Deteksi Website *Phishing* Menggunakan Teknik *Machine Learning*

Lukito, Wilfridus Bambang Triadi Handaya[2]
Program Studi Informatika, Fakultas Teknologi Industri, Universitas Atma Jaya Yogyakarta
Jl. Babarsari No.44, 55281, Daerah Istimewa Yogyakarta, Indonesia
Email: [1]200710677@students.uajy.ac.id, [2]wilfridus.handaya@uajy.ac.id

Abstract. *Phishing is a fraudulent technique that involves masquerading as a trusted entity to steal sensitive data. Machine learning-based detection methods, including XGBoost, Random Forest, and Decision Tree, have been demonstrated to be effective in recognizing patterns indicative of phishing websites. The findings indicate that XGBoost with hyperparameter tuning attains the highest level of accuracy, reaching 96%. After this analysis, the model is implemented in a web service using Flask or FastAPI, enabling users to verify URLs in real time. The system is further equipped with a feature extraction mechanism, encompassing URL analysis, domain age, page content, and automatic labeling to facilitate model retraining. Furthermore, the integration of the system with Telegram bots serves to enhance accessibility, thereby enabling users to perform phishing detection at any time via instant messaging without the constraints of location or device restrictions.*
Keywords: *Machine Learning, Supervised Learning, Phishing*

Abstrak. *Phishing merupakan teknik penipuan yang memanfaatkan penyamaran sebagai entitas terpercaya untuk mencuri data sensitif. Metode deteksi berbasis machine learning, seperti XGBoost, Random Forest, dan Decision Tree, efektif dalam mengenali pola website phishing. Hasil penelitian menunjukkan bahwa XGBoost dengan hyperparameter tuning memberikan akurasi tertinggi, yaitu 96%. Model ini kemudian diimplementasikan dalam web service menggunakan Flask atau FastAPI, sehingga pengguna dapat memeriksa URL secara real-time. Sistem juga dilengkapi mekanisme ekstraksi fitur, termasuk analisis URL, domain age, dan konten halaman, serta pelabelan otomatis untuk memudahkan pelatihan ulang model. Selain itu, integrasi dengan bot Telegram memperluas aksesibilitas, karena pengguna dapat melakukan deteksi phishing kapan saja melalui pesan instan tanpa batasan lokasi atau perangkat.*
Kata Kunci: *Machine Learning, Supervised Learning, Phishing*

1. Pendahuluan

Phishing merupakan teknik penipuan yang berupaya memperoleh informasi sensitif, seperti nama pengguna, kata sandi, dan detail kartu kredit, dengan menyamar sebagai entitas terpercaya melalui komunikasi elektronik [1]. Perkembangan teknologi yang pesat menyebabkan ancaman ini terus meningkat, terutama melalui *email* dan situs web palsu. Berdasarkan laporan Cloudflare pada tahun 2023, sebanyak 90% serangan *cyber* yang berhasil dilakukan memanfaatkan *email* sebagai sarana utama. Dalam kurun waktu Mei 2022 hingga Mei 2023, Cloudflare memproses sekitar 13 juta *email* serta memblokir 250 juta pesan berbahaya sebelum mencapai pengguna [2]. Mengingat kerugian yang ditimbulkan oleh serangan *phishing* kian signifikan, berbagai upaya mitigasi perlu dilakukan. Salah satu pendekatan yang banyak mendapatkan perhatian adalah penerapan sistem deteksi *phishing* menggunakan *machine learning* (ML), khususnya metode *supervised learning* seperti XGBoost, Decision Tree, dan Random Forest dikarenakan kemampuannya dalam menangani data yang kompleks, memberikan hasil yang akurat serta mendukung optimasi model dalam berbagai jenis *dataset*. Model-model ini memanfaatkan *dataset* berlabel untuk mengidentifikasi pola-pola yang membedakan situs web *phishing* dari situs web yang sah [3]–[14].

Pemilihan algoritma XGBoost, Decision Tree, dan Random Forest dalam sistem deteksi *phishing* didasarkan pada bukti empiris yang menunjukkan akurasi tinggi dan kemampuan adaptasi terhadap data yang kompleks. Random Forest terbukti konsisten memberikan hasil terbaik di berbagai penelitian, seperti yang ditunjukkan oleh Ahmad, dkk. dengan akurasi

mencapai 99,55% [10] dan oleh Zaini, dkk. dengan akurasi 94,79% [11]. Sangra, dkk. juga meningkatkan performa Random Forest hingga 95,3% melalui metode pemilihan fitur berbasis korelasi Pearson [3]. XGBoost, di sisi lain, menunjukkan keunggulan dalam mendeteksi *phishing* melalui strategi pemilihan fitur yang canggih dengan akurasi mencapai 99,2% menurut Bahaghighat, dkk. [5], sementara Gradient Boosting yang merupakan model yang sejenis dengan XGBoost mencatat akurasi tertinggi sebesar 92% dalam studi oleh Komalasari, dkk. [7].

Algoritma Decision Tree juga layak dipilih karena kemampuannya memberikan akurasi tinggi disertai efisiensi komputasi. Muliono, dkk. mencatat akurasi rata-rata 97,62% dengan waktu pemrosesan tercepat dibandingkan algoritma lain seperti KNN [9]. Selain itu, metode ensemble seperti yang digunakan pada Random Forest dan XGBoost terbukti meningkatkan stabilitas prediksi terhadap perubahan pola data, sebagaimana dilaporkan oleh Ponnusamy dan Dhandayudam dengan akurasi hingga 96% [6]. Faktor-faktor seperti ukuran *dataset* dan strategi pemilihan fitur yang relevan, sebagaimana ditegaskan oleh Adane, dkk. [4], juga menunjukkan bahwa ketiga algoritma ini memiliki keunggulan dalam menyesuaikan diri dengan data *phishing* yang kompleks dan dinamis.

Penelitian ini bertujuan untuk membandingkan tiga algoritma *machine learning* XGBoost, Decision Tree, dan Random Forest dengan tujuan memilih salah satu model yang paling sesuai untuk diterapkan sebagai sistem deteksi *phishing*. Ketiga algoritma tersebut dipertimbangkan berdasarkan performa unggul yang telah ditunjukkan dalam berbagai studi terdahulu, khususnya dalam mengklasifikasikan data kompleks seperti fitur URL dan konten situs web. Evaluasi dilakukan berdasarkan metrik kinerja seperti akurasi, presisi, recall, dan F1-score. Hasil analisis komparatif ini diharapkan dapat mengidentifikasi model yang paling efektif dan andal untuk digunakan sebagai komponen utama dalam pengembangan sistem deteksi *phishing* yang responsif dan adaptif terhadap perubahan pola serangan.

Serangan *phishing* yang semakin kompleks dan dinamis menjadi salah satu ancaman serius di dunia maya, khususnya dalam mengeksploitasi pengguna melalui URL website palsu yang menyerupai situs resmi. Tantangan utama dalam mendeteksi *phishing* terletak pada kebutuhan akan sistem yang mampu secara akurat dan efisien mengidentifikasi URL berbahaya berdasarkan pola yang tersembunyi dalam struktur URL dan konten terkait. Oleh karena itu, dibutuhkan pendekatan berbasis *machine learning* yang andal, yang tidak hanya akurat tetapi juga mampu beradaptasi dengan karakteristik data yang berubah-ubah.

Penelitian ini bertujuan untuk merancang sistem deteksi *phishing* pada URL situs web dengan memanfaatkan tiga algoritma *supervised learning*, yaitu XGBoost, Decision Tree, dan Random Forest. Ketiga algoritma ini dipilih karena telah terbukti efektif dalam berbagai studi sebelumnya dalam menangani data kompleks, menghasilkan akurasi tinggi, serta mendukung teknik optimasi model seperti pemilihan fitur dan *tuning hyperparameter*. Perbandingan dilakukan untuk mengidentifikasi model terbaik yang paling sesuai digunakan dalam konteks deteksi *phishing* berbasis URL.

Selanjutnya, model deteksi yang terpilih akan diintegrasikan ke dalam sebuah layanan web yang interaktif, sehingga dapat digunakan langsung oleh pengguna untuk memverifikasi URL secara langsung. Selain itu, sistem ini juga akan dikembangkan agar dapat terhubung dengan platform Telegram melalui implementasi bot, guna memperluas aksesibilitas dan kemudahan penggunaan layanan deteksi. Diharapkan hasil penelitian ini tidak hanya menghasilkan model yang efektif dan efisien, tetapi juga dapat diimplementasikan secara praktis untuk memberikan perlindungan terhadap serangan *phishing* di berbagai platform digital.

2. Tinjauan Pustaka

Penelitian di bidang pendeteksian *phishing* menggunakan *machine learning* terus berkembang seiring meningkatnya ancaman siber yang memanfaatkan tautan berbahaya. Sangra, dkk. mengoptimalkan kinerja Random Forest dengan metode korelasi Pearson dengan hasil akurasi 95,3% [3]. Menurut Adane, dkk., ukuran himpunan data dan metode pemilihan fitur sangat memengaruhi kinerja model *machine learning* [4]. Bahaghighat, dkk. merancang

strategi pemilihan fitur canggih yang efektif dalam meningkatkan rasio deteksi *phishing* dengan menggunakan algoritma XGBoost dengan hasil akurasi 99,2% [5]. Ponnusamy dan Dhandayudam mengajukan metode *ensemble feature selection* dan *bagging* dengan hasil akurasi deteksi mencapai 96% untuk memperkuat stabilitas prediksi saat pola data berubah [6].

Komalasari, dkk. juga menggunakan pendekatan serupa dengan menerapkan berbagai algoritma seperti Decision Tree, Random Forest, Support Vector Machine, dan Gradient Boosting untuk mengidentifikasi domain *phishing*. Dalam penelitian ini, model Gradient Boosting menunjukkan akurasi tertinggi sebesar 92% [7], menegaskan pentingnya pemilihan model yang tepat dalam meningkatkan efektivitas deteksi *phishing*. Mahmud dan Wirawan juga membandingkan beberapa algoritma, termasuk Decision Tree, Random Forest, dan K-Nearest Neighbors (KNN), dalam mendeteksi website *phishing* berdasarkan fitur URL. Hasil penelitian menunjukkan bahwa Random Forest memiliki kinerja terbaik dengan akurasi 83,4%, presisi 86%, recall 83%, dan F1-score 83%, menjadikannya pilihan yang lebih unggul dibandingkan KNN yang hanya mencapai akurasi 48,2% [8].

Penelitian oleh Muliono, dkk. mengusulkan model klasifikasi *phishing* berbasis *machine learning* yang menggunakan 18 fitur untuk mendeteksi situs *phishing*, termasuk panjang URL, penggunaan karakter spesial, dan analisis konten HTML dan JavaScript. Algoritma Decision Tree menunjukkan hasil terbaik dengan akurasi rata-rata 97,62% dan waktu komputasi tercepat, menjadikannya pilihan unggul untuk mendeteksi serangan *phishing* dibandingkan dengan algoritma K-Nearest Neighbor (KNN) yang mencapai akurasi 96,79% pada nilai k = 12 [9]. Ahmad, dkk. mengevaluasi berbagai teknik *machine learning* untuk mendeteksi serangan *phishing* menggunakan *dataset* dari PhishTank dan UCI. Hasil penelitian menunjukkan bahwa model Random Forest memiliki akurasi tertinggi, mencapai 99,55% untuk K-*fold cross-validation*, 99% untuk pemilihan fitur, dan 99,45% untuk *tuning hyperparameter*, menjadikannya salah satu model paling akurat dalam mendeteksi *phishing* dibandingkan model lain seperti SVM dan Logistic Regression [10].

Zaini, dkk. menerapkan pendekatan berbasis *heuristik* untuk mendeteksi serangan *phishing* menggunakan beberapa algoritma *machine learning*, termasuk Random Forest, J48, Multi-Layer Perceptron (MLP), dan K-Nearest Neighbors (KNN). Hasil penelitian menunjukkan bahwa Random Forest mencapai akurasi tertinggi sebesar 94,79%, menjadikannya algoritma paling efektif untuk mendeteksi *phishing* dibandingkan metode lain seperti KNN yang hanya mencapai akurasi 93,08% [11]. Borate, dkk. melakukan tinjauan komprehensif terhadap berbagai teknik pendeteksian serangan *phishing* menggunakan *machine learning*. Dalam penelitian ini, algoritma Random Forest menunjukkan akurasi tertinggi sebesar 97%, diikuti oleh Decision Tree dengan akurasi 92%, dan Support Vector Machines (SVM) dengan akurasi 89%. Meskipun Random Forest menunjukkan hasil terbaik dalam klasifikasi URL *phishing*, studi ini juga menyoroti tantangan seperti ketidakseimbangan data dan kebutuhan untuk meningkatkan fitur ekstraksi untuk hasil yang lebih akurat [12].

Lokesh dan BoreGowda mengembangkan sistem klasifikasi *phishing* yang menggunakan berbagai algoritma *machine learning* seperti Random Forest, K-Nearest Neighbors (KNN), Decision Tree, dan Linear SVC untuk menganalisis metadata URL. Pendekatan ini bertujuan untuk mengatasi keterbatasan deteksi *phishing* tradisional dengan mengekstraksi fitur numerik yang lebih akurat dalam membedakan antara situs *phishing* dan situs yang sah [13]. Hannousse dan Yahiouche mengusulkan skema untuk membangun *dataset phishing* yang dapat di replikasi dan diperluas, dengan menggunakan 87 fitur umum yang dievaluasi untuk relevansi dan kinerja. Hasil penelitian menunjukkan bahwa Random Forest adalah algoritma dengan prediksi terbaik, mencapai akurasi hingga 96,83% ketika menggunakan kombinasi fitur eksternal dan konten halaman web [14].

3. Metodologi Penelitian

Penelitian ini difokuskan pada pembuatan sistem deteksi *phishing* berbasis *machine learning* yang diintegrasikan dengan layanan web serta bot Telegram. Pada tahap awal, data dikumpulkan dari *"Datasets for Phishing Websites Detection"* [15] yang memuat 58.645

instance, terdiri atas 30.647 data *phishing* dan 27.998 data *legitimate*. Data tersebut kemudian dipisahkan menjadi 80% data pelatihan dan 20% data pengujian. Proses selanjutnya adalah ekstraksi 111 fitur yang terbagi dalam enam kategori, yakni *full* URL, domain, *directory*, *file*, parameter, dan *resolving* URL.

Tahap pengembangan model dilakukan dengan memilih tiga algoritma, yaitu Random Forest, Decision Tree, dan XGBoost. Ketiga algoritma tersebut digunakan karena memiliki kemampuan yang baik dalam menangani data beragam serta memberikan kinerja yang andal pada tugas klasifikasi. Model dilatih dengan parameter terbaik melalui teknik *hyperparameter tuning*, kemudian dievaluasi dengan beberapa metrik seperti akurasi, presisi, *recall*, dan *F1-score*. Model yang memiliki kinerja terbaik akan digunakan pada tahap implementasi sistem.

Setelah model dipilih, sistem pendeteksi *phishing* diimplementasikan menggunakan Flask sebagai kerangka kerja (*framework*) utama. Arsitektur yang dibangun meliputi antarmuka pengguna (*frontend*), bagian *backend* untuk memanggil model, serta model *machine learning* itu sendiri. Alur kerja sistem mencakup penerimaan *input* URL, ekstraksi fitur oleh *backend*, dan proses prediksi menggunakan model terlatih. Hasil prediksi selanjutnya dikembalikan ke antarmuka agar pengguna dapat mengetahui apakah suatu URL tergolong *phishing* atau tidak. Dalam sistem ini, konsep *adversarial system* diterapkan untuk mempertahankan relevansi model terhadap perubahan pola serangan. Setiap URL yang dimasukkan oleh pengguna dapat ditambahkan kembali ke dalam *dataset* apabila diperlukan, lalu dilakukan pelatihan ulang agar model kian adaptif.

Untuk memudahkan akses pengguna di luar platform web, model juga diintegrasikan dengan Telegram. Tahap ini memanfaatkan *library* python-telegram-bot sebagai jembatan komunikasi antara bot dan layanan web Flask. Ketika pengguna mengirimkan URL ke bot Telegram, permintaan akan diteruskan kepada layanan Flask untuk proses ekstraksi fitur dan prediksi. Hasil penilaian risiko *phishing* kemudian dikembalikan ke bot agar pengguna dapat melihatnya secara langsung di aplikasi Telegram. Dengan demikian, implementasi lintas platform ini diharapkan dapat meningkatkan jangkauan serta kemudahan akses sistem deteksi *phishing*.

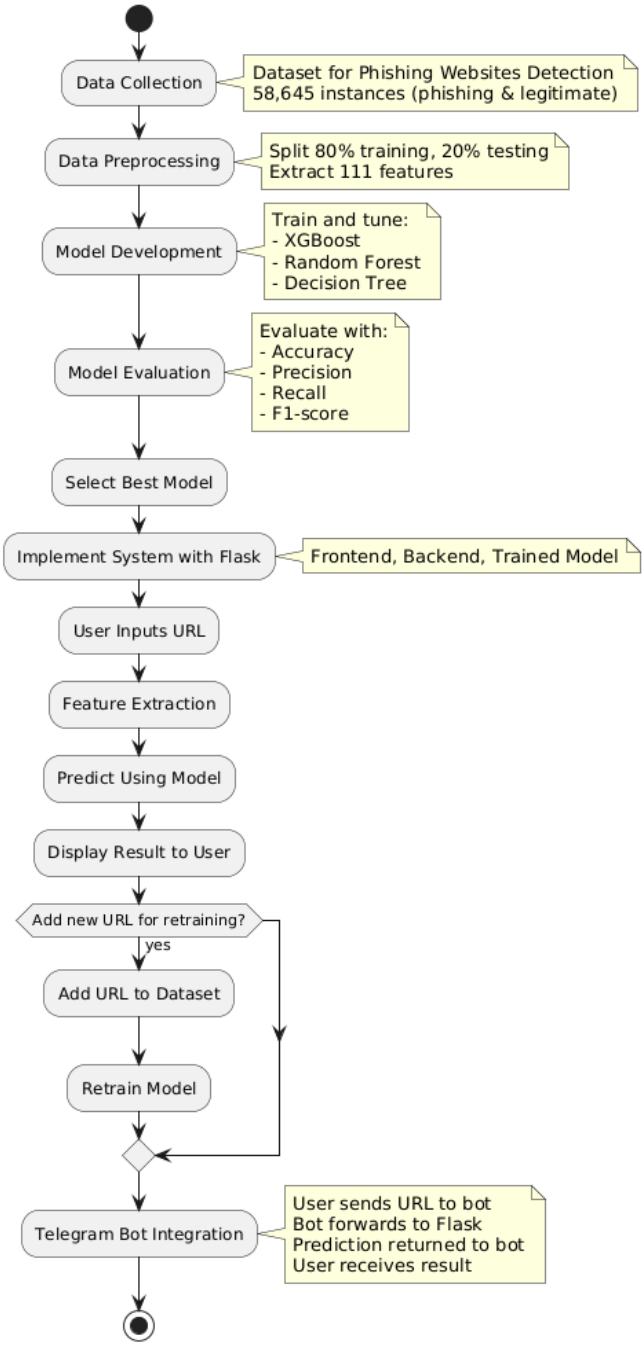
Flowchart metode penelitian yang menggambarkan tahapan-tahapan tersebut dapat dilihat pada Gambar 1.

4. Hasil dan Diskusi

4.1 Pengembangan dan Evaluasi Model

Dataset yang digunakan untuk pengembangan model memiliki rasio 80:20, dengan 80% data untuk *training* dan 20% data untuk testing. *Dataset phishing* terdiri dari 30.647 data, sedangkan *dataset legitimate* terdiri dari 27.998 data. Penelitian ini membandingkan kinerja model Decision Tree, Random Forest, dan XGBoost, baik dengan konfigurasi *default* maupun setelah dilakukan *Randomized Search dan Grid Search* untuk *hyperparameter tuning*. Setiap model dievaluasi berdasarkan metrik *accuracy, precision, recall,* dan *F1-score*. Hasil dalam Gambar 2 menunjukkan bahwa XGBoost dengan *tuning Grid Search* mencapai performa terbaik, dengan akurasi 96,14%, *precision* 96,17%, *recall* 96,48%, dan F1-score 96,32%. Sehingga sistem deteksi menggunakan model XGBoost untuk melakukan deteksi. Gambar 3 menampilkan parameter terbaik untuk setiap model yang telah dituning.

Model Decision Tree mencapai performa optimal dengan parameter max_depth: 10 dan min_sample_split: 2. Model Random Forest bekerja dengan baik dengan parameter min_sample_split: 2 dan n_estimators: 100. Model XGBoost mencapai performa optimal dengan parameter learning_rate: 0,2, max_depth: 10, dan n_estimators: 200.



Gambar 1. Flowchart Kerja Sistem Deteksi

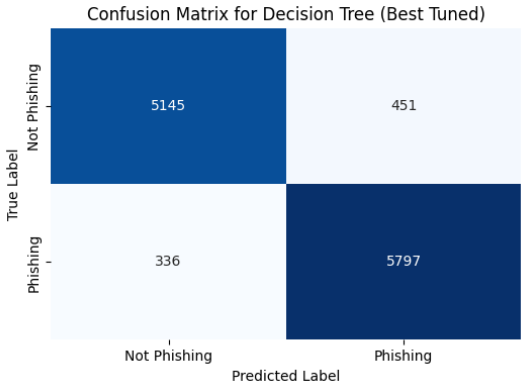
```
=== Comparison of Results ===
               Model             Stage                                                 Parameters  Accuracy  Precision   Recall  F1 Score
Decision Tree          Default                                {'max_depth': None, 'min_samples_split': 2}  0.931964   0.934517 0.935431  0.934974
Random Forest          Default {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 100}  0.958053   0.956909 0.963150  0.960020
      XGBoost          Default {'learning_rate': None, 'max_depth': None, 'n_estimators': None}  0.957456   0.959244 0.959400  0.959322
Decision Tree Randomized Search                     {'min_samples_split': 15, 'max_depth': 10}  0.932475   0.927622 0.944562  0.936016
Decision Tree       Grid Search                      {'max_depth': 10, 'min_samples_split': 2}  0.932901   0.927817 0.945214  0.936435
Random Forest Randomized Search  {'n_estimators': 50, 'min_samples_split': 2, 'max_depth': None}  0.958308   0.957078 0.963476  0.960267
Random Forest       Grid Search {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 100}  0.958053   0.956909 0.963150  0.960020
      XGBoost Randomized Search    {'n_estimators': 200, 'max_depth': 15, 'learning_rate': 0.05}  0.960951   0.960858 0.964618  0.962734
      XGBoost       Grid Search     {'learning_rate': 0.2, 'max_depth': 10, 'n_estimators': 200}  0.961463   0.961795 0.964618  0.963204
```

Gambar 2. Hasil Performa Setiap Model

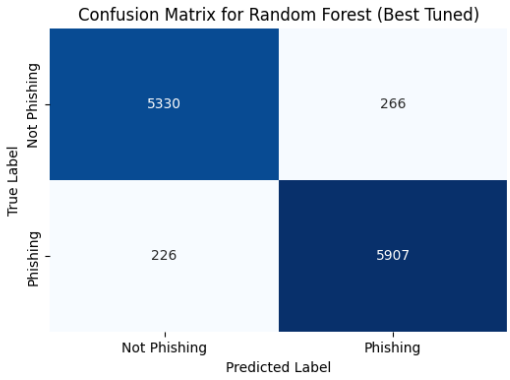
```
=== Best Parameters for Each Model ===

Decision Tree:
  max_depth: 10
  min_samples_split: 2

Random Forest:
  max_depth: None
  min_samples_split: 2
  n_estimators: 100

XGBoost:
  learning_rate: 0.2
  max_depth: 10
  n_estimators: 200
```

Gambar 3. Parameter Terbaik Untuk Setiap Model

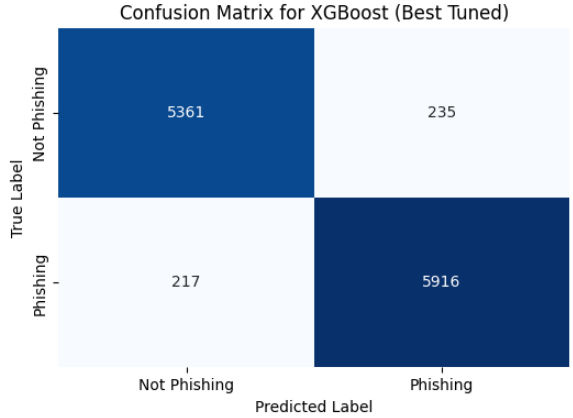
Berdasarkan *Confusion* Matrix yang ditampilkan pada Gambar 4, performa model Decision Tree yang telah dituning menunjukkan 5.145 *True Positive* (TP), 451 *False Positive* (FP), 336 *False Negative* (FN), dan 5.797 *True Negative* (TN). Selanjutnya, *Confusion* Matrix pada Gambar 5 mengilustrasikan performa model Random Forest yang telah dituning, dengan 5.330 TP, 266 FP, 226 FN, dan 5.907 TN. Terakhir, *Confusion* Matrix XGBoost yang telah dituning pada Gambar 6 memperlihatkan 5.361 TP, 235 FP, 217 FN, dan 5.916 TN. Dalam deteksi URL *phishing*, *False Positive* dapat berbahaya karena model dapat salah mengklasifikasikan sebuah URL *phishing* menjadi URL *legitimate*, yang berpotensi membahayakan pengguna.



Gambar 4. *Confusion Matrix Decision Tree*



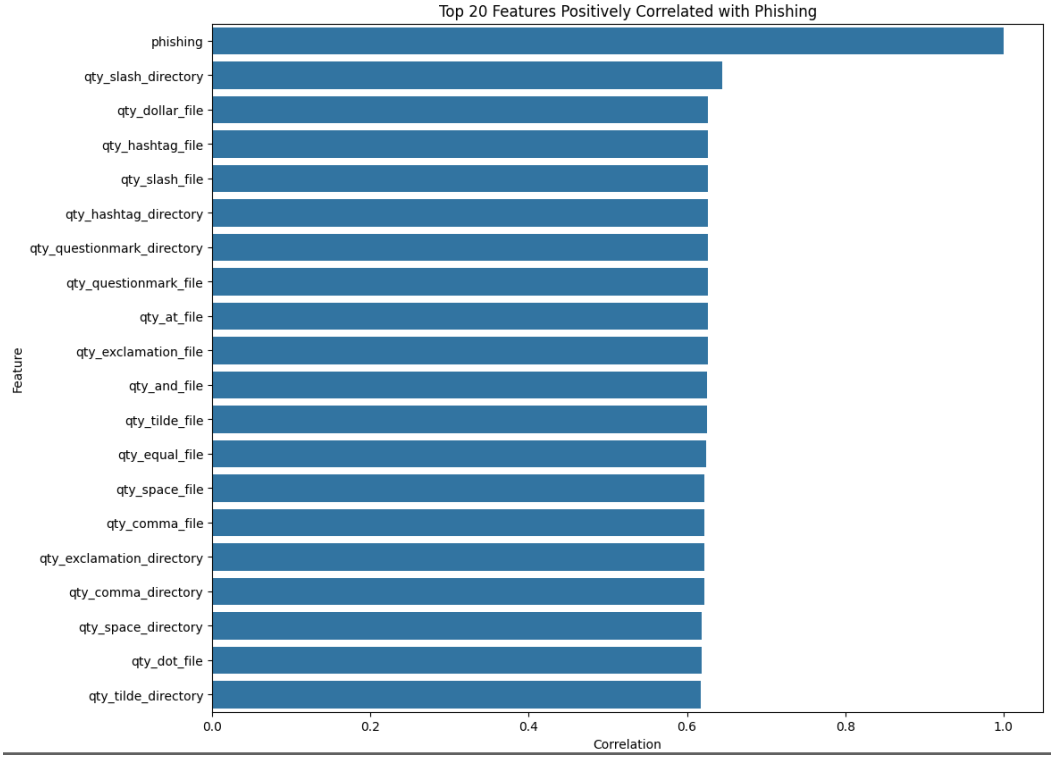
Gambar 5. *Confusion Matrix Random Forest*



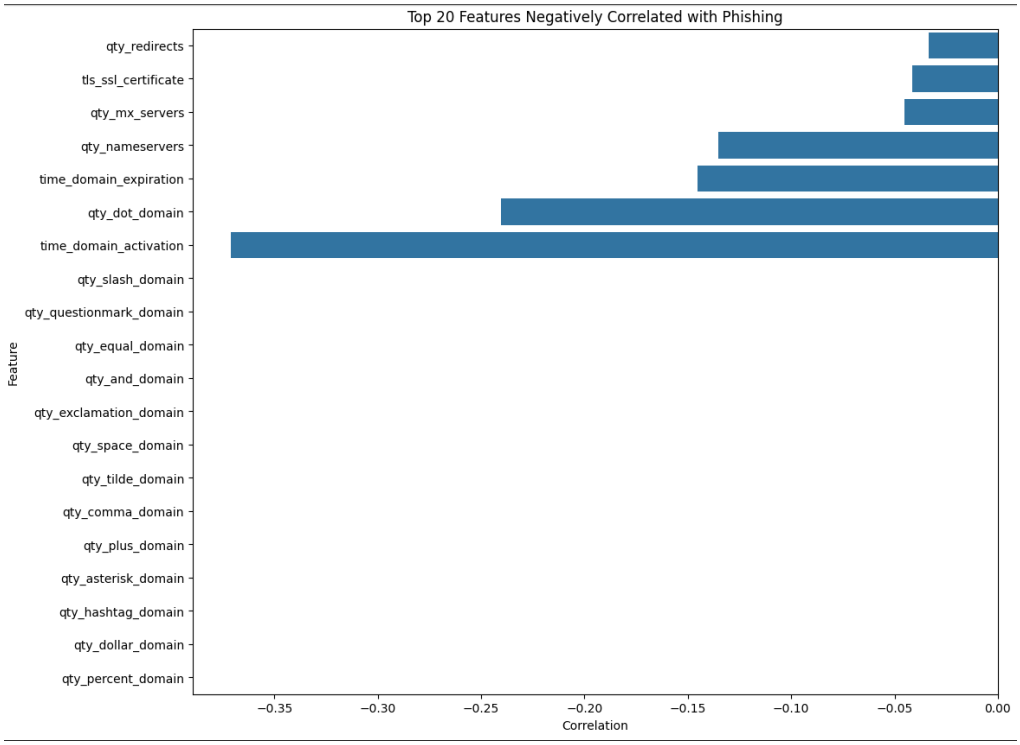
Gambar 6. *Confusion Matrix XGBoost*

4.2 Analisis Korelasi Atribut URL dalam Dataset terhadap *Phishing*

Berdasarkan analisis korelasi terhadap label *phishing* pada Gambar 7, ditemukan bahwa pola tertentu dalam struktur URL dapat menjadi indikator penting untuk mengidentifikasi apakah sebuah situs bersifat *phishing* atau bukan. Salah satu temuan utama ialah korelasi tinggi antara jumlah *subdirectory* (*qty_slash_directory*) dengan *phishing*, yang menunjukkan bahwa situs *phishing* kerap memiliki struktur URL lebih panjang dan kompleks. Selain itu, penggunaan simbol tertentu seperti dolar ($), pagar (#), tanda tanya (?), *at* (@), serta tanda seru (!) dalam *directory* maupun *file* URL juga berkorelasi erat dengan *phishing*. Di sisi lain, pada Gambar 8, fitur-fitur yang berkaitan dengan domain misalnya jumlah *ampersand* (&), spasi, atau karakter khusus lain tidak memiliki korelasi signifikan, bahkan beberapa data tercatat sebagai *NaN*. Dengan demikian, perbedaan mencolok antara URL *phishing* dan non-*phishing* umumnya terlihat pada struktur *file* dan *directory*, bukan pada domain. Oleh karena itu, langkah deteksi *phishing* dapat difokuskan pada analisis pola dalam struktur URL, terutama jumlah serta jenis karakter khusus yang digunakan, diikuti dengan visualisasi dan pemodelan *machine learning* untuk meningkatkan keakuratan deteksi.



Gambar 7. Atribut Dataset yang Positif berkorelasi dengan *Phishing*



Gambar 8. Atribut Dataset yang Negatif berkorelasi dengan *Phishing*

4.3 Implementasi Model

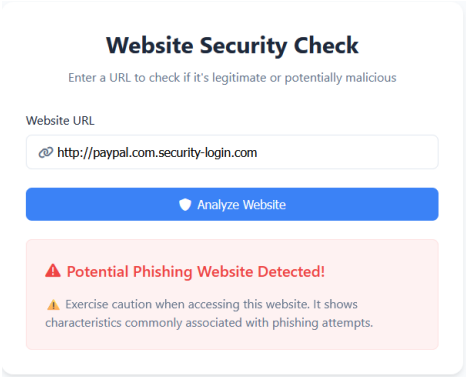
Pada tahap implementasi, model *machine learning* yang telah dilatih untuk mendeteksi *phishing* akan diintegrasikan ke dalam sebuah web *service* menggunakan Flask. Flask dipilih karena sifatnya yang ringan, fleksibel, dan mudah diimplementasikan, sehingga cocok untuk membangun sistem deteksi *phishing* berbasis web yang sederhana dan cepat.

4.4 Implementasi Sistem

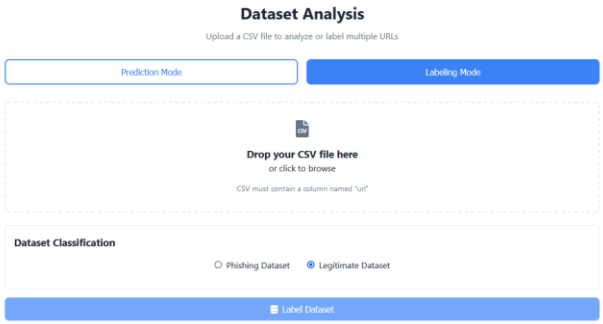
Sistem deteksi website *phishing* berbasis Flask ini terdiri atas tiga komponen utama, yaitu antarmuka pengguna (*frontend*), server aplikasi (*backend*), dan model *machine learning*. Bagian *frontend* menyediakan tampilan bagi pengguna untuk memasukkan URL yang akan diperiksa, sementara *backend* bertugas mengelola permintaan dari *frontend* serta berinteraksi dengan model *machine learning* yang telah dilatih dan disimpan. Ketika pengguna memasukkan sebuah URL, data tersebut dikirim ke *backend* untuk diekstraksi fiturnya, seperti struktur URL dan parameter lain yang relevan bagi model. Setelah fitur berhasil diekstraksi, model *machine learning* memproses data tersebut untuk memprediksi apakah URL tergolong *phishing* atau *legitimate*. Hasil prediksi kemudian dikembalikan ke antarmuka pengguna dalam format yang mudah dipahami. Pada proses integrasinya, model *machine learning* dimuat saat aplikasi berjalan, sehingga setiap URL yang dimasukkan dapat langsung diekstraksi fiturnya dan diprediksi oleh model. Dengan demikian, pengguna dapat memperoleh hasil deteksi secara real-*time* melalui halaman web Flask yang sederhana dan cepat.

Gambar 9 menampilkan fitur deteksi *phishing*, di mana pengguna dapat memasukkan URL yang ingin diperiksa dan memperoleh hasil prediksi. Selanjutnya, Gambar 10 menunjukkan fitur *adversarial system* yang berfungsi untuk melatih ulang model *machine learning* sehingga akurasi deteksi *phishing* dapat ditingkatkan. Pada fitur *adversarial system* ini, pengguna diberi kesempatan melakukan pelabelan URL dengan menentukan apakah URL tergolong *legitimate* atau *phishing*, kemudian hasilnya akan di tampilkan seperti pada Gambar 11. Label yang diberikan kemudian disimpan dalam *storage* dengan bentuk *file* CSV, lalu

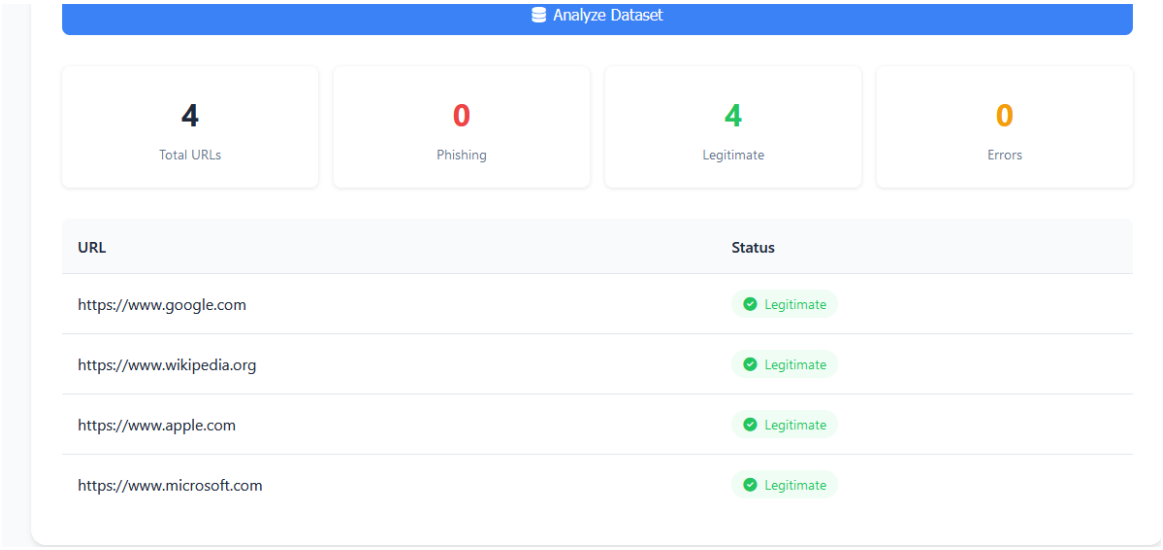
digabungkan dengan *dataset* lama untuk dilatih kembali menggunakan model sebelumnya. Proses ini ditunjukkan dalam Gambar 12, yang menampilkan performa model lama yang sempat keliru mendeteksi URL *phishing* sebagai *legitimate*. Setelah pelatihan ulang, sebagaimana tergambar pada Gambar 13, akurasi sistem meningkat dan model dapat mengenali URL *phishing* dengan lebih baik. Gambar 14 mengilustrasikan fitur bot Telegram yang memanfaatkan fungsi deteksi serupa dengan Gambar 9, sehingga pengguna dapat melakukan pengecekan URL *phishing* langsung melalui aplikasi Telegram.



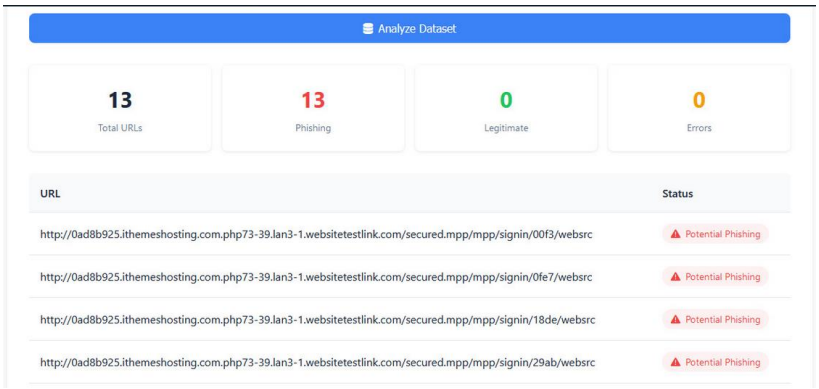
Gambar 9. Fitur Deteksi Phishing



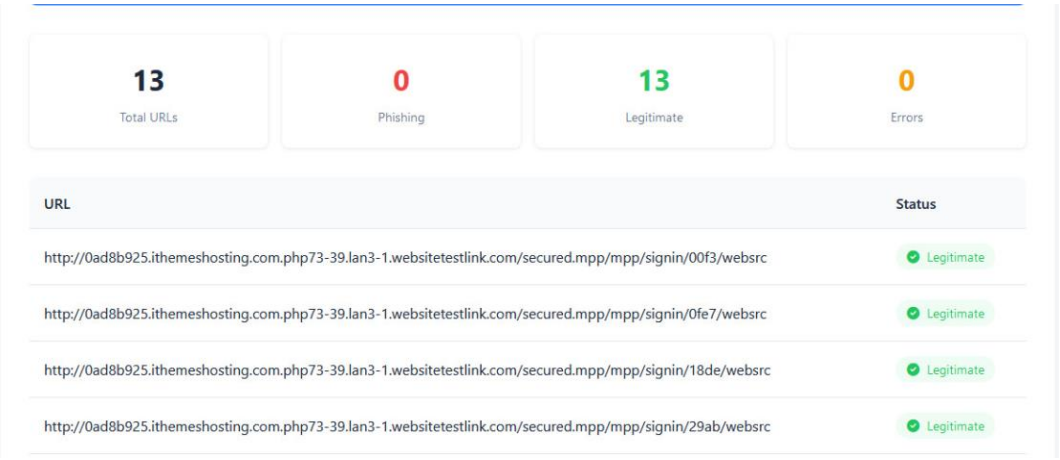
Gambar 10. Fitur Adversarial System



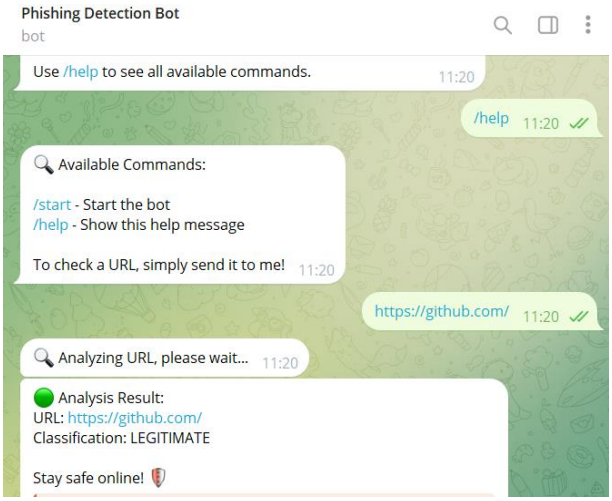
Gambar 11. Hasil Pelabelan



Gambar 12. Performa Model Lama



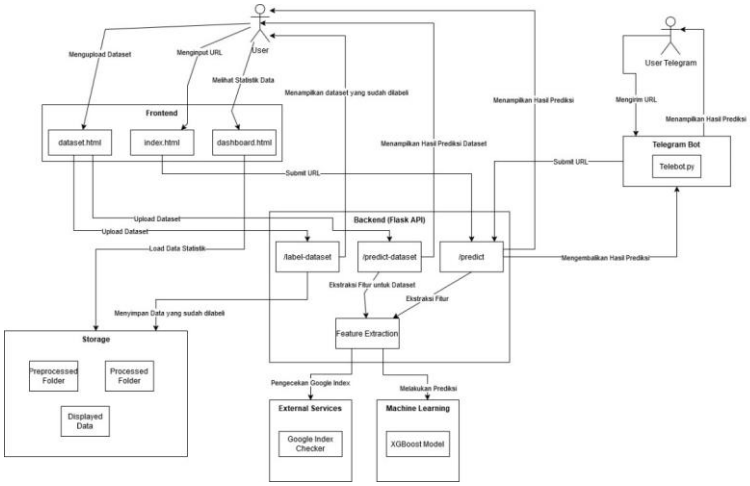
Gambar 13. Performa Model Baru



Gambar 14. Fitur Bot Telegram

Gambar 15 menggambarkan alur kerja sistem deteksi *phishing* berbasis *machine learning* yang melibatkan interaksi antara pengguna, sistem *backend*, dan layanan eksternal. Pengguna dapat mengunggah *dataset* dan memasukkan URL untuk dianalisis. Dataset yang diunggah kemudian diproses oleh *backend* menggunakan Flask API, yang mengatur ekstraksi fitur dan prediksi menggunakan model XGBoost. Hasil dari analisis ini dapat dilihat melalui *frontend*, di mana pengguna dapat melihat statistik dan hasil prediksi.

Selain itu, sistem ini juga terintegrasi dengan bot Telegram yang memungkinkan pengguna untuk mengirim URL melalui aplikasi Telegram dan mendapatkan hasil prediksi langsung. Proses *backend* juga mencakup pengecekan status indeks Google melalui layanan eksternal, serta penyimpanan *dataset* yang sudah diberi label di folder yang sesuai untuk penggunaan lebih lanjut. Sistem ini menggabungkan beberapa komponen untuk memberikan deteksi *phishing* yang efisien dan mudah diakses.



Gambar 15. Alur Kerja Sistem Deteksi

5. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan hasil perbandingan antara model XGBoost, Decision Tree, dan Random Forest, dapat disimpulkan bahwa XGBoost merupakan model yang paling optimal untuk mendeteksi URL *phishing* dalam penelitian ini, dengan performa terbaik setelah dilakukan t*uning hyperparameter*. Integrasi model ke dalam layanan web menunjukkan bahwa sistem mampu mendeteksi URL *phishing* dan *non-phishing* secara interaktif, meskipun masih terdapat ruang untuk peningkatan akurasi. Penggunaan fitur ekstraksi dan pelabelan otomatis memberikan potensi bagi sistem untuk terus dikembangkan melalui pelatihan ulang model. Selain itu, integrasi dengan bot Telegram membuktikan bahwa sistem deteksi ini dapat diakses secara luas dan mudah digunakan, menjawab kebutuhan akan solusi praktis dan responsif dalam mengidentifikasi serangan *phishing* secara langsung.

5.2 Saran

Penelitian selanjutnya dapat meningkatkan jumlah dan keragaman *dataset* yang digunakan agar model mampu memahami pola website *phishing* yang lebih kompleks dengan akurasi lebih tinggi, serta mengoptimalkan model menggunakan teknik yang lebih baik atau mencoba algoritma lain demi meningkatkan performa deteksi. Selain itu, sistem deteksi juga dapat diintegrasikan dengan platform lain, seperti ekstensi peramban (browser *extension*), aplikasi seluler, atau sistem keamanan lainnya, guna meningkatkan kemudahan akses dan efektivitas deteksi.

Referensi

[1] S. Kurniawan, “Phising: Pengertian, Cara Kerja dan Langkah Mengatasinya,” Niagahoster Blog, May 22, 2023. https://www.niagahoster.co.id/blog/mengatasi-phishing/

[2] E. Dzuba, “Introducing Cloudflare’s 2023 phishing threats report,” The Cloudflare Blog, Oct. 26, 2023. https://blog.cloudflare.com/2023-phishing-report/

[3] E. Sangra, R. Agrawal, P. R. Gundalwar, K. Sharma, D. Bangri, and D. Nandi, "Malicious Website Detection Using Random Forest and Pearson Correlation for Effective Feature Selection," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 15, no. 8, pp. 772-780, 2024, http://dx.doi.org/10.14569/IJACSA.2024.0150876

[4] K. Adane, Berhanu Beyene, and Mohammed Abebe, “ML and DL-based Phishing Website Detection: The Effects of Varied Size Datasets and Informative Feature Selection Techniques”, *Journal of Artificial Intelligence and Technology*, vol. 4, no. 1, pp. 18–30, Sep. 2023.

[5] M. Bahaghighat, M. Ghasemi, and F. Ozen, "A high-accuracy phishing website detection method based on machine learning," *Journal of Information Security and Applications*, vol. 77, p. 103553, 2023. Available: https://doi.org/10.1016/j.jisa.2023.103553.

[6] P. and Dhandayudam, P., "An Optimized Bagging Learning with Ensemble Feature Selection Method for URL Phishing Detection", *Journal of Electrical Engineering & Technology*, vol. 19, no. 3, Springer, pp. 1881–1889, 2023. doi:10.1007/s42835-023-01680-z.

[7] D. Komalasari, T. B. Kurniawan, D. A. Dewi, M. Z. Zakaria, Z. Abdullah, and A. Alanda, "Phishing Domain Detection Using Machine Learning Algorithms", *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 15, no. 1, pp. 318–327, Feb. 2025. doi: https://doi.org/10.18517/ijaseit.15.1.12553

[8] A. F. Mahmud and S. Wirawan, "Deteksi Phishing Website menggunakan Machine Learning Metode Klasifikasi," *Sistemasi: Jurnal Sistem Informasi*, vol. 13, no. 4, pp. 1368-1380, 2024, doi: https://doi.org/10.32520/stmsi.v13i4.3456.

[9] Y. Muliono, M. A. Ma'ruf, and Z. M. Azzahra, "Phishing Site Detection Classification Model Using Machine Learning Approach," *Jurnal EMACS (Engineering, Mathematics and Computer Science)*, vol. 5, no. 2, pp. 63-67, May 2023, doi: 10.21512/emacsjournal.v5i2.995163.

[10] S. K. Ahmad, B. A. Dapshima, and Y. C. Essa, "Detection of Phishing Attacks Using Machine Learning Techniques," *International Research Journal of Modernization in Engineering, Technology and Science*, vol. 6, no. 7, pp. 1166, July 2024, doi: 10.56726/IRJMETS60054.

[11] N. S. Zaini, D. Stiawan, M. F. Ab Razak, A. Firdaus, W. I. S. Wan Din, S. Kasim, and T. Sutikno, "Phishing Detection System Using Machine Learning Classifiers," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 17, no. 3, pp. 1165-1171, Mar. 2020, doi: 10.11591/ijeecs.v17.i3.pp1165-1171.

[12] V. Borate, D. Adsul, M. Dhakane, M. Gawade, M. Ghodake, and M. Jadhav, "A Comprehensive Review of Phishing Attack Detection Using Machine Learning Techniques," *International Journal of Advanced Research in Science Communication and Technology*, vol. 4, pp. 435-441, 2024, doi: 10.48175/IJARSCT-19963.

[13] G. Harinahalli Lokesh and G. BoreGowda, "Phishing Website Detection Based on Effective Machine Learning Approach," *Journal of Cyber Security Technology*, vol. 5, no. 1, pp. 1-14, 2020, doi: 10.1080/23742917.2020.1813396.

[14] A. Hannousse and S. Yahiouche, "Towards Benchmark Datasets for Machine Learning Based Website Phishing Detection: An Experimental Study," *Engineering Applications of Artificial Intelligence*, vol. 104, 2021, Art. no. 104347, doi: 10.1016/j.engappai.2021.104347.

[15] G. Vrbančič, I. Fister, and V. Podgorelec, "Datasets for phishing websites detection," Data in Brief, vol. 33, p. 106438, Oct. 2020, doi: 10.1016/j.dib.2020.106438.