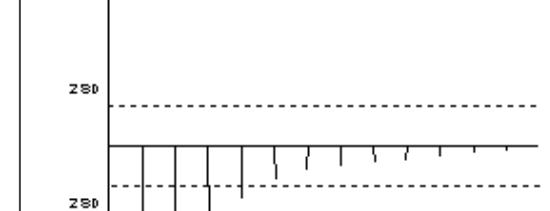
Contoh ACF dan PACF untuk AR (2)

ACF

ACF sinus/cosinus, PACF cuts off pada lag 2

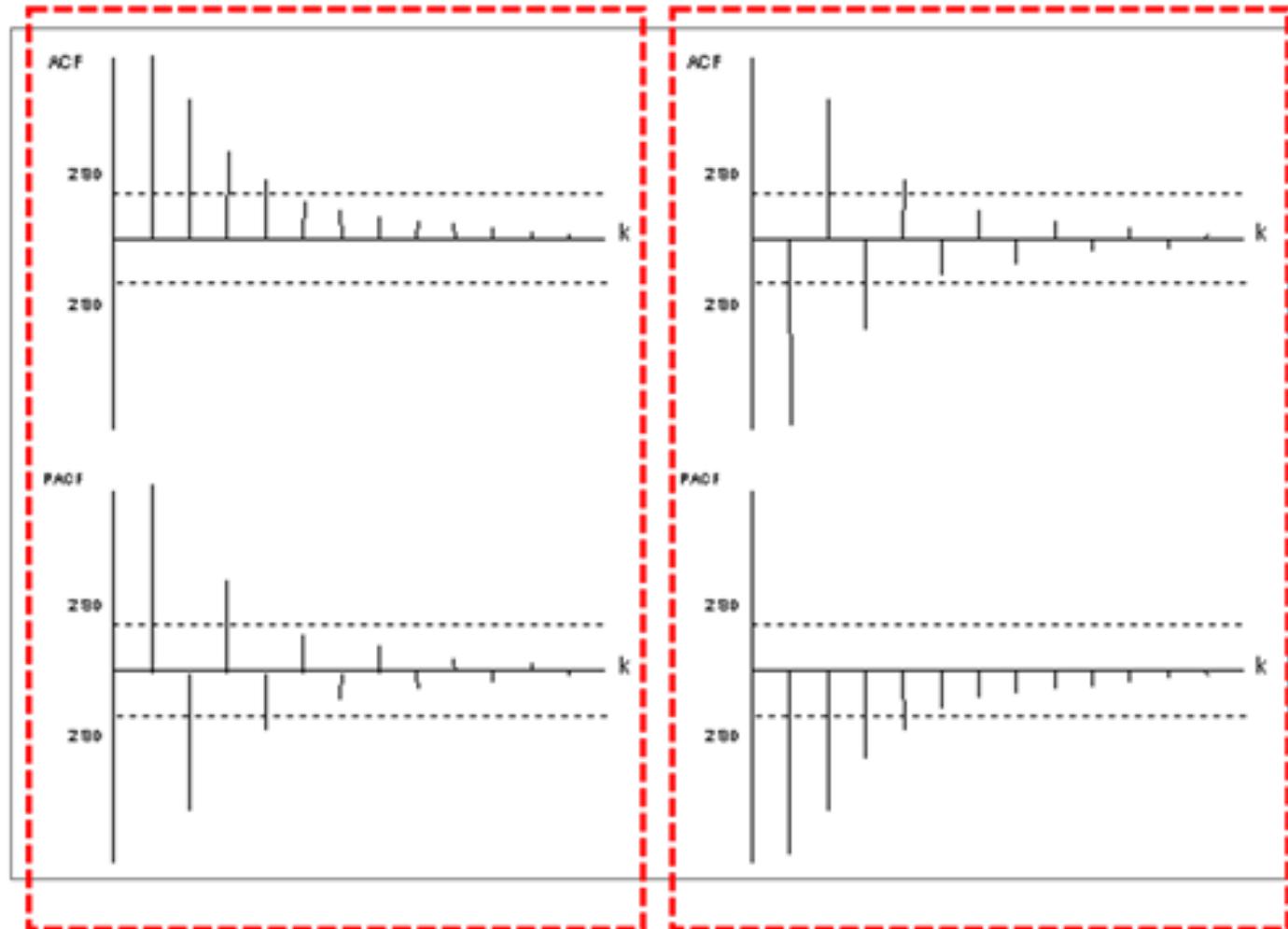
**PACF****ACF**

ACF cuts off lag 2, ACF dies down

**PACF****PACF**

Identifikasi Model

Identifikasi Model



**ACF and PACF
for ARMA(1,1)**

ACF

ACF *dies down*
PACF *dies down*

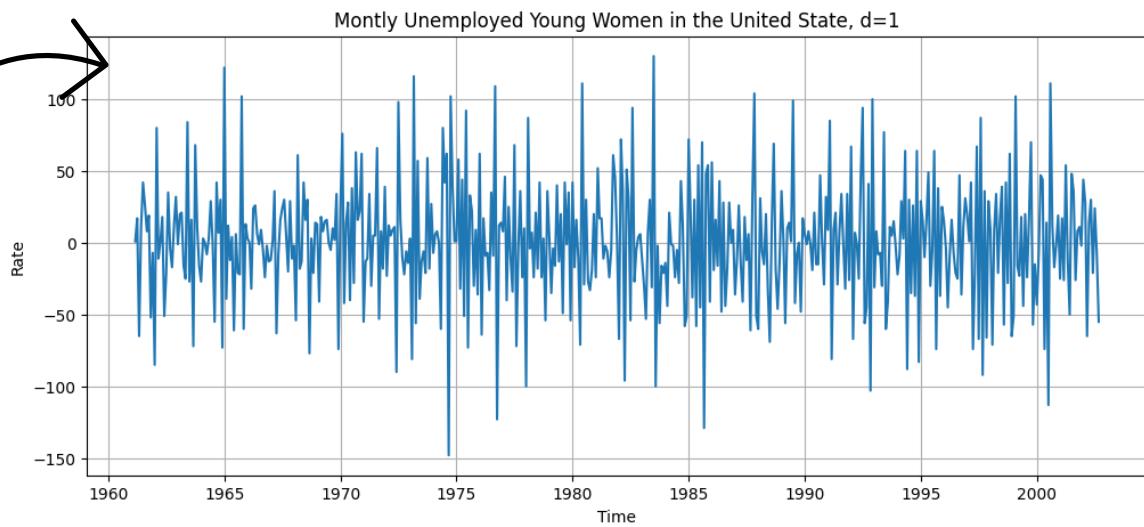
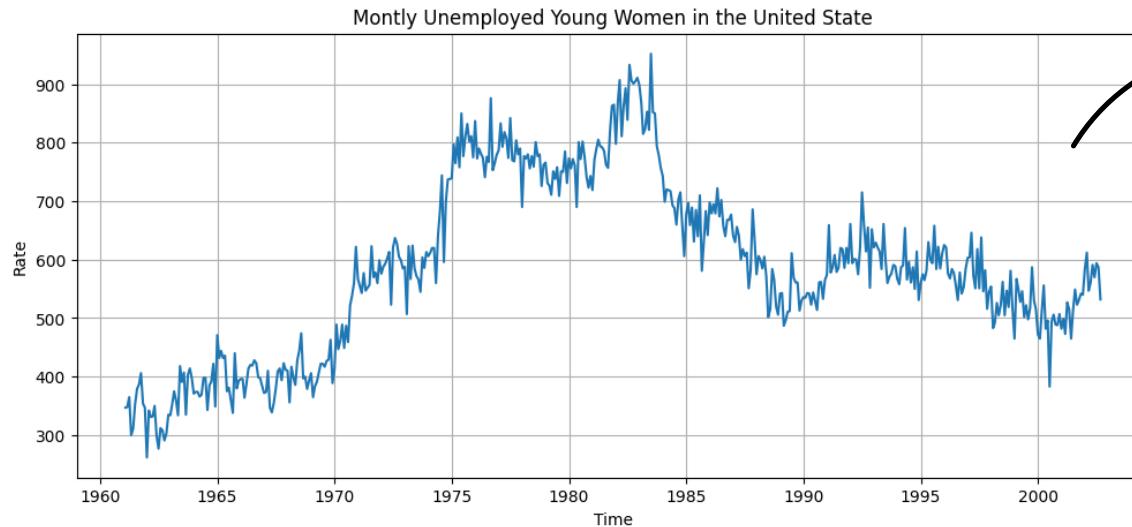
PACF

Contoh :

Monthly Unemployed Young Women in the United State

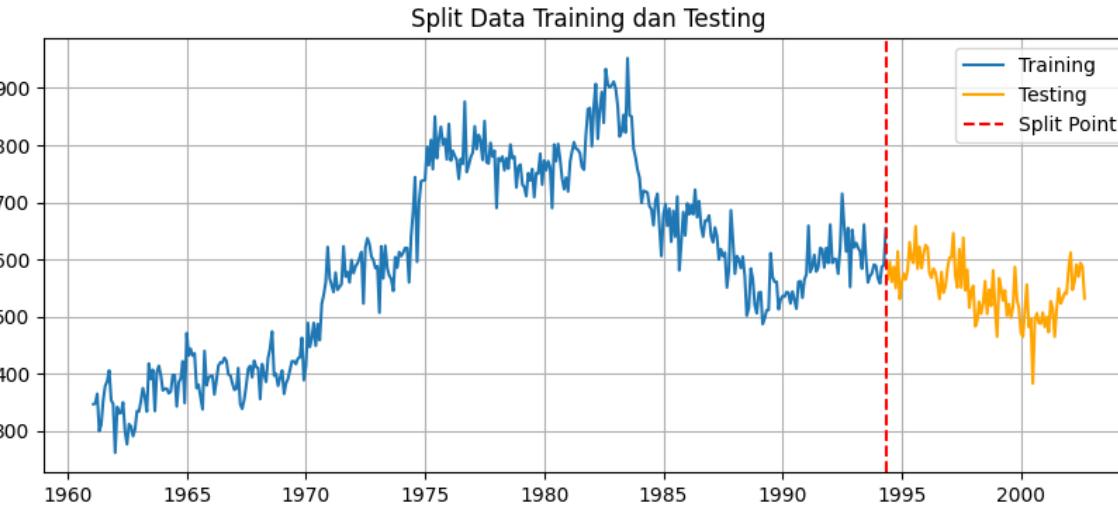
(Wei, 2006)

$$\text{Differencing} = Y_t - Y_{t-1}$$

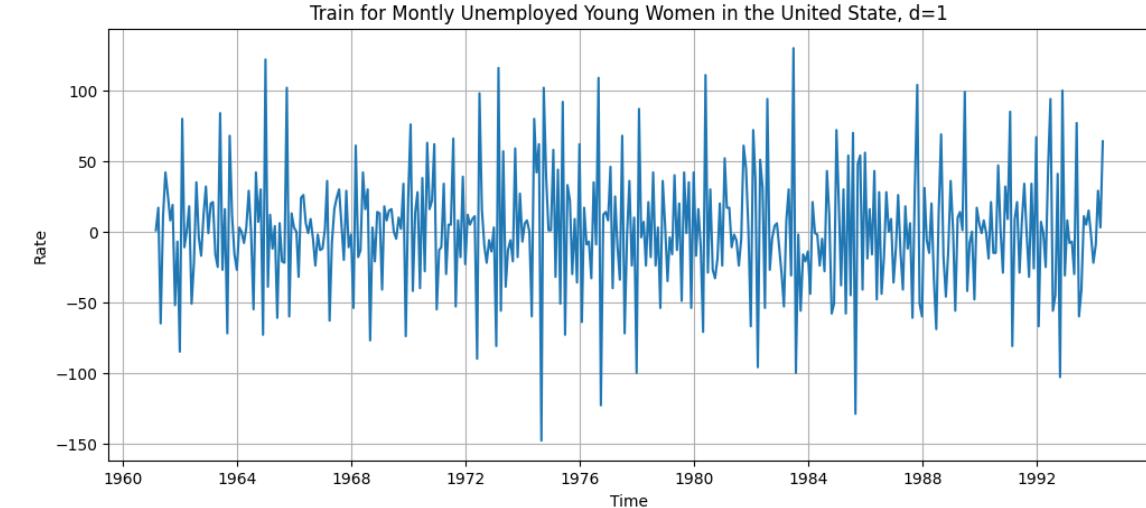


Contoh :

Monthly Unemployed Young Women in the United State (Wei, 2006)



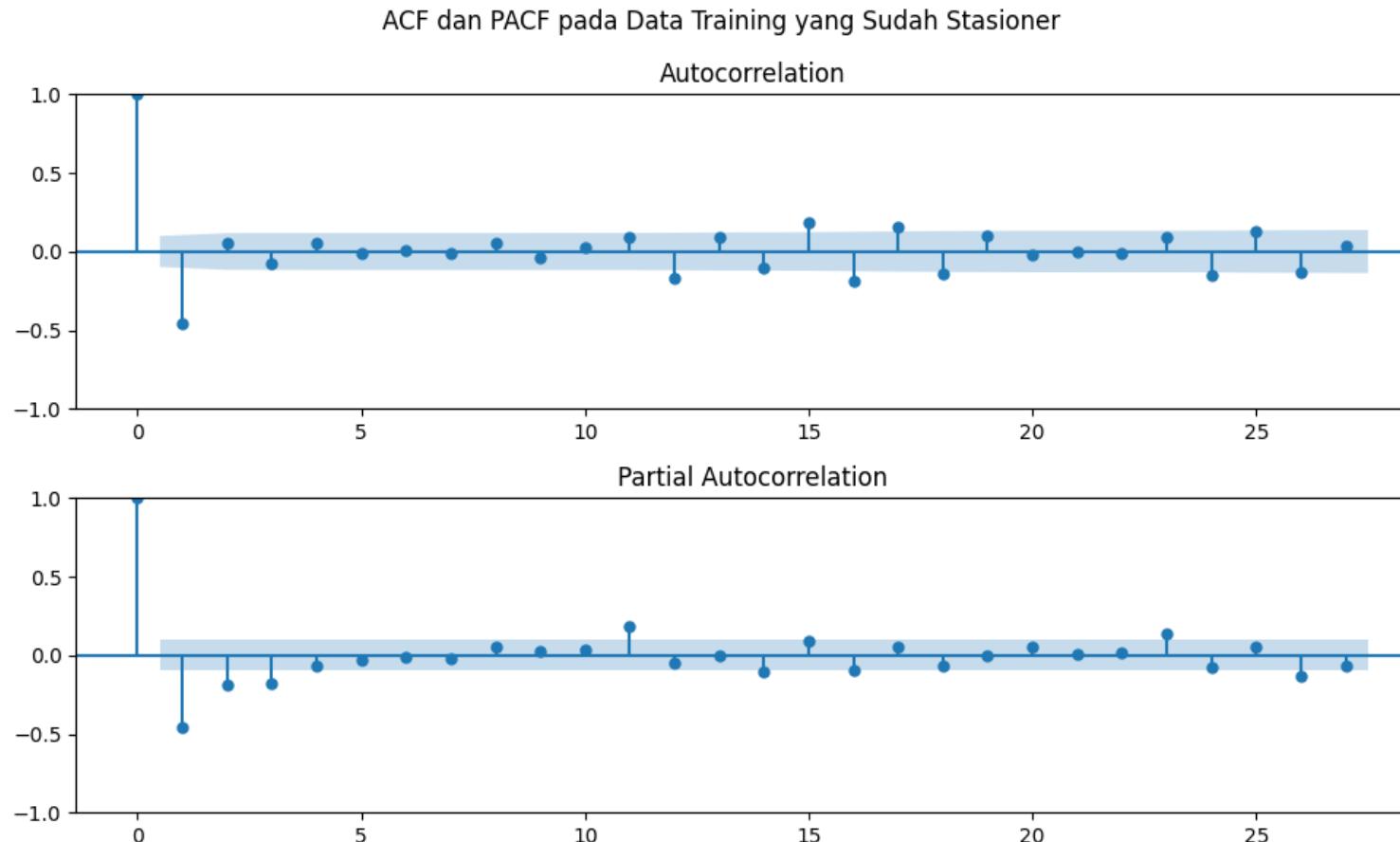
ADF Statistic: -1.934
p-value: 0.316



ADF Statistic setelah differencing: -4.700
p-value: 8.412e-05

Contoh :

Monthly Unemployed Young Women in the United State (Wei, 2006)

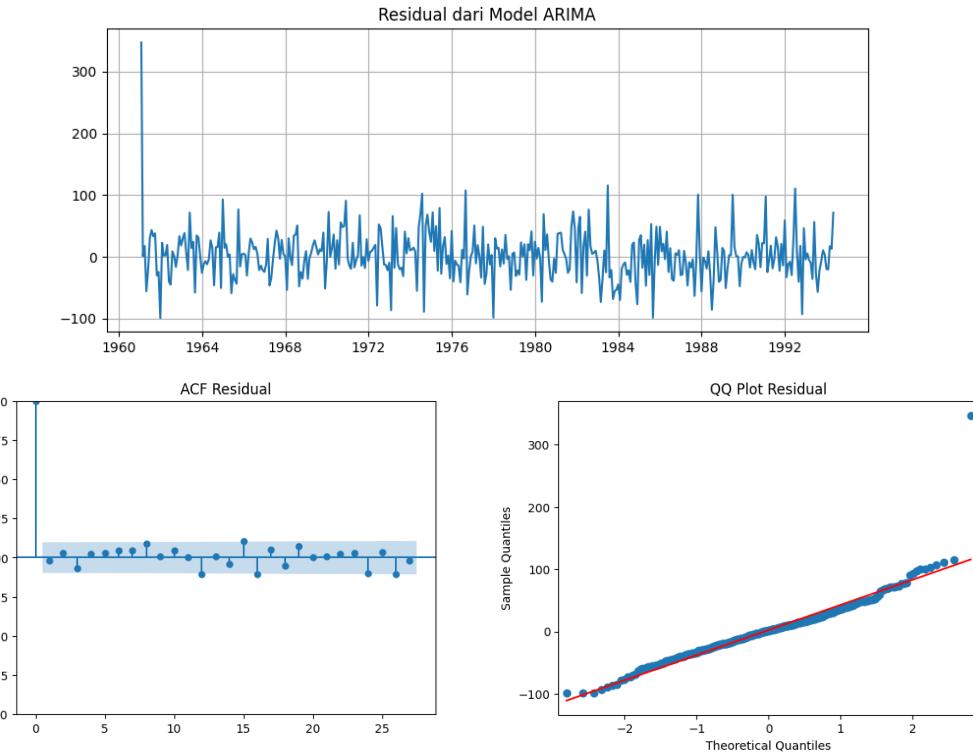


Model tentative :
ARIMA(0,1,1)

Contoh :

Monthly Unemployed Young Women in the United State

```
SARIMAX Results
=====
Dep. Variable:      Un_Young_W    No. Observations:             400
Model:              ARIMA(0, 1, 1)   Log Likelihood:          -2002.308
Date:                Mon, 19 May 2025 AIC:                   4008.616
Time:                  13:51:15       BIC:                   4016.594
Sample:               01-31-1961   HQIC:                  4011.775
                           - 04-30-1994
Covariance Type:    opg
=====
            coef    std err        z     P>|z|      [0.025      0.975]
-----
ma.L1     -0.5776    0.037   -15.672      0.000      -0.650      -0.505
sigma2    1336.3917  82.761    16.148      0.000     1174.183     1498.600
=====
Ljung-Box (L1) (Q):      0.21    Jarque-Bera (JB):      7.83
Prob(Q):                  0.65    Prob(JB):           0.02
Heteroskedasticity (H):    1.40    Skew:                 0.17
Prob(H) (two-sided):      0.05    Kurtosis:            3.59
=====
```



Contoh :

Monthly Unemployed Young Women in the United State

SARIMAX Results

```
=====
Dep. Variable: Un_Young_W No. Observations: 400
Model: ARIMA(0, 1, 2) Log Likelihood -2002.109
Date: Mon, 19 May 2025 AIC 4010.219
Time: 11:58:06 BIC 4022.185
Sample: 01-31-1961 HQIC 4014.958
- 04-30-1994
Covariance Type: opg
=====
```

	coef	std err	z	P> z	[0.025	0.975]
ma.L1	-0.5967	0.047	-12.640	0.000	-0.689	-0.504
ma.L2	0.0291	0.053	0.550	0.582	-0.075	0.133
sigma2	1335.0450	82.630	16.157	0.000	1173.093	1496.997

```
=====
Ljung-Box (L1) (Q): 0.00 Jarque-Bera (JB): 8.08
Prob(Q): 0.95 Prob(JB): 0.02
Heteroskedasticity (H): 1.40 Skew: 0.18
Prob(H) (two-sided): 0.05 Kurtosis: 3.60
=====
```

Overfitting:
ARIMA(0,1,2)

SARIMAX Results

```
=====
Dep. Variable: Un_Young_W No. Observations: 400
Model: ARIMA(1, 1, 1) Log Likelihood -2002.100
Date: Mon, 19 May 2025 AIC 4010.200
Time: 11:58:28 BIC 4022.167
Sample: 01-31-1961 HQIC 4014.939
- 04-30-1994
Covariance Type: opg
=====
```

	coef	std err	z	P> z	[0.025	0.975]
ar.L1	-0.0528	0.091	-0.580	0.562	-0.231	0.126
ma.L1	-0.5446	0.074	-7.403	0.000	-0.689	-0.400
sigma2	1334.9588	82.556	16.170	0.000	1173.151	1496.767

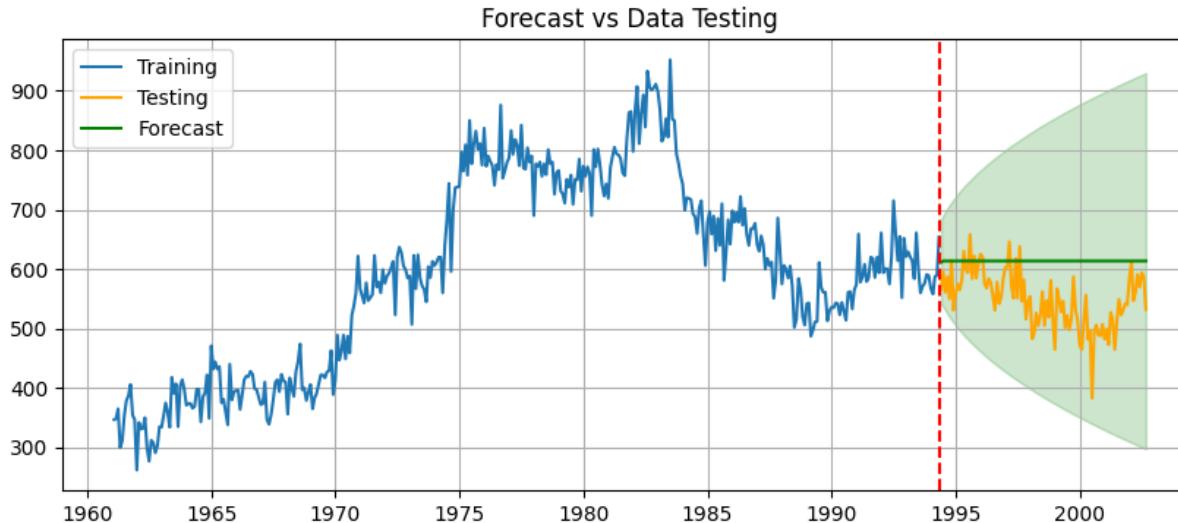
```
=====
Ljung-Box (L1) (Q): 0.00 Jarque-Bera (JB): 8.05
Prob(Q): 0.96 Prob(JB): 0.02
Heteroskedasticity (H): 1.40 Skew: 0.18
Prob(H) (two-sided): 0.05 Kurtosis: 3.60
=====
```

Overfitting:
ARIMA(1,1,1)

Model terbaik → ARIMA(0,1,1)

Contoh :

Monthly Unemployed Young Women in the United State



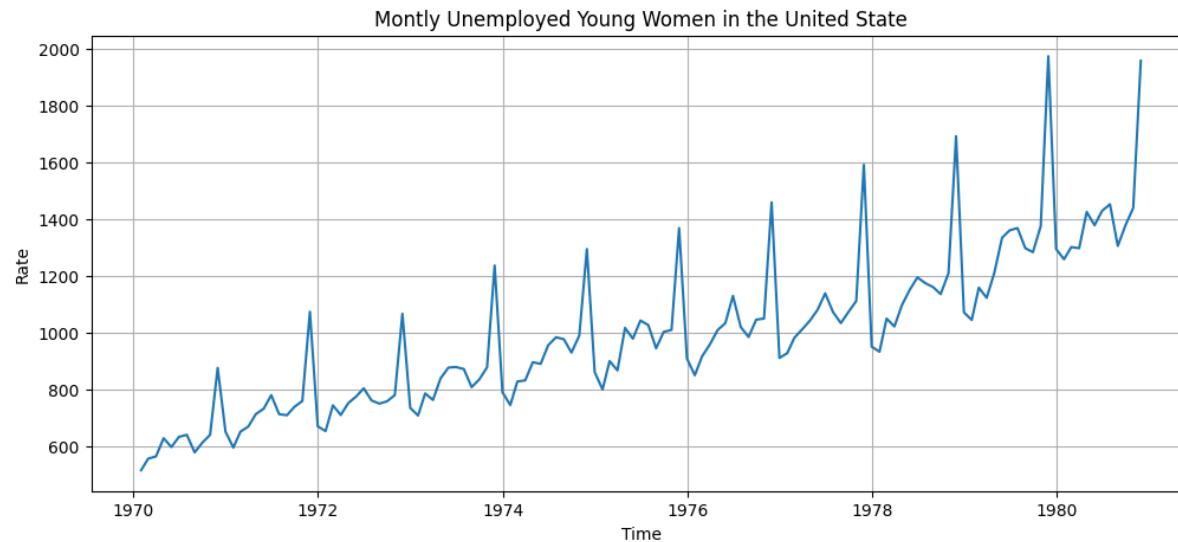
MAPE: 13.19%

MSE: 6471.29

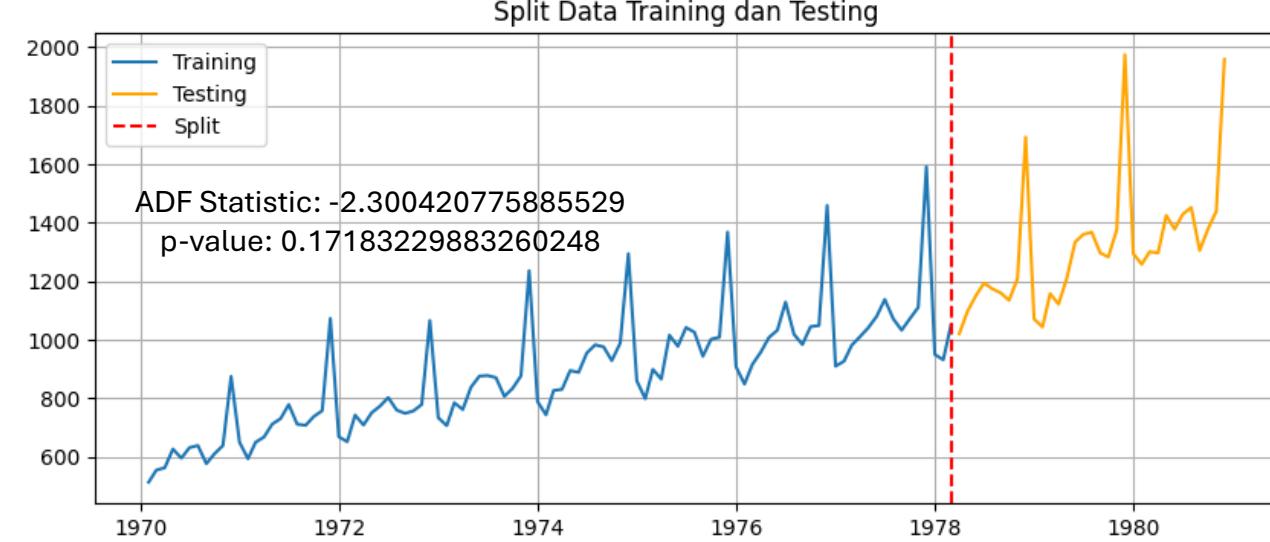
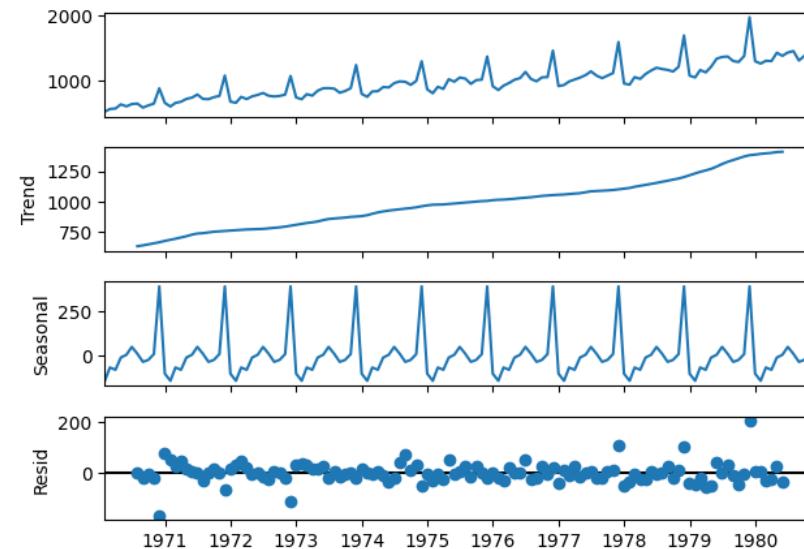
MAE: 68.00

# Summary hasil forecast					
print(forecast_result.summary_frame())					
Un_Young_W	mean	mean_se	mean_ci_lower	mean_ci_upper	
2000-08-31	465.181971	36.728613	393.195213	537.168730	
2000-09-30	467.419300	39.145888	390.694769	544.143831	
2000-10-31	467.245602	42.045288	384.838351	549.652852	
2000-11-30	467.259087	44.709542	379.629995	554.888179	
2000-12-31	467.258040	47.227217	374.694396	559.821685	
2001-01-31	467.258122	49.617046	370.010498	564.505745	
2001-02-28	467.258115	51.896961	365.541941	568.974289	
2001-03-31	467.258116	54.080843	361.261610	573.254621	
2001-04-30	467.258116	56.179896	357.147543	577.368689	
2001-05-31	467.258116	58.203297	353.181749	581.334482	
2001-06-30	467.258116	60.158681	349.349267	585.166964	
2001-07-31	467.258116	62.052478	345.637493	588.878738	
2001-08-31	467.258116	63.890165	342.035694	592.480537	
2001-09-30	467.258116	65.676451	338.534637	595.981595	
2001-10-31	467.258116	67.415424	335.126313	599.389918	
2001-11-30	467.258116	69.110654	331.803723	602.712508	
2001-12-31	467.258116	70.765285	328.560706	605.955526	
2002-01-31	467.258116	72.382102	325.391803	609.124428	
2002-02-28	467.258116	73.963584	322.292155	612.224077	
2002-03-31	467.258116	75.511952	319.257410	615.258822	
2002-04-30	467.258116	77.029202	316.283654	618.232578	
2002-05-31	467.258116	78.517139	313.367351	621.148880	
2002-06-30	467.258116	79.977398	310.505296	624.010935	
2002-07-31	467.258116	81.411469	307.694568	626.821663	
2002-08-31	467.258116	82.820712	304.932502	629.583729	

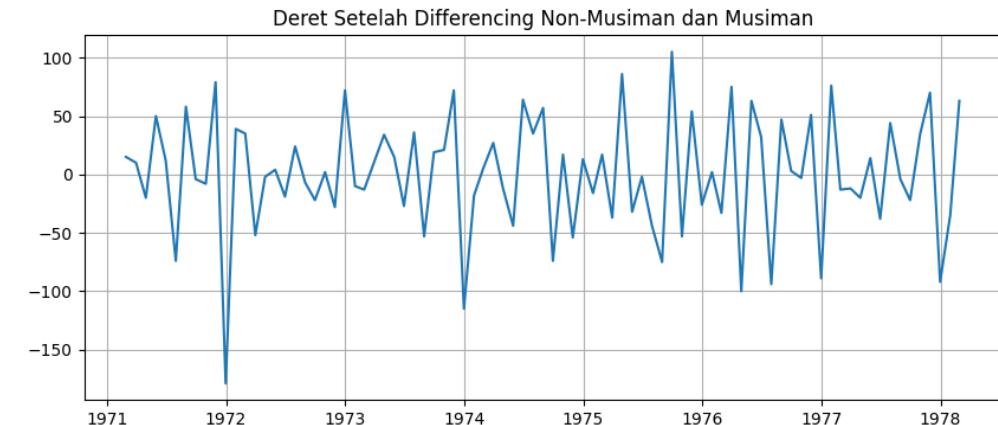
Pemodelan SARIMA



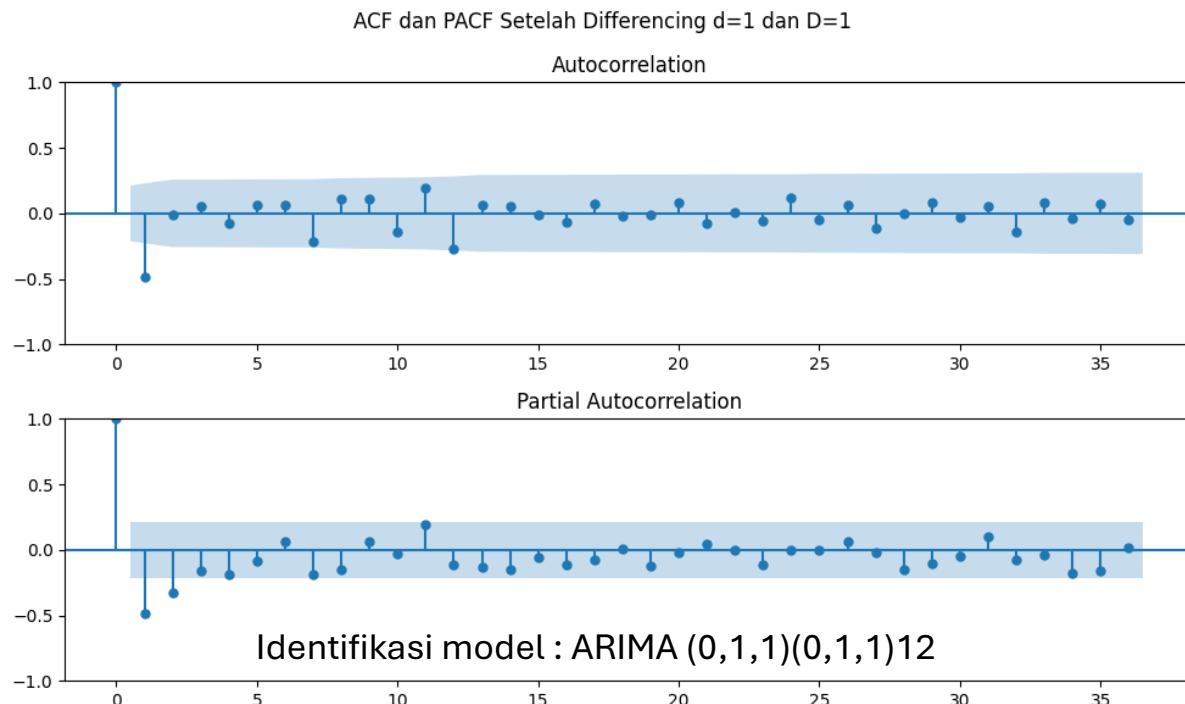
Dekomposisi Deret Waktu



Karena data tidak stasioner, maka dilakukan pembedaan di level non-musiman dan musiman ($d=1$ dan $D=1$)



Pemodelan SARIMA

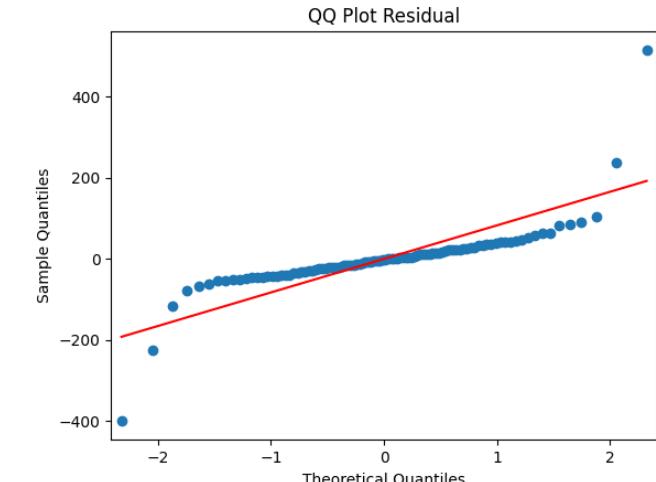
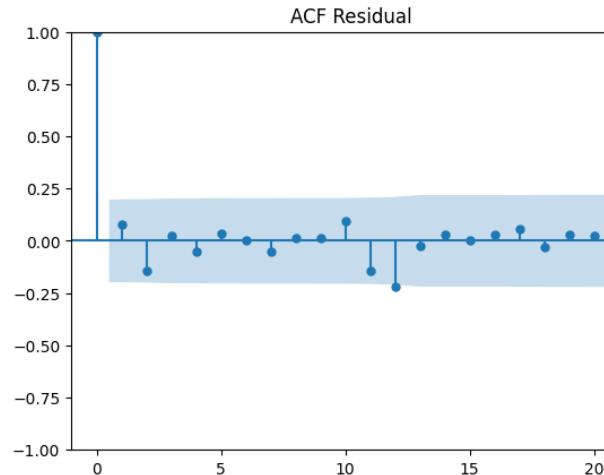
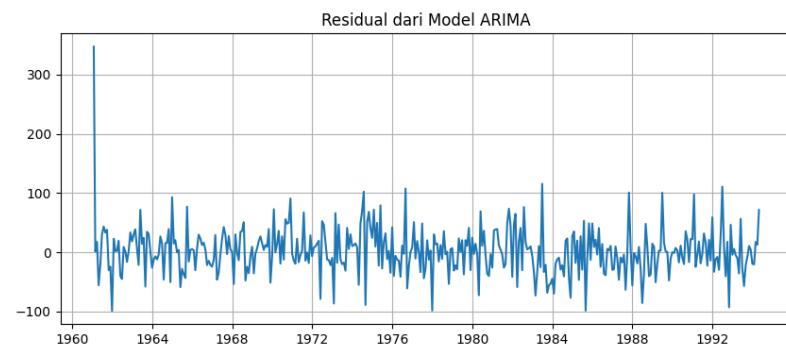


SARIMAX Results

Dep. Variable: 580 No. Observations: 98
 Model: SARIMAX(0, 1, 1)x(0, 1, 1, 12) Log Likelihood: -355.710
 Date: Wed, 21 May 2025 AIC: 717.421
 Time: 04:42:10 BIC: 724.209
 Sample: 01-31-1970 HQIC: 720.120
 - 02-28-1978
 Covariance Type: opg

	coef	std err	z	P> z	[0.025	0.975]
ma.L1	-0.7760	0.094	-8.234	0.000	-0.961	-0.591
ma.S.L12	-0.3047	0.113	-2.698	0.007	-0.526	-0.083
sigma2	1313.1245	209.719	6.261	0.000	902.083	1724.166

Ljung-Box (L1) (Q): 0.01 Jarque-Bera (JB): 6.35
 Prob(Q): 0.93 Prob(JB): 0.04
 Heteroskedasticity (H): 1.52 Skew: 0.68
 Prob(H) (two-sided): 0.31 Kurtosis: 3.53



Hands-On Session #3

- Pemodelan Box-Jenkins ARIMA dan SARIMA



Machine Learning dan Deep Learning untuk Time Series

Pengantar ML untuk time series

- Metode klasik (seperti ARIMA) biasanya akan mempunya performa yang baik pada data yang linier, namun lemah pada data dengan missing value, non-linier, hubungan yang kompleks antar beberapa peubah time series dan peramalan multi-langkah (multi-step).
- ARIMA dan SARIMA adalah teknik statistik klasik untuk peramalan deret waktu. Mereka bukan termasuk *machine learning* maupun *deep learning*, tapi tetap penting karena: sangat andal sebagai model baseline, mudah diinterpretasikan, dan seringkali mengungguli DL saat data sedikit.
- ML sebagai pendekatan alternatif/statistik modern.
- Kapan ML digunakan: banyak variabel (multivariat), pola non-linier, dan data besar serta kompleks
- Prinsip Dasar Machine Learning :
 - Supervised learning
 - Proses umum:
 1. Data → Fitur
 2. Training → Model
 3. Prediksi dan evaluasi

Pengantar DL untuk time series

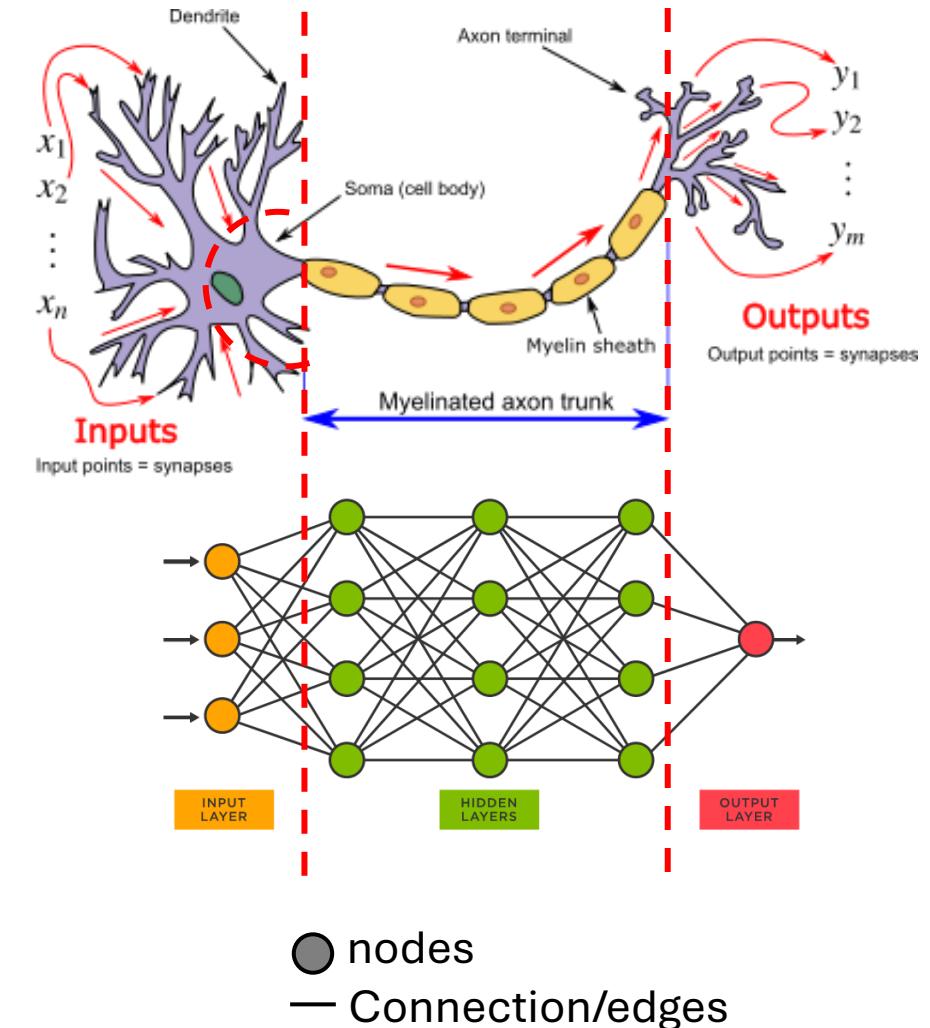
- Metode tradisional time series sering kali kurang fleksibel untuk menangani data berskala besar, pola non-linear, interaksi kompleks antar variable serta long-term dependencies (ketergantungan jarak jauh, misal: pengaruh data bulan lalu terhadap hari ini)
- *Deep Learning* hadir sebagai solusi revolusioner
- Dengan kemampuan *neural networks* untuk mempelajari representasi data secara otomatis (feature learning), teknik seperti RNN, LSTM, GRU, dan Transformer mampu mengekstrak pola temporal yang sulit ditangkap metode statistik konvensional
- Keunggulan: Akurasi lebih tinggi, adaptif terhadap data non-stasioner, dan mampu memproses multivariate time series.

Aspek	Machine Learning (ML)	Deep Learning (DL)
Pendekatan	Data → fitur → model	Data mentah → model langsung
Kemampuan tangkap pola waktu	Terbatas, bergantung fitur	Kuat – otomatis belajar dependensi waktu
Jenis model umum	Linear regression, random forest, XGBoost, SVR	MLP, CNN, LSTM, GRU, Transformer
Multivariat input	Bisa, tapi terbatas	Sangat fleksibel
Musiman/tren	Harus dibuat manual	Bisa dipelajari langsung
Kebutuhan data	Lebih sedikit	Butuh data besar agar efektif
Kecepatan latih	Cepat (ringan)	Lambat (butuh GPU)
Interpretasi model	Lebih mudah dijelaskan	Cenderung "black-box"
Cocok untuk	Dataset kecil-sedang, hubungan sederhana	Dataset besar, masalah kompleks & long-term

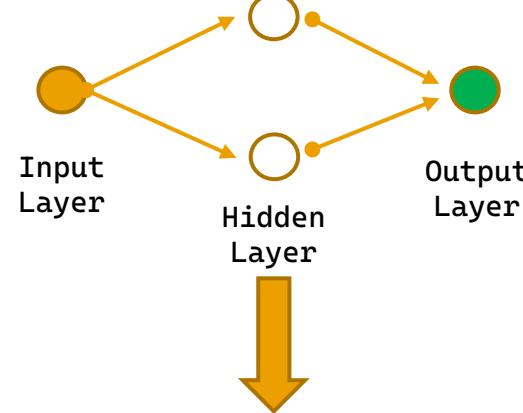
Pengantar Neural Network

Mengadopsi kemampuan jaringan saraf manusia yang mampu menerima stimulasi/rangsangan, melakukan proses, dan memberikan output.

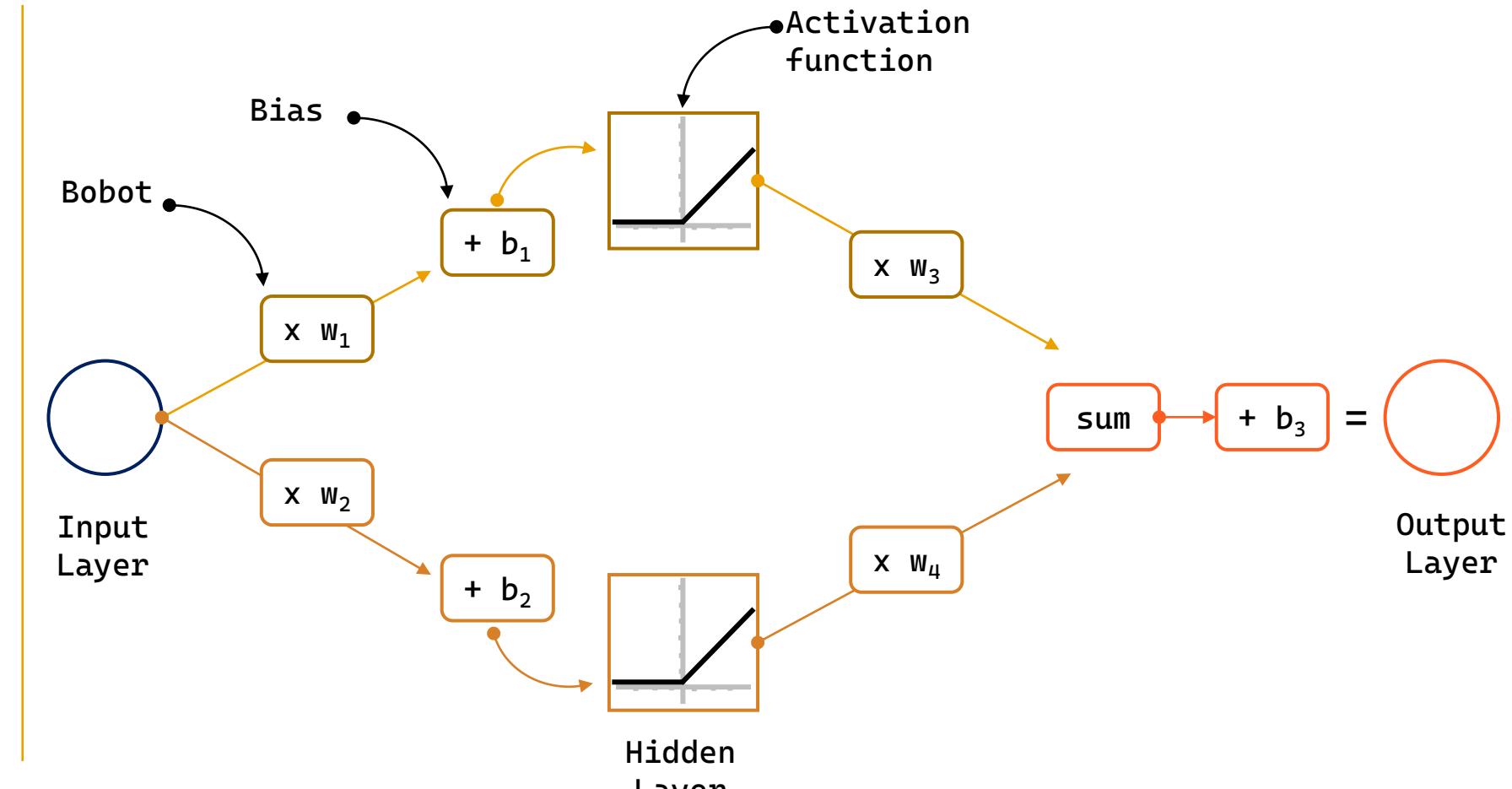
Pertama kali diperkenalkan oleh Warren McCulloch dan Walter Pitts untuk peningkatan komputasi. Selanjutnya dikembangkan oleh Rosenblatt (1950): two-layer network disebut perceptron.



Neural Network – One input & Output

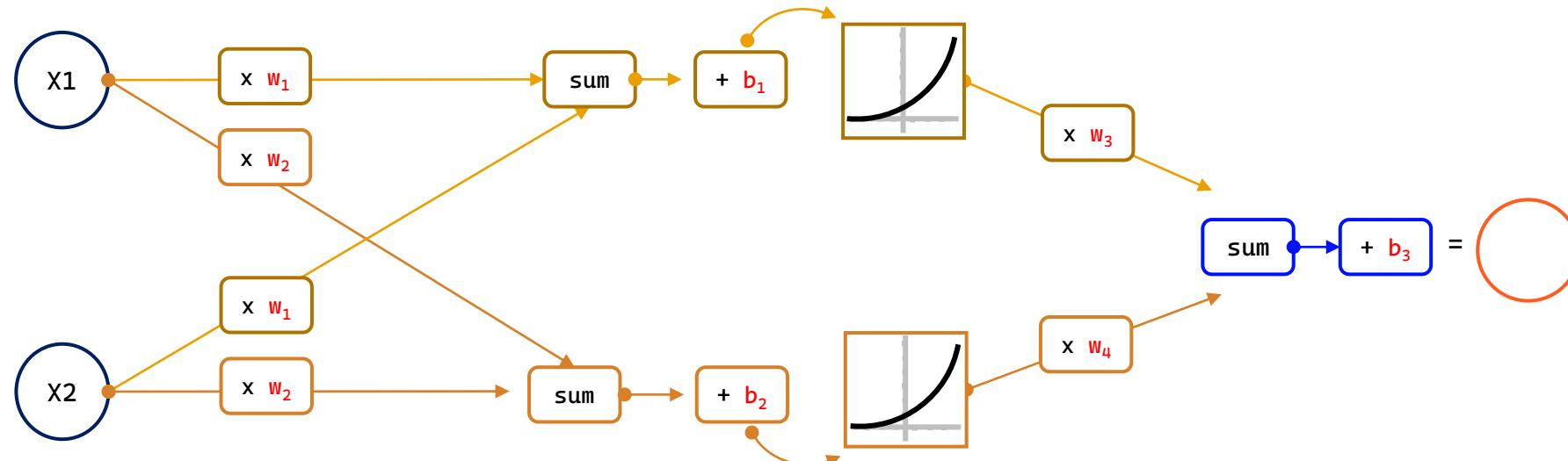


1 Input Layer (X)
1 Hidden Layer, 2 Neuron
1 Output Layer (Y)

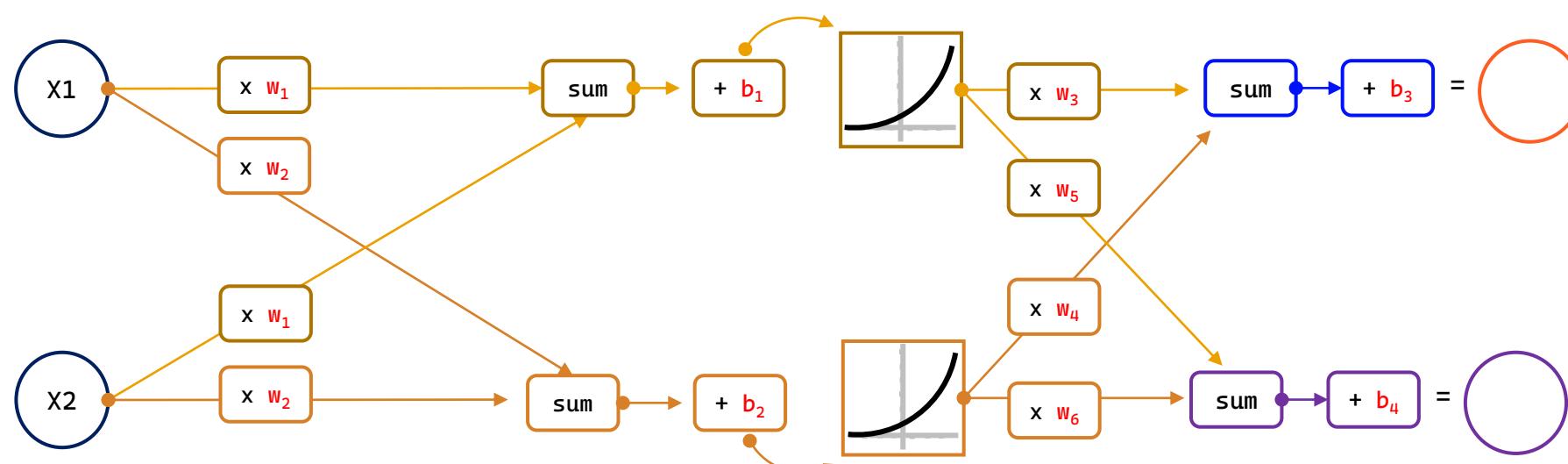


- ✓ Input layer: Menerima fitur dari data (misal suhu, waktu, kelembaban).
- ✓ Hidden layer: Memproses pola dan hubungan antar fitur.
- ✓ Output layer: Menghasilkan prediksi akhir (misalnya nilai suhu ke depan).

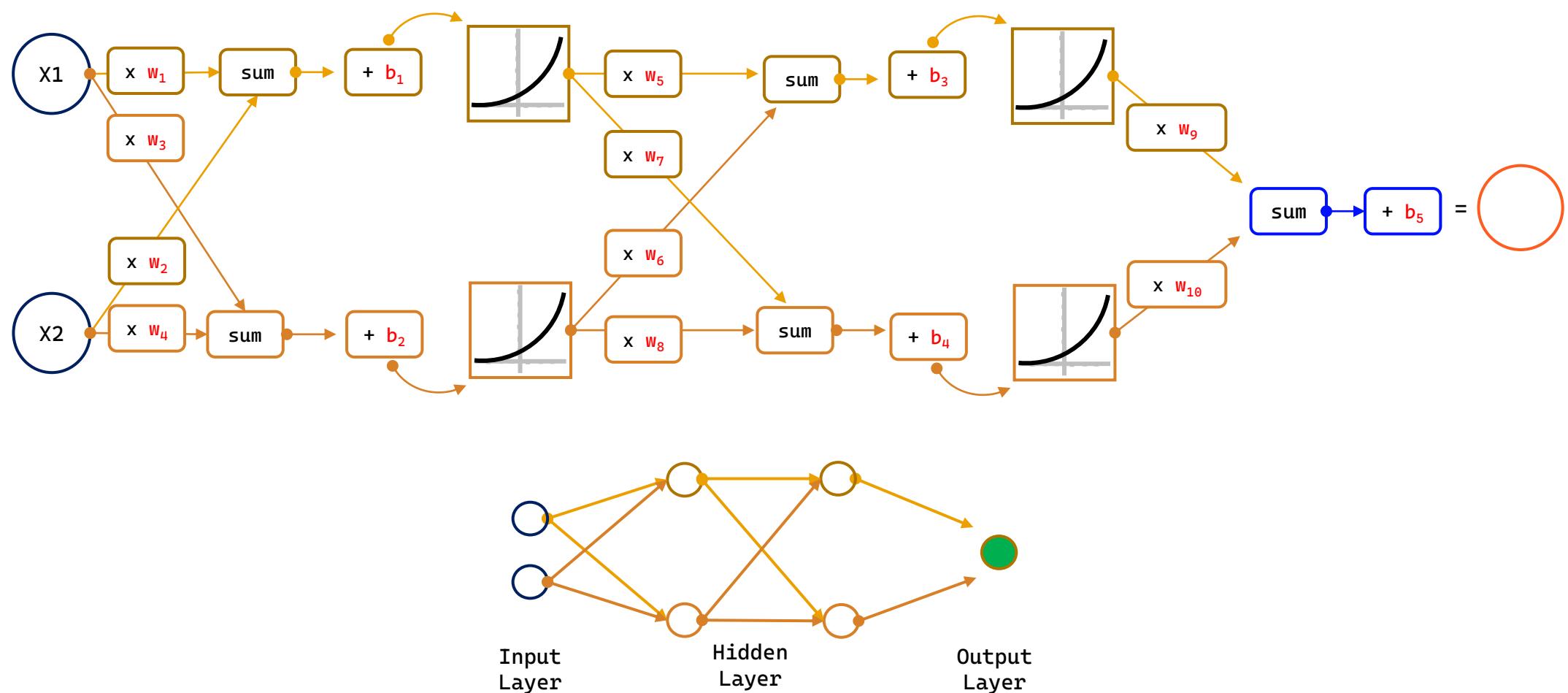
Multi Input



Multi Output



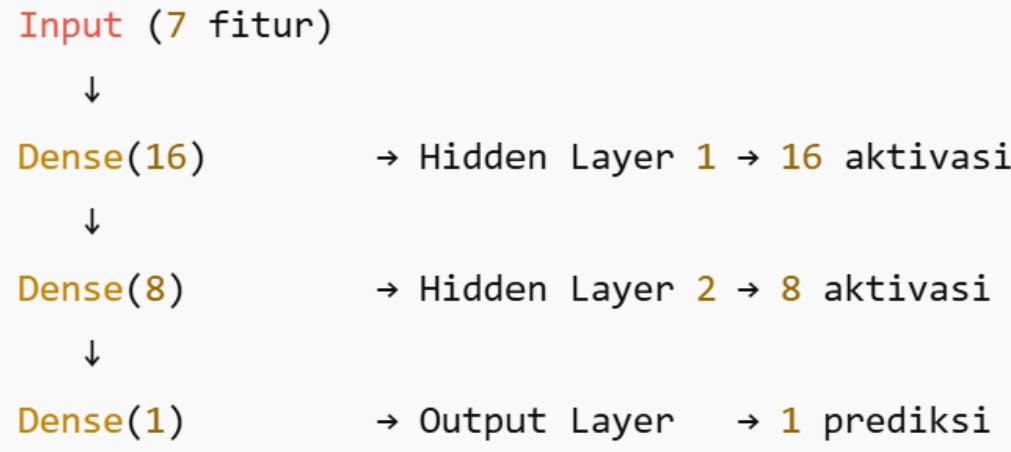
Multiple hidden layer



ANN untuk Forecasting Time Series

- ANN tradisional (seperti MLP/Multi-Layer Perceptron) tidak dirancang khusus untuk data berurutan/time series
- ANN tradisional (MLP) memperlakukan input sebagai data independen, tidak menyimpan informasi dari langkah waktu sebelumnya, akibatnya gagal menangkap *long-term dependencies* (misal: pengaruh musiman tahun lalu)
- ANN tidak bisa langsung memproses data deret waktu mentah. Perlu windowing, normalisasi, atau ekstraksi fitur manual.
- ANN cocok untuk data time series dengan pola jangka pendek (misal: prediksi suhu 1 hari ke depan)

ANN untuk Forecasting Time Series



◆ 1. Input Layer → Hidden Layer 1 (Dense 16)

- Input: 7 nilai (misalnya suhu hari 1–7)
- Setiap dari **16 neuron** menerima semua 7 input
- Masing-masing neuron punya **7 bobot + 1 bias**
- Hasil: **16 aktivasi** = a_1, a_2, \dots, a_{16}

$$a_i = \text{ReLU}(w_{i1}x_1 + w_{i2}x_2 + \cdots + w_{i7}x_7 + b_i)$$

◆ 2. Hidden Layer 1 → Hidden Layer 2 (Dense 8)

- Input: 16 aktivasi dari layer sebelumnya
- **8 neuron**, masing-masing menerima semua 16 aktivasi sebagai input
- Hasil: **8 aktivasi baru** = h_1, h_2, \dots, h_8

$$h_j = \text{ReLU}(v_{j1}a_1 + v_{j2}a_2 + \cdots + v_{j16}a_{16} + c_j)$$

◆ 3. Hidden Layer 2 → Output Layer (Dense 1)

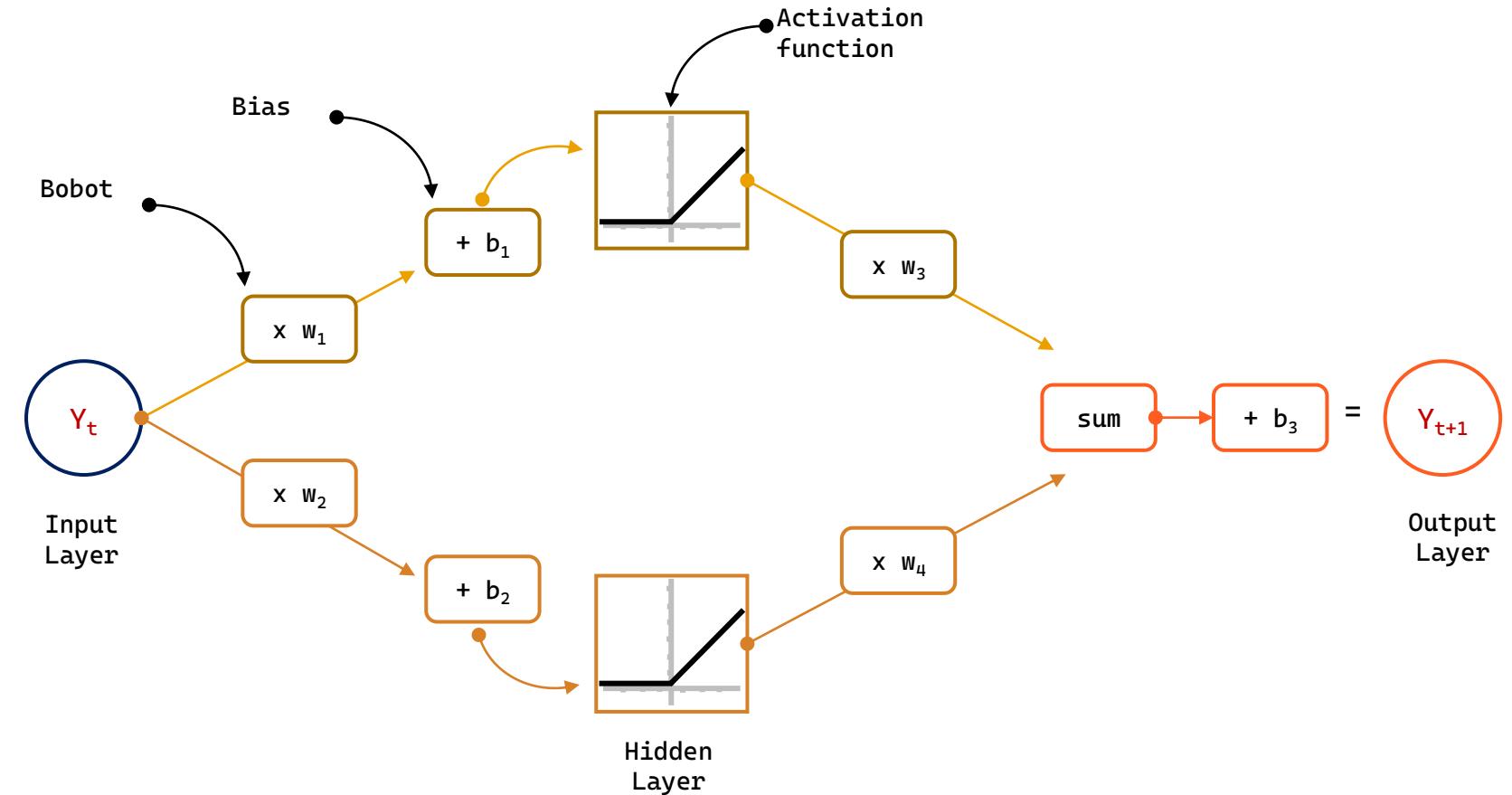
- Input: 8 aktivasi dari layer sebelumnya
- **1 neuron output**, menerima semua 8 input
- Hasil: **1 nilai prediksi akhir** \hat{y}

$$\hat{y} = w_1h_1 + w_2h_2 + \cdots + w_8h_8 + b$$

ANN untuk Forecasting Time Series

$$y_{t+1} = f(y_t) + \epsilon_t$$

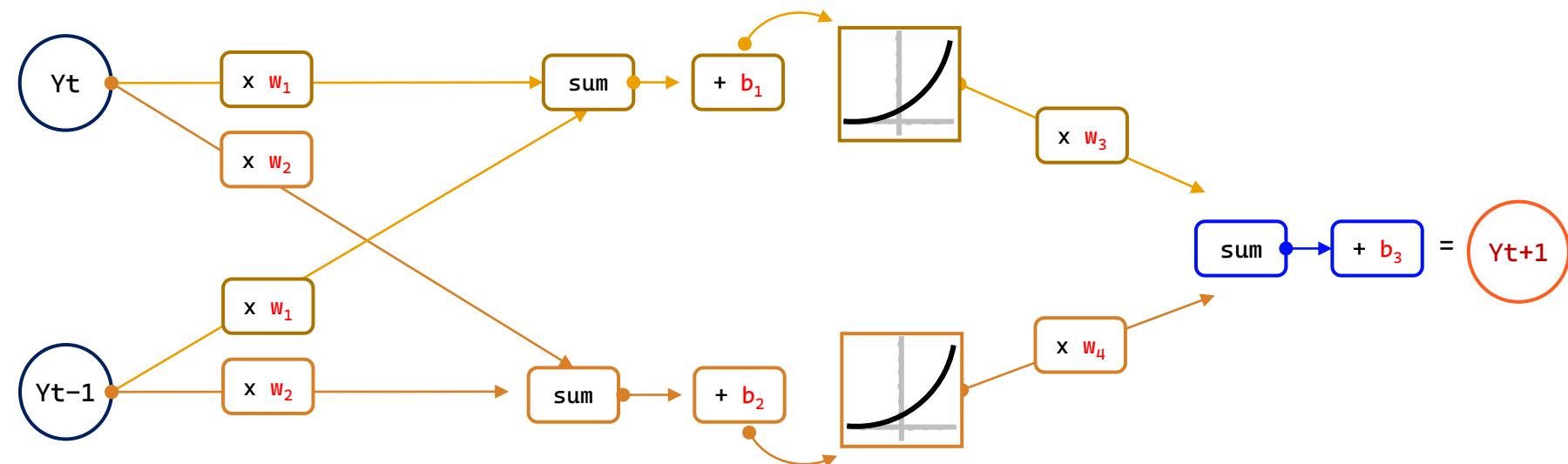
yt	Y	X
y ₁	y ₂	y ₁
y ₂	y ₃	y ₂
y ₃
...	y _T	y _{T-1}
yt		



ANN untuk Forecasting Time Series

$$y_{t+1} = f(y_t, y_{t-1}) + \epsilon_t$$

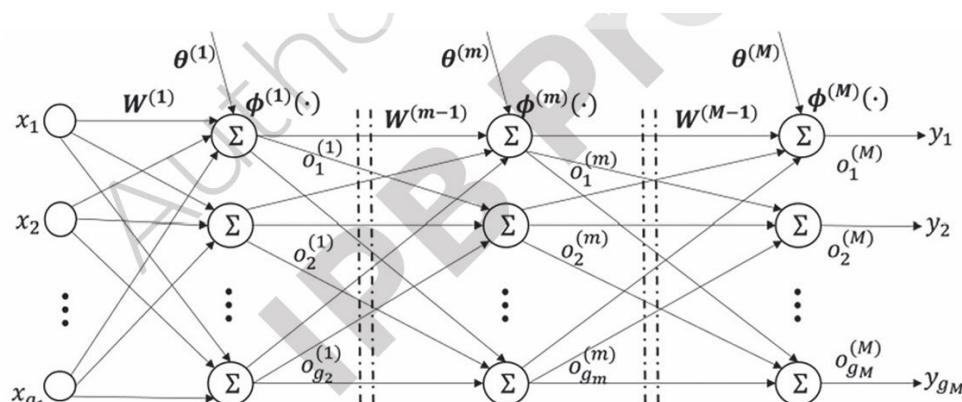
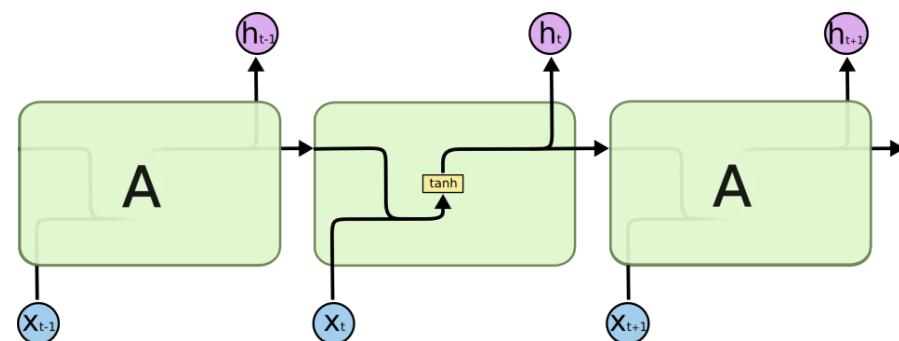
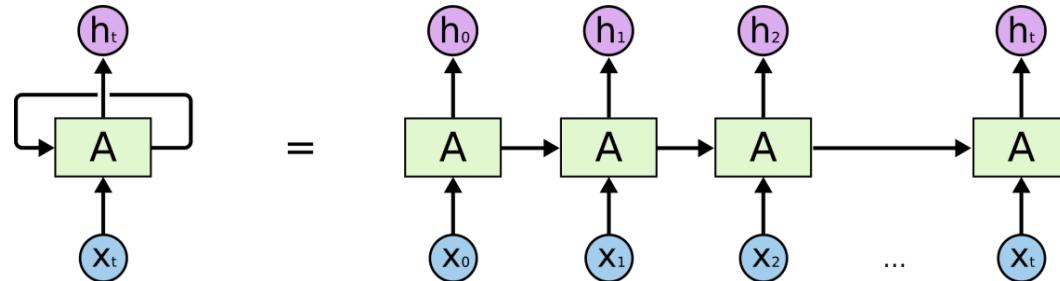
yt	Y	X1	X2
y ₁	y ₃	y ₂	y ₁
y ₂	y ₄	y ₃	y ₂
y ₃
...	y _T	y _{T-1}	y _{T-2}
y _T			



Recurrent Neural Network (RNN)

- Model ANN sebelumnya akan menjadi sangat kompleks ketika pengaruh jangka panjang terjadi
- Urutan dalam data time series penting, tapi model ANN sebelumnya tidak memperhatikan urutan pada data
- RNN memperkenalkan *looping mechanism* (mekanisme perulangan) yang memungkinkan jaringan menyimpan informasi dari langkah sebelumnya.
- Cara kerja RNN : Setiap neuron RNN memiliki **hidden state** yang menyimpan informasi dari input sebelumnya.

Arsitektur 1 Unit RNN



Misalnya kamu ingin prediksi suhu hari ke-8 berdasarkan 7 hari sebelumnya.

Input: [30, 32, 33, 35, 34, 36, 37]

Satu unit RNN bekerja sebagai berikut:

1. Step t=1: input = 30 → menghasilkan hidden state h_1
2. Step t=2: input = 32 + h_1 → menghasilkan h_2
3. Step t=3: input = 33 + h_2 → ...
4. ...
5. Step t=7: input = 37 + h_6 → menghasilkan h_7
6. h_7 dikirim ke Dense(1) → suhu prediksi hari ke-8

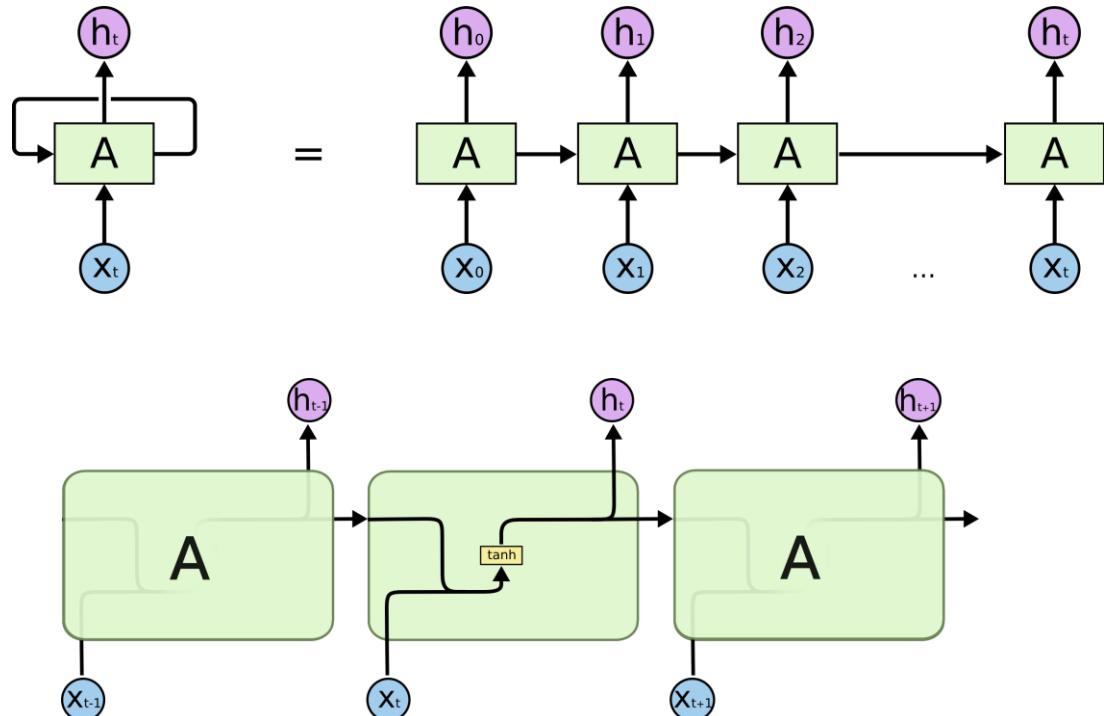
```

x1 → RNN → h1
x2 + h1 → RNN → h2
...
x7 + h6 → RNN → h7 → Dense → Prediksi
  
```

Satu unit RNN itu seperti otak mini yang:

- ✓ Menerima 1 data suhu per-t
- ✓ Mengolahnya
- ✓ Mengingat informasi penting dari hari sebelumnya

Arsitektur RNN



$$\hat{y} = w_1 h_1^{(2)} + w_2 h_2^{(2)} + \dots + w_8 h_8^{(2)} + b$$

Misalnya ada 7 data suhu harian:

[30, 31, 29, 32, 33, 34, 35]

- Ingin diprediksi suhu ke-8 berdasarkan pola 7 suhu sebelumnya.
- Menggunakan 2 hidden layer 16 dan 8

Pada layer 1 : 16 unit RNN

Bayangkan ada **16 otak mini (neuron)** yang bekerja **bersamaan**
Apa yang terjadi?

1. Hari ke-1 (input pertama: 30)

1. Semua 16 unit mengolah angka 30
2. Karena ini hari pertama, belum ada memori sebelumnya (pakai memori awal = nol)
3. Masing-masing menyimpan sedikit informasi (disebut hidden state)

2. Hari ke-2 (input: 31)

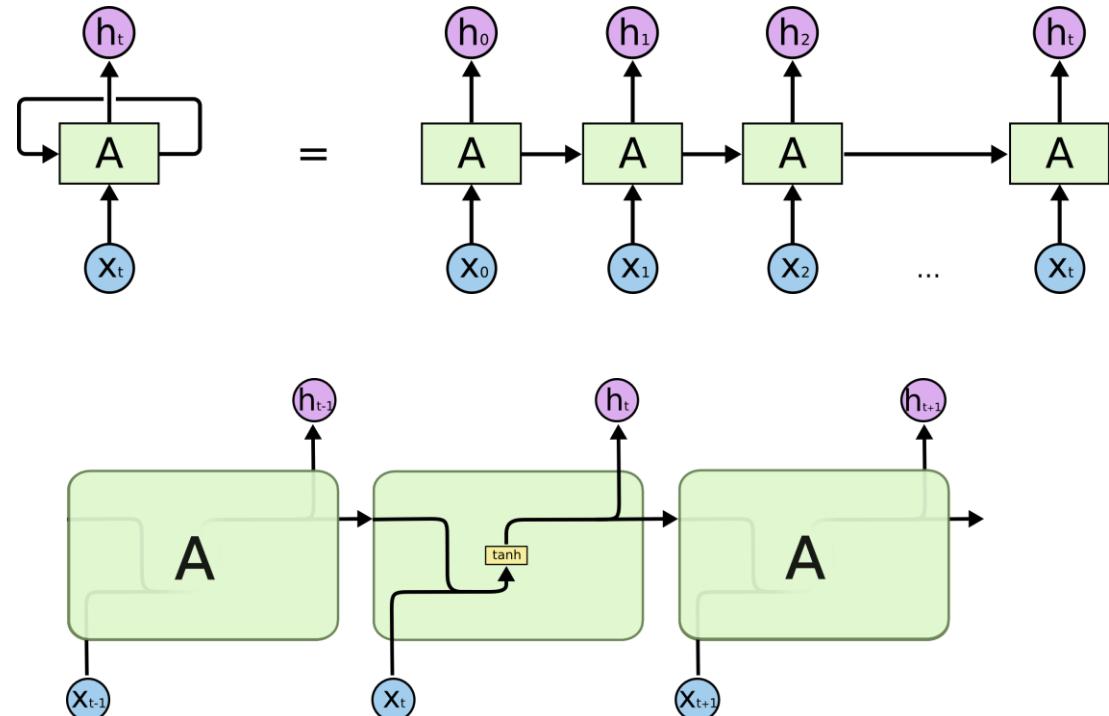
1. Setiap unit memproses angka 31 + informasi dari hari ke-1 (memori sebelumnya)
2. Lalu menyimpan memori baru

3. Begitu seterusnya sampai hari ke-7...

Akhirnya:

- Setiap unit punya memori terakhir setelah hari ke-7
- Jadi, di akhir layer pertama, kita punya 16 angka (1 dari tiap neuron) → ini adalah ringkasan informasi suhu 7 hari versi layer pertama

Arsitektur RNN



Pada layer 2 : 8 unit RNN

Di layer 2, ada **8 otak mini baru**.

Apa input layer kedua?

- ✓ Bukan lagi angka suhu
- ✓ Tapi **hasil olahan dari 16 neuron sebelumnya**

Prosesnya:

1. Hari ke-1:

1. Masing-masing dari 8 neuron menerima **16 angka** dari layer pertama (hasil pemrosesan data hari ke-1)
2. Mereka memulai dengan memori awal (nol)

2. Hari ke-2 sampai hari ke-7:

1. Mereka terus menerima hasil dari layer 1 dan meng-update memorinya

Akhirnya:

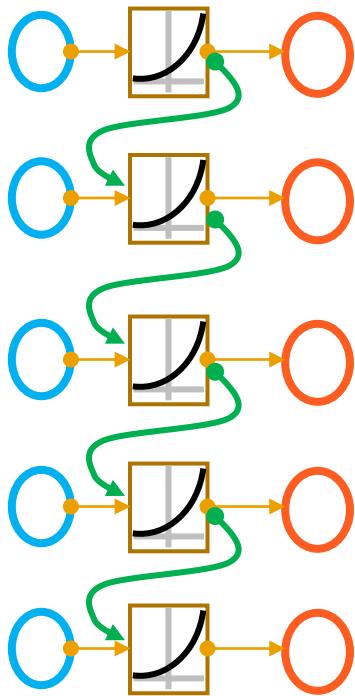
- ✓ Di akhir hari ke-7, setiap dari 8 neuron menyimpan 1 angka (memori terakhirnya)
- ✓ Jadi, layer kedua memberikan kita **8 angka sebagai hasil akhir**

Layer Output :

Pada layer ini terdapat 1 neuron output (misalnya untuk memprediksi suhu hari ke-8). Neuron ini menerima 8 angka dari layer kedua, lalu menghitung:

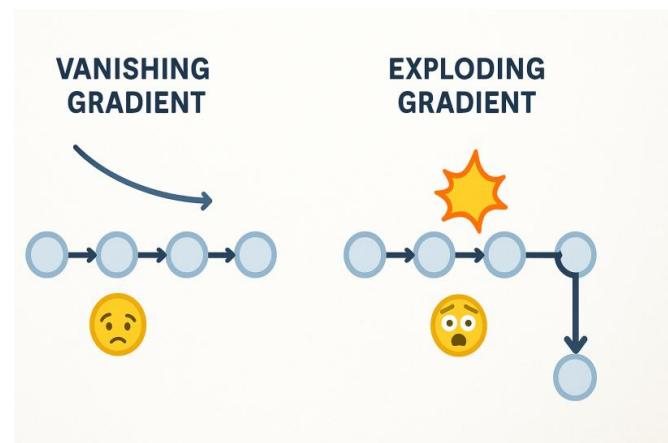
$$\hat{y} = w_1 h_1^{(2)} + w_2 h_2^{(2)} + \cdots + w_8 h_8^{(2)} + b$$

Permasalahan di RNN kalau rantainya panjang



Saat kita melatih model RNN (Recurrent Neural Network), kita menggunakan proses yang disebut **backpropagation** untuk memperbaiki bobot-bobot model.

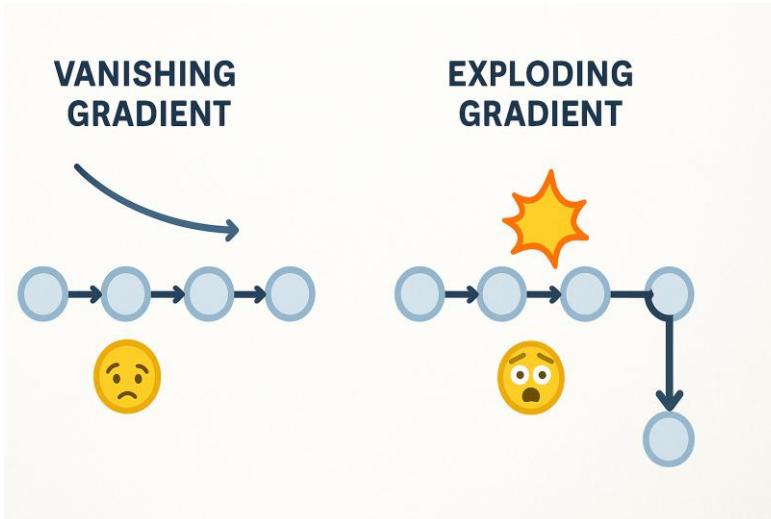
Proses ini mirip seperti memberi tahu model: "Hei, kamu salah sejaui ini. Yuk, perbaiki bobotmu sedikit demi sedikit."



Perbaikan bobot tersebut dapat mengakibatkan salah satu dari masalah ini:

- Exploding gradient problem → membuat proses estimasi model sulit mencapai konvergensi karena perubahan antar iterasi terlalu besar
- Vanishing gradient problem → membuat proses estimasi model menjadi sangat lama mencapai konvergensi karena perubahan antar iterasi sangat kecil

Permasalahan di RNN kalau rantainya panjang



Mengapa Ini Terjadi di RNN?

- Karena RNN memproses data secara berurutan (step-by-step) dan mengalirkan informasi dari waktu ke waktu.
- Setiap langkah waktu melibatkan perkalian berulang dengan matriks bobot, sehingga:
 - ✓ Kalau bobotnya kecil (misalnya < 1), dikalikan terus-terusan = semakin kecil \rightarrow vanishing.
 - ✓ Kalau bobotnya besar (misalnya > 1), dikalikan terus-terusan = semakin besar \rightarrow exploding.



Solusinya?

- Untuk vanishing gradient: Gunakan arsitektur seperti LSTM atau GRU yang dirancang untuk mengatasi masalah ini.
- Untuk exploding gradient: Gunakan teknik gradient clipping (memotong gradient agar tidak terlalu besar seperti norm clipping/Value Clipping).

★ Exploding Gradient (Gradient Meledak)

- Bayangkan kamu memberi tahu temanmu dengan **teriakan super keras** sampai dia **panik dan malah makin kacau**.

Dalam RNN:

- Kalau gradient-nya **sangat besar**, bobot model bisa **berubah secara liar**. Ini menyebabkan **angka-angka jadi tidak stabil**, model **susah belajar**, bahkan kadang bisa menghasilkan **NaN** (bukan angka).

★ Akibatnya: Model jadi **tidak bisa belajar dengan baik** atau **berperilaku aneh**.



Vanishing Gradient (Gradient Menghilang)

- Bayangkan kamu memberi tahu temanmu untuk memperbaiki kesalahan, tapi kamu **berbisik sangat pelan**. Saking pelannya, temanmu **nggak bisa dengar**. Akibatnya, dia **nggak tahu harus memperbaiki apa**.

Dalam RNN:

- Kalau jaringan RNN punya **rantai panjang** (misalnya memproses data deret waktu yang panjang), maka di bagian awal (jauh dari akhir), gradient bisa jadi **sangat kecil**. Karena itu, **neuron-neuron di awal tidak belajar apa-apapula**.

★ Akibatnya: Model susah mengingat informasi dari masa lalu.

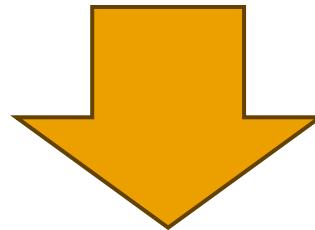
Di deep learning, gradient adalah "sinyal koreksi" yang membantu model belajar dari kesalahan. Jika sinyal ini terlalu kecil/besar, model tidak bisa memperbaiki diri.

Dampaknya : Model gagal belajar ketergantungan jangka panjang (misal: pengaruh data bulan lalu terhadap prediksi hari ini).

Dengan bahasa sederhana

RNN memang bisa mengingat data sebelumnya, **tapi hanya sebentar**.

- Jika kita punya data suhu dari 30 hari lalu,
- Mungkin RNN hanya kuat mengingat 5–10 hari terakhir,
- Sisanya "lupa" karena **masalah yang disebut *vanishing gradient*** (angka memori mengecil terus saat training)



LSTM

LSTM (*Long Short-Term Memory*)

- LSTM adalah jenis RNN yang mampu mengingat informasi jangka panjang maupun pendek, dan menghindari masalah vanishing gradient
- LSTM memiliki tiga gerbang utama:
 - **Forget Gate** – menentukan apa yang perlu dilupakan (Apa yang mau **dilupakan**)
 - **Input Gate** – menentukan informasi baru yang masuk (Apa yang mau **diingat**)
 - **Output Gate** – menentukan apa yang akan dikeluarkan ke langkah berikutnya (Apa yang mau **digunakan sebagai output saat ini**)
- Misalkan hanya ada satu fitur (misalnya: suhu harian), dan ingin memprediksi nilai ke depan.
 - Ilustrasi input: Jika lag = 3, maka: Input = [30.0, 30.5, 31.0] Output = 30.2 (nilai suhu berikutnya)
- LSTM akan memperhatikan urutan dari data sebelumnya untuk memahami polanya.

Bagaimana LSTM Bekerja?

Misalnya ada 7 data suhu harian:

[30, 31, 29, 32, 33, 34, 35]

Ingin diprediksi suhu ke-8 berdasarkan pola 7 suhu sebelumnya.

Bayangkan satu unit LSTM seperti lemari arsip pintar, yang punya:

- 1. Pintu masuk** (gate masuk / input gate): Menentukan apakah data suhu hari ini layak disimpan
- 2. Pintu hapus** (forget gate): Menentukan apakah memori lama perlu dibuang
- 3. Pintu keluar** (output gate): Menentukan apakah kita perlu menggunakan informasi dari memori sekarang untuk prediksi

LSTM akan bekerja berurutan per hari, dengan proses:

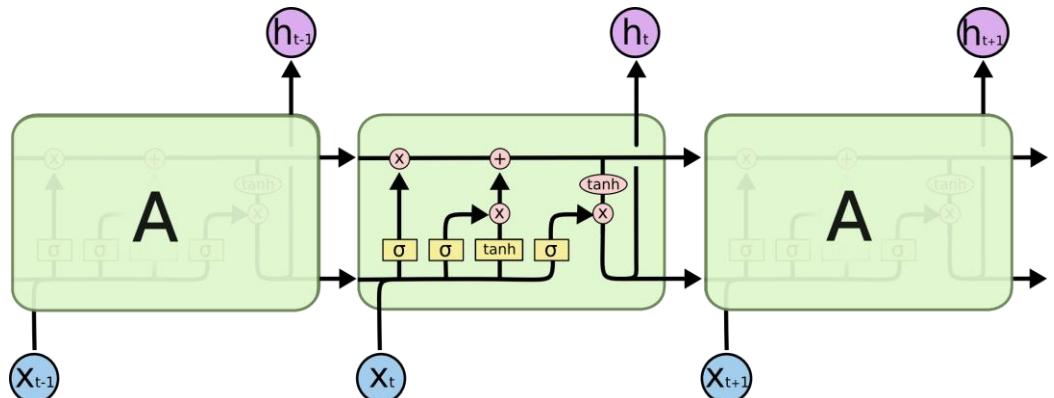
- ❑ Hari ke-1 (suhu = 30):Lemari arsip masih kosongLSTM: "Oke, simpan suhu 30 ke memori"
- ❑ Hari ke-2 (suhu = 31):LSTM: "Masukkan suhu 31 ke memori""Apakah suhu 30 masih penting? Kalau tidak, buang""Perlu keluarkan info sekarang untuk prediksi? Belum"
- ❑ ...hingga Hari ke-7:Memori diperbarui terusInfo lama yang tidak penting dibuang, info baru yang penting disimpan

Setelah Hari ke-7:

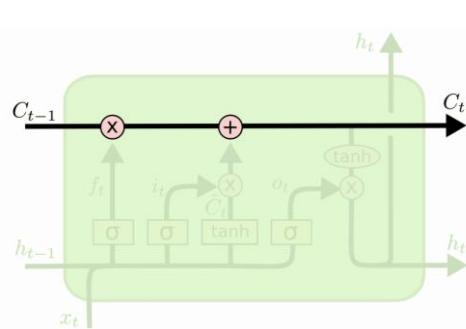
LSTM akan punya memori ringkasan tentang suhu 7 hari terakhir, dan menggunakan itu untuk prediksi suhu hari ke-8.

LSTM

Long Short Term Memory



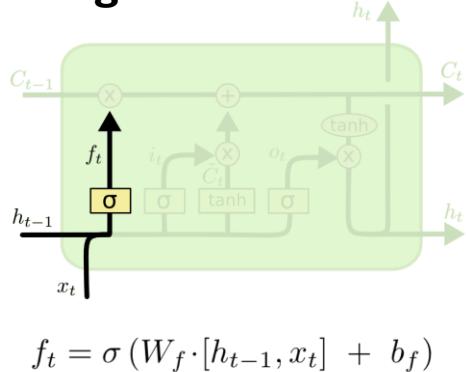
Cell State



Bagian yang menampung informasi penting yang untuk dialirkan dari waktu ke waktu. Dalam perjalannya, informasi pada cell state dapat mengalami perubahan:

- Yang tidak relevan akan dilupakan
- Informasi baru akan ditambahkan

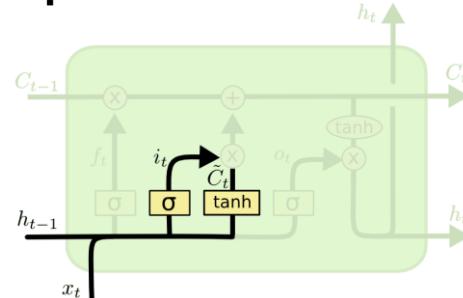
Forget Gate



Gate ini menghasilkan faktor pengganli 0 – 1 yang menentukan seberapa besar C_{t-1} untuk tetap diingat.

Besar kecilnya proporsi yang akan dingat tergantung pada output sebelumnya dan nilai saat ini.

Input Gate

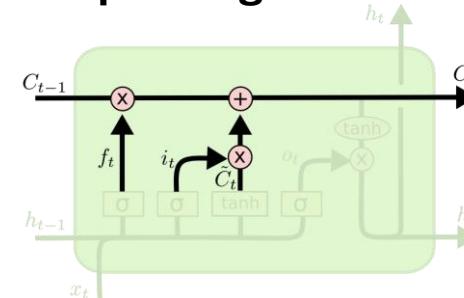


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

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

Gate ini menentukan besarnya nilai yang ditambahkan untuk disimpan pada cell state. Nilai yang disimpan adalah kombinasi antara nilai input pada waktu ke-t dan nilai output yang dialirkan dari periode t – 1

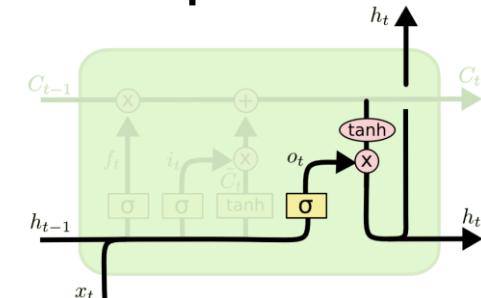
Updating Cell State



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

Pada tahapan berikutnya, dilakukan updating terhadap cell state berdasarkan nilai cell state sebelumnya (uang telah difilter oleh forget gate) dan nilai hasil input gate.

Output Gate

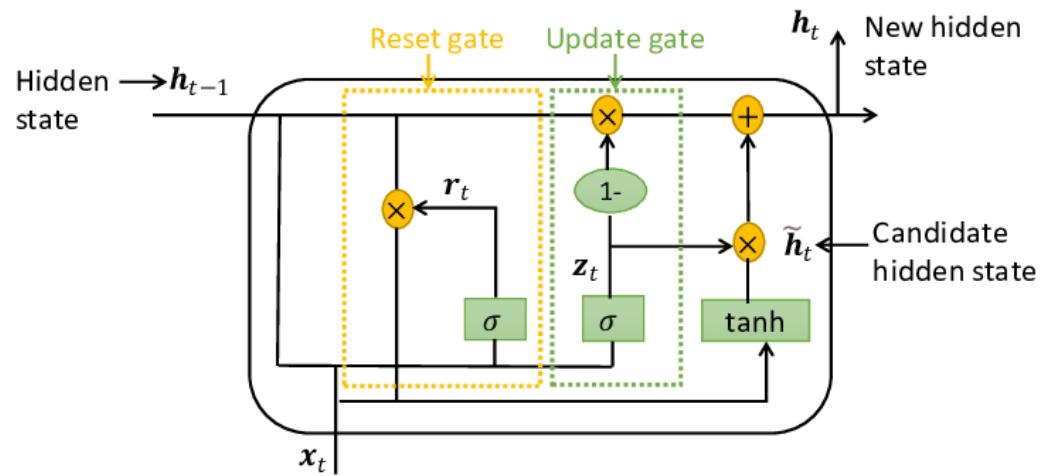


$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

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

Terakhir, adalah output yang dihasilkan yang merupakan kombinasi dari nilai cell state dan input yang didapatkan pada periode ke-t

GRU (Gated Recurrent Unit)



$$\begin{aligned}
 z_t &= \sigma(W_z x_t + W_z h_{t-1} + b_z), \\
 r_t &= \sigma(W_r x_t + W_r h_{t-1} + b_r), \\
 \tilde{h}_t &= \tanh(W_h x_t + W_h (r_t \circ h_{t-1}) + b_h) \\
 h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t,
 \end{aligned}$$

- ✓ GRU adalah versi lebih sederhana dan ringan dari LSTM.
- ✓ GRU hanya menggunakan dua gerbang:
 - ✓ **Update Gate** – mengontrol seberapa besar informasi lama yang disimpan
 - ✓ **Reset Gate** – mengontrol seberapa besar informasi lama yang dihapus
- ✓ Meski lebih sederhana, GRU sering kali memiliki performa yang mirip atau bahkan lebih baik dari LSTM, terutama pada dataset kecil/menengah.

Membuat struktur input di LSTM dan GRU

data = [30.1, 30.3, 30.2, 30.4, 30.3, 30.5, 30.6, 30.7, 30.9, 31.0]

Multi-input, One output

- Prediksi suhu hari ke-6 berdasarkan 5 hari sebelumnya.
- Misalnya lag = 5, berarti setiap input terdiri dari 5 nilai sebelumnya

Input (X)	Output (y)
[30.1, 30.3, 30.2, 30.4, 30.3]	30.5
[30.3, 30.2, 30.4, 30.3, 30.5]	30.6
[30.2, 30.4, 30.3, 30.5, 30.6]	30.7
...	...

Multi-input dan output

- Input: suhu hari ke-1 s/d ke-7
Output: suhu hari ke-8, ke-9, ke-10

Hari ke	Input: Suhu Hari ke-1 s/d ke-7	Output: Suhu Hari ke-8-10
1	30.1, 30.3, 30.2, 30.4, 30.3, 30.5, 30.6	30.7, 30.9, 31.0
2	30.3, 30.2, 30.4, 30.3, 30.5, 30.6, 30.7	30.9, 31.0, 31.2
3	30.2, 30.4, 30.3, 30.5, 30.6, 30.7, 30.9	31.0, 31.2, 31.3

Membuat struktur input di LSTM dan GRU

```
data = [30.1, 30.3, 30.2, 30.4, 30.3, 30.5, 30.6, 30.7, 30.9, 31.0]
```

Multi-input, One output

```
import numpy as np

def create_dataset(series, lag):
    X, y = [], []
    for i in range(len(series) - lag):
        X.append(series[i:i+lag])
        y.append(series[i+lag])
    return np.array(X), np.array(y)

# Contoh data suhu
data = np.array([30.1, 30.3, 30.2, 30.4, 30.3, 30.5, 30.6, 30.7, 30.9, 31.0])

# Buat dataset untuk LSTM/GRU dengan Lag 5
lag = 5
X, y = create_dataset(data, lag)

# Bentuk untuk LSTM: (samples, time_steps, features)
X = X.reshape((X.shape[0], X.shape[1], 1))

print("X shape:", X.shape) # (5, 5, 1)
print("y shape:", y.shape) # (5,)
```

Multi-input dan output

```
import numpy as np

def create_multistep_dataset(data, time_steps, horizon):
    X, y = [], []
    for i in range(len(data) - time_steps - horizon + 1):
        X.append(data[i:i+time_steps])
        y.append(data[i+time_steps:i+time_steps+horizon])
    return np.array(X), np.array(y)

# Data suhu
data = [30.1, 30.3, 30.2, 30.4, 30.3, 30.5, 30.6, 30.7, 30.9, 31.0, 31.2, 31.3]

# Buat dataset
time_steps = 7
horizon = 3

X, y = create_multistep_dataset(data, time_steps, horizon)
X = X.reshape((X.shape[0], X.shape[1], 1))
print("X (input):")
print(X)
print("\ny (target):")
print(y)
```

Studi Kasus:

3 variabel (fitur) harian:

- Curah hujan (target output)
- Suhu
- Kelembapan

Spesifikasi Model:

Input: 2 periode (hari), 3 fitur per hari
 → Ukuran input: (2, 3)

Output: 1 angka = prediksi curah hujan hari ke-3

Hidden layers:

Layer 1: LSTM dengan 16 unit

Layer 2: LSTM dengan 8 unit

Proses Kerja LSTM dengan 3 Fitur

1. Input Masuk ke Layer 1 (16 unit)

Untuk setiap waktu ($t=1$ dan 2):

- Input ke layer 1 = [curah hujan, suhu, kelembapan]
- Setiap unit LSTM menerima 3 angka sekaligus per waktu
- LSTM akan mengolah input + memori sebelumnya menggunakan gate-gate internal (input gate, forget gate, output gate)

Setelah $t=2$, setiap dari 16 neuron menghasilkan **hidden state terakhir** (atau seluruh urutan, tergantung pengaturan).

2. Hasil Layer 1 Masuk ke Layer 2 (8 unit)

- Sekarang, input ke Layer 2 bukan 3 fitur lagi, tapi 16 angka per waktu (hasil dari Layer 1).
- Layer ke-2 memproses hasil Layer 1 dari hari 1 dan 2
- Sama seperti sebelumnya, tapi lebih tinggi tingkatannya – dia menangkap pola yang lebih kompleks (kombinasi pola antar fitur dan antar waktu)

Setelah $t=2$, kita dapat 8 nilai akhir (hidden state) dari layer ke-2.

3. Layer Output (Dense Layer)

Dense layer menerima 8 angka dari layer ke-2

Lalu menggunakan fungsi linear seperti: $\hat{y} = w_1 h_1^{(2)} + w_2 h_2^{(2)} + \dots + w_8 h_8^{(2)} + b$

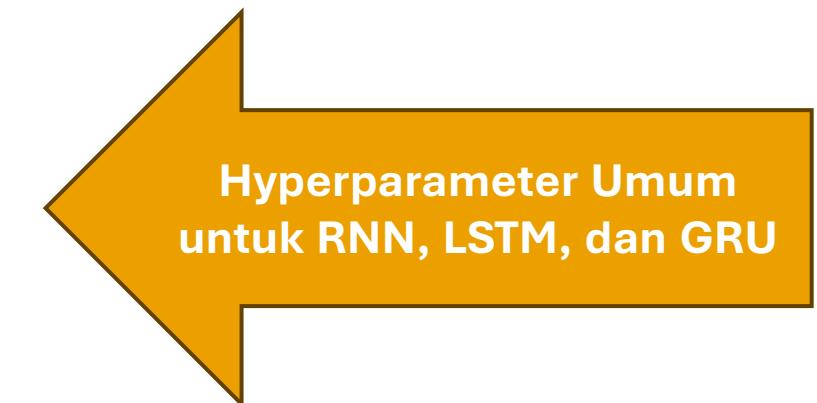
Analogi Sederhana

LSTM layer 1 = pengamat harian, mencatat suhu, hujan, dan kelembapan lalu menyaring info penting.

LSTM layer 2 = penyimpul, melihat hasil catatan dan mencoba mencari pola antar hari.

Dense layer = penjawab, menyimpulkan: "Kalau begitu, besok kemungkinan curah hujan sekian."

Hyperparameter	Fungsi
input_shape / input_size	Menentukan bentuk data input: (jumlah time steps, jumlah fitur)
units (atau hidden_size)	Jumlah neuron/unit dalam layer. Semakin besar, model bisa belajar pola lebih kompleks.
num_layers	Jumlah layer tersembunyi (hidden layers). Lebih banyak = lebih dalam
dropout	Mencegah overfitting dengan menghilangkan beberapa neuron saat training
recurrent_dropout	Sama seperti dropout, tapi berlaku untuk koneksi memori antar waktu
activation	Fungsi aktivasi untuk output (tanh, relu, dsb)
return_sequences	Apakah setiap waktu step mengeluarkan output (True) atau hanya di akhir (False)
return_state	Apakah mengembalikan state terakhir (berguna untuk stateful prediction)
batch_size	Jumlah data yang diproses sekaligus dalam 1 langkah training
learning_rate	Kecepatan model belajar. Terlalu tinggi = tidak stabil, terlalu kecil = lambat
optimizer	Algoritma optimasi (SGD, Adam, RMSProp) untuk memperbarui bobot
loss_function	Mengukur seberapa besar kesalahan prediksi dibanding nilai asli
epochs	Berapa kali seluruh data dilatih ulang
sequence_length	Berapa panjang input time steps yang digunakan sebagai input



Hyperparameter Khusus LSTM

Hyperparameter	Fungsi
recurrent_activation	Fungsi aktivasi di dalam gate LSTM (biasanya sigmoid)
unit_forget_bias	Bias di gate forget, bisa membantu stabilitas
kernel_initializer	Inisialisasi bobot input-to-hidden
recurrent_initializer	Inisialisasi bobot hidden-to-hidden (antar waktu)
bias_initializer	Inisialisasi bias untuk gate-gate LSTM
stateful	Menjaga state antar batch (berguna untuk prediksi jangka panjang)

Hyperparameter Khusus GRU

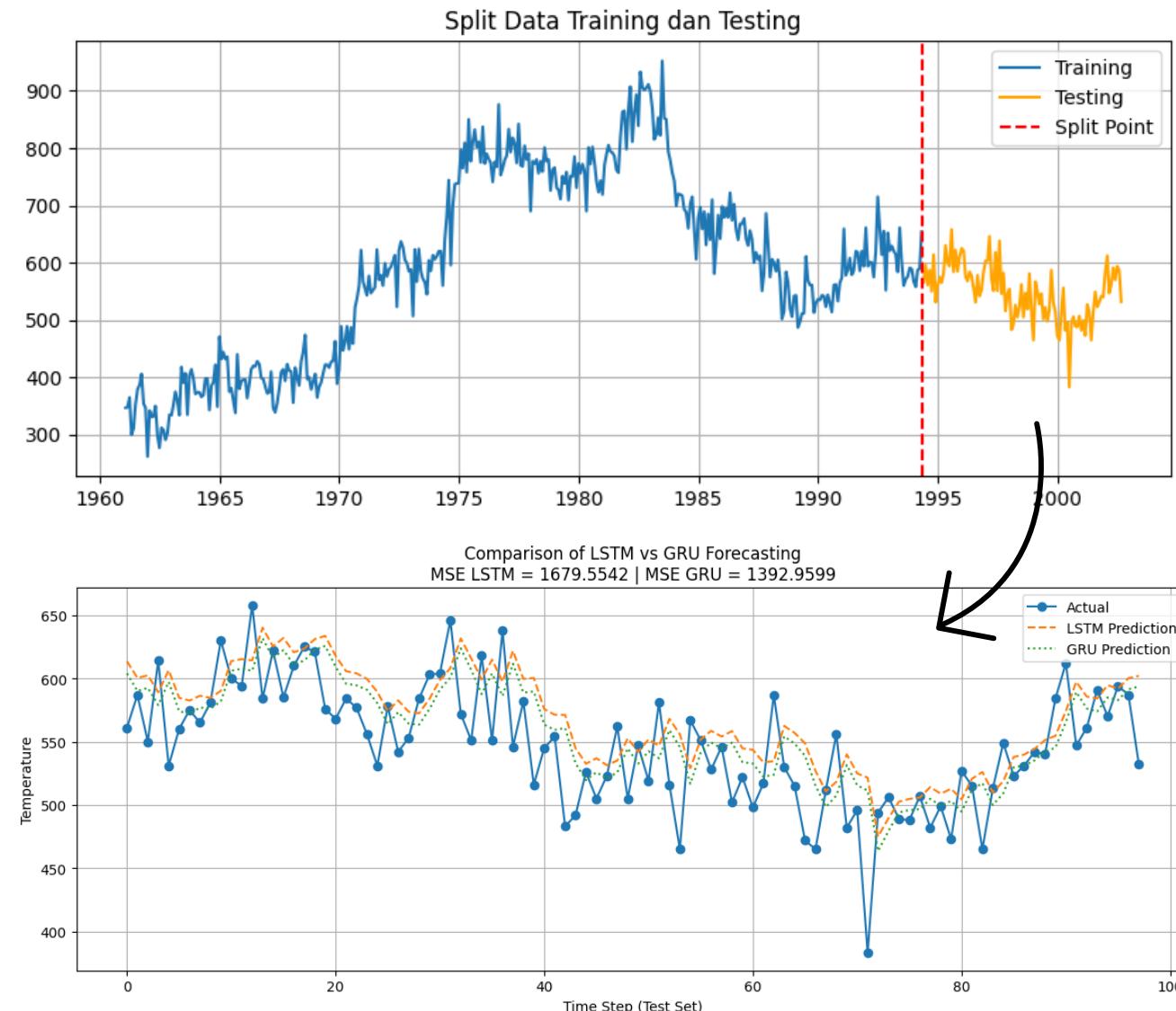
Hyperparameter	Fungsi
recurrent_activation	Fungsi aktivasi dalam gate GRU
reset_after	Urutan perhitungan reset gate (pengaruh terhadap efektivitas)
implementation	Opsi efisiensi eksekusi di backend

Monthly Unemployed Young Women in the United State with LSTM and GRU

```
# Model LSTM
model_lstm = Sequential([
    LSTM(32, activation='relu', input_shape=(lag, 1)),
    Dense(1)
])
model_lstm.compile(optimizer=Adam(0.01), loss='mse')
model_lstm.fit(X_train, y_train, epochs=100, verbose=0)

# Model GRU
model_gru = Sequential([
    GRU(32, activation='relu', input_shape=(lag, 1)),
    Dense(1)
])
model_gru.compile(optimizer=Adam(0.01), loss='mse')
model_gru.fit(X_train, y_train, epochs=100, verbose=0)
```

	ARIMA	LSTM	GRU
MAPE	13.19%	6.17%	5.53%
MSE	6471.29	1679.55	1392.96
MAE	68.00	32.52	29.38



Pendekatan utama forecasting pada DL

1. Direct Forecasting (Langsung)

Konsep:

Model dilatih untuk memprediksi langsung nilai-nilai beberapa langkah ke depan sekaligus.

Output:

Multi-output (misalnya prediksi suhu hari ke-1 sampai ke-3 sekaligus).

Contoh:

Input: suhu hari ke-1 s/d ke-7

Output: suhu hari ke-8, ke-9, ke-10

2. Recursive Forecasting (Autoregressive / One-step-ahead prediction)

Konsep:

Model memprediksi 1 langkah ke depan, lalu hasil prediksi digunakan sebagai input untuk prediksi selanjutnya.

Proses:

1. Prediksi suhu hari ke-8 pakai data hari ke-1 sampai ke-7.
2. Lalu pakai hari ke-2 sampai ke-8 (dengan prediksi ke-8) untuk prediksi ke-9.
3. Ulangi sampai horizon yang diinginkan.

3. Hybrid / Multi-Model (Direct-Recursive Hybrid)

Konsep:

Melatih model yang berbeda untuk setiap langkah ke depan (1-step model, 2-step model, dst).

Contoh:

- Model-1 → prediksi hari ke-8
- Model-2 → prediksi hari ke-9
- Model-3 → prediksi hari ke-10

Kunci utama dalam pemodelan deep learning

Kualitas dan Persiapan Data

- Garbage in, garbage out — kualitas data menentukan kualitas model.
- Harus bersih dari noise ekstrem atau outlier/anomali yang merusak pola.
- Normalisasi/standarisasi penting, terutama untuk LSTM/GRU/NN.
- Untuk time series: penting menjaga urutan waktu, tidak boleh shuffle.

📌 Tips: Visualisasikan data sebelum modeling!

Pemilihan Fitur yang Tepat

- Untuk univariate, hanya satu fitur dipakai (misal: suhu).
- Untuk multivariate, bisa gunakan fitur tambahan (kelembapan, tekanan udara).
- Fitur lag, moving average, atau transformasi (trend/seasonal) bisa menambah kekuatan model.

📌 Tools bantu: ACF/PACF, teknik feature engineering time series.

Struktur Model yang Sesuai

- Untuk data sequence seperti time series: gunakan RNN, LSTM, GRU.
- Untuk forecasting jangka panjang: bisa coba stacked LSTM atau encoder-decoder.
- Model harus menyesuaikan panjang input dan tujuan output (single-step vs multi-step).

📌 Contoh: `input_shape = (lag, 1)` untuk LSTM univariate.

Kunci utama dalam pemodelan deep learning

Strategi Training yang Efektif

- Perlu early stopping agar tidak overfitting.
- Gunakan validation set untuk tuning hyperparameter.
- Optimizer (adam, rmsprop) dan learning rate yang tepat sangat berpengaruh.

Evaluasi Model yang Jujur

- Gunakan metrik seperti **MAE**, **RMSE**, **MAPE** untuk evaluasi.
- Gunakan **data test terakhir** (bukan acak) untuk time series.
- Plot hasil prediksi vs aktual untuk melihat pola dan deviasi.

Handling Overfitting / Underfitting

- **Overfitting:** training bagus tapi gagal di test → model terlalu kompleks
- **Underfitting:** training dan test sama-sama jelek → model kurang kuat
- Solusi: dropout, lebih banyak data, model lebih ringan/kuat

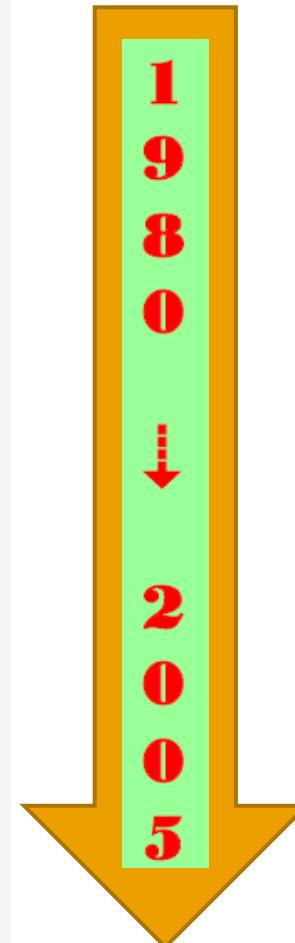
Hands-On Session #4

Forecasting dengan Deep Learning



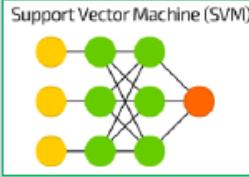
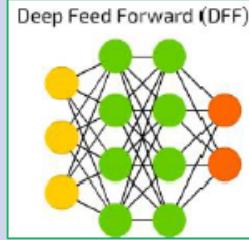
25 years of time series forecasting

- 25 years of time series forecasting
 - Introduction
 - Exponential smoothing
 - Preamble
 - Variations
 - State space models
 - Method selection
 - Robustness
 - Prediction intervals
 - Parameter space and model properties
 - ARIMA models
 - Preamble
 - Univariate
 - Transfer function
 - Multivariate
 - Seasonality
 - State space and structural models and the Kalman filter



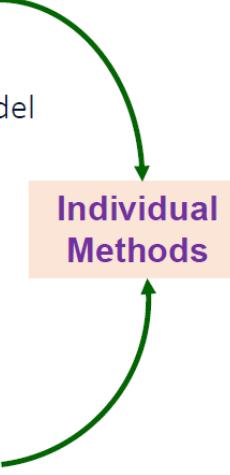
- Nonlinear models
 - Preamble
 - Regime-switching models
 - Functional-coefficient model
 - Neural nets
 - Deterministic versus stochastic dynamics
 - Miscellaneous
 - Long memory models
 - ARCH/GARCH models
 - Count data forecasting
 - Forecast evaluation and accuracy measures
 - Combining
 - Prediction intervals and densities
 - A look to the future
 - Acknowledgments
- References

Quantitative Forecasting Method in Time Series Data

Approach	Variable	Technique	Methods
 Causal	Univariate	Classical models	Multiple Regression, MARS, ...
		Machine Learning	NN, SVR, ANFIS, Deep Learning NN, ...
		Hybrid	Regression & NN, ...
	Multivariate	Classical models	Multivariate Linear Regression (MLR), ...
		Machine Learning	Multi output NN, Deep Learning NN, ...
		Hybrid	MLR & Multi output NN, ...
 Time Series	Univariate	Classical models	<u>TSR</u> , ARIMA, <u>ARIMAX</u> , ...
		Machine Learning	<u>NN</u> , SVR, ANFIS, <u>Deep Learning NN</u> , ...
		Hybrid	ARIMA & NN, ARIMAX & NN, ...
	Multivariate	Classical models	VARIMA, GSTAR, VARIMAX, GSTARX, ...
		Machine Learning	Multi output NN, Deep Learning NN, ...
		Hybrid	VARIMA & NN, GSTAR & NN, ...

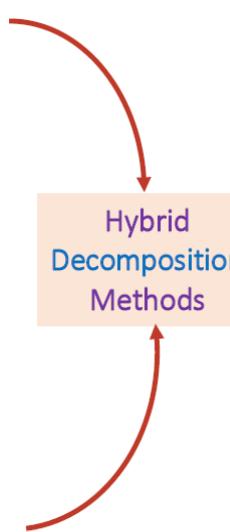
1. Exponential Smoothing: Holt-Winter's method, Holt-Winter-Taylor's method
2. Decomposition Method
3. Trend Analysis & Time Series Regression
4. ARIMA & ARIMAX: Intervention Analysis, Transfer Function model
5. Neural Networks: FFNN, RBFN, GRNN, RNN
6. Adaptive Neuro Fuzzy Inference Systems (ANFIS)
7. Multiresolution Autoregressive (MAR)
8. Wavelet Neural Networks (WNN)
9. Support Vector Regression (SVR)
10. Fuzzy Time Series (FTS)
11. Quantile Regression Autoregressive & ARIMAX
12. Quantile Regression Neural Networks (QRNN)
13. Singular Spectrum Analysis (SSA) - Linear Recurrence Relations (LRR)
14. Deep Learning Neural Networks (DLNN)

Individual Methods



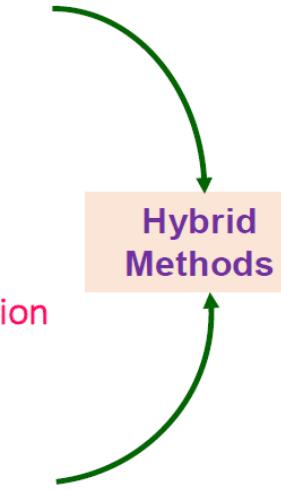
1. Decomposition method & ARIMA
2. Decomposition method & Neural Networks
3. Decomposition method & Fuzzy Time Series
4. Wavelet Transform - MODWT & Autoregressive Model
5. Wavelet Transform - MODWT & Neural Networks
6. Wavelet Transform - MODWT & ANFIS
7. Singular Spectrum Analysis & Time Series Regression
8. Singular Spectrum Analysis & ARIMA
9. Singular Spectrum Analysis & Neural Networks
10. Singular Spectrum Analysis & ANFIS
11. Singular Spectrum Analysis & Support Vector Regression
12. Singular Spectrum Analysis & Deep Learning Neural Networks

Hybrid Decomposition Methods



1. Winter's model & ARIMA
2. Winter's model & Neural Networks
3. Winter's model & Fuzzy Time Series
4. Decomposition Method & ARIMA
5. Decomposition Method & Neural Networks
6. Time Series Regression & Neural Networks
7. Time Series Regression & ANFIS
8. Time Series Regression & Support Vector Regression
9. ARIMAX & Neural Networks
10. ARIMAX & ANFIS
11. ARIMAX & Support Vector Regression
12. ARIMAX & Quantile Regression Neural Networks
13. ARIMAX & Deep Learning Neural Networks

Hybrid Methods



Future WORK

Identifikasi Model

Extended Autocorrelation Function (EACF)

Theoretical Extended ACF (EACF) for an ARMA(1,1) Model														
AR/MA	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1	x	0*	0	0	0	0	0	0	0	0	0	0	0	0
2	x	x	0	0	0	0	0	0	0	0	0	0	0	0
3	x	x	x	0	0	0	0	0	0	0	0	0	0	0
4	x	x	x	x	0	0	0	0	0	0	0	0	0	0
5	x	x	x	x	x	0	0	0	0	0	0	0	0	0
6	x	x	x	x	x	x	0	0	0	0	0	0	0	0
7	x	x	x	x	x	x	x	0	0	0	0	0	0	0

Sample EACF for Simulated ARMA(1,1) Series														
AR / MA	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	x	x	x	x	0	0	0	0	0	0	0	0	0	0
1	x	0	0	0	0	0	0	0	0	0	0	0	0	0
2	x	0	0	0	0	0	0	0	0	0	0	0	0	0
3	x	x	0	0	0	0	0	0	0	0	0	0	0	0
4	x	0	x	0	0	0	0	0	0	0	0	0	0	0
5	x	0	0	0	0	0	0	0	0	0	0	0	0	0
6	x	0	0	0	x	0	0	0	0	0	0	0	0	0
7	x	0	0	0	x	0	0	0	0	0	0	0	0	0

p = 1, q = 1
p = 2, q = 1