# DETEKSI *ZERO-WIDTH CHARACTERS* YANG DISAMARKAN PADA URL *PHISHING* DENGAN MENGGUNAKAN ALGORITMA XGBOOST

**SKRIPSI**

**DISUSUN OLEH**

**AHMAD ASADEL**

**NPM. 2109020157**


**PROGRAM STUDI TEKNOLOGI INFORMASI**

**FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI**

**UNIVERSITAS MUHAMMADIYAH SUMATERA UTARA**

**MEDAN**

**2025**

# DETEKSI *ZERO-WIDTH CHARACTERS* YANG DISAMARKAN PADA URL *PHISHING* DENGAN MENGGUNAKAN ALGORITMA XGBOOST

**SKRIPSI**

**Ditujukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer (S.Kom) dalam Program Studi Teknologi Informasi pada Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Muhammadiyah Sumatra Utara**

**AHMAD ASADEL**
**NPM : 2109020157**

**PROGRAM STUDI TEKNOLOGI INFORMASI**
**FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI**
**UNIVERSITAS MUHAMMADIYAH SUMATERA UTARA**
**MEDAN**
**2025**

## LEMBAR PENGESAHAN

| | | |
|---|---|---|
| Judul Skripsi | : | DETEKSI ZERO-WIDTH CHARACTERS YANG DISAMARKAN PADA URL PHISHING DENGAN MENGGUNAKAN ALGORITMA XGBOOST |
| Nama Mahasiswa | : | Ahmad Asadel |
| NPM | : | 2109020157 |
| Program Studi | : | Teknologi Informasi |

Menyetujui
Komisi Pembimbing

**(Andi Zulherry, S.Kom., M.Kom)**
**NIDN. 0129117901**

| **Ketua Program Studi** | **Dekan** |
|---|---|
| **(Fatma Sari Hutagalung, S.Kom., M.Kom.)** | **(Dr. Al-Khowarizmi, S.Kom., M.Kom.)** |
| **NIDN. 0117019301** | **NIDN. 0127099201** |

# PERNYATAAN ORISINALITAS

**DETEKSI *ZERO-WIDTH CHARACTERS* YANG DISAMARKAN PADA URL *PHISHING* DENGAN MENGGUNAKAN ALGORITMA XGBOOST**

**SKRIPSI**

Saya menyatakan bahwa karya tulis ini adalah hasil karya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing disebutkan sumbernya.

Medan, 10 September 2025

Yang membuat pernyataan


Ahmad Asadel

NPM. 2109020157

# PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademika Universitas Muhammadiyah Sumatera Utara, saya bertanda tangan dibawah ini:

| | |
|---|---|
| Nama | : Ahmad Asadel |
| NPM | : 219020157 |
| Program Studi | : Teknologi Informasi |
| Karya Ilmiah | : Skripsi |

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Muhammadiyah Sumatera Utara Hak Bedas Royalti Non-Eksekutif *(Non-Exclusive Royalty free Right)* atas penelitian skripsi saya yang berjudul:

**DETEKSI *ZERO-WIDTH CHARACTERS* YANG DISAMARKAN PADA URL *PHISHING* DENGAN MENGGUNAKAN ALGORITMA XGBOOST**

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non-Eksekutif ini, Universitas Muhammadiyah Sumatera Utara berhak menyimpan, mengalih media, memformat, mengelola dalam bentuk database, merawat dan mempublikasikan Skripsi saya ini tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis dan sebagai pemegang dan atau sebagai pemilik hak cipta.

Demikian pernyataan ini dibuat dengan sebenarnya.

Medan, 10 September 2025

Yang membuat pernyataan

Ahmad Asadel

NPM. 2109020157

# RIWAYAT HIDUP

DATA PRIBADI

| | |
|---|---|
| Nama Lengkap | : Ahmad Asadel |
| Tempat dan Tanggal Lahir | : Pekanbaru, 22 November 2002 |
| Alamat Rumah | : Jalan Parit Indah, Perumahan Permata Ratu (Blok R No 1) Tengkerang Labuai, Kecamatan Bukit Raya, Kota Pekanbaru, Riau |
| Telepon/Faks/HP | : 085163146431 |
| E-mail | : ahmadasadel04@gmail.com |
| Instansi Tempat Kerja | : - |
| Alamat Kantor | : - |

DATA PENDIDIKAN

| | | |
|---|---|---|
| SD | : MUHAMMADIYAH 3 UNGGULAN | TAMAT: 2014 |
| SMP | : MTsN ANDALAN | TAMAT: 2017 |
| SMA | : SMA IT IMAM SYAFI'I 2 | TAMAT: 2020 |

# KATA PENGANTAR

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

Puji syukur penulis ucapkan kepada Allah SWT atas rahmat-Nya Penulis dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Komputer pada Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Muhammadiyah Sumatera Utara.

Penulis menyadari bahwa tanpa bantuan, bimbingan, dan dukungan dari berbagai pihak sejak masa perkuliahan hingga masa penyusunan skripsi, sulit untuk menyelesaikan skripsi ini. Oleh karena itu dengan segala kerendahan hati, Penulis ingin menyampaikan terima kasih dan rasa hormat kepada:

1. Bapak Prof. Dr. Agussani, M.AP., selaku Rektor Universitas Muhammadiyah Sumatera Utara (UMSU)
2. Bapak Dr. Al-Khowarizmi, S.Kom., M.Kom., selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi (FIKTI) UMSU.
3. Bapak Halim Maulana, S.T., M.Kom., selaku Wakil Dekan I Fakultas Ilmu Komputer dan Teknologi Informasi.
4. Bapak Dr. Lutfi Basit, S.Sos., M.I.Kom., selaku Wakil Dekan III Fakultas Ilmu Komputer dan Teknologi Informasi.
5. Ibu Fatma Sari Hutagalung, S.Kom., M.Kom., selaku Ketua Program Studi Teknologi Informasi yang telah memberikan bimbingan dan bantuan selama Penulis menempuh studi serta yang selalu mengingatkan penulis untuk segera menyelesaikan skripsi ini.

6. Bapak Mhd. Basri, S.Si., M.Kom., selaku Sekretaris Program Studi Teknologi Informasi.
7. Bapak Andi Zulherry, S.Kom., M.Kom., selaku Dosen Pembimbing yang telah banyak membantu penulis baik dukungan, saran, dan bimbingan dalam penyusunan penelitian ini.
8. Bapak Yasirwan, S.E., S.H., M.H., dan Ibu Aminah Hidayani, S.E., selaku Kedua Orang Tua penulis yang selalu mendoakan dan memberi dukungan baik mental maupun finansial selama merantau di kota medan.
9. Puan Azzahra, selaku adik penulis yang senantiasa memberikan dukungan dan semangat dalam penyelesaian skripsi ini.
10. Ismi Qontas Lubis dan Daris Fauzan Abila, rekan seperjuangan yang telah menjadi partner terbaik dalam proses pengerjaan skripsi ini. Diskusi, dukungan, dan semangat dari kalian sangat membantu penulis hingga sampai di tahap ini.
11. Seluruh teman dan rekan seperjuangan dari grup Mulitmedia, D1-Esport, anak-anak DV, dan ‘Kursi Duduk’. Kalian adalah alasan mengapa penulis masih bisa tertawa di tengah stres, dan pengingat untuk tidak pernah menyerah. Dukungan dan masukan dari kalian semua sangat berarti hingga penulis bisa sampai di titik ini.
12. Diri sendiri, yang sudah berusaha keras dan berjuang sejauh ini. Untuk semua energi yang terkuras, untuk kesabaran yang tak terbatas dalam menghadapi diri yang penuh drama, dan untuk kekuatan yang muncul di saat-saat paling genting. Perjuangan melawan rasa malas ini akhirnya terbayar lunas.

Penulis menyadari sepenuhnya bahwa skripsi ini masih jauh dari kata sempurna dan memiliki banyak keterbatasan. Meskipun demikian, harapan terbesar penulis adalah semoga karya sederhana ini tetap dapat memberikan sumbangsih yang bermanfaat, baik bagi para pembaca, akademisi, maupun sebagai referensi awal bagi penelitian selanjutnya di bidang yang sama.

Medan, 4 Agustus 2025

Penulis

Ahmad Asadel

**DETEKSI *ZERO-WIDTH CHARACTERS* YANG DISAMARKAN PADA URL *PHISHING* DENGAN MENGGUNAKAN ALGORTMA XGBOOST**

# ABSTRAK

Serangan *phishing* merupakan salah satu ancaman siber yang paling umum dan merusak, dengan teknik yang terus berevolusi menjadi semakin canggih dan sulit dideteksi. Salah satu metode penyamaran terbaru yang menjadi perhatian adalah penggunaan *Zero-Width Characters* (ZWC)—karakter Unicode tak kasat mata yang disisipkan ke dalam URL untuk mengelabui sistem deteksi tradisional dan persepsi visual manusia. Penelitian ini bertujuan untuk mengembangkan dan mengevaluasi model machine learning yang efektif dan andal untuk mendeteksi URL *phishing* yang telah disamarkan menggunakan ZWC. Algoritma eXtreme Gradient Boosting (XGBoost) dipilih karena kemampuannya yang terbukti unggul dalam menangani data yang kompleks dan kemampuannya untuk mengoptimalkan performa.Penelitian ini menggunakan *dataset* publik dari Kaggle yang terdiri dari 11.430 sampel URL, yang kemudian dimodifikasi melalui proses rekayasa fitur. Secara spesifik, 50% dari URL *phishing* disisipi salah satu dari lima jenis ZWC (ZWSP, ZWNJ, ZWJ, RLM, LRM), dan sebuah fitur biner khusus diciptakan untuk menandai keberadaan karakter-karakter tersebut. Pada pelatihan awal, model menunjukkan adanya overfitting ringan. Oleh karena itu, dilakukan proses *hyperparameter* tuning dengan mengatur parameter max_depth dan min_child_weight untuk menciptakan model yang lebih robust. Model final dievaluasi menggunakan 20% data uji dan menunjukkan kinerja yang sangat tinggi, dengan pencapaian akurasi 97.24%, Presisi 97.03%, Recall 97.37%, dan skor AUC 0.9972. Nilai recall yang tinggi sangat krusial, membuktikan kemampuan model yang andal dalam meminimalkan risiko lolosnya ancaman berbahaya. Penelitian ini berhasil membuktikan bahwa pendekatan XGBoost dengan rekayasa fitur yang ditargetkan mampu menjadi solusi yang efektif untuk melawan serangan *phishing* canggih.

**Kata Kunci:** *Phishing, Zero-Width Characters, XGBoost, Deteksi URL, Keamanan Siber, Machine Learning.*

**DETECTION OF DISGUISED ZERO-WIDTH CHARACTERS IN PHISHING URLS USING XGBOOST ALGORITHM**

# ABSTRACT

Phishing attacks represent one of the most common and damaging cyber threats, with techniques continuously evolving to become more sophisticated and harder to detect. One of the latest evasion methods of concern is the use of Zero-Width Characters (ZWC)—invisible Unicode Characters inserted into URLs to deceive traditional detection systems and human visual perception. This research aims to develop and evaluate an effective and reliable machine learning model to detect phishing URLs that have been obfuscated using ZWC. The eXtreme Gradient Boosting (XGBoost) algorithm was chosen for its proven superiority in handling complex data and its performance optimization capabilities. This study utilized a public dataset from Kaggle consisting of 11,430 URL samples, which was then modified through a feature engineering process. Specifically, 50% of the phishing URLs were injected with one of five types of ZWC (ZWSP, ZWNJ, ZWJ, RLM, LRM), and a dedicated binary feature was created to flag the presence of these Characters. Initial training revealed signs of minor overfitting. Consequently, a hyperparameter tuning process was conducted by adjusting the max_depth and min_child_weight parameters to create a more robust model. The final model was evaluated on 20% of the test data and demonstrated exceptionally high performance, achieving an Accuracy of 97.24%, Precision of 97.03%, Recall of 97.37%, and an AUC score of 0.9972. The high Recall value is particularly crucial, proving the model's reliability in minimizing the risk of missed threats. This research successfully proves that an XGBoost-based approach with targeted feature engineering can be an effective solution against advanced phishing attacks.

**Keywords:** Phishing, Zero-Width Characters, XGBoost, URL Detection, Cybersecurity, Machine Learning.

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# BAB I.
# PENDAHULUAN

## 1.1. Latar Belakang

Internet kini menjadi bagian yang tidak terpisahkan dari kehidupan sehari-hari. Internet telah merasuk ke berbagai aspek kehidupan kita, mulai dari komunikasi, bisnis, hingga pendidikan (Wijoyo et al., 2023). Kemajuan teknologi ini membuat informasi menjadi lebih cepat dan mudah diakses. Selain itu, perkembangan ini secara signifikan meningkatkan efisiensi dan keterhubungan di seluruh dunia. Namun, di sisi lain, hal ini juga menimbulkan ancaman baru yaitu meningkatnya risiko terjadinya kejahatan siber, salah satunya adalah *phishing*.

Serangan *phishing* merupakan salah satu bentuk serangan siber yang paling merusak dan umum terjadi (Wijoyo et al., 2023). Berdasarkan laporan terbaru dari APWG (Anti-*Phishing* Working Group) tahun 2024, serangan *phishing* mengalami peningkatan signifikan setiap tahunnya dengan kerugian finansial yang terus meningkat sejak Juni 2023, jumlah serangan *phishing* dilaporkan berada di kisaran 290.000 hingga 370.000 serangan tiap bulannya. Mengutip laporan dari Intersile Consulting, tercatat adanya peningkatan tahunan sejumlah 50.000 serangan, sehingga totalnya hampir mencapai 1,9 juta serangan di periode Mei 2023 hingga April 2024. Serangan *phishing* biasanya dilakukan melalui manipulasi psikologis (social engineering) dengan tujuan mencuri data sensitif seperti password, nomor kartu kredit, atau informasi pribadi pengguna.

Saat ini, metode *phishing* yang digunakan penyerang juga semakin beragam dan canggih. Salah satu teknik baru yang mulai muncul dan menjadi perhatian komunitas keamanan adalah penggunaan *Zero-Width Characters*. ZWC merupakan

karakter Unicode yang tidak tampak secara visual atau tidak memakan ruang saat ditampilkan di layar. (Pajola & Conti, 2021). Jenis-jenis karakter ini mencakup *Zero-Width Space* (U+200B), *Zero-Width Non-Joiner* (U+200C), *Zero-Width Joiner* (U+200D), *Left-To-Right Mark* (U+200E), dan *Right-To-Left Mark* (U+200F).

Teknik *phishing* ini diterapkan dengan memanfaatkan Unicode dikenal sebagai *Zero-Width Attack on Security & Privacy* (Z-WASP) (Kaavija, 2025). Cara kerja teknik ini ZWC disisipkan ke dalam kalimat yang terlihat normal bagi manusia tetapi akan mempengaruhi cara mesin menganalisis dan mengindeks teks. Karakter-karakter ini digunakan untuk menghindari deteksi oleh sistem yang memeriksa teks, seperti dalam konteks serangan terhadap sistem deteksi ujaran kebencian atau aplikasi terjemahan (Pajola & Conti, 2021).Sebagai contoh, penyerang dapat menampilkan tautan yang terlihat seperti www.umsumedan.com. Namun, sebenarnya pelaku telah menyisipkan ZWC di antara beberapa huruf, misalnya antara huruf d dan a sehingga membentuk www.umsumedan.com (dengan simbol U+200B—Zero-Width Space). Namun, secara teknis browser akan membaca URL yang sudah dimodifikasi tersebut, lalu mengarahkan pengguna ke situs palsu.

Selain mengecoh pengguna secara visual, karakter ini juga mampu mengelabui sistem keamanan otomatis yang berbasis teks atau URL karena sulit dideteksi oleh pemeriksaan biasa yang berbasis pencocokan teks sederhana. Sebagai contoh nyata, serangan *phishing* menggunakan *Zero-Width* pernah berhasil mengelabui sistem keamanan email Microsoft Office 365 pada tahun 2019 (microsoft.com, 2021).

Permasalahan yang muncul adalah bahwa sistem keamanan yang umum digunakan saat ini, seperti antivirus berbasis *signature* maupun sistem filter email berbasis aturan, masih kesulitan mendeteksi serangan *phishing* jenis ini. Oleh karena itu, dibutuhkan pendekatan baru yang lebih cerdas dan adaptif untuk mendeteksi serangan *phishing* berbasis *Zero-Width Characters* secara efektif.

Algoritma *Machine Learning* (ML) telah terbukti berhasil dalam mendeteksi berbagai serangan siber. Metode ini memungkinkan sistem untuk mengidentifikasi pola dalam data dan membuat prediksi atau klasifikasi tanpa harus diprogram untuk setiap scenario. *Extreme Gradient Boosting* (XGBoost) adalah salah satu algoritma pembelajaran mesin yang paling baik untuk tugas klasifikasi. XGBoost sangat dikenal karena kemampuan untuk mengelola dataset yang sangat besar dan kompleks.

Dalam konteks deteksi *phishing*, XGBoost bekerja dengan membangun serangkaian pohon keputusan secara berurutan. Setiap pohon yang baru ditambahkan akan berfokus untuk memperbaiki kesalahan prediksi yang dilakukan oleh pohon-pohon sebelumnya. Ini berarti, jika pohon pertama salah mengklasifikasikan suatu URL sebagai *non-phishing* padahal mengandung ZWC, pohon berikutnya akan belajar dari kesalahan tersebut dan mencoba mengklasifikasikannya dengan benar. Proses iteratif ini memungkinkan XGBoost untuk secara bertahap meningkatkan akurasi keseluruhan model dalam mengidentifikasi pola-pola halus dan kompleks dalam URL, termasuk keberadaan *Zero-Width Characters* yang disamarkan.

Algoritma ini efektif dalam klasifikasi dan regresi, serta dapat mengoptimalkan deteksi *phishing* yang menggunakan ZWC (Erdiyanto, 2023).

XGBoost telah berhasil diterapkan dalam berbagai deteksi anomali, deteksi intrusi, maupun klasifikasi email *phishing* dengan tingkat akurasi yang tinggi terhadap perubahan pola data.

Selain itu, fitur-fitur teknis dan struktural dari URL akan dianalisis dan diubah menjadi format numerik yang dapat diproses oleh model. Selanjutnya, fitur-fitur teknis dan struktural dari URL akan dianalisis dan digunakan sebagai dasar untuk proses klasifikasi. Untuk mengukur performa model, akan diterapkan evaluasi menggunakan *Confusion Matrix* dan metrik-metrik klasifikasi lainnya, meliputi akurasi, presisi, *recall*, dan *F1-score*. Pengujian ini bertujuan untuk memvalidasi bahwa hasil riset ini akurat dan dapat diterapkan pada situasi sebenarnya.

Tujuan dari penelitian ini adalah untuk membuat model yang efektif untuk mendeteksi serangan *phishing* dengan menggunakan algoritma XGBoost, yang terkenal karena kemampuannya dalam menangani data besar dan kompleks. Algoritma ini dipilih karena kemampuannya yang luar biasa untuk mengidentifikasi pola yang tersembunyi dalam data, yang sangat penting untuk mendeteksi *Zero-Width Characters* yang disamarkan ke dalam URL *Phishing*.

Selain itu, penelitian ini akan mengevaluasi kinerja model XGBoost dalam mendeteksi URL *phishing* dengan menggunakan berbagai metrik evaluasi, seperti akurasi, presisi, recall, dan skor F1. Hasil evaluasi ini akan menunjukkan seberapa efektif model dalam situasi dunia nyata dan menemukan area mana yang membutuhkan perbaikan tambahan. Berdasarkan hasil evaluasi, penelitian ini akan memberikan rekomendasi praktis untuk pengembangan sistem deteksi *phishing* yang lebih baik dan efisien.

## 1.2. Rumusan Masalah

Dengan mempertimbangkan masalah-masalah di atas, penelitian ini akan berkonsentrasi pada beberapa masalah utama berikut:

1. Bagaimana penerapan algoritma XGBoost dalam membangun model yang mampu mengidentifikasi URL *phishing* yang secara khusus disamarkan menggunakan *Zero-Width Characters.*
2. Penelitian ini menganalisis fitur-fitur dari URL seperti panjang URL, panjang hostname, jumlah titik, tanda hubung, serta elemen-elemen lain yang relevan, untuk membedakan antara URL *phishing* dan *non-phishing* dalam *dataset* yang mengandung karakter *Zero-Width.*
3. Evaluasi efektivitas model XGBoost dalam mendeteksi URL *phishing* yang disamarkan dengan *Zero-Width Characters* berdasarkan *dataset* yang digunakan, untuk mengukur tingkat akurasi, presisi, recall, dan F1-score model dalam mengidentifikasi *phishing*.

## 1.3. Batasan Masalah

Penelitian ini dilakukan dengan batasan yang ditetapkan secara hati-hati untuk menjaga penelitian tetap fokus dan terarah. Batasan-batasan ini juga digunakan untuk mengelola ruang lingkup penelitian secara efektif dan menghasilkan hasil yang sesuai dengan tujuan penelitian. Batasan yang terkait dengan subjek penelitian ini adalah sebagai berikut:

a) *Dataset* yang digunakan merupakan *dataset* yang didapatkan dari website https://www.kaggle.com/*dataset*s/shashwatwork/web-page-*phishing*-detection-*dataset*/data yang kemudian dimodifikasi sehingga URL dengan status *phishing* memiliki *Zero-Width Characters.*

b) Penelitian ini membatasi analisis hanya pada karakter *Zero-Width Space* (ZWSP), *Zero-Width Non-Joiner* (ZWNJ), *dan Zero-Width Joiner* (ZWJ), *Right-to-Left Mark* (RLM)*, Left-to-Right Mark* (LRM)*,* yang disisipkan dalam URL *phishing*.

c) Penelitian hanya menggunakan algoritma XGBoost untuk klasifikasi, tanpa melakukan perbandingan dengan algoritma *machine learning* lainnya.

d) Evaluasi model hanya menggunakan *Confusion Matrix*, akurasi, presisi, recall, dan F1-score sebagai parameter penilaian.

e) Penelitian ini hanya membahas deteksi *phishing* berbasis URL, yang disamarkan dengan *Zero-Width Characters*.

## 1.4. Tujuan Penelitian

Berdasarkan rumusan masalah sebelumnya, tujuan penelitian ini adalah:

a) Melakukan evaluasi efektivitas model XGBoost untuk menemukan URL *phishing.* pengukuran efektivitas tersebut didasarkan pada dataset yang digunakan dan dianalisis melalui metrik akurasi, presisi, *recall*, serta *F1-score*.

b) Tujuan dari penelitian ini adalah untuk mengembangkan model deteksi yang efisien yang dapat mendeteksi URL phishing yang disamarkan menggunakan karakter Zero-Width menggunakan algoritma XGBoost.

## 1.5. Manfaat Penelitian

Manfaat yang diharapkan dari dilakukannya penelitian ini yaitu :

a) Menjadi referensi bagi penelitian di bidang keamanan siber dan *machine learning*, khususnya terkait implementasi XGBoost dalam deteksi *phishing* berbasis URL.
b) Memberikan kontribusi bagi sistem keamanan siber dalam mendeteksi *phishing* berbasis *Zero-Width Characters,* yang sulit dideteksi oleh sistem keamanan konvensional.
c) Meningkatkan kesadaran masyarakat tentang ancaman *phishing* yang lebih canggih, serta pentingnya memahami keamanan URL sebelum mengaksesnya.
d) Memberikan wawasan mengenai bagaimana karakter *Zero-Width* dapat digunakan sebagai teknik penyamaran dalam serangan *phishing*, serta cara mengidentifikasinya.

# BAB II.
# LANDASAN TEORI

## 2.1. Pengertian *Phishing*

*Phishing* adalah jenis penipuan di dunia maya di mana pelaku menggunakan berbagai teknik seperti ancaman atau jebakan untuk mengelabui orang yang mereka tuju(Febrika Ardy et al., 2024). Tujuan utamanya adalah agar korban secara tidak sadar mengungkapkan kredensial atau informasi pribadi kepada pelaku.

Taktik ini berdampak signifikan baik terhadap individu maupun organisasi, dapat menyebabkan kerugian data penting dan keuangan. Sejak pertama kali diidentifikasi pada akhir tahun 1990-an, teknik *phishing* terus berkembang. Mulai dari email yang menyamar sebagai komunikasi dari lembaga keuangan untuk menipu korban, kini meluas ke teknik lebih sofistikasi seperti *Spear Phishing* dan *Whaling* (BSSN, 2024). *Spear Phishing* adalah serangan yang sangat personalisasi, menargetkan individu atau entitas dengan informasi yang sangat spesifik untuk meningkatkan kepercayaan korban, sementara *Whaling* berfokus pada target tingkat tinggi seperti eksekutif perusahaan (Febrika Ardy et al., 2024).

Kemajuan terbaru dalam teknik *phishing* mencakup penggunaan karakter *Zero-Width Characters*, yang memanipulasi cara URL ditampilkan, membuatnya sulit untuk dideteksi oleh korban tetapi mudah diolah oleh mesin (microsoft.com, 2021). Penambahan karakter ini dalam URL adalah contoh dari bagaimana teknik manipulasi dalam *phishing* terus berkembang, memerlukan pendekatan yang lebih dinamis dan teknologi deteksi yang lebih canggih untuk melawan serangan *phishing* yang semakin kompleks.

## 2.2. Karakter Zero-Width dalam URL

*Zero-Width Characters* (ZWC) merupakan elemen Unicode yang tidak terlihat secara visual namun memiliki keberadaan dalam teks, sehingga dapat memengaruhi tampilan maupun pengolahan. Berikut ini adalah beberapa jenis *Zero-Width Characters* yang sering dimanfaatkan dalam serangan *phishing* beserta cara penyisipannya ke dalam URL:

a) Zero Width Space (ZWSP) *Characters* (U+200B):

ZWSP digunakan untuk mengindikasikan peluang jeda kata atau baris tanpa menghabiskan ruang apa pun secara visual. Biasanya digunakan dalam bahasa seperti Thailand, Myanmar, Khmer, dan Jepang, di mana tidak ada spasi yang terlihat di antara kata-kata tetapi diperlukan pemisah kata atau pemisah baris.

b) Zero Width Joiner (ZWJ) *Characters* (U+200D):

ZWJ digunakan untuk meminta rendering yang terhubung dari karakter yang berdekatan, seperti membentuk ligatur atau bergabung secara kursif. Ini digunakan untuk mengontrol koneksi antara karakter yang dapat membentuk ligatur atau koneksi kursif. Jika dua karakter dapat membentuk ligatur tetapi tidak secara default, ZWJ meminta agar ligatur digunakan.

c) Zero Width Non-Joiner (ZWNJ) *Characters* (U+200C):

Berbeda dengan ZWJ, ZWNJ justru mencegah penggabungan karakter. ZWNJ digunakan untuk mencegah pembentukan ligatur atau koneksi kursif antara karakter yang berdekatan. Ini mematahkan ligatur dan koneksi kursif, memastikan karakter dirender tanpa menghubungkan. Dalam serangan *phishing*, ZWNJ digunakan untuk memisahkan elemen-

elemen dalam URL yang biasanya dianggap bagian dari domain sah, sehingga sistem keamanan yang kurang canggih kesulitan mendeteksi adanya keanehan.

d) Left-To-Right Mark (LTR Mark/LRM) (U+200E):

Karakter ini digunakan untuk memaksa teks agar ditulis dengan arah kiri-ke-kanan, terutama dalam konteks teks yang mengandung campuran dari skrip dengan arah penulisan yang berbeda (seperti teks Arab yang kanan-ke-kiri dan teks Latin yang kiri-ke-kanan)

e) Right-To-Left Mark (RTL Mark/RLM) (U+200F):

Berkebalikan dengan LRM, RLM digunakan untuk memaksa teks yang mengikuti karakter ini untuk ditulis dalam urutan kanan-ke-kiri. Ini sangat berguna dalam teks yang mencampurkan bahasa yang ditulis kanan-ke-kiri (seperti bahasa Arab atau Ibrani) dengan bahasa yang ditulis kiri-ke-kanan.

Mekanisme penyisipan karakter *Zero-Width Characters* dalam URL memanfaatkan celah dalam sistem deteksi tradisional yang sering kali hanya mengidentifikasi kesalahan visual, tanpa menganalisis komponen tidak terlihat dalam kode. Hal ini menciptakan tantangan signifikan dalam deteksi otomatis, karena sistem keamanan harus mampu menguraikan dan mengenali struktur internal URL di luar sekadar representasi visualnya (Pajola & Conti, 2021).

## 2.3. Machine learning

*Machine learning* atau Pembelajaran Mesin merupakan teknik pendekatan dari Artificial Intelligent (AI) yang digunakan untuk menirukan hingga menggantikan peran manusia dalam melakukan aktivitas hingga memecahkan

masalah (Suhendra & Cs, 2021). *Machine learning* menggunakan algoritme dan model statistik untuk menganalisis dan menarik kesimpulan dari data besar, dan secara progresif memperbaiki kinerjanya berdasarkan pengalaman yang diperoleh melalui data yang diproses.

*Machine learning* dapat dibagi menjadi empat kategori utama berdasarkan bagaimana algoritmenya belajar dari data: *Supervised learning*, *Unsupervised learning*, *Semi-supervised learning*, dan *Reinforcement learning*. *Supervised learning* melibatkan pelatihan model dengan data berlabel, di mana model belajar memprediksi output dari input yang diberikan. *Unsupervised learning* adalah algoritma yang tidak membutuhkan data berlabel. Algoritma ini digunakan untuk mendeteksi pola dan pemodelan deskriptif tanpa menggunakan kategori atau output berlabel. Kategori dan output berlabel adalah dasar algoritma untuk mencari model yang tepat. Metode ini digunakan untuk *clustering*, yang merupakan teknik dalam *Unsupervised learning* untuk mengelompokkan data ke dalam kelompok berdasarkan kesamaan atau kedekatannya, serta untuk *Association rule*, yang digunakan untuk menemukan hubungan atau pola yang sering terjadi antar item dalam *dataset*, seperti dalam analisis transaksi (Suhendra & Cs, 2021).

*Semi-supervised learning* adalah algoritma yang tidak dapat dikategorikan ke dalam *Supervised* atau *Unsupervised learning*. Itu cocok untuk data yang sangat besar yang dibagi menjadi dua bagian, masing-masing diberi label dan tidak diberi label (Suhendra & Cs, 2021). Terakhir, *Reinforcement learning* melibatkan agen yang belajar melalui interaksi dengan lingkungan, menerima umpan balik berupa *reward* atau *punishment* untuk meningkatkan strateginya, seperti yang digunakan dalam permainan dan robotika (Murphy, 2024).

Dalam penerapannya, *machine learning* melibatkan pelatihan algoritma atau serangkaian proses statistik untuk mengidentifikasi pola dan fitur dalam Kumpulan data yang besar. Tujuannya adalah untuk membuat keputusan atau prediksi berdasarkan informasi yang ditemukan. Semakin baik kinerja algoritmanya, semakin tinggi pula akurasi keputusan dan prediksi sistem (Yudhana et al., 2022).

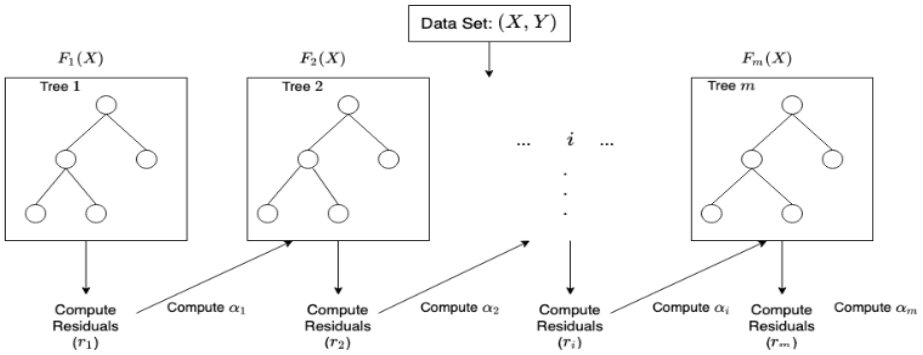
### 2.3.1. XGBoost

XGBoost (eXtreme Gradient Boosting) adalah salah satu algoritma pengajaran mesin yang paling banyak digunakan untuk menyelesaikan masalah regresi dan klasifikasi. Algoritma ini menjadi andalan karena kemampuannya yang unggul dalam memproses data berskala besar dengan tingkat kompleksitas yang tinggi (Mim Hanifah Permana, 2024). Alasan utama pemilihan algoritma XGBoost pada riset ini didasarkan pada efisiensinya yang tinggi. XGBoost menawarkan optimisasi pada kecepatan pemrosesan dan penggunaan memori melalui pemanfaatan cache prosesor yang lebih baik serta dukungan terhadap komputasi paralel dan multicore. Kapabilitas tersebut menjadikan sistem mampu beroperasi dengan waktu eksekusi yang lebih singkat daripada algoritma konvensional

XGBoost mengoptimalkan model prediktif melalui ensemble dari pohon keputusan yang saling memperbaiki kesalahan satu sama lain. Setiap pohon baru yang ditambahkan berfokus pada kesalahan yang ditinggalkan oleh pohon sebelumnya, secara bertahap meningkatkan akurasi keseluruhan model (Ryan Afrizal et al., 2018).

Algoritma XGBoost menggabungkan berbagai model pembelajaran mesin untuk meningkatkan klasifikasi yang masih lemah, seperti metode Gradient Boosting lainnya. Tujuannya adalah untuk meningkatkan kinerja melalui pelatihan

bertahap. Agar kinerja algoritma dengan metode kelompok ini ditingkatkan, hasil klasifikasi dari model pembelajaran mesin sebelumnya, yang dikenal sebagai residuals atau error, digunakan untuk evaluasi. Algoritma XGBoost dirancang untuk meningkatkan waktu eksekusi, terutama saat memproses data yang besar (I. Karo Karo, 2022).




**Gambar 2.1 Cara kerja peningkatan pohon gradien (XGBoost).**

Kemudian XGBoost secara bertahap memperbaiki kesalahan prediksi sebelumnya dengan menambahkan pohon-pohon bar uke dalam modelnya, di mana setiap pohon berusaha memperbaiki kesalahan dari pohon sebelumnya. Ini memungkinkan model untuk mempelajari pola tersembunyi dan kompleks dalam data dengan efektif.

Selain itu, dengan menggunakan fungsi tujuan yang teratur, XGBoost dapat mengoptimalkan modelnya untuk mencapai keseimbangan yang tepat antara presisi dan kompleksitas, yang membantu mencegah model dari overfitting pada data pelatihan dengan adanya regularisasi bawaan (L1 dan L2). Adapun parameter tersebut meliputi *max_depth* untuk mengontrol kedalaman pohon, dan *min_child_weight* yaitu jumlah minimum bobot instans yang diperlukan pada anak (Geeksforgeeks.org, 2024). Fokus utama dalam XGBoost adalah pada penurunan nilai fungsi objektif teratur, yang didefinisikan sebagai berikut:

$$Ob_j(\theta) = \sum_{i=1}^{n} L\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{k=1}^{k} \Omega(f_k) \text{.....(2.1)}$$

Keterangan :

$\Theta$ : Parameter model

$n$ : Jumlah sampel dalam *dataset*

$L\left(y_i, \hat{y}_i^{(t)}\right)$ : Fungsi kerugian yang mengukur beda antara label aktual $y_i$ dan prediksi $\hat{y}_i^{(t)}$ pada iterasi $t$

$\Omega(f_k)$ : Fungsi regularisasi pohon $f_k$, yang bertujuan untuk mengontrol kompleksitas model

$t$ : Jumlah pohon dalam model

## 2.4. ROC-AUC

Area Di Bawah Kurva (AUC) adalah ukuran luas di bawah kurva ROC (*Characteristics of Receiver Operating*), yang digunakan untuk mengevaluasi kinerja model klasifikasi. Kinerja model yang lebih baik dalam membedakan kelas positif dan negatif ditunjukkan oleh nilai AUC yang lebih tinggi dalam rentang 0,5–1,0. Lima kategori interpretasi nilai AUC adalah sebagai berikut:

a) 0,5–0,6 (akurasi rendah/salah)

b) 0,6–0,7 (akurasi lemah)

c) 0,7–0,8 (akurasi sedang)

d) 0,8–0,9 (akurasi tinggi)

e) 0,9–1,0 (akurasi sangat tinggi).

ROC-AUC bukanlah satu rumus tunggal seperti Akurasi, melainkan sebuah konsep evaluasi yang terdiri dari dua bagian:

### 2.4.1. Kurva ROC (Reciever Operating Characteristic)

Kurva ROC adalah sebuah grafik yang memvisualisasikan seberapa baik sebuah model bisa membedakan antara kelas positif dan kelas negatif. Pada kasus ini kelas positif itu adalah *Phishing* dan kelas positif adalah *Legitimate.* Kurva ini dibuat dengan memplot dua nilai:

a) True Positive Rate (TPR) di sumbu Y. Dari semua *phishing* yang sebenarnya, berapa persen yang berhasil dideteksi oleh model. Rumusnya:

$$TPR = \frac{TP}{TP+FN} \quad \text{.....(2.2)}$$

b) False Positive Rate (FPR) di sumbu X. Dari semua URL aman yang sebenarnya, berapa persen yang salahh ditandai sebagai *phishing*. Rumusnya:

$$FPR = \frac{FP}{FP+TN} \quad \text{.....(2.3)}$$

### 2.4.2. AUC (Area Under the Curve)

AUC adalah nilai skor yang mengukur luas area di bawah kurva ROC. Nilainya berkisar dari 0.0 hingga 1.0. 0.5 berarti model tidak lebih baik dari tebakan acak. Sedangkan 1.0 berarti model adalah pemisah yang sederhana. AUC sangat berguna untuk mengevaluasi model pada *dataset* yang tidak seimbang dan memberikan gambaran yang lebih komprehensif tentang kemampuan diskriminatif model.

## 2.5. Evaluasi Model

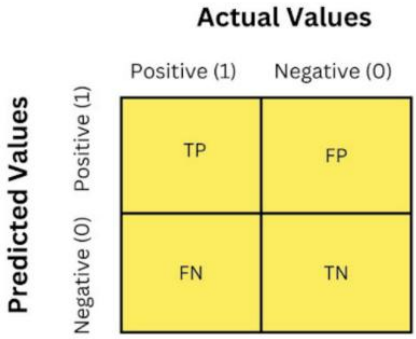
Salah satu langkah penting dalam penelitian yang menggunakan *machine learning* adalah evaluasi model. Ini dilakukan dengan tujuan untuk mengetahui seberapa baik model klasifikasi dapat mengenali dan mengklasifikasikan data dengan benar. Evaluasi dilakukan dengan menggunakan beberapa metrik penting

yang biasa digunakan dalam penelitian klasifikasi, yaitu *Confusion Matrix*, *accuracy, precision, recall,* dan F1-Score (Fatunnisa & Marcos, 2024).

### 2.5.1. Confusion Matrix

Matriks konfusi adalah metode untuk mengevaluasi kinerja metode klasifikasi. Metode ini menampilkan tabel yang berisi ringkasan hasil prediksi, yang memisahkan prediksi yang tepat dan keliru dari model klasifikasi pada persoalan klasifikasi biner (Yulianti et al., 2022). Karena metrik ini menunjukkan kesalahan yang dibuat oleh model dan jenis kesalahan tersebut, metrik ini sangat berguna untuk mengevaluasi kinerja model pembelajaran mesin. Matriks tersebut terdiri dari empat komponen:

a) True Positives (TP): Kasus di mana model menghasilkan hasil kelas positif.
b) True Negatives (TN): Kasus di mana model menghasilkan hasil kelas negatif.
c) False Positives (FP): Kasus di mana model salah menganggap kelas negatif sebagai kelas positif.
d) False Negatives (FN): Kasus di mana model salah menganggap kelas positif sebagai kelas negatif.




**Gambar 2.2 Ilustrasi *Confusion Matrix***

Sebelum nilai TF-IDF dihitung, URL terlebih dahulu akan dipecah menjadi token-token yang lebih kecil berdasarkan tanda baca seperti titik, garis miring, atau simbol lainnya. Contohnya, URL 'https://contoh-domain.com/login' akan dipecah menjadi token seperti 'https', 'contoh', 'domain', 'com', dan 'login'. Token-token inilah yang kemudian dihitung bobot TF-IDF-nya.

### 2.5.2. Accuracy, Precision, Recall, dan F1-Score

a) Akurasi : Metrik ini mengukur proporsi semua prediksi yang benar (baik positif maupun negatif) terhadap semua kasus dalam *dataset*. Rumusnya adalah:

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{.....(2.4)}$$

b) Presisi : Metrik ini mengukur proporsi prediksi positif yang benar-benar positif. Rumusnya adalah:

$$Presisi = \frac{TP}{TP+FP} \quad \text{.....(2.5)}$$

c) Recall (Sensitivitas) : Metrik ini mengukur kemampuan model untuk menemukan semua kasus positif yang relevan dalam *dataset*. Rumusnya adalah:

$$Recall = \frac{TP}{TP+FN} \quad \text{.....(2.6)}$$

d) F1-Score : Nilai F1-Score membantu mengukur keseimbangan antara presisi dan recall dalam satu angka. Rumus ini adalah:

$$F1 - Score = 2 \times \frac{Presisi \times Recall}{Presisi+Recall} \quad \text{.....(2.7)}$$

## 2.6. Python

Python, bahasa pemrograman tingkat tinggi yang diciptakan oleh Guido Van Rossum dan dirilis pada tahun 1991, dimana pada zaman sekarang bahasa ini sangat popular (Riziq Sirfatullah Alfarizi et al., 2023). Python dikenal sebagai bahasa pemrograman yang mudah dipelajari oleh pemula sekaligus cukup kuat untuk keperluan industri skala besar. Python cukup popular digunakan untuk berbagai kalangan mulai dari pemula hingga industri besar karena sintaksnya yang sederhana, dapat digunakan di berbagai platform, memiliki banyak library dan framework seperti NumPy, *Pandas*, *Scikit-learn*, *Matplotlib*, TensorFlow, dan XGBoost, serta memiliki komunitas yang luas (Maesaroh et al., 2024).

Python memungkinkan peneliti untuk memproses data URL dengan lebih mudah menggunakan *library Pandas*, melakukan ekstraksi fitur teks menggunakan teknik TF-IDF melalui *Scikit-learn*, serta menerapkan algoritma klasifikasi menggunakan *library* khusus *machine learning* seperti XGBoost. Selain itu, Python juga dapat digunakan untuk mengevaluasi model yang ada dengan *Confusion Matrix* dan metrik seperti akurasi, presisi, recall, serta F1-score dengan memanfaatkan library yang ada.

## 2.7. Penelitian Terkait

Beberapa penelitian sebelumnya telah memanfaatkan algoritma *machine learning* seperti XGBoost untuk mendeteksi *phishing* dengan berbagai pendekatan. Penelitian oleh Muhammad Ryan Afrizal dkk. (2024) menyoroti keunggulan XGBoost yang dipadukan dengan *Random Search Hyperparameter* Tuning. Hasil penelitian mereka menunjukkan bahwa XGBoost setelah tuning *hyperparameter*

dapat mencapai akurasi hingga 97,69%, yang lebih tinggi dibandingkan tanpa tuning sebesar 95,34%.

Penelitian lain oleh Nishant Nityanand Naik (2021) menegaskan potensi XGBoost dalam mendeteksi *phishing*. Dalam eksperimennya yang membandingkan beberapa algoritma, XGBoost berhasil mencapai tingkat akurasi sebesar 85,8%, menunjukkan performa yang lebih stabil dibanding algoritma pohon keputusan lainnya seperti Decision Tree yang mencapai 85,94% dan Random Forest dengan akurasi 85,9%.

Kemudian penelitian oleh Farashazillah Yahya dkk. (2021) membandingkan tiga algoritma yaitu Decision Tree, Random Forest, dan KNN dalam mendeteksi website *phishing*. Hasil penelitian tersebut menunjukkan bahwa algoritma KNN memiliki tingkat akurasi tertinggi yaitu sebesar 97,6%, sementara Random Forest mencapai 94,44%, dan Decision Tree sebesar 91,16%. Walaupun penelitian tersebut belum secara langsung menguji XGBoost, tetapi hasil tersebut menegaskan bahwa algoritma berbasis pohon memiliki performa klasifikasi *phishing* yang sangat baik, yang mendukung penggunaan XGBoost dalam penelitian ini.

Selain itu, penelitian Mim Hanifah Permana (2024) menunjukkan pengembangan model peramalan penjualan dengan XGBoost. Hasil penelitian ini membuktikan bahwa XGBoost efektif dalam analisis prediksi kompleks dan mampu memberikan hasil peramalan akurat dengan *RMSE* rendah (61,89) dan *MAPE* (10,2%). Hal ini mendukung penggunaan algoritma XGBoost pada penelitian ini yang berfokus pada deteksi URL *phishing* dengan *Zero-Width Characters* yang memerlukan kemampuan analisis fitur kompleks.

Penelitian Nurkumala Lubis (2024), membandingkan performa algoritma Support Vector Machine (SVM) dan Random Forest dalam klasifikasi *phishing* melalui email. Penelitian ini menemukan bahwa SVM memiliki akurasi lebih tinggi (97,27%) dibanding Random Forest (96,51%). Meskipun penelitiannya tidak langsung menggunakan XGBoost, hasilnya menegaskan bahwa algoritma berbasis pohon seperti Random Forest (dan secara tidak langsung juga XGBoost yang merupakan varian pohon keputusan yang lebih baik) sangat efektif dalam klasifikasi *phishing*.

**Tabel 2.1 Penelitian Terkait**

| NO | Penulis | Tahun | Judul | Metode | Hasil |
|---|---|---|---|---|---|
| 1 | Muhammad Ryan Afrizal dkk. | 2024 | XGBoost dengan *Random Search Hyperparameter* Tuning Untuk Klasifikasi Situs *Phishing* | XGBoost, *Random Search* | Akurasi mencapai 97,69% dengan tuning *hyperparameter*, sebelumnya 95,34% |
| 2 | Nishant Nityanand Naik | 2021 | Modelling Enhanced *Phishing* Detection using XGBoost | XGBoost, Decision Tree, Random Forest | Akurasi XGBoost 85,8%, lebih baik disbanding Decision Tree 85,94%, dan Random Forest 85,9% |
| 3 | Farashazillah Yahya dkk. | 2021 | Detection of *Phishing* Websites Using *Machine learning* Approaches | Decision Tree, KNN, Random Forest | KNN memiliki akurasi terbaik (97,6%), diikuti Random Forest (94,44%) dan Decision Tree 91,16%. |
| 4 | Mim Hanifah Permana | 2024 | Pengembangan Model Peramalan Penjualan dengan XGBoost | XGBoost | Menunjukkan XGBoost efektif dalam analisis prediksi kompleks dan mampu memberikan hasil peramalan akurat dengan RMSE rendah |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | (61,89) dan MAPE (10,2%) |
| 5 | Nurkumala Lubis | 2024 | Analisis Perbandingan SVM dan Random Forest untuk Klasifikasi Email *Phishing* | SVM, Random Forest | SVM lebih baik dengan akurasi 97,27%, Random Forest 96,51% |

# BAB III.
# METODOLOGI PENELITIAN

## 3.1. Instrumen Penelitian

Hasil penelitian menunjukkan penggunaan berbagai spesifikasi atau instrumen untuk mendukung penelitian, termasuk perangkat keras, perangkat lunak, dan sistem operasi yang mendukung proses pengembangan aplikasi. Berikut adalah ringkasan spesifikasi yang digunakan dalam penelitian ini:

**Tabel 3.1 Instrumen Penelitian**

| Instrumen | Spesifikasi |
|---|---|
| Processor | 12$^{th}$ Gen Intel® Core(TM) I7-12700H (20 CPUs), ~2.3GHz |
| Memory | 24 GB |
| Solid State Drive | 1 TB |
| Series | Lenovo Legion 5I |
| Python | 3.11.9 |

Untuk mendukung proses implementasi dan eksperimen, penelitian ini memanfaatkan beberapa pustaka (*library*) utama dari bahasa pemrograman Python yang berjalan di lingkungan Google Colaboratory. *Pandas* digunakan sebagai fondasi untuk seluruh operasi data, mulai dari memuat hingga memanipulasi *dataset*. Algoritma klasifikasi inti dibangun menggunakan pustaka XGBoost.

Seluruh proses evaluasi, termasuk pembagian data dan perhitungan metrik kinerja, difasilitasi oleh pustaka *Scikit-learn*. Untuk visualisasi data seperti *Confusion Matrix* dan Kurva ROC, digunakan kombinasi pustaka *Matplotlib* dan

*Seaborn*. *Matplotlib* digunakan sebagai dasar untuk membuat grafik seperti diagram batang dan Kurva ROC, dan *Seaborn* digunakan di atas *Matplotlib* untuk menghasilkan visualisasi yang lebih informatif dan estetis, khususnya dalam pembuatan *heatmap* untuk *Confusion Matrix*.

Terakhir, untuk membangun antarmuka pengguna (UI) sebagai bukti konsep, digunakan pustaka *Gradio* yang dibantu oleh tldextract untuk memvalidasi input URL.

### 3.2. Jenis dan Pendekatan Penelitian

Penelitian ini menggunakan pendekatan kuantitatif karena jenis penelitian ini berfokus pada pengembangan dan evaluasi model (model *development* and *evaluation*), yang memiliki elemen eksperimental. Pendekatan kuantitatif dipilih karena penelitian ini akan menguji hipotesis tentang efektivitas algoritma XGBoost melalui pengumpulan dan analisis data numerik.

Secara spesifik, penelitian ini mengembangkan sebuah model klasifikasi machine learning, melatihnya menggunakan *dataset* URL *phishing* yang telah dimodifikasi dengan menyisipkan *Zero-Width Characters*, dan secara sistematis menguji kinerjanya menggunakan data pengujian terpisah.

Proses pengolahan data meliputi rekayasa fitur untuk mendeteksi *Zero-Width Characters* dan memastikan semua fitur dalam format numerik yang siap digunakan oleh model XGBoost, pembagian *dataset*, pelatihan model, dan evaluasi menggunakan metrik kuantitatif seperti *Confusion Matrix*, akurasi, presisi, recall, dan F1-score merupakan ciri khas dari pendekatan kuantitatif dalam pengembangan dan validasi model prediktif di bidang ilmu komputer. Elemen eksperimental terlihat dari adanya manipulasi data (penyisipan karakter *Zero-Width*) dan

pengukuran hasil (kinerja deteksi model) untuk menilai efektivitas metode yang diusulkan.

### 3.3. Prosedur Pengumpulan Data

Untuk mendukung penelitian ini, diperlukan data yang akan digunakan untuk membangun model deteksi URL *phishing* dengan algoritma XGBoost. *Dataset* yang digunakan berasal dari sumber publik di https://www.kaggle.com/s/shashwatwork/web-page-*phishing*-detection-*dataset*/data, yang memuat data URL dengan label *phishing* dan *non-phishing*. Kemudian *dataset* tersebut dimodifikasi dengan cara menyisipkan beberapa jenis karakter *Zero-Width* seperti *Zero-Width* Space (ZWSP), *Zero-Width* Non-Joiner (ZWNJ), dan *Zero-Width* Joiner (ZWJ) di antara huruf-huruf pada URL yang mengarah ke situs *phishing*.

Tujuan dari modifikasi ini adalah untuk meniru cara serangan *phishing* berbasis *Zero-Width* dilakukan, sehingga URL yang tampaknya sah di mata pengguna, namun ketika diproses oleh sistem, URL tersebut mengarah ke situs berbahaya. Agar dapat digunakan oleh algoritma *machine learning* XGBoost untuk dapat belajar dan memahami pola yang ada dari data tersebut sesuai dengan arah penelitian. Berikut Langkah-langkah modifikasinya:

a) Identifikasi URL dengan status *Phishing*.
b) Menyisipkan *Zero-Width Characters* (ZWSP, ZWNJ, ZWJ, LRM, RLM).
c) Distribusikan *Zero-Width Characters* secara acak dan merata.
d) Penambahan Kolom URL yang dimodifikasi (kolom modified_url) agar dapat membandingkan URL asli dengna versi yang telah dimodifikasi.

*Dataset* modifikasi ini berisikan 11.430 entri, yang terdiri dari 5.715 URL legitimate dan 5.715 URL phishing. Dari total URL *phishing*, sebanyak 2.857 di

antaranya telah dimodifikasi dengan menyisipkan karakter Zero-Width secara acak. Adapun distribusi jumlah untuk setiap jenis karakter Zero-Width yang disisipkan pada *dataset* ini adalah sebagai berikut:

a) *Zero-Width Space (ZWSP): 568 karakter.*

b) *Zero-Width Non-Joiner* (ZWNJ): 575 karakter.

c) *Zero-Width Joiner* (ZWJ): 600 karakter.

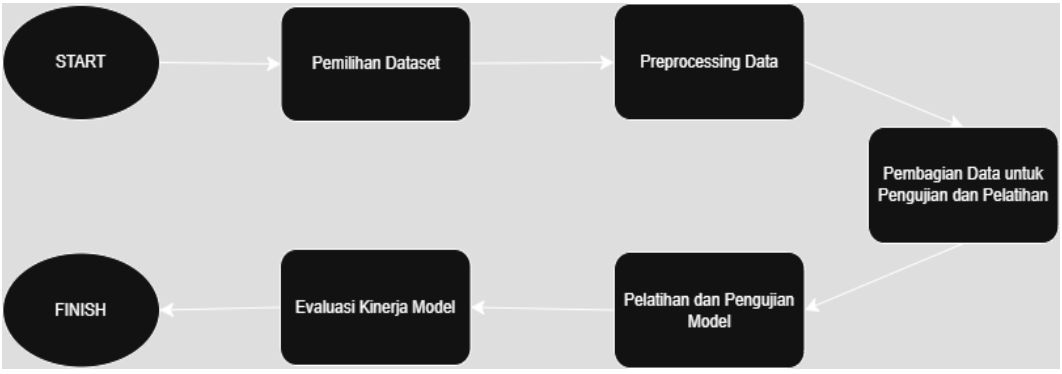
d) *Left-to-Right Mark* (LRM): 544 karakter.

e) *Right-to-Left Mark* (RLM): 570 karakter.

Dengan memanfaatkan *dataset* ini, diharapkan dapat dikembangkan model deteksi *phishing* yang lebih akurat dan efektif, khususnya dalam mengidentifikasi URL *phishing* yang telah disamarkan menggunakan karakter *Zero-Width*.

### 3.4. Alur Penelitian (Workflow)

Penelitian ini bertujuan untuk mengembangkan model deteksi *phishing* menggunakan algoritma XGBoost, yang dirancang untuk mendeteksi URL *phishing* yang telah disamarkan dengan karakter *Zero-Width*. Dalam proses penelitian ini, diperlukan tahapan-tahapan sehingga model yang dibangun




**Gambar 3.1 Alur proses penelitian**

merupakan model yang prediktif dan dapat melakukan proses prediksi dengan akurat. Alur penelitian dapat dilihat pada gambar 3.1.

Gambar 3.1 menunjukkan alur proses yang dilakukan dalam membangun model klasifikasi URL pishing. Untuk penjelasan tahap alur penelitian diberikan dibawah ini.

a) Penelitian *Dataset*

Penelitian dimulai dengan pemilihan *dataset* yang sebelumnya didapatkan dari Kaggle.com yang terdiri dari URL yang dilabeli sebagai *phishing* dan *legitimate*. *Dataset* ini kemudian dimodifikasi dengan menambahkan karakter *Zero-Width* pada URL *phishing*.

b) Preprocessing Data

Data yang ada akan dibersihkan dan diproses terlebih dahulu dari entri yang tidak relevan atau rusak, dan memastikan bahwa setiap URL sudah memiliki label yang jelas. Kemudian pada tahap ini, URL *phishing* yang ada akan dimodifikasi dengan *Zero-WidthCharacters*, seperti *Zero-Width Space* (ZWSP), *Zero-Width Non-Joiner* (ZWNJ), *Zero-Width Joiner* (ZWJ), *Left-to-Right Mark* (LRM), *dan Right-to-Left Mark* (RLM). Selain itu, fitur terkait URL, seperti panjang URL, jumlah karakter khusus, dan jumlah karakter *Zero-Width*, akan dihitung.

Kemudian, fitur ‘mengandung_zwc’ yang baru dibuat akan digabungkan dengan 88 fitur numerik yang sudah ada dari *dataset*. Vektor fitur gabungan inilah yang akan digunakan oleh model untuk mempelajari pola, karena algoritma machine learning membutuhkan data dalam format

numerik untuk melakukan perhitungan. Pembagian Data untuk Pengujian dan Pelatihan.

*Dataset* dibagi menjadi dua bagian: satu untuk pelatihan model (80%) dan satu lagi untuk pengujian model dalam mengklasifikasikan URL *phishing* dan *legitimate* (20%). Data pengujian digunakan untuk mengevaluasi kinerja model dalam melakukan prediksi yang akurat, sementara data pelatihan digunakan untuk melatih model agar dapat mengidentifikasi pola dalam data.

c) Pelatihan dan Pengujian Model

Pada tahap ini algoritma XGBoost digunakan untuk melatih model deteksi *phishing*. Model dilatih menggunakan kombinasi 89 fitur numerik (88 fitur asli dan 1 fitur ZWC) yang telah disiapkan sebelumnya. Fitur-fitur inilah yang akan diproses oleh XGBoost untuk mempelajari perbedaan antara URL *phishing* dan *legitimate*.

d) Evaluasi Kinerja Model

Hasil dari proses pengujian tersebut dapat dilakukan analisis dengan menggunakan *Confusion Matrix* sehingga dapat diambil kesimpulan performa dari model dengan algoritma XGBoost yang dibangun sehingga memiliki performa paling optimal.

### 3.5. Jadwal Penelitian

Dalam penelitian yang dilakukan, agar penelitian dapat lebih terstruktur dan optimal, maka dirancang jadwal penelitian yang dilakukan. Untuk jadwal penelitian yang dilakukan pada penelitian ini diberikan pada tabel 3.2.

**Tabel 3.2 Jadwal Penelitian**

| No | Nama Kegiatan | Bulan | | | | | |
|---|---|---|---|---|---|---|---|
| | | 2 | 3 | 4 | 5 | 6 | 7 |
| | Pemilihan *dataset* yang akan digunakan | ■ | | | | | |
| | Pemilihan metode penelitian yang akan diterapkan untuk mengolah *dataset* | ■ | | | | | |
| | Penentuan proses pengolahan data yang akan dilakukan | ■ | | | | | |
| | Pemasangan tools yang akan digunakan dalam pembangunan model pada penelitian | | ■ | ■ | | | |
| | Merancang skema penelitian | | ■ | ■ | | | |
| | Penulisan proposal penelitian | | ■ | ■ | ■ | | |
| | Pelatihan dan pengujian Model | | | | | ■ | |
| | Penulisan bab 4 dan 5 | | | | | ■ | |
| | Evaluasi Model dan Visualisasi data | | | | | ■ | |
| | Penulisan laporan dan finalisasi laporan akhir | | | | | | ■ |

# BAB IV.
# HASIL DAN PEMBAHASAN

## 4.1. Pengumpulan dan Persiapan Data

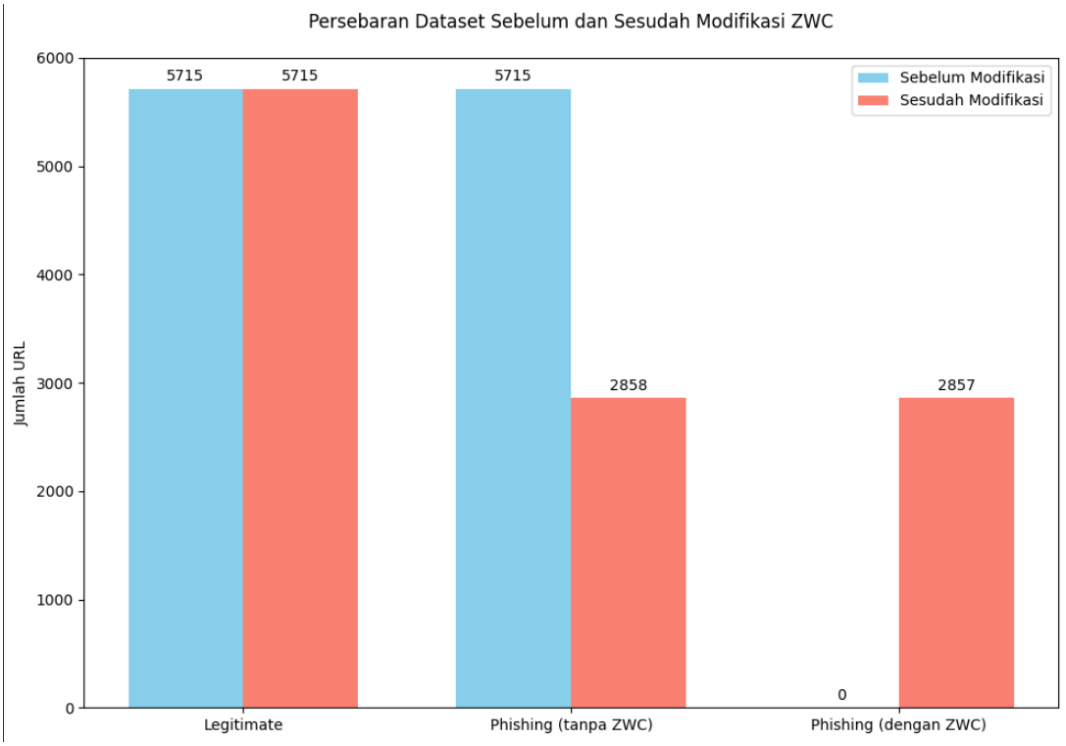
Penelitian ini menggunakan *dataset* publik berjudul "Web Page *Phishing* Detection" yang diperoleh dari platform Kaggle. *Dataset* mentah ini terdiri dari 11.430 sampel URL dengan 88 fitur yang telah diekstraksi. Setiap sampel diklasifikasikan ke dalam dua kategori, yaitu *legitimate* untuk URL yang sah dan *phishing* untuk URL berbahaya.

Selanjutnya, dilakukan proses rekayasa fitur dengan memodifikasi *dataset* asli untuk membangun kemampuan deteksi terhadap *Zero-Width Characters* (ZWC). Langkah pertama adalah menambahkan satu kolom fitur baru bernama "mengandung_zwc", yang pada awalnya diisi dengan nilai 0 untuk semua sampel. Kemudian, sebanyak 50% dari total URL yang berstatus *phishing* dipilih secara acak untuk disisipi salah satu dari lima jenis karakter ZWC yang telah ditetapkan dalam batasan masalah, yaitu *Zero-Width Space* (U+200B), *Zero-Width Non-Joiner* (U+200C), *Zero-Width Joiner* (U+200D), *Right-to-Left Mark* (U+200F), dan *Left-to-Right Mark* (U+200E). Untuk setiap URL yang telah dimodifikasi, nilai pada kolom mengandung_zwc diubah menjadi 1.

Untuk memastikan bahwa proses modifikasi hanya diterapkan pada kelas yang tepat, dilakukan verifikasi data seperti yang ditunjukkan pada Tabel 4.1. Berdasarkan tabel tersebut, dapat dilihat bahwa nilai mengandung_zwc = 1 hanya terdapat pada URL dengan status *phishing* dan berjumlah nol pada URL *legitimate*, sehingga validitas proses rekayasa fitur dapat teruji.

**Tabel 4.1 Verifikasi Hasil Modifikasi Data**

| Keberadaan Zero-*Width* Character | *Legitimate* | *Phishing* |
|---|---|---|
| 0 | 5715 | 2858 |
| 1 | - | 2857 |

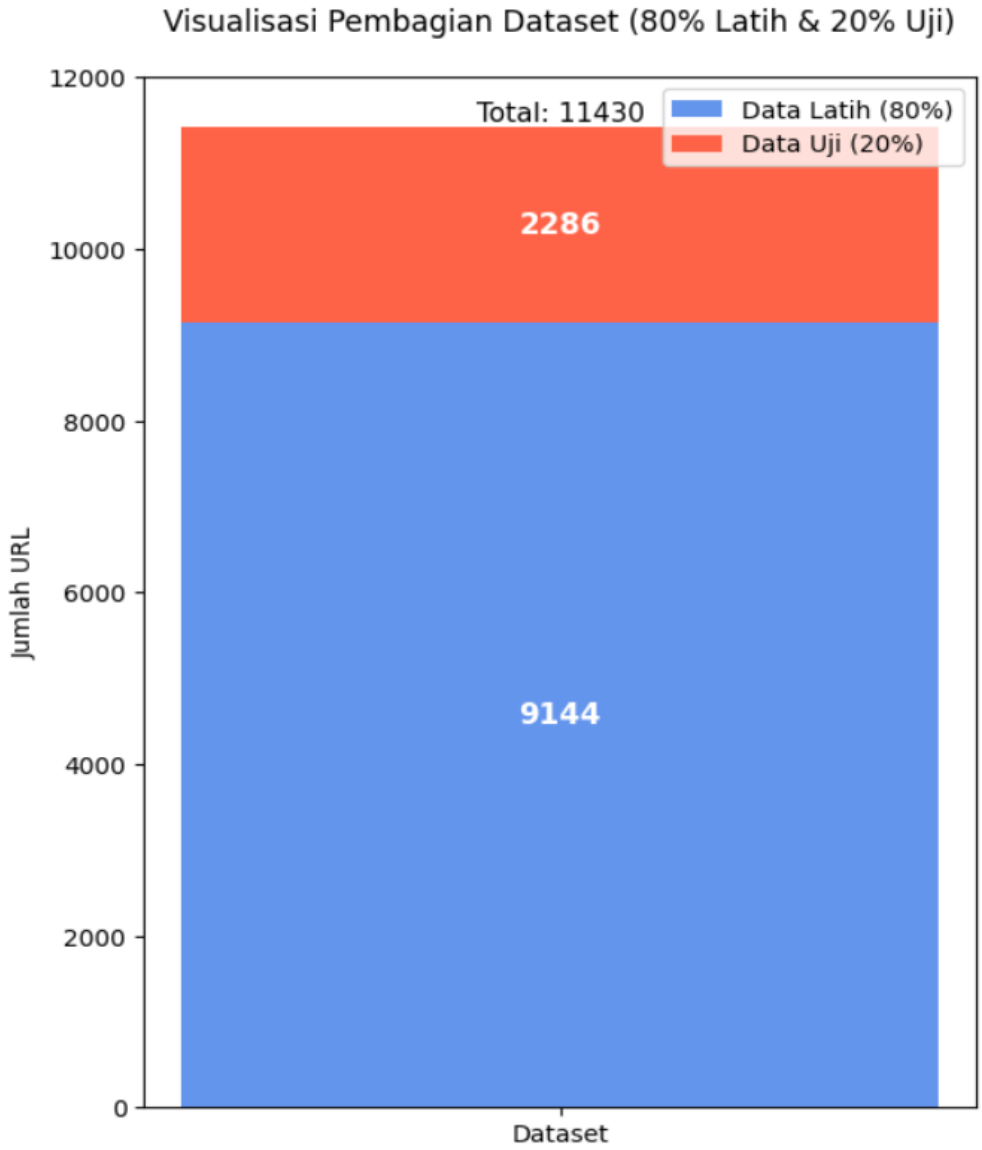



**Gambar 4.1 Persebaran Dataset Sebelum dan Sesudah Modifikasi ZWC**

Langkah terakhir dalam tahap persiapan adalah membagi *dataset* final yang telah dimodifikasi menjadi dua bagian utama, 80% tersebut digunakan sebagai data latihan (training data) dan 20% digunakan sebagai data uji (testing data).

Gambar 4.2 di bawah ini mengilustrasikan pembagian tersebut secara visual, di mana dari total 11.430 URL, sebanyak 9.144 URL dialokasikan sebagai data latih dan sisa 2.286 URL digunakan sebagai data uji. Proses pembagian ini menggunakan metode stratified split untuk memastikan bahwa proporsi antara kelas *legitimate*

dan *phishing* tetap seimbang pada kedua set data tersebut, sehingga dapat mencegah bias selama proses pelatihan dan pengujian model.



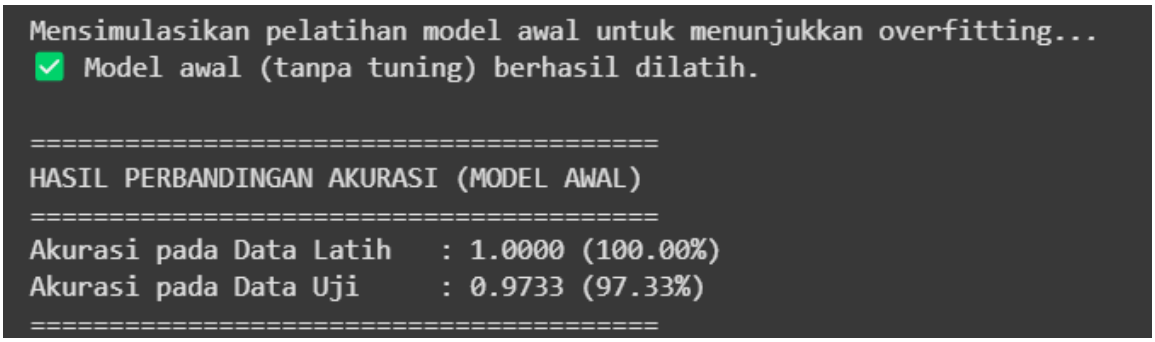

**Gambar 4.2 Visualisasi Pembagian Dataset**

## 4.2. Pelatihan dan Tuning Model

Setelah data disiapkan, langkah selanjutnya adalah melatih model klasifikasi menggunakan algoritma XGBoost. Proses ini bertujuan untuk membangun model yang mampu mengenali pola antara fitur-fitur URL dengan statusnya, apakah *legiimate* atau *phishing*.

Saat pelatihan tahap pertama dengan parameter bawaan, model menunjukkan gejala *overfitting* ringan. Fenomena *overfitting* ini muncul saat model menunjukkan kinerja yang sangat tinggi pada data latih, namun kinerjanya menurun ketika diuji dengan data baru. Indikasi ini terlihat jelas dari hasil akurasi: model berhasil mencapai 100% pada data latih, tetapi hanya memperoleh 98.03% pada data uji. Perbedaan atau "celah" ini menandakan bahwa model cenderung "menghafal" data latih daripada menggeneralisasi polanya.

Untuk mengatasi masalah tersebut, dilakukan proses *hyperparameter tuning* dengan tujuan mengurangi kompleksitas model. Dua parameter utama diatur ulang: max_depth dibatasi menjadi 4 dan min_child_weight diatur menjadi 3. Dua parameter utama diatur ulang untuk membatasi kemampuan model dalam 'menghafal' data.

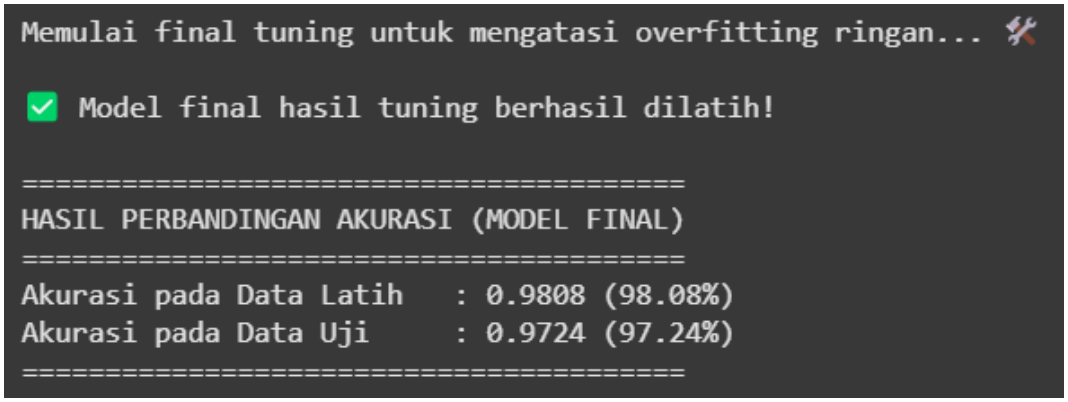



**Gambar 4.3 Bukti Pelatihan Awal Menghasilkan Overfitting**

Setelah proses *tuning*, model dilatih ulang dan dievaluasi kembali. Hasilnya, seperti yang dirangkum pada Gambar 4.4, menunjukkan perbaikan yang signifikan. "Celah" antara akurasi latih dan uji berhasil ditekan hingga kurang dari 1%, menandakan bahwa model final tidak lagi *overfitting* dan memiliki kemampuan generalisasi yang lebih baik. Model hasil *tuning* inilah yang kemudian digunakan sebagai model final dalam penelitian ini.

a) Max_depth : Parameter ini mengatur batas seberapa detail atau rumit aturan yang bisa dipelajari oleh model. Dengan membatasinya menjadi 4, model dipaksa untuk fokus pada pola-pola yang paling umum dan signifikan saja, serta dihindarkan dari membuat aturan yang terlalu spesifik dan hanya berlaku untuk beberapa sampel di data latih.

b) Min_child_weight : Parameter ini menentukan jumlah minimum 'bukti' atau sampel yang diperlukan sebelum model diizinkan membuat cabang aturan baru. Dengan mengaturnya menjadi 3, model dilarang mengambil kesimpulan dari pola yang hanya didukung oleh satu atau dua sampel saja. Ini memaksa model untuk hanya belajar dari pola yang konsisten dan signifikan, sehingga mengabaikan 'noise' atau kebetulan yang ada di data latih.




**Gambar 4.4 Hasil Tuning Untuk Mengatasi Overfitting**

## 4.3. Hasil dan Evaluasi Model

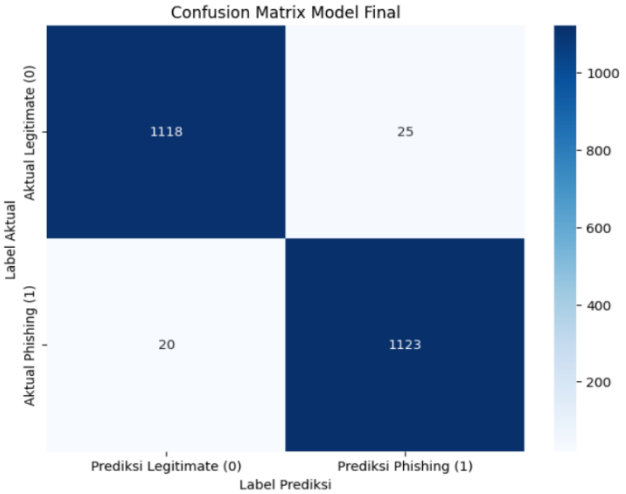
Setelah melalui tahap pelatihan dan tuning, model diuji menggunakan 20% data uji yang belum pernah diproses sebelumnya. Tahap evaluasi ini bertujuan untuk mengukur kinerja model secara objektif dalam mengklasifikasikan URL.

Pengukuran performa dilakukan berdasarkan empat metrik utama, yaitu Accuracy, Precision, Recall, dan F1-Score.

Hasil evaluasi kinerja model pada data uji dirangkum secara lengkap pada Tabel 4.2. Model berhasil mencapai akurasi sebesar 97.24%, yang menandakan bahwa model mampu mengklasifikasikan mayoritas URL dengan benar. Selain itu, nilai *Precision* sebesar 97.03% menunjukkan rendahnya tingkat kesalahan dalam memprediksi URL yang aman sebagai *phishing* (*false positive*). Yang tidak kalah penting, nilai *Recall* yang tinggi sebesar 97.37% membuktikan kemampuan model yang sangat baik dalam menemukan sebagian besar URL *phishing* yang sebenarnya, sehingga meminimalkan risiko ancaman yang terlewatkan (*false negative*).

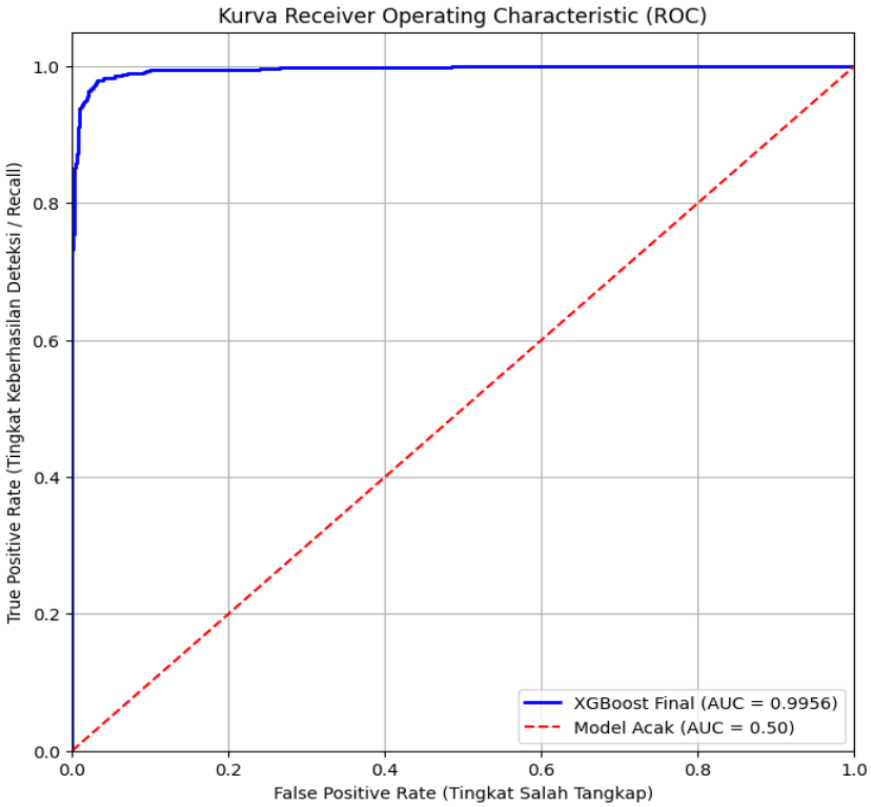
**Tabel 4.2 Hasil Metrik Evaluasi Model**

| Metrik | Nilai |
|---|---|
| Accuracy | 97.24% |
| Precision | 97.03% |
| Recall | 97.37% |
| F1-Score | 97.17% |




**Gambar 4.5 Visualisasi Confusion Matrix**

Selanjutnya, kurva *Receiver Operating Characteristic* (ROC) dimanfaatkan untuk mengevaluasi kapabilitas model dalam memisahkan kelas *legitimate* dan *phishing* pada berbagai tingkat probabilitas. Nilai *Area Under the Curve* (AUC) dari kurva tersebut kemudian menjadi skor tunggal yang menunjukkan seberapa baik kemampuan model dalam melakukan diskriminasi. Seperti yang ditunjukkan pada Gambar 4.6, model final berhasil mencapai skor AUC sebesar 0.9972, yang mendekati nilai sempurna 1.0. Hal ini menandakan bahwa model memiliki kapabilitas yang sangat luar biasa dalam memisahkan kedua kelas tersebut.




**Gambar 4.6 Kurva ROC-AUC Model**

## 4.4. Pembahasan Hasil

Berdasarkan hasil evaluasi yang dipaparkan pada sub-bab sebelumnya, dapat dilakukan pembahasan yang lebih mendalam untuk menginterpretasi kinerja model dan implikasinya. Secara keseluruhan, model XGBoost yang telah

dioptimisasi menunjukkan performa yang sangat tinggi dan efektif dalam mendeteksi URL *phishing*, termasuk yang telah disamarkan menggunakan *Zero-Width Characters* (ZWC).

Pencapaian akurasi sebesar 97.24% menunjukkan bahwa model memiliki kemampuan yang sangat andal dalam mengklasifikasikan URL secara benar pada sebagian besar kasus. Namun, dalam konteks keamanan siber, metrik *Recall* yang mencapai 97.37% memiliki signifikansi yang lebih krusial. Nilai *Recall* yang tinggi ini mengindikasikan bahwa model berhasil mengidentifikasi hampir semua ancaman *phishing* yang ada di dalam data uji. Hal ini sangat penting karena tujuan utama sistem deteksi adalah untuk meminimalkan jumlah ancaman yang lolos atau tidak terdeteksi (*False Negative*), yang merupakan risiko paling berbahaya bagi pengguna.

Analisis lebih lanjut pada *Confusion Matrix* (Gambar 4.5) memberikan wawasan mengenai jenis kesalahan yang dibuat oleh model. Tercatat hanya terdapat 20 kasus *False Negative*, yang berarti hanya 20 dari 1.143 URL *phishing* yang keliru dianggap aman. Di sisi lain, terdapat 25 kasus *False Positive*, di mana URL yang sah keliru diidentifikasi sebagai *phishing*.

Salah satu contoh kasus *False Positive* yang menarik ditemukan saat pengujian pada beberapa URL sah milik Universitas Gunadarma (contoh: https://praktikum.gunadarma.ac.id/login). Model keliru mengklasifikasikannya sebagai *phishing* kemungkinan besar karena URL tersebut memiliki beberapa karakteristik teknis yang secara kebetulan mirip dengan pola URL berbahaya, seperti penggunaan subdomain yang spesifik (praktikum) dan adanya kata kunci sensitif (login) di dalam struktur URL. Kasus seperti ini menunjukkan bahwa

meskipun sangat akurat, model berbasis fitur teknis terkadang dapat membuat kesalahan tanpa pemahaman kontekstual yang dimiliki manusia.

Selama tahap pengujian, ditemukan pula kasus di mana model menunjukkan keraguan (skor kepercayaan ~50%) saat dihadapkan pada URL *phishing* konvensional yang dibuat dengan sangat "halus" atau rapi. Keraguan ini terjadi karena URL tersebut hanya memiliki sedikit ciri-ciri teknis yang mencurigakan, sehingga total "skor bahaya" yang diakumulasi oleh model tidak cukup untuk membuat keputusan yang sangat percaya diri.

Fenomena ini justru semakin menyoroti pentingnya fitur mengandung_zwc yang menjadi inti dari penelitian ini. Jika serangan *phishing* yang halus tersebut ditambahkan dengan trik ZWC, fitur mengandung_zwc akan berfungsi sebagai sinyal bahaya yang sangat kuat. Fitur ini akan menjadi "pemecah kebuntuan" (*tie-breaker*), yang mengubah keraguan model menjadi sebuah keputusan "*Phishing*" yang sangat meyakinkan. Dengan demikian, dapat disimpulkan bahwa fitur deteksi ZWC tidak hanya menangani ancaman spesifik, tetapi juga berfungsi sebagai lapisan pertahanan krusial untuk memperkuat kemampuan model dalam menghadapi serangan siber yang semakin canggih dan ambigu.

### 4.5. Implementasi Antarmuka (UI)

Untuk mendemonstrasikan kapabilitas model secara praktis dan interaktif, sebuah antarmuka pengguna (UI) sederhana dikembangkan. Tujuan dari implementasi ini adalah untuk memberikan bukti nyata bahwa model yang telah dilatih tidak hanya valid secara statistik, tetapi juga dapat diaplikasikan untuk menganalisis URL secara *real-time*.

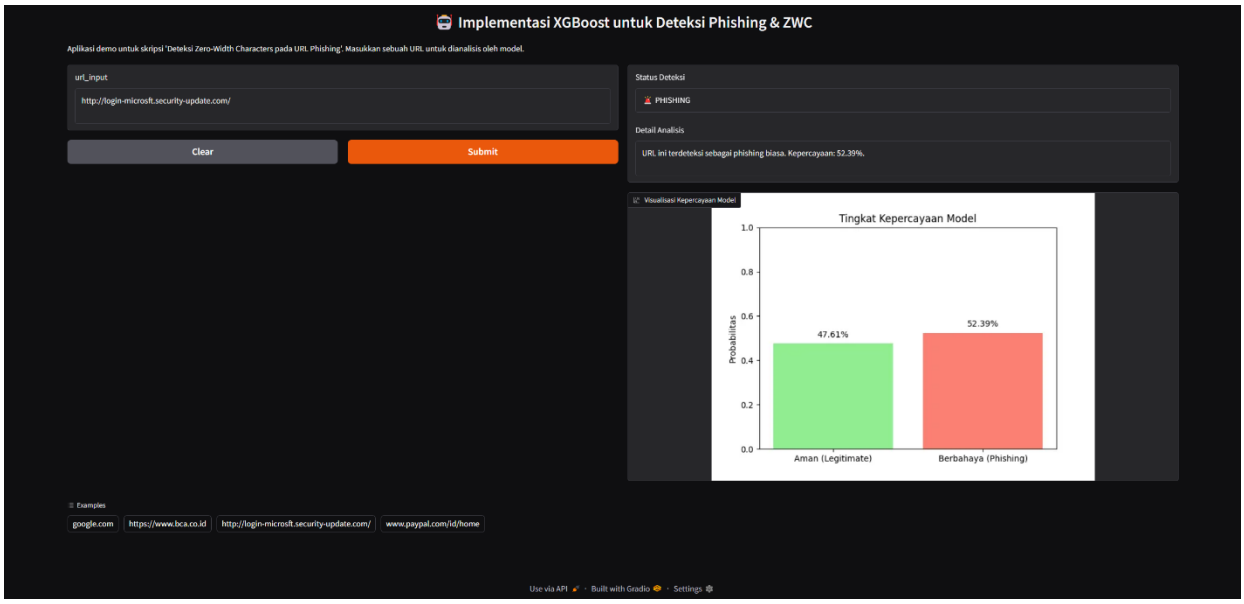
Antarmuka ini dibangun menggunakan *library Gradio* di dalam lingkungan Google Colaboratory. *Gradio* dipilih karena kemampuannya untuk dengan cepat membuat dan membagikan aplikasi web interaktif langsung dari kode Python. UI yang dikembangkan memiliki tiga komponen utama:

a) Area Input: Sebuah kotak teks di mana pengguna dapat memasukkan atau menempelkan URL yang ingin diuji.

b) Tombol Eksekusi: Sebuah tombol bertuliskan "Deteksi Sekarang" yang akan memicu proses analisis.

c) Area Output: Dua buah kotak teks keluaran yang menampilkan hasil analisis, yaitu "Status Deteksi" dan "Detail Analisis".

Gambar 4.7 di bawah ini menunjukkan tampilan dari antarmuka yang telah diimplementasikan.




**Gambar 4.7 Tampilan UI Aplikasi Deteksi *Phishing***

Mekanisme kerja di balik antarmuka ini dimulai ketika pengguna menekan tombol deteksi. Pertama, sistem akan melakukan validasi input untuk memastikan bahwa teks yang dimasukkan memiliki format URL yang valid. Jika input valid,

sebuah fungsi akan mengekstrak fitur-fitur dari URL tersebut, termasuk memeriksa keberadaan *Zero-Width Characters* (ZWC). Vektor fitur yang dihasilkan kemudian dimasukkan ke dalam model final yang telah dilatih dan dimuat sebelumnya.

Secara khusus, logika pada UI dirancang untuk memberikan tiga jenis output yang informatif sesuai dengan tujuan penelitian:

a) AMAN (*Legitimate*): Jika model memprediksi URL sebagai kelas 0 (sah).
b) *PHISHING* (Konvensional): Jika model memprediksi URL sebagai kelas 1 (berbahaya) dan pemeriksaan lebih lanjut tidak menemukan adanya ZWC.
c) *PHISHING* (dengan *Zero-Width* Character): Jika model memprediksi URL sebagai kelas 1 (berbahaya) dan pemeriksaan lebih lanjut menemukan adanya ZWC.

# BAB V.
# KESIMPULAN DAN SARAN

## 5.1. Kesimpulan

Dari analisis dan pembahasan yang telah dilakukan dalam penelitian tentang "Deteksi *Zero-Width Characters* yang Disamarkan pada URL *Phishing* Menggunakan Algoritma XGBoost", dapat ditarik beberapa poin kesimpulan berikut:

a) Model deteksi phishing menggunakan algoritma XGBoost telah berhasil dikembangkan dan diimplementasikan. Model ini mampu mengidentifikasi URL Phishing yang disamarkan dengan Zero-Width Characters (ZWC) melalui proses rekayasa fitur, di mana sebuah fitur khusus (mengandung_zwc) dibuat untuk mengenali keberadaan lima jenis ZWC (ZWSP, ZWNJ, ZWJ, RLM, dan LRM).

b) Model membedakan antara URL *Phishing* dan *legitimate* dengan menganalisis kombinasi dari 88 fitur teknis dan struktural URL—seperti panjang URL, jumlah titik, dan reputasi domain—serta fitur mengandung_zwc yang secara spesifik menargetkan serangan ZWC. Keberadaan fitur ZWC terbukti menjadi indikator yang sangat kuat untuk mendeteksi serangan yang lebih canggih.

c) Model XGBoost yang telah dioptimisasi melalui *hyperparameter* tuning terbukti sangat efektif dalam mendeteksi URL *phishing*. Hal ini dibuktikan dengan pencapaian metrik evaluasi yang sangat tinggi pada data uji, yaitu Akurasi sebesar 97.24%, Presisi 97.03%, Recall 97.37%, dan skor AUC 0.9972. Pencapaian skor *Recall* yang tinggi

mengindikasikan bahwa model memiliki keandalan yang sangat baik untuk menekan potensi ancaman berbahaya yang tidak terdeteksi.

## 5.2. Saran

Meskipun penelitian ini telah berhasil mencapai tujuannya, terdapat beberapa peluang pengembangan yang dapat menjadi acuan untuk penelitian selanjutnya. Dari sisi pengembangan model, dapat dilakukan studi komparatif dengan membandingkan kinerja XGBoost dengan algoritma lain seperti Support Vector Machine (SVM) atau bahkan pendekatan *deep learning* seperti LSTM untuk menangkap pola sekuensial pada URL.

Selain itu, eksplorasi teknik *hyperparameter tuning* yang lebih sistematis seperti *Grid Search* atau *Random Search* dapat dilakukan untuk menemukan kombinasi parameter yang lebih optimal. Dari sisi data dan fitur, model di masa depan dapat disempurnakan dengan menambahkan mekanisme *whitelist* untuk domain-domain terpercaya (contoh: .ac.id, .go.id) guna mengurangi kasus *False Positive.*

Penggunaan *dataset* yang lebih besar dan beragam, terutama yang berisi lebih banyak contoh *phishing* "halus" (*subtle phishing*), juga akan meningkatkan kemampuan generalisasi model. Terakhir, dari sisi implementasi praktis, aplikasi demo yang telah dibangun dapat dikembangkan lebih lanjut menjadi produk nyata yang bermanfaat bagi masyarakat, seperti sebuah ekstensi peramban (*Browser extension*) yang memberikan peringatan secara *real-time* atau sebuah *API* publik yang dapat diintegrasikan dengan sistem keamanan lainnya.

# DAFTAR PUSTAKA


BSSN. (2024). *Lanskap Keamanan Siber Indonesia 2024*. https://ilmubersama.com/2025/03/30/lanskap-keamanan-siber-indonesia-2024-bssn

Erdiyanto, R. P. (2023). Penipuan Mengatasnamakan Bank Berbentuk Phising. In *Jurnal Inovasi Global* (Vol. 1, Issue 2). https://doi.org/10.58344/jig.v1i2.11

Fatunnisa, A., & Marcos, H. (2024). Prediksi Kelulusan Tepat Waktu Siswa SMK Teknik Komputer Menggunakan Algoritma Random Forest. *Jurnal Manajemen Informatika (JAMIKA)*, *14*(1), 101–111. https://doi.org/10.34010/jamika.v14i1.12114

Febrika Ardy, L. A., Istiqomah, I., Ezer, A. E., & Neyman, S. N. (2024). *Phishing* di Era Media Sosial: Identifikasi dan Pencegahan Ancaman di Platform Sosial. *Journal of Internet and Software Engineering*, *1*(4), 11. https://doi.org/10.47134/pjise.v1i4.2753

Geeksforgeeks.org. (2024). *Perbedaan Antara Random Forest dan XGBoost _ GeeksforGeeks*. https://www.geeksforgeeks.org/machine-learning/difference-between-random-forest-vs-xgboost

Kaavija. (2025). *New Phishing Attack Using Zero-Width Characters to Bypass Security Filters*. https://*cybersecuritynews.com/phishing-attack-using-zero-width-characters*

Karo Karo, I. M. (2022). Implementasi Metode XGBoost dan Feature Important untuk Klasifikasi pada Kebakaran Hutan dan Lahan. *Journal of Software Engineering, Information and Communication Technology (SEICT)*, *1*(1), 11–18. https://doi.org/10.17509/seict.v1i1.29347

Maesaroh, S., Ageng, S., Afiyati, A., & Yusuf, M. (2024). *Bahasa Pemrograman Python Eza Budi Perkasa STMIK Atma Luhur*.

Microsoft.com. (2021). *Trend-spotting email techniques: How modern phishing emails hide in plain sight.* https://www.microsoft.com/en-us/security/blog/2021/08/18/trend-spotting-email-techniques-how-modern-phishing-emails-hide-in-plain-sight

Mim Hanifah Permana. (2024). *Pengembangan Model Peramalan Penjualan Fast Food Dengan XGBoost Menggunakan Python.* https://repository.uinjkt.ac.id/dspace/bitstream/123456789/81847/1/MIM%20HANIFAH%20PERMANA-FST.pdf

Murphy, K. (2024). *Reinforcement learning: A Comprehensive Overview*. https://arxiv.org/abs/2412.05265

Pajola, L., & Conti, M. (2021). *Fall of Giants: How popular text-based MLaaS fall against a simple evasion attack*. http://arxiv.org/abs/2104.05996

Alfarizi, M. R. S., Al-farish, M. Z., Taufiqurrahman, M., Ardiansah, G., & Elgar, M. (2023). Penggunaan Python sebagai bahasa pemrograman untuk machine learning dan deep learning. *Karimah Tauhid*, *2*(1), 1-6.

Ryan Afrizal, M., Adi Nugroho, R., Kartini, D., Herteno, R., Ahmad Yani Km, J., & Selatan, K. (2018). *Xgboost Dengan Random Search Hyper-Parameter Tuning Untuk Klasifikasi Situs Phising*.

Sitohang, N. (2023). Jurnal Sains Informatika Terapan ( JSIT ). *Penerapan Data Mining Untuk Peringatan Dini Banjir Menggunakan Metode Klastering K-Means*, *2*(1), 16–20. http://dx.doi.org/10.62357/jsit.v2i2.165

Suhendra, T., & Cs, M. (2021.). *Makalah Pembelajaran Mesin (Machine Learning ) Dosen Pengampu*.

Wijoyo, A., Saputra, A., Rio Arya Pratama, M., & Rahman, R. (2023). *Analisis Serangan Phising dan Strategi Deteksinya*.

Yudhana, A., Umar, R., & Saputra, S. (2022). Fish Freshness Identification Using Machine Learning: Performance Comparison of k-NN and Naïve Bayes Classifier. *Journal of Computing Science and Engineering*, *16*(3), 153–164. https://doi.org/10.5626/JCSE.2022.16.3.153

Yulianti, E. H., Soesanto, O., & Sukmawaty, Y. (2022). Penerapan Metode Extreme Gradient Boosting (XGBOOST) pada Klasifikasi Nasabah Kartu Kredit. *JOMTA Journal of Mathematics: Theory and Applications*, *4*(1). https://doi.org/10.31605/jomta.v4i1.1792

# LAMPIRAN

Lampiran 1 : Implementasi model

```
#
==========================================================
=======================

# LANGKAH 1: HUBUNGKAN GOOGLE COLAB DENGAN GOOGLE
DRIVE

#
==========================================================
=======================

# memunculkan jendela pop-up untuk meminta izin akses.

from google.colab import drive

drive.mount('/content/drive')



print(" ✅ Google Drive berhasil terhubung.")



#
==========================================================
=======================

# LANGKAH 2: IMPORT LIBRARY YANG DIBUTUHKAN

#
==========================================================
=======================

import Pandas as pd

import numpy as np

import random

import xgboost as xgb

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix
```

```
import warnings


warnings.filterwarnings('ignore')

print("✅ Library yang dibutuhkan telah di-import.")


#
==========================================================
======================

# LANGKAH 3: TENTUKAN LOKASI FILE DAN LOAD DATASET

#
==========================================================
======================

file_path = '/content/drive/MyDrive/DATASET/dataset_phishing.csv'


try:

    df = pd.read_csv(file_path)

    print(f"✅ Dataset berhasil dimuat dari: {file_path}")

    print("Shape dataset awal:", df.shape)

except FileNotFoundError:

    print(f"❌ ERROR: File tidak ditemukan di '{file_path}'.")

    print("Pastikan file dan path-nya sudah benar.")

    # Hentikan eksekusi jika file tidak ditemukan

    raise
```

Lampiran 2 : Modifikasi *dataset* dengan menambahkan ZWC

```
#
```

```
# ==========================================================
# LANGKAH 2: MODIFIKASI DATA (REKAYASA FITUR)
# ==========================================================
print("Memulai proses rekayasa fitur ...")

# Daftar 5 karakter ZWC
ZWC_list = [
    '\u200b',  # Zero-Width Space (ZWSP)
    '\u200c',  # Zero-Width Non-Joiner (ZWNJ)
    '\u200d',  # Zero-Width Joiner (ZWJ)
    '\u200f',  # Right-to-Left Mark (RLM)
    '\u200e'   # Left-to-Right Mark (LRM)
]

# Reset kolom 'mengandung_zwc' untuk memulai dari awal
if 'mengandung_zwc' in df.columns:
    df = pd.read_csv(file_path) # Muat ulang dataset asli

df['mengandung_zwc'] = 0

# Ambil indeks dari baris yang statusnya 'phishing'
phishing_indices = df[df['status'] == 'phishing'].index

# Pilih 50% dari URL phishing secara acak untuk disisipi ZWC
```

```
jumlah_modifikasi = len(phishing_indices) // 2

indices_to_modify = random.sample(list(phishing_indices), jumlah_modifikasi)


# Lakukan iterasi untuk memodifikasi URL

for index in indices_to_modify:

    url = df.loc[index, 'url']

    if len(url) > 1:

        insert_pos = random.randint(1, len(url) - 1)

        random_zwc = random.choice(ZWC_list)

        modified_url = url[:insert_pos] + random_zwc + url[insert_pos:]

        df.loc[index, 'url'] = modified_url

        df.loc[index, 'mengandung_zwc'] = 1


print(f" ✅ Rekayasa fitur selesai. {len(indices_to_modify)} URL phishing telah disisipi ZWC.")
```

Lampiran 3 : Program untuk verifikasi modifikasi

```
# Sel untuk verifikasi data

print("Mengecek distribusi data yang telah dimodifikasi:")

print("-------------------------------------------------")


# Membuat tabel silang antara kolom 'mengandung_zwc' dan 'status'

verifikasi_tabel = pd.crosstab(df['mengandung_zwc'], df['status'])


print(verifikasi_tabel)
```

```
print("\n")


# Penjelasan hasil
if verifikasi_tabel.loc[1, 'legitimate'] == 0:
    print("✅ Verifikasi Berhasil!")
    print("Penjelasan: Tidak ada (0) URL 'legitimate' yang disisipi ZWC.")
    print("Modifikasi hanya diterapkan pada URL 'phishing', sesuai harapan.")
else:
    print("❌ Verifikasi Gagal!")
    print("Penjelasan: Ditemukan ada URL 'legitimate' yang ikut termodifikasi.")
```

Lampiran 4 : Kode pembagian *dataset*

```
#
==========================================================
======================

# LANGKAH 5: PERSIAPAN DAN PEMBAGIAN DATA
#
==========================================================
======================

from sklearn.model_selection import train_test_split


print("Mempersiapkan data untuk pelatihan model...")


# 1. Memisahkan Fitur (X) dan Target (y)
# Fitur (X) adalah semua kolom kecuali 'url' (karena bukan numerik) dan 'status'
(karena ini targetnya).
X = df.drop(columns=['url', 'status'])
```

```
# Target (y) adalah kolom 'status' yang diubah menjadi angka (1 untuk phishing, 0 untuk legitimate).
y = df['status'].map({'phishing': 1, 'legitimate': 0})


# 2. Membagi Data menjadi Data Latih dan Data Uji
# test_size=0.2 artinya 20% data untuk pengujian.
# random_state=42 memastikan hasil pembagian data selalu sama setiap kali kode dijalankan.
# stratify=y memastikan proporsi phishing dan legitimate sama di data latih dan data uji.
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state=42,
                                                    stratify=y)


print("\n✅ Data telah berhasil dibagi.")
print(f"Ukuran Data Latih (X_train): {X_train.shape}")
print(f"Ukuran Data Uji (X_test):   {X_test.shape}")
```

Lampiran 5 : Kode verifikasi pembagian *dataset*

```
# Menghitung persentase kelas pada dataset asli sebelum dibagi
print("Proporsi kelas pada dataset KESELURUHAN (y):")
print(y.value_counts(normalize=True))
print("-----------------------------------------------")
```

```
# Menghitung persentase kelas pada DATA LATIH (y_train)
print("Proporsi kelas pada DATA LATIH (y_train):")
print(y_train.value_counts(normalize=True))
print("-----------------------------------------------")

# Menghitung persentase kelas pada DATA UJI (y_test)
print("Proporsi kelas pada DATA UJI (y_test):")
print(y_test.value_counts(normalize=True))
```

Lampiran 6 : Kode visualisasi persebaran *dataset*

```
#
==========================================================
======================

# Visualisasi persebaran dataset
#
==========================================================
======================


import Matplotlib.pyplot as plt
import Pandas as pd
import numpy as np


# Pastikan variabel df (dataframe yang sudah dimodifikasi) ada di memori.
# Jika tidak, jalankan ulang sel yang memuat dan memodifikasi data.
try:
    df
```

```
except NameError:
    print("DataFrame 'df' tidak ditemukan. Harap jalankan sel modifikasi data terlebih dahulu.")
else:
    # --- Menyiapkan Data untuk Visualisasi ---

    # 1. Data SEBELUM Modifikasi
    # Di kondisi awal, semua phishing tidak memiliki ZWC.
    count_legitimate_before = df[df['status'] == 'legitimate'].shape[0]
    count_phishing_before = df[df['status'] == 'phishing'].shape[0]

    # 2. Data SESUDAH Modifikasi
    count_legitimate_after = count_legitimate_before
    count_phishing_with_zwc = df[df['mengandung_zwc'] == 1].shape[0]
    count_phishing_no_zwc = count_phishing_before - count_phishing_with_zwc

    # --- Membuat Grafik ---
    labels = ['Legitimate', 'Phishing (tanpa ZWC)', 'Phishing (dengan ZWC)']
    before_counts = [count_legitimate_before, count_phishing_before, 0]
    after_counts = [count_legitimate_after, count_phishing_no_zwc, count_phishing_with_zwc]

    x = np.arange(len(labels))  # Lokasi label
    width = 0.35  # Lebar bar

    fig, ax = plt.subplots(figsize=(10, 7))
    rects1 = ax.bar(x - width/2, before_counts, width, label='Sebelum Modifikasi',
```

```
color='skyblue')

    rects2 = ax.bar(x + width/2, after_counts, width, label='Sesudah Modifikasi',
color='salmon')



    # Tambahkan label, judul, dan legenda

    ax.set_ylabel('Jumlah URL')

     ax.set_title('Persebaran Dataset Sebelum dan Sesudah Modifikasi ZWC',
pad=20)

    ax.set_xticks(x)

    ax.set_xticklabels(labels)

    ax.legend()



    # Tambahkan label angka di atas setiap bar

    ax.bar_label(rects1, padding=3)

    ax.bar_label(rects2, padding=3)



    fig.tight_layout()

    plt.show()
```

Lampiran 7 : Kode visualisasi pembagian *dataset*

```
#
==========================================================
======================

# Visualisasi pembagian dataset

#
==========================================================
======================
```

```
import Matplotlib.pyplot as plt


# Pastikan variabel X_train dan X_test ada di memori.
try:
    X_train
    X_test
except NameError:
     print("Variabel X_train atau X_test tidak ditemukan. Harap jalankan sel pembagian data terlebih dahulu.")
else:
    # --- Menyiapkan Data untuk Visualisasi ---
    total_data = len(df)
    train_count = len(X_train)
    test_count = len(X_test)

    train_percentage = (train_count / total_data) * 100
    test_percentage = (test_count / total_data) * 100

    # --- Membuat Grafik ---
    labels = ['Dataset']
    width = 0.5

    fig, ax = plt.subplots(figsize=(6, 8))

    # Membuat bar untuk data latih (bagian bawah)
            ax.bar(labels, [train_count], width, label=f'Data Latih
```

```
({train_percentage:.0f}%)', color='cornflowerblue')

    # Membuat bar untuk data uji (di atas data latih)

    ax.bar(labels, [test_count], width, bottom=[train_count],

           label=f'Data Uji ({test_percentage:.0f}%)', color='tomato')



    # Menambahkan label angka di dalam setiap bagian bar

        ax.text(0, train_count/2, str(train_count), ha='center', va='center', color='white', fontsize=12, fontweight='bold')

    ax.text(0, train_count + (test_count/2), str(test_count), ha='center', va='center', color='white', fontsize=12, fontweight='bold')



    # Menambahkan label, judul, dan legenda

    ax.set_ylabel('Jumlah URL')

    ax.set_title('Visualisasi Pembagian Dataset (80% Latih & 20% Uji)', pad=20)

    ax.legend()



    # Menampilkan total data di puncak bar

    total_bar_height = train_count + test_count

        ax.text(0, total_bar_height, f'Total: {total_bar_height}', ha='center', va='bottom', fontsize=11)



    plt.show()
```

Lampiran 8 : Kode demonstrasi model awal (overfitting)

```
#
==========================================================
==
```

```
# SEL KHUSUS UNTUK DEMONSTRASI OVERFITTING
#
==========================================================
==
import xgboost as xgb
from sklearn.metrics import accuracy_score


print("Mensimulasikan pelatihan model awal untuk menunjukkan overfitting...")


# 1. Inisialisasi Model Awal (Tanpa Tuning)
# Ini adalah model "agresif" yang pertama kali kita buat.
model_awal = xgb.XGBClassifier(use_label_encoder=False,
                               eval_metric='logloss',
                               random_state=42)


# 2. Latih Model Awal
# Pastikan X_train dan y_train sudah ada dari sel sebelumnya.
model_awal.fit(X_train, y_train)
print("✅ Model awal (tanpa tuning) berhasil dilatih.")


# 3. Lakukan Pengecekan Overfitting pada Model Awal
print("\n" + "="*40)
print("HASIL PERBANDINGAN AKURASI (MODEL AWAL)")
print("="*40)


# Cek akurasi pada data latih
```

```
y_train_pred_awal = model_awal.predict(X_train)

accuracy_train_awal = accuracy_score(y_train, y_train_pred_awal)

print(f"Akurasi pada Data Latih : {accuracy_train_awal:.4f} ({accuracy_train_awal:.2%})")



# Cek akurasi pada data uji

y_test_pred_awal = model_awal.predict(X_test)

accuracy_test_awal = accuracy_score(y_test, y_test_pred_awal)

print(f"Akurasi pada Data Uji : {accuracy_test_awal:.4f} ({accuracy_test_awal:.2%})")

print("="*40)



gap = accuracy_train_awal - accuracy_test_awal

print(f"\nTerlihat 'celah' performa yang signifikan sebesar: {gap:.2%}")

print("termasuk overfitting.")
```

Lampiran 9 : Kode pelatihan dan validasi model final

```
# ============================================================
# LANGKAH 6 : MELATIH MODEL DENGAN REGULARISASI
# ============================================================

import xgboost as xgb

from sklearn.metrics import accuracy_score
```

```
print("Memulai final tuning untuk mengatasi overfitting... 🛠️")


# Inisialisasi Model dengan Hyperparameter yang lebih ketat
# max_depth=4: Membuat pohon lebih dangkal lagi.
# min_child_weight=3: Mencegah model belajar dari pola yang terlalu spesifik.
model_final = xgb.XGBClassifier(use_label_encoder=False,
                                eval_metric='logloss',
                                random_state=42,
# -- Parameter Tuning Diperketat --
                                max_depth=4,
                                eta=0.1,
                                min_child_weight=3)


# Proses Pelatihan Ulang
model_final.fit(X_train, y_train)


print("\n✅ Model hasil tuning berhasil dilatih!")


# ============================================================
# PENGECEKAN OVERFITTING MODEL
# ============================================================
print("\n" + "="*40)
print("HASIL PERBANDINGAN AKURASI MODEL")
```

```
print("="*40)


# Cek akurasi pada data latih
y_train_pred_final = model_final.predict(X_train)
accuracy_train_final = accuracy_score(y_train, y_train_pred_final)
print(f"Akurasi pada Data Latih : {accuracy_train_final:.4f} ({accuracy_train_final:.2%})")


# Cek akurasi pada data uji
y_test_pred_final = model_final.predict(X_test)
accuracy_test_final = accuracy_score(y_test, y_test_pred_final)
print(f"Akurasi pada Data Uji : {accuracy_test_final:.4f} ({accuracy_test_final:.2%})")
print("="*40)
```

Lampiran 10 : Kode pengecekan data duplikat

```
# OPSIONAL: Mengecek data duplikat sebelum preprocessing
print(f"Jumlah baris data sebelum pengecekan duplikat: {df.shape[0]}")


# Menghitung jumlah baris yang duplikat
jumlah_duplikat = df.duplicated().sum()
print(f"Jumlah baris duplikat yang ditemukan: {jumlah_duplikat}")


if jumlah_duplikat > 0:
    # Menghapus baris duplikat jika ada
```

```
df.drop_duplicates(inplace=True)

print("Baris duplikat telah dihapus.")

print(f"Jumlah baris data setelah penghapusan: {df.shape[0]}")
```

Lampiran 11 : Kode evaluasi kinerja model final

```
#
==========================================================
=======================

# LANGKAH 7: EVALUASI KINERJA MODEL FINAL PADA DATA UJI

#
==========================================================
=======================

from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix

import Seaborn as sns

import Matplotlib.pyplot as plt



print("Mengevaluasi kinerja model hasil tuning pada data uji... 📝")



# 1. Membuat Prediksi

# Menggunakan model_tuned untuk menebak label dari data uji (X_test).

y_pred_tuned = model.predict(X_test)



# 2. Menghitung Metrik Evaluasi

# Membandingkan hasil prediksi (y_pred_tuned) dengan jawaban sebenarnya
(y_test).

accuracy = accuracy_score(y_test, y_pred_tuned)
```

```
precision = precision_score(y_test, y_pred_tuned)
recall = recall_score(y_test, y_pred_tuned)
f1 = f1_score(y_test, y_pred_tuned)
cm = confusion_matrix(y_test, y_pred_tuned)

# 3. Menampilkan Hasil Numerik
print("\n" + "="*35)
print("HASIL EVALUASI MODEL FINAL")
print("="*35)
print(f"Akurasi   : {accuracy:.4f} ({accuracy:.2%})")
print(f"Presisi   : {precision:.4f} ({precision:.2%})")
print(f"Recall    : {recall:.4f} ({recall:.2%})")
print(f"F1-Score  : {f1:.4f} ({f1:.2%})")
print("="*35)

# 4. Visualisasi Confusion Matrix
print("\nVisualisasi Confusion Matrix:")
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Prediksi Legitimate (0)', 'Prediksi Phishing (1)'],
            yticklabels=['Aktual Legitimate (0)', 'Aktual Phishing (1)'])
plt.ylabel('Label Aktual')
plt.xlabel('Label Prediksi')
plt.title('Confusion Matrix Model Final')
plt.show()
```

Lampiran 12 : Kode pengecekan overfitting pada model final

```
#
==========================================================
======================

# PENGECEKAN OVERFITTING UNTUK MODEL FINAL

#
==========================================================
======================

from sklearn.metrics import accuracy_score


print("Mengecek performa model_final pada data latih dan data uji...")

print("\n" + "="*40)

print("HASIL PERBANDINGAN AKURASI (MODEL FINAL)")

print("="*40)


# 1. Cek akurasi pada DATA LATIH menggunakan model_final

y_train_pred_final = model_final.predict(X_train)

accuracy_train_final = accuracy_score(y_train, y_train_pred_final)

print(f"Akurasi pada Data Latih : {accuracy_train_final:.4f}
({accuracy_train_final:.2%})")


# 2. Cek akurasi pada DATA UJI menggunakan model_final

y_test_pred_final = model_final.predict(X_test)

accuracy_test_final = accuracy_score(y_test, y_test_pred_final)

print(f"Akurasi pada Data Uji : {accuracy_test_final:.4f}
({accuracy_test_final:.2%})")

print("="*40)
```

```
# 3. Analisis Celah (Gap)
gap = accuracy_train_final - accuracy_test_final
print(f"\nCelah (Gap) antara akurasi latih dan uji: {gap:.2%}")
if gap < 0.05: # Celah di bawah 5% dianggap sehat
    print("✅ Hasil bagus! Celahnya kecil, model tidak overfitting.")
else:
    print("⚠️ Celahnya cukup besar, model masih sedikit overfitting.")
```

Lampiran 13 : Kode visualisasi kurva ROC-AUC

```
#
==========================================================
======================

# VISUALISASI KURVA ROC & AUC UNTUK MODEL FINAL

#
==========================================================
======================

from sklearn.metrics import roc_curve, roc_auc_score
import Matplotlib.pyplot as plt


print("Membuat visualisasi Kurva ROC dan menghitung AUC untuk
model_final...")


# 1. Dapatkan probabilitas prediksi untuk kelas positif (phishing)
# Metode .predict_proba() memberikan skor kepercayaan model
y_pred_proba_final = model_final.predict_proba(X_test)[:, 1]
```

```
# 2. Hitung nilai AUC

auc_final = roc_auc_score(y_test, y_pred_proba_final)

print(f"\nNilai AUC (Area Under Curve): {auc_final:.4f}")



# 3. Hitung False Positive Rate (FPR) dan True Positive Rate (TPR)

fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba_final)



# 4. Plot Kurva ROC

plt.figure(figsize=(8, 8))

plt.plot(fpr, tpr, label=f'XGBoost Final (AUC = {auc_final:.4f})', color='blue', linewidth=2)

# Plot garis diagonal sebagai pembanding (model acak)

plt.plot([0, 1], [0, 1], 'r--', label='Model Acak (AUC = 0.50)')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate (Tingkat Salah Tangkap)')

plt.ylabel('True Positive Rate (Tingkat Keberhasilan Deteksi / Recall)')

plt.title('Kurva Receiver Operating Characteristic (ROC)')

plt.legend(loc="lower right")

plt.grid(True)

plt.show()
```

Lampiran 14 : Kode untuk menyimpan *dataset* yang telah dilatih

```
# Tentukan lokasi dan nama file model

model_final_save_path                                                    =
```

```
'/content/drive/MyDrive/DATASET/model_phishing_final.json'


# metode .save_model() untuk menyimpan model_final
model_final.save_model(model_final_save_path)


print(f"✅ Model final berhasil disimpan di: {model_final_save_path}")
```

Lampiran 15 : Kode untuk tampilan UI

```
#
==========================================================
======================

# KODE LENGKAP UI GRADIO DENGAN VISUALISASI
#
==========================================================
======================


# Langkah 1: Instal Gradio & Import Library
print("Menginstal Gradio...")
!pip install Gradio -q
!pip install tldextract -q


import Gradio as gr
import Pandas as pd
import xgboost as xgb
from urllib.parse import urlparse
import re
```

```
import tldextract

from google.colab import drive

import Matplotlib.pyplot as plt


# Menghubungkan ke Google Drive

print("\nMenghubungkan ke Google Drive...")

try:

    drive.mount('/content/drive', force_remount=True)

except Exception as e:

    print(f"Error saat menghubungkan Drive: {e}")


# Langkah 2: Muat Model & Siapkan Konfigurasi

print("\nMemuat model terlatih dari Google Drive...")

model_final_load_path =
'/content/drive/MyDrive/DATASET/model_phishing_final.json'

try:

    model_siap_pakai = xgb.XGBClassifier()

    model_siap_pakai.load_model(model_final_load_path)

    print("✅ Model final berhasil dimuat dan siap digunakan!")

except Exception as e:

    print(f"❌ Gagal memuat model. Pastikan path '{model_final_load_path}'
sudah benar.")

    raise


FEATURE_COLUMNS = [

    'length_url', 'length_hostname', 'ip', 'nb_dots', 'nb_hyphens', 'nb_at', 'nb_qm',
'nb_and', 'nb_or', 'nb_eq',
```

```
    'nb_underscore', 'nb_tilde', 'nb_percent', 'nb_slash', 'nb_star', 'nb_colon', 'nb_comma', 'nb_semicolumn',

    'nb_dollar', 'nb_space', 'nb_www', 'nb_com', 'nb_dslash', 'http_in_path', 'https_token', 'ratio_digits_url',

    'ratio_digits_host', 'punycode', 'port', 'tld_in_path', 'tld_in_subdomain', 'abnormal_subdomain', 'nb_subdomains',

    'prefix_suffix', 'random_domain', 'shortening_service', 'path_extension', 'nb_redirection', 'nb_external_redirection',

  'length_words_raw', 'char_repeat', 'shortest_words_raw', 'shortest_word_host', 'shortest_word_path', 'longest_words_raw',

  'longest_word_host', 'longest_word_path', 'avg_words_raw', 'avg_word_host', 'avg_word_path', 'phish_hints', 'domain_in_brand',

    'brand_in_subdomain', 'brand_in_path', 'suspecious_tld', 'statistical_report', 'nb_hyperlinks', 'ratio_intHyperlinks', 'ratio_extHyperlinks',

            'ratio_nullHyperlinks', 'nb_extCSS', 'ratio_intRedirection', 'ratio_extRedirection', 'ratio_intErrors', 'ratio_extErrors', 'login_form',

        'external_favicon', 'links_in_tags', 'submit_email', 'ratio_intMedia', 'ratio_extMedia', 'sfh', 'iframe', 'popup_window', 'safe_anchor',

            'onmouseover', 'right_clic', 'empty_title', 'domain_in_title', 'domain_with_copyright', 'whois_registered_domain', 'domain_registration_length',

    'domain_age', 'web_traffic', 'dns_record', 'google_index', 'page_rank', 'mengandung_zwc'

]



# Langkah 3: Fungsi Ekstraksi Fitur

def extract_features_from_url(url):

  features = {}

  if not url.lower().startswith('http'):

    url = 'http://' + url

  try:
```

```
        parsed_url = urlparse(url)
        ext = tldextract.extract(url)
        hostname = parsed_url.hostname if parsed_url.hostname else ''
        path = parsed_url.path if parsed_url.path else ''
    except Exception:
        hostname = ''
        path = ''
    features['length_url'] = len(url)
    features['length_hostname'] = len(hostname)
        features['ip'] = 1 if re.match(r"^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$",
hostname) else 0
    features['nb_dots'] = url.count('.')
    features['nb_hyphens'] = url.count('-')
    features['nb_at'] = url.count('@')
    features['nb_qm'] = url.count('?')
    features['nb_and'] = url.count('&')
    features['nb_eq'] = url.count('=')
    features['nb_underscore'] = url.count('_')
    features['nb_slash'] = url.count('/')
    features['https_token'] = 1 if url.startswith('https') else 0
    features['ratio_digits_url'] = sum(c.isdigit() for c in url) / len(url) if len(url) >
0 else 0
        features['ratio_digits_host'] = sum(c.isdigit() for c in hostname) /
len(hostname) if len(hostname) > 0 else 0
    features['nb_www'] = url.lower().count('www')
    features['nb_com'] = url.lower().count('com')
    features['mengandung_zwc'] = 1 if any(zwc in url for zwc in ['\u200b', '\u200c',
```

```
'\u200d', '\u200f', '\u200e']) else 0
    for col in FEATURE_COLUMNS:
        if col not in features:
            features[col] = 0
    return pd.DataFrame([features], columns=FEATURE_COLUMNS)


# Langkah 4: Fungsi Utama untuk Prediksi
def deteksi_phishing(url_input):
    url_input = url_input.strip()
    if not url_input:
        return " ⚠️ Harap masukkan URL terlebih dahulu.", "", None
    ext = tldextract.extract(url_input)
    if not ext.domain or not ext.suffix or ' ' in url_input:
        return " ❌ Input tidak valid. Harap masukkan format URL yang benar.", "",
None


    try:
        features_df = extract_features_from_url(url_input)
        prediction = model_siap_pakai.predict(features_df)
            prediction_proba = model_siap_pakai.predict_proba(features_df)[0] #
mendapatkan array probabilitas


        # --- Membuat Visualisasi ---
        labels = ['Aman (Legitimate)', 'Berbahaya (Phishing)']
        probabilities = [prediction_proba[0], prediction_proba[1]]
```

```
        fig, ax = plt.subplots()

        bars = ax.bar(labels, probabilities, color=['lightgreen', 'salmon'])

        ax.set_ylabel('Probabilitas')

        ax.set_title('Tingkat Kepercayaan Model')

        ax.set_ylim(0, 1) # Set y-axis limit from 0 to 1 (100%)


        # Add percentage labels on top of bars

        for bar in bars:

            yval = bar.get_height()

                ax.text(bar.get_x() + bar.get_width()/2.0, yval + 0.02, f'{yval:.2%}',
ha='center', va='bottom')

        # ------------------------


        if prediction[0] == 1:

            confidence = probabilities[1] * 100

            hasil_utama = " 🚨 *PHISHING*"

            if features_df['mengandung_zwc'].iloc[0] == 1:

                hasil_detail = f"URL ini sangat berbahaya dan menggunakan trik ZWC!
Kepercayaan: {confidence:.2f}%."

            else:

                hasil_detail = f"URL ini terdeteksi sebagai *phishing* biasa. Kepercayaan:
{confidence:.2f}%."

        else:

            confidence = probabilities[0] * 100

            hasil_utama = f" ✅ AMAN (*Legitimate*)"

            hasil_detail = f"URL ini terlihat sebagai situs yang sah. Kepercayaan:
{confidence:.2f}%."
```

```
        return hasil_utama, hasil_detail, fig


    except Exception as e:
        return f"❌ Terjadi kesalahan saat analisis.", f"Error: {e}", None


# Langkah 5: Luncurkan Antarmuka Gradio dengan Output Plot
iface = gr.Interface(
    fn=deteksi_phishing,
    inputs=gr.Textbox(lines=2, placeholder="Contoh: google.com atau URL yang dirasa mencurigakan ..."),
    outputs=[
        gr.Textbox(label="Status Deteksi"),
        gr.Textbox(label="Detail Analisis"),
        gr.Plot(label="Visualisasi Kepercayaan Model") # Add the Plot component
    ],
    title="🤖 Implementasi XGBoost untuk Deteksi Phishing & ZWC",
     description="Aplikasi demo untuk skripsi 'Deteksi Zero-Width Characters pada URL Phishing'. Masukkan sebuah URL untuk dianalisis oleh model.",
    examples=[
        ["google.com"],
        ["https://www.bca.co.id"],
        ["http://login-microsft.security-update.com/"],
        ["www.paypal.com/id/home"]
    ],
    flagging_mode='never' # Corrected parameter value
```

```
)

iface.launch(share=True, debug=True)
```

SKRIPSI_AHMAD-ASADEL_2109020157-1757910085913

ORIGINALITY REPORT

| 17% | 14% | 11% | 7% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | files.osf.io Internet Source | 1% |
| 2 | www.mclean.net.nz Internet Source | 1% |
| 3 | repository.upi.edu Internet Source | 1% |
| 4 | journal.pubmedia.id Internet Source | 1% |
| 5 | docplayer.info Internet Source | <1% |
| 6 | Submitted to Universitas Diponegoro Student Paper | <1% |
| 7 | journal.lembagakita.org Internet Source | <1% |
| 8 | journal.thamrin.ac.id Internet Source | <1% |
| 9 | komsciguy.com Internet Source | <1% |

| | | |
|---|---|---|
| 10 | Submitted to Universitas Indonesia Student Paper | <1% |
| 11 | jurnal.polgan.ac.id Internet Source | <1% |
| 12 | rcf-indonesia.org Internet Source | <1% |
| 13 | Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper | <1% |
| 14 | Submitted to Universitas Muhammadiyah Sumatera Utara Student Paper | <1% |
| 15 | ocelot.ca Internet Source | <1% |
| 16 | eprints.uad.ac.id Internet Source | <1% |
| 17 | ejournal.nusamandiri.ac.id Internet Source | <1% |
| 18 | jurnal.harianregional.com Internet Source | <1% |
| 19 | repository.amikomsolo.ac.id Internet Source | <1% |
| 20 | www.bagiteknologi.com Internet Source | <1% |

| | |
|---|---|
| 21 aip.vse.cz Internet Source | <1% |
| 22 repository.umsu.ac.id Internet Source | <1% |
| 23 Merry Anggraeni, Hillman Akhyar Damanik. "Implementation of MTCNN Architecture on Different Pixel Dimension Classes and Plotting Multi-Face on Detection Results", Jurnal Pekommas, 2025 Publication | <1% |
| 24 Muhammad Farrel Golfantara. "PENGGUNAAN ALGORITMA YOLO V8 UNTUK IDENTIFIKASI REMPAH-REMPAH", Jurnal Informatika dan Teknik Elektro Terapan, 2024 Publication | <1% |
| 25 eprints.polsri.ac.id Internet Source | <1% |
| 26 www.invisible-character.top Internet Source | <1% |
| 27 Panji Novantara, Risteruw Leonardo Firmansyah, Marrilyn Arismawati. "Deteksi Hama Penyakit Daun Padi Dengan Menggunakan Teknik Optimasi Deep Learning Convolutional Neural Network", bit-Tech, 2025 Publication | <1% |
| 28 Submitted to Bath Spa University College Student Paper | <1% |

| | | |
|---|---|---|
| 29 | Submitted to Murdoch University Student Paper | <1% |
| 30 | eprints.unpak.ac.id Internet Source | <1% |
| 31 | www.mediawiki.org Internet Source | <1% |
| 32 | www.scribd.com Internet Source | <1% |
| 33 | Fanny Ramadhani, Dian Septiana, Sisti NadiaAmalia, Putri Maulidina Fadilah, Andy Satria. "KLASIFIKASI RISIKO GIZI BURUK PADA IBU HAMIL MENGGUNAKAN METODE RANDOM FOREST", Djtechno: Jurnal Teknologi Informasi, 2024 Publication | <1% |
| 34 | community.intel.com Internet Source | <1% |
| 35 | pdfcookie.com Internet Source | <1% |
| 36 | repository.uin-suska.ac.id Internet Source | <1% |
| 37 | Submitted to Coventry University Student Paper | <1% |

| | | |
|---|---|---|
| 38 | Dadang Iskandar Mulyana, Guruh Taruno Putra. "Deteksi Objek bertumpuk Gerak Tangan Bahasa Isyarat SIBI dengan Algoritma LSTM-Holistic", INTECOMS: Journal of Information Technology and Computer Science, 2025<br>Publication | <1% |
| 39 | Submitted to Kaplan College<br>Student Paper | <1% |
| 40 | Rizky Wahyudi, Kristin Impana Manik, Muhammad Alfin, John Bush Henrydunan, Muhammad Haikal Al Majid, Kana Saputra. "KLASIFIKASI PENYAKIT MIGRAIN MENGGUNAKAN METODE SUPPORT VECTOR MACHINE", Jurnal Manajemen Informatika dan Sistem Informasi, 2025<br>Publication | <1% |
| 41 | Submitted to Universitas Prima Indonesia<br>Student Paper | <1% |
| 42 | Submitted to Universitas Putera Batam<br>Student Paper | <1% |
| 43 | adoc.pub<br>Internet Source | <1% |
| 44 | core.ac.uk<br>Internet Source | <1% |

| | | |
|---|---|---|
| 45 | cybersecuritynews.com Internet Source | <1% |
| 46 | www.coursehero.com Internet Source | <1% |
| 47 | Talitha Hurin Salsabila, Tri Mei Indrawati, Revienda Anita Fitrie. "Meningkatkan Efisiensi Pengambilan Keputusan Publik melalui Kecerdasan Buatan", Journal of Internet and Software Engineering, 2024 Publication | <1% |
| 48 | Submitted to UM Surabaya Student Paper | <1% |
| 49 | Wanda Putra Ramadhan, Didi Juardi. "ANALISIS SENTIMEN ULASAN APLIKASI BTN MOBILE MENGGUNAKAN ALGORITMA NAIVE BAYES", Jurnal Informatika dan Teknik Elektro Terapan, 2025 Publication | <1% |
| 50 | journal.aritekin.or.id Internet Source | <1% |
| 51 | journal.nurulfikri.ac.id Internet Source | <1% |
| 52 | repository.unj.ac.id Internet Source | <1% |

| | | |
|---|---|---|
| 53 | Anindita Puspa Ayu Prayogi, Altha Inas Shofyana, Dewi Putriani. "Prediksi *Credit Card Approval* Menggunakan Algoritma *Random Forest*", Jurnal Ilmu Komputer dan Multimedia, 2025<br>Publication | <1% |
| 54 | Sartini, Sumarna, Abdul Hamid, Ahmad Hafidzul Kahfi, Nicodias Palasara. "REVOLUSI DIAGNOSIS: OPTIMASI RANDOM TREE-PSO UNTUK PENYAKIT GINJAL KRONIS", Jurnal Informatika dan Rekayasa Elektronik, 2025<br>Publication | <1% |
| 55 | broonet.com<br>Internet Source | <1% |
| 56 | cahayasenyuman.blogspot.com<br>Internet Source | <1% |
| 57 | es.scribd.com<br>Internet Source | <1% |
| 58 | Muhammad Wisnu Nugroho. "Analisis Performa Algoritma Random Forest dalam Mengatasi Overfitting pada Model Prediksi", Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi), 2025<br>Publication | <1% |
| 59 | Zian Asti Dwiyanti, Cahyo Prianto. "Prediksi Cuaca Kota Jakarta Menggunakan Metode Random Forest", Jurnal Tekno Insentif, 2023<br>Publication | <1% |

| | | |
|---|---|---|
| 60 | ejournal.uniska-kediri.ac.id Internet Source | <1% |
| 61 | ejurnal.stmik-budidarma.ac.id Internet Source | <1% |
| 62 | eprints.itn.ac.id Internet Source | <1% |
| 63 | eprints.upj.ac.id Internet Source | <1% |
| 64 | repository.unand.ac.id Internet Source | <1% |
| 65 | repository.usni.ac.id Internet Source | <1% |
| 66 | semnasristek.sakaintek.com Internet Source | <1% |
| 67 | www.muftins.gov.my Internet Source | <1% |
| 68 | DHIKA MALITA PUSPITA ARUM, KARTIKA IMAM SANTOSO, ANDRI TRIYONO, EKO SUPRIYADI, AGUS SUSILO NUGROHO, Edy Widodo. "Algoritma Random Forest, Decision Tree dan XGboost Untuk Klasifikasi Stunting Pada Balita", Jurnal Transformatika, 2025 Publication | <1% |

| | |
|---|---|
| 69 Royan Habibie Sukarna, Holilah Holilah, Fitri Damyati, Mohamad Hilman. "Analisis Prediksi Kelulusan Mahasiswa Universitas Sultan Ageng Tirtayasa Menggunakan Algoritma Machine Learning dan Feature Selection", Jurnal Ilmiah Sains dan Teknologi, 2024<br>Publication | <1% |
| 70 ejournal.istn.ac.id Internet Source | <1% |
| 71 eprints.umpo.ac.id Internet Source | <1% |
| 72 ms.wikipedia.org Internet Source | <1% |
| 73 repository.its.ac.id Internet Source | <1% |
| 74 tahtamedia.co.id Internet Source | <1% |
| 75 www.microsoft.com Internet Source | <1% |
| 76 zh.scribd.com Internet Source | <1% |

| | | |
|---|---|---|
| 77 | Galih Hermawan. "Klasifikasi Pengemudi Terganggu Berdasarkan Citra Menggunakan ResNet-50", Jurnal Informatika Komputer, Bisnis dan Manajemen, 2024<br>Publication | <1% |
| 78 | Heru Teguh Santoso Teguh. "Implementasi Data Minning Clustering Dalam Mengelompokan Kasus Perceraian Yang Terjadi Di Provinsi Jawa Timur Menggunakan Algoritma K-Means", JAMI: Jurnal Ahli Muda Indonesia, 2025<br>Publication | <1% |
| 79 | Muhammad Reyan, Franindya Purwaningtyas. "ANALISIS SENTIMEN ULASAN APLIKASI IBI LIBRARY PADA GOOGLE PLAY STORE MENGGUNAKAN NAIVE BAYES CLASSIFIER", Djtechno: Jurnal Teknologi Informasi, 2025<br>Publication | <1% |
| 80 | Rafi Rahmadani, Abdul Rahim, RudimanRudiman. "ANALISIS SENTIMEN ULASAN OJOL THE GAME" DI GOOGLE PLAY STORE MENGGUNAKAN ALGORITMA NAIVE BAYES DAN MODEL EKSTRAKSI FITUR TF-IDF UNTUK MENINGKATKAN KUALITAS GAME", Jurnal Informatika dan Teknik Elektro Terapan, 2024<br>Publication | <1% |

| | | | |
|---|---|---|---|
| Exclude quotes | On | Exclude matches | Off |
| Exclude bibliography | Off | | |