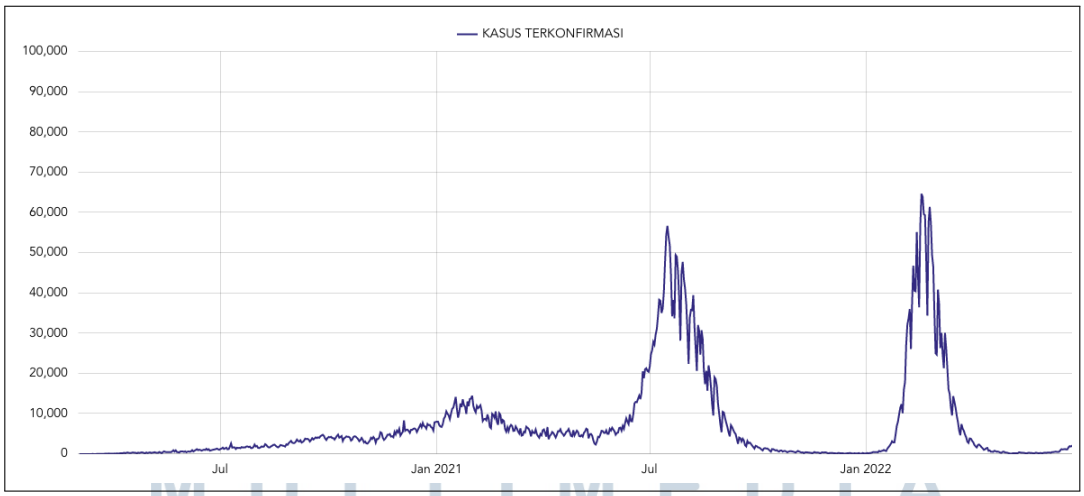
BAB 2
LANDASAN TEORI

2.1 Coronavirus disease (COVID-19)

Coronavirus disease (COVID-19) merupakan suatu penyakit menular yang disebabkan oleh virus SARS-CoV-2 [1]. Virus ini pertama kali muncul di Wuhan, China pada tahun 2019. Virus ini dapat menyebar dengan cepat melalui saluran pernapasan orang yang terinfeksi. Menurut WHO [1], orang yang terinfeksi virus ini akan merasakan penyakit pernafasan ringan hingga sedang dan akan sembuh tanpa memerlukan perawatan khusus. Umumnya orang yang memiliki penyakit bawaan seperti penyakit kardiovaskular, diabetes, penyakit pernapasan kronis, atau kanker lebih rentan terkena penyakit serius. Sayangnya, setiap orang dapat terkena virus ini dan menyebabkan sakit kronis hingga meninggal dunia.

Gambar 2.1 menunjukan jumlah kasus terkonfirmasi di Indonesia. Virus ini pertama kali masuk di Indonesia pada tanggal 02 Maret 2020. Kemudian virus B.1.617.2 (*Delta*) masuk pada bulan Mei 2021, dan B.1.1.529 (*Omicron*) pada bulan Desember 2021. Hingga saat ini penyebaran virus masih terus terjadi. Oleh karena itu, dibutuhkan model peramalan untuk memprediksi ke depan.



Gambar 2.1. Kasus Terkonfirmasi positif di Indonesia
Sumber: covid19.go.id [11]

2.2 Peramalan Deret Waktu

Peramalan digunakan untuk mendapatkan informasi, dan digunakan sebagai acuan untuk mengambil keputusan dalam perancangan strategi jangka panjang [12]. Menurut Hyndman dkk. [12], model peramalan yang akurat sangat bergantung dengan data yang tersedia. Jika jumlah data tidak tersedia dan tidak relevan, maka metode peramalan kualitatif harus digunakan. Sebaliknya, jika data tersebut tersedia dan relevan, maka metode kuantitatif digunakan. Salah satu contoh metode peramalan kuantitatif adalah deret waktu. Menurut Hyndman dkk [12], deret waktu memiliki kriteria seperti informasi numerik tentang masa lalu, dan memiliki pola yang mirip dimasa depan. Deret waktu memiliki komposisi seperti *trend*, *seasonality*, dan *cyclic*.

1. *Trend* terjadi ketika ada kenaikan atau penurunan jangka panjang dalam data deret waktu.

2. *Seasonality* terjadi ketika deret waktu dipengaruhi oleh faktor musiman seperti seminggu, sebulan, maupun setahun. *Seasonality* terjadi ketika data menunjukan naik dan turun dan bukan dari frekuensi tetap.

3. *Cyclic* biasanya disebabkan oleh fluktuasi kondisi ekonomi, dan sering dikaitkan dengan siklus bisnis. *Cyclic* bisanya terjadi oleh faktor yang tak tentu.

2.3 Facebook Prophet

Facebook Prophet merupakan sebuah model yang dibuat oleh Facebook Ai Researcher untuk menyelesaikan masalah yang sering terjadi dalam melakukan peramalan. Masalah utama dari peramalan deret waktu adalah memilih metode yang tepat untuk menyelesaikan masalah deret waktu. Kedua dibutuhkan keahlian dalam merancang model deret waktu. Oleh karena itu, Facebook Ai Researcher mengembangkan sebuah *framework* yang dapat menghasilkan kualitas peramalan yang bagus. Model Facebook Prophet menggunakan metode *analyst-in-the-loop* yang dapat membuat praktisi deret waktu melakukan automasi terhadap model. Jadi jika terjadi *error* pada model, Facebook Prophet akan memberikan *feedback* kepada praktisi untuk melakukan inspeksi pada model. [6]. Penelitian Taylor dan Letham [6], mengatakan *framework* Facebook Prophet menggunakan *decomposable time*

series atau membagi *time series* menjadi tiga model model utama: *trend*, *seasonality*, dan *holidays*. Model ini mirip dengan metode *Generalized Additive Model* (GAM), yang dimana *output* dari hasil fungsi dijumlahkan sehingga memperoleh hasil prediksi. Berikut formulasi dari Facebook Prophet:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon(t) \tag{2.1}$$

Keterangan:

g(t): fungsi *trend* dimana model *non periodic* dapat berubah dalam nilai *time series*.

s(t): fungsi *seasonality*, mewakili perubahan periode musiman seperti (mingguan, bulanan, dan tahunan)

h(t): fungsi *holidays*, menunjukkan efek liburan yang mempengaruhi *times series* dalam sehari ataupun beberapa hari.

ε(t): fungsi *error term*, menunjukan perubahan *idiosyncratic* yang tidak di akomodasi oleh model.

2.3.1 Trend atau Growth

Trend dalam Facebook Prophet menggunakan dua fungsi yang bergantung pada model *growth*. Jika model *growth logistic* atau *non-linear* maka menggunakan model *saturating growth*, sedangkan jika model *growth linear* maka menggunakan model *piece-wise*.

A. Saturating Growth (Non-linear)
   Model ini digunakan apabila data yang diharapkan akan berkembang lagi. Model *growth* biasanya mirip dengan perkembangan dalam ekosistem natural. Pertumbuhan biasanya tidak konstan atau *non-linear*, oleh karena itu perkembagan dibatasi dengan memasukkan *ceiling*, dan *floor*.

B. Piece-wise Growth (Linear)
   Model ini digunakan apabila data adalah linear atau konstan. Model ini akan menunjukan perubahan secara *piece-wise*, yang dimana apabila terdapat perubahan dalam *trend*, model akan menambahkan titik ke dalam *trend*.

2.3.2 Seasonality

Seasonality dalam model Facebook Prophet adalah sebuah fungsi *fourier* untuk memprediksi mingguan, bulanan, dan tahunan. *Fourier series* dipakai untuk memberikan model periodik yang fleksibel. User dapat mengatur efek seasonality dalam periode N, apabila periode N ditingkatkan maka pola seasonal akan berubah dengan cepat, walaupun dapat meningkatkan resiko overfitting [6]. Secara *default* Facebook Prophet memberikan order N=10 untuk tahunan, dan order N=3 untuk mingguan [6].

2.3.3 Holidays dan Event

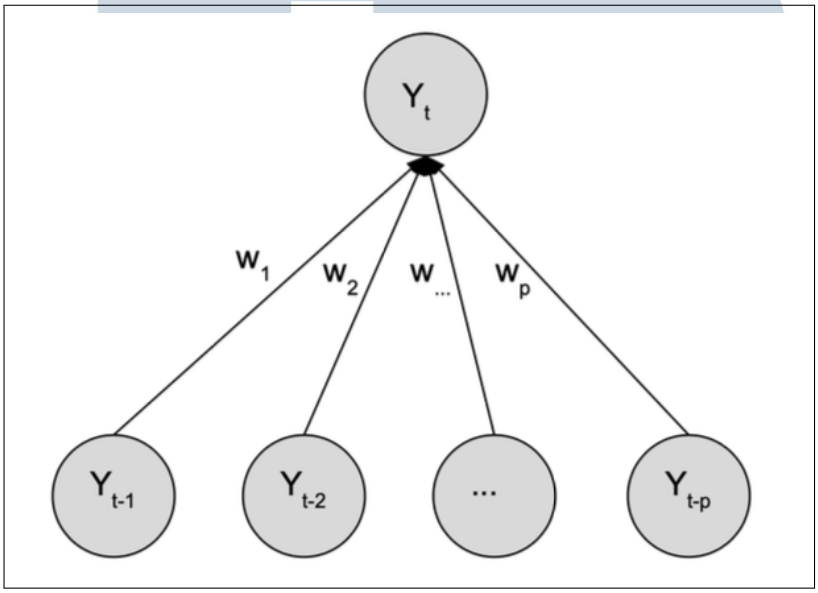
Efek *holidays* dan *event* sangat mempengaruhi efek deret waktu bisnis, dan kadang tidak mengikuti pola periodik. Jadi efek *holidays* tidak membentuk *smooth cycle*. Facebook Prophet menyediakan *custom list* untuk *event* sebelum dan ke depannya, dan diidentifikasi dengan nama unik. Kemudian Facebook Prophet menggabungkan *list holiday* ke dalam model dengan asumsi efek *holidays* adalah *independent*.

Dalam *holidays* dan *event* praktisi dapat mengatur *upper window* dan *lower window*. Untuk *upper window* adalah hari yang ditujukan setelah *event* atau *holidays* berlangsung. Sebaliknya *lower window* adalah hari yang ditujukan sebelum *event* atau *holidays* berlangsung. Seperti apabila ingin menambahkan hari malam natal maka *lower window* dapat diatur menjadi 1 [6].

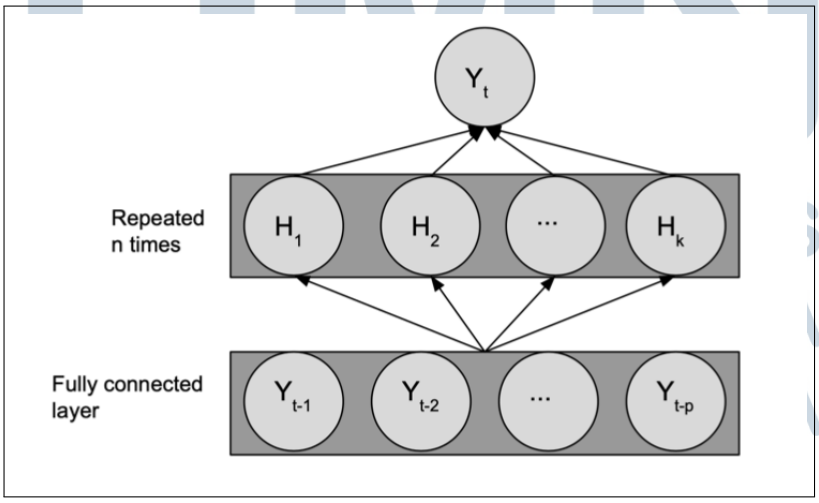
2.4 AR-Net

AR-Net adalah sebuah model yang dikembangkan oleh Triebe dkk. [13], model ini bertunjuan untuk menyelesaikan masalah skalabilitas yang terdapat pada *Classical Autoregression* (AR) model. Masalah tersebut terjadi ketika berhadapan dengan dependensi jarak jauh, *Classic Autoregression* (AR) model bisa menjadi lambat ketika data di-*fit* dengan skala data yang besar. Model *sequence-to-sequence* seperti *Recurrent Neural Network* (RNN) dapat mengatasi masalah ini, namun Triebe [13] menjelaskan bahwa *Recurrent Neural Network* (RNN) dinilai terlalu kompleks untuk model *time series* biasa dan interpretasi data akan berkurang. Maka dari itu Triebe dkk. [13], membuat sebuah model AR-Net dengan menggabungkan *Classical AR* dengan *feed-forward neural network*. Model ini mampu meningkatkan kapabilitas dan juga dapat memudahkan interpretasi seperti

Classical AR. Dalam penelitian Triebe [13], AR-Net model dapat meniru proses tradisional AR dengan *neural network*. AR-Net didesain agar parameter dari layer pertama sama dengan AR-*coefficient* (2.2). AR-Net juga dapat di-*extend* menjadi beberapa *hidden layer* agar mendapatkan tinggkat akurasi yang tinggi (Gambar 3). Dimana arsitektur AR-Net dengan *lag* $y_(t-1),\ldots,y_(t-p)$ adalah sebagai *input* dan y_t adalah sebagai *target*, dan $w_t,\ldots,w_p$ adalah *weight*, dan $H_1,\ldots,H_k$ adalah jumlah *neuron* dalam *n hidden layer*.

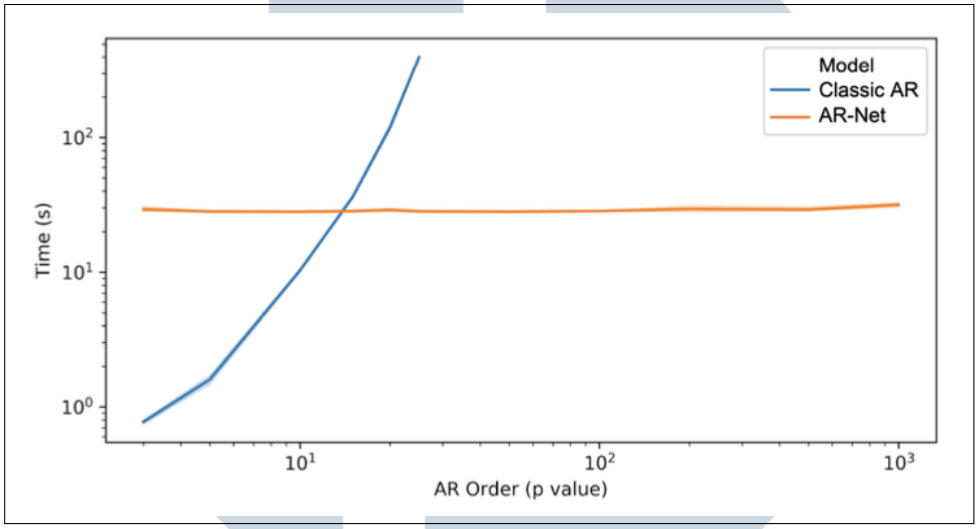


Gambar 2.2. AR-Coefficient sama dengan arsitektur neural network
Sumber: Triebe dkk. [13]



Gambar 2.3. AR neural network dengan n hidden layer berukuran k
Sumber: Triebe dkk. [13]

Hasil penelitian Triebe dkk. [13], AR-Net mampu mengatasi skalabilitas yang terdapat pada Classical *Autoregression* (AR). AR-Net dapat menangani AR-*order* dengan jumlah yang banyak dengan *time complexity* yang linear dibandingkan dengan *Classical AutoRegression*(AR) yang *quadratic*.



Gambar 2.4. Time complexity AR-Net dengan Classical AR

Sumber: Triebe dkk. [13]

2.5 Classical Autoregression

Dalam model *Classical Autoregression* (AR), untuk memprediksi variabel yang diinginkan dapat menggunakan kombinasi linear dari variabel masa lalu. Istilah *autoregression* adalah regresi variabel terhadap dirinya sendiri [12]. Jadi *autoregressive* sebagai p bisa ditulis sebagai berikut:

$$y_t = c + \phi_1 y_{t-1} + ... + \phi_p y_{t-p} + \varepsilon_t \tag{2.2}$$

Keterangan:

c: konstan

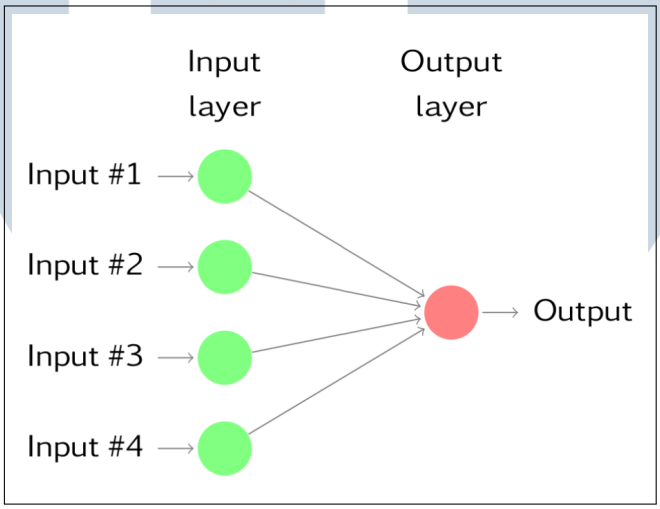
$y_(t-p)$: p adalah jumlah *lag* yang digunakan untuk memprediksi y_t.

ϕ_p: adalah *weight* yang akan dikali dengan jumlah *lag* p. Jadi $\phi_p y_{(t-p)}$ dapat dikatakan sebagai AR-*coefficient*.

ε_t: adalah banyaknya jumlah *noise*.

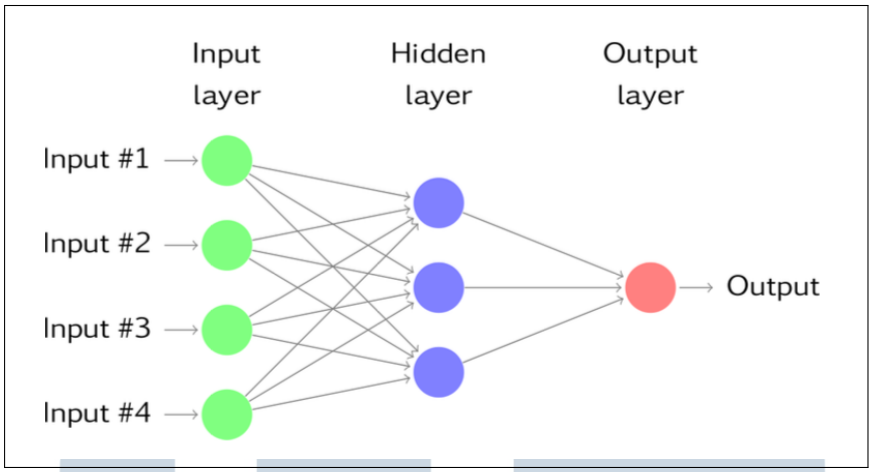
2.6 Neural Network

Neural network sederhanya adalah sebuah jaringan saraf tiruan yang terdiri dari beberapa *layer* linear dan *non-linear*, yang di-*fit* ke dalam *target* dengan menggunakan *stochastic gradient descent* (SGD) sebagai *loss term*. Menambah *layer* sebagai *"deep" neural network* memungkinkan untuk memodelkan hubungan *non-linear* yang kompleks antara variabel respons dan variabel prediktornya. Prediktor atau *input* biasanya membentuk *bottom layer* dan *forecast* atau *output* biasanya membentuk *top layer*. [13]



Gambar 2.5. Neural network tanpa hidden layer.
Sumber: Triebe dkk. [13]

Menurut Triebe dkk. [13], bentuk sederhana dari *neural network* tidak memiliki *hidden layers*, sehingga membuat model ini mirip dengan *linear regression*. *Coefficient* yang ada pada prediktor bisa disebut sebagai *weights* dan digunakan untuk mengambil *input* dengan kombinasi *linear*. Berbeda dengan saat menambah *hidden layer*, *neural network* menjadi *non-linear*. Model ini biasanya disebut sebagai *multilayer feed-forward network*, dimana setiap *node layer* menerima *input* dari *layer* sebelumnya, *input* tersebut digabung menggunakan *weight* kombinasi *linear*. Hasil tersebut diubah oleh fungsi *nonlinear* seperti *sigmoid*, atau *relu* sebelum dijadikan *input* ke *layer* selanjutnya.



Gambar 2.6. Neural network dengan satu hidden layer.

Sumber: Triebe dkk. [13]

2.7 Neural Prophet

Model Neural Prophet tidak jauh berbeda dengan model Facebook Prophet. Model ini dikembangkan untuk mempermudah peramalan dengan mempertahankan tujuan dari Facebook Prophet seperti, interpretasi, konfigurasi. Neural Prophet menyediakan lebih banyak ekstensibilitas seperti *automatic differencing* dengan PyTorch sebagai *backend*. Neural Prophet dibuat sepenuhnya modular, sehingga ke depannya dapat diskalakan untuk menambah beberapa komponen baru. [7]

Dalam penelitian Triebe dkk. [7], konsep utama dari model Neural Prophet adalah *modular composability*, yang dimana setiap modul akan berkontribusi menjadi *additive component* yang kemudian dipakai untuk peramalan. Sebagian komponen dapat dikonfigurasi untuk diskalakan berdasarkan tren sehingga membentuk efek *multiplicative*. Setiap modul memiliki *input* dan proses model tersendiri. Akan tetapi semua modul harus menghasilkan h *output*, dimana h didefinisikan sebagai *number of steps* yang akan diramalkan kedepannya secara bersamaan. Ini adalah tambahan sebagai *predicted value* $\hat{y}_t, \ldots, \hat{y}_{(t+h-1)}$ untuk *future value* $y_t, \ldots, y_{(t+h-1)}$. Jika model hanya *time dependent*, jumlah acak *number of forecasts*dapat dilakukan. Berikut formulasi yang setara dengan prediksi *one-step ahead* dengan h=1.

$$\hat{y}_t = T(t) + S(t) + E(t) + F(t) + A(t) + L(t) \tag{2.3}$$

Keterangan:

T(t): Fungsi *trend* dalam waktu t.

S(t): Fungsi *seasonality* dalam waktu t.
E(t): Fungsi *event* dan *holidays* dalam waktu t.
F(t): Fungsi regressi dalam waktu t untuk variable eksogen yang sudah diketahui dimasa depan.
A(t): Fungsi *autoregression* dalam waktu t berdasarkan pengamatan sebelumnya.
L(t): Fungsi regresi dalam waktu t untuk pengamatan *lag* dari variabel eksogen.
$\hat{y}_t$: hasil prediksi.

Semua komponen modul tersebut dapat dikonfigurasi dan digabungkan untuk membuat model. Jika semua modul tidak dipakai, parameter statis akan di fit sebagai komponen tren. Secara default hanya model trend dan *seasonality* yang diaktifkan. [7]

2.8 Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE) menggunakan adalah sebuah sebuah *scale dependent error* dimana nilai *error* bergantung dengan nilai skala dan tidak dapat dibandingkan dengan skala yang berbeda. Untuk menghitung *Root Mean Square Error* (RMSE) adalah dengan melakukan *square root* dari nilai rata-rata prediksi dan aktual.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n}} \tag{2.4}$$

Keterangan:
n: banyaknya data
y_i: nilai aktual
$\hat{y}_i$: nilai prediksi

2.9 Mean Absolute Error (MAE)

Mean Absolute Error (MAE) digunakan untuk mengukur nilai rata rata antara nilai aktual *time series* dan hasil nilai prediksi. MAE termasuk dalam *scale dependent error* dimana nilai *error* tergantung dengan nilai skala dan tidak dapat dibandingkan dengan nilai skala yang berbeda. Untuk menghitung *Mean Absolute Error* (MAE) adalah dengan mengurangi nilai rata-rata dengan nilai prediksi, lalu

dilakukan *absolute* terhadap nilai yang negatif. Kemudian dijumlahkan secara keseluruhan, lalu dibagi dengan banyaknya data n.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \tag{2.5}$$

Keterangan:

$\hat{y}_i$: nilai prediksi

y: nilai aktual

n: banyaknya data