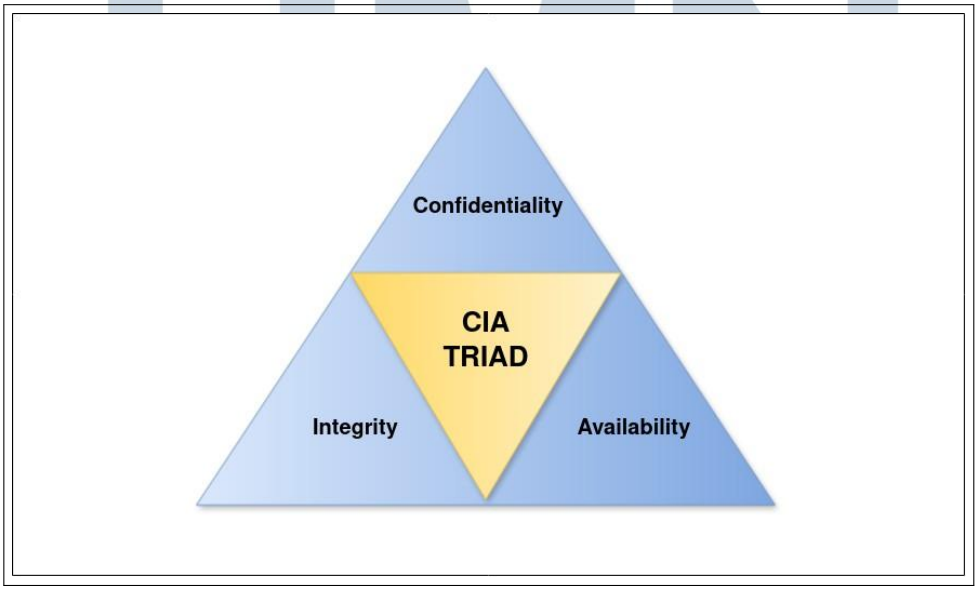
BAB 2
LANDASAN TEORI

2.1 Cyber Security dan Phishing

Cyber security adalah praktik untuk melindungi komputer, server, perangkat seluler, sistem elektronik, jaringan, dan data dari serangan berbahaya, istilah ini juga dikenal sebagai keamanan teknologi informasi [26]. Penelitian lain menyebut bahwa *cyber security* adalah kumpulan teknologi dan proses yang dirancang untuk melindungi komputer, jaringan, program, dan data dari serangan, kerusakan, atau akses yang tidak sah [27]. Dalam beberapa waktu terakhir, keamanan siber mengalami pergeseran besar-besaran dalam teknologi dan operasinya dalam konteks komputasi, dan *data science* menjadi pendorong perubahan tersebut, di mana *machine learning*, bagian inti dari *articial inteligence* (AI), dapat memainkan peran vital dalam mengungkap wawasan dari data [28].

2.1.1 CIA Triad

Dalam *cybersecurity*, *CIA Triad* merupakan pedoman dan panduan untuk membantu individu atau organisasi tertentu dalam membentuk sebuah sistem keamanan pada aplikasi, website, dan sebagainya.



Gambar 2.1. CIA *Triad*

Sumber: [29]

Pada gambar Gambar 2.1, CIA *Triad* sendiri terdiri dari 3 sifat yang penting yaitu [30][29]:

a) *Confidentiality*

Merupakan prinsip dasar dalam keamanan sistem yang bertujuan memastikan bahwa data hanya dapat diakses oleh pihak yang berwenang. Baik individu maupun organisasi memiliki tanggung jawab untuk menjaga agar informasi tetap bersifat pribadi dan tidak disalahgunakan. Untuk mencapai hal tersebut, akses terhadap informasi seperti *login credential* harus dikendalikan secara ketat guna mencegah kebocoran data, baik yang terjadi secara sengaja maupun tidak disengaja. Pembatasan akses dilakukan agar hanya pihak yang memiliki otorisasi yang dapat melihat atau menggunakan data tertentu. Selain itu, data juga dapat diklasifikasikan berdasarkan tingkat risiko kerugian apabila informasi tersebut jatuh ke pihak yang tidak berhak, sehingga langkah perlindungan yang diterapkan dapat disesuaikan dengan tingkat sensitivitas data tersebut.

b) *Integrity*

Merupakan aspek keamanan yang berfokus pada upaya menjaga konsistensi, keakuratan, dan keandalan data dalam suatu sistem atau aplikasi. Suatu sistem dianggap memiliki integritas yang baik apabila data yang tersimpan tetap asli, akurat, dan tidak mengalami perubahan yang tidak sah. Dalam praktiknya, organisasi perlu memastikan kualitas data agar tidak terjadi berbagai permasalahan seperti data *corrupted*, *inconsistency*, *redundancy*, maupun *isolation*. Kondisi-kondisi tersebut dapat mengurangi keandalan data serta berpotensi menimbulkan kerugian bagi organisasi. Oleh karena itu, diperlukan penerapan *restrictions* atau pembatasan akses agar perubahan data hanya dapat dilakukan oleh pihak yang memiliki kewenangan. Selain itu, mekanisme *backup* juga perlu disediakan untuk memulihkan data apabila terjadi kerusakan atau kehilangan sehingga kondisi data dapat dikembalikan seperti semula.

c) *Availability*

Aspek penting lainnya dalam keamanan informasi adalah *availability*. Prinsip ini menekankan bahwa sistem, aplikasi, jaringan, serta data harus selalu tersedia dan dapat diakses ketika dibutuhkan oleh pengguna yang berwenang. Suatu sistem yang aman tidak hanya melindungi kerahasiaan dan keakuratan

data, tetapi juga memastikan bahwa informasi tersebut dapat digunakan tanpa hambatan. Oleh karena itu, organisasi perlu memastikan bahwa layanan tetap berjalan dengan baik meskipun terjadi gangguan seperti pemadaman listrik atau gangguan sistem lainnya. Selain itu, akses terhadap data juga harus dapat dilakukan secara efisien tanpa waktu tunggu yang terlalu lama agar operasional sistem tetap berjalan secara optimal.

Kejahatan siber pada dasarnya adalah kejahatan yang menggunakan internet atau ruang siber sebagai media untuk melakukan tindakan penipuan yang direncanakan. Penjahat siber menargetkan pengguna internet demi keuntungan finansial atau jenis penipuan lainnya. *Phishing* merupakan salah satu kejahatan siber utama yang cukup marak saat ini. *Phishing* adalah jenis kejahatan siber di mana penyerang menyamar sebagai entitas tepercaya. Penyerang berusaha memikat korban dengan menawarkan godaan yang mudah membuat korban terjebak. Penyerang biasanya mencuri data sensitif korban, seperti detail kartu kredit atau kredensial *login* [31]. Hal ini terjadi ketika korban mengklik tautan yang dikirim oleh penyerang yang menyamar sebagai entitas asli, atau ketika pengguna mengungkapkan data kepada penyerang melalui panggilan telepon. Dengan memikat korban dan menawarkan keuntungan palsu, penyerang memperoleh detail pengguna dan menggunakannya secara jahat terhadap korban.

2.2 Karakteristik URL Phishing

URL *phishing* memiliki sejumlah karakteristik tertentu yang dapat digunakan sebagai indikator untuk mengidentifikasi apakah suatu URL berpotensi sebagai *phishing* yang dapat dilihat pada tabel 2.1 dibawah ini.

Pada penelitian S.Shabudin et.al.[33], terlihat tabel 2.1 yang menunjukkan karakteristik situs web *phishing* bahwa terdapat berbagai indikator yang dapat digunakan untuk mengidentifikasi apakah suatu situs web bersifat mencurigakan (*phishing*) atau tidak (*legitimate*). Indikator tersebut dapat dianalisis melalui beberapa aspek, yaitu fitur pada *address bar*, abnormalitas pada struktur tautan, dan penggunaan elemen *Javascript*, serta informasi terkait domain. Dengan menganalisis berbagai karakteristik tersebut, sistem deteksi berbasis *machine learning* dapat memanfaatkan fitur-fitur tersebut sebagai variabel dalam proses klasifikasi untuk membedakan antara situs web *phishing* dan situs web yang sah.

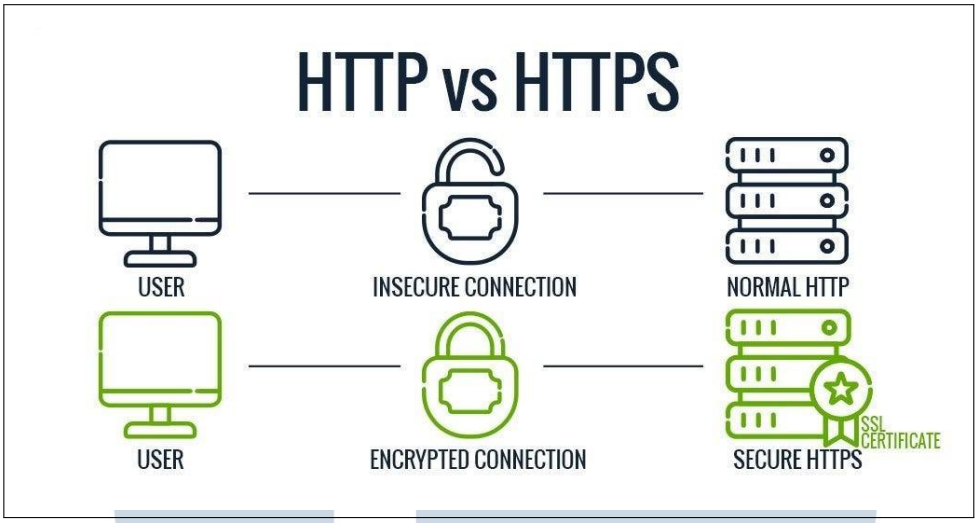
Tabel 2.1. Karakteristik Phishing Website

Sumber: [32]

Fitur	**Sub-Fitur**
Fitur berdasarkan Address Bar	Terdapat IP Address pada nama domain
	Terdapat URL yang panjang untuk menutupi bagian yang mencurigakan pada nama domain
	Terdapat URL shortening services (TinyURL)
	Redirection menggunakan "//"
	Menambahkan prefiks atau sufiks yang dipisahkan dengan simbol "-" pada domain
	Terdapat simbol "@" pada nama domain
	Memiliki multi sub-domain
	Terdapat non-standar port
	URL yang terlalu panjang
	Menggunakan favicon
	Terdapat token http/https pada nama domain
Fitur berdasarkan abnormalitas	Request URL
	URL of Anchor
	Tautan pada tag <Meta>, <Script>, dan <Link>
	Server Form Handler (SFH)
	Input informasi ke Email
	Abnormal URL
Fitur berdasarkan Javascript	Website Forwarding
	Kustomisasi Status Bar
	Menonaktifkan Right Click
	Terdapat pop-up window
	Iframe Redirection
Fitur berdasarkan domain	Umur dari domain
	DNS Record

2.2.1 HTTP dan HTTPS

Pada gambar 2.2 dibawah, disajikan perbedaan antara HTTP (*Hypertext Transfer Protocol*) dan HTTPS (*Hypertext Transfer Protocol Secure*).



Gambar 2.2. Perbedaan HTTP/HTTPS
Sumber: [34]

HTTP (*Hypertext Transfer Protocol*) adalah protokol pada lapisan aplikasi (*application layer*) yang menjadi dasar dalam sistem informasi yang bersifat terdistribusi dan kolaboratif. Protokol ini memiliki sifat *generic* dan *stateless*, yang berarti tidak menyimpan sebuah informasi dari permintaan sebelumnya. Karakteristik *generic* memungkinkan HTTP digunakan tidak hanya untuk *hypertext*, tetapi juga untuk berbagai kebutuhan lain seperti *name servers* melalui mekanisme perluasan pada metode permintaan, kode kesalahan, serta *headers*. Salah satu fitur yang cukup penting dari protokol ini adalah proses *typing* dan *negotiation of data representation*, yang memungkinkan sistem dikembangkan secara mandiri dari jenis data yang sedang ditransfer. Sementara itu, HTTPS (Hypertext Transfer Protocol Secure) merupakan versi yang lebih aman dari HTTP karena menggunakan mekanisme enkripsi dalam proses komunikasi antara browser dan situs web. Pada HTTPS, data yang dikirimkan dilindungi melalui protokol keamanan seperti SSL (Secure Sockets Layer) atau TLS (Transport Layer Security). Dengan adanya enkripsi ini, informasi sensitif seperti data login maupun informasi pembayaran dapat terlindungi dari risiko penyadapan atau akses oleh pihak yang tidak berwenang [35].

2.3 Machine Learning

Machine *learning* adalah bidang interdisipliner yang mendasarkan pada teori statistik, ilmu komputer, dan bidang-bidang terkait. Peningkatan yang pesat

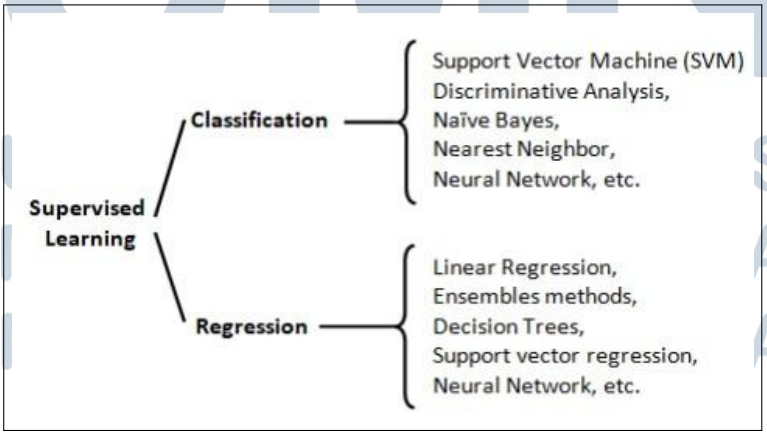
dalam beberapa tahun terakhir telah menempatkannya sebagai kekuatan utama dalam kemajuan teknologi. Sebelumnya dikenal dengan istilah-istilah seperti pembelajaran statistik atau pengenalan pola, *machine learning* kini mencakup berbagai metode yang telah diteliti secara mendalam, disempurnakan, dan dioptimalkan. Pada inti dari *machine learning* terdapat konsep pembelajaran mandiri, suatu proses yang memanfaatkan pemodelan statistik yang berurusan dengan penemuan hubungan antara variabel untuk memprediksi hasil [32]. Kemampuan belajar mandiri yang dicapai melalui pemodelan statistik dan wawasan berbasis data ini telah menjadi fitur kunci dalam *machine learning* [36].

2.3.1 Klasifikasi Machine Learning

Machine *learning* secara umum diklasifikasikan menjadi tiga jenis, yaitu *supervised learning*, *unsupervised learning*, dan *reinforcement learning* [37] yang dijabarkan pada poin-poin berikut ini:

a) *Supervised Learning*

Supervised learning merupakan jenis *machine learning* yang menggunakan dataset berlabel dalam proses pelatihan model. Setiap data masukan memiliki nilai keluaran atau label yang telah ditentukan, sehingga model dapat mempelajari hubungan antara fitur dan kelas target. Pendekatan ini umumnya digunakan untuk menyelesaikan permasalahan klasifikasi dan regresi. Klasifikasi digunakan ketika keluaran yang diprediksi berupa kategori atau label diskret, sedangkan regresi digunakan ketika keluaran berupa nilai numerik atau kontinu [37][38].

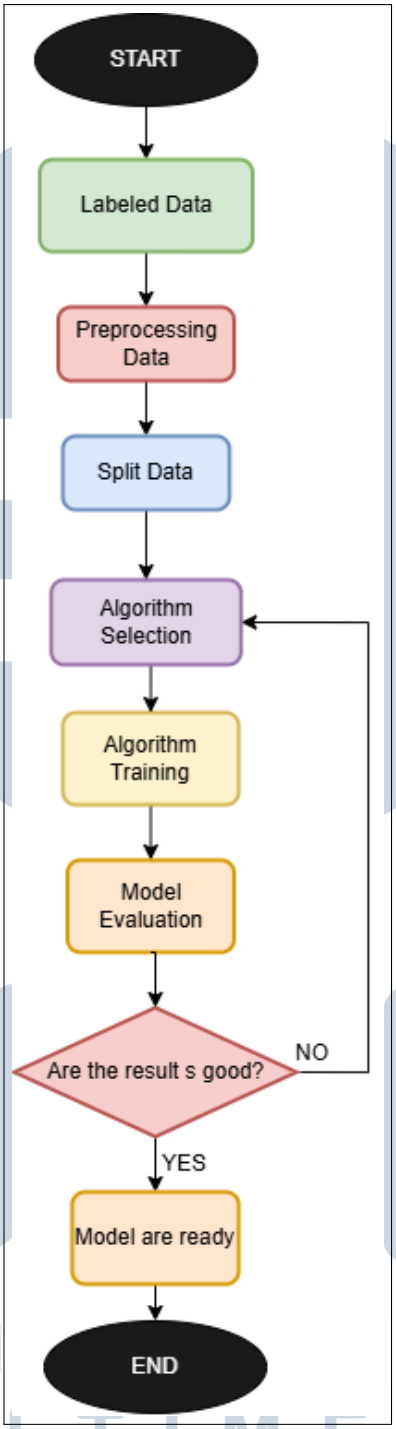


Gambar 2.3. Klasifikasi *supervised learning*

Referensi: [37]

Berdasarkan Gambar 2.3, *supervised learning* dapat dibedakan menjadi klasifikasi dan regresi berdasarkan jenis keluaran yang ingin diprediksi. Klasifikasi digunakan ketika keluaran berbentuk kategori, seperti menentukan apakah suatu URL termasuk kelas *phishing* atau *legitimate*. Sementara itu, regresi digunakan ketika keluaran yang diprediksi berupa nilai kontinu, seperti prediksi suhu atau harga.

Penelitian ini termasuk dalam permasalahan klasifikasi karena model digunakan untuk memprediksi dua kelas, yaitu URL *phishing* dan URL *legitimate*. Algoritma klasifikasi yang digunakan adalah *Decision Tree* dan *Random Forest*.



Gambar 2.4. Alur kerja *supervised learning*

Referensi: [38]

Berdasarkan Gambar 2.4, alur kerja *supervised learning* dimulai dengan mendefinisikan permasalahan, mengumpulkan data, melakukan praproses data, serta membagi dataset menjadi data latih dan data uji. Data latih

digunakan untuk membangun model, sedangkan data uji digunakan untuk mengevaluasi kemampuan model dalam memprediksi data yang belum pernah digunakan selama pelatihan [37].

Tahap berikutnya adalah memilih algoritma yang sesuai dengan karakteristik permasalahan. Model kemudian dilatih menggunakan data latih dan dapat melalui proses penyesuaian parameter untuk memperoleh konfigurasi yang optimal. Setelah model terbentuk, evaluasi dilakukan menggunakan data uji dengan metrik yang sesuai, seperti *accuracy*, *precision*, *recall*, dan *F1-score*.

Proses tersebut dapat dilakukan secara iteratif. Apabila hasil evaluasi belum memenuhi kriteria yang ditentukan, perbaikan dapat dilakukan pada tahap praproses, pemilihan fitur, pemilihan algoritma, maupun penyesuaian parameter. Siklus tersebut dilakukan hingga diperoleh model klasifikasi yang memiliki performa sesuai dengan kebutuhan penelitian [37][39].

b) *Unsupervised Learning*

Unsupervised learning merupakan jenis *machine learning* yang menggunakan data tanpa label. Model berusaha menemukan pola, kemiripan, atau struktur tersembunyi dalam data tanpa diberikan kelas keluaran yang telah ditentukan. Pendekatan ini umumnya digunakan untuk pengelompokan data atau *clustering* dan pengurangan dimensi. Contoh algoritma *unsupervised learning* adalah *K-Means*, *Hierarchical Clustering*, dan DBSCAN [40]. Pendekatan ini tidak digunakan dalam penelitian karena dataset PhiUSIIL telah memiliki label kelas *phishing* dan *legitimate*.

c) *Reinforcement Learning*

Reinforcement learning merupakan pendekatan pembelajaran yang melibatkan interaksi antara agen dan lingkungan. Agen memilih suatu tindakan berdasarkan kondisi lingkungan, kemudian memperoleh *reward* atau *penalty* sebagai umpan balik. Proses tersebut dilakukan secara berulang agar agen dapat mempelajari tindakan yang menghasilkan total *reward* paling tinggi dalam jangka panjang [41]. Pendekatan ini tidak digunakan dalam penelitian karena proses deteksi *phishing* dilakukan sebagai permasalahan klasifikasi menggunakan dataset berlabel, bukan melalui interaksi antara agen dan lingkungan.

2.4 CRISP-DM

Cross-Industry Standard Process for Data Mining atau CRISP-DM merupakan model proses yang digunakan untuk mengarahkan pelaksanaan proyek *data mining* secara sistematis. CRISP-DM terdiri atas enam tahapan, yaitu *Business Understanding*, *Data Understanding*, *Data Preparation*, *Modeling*, *Evaluation*, dan *Deployment*. Keenam tahapan tersebut tidak harus dilakukan secara linear karena hasil pada suatu tahap dapat menyebabkan proses kembali ke tahap sebelumnya untuk melakukan perbaikan atau penyesuaian [42, 43].

Dalam konteks penelitian akademik, tahap *Business Understanding* dapat disesuaikan menjadi pemahaman terhadap masalah dan tujuan penelitian. Adapun tahapan CRISP-DM yang diterapkan dalam penelitian ini dijelaskan sebagai berikut.

a. *Business Understanding*

Tahap *Business Understanding* berfokus pada pemahaman masalah, tujuan, kebutuhan, dan kriteria keberhasilan proyek. Permasalahan yang telah diidentifikasi kemudian diterjemahkan menjadi tujuan *data mining* yang dapat diselesaikan menggunakan data dan teknik pemodelan tertentu. Pada tahap ini juga ditentukan jenis permasalahan, teknik analisis yang dibutuhkan, serta ukuran keberhasilan model [42].

Dalam penelitian ini, masalah yang diangkat adalah kebutuhan untuk mendeteksi URL *phishing* berdasarkan karakteristik URL. Tujuan penelitian ditetapkan untuk membandingkan performa algoritma *Decision Tree* dan *Random Forest*, mengidentifikasi fitur yang berpengaruh terhadap klasifikasi, serta menganalisis kasus *false negative*. Kriteria keberhasilan model diukur menggunakan *accuracy*, *precision*, *recall*, *F1-Score*, *ROC-AUC*, *False Negative Rate*, dan *specificity*.

b. *Data Understanding*

Tahap *Data Understanding* dilakukan untuk memperoleh pemahaman awal mengenai data yang digunakan. Aktivitas pada tahap ini mencakup pengumpulan data, pemeriksaan struktur dan tipe data, analisis distribusi atribut dan kelas, identifikasi nilai hilang, pemeriksaan data duplikat, serta evaluasi awal terhadap kualitas data. Statistik deskriptif dan visualisasi dapat digunakan untuk membantu memahami karakteristik data [42].

Pada penelitian ini, data yang digunakan adalah *dataset* PhiUSIIL *Phishing URL* yang terdiri atas 235.795 sampel URL. Proses pemahaman data dilakukan melalui pemeriksaan jumlah baris dan kolom, tipe data, distribusi label *phishing* dan *legitimate*, keberadaan *missing value*, data duplikat, dan karakteristik awal setiap fitur. Label 0 merepresentasikan kelas *phishing*, sedangkan label 1 merepresentasikan kelas *legitimate*.

c. *Data Preparation*

Tahap *Data Preparation* bertujuan menghasilkan data yang siap digunakan pada proses pemodelan. Tahap ini dapat mencakup pemilihan data, pembersihan data, penghapusan atribut yang tidak relevan, penanganan nilai hilang, transformasi data, pembentukan atribut, dan seleksi fitur. Teknik persiapan data dapat berbeda bergantung pada tujuan penelitian, karakteristik data, dan model yang digunakan [42].

Dalam penelitian ini, atribut yang bersifat *identifier*, bergantung pada konten halaman, berada di luar ruang lingkup penelitian, atau berpotensi menjadi *label proxy* dihapus dari data pemodelan. Setelah penyaringan awal, jumlah fitur kandidat berkurang menjadi 24 fitur. Data duplikat berdasarkan kombinasi fitur pemodelan dan label kemudian dihapus sehingga tersisa 233.845 data.

Data selanjutnya dibagi menjadi data latih dan data uji dengan rasio 80:20 menggunakan *Stratified Train-Test Split*. Metode *stratified* digunakan untuk mempertahankan proporsi kelas pada kedua subset. Tahapan *SimpleImputer* dan *SelectFromModel* berbasis *Random Forest* ditempatkan di dalam *Pipeline*, sehingga proses imputasi dan seleksi fitur hanya menggunakan data latih pada setiap iterasi validasi silang. Proses seleksi fitur menghasilkan 12 fitur terpilih dari 24 fitur kandidat.

d. *Modeling*

Tahap *Modeling* mencakup pemilihan algoritma, penyusunan skema pengujian, pelatihan model, penentuan parameter, dan pemilihan model terbaik. Pemilihan algoritma harus disesuaikan dengan tujuan analisis dan karakteristik data. Pada tahap ini, beberapa algoritma dapat dibandingkan untuk mengetahui model yang memberikan performa terbaik [42].

Penelitian ini menggunakan algoritma *Decision Tree* sebagai model *baseline* dan *Random Forest* sebagai model berbasis *ensemble learning*. Tahapan

SimpleImputer, *SelectFromModel*, dan algoritma klasifikasi diintegrasikan dalam satu *Pipeline*. Optimasi *hyperparameter* dilakukan menggunakan *GridSearchCV* dengan skema *10-fold Stratified Cross-Validation* dan metrik *f1_weighted* sebagai fungsi evaluasi utama.

e. *Evaluation*

Tahap *Evaluation* bertujuan menilai apakah model yang dihasilkan telah memenuhi tujuan dan kriteria keberhasilan yang ditentukan pada tahap awal. Evaluasi tidak hanya dilakukan terhadap nilai metrik, tetapi juga terhadap proses pemodelan, kesesuaian hasil dengan tujuan penelitian, keterbatasan model, dan kemungkinan terjadinya kesalahan selama pengolahan data. Visualisasi seperti *confusion matrix* juga dapat digunakan untuk memperjelas hasil evaluasi [42].

Dalam penelitian ini, model dievaluasi menggunakan *confusion matrix*, *accuracy*, *precision*, *recall*, *F1-Score*, *ROC-AUC*, *False Negative Rate*, dan *specificity*. Evaluasi juga mencakup perbandingan performa *Decision Tree* dan *Random Forest*, analisis *feature importance*, analisis kasus *false negative*, serta pemeriksaan selisih antara *train score* dan *cross-validation score*. Pemeriksaan tambahan terhadap fitur berpotensi menjadi *label proxy* dan irisan data latih dan data uji dilakukan untuk meminimalkan risiko *data leakage*.

f. *Deployment*

Tahap *Deployment* merupakan tahap penggunaan atau penyajian hasil model agar dapat dimanfaatkan oleh pengguna. Bentuk *deployment* tidak selalu berupa sistem produksi, tetapi dapat berupa laporan penelitian, rekomendasi, komponen perangkat lunak, atau *prototype* aplikasi. Tahap ini juga dapat mencakup perencanaan pemantauan dan pemeliharaan model [42]. CRISP-DM bersifat fleksibel sehingga tahapan dan hasil akhirnya dapat disesuaikan dengan kebutuhan proyek atau bidang penerapannya [43].

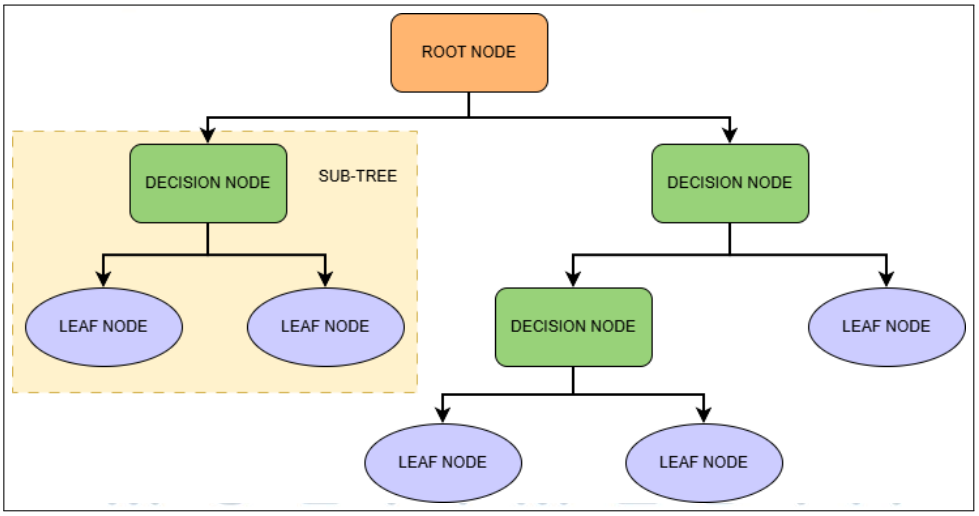
Dalam penelitian ini, model *Random Forest* terbaik diimplementasikan ke dalam *prototype* berbasis *Streamlit*. *Prototype* digunakan untuk mendemonstrasikan proses prediksi terhadap data uji dengan menampilkan hasil prediksi, label asli, probabilitas kelas, status kebenaran prediksi, dan nilai fitur yang digunakan oleh model. Implementasi tersebut tidak diklaim

sebagai sistem deteksi URL *phishing* secara *real-time*, melainkan sebagai demonstrasi penggunaan model terbaik yang dihasilkan dalam penelitian.

Penerapan CRISP-DM pada penelitian ini memberikan alur yang terstruktur mulai dari identifikasi masalah hingga implementasi *prototype*. Meskipun terdiri atas enam tahap utama, CRISP-DM tetap memungkinkan proses iterasi apabila ditemukan permasalahan pada data, model, atau hasil evaluasi. Fleksibilitas tersebut menjadikan CRISP-DM sesuai digunakan dalam proyek *machine learning* yang memerlukan proses pengolahan, pemodelan, evaluasi, dan perbaikan secara berulang [42, 43].

2.5 Algoritma Decision Tree

Decision tree merupakan algoritma yang memiliki struktur berbentuk *tree* dan menyerupai diagram alur, di mana setiap simpul merepresentasikan pengujian terhadap suatu atribut, setiap cabang menunjukkan hasil dari pengujian tersebut, dan setiap daun pada *tree* menggambarkan distribusi kelas tertentu. Model *decision tree* juga dapat dengan mudah diubah menjadi seperangkat aturan klasifikasi yang lebih sederhana untuk dipahami [44] [45]. Pada gambar 2.5 dibawah, ditunjukkan proses dari algoritma *decision tree*.



Gambar 2.5. Proses algoritma *decision tree*

Referensi: [46]

Struktur sebuah *Decision Tree* diawali oleh *Root Node*, yaitu titik awal yang mewakili seluruh kumpulan data dan menentukan fitur paling penting untuk

pembagian awal. Dari titik ini, data akan mengalir ke *Decision Node*, di mana terjadi proses pengujian atau pengecekan nilai atribut tertentu yang memecah data menjadi beberapa jalur baru. Keseluruhan bagian dari percabangan tersebut dapat dikategorikan sebagai *Sub-Tree*, yang merupakan struktur textittree dalam skala lebih kecil. Proses pembagian ini terus berlanjut hingga mencapai *Leaf Node*, yaitu simpul akhir yang tidak dapat dibagi lagi karena telah menghasilkan keputusan, klasifikasi, atau hasil akhir dari seluruh proses analisis tersebut [47][48].

2.5.1 Entropy

Pada algoritma *Decision Tree*, proses pemilihan atribut terbaik untuk membangun struktur pohon keputusan didasarkan pada ukuran tingkat ketidakpastian data. Salah satu ukuran yang umum digunakan adalah entropy, yaitu ukuran yang menggambarkan tingkat ketidakteraturan atau *impurity* dalam suatu dataset. Nilai *entropy* digunakan untuk mengetahui seberapa homogen distribusi kelas dalam suatu kumpulan data. Semakin kecil nilai *entropy*, maka data tersebut semakin homogen dan lebih mudah untuk diklasifikasikan. Rumus entropy dapat dinyatakan sebagai berikut [44] [48]:

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2 p_i \quad (2.1)$$

Keterangan:
A = dataset
c = jumlah kelas
Pi= probabilitas atau proporsi data pada kelas ke-i
Pada rumus 2.1, entropy bernilai 0 apabila seluruh data dalam suatu node berasal dari kelas yang sama, sedangkan nilai entropy akan semakin besar apabila distribusi kelas semakin tidak seimbang.

2.5.2 Information Gain

Setelah nilai *entropy* dari dataset diketahui, langkah selanjutnya dalam pembentukan *Decision Tree* adalah menentukan atribut terbaik yang akan digunakan sebagai node pemisah (*splitting node*). Proses ini dilakukan dengan menghitung *information gain*, yaitu ukuran yang menunjukkan seberapa besar pengurangan *entropy* yang diperoleh setelah data dibagi berdasarkan suatu atribut tertentu. Atribut dengan nilai information *gain* tertinggi dipilih sebagai *node* dalam

pohon keputusan karena dianggap paling efektif dalam memisahkan kelas data. Rumus *information gain* dinyatakan sebagai berikut [44]:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \quad (2.2)$$

Keterangan:
S = dataset awal
A = atribut yang diuji
Sv = subset data dari atribut A dengan nilai v
$|S_v|$ = jumlah data pada subset
$|S|$ = jumlah seluruh data
Berdasarkan rumus 2.2 diatas, nilai *information gain* yang lebih besar menunjukkan bahwa atribut tersebut lebih baik dalam membagi data menjadi kelas yang lebih homogen.

2.5.3 Gini Index

Selain menggunakan *entropy* dan *information gain*, beberapa implementasi *Decision Tree* juga menggunakan ukuran lain yang disebut *Gini Index*. *Gini Index* merupakan ukuran *impurity* yang digunakan untuk menentukan seberapa sering suatu elemen dalam dataset salah diklasifikasikan apabila dipilih secara acak berdasarkan distribusi label dalam dataset tersebut. Nilai *Gini* yang lebih kecil menunjukkan bahwa data dalam suatu node lebih homogen. Rumus *Gini Index* dapat dituliskan seperti 2.3 [44]:

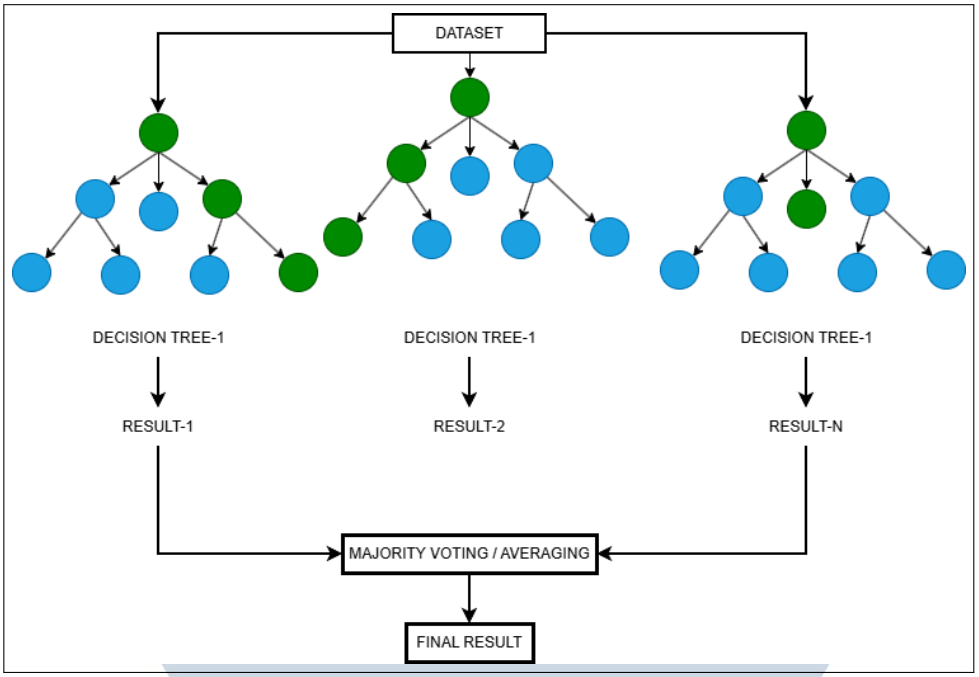
$$Gini(S) = 1 - \sum_{i=1}^{c} p_i^2 \quad (2.3)$$

Keterangan:
S = dataset awal
Pi= probabilitas atau proporsi data pada kelas ke-i
Dalam algoritma CART (*Classification and Regression Tree*), atribut yang menghasilkan nilai *Gini* terkecil biasanya dipilih sebagai *node* pemisah karena menghasilkan pembagian data yang lebih baik.

2.6 Algoritma Random Forest dan Mekanismenya

Random Forest adalah algoritma *supervised learning* yang menggunakan data pelatihan dan data pengujian. Algoritma ini diperkenalkan oleh seorang ahli statistika Amerika bernama Leo Breiman pada tahun 2001 [49]. *Random Forest* merupakan algoritma gabungan (*ensemble*) hasil pengembangan dari algoritma *decision tree*

Random forest banyak digunakan untuk menyelesaikan berbagai permasalahan, seperti *classification*, *regression*, dan beberapa tugas analisis data lainnya. Algoritma ini bekerja dengan membangun sejumlah *decision tree* selama proses pelatihan, kemudian menggabungkan hasil prediksi dari seluruh pohon tersebut untuk meningkatkan kinerja serta ketahanan model terhadap kesalahan. Algoritma ini disebut *random* karena dua alasan utama. Pertama, setiap pohon dibangun menggunakan *bootstrap instance* yang berbeda, yaitu sampel data yang diambil secara acak dari *training data*. Kedua, pada setiap simpul pemisahan selama proses pembentukan *decision tree*, hanya sebagian variabel sebanyak *m* yang dipilih secara acak dari kumpulan fitur yang tersedia, kemudian variabel tersebut digunakan untuk menentukan pemisahan terbaik pada simpul tersebut [50]. Secara konseptual, algoritma ini merupakan kombinasi dari beberapa prediktor berbasis pohon atau *decision tree*, di mana setiap pohon dibangun menggunakan nilai *random vector* yang berfungsi sebagai sampel independen dengan distribusi yang sama pada seluruh pohon di dalam hutan. Hasil prediksi dari model *random forest* diperoleh melalui mekanisme *majority voting* pada tugas *classification*, sedangkan pada tugas *regression* prediksi diperoleh dengan menghitung nilai rata-rata dari seluruh hasil prediksi yang dihasilkan oleh masing-masing *decision tree* [50]. Proses dari algoritma ini terdapat pada gambar 2.6 dibawah ini.



Gambar 2.6. Proses algoritma *random forest*
Referensi: [51]

Proses dimulai dengan mengambil sampel dari dataset awal menggunakan teknik bootstrapping atau pengambilan sampel acak untuk melatih beberapa *decision tree* secara bersamaan. Setiap pohon tersebut dibangun secara independen (seperti Tree-1, Tree-2, hingga Tree-N) agar masing-masing dapat mempelajari pola yang sedikit berbeda dari subset data yang berbeda pula. Setelah setiap pohon menghasilkan prediksinya sendiri, tahap berikutnya adalah penggabungan melalui mekanisme *majority voting* untuk masalah klasifikasi dan averaging (rata-rata) untuk masalah regresi. Melalui proses kombinasi ini, hasil akhir yang diperoleh menjadi lebih stabil, lebih akurat, dan efektif dalam mengurangi risiko *overfitting* dibandingkan jika hanya mengandalkan satu pohon keputusan saja [51].

2.7 Feature Importance

Pada model *machine learning* berbasis *decision tree*, salah satu keunggulan yang dapat dinikmati dari model tersebut adalah kemampuan untuk mengidentifikasi fitur mana yang paling berpengaruh dalam proses klasifikasi. Hal ini sering disebut sebagai *feature importance*, yaitu suatu metode yang digunakan untuk mengukur tingkat kontribusi fitur terhadap keputusan yang dihasilkan oleh model klasifikasi. Pada algoritma *Random Forest*, *feature importance* umumnya dihitung berdasarkan

penurunan *impurity* yang dihasilkan oleh suatu fitur pada seluruh *decision tree* dalam model. Fitur yang menghasilkan penurunan *impurity* yang lebih besar dianggap memiliki kontribusi yang lebih penting dalam proses klasifikasi.
Secara matematis, feature importance dapat diwakili oleh rumus **??** dibawah ini:

$$FI_j = \frac{1}{T} \sum_{t=1}^{T} \Delta I_{t,j} \tag{2.4}$$

Keterangan:
FI_j = nilai importance untuk fitur ke- j
T = jumlah pohon keputusan dalam Random Forest
$\Delta I_{t,j}$ = penurunan impurity yang dihasilkan oleh fitur j pada pohon ke-t
Nilai feature importance yang lebih tinggi menunjukkan bahwa fitur tersebut memiliki pengaruh yang lebih besar terhadap hasil klasifikasi model.

2.8 Confusion Matrix

Dalam evaluasi model klasifikasi, salah satu metode yang umum digunakan adalah *confusion matrix* yaitu tabel yang digunakan untuk menggambarkan performa dari suatu model dengan membandingkan antara hasil prediksi model dan label sebenarnya dari data. *Confusion matrix* terdiri dari empat komponen utama yang tertera pada tabel 2.2 dibawah ini [52].

	Predicted Positive	**Predicted Negative**
Actual Positive	*True Positive* (TP)	*False Negative* (FN)
Actual Negative	*False Positive* (FP)	*True Negative* (TN)

Tabel 2.2. *Confusion Matrix*
Referensi: [53]

Keterangan:

a. *True Positive* (*TP*): URL *phishing* yang benar diklasifikasikan sebagai *phishing* oleh model.

b. *True Negative* (*TN*): URL *legitimate* yang benar diklasifikasikan sebagai *legitimate* oleh model.

c. *False Positive* (*FP*): URL *legitimate* yang salah diklasifikasikan sebagai *phishing* oleh model.

d. *False Negative* (*FN*): URL *phishing* yang salah diklasifikasikan sebagai *legitimate* oleh model.

Dalam konteks deteksi *phishing*, kesalahan *False Negative* merupakan kondisi yang sangat berbahaya karena situs *phishing* diklasifikasikan sebagai situs yang sah, sehingga berpotensi membahayakan pengguna [53]. Untuk mengukur kinerja model klasifikasi secara kuantitatif, digunakan beberapa metrik evaluasi yang dihitung berdasarkan nilai pada *confusion matrix*. Beberapa metrik yang umum digunakan dalam evaluasi model *machine learning* antara lain *accuracy*, *precision*, *recall*, dan *F1-score*.

a. ***Accuracy***

Accuracy merupakan metrik yang menunjukkan persentase jumlah prediksi yang benar dibandingkan dengan seluruh data yang diuji [54].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}$$

Meskipun *accuracy* merupakan metrik evaluasi yang paling sederhana dan mudah dipahami, penggunaan metrik ini memiliki beberapa keterbatasan. *Accuracy* hanya menghitung proporsi prediksi yang benar terhadap seluruh data uji tanpa membedakan jenis kesalahan yang terjadi, yaitu *false positive* dan *false negative*. Dalam konteks deteksi *phishing*, permasalahan ini menjadi penting karena kesalahan dalam bentuk *false negative* dapat menyebabkan situs *phishing* terklasifikasi sebagai situs yang sah. Oleh karena itu, penggunaan *accuracy* saja tidak cukup untuk mengevaluasi performa model klasifikasi secara menyeluruh [55].

b. ***Precision* dan *Recall***

Precision menunjukkan tingkat ketepatan model dalam memprediksi kelas positif dari seluruh prediksi positif yang dihasilkan model, sedangkan *recall* menunjukkan kemampuan model dalam mengidentifikasi seluruh data yang benar-benar termasuk ke dalam kelas positif. Pada kasus deteksi *phishing*, metrik *recall* sering menjadi perhatian utama karena kesalahan *false negative* dapat menyebabkan situs *phishing* diklasifikasikan sebagai situs yang sah [54].

Precision

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

Recall

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

Metrik *precision* dan *recall* dikembangkan untuk mengatasi keterbatasan *accuracy*, terutama pada kondisi *dataset* yang tidak seimbang. *Precision* berfokus pada ketepatan model dalam memprediksi kelas positif, sedangkan *recall* berfokus pada kemampuan model dalam mendeteksi seluruh data yang benar-benar termasuk dalam kelas positif. Selain itu, kedua metrik ini sering kali menunjukkan hubungan yang saling berlawanan (*trade-off*). Peningkatan nilai *precision* dapat menyebabkan penurunan nilai *recall*, dan sebaliknya. Oleh karena itu, diperlukan metrik evaluasi yang mampu menyeimbangkan kedua nilai tersebut [55] [56].

c. ***F1-Score***

F1-score merupakan rata-rata harmonis dari nilai *precision* dan *recall* yang digunakan untuk menyeimbangkan kedua metrik tersebut [54].

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.8}$$

Untuk mengatasi keterbatasan-keterbatasan yang disebutkan diatas, digunakan metrik *F1-score* yang merupakan rata-rata harmonis dari *precision* dan *recall*. Metrik ini memberikan ukuran yang lebih seimbang dalam mengevaluasi performa model klasifikasi, terutama ketika distribusi kelas pada *dataset* tidak seimbang [55]. Dengan menggunakan *F1-score*, evaluasi model tidak hanya mempertimbangkan ketepatan prediksi positif, tetapi juga kemampuan model dalam mendeteksi seluruh data positif secara lebih komprehensif [56].

Metrik-metrik tersebut menjadi sangat penting terutama ketika *dataset* memiliki distribusi kelas yang tidak seimbang, karena dapat memberikan gambaran yang lebih akurat mengenai performa model dibandingkan hanya menggunakan *accuracy*.

2.9 K-Fold Cross Validation

Cross validation adalah teknik dalam *machine learning* yang digunakan untuk menilai sejauh mana hasil analisis statistik dapat diterapkan pada kumpulan data independen [57]. Proses ini sangat penting guna memastikan bahwa model-model tersebut dapat bekerja dengan baik pada data yang belum pernah dilihat sebelumnya. Konsep *cross validation*awalnya digunakan pada bidang statistika dan rancangan eksperimen [58].

Dalam *K-fold Cross Validation*, kumpulan data dibagi menjadi k bagian yang sama besar (*fold*). Model dilatih dan divalidasi sebanyak k kali, di mana pada setiap iterasi satu *fold* digunakan sebagai kumpulan data validasi dan $k-1$ *fold* sisanya digunakan sebagai kumpulan data pelatihan.

Asumsikan memiliki dataset D dibawah ini [58]:

$$D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_n, y_n)\} \tag{2.9}$$

Keterangan:

D = dataset

x_i = vektor fitur

y_i = label yang sesuai untuk pengamatan ke-i

x_i mewakili vektor fitur dan y_i mewakili label yang sesuai untuk pengamatan ke-i.

Iterasi dilakukan mulai dari $i = 1$ hingga k.

Kesalahan *K-fold Cross Validation* dihitung sebagai berikut:

$$\text{K-Fold CV Error} = \frac{1}{k}\sum_{i=1}^{k} E_i \tag{2.10}$$

Keterangan:

K-Fold CV Error = rata-rata error dari seluruh fold

k = jumlah fold dalam cross validation

E_i = error pada fold ke-i

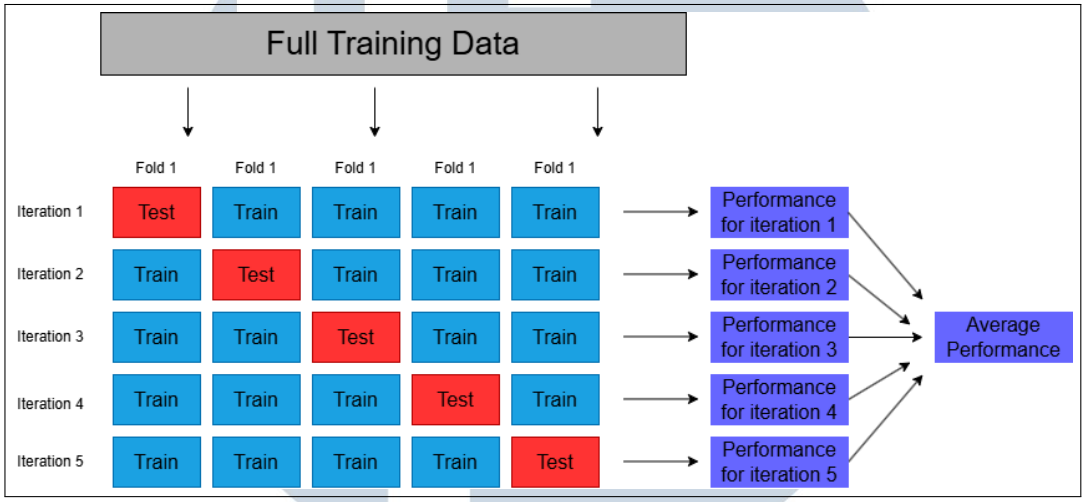
di mana E_i adalah kesalahan pada *fold* ke-i, yang didefinisikan sebagai:

$$E_i = \frac{1}{|D_i|}\sum_{j \in D_i} L(y_j, \hat{y_j}) \tag{2.11}$$

Keterangan:

E_i = error pada fold ke-i

k = jumlah fold dalam cross validation
D_i = himpunan data pada fold ke-i yang digunakan sebagai validation set
$L(y_j, \hat{y}_j)$ = fungsi loss antara nilai aktual y_j dan prediksi $\hat{y}_j$
dengan D_i mewakili himpunan validasi untuk *fold* ke-i dan L merupakan *loss function*. Secara visual, *workflow* dari *K-fold Cross Validation* dapat dilihat pada gambar 2.7 dibawah.



Gambar 2.7. Proses *k-fold cross validation*
Referensi: [58]

Dalam kasus *5-fold Cross Validation*, data dibagi menjadi lima *fold*. Model dilatih dan divalidasi beberapa kali pada iterasi yang berbeda. Pada setiap iterasi, satu *fold* dipilih sebagai himpunan validasi, sedangkan empat *fold* lainnya digunakan untuk pelatihan. Kinerja model dievaluasi pada *fold* validasi untuk setiap iterasi. Setiap *fold* digunakan tepat satu kali sebagai himpunan validasi. Kinerja model secara keseluruhan kemudian diperoleh dengan menghitung rata-rata hasil dari kelima iterasi tersebut.

2.10 Penelitian Terdahulu

Penelitian mengenai deteksi *phishing* berbasis *machine learning* telah banyak dilakukan dengan menggunakan berbagai algoritma klasifikasi diantaranya adalah *Decision Tree*, *Support Vector Machine*, *Random Forest*, *Naive Bayes*, dan *K-Nearest Neighbors* [25]. Sebagian besar penelitian mengevaluasi kinerja model berdasarkan metrik *accuracy*, *precision*, *recall*, dan *F1-score*. Namun, masih terdapat beberapa perbedaan dalam penggunaan dataset, algoritma, teknik seleksi fitur, dan analisis kesalahan klasifikasi.

Windarni et al. [10] membandingkan algoritma *Naive Bayes*, *Decision Tree*, dan *Random Forest* untuk mendeteksi situs *phishing* menggunakan teknik seleksi fitur *Pearson Correlation*. Hasil penelitian menunjukkan bahwa *Naive Bayes* memperoleh akurasi sebesar 60,4%, *Decision Tree* sebesar 94,4%, dan *Random Forest* sebesar 96,3%. Hasil tersebut menunjukkan bahwa *Random Forest* memberikan performa terbaik. Namun, seleksi fitur dilakukan secara terpisah dari proses pemodelan dan penelitian tersebut belum menganalisis kesalahan *false negative* secara khusus.

Mahmud dan Wirawan [23] membandingkan algoritma *Decision Tree*, *Random Forest*, dan *K-Nearest Neighbors* untuk mendeteksi situs *phishing* berdasarkan fitur URL. Hasil penelitian menunjukkan bahwa *Decision Tree* memperoleh akurasi sebesar 83,3%, *Random Forest* sebesar 83,4%, dan *K-Nearest Neighbors* sebesar 48,2%. Perbedaan performa antara *Decision Tree* dan *Random Forest* pada penelitian tersebut relatif kecil. Penelitian tersebut juga belum membahas karakteristik kesalahan klasifikasi, khususnya URL *phishing* yang salah diklasifikasikan sebagai URL *legitimate*.

Suwarno dan Hardjianto [59] mengembangkan sistem deteksi *phishing* berbasis URL menggunakan algoritma *Random Forest* dengan 30 fitur. Model yang dikembangkan memperoleh nilai *precision* sebesar 96%, *recall* sebesar 92%, akurasi sebesar 94%, dan *F-score* sebesar 93%. Meskipun menghasilkan performa yang baik, penelitian tersebut hanya menggunakan satu algoritma sehingga tidak memiliki model dasar sebagai pembanding. Selain itu, penelitian tersebut belum mengintegrasikan proses seleksi fitur ke dalam skema validasi silang.

Taha et al. [60] melakukan perbandingan terhadap lima algoritma klasifikasi, yaitu *Logistic Regression*, *Adaptive Boosting*, *Decision Tree*, *Random Forest*, dan XGBoost. Hasil penelitian menunjukkan bahwa *Random Forest* memperoleh akurasi tertinggi sebesar 97%, sedangkan *Decision Tree* dan XGBoost masing-masing memperoleh akurasi sebesar 94%. Penelitian tersebut menunjukkan keunggulan *Random Forest* dalam deteksi *phishing*. Namun, analisis terhadap *feature importance* dan kesalahan *false negative* belum dilakukan secara mendalam. Berdasarkan penelitian-penelitian tersebut, *Random Forest* secara umum menunjukkan performa yang lebih baik dibandingkan *Decision Tree* maupun beberapa algoritma klasifikasi lainnya.

Meskipun demikian, masih terdapat beberapa keterbatasan pada penelitian sebelumnya. Pertama, sebagian penelitian menggunakan dataset berukuran kecil atau menengah sehingga kemampuan generalisasi model belum dapat dibandingkan secara optimal. Kedua, proses

seleksi fitur umumnya dilakukan sebelum pemodelan dan berada di luar proses validasi silang. Ketiga, analisis kesalahan klasifikasi, khususnya *false negative*, masih jarang dibahas meskipun kesalahan tersebut memiliki risiko tinggi dalam deteksi *phishing*. Perbandingan antara penelitian terdahulu dan penelitian yang dilakukan dirangkum pada Tabel 2.3.

Tabel 2.3. Perbandingan Penelitian Terdahulu

Peneliti	**Dataset**	**Algoritma dan Hasil**	**Keterbatasan**
Windarni et al. (2023) [10]	Dataset Kaggle	*Naive Bayes*, *Decision Tree*, dan *Random Forest* dengan seleksi fitur *Pearson Correlation*. Akurasi tertinggi diperoleh *Random Forest* sebesar 96,3%.	Seleksi fitur dilakukan di luar *pipeline* dan belum terdapat analisis *false negative*.
Mahmud dan Wirawan (2024) [23]	Dataset URL dengan fitur leksikal	*Decision Tree*, *Random Forest*, dan *K-Nearest Neighbors*. Akurasi tertinggi diperoleh *Random Forest* sebesar 83,4%.	Perbedaan performa *Decision Tree* dan *Random Forest* relatif kecil serta belum terdapat analisis *false negative*.
Suwarno dan Hardjianto (2024) [59]	Dataset URL dengan 30 fitur	*Random Forest* memperoleh akurasi 94%, *precision* 96%, *recall* 92%, dan *F-score* 93%.	Hanya menggunakan satu algoritma dan seleksi fitur belum diintegrasikan ke dalam validasi silang.
Taha et al. (2024) [60]	Dataset *phishing website*	*Logistic Regression*, *Adaptive Boosting*, *Decision Tree*, *Random Forest*, dan XGBoost. *Random Forest* memperoleh akurasi tertinggi sebesar 97%.	Belum terdapat analisis *feature importance* dan *false negative* secara mendalam.