BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

Tinjauan pustaka ini mencakup beberapa konsep dan penelitian penting yang menjadi dasar penelitian ini, meliputi prediksi gempa dan seismologi, model prediksi *Time series*, evaluasi model, pengembangan aplikasi web, visualisasi data, serta metode *hosting* dan *Deployment*. Setiap topik tersebut memberikan landasan teoritis dan teknis yang relevan untuk memahami konteks dan metode penelitian yang diterapkan

2.1.1 Gempa Bumi

2.1.1.1 Pengertian Gempa Bumi

Gempa bumi adalah fenomena alam yang terjadi akibat pelepasan energi secara tiba-tiba dari dalam bumi, yang menyebabkan getaran pada permukaan. Energi ini biasanya dilepaskan sebagai akibat dari pergerakan lempeng tektonik yang membentuk kerak bumi. Ketika lempeng-lempeng ini bergerak, mereka dapat saling bertumbukan, meluncur di atas satu sama lain, atau terpisah, dan interaksi ini menimbulkan ketegangan yang kemudian dilepaskan dalam bentuk gelombang seismik.

Proses pelepasan energi inilah yang memicu getaran di permukaan bumi, yang disebut sebagai gempa bumi. Gempa bumi sering kali diklasifikasikan

berdasarkan penyebab dan kedalamannya. Terdapat beberapa jenis gempa, yaitu gempa tektonik, vulkanik, dan runtuhan, di mana gempa tektonik adalah yang paling umum dan terjadi akibat pergerakan lempeng tektonik (Ruyani, 2023).

2.1.1.2 Penyebab Gempa Bumi

Gempa bumi terutama disebabkan oleh aktivitas tektonik, yaitu pergerakan dan interaksi antar lempeng tektonik di bawah permukaan bumi. Bumi terbagi menjadi beberapa lempeng besar yang disebut lempeng tektonik, yang terus bergerak sangat lambat di atas lapisan mantel bumi (Ruyani, 2023). Ada tiga jenis pergerakan lempeng yang dapat menyebabkan gempa yaitu :

1. *Konvergen*

Ketika dua lempeng bertabrakan, salah satunya biasanya menyelam ke bawah yang lain (zona subduksi), menyebabkan tekanan besar yang pada akhirnya dilepaskan dalam bentuk gempa besar.

2. *Divergen*

Ketika dua lempeng bergerak menjauh satu sama lain, mereka menciptakan celah di mana magma dapat naik dan membentuk gempa vulkanik.

3. *Transform*

Ketika dua lempeng bergeser atau bergerak saling mendatar di sepanjang satu sama lain, tekanan yang terakumulasi di garis patahan dapat menyebabkan gempa.

Di wilayah seperti Selat Sunda, gempa bumi disebabkan oleh interaksi antara Lempeng Indo-Australia yang menyelam di bawah Lempeng Eurasia dalam proses

yang dikenal sebagai subduksi. Tekanan yang terjadi di zona subduksi ini dapat menyebabkan gempa bumi besar, yang pada gilirannya juga dapat memicu tsunami.

2.1.1.3 Seismologi dan Data Seismik

Seismologi merupakan cabang geofisika yang berfokus pada studi tentang getaran atau goncangan yang terjadi di permukaan bumi akibat pelepasan energi dari dalam bumi. Biasanya, getaran ini disebabkan oleh aktivitas tektonik, seperti pergerakan lempeng bumi, atau aktivitas vulkanik, yang memicu gempa bumi. Secara sederhana, seismologi adalah ilmu tentang gempa dan proses-proses geologis yang berhubungan dengannya (Ronoatmojo & Burhannudinnur, 2021).

Dalam penelitian ini, seismologi menjadi dasar teoritis penting yang membantu memahami mekanisme terjadinya gempa, terutama di wilayah yang rawan seperti Selat Sunda. Pengetahuan ini digunakan untuk mengenali karakteristik gempa seperti *Magnitudo*, frekuensi, kedalaman, dan pola-pola historisnya, yang penting dalam pengembangan model prediksi.

Data seismik merupakan data yang diperoleh dari berbagai instrumen pengukur getaran bumi, seperti seismometer, yang mampu merekam gelombang seismik yang dihasilkan oleh gempa bumi. Data ini mencakup informasi penting mengenai waktu kejadian, intensitas gempa (*Magnitudo*), kedalaman sumber gempa, dan lokasi episenter gempa. Pengolahan data seismik sangat berguna dalam analisis dan prediksi gempa, serta dalam memetakan wilayah-wilayah yang berisiko tinggi terhadap goncangan kuat (Ronoatmojo & Burhannudinnur, 2021).

Melalui analisis data seismik yang komprehensif, seismolog dapat memahami pola historis gempa dan memprediksi kemungkinan terjadinya gempa di masa depan. Hal ini berperan besar dalam pengembangan sistem peringatan dini dan strategi mitigasi bencana di daerah rawan gempa.

2.1.2 ARIMA

2.1.2.1 Pengertian ARIMA

ARIMA, atau *Autoregressive Integrated Moving Average*, adalah metode peramalan yang membuat prediksi berdasarkan pola yang ada dalam data historis. Model ini menggunakan data masa lalu dari variabel target untuk menghasilkan prediksi jangka pendek yang akurat, tanpa memerlukan variabel tambahan. Karena sifatnya yang tidak melibatkan variabel independen, ARIMA cocok untuk analisis yang hanya bergantung pada data masa lalu dari variabel utama (Rusyida, 2022).

Dalam praktiknya, ARIMA banyak digunakan di berbagai bidang, seperti peramalan ekonomi, analisis anggaran, pengendalian kualitas dan manajemen proses, serta analisis sensus. Sifatnya yang fleksibel dan ekonomis membuat ARIMA menjadi salah satu model populer dalam analisis deret waktu. Misalnya, dalam peramalan ekonomi dan keuangan, model ini dapat digunakan untuk menganalisis tren harga saham atau produk keuangan lainnya.

Pendekatan yang digunakan dalam ARIMA dikenal sebagai metode *Box-Jenkins*. Metode ini memungkinkan prediksi berbagai variabel secara cepat, mudah, dan efisien, hanya membutuhkan data dari variabel yang diprediksi tanpa perlu

melibatkan data tambahan dari variabel lain. Hal ini membuat metode *Box-Jenkins* sangat cocok untuk pemodelan prediksi yang sederhana namun akurat(Rusyida, 2022).

Dengan kemampuannya untuk mengidentifikasi pola jangka pendek dalam data historis, ARIMA menjadi alat yang efektif dalam peramalan yang membutuhkan ketepatan dalam waktu singkat. Model ini juga cocok untuk data yang menunjukkan sifat stasioner atau yang dapat distasionerkan melalui *differencing*, sehingga hasil prediksinya dapat lebih stabil.

2.1.2.2 Komponen ARIMA

Model ARIMA terdiri dari tiga komponen utama yang saling bekerja sama untuk menangkap pola dalam data *Time series*, yaitu *Autoregressive* (AR), *Integrated* (I), dan *Moving Average* (MA) (Sudirman, 2023).

1. *Autoregressive* (AR)

Komponen ini mempertimbangkan hubungan antara nilai observasi saat ini dengan nilai-nilai sebelumnya. AR menggunakan persamaan regresi untuk memprediksi nilai sekarang berdasarkan kombinasi linier dari observasi sebelumnya.

Model AR dengan orde p dapat dituliskan sebagai :

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \epsilon t \qquad (1)$$

Berdasarkan rumus diatas dapat disimpulkan bahwa

a. Y_t = Nilai variabel pada waktu t
b. c = Konstanta

c. $\phi_1,\phi_2,\ldots,\phi_p$ = Koefisien dari *lag* (nilai masa lalu)

d. ϵ_t = *Error* atau noise pada waktu t

2. *Integrated* (I)

Komponen ini mengatasi tren atau perubahan dalam data *Time series* yang dapat mengganggu kestabilan prediksi. Dengan menerapkan proses *differencing*, data diubah menjadi bentuk yang stasioner, di mana rata-rata dan varian tetap stabil sepanjang waktu.

Model *Integrated* dengan orde d dapat dituliskan sebagai:

$$Yt' = Yt - Yt - 1 \quad (2)$$

3. *Moving Average* (MA)

Komponen MA memperhitungkan hubungan antara observasi saat ini dan kesalahan atau gangguan dari prediksi sebelumnya. Metode ini menggunakan nilai *residual* dari komponen AR untuk mengoreksi prediksi, yang membantu mengurangi dampak dari gangguan atau kesalahan di masa lalu.

Model MA dengan orde q dapat ditulis sebagai:

$$\mathrm{Yt} = \mathrm{c} + \epsilon\mathrm{t} + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q} \quad (3)$$

Berdasarkan rumus diatas dapat disimpulkan bahwa

a. $\theta_1,\theta_2,\ldots,\theta_q$ = Koefisien dari *error* sebelumnya

b. ϵ_{t-i} = *Error* pada periode sebelumnya

Dengan menggabungkan ketiga komponen ini, ARIMA dapat memodelkan data *Time series* secara menyeluruh, menghasilkan prediksi yang akurat berdasarkan pola historis yang ada dalam data. Parameter ARIMA ditentukan oleh tiga nilai:

a. p = Orde komponen AR

b. d = Orde *differencing* (komponen I)

c. q = Orde komponen MA

Model ARIMA umum dirumuskan sebagai:

$$Yt = c + \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \epsilon t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} \quad (4)$$

Identifikasi parameter dilakukan dengan menganalisis ACF (*Autocorrelation Function*) dan PACF (*Partial Autocorrelation Function*).

2.1.2.3 ACF dan PACF

Autocorrelation Function (ACF) dan *Partial Autocorrelation Function* (PACF) adalah dua alat analisis penting dalam pemodelan deret waktu yang sering digunakan untuk menentukan parameter terbaik dalam model ARIMA. Keduanya membantu mengidentifikasi pola keterkaitan antara nilai dalam data *Time series*, yang mendasari proses prediksi jangka pendek (Rusyida, 2022; Sudirman, 2023). *Autocorrelation Function* (ACF) mengukur kekuatan hubungan antara nilai observasi saat ini dan nilai-nilai pada berbagai interval waktu sebelumnya, yang dikenal sebagai *lag*. Pada setiap *lag*, ACF menunjukkan korelasi antara nilai saat ini dengan nilai sebelumnya dalam deret waktu, memberikan gambaran seberapa jauh pengaruh masa lalu masih signifikan terhadap nilai sekarang. Sebagai contoh, ACF pada *lag*-1 mengukur korelasi antara nilai observasi saat ini dengan satu periode sebelumnya, ACF pada *lag*-2 dengan dua periode sebelumnya, dan seterusnya. ACF membantu menentukan orde komponen *Moving Average* (MA)

yang terbaik untuk menangkap pola fluktuasi dalam data *Time series*. Rumus ACF dijabarkan sebagai berikut.

$$ACF(k) = \frac{\sum_{t=1}^{N-k}(Yt - \bar{Y})\,(Y_{t+k} - \bar{Y})}{\sum_{t=1}^{N}(Yt - \bar{Y})\text{^}2}$$

Partial Autocorrelation Function (PACF), berbeda dengan ACF, mengukur korelasi antara nilai observasi saat ini dengan nilai pada *lag* tertentu setelah mengendalikan pengaruh dari *lags* antara keduanya. PACF memberikan informasi tentang pengaruh langsung atau parsial dari *lag* tertentu terhadap nilai saat ini tanpa campur tangan dari nilai *lag* lainnya. PACF sering digunakan untuk menentukan parameter dalam komponen *Autoregressive* (AR), di mana jumlah *lag* signifikan yang terdeteksi oleh PACF memberi panduan tentang seberapa banyak nilai masa lalu yang perlu dimasukkan dalam model untuk menghasilkan prediksi yang akurat. Rumus PACF dapat dijabarkan sebagai berikut.

$$PACF(k) = Correlation(Y_t, Y_{t-k} \mid Y_{t-1}, Y_{t-2}, \dots, Y_{t-k+1}) \quad (5)$$

Melalui analisis ACF dan PACF, parameter yang sesuai untuk model ARIMA dapat ditentukan. Jika grafik ACF menunjukkan penurunan setelah sejumlah *lag* tertentu, ini menunjukkan jumlah *lag* yang diperlukan untuk komponen MA. Begitu pula, cut-off pada grafik PACF menunjukkan jumlah *lag* untuk komponen AR. Dengan demikian, ACF dan PACF memainkan peran penting dalam menyesuaikan model ARIMA agar dapat menangkap pola kompleks dalam data *Time series*.

2.1.2.4 Implementasi ARIMA

Dalam praktiknya, penerapan model ARIMA melibatkan beberapa langkah, termasuk

1. Identifikasi

   Menganalisis data untuk menentukan urutan AR dan MA yang sesuai serta tingkat integrasi yang dibutuhkan untuk membuat data stasioner. Identifikasi parameter dilakukan dengan menganalisis ACF (*Autocorrelation Function*) dan PACF (*Partial Autocorrelation Function*).

2. Estimasi Parameter

   Menghitung parameter model yang paling sesuai dengan data yang tersedia.

3. Diagnostik

   Memastikan bahwa model yang dibangun sesuai dan memberikan hasil yang baik dengan melakukan analisis residual.

4. Peramalan

   Menggunakan model yang telah dibangun untuk membuat prediksi nilai masa depan berdasarkan data historis.

2.1.3 Transformer

Arsitektur *Transformer* didasarkan pada *self-attention mechanism* yang memungkinkan model untuk memperhitungkan seluruh konteks *sequen input* secara simultan tanpa bergantung pada urutan. Pendekatan ini berbeda dari RNN yang memproses data secara berurutan, sehingga memiliki keterbatasan dalam mengakses informasi yang jauh dalam sekuens data.

Dalam *Transformer*, komponen utama adalah lapisan *encoder* dan *decoder*, yang menggunakan mekanisme *multi-head attention* untuk menemukan hubungan antar-kata atau antar-token dalam suatu kalimat atau dokumen (Liu et al., 2023).

2.1.3.1 *Transformer* dalam Pemrosesan Bahasa Alami (NLP)

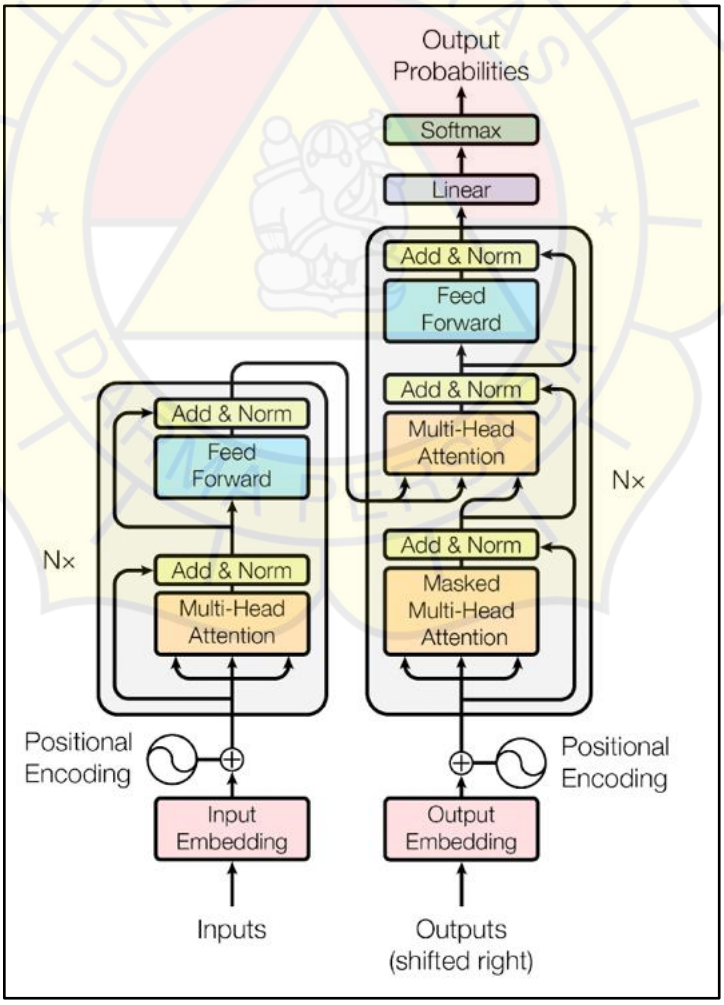
Arsitektur Transformer diperkenalkan oleh Vaswani et al. pada tahun 2017 melalui makalah berjudul "Attention Is All You Need". Sejak saat itu, Transformer telah menjadi landasan utama dalam pengembangan berbagai model modern di bidang Pemrosesan Bahasa Alami (Natural Language Processing/NLP), seperti dalam tugas penerjemahan bahasa, rangkuman otomatis (text summarization), penjawaban pertanyaan (question answering), hingga dialog interaktif.

Tidak seperti arsitektur sebelumnya seperti Recurrent Neural Network (RNN) atau Long Short-Term Memory (LSTM), yang memproses data secara berurutan dan cenderung kesulitan dalam menangani dependensi jangka panjang dalam teks, Transformer memungkinkan pemrosesan paralel seluruh urutan input. Hal ini dimungkinkan berkat mekanisme self-attention, yang memungkinkan model untuk memperhitungkan konteks global dari setiap token (kata atau sub-kata) dalam satu langkah komputasi. Dengan kata lain, Transformer dapat menentukan kata-kata mana yang saling berkaitan dalam satu kalimat atau paragraf, tanpa harus melalui proses iteratif.

Keunggulan lain dari Transformer adalah skalabilitasnya. Arsitektur ini mendukung pelatihan pada dataset besar dan arsitektur berskala besar, seperti GPT, BERT, T5, dan lain sebagainya, yang merupakan model-model besar berbasis

Transformer yang kini mendominasi berbagai benchmark NLP. Selain itu, struktur non-sekuensial dari Transformer juga membuatnya lebih efisien untuk dijalankan di perangkat keras modern seperti GPU dan TPU, karena dapat memanfaatkan komputasi paralel secara maksimal.

Dengan kemampuannya dalam menangani konteks panjang, efisiensi dalam pelatihan, serta kemampuannya untuk ditransfer ke berbagai tugas NLP, Transformer telah menjadi arsitektur utama dalam era modern NLP dan mendorong kemajuan pesat dalam teknologi bahasa mesin (Rothman, 2021).



Gambar 2.1 Arsitektur Transformer

Komponen Utama dalam Arsitektur *Transformer* pada gambar 2.1 adalah

1. *Self-attention Mechanism*

   Self-attention memungkinkan model untuk mempertimbangkan setiap kata dalam teks terhadap semua kata lainnya, memberikan bobot perhatian lebih besar pada kata-kata yang lebih relevan dalam konteks tertentu (Vaswani et al., 2017).

2. *Multi-head attention*

   Transformer menggunakan beberapa "kepala" perhatian untuk melihat berbagai aspek atau pola dalam data. *Multi-head attention* ini memungkinkan model menangkap hubungan antar-kata yang lebih kompleks.

3. *Positional Encoding*

   Karena *Transformer* tidak memiliki urutan pemrosesan bawaan, setiap posisi token dalam teks diberi nilai tambahan yang disebut positional encoding. Ini membantu model memahami urutan kata dalam kalimat.

4. *Layer Norm dan Feed Forward Network*

   Setiap lapisan *Transformer* memiliki lapisan normalisasi (layer norm) dan jaringan feed-forward untuk memperkuat representasi fitur.

Transformer terdiri dari dua bagian utama yaitu *encoder* untuk mengonversi *input* menjadi representasi vektor, dan *decoder* untuk mengonversi vektor menjadi *output*, seperti dalam tugas terjemahan bahasa.

2.1.3.2 Perkembangan *Transformer* untuk *Time series*

Walaupun *Transformer* awalnya dirancang untuk tugas NLP, arsitektur ini segera menunjukkan potensi dalam menangani data sekuensial lainnya, termasuk *Time series*. Data *Time series* memiliki tantangan berbeda dibandingkan teks, terutama dalam hal pola musiman, tren jangka panjang, dan korelasi antar kejadian yang terkadang jauh rentangnya (Rothman, 2021).

Dalam pemodelan *Time series*, *Transformer* menghadapi tantangan utama, yaitu :

a. Kompleksitas Komputasi

   Transformer memiliki kompleksitas komputasi *O(L^2)*, di mana *L* adalah panjang sekuens. Ini menyebabkan kebutuhan komputasi yang tinggi untuk data *Time series* yang panjang.

b. Redundansi Data

   Data *Time series* sering kali memiliki banyak data berulang atau kurang relevan, yang meningkatkan jumlah interaksi yang tidak diperlukan.

Untuk mengatasi tantangan ini, berbagai modifikasi *Transformer* mulai dikembangkan, termasuk model *Informer* yang dirancang khusus untuk efisiensi dalam pemodelan *Time series*.

2.1.3.3 *Informer*

Informer adalah salah satu model *deep learning* yang dikembangkan untuk menangani prediksi deret waktu (time-series *forecasting*) dengan panjang urutan data yang sangat besar dan kompleks. *Informer* merupakan pengembangan dari arsitektur *transformer*, sebuah model yang pertama kali diperkenalkan dalam dunia pemrosesan bahasa alami (*natural language processing*).

Penggunaan model *transformer* telah terbukti efektif dalam menangani tugas-tugas seperti penerjemahan bahasa, tetapi *Informer* dirancang khusus untuk memperluas manfaat arsitektur ini ke dalam domain deret waktu (Zhou et al., 2020).

Salah satu keunggulan utama dari *Informer* adalah kemampuannya dalam menangkap dependensi jangka panjang dalam data, yang sering kali sulit ditangani oleh metode statistik tradisional seperti ARIMA. Gempa bumi, yang terjadi di wilayah dengan pola tektonik yang kompleks seperti Selat Sunda, sering kali menunjukkan pola yang sulit dipahami oleh model linier sederhana. Oleh karena itu, kemampuan *Informer* untuk menangkap pola-pola non-linier dan korelasi jangka panjang memberikan keunggulan tersendiri dalam konteks prediksi gempa bumi.

Informer mengadopsi mekanisme *attention* yang canggih untuk memproses data. *Attention* memungkinkan model untuk memperhatikan bagian-bagian penting dari data secara selektif, sehingga tidak semua informasi dalam deret waktu diproses dengan cara yang sama (Vaswani et al., 2017). Hal ini membuat *Informer* lebih efisien dalam mengatasi masalah kompleksitas skala data, di mana panjang

urutan data yang sangat besar sering kali menjadi kendala dalam model *deep learning* konvensional.

Selain itu, dengan teknik optimasi yang disebut sebagai ProbSparse *self-attention*, *Informer* dapat mengurangi kompleksitas komputasi dari $O(L^2)$ menjadi O(LlogL), di mana L adalah panjang urutan data, membuat model ini sangat efisien untuk diterapkan pada data deret waktu skala besar seperti data gempa bumi.

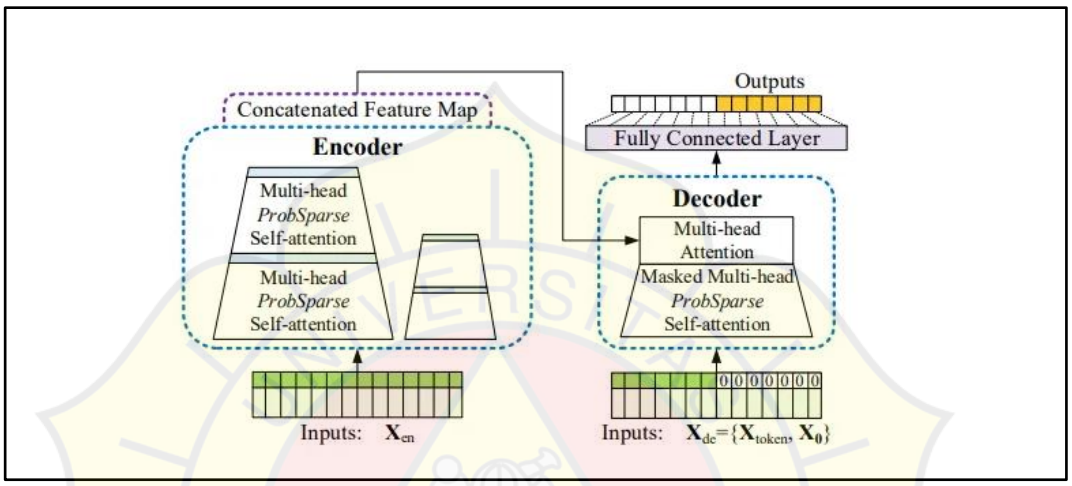
Dengan kemampuannya yang lebih unggul dalam menangkap pola non-linier dan anomali dalam data gempa, *Informer* diharapkan dapat memberikan prediksi yang lebih akurat dibandingkan metode berbasis statistik. Sering kali, pola gempa tidak hanya mengikuti satu tren atau musim, melainkan dipengaruhi oleh banyak faktor kompleks yang sulit diprediksi. Penggunaan *Informer* memungkinkan model untuk mengintegrasikan pola tersebut ke dalam perhitungan prediksi, sehingga menghasilkan prediksi yang lebih *robust* dan fleksibel, meskipun dihadapkan pada data yang sangat bervariasi dan tidak menentu.

Model ini juga dirancang untuk menangani ketergantungan jangka panjang dalam data. Ini sangat penting dalam konteks prediksi gempa, karena peristiwa gempa tidak selalu berurutan atau terjadi dalam pola yang teratur. Sebagai contoh, gempa besar mungkin dipicu oleh aktivitas seismik bertahun-tahun sebelumnya, dan hal ini menuntut adanya model yang dapat melihat ke belakang dalam deret waktu untuk memahami pola-pola historis yang relevan.

Selain itu, *Informer* juga mendukung multi-step *forecasting*, artinya model ini tidak hanya memprediksi satu titik waktu di masa depan, tetapi dapat memprediksi banyak langkah ke depan secara bersamaan. Ini membuatnya cocok

untuk digunakan dalam aplikasi peringatan dini, di mana prediksi dalam beberapa hari atau minggu ke depan sangat penting untuk memberikan waktu yang cukup bagi mitigasi bencana

Berikut merupakan arsitektur dari *Informer* :



Gambar 2.2 Arsitektur Informer

Gambar 2.2 menunjukan arsitektur *Informer* dengan penjelasan sebagai berikut:

1. *Encoder-Decoder* Architecture *Informer* menggunakan arsitektur berbasis *encoder-decoder*, serupa dengan arsitektur *transformer* standar. Namun, *Informer* dioptimalkan untuk prediksi deret waktu yang panjang (*long sequence time-series forecasting*) dengan beberapa inovasi untuk meningkatkan efisiensi komputasi dan kapasitas prediksi.

    a. *Encoder*

    Bagian *encoder* menerima *input* data sekuensial yang panjang, mengubahnya menjadi representasi fitur, dan mengelola hubungan jangka panjang antar elemen dalam sekuens tersebut. *Encoder* ini menggunakan mekanisme *ProbSparse self-attention* yang lebih efisien dalam memproses *input* berukuran besar.

b. *Decoder*

Decoder menggunakan arsitektur *generative style*, yang memungkinkan prediksi dilakukan dalam satu langkah maju (*forward step*). Model ini tidak melakukan prediksi langkah demi langkah, sehingga mengurangi akumulasi kesalahan yang terjadi dalam proses prediksi panjang.

2. *ProbSparse Self-attention* adalah inovasi utama *Informer* yang mengatasi masalah komputasi kuadratik pada *self-attention* standar. *Self-attention* standar memerlukan waktu komputasi $O(L^2)$ untuk memproses panjang urutan L, sedangkan *ProbSparse* mengurangi kompleksitas ini menjadi O(L log L). Ini dicapai dengan hanya fokus pada elemen-elemen penting dalam urutan berdasarkan pengukuran keanekaragaman (*diversity measurement*) yang menggunakan *query sparsity* untuk memilih elemen kunci yang relevan.
3. *Self-attention Distilling Informer* memperkenalkan mekanisme *self-attention distilling* yang menyederhanakan peta fitur (*feature map*) dari *layer-layer self-attention* dengan mempertahankan hanya perhatian yang dominan. Mekanisme ini memungkinkan *encoder* untuk mengelola *input* urutan panjang dengan lebih efisien, dan secara bertahap mengurangi dimensi waktu tanpa kehilangan informasi penting.
4. *Generative Style Decoder* merupakan bagian *decoder* pada *Informer* menggunakan pendekatan generatif, di mana model memprediksi semua langkah *output* sekaligus, bukan secara bertahap. *Decoder* diberi token awal

(*start token*) yang memungkinkan prediksi panjang dilakukan dengan lebih cepat dan menghindari akumulasi kesalahan yang sering terjadi pada prediksi berbasis autoregresif. Pendekatan ini meningkatkan kecepatan prediksi dan mengurangi kompleksitas waktu komputasi.

2.1.4 Perhitungan *Error*

Dalam proses prediksi deret waktu, salah satu komponen penting untuk mengevaluasi kualitas model prediksi adalah dengan mengukur kesalahan *(error)* antara hasil prediksi dan nilai sebenarnya. Beberapa metrik evaluasi umum digunakan dalam penelitian prediksi deret waktu, termasuk dalam konteks gempa bumi. Berikut adalah beberapa metrik kesalahan yang gunakan dalam mengukur kinerja model prediksi seperti ARIMA dan *Informer*

2.1.4.*1 Mean Absolute Percentage Error* (MAPE)

MAPE mengukur kesalahan prediksi dalam bentuk persentase, yang membuatnya lebih mudah dibandingkan antar *Dataset* yang memiliki skala atau unit yang berbeda. MAPE memberikan proporsi kesalahan prediksi terhadap nilai aktual, sehingga lebih mudah untuk menginterpretasi dalam konteks bisnis atau aplikasi lain (Chiulli, 2018).

Rumus perhitungannya dalam persen adalah

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_1}{y_i}\right| \times 100$$

Namun, MAPE memiliki kelemahan jika terdapat nilai aktual yang sangat kecil, karena perhitungan persentase akan memberikan kesalahan yang sangat besar meskipun perbedaannya kecil.

2.1.4.2 Varian *Mean Absolute Percentage Error* (MAPE)

1. MAPE Standar

MAPE mengukur rata-rata persentase kesalahan absolut antara nilai prediksi dan aktual (dalam persen):

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_1}{y_i}\right| \times 100$$

Mean Absolute Percentage Error (MAPE) adalah salah satu metrik evaluasi yang sering digunakan untuk mengukur akurasi prediksi dalam bentuk persentase relatif terhadap nilai aktual. MAPE dihitung dengan mengambil rata-rata dari persentase kesalahan absolut antara nilai prediksi dan nilai actual (Chiulli, 2018).

Kelebihan MAPE terletak pada interpretasinya yang mudah dipahami karena disajikan dalam bentuk persen, sehingga pembaca dapat langsung menilai tingkat kesalahan model.

Meski demikian, MAPE memiliki keterbatasan, terutama ketika nilai aktual sama dengan nol, yang dapat menghasilkan nilai tidak terdefinisi (NaN). Selain itu, MAPE juga cenderung memberi bobot lebih besar pada kesalahan dari nilai aktual yang sangat kecil

2. *Symmetric* MAPE (sMAPE)

sMAPE menyesuaikan denominator dengan rata-rata antara nilai aktual dan prediksi, sehingga mengurangi masalah pembagian nol:

$$sMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_1|}{(|y_i|+|\hat{y}_1|)/2} \times 100$$

Untuk mengatasi keterbatasan MAPE standar, dikembangkan *Symmetric* MAPE atau sMAPE. Metode ini menyesuaikan perhitungan dengan menggunakan rata-rata antara nilai aktual dan prediksi sebagai basis pembagiannya (Ballina et al., 2024).

Keunggulan sMAPE adalah kemampuannya menghindari nilai NaN ketika nilai aktual sama dengan nol serta memberikan perhitungan kesalahan yang lebih simetris antara prediksi yang terlalu tinggi dan terlalu rendah.

Namun, sMAPE tetap sensitif terhadap kesalahan pada nilai yang sangat kecil, dan interpretasi persentasenya sedikit berbeda dibanding MAPE standar karena dasar perhitungannya bukan hanya nilai aktual

3. Modified / Adjusted MAPE (MAPE Modifikasi)

Varian lain yang dikenal adalah *Modified* atau *Adjusted* MAPE. Dalam metode ini, ditambahkan nilai epsilon kecil pada denominator untuk mencegah pembagian dengan nol (Armstrong, 2001).

$$MAPEadj = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_1|}{|y_i|+\epsilon} \times 100$$

Pendekatan ini memungkinkan MAPE tetap dapat digunakan pada data yang memiliki nilai nol, seperti data penjualan harian atau sensor yang kadang menghasilkan nol. Pemilihan nilai epsilon harus dilakukan secara hati-hati agar tidak memengaruhi hasil metrik secara signifikan

4. *Weighted* MAPE (wMAPE)

Weighted MAPE atau wMAPE adalah varian yang memberikan bobot proporsional terhadap nilai aktual sehingga kesalahan relatif terhadap nilai yang lebih besar menjadi lebih diperhatikan (Gilliland et al., 2016).

$$wMAPE = \frac{\Sigma|y_i - \hat{y}_1|}{\Sigma|y_i|} \times 100$$

Kelebihan wMAPE adalah mampu menampilkan kesalahan yang lebih representatif pada dataset dengan variasi nilai besar dan kecil, karena kesalahan pada nilai besar tetap dihitung secara signifikan. Kelemahannya, error kecil pada nilai kecil cenderung "tersembunyi" karena bobotnya lebih rendah

Memahami varian-varian MAPE ini penting untuk memilih metrik yang tepat sesuai karakteristik data dan task penelitian. sMAPE dan Modified MAPE cocok untuk mengatasi nilai nol, sedangkan wMAPE lebih cocok untuk dataset dengan variasi nilai besar-kecil. Pemilihan metrik yang tepat membantu menghindari masalah NaN dan memberikan evaluasi performa model yang lebih akurat dan representatif.

2.1.4.3 *Mean Squared Error* (MSE)

MSE menghitung rata-rata dari kuadrat kesalahan prediksi. Karena MSE mengkuadratkan perbedaan antara prediksi dan nilai aktual, metrik ini lebih sensitif terhadap kesalahan besar, seperti halnya RMSE. MSE sering kali digunakan sebagai fungsi kerugian dalam proses pelatihan model, termasuk dalam model *deep learning* seperti *Informer*.

Rumus perhitungannya adalah:

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

MSE dihitung dalam unit kuadrat dari data asli, sehingga sering diikuti dengan RMSE untuk interpretasi dalam unit yang sama dengan data (Novaliendry, 2024; Christy N., 2019).

2.1.4.4 Jenis Jenis Metriks *Error* untuk *Taks* yang Berbeda

1. *Mean Absolute Error* (MAE)

Mean Absolute Error (MAE) adalah salah satu metrik evaluasi yang digunakan untuk mengukur rata-rata kesalahan absolut antara nilai dan nilai aktual. MAE dihitung dengan rumus:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_1|$$

di mana n adalah jumlah data pengamatan. MAE memberikan ukuran kesalahan dalam satuan yang sama dengan data asli, sehingga mudah diinterpretasikan (Gilliland et al., 2016).

Kelebihan MAE adalah kesederhanaannya dan kemampuannya memberikan gambaran error secara langsung tanpa memperbesar efek outlier. Setiap kesalahan diperlakukan sama, sehingga hasil rata-rata menunjukkan error tipikal model secara proporsional. Oleh karena itu, MAE sangat cocok digunakan ketika tujuan evaluasi adalah untuk memahami kesalahan rata-rata model secara sederhana dan tidak ingin penalti berlebihan terhadap nilai ekstrem.

Namun, MAE juga memiliki keterbatasan. Karena tidak memperbesar pengaruh error yang besar, MAE kurang sensitif terhadap outlier atau kesalahan ekstrem. Jika dalam dataset terdapat beberapa prediksi yang sangat meleset, MAE mungkin tidak mencerminkan besarnya kesalahan tersebut secara signifikan. Oleh karena itu, MAE lebih cocok untuk situasi di mana distribusi kesalahan relatif merata dan tidak terdapat nilai ekstrem yang kritis.

2. *Root Mean Squared Error* (RMSE)

Root Mean Squared Error (RMSE) adalah metrik yang mengukur kesalahan prediksi dengan menghitung akar dari rata-rata kuadrat selisih antara nilai prediksi dan nilai aktual:

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

Karena selisih dikuadratkan sebelum dirata-ratakan, RMSE memberikan penalti lebih besar pada kesalahan yang ekstrem dibandingkan MAE. Hal ini membuat RMSE menjadi metrik yang lebih sensitif terhadap *outlier*, sehingga *error* besar

akan lebih terlihat dalam hasil evaluasi. RMSE tetap menggunakan satuan yang sama dengan data asli, sehingga interpretasinya lebih intuitif dibanding metrik persentase, tetapi nilainya dapat didominasi oleh beberapa *error* besar(Chiulli, 2018).

Kelebihan RMSE terletak pada kemampuannya untuk menekankan prediksi yang meleset secara signifikan. Dalam banyak penelitian, terutama *forecasting* atau regresi, RMSE digunakan untuk memastikan bahwa model tidak hanya memiliki rata-rata *error* yang rendah, tetapi juga mampu meminimalkan kesalahan besar yang dapat berdampak kritis.

Namun, kelemahan RMSE adalah sensitivitasnya terhadap *outlier* yang berlebihan, yang bisa membuat evaluasi model terkesan lebih buruk daripada kondisi rata-rata sebenarnya. Selain itu, karena satuannya sama dengan data asli, interpretasi secara relatif atau dalam persentase tidak langsung, sehingga sering dikombinasikan dengan metrik lain seperti MAPE atau sMAPE untuk memberikan perspektif yang lebih lengkap.

3. Alasan MAE dan RMSE tidak digunakan pada prediksi gempa

Dalam konteks prediksi gempa bumi, baik MAE maupun RMSE memiliki keterbatasan yang membuatnya kurang tepat. Gempa bumi merupakan fenomena yang sangat jarang namun ekstrem, sehingga data memiliki distribusi yang tidak merata dan sering mengandung nilai nol atau hampir nol pada periode tanpa gempa. MAE cenderung mengabaikan efek dari kejadian ekstrem ini karena memberikan bobot yang sama pada semua kesalahan, sedangkan RMSE bisa terlalu dipengaruhi

oleh beberapa kejadian besar sehingga hasil evaluasi menjadi tidak stabil atau menyesatkan.

Sebaliknya, metrik seperti MAPE dan variannya (sMAPE, Modified MAPE, wMAPE) lebih sesuai karena mampu memberikan penilaian kesalahan dalam bentuk relatif atau persentase, mengatasi masalah nilai nol, dan memberikan interpretasi yang lebih mudah dimengerti dalam konteks prediksi fenomena yang jarang terjadi tetapi berdampak besar.

Oleh karena itu, meskipun MAE dan RMSE tetap penting untuk banyak task, dalam penelitian ini fokus evaluasi menggunakan MAPE dan variannya untuk memberikan hasil yang lebih representatif dan realistis terhadap karakteristik data gempa.

2.1.5 Penggunaan *Python* dalam Pengembangan Aplikasi Prediksi Kejadian Gempa

2.1.5.1 Pengertian *Python*

Python adalah bahasa pemrograman tingkat tinggi yang terkenal karena sintaksisnya yang sederhana dan mudah dibaca. Dikenal sebagai bahasa yang sangat serbaguna, *Python* digunakan dalam berbagai bidang, termasuk pengembangan web, analisis data, pembelajaran mesin, otomatisasi, dan lainnya. Dikenalkan oleh Guido van Rossum pada tahun 1991, *Python* kini menjadi salah satu bahasa pemrograman yang paling populer di dunia(Sarno et al., 2023). Beberapa pustaka *Python* yang relevan meliputi:

a. *NumPy*

Pustaka untuk komputasi numerik yang menyediakan dukungan untuk array dan matriks, serta berbagai fungsi matematika.

b. *Pandas*

Pustaka untuk manipulasi dan analisis data yang memudahkan pengelolaan data dalam bentuk tabel.

c. *Matplotlib* dan *Seaborn*

Pustaka visualisasi yang memungkinkan pembuatan grafik dan *plot* untuk menganalisis dan menampilkan data secara visual.

2.1.5.2 Karakteristik Utama *Python*

Phyton memiliki karakteristik sebagai berikut :

a. Mudah Dipelajari

Sintaksis yang jelas dan ringkas membuat *Python* mudah dipelajari oleh pemula.

b. Dukungan Pustaka yang Luas

Python memiliki ekosistem pustaka yang kaya, seperti NumPy, Pandas, Mat*plot*lib untuk analisis data, dan TensorFlow serta PyTorch untuk pembelajaran mesin (Wardana, 2024).

c. Interoperabilitas

Python dapat diintegrasikan dengan bahasa lain, seperti C/C++ dan Java, memungkinkan penggunaan modul yang ditulis dalam bahasa lain.

d. Komunitas Besar

Dengan komunitas pengembang yang besar, *Python* menawarkan banyak sumber daya, tutorial, dan dukungan untuk belajar dan pengembangan.

2.1.5.3 Editor Jupyter

Jupyter Notebook adalah aplikasi web open-source yang memungkinkan pengembang dan peneliti untuk membuat dan berbagi dokumen yang berisi kode, visualisasi, dan narasi teks(Barry, 2016; Wintjen & Vlahutin, 2020). Jupyter mendukung berbagai bahasa pemrograman, tetapi paling sering digunakan dengan *Python*. Fitur utamanya meliputi:

a. Interaktivitas

   Pengguna dapat menjalankan kode secara interaktif dalam sel, memudahkan eksperimen dan eksplorasi data.

b. Visualisasi

   Mendukung integrasi dengan pustaka visualisasi seperti Mat*plot*lib, Seaborn, dan *Plot*ly, memungkinkan pengguna untuk menghasilkan grafik dan *plot* yang informatif.

c. Dokumentasi

   Memungkinkan pengguna untuk menambahkan penjelasan dan komentar di antara sel kode, menjadikan dokumentasi lebih mudah dan terstruktur.

d. Penggunaan di Data Science

   Jupyter sangat populer di kalangan ilmuwan data untuk analisis data, pemodelan, dan pembelajaran mesin, serta dapat digunakan untuk menguji dan mengembangkan model prediksi.

2.1.5.4 *Library* scikit-learn

Scikit-learn adalah pustaka *Python* yang sangat populer untuk pembelajaran mesin (Siahaan & Sianipar, 2021). *Library* ini menyediakan berbagai algoritma dan alat untuk analisis data dan pemodelan.

Fitur utama dari *Scikit-learn* meliputi:

a. Algoritma Pembelajaran Mesin

   Algoritma yang mencakup berbagai algoritma untuk klasifikasi, regresi, dan pengelompokan, seperti Random Forest, SVM, K-Means, dan lain-lain.

b. Praproses Data

   Scikit-learn menyediakan fungsi untuk praproses data, termasuk normalisasi, pengkodean kategori, dan pengisian nilai yang hilang.

c. Evaluasi Model

   Scikit-learn menawarkan metrik evaluasi yang berbeda, seperti akurasi, *precision, recall, dan cross-validation,* untuk membantu peneliti menilai performa model.

d. Integrasi Mudah

   Scikit-learn dapat dengan mudah diintegrasikan dengan pustaka lain seperti NumPy dan Pandas, memudahkan alur kerja analisis data.

Library lainnya yang juga digunakan

Selain pustaka yang disebutkan di atas, terdapat beberapa pustaka lain yang juga berperan penting dalam pengembangan aplikasi prediksi:

a. *TensorFlow*

Framework open-source untuk pembelajaran mesin yang menyediakan berbagai alat untuk membangun dan melatih model *deep learning*, termasuk arsitektur *Transformer* seperti *Informer* (Siahaan & Sianipar, 2021).

b. *Keras*

Pustaka tinggi untuk TensorFlow yang memungkinkan pengembang untuk membangun dan melatih model *Neural Network* dengan lebih mudah (Siahaan & Sianipar, 2021).

c. *Flask*

Sebagai micro-framework untuk *Python*, *Flask* digunakan untuk mengembangkan aplikasi web yang mengintegrasikan model prediksi dan memungkinkan interaksi pengguna (Thanh, 2019).

2.1.5.5 *Flask*: Micro-Framework untuk Pengembangan Web

Flask adalah *micro-framework* untuk *Python* yang dirancang untuk memudahkan pengembangan aplikasi web dengan fitur minimal tetapi fleksibel. *Flask* didasarkan pada Werkzeug (sebagai server dan alat pengembangan) dan Jinja2 (sebagai template engine), menjadikannya ringan dan modular (Adedeji, 2023).

Karakteristik Utama *Flask* adalah sebagai berikut :

a. Minimalis

Flask memberikan struktur dasar untuk aplikasi web tanpa mengikat pengembang pada konvensi tertentu, memberikan kebebasan untuk memilih komponen tambahan.

b. Ekstensibilitas

Dengan banyak ekstensi yang tersedia, seperti *Flask*-SQLAlchemy untuk interaksi basis data dan *Flask*-Login untuk manajemen sesi pengguna, pengembang dapat dengan mudah menambahkan fitur sesuai kebutuhan.

c. Template Engine

Menggunakan Jinja2, *Flask* memungkinkan pengembang untuk memisahkan logika aplikasi dari presentasi, menjadikan pengembangan lebih terorganisir.

d. *Routing* yang Sederhana

Flask menyediakan sistem routing yang intuitif, memungkinkan pengembang untuk menentukan URL dan menghubungkannya dengan fungsi yang sesuai.

Struktur *Flask* merupakan gambaran umum tentang bagaimana aplikasi *Flask* biasanya disusun. Dan *flask* disusun menggunakan app.py sebagai dasar aplikasi dalam bentuk python, templates sebagai user *interface* aplikasi *flask* dan juga static untuk menjalankan CSS, *JavaScript* dan juga *Image.*

File Strukturnya adalah sebagai berikut :

```
/my_Flask_app
    ├── app.py              # File utama aplikasi
    ├── templates/          # Folder untuk file HTML
    └── static/             # Folder untuk file CSS, JavaScript, dan
gambar
```

Berikut merupakan contoh kode aplikasi *Flask* :

```
from Flask import Flask, render_template
app = Flask(__name__)
@app.route('/')
def home():
    return render_template('index.html')
if __name__ == '__main__':
    app.run(debug=True)
```

Langkah-langkah Pengembangan Aplikasi Web dengan *Flask* yang lakukan adalah sebagai berikut ;

1. Instalasi *Flask*: menginstal *Flask* menggunakan pip

```
pip install Flask
```

2. Membuat Aplikasi Dasar membuat file app.py dan tulis kode seperti di atas.
3. Menambahkan Routing menentukan berbagai URL untuk menghubungkan ke fungsi yang berbeda, seperti halaman utama, halaman prediksi, dan lain-lain.
4. Menggunakan Template HTML membuat file HTML di dalam folder templates/ dan gunakan fungsi render_template() untuk menampilkannya.
5. Mengelola Data mengintegrasikan dengan basis data menggunakan *Flask*-SQLAlchemy untuk menyimpan dan mengambil data terkait gempa.
6. Mengimplementasikan Model Prediksi menghubungkan logika model prediksi (yaitu ARIMA dan *Informer*) dengan aplikasi web untuk menghasilkan prediksi berdasarkan *input* pengguna.

Keuntungan Menggunakan *Flask* untuk Aplikasi Web yaitu:

a. Kecepatan Pengembangan

Penggunaa *Flask* memungkinkan pengembang untuk membangun aplikasi dengan cepat, meminimalkan overhead.

b. Fleksibilitas

selaku pengembang memiliki kontrol penuh atas struktur dan komponen aplikasi, memungkinkan penyesuaian sesuai kebutuhan proyek.

c. Dukungan untuk RESTful APIs

Penggunaan *Flask* sangat cocok untuk membangun API, memudahkan integrasi dengan *frontend* dan aplikasi lain.

Python, sebagai bahasa pemrograman yang kuat dan fleksibel, sangat mendukung pengembangan aplikasi web melalui framework seperti *Flask*. Dengan pendekatan minimalis dan ekstensibilitas, *Flask* memungkinkan pengembang untuk membangun aplikasi prediksi kejadian gempa dengan efisien, memberikan hasil yang dapat diakses oleh pengguna atau instansi terkait. Keterampilan dalam menggunakan *Python* dan *Flask* tidak hanya meningkatkan kualitas aplikasi, tetapi juga memungkinkan penelitian Anda untuk disajikan dengan cara yang lebih interaktif dan informatif.

2.1.6 *Data Processing*

Data processing merupakan tahap penting dalam penelitian ini karena kualitas data sangat menentukan keakuratan dan reliabilitas model prediksi. Tahap ini meliputi serangkaian langkah yang bertujuan untuk membersihkan, menyiapkan, dan menyesuaikan dataset agar siap digunakan dalam pemodelan. Proses ini mencakup pembersihan data, imputasi, interpolasi, normalisasi, transformasi, deteksi dan penanganan *outlier,* serta *feature engineering.*

Data gempa bumi yang digunakan dalam penelitian ini biasanya berasal dari sensor seismik dan catatan historis, yang memiliki beberapa karakteristik khusus, seperti ketidaklengkapan data, interval waktu yang tidak merata, fluktuasi nilai yang ekstrem, dan noise akibat gangguan sensor (Wilson, 2021).

Tanpa tahap data processing yang tepat, model prediksi bisa menghasilkan hasil yang bias, tidak stabil, atau gagal menangkap pola temporal yang sebenarnya. Dengan melakukan *data processing* secara menyeluruh, dataset menjadi lebih konsisten, kontinu, dan representatif terhadap fenomena gempa, sehingga model dapat mempelajari tren dan pola dengan lebih baik.

2.1.6.1 Imputasi

Imputasi adalah proses untuk mengisi nilai-nilai yang hilang atau tidak tercatat dalam dataset. *Missing values* dapat muncul karena gangguan sensor, kesalahan pencatatan, atau periode tanpa kejadian (Sukasih & Scott, 2023). Tanpa imputasi, analisis data dan pemodelan dapat menghasilkan bias atau kesalahan tinggi.

Metode imputasi dibagi menjadi beberapa kategori.

1. Imputasi sederhana: menggunakan nilai *mean, median,* atau *mode* dari data yang tersedia. Metode ini mudah diterapkan dan cocok untuk dataset dengan distribusi stabil dan tanpa tren musiman yang kompleks.
2. Imputasi berbasis model: menggunakan metode regresi*, k-nearest neighbors* (KNN), atau *machine learning* untuk memperkirakan nilai yang hilang berdasarkan pola yang ditemukan dalam data lain. Metode ini lebih

akurat, terutama untuk dataset yang memiliki hubungan *non-linear* antar fitur.

3. Imputasi berbasis waktu: untuk data deret waktu, nilai hilang dapat diestimasi menggunakan metode *forward fill, backward fill,* atau *moving average.*

Pemilihan metode imputasi harus disesuaikan dengan karakteristik data dan tujuan penelitian. Dalam penelitian ini, imputasi dilakukan untuk memastikan dataset gempa bumi tetap lengkap dan konsisten, sehingga prediksi model tidak terganggu oleh missing values.

2.1.6.2 Interpolasi

Interpolasi merupakan teknik khusus dalam imputasi yang digunakan untuk mengestimasi nilai yang hilang dengan memperhitungkan data sekitarnya. Interpolasi sangat berguna ketika data memiliki interval waktu yang tidak merata atau *missing values* yang acak, karena teknik ini dapat menghasilkan data yang kontinu dan lebih rapi (Luo, 2024).

Terdapat beberapa jenis interpolasi yang umum digunakan:

1. *Linear interpolation*: mengasumsikan perubahan antar titik data terjadi secara linear, mudah diterapkan, dan memberikan estimasi sederhana. Nilai yang hilang pada titik x antara (x_1,y_1) dan (x_2,y_2) dapat dihitung dengan:

$$\hat{y}(x) = y_1 + (x - x_1)\frac{y_2 - y_1}{x_2 - x_1}$$

2. *Polynomial interpolation*: menggunakan kurva polinomial untuk memperkirakan nilai, cocok untuk data yang memiliki pola non-*linear*, namun lebih sensitif terhadap fluktuasi ekstrem.
3. *Spline interpolation*: menggunakan fungsi *spline* untuk menciptakan kurva halus yang melewati semua titik data, efektif untuk menangkap pola yang kompleks dan perubahan gradual.

Beberapa penelitian dan jurnal menyatakan bahwa interpolasi diperbolehkan untuk data yang tidak rapi atau tidak lengkap, selama metodologi dijelaskan dengan jelas (Kazijevs & Samad, 2023; Luo, 2024). Alasan utamanya:

1. Menjaga struktur temporal: model deret waktu umumnya memerlukan data dengan interval yang seragam. Interpolasi memastikan tidak ada celah besar yang dapat mengganggu pembelajaran pola.
2. Estimasi berbasis informasi yang ada: interpolasi bukan menciptakan data baru secara acak, melainkan melakukan perkiraan yang konsisten dengan tren lokal dari data aktual.
3. Mencegah bias akibat data hilang: *missing values* yang dibiarkan kosong dapat menyebabkan model salah memahami tren atau gagal melakukan training.
4. Mendukung akurasi prediksi: dengan data yang rapi dan kontinu, model dapat lebih mudah menangkap tren jangka panjang maupun variasi musiman (Lepot et al., 2017).

Dengan demikian, interpolasi dipilih dalam penelitian ini karena *dataset* gempa bumi memiliki nilai hilang serta *interval* pengukuran yang tidak seragam. Penerapan interpolasi memastikan dataset tetap terstruktur dengan baik dan siap digunakan dalam tahap pemodelan peramalan.

2.1.6.3 Normalisasi dan Standarisasi

Normalisasi dan standarisasi adalah proses penyamaan skala data agar fitur dengan rentang nilai berbeda dapat diproses secara setara oleh model (Morimoto & Nakahara, 2024). Normalisasi mengubah skala data ke rentang tertentu, misalnya 0 hingga 1, Standarisasi mengubah data sehingga memiliki rata-rata 0 dan standar deviasi 1.

Kedua teknik ini penting untuk algoritma machine learning, terutama deep learning, agar konvergensi training lebih cepat dan hasil model lebih stabil.

2.1.6.4 Deteksi dan Penanganan Outlier

Outlier adalah nilai ekstrem yang tidak sesuai dengan pola umum data. Dalam data gempa bumi, outlier bisa muncul karena kesalahan sensor atau kejadian gempa yang sangat langka. Outlier dapat mengganggu performa model dan mempengaruhi metrik error (Ali et al., 2025).

Beberapa teknik untuk mendeteksi dan menangani outlier meliputi:

1. Metode statistik, misalnya z-score atau IQR (interquartile range).
2. Capping atau Winsorization, yaitu membatasi nilai ekstrem pada batas tertentu.

3. Penghapusan outlier, jika nilai tersebut jelas bukan bagian dari fenomena alami.

Pemilihan metode penanganan outlier harus disesuaikan dengan karakteristik data agar informasi penting tidak hilang.

2.1.6.5 Feature Engineering

Feature engineering adalah proses pembuatan atau transformasi variabel untuk meningkatkan kemampuan model dalam menangkap pola data (Duboue, 2020). Pada prediksi gempa bumi, langkah ini dapat meliputi:

1. Membuat fitur lag dari data deret waktu, untuk menangkap pengaruh kejadian sebelumnya.
2. Menghitung rolling mean atau rolling std untuk menangkap tren dan volatilitas temporal.
3. Transformasi logaritmik atau scaling untuk mengurangi skewness pada data magnitude gempa.

2.2 Kajian Penelitian Terdahulu

2.2.1 Paper 1

Paper pertama berjudul *"Prediksi Jumlah Gempa Tektonik di Wilayah Jawa Timur dengan Menggunakan Metode ARIMA Box Jenkins dan Kalman Filter"* yang ditulis oleh Zulfatul Aizzah, Putroue Keumala Intan, dan Wika Dianita Utami, dipublikasikan dalam Jurnal Riset Sains dan Teknologi (JRST) pada tahun 2021 dan terindeks di Sinta 3. Dalam penelitian ini, menggunakan dua metode analisis, yaitu

ARIMA *Box Jenkins* dan *Kalman Filter*, untuk memprediksi jumlah gempa tektonik yang terjadi di wilayah Jawa Timur. Metode ARIMA *Box Jenkins* digunakan untuk menganalisis data deret waktu yang berkaitan dengan pola gempa, sementara *Kalman Filter* digunakan untuk memperbaiki estimasi model dengan memanfaatkan pengukuran yang ada, sehingga diharapkan dapat memberikan prediksi yang lebih akurat terkait aktivitas gempa di wilayah tersebut

2.2.1.1 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk memprediksi jumlah gempa tektonik di wilayah Jawa Timur sebagai upaya mitigasi bencana, mengingat Indonesia merupakan negara yang terletak di wilayah cincin api pasifik yang aktif dan sering mengalami gempa bumi. Penelitian ini menggunakan metode ARIMA yang diperbaharui dengan estimasi *Kalman Filter* untuk meningkatkan akurasi prediksi (Aizzah et al., 2022).

2.2.1.2 Metodologi yang digunakan

Metodologi penelitian yang digunakan dalam jurnal ini melibatkan dua metode utama, yaitu ARIMA *Box Jenkins* dan *Kalman Filter*. Berikut adalah penjelasan rinci mengenai kedua metode tersebut:

1. Metode ARIMA *Box Jenkins*:
    a. Identifikasi Model: Proses dimulai dengan identifikasi korelasi antar deret waktu. Model ARIMA digunakan untuk memprediksi data

Time series dengan mengamati data saat ini yang tergantung pada beberapa pengamatan dari data tahun-tahun sebelumnya.

b. Pengecekan Kelayakan Data: Data yang digunakan diperiksa kestasionerannya, baik dalam rata-rata maupun dalam varians. Jika data belum stasioner, transformasi *Box-Cox* dilakukan untuk mencapai kestasioneran.

c. Uji Kestasioneran: Setelah transformasi, kestasioneran data terhadap mean diperiksa menggunakan uji *Augmented Dickey-Fuller (ADF).* Jika p-value > 0.05, data dianggap belum stasioner.

d. Penghitungan dan Uji Signifikansi Model: Setelah mendapatkan model yang sesuai, dilakukan penghitungan dan uji signifikansi untuk memastikan model yang dipilih adalah yang terbaik.

2. Metode *Kalman Filter*:

a. Estimasi dan Koreksi: *Kalman Filter* digunakan untuk memperbaiki hasil prediksi dari model ARIMA yang memiliki nilai *error* relatif tinggi. Proses ini melibatkan beberapa tahapan, yaitu identifikasi model sistem dan pengukuran, inisialisasi, prediksi, dan koreksi.

b. Algoritma Diskrit: Algoritma *Kalman Filter* bertipe diskrit, yang berarti ia bekerja dengan data yang diambil pada interval waktu tertentu. Proses ini bertujuan untuk memperbarui hasil prediksi dengan mengoreksi estimasi berdasarkan pengukuran yang baru.

Dengan menggabungkan kedua metode ini, penelitian bertujuan untuk meningkatkan akurasi prediksi jumlah gempa tektonik di wilayah Jawa Timur. Hasil

dari model ARIMA yang awalnya memiliki nilai *error* yang tinggi diperbaiki dengan estimasi dari *Kalman Filter*, sehingga diharapkan menghasilkan prediksi yang lebih akurat.

2.2.1.3 Temuan Utama

Temuan utama dari penelitian ini adalah bahwa penggunaan metode ARIMA *Box Jenkins* yang diperbaharui dengan estimasi *Kalman Filter* dapat meningkatkan akurasi prediksi jumlah gempa tektonik di wilayah Jawa Timur. Penelitian menunjukkan bahwa model ARIMA memiliki nilai *error* yang relatif tinggi, sehingga perlu diperbaiki dengan menggunakan *Kalman Filter* untuk mendapatkan hasil yang lebih akurat.

Dalam penelitian ini, setelah melakukan transformasi data untuk mencapai kestasioneran, model ARIMA yang dihasilkan diuji dan dibandingkan dengan estimasi dari *Kalman Filter*. Hasil analisis menunjukkan bahwa kombinasi kedua metode ini memberikan prediksi yang lebih baik dibandingkan dengan penggunaan model ARIMA saja.

Selain itu, penelitian ini juga menekankan pentingnya prediksi gempa sebagai upaya mitigasi bencana, mengingat Indonesia berada di wilayah cincin api pasifik yang rentan terhadap gempa bumi. Dengan demikian, temuan ini memberikan kontribusi signifikan terhadap pengembangan metode prediksi gempa yang lebih akurat dan efektif.

2.2.1.4 Kelemahan Penelitian

Kelemahan penelitian ini terletak pada penggunaan model ARIMA yang memiliki nilai *error* yang relatif tinggi. Hal ini menunjukkan bahwa model ARIMA saja tidak cukup akurat dalam memprediksi jumlah gempa tektonik, sehingga perlu diperbaiki dengan menggunakan metode lain seperti *Kalman Filter* untuk meningkatkan akurasi prediksi.

Untuk mengembangkan penelitian ini, beberapa langkah yang kemungkinan dapat diambil oleh antara lain:

1. Penggunaan Metode Alternatif

Mengintegrasikan metode lain selain ARIMA dan *Kalman Filter*, seperti metode *Machine learning* atau *deep learning*, yang mungkin dapat memberikan hasil prediksi yang lebih baik dan lebih akurat.

2. Peningkatan Kualitas Data

Mengumpulkan data yang lebih banyak dan lebih berkualitas, termasuk data historis gempa yang lebih lengkap, untuk meningkatkan model prediksi.

3. Analisis Variabel Lain

Mempertimbangkan variabel lain yang mungkin mempengaruhi kejadian gempa, seperti aktivitas seismik di daerah sekitar, untuk meningkatkan akurasi model.

4. Uji Coba Model yang Berbeda

Melakukan perbandingan dengan model-model lain, seperti *Informer*, untuk melihat mana yang memberikan hasil prediksi yang lebih baik dalam konteks kejadian gempa di Selat Sunda, seperti yang dilakukan dalam penelitian

2.2.2 Paper 2

Paper kedua berjudul *"Prediksi Gempa Bumi di Indonesia Menggunakan R-Shiny"* yang ditulis oleh Yova Rezky Amanda, Mumtazah Nurul ’Aini, dan Melliyuma Miyoze, dipublikasikan dalam Jurnal Sains dan Seni ITS pada tahun 2022 dan terindeks di Sinta 4. Dalam penelitian ini, menggunakan aplikasi berbasis R-Shiny untuk memprediksi gempa bumi di Indonesia. R-Shiny, sebagai platform untuk membuat aplikasi interaktif berbasis web, memungkinkan pengguna untuk melakukan analisis data gempa secara real-time dengan antarmuka yang mudah digunakan. Penelitian ini bertujuan untuk memberikan alat prediksi yang dapat membantu dalam mitigasi bencana dan memberikan informasi yang lebih cepat serta akurat mengenai potensi gempa bumi yang dapat terjadi di Indonesia.

2.2.2.1 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk merancang dan menjalankan sistem deteksi gempa bumi di Indonesia, serta membangun aplikasi web interaktif yang dapat memprediksi frekuensi gempa bumi berdasarkan data historis. Penelitian ini bertujuan agar praktisi dapat lebih memahami cara membangun sistem prediksi gempa menggunakan software, dan bagi pembaca, dapat mengetahui informasi terkini mengenai prediksi gempa yang akan terjadi di masa mendatang (Amanda et al., 2022).

2.2.2.2 Metodologi yang digunakan

A. Sumber Data

Penelitian ini menggunakan data sekunder yang diperoleh dari situs web repogempa.bmkg.go.id. Data tersebut diakses pada hari Minggu, 28 Maret 2021, di wilayah Jawa Timur.

B. Variabel Penelitian

Tabel 2.1 menunjukkan variabel yang digunakan dalam penelitian ini untuk memprediksi frekuensi kejadian gempa bumi di Indonesia.

Tabel 2.1 Variabel Penelitian

Variabel ke-	Keterangan	Skala
1	Waktu terjadinya gempa	Ratio
2	*Latitude*	Ratio
3	*Longitude*	Ratio
4	Depth	Ratio
5	*Magnitudo* (M)	Ratio
6	Region	Nominal

C. Tahapan Analisis

Penelitian ini dilakukan melalui tahapan analisis sebagai berikut:

1. Mengimpor Data

Data gempa bumi di Indonesia dari lima tahun terakhir, yaitu dari Februari 2016 hingga Februari 2021, dimasukkan dalam format Excel.

2. Melakukan Prediksi

Prediksi gempa dilakukan berdasarkan data yang telah diperoleh, meliputi:

a. Memetakan wilayah dalam data ke peta wilayah Indonesia.
b. Mengelompokkan data kejadian gempa.
c. Prediksi dibuat menggunakan model *Moving Average* (MA) orde 1, karena fungsi Autocorrelation (ACF) data menunjukkan nilai tidak signifikan setelah *lag* 1.
d. Menghitung tingkat kesalahan dari hasil prediksi.
e. Menyimpan hasil prediksi dalam format file RDS.

3. Visualisasi Data

Data gempa divisualisasikan menggunakan aplikasi Shiny, di mana prediksi frekuensi gempa ditampilkan pada peta Indonesia di dalam aplikasi tersebut.

2.2.2.3 Temuan Utama

Temuan utama dari penelitian ini adalah sebagai berikut:

1. Pentingnya Prediksi Gempa Bumi

Penelitian menekankan bahwa memprediksi gempa bumi, yang dapat disebabkan oleh aktivitas vulkanis atau tektonik, sangat penting untuk meningkatkan kewaspadaan masyarakat terhadap potensi bencana.

2. Pengembangan Aplikasi Shiny

Penelitian ini berhasil mengembangkan aplikasi web interaktif menggunakan R-Shiny yang memungkinkan pengguna untuk memprediksi dan memvisualisasikan frekuensi gempa bumi di Indonesia berdasarkan data historis dari Februari 2016

hingga Februari 2021. Aplikasi ini memiliki antarmuka pengguna (UI) yang memungkinkan interaksi dengan data dan model.

3. Komponen Aplikasi

Aplikasi ini terdiri dari beberapa komponen kunci, termasuk:

a. *User* Interface (UI)

Menyediakan panel kontrol untuk *input* dan tampilan *output* seperti grafik dan tabel, yang dirancang untuk memudahkan interaksi pengguna.

b. Server Logic

Memproses *input* pengguna, melakukan analisis data, dan menghasilkan prediksi menggunakan metode statistik seperti Moving Average.

c. Visualisasi Data

Menampilkan frekuensi gempa historis dan yang diprediksi pada peta choropleth Indonesia, memungkinkan pengguna untuk melihat variasi frekuensi gempa di berbagai wilayah.

4. Metrik Kesalahan

Aplikasi ini juga menyediakan metrik kesalahan (Mean *Error*, RMSE, MAE) untuk prediksi, yang membantu pengguna menilai akurasi dari ramalan yang dihasilkan.

5. Evaluasi Teknik Peramalan

Penelitian ini menggunakan *Mean Squared Error* (MSE) sebagai metode untuk mengevaluasi teknik peramalan, yang menunjukkan bahwa MSE cenderung memberikan penalti lebih berat pada kesalahan yang lebih besar.

6. Penggunaan R dan R-Studio

Penelitian ini memanfaatkan R dan R-Studio sebagai lingkungan pengembangan terintegrasi untuk analisis statistik dan visualisasi data, serta untuk membangun aplikasi web interaktif.

Secara keseluruhan, penelitian ini memberikan kontribusi signifikan dalam memahami dan memprediksi kejadian gempa bumi di Indonesia, serta menyediakan alat yang berguna bagi masyarakat dan praktisi untuk meningkatkan kesiapsiagaan terhadap bencana.

2.2.2.4 Kelemahan Penelitian

Kelemahan dari penelitian ini adalah :

1. Model Prediksi yang Sederhana

Penelitian ini menggunakan metode Single *Moving Average* (SMA) untuk memprediksi frekuensi gempa bumi. SMA sering kali tidak cukup sensitif terhadap pola perubahan data yang kompleks, sehingga hasil prediksi mungkin kurang akurat jika dibandingkan dengan model yang lebih canggih seperti ARIMA atau model berbasis *Machine learning*.

2. Keterbatasan Data Historis

Penelitian ini hanya menggunakan data gempa dari lima tahun (2016-2021). Rentang data yang terbatas bisa memengaruhi akurasi prediksi karena model tidak mendapatkan cukup variasi kejadian untuk menangkap pola jangka panjang yang mungkin muncul di data yang lebih luas.

3. Visualisasi yang Terbatas pada Wilayah Indonesia

Penelitian ini hanya memetakan frekuensi gempa di Indonesia, tanpa memperhitungkan koneksi data gempa dari wilayah tetangga. Wilayah seismik yang lebih luas dapat memperkaya analisis pola gempa yang berpotensi terjadi.

Untuk mengembangkan penelitian ini, beberapa langkah yang kemungkinan dapat diambil oleh antara lain:

1. Menggunakan Model Prediksi yang Lebih Kompleks

   Mengganti atau mengkombinasikan SMA dengan model yang lebih canggih seperti ARIMA atau *Informer* bisa meningkatkan akurasi dan sensitivitas prediksi, terutama dalam menangkap pola musiman dan tren jangka panjang.

2. Memperluas Rentang Data

   Menggunakan data historis yang lebih panjang, atau bahkan data dari sumber-sumber tambahan, akan memberikan model lebih banyak informasi untuk menghasilkan prediksi yang lebih stabil dan dapat diandalkan.

3. Implementasi dengan Teknologi Baru

   Penggunaan model dengan web berbasis framework yang lebih modern, seperti *Flask* atau Django, serta integrasi dengan API geospasial yang lebih kaya fitur, bisa meningkatkan kapabilitas dan fleksibilitas aplikasi ini.

4. Mengintegrasikan Data dari Wilayah Seismik Tetangga

   Untuk memperkaya analisis prediksi, data gempa dari negara-negara tetangga yang juga rentan terhadap aktivitas seismik dapat diintegrasikan agar pola pergerakan yang lebih besar bisa dianalisis.

Pengembangan ini bisa meningkatkan efektivitas aplikasi dalam memberikan informasi yang lebih akurat dan prediktif kepada pengguna, sehingga bisa memberikan manfaat yang lebih besar dalam mitigasi risiko gempa bumi.

2.2.3 Paper 3

Paper ketiga berjudul *"Uji Performa Prediksi Gempa Bumi di Jawa Timur dengan Artificial Neural Network"* yang ditulis oleh Muhammad Aji Permana dan M. Faisal, dipublikasikan dalam EULER: Jurnal Ilmiah Matematika, Sains dan Teknologi pada tahun 2023 dan terindeks di Sinta 4. Penelitian ini menguji performa model prediksi gempa bumi di wilayah Jawa Timur menggunakan metode Artificial *Neural Network* (ANN). Dengan memanfaatkan kemampuan ANN dalam mengenali pola-pola kompleks pada data gempa, penelitian ini bertujuan untuk mengembangkan model yang lebih akurat dalam memprediksi kejadian gempa bumi di wilayah tersebut. Hasil uji performa menunjukkan bahwa ANN dapat menjadi salah satu alternatif yang efektif dalam meningkatkan akurasi prediksi gempa, yang pada akhirnya dapat membantu dalam perencanaan mitigasi bencana yang lebih baik di Jawa Timur.

2.2.3.1 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk memperoleh arsitektur jaringan terbaik yang diterapkan pada data frekuensi kejadian gempa bumi per bulan di Provinsi Jawa Timur. Penelitian ini bertujuan untuk meningkatkan kesiapsiagaan

dalam upaya mengurangi risiko gempa bumi dengan melakukan prediksi mengenai frekuensi kejadian gempa (Permana & Faisal, 2023).

2.2.3.2 Metodologi yang digunakan

Metodologi penelitian yang digunakan dalam paper ini melibatkan beberapa langkah sebagai berikut:

1. Pengumpulan Data

Penelitian ini menggunakan data katalog parameter kejadian gempa bumi yang bersumber dari BMKG (Badan Meteorologi, Klimatologi, dan Geofisika) Stasiun Geofisika Nganjuk. Data yang dicatat adalah kejadian gempa bumi di wilayah Jawa Timur dan sekitarnya dengan periode data antara tahun 2016 - 2021, yang mencakup 4029 kejadian gempa bumi. Data tersebut kemudian dikelompokkan menjadi frekuensi kejadian gempa bumi per bulan, menghasilkan 72 *Dataset* selama 6 tahun.

2. Pembagian Data

Dataset yang telah dikumpulkan dibagi menjadi dua bagian: 70% digunakan sebagai data latih dan 30% sebagai data uji. Pembagian ini bertujuan untuk melatih model dan kemudian menguji kinerjanya.

3. Pemilihan Arsitektur Jaringan

Penelitian ini melakukan percobaan dengan beberapa variabel arsitektur *Neural Network*. Kombinasi jumlah node *input*, jumlah neuron hidden, dan jumlah *output* diuji untuk menemukan arsitektur terbaik dengan nilai *error* yang lebih kecil.

Salah satu arsitektur yang diuji adalah 3-5-1, yang terdiri dari 3 *input* dan 5 neuron hidden.

4. Pelatihan Model

Proses pelatihan dilakukan menggunakan software R dengan package neuralnet. Fungsi "neuralnet" digunakan untuk menyesuaikan bobot dan bias dalam jaringan neural berdasarkan data yang diberikan. Proses pelatihan dilakukan dengan beberapa iterasi untuk mencapai model yang optimal.

5. Prediksi dan Evaluasi

Setelah model dilatih, model yang sudah dilatih digunakan untuk melakukan prediksi pada data pengujian atau data baru. Kinerja model dievaluasi dengan membandingkan hasil prediksi dengan nilai yang sebenarnya untuk menilai akurasi dan efektivitas model yang dibangun.

Metodologi ini bertujuan untuk mendapatkan arsitektur jaringan terbaik yang dapat digunakan untuk memprediksi frekuensi kejadian gempa bumi di wilayah Jawa Timur, sehingga dapat meningkatkan kesiapsiagaan dalam mengurangi risiko gempa bumi.

2.2.3.3 Temuan Utama

Temuan utama dari penelitian ini adalah bahwa arsitektur jaringan terbaik untuk memprediksi frekuensi kejadian gempa bumi di Provinsi Jawa Timur adalah model dengan arsitektur 9-30-1, yang menghasilkan nilai *error* sebesar 0.1958. Penelitian ini menunjukkan bahwa variasi *input* memiliki kontribusi yang signifikan dan korelasi yang lebih besar dibandingkan dengan variasi jumlah

neuron tersembunyi dalam jaringan. Selain itu, penelitian ini menggunakan data terbaru dari BMKG Stasiun Geofisika Nganjuk, yang mencatat 4029 kejadian gempa bumi selama periode 2016-2021, dan data tersebut dikelompokkan menjadi frekuensi kejadian per bulan, menghasilkan 72 *Dataset* untuk analisis.

2.2.3.4 Kelemahan Penelitian

Kelemahan dari penelitian ini adalah :

1. Rentang Data yang Terbatas

Penelitian ini menggunakan data frekuensi gempa hanya untuk periode 2016-2021. Rentang data yang terbatas ini dapat mengurangi kemampuan model dalam menangkap pola jangka panjang, yang penting dalam prediksi bencana alam seperti gempa yang bisa memiliki interval puluhan hingga ratusan tahun.

2. Hanya Menggunakan Arsitektur ANN Tunggal

Metode yang dipakai dalam penelitian ini terbatas pada ANN dengan arsitektur tertentu, yaitu kombinasi jumlah node *input* dan hidden layer. Tidak ada eksplorasi model alternatif seperti Long Short-Term Memory (LSTM) atau metode lain berbasis *Time series*, seperti SSA (Singular Spectrum Analysis) atau ARIMA. Metode alternatif tersebut mungkin memiliki keunggulan dalam menangani data *Time series* atau pola musiman.

3. Ketergantungan pada Data BMKG Lokal

Penelitian ini hanya menggunakan data dari BMKG Nganjuk, Jawa Timur, tanpa mempertimbangkan sumber data lain yang mungkin lebih lengkap, seperti

katalog USGS atau data internasional lainnya yang mencakup wilayah lebih luas atau rentang waktu lebih panjang.

4. Kurangnya Pengujian Kinerja Lintas Wilayah

Model yang dikembangkan belum diuji untuk data gempa dari wilayah berbeda, sehingga belum diketahui apakah model tersebut memiliki generalisasi yang baik atau hanya cocok untuk data Jawa Timur.

Untuk mengembangkan penelitian ini, beberapa langkah yang kemungkinan dapat diambil oleh antara lain:

1. Memperluas Rentang Data dan Sumber Data

Mengumpulkan data dari rentang waktu yang lebih panjang serta menggabungkan data dari katalog gempa internasional seperti USGS dapat membantu model menangkap pola yang lebih bervariasi dan kompleks. Ini akan meningkatkan ketepatan prediksi, terutama dalam mengidentifikasi pola frekuensi gempa dengan lebih baik.

2. Menggunakan Model Lain yang Lebih Canggih

Memanfaatkan model lain seperti LSTM, GRU, atau SSA untuk mengatasi pola non-linear atau musiman pada data gempa bumi. Model-model ini dikenal mampu menangani data *Time series* dengan lebih baik dan mungkin memberikan akurasi yang lebih tinggi dibandingkan ANN.

3. Menggabungkan Data Geospasial

Integrasi data geospasial dan faktor lain seperti kedalaman gempa, *Magnitudo*, dan jarak dari sesar aktif dapat memberikan informasi tambahan yang

dapat meningkatkan akurasi model prediksi. Model bisa dikembangkan untuk mengolah data multivariat dengan mempertimbangkan variabel-variabel tersebut.

4. Menguji Model pada Data dari Wilayah Lain

Menguji kinerja model menggunakan data dari wilayah lain akan membantu mengukur generalisasi model dan kemampuannya untuk diaplikasikan di luar Jawa Timur. Dengan demikian, model dapat dinilai secara lebih komprehensif dan dikembangkan menjadi lebih universal.

5. Optimasi Parameter Model

Menerapkan teknik optimasi hyperparameter, seperti menggunakan Grid Search atau Bayesian Optimization, dapat membantu menemukan konfigurasi terbaik dari parameter model, seperti jumlah node, learning rate, atau algoritma aktivasi yang paling cocok.