KLASIFIKASI METRIK *COMMON VULNERABILITY SCORING SYSTEM* (CVSS) BERDASARKAN DESKRIPSI *COMMON VULNERABILITIES AND EXPOSURES* (CVE) MENGGUNAKAN *BIDIRECTIONAL ENCODER FROM TRANSFORMERS*

LAPORAN TUGAS AKHIR

Laporan ini disusun guna memenuhi salah satu syarat untuk menyelesaikan program studi Teknik Informatika S1 pada Fakultas Teknologi Industri Universitas Islam Sultan Agung Semarang

Disusun Oleh:

BELGIS ANGGITA

NIM 32602100004

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM SULTAN AGUNG

SEMARANG

2025

CLASSIFICATION OF COMMON VULNERABILITY SCORING SYSTEM (CVSS) METRICS BASED ON COMMON VULNERABILITIES AND EXPOSURES (CVE) DESCRIPTIONS USING BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS

FINAL PROJECT

Proposed to complete the requirement to obtain a bachelor's degree (S1)
at Informatics Engineering Departement of Industrial Technology Faculty
Sultan Agung Islamic University

Arranged By :

BELGIS ANGGITA

NIM 32602100004

FACULTY OF INDUSTRIAL TECHNOLOGY
SULTAN AGUNG ISLAMIC UNIVERSITY
SEMARANG
2025

LEMBAR PENGESAHAN
TUGAS AKHIR

KLASIFIKASI METRIK *COMMON VULNERABILITY SCORING SYSTEM* (CVSS) BERDASARKAN DESKRIPSI *COMMON VULNERABILITIES AND EXPOSURES* (CVE) MENGGUNAKAN *BIDIRECTIONAL ENCODER FROM TRANSFORMERS*

BELGIS ANGGITA
NIM 32602100004

Telah dipertahankan di depan tim penguji ujian sarjana tugas akhir
Program Studi Teknik Informatika
Universitas Islam Sultan Agung
Pada tanggal : ..**3 Juli 2025**.....

TIM PENGUJI UJIAN SARJANA :

Ir. Sri Mulyono, M.Eng NIK. 210616049 (Ketua Penguji)		15-07-2025 .....
Imam Much Ibnu Subroto, ST,M.Sc.,Ph.D NIK. 210600017 (Anggota Penguji)	.....	14-07-2025 .....
Badie'ah, ST,M.Kom NIK. 210615044 (Pembimbing)	.....	23-07-2025 .....

Semarang, **31 Juli 2025**

Mengetahui,
Kaprodi Teknik Informatika
Universitas Islam Sultan Agung

Moch. Taufik, S.T, M.IT
NIK. 210604034

SURAT PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan dibawah ini :

Nama : Belgis Anggita

NIM : 32602100004

Judul Tugas Akhir : Klasifikasi Metrik *Common Vulnerability Scoring System* (CVSS) Berdasarkan Deskripsi *Common Vulnerabilities and Exposures* (CVE) Menggunakan *Bidirectional Encoder From Transformers*

Dengan bahwa ini saya menyatakan bahwa judul dan isi Tugas Akhir yang saya buat dalam rangka menyelesaikan Pendidikan Strata Satu (S1) Teknik Informatika tersebut adalah asli dan belum pernah diangkat, ditulis ataupun dipublikasikan oleh siapapun baik keseluruhan maupun sebagian, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka, dan apbila di kemudian hari ternyata terbukti bahwa judul Tugas Akhir tersebut pernah diangkat, ditulis ataupun dipublikasikan, maka saya bersedia dikenakan sanksi akademis. Demikian surat pernyataan ini saya buat dengan sadar dan penuh tanggung jawab.

Semarang, 16 Juli 2025

Yang Menyatakan,

Belgis Anggita

PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH

Saya yang bertanda tangan dibawah ini :

Nama : Belgis Anggita

NIM : 32602100004

Program Studi : Teknik Informatika

Fakultas : Teknologi industri

Dengan ini menyatakan Karya Ilmiah berupa Tugas akhir dengan Judul : Klasifikasi Metrik *Common Vulnerability Scoring System* (CVSS) Berdasarkan Deskripsi *Common Vulnerabilities and Exposures* (CVE) Menggunakan *Bidirectional Encoder From Transformers*

Menyetujui menjadi hak milik Universitas Islam Sultan Agung serta memberikan Hak bebas Royalti Non-Eksklusif untuk disimpan, dialihmediakan, dikelola dan pangkalan data dan dipublikasikan diinternet dan media lain untuk kepentingan akademis selama tetap menyantumkan nama penulis sebagai pemilik hak cipta. Pernyataan ini saya buat dengan sungguh-sungguh. Apabila dikemudian hari terbukti ada pelanggaran Hak Cipta/Plagiarisme dalam karya ilmiah ini, maka segala bentuk tuntutan hukum yang timbul akan saya tanggung secara pribadi tanpa melibatkan Universitas Islam Sultan agung.

Semarang, 16 Juli 2025

Yang menyatakan,

Belgis Anggita

KATA PENGANTAR

Puji syukur kepada Allah SWT, yang telah memberikan rahmat dan karunianya, sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "Klasifikasi Metrik *Common Vulnerability Scoring System* (CVSS) Berdasarkan Deskripsi *Common Vulnerabilities and Exposures* (CVE) Menggunakan *Bidirectional Encoder From Transformers*" ini dengan baik. Dengan penuh rasa hormat, penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Rektor UNISSULA Bapak Prof. Dr. H. Gunarto, SH., M.Hum yang mengizinkan penulis menimba ilmu di kampus ini.
2. Dekan Fakultas Teknologi Industri Ibu Dr. Ir. Hj. Novi Marlyana, S.T., M.T
3. Dosen pembimbing penulis Ibu Badie'ah, ST., M.Kom yang telah memberikan bimbingan dan saran dalam proses penelitian hingga penulisan laporan ini.
4. Dosen-dosen Program Studi Teknik Informatika, Fakultas Teknologi Industri UNISSULA yang telah membekali penulis dengan ilmu dan pengetahuan selama masa studi.
5. Orang tua penulis yang selalu memberikan doa, dukungan, dan motivasi selama proses penyelesaian tugas akhir ini.
6. Dan kepada seluruh pihak yang tidak dapat saya sebutkan satu per satu.

Penulis menyadari bahwa laporan masih memiliki banyak kekurangan dalam hal kuantitas, kualitas, dan ilmu pengetahuan. Oleh karena itu, penulis merima kritikan dan saran yang membangun untuk membantu laporan ini menjadi lebih baik di masa depan.

Semarang 24 Juni 2025

. Belgis Anggita

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

Dalam era digital yang semakin kompleks, identifikasi dan penilaian kerentanan keamanan siber menjadi prioritas utama bagi banyak organisasi. *Common Vulnerabilities and Exposures* (CVE) menyediakan deskripsi standar mengenai kerentanan, sementara *Common Vulnerability Scoring System* (CVSS) digunakan untuk mengukur tingkat keparahan kerentanan tersebut melalui sejumlah metrik. Namun, penilaian metrik CVSS secara manual terhadap deskripsi CVE bersifat memakan waktu, tidak konsisten, dan rawan subjektivitas. Untuk mengatasi tantangan ini, penelitian ini mengusulkan pendekatan otomatisasi klasifikasi metrik CVSS berdasarkan deskripsi CVE menggunakan metode *Natural Language Processing* (NLP). Model yang digunakan adalah *Bidirectional Encoder Representations from Transformers* (BERT), yang dilatih untuk mengklasifikasikan masing-masing metrik CVSS. Sistem ini dapat meningkatkan kecepatan, akurasi, dan konsistensi proses penilaian, serta mendukung pengambilan keputusan yang lebih objektif dalam manajemen kerentanan siber.

Kata kunci: Klasifikasi, *Common Vulnerability Scoring System*, *Natural Language Processing*, *Bidirectional Encoder Representations for Transformers*

ABSTRACT

In today's increasingly complex digital era, the identification and assessment of cybersecurity vulnerabilities have become a top priority for many organizations. The Common Vulnerabilities and Exposures (CVE) system provides standardized descriptions of vulnerabilities, while the Common Vulnerability Scoring System (CVSS) is used to measure their severity through a set of defined metrics. However, manually assessing CVSS metrics based on CVE descriptions is time-consuming, inconsistent, and prone to subjectivity. To address these challenges, this study proposes an automated approach for classifying CVSS metrics based on CVE descriptions using Natural Language Processing (NLP) techniques. The model employed is the Bidirectional Encoder Representations from Transformers (BERT), which is trained to classify each CVSS metric. This system improve the speed, accuracy, and consistency of the assessment process, and support more objective decision-making in cybersecurity vulnerability management.

Keywords: *Classification, Common Vulnerability Scoring System, Natural Language Processing, Bidirectional Encoder Representations from Transformers*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Di era digital yang terus berkembang pesat, transformasi teknologi telah mengubah cara industri beroperasi dan berinteraksi dengan pelanggan, mitra, serta pemangku kepentingan lainnya (Yang & Gu, 2021). Namun, dengan kemajuan ini, keamanan informasi menjadi salah satu aspek yang paling krusial dan harus diperhatikan oleh setiap organisasi (Renaldy dkk., 2023). Ancaman terhadap sistem informasi yang semakin kompleks memerlukan penerapan solusi keamanan yang lebih canggih untuk melindungi aset digital yang sangat berharga (Weinberg & Cohen, 2024). Risiko ancaman dan serangan siber yang terus meningkat menambah tantangan besar bagi administrator teknologi informasi dalam menjaga integritas dan kerahasiaan data yang ada dalam ekosistem digital perusahaan (Beretas, 2024).

Administrator teknologi informasi penting untuk dapat mengidentifikasi kerentanan keamanan komputer yang paling kritis dan membutuhkan penanganan segera (Ozkan-okay dkk., 2023). Untuk memfasilitasi hal tersebut, *Common Vulnerability and Exposure* (CVE) menjadi alat yang sangat berguna. Perusahaan MITRE menciptakan CVE pada tahun 1999 sebagai katalog referensi untuk mengategorikan kerentanan keamanan pada perangkat lunak dan firmware yang cukup rentan hingga berpotensi menjadi sasaran serangan siber, di mana setiap kerentanan baru akan tercatat dengan memberikan identifikasi unik, deskripsi rinci, dan rincian vektor lainnya (Simonetto & Bosch, 2024). Dengan CVE organisasi dapat mendiskusikan dan berbagi informasi mengenai kerentanan keamanan siber untuk menilai tingkat keparahan kerentanan.

Program CVE diawasi oleh CVE Editorial Board yang terdiri dari berbagai organisasi keamanan siber, akademisi, lembaga penelitian, pemerintah, dan pakar terkemuka lainnya (Sönmez, 2021). Tugas utama Dewan ini adalah menyetujui sumber data, cakupan produk, dan entri CVE, serta mengelola penugasan entri baru (Tasmiha Khan, 2024). CVE melibatkan *CVE Numbering Authorities* (CNA), entitas yang berwenang mengeluarkan ID CVE untuk kerentanan yang baru

ditemukan, berdasarkan pedoman yang telah ditetapkan dan diawasi oleh *CVE Editorial Board* (Tasmiha Khan, 2024). Setelah kerentanan tercatat, tingkat keparahannya dinilai menggunakan *Common Vulnerability Scoring System* (CVSS), yang dikelola oleh *Forum of Incident Response and Security Teams*.

Penilaian CVSS yang telah selesai akan didokumentasikan dan dipublikasikan mengenai informasi terkait kerentanan dan skor CVSS ke dalam *National Vulnerability Database* (NVD) yang dikelola oleh *National Institute of Standards and Technology* (NIST) (Shahid & Debar, 2021). Namun, proses penilaian CVSS ini dilakukan secara manual dengan menganalisis deskripsi kerentanan, akan tetapi vendor sering kali tidak memberikan detail yang cukup, sehingga mempengaruhi akurasi skor. Selain itu, beberapa metrik CVSS bersifat subjektif dan bergantung pada pengalaman sebelumnya. Proses ini juga terkendala oleh ketidaksesuaian waktu, di mana diperlukan waktu rata-rata 19 hari untuk menentukan skor CVSS, sementara kerentanan baru ditemukan setiap 6 hari (Costa dkk., 2022).

Seiring dengan perkembangan teknologi, penggunaan *Artificial Intelligence* (AI) semakin berkembang, termasuk dalam bidang keamanan siber. Salah satu penerapan AI adalah *Natural Language Processing* (NLP), yang dapat menjadi terobosan dalam menangani masalah penentuan metrik CVSS dengan menganalisis deskripsi kerentanan keamanan komputer. Beberapa penelitian telah dilakukan oleh peneliti yaitu menerapkan teknologi NLP untuk klasifikasi CVSS, salah satunya (Costa dkk., 2022) melakukan pengujian dari berbagai model NLP dari keluarga BERT seperti BERT, RoBERTa, ALBERT, hingga DistilBERT. Namun, penelitian tersebut masih memiliki kelemahan berupa ketidakseimbangan kelas dalam dataset CVSS, yang menyebabkan model cenderung bias terhadap kelas mayoritas dan mengabaikan kelas minoritas. Selain model-model diatas yang telah diuji cobakan, ada model versi lebih ringan dari BERT yaitu BERT-small yang efektif dalam memahami konteks bahasa dan meningkatkan akurasi dalam mengekstraksi informasi kritis dari deskripsi kerentanan (Shahid & Debar, 2021).

Oleh karena itu, dalam penelitian selanjutnya fokus untuk menangani permasalahan kelas tidak seimbang, seperti melalui pendekatan *resampling* untuk

meningkatkan kemampuan generalisasi model serta akurasi prediksi metrik CVSS yang lebih adil dan andal di berbagai kondisi, dan menerapkan pre-trained model BERT-small dalam mengklasifikasikan metrik CVSS.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dalam penelitian ini adalah: Bagaimana penerapan teknologi *Natural Language Processing* (NLP), dengan model *Bidirectional Encoder Representations from Transformers (*BERT) mengklasifikasi Metrik CVSS berdasarkan deskripsi *Common Vulnerabilites and Exposures*.

1.3 Batasan Masalah

Batasan masalah ini bertujuan agar permasalahan yang diuraikan tidak meluas dan mampu dibahas dengan baik sesuai batasan keilmuan peneliti. Beberapa batasan masalah yang terdapat pada penelitian ini sebagai berikut:

1. Penelitian ini hanya menggunakan deskripsi *Common Vulnerabilities and Exposures* sebagai input yang akan diproses dan dianalisis untuk memprediksi metrik CVSS.
2. Penelitian ini hanya akan menghasilkan prediksi yang akurat jika deskripsi kerentanan keamanan komputer sesuai dengan konteks keamanan dan terdefinisi dengan jelas seperti kriteria deskripsi CVE. Sementara input yang tidak relevan dengan konteks keamanan atau ambigu akan menghasilkan output yang acak atau tidak valid.
3. Penelitian ini hanya berfokus pada prediksi nilai *Base Score* CVSS dan tidak mencakup penilaian komponen lainnya seperti *Temporal Score* atau *Environmental Score*.
4. Penelitian ini tidak mempertimbangkan faktor eksternal yang dapat mempengaruhi penilaian kerentanan komputer, seperti konfigurasi sistem atau konteks lingkungan tertentu.

5. Penelitian ini hanya fokus pada penilaian risiko berdasarkan deskripsi yang ada, tidak mencakup solusi yang dapat mengurangi dampak dari kerentanannya.
6. Deskripsi CVE yang digunakan dalam penelitian ini seluruhnya menggunakan bahasa Inggris, model tidak dirancang untuk memproses input dalam bahasa lain.
7. Model tetap menghasilkan prediksi meskipun input tidak relevan atau tidak mengandung informasi kerentanan, sehingga pengguna perlu memastikan deskripsi yang dimasukkan sesuai dengan konteks keamanan.

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk menerapkan model *Natural Language Processing* (NLP) yaitu *Bidirectional Encoder Representations from Transformers* (BERT) dalam proses klasifikasi metrik CVSS berdasarkan deskripsi kerentanan keamanan komputer, guna membantu organisasi untuk mengidentifikasi kerentanan yang paling kritis dan membutuhkan penanganan segera.

1.5 Manfaat

Manfaat penelitian ini antara lain: peningkatan akurasi dalam penilaian kerentanan CVSS, mengurangi waktu yang dibutuhkan untuk penilaian CVSS, meningkatkan efisiensi dalam mengidentifikasi kerentanan keamanan komputer, memberikan referensi baru dalam penggunaan AI untuk keamanan siber, membantu pengambilan keputusan dalam manajemen risiko dari yang paling tinggi ke yang paling rendah.

1.6 Sistematika Penulisan

BAB I PENDAHULUAN: Menguraikan latar belakang masalah agar dapat dipilih sebagai judul penelitian, perumusan masalah untuk menguraikan masalah yang akan dipecahkan, batasan masalah agar ruang lingkup masalah tidak terlalu luas, tujuan yang ingin dicapai, manfaat dari pembuatan sistem, dan

sistematika penulisan yang mencakup uraian dari penulisan laporan tugas akhir.

BAB II TINJAUAN PUSTAKA DAN DASAR TEORI: Acuan untuk menyusun Tugas Akhir, mempelajari tinjauan literatur dan dasar teori yang digunakan untuk mendukung analisis masalah.

BAB III METODE PENELITIAN: Menganalisis tentang proses sistem klasifikasi CVSS menggunakan BERT. Analisa ini mencakup perancangan system dan desain antarmuka.

BAB IV HASIL DAN PEMBAHASAN: Memuat hasil pengujian program, pembahasan tentang prosedur kerja program, dan tampilannya.

BAB V KESIMPULAN DAN SARAN: Memuat tentang kesimpulan dan saran dari penulis terhadap penelitian yang telah dilakukan.

BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Common Vulnerability Scoring System (CVSS) adalah sebuah sistem standar yang digunakan untuk mengukur dan menentukan tingkat keparahan kerentanan pada perangkat lunak dan sistem informasi. Dengan menggunakan CVSS, organisasi dapat menilai seberapa besar dampak kerentanan terhadap keamanan dan integritas sistem mereka, serta memberikan pedoman untuk prioritas penanganan (Simonetto & Bosch, 2024).

Penelitian dari (Duraz dkk., 2024) mengeksplorasi penggunaan *Common Vulnerability Scoring System* (CVSS) dalam meningkatkan efektivitas sistem deteksi intrusi (*Intrusion Detection Systems*—IDS). Mereka menunjukkan bahwa dengan mempertimbangkan skor CVSS, IDS dapat lebih akurat dalam membedakan ancaman berdasarkan tingkat keparahannya. Penelitian ini juga mengusulkan metrik baru, seperti *Miss Cost (MC)* dan *False Alarm Cost (FAC)*, yang memperhitungkan skor CVSS untuk memberikan evaluasi yang lebih kontekstual terhadap kinerja IDS. Namun, pendekatan yang digunakan hanya berfokus pada deteksi berbasis pemindaian aktif dan tidak mempertimbangkan analisis deskripsi kerentanan CVE untuk memprediksi tingkat keparahan atau karakteristik teknisnya.

Pada penelitian (Elbaz dkk., 2020) mengusulkan sebuah pendekatan untuk memprediksi metrik vektor dasar CVSS dan skor keparahan terkait berdasarkan deskripsi kerentanan CVE. Deskripsi kerentanan diubah menjadi representasi *bag of words*, di mana setiap dimensi vektor mewakili jumlah kemunculan kata tertentu, dengan nilai 0 menunjukkan kata tersebut tidak ada. Kata-kata yang tidak relevan dihapus untuk mengurangi dimensi vektor kerentanannya. Model regresi linier kemudian digunakan untuk memprediksi skor untuk setiap metrik dalam vektor CVSS, dan nilai metrik ditentukan berdasarkan skor yang diprediksi. Keuntungan utama dari pendekatan ini adalah penggunaan regresi linier, yang memberikan kejelasan dalam pemahaman model karena bobot kata-kata yang relevan dapat

mengindikasikan kontribusi kata terhadap prediksi. Namun, regresi linier mengasumsikan adanya hubungan linier antara input dan output, yang membuatnya kurang efektif dalam menangani kompleksitas bahasa alami, sehingga membatasi kinerja model. Selain itu, representasi *bag of words* tidak mempertimbangkan konteks dan urutan kata, yang menghasilkan representasi data teks yang tidak optimal.

Pada penelitian (Shahid & Debar, 2021) melakukan analisis terhadap hubungan antara kata-kata yang paling sering diasosiasikan dengan nilai-nilai spesifik dari metrik CVSS untuk masing-masing kategori. Analisis ini bertujuan untuk mengidentifikasi kata kunci dalam deskripsi kerentanan keamanan siber yang membantu sistem dalam memahami dan memberikan nilai metrik CVSS yang akurat. (Shahid & Debar, 2021) juga menggunakan metode *gradient-based input saliency* untuk menjelaskan kata-kata input yang paling relevan dalam memberikan prediksi nilai untuk setiap metrik CVSS. Hasilnya, model BERT yang diuji mencapai akurasi tinggi, yaitu 83,79% hingga 96,07%. Namun, penelitian ini tidak memperhatikan *balance accuracy* sebagai metrik evaluasi tambahan, yang dapat mempengaruhi interpretasi hasil terutama dalam kasus distribusi kelas yang tidak seimbang. Meskipun akurasi tinggi tercapai, penting untuk mempertimbangkan *balance accuracy* untuk memastikan bahwa model dapat memberikan hasil yang adil di semua kategori kerentanannya, tanpa bias terhadap kelas tertentu.

Penelitian (Costa dkk., 2022) mengeksploasi penggunaan *Natural Language Processing* untuk prediksi metrik CVSS menggunakan deskripsi kerentanan keamanan komputer dari *National Vulnerability Database* (NVD). Penelitian ini melakukan percobaan dengan menggunakan teknik *text-preprocessing* seperti *lemmatization* dan *stemming*, serta penambahan kosakata sebanyak 5.000, 10.000, dan 25.000 kata. Model yang diuji meliputi *DistilBERT*, *BERT*, *RoBERTa*, dan *XLNet*. Setiap model dilatih dengan *learning rate* yang berbeda untuk setiap model, 1.5×10^{-5} untuk *RoBERTa*, 5×10^{-55} untuk *DistilBERT*, dan 3×10^{-53} untuk *BERT*, *ALBERT*, dan *DeBERTa,* serta epoch 3 untuk *BERT, ALBERT, DistilBERT* serta jumlah *epoch, batch size, weight decay* yang berbeda pada setiap model. Evaluasi performa dilakukan menggunakan metrik *accuracy, F1*

score, dan *balanced accuracy* dengan hasil penelitian menunjukkan bahwa *DistilBERT* dikombinasi dengan *lemmatization* ditambah dengan penambahan 5.000 kata memberikan performa yang terbaik dalam prediksi CVSS. Namun pendekatan dalam penelitian ini dapat menyebabkan kehilangan makna tertentu dalam teks yang lebih kompleks, terutama jika kata-kata teknis atau istilah spesifik mengalami perubahan bentuk yang mengaburkan konteks aslinya. Selain itu, *balance accuracy* yang masih rendah menunjukkan bahwa model cenderung tidak seimbang dalam memprediksi kelas tertentu, sehingga dapat menyebabkan bias dalam klasifikasi.

Dari penelitian-penelitian di atas dapat disimpulkan bahwa Teknik *Natural Language Processing* (NLP) dengan menggunakan *Bidirectional Encoder Representations from Transformers* (BERT) pada sistem prediksi metrik CVSS berdasarkan deskripsi CVE memudahkan organisasi untuk mengidentifikasi kerentanannya yang paling kritis dan membutuhkan penanganan segera.

2.2 Dasar Teori

2.2.1 *Common Vulnerability Scoring System* v3.1

Common Vulnerability Scoring System (CVSS) adalah standar internasional yang secara akurat menetapkan skor kerentanan keamanan komputer untuk membantu para profesional keamanan untuk memprioritaskan dan menangani kerentanan sesuai dengan tingkat keparahannya. CVSS ini dikelola oleh *Forum of Incident Response and Security Teams* (FIRST). Skor yang diberikan berdasarkan analisis deskripsi CVE yang tersedia dalam *National Vulnerability Database* (NVD).

Salah satu komponen utama dalam CVSS v3.1 adalah *Base Metrics*, yang mencerminkan tingkat keparahan kerentanan komputer berdasarkan karakteristik yang tidak berubah dari waktu ke waktu dan di berbagai lingkungan. Base Metrics ini terdiri dari dua kategori utama:

- *Exploitability Metrics*: Mengukur kemudahan eksploitasi suatu kerentanan sistem berdasarkan faktor seperti *Attack Vector, Attack Complexity, Privileges*

Required, dan User Interaction. Metrik ini menggambarkan seberapa mudah atau sulitnya sebuah kerentanan komputer dapat dieksploitasi oleh penyerang.

- *Scope*: Mengindikasikan apakah dampak eksploitasi hanya terbatas pada komponen yang terkena atau meluas ke komponen lain, yang berpotensi mengubah otoritas keamanan dari sistem yang terdampak.
- *Impact Metrics*: Mengukur dampak yang ditimbulkan oleh keberhasilan eksploitasi, yang meliputi *Confidentiality Impact, Integrity Impact, dan Availability Impact*. Metrik ini menilai seberapa besar kerusakan yang dapat terjadi pada aspek kerahasiaan, integritas, dan ketersediaan data atau sistem.

Tabel 2.1 merupakan contoh data deskripsi kerentanan CVE dan metrik CVSS.

Tabel 2. 1 Contoh Data Metriks CVSS

English Description	**Metriks CVSS**							
	Attack Vector	***Attack Complexity***	***Privileges Required***	***User Interaction***	***Scope***	***Confidentiality Impact***	***Integrity Impact***	***Availability Impact***
Buffer Access with Incorrect Length Value in GitHub repository radareorg/radare2 prior to 5.6.2.	*LOCAL*	*LOW*	*NONE*	*REQUIRED*	*UNCHANGED*	*HIGH*	*NONE*	*HIGH*
Mediatrix 4102 before v48.5.2718 allows local attackers to gain root access via the UART port.	*PHYSICAL*	*LOW*	*NONE*	*NONE*	*UNCHANGED*	*HIGH*	*HIGH*	*HIGH*
Remote desktop takeover via phishing	*NETWORK*	*LOW*	*NONE*	*REQUIRED*	*CHANGED*	*HIGH*	*HIGH*	*HIGH*

2.2.1.1 *CVE Description Template*

Dalam proses penilaian kerentanan keamanan komputer menggunakan *Common Vulnerability Scoring System* (CVSS), kualitas dan kelengkapan deskripsi CVE menjadi elemen krusial (Sun dkk., 2021). CVSS membutuhkan informasi kontekstual dari kerentanan yang dilaporkan, yang sebagian besar diperoleh dari deskripsi CVE (Sun dkk., 2021). Deskripsi ini mencakup jenis kerentanan (misalnya, luapan buffer, dereferensi penunjuk NULL, atau pemalsuan permintaan lintas situs) vendor produk, dan basis kode yang terpengaruh, peneliti keamanan juga dapat memberikan informasi lebih lanjut, seperti dampak yang terjadi, vektor serangan, atau status perbaikan (NVD, 2024). Berikut merupakan *template* atau kriteria deskripsi yang baik mencakup:

- *[VULNERABILY TYPE] in [COMPONENT] in [VENDOR] [PRODUCT] [VERSION] allows [ATTACKER] to [IMPACT] via [VECTOR]*
- *[COMPONENT] in [VENDOR] [PRODUCT] [VERSION] [ROOT CAUSE], which allows [ATTACKER] to [IMPACT] via [VECTOR]*

  Contoh:

Gambar 2. 1 CVE-2010-4946 NVD (Sun dkk., 2021)

Gambar 2.1 menunjukkan contoh nyata dari entri CVE-2010-4946 dalam basis data NVD. Gambar ini menandai langsung terhadap bagian-bagian penting

dari deskripsi kerentanan CVE, termasuk *Vulnerability Type*, *Component*, *Product Version*, *Attacker Type*, *Impact*, dan *Attack Vector*.

2.2.1.2 *Attack Vector* (AV)

Attack Vector (AV) adalah jalur atau metode yang digunakan oleh penyerang untuk mengakses atau mengeksploitasi kerentanan keamanan komputer di dalam suatu sistem, jaringan, atau perangkat (FIRST, 2025). Tabel 2.2 menunjukkan nilai metrik dan keterangan pada *Attack Vector*:

Tabel 2. 2 Nilai metrik *Attack Vector* (FIRST, 2025)

Kategori	Skor	Keterangan
Network(N)	0,85	Eksploitasi dilakukan dari jarak jauh melalui jaringan
Adjacent (A)	0,62	Eksploitasi dilakukan jika penyerang berada dalam jaringan yang sama
Local (L)	0,55	Eksploitasi memerlukan akses langsung ke sistem
Physical (P)	0,2	Eksploitasi memerlukan akses fisik ke perangkat

2.2.1.3 *Attack Complexity* (AC)

Attack Complexity (AC) merujuk pada tingkat kesulitan atau kompleksitas yang terlibat dalam melaksanakan suatu serangan terhadap kerentanan sistem (FIRST, 2025). Tabel 2.3 menunjukkan nilai metrik dan keterangan pada *Attack Complexity*:

Tabel 2. 3 Nilai metrik *Attack Complexity* (FIRST, 2025)

Kategori	Skor	Keterangan
Low (L)	0,77	Eksploitasi dapat dilakukan dengan mudah tanpa syarat tambahan
High (H)	0,44	Eksploitasi memerlukan kondisi spesifik, seperti perlunya bypass proteksi atau eksploitasi kondisi race condition.

2.2.1.4 *Privileges Required* (PR)

Privileges Required (PR) adalah istilah yang digunakan untuk menggambarkan sejauh mana eksploitasi suatu kerentanan sistem memerlukan akses atau kredensial tertentu pada sistem target (FIRST, 2025). Tabel 2.4 menunjukkan nilai metrik dan keterangan pada *Privileges Required*:

Tabel 2. 4 Nilai metrik *Privilileges Required* (FIRST, 2025)

Kategori	Skor	Keterangan
None (N)	0,85	Eksploitasi dapat dilakukan tanpa autentikasi atau izin khusus.
Low (L)	0,62	Penyerang memerlukan hak akses terbatas ke sistem target.
High (H)	0,27	Eksploitasi hanya dapat dilakukan oleh pengguna dengan hak administratif atau root.

2.2.1.5 *User Interaction* (UI)

User Interaction (UI) mengacu pada tindakan atau partisipasi yang diperlukan dari pengguna dalam konteks eksploitasi kerentanan sistem. UI merujuk pada sejauh mana serangan atau eksploitasi membutuhkan keterlibatan aktif dari pengguna target untuk berhasil (FIRST, 2025). Tabel 2.5 menunjukkan nilai metrik dan keterangan pada *User Interaction*:

Tabel 2. 5 Nilai metrik *User Interaction* (FIRST, 2025)

Kategori	Skor	Keterangan
None (N)	0,85	Eksploitasi dapat dilakukan tanpa keterlibatan pengguna.
Required (R)	0,62	Eksploitasi hanya berhasil jika pengguna melakukan tindakan tertentu, seperti membuka tautan berbahaya atau mengunduh file.

2.2.1.6 *Scope* (S)

Scope merujuk pada sejauh mana eksploitasi suatu kerentanan mempengaruhi sistem komputer atau komponen lainnya dalam lingkungan yang lebih besar, apakah perubahan tersebut terbatas pada komponen yang terpengaruh atau meluas ke bagian lain dari sistem (FIRST, 2025). Tabel 2.6 menunjukkan nilai metrik dan keterangan pada *Scope*:

Tabel 2. 6 Nilai metrik *Scope* (FIRST, 2025)

Kategori	Skor	Keterangan
Unchanged (U)	0,68	Eksploitasi hanya berdampak pada komponen yang rentan

Change (C)	0,5	Eksploitasi hanya berhasil jika pengguna melakukan tindakan tertentu, seperti membuka tautan berbahaya atau mengunduh file.

2.2.1.7 *Confidentiality Impact* (C)

Confidentiality Impact (C) adalah ukuran sejauh mana kerentanan sistem dapat mempengaruhi kerahasiaan informasi dalam suatu sistem. Kerahasiaan berhubungan dengan perlindungan informasi agar hanya dapat diakses oleh pihak yang berwenang (FIRST, 2025). Tabel 2.7 menunjukkan nilai metrik dan keterangan pada *Confidentiality Impact*:

Tabel 2. 7 Nilai metrik *Confidentiality Impact* (FIRST, 2025)

Kategori	Skor	Keterangan
High (H)	0,56	Penyerang dapat mengakses semua informasi sensitif tanpa batasan.
Low (L)	0,22	Sebagian informasi sensitif dapat diakses, tetapi tidak semua.
None (N)	0	Tidak ada dampak pada kerahasiaan data.

2.2.1.8 *Integrity Impact* (I)

Integrity Impact merupakan sejauh mana kerentanan sistem dapat mempengaruhi keaslian, konsistensi, dan akurasi data atau sistem. Integritas berhubungan dengan memastikan bahwa informasi tetap utuh dan tidak dimodifikasi tanpa izin atau secara tidak sah (FIRST, 2025). Tabel 2.8 menunjukkan nilai metrik dan keterangan pada *Integrity Impact*:

Tabel 2. 8 Nilai metrik *Integrity Impact* (FIRST, 2025)

Kategori	Skor	Keterangan
High (H)	0,56	Penyerang dapat melakukan perubahan besar pada data atau sistem.
Low (L)	0,22	Penyerang dapat mengubah sebagian data, tetapi dampaknya terbatas.
None (N)	0	Tidak ada dampak pada kerahasiaan data.

2.2.1.9 *Availability Impact* (A)

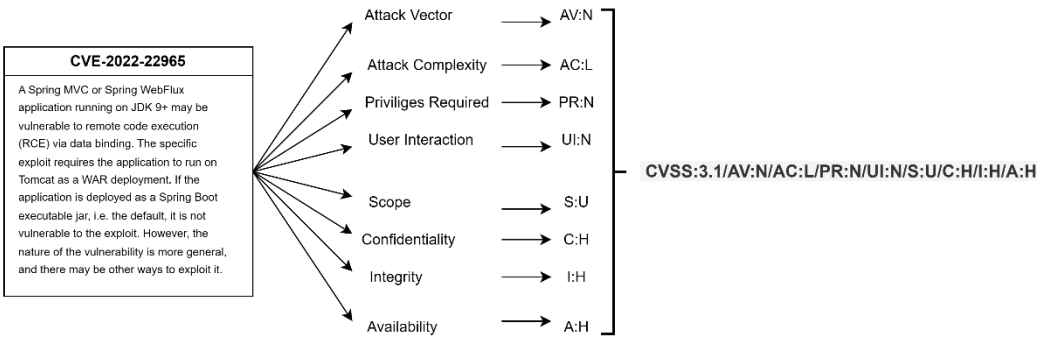
Availability Impact merupakan sejauh mana kerentanan dari sebuah sistem dapat mempengaruhi ketersediaan sistem, layanan, atau data yang dibutuhkan oleh pengguna yang sah. *Availability* berhubungan dengan memastikan bahwa informasi dan layanan dapat diakses dan digunakan oleh pihak yang berwenang saat dibutuhkan (FIRST, 2025). Tabel 2.9 menunjukkan nilai metrik dan keterangan pada *Availability Impact*:

Tabel 2. 9 Nilai metrik *Availability Impact* (FIRST, 2025)

Kategori	Skor	Keterangan
High (H)	0,56	Sistem atau layanan dapat menjadi tidak tersedia sepenuhnya.
Low (L)	0,22	Ada dampak pada ketersediaan, tetapi layanan tetap bisa diakses meskipun dengan gangguan.
None (N)	0	Tidak ada dampak pada ketersediaan sistem.

2.2.1.10 *Base Vector & Base Score*

Base Vector adalah representasi teknis dari karakteristik sebuah kerentanan dalam sistem CVSS. Ini menggambarkan faktor-faktor dasar yang mempengaruhi tingkat keparahan kerentanan, yang mencakup aspek seperti cara kerentanan sistem dapat dieksploitasi, dampaknya terhadap sistem, serta kesulitan atau kemudahan eksploitasi. Base Vector terdiri dari beberapa komponen yang saling berhubungan, yang dihitung untuk menghasilkan *Base Score*, yang menggambarkan tingkat keparahan kerentanan keamanan sistem (FIRST, 2025). Contoh:



Gambar 2. 2 Contoh *Base Vector*

Gambar 2.2 adalah contoh Base Vector dari entri CVE-2022-22965. Setiap komponen pada Base Vector diberi representasi singkatan, seperti AV:N untuk *Attack Vector: Network*, PR:N untuk *Privileges Required: None*, dan seterusnya. Seluruh kombinasi nilai komponen ini disusun dalam format yang disebut sebagai *CVSS Vector String*, yang menjadi dasar perhitungan nilai *Base Score*.

Langkah berikutnya setelah *Base Vector* ditentukan adalah penentuan *Base Score.* Penentuan ini bertujuan memberikan pemahaman yang jelas mengenai tingkat keparahan kerentanan. Semakin tinggi nilai *Base Score*, semakin besar potensi kerusakan yang bisa disebabkan oleh eksploitasi kerentanan. Berikut Tabel 2.10 yang memuat skala tingkat keparahan kerentanan CVE:

Tabel 2. 10 Skala tingkat keparahan (FIRST)

Severity Level	CVSS Score
None	0
Low	0,1 – 3,9
Medium	4,0 – 6,9
High	7,0 – 8,9
Critical	9,0 – 10,0

Langkah perhitungan:

1. *Exploitability Subscore*

$$Exploitability = 8.22 \times AV \times AC \times PR \times UI \quad (1)$$

2. *Impact Subscore* (ISS)

$$ISS = (1 - (1 - C) \times (1 - I) \times (1 - A) \quad (2)$$

- Jika *scope Unchanged*

$$Impact = 6{,}42 \times ISS \quad (3)$$

- Jika *scope Changed*

$$Impact = 7{,}52 \times (ISS - 0{,}029) - 3{,}25 \times (ISS - 0{,}02)^{15} \quad (4)$$

3. *Base Score*

- Jika Impact \<= 0, maka 0, *else*
- Jika *Scope Unchanged*

$$BaseScore = round(Min[(Impact + Exploitability), 10] \quad (5)$$

- Jika *Scope Unchanged*

$$BaseScore = round(Min[1{,}08 \times (Impact + Exploitability), 10] \quad (6)$$

2.2.2 *Natural Language Processing*

Natural Language Processing (NLP) adalah cabang *Artificial Intelligence* (Kecerdasan buatan) yang berfokus pada interaksi antara komputer dan bahasa manusia. Tujuannya adalah untuk memungkinkan komputer untuk memahami, memproses, dan menghasilkan bahasa manusia secara otomatis (Galassi dkk., 2021). NLP mencakup teknik seperti analisis sintaksis, semantik, dan pengenalan entitas, yang digunakan dalam berbagai aplikasi seperti penerjemahan otomatis, analisis sentimen, dan ekstraksi informasi dari teks (Galassi dkk., 2021).

2.2.3 Data *Pre Processing*

2.2.3.1 *Tokenizers*

Tokenization adalah proses pemecahan teks menjadi unit-unit yang lebih kecil yang disebut token. Proses ini penting dalam pemrosesan bahasa alami (NLP) karena model-machine learning tidak dapat langsung memahami teks mentah, sehingga perlu ada representasi numerik dari kata atau frase. Tokenizers bertanggung jawab untuk mengubah teks menjadi token yang relevan dan sesuai dengan kebutuhan model (Huang dkk., 2025). Selain itu, tokenizer juga dapat menangani masalah seperti kata yang belum pernah dilihat sebelumnya dengan menggunakan teknik seperti subword tokenization. Dengan memecah kata-kata atau karakter ke dalam unit-unit yang lebih kecil, model dapat memahami variasi kata dan meningkatkan kemampuan generalisasi terhadap data yang belum dikenal (Hopton dkk., 2025).

2.2.3.2 Data *Labeling*

Data labeling dalam NLP adalah proses memberi anotasi pada data untuk melatih model machine learning, seperti dalam klasifikasi teks atau analisis sentimen. Proses ini memerlukan waktu dan biaya yang besar, sehingga active learning diperkenalkan untuk memilih data yang paling informatif untuk dilabeli, mengurangi jumlah anotasi manual tanpa mengorbankan kinerja model. Berbagai strategi query seperti uncertainty sampling dan gradient-based methods digunakan untuk meningkatkan efisiensi labeling. Tugas seperti structured prediction yang

membutuhkan anotasi lebih kompleks juga dapat diatasi dengan metode partial-structure annotation. Secara keseluruhan, active learning menawarkan solusi untuk mengurangi biaya dan meningkatkan efisiensi dalam proses labeling (Zhang dkk., 2022).

2.2.3.3 *Padding* dan *Truncation*

Padding dan truncation digunakan untuk memastikan bahwa input teks memiliki panjang yang konsisten, yang penting karena model seperti BERT memerlukan input dengan panjang tetap. Padding menambahkan token khusus, seperti `[PAD]`, sehingga input teks yang lebih pendek memiliki panjang yang sesuai dengan panjang maksimum yang ditentukan. Sementara itu, truncation memotong teks yang lebih panjang dari batasan panjang input yang diinginkan. Proses ini membantu dalam mengoptimalkan pelatihan dan prediksi model, karena model dapat bekerja lebih efisien dengan input yang memiliki panjang seragam (HuggingFace, 2020a).

2.2.3.4 Konversi ke Pytorch

BERT sering digunakan bersama *framework* seperti PyTorch atau TensorFlow dalam pemrosesan bahasa alami. Untuk menggunakan BERT dengan PyTorch, data yang telah melalui proses tokenisasi dan padding dikonversi menjadi objek tensor PyTorch agar dapat diproses oleh model. Konversi ini memungkinkan integrasi langsung dengan API PyTorch yang mengandalkan struktur tensor untuk menjalankan operasi matematika, optimasi, dan inference secara efisien. Dengan menggunakan tensor PyTorch, proses pelatihan dan inferensi dapat dilakukan secara optimal, terutama dengan bantuan perangkat keras seperti GPU. Konversi ini juga mempermudah pemanfaatan pustaka seperti HuggingFace Transformers, yang menyediakan antarmuka siap pakai untuk model berbasis BERT dalam ekosistem PyTorch (HuggingFace, 2020b).

2.2.4 *Imbalance Dataset*

Imbalance data atau data tidak seimbang merupakan kondisi di mana distribusi kelas dalam suatu dataset tidak merata, yaitu jumlah data pada satu kelas (*instance*) jauh lebih banyak atau lebih sedikit dibandingkan dengan kelas lainnya. Kelas dengan jumlah data yang lebih sedikit disebut sebagai kelas minoritas

(*minority class*), sedangkan kelas dengan jumlah data yang lebih banyak disebut kelas mayoritas (*majority class*) (Carvalho dkk., 2025).

Ketidakseimbangan kelas ini merupakan hal yang umum dijumpai pada data nyata (real-world data) yang diambil langsung dari database. Kondisi ini menjadi permasalahan utama dalam proses pembelajaran mesin, karena model cenderung bias terhadap kelas mayoritas dan kesulitan dalam melakukan generalisasi terhadap kelas minoritas (Carvalho dkk., 2025).

2.2.4.1 *Text Data Augmentation*

Text Data Augmentation adalah metode yang digunakan untuk meningkatkan keberagaman data dengan membuat data baru tanpa harus mengumpulkan data baru secara langsung (Feng dkk., 2021). Teknik-teknik ini menciptakan variasi data, baik dengan memodifikasi data yang sudah ada maupun menghasilkan data sintetis (Feng dkk., 2021).

Tujuan utamanya dari *Data Augmentation* untuk memperbanyak distribusi data, sehingga dapat membantu model dalam menggeneralisasi lebih baik, mengurangi *overfitting*, dan meningkatkan performa, khususnya dalam kondisi data terbatas. *Data augmentation* telah diterapkan dalam berbagai teknik seperti substitusi sinonim, back-translation, interpolasi antar contoh, dan pemanfaatan model generatif (Feng dkk., 2021).

2.2.4.2 *Synonym Replacement*

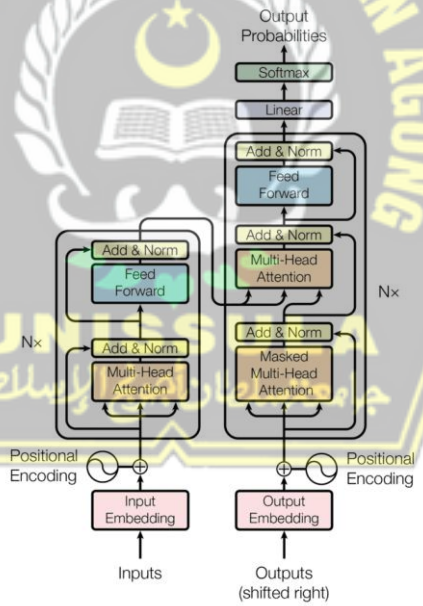
Synonym Replacement adalah teknik augmentasi data teks yang mengganti kata-kata dalam sebuah kalimat dengan sinonimnya. S*ynonym replacement* diterapkan sebagai bagian dari strategi untuk meningkatkan performa model pada tugas-tugas *natural language processing* (NLP) seperti *sentiment analysis* dan *text categorization* (Joshi dkk., 2023). Dengan mengganti kata-kata menggunakan sinonim, model diharapkan mampu menangkap makna semantik yang serupa dalam bentuk yang berbeda (Joshi dkk., 2023).

Untuk mempermudah implementasi teknik seperti *synonym replacement*, *library nlpaug* hadir sebagai solusi yang fleksibel dan komprehensif dalam melakukan augmentasi teks. *Library* ini menyediakan berbagai jenis augmenter, termasuk berbasis sinonim menggunakan *WordNet* maupun *embedding-based*

(seperti word2vec dan BERT). Dengan memanggil fungsi tertentu dalam *nlpaug* untuk langsung menerapkan augmentasi tanpa harus membangun algoritma dari awal. Dalam konteks penelitian, nlpaug tidak hanya mempercepat proses eksperimen, tetapi juga memberikan standar yang dapat direplikasi untuk berbagai tugas NLP, menjadikannya alat penting dalam pengembangan model yang lebih robust dan generalis (Joshi dkk., 2023).

2.2.5 *Transformers*

Transformer adalah sebuah arsitektur model *deep learning* yang menggunakan mekanisme *self-attention* untuk memproses hubungan semua kata dalam sebuah kalimat secara cepat dan scalable untuk dataset besar. Hal ini dikarenakan *Transformer* tidak menggunakan rekursi atau pemrosesan tanpa bergantung pada urutan input. Serta memiliki kemampuan memproses data secara paralel dan lebih efisien (Vaswani dkk., 2017).



Gambar 2. 3 Model Arsitektur *Transformers* (Vaswani et. al, 2017)

Gambar 2.3 menampilkan arsitektur umum dari model Transformer, yang terdiri atas dua komponen utama: *encoder* dan *decoder*. Setiap bagian dibangun dari lapisan-lapisan yang distack secara berurutan, terdiri dari *multi-head self-attention* dan *fully connected feed-forward networks*. Arsitektur umum ini diikuti oleh

transformator dengan menggunakan lapisan *self-attention stacked* dan *point-wise*, sepenuhnya terhubung dengan *encoder* dan *decoder*.

2.2.5.1 *Encoder* dan *Decoder*

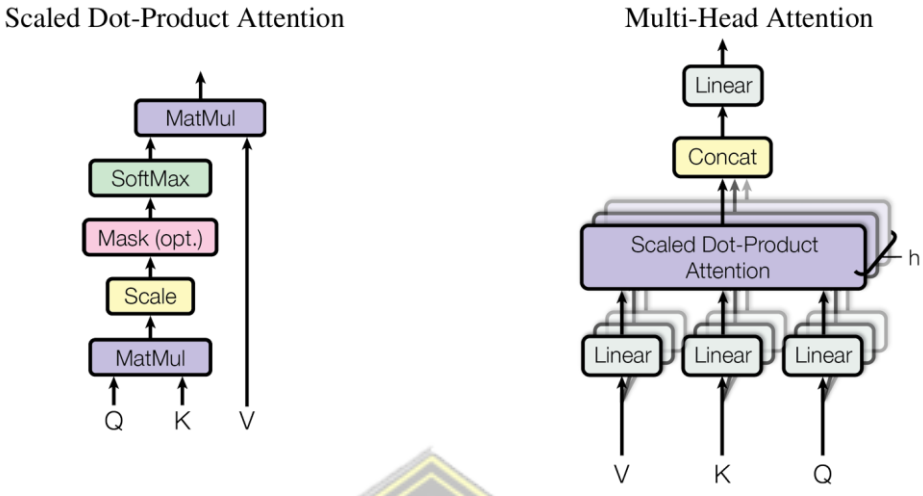
Arsitektur *Transformer* terdiri dari dua bagian utama: *encoder* dan *decoder*. *Encoder* bertugas untuk memproses *input* dan menghasilkan representasi internal, sementara *decoder* menggunakan representasi tersebut untuk menghasilkan *output*.

Encoder terdiri dari tumpukan N = 6 lapisan yang identik. Setiap lapisan memiliki dua sub-lapisan. Yang pertama adalah mekanisme *multi-head self-attention*, dan yang kedua adalah jaringan *feed-forward* yang sederhana dan terhubung sepenuhnya secara posisi.

Decoder terdiri dari tumpukan N = 6 lapisan yang identik. Selain dua sub-lapisan di setiap lapisan *encoder, decoder* menyisipkan sub-lapisan ketiga, yang melakukan *multi-head attention* dari *output* tumpukan *encoder* (Vaswani dkk., 2017).

2.2.5.2 *Attention*

Attention bekerja dengan mengukur kesamaan antara elemen-elemen dalam *input sequence* dan menentukan seberapa besar perhatian yang diberikan pada masing-masing elemen. Semakin besar kesamaan antara elemen, semakin besar perhatian yang diberikan pada elemen tersebut. *Attention* ini berfungsi untuk memfokuskan diri pada bagian-bagian penting dari input tanpa harus memproses urutan secara berurutan. Fungsi *attention* dapat digambarkan sebagai pemetaan *query* dan penetapan *key-value* ke sebuah *output* Dalam hal ini *query, key, value,* dan *output* adalah vektor, dan jumlah tertimbang dari nilai-nilai dihitung sebagai *output.* Selain itu, fungsi kompatibilitas *query* dengan *key* yang sesuai menentukan bobot yang ditetapkan untuk setiap *value* (Vaswani dkk., 2017).



Gambar 2. 4 (kiri) *Scaled Dot-product Attention*. (kanan) *Multi-head Attention* terdiri dari beberapa lapisan perhatian yang berjalan secara parallel (Vaswani et. al, 2017)

Gambar 2.4 menggambarkan dua bentuk penting dari mekanisme attention. Bagian kiri memperlihatkan *Scaled Dot-Product Attention*, yang merupakan bentuk dasar dari perhitungan perhatian berdasarkan dot product yang diskalakan. Sedangkan bagian kanan menunjukkan *Multi-head Attention*, yaitu perluasan dari mekanisme dasar dengan menjalankan beberapa lapisan perhatian secara paralel untuk menangkap informasi dari berbagai representasi subruang yang berbeda (Vaswani dkk., 2017).

2.2.5.3 *Attention Masking*

Attention masking adalah teknik yang digunakan dalam model berbasis perhatian (attention-based models) seperti Transformer untuk mengontrol bagian mana dari input yang dapat diperhatikan oleh model selama proses pelatihan atau inferensi. Masking ini sangat penting ketika menggunakan padding, karena token padding tidak seharusnya berkontribusi pada perhitungan perhatian model. Dalam model Transformer, attention mask memastikan bahwa posisi padding tidak dihitung dalam perhitungan perhatian dan karenanya tidak memengaruhi hasil model. Teknik ini membantu meningkatkan efisiensi komputasi serta kualitas prediksi yang dihasilkan (Vaswani dkk., 2017).

2.2.5.4 *Scaled Dot-Product Attention*

Scaled Dot-Product Attention adalah bentuk dasar dari mekanisme perhatian di Transformer yang menghitung kesamaan antar elemen dengan menggunakan

hasil perkalian titik antara *query* (Q), *key* (K), dan *value* (V). Ini adalah ide dasar dari *self-attention*. Perhitungan dilakukan antara vektor *query* dan vektor *key* untuk setiap pasangan *query-key* menghasilkan berbagai skor perhatian yang menunjukkan seberapa banyak komponen dalam data input berkontribusi pada komponen lain dalam konteks *self-attention.* Skalabilitas, atau akar kuadrat dari dimensi vektor kueri, dibagi oleh nilai skor perhatian agar tetap stabil saat dimasukkan ke dalam fungsi *softmax*. Untuk mengontrol banyaknya skor dan memastikan bahwa model dapat belajar perhatian yang halus, ini disebut *"scaling"*. Skor perhatian yang telah disesuaikan dengan skala digunakan sebagai bobot untuk menghitung vektor nilai rata-rata tertimbang, menunjukkan seberapa besar kontribusi setiap nilai terhadap vektor hasil akhir. Proses *Scaled Dot-Product Attention* menghasilkan vektor hasil perhatian. Mekanisme ini diulang untuk setiap pasangan kunci pertanyaan untuk memahami hubungan antara elemen-elemen tersebut. Hasilnya adalah matriks perhatian yang menunjukkan relevansi setiap elemen dalam data (Vaswani dkk., 2017). Kueri ini dikemas ke dalam sebuah matriks Q, kunci dan nilai juga dikemas ke dalam matriks K dan V (Vaswani *dkk.*, 2017). Matriks keluaran dihitung sebagai berikut:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (7)$$

Keterangan:

Q = Kueri

K = Kunci

V = Nilai

d_k = Kueri dan Kunci dimensi

d_v = Nilai dari dimensi

2.2.5.5 *Multi-Head Attention*

Multi-Head Attention adalah pengembangan dari *Scaled Dot-Product Attention*, di mana proses perhatian dilakukan beberapa kali secara paralel dengan kepala perhatian yang berbeda untuk menangkap informasi dari berbagai ruang perhatian. Pada *Multi-Head Attention*, *query*, *key*, dan *value* dibagi menjadi beberapa "*head*", dan perhatian dihitung secara terpisah untuk setiap *head*.

Setiap *head* menghitung skor perhatian mereka sendiri, yang menghasilkan sejumlah matriks perhatian yang berbeda. Ini dicapai melalui perhitungan produk dot antara vektor kueri dan vektor kunci setelah proyeksi linier. Setelah itu, *softmax* dan *scaling* digunakan untuk perhitungan perhatian. Matriks perhatian akhir dibuat dengan menggabungkan hasil dari setiap *head*. Hasil akhir yang lebih halus dapat dicapai dengan menggunakan proyeksi linier tambahan. Seperti pada *Scaled Dot-Product Attention*, matriks perhatian digunakan untuk mengambil rata-rata tertimbang dari vektor nilai. Ini menghasilkan vektor hasil yang menunjukkan interaksi antara komponen dalam data *input* (Vaswani dkk., 2017).

$$MultiHead(Q,K,V) = Concat(head_1, head_2, \dots, head_h)W^O \quad (8)$$

Dimana setiap *head* dihitung dengan:

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \quad (9)$$

Keterangan:

- $MultiHead(Q,K,V) =$ operasi *multi-head attention* mengambil tiga input, yaitu matriks *query* (Q), matriks *key* (K), dan matriks *value* (V)
- $Concat(head_1, head_2, \dots, head_h)$ = operasi penggabungan (*concatenation*) dari hasil-hasil dari setiap kepala (*head*) yang diberikan oleh fungsi *Attention.* Hasil dari setiap kepala dikonkatenasi menjadi satu vektor besar.
- W^O= matriks pembobot akhir yang digunakan untuk mentransformasi hasil penggabungan (*Concat*) dari kepala-kepala *attention* menjadi representasi akhir.
- QW_i^Q, KW_i^K, VW_i^V= digunakan untuk mengubah matriks *query* (Q), *key* (K), dan *value* (V) menjadi representasi yang sesuai untuk kepala tersebut.

2.2.5.6 *Position-wise Feed-Forward Network*

Position-wise Feed-Forward Network adalah lapisan dalam *Transformer* yang diterapkan pada setiap posisi dalam urutan secara independen setelah mekanisme perhatian. Lapisan ini terdiri dari dua lapisan linear dengan fungsi aktivasi non-linear (ReLU atau GELU) di antara keduanya. Ini digunakan untuk memproses informasi pada setiap posisi dan menghasilkan output yang lebih kompleks.

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (10)$$

Sementara transformasi linear sama tetapi di posisi yang berbeda, menggunakan parameter yang berbeda dari lapisan ke lapisan. Cara lain untuk menggambarkannya adalah sebagai dua konvolusi dengan ukuran: dimensi *input* dan *output* ($d_{model} = 512$), dan dimensi *inner-layer* ($d_{ff} = 2048$).

2.2.5.7 *Embeddings and Softmax*

Embeddings adalah representasi vektor dari kata-kata atau token dalam ruang dimensi rendah, yang digunakan untuk merepresentasikan *input* dan *output* dalam model *Transformer*. *Softmax* adalah fungsi aktivasi yang mengubah output menjadi distribusi probabilitas. *Embeddings* digunakan untuk mengubah kata atau token menjadi vektor numerik sebelum diproses oleh model. *Softmax* digunakan di layer output decoder untuk mengonversi logit menjadi probabilitas, sehingga model bisa menghasilkan token output dengan probabilitas tertinggi (Vaswani dkk., 2017).

2.2.5.8 *Positional Encoding*

Positional Encoding adalah teknik yang digunakan untuk memberi informasi urutan posisi kepada model Transformer. *Positional Encoding* ditambahkan pada *embedding input* untuk memberikan informasi mengenai posisi relatif dari kata-kata dalam urutan. Teknik ini menggunakan fungsi sinusoidal untuk menghasilkan vektor posisi. Untuk melakukannya, "*positional encodings*" harus ditambahkan ke penyematan input di bagian bawah tumpukan *encoder* dan *decoder*. *Positional Encodings* memiliki dimensi yang sama dengan vektor yang sama dengan *embedding*, sehingga keduanya dapat digabungkan. Banyak opsi untuk *positional encodings* yang dipelajari dan diperbaiki. Fungsi sinus dan kosinus digunakan dengan frekuensi yang berbeda dalam tugas ini:

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{model}}\right) \quad (11)$$

$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_{model}}\right) \quad (12)$$

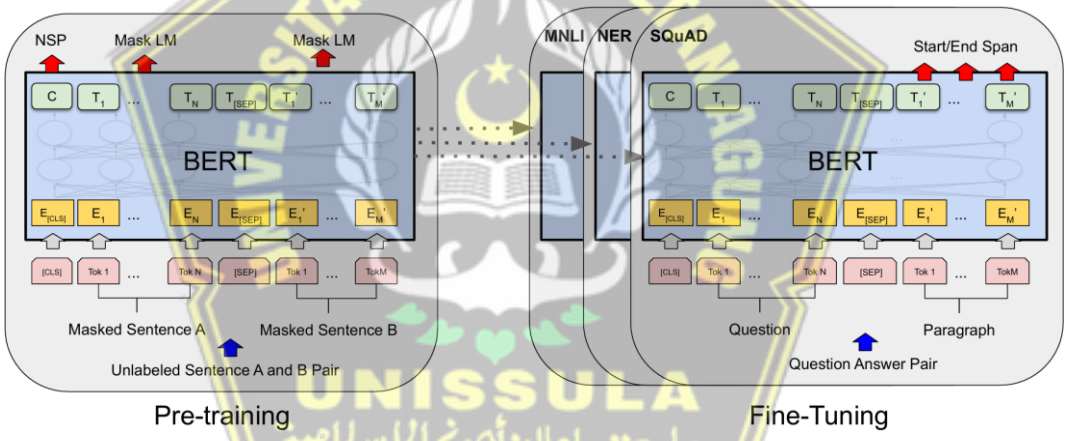
Dimensi adalah i dan pos adalah posisi. Dengan kata lain, setiap dimensi pengkodean posisi berkorelasi dengan *sinusoid*. Panjang gelombang membentuk perkembangan geometris dari 2π hingga 10000 . 2π. Fungsi ini memungkinkan model belajar dengan mudah untuk hadir dengan posisi relatif, karena PE_{pos+k}

dapat digambarkan sebagai fungsi linier PE_{pos} untuk setiap *offset* tetap k (Vaswani dkk., 2017).

2.2.6 *Bidirectional Encoder Representations from Transformers (BERT)*

Birectional Encoder Representations from Transformers (BERT) adalah model *deep learning* yang dikembangkan oleh Google untuk meningkatkan performa dari tugas NLP. BERT menggunakan arsitektur transformer untuk memahami konteksual kata-kata dalam sebuah kalimat dengan mempertimbangkan kata-kata sebelum dan sesudahnya. BERT bertujuan untuk mengajarkan representasi dua arah yang mendalam dari teks yang tidak berlabel dengan mengkondisikan secara bersama-sama disetiap lapisan konteks (Devlin dkk., 2019).

2.2.6.1 *Pre-Training dan Fine-Tuning*



Gambar 2. 5 Arsitektur *Pre-Training* dan *Fine-Tuning* dalam BERT.

Gambar 2.5 memperlihatkan arsitektur umum BERT yang digunakan dalam kedua tahap tersebut. Bagian atas menunjukkan proses *pre-training*, di mana model belajar dari teks tanpa supervisi eksplisit. Proses *pre-training* dan *fine-tuning* BERT secara keseluruhan menggunakan arsitektur yang sama, dengan *pre-training* melibatkan pelatihan model pada korpus teks besar tanpa label untuk mempelajari representasi bahasa umum. Pada tahap *pre-training*, model BERT dilatih untuk menyelesaikan dua tugas utama: *Masked Language Model* (MLM) dan *Next Sentence Prediction* (NSP). MLM mengharuskan model untuk memprediksi kata yang hilang dalam kalimat, sementara NSP bertugas memprediksi apakah dua kalimat berurutan dalam teks tersebut merupakan pasangan yang saling terkait.

Selama *pre-training*, model BERT hanya mempelajari representasi umum dari bahasa tanpa bergantung pada label khusus. Sementara itu, bagian bawah gambar menunjukkan proses *fine-tuning*, di mana model yang telah dilatih sebelumnya disesuaikan lebih lanjut menggunakan dataset yang berlabel, tergantung pada tugas spesifik yang akan diselesaikan.

Pada tahap *fine-tuning*, parameter yang telah dipelajari selama *pre-training* tidak dibekukan, melainkan dioptimalkan kembali dengan tambahan lapisan keluaran sesuai tugas, seperti *softmax layer* untuk klasifikasi. Dengan pendekatan ini, BERT mampu menghasilkan performa tinggi pada berbagai tugas pemrosesan bahasa alami, termasuk klasifikasi teks, pengenalan entitas bernama (*Named Entity Recognition*), dan menjawab pertanyaan (Devlin dkk., 2019).

2.2.6.2 *Bidirectional Context*

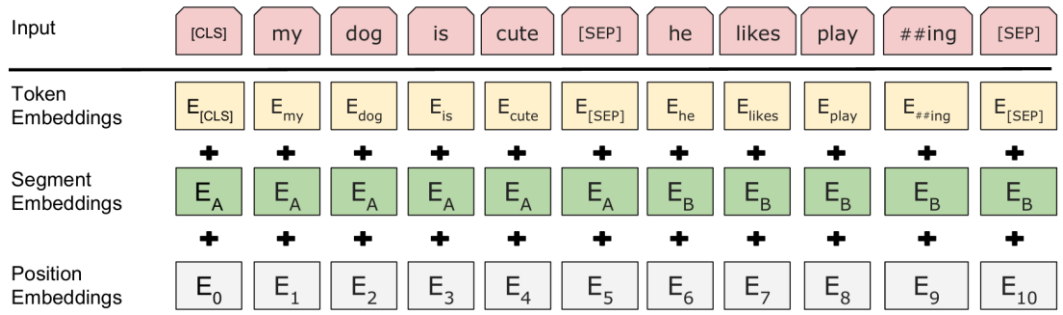
BERT unggul dalam berbagai tugas pemrosesan bahasa alami karena "*bidirectional context*", kemampuan model untuk memahami kata-kata dalam konteks dua arah, yang memungkinkan pemahaman konteks yang lebih luas dan pemrosesan bahasa yang lebih efisien (Devlin dkk., 2019).

2.2.6.3 *Masked Language Model*

Masked Language Model (MLM) adalah tugas pre-training utama pada BERT, di mana beberapa kata dalam kalimat secara acak diganti dengan token khusus [MASK], dan model diharapkan untuk memprediksi kata yang hilang tersebut (Devlin dkk., 2019).

2.2.6.4 *Next Sentence Prediction (NSP)*

Next Sentence Prediction (NSP) adalah salah satu tugas *pre-training* yang membantu model memahami hubungan antara dua kalimat dalam sebuah teks, yang sangat penting untuk tugas-tugas seperti analisis hubungan antar kalimat, pertanyaan-jawaban, atau pemahaman teks secara keseluruhan.



Gambar 2. 6 Penyematan input BERT adalah jumlah penyematan token, segmentasi, dan posisi (Devlin et al., 2019)

Gambar 2.6 menunjukkan bagaimana BERT menyusun penyematan input yang terdiri dari tiga komponen utama: *token embedding*, *segment embedding*, dan *position embedding*. *Token embedding* merepresentasikan kata atau sub-kata dalam bentuk vektor, *segment embedding* digunakan untuk membedakan antara kalimat pertama dan kedua, sedangkan *position embedding* menyandikan posisi token dalam urutan input. Tugas ini membantu model BERT memprediksi apakah kalimat kedua adalah kalimat berikutnya dari kalimat pertama (Devlin dkk., 2019).

2.2.6.5 *Self-Attention Mechanism*

Self-attention adalah mekanisme dalam *transformer* yang memungkinkan model untuk memberi perhatian pada kata-kata lain dalam kalimat untuk memperhitungkan hubungan antar kata. Mekanisme ini sangat penting dalam BERT karena memberikan cara untuk memahami konteks global sebuah kalimat. Ini adalah metode yang memungkinkan model untuk memperhatikan hubungan antara kata-kata dalam kalimat atau urutan data yang dimasukkan. Mekanisme *self-attention* terdiri dari dua operasi utama yaitu: perhitungan skor perhatian (*attention scores*) dan penggabungan jumlah berat dari nilai vektor-vektor (Devlin dkk., 2019).

2.2.6.6 *Transfer Learning*

Transfer learning adalah teknik *machine learning* di mana model yang dilatih untuk tugas tertentu digunakan kembali untuk tugas lain yang memiliki keterkaitan. BERT memanfaatkan *transfer learning* dengan menggunakan model yang sudah dilatih dan menyesuaikan model tersebut untuk tugas spesifik melalui *fine-tuning*. Setelah tahap *pre-training*, model yang telah dilatih pada data besar

digunakan untuk tugas tertentu, seperti klasifikasi atau pertanyaan-jawaban, dengan cara *fine-tuning* menggunakan dataset yang lebih kecil dan spesifik (Devlin dkk., 2019).

2.2.7 *BERT-Small*

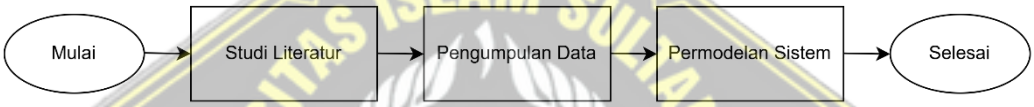
BERT-Small adalah salah satu varian dari arsitektur BERT yang dikembangkan untuk mencapai keseimbangan antara efisiensi komputasi dan kinerja. Model ini memiliki sekitar 29,1 juta parameter, jauh lebih kecil dibandingkan dengan BERT-Base (110 juta) atau BERT-Large (340 juta). Tujuan dari model ini adalah memberikan solusi yang lebih ringan dan cepat untuk tugas-tugas NLP, sambil tetap mempertahankan akurasi yang dapat diterima. Dalam penelitian ini, BERT-Small digunakan sebagai bagian dari eksperimen generalisasi dalam *Natural Language Inference* (NLI) untuk mengevaluasi kemampuannya dalam menangani dataset yang memiliki bias atau pola heuristik tertentu (Bhargava dkk., 2021).

Model BERT-Small memiliki 4 *hidden layers* dengan masing-masing 512d, 8 *attention heads*, dan *feed-forward hidden layers* sebanyak 2.048 (Distillation, 2020). Framework Huggingface digunakan untuk membaca dan memproses teks dalam blok dokumen, dengan konfigurasi pelatihan menggunakan 256 token per batch. BERT-Small dilatih menggunakan dataset yang mencakup berbagai sumber teks berbahasa Inggris, dengan total lebih dari 100 juta kata yang berasal dari korpus berita, Wikipedia, dan sumber teks lainnya. Model ini menggunakan kosakata WordPiece dengan sekitar 30.000 entri. (Bhargava dkk., 2021).

BAB III
METODOLOGI PENELITIAN

3.1 Metode Penelitian

Penelitian ini menggunakan metode *Natural Language Processing* (NLP) dengan model BERT-small atau *Bidirectional Encoder Representation from Transformers*. Keluaran dari sistem ini adalah prediksi metrik *Common Vulnerability Scoring System* (CVSS) yang mampu menganalisis tingkat kerentanan dari suatu sistem atau perangkat lunak berdasarkan deskripsi kelemahan atau *vulnerability*. Dalam penelitian ini ada beberapa langkah – langkah yang di lakukan seperti *Flowchart* pada gambar 3.1 di bawah ini:



Gambar 3. 1 *Flowchart* Metodologi Penelitian

3.1.1 Studi Literatur

Langkah pertama yang dilakukan adalah studi literatur mengenai topik penelittian terkait prediksi metrik CVSS menggunakan *Artificial Intelligence,* dilakukan dengan meninjau beberapa e-book, makalah, jurnal, tesis, skripsi terdahulu, dan berbagai situs web. Ditemukan beberapa jurnal mengenai prediksi CVSS menggunakan metode seperti *Machine Learning, Natural Language Processing, deep learning* ditemukan saat peninjauan ini. Hasil dari studi literatur ini, penulis memutuskan menggunakan model *Bidirectional Encoder Representations from Transformers* (BERT) untuk memprediksi metrik CVSS.

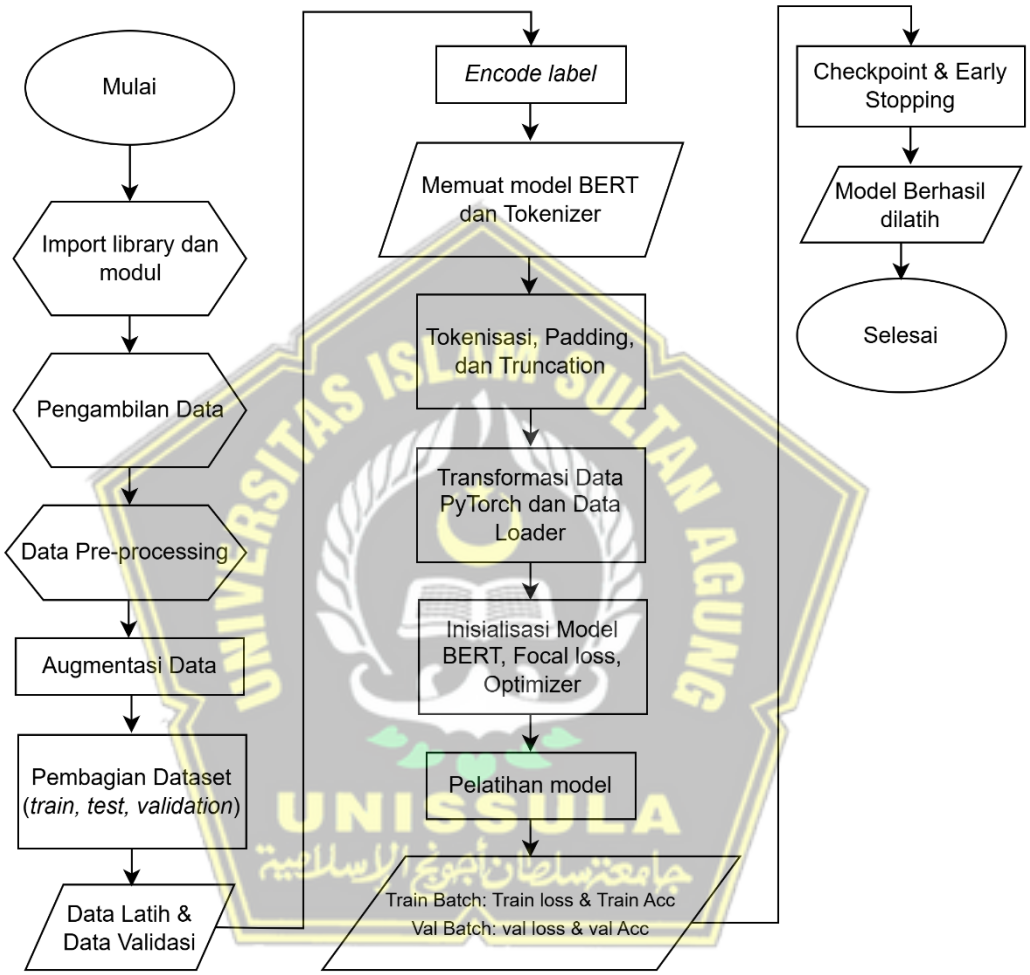
3.1.2 Pengumpulan Data

Pada tahap ini, penulis mengumpulkan dataset menggunakan metode *web data extraction* dari *National Vulnerability Database* (NVD)[1] melalui NVD CVE JSON Feed API. Data yang diambil mencakup deskripsi kerentanan, dan metrik CVSS seperti: *Attack Vector*, *Attack Complexity*, *Privilege Required*, *User Interaction*, *Scope*, *Confidentiality Impact*, *Integrity Impact*, dan *Availability*

[1] National Institute of Standards and Technology, *National Vulnerability Database*, U.S. Department of Commerce, accessed May 3, 2025, https://nvd.nist.gov/.

Impact. Deskripsi kerentanan pada dataset ini memuat informasi terkait insiden keamanan. Dataset ini lah yang akan digunakan sebagai input untuk melatih model BERT dalam memprediksi metrik CVSS.

3.1.3 Pemodelan Sistem



Gambar 3. 2 *Flowchart* sistem training dan validation Model BERT

Pada gambar 3.2 menunjukkan alur pemodelan sistem pada tahap training dan validasi model BERT untuk klasifikasi metrik CVSS. Berikut Penjelasannya:

1. Persiapan Data

   Peneliti mengambil data kerentanan CVE dari *National Vulnerability Database* (NVD) menggunakan API yang disediakan oleh NIST. File yang diunduh berupa file `.json.gz` yang berisi daftar entri CVE untuk setiap tahun dari 2022 hingga 2024. Setelah diunduh, data diekstrak dan difilter agar hanya

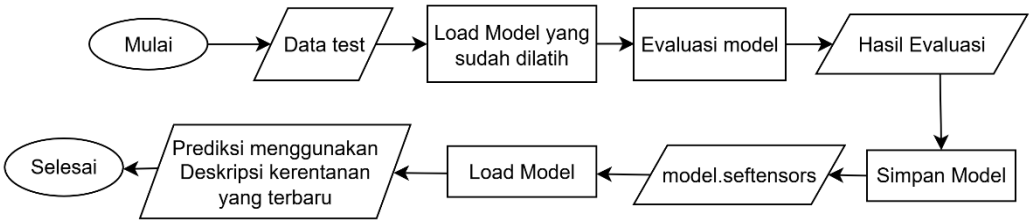
menyertakan entri dengan metrik CVSS v3 dan deskripsi dalam bahasa Inggris. Dataset hasil ekstraksi dikompilasi menjadi sebuah file CSV. Tahap ini juga mencakup analisis awal data seperti pemeriksaan jumlah kelas, distribusi perkelas, dan outlier pada panjang deskripsi.

2. *Data Preprocessing*

   Pada tahap ini dilakukan proses pembersihan teks deskripsi seperti normalisasi *unicode*, penghapusan karakter asing, *lowercase*, dan penghapusan duplikat, kemudian dilakukan augmentasi data pada kelas minor menggunakan metode sinonim berbasis NLPaug dan filtering berbasis *cosine similarity*. Setelah itu, data dibagi menjadi tiga bagian menggunakan *stratified* split agar proporsi kelas tetap seimbang: 64% data pelatihan, 16% data validasi, dan 20% data uji. Label dikodekan ke format numerik menggunakan `LabelEncoder` dari *Scikit-learn*.

3. Modeling

   Model yang digunakan adalah pretrained BERT ringan (`prajjwal1/bert-small`) yang diintegrasikan dengan *tokenizer*. *Tokenizer* melakukan tokenisasi teks, konversi token ke ID numerik, serta *padding* dan truncation hingga maksimal 128 token. Data yang telah ditokenisasi diubah menjadi PyTorch Dataset dan dimuat ke dalam *DataLoader*. Pelatihan dilakukan per epoch dengan strategi *gradual unfreezing*, di mana satu per satu layer BERT dibuka agar lebih stabil. Proses *training* menggunakan *Focal Loss* dan *optimizer* AdamW. Evaluasi dilakukan terhadap data validasi dengan menghitung metrik seperti akurasi dan loss. Jika model menunjukkan performa validasi terbaik, model disimpan ke dalam *checkpoint*.



Gambar 3. 3 *Flowchart* sistem pengujian menggunakan data *test*

Pada gambar 3.3 menunjukkan alur pengujian model menggunakan data test, setelah proses pelatihan selesai. Berikut penjelasannya:

1. Tokenisasi dan Preprocessing Data Uji

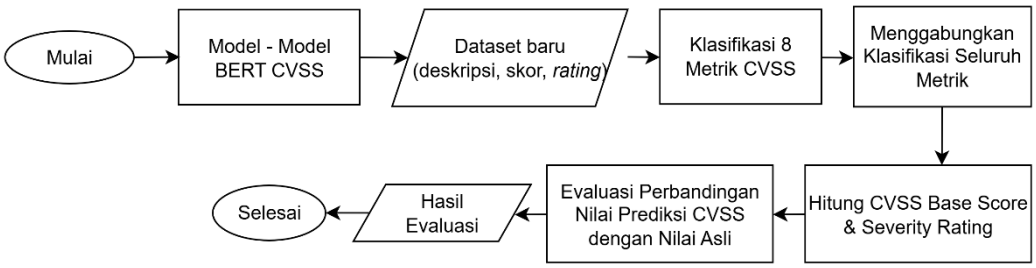
   Sama seperti proses pelatihan, data uji melalui proses tokenisasi menggunakan tokenizer BERT, kemudian dilakukan *padding* dan *truncation*. Data diubah ke dalam format tensor dan dimasukkan ke DataLoader.

2. Evaluasi Model

   Model dijalankan dalam mode evaluasi dan melakukan prediksi. Hasil prediksi dibandingkan dengan label sebenarnya pada data uji. Evaluasi dilakukan dengan menghitung metrik *accuracy*, *precision*, *recall*, *F1-score*, dan *confusion matrix*. Hasil ini mencerminkan kemampuan generalisasi model.

3. Simpan Model Final

   Jika performa model sudah baik, maka model dan tokenizer disimpan secara permanen untuk digunakan dalam implementasi prediksi data baru.



Gambar 3. 4 Evaluasi *end-to-end* Sistem Klasifikasi CVSS *Base Score* dan *Severity*

Pada gambar 3.4 menunjukkan alur evaluasi sistem secara keseluruhan (*end-to-end*), yaitu tahap implementasi model dalam skenario nyata. Deskripsi CVE dari dataset baru diproses oleh delapan model klasifikasi BERT untuk memprediksi metrik-metrik CVSS. Berikut Penjelasannya:

1. Load Dataset Evaluasi dan Model BERT

   Dataset evaluasi yang berbeda dari data training digunakan. Dataset ini hanya berisi deskripsi kerentanan CVE, nilai *base score* asli, dan *severity rating*. Seluruh model BERT yang telah dilatih (8 model untuk masing-masing metrik CVSS) dimuat beserta tokenizernya.

2. Prediksi 8 Metrik CVSS dari Deskripsi

   Setiap deskripsi diproses satu per satu oleh delapan model BERT untuk memprediksi: *Attack Vector*, *Attack Complexity*, *Privileges Required*, *User Interaction*, *Scope*, *Confidentiality Impact*, *Integrity Impact*, dan *Availability Impact*.

3. Hitung Skor dan Severity

   Hasil prediksi metrik dikombinasikan dan dimasukkan ke dalam kalkulator CVSS. Dari kalkulasi ini diperoleh nilai *base score* prediksi serta *severity level* (*Low*, *Medium*, *High*, atau *Critical*).

4. Evaluasi Akhir Sistem

   Prediksi dibandingkan dengan nilai sebenarnya untuk mengukur performa sistem secara numerik menggunakan MAE, MSE, RMSE, *Exact Match*, dan *Close Match*. Evaluasi juga dilakukan dalam bentuk klasifikasi ordinal menggunakan *precision*, *recall*, dan *F1-score* untuk setiap tingkat severity. Hasil evaluasi ini mencerminkan performa sistem dalam konteks aplikasi nyata.

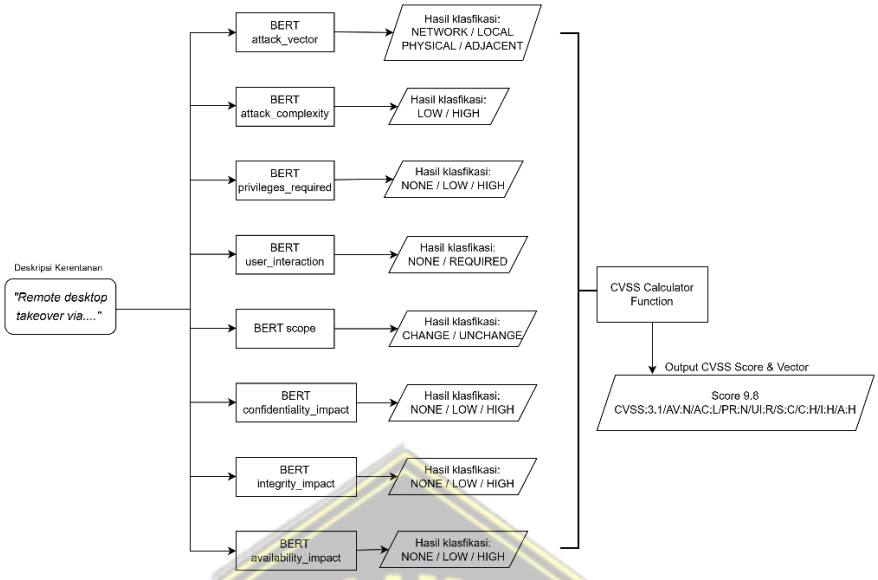
3.2 Analisis Sistem

Pada Penelitian ini penulis akan membuat sistem klasifikasi berbasis website yang bertujuan untuk memudahkan ahli kerentanan sistem dalam mengklasifikasikan kerentanan keamanan komputer. Untuk alurnya digambarkan pada gambar 3.4.

Gambar 3. 5 *Flowchart* Interaksi User dalam Proses Klasifikasi Kerentanan

Pada gambar 3.5 adalah *flowchart* yang menggambarkan proses dari sistem klasifikasi. Tahapan pada gambar 3.4 sebagai berikut:

a. Pertama *User* memasuki halaman tampilan awal dari website aplikasi klasfikasi.
b. *User* memasukkan deskripsi kerentanan sesuai dengan standar deskripsi CVE seperti gambar 2.1, kemudian *User* menekan *button* "*Submit*".
c. Sistem klasifikasi memproses input deskripsi CVE yang diberikan oleh user dengan mengklasifikikasikan kerentanan berdasarkan data yang sudah dilatih.
d. Terakhir, Sistem klasifikasi mengeluarkan hasil klasifikasi berupa *pop up* dengan isi metrik CVSS, *score* CVSS, dan *Severity*.



Gambar 3. 6 Arsitektur Sistem Klasifikasi CVSS dari Input User hingga Skor

Berdasarkan gambar 3.5 yang menunjukkan *flowchart* interaksi user, gambar 3.6 menggambarkan arsitektur tahapan ke tiga gambar 3.5 yaitu saat sistem klasifikasi memproses input deskripsi kerenanan yang diberikan oleh *User*. Arsitektur ini menjelaskan bagaimana input dari user diproses secara sistematis hingga menghasilkan skor akhir. Tahapan arsitektur sebagai berikut:

a. Deskripsi kerentanan komputer yang dikirim oleh *User* masuk kedalam sistem melalui antarmuka aplikasi. Deskripsi ini mengacu pada format standar CVE (gambar 2.1).
b. Deskripsi yang masuk kemudian melalui tahap preprocessing, seperti tokenisasi, normalisasi teks, dan pembersihan karakter tidak relevan untuk mempersiapkan input agar sesuai dengan kebutuhan model.
c. Setelah preprocessing, data dimasukkan ke dalam model klasifikasi berbasis BERT. Model ini berjumlah 8 yang mana masing-masing dilatih dan digunakan untuk mengklasifikasi satu persatu metrik CVSS: *Attack Vector*, *Attack Complexity*, *Privileges Required*, *User Interaction*, *Scope*, dan dampak *Confidentiality*, *Integrity*, *Availability*.
d. Vektor metrik hasil klasifikasi kemudian dimasukkan ke dalam *function calculator CVSS* untuk menghasilkan skor CVSS. Skor ini menggambarkan tingkat keparahan kerentanan.

e. Terakhir, sistem akan menampilkan hasil klasifikasi berupa *Base Vector* dan *Score* CVSS terhadap *User*.

3.3 Analisis Kebutuhan

Pada tahap analisis kebutuhan, peneliti memeriksa semua perangkat lunak yang diperlukan untuk membuat aplikasi klasfikasi metrik CVSS ini beroperasi dengan baik dan menghasilkan hasil yang diinginkan. Sistem ini dibuat dengan menggunakan program berikut:

1. Visual Studio Code

   Visual Studio Code (VS Code) adalah editor kode sumber yang ringan namun kuat, mendukung berbagai bahasa pemrograman dan dilengkapi dengan ekstensi yang memperkaya pengembangan perangkat lunak. Dalam penelitian ini, VS Code digunakan sebagai lingkungan pengembangan utama untuk menulis, mengelola, dan mengeksekusi kode Python serta mengelola file proyek secara efisien.

2. Python 3.11.9

   Python merupakan bahasa pemrograman tingkat tinggi yang dikenal karena sintaksnya yang sederhana dan mudah dibaca. Dalam penelitian ini, Python digunakan sebagai bahasa utama karena memiliki ekosistem library yang kaya dan mendukung pengembangan berbagai model pembelajaran mesin dan analisis data secara efisien. Penulis memilih Python versi 3.11.9 karena performa yang optimal dan kompatibilitasnya dengan pustaka terbaru.

3. *Library JSON*

   Library JSON pada Python digunakan untuk membaca dan menulis data dalam format JSON (*JavaScript Object Notation*). Dalam penelitian ini, *library* ini dimanfaatkan untuk memuat data input atau menyimpan hasil pemrosesan dalam format yang terstruktur dan mudah dipahami manusia dan mudah diproses oleh mesin

4. *Library Requests*

   Requests adalah library Python yang digunakan untuk melakukan permintaan HTTP secara sederhana dan efisien. Dalam penelitian ini, *Requests*

dimanfaatkan untuk mengambil data kerentanan CVE dari *National Vulnerability Database (NVD)* melalui API resmi yang disediakan.

5. *Library Numpy*

   NumPy adalah library Python yang digunakan untuk komputasi numerik. Ia menyediakan objek array multidimensi yang sangat efisien dan fungsi-fungsi matematika tingkat tinggi. Dalam penelitian ini, *NumPy* digunakan untuk manipulasi data numerik yang efisien, seperti pengolahan vektor atau matriks yang menjadi bagian dari proses analisis.

6. *Library Matplotlib & Seaborn*

   Matplotlib dan *Seaborn* adalah library visualisasi data di Python. Matplotlib memungkinkan pembuatan grafik dasar seperti garis, batang, dan histogram. Seaborn dibangun di atas Matplotlib dan menyediakan antarmuka tingkat tinggi untuk membuat grafik statistik yang lebih menarik dan informatif. Keduanya digunakan dalam penelitian ini untuk menyajikan hasil analisis data secara visual.

7. *Library Transformers*

   Transformers adalah library dari Hugging Face yang menyediakan implementasi model-model *deep learning state-of-the-art* untuk *natural language processing* (NLP). Dalam penelitian ini, penggunaan model NLP dengan model BERT untuk memahami bahasa dengan baik dan klasifikasi lebih efisien

8. *Library PyTorch*

   PyTorch adalah *framework deep learning* yang fleksibel dan mudah digunakan untuk penelitian dan pengembangan model *machine learning* maupun *deep learning*. Dalam penelitian ini, *PyTorch* digunakan sebagai backend untuk menjalankan model-model *neural network*, terutama untuk model transformer yang digunakan dalam proses klasifikasi. PyTorch menyediakan kemudahan dalam debugging dan pengembangan model secara iteratif.

9. *Library Pandas*

   Pandas adalah *library* Python yang powerful untuk manipulasi dan analisis data terstruktur. *Library* ini menyediakan struktur data seperti *DataFrame* yang memudahkan pembacaan, pembersihan, transformasi, dan analisis data. Dalam penelitian ini, Pandas digunakan untuk memproses dataset kerentanan, mengelola data CVSS, dan melakukan preprocessing data sebelum digunakan dalam model BERT.

10. *Library Scikit-Learn*

    Scikit-Learn adalah *library machine learning* yang komprehensif untuk Python, menyediakan berbagai algoritma untuk klasifikasi, regresi, clustering, dan preprocessing data. Dalam penelitian ini, *Scikit-Learn* digunakan untuk implementasi algoritma *machine learning* tradisional, evaluasi model (seperti *accuracy, precision, recall, F1-score*), *preprocessing* data (*scaling*, *encoding*), dan sebagai *baseline* untuk membandingkan performa dengan model *deep learning*.

11. *Library* CVSS

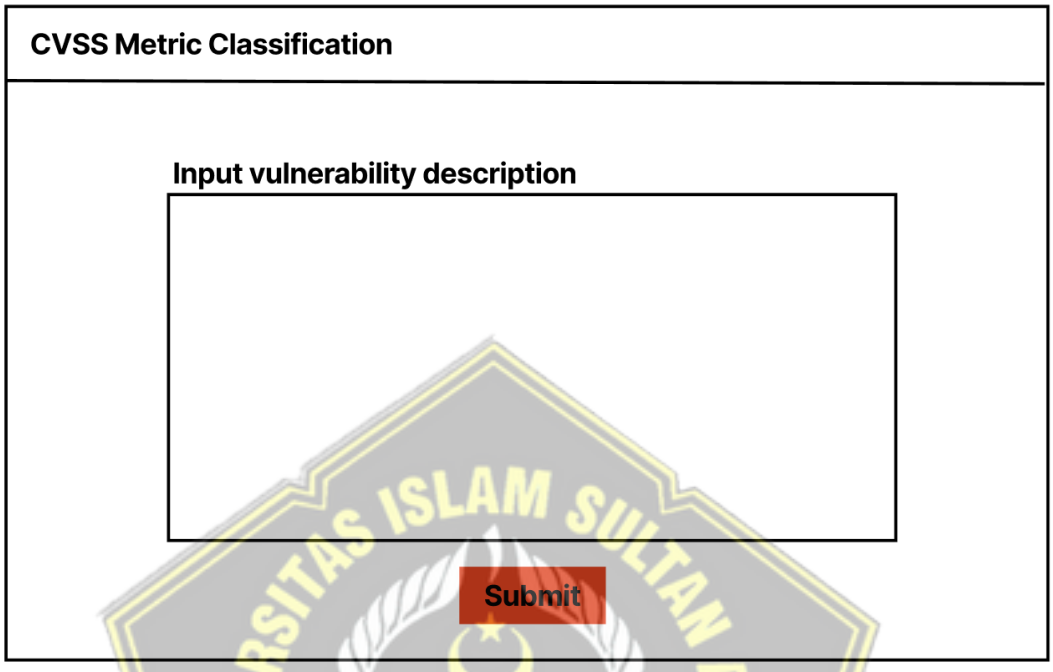
    Library CVSS adalah pustaka Python yang digunakan untuk memproses dan menghitung skor *Common Vulnerability Scoring System* berdasarkan metrik-metrik yang tersedia. Dalam penelitian ini, library ini digunakan untuk membantu validasi hasil klasifikasi dengan menghitung skor akhir CVSS berdasarkan prediksi setiap metrik, sehingga memudahkan evaluasi performa sistem secara terintegrasi.

12. *Flask*

    Flask adalah micro web *framework* untuk Python yang ringan dan fleksibel, digunakan untuk membangun aplikasi web. Dalam penelitian ini, Flask digunakan untuk mengembangkan antarmuka web aplikasi klasifikasi metrik CVSS, memungkinkan pengguna untuk mengakses sistem melalui browser web, mengunggah data kerentanan, dan melihat hasil klasifikasi dalam format yang *user-friendly* melalui web *interface*.

3.4 Perancangan User Interface

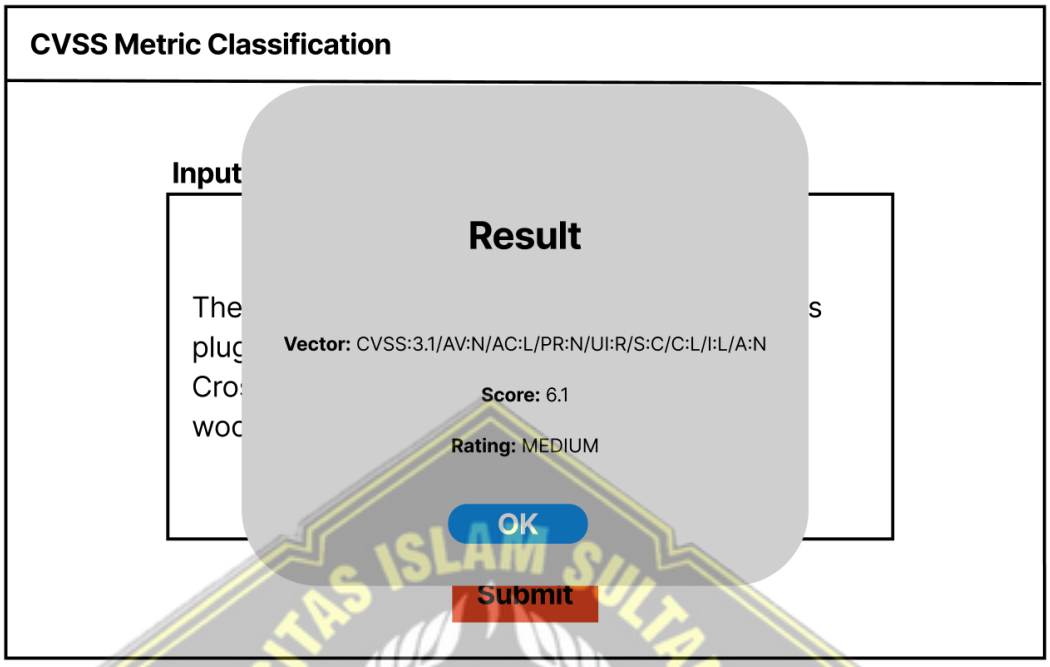
3.4.1 Halaman Awal Aplikasi



Gambar 3. 7 Halaman Awal Aplikasi

Gambar 3.7 adalah halaman awal aplikasi menampilkan antarmuka yang sederhana dengan judul "*CVSS Metric Classification*" di bagian atas. Halaman ini memiliki area input berupa text box besar dengan label "*Input vulnerability description*" yang digunakan pengguna untuk memasukkan deskripsi CVE yang ingin diklasifikasikan. Di bagian bawah terdapat tombol "Submit" yang berfungsi untuk memproses input yang telah dimasukkan pengguna. Desain halaman ini mengutamakan kesederhanaan dan kemudahan penggunaan, dengan layout yang bersih dan fokus pada fungsi utama yaitu input deskripsi kerentanan komputer.

3.4.2 Halaman Hasil Klasifikasi



Gambar 3. 8 Halaman Hasil Klasifikasi

Gambar 3.8 adalah halaman hasil klasifikasi menampilkan output dalam bentuk *pop-up* yang muncul di atas halaman utama setelah pengguna menekan tombol Submit. *Pop-up* ini menampilkan hasil analisis CVSS dalam format yang terstruktur, mencakup Vector CVSS lengkap (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N), Score numerik (6.1), dan Rating kategori (MEDIUM). Terdapat tombol "*OK*" di bagian bawah pop-up yang memungkinkan pengguna untuk kembali ke halaman input utama. Desain *pop-up* ini efektif karena memberikan informasi hasil klasifikasi secara jelas tanpa mengalihkan pengguna dari konteks halaman utama, sehingga memudahkan pengguna untuk melakukan klasifikasi berulang jika diperlukan.

BAB IV
HASIL DAN ANALISIS PENELITIAN

4.1 Hasil Penelitian

4.1.1 Persiapan Dataset

1. Pengambilan Dataset

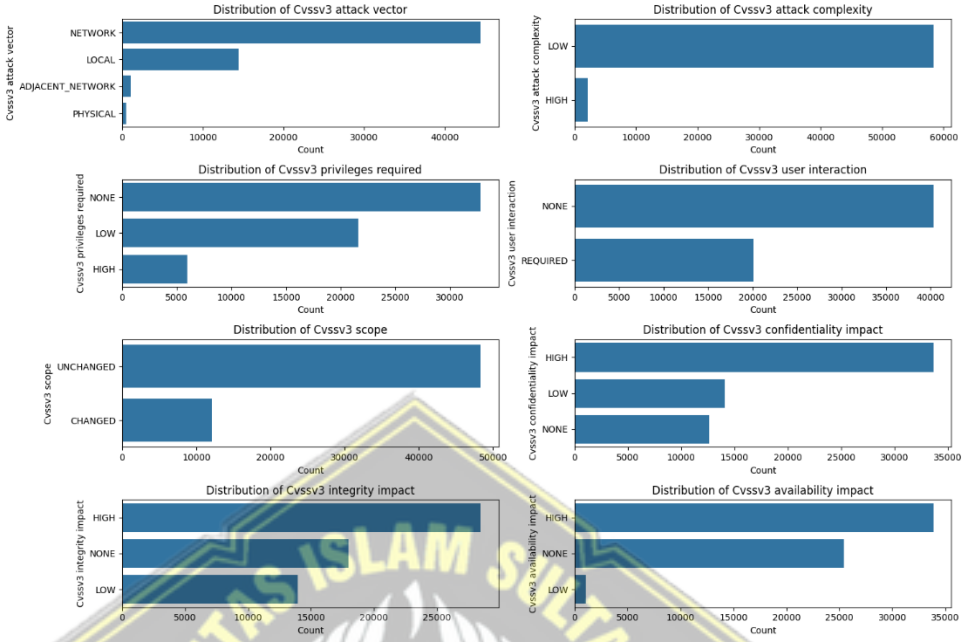
Dataset yang digunakan dalam penelitian ini merupakan data kerentanan sistem yang bersumber dari *National Vulnerability Database* (NVD). Pengambilan data dilakukan menggunakan metode *web data extraction*, di mana NVD menyediakan *open access* kepada publik. Proses ini memanfaatkan *API Data Retrieval* NVD[2], yang menyediakan data dalam format file JSON terkompresi (`.json.gz`). File-file tersebut diunduh dan diproses secara otomatis menggunakan bahasa pemrograman Python.

Pada tahap awal, dilakukan proses *filtering* dan ekstraksi informasi penting, yang difokuskan hanya pada entri yang memiliki metrik CVSS serta deskripsi kerentanan dalam bahasa Inggris. Beberapa metrik yang diambil antara lain: *Attack Vector, Attack Complexity, Privileges Required, User Interaction, Scope, Confidentiality Impact, Integrity Impact,* dan *Availability Impact*. Entri yang tidak memenuhi kriteria tersebut secara otomatis dilewati agar hasil dataset lebih relevan dan bersih.

Proses pengambilan data dilakukan dari 4 tahun yakni 2022 hingga 2024 dan seluruh hasilnya digabungkan menjadi satu dataset besar dengan jumlah 60.400 data. Selanjutnya dilakukan pembersihan data, khususnya pada kolom deskripsi dengan menghilangkan karakter *newline* dan menambahkan tanda kutip untuk menjaga konsistensi format teks. Data hasil akhir kemudian disimpan dalam bentuk file CSV untuk keperluan analisis lebih lanjut.

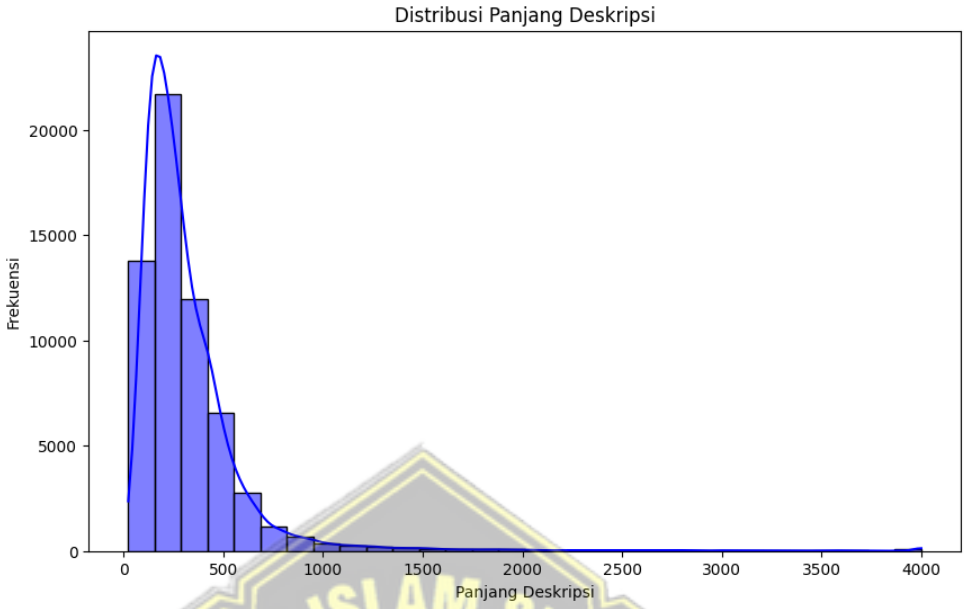
[2] URL Data NVD https://nvd.nist.gov/feeds/json/cve/1.1/nvdcve-1.1-{year}.json.gz

2. Analisis Data



Gambar 4. 1 Distribusi Metrik CVSS

Gambar 4.1 menampilkan distribusi metrik CVSS menunjukkan adanya ketidakseimbangan data yang cukup signifikan pada sebagian besar fitur. Ketimpangan ini dapat memengaruhi performa model, khususnya jika digunakan dalam klasifikasi, karena model cenderung akan lebih akurat dalam memprediksi kelas mayoritas dan mengabaikan kelas minoritas. Oleh karena itu, penting untuk mempertimbangkan teknik penanganan imbalance seperti augmentasi data atau penggunaan metrik evaluasi yang sensitif terhadap distribusi kelas saat membangun model klasifikasi, agar hasilnya tetap adil dan representatif untuk semua kategori.



Gambar 4. 2 Visualisasi Panjang deskripsi

Analisis terhadap panjang deskripsi kerentanan CVE gambar 4.2 menunjukkan distribusi yang sangat miring ke kanan (*right-skewed*). Sebagian besar deskripsi memiliki panjang antara 100-400 karakter dengan puncak distribusi berada di sekitar 200 karakter. Frekuensi tertinggi mencapai lebih dari 20.000 untuk deskripsi dengan panjang sekitar 150-200 karakter.

Pola distribusi pada gambar 4.2 mengindikasikan bahwa dokumentasi kerentanan dalam NVD mengikuti standar penulisan yang konsisten. Data outlier dengan deskripsi yang sangat panjang merupakan kerentanan yang sangat kompleks atau memerlukan penjelasan teknis yang detail.

4.1.2 *Data Preprocessing*

Tahap *preprocessing* dilakukan untuk mempersiapkan data agar dapat digunakan oleh model. Berikut merupakan tahapan *preprocessing* yang dilakukan:

1. *Dropna*, *Normalize*, dan *Lowercase*

   Pada tahap ini dilakukan proses pembersihan dan standarisasi data pada kolom deskripsi kerentanan. Langkah-langkah yang dilakukan meliputi penghapusan data kosong, normalisasi teks untuk menghilangkan karakter-karakter non-standar atau asing, penghapusan spasi berlebih, serta pengubahan seluruh teks menjadi huruf kecil. Hasil dari proses ini adalah data teks yang

lebih bersih, konsisten, dan seragam, sehingga siap digunakan untuk proses analisis lebih lanjut seperti pemodelan dan klasifikasi. Dengan mengurangi noise dalam data, proses ini membantu meningkatkan efektivitas dan akurasi pada tahap-tahap selanjutnya.

2. Hapus Duplikat

Proses penghapusan data duplikat dilakukan berdasarkan isi deskripsi kerentanan dan metrik CVSS. Duplikasi yang tidak ditangani dapat menimbulkan bias atau memperbesar bobot suatu entri secara tidak proporsional dalam analisis. Dengan menghilangkan entri-entri yang identik, data menjadi lebih representatif dan tidak berlebihan. Proses ini berkontribusi pada efisiensi analisis dan pemodelan karena setiap data yang digunakan bersifat unik.

3. Distribusi Perkelas Setelah Data Preprocessing

Setelah melalui dua tahapan preprocessing, yaitu pembersihan data dengan *dropna*, *normalize*, dan *lowercase*, serta penghapusan duplikat. Langkah selanjutnya adalah melihat distribusi data per kelas untuk memastikan bahwa data yang telah dibersihkan representatif. Berikut adalah tabel 4.1 yang menunjukkan distribusi data perkelas:

Tabel 4. 1 Distribusi perkelas setelah *Data Preprocessing*

			Target *Balance*
Attack Vector	NETWORK	42.949	42.949
	LOCAL	13.115	
	PHYSICAL	506	
	ADJACENT_NETWORK	974	
Attack Complexity	LOW	55.529	55.529
	HIGH	2004	
Privileges Required	NONE	31.653	31.653
	LOW	20.358	
	HIGH	5.616	
User Interaction	NONE	38.618	38.618
	REQUIRED	18.916	

Scope	UNCHANGE	46.031	46.031
	CHANGE	11.475	
Confidentiality Impact	HIGH	31.880	31.880
	LOW	13.563	
	NONE	12.125	
Integrity Impact	HIGH	27.060	27.060
	LOW	17.106	
	NONE	13.395	
Availability Impact	HIGH	32.100	32.100
	LOW	24.425	
	NONE	1.014	

4.1.3 Data Augmentasi

Berdasarkan Tabel 4.1, meskipun data telah melalui proses pembersihan dan normalisasi, distribusi data CVSS pada masing-masing kelas masih menunjukkan ketidakseimbangan. Untuk mengatasi ketidakseimbangan tersebut dan mencapai angka "Target *Balance*" pada masing-masing kelas sebagaimana ditampilkan pada Tabel 4.12 penulis menerapkan metode augmentasi data.

1. Augmentasi Menggunakan Sinonim

   Augmentasi dilakukan dengan menambahkan variasi data baru menggunakan teknik penggantian sinonim. roses ini dilakukan dengan bantuan library `nlpaug`, yang memanfaatkan sumber sinonim dari *WordNet* untuk menghasilkan teks-teks baru dengan sinonim yang mana maknanya tetap relevan. Proses ini membantu memperbanyak keragaman data dalam setiap kelas minoritas, serta mempertahankan konteks yang relevan untuk proses pelatihan model. Hasil augmentasi menunjukkan peningkatan jumlah data pada kelas minoritas, sehingga membantu menyeimbangkan distribusi data secara keseluruhan.

2. *Filter Cosine Similarity*

   Upaya untuk menjaga kualitas data sintetis hasil augmentasi dilakukan melalui seleksi berbasis nilai *cosine similarity*. Hanya teks hasil augmentasi yang memiliki tingkat kemiripan dalam rentang 0.85 hingga 0.95 terhadap teks

aslinya yang dipertahankan. Proses ini memastikan bahwa data baru tidak terlalu mirip sehingga redundan, dan juga tidak terlalu berbeda sehingga menyimpang dari konteks asli. Hasil seleksi ini menunjukkan bahwa variasi data yang dihasilkan tetap relevan dan memiliki kemiripan semantik yang wajar, sehingga efektif dalam memperkuat pembelajaran model tanpa mengorbankan validitas data.

3. *Self Similarity*

Setelah proses augmentasi dilakukan pada masing-masing kelas, langkah berikutnya adalah mengevaluasi kualitas hasil augmentasi pada setiap kelas dalam dataset yang telah diseimbangkan. Evaluasi ini dilakukan dengan mengukur *self similarity* antara teks-teks hasil augmentasi dan teks asli dalam setiap kelas. Untuk setiap kelas unik, terlebih dahulu diambil seluruh teks asli dari dataset mentah yang termasuk dalam kelas tersebut. Selanjutnya, representasi teks dibentuk menggunakan metode TF-IDF dengan menggunakan *TfidfVectorizer*, yang mengakomodasi unigram dan bigram serta membatasi jumlah fitur maksimum sebanyak 5000. Proses ini menghasilkan vektor representasi teks yang kemudian digunakan untuk menghitung rata-rata kemiripan antar teks dalam kelas yang sama. Nilai *self similarity* ini menjadi indikator sejauh mana variasi data hasil augmentasi tetap berada dalam konteks semantik yang sesuai dengan kelas aslinya. Berikut tabel 4.2 berisi informasi tentang nilai *Self Similarity* pada masing-masing kelas:

Tabel 4. 2 Tabel *Self Similarity*

Attack Vector	NETWORK	1.00 (kelas mayoritas tidak ada augmentasi)
	LOCAL	0.910
	PHYSICAL	0.915
	ADJACENT_NETWORK	0.914
Attack Complexity	LOW	1.00 (kelas mayoritas tidak ada augmentasi)
	HIGH	0.913
Privileges Required	NONE	1.00 (kelas mayoritas tidak ada augmentasi)

	LOW	0.910
	HIGH	0.909
User Interaction	NONE	1.00 (kelas mayoritas tidak ada augmentasi)
	REQUIRED	0.909
Scope	UNCHANGE	1.00 (kelas mayoritas tidak ada augmentasi)
	CHANGE	0.908
Confidentiality Impact	HIGH	1.00 (kelas mayoritas tidak ada augmentasi)
	LOW	0.909
	NONE	0.911
Integrity Impact	HIGH	1.00 (kelas mayoritas tidak ada augmentasi)
	LOW	0.909
	NONE	0.911
Availability Impact	HIGH	1.00 (kelas mayoritas tidak ada augmentasi)
	LOW	0.914
	NONE	0..910

Hasil evaluasi *self cosine similarity* pada tabel 4.2 menunjukkan bahwa nilai rata-rata kemiripan antara teks asli dan hasil augmentasi berada dalam rentang 0,85–0,95. Rentang ini menunjukkan bahwa makna dan konteks utama dari teks tetap terjaga, namun telah mengalami variasi semantik yang cukup untuk meningkatkan keragaman data. Nilai-nilai tersebut mengindikasikan bahwa proses augmentasi berhasil menyeimbangkan distribusi kelas tanpa menyebabkan distorsi makna. Penetapan ambang batas ini merujuk pada penelitian (Li, 2024), yang mengukur kesamaan semantik antar kalimat, dengan rentang 0,85 hingga 0,95 kemiripan dianggap masih bersifat variatif namun tetap mempertahankan makna atau konteks asli. Skor di atas 0,95 menunjukkan kemiripan yang sangat tinggi, bahkan berpotensi duplikasi, sedangkan di bawah 0,85 tingkat variasi lebih banyak namun cenderung disertai pergeseran makna atau perubahan konteks dari data aslinya. Penelitian tersebut juga menegaskan bahwa metode ini tetap robust terhadap gangguan

seperti kesalahan karakter (*OCR error*), sehingga cocok digunakan untuk mengevaluasi kestabilan makna dalam teks hasil augmentasi.

4.1.4 *Split Data (Train, Test, Validation)*

Dataset yang telah diseimbangkan kemudian dibagi menjadi tiga subset, yaitu *train*, *validation*, dan *test*, dengan menggunakan stratifikasi. Hal ini diterapkan untuk menjaga proporsi distribusi label tetap konsisten di setiap subset. Data awal dibagi menjadi 80% untuk *train* dan *validation*, serta 20% untuk *test*. Selanjutnya, dari bagian 80% tersebut, dilakukan pembagian lanjutan menjadi 64% untuk *train* dan 16% untuk *validation*. Dengan demikian, proporsi akhir pembagian dataset adalah 64% *train*, 16% *validation*, dan 20% *test*, dengan distribusi label yang seimbang pada setiap bagian. Proses ini penting untuk memastikan model dapat belajar dari data yang representatif dan divalidasi serta diuji secara adil terhadap data yang belum pernah dilihat sebelumnya.

Sebelum melakukan training model BERT, langkah penting yang harus dilakukan adalah memastikan bahwa tidak terjadi *data leakage* antara data *train, test,* dan *validation*. *Data leakage* dapat menyebabkan hasil evaluasi model menjadi tidak akurat karena informasi dari data pengujian secara tidak sengaja telah bocor ke dalam proses pelatihan, sehingga model terlihat berkinerja lebih baik dari kenyataan. Untuk mencegah hal ini, diperlukan serangkaian pemeriksaan terhadap integritas data yang telah dibagi.

1. Memeriksa duplikat data antar data yg sudah dibagi

   Langkah pertama dalam memastikan tidak terjadi data leakage adalah dengan memeriksa apakah terdapat data yang persis sama (duplikat) antara dataset pelatihan dan dataset pengujian. Hal ini penting karena keberadaan data yang identik di kedua set dapat menyebabkan model belajar dari informasi yang sudah tersedia di data pengujian, sehingga menghasilkan performa yang tampak lebih baik dari sebenarnya.

```
Literal duplicates antara TRAIN dan VALIDATION: 0
------------------------------------------------
Literal duplicates antara TRAIN dan TEST: 0
------------------------------------------------
Literal duplicates antara VALIDATION dan TEST: 0
------------------------------------------------
```

Gambar 4. 3 Menampilkan jumlah duplikat data yang sudah dibagi

Gambar 4.3 menampilkan hasil pemeriksaan jumlah duplikat literal antar kombinasi dataset *train*, *validation*, dan *test* yang menunjukkan bahwa tidak ditemukan duplikasi data antar ketiganya, sehingga dapat dipastikan tidak terjadi kebocoran data (*leakage*) pada tahap pembagian data.

2. Memeriksa *near-duplicate* data antar data yang sudah dibagi

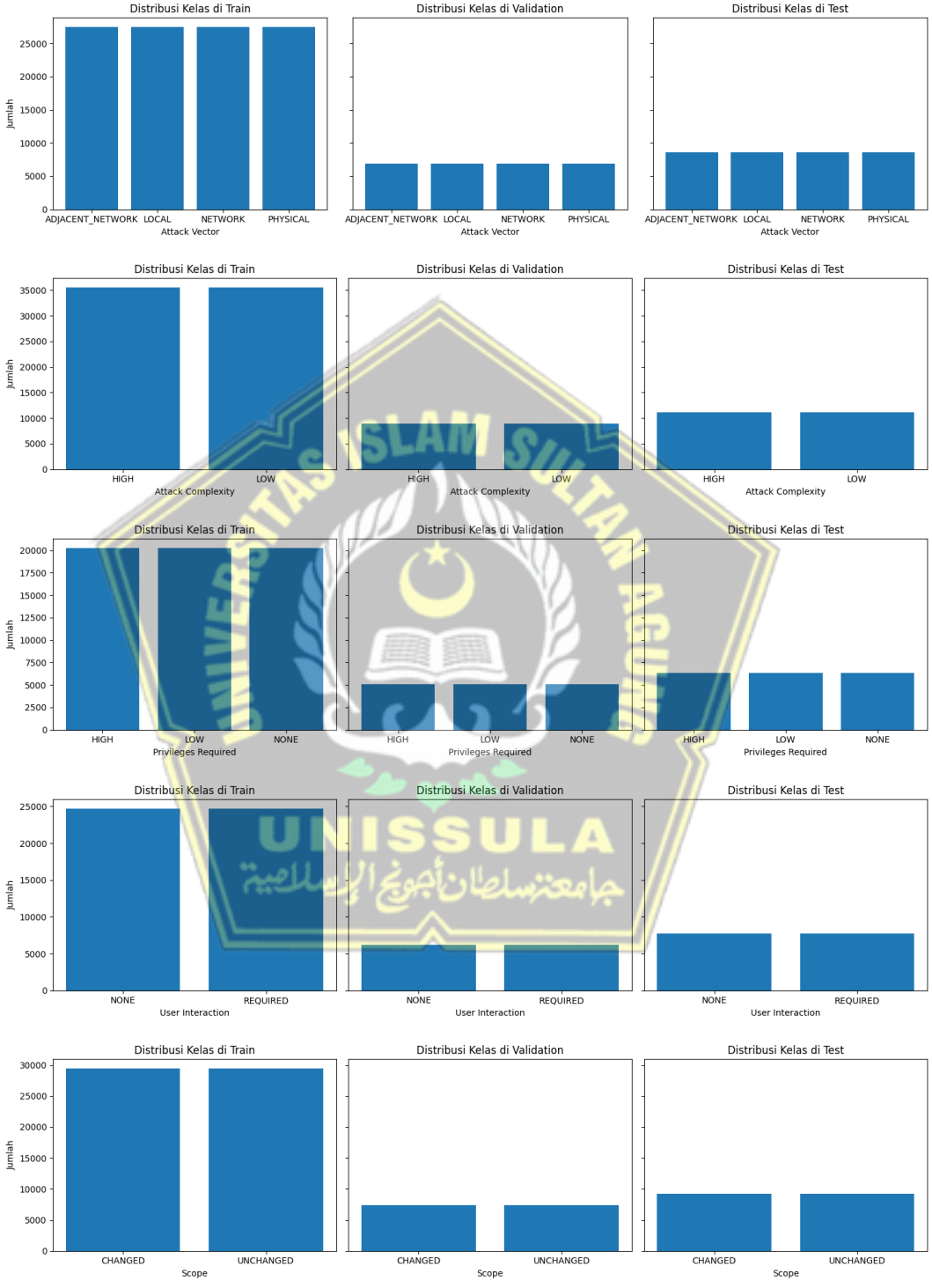
Selain duplikat yang identik, penting juga untuk memeriksa keberadaan *near-duplicate* karena sebelumnya *balancing* data dengan menggunakan teknik augmentasi yang berpotensi menghasilkan data dengan kemiripan tinggi. Untuk mengidentifikasinya, digunakan *cosine similarity*, yaitu teknik pengukuran kemiripan antara dua vektor teks berdasarkan sudut di antara data yang sudah dibagi.

Tabel 4. 3 Memeriksa *near-duplicate* antara data yang sudah dibagi

	Train dengan Validation	Train dengan Test
Attack Vector	0.8517	0.8508
Attack Complexity	0.8196	0.8182
Privileges Required	0.7987	0.7987
User Interaction	0.7750	0.7766
Scope	0.7908	0.7896
Confidentiality Impact	0.8050	0.8071
Integrity Impact	0.7864	0.7862
Availability Impact	0.8092	0.8081

Berdasarkan hasil perhitungan *cosine similarity* pada tabel 4.3 menunjukkan nilai antara 77 – 85 pada perbandingan data Train-Validation dan Train-Test, serta didukung oleh pemeriksaan *literal duplication* yang tidak menemukan adanya data yang identik, maka dapat disimpulkan bahwa dataset final yang telah melalui proses *balancing* memiliki potensi *data leakage* yang sangat minim dari *data training*. Pemisahan data telah dilakukan dengan baik

dan tidak terdapat kemiripan signifikan yang dapat memengaruhi hasil evaluasi model secara tidak adil.



Gambar 4. 4 Distribusi *train, test, validation* setiap kelas di setiap metrik CVSS

Selain itu, penting untuk memastikan distribusi setiap kelas pada masing-masing metrik CVSS berada dalam kondisi seimbang. Visualisasi yang ditunjukkan oleh gambar 4.4 menunjukkan bahwa proses *balancing* telah berhasil dilakukan dengan baik.

4.1.5 *Encode Labels*

Tahapan ini bertujuan mengubah label kelas yang semula berbentuk teks menjadi representasi numerik. Proses ini penting karena model pembelajaran mesin tidak dapat memproses label dalam bentuk teks secara langsung. Label dari data pelatihan digunakan sebagai acuan utama dalam proses encoding, dan hasilnya kemudian diterapkan secara konsisten pada data validasi dan data pengujian.

Label yang telah diencoding disimpan dalam format file terpisah agar dapat digunakan kembali, misalnya saat proses inferensi atau evaluasi model di tahap selanjutnya. Dengan menyimpan label asli, proses pelabelan balik (*decoding*) dari

output model ke format aslinya dapat dilakukan dengan mudah dan akurat. Tahapan ini menjadi krusial dalam menjaga konsistensi label selama seluruh alur pelatihan dan pengujian model.

4.1.6 Memuat Model BERT dan Tokenisasi

Pada tahap ini, digunakan model BERT ringan yaitu *prajjwal1/bert-small*, yang merupakan versi lebih kecil dari BERT standar dan telah dilatih sebelumnya pada korpus bahasa Inggris. Model ini dipilih karena memiliki ukuran lebih ringan dan waktu pelatihan yang lebih cepat, namun tetap efektif dalam menyelesaikan tugas klasifikasi teks. Proses tokenisasi dilakukan dengan batas maksimum panjang input sebesar 128 token, yang berarti setiap teks akan dipotong atau dilengkapi agar memiliki panjang seragam. Tokenisasi sangat penting karena model BERT hanya dapat menerima input dalam bentuk token numerik, bukan teks mentah. Dengan tokenisasi yang sesuai dengan model pra-latih, struktur input menjadi kompatibel dengan arsitektur internal BERT, yang mendukung kestabilan dan efektivitas pelatihan.

4.1.7 Transformasi Data ke Format Dataset dan DataLoader

Data hasil tokenisasi kemudian diubah ke dalam format yang dapat digunakan oleh PyTorch. Untuk itu, dataset dibungkus dalam struktur khusus yang memungkinkan pengambilan data per-batch secara efisien selama proses pelatihan. Seluruh data pelatihan, validasi, dan pengujian kemudian dikemas menggunakan *DataLoader*, yang memungkinkan pengolahan data dalam mini-batch serta mendukung fitur *shuffling*. Penggunaan batch size sebesar 16 bertujuan untuk menyeimbangkan antara efisiensi pemrosesan dan stabilitas pembelajaran. Tahapan ini memastikan alur pelatihan berjalan efisien dan mendukung proses optimasi yang lebih baik.

4.1.8 Inisialisasi Model dan Strategi *Fine-Tuning*

Model klasifikasi yang digunakan merupakan turunan dari *BertForSequenceClassification*, dengan arsitektur dasar *prajjwal1/bert-small*.

Model ini diinisialisasi dengan jumlah label yang sesuai dengan jumlah kelas target, dan dijalankan pada perangkat GPU untuk mempercepat proses pelatihan. Dalam tahap awal fine-tuning, seluruh parameter model dibekukan agar tidak langsung diperbarui, guna menghindari overfitting dan ketidakstabilan. Selanjutnya, diterapkan strategi *gradual unfreezing*, yaitu membuka parameter model secara bertahap mulai dari layer atas ke bawah pada setiap epoch. Strategi ini bertujuan untuk menyesuaikan model secara perlahan terhadap domain data baru dengan cara yang lebih terkontrol dan stabil.

4.1.9 *Focal Loss* untuk Menangani Ketidakseimbangan Kelas

Focal Loss mengurangi kontribusi dari sampel yang mudah diklasifikasi dan memfokuskan pelatihan pada sampel yang sulit. Untuk mengatasi potensi bias akibat ketidakseimbangan kelas yang masih tersisa setelah augmentasi, digunakan fungsi loss *Focal Loss*. Fungsi ini dirancang untuk mengurangi kontribusi dari sampel yang mudah diklasifikasi, dan lebih memfokuskan pembelajaran pada sampel yang sulit. Pengaturan nilai gamma yang digunakan bertujuan untuk meningkatkan fokus tersebut. Dengan menggunakan *Focal Loss*, model menjadi lebih sensitif terhadap kelas minoritas dan tidak terlalu didominasi oleh kelas mayoritas, sehingga meningkatkan performa klasifikasi secara menyeluruh.

4.1.10 Optimizer dan Penjadwalan Pelatihan Model

Proses pelatihan model dikendalikan menggunakan optimizer *AdamW*, yang merupakan varian dari *Adam* dengan penambahan regularisasi melalui *weight decay*. Optimizer ini hanya mengatur parameter yang sedang aktif (tidak dibekukan), sehingga mendukung strategi *unfreezing* bertahap yang telah diterapkan. Setiap kali parameter baru dibuka selama pelatihan, optimizer dibangun ulang untuk memasukkan parameter tersebut ke dalam proses pembaruan. Proses ini tidak hanya menjaga stabilitas pelatihan, tetapi juga memastikan bahwa proses fine-tuning dilakukan secara efisien dan terfokus.

4.1.11 Pelatihan Model

Tahap pelatihan dilakukan untuk melatih model menggunakan data yang telah diproses sebelumnya. Berikut merupakan tahapan pelatihan yang dilakukan:

1. Strategi *Unfreezing* Bertahap

   Proses pelatihan model dilakukan dengan pendekatan *fine-tuning* bertahap melalui strategi *gradual unfreezing*. Pada setiap epoch, hanya satu lapisan encoder dari model BERT yang diaktifkan untuk pembaruan bobot, dimulai dari lapisan paling atas. Strategi ini digunakan untuk mencegah perubahan drastis pada parameter model yang telah dilatih sebelumnya, serta memberikan adaptasi bertahap terhadap data baru. Proses pelatihan menjadi ini lebih stabil dan risiko *overfitting* dapat diminimalkan.

2. Proses Pelatihan Per *Epoch*

   Pada setiap epoch, model menjalankan proses pelatihan dengan membagi data ke dalam mini-batch. Model berada dalam mode pelatihan yang memungkinkan pembaruan parameter berdasarkan hasil prediksi dan nilai *loss*. Akurasi dan *loss* dihitung secara akumulatif selama pelatihan, sebagai indikator seberapa baik model mempelajari pola dari data latih. Akurasi dihitung berdasarkan persentase prediksi yang benar terhadap label asli, sedangkan nilai *loss* digunakan untuk mengevaluasi seberapa jauh prediksi model dari target sebenarnya.

3. Validasi Model

   Setelah setiap epoch selesai, model dievaluasi menggunakan data validasi. Pada tahap ini, model berada dalam mode evaluasi untuk menonaktifkan mekanisme pelatihan seperti *dropout*. Evaluasi dilakukan tanpa pembaruan bobot, sehingga prosesnya lebih ringan dan cepat. Nilai *loss* dan akurasi pada data validasi dihitung untuk menilai apakah model mengalami peningkatan atau justru *overfitting*. Hasil validasi ini menjadi acuan dalam menentukan langkah lanjutan pada proses pelatihan.

4. *Early Stopping* dan *Checkpointing*

   Proses pelatihan model dilengkapi dengan mekanisme *early stopping* untuk mencegah pelatihan berlebihan (*overfitting*) dan penggunaan sumber

daya secara tidak efisien. Setiap kali validasi menghasilkan nilai *val_loss* yang lebih rendah dibandingkan sebelumnya, model akan disimpan sebagai model terbaik. Namun, apabila tidak terjadi perbaikan nilai *val_loss* selama beberapa epoch berturut-turut, *patience* ditetapkan sebanyak 2 epoch, maka proses pelatihan akan dihentikan secara otomatis.

Selain *early stopping*, diterapkan pula sistem *checkpointing* yang menyimpan salinan status model pada titik-titik tertentu selama pelatihan, seperti setelah setiap epoch atau saat terjadi peningkatan performa validasi. Dengan adanya sistem ini, model terbaik dapat dengan mudah dimuat kembali untuk proses evaluasi atau prediksi, tanpa perlu mengulangi pelatihan dari awal.

5. Memuat Model Terbaik

   Setelah pelatihan selesai, bobot model terbaik yang disimpan selama proses *checkpointing* dimuat kembali. Model ini akan digunakan untuk evaluasi akhir pada data uji dan untuk kebutuhan prediksi selanjutnya. Pemilihan model terbaik berdasarkan performa pada data validasi memastikan bahwa model yang digunakan memiliki kemampuan generalisasi yang baik dan tidak hanya unggul pada data latih.

4.1.12 Evaluasi Permodel

Setelah model selesai dilatih dan bobot terbaik dimuat, langkah selanjutnya adalah melakukan evaluasi pada data uji untuk mengukur performa akhir model. Evaluasi ini bertujuan untuk melihat seberapa baik model mampu menggeneralisasi terhadap data yang belum pernah dilihat sebelumnya.

Penilaian dilakukan menggunakan metrik *balanced accuracy* untuk mengatasi ketidakseimbangan distribusi kelas. Selain itu, metrik lainnya seperti *precision*, *recall*, dan *F1-score* dihitung menggunakan pendekatan rata-rata tertimbang (*weighted average*) agar mencerminkan proporsi tiap kelas secara adil. Untuk analisis yang lebih mendalam, juga dihasilkan *confusion matrix* dan *classification report* yang menggambarkan performa model dalam mengklasifikasikan tiap kelas secara spesifik.

4.1.13 *Save Model & Predict*

Setelah proses evaluasi selesai, model yang telah dilatih dan tokenizernya disimpan agar dapat digunakan kembali tanpa perlu dilakukan pelatihan ulang. Penyimpanan ini penting untuk mendukung efisiensi dalam proses inferensi dan penerapan model di tahap produksi atau sistem riil, karena memungkinkan pemuatan cepat terhadap model yang sudah siap pakai.

```
Prompt: Vulnerability in the Oracle VM VirtualBox product of Oracle Virtualization (component: Core). Supported versions that
are affected are Prior to 7.0.24 and prior to 7.1.6. Easily exploitable vulnerability allows high privileged attacker with lo
gon to the infrastructure where Oracle VM VirtualBox executes to compromise Oracle VM VirtualBox. While the vulnerability is
in Oracle VM VirtualBox, attacks may significantly impact additional products (scope change). Successful attacks of this vuln
erability can result in unauthorized creation, deletion or modification access to critical data or all Oracle VM VirtualBox a
ccessible data as well as unauthorized read access to a subset of Oracle VM VirtualBox accessible data and unauthorized abili
ty to cause a partial denial of service (partial DOS) of Oracle VM VirtualBox. CVSS 3.1 Base Score 7.3 (Confidentiality, Inte
grity and Availability impacts).
Predicted Attack Vector: LOCAL (89.84%)

Probabilities per class:
 - ADJACENT_NETWORK    :  0.22%
 - LOCAL               : 89.84%
 - NETWORK             :  9.69%
 - PHYSICAL            :  0.24%
```

Gambar 4. 5 *Predict new case*

Pada Gambar 4.5 menunjukkan hasil prediksi model terhadap data baru. Dalam proses ini, deskripsi kerentanan baru dimasukkan sebagai input, kemudian diproses melalui tokenizer dan model yang telah disimpan sebelumnya. Output prediksi menampilkan label dengan probabilitas tertinggi beserta tingkat kepercayaannya, serta distribusi probabilitas ke seluruh kelas yang ada. Hasil ini memberikan gambaran yang jelas dan informatif mengenai bagaimana model menginterpretasikan input dan menentukan klasifikasinya secara probabilistik.

4.1.14 Evaluasi Keseluruhan Model

Setelah dilakukan pelatihan dan evaluasi terhadap masing-masing model BERT untuk setiap metrik CVSS (*Base Metrics*), langkah selanjutnya adalah menguji performa sistem secara keseluruhan dalam konteks implementasi aplikasi. Evaluasi ini bertujuan untuk mengukur sejauh mana akurasi sistem *end-to-end* dalam memprediksi nilai CVSS *Base Score* dan kategorisasi *severity rating* berdasarkan input deskripsi CVE.

Proses evaluasi menyeluruh dilakukan dengan memproses deskripsi dari dataset evaluasi melalui sistem klasifikasi yang terdiri dari delapan model BERT. Dataset evaluasi yang digunakan berbeda dengan dataset pelatihan. Dataset ini

berisi deskripsi kerentanan, nilai *base score* asli, serta *severity rating*-nya (*None, Low, Medium, High, Critical*), tanpa memuat label-label metrik CVSS.

Hasil klasifikasi masing-masing metrik kemudian dikombinasikan dan dimasukkan ke dalam kalkulator CVSS untuk menghasilkan prediksi nilai base *score* dan *severity rating*. Untuk mengukur seberapa dekat hasil prediksi sistem terhadap nilai asli dari dataset.

1. Regresi Numerik
    - *Mean Absolute Error* (MAE): mengukur rata-rata selisih absolut antara skor prediksi dan skor aktual.
    - *Mean Squared Error* (MSE): mengukur rata-rata kuadrat dari selisih prediksi terhadap nilai sebenarnya.
    - *Root Mean Squared Error* (RMSE): akar dari MSE, yang memberikan interpretasi dalam satuan yang sama dengan skor CVSS.
    - *Exact Numeric Match*: proporsi prediksi yang benar-benar identik dengan skor aktual.
    - *Close Numeric Match*: proporsi prediksi yang berbeda tidak lebih dari ±1 poin dari skor aktual.
2. Klasifikasi Multikelas

    Skor CVSS secara resmi dikategorikan ke dalam level keparahan (*severity*) yaitu *LOW*, *MEDIUM*, *HIGH*, dan *CRITICAL*, sehingga dilakukan pula evaluasi dalam bentuk klasifikasi multikelas. Beberapa metrik evaluasi yang digunakan antara lain:

    - *Precision, Recall,* dan *F1-Score* untuk masing-masing kelas maupun secara keseluruhan.
    - *Weighted average* sebagai indikator performa rata-rata dengan mempertimbangkan distribusi jumlah sampel pada tiap kelas.

4.2 Hasil Evaluasi

4.2.1 Hasil Evaluasi Per Model Metrik CVSS

Tabel 4. 4 Evaluasi *training, validation, test*

Kategori	*Training*		*Validation*		*Test*	
	Loss	***Accuracy***	***Loss***	***Accuracy***	***Loss***	***Accuracy***
Attack Vector	0,0179	0,9793	0,0176	0,9801	0,0185	0,9789
Attack Complexity	0,0105	0,9863	0,0135	0,9836	0,0126	0,9841
Privileges Required	0,0579	0,9232	0,1053	0,8851	0,1087	0,8860
User Interaction	0,0341	0,9533	0,0369	0,9496	0,0360	0,9503
Scope	0,0210	0,9742	0,0212	0,9749	0,0207	0,9770
Confidentiality Impact	0,0645	0,9243	0,0913	0,9049	0,0891	0,9075
Integrity Impact	0,0541	0,9371	0,1030	0,9032	0,1048	0,9031
Availability Impact	0,0401	0,9485	0,0522	0,9411	0,0573	0,9376

Tabel 4.4 menunjukkan nilai loss dan akurasi untuk setiap kategori CVSS pada data training, validasi, dan pengujian. Secara keseluruhan, model menunjukkan performa yang stabil dan tinggi, dengan sebagian besar kategori memperoleh akurasi di atas 90%. Kategori seperti *Attack Vector*, *Attack Complexity*, dan *Scope* memiliki nilai loss yang sangat rendah dan akurasi yang tinggi pada ketiga data, menunjukkan bahwa model mampu melakukan generalisasi dengan baik. Namun, kategori *Privileges Required* dan *Integrity Impact* memiliki nilai loss yang lebih tinggi dan akurasi yang lebih rendah, terutama pada data validasi dan pengujian, yang menunjukkan bahwa kategori ini lebih sulit diprediksi secara akurat, dikarenakan ketidakseimbangan kelas atau kompleksitas fitur yang lebih tinggi.

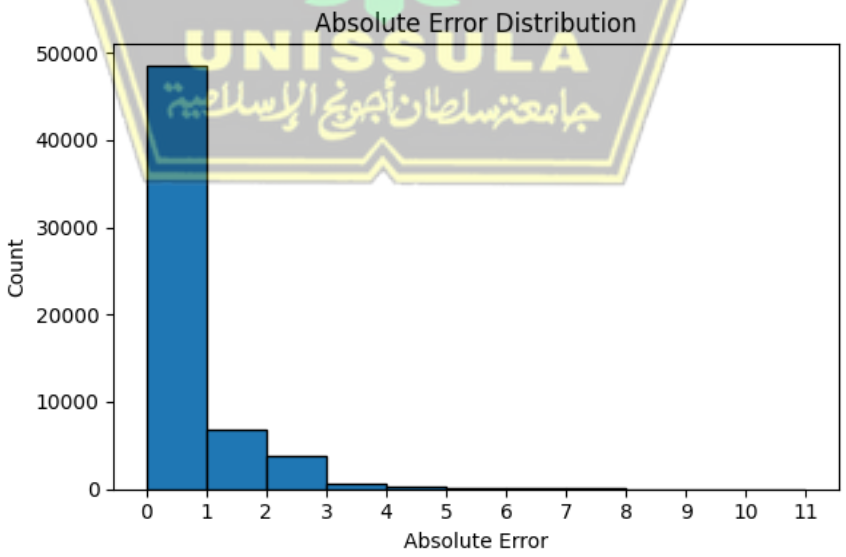
Tabel 4. 5 Evaluasi Metrik per Model Metrik CVSS

Categories	***Accuracy***	***Balance Accuracy***	***Precision***	***Recall***	***F1-Score***
Attack Vector	0,9789	0,9789	0,9789	0,9789	0,9789
Attack Complexity	0,9841	0,9841	0,9841	0,9841	0,9841
Privileges Required	0,8860	0,8860	0,8855	0,8860	0,8857

User Interaction	0,9503	0,9503	0,9504	0,9503	0,9503
Scope	0,9770	0,9770	0,9770	0,9770	0,9770
Confidentiality Impact	0,9075	0,9075	0,9075	0,9075	0,9073
Integrity Impact	0,9031	0,9031	0,9034	0,9031	0,9032
Availability Impact	0,9376	0,9376	0,9375	0,9376	0,9375

Tabel 4.5 menunjukkan performa model klasifikasi untuk masing-masing kategori CVSS pada data pengujian. Penilaian metrik dihitung termasuk *accuracy*, *balanced accuracy*, *precision*, *recall*, dan *F1-score*. Secara umum, model menunjukkan performa tinggi dan konsisten, dengan akurasi dan *balanced accuracy* berkisar antara 88,60% hingga 98,41%. Kategori yang paling mudah diprediksi adalah *Attack Complexity* (AC) dan *Scope* (S), yang keduanya mencatat *accuracy* dan *balanced accuracy* tertinggi sebesar 98,41% dan 97,70%. Di sisi lain, kategori *Privileges Required* (PR) menjadi yang paling sulit, dengan *accuracy* dan *balanced accuracy* terendah yaitu 88,60%. Tingginya nilai *balanced accuracy* pada sebagian besar kategori menunjukkan bahwa model mampu mengklasifikasikan masing-masing kelas secara merata dan tidak bias terhadap kelas tertentu.

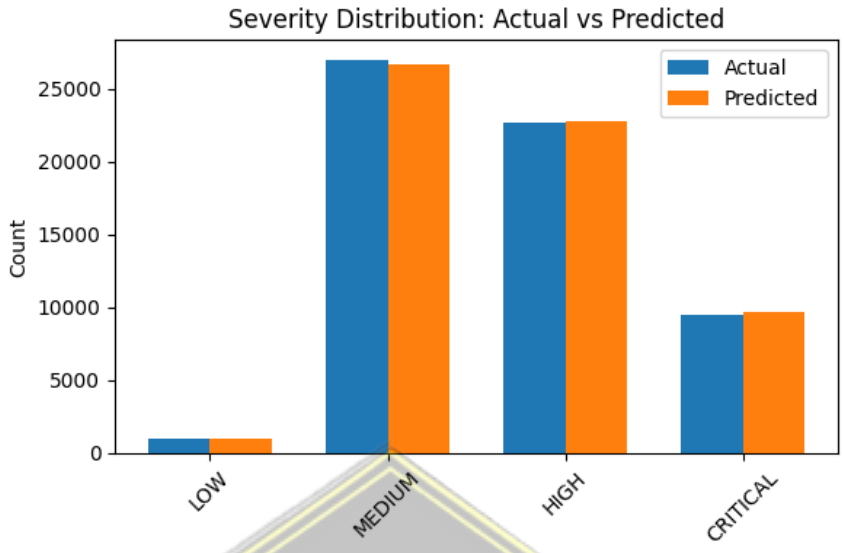
4.2.2 Hasil Evaluasi Keseluruhan model



Gambar 4. 6 *Absolute error distrbution*

Distribusi kesalahan absolut ditampilkan pada gambar 4.6, yang menunjukkan bahwa sebagian besar hasil prediksi hanya berbeda sedikit dari nilai sebenarnya yaitu selisihnya kurang dari satu angka pada skala skor CVSS. Dari hasil prediksi vektor CVSS Base, di hitung skor tingkat keparahan CVSS untuk setiap kerentanan CVE pada data uji. *Mean Squared Error* (MSE) dan *Root Mean Squared Error* (RMSE) antara skor prediksi dan skor sebenarnya masing-masing adalah 0.9543 dan 0.9769. Sementara itu, *Mean Absolute Error* (MAE) tercatat sebesar 0.4301. Skor prediksi yang benar-benar sama dengan skor sebenarnya (MAE = 0) ditemukan pada 70.88% kerentanan CVE dalam data uji. Selain itu, nilai MAE kurang dari 1 pada 80.35% kerentanan, yang menunjukkan bahwa sebagian besar prediksi memiliki tingkat kesalahan yang kecil terhadap nilai sebenarnya.

Secara teoretis, skor CVSS berada dalam rentang 0.0 hingga 10.0, sehingga nilai maksimum MAE secara matematis adalah 10.0, yang dapat terjadi hanya dalam skenario ekstrem ketika semua prediksi sepenuhnya meleset, misalnya semua label sebenarnya adalah 10.0 namun model selalu memprediksi 0.0. Dalam praktiknya, model yang tidak terlatih atau hanya menebak rata-rata biasanya menghasilkan MAE yang jauh lebih tinggi, berkisar antara 3.0 hingga 5.0. Oleh karena itu, nilai MAE sebesar 0.4301 yang diperoleh dalam penelitian ini mencerminkan performa model yang sangat baik dan presisi tinggi. Sebagai perbandingan, model CVSS-BERT oleh (Shahid & Debar, 2021) mencatat MSE sebesar 1.79 dan RMSE sebesar 1.34, dengan 55.3% prediksi identik (exact match) dan 75% prediksi memiliki selisih kurang dari satu poin (close match). Dengan MAE sebesar 0.4301, serta 70.88% *exact match* dan 80.35% *close match* pada penelitian ini, menunjukkan performa model yang lebih baik daripada penelitian sebelumnya.



Gambar 4. 7 *Severity Distribution: Actual VS Predicted*

Gambar 4.7 memperlihatkan perbandingan distribusi tingkat keparahan antara nilai sebenarnya dan hasil prediksi. Tingkat keparahan dikelompokkan dalam empat kategori: *Low*, *Medium*, *High*, dan *Critical*. Grafik menunjukkan bahwa model prediksi mampu mereplikasi pola distribusi tingkat keparahan dengan sangat baik. Jumlah kasus pada masing-masing kategori hampir identik antara nilai sebenarnya dan prediksi, yang mengindikasikan bahwa model tidak hanya akurat dalam skala numerik, tetapi juga cukup andal dalam mengelompokkan kerentanan ke dalam kategori tingkat keparahan yang sesuai.

Tabel 4. 6 Evaluasi Metrik keseluruhan model

Severity level	Precision	Recall	F1-Score
CRITICAL	0.7927	0.8103	0.8014
HIGH	0.8234	0.8283	0.8258
MEDIUM	0.8978	0.8879	0.8928
LOW	0.5564	0.5300	0.5429
Accuracy			0.8472
Weighted Avg	0.8474	0.8472	0.8473

Tabel 4.6 menampilkan evaluasi metrik keseluruhan model klasifikasi berdasarkan tingkat keparahan (*Severity level*) pada data pengujian, dengan metrik *precision*, *recall*, dan *F1-score* untuk kategori *CRITICAL*, *HIGH*, *MEDIUM*, dan

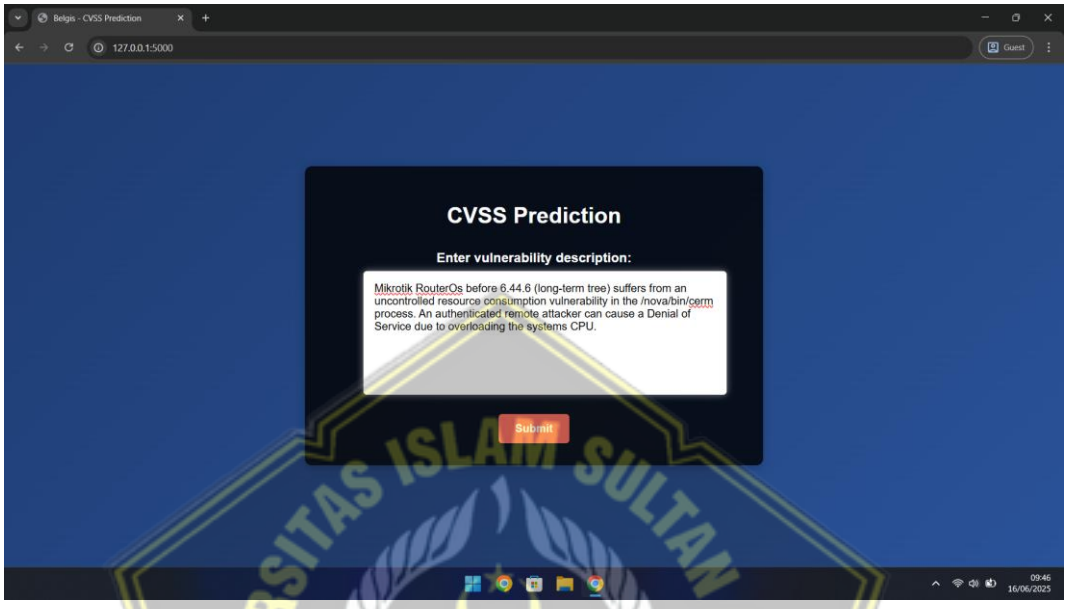
LOW. Model menunjukkan performa terbaik pada kategori *MEDIUM* dengan nilai *F1-score* tertinggi 0.8928, sementara kategori *LOW* memiliki performa paling rendah dengan *F1-score* 0.5429. Rendahnya hasil pada kategori *LOW* ini disebabkan oleh jumlah data yang terbatas sehingga model kesulitan dalam mempelajari pola kelas tersebut secara efektif. Akurasi keseluruhan model mencapai 0.8472, dengan *weighted average* metrik yang seimbang, mengindikasikan model mampu mengklasifikasikan berbagai kelas secara proporsional meskipun performa pada kelas dengan data sedikit menjadi lebih rendah.

4.3 Hasil Perancangan Aplikasi

Gambar 4. 8 Halaman awal aplikasi

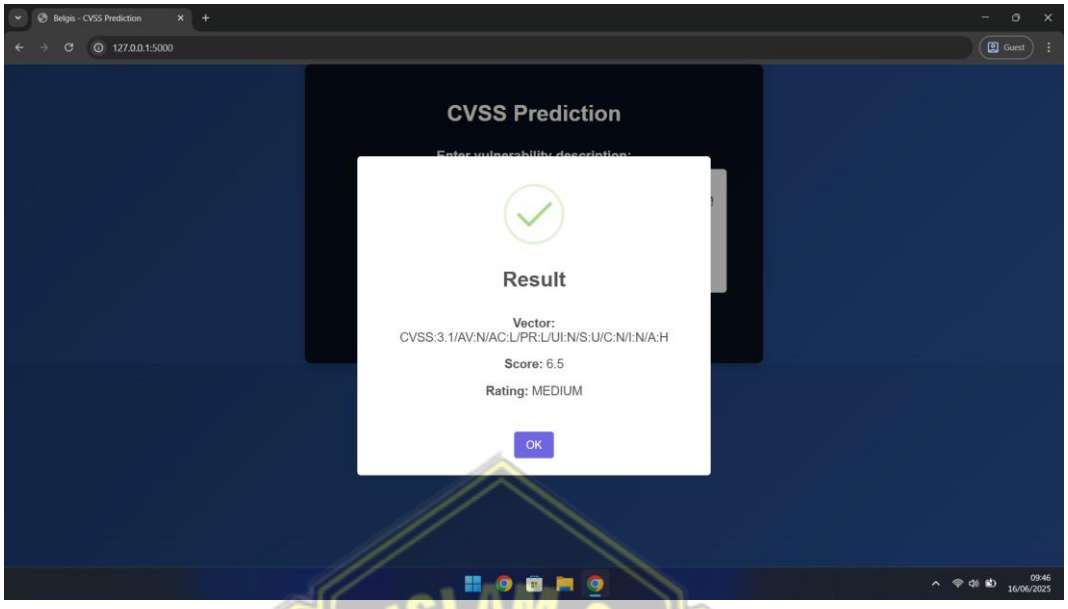
Pada gambar 4.8 merupakan implementasi desain halaman awal klasifikasi CVSS yang dilihat oleh pengguna pertama kali, dengan judul "*CVSS Prediction"* yang ditampilkan di bagian atas dalam sebuah *card* atau kotak container berwarna gelap. Pada bagian bawah judul terdapat teks *"Enter vulnerability description:"* yang berfungsi sebagai label untuk input area. Sebuah *textarea* yang digunakan untuk mengetikkan deskripsi kerentanan yang akan diprediksi. Di bagian bawah form, terdapat tombol **"***Submit*" berwarna merah yang berfungsi untuk mengirimkan input deskripsi tersebut ke backend untuk dilakukan klasifikasi CVSS

dan prediksi skornya. Seluruh komponen ini ditata secara simetris di tengah layar dengan latar belakang biru yang bersih untuk memberikan kesan profesional dan fokus kepada fungsi utama aplikasi.



Gambar 4. 9 Halaman deskripsi kerentanan sudah dimasukkan

Pada gambar 4.9 merupakan implementasi lanjutan dari halaman awal klasifikasi CVSS, di mana pengguna telah mengisi input deskripsi kerentanan ke dalam kolom yang tersedia. Contoh input yang dimasukkan adalah deskripsi kerentanan terhadap MikroTik RouterOS versi tertentu yang mengalami masalah *uncontrolled resource consumption* pada proses tertentu. Deskripsi ini bertujuan untuk diuji oleh sistem prediksi CVSS. Setelah deskripsi dituliskan, pengguna dapat menekan tombol "*Submit*" yang berada di bawah kolom input untuk memproses prediksi skor kerentanan komputer.



Gambar 4. 10 *Pop-up* Hasil Klasifikasi

Pada gambar 4.10 merupakan tampilan hasil dari proses klasifikasi deskripsi CVE menggunakan model prediksi CVSS. Setelah pengguna menekan tombol "*Submit*", sistem menampilkan pop-up berisi ringkasan hasil prediksi. Pada bagian atas terdapat ikon centang hijau sebagai indikator bahwa proses berhasil dilakukan. Kemudian ditampilkan informasi "*Result*" yang mencakup tiga elemen penting:

- Vector: representasi vektor metrik CVSS, yaitu CVSS:3.1/AV:N/AC:L/PR:U/UI:N/S:U/C:N/I:N/A:H,
- Score: nilai CVSS yang dihasilkan yaitu 6.5,
- Rating: tingkat keparahan yang dikategorikan sebagai MEDIUM.

Seluruh informasi ini ditampilkan dalam sebuah kotak dialog berwarna putih dengan tombol OK berwarna ungu di bagian bawah sebagai interaksi untuk menutup dialog. Tampilan ini memberikan hasil yang ringkas, informatif, dan mudah dipahami oleh pengguna akhir.

BAB V
KESIMPULAN DAN SARAN

7.1 Kesimpulan

Berdasarkan hasil penelitian ini, dapat diambil kesimpulan tentang sistem klasifikasi CVSS berbasis deskripsi CVE menunjukkan performa yang tinggi dan konsisten pada data pengujian. Model *Bidirectional Encoder Representations from Transformers* (BERT) yang telah dilatih mampu mengklasifikasikan sebagian besar metrik CVSS dengan akurasi dan balanced accuracy yang sangat baik, yakni berkisar antara 88,60% hingga 98,41%, dengan performa terbaik pada kategori *Attack Complexity* dan performa yg cukup pada *Privileges Required*. Evaluasi regresi pada skor CVSS menghasilkan nilai *Mean Absolute Error* (MAE) sebesar 0.4301, dengan 70,88% prediksi identik terhadap skor sebenarnya dan lebih dari 80% prediksi memiliki selisih kurang dari satu poin. Nilai ini tergolong sangat rendah mengingat MAE maksimum secara teoretis adalah 10.0. Hal ini menunjukkan bahwa model tidak hanya akurat dalam klasifikasi vektor, tetapi juga presisi dalam menghasilkan skor CVSS, sehingga dapat diandalkan untuk mendukung analisis kerentanan komputer secara otomatis.

7.2 Saran

Saran untuk pengembangan sistem prediksi CVSS ini adalah dengan menambahkan tahap praproses berupa pemfilteran deskripsi menggunakan model berbasis *Large Language Model* (LLM), seperti GPT, untuk memastikan bahwa input yang diterima benar-benar relevan dengan konteks keamanan siber. Hal ini penting karena model masih menghasilkan prediksi meskipun diberikan input yang tidak sesuai, seperti frasa umum yang sama sekali tidak memiliki kaitan dengan kerentanan komputer, namun tetap menghasilkan skor CVSS yang acak. Dengan adanya modul pemilah atau validasi awal berbasis LLM, sistem dapat mengenali dan menyaring deskripsi yang ambigu, tidak relevan, atau mengandung noise sebelum diproses lebih lanjut. Pendekatan ini diharapkan dapat meningkatkan akurasi dan keandalan prediksi, sekaligus mencegah penyalahgunaan atau

kesalahan interpretasi hasil oleh pengguna. Selain itu, integrasi LLM juga dapat dikembangkan untuk memberikan umpan balik otomatis terhadap kualitas deskripsi yang dimasukkan oleh pengguna.

DAFTAR PUSTAKA

Beretas, C. (2024). Information Systems Security, Detection and Recovery from Cyber Attacks. *Universal Library of Engineering Technology*, *01*(01), 27–40. https://doi.org/10.70315/uloap.ulete.2024.0101005

Bhargava, P., Drozd, A., & Rogers, A. (2021). *Generalization in NLI: Ways (Not) To Go Beyond Simple Heuristics*. 125–135. https://doi.org/10.18653/v1/2021.insights-1.18

Carvalho, M., Pinho, A. J., & Brás, S. (2025). Resampling approaches to handle class imbalance: a review from a data perspective. *Journal of Big Data*, *12*(1). https://doi.org/10.1186/s40537-025-01119-4

Costa, J. C., Roxo, T., Sequeiros, J. B. F., Proenca, H., & Inacio, P. R. M. (2022). Predicting CVSS Metric via Description Interpretation. *IEEE Access*, *10*, 59125–59134. https://doi.org/10.1109/ACCESS.2022.3179692

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, *1*(Mlm), 4171–4186.

Distillation, P. (2020). *WELL-READ STUDENTS LEARN BETTER : ON THE IM - PORTANCE OF PRE-TRAINING COMPACT MODELS. MLM*, 1–13.

Duraz, R., Espes, D., Francq, J., & Vaton, S. (2024). Intrusion Detection Systems. *Journal of Universal Computer Science*, *2024*(9). https://doi.org/10.3897/jucs.131659ï

Elbaz, C., Rilling, L., & Morin, C. (2020). *Day Vulnerabilities with Automated CVSS Vector Prediction at Disclosure*. https://inria.hal.science/hal-02895913v1

Feng, S. Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., & Hovy, E. (2021). A Survey of Data Augmentation Approaches for NLP. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 968–988. https://doi.org/10.18653/v1/2021.findings-acl.84

FIRST. (2025). *Common Vulnerability Scoring System version 3.1 Specification Document Revision 1*. 1–24. https://www.first.org/cvss/

Galassi, A., Lippi, M., & Torroni, P. (2021). Attention in Natural Language Processing. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(10), 4291–4308. https://doi.org/10.1109/TNNLS.2020.3019893

Hopton, Z. W., Scherrer, Y., & Samardzic, T. (2025). *Functional Lexicon in Subword Tokenization*. *1*, 7839–7853.

Huang, H., Zhu, D., Wu, B., Zeng, Y., Wang, Y., Min, Q., & Zhou, X. (2025). *Over-Tokenized Transformer: Vocabulary is Generally Worth Scaling*. http://arxiv.org/abs/2501.16975

HuggingFace. (2020a). *Transformers Documentation, huggingface.co.* https://huggingface.co/docs/transformers/pad_truncation.%0A

HuggingFace. (2020b). *Transformers Documentation, huggingface.co.*

Joshi, D., Aryan, S., Shreya, D., Om, D., Dipasha, S., & Jagtap, S. (2023). Text Data Augmentation. *Proceedings of the 2023 International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)*, 392–396. https://doi.org/10.1109/ICAICCIT60255.2023.10465716

Li, L. (2024). Tracing the Genealogies of Ideas with Sentence Embeddings. *NLP4DH 2024 - 4th International Conference on Natural Language Processing for Digital Humanities, Proceedings of the Conference*, 9–16.

NVD. (2024). *CVEs and the NVD Process*. National Vulnerability Database. https://nvd.nist.gov/general/cve-process

Ozkan-okay, M., Yilmaz, A. A., Akin, E., Aslan, A., & Aktug, S. S. (2023). A Comprehensive Review of Cyber Security Vulnerabilities ,. *Electronics*, *12*(1333).

Renaldy, A., Fauzi, A., Shabrina, A. N., Novita Ramadhan, H., Novita Ramadhani, M., Aprilia Hikayatuni'mah, P., & Iskandar, O. (2023). Peran Sistem Informasi dan Teknologi Informasi Terhadap Peningkatan Keamanan Informasi Perusahaan. *Jurnal Ilmu Multidisplin*, *2*(1), 15–22. https://doi.org/10.38035/jim.v2i1.212

Shahid, M. R., & Debar, H. (2021). CVSS-BERT: Explainable Natural Language Processing to Determine the Severity of a Computer Security Vulnerability from its Description. *Proceedings - 20th IEEE International Conference on Machine Learning and Applications, ICMLA 2021*, 1600–1607. https://doi.org/10.1109/ICMLA52953.2021.00256

Simonetto, S., & Bosch, P. (2024). *Comprehensive threat analysis and systematic mapping of CVEs to MITRE framework*. https://cwe.mitre.org/

Sönmez, F. Ö. (2021). Classifying Common Vulnerabilities and Exposures Database Using Text Mining and Graph Theoretical Analysis. In *Studies in Computational Intelligence* (Vol. 919, hal. 313–338). Springer Science and Business Media Deutschland GmbH. https://doi.org/10.1007/978-3-030-57024-8_14

Sun, J., Xing, Z., Guo, H., Ye, D., Li, X., Xu, X., & Zhu, L. (2021). *Generating Informative CVE Description From ExploitDB Posts by Extractive Summarization*. http://arxiv.org/abs/2101.01431

Tasmiha Khan, M. G. (2024). *What is Common Vulnerabilities and Exposures (CVE)?* IBM. https://www.ibm.com/id-id/think/topics/cve

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-Decem*(Nips), 5999–6009.

Weinberg, A. I., & Cohen, K. (2024). *Zero Trust Implementation in the Emerging Technologies Era: Survey*. https://doi.org/10.20517/ces.2024.41

Yang, F., & Gu, S. (2021). Industry 4.0, a revolution that requires technology and national strategies. *Complex and Intelligent Systems*, *7*(3), 1311–1325. https://doi.org/10.1007/s40747-020-00267-9

Zhang, Z., Strubell, E., & Hovy, E. (2022). A Survey of Active Learning for Natural Language Processing. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, 6166–6190. https://doi.org/10.18653/v1/2022.emnlp-main.414