e-ISSN 2962-6129
Seminar Nasional Mahasiswa Ilmu Komputer dan Aplikasinya (SENAMIKA)
Jakarta-Indonesia, 20 Agustus 2022

Implementasi Ekstensi Google Chrome Dalam Mendeteksi Situs Web *Phishing* Menggunakan Algoritma *Random Forest*

Muhamad Abdul Ghanni Al Ghifari[1], Bayu Hananto, S.Kom., M.Kom.[2*], Bambang Tri Wahyono, S.Komp., M.Si.[3]
Program Studi Informatika, Universitas Pembangunan Nasional Veteran Jakarta
Jl. RS. Fatmawati Raya, Pd. Labu, Kec. Cilandak, Kota Depok, Jawa Barat 12450
[1]muhamadabdul@upnvj.ac.id, [2]bayuhananto@upnvj.ac.id, [3]bambang.triwahyono@upnvj.ac.id

Abstrak. Di zaman teknologi ini penggunaan internet mengubah kehidupan sehari-hari masyarakat menjadi lebih mudah. Namun dengan segala manfaat internet terdapat oknum tertentu yang melakukan berbagai macam kejahatan yaitu salah satunya membuat situs web *phishing*. Karena hal tersebut penelitian ini mengusulkan pendeteksian situs web *phishing*. Untuk mendeteksi situs web *phishing* pendekatan yang paling umum digunakan yaitu metode *blacklist* dan *whitelist*, namun metode ini mempunyai beberapa kekurangan. Oleh sebab itu penelitian ini bertujuan menggunakan pendekatan *machine learning* yaitu metode *Random Forest*, dengan mengimplementasikannya ke dalam ekstensi peramban seperti Google Chrome. Ekstensi peramban dibangun tanpa bergantung kepada *web service*, supaya pendeteksian dapat lebih cepat. Hasil evaluasi model klasifikasi mempunyai hasil akurasi 90,2%, *recall* 88,8% dan presisi 88,8%. Ekstensi peramban dilakukan evaluasi kinerja menggunakan data baru dengan akurasi 88%, *recall* 84% dan presisi 91,3%.

Kata kunci: Deteksi *Phishing*, *Machine Learning*, Klasifikasi, *Random Forest*, Ekstensi Peramban.

1. PENDAHULUAN

Di zaman teknologi ini penggunaan internet merupakan suatu hal yang lumrah bahkan vital di kehidupan keseharian masyarakat. Dengan bertambahnya kebutuhan akan informasi, internet membantu kehidupan menjadi lebih mudah dalam berkomunikasi, edukasi, berbelanja dan berbisnis. Hampir segala aktivitas maupun transaksi sekarang ini dapat dilakukan dan diakses secara daring tanpa mengenal tempat dengan bantuan internet.

Namun, dengan segala manfaatnya internet dapat digunakan oleh oknum tertentu untuk melakukan berbagai macam kejahatan yaitu salah satunya membuat situs web *phishing* sebagai alat bantu dalam melancarkan penipuan, mereka menargetkan para pengguna internet yang masih awam atau lalai terhadap keamanan bertransaksi di dunia maya. Data pribadi seperti ID pengguna, kata sandi, bahkan informasi finansial seperti akun bank dan data kartu kredit merupakan hal yang diincar para penjahat internet ini.

Menurut Phishing.org [1] *phishing* adalah *cybercrime* yang berusaha memancing informasi penting atau rahasia seperti detail perbankan atau kartu kredit serta kata sandi dari pengguna yang biasanya dikirim melalui *email*, telepon atau teks pesan yang dilakukan dengan menyamar menjadi institusi atau individu dan membuat situs web palsu dengan sedemikian rupa sehingga menyerupai situs web yang autentik dan sah.

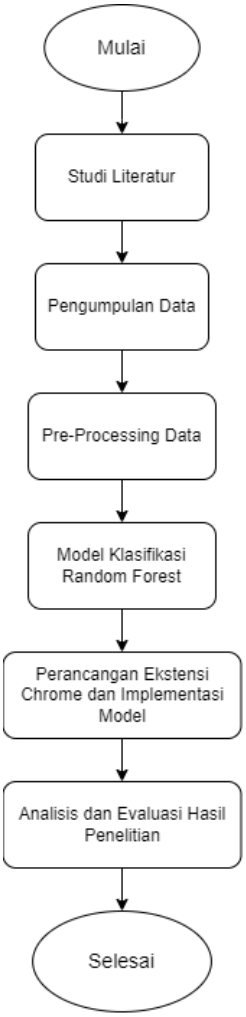
Menurut laporan transparansi Google [2] serangan *phishing* meningkat tajam hingga 4-5 kali lipat saat terjadinya pandemi COVID-19. Situs web seperti *e-commerce*, perbankan atau yang baru-baru ini situs web dompet digital untuk *cryptocurrency* kerap dijadikan sasaran *phishing*. Situs web berbasis *social media* juga tidak luput dari para penjahat internet untuk mengambil akun para korban. Terlepas dari pencurian data, situs web *phishing* juga dapat digunakan untuk melakukan menyebarkan virus atau *malware* dengan mengatasnamakan situs web asli.

Dilihat dari bahaya dan maraknya kejahatan *phishing* para peneliti mencoba untuk mengatasi hal tersebut dengan berbagai metode seperti pendeteksian situs web *phishing*. Untuk mendeteksi situs web *phishing* pendekatan yang paling umum digunakan yaitu metode *blacklist* dan *whitelist*, metode ini menggunakan

database yang berisi kumpulan situs web yang sudah diklasifikasikan sebagai situs web *phishing* atau bukan. Pendekatan ini mempunyai kelemahan yaitu cakupannya yang kurang luas, karena tentunya tidak semua URL atau URL yang baru dibuat langsung ada pada *database* tersebut. [3] Oleh karena itu untuk mengatasi hal tersebut penulis menggunakan pendekatan *machine learning* dengan membuat model yang dapat membedakan situs web *phishing* atau situs web yang autentik dan sah secara langsung. Berdasarkan penelitian terdahulu algoritma *machine learning* yang baik dalam mendeteksi situs web *phishing* yaitu *Random Forest* sehingga penulis menggunakan algoritma tersebut dalam penelitian ini.

Salah satu cara agar metode ini bisa dimanfaatkan oleh pengguna yaitu dengan mengimplementasikannya ke dalam ekstensi peramban seperti Google Chrome. Cara tersebut akan memperingatkan secara langsung ketika pengguna mengunjungi situs web *phishing*. Sistem pendeteksi situs web *phishing* dengan pendekatan *machine learning* yang sudah ada saat ini yaitu dengan cara mengirim URL ke server untuk diklasifikasikan dan hasilnya akan dikembalikan lagi. Dengan pendekatan tersebut tentunya privasi pengguna terganggu (mengirim URL situs web apa yang dikunjungi) dan juga deteksi akan terhambat oleh jaringan internet yang mana mempunyai kemungkinan gagal dalam memperingatkan pengguna di waktu yang tepat. Karena pentingnya masalah privasi dan keamanan, penulis menggunakan implementasi ekstensi Google Chrome yang bisa mengklasifikasikan situs web *phishing* menggunakan algoritma *Random Forest* secara langsung tanpa bergantung dengan server dan *web service*.

2. METODOLOGI PENELITIAN



Gambar 1. Metode Penelitian

2.1 Studi Literatur

Studi literatur dilakukan untuk mengetahui berbagai teori-teori yang relevan tentang penelitian yang diangkat sebagai bahan referensi dalam pembahasan hasil penelitian. Ini dilakukan dengan membaca beberapa jurnal, artikel, buku, maupun referensi lain yang terkait dengan penelitian ini.

2.2 Pengumpulan Data

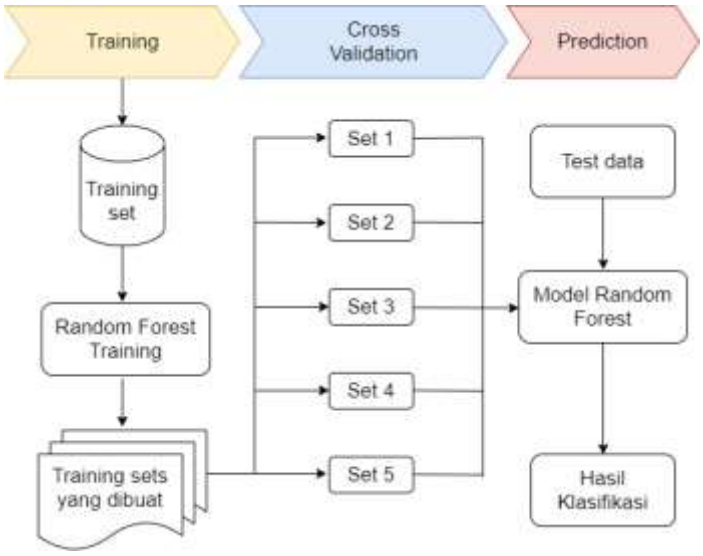
Dataset yang digunakan dalam penelitian ini berasal dari UCI Machine Learning Repository yang sudah diriset dengan baik dan menjadi acuan bagi para peneliti lainnya dalam pendeteksian *phishing*.[4] *Dataset* itu berisi 11055 data yang terdiri dari 6157 situs web *phishing* dan 4898 situs web asli. Setiap situs web pada *dataset* tersebut mempunyai 30 fitur dan setiap fitur itu mempunyai suatu aturan (*rule*) yang dapat menentukan suatu situs web *phishing* atau bukan.

2.3 Pre-processing Data

Sebelum dilakukan pemodelan dilakukan *pre-processing* data. Pada *dataset* ini dilakukan seleksi fitur secara manual. Setiap fitur dianalisis dan dipelajari bagaimana mereka dapat diekstraksi dari URL dan halaman situs web. Penulis men-*drop* 14 fitur dari 30. Fitur yang diseleksi kebanyakan fitur yang bergantung pada *web service* contohnya fitur usia domain yang mana bila mengekstrak fitur ini harus mengecek usia situs web menggunakan *third-party* yang kemungkinan mengakibatkan terjadinya *delay* sehingga pendeteksian menjadi lebih lama oleh karena itu dipilih fitur yang bisa diekstraksi secara *client-based* saja sehingga pendeteksian bisa lebih cepat.

2.4 Model Klasifikasi Random Forest

Berdasarkan studi literatur yang telah dilakukan mengenai deteksi situs web *phishing* dengan pendekatan *machine learning*, beberapa *classifier* sudah dibandingkan hasilnya, dimana *classifier* dengan kinerja deteksi terbaik yang didapatkan yaitu *Random Forest* dibanding algoritma lain. Oleh karena itu *Random Forest* digunakan pada penelitian ini. Model *Random Forest* ini dilatih pada *dataset* menggunakan *library* python yaitu Scikit-Learn [5].



Gambar 1. Deskripsi Tahapan Dari Pembuatan Model Random Forest

Berikut merupakan tahapan yang dilakukan dalam klasifikasi dan pembuatan model menggunakan metode *Random Forest* :

1. Tahapan awal yang dilakukan yaitu memasukan *dataset* ke IDE yang digunakan.
2. Membagi *dataset* menjadi dua yaitu *training* set dan *test* set dengan jumlah *training* set sebesar 70% dan *test* set 30%.
3. Selanjutnya membuat model menggunakan klasifikasi *Random Forest* menggunakan *training* set serta melakukan *cross-validation* dan melakukan *tuning* parameter untuk menentukan jumlah pohon yang optimum.
4. Setelah dilakukan klasifikasi dan telah mendapatkan model maka selanjutnya mengevaluasi hasil tersebut menggunakan *confusion matrix* untuk diketahui akurasi, *recall* dan presisi.
5. Jika hasil memuaskan maka akan dilakukan proses selanjutnya bila tidak akan dilakukan kembali tahapan

sebelumnya sampai hasil yang didapatkan terbaik.

2.5 Kinerja Model

Model klasifikasi yang telah dibuat akan diukur kinerjanya, Untuk mengevaluasi kinerjanya, pada penelitian ini menggunakan :

1. Akurasi : Ini adalah rasio jumlah prediksi yang benar dari total input sampel. Karena tujuan penelitian harus diklasifikasikan dengan akurasi yang tinggi, maka digunakanlah metrik ini.
   Perumusan Akurasi :

$$\frac{TP + TN}{TP + FP + TN + FN}$$

2. Recall : Merupakan rasio prediksi aktual positif di dalam data populasi yang telah diuji. Karena situs web yang ingin diprediksi positif mungkin, maka *recall* yang tinggi merupakan metrik yang diharapkan
   Rumus *recall* :

$$\frac{TP}{TP + FN}$$

3. Presisi : Merupakan proporsi kasus yang diprediksi positif yang juga benar positif pada data yang sebenarnya. Dalam kasus ini adalah bagian dari URL yang diklasifikasikan dengan benar sebagai *phishing* yang sebenarnya *phishing*.
   Rumus presisi :

$$\frac{TP}{TP + FP}$$

4. Confusion matrix : Merupakan suatu metode yang digunakan untuk melakukan perhitungan akurasi. Hasil dari pengelompokan data yang telah diprediksi oleh model merupakan informasi utama dalam *confusion matrix*. Hasil kinerja dari model akan ditampilkan dalam bentuk matriks.[6] Berikut merupakan bentuk dan model perhitungan *Confusion matrix.*

Tabel 1. *Confusion Matrix* untuk Dua Kelas

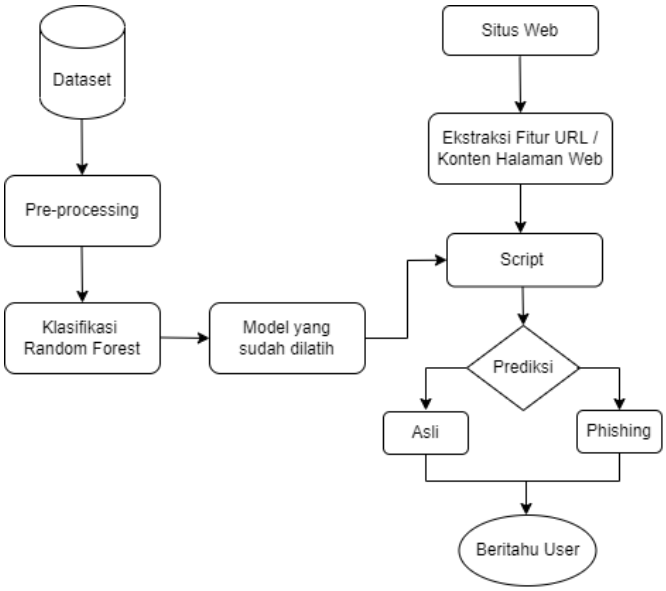
		True Values	
		Positive	*Negative*
Prediction	*Positive*	*True Positive (TP)*	*False Positive (FP)*
	Negative	*False Negative (FN)*	*True Negative (TN)*

True Positive : merupakan data positif dan diprediksi benar
True Negative : merupakan data negatif dan diprediksi benar.
False Positive : merupakan data negatif tetapi diprediksi untuk sebagai data positif.
False Negative : merupakan data positif tetapi diprediksi untuk sebagai data negatif.

2.6 Ekstensi Chrome dan Implementasi Model

Ekstensi ini dikembangkan menggunakan JavaScript. Implementasi ini bertujuan untuk membangun ekstensi peramban dengan pendekatan teknik *machine learning* untuk deteksi situs web *phishing* menggunakan model *Random Forest* yang sudah dilatih sebelumnya. Dibandingkan dengan deteksi *phishing* dengan proteksi *default* dari *Google Safe Browsing* (GSB) dengan rata-rata akurasinya yaitu ~45% saja.[7] Penulis berharap ekstensi yang akan dibangun ini dapat menghasilkan akurasi tinggi dan waktu pendeteksian yang cepat.

2.6.1 Sistem Arsitektur



Gambar 3. Diagram Sistem Arsitektur Ekstensi Peramban

Dataset sebelumnya dilakukan preproses yaitu seleksi fitur yang mana 16 fitur dipilih dari 30 jumlah fitur, fitur yang dipilih kemudian dipisahkan untuk pelatihan dan pengujian lalu dilakukan pelatihan model. Model *random forest* yang sudah dilatih akan diubah ke format *file* JSON. Model yang sudah berbentuk JSON dan fitur yang sudah diekstraksi dari URL atau halaman situs web tersebut akan diproses menggunakan JavaScript pada ekstensi peramban untuk mengklasifikasikan situs web yang sedang dimuat di tab yang aktif tanpa menggunakan server atau web *service*. Hasil dari proses itulah yang menentukan suatu situs web dapat dikatakan situs *phishing* atau bukan yang kemudian peringatan akan ditampilkan jika situs web diklasifikasikan sebagai *phishing* kepada pengguna.

2.6.2 Ekstraksi Fitur

Pada penelitian ini ekstraksi fitur dilakukan untuk mengekstrak 16 fitur yang ada pada URL dan halaman web secara langsung saat halaman sedang dimuat. Untuk mengekstrak fitur dibuat fungsi pada setiap fitur yang mau diekstraksi menggunakan JavaScript agar dapat mengakses fitur-fitur pada halaman web. Skrip tersebut secara otomatis berjalan saat memuat di setiap halaman.

Fitur yang sudah diekstraksi akan disimpan kedalam *array* dan selanjutnya siap digunakan untuk pengklasifikasian atau prediksi melalui algoritma *Random Forest*.

2.6.3 Format Model untuk Diklasifikasi

Model yang sudah di-*training* akan diubah ke bentuk file format JSON yang disimpan secara lokal agar bisa digunakan dan diklasifikasikan menggunakan algoritma *Random Forest* melalui *JavaScript*. JSON atau kepanjangannya *JavaScript Object Notation* adalah format berbasis teks standar yang menampilkan data terstruktur berdasarkan sintaks objek JavaScript. Biasanya digunakan untuk mentransmisikan data dalam aplikasi web (misalnya, mengirim beberapa data dari server ke klien, sehingga dapat ditampilkan di halaman web, atau menerima input pengguna dan mengirimkannya kembali ke server). [8]

Karena JSON struktur hierarkinya berbentuk *tree* jadi untuk implementasi model klasifikasi menggunakan *Random Forest* menjadi lebih mudah. Model dalam berbentuk JSON berisi kumpulan *Decision Tree* beserta nilai *threshold* pada setiap node pada model tersebut yang disimpan secara lokal agar bisa langsung digunakan untuk prediksi.

2.7 Analisis dan Evaluasi Hasil Penelitian

Evaluasi hasil penelitian yaitu ekstensi peramban akan dilakukan dengan menggunakan data baru untuk melihat kinerjanya dalam memprediksi situs web menggunakan *confusion matrix* dan dianalisis apakah hasil yang didapatkan dengan data baru tersebut baik atau tidak.

3. HASIL DAN PEMBAHASAN

3.1 Data

Data yang digunakan dalam penelitian ini merupakan data yang didapatkan dari arsip PhishTank dan arsip MillerSmiles yaitu *database* dari kumpulan situs web *phishing* yang sebelumnya sudah di lakukan pengumpulan data menjadi *dataset* dan diterbitkan secara publik di *UCI Machine Learning Repository. Dataset* berisi 11055 URL yang terdiri dari 6157 situs web *phishing* dan 4898 situs web asli. Setiap URL tersebut mempunyai 30 fitur dan setiap fitur itu mempunyai suatu aturan (*Rule*) yang dapat menentukan suatu situs *phishing* atau bukan.

3.2 Pre-processing Data

Data yang ada dilakukan tahap preproses data sebelum digunakan sebagai model pada ekstensi yang dibuat. Pada tahap ini data dilakukan seleksi fitur secara manual untuk menghilangkan beberapa fitur yang tidak digunakan. Untuk penelitian ini dipilih 16 fitur dari 30. Fitur dipilih kebanyakan berdasarkan apakah fitur tersebut bisa diekstraksi dari sisi *client* saja tanpa bergantung *web service* atau pada pihak ketiga. Fitur yang tidak digunakan salah satunya seperti fitur usia domain dimana untuk mengecek usia domain harus dilakukan menggunakan pihak ketiga yang mana prediksi situs web yang dikunjungi membutuhkan waktu yang lebih lama. Oleh karena itu fitur tersebut tidak digunakan agar kemampuan dalam mendeteksi situs web *phishing* bisa lebih cepat.

Tabel 2. Fitur yang Digunakan Setelah Seleksi Fitur

No.	Nama Fitur
1	URL berbentuk IP address
2	URL panjang
3	URL dipendekkan
4	Simbol @ pada URL
5	Terdapat "//" pada URL
6	Terdapat "-" pada URL
7	Jumlah *subdomain* pada URL
8	Favicon
9	Terdapat "https" pada nama URL
10	*Request* URL
11	URL *anchor tag*
12	*Tag script*
13	*Server Form Handler*
14	Penggunaan mailto
15	Kostumisasi Status *bar*
16	Penggunaan IFrame

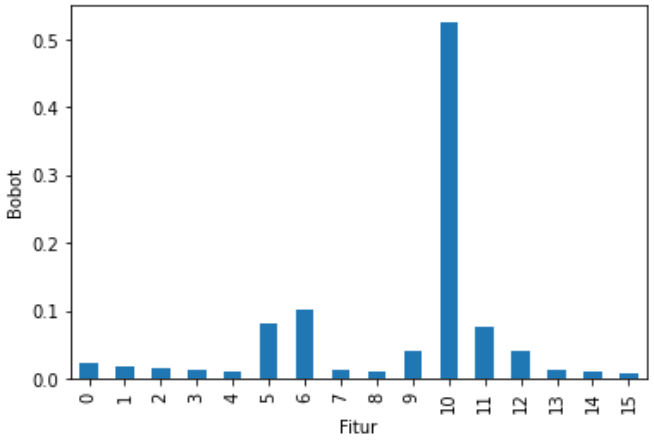
3.3 Pelatihan Model

Pada proses pelatihan model, metode yang digunakan adalah algoritma *Random Forest*. Sebelum proses dimulai, pembagian data dilakukan untuk membagi data menjadi data latih dan data uji. Data latih akan digunakan oleh algoritma klasifikasi untuk menciptakan sebuah model klasifikasi, sedangkan data uji akan digunakan untuk menguji performa dan kebenaran model dalam melakukan klasifikasi. Pembagian data yang digunakan pada penelitian ini yaitu dengan membagi data dari data total menjadi 70% untuk data latih dan 30% untuk data uji.

Tabel 1. **Jumlah data setelah pembagian data**

Data Latih	**Data Uji**	**Total Data**
70%	30%	100%
7739	3316	11055

Untuk menentukan parameter terbaik dan melakukan *cross-validation*, digunakan *grid search* yaitu GridSearchCV() dan RandomForestClassifier() sebagai estimatornya menggunakan *library* Scikit-Learn [9]pada Python dengan jumlah *cross-validation* 5 kali. Didapatkan dari melalui *tuning* parameter bahwa 80 adalah jumlah pohon optimal untuk *dataset* ini.

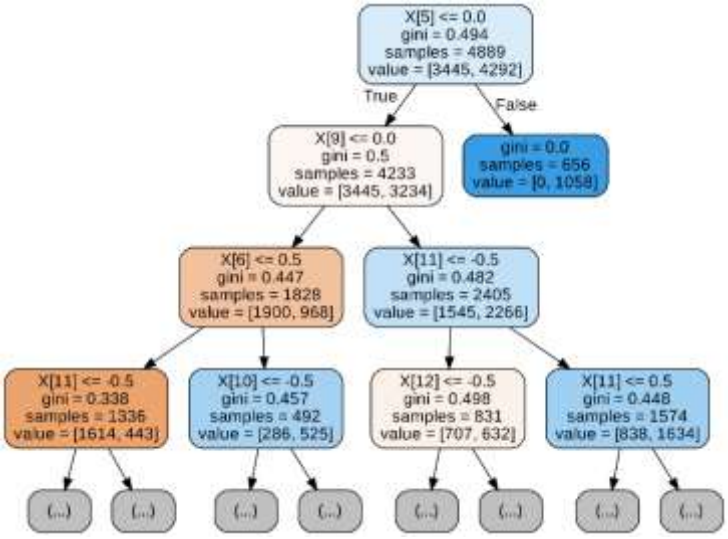


Gambar 4. Grafik Bar dari Feature Important

Berikut merupakan langkah-langkah dalam melakukan pembuatan model menggunakan metode *Random Forest* di-*library scikit-learn*.

1. Tahapan pertama melakukan *bootstrapping* data dimana sampel diambil secara acak berisi data maupun fitur dan dibuat *subset* data dari *dataset* dengan pengembalian (*replacement*).

2. Membuat setiap individu *Decision Tree* berdasarkan subset data yang diambil dari *dataset* yang sudah di-*bootstrap*. Berikut merupakan visualisasi sebagian dari salah satu *tree* yang dibuat pada model. Karena keterbatasan tempat tidak bisa menampilkan seluruh *tree*.



Gambar 5. Visualisasi Pohon pada Salah Satu Decision Tree

3. Mengulang langkah (1) dan (2) sebanyak jumlah pohon pada *Random Forest* yang dibuat.

4. Untuk memprediksi akan dilakukan *majority voting* dari seluruh *Decision Tree* yang dibuat untuk menentukan output akhiran.

3.4 Evaluasi Model

Selanjutnya untuk mengukur kinerja dari model yang telah dibuat maka digunakan teknik *confusion matrix* dengan bantuan *library* Scikit-Learn [10]. Pada pemrograman Python. Dengan memanfaatkan fungsi confusion_matrix()

didapatkan perhitungan *confusion matrix* pada model dan mengembalikan hasil dengan bentuk array. Perhitungan yang digunakan yaitu akurasi, *recall* dan presisi. Berikut *confusion matrix* pada *dataset* menggunakan metode *Random Forest*:

Tabel 4. Tabel Confusion Matrix

		True Values	
		Positive	*Negative*
Prediction	*Positive*	1293 (TP)	163 (FP)
	Negative	162 (FN)	1699 (TN)

1. Akurasi :

$$\frac{1293 + 1699}{1293 + 163 + 1699 + 162} = 0{,}902 = 90{,}2\%$$

2. *Recall* :

$$\frac{1293}{1293 + 162} = 0{,}888 = 88{,}8\%$$

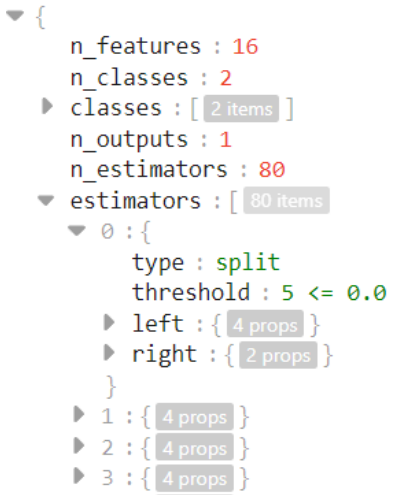
3. Presisi :

$$\frac{1293}{1293 + 163} = 0{,}888 = 88{,}8\%$$

Meskipun beberapa fitur dari *dataset* tidak digunakan tetapi dari evaluasi model ini didapatkan hasil yang cukup baik untuk memprediksi situs *web phishing* sehingga model ini bisa digunakan ke tahap selanjutnya yaitu mengimplementasikannya ke ekstensi peramban.

3.5 Mengubah Model ke Format JSON

Model yang sudah dilatih diubah kebentuk JSON agar model tersebut bisa digunakan untuk prediksi menggunakan JavaScript. JSON tersebut berisi jumlah fitur, jumlah kelas, jumlah *output*, jumlah estimator (*tree*), estimator yaitu *Decision Tree* yang mempunyai *threshold* pada setiap *node*. Atribut-atribut yang ada diambil dari model yang sudah dilatih menggunakan bantuan *library* tree pada Sklearn dan JSON untuk men-*dump file*.



Gambar 6. Bentuk Sebagian Isi Model Setelah Diubah ke JSON

3.6 Ekstensi Chrome dan Implementasi Model

Pembangunan ekstensi peramban dapat mendeteksi situs web *phishing* secara *real time* tanpa menggunakan server atau *client based* saja sehingga pendeteksian bisa dilakukan secara cepat dan tidak tergantung pada koneksi internet. Pada ekstensi yang dibuat ini ada tiga bagian utama di dalamnya yaitu *background, content* dan *popup*. Tahapannya pada proses ini yaitu ekstraksi fitur, klasifikasi web dan evaluasi kinerja.

3.7 Ekstraksi Fitur pada Situs Web

Ekstraksi Fitur diperlukan untuk mendapatkan data pada halaman web yang mana data tersebut digunakan untuk klasifikasi apakah website tersebut merupakan situs *phishing* atau bukan. *Content*.js merupakan bagian dalam ekstensi ini yang berisi fungsi-fungsi untuk mengekstrak data sesuai fitur yang diperlukan untuk klasifikasi. Fitur pada data ini bisa dikatakan merupakan ciri-ciri situs web *phishing*. Bila suatu ciri tersebut terdapat pada website maka fungsi yang ada akan mengembalikan nilai 1 ("Tidak"), jika tidak akan mengembalikan nilai -1 dan 0 yang masing-masing mempunyai arti yaitu "Ya" dan "Mungkin". Beberapa fitur mempunyai aturan yang mana bila data yang didapatkan dari suatu situs web berada dalam kategori antara ciri *phishing* atau tidak contohnya fitur URL panjang yaitu jika panjang URL mempunyai kurang dari 54 karakter bisa dikatakan "Ya", jika panjang URL mempunyai lebih dari atau sama dengan 75 karakter maka akan diberi nilai 1 dan jika URL berada diantara sama dengan atau lebih dari 54 dan 74 maka akan diberi nilai 0 ("Mungkin") pada data. Untuk lebih jelasnya berikut merupakan tahapan yang dilakukan diproses ini.

1. Mengunjungi situs web dan ekstensi sudah terpasang.
2. Mengambil atau mengekstraksi fitur-fitur melalui URL dan DOM halaman situs web yang dikunjungi dengan fungsi-fungsi di *Content*.js dengan beberapa *rule* yang sudah ditentukan.
3. Setelah fitur diekstraksi dari situs web, simpan hasil tersebut dalam array dan data tersebut siap untuk dilakukan prediksi.

Tabel 2. Nilai fitur-fitur setelah diekstraksi pada suatu situs web

No.	Nama Fitur	Nilai
1	URL Berbentuk IP address	-1
2	URL Panjang	1
3	URL Dipendekkan	-1
4	Simbol @ pada URL	-1
5	Terdapat "//" pada URL	1
6	Terdapat "-" pada URL	-1
7	*Subdomain* pada URL	-1
8	Favicon	1
9	Terdapat HTTPS pada nama URL	-1
10	*Request* URL	0
11	URL *anchor tag*	0
12	*Tag script*	1
13	*Server Form Handler*	1
14	Penggunaan mailto	-1
15	Kostumisasi Status *bar*	-1
16	Penggunaan IFrame	1

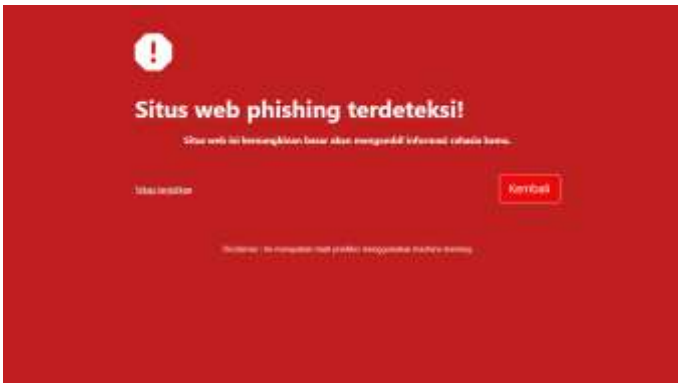
3.8 Klasifikasi Situs Web

Setelah fitur sudah diekstraksi maka tahap selanjutnya yaitu menggunakan data tersebut untuk melakukan klasifikasi atau prediksi apakah situs web yang dikunjungi merupakan situs web *phishing* atau tidak. Karena di JavaScript tidak ada *library Random Forest* seperti pada python maka dibuat fungsi khusus untuk melakukan prediksi sesuai dengan algoritma

tersebut. Fungsi tersebut menerima nilai hasil ekstraksi fitur lalu membandingkannya ke *threshold* dari setiap *node Decision Tree* yang ada pada model *Random Forest* dan *output*-nya berdasarkan nilai dari *leaf node.* Setelah hasil prediksi dari seluruh *Decision Tree* didapatkan maka dilakukan *voting* terbanyak untuk menentukan suatu situs web tersebut *phishing* atau bukan.

3.9 Tampilan Ekstensi Chrome

Gambar 7. Tampilan Popup Ekstensi



Gambar 8. Tampilan bila mengunjungi situs *phishing*

3.10 Hasil Kinerja Ekstensi Chrome

Ekstensi chrome yang sudah dibuat akan diketahui kinerja klasifikasinya dalam menentukan situs web *phishing* atau bukan dengan mengklasifikasikan 100 data baru yaitu situs web yang berisi 50 situs web asli dan 50 situs web *phishing* yang didapatkan dari PhishTank.org [11] dan RankRanger.com [12]. Berikut hasilnya

Tabel 6. Tabel *Confusion matrix*

		True Values	
		Positive	*Negative*
Prediction	*Positive*	42 (TP)	4 (FP)
	Negative	8 (FN)	46 (TN)

1. Akurasi :

$$\frac{42 + 46}{100} = 0.88 = 88\%$$

2. Recall :

$$\frac{42}{42 + 8} = 0{,}84 = 84\%$$

3. Presisi :

$$\frac{42}{42 + 4} = 0{,}913 = 91{,}3\%$$

Dari hasil diatas dapat diketahui bahwa kinerja ekstensi yang sudah dibuat menggunakan klasifikasi *Random Forest* dalam mendeteksi situs web *phishing* dengan akurasi 88%, *recall* 84% dan presisi 91,3% yang hasil tersebut sedikit lebih rendah dibanding hasil dari model aslinya namun masih cukup baik.

4. KESIMPULAN DAN SARAN

Berdasarkan penelitian yang telah dilakukan, maka dapat diambil kesimpulan bahwa model menggunakan algoritma *Random Forest* untuk pendeteksian situs web *phishing* didapatkan hasil akurasi 88%, *recall* 84% dan presisi 91,3%. Model yang sudah dibangun diimplementasikan ke ekstensi peramban Google Chrome dengan mengubahnya ke bentuk format JSON. Pengujian ekstensi Google Chrome dilakukan menggunakan 100 data baru yang berisi 50 situs web *phishing* dan 50 situs web asli. Hasil kinerja dari ekstensi yang dibuat menggunakan model algoritma *Random Forest* mempunyai akurasi 88%, *recall* 84% dan presisi 91,3%.

Berdasarkan penelitian yang telah dilakukan maka penulis menyarankan menggunakan *dataset* yang mempunyai fitur dan jumlah data yang lebih banyak agar hasil klasifikasi dapat lebih akurat dan mencoba algoritma *machine learning* lainnya yang bisa dipadukan dengan pengambilan ekstraksi fitur seperti contoh menggunakan citra dari tampilan suatu situs web agar diketahui ciri suatu situs *web phishing* tidak hanya dari URL atau DOM saja.

5. DAFTAR PUSTAKA

[1] "What Is Phishing?" https://www.phishing.org/what-is-phishing (accessed Nov. 17, 2022).

[2] "Google Safe Browsing – Google Transparency Report." https://transparencyreport.google.com/safe-browsing/overview (accessed Nov. 18, 2021).

[3] P. D. Dudhe and P. Ramteke, "A REVIEW ON PHISHING DETECTION APPROACHES," 2015.

[4] R. M. Mohammad, L. McCluskey, and F. Thabtah, "UCI Machine Learning Repository: Phishing Websites Data Set," 2015. https://archive.ics.uci.edu/ml/datasets/phishing+websites (accessed Jun. 23, 2022).

[5] "sklearn.ensemble.RandomForestClassifier — scikit-learn 1.1.1 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (accessed Nov. 20, 2021).

[6] D. M. W. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, pp. 37–63, Oct. 2011, doi: 10.48550/arxiv.2010.16061.

[7] A. Butnaru, A. Mylonas, and N. Pitropakis, "Towards Lightweight URL-Based Phishing Detection," *Future Internet 2021, Vol. 13, Page 154*, vol. 13, no. 6, p. 154, Jun. 2021, doi: 10.3390/FI13060154.

[8] A. Dechalert, "JavaScript Fundamentals: JSON." https://www.dottedsquirrel.com/js-json/ (accessed Jun. 11, 2022).

[9] "sklearn.model_selection.GridSearchCV — scikit-learn 1.1.1 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html (accessed Nov. 20, 2021).

[10] "sklearn.metrics.confusion_matrix — scikit-learn 1.1.1 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html (accessed Nov. 20, 2020).

[11] "PhishTank." https://phishtank.org/ (accessed May 22, 2022).

[12] "Rank Ranger." https://www.rankranger.com/ (accessed May 22, 2022).