 ISSN:2302-8149
e-ISSN:2540-9719

Optimasi Kinerja Deteksi *Phishing* dengan Analisis Korelasi Variabel dan *Imbalance Learning*

Optimization of Phishing Detection Performance with Variable Correlation Analysis and Imbalance Learning

[1]Samsul Arifin, [2]Fandy Setyo Utomo*
[1,2]Program Studi Informatika, Fakultas Ilmu komputer, Universitas Amikom Purwokerto
[1,2]Jl. Letjend Pol. Soemarto No.127, Watumas, Purwokerto Utara, Banyumas, Jawa Tengah 53127
*e-mail: fandy_setyo_utomo@amikompurwokerto.ac.id

(***received*:** 8 October 2024, ***revised*:** 18 February 2025, ***accepted*:** 19 February 2025)

Abstrak

Phishing adalah ancaman terhadap keamanan dunia maya yang sering terjadi, di mana para pelaku mencoba menipu pengguna untuk menyerahkan informasi pribadi seperti kata sandi, nomor kartu kredit, dan berbagai data sensitif lainnya. Dengan perkembangan teknologi, teknik *phishing* semakin canggih dan sulit terdeteksi oleh metode tradisional. Oleh karena itu, sangat penting untuk merancang teknik yang mampu mendeteksi situs *phishing* dengan akurasi yang tinggi. Penelitian ini bertujuan untuk mengoptimasi kinerja deteksi *phishing* dengan mengintegrasikan analisis korelasi variabel untuk seleksi fitur dan penggunaan teknik *imbalance learning* guna mengatasi ketidakseimbangan data. Tahapan penelitian mencakup *Data Collection, Data Preprocessing, Data Exploration* meliputi analisis korelasi, pembersihan Fitur yang berkorelasi rendah, dan visualisasi data. Pada tahap *Model Building and Training*, dilakukan pembagian fitur dan label, Training data, dan penerapan teknik penyeimbangan data, diakhiri dengan *Model Evaluation*. Algoritma yang diuji mencakup *Logistic Regression, Naive Bayes, K-Nearest Neighbors, Support Vector Machine, Multi-Layer Perceptron, Decision Tree, Random Forest, Gradient Boosting, CatBoost.* Hasil penelitian ini menunjukkan Algoritma *KNN* memberikan kinerja paling baik dengan akurasi mencapai 91,25%. serta hasil optimal pada metrik *Precision, Recall*, dan *F1-Score*, masing-masing sebesar 0,906943, 0,927858, dan 0,922141, serta *Hamming Loss* terendah sebesar 0,0875. Sebaliknya, *SVM* menunjukkan hasil terendah dibandingkan algoritma lainnya. Implementasi metode ini diharapkan dapat berkontribusi pada pengembangan sistem deteksi *phishing* yang lebih andal dan akurat di masa mendatang.

Kata kunci: *deteksi phishing*, optimasi kinerja model, *machine learning*

Abstract

Phishing is a common cyber security threat in which attackers attempt to deceive users into disclosing personal information such as passwords, credit card numbers, and other sensitive data. With the rapid advancement of technology, phishing techniques have become increasingly sophisticated and harder to detect using traditional methods. Therefore, it is essential to develop detection techniques capable of identifying phishing websites with high accuracy. This study aims to optimize phishing detection performance by integrating variable correlation analysis for feature selection and applying imbalanced learning techniques to address data imbalance. The research stages include Data Collection, Data Preprocessing, and Data Exploration, which involve correlation analysis, removal of low-correlation features, and data visualization. In the Model Building and Training phase, the dataset is split into features and labels, followed by training and the application of data balancing techniques, ending with Model Evaluation. The evaluated algorithms include Logistic Regression, Naive Bayes, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Multi-Layer Perceptron, Decision Tree, Random Forest, Gradient Boosting, and CatBoost. The results show that the KNN algorithm delivers the best performance, achieving an accuracy of 91.25% and optimal scores in Precision (0.906943), Recall (0.927858), and F1-Score (0.922141), along with the lowest Hamming Loss at 0.0875. In contrast, the SVM algorithm recorded the lowest performance

 ISSN:2302-8149
e-ISSN:2540-9719

among the tested models. The implementation of this method is expected to contribute to the development of more reliable and accurate phishing detection systems in the future.

Keywords: *phishing detection, model performance optimization, machine learning*

1 Pendahuluan

Phishing adalah ancaman terhadap keamanan dunia maya yang sering terjadi, di mana para pelaku mencoba menipu pengguna untuk menyerahkan informasi pribadi seperti kata sandi, nomor kartu kredit, dan berbagai data sensitif lainnya. Dengan perkembangan teknologi, teknik phishing semakin canggih dan sulit untuk dideteksi menggunakan metode tradisional. Oleh karena itu, penting untuk mengembangkan teknik deteksi yang mampu mengidentifikasi situs phishing dengan tingkat akurasi tinggi [1]. Penelitian mengenai deteksi situs phishing umumnya berfokus pada pemanfaatan fitur URL, konten situs, dan pola perilaku pengguna. Beragam pendekatan telah diusulkan, mulai dari penerapan algoritma machine learning seperti *Naive Bayes dan Support Vector Machine* (SVM), *Decision Tree*, hingga teknik *deep learning* yang lebih kompleks. Namun, setiap pendekatan tersebut memiliki kekurangan dalam hal akurasi, kecepatan pemrosesan, serta kemampuan untuk mendeteksi serangan phishing baru dan yang belum dikenal sebelumnya *(zero-day attack)* [2].

Penelitian sebelumnya yang dilakukan oleh Mahmud et al [3] menganalisis beberapa algoritma, termasuk Random Forest, Decision Tree, dan KNN. Dari ketiga algoritma tersebut, Random Forest menunjukkan performa terbaik dalam mendeteksi situs phishing. Akan tetapi, penelitian ini masih memiliki sejumlah keterbatasan yang perlu diperhatikan. Salah satunya adalah keterbatasan jumlah dataset, di mana hanya digunakan 10.000 sampel yang terbagi menjadi dua kategori, yaitu Phishing dan Non-Phishing. Jumlah data tersebut kurang representatif, sehingga dapat berpengaruh pada penurunan akurasi dan kemampuan model untuk menggeneralisasi ketika diterapkan pada data baru.

Kedua, algoritma yang digunakan masih tergolong sedikit, yang berarti belum ada evaluasi mengenai kinerja algoritma lain yang mungkin lebih efektif dalam mendeteksi phishing. Selain itu, penelitian tersebut juga belum menerapkan teknik penyeimbangan data yang penting untuk menangani ketidakseimbangan kelas pada dataset. Tanpa adanya teknik penyeimbang data, model cenderung lebih fokus pada kelas mayoritas, yang dalam konteks ini bisa menyebabkan situs phishing tidak terdeteksi dengan baik. Selanjutnya, belum diterapkannya teknik korelasi variabel menjadi kelemahan lain, di mana analisis korelasi sangat penting untuk memahami hubungan antar fitur sehingga pemilihan fitur yang signifikan dapat membantu meningkatkan performa model.

Dengan keterbatasan-keterbatasan tersebut, Tujuan dari penelitian ini adalah untuk mengoptimasi kinerja deteksi phishing dengan mengintegrasikan analisis korelasi variabel untuk seleksi fitur dan penggunaan teknik imbalance learning guna mengatasi ketidakseimbangan data. serta melakukan evaluasi yang lebih komprehensif dengan memperhatikan aspek-aspek penting seperti peningkatan jumlah data, penggunaan algoritma yang lebih beragam, penerapan teknik penyeimbang data, analisis korelasi variabel, dan penggunaan metrik evaluasi yang lebih beragam agar dapat menghasilkan sistem deteksi situs phishing yang lebih akurat dan handal.

2 Tinjauan Literatur

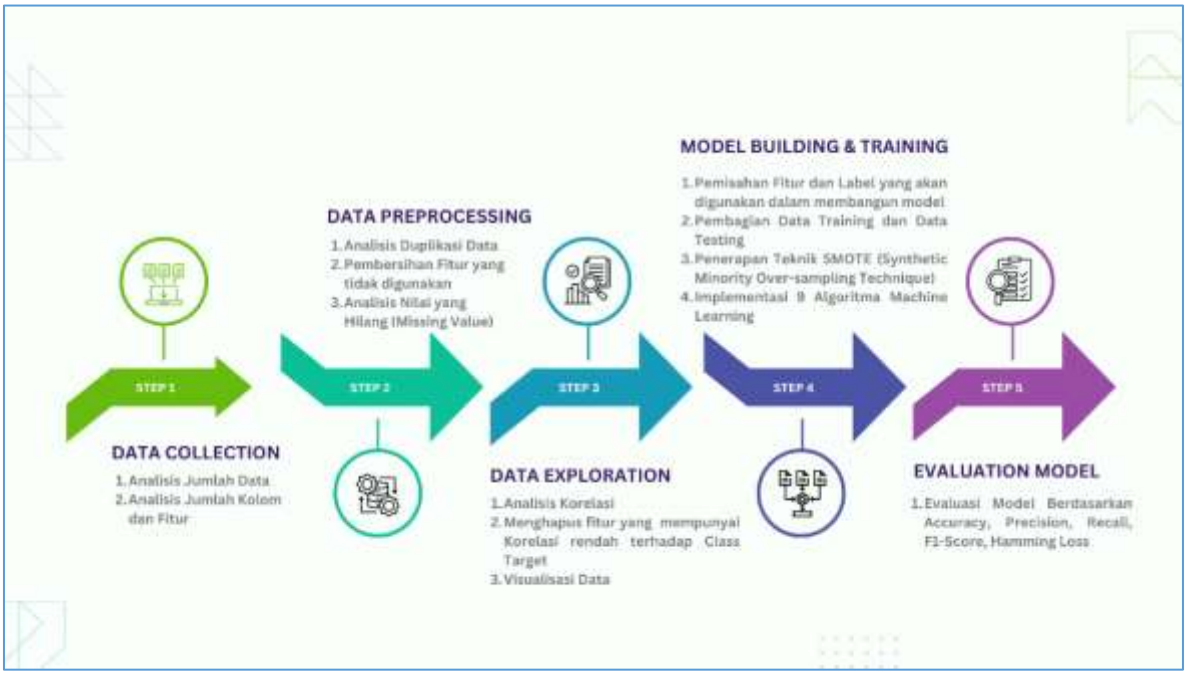
Menurut penelitian yang dilakukan oleh Irawan et al [4], dilakukan perbandingan untuk menemukan algoritma terbaik di antara *Decision Tree*, *SVM*, *Naive Bayes*, dan *Multi-Layer* Perceptron. Hasil penelitian tersebut mengungkapkan bahwa Multi-Layer Perceptron memberikan kinerja terbaik dengan akurasi mencapai 93,15%. Sementara itu, penelitian yang dilakukan oleh Mahmud et al [3] membandingkan tiga algoritma, yakni Decision Tree, Random Forest, dan KNN. Dari penelitian tersebut, *Random Forest* tercatat memiliki akurasi tertinggi, yaitu 0,834. Selain itu, penelitian yang dilakukan oleh Sunge Supriyadi [5] membandingkan beberapa algoritma, termasuk *Decision Tree, Naive Bayes, Support Vector Machine, K-Nearest Neighbor,* dan *Multi-Layer Perceptron,* di mana hasilnya menunjukkan bahwa KNN keluar sebagai algoritma terbaik. Penelitian lain juga melakukan perbandingan yang sama seperti yang dilakukan oleh Waseso dan Setyanto [6], penelitian tersebut membandingakan performa 2 algoritma *K-Nearest Neighbor* dengan *Naïve Bayes* untuk mengklasifikasikan situs web phishing. Berdasarkan hasil uji akurasi kombinasi *KNN* dan *Naïve Bayes*, dihasilkan akurasi KNN sebesar 93,44%. Hasil ini 6,25% lebih unggul dibandingkan

 ISSN:2302-8149
e-ISSN:2540-9719

menggunakan hanya satu *classifier*. Penelitian yang di lakukan oleh topcu et al [7] melakukan perbandingan 3 algoritma *Naive bayes, C4.5* dan *KNN*. dari ketiga algoritma tersebut KNN keluar sebagai algoritma dengan nilai akurasi tertinggi di atas 90%.

3 Metode Penelitian

Penelitian ini disusun melalui beberapa tahap, dimulai dari Pengumpulan Data, Prapemrosesan Data, Eksplorasi Data, Pembangunan & Pelatihan Model, hingga Evaluasi Model. Fokus penelitian ini adalah mengoptimasi kinerja deteksi phishing dengan mengintegrasikan analisis korelasi variabel untuk seleksi fitur dan penggunaan teknik *Imbalance Learning* guna mengatasi ketidakseimbangan data. Berikut ini Tahapan Detail Alur Metode penelitian yang dilakukan penulis seperti pada Gambar 1.



Gambar 1. Detail alur metode penelitian

3.1 Data Collection

Langkah pertama adalah *Data Collection*, data diperoleh dari berbagai sumber terpercaya yang menyediakan Dataset URL phishing dan non-phishing. Sumber data yang digunakan bersumber dari website kaggle *https://www.kaggle.com/datasets/eswarchandt/phishing-website-detector/data* dan website Phistank *http://data.phishtank.com/data/online-valid.csv*. Dataset yang digunakan berjumlah 10.000 data dengan 32 fitur, yang mencakup *URL phishing* dan *URL non-phishing*. Fitur tersebut antara lain *Index, UsingIP, LongURL, ShortURL, Symbol@, Redirecting//, PrefixSuffix-, SubDomains, HTTPS, DomainRegLen, Favicon, NonStdPort, HTTPSDomainURL, RequestURL, AnchorURL, LinksInScriptTags, ServerFormHandler, InfoEmail, AbnormalURL, WebsiteForwarding, StatusBarCust, DisableRightClick, UsingPopupWindow, IframeRedirection, AgeofDomain, DNSRecording, WebsiteTraffic, PageRank, GoogleIndex, LinksPointingToPage, StatsReport, class.*

Dataset yang digunakan untuk deteksi situs phishing, data diklasifikasikan menjadi dua kategori utama yaitu *Phishing* dan *Non-Phishing*. Kategori 1 mewakili URL Phishing, yang berjumlah 16.643 data, sementara kategori -1 mewakili URL Non-Phishing, dengan jumlah 13.357 data. Total keseluruhan data adalah 30.000, dengan distribusi yang sedikit lebih banyak pada URL phishing dibandingkan non-phishing.

3.2 Data Preprocessing

Tahap kedua adalah *Data Preprocessing.* Data Preprocessing adalah serangkaian proses yang bertujuan untuk menyiapkan data mentah agar dapat dimanfaatkan secara optimal dalam model

 ISSN:2302-8149
e-ISSN:2540-9719

pembelajaran mesin [8]. Proses ini melibatkan serangkaian langkah yang dirancang untuk membersihkan, mengubah, dan mengekstrak fitur penting dari data URL *Phishing dan Non-Phishing* [9], sehingga dapat meningkatkan performa model deteksi phishing yang akan diterapkan. Proses ini mencakup analisis duplikasi data, pembersihan fitur yang tidak digunakan, analisis nilai yang hilang *(Missing Value)* yang bisa mempengaruhi hasil klasifikasi.

3.3 Data Exploration

Tahap Ketiga adalah *Data Exploration,* pada tahap ini dilakukan analisis eksploratif untuk memahami struktur dan distribusi data [10]. Langkah pertama adalah Analisis Korelasi, Pada tahap analisis korelasi, terdapat dua metode yang umum digunakan untuk mengukur hubungan antara fitur-fitur dalam data, yaitu korelasi *Pearson* dan korelasi *Spearman.* Kedua metode ini menggunakan pendekatan yang berbeda dalam menentukan kekuatan dan arah hubungan antara variabel, analisis korelasi dilakukan untuk mengukur hubungan antara fitur-fitur URL dengan *Class Target (Phishing atau Non-Phishing)*[11]. Korelasi yang kuat dapat menjadi indikator bahwa fitur tersebut relevan untuk prediksi. Pemahaman yang diperoleh dari eksplorasi ini menjadi dasar dalam memilih fitur yang paling berpengaruh dalam membangun model yang lebih efektif. Langkah kedua adalah Menghapus Fitur yang mempunyai korelasi rendah terhadap *Class Target*. Tahap ketiga adalah Visualisasi Data, Visualisasi membantu mengidentifikasi pola, relasi, dan distribusi data dengan cara yang lebih intuitif. Ini meliputi histogram, scatter plot, line chart, box plot, dan lain-lain [12].

3.4 Model Building and Training

Tahap Keempat adalah *Model Building and Training,* Tahap Pertama dari *Model Building and Training* adalah memisahkan fitur dan label yang akan digunakan dalam membangun model. Selanjutnya, data dipisahkan menjadi Data Pelatihan dan Data Pengujian. Selanjutnya, data dipisahkan menjadi Data Pelatihan dan Data Pengujian. Setelah itu, teknik *SMOTE (Synthetic Minority Over-sampling Technique)* digunakan untuk mengatasi masalah ketidakseimbangan data dalam pembelajaran mesin, khususnya dalam klasifikasi. *SMOTE* bekerja dengan cara meningkatkan jumlah data pada kelas minoritas secara sintetis, tanpa melakukan penggandaan data yang ada.

Secara lebih spesifik, *SMOTE* menghasilkan sampel baru dari kelas minoritas dengan membuat data sintetis berdasarkan kombinasi linier antara titik data yang ada. *SMOTE* memilih titik-titik data minoritas, lalu menciptakan sampel baru di sepanjang garis yang menghubungkan titik-titik tersebut. Hal ini memberikan lebih banyak representasi untuk kelas minoritas, sehingga dataset yang awalnya tidak seimbang menjadi lebih seimbang. Dengan demikian, algoritma pembelajaran mesin dapat mempelajari pola dari kedua kelas secara lebih efektif dan akurat.

Keempat Implementasi Algoritma Machine Learning, penelitian yang di lakukan oleh Mahmud et al 2021 [3] menggunakan 3 algoritma klasifikasi yaitu *KNN, Decision Tree* dan *Random Forest*. di Penelitian ini 9 algoritma pembelajaran mesin diterapkan untuk membangun model klasifikasi. Algoritma yang diterapkan meliputi *Decision Tree, Random Forest, Gradient Boosting, CatBoost, Multi-Layer Perceptron, Support Vector Machine, Naive Bayes, K-Nearest Neighbors,* dan *Logistic Regression.*

3.5 Model Evaluation

Pada tahap terakhir, yaitu *Model Evaluation*, kinerja setiap algoritma dievaluasi berdasarkan beberapa metrik evaluasi. Akurasi digunakan untuk mengukur seberapa sering model membuat prediksi yang benar, sedangkan Presisi mengevaluasi seberapa tepat prediksi phishing dari semua prediksi phishing yang dilakukan oleh model. *Recall* mengukur kemampuan model dalam mendeteksi URL phishing dari keseluruhan data phishing yang ada. Untuk menyeimbangkan antara presisi dan recall, Metrik F1-score digunakan, yang merupakan rata-rata harmonis dari presisi dan recall. Berdasarkan hasil evaluasi, algoritma dengan kinerja terbaik dipilih sebagai model paling efektif untuk mendeteksi situs phishing [13]. Selain akurasi, presisi, recall, dan F1-score, *Hamming Loss* juga diterapkan untuk mengevaluasi kemampuan model dalam mendeteksi phishing, *Hamming Loss* adalah metrik yang mengukur proporsi prediksi yang salah dari keseluruhan prediksi yang dilakukan oleh model. Metrik ini mencakup kesalahan klasifikasi baik untuk prediksi positif (Phishing) maupun negatif (Non-Phishing). Dalam konteks pendeteksian phishing, *Hamming Loss* sangat berguna karena

 ISSN:2302-8149
e-ISSN:2540-9719

memberikan informasi mengenai berapa banyak kesalahan yang terjadi di antara semua sampel yang ada.

4 Hasil dan Pembahasan

Tahap Pertama dalam penelitian ini adalah Pengumpulan data atau *Data Collection*, dimana data di peroleh dari sumber yang sudah di sebutkan sebelumnya. Jumlah data yang di gunakan berjumlah 30.000 dengan 32 fitur.

```
data = pd.read_csv('/content/drive/MyDrive/AKU SKRIPSI 1/phishing dataset.csv')
data.head()
```

	Index	UsingIP	LongURL	ShortURL	Symbol@	Redirecting//	PrefixSuffix-	SubDomains	HTTPS	DomainRegLen	...	UsingPopupWindow
0	0	1	1	1	1	1	-1	0	1	-1	...	1
1	1	1	0	1	1	1	-1	-1	-1	-1	...	1
2	2	1	0	1	1	1	-1	-1	-1	1	...	1
3	3	1	0	-1	1	1	-1	1	1	-1	...	-1
4	4	-1	0	-1	1	-1	-1	1	1	-1	...	1

5 rows × 32 columns

Gambar 2. Loading dataset

```
[45]  # menampilkan jumlah data
      data.shape

(30000, 32)
```

Gambar 3. Jumlah data dan fitur

```
# jumlah data distribusi data phising dan non phising
print(data['class'].value_counts())

class
 1    16643
-1    13357
Name: count, dtype: int64
```

Gambar 4. Distribusi data phising dan non phising

Gambar 2, 3 dan 4 menjelaskan Dataset yang digunakan untuk deteksi situs phishing, data diklasifikasikan menjadi dua kategori utama yaitu *Phishing* dan *Non-Phishing*. Kategori 1 mewakili URL Phishing, yang berjumlah 16.643 data, sementara kategori -1 mewakili URL Non-Phishing, dengan jumlah 13.357 data. Total keseluruhan data adalah 30.000, dengan distribusi yang sedikit lebih banyak pada URL phishing dibandingkan non-phishing.

Tahap kedua adalah *Data Preprocessing*, Proses ini mencakup analisis duplikasi data, pembersihan fitur yang tidak digunakan, analisis nilai yang hilang *(Missing Value)*.

 ISSN:2302-8149
e-ISSN:2540-9719

```
# ANALISIS APAKAH ADA DATA YG DUPLIKAT
duplicate_rows = data[data.duplicated()]
print("Jumlah baris duplikat:", len(duplicate_rows))
if not duplicate_rows.empty:
  print("Baris duplikat:")
  print(duplicate_rows)
else:
  print("Tidak ada baris duplikat dalam DataFrame.")
```

Gambar 5. Analisis duplikasi data

Gambar 5 merupakan Analisis Duplikasi Data. Analisis Duplikasi Data adalah langkah penting dalam preprocessing data sebelum membangun model machine learning. Data duplikat dapat menyebabkan model machine learning belajar pola yang tidak akurat dan mengalami overfitting. Selain itu, data duplikat memperlambat proses training, menghabiskan sumber daya komputasi, dan bisa menunjukkan adanya kesalahan dalam pengumpulan atau pemrosesan data. Akibatnya, model sulit mempelajari pola yang sebenarnya, mengganggu kinerja secara keseluruhan.

```
# menghapus kolom 'Index' dari DataFrame 'data'.
data.drop('Index', axis=1, inplace=True)
```

Gambar 6. Menghapus kolom index

Pada Gambar 6 Kolom Index di hapus karena kolom Index kemungkinan besar merupakan Index baris yang dihasilkan secara otomatis oleh dataset atau alat pengumpulan data. Dalam analisis, Index baris umumnya tidak memberikan informasi yang berarti tentang data itu sendiri dan dapat dihapus untuk menyederhanakan model. Menghapus Index dapat meningkatkan efisiensi dan mengurangi kompleksitas pemodelan karena fitur yang tidak relevan dihilangkan.

```
# Periksa missing value pada setiap kolom
missing_values = data.isnull().sum()

# Tampilkan kolom yang memiliki missing value
print(missing_values[missing_values > 0])

# Jika tidak ada kolom dengan missing value, tampilkan pesan
if missing_values.sum() == 0:
  print("Tidak ada missing value dalam dataset.")
```

```
Series([], dtype: int64)
Tidak ada missing value dalam dataset.
```

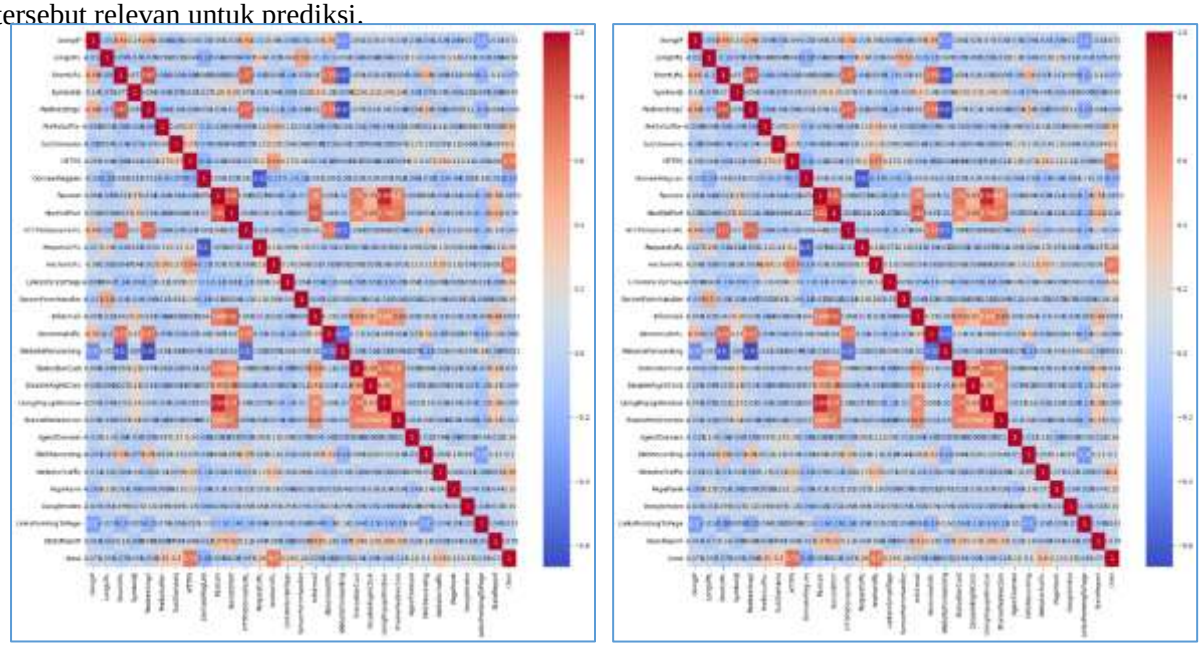
Gambar 7. Analisis missing value

Gambar 7 adalah Analisis *Missing Value*. Analisis Missing Value dilakukan untuk mendeteksi apakah terdapat nilai yang hilang atau kosong dalam dataset. Kehadiran missing value sering kali menjadi masalah dalam pemrosesan data karena dapat mempengaruhi performa model,

 ISSN:2302-8149
e-ISSN:2540-9719

mempengaruhi kualitas analisis. Dengan analisis ini diharapkan dapat menjaga kualitas data serta hasil analisis yang akurat [14].

Tahap Ketiga adalah tahap *Data Exploration*, Tahap ini mencakup Analisis Korelasi, Pembersihan Fitur yang tidak memiliki korelasi yang signifkan terhadap *Class Target*, dan Visualisasi Data. Tahap Analisis Korelasi menggunakan 2 metode yang umum yang digunakan yaitu metode *Pearson* dan *Spearman.* Analisis ini bertujuan untuk mengukur hubungan antara fitur-fitur URL dengan Class Target (Phishing atau Non-Phishing). Korelasi yang kuat menunjukkan bahwa fitur tersebut relevan untuk prediksi.



(a) (b)

Gambar 8. Analisis korelasi (a) spearman dan (b) pearson

```
# Mencari kolom yang berkorelasi dengan kolom 'class'

corr_matrix = data.corr()
correlated_columns = corr_matrix[corr_matrix['class'] > 0.6]['class'].index.
tolist()
print(correlated_columns)

['HTTPS', 'AnchorURL', 'class']
```

Gambar 9. Hasil analisis korelasi

Hasil analisis korelasi pada Gambar 8 dan 9 mengindikasikan adanya dua fitur utama yang memiliki korelasi kuat dengan variabel *Class Target*, yaitu *HTTPS* dan *AnchorURL*. Keduanya memiliki nilai korelasi diatas 0.6, yang menunjukkan bahwa fitur-fitur ini relevan untuk membedakan URL phishing dari non-phishing.

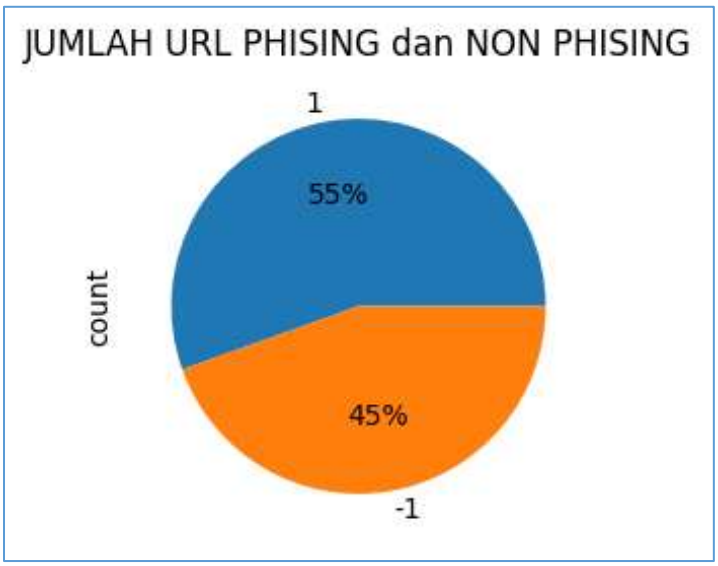
Setelah fitur yang berkorelasi tinggi didapatkan selanjutnya adalah pembersihan fitur-fitur yang memiliki korelasi rendah terhadap variabel target, sehingga hanya mempertahankan kolom yang dianggap signifikan. Langkah ini penting untuk mengurangi kompleksitas data dan meningkatkan akurasi model prediksi.

```
[17] # Menghapus semua kolom kecuali 'HTTPS', 'AnchorURL', dan 'class'

     columns_to_keep = ['HTTPS', 'AnchorURL', 'class']
     data = data[columns_to_keep]
```

Gambar 10. Pembersihan fitur

 ISSN:2302-8149
e-ISSN:2540-9719

Tahap selanjutnya adalah Visualisasi Data, Tujuan utama Visualisasi Data adalah untuk Memahami Distribusi Data.



Gambar 11. Distribusi data dalam diagram

Gambar 11 menunjukkan Distribusi URL Phishing (55%) dan Non-Phishing (45%) sebelum dilakukan *Oversampling (SMOTE).* Dataset sedikit tidak seimbang, dengan URL Phishing lebih dominan. Ketidakseimbangan ini perlu diperbaiki menggunakan *SMOTE* agar model dapat mendeteksi kedua kelas dengan lebih baik dan menghindari bias terhadap kelas yang lebih banyak.

Tahap Keempat adalah *Model Building and Training,* Tahap ini mencakup memisahkan Fitur dan Label yang akan digunakan dalam membangun model, memisahkan data menjadi Data Training dan Data Testing, Penerapan Teknik *SMOTE*, dan Implementasi Algoritma Machine Learning.

```
0d
# Memisahkan fitur dan label

X = data[['HTTPS', 'AnchorURL']]
y = data['class']
```

Gambar 12. Pemisahan fitur dan label

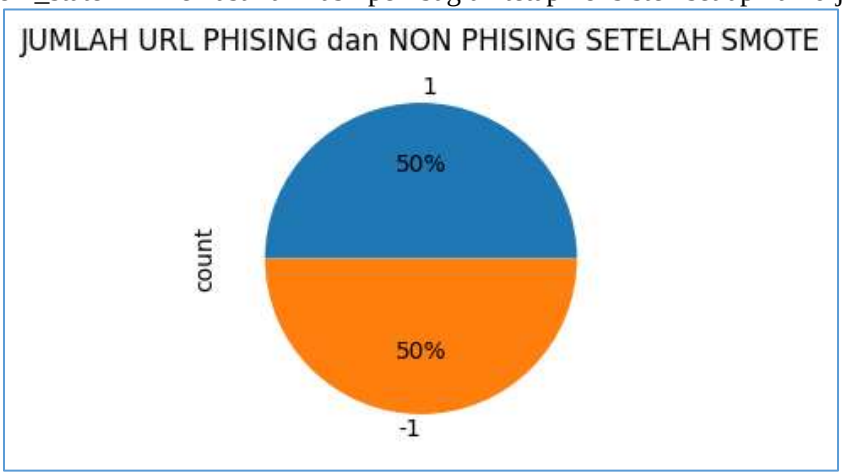
Pada Gambar 12, X adalah variabel fitur yang berisi kolom *HTTPS* dan *AnchorURL*. Kolom-kolom ini merupakan fitur yang relevan yang akan digunakan oleh model untuk mempelajari pola-pola dalam data. y adalah variabel target yang menyimpan nilai dari kolom *Class*. *Class* merupakan label yang menunjukkan apakah URL tersebut adalah Phishing (1) atau Non-Phishing (-1). Memisahkan Fitur (X) dan Label (y) diperlukan agar model dapat dilatih menggunakan fitur untuk memprediksi nilai target (class). Dengan pemisahan ini, proses pelatihan dan evaluasi model dapat dilakukan dengan lebih jelas dan terstruktur.

```
0d
# Membagi data menjadi data training dan data testing

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

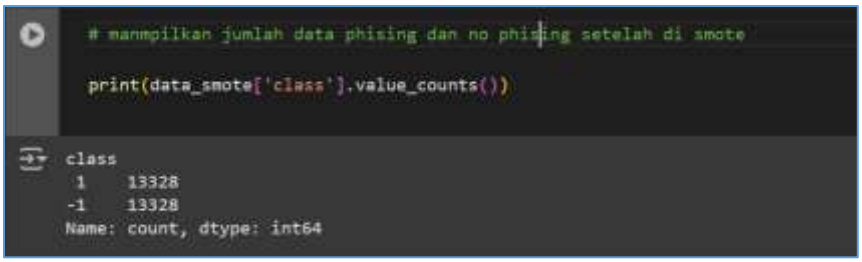
Gambar 13. Pemisahan data training dan testing

 ISSN:2302-8149
e-ISSN:2540-9719

Gambar 13 menunjukkan Dataset dibagi menjadi data training (80%) dan data testing (20%) menggunakan metode train_test_split. Data training (X_train, y_train) digunakan untuk melatih model. sedangkan data testing (X_test, y_test) digunakan untuk mengevaluasi kinerja model. Parameter random_state=42 memastikan hasil pembagian tetap konsisten setiap kali dijalankan [15].



Gambar 14. Distribusi data setelah SMOTE



Gambar 15. Data setelah SMOTE

Gambar 14 dan 15 menunjukkan Distribusi kelas dalam dataset setelah dilakukan *Oversampling* menggunakan *SMOTE*. setelah penerapan *SMOTE*, dataset kini memiliki jumlah yang sama untuk kedua kelas. Setelah penerapan Teknik *SMOTE* jumlah data akan mengalami pengurangan hal ini di karenakan data yang diproses oleh *SMOTE* adalah data training saja (X_train), bukan seluruh data. Proses train_test_split telah membagi data menjadi training dan testing, dan SMOTE hanya diterapkan pada bagian training. Data testing (X_test dan y_test) tetap ada dan tidak ikut diproses oleh SMOTE. Jadi, jumlah data akhir akan lebih sedikit daripada data awal. Keseimbangan ini akan meningkatkan kemampuan model dalam mengklasifikasikan kedua kelas dengan lebih akurat, serta mengurangi risiko *Overfitting* pada kelas yang lebih dominan.

Tahap selanjutnya adalah Implementasi Model menggunakan berbagai algoritma machine learning. Pada penelitian ini, akan digunakan 9 algoritma klasifikasi untuk mendeteksi URL phishing dan non-phishing.

```
# Membuat model Logistic Regression
model = LogisticRegression()

# Melatih model dengan data training yang telah di-SMOTE
model.fit(X_train_smote, y_train_smote)

# Memprediksi label pada data testing
y_pred = model.predict(X_test)
```

Gambar 16. Model logistik regression

 ISSN:2302-8149
e-ISSN:2540-9719

```
# Membuat model KNN
knn_model = KNeighborsClassifier(n_neighbors=5)

# Melatih model dengan data training yang telah di-SMOTE
knn_model.fit(X_train_smote, y_train_smote)

# Memprediksi label pada data testing
y_pred_knn = knn_model.predict(X_test)
```

Gambar 17. Model KNN

```
# Membuat model SVM
svm_model = SVC(kernel='linear')  | poly, sigmoid)

# Melatih model dengan data training yang telah di-SMOTE
svm_model.fit(X_train_smote, y_train_smote)

# Memprediksi label pada data testing
y_pred_svm = svm_model.predict(X_test)
```

Gambar 18. Model SVM

```
# Membuat model Naive Bayes
nb_model = GaussianNB()

# Melatih model dengan data training yang telah di-SMOTE
nb_model.fit(X_train_smote, y_train_smote)

# Memprediksi label pada data testing
y_pred_nb = nb_model.predict(X_test)
```

Gambar 19. Model naive bayes

```
# Membuat model Decision Tree
dt_model = DecisionTreeClassifier()

# Melatih model dengan data training yang telah di-SMOTE
dt_model.fit(X_train_smote, y_train_smote)

# Memprediksi label pada data testing
y_pred_dt = dt_model.predict(X_test)
```

Gambar 20. Model decision tree

```
# Membuat model Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
dapat mengubah n_estimators

# Melatih model dengan data training yang telah di-SMOTE
rf_model.fit(X_train_smote, y_train_smote)

# Memprediksi label pada data testing
y_pred_rf = rf_model.predict(X_test)
```

Gambar 21. Model random forest

 ISSN:2302-8149
e-ISSN:2540-9719

```
# Membuat model Gradient Boosting
gb_model = GradientBoostingClassifier(n_estimators=100, random_state=42)

# Melatih model dengan data training yang telah di-SMOTE
gb_model.fit(X_train_smote, y_train_smote)

# Memprediksi label pada data testing
y_pred_gb = gb_model.predict(X_test)
```

Gambar 22. Model gradient boosting

```
# Membuat model CatBoost
cb_model = CatBoostClassifier(iterations=100, learning_rate=0.1, depth=6,
random_seed=42)

# Melatih model dengan data training yang telah di-SMOTE
cb_model.fit(X_train_smote, y_train_smote, verbose=False)

# Memprediksi label pada data testing
y_pred_cb = cb_model.predict(X_test)
```

Gambar 23. Model catboost

```
# Membuat model Multi-Layer Perceptron
mlp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, random_state=42)

# Melatih model dengan data training
mlp.fit(X_train, y_train)

# Memprediksi label untuk data testing
y_pred_mlp = mlp.predict(X_test)
```

Gambar 24. Model MLP

Gambar 16-24 menjelaskan sembilan algoritma yang digunakan dalam penelitian ini. Gambar 16 menunjukkan *K-Nearest Neighbors* (KNN), yang mengklasifikasikan data berdasarkan mayoritas *k* tetangga terdekatnya. Gambar 17 merepresentasikan *Naïve Bayes*, algoritma probabilistik berbasis *Bayes' Theorem* dengan asumsi independensi antar fitur. Gambar 18 menggambarkan *Support Vector Machine* (SVM), yang mencari *hyperplane* optimal untuk memisahkan kelas data. Gambar 19 menampilkan *Multi-Layer Perceptron* (MLP), jaringan saraf tiruan dengan lapisan tersembunyi untuk menangani klasifikasi non-linear. Gambar 20 menunjukkan *CatBoost*, teknik *gradient boosting* yang dioptimalkan untuk data kategori. Gambar 21 merepresentasikan *Gradient Boosting*, metode *ensemble learning* yang meningkatkan akurasi dengan mengurangi kesalahan residu model sebelumnya. Gambar 22 memperlihatkan *Random Forest*, algoritma berbasis pohon keputusan yang menggabungkan banyak pohon untuk meningkatkan stabilitas dan mengurangi overfitting. Gambar 23 menampilkan *Decision Tree*, model berbasis aturan keputusan yang membagi dataset secara hierarkis. Terakhir, Gambar 24 menunjukkan *Logistic Regression*, metode statistik yang memanfaatkan fungsi sigmoid untuk klasifikasi biner. Kesembilan algoritma ini diuji dalam penelitian untuk menentukan metode paling optimal dalam deteksi *phishing.*

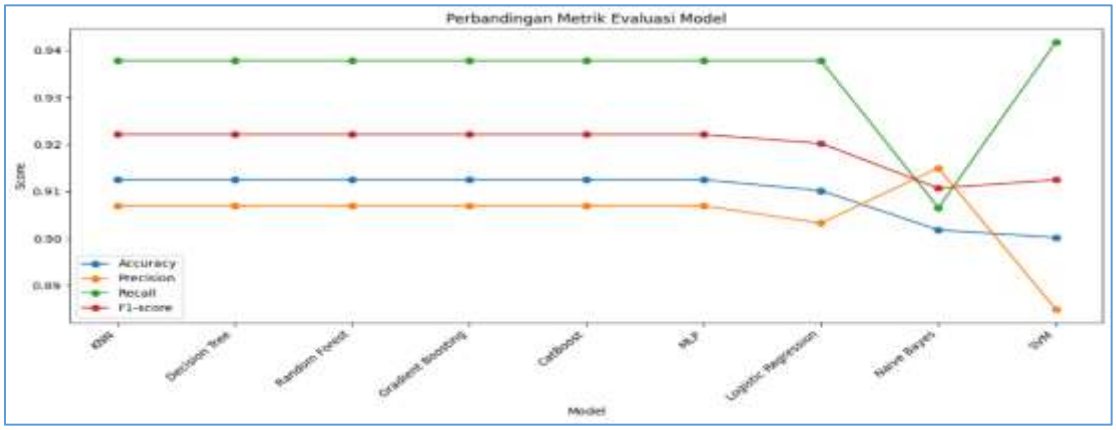
Tahap terakhir adalah Evaluasi Model, di mana kinerja setiap algoritma dievaluasi menggunakan beberapa metrik, seperti Accuracy, Precision, Recall, dan F1-Score. dan *Hamming Loss.*

 ISSN:2302-8149
e-ISSN:2540-9719

Hasil Evaluasi Semua Model:

	Model	Accuracy	Precision	Recall	F1-score	Hamming Loss
1	KNN	0.912500	0.906943	0.937858	0.922141	0.0875
4	Decision Tree	0.912500	0.906943	0.937858	0.922141	0.0875
5	Random Forest	0.912500	0.906943	0.937858	0.922141	0.0875
6	Gradient Boosting	0.912500	0.906943	0.937858	0.922141	0.0875
7	CatBoost	0.912500	0.906943	0.937858	0.922141	0.0875
8	MLP	0.912500	0.906943	0.937858	0.922141	0.0875
0	Logistic Regression	0.910167	0.903254	0.937858	0.920231	0.0898
3	Naive Bayes	0.901800	0.915000	0.906500	0.910700	0.0982
2	SVM	0.900167	0.884921	0.941780	0.912465	0.0998

Gambar 25. Hasil evaluasi model



Gambar 26. Perbandingan metrik evaluasi model

Gambar 25 dan Gambar 26 menunjukan hasil evaluasi Model, Sembilan algoritma telah diuji untuk mendeteksi URL phishing menggunakan beberapa metrik utama, yaitu *Accuracy*, *Precision*, *Recall*, *F1-Score*, dan *Hamming Loss*. Hasil analisis menunjukkan bahwa *K-Nearest Neighbors (KNN)*, *Decision Tree*, *Random Forest*, *dan Gradient Boosting*, *CatBoost*, dan *Multi-Layer Perceptron (MLP)* memiliki performa yang setara dengan akurasi sebesar 91.25%. Selain itu, keenam model ini juga mencapai nilai Precision, Recall, dan F1-Score yang sama, masing-masing sebesar 0.906943, 0.937858, dan 0.922141, serta Hamming Loss terendah, yaitu 0.0875. Hasil ini mengindikasikan bahwa keenam algoritma tersebut mampu memberikan kinerja yang seimbang dan konsisten dalam mendeteksi URL phishing maupun non-phishing.

Sementara itu, *Logistic Regression* mencatatkan hasil yang sedikit lebih rendah dengan Akurasi 91.02% dan Precision 0.903254. Meskipun performanya cukup mendekati, model ini memiliki Hamming Loss yang lebih tinggi, yaitu 0.0898, menunjukkan bahwa *Logistic Regression* membuat sedikit lebih banyak kesalahan prediksi dibandingkan dengan keenam model sebelumnya. Di sisi lain, *Naive Bayes* mencapai Precision tertinggi sebesar 0.915000 dengan Akurasi 90.18%. Namun, nilai Recall yang lebih rendah (0.906500) menyebabkan F1-Score-nya sedikit menurun menjadi 0.910700 dengan Hamming Loss 0.0982. Ini menunjukkan bahwa Naive Bayes cenderung lebih baik dalam mendeteksi URL phishing, tetapi tidak seimbang dalam menangani URL non-phishing.

Algoritma yang menunjukkan kinerja paling rendah adalah Support Vector Machine (SVM), dengan Akurasi 90.01% dan Precision 0.884921. Meskipun nilai Recall-nya cukup tinggi (0.941780), Hamming Loss sebesar 0.0998 menunjukkan bahwa SVM lebih sering membuat kesalahan prediksi dibandingkan dengan model lain.

5 Kesimpulan

Penelitian ini berhasil meningkatkan kinerja deteksi phishing dengan mengoptimalkan seleksi fitur melalui analisis korelasi variabel dan mengatasi ketidakseimbangan data menggunakan *imbalance learning*. Hasil menunjukkan bahwa algoritma *K-Nearest Neighbors* (KNN) memberikan

performa terbaik dengan akurasi 91,25%, precision 0,906943, recall 0,927858, F1-score 0,922141, dan *Hamming Loss* terendah sebesar 0,0875. Sebaliknya, *Support Vector Machine* (SVM) menunjukkan performa terendah, sehingga kurang efektif untuk deteksi phishing. Dibandingkan penelitian sebelumnya, model yang diuji dalam penelitian ini menunjukkan peningkatan signifikan dalam akurasi dan konsistensi metrik performa. Dengan demikian, pendekatan yang digunakan dapat menjadi solusi lebih efektif dalam mengembangkan sistem deteksi *phishing* yang lebih akurat dan andal.

Referensi

[1] T. A. Assegie*, "*K-Nearest Neighbor Based URL Identification Model for Phishing Attack Detection*," *IJAINN*, Vol. 1, No. 2, pp. 18–21, Apr. 2021, doi: 10.35940/ijainn.B1019.041221.

[2] Y. Muliono, M. A. Ma'ruf, and Z. M. Azzahra, "*Phishing Site Detection Classification Model using Machine Learning Approach*," *EMACS Journal*, Vol. 5, No. 2, pp. 63–67, May 2023, doi: 10.21512/emacsjournal.v5i2.9951.

[3] A. F. Mahmud and S. Wirawan, "*Phishing Website Detection using Machine Learning Classification Method*," *SISTEMASI*, Vol. 13, No. 4, p. 1368, Jul. 2024, doi: 10.32520/stmsi.v13i4.3456.

[4] A. S. Y. Irawan, N. Heryana, H. S. Hopipah, and D. Rahma, "Identifikasi *Website Phishing* dengan Perbandingan Algoritma Klasifikasi," *Syntax J. Inf.*, Vol. 10, No. 01, pp. 57–67, Jun. 2021, doi: 10.35706/syji.v10i01.5292.

[5] A. S. Sunge, "Komparasi *Machine Learning* Memprediksi *Phising* dalam Keamanan *Website*," 2022.

[6] B. M. P. Waseso and N. A. Setiyanto, "*Web Phishing Classification using Combined Machine Learning Methods*," *J. Comput. Theor. Appl.*, Vol. 1, No. 1, pp. 11–18, Aug. 2023, doi: 10.33633/jcta.v1i1.8898.

[7] A. E. Topcu, Y. I. Alzoubi, E. Elbasi, and E. Camalan, "*Social Media Zero-Day Attack Detection using TensorFlow*," *Electronics*, Vol. 12, No. 17, p. 3554, Aug. 2023, doi: 10.3390/electronics12173554.

[8] F. F. Tampinongkol, A. R. Kamila, A. C. Wardhana, A. W. Candra, and D. Revaldo, "*Implementation of Random Forest Classification and Support Vector Machine Algorithms for Phishing Link Detection*," 2024.

[9] M. Vebriani and W. Yustanti, "Klasifikasi Deteksi Link Phising DANA Kaget menggunakan Metode *Support Vector Machine* berbasis *Website*," *Journal of Informatics and Computer Science*, Vol. 3 No. 2 Juli 2024 Hal. 82-91, doi: https://doi.org/10.22303/upu.1.1.2021.01-10.

[10] A. D. Harahap, D. Juardi, and A. S. Y. Irawan, "Rancang Bangun Sistem Pendeteksi Link *Phishing* menggunakan Algoritma *Random Forest* berbasis *Web*," *JITET*, Vol. 12, No. 3, Aug. 2024, doi: 10.23960/jitet.v12i3.4858.

[11] O. Kayode-Ajala, "*Applying Machine Learning Algorithms for Detecting Phishing Websites: Applications of SVM, KNN, Decision Trees, and Random Forests*".

[12] Nor Hapiza Mohd Ariffin, Muhammad Imtiaz Mohamed Iqbal, Marina Yusoff, and Nurul Akhmal Mohd Zulkefli, "*A Study on the Best Classification Method for an Intelligent Phishing Website Detection System*," *ARASET*, Vol. 48, No. 2, pp. 197–210, Jul. 2024, doi: 10.37934/araset.48.2.197210.

[13] F. Raihan and R. Renaldy, "Efektivitas Algoritma *Artificial Intelligence* dalam Melawan Serangan Zero-Day," Vol. 2, 2024.

[14] L. Lakshmi, M. P. Reddy, C. Santhaiah, and U. J. Reddy, "*Smart Phishing Detection in Web Pages using Supervised Deep Learning Classification and Optimization Technique ADAM*," *Wireless Pers Commun*, Vol. 118, No. 4, pp. 3549–3564, Jun. 2021, doi: 10.1007/s11277-021-08196-7.

[15] D. Wahyudi, M. Niswar, and A. A. P. Alimuddin, "*Website Phising Detection Application using Support Vector Machine (SVM)*," *JITU*, Vol. 5, No. 1, pp. 18–24, Jun. 2022, doi: 10.56873/jitu.5.1.4836.