BAB I
PENDAHULUAN

A. Latar Belakang

Perkembangan teknologi yang sangat pesat di Indonesia bahkan dunia mempengaruhi berbagai bidang aspek kehidupan [1]. Teknologi kini menjadi infrastruktur penting yang mendukung aktivitas sehari-hari. Namun, kemudahan ini juga membuka peluang bagi kejahatan siber yang semakin meresahkan, seperti *Phishing*.

Phishing adalah tindakan kejahatan siber yang bertujuan memancing korban dengan cara menipu atau menjebak. Dengan cara ini, pelaku bisa mendapatkan informasi penting dari korban tanpa disadari. *Phishing* merupakan salah satu bentuk kejahatan dunia maya [2]. Laporan keamanan siber global menunjukkan bahwa serangan *Phishing* terus meningkat setiap tahun dan menyebabkan kerugian finansial besar bagi individu maupun institusi.

Untuk menghadapi ancaman *Phishing*, berbagai metode deteksi telah dikembangkan, mulai dari *blacklisting* hingga *Machine Learning*. *Machine Learning*, sebagai bagian dari kecerdasan buatan, menawarkan pendekatan adaptif yang dapat mempelajari pola data, mengenali aktivitas mencurigakan, dan mendeteksi ancaman secara otomatis tanpa aturan tetap [3]. Salah satu metode yang efisien adalah deteksi berbasis fitur URL, yang tidak memerlukan akses ke situs web target sehingga lebih cepat dan aman bagi pengguna.

Dalam *Machine Learning*, ada banyak algoritma klasifikasi yang bisa digunakan. Dua algoritma yang populer dan sering dibandingkan adalah *Random Forest* dan *Extreme Gradient Boosting* (*XGBoost*). *Random Forest* adalah metode *ensemble learning* yang menggabungkan banyak pohon keputusan untuk menghasilkan prediksi yang lebih stabil dan akurat [4]. *XGBoost* adalah algoritma berbasis *gradient boosting decision tree* (GBDT) yang dirancang untuk meningkatkan efisiensi komputasi dan akurasi prediksi [5].

Pemilihan kedua algoritma ini didukung oleh bukti empiris dari studi komparatif sebelumnya [6]. Penelitian tersebut menyimpulkan bahwa untuk mendeteksi halaman *web Phishing*, dari 16 pengklasifikasi AI yang diuji, *GradientBoostingClassifier* dan *RandomForestClassifier* memiliki tingkat akurasi

tertinggi, sekitar 97% [6]. Berdasarkan literatur ini, kedua algoritma tersebut terbukti lebih unggul dibandingkan metode klasifikasi lainnya. Karena itu, penelitian ini akan membandingkan kinerja kedua algoritma ini secara khusus untuk kasus URL *Phishing* obfuskasi.

Tantangan terbesar dalam deteksi Phishing saat ini adalah situs web Phishing yang hampir sama persis dengan situs web asli. Penipu meniru situs web sah tanpa kesalahan, sehingga sulit membedakan antara situs asli dan tiruan dengan mata telanjang. Mereka memastikan antarmuka dan fungsi pengguna pada situs Phishing identik dengan situs sah [7]. Penyerang siber juga terus mengembangkan teknik manipulasi untuk menyamarkan identitas situs mereka, seperti penggunaan karakter khusus, kata manipulasi, penyisipan karakter acak, dan subdomain yang kompleks. Teknik ini dikenal sebagai teknik obfuskasi, yang dirancang untuk mengelabui deteksi otomatis berbasis aturan sederhana. Akibatnya, model yang unggul pada dataset bersih bisa mengalami penurunan performa drastis saat menghadapi URL yang telah diobfuskasi.

Penelitian sebelumnya tentang deteksi *Phishing link* dengan algoritma *Bagging* dan *Boosting* mengevaluasi performa model menggunakan beberapa metrik utama. Studi literatur tersebut menunjukkan bahwa *Extreme Gradient Boosting* (*XGBoost*) memiliki keunggulan mutlak dan mengungguli *Random Forest* di semua metrik evaluasi, dengan Akurasi 92%, Presisi 91%, dan *Recall* 94%. Nilai ini secara konsisten lebih tinggi dibandingkan *Random Forest* yang memperoleh Akurasi 91%, Presisi 90%, dan *Recall* 92% [8].

Walaupun perbandingan kinerja antara *Random Forest* dan *XGBoost* sudah sering dilakukan, sebagian besar penelitian sebelumnya masih berfokus pada metrik evaluasi standar di dataset umum tanpa mempertimbangkan efektivitas model dalam menangani variasi URL *Phishing* obfuskasi [9]. Masih ada celah penelitian untuk mengetahui bagaimana kedua algoritma ini mengenali pola fitur URL yang sengaja dibuat untuk menyamarkan tanda-tanda *Phishing*.

Oleh karena itu, penelitian ini bertujuan membandingkan kinerja algoritma *Random Forest* dan *XGBoost* untuk klasifikasi URL *Phishing* obfuskasi. Dengan mengukur performa kedua algoritma dalam mengklasifikasikan data yang telah melalui berbagai teknik penyamaran, penelitian ini diharapkan dapat memberikan

rekomendasi algoritma paling optimal untuk sistem keamanan siber yang menghadapi taktik obfuskasi dinamis di dunia nyata.

B. Perumusan Masalah

Berdasarkan latar belakang di atas, terdapat rumusan masalah dalam penelitian ini, yaitu bagaimana perbandingan kinerja algoritma *Random Forest* dan *XGBoost* dalam mengklasifikasikan situs *Phishing* yang menggunakan teknik obfuskasi pada fitur URL, diukur berdasarkan metrik akurasi, presisi, *Recall*, dan *F1-Score*?.

C. Batasan Penelitian

Agar penelitian ini lebih terarah, penulis menetapkan batasan masalah sebagai berikut:

1. Data yang digunakan adalah dataset URL yang dikumpulkan dari sumber publik seperti *PhishTank* untuk URL *phishing* dan *Tranco List* untuk URL aman(*Legitimate*).
2. Kelas URL yang digunakan hanya kelas *Phishing* dan kelas aman (*Legitimate*).
3. Dataset yang digunakan fokus pada URL yang mengandung teknik obfuskasi, yaitu manipulasi leksikal, struktural, atau penggunaan karakter unik untuk menyamarkan identitas asli situs.
4. Fitur yang digunakan terbatas pada fitur URL statis (leksikal, struktural, dan statistik) dan tidak melibatkan analisis konten halaman web atau waktu respons server.
5. Algoritma yang dibandingkan dalam klasifikasi ini hanya *Random Forest* dan *XGBoost*.
6. Metrik evaluasi yang digunakan untuk mengukur kinerja klasifikasi meliputi *Accuracy, Precision, Recall, dan F1-Score* berdasarkan hasil dari *Confusion Matrix*.

D. Tujuan Penelitian

Pada penelitian sebelumnya telah dilakukan perbandingan kinerja algoritma *Random Forest* dan *XGBoost* dalam mendeteksi situs *Phishing* menggunakan dataset umum tanpa mengkhususkan tingkat deteksi pada level obfuskasi pada fitur URL, lalu pada penelitian sebelumnya juga hanya memberikan hasil berdasarkan metrik akurasi, presisi, dan *Recall*. Sementara, tujuan dari penelitian ini adalah menganalisis dan membandingkan kinerja algoritma *Random Forest* dan *XGBoost* dalam mengklasifikasikan situs *Phishing* yang menggunakan teknik obfuskasi pada fitur URL, guna menentukan algoritma mana yang memberikan hasil paling optimal berdasarkan *confusion matrix* yang menghasilkan metrik *Accuracy, Precision, Recall,* dan *F1-Score*.

E. Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat sebagai berikut:

1. Manfaat Teoretis: Manfaat Teoretis: Menambah wawasan dan referensi akademis mengenai perbandingan kinerja algoritma *Machine Learning* dalam keamanan siber, khususnya terkait efektivitas model klasifikasi dalam mendeteksi taktik penyamaran (obfuskasi) pada fitur struktural URL.
2. Manfaat Praktis: Memberikan rekomendasi teknis kepada pengembang sistem keamanan siber atau *security analyst* mengenai pemilihan algoritma yang paling efektif untuk mendeteksi situs *Phishing* yang menggunakan teknik obfuskasi di dunia nyata.

BAB II
KAJIAN LITERATUR

Bab ini menyajikan landasan teori dan tinjauan pustaka yang relevan dengan penelitian mengenai klasifikasi URL *Phishing* yang menggunakan teknik obfuskasi. Pembahasan dimulai dengan mengulas penelitian – penelitian terdahulu yang membandingkan algoritma *Random Forest* dan *XGBoost* dalam *Domain* keamanan siber. Selanjutnya, bab ini memaparkan teori dasar mengenai mekanisme *Phishing*, anatomi URL, serta teknik obfuskasi yang sering digunakan oleh penyerang untuk mengelabui sistem deteksi. Selain itu, dijelaskan pula konsep dasar *Machine Learning* dan algoritma *ensemble learning* yang menjadi fokus utama dalam melakukan analisis komparatif kinerja pada penelitian ini

A. Tinjauan Pustaka

Metode perlindungan siber terhadap serangan *Phishing* telah berkembang pesat, dari penggunaan *blacklist* ke ekstraksi cerdas berbasis analisis tautan leksikal. URL bukan hanya alamat acak, tetapi juga kumpulan data yang mencerminkan niat pembuatnya. Hal ini menjadi dasar bagi banyak penelitian keamanan siber modern yang fokus pada deteksi dini melalui analisis karakter URL. Penelitian oleh Ghalechyan dkk. menunjukkan bahwa metrik leksikal dasar sangat berkaitan dengan niat jahat. Baik model deterministik maupun probabilistik, metode leksikal secara konsisten memberikan akurasi tinggi dalam mengklasifikasi URL panjang dan pendek (kurang dari 50 karakter) [10]. Pentingnya panjang URL juga didukung oleh studi komparatif lain. Eksperimen pada berbagai arsitektur hibrida membuktikan bahwa hanya dengan mengekstrak fitur leksikal dari struktur tautan, tanpa memindai isi halaman, deteksi bisa dilakukan jauh lebih efisien [11]. Berbagai jurnal *peer-reviewed* menegaskan bahwa panjang URL, penggunaan titik tidak biasa pada *Subdomain*, dan karakter khusus adalah metrik utama yang terbukti efektif secara ilmiah untuk mendeteksi situs *Phishing* sebelum kode berbahaya dijalankan.

Meskipun analisis fitur leksikal dasar terbukti efisien, ancaman *Phishing* kini semakin canggih dengan teknik penyembunyian (obfuskasi) yang terstruktur. Pelaku *Phishing* secara sistematis menggunakan rekayasa sosial dan obfuskasi agar

URL buatan mereka mirip dengan URL yang sah. Mereka sering menambahkan kata kunci seperti "*secure*", "*login*", atau "*banking*", atau menyembunyikan identitas *server* di balik karakter Unicode yang rumit. Ancaman ini semakin jelas setelah munculnya model deteksi anomali *TabNet* dalam penelitian terbaru [12]. Laporan tersebut menunjukkan kelemahan sistem keamanan tradisional dan menegaskan bahwa identifikasi URL yang telah disamarkan kini menjadi tantangan besar di dunia *cybersecurity* [12]. Manipulasi ini membuat metode penghitungan karakter biasa menjadi kurang efektif dan mudah dilewati. Untuk mengatasi kelemahan ini, pendekatan baru harus mampu mengenali pola laten. Solusi ini diwujudkan dengan kerangka deteksi cerdas yang tidak hanya menghitung fitur dasar, tetapi juga menggabungkan properti leksikal, struktur *host*, dan atribut struktural URL secara menyeluruh [13]. Dengan integrasi fitur ini, sistem deteksi dapat mengubah anomali kecil, seperti perpindahan tanda hubung atau penggunaan titik tidak biasa, menjadi probabilitas ancaman yang terukur. Oleh karena itu, menambahkan variabel deteksi kata manipulatif dan menghitung kedalaman jalur sangat relevan untuk melawan serangan obfuskasi URL.

Karena data fitur URL kini sangat kompleks, dengan puluhan hingga ratusan atribut numerik, algoritma *Machine Learning* konvensional sudah tidak cukup. Peneliti siber kini banyak menggunakan algoritma klasifikasi berbasis Pohon Keputusan, terutama *Random Forest*, yang dikenal sangat andal untuk data multivariabel. *Random Forest* menggunakan pendekatan ansambel yang efektif mencegah *Overfitting* dengan membangun banyak pohon secara acak (*bagging*) dan menggabungkan hasilnya. Keunggulan *Random Forest* dalam mendeteksi *Phishing* telah dibuktikan lewat berbagai eksperimen besar. Dalam uji coba yang membandingkan enam algoritma, termasuk regresi logistik, *K-Nearest Neighbors*, *Naive Bayes, Random Forest, Support Vector Machine* (SVM), dan *XGBoost*, *Random Forest* menunjukkan keunggulan mutlak atas metode statistik klasik [14]. Keunggulan performa dari metode gabungan *ensemble tree* ini turut diverifikasi ulang pada serangkaian uji coba terapan yang dirancang khusus untuk melindungi infrastruktur jaringan perangkat *Internet of Things* (IoT) pada industri 4.0. Melalui kerangka pengujian berskala industri yang secara independen mengadu *Random Forest*, SVM, dan *XGBoost*, hasil eksperimen secara konsisten menunjukkan

tingginya akurasi dari pendekatan *tree-bas Keunggulan ini juga terbukti dalam uji coba untuk melindungi jaringan perangkat IoT di industri 4.0. Pengujian industri yang membandingkan Random Forest, SVM, dan XGBoost secara konsisten menunjukkan akurasi tinggi dari metode tree-based* [15]. Hasil eksperimen lintas negara ini menjadi dasar kuat bahwa memilih *Random Forest* sebagai algoritma utama dalam skripsi ini adalah keputusan metodologis yang tepat, karena sistem ini stabil, mudah dipahami, dan tahan terhadap variasi leksikal.

Meski *Random Forest* masih dominan, perkembangan algoritma kecerdasan buatan telah melahirkan pesaing kuat, yaitu *XGBoost* (*eXtreme Gradient Boosting*). Berbeda dengan *Random Forest* yang membangun banyak pohon secara paralel (*bagging*), *XGBoost* membangun pohon secara berurutan, di mana setiap pohon baru memperbaiki kesalahan pohon sebelumnya. Proses ini didukung oleh regularisasi bawaan yang mencegah pohon menjadi terlalu kompleks, sehingga *XGBoost* dapat melatih model dengan banyak parameter secara efisien. Dalam berbagai perbandingan ketat, *XGBoost* sering bersaing bahkan melampaui algoritma *deep learning* yang lebih berat. Hal ini dibuktikan dalam eksperimen pada platform *PhishNet* versi 1.0. Dengan seleksi fitur tingkat lanjut seperti Boruta, *XGBoost* menunjukkan performa komputasi yang sangat kompetitif bersama *Random Forest*, keduanya mencapai akurasi di atas 95,9% [16]. Namun, studi juga menekankan bahwa performa maksimal kedua algoritma ini sangat bergantung pada struktur dan kualitas data pelatihan [15]. Persaingan antara *Random Forest* dan *XGBoost* yang tercatat di banyak literatur membuktikan bahwa membandingkan performa keduanya tetap menjadi topik penelitian yang relevan, terutama di tengah perkembangan dataset penipuan siber.

Dari tinjauan literatur internasional terbaru tahun 2024-2026, dapat disimpulkan bahwa seleksi fitur leksikal pada URL sudah menjadi standar metodologis, dan persaingan antara *Random Forest* dan *XGBoost* terus mendominasi klasifikasi siber. Namun, sebagian besar studi masih menggunakan dataset *Phishing* standar tanpa penyaringan khusus. Belum banyak penelitian yang secara khusus membandingkan *Random Forest* dan *XGBoost* dengan dataset yang didominasi teknik penyembunyian dan obfuskasi struktural. Selain itu, teknik penyelarasan dataset (*dataset pairing*) untuk menyeimbangkan rasio dan panjang

URL antar kelas sangat direkomendasikan, tetapi masih jarang diterapkan pada fitur leksikal berobfuskasi [17]. Celah penelitian ini menjadi dasar eksperimen dalam skripsi ini. Dengan merekayasa dan mengisolasi atribut manipulatif seperti titik pada *Subdomain*, kedalaman *path*, dan alamat IP tersembunyi, skripsi ini memastikan model tidak hanya belajar secara dangkal, tetapi benar-benar mengidentifikasi obfuskasi leksikal. Pengukuran kinerja model akan dilakukan dengan standar evaluasi algoritma yang telah divalidasi, termasuk Akurasi, Presisi, *Recall*, *F1-Score*, dan *Confusion Matrix* [18]. Pendekatan ini diharapkan dapat memberikan kesimpulan ilmiah yang jelas tentang algoritma ansambel mana, antara *Random Forest* dan *XGBoost*, yang paling tangguh dan konsisten untuk mendeteksi teknik *Phishing* dan obfuskasi terbaru.

B. Landasan Teori

2.1. *Phishing*

Phishing merupakan salah satu bentuk ancaman keamanan siber yang menggunakan teknik rekayasa sosial (*social engineering*) untuk mengelabui target agar memberikan informasi sensitif. Informasi tersebut dapat berupa kredensial *login* (nama pengguna dan kata sandi), data kartu kredit, hingga informasi pribadi lainnya yang bersifat rahasia[19].

2.1.1 Definisi dan Mekanisme Kerja

Phishing (password harvesting fishing) adalah kata yang berasal dari bahasa Inggris yaitu *fishing* yang berarti memancing. *Phishing* merupakan penipuan yang dilakukan dengan cara mengelabui target sehingga pelaku bisa mendapatkan data sensitif dan bersifat rahasia.[20].

Mekanisme bagaimana cara *Phishing* bekerja umumnya melibatkan beberapa tahapan berikut:

1. Perencanaan: Penyerang menentukan target dan membuat materi tiruan (situs web atau *email*) yang sangat mirip dengan aslinya.
2. Pengiriman: Umpan dikirimkan melalui berbagai media, paling umum adalah melalui *email*, pesan instan, atau SMS (*smishing*).
3. Eksekusi: Korban yang terpedaya akan mengklik tautan (*URL*) berbahaya yang mengarah ke situs palsu.

4. Pengumpulan Data: Korban memasukkan data sensitif pada situs palsu tersebut, yang kemudian direkam dan disimpan oleh penyerang.

2.1.2 Jenis-Jenis *Phishing*

Berdasarkan target dan metodenya, *Phishing* dapat dikategorikan menjadi beberapa jenis[21]:

1. Email *Phishing*, Jenis *Phishing* ini akan menggunakan *email* - *email* palsu untuk mengelabui korbannya dengan email yang dikirim secara acak kepada calon korban. Pelaku akan membuat *email* dengan format menyerupai *email* asli dan resmi sehingga korban percaya kalau itu berasal dari forum resmi.
2. Web *Phishing*, biasanya menggunakan media *website* palsu untuk menjerat calon korban. Di mana, pelaku akan membuat *email* dengan nama *Domain* yang seolah-olah resmi atau juga disebut dengan *Domain spoofing*.
3. *Spear Phishing*: merupakan bentuk lain dari *email Phishing* dengan pelaku yang sudah memiliki data pribadi korban berupa nama dan alamat, nantinya akan berusaha mendapatkan informasi pribadi lanjutan dari korban
4. *Whaling*: merupakan kejahatan *Phishing* dengan menargetkan orang-orang dalam kewenangan besarseperti manajer personalia, penanggung jawab suatu kegiatan, bahkan pemilik bisnis. Data korban dengan kewenangan besar tentunya akan menarik korban seperti perekrutan karyawan palsu.

2.1.3 *Phishing* Berbasis URL

Fokus utama dalam penelitian ini adalah pada URL *Phishing*. Pada jenis ini, penyerang memanfaatkan ketidakmampuan manusia dalam membedakan struktur URL yang asli dengan yang dimanipulasi. Penyerang sering kali menggunakan teknik obfuskasi untuk menyembunyikan alamat tujuan yang sebenarnya, seperti pemanfaatan *Subdomain* yang mengecoh, penyisipan karakter khusus, modifikasi parameter, atau penggunaan path direktori yang panjang guna mengelabui pengguna sekaligus menghindari sistem pendeteksi keamanan (*security filters*).

2.2 Fitur URL (*Uniform Resource Locator*)

URL adalah rangkaian karakter menurut format standar tertentu, yang digunakan untuk menentukan alamat suatu sumber daya (seperti dokumen atau gambar) di internet. pengetesan URL merupakan salah satu cara untuk mencari ciri dari suatu URL *Phishing*, dan juga sebagai penentuan hasil klasifikasi [22].

2.2.1 Struktur Anatomi URL

Untuk mengekstraksi fitur, terlebih dahulu harus dipahami struktur dasar sebuah URL yang terdiri dari beberapa komponen utama[23]:

1. Protokol (*Scheme*): Menunjukkan metode akses yang digunakan, seperti http:// atau https://.
2. *Subdomain*: Bagian tambahan sebelum *Domain* utama (misal: m. pada *m.facebook.com*).
3. Nama *Domain*(SLD): Nama spesifik yang mewakili identitas pemilik atau nama entitas dari sebuah situs web (misal: *google* pada *google.com*)
4. Top Level *Domain* (TLD): Akhiran atau ekstensi yang terletak di bagian paling kanan dari sebuah nama *Domain*, yang menunjukkan kategori atau wilayah asal *Domain* tersebut (misal: *.com, .id*, atau *.org*).
5. *Path*: Menunjukkan lokasi spesifik file atau direktori di dalam server.
6. *Query String*: Parameter tambahan yang sering diawali dengan tanda tanya(?).

2.2.2 Kategori Fitur URL untuk Deteksi *Phishing*

Fitur-fitur yang diekstraksi dari URL umumnya dikelompokkan ke dalam beberapa kategori berdasarkan karakteristiknya[24]:

1. Struktur dan Panjang URL: URL yang memiliki panjang berlebihan atau mengandung terlalu banyak karakter acak cenderung digunakan untuk menyembunyikan *Domain* sebenarnya. Sistem akan menghitung panjang total URL & dibandingkan dengan batas rata-rata situs normal.
2. Protokol yang Digunakan: Penggunaan protokol HTTP tanpa enkripsi dianggap berisiko tinggi, sementara protokol HTTPS menunjukkan

tingkat keamanan yang lebih baik. Sistem akan menandai URL tanpa HTTPS sebagai potensi ancaman.

3. Jumlah dan Posisi *Subdomain*: Situs *Phishing* sering kali memanfaatkan banyak *Subdomain* untuk meniru *Domain* resmi. Sistem menganalisis struktur *Domain* dan menghitung jumlah *Subdomain* untuk mendeteksi indikasi tersebut.
4. Penggunaan Karakter Spesial dan Numerik: Fitur ini menilai seberapa banyak karakter seperti "-", "_", "%", atau angka digunakan dalam URL. URL berbahaya umumnya mengandung karakter seperti itu untuk mengelabui pengguna.
5. Penggunaan Kata Kunci Mencurigakan Sistem mendeteksi keberadaan kata kunci seperti "*login*", "*secure*", "*verify*", atau "*update*", yang sering digunakan oleh pelaku *Phishing* kepercayaan korban.
6. Port Non-Standar dan Redireksi menarik Beberapa situs berbahaya menggunakan port yang tidak umum atau teknik pengalihan (*redirect*) untuk menyembunyikan aktivitas berbahaya di balik server lain.

2.2.3 Peran Fitur dalam Model Prediksi

Fitur-fitur di atas berfungsi sebagai variabel independen yang akan diubah menjadi representasi numerik (vektor). Semakin banyak dan relevan fitur yang diekstraksi, semakin mampu algoritma seperti *Random Forest* atau *XGBoost* dalam mengenali batas pemisah antara kelas *legitimate* (aman) dan *Phishing*.

2.3 *Random Forest*

Random Forest adalah algoritma pembelajaran mesin yang termasuk dalam kategori *ensemble learning* dengan menggunakan teknik *Bagging* (*Bootstrap Aggregating*). Metode *Random Forest* merupakan pengembagan dari Decision Tree dengan menggunakan *ensemble* untuk meningkatkan akurasi dan mengurangi resiko *Overfitting*[25].

2.3.1 Konsep Ensemble Learning dan Bagging

Ide utama di balik *Random Forest* adalah bahwa sekumpulan model yang bekerja bersama akan memberikan hasil yang lebih akurat dan stabil dibandingkan dengan satu model tunggal.

1. *Bootstrap*: Algoritma mengambil sampel acak dari dataset asli dengan pengembalian untuk membangun setiap pohon.
2. *Aggregating*: Hasil prediksi dari seluruh pohon dikumpulkan dan ditentukan melalui mekanisme *voting* terbanyak (untuk klasifikasi) atau rata-rata (untuk regresi).

2.3.2 Mekanisme Kerja *Random Forest*

Proses pembentukan algoritma *Random Forest* melibatkan beberapa langkah-langkah berikut:

1. Pemilihan Sampel: Dari total data yang tersedia, diambil beberapa sampel acak secara *bootstrap*.
2. Konstruksi Pohon: Setiap sampel digunakan untuk membangun satu *Decision Tree*. Pada setiap simpul (*node*) pohon, pemilihan fitur dilakukan secara acak dari total fitur yang tersedia (misal: hanya memilih beberapa fitur dari 30 fitur URL).
3. Prediksi: Ketika ada data URL baru, setiap pohon dalam "hutan" akan memberikan prediksi (*Phishing* atau Aman).
4. Majority Vote: Prediksi akhir ditentukan berdasarkan suara terbanyak dari seluruh pohon tersebut.

2.3.3 Keunggulan dalam Deteksi *Phishing*

Random Forest sering dipilih dalam penelitian deteksi *Phishing* karena beberapa alasan berikut:

1. Ketahanan terhadap *Overfitting*: Dengan menggabungkan banyak pohon, risiko model hanya "menghafal" data pelatihan (*Overfitting*) menjadi lebih kecil.
2. Menangani Data Dimensi Tinggi: Sangat efektif ketika digunakan pada dataset yang memiliki banyak fitur, seperti analisis URL yang melibatkan puluhan parameter.

2.3.3 Formulasi Matematis Algoritma *Random Forest*

Random Forest pertama kali diperkenalkan oleh Leo Breiman (2001). *Random Forest* merupakan salah satu metode yang dapat meningkatkan hasil akurasi dalam membangkitkan atribut untuk setiap *node* yang dilakukan secara acak. *Random Forest* terdiri dari sekumpulan *decision tree*, di mana kumpulan pohon keputusan ini digunakan untuk mengklasifikasi data ke suatu kelas[26].

Membentuk pohon keputusan pada metode *Random Forest* sama dengan proses pada *Classification and Regression Tree* (*CART*), hanya saja pada *Random Forest* tidak dilakukan *prunning* (pemangkasan). Indeks *Gini* digunakan untuk memilih fitur di setiap simpul internal dari pohon keputusan. Nilai Indeks Gini dapat dihitung sebagai berikut[26]:

$$Gini(Si) = 1 - \sum_{i=0}^{c-1} p_i^2 \quad (1)$$

Keterangan:

1. p_i = frekuensi relatif kelas C_i di dalam *set*.
2. C_i = kelas untuk $i, \ldots, c-1$ dan adalah c jumlah kelas yang telah ditentukan.

Kualitas *split* (pemecahan) pada fitur k ke dalam subset Si merupakan jumlah sampel milik kelas C_i, kemudian dihitung sebagai jumlah pertimbangan indikasi Gini dari subset yang dihasilkan. Data dapat dihitung dengan rumus perhitungan Gini *Split* sebagai berikut:

$$Gini_{Split} = \sum_{i=0}^{k-1} \left(\frac{n_i}{n}\right) Gini(\ S_i) \quad (2)$$

Keterangan:

1. n_i = jumlah sampel dalam subset setelah di-*split*.
2. n = jumlah sampel di *node* yang diberikan.

2.4 *XGBoost* (*Extreme Gradient Boosting*)

XGBoost adalah implementasi lanjutan dari algoritma *Gradient Boosting* yang dirancang khusus untuk efisiensi, fleksibilitas, dan portabilitas. Algoritma ini merupakan salah satu teknik paling populer dalam kompetisi *data science* karena kemampuannya dalam menghasilkan performa yang sangat tinggi pada data terstruktur, seperti fitur-fitur numerik yang diekstraksi dari URL.

2.4.1 Konsep *Gradient Boosting*

Berbeda dengan *Random Forest* yang membangun pohon secara sejajar (*parallel*), *XGBoost* membangun pohon secara berurutan (*sequential*).

1. Koreksi Kesalahan: Pohon kedua dibangun untuk memperbaiki kesalahan (residual) yang dihasilkan oleh pohon pertama. Proses ini berlanjut hingga jumlah pohon yang ditentukan tercapai atau tidak ada lagi peningkatan yang signifikan.
2. *Gradient Descent*: Algoritma ini menggunakan prinsip *gradient descent* untuk meminimalkan fungsi kerugian (*loss function*) saat menambahkan model baru.

2.4.2 Keunggulan Teknis *XGBoost*

XGBoost memiliki beberapa fitur teknis yang membuatnya disebut sebagai "*Extreme*":

1. Regularisasi (L1 dan L2): Memiliki sistem "penalti" untuk mencegah model menjadi terlalu kompleks, sehingga sangat efektif dalam menghindari *Overfitting*.
2. *Sparsity Awareness*: Mampu menangani data yang hilang (*missing values*) secara otomatis, yang sering terjadi pada dataset URL yang tidak lengkap.
3. *Parallel Processing*: Meskipun algoritmanya sekuensial, proses pencarian titik potong (*split*) pada fitur dilakukan secara paralel, sehingga waktu *training* jauh lebih cepat dibandingkan *Gradient Boosting* standar.

2.4.3 Formulasi Matematis Algoritma *XGBoost*

XGBoost adalah algoritma *ensemble tree* (ansambel pohon) yang sangat efisien dan populer, yang didasarkan pada kerangka *Gradient Boosting*

Decision Tree (GBDT). *XGBoost* bekerja dengan membangun pohon keputusan secara sekuensial (bertahap), di mana setiap pohon baru dilatih untuk mengoreksi kesalahan (*residual*) dari pohon-pohon sebelumnya.[27]

Keunggulan utama *XGBoost* adalah kemampuannya mencegah *Overfitting* melalui fungsi objektif yang ter-regularisasi. Fungsi objektif yang dioptimalkan oleh *XGBoost* didefinisikan pada persamaan berikut:

$$O = \sum_{i=1}^{n} L\left(y_i, F(x_i)\right) + \sum_{k=1}^{t} R(f_k) + C \quad (3)$$

Pada fungsi objektif tersebut, fungsi $L\left(y_i, F(x_i)\right)$ berperan sebagai *loss function* yang digunakan untuk mengukur tingkat kesalahan prediksi yang dihasilkan model pada setiap data. Sementara itu, $R(f_k)$ merupakan fungsi regularisasi yang berfungsi mengendalikan kompleksitas model sehingga dapat meminimalkan potensi terjadinya *Overfitting*.[27]

Untuk memperjelas definisi dari fungsi regularisasi, bentuk persamaannya ditunjukkan pada rumus berikut:

$$R(f_k) = \alpha H + \frac{1}{2} n \sum_{j=1}^{H} \omega_j^2 \quad (4)$$

Keterangan:

1. α = kompleksitas daun.
2. H = jumlah daun.
3. n = parameter penalti.
4. ω_j^2 = hasil *output* dari setiap simpul daun.
5. C = konstanta yang dapat dihilangkan secara efektif.

2.4.3 Penerapan dalam Klasifikasi *Phishing*

Dalam konteks skripsi ini, *XGBoost* digunakan sebagai pembanding utama bagi *Random Forest*. Keunggulan utamanya dalam deteksi *Phishing* meliputi:

1. Presisi Tinggi: Sangat handal dalam membedakan pola halus antara URL yang sangat mirip (misal: antara situs bank asli dan kloningannya).
2. Efisiensi Sumber Daya: Mampu memproses ribuan baris dataset URL dengan waktu komputasi yang minimal, sehingga cocok untuk implementasi sistem deteksi *real-time*.

2.5 *Machine Learning* (Pembelajaran Mesin)

Machine Learning adalah cabang dari Kecerdasan Buatan (*Artificial Intelligence*) yang berfokus pada pengembangan algoritma dan model statistik yang memungkinkan komputer untuk "belajar" dari data tanpa diprogram secara eksplisit untuk setiap tugas tertentu. Oleh karena itu, penerapan pendekatan *Machine Learning* mulai banyak diteliti, mengingat kemampuannya dalam mengenali pola data yang kompleks dan sulit ditangkap oleh metode tradisional. [28]

2.5.1 Paradigma *Supervised Learning*

Berdasarkan jenis data dan tujuan penelitian ini, pendekatan yang digunakan adalah *Supervised Learning* (Pembelajaran Terawasi).

1. Mekanisme: Model dilatih menggunakan *dataset* yang sudah memiliki label (misalnya: label '1' untuk *Phishing* dan '0' untuk *legitimate*).
2. Proses: Algoritma akan mempelajari pemetaan antara fitur input (X) dan target output (Y). Setelah fase pelatihan (*training*) selesai, model yang dihasilkan diharapkan mampu memprediksi label dari data URL baru yang belum pernah dilihat sebelumnya.

2.5.2 Klasifikasi (*Classification*)

Tugas utama dalam deteksi *Phishing* adalah Klasifikasi Biner. Klasifikasi adalah teknik *Machine Learning* untuk memprediksi kategori atau label diskrit. Dalam penelitian ini, model akan mengklasifikasikan setiap URL ke dalam salah satu dari dua kelas:

1. *Positive Class* (1): Website *Phishing*.
2. *Negative Class* (0): Website Aman (*Legitimate*).

2.5.3 Evaluasi Kinerja Model

Untuk mengukur seberapa efektif model dalam melakukan klasifikasi, digunakan metrik evaluasi yang umum dalam literatur ilmiah. Seluruh metrik evaluasi klasifikasi ini pada dasarnya dihitung menggunakan parameter yang dihasilkan dari *Confusion Matrix***.** *Confusion Matrix* adalah tabel yang digunakan untuk mengevaluasi kinerja model klasifikasi dengan membandingkan hasil prediksi model dengan nilai aktual (data sebenarnya). [29].

Di mana:

1. *True Positive* (*TP*): Data *Phishing* diprediksi *Phishing*.
2. *True Negative* (*TN*): Data *legitimate* diprediksi *legitimate*.
3. *False Positive (FP): Data legitimate diprediksi Phishing.*
4. *False Negative (FN):* Data *Phishing* diprediksi *legitimate*.

Berdasarkan nilai-nilai pada *Confusion Matrix* tersebut, metrik evaluasi yang digunakan dalam penelitian ini antara lain:

A. *Accuracy*: Persentase total prediksi yang benar (baik *Phishing* maupun aman). *Accuracy* mengukur persentase prediksi yang benar secara keseluruhan.

$$Accuracy = \frac{\text{TP+TN}}{TP+TN+FP+FN} \quad (5)$$

B. Precision: Seberapa akurat model saat memprediksi sebuah URL sebagai *Phishing*. *Precision* mengukur seberapa akurat prediksi positif yang dilakukan model.

$$Precision = \frac{\text{TP}}{TP+FP} \quad (6)$$

C. *Recall (Sensitivity)*: Kemampuan model dalam menemukan seluruh situs *Phishing* yang ada dalam data. *Recall* mengukur kemampuan model dalam menemukan semua data positif yang sebenarnya.

$$Recall = \frac{\text{TP}}{TP+FN} \quad (7)$$

D. *F1-Score*: Rata-rata harmonik antara *Precision* dan *Recall*, yang berguna jika terdapat ketidakseimbangan data (*data imbalance*). *F1-Score* adalah rata-rata harmonik antara *Precision* dan *Recall*, berguna untuk menyeimbangkan kedua metrik tersebut.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (8)$$

2.6 Obfuskasi URL *Phishing*

Dalam pengembangan model *Machine Learning*, tantangan utama bukan hanya klasifikasi data standar, melainkan kemampuan model dalam mengenali data yang telah dimanipulasi. Obfuskasi URL didefinisikan sebagai teknik penyamaran struktur alamat situs web yang dilakukan secara sengaja oleh penyerang untuk menyembunyikan identitas asli situs *Phishing*. Tujuan utama dari teknik ini adalah untuk menurunkan sensitivitas sistem deteksi otomatis dan mengelabui kewaspadaan pengguna manusia. Penyerang menggunakan teknik penipuan yang canggih untuk membuat URL yang tidak sah tampak sah dengan menggunakan pemalsuan (*spoofing*) dan penyamaran (*obfuscation*) URL, dan sebagainya, guna menghindari deteksi dan menyerang pengguna yang tidak waspada.[30]

2.6.1 Konsep dan Mekanisme Obfuskasi

Obfuskasi bekerja dengan cara mengubah karakteristik leksikal, struktural, atau statistik dari sebuah URL tanpa menghilangkan fungsionalitas akses ke server tujuan. Dalam konteks keamanan siber, pemahaman terhadap teknik obfuskasi sangat krusial karena penyerang akan selalu mencari celah untuk membuat URL berbahaya terlihat semirip mungkin dengan data normal (*legitimate*).

2.6.2 Teknik-Teknik Obfuskasi pada URL

Beberapa bentuk manipulasi fitur yang sering digunakan oleh penyerang dalam teknik obfuskasi meliputi:

1. Obfuskasi Leksikal: Menyamarkan nama *Domain* asli dengan menambahkan banyak *Subdomain*, atau menggunakan karakter yang membingungkan.
2. Manipulasi Statistik: Mengubah distribusi karakter dalam URL, seperti penggunaan angka atau simbol tertentu secara berlebihan,

untuk memanipulasi nilai entropi agar terlihat seperti teks yang dibuat oleh manusia atau sebaliknya.

3. *Structural Mimicry*: Menyisipkan kata-kata kunci tepercaya seperti "*secure*", "*login*", atau "*official-bank*" ke dalam bagian *path* atau *query* URL guna mengelabui fitur deteksi berbasis kata kunci.

2.6.3 Pentingnya Klasifikasi pada Data Obfuskasi

Penelitian yang hanya berfokus pada akurasi standar sering kali menghasilkan model yang mengalami *Overfitting* pada pola tertentu dan gagal mengenali pola penyamaran baru. Dengan memfokuskan klasifikasi pada URL yang diobfuskasi, penelitian ini dapat:

1. Mengidentifikasi Titik Lemah: Mengetahui fitur URL mana yang paling rentan dikelabui oleh teknik penyamaran tertentu.
2. Menjamin Realibilitas Sistem: Memberikan rekomendasi algoritma mana (antara *Random Forest* atau *XGBoost*) yang lebih efektif dan konsisten dalam mengenali ancaman di lingkungan dunia nyata yang dinamis.