Pengembangan Chrome Extension untuk Mengidentifikasi Phishing Website berdasarkan URL dengan Algoritma Random Forest

Kevin Benedict, Agustinus Noertjahyana, Endang Setyati
Program Studi Informatika, Fakultas Teknologi Industri
Universitas Kristen Petra, Surabaya
Indonesia
kevinbenedict28@gmail.com , agust@petra.ac.id, endang@stts.edu

ABSTRAK

Teknologi yang terus berkembang membuat internet menjadi salah satu bagian terpenting dalam aktivitas manusia sehari-hari. Perkembangan ini juga diikuti dengan meningkatnya aktivitas *phishing* yang tidak hanya bertambah jumlahnya tetapi juga tekniknya yang semakin beragam. Kerugian yang disebabkan oleh *phishing* pun cukup besar. Ada banyak aplikasi untuk menangkal serangan *phishing*, namun sebagian besar masih kurang akurat. Beberapa penelitian menunjukkan bahwa algoritma *ensemble learning* memiliki kemampuan yang baik dalam mendeteksi *phishing website*.

Dalam penelitian ini telah dikembangkan aplikasi *chrome extension* yang menggunakan model *Random Forest* dalam mendeteksi *phishing website*. *Random Forest* merupakan salah satu metode *ensemble learning* yang paling dikenal. Beberapa *hyperparameter* yang paling penting dan akan dieksperimentasi antara lain *n_estimators, min_samples_leaf, min_samples_split, max_features,* dan *max_depth.* Fitur yang digunakan merupakan *Lexical features* yang disusun berdasarkan referensi dari penelitian lain dan *Domain-based features* yang merupakan fitur baru yang diajukan, terdiri dari *Global Page Rank, Average Daily Time, Sites Linking In, Domain Age,* dan *Registration Period*. Semua fitur didapatkan hanya dari URL.

Penelitian ini menunjukkan bahwa kualitas *dataset* merupakan faktor yang paling berpengaruh dalam membuat suatu model yang baik. *Hyperparameter tuning* juga merupakan tahap yang penting namun terbatas dalam skenario tertentu saja. Fitur baru yang diajukan dapat memberikan peningkatan terhadap performa model. Dari beberapa pengujian, dapat disimpulkan bahwa penggunaan *Lexical* dan *Domain-based features* berhasil memperoleh akurasi terbaik sebesar 98.28%.

Kata Kunci: *Phishing Website, Ensemble Learning, Flask, Chrome Extension.*

ABSTRACT

The ever developing technology makes internet one of the most important part in human's daily activity. This development is also followed by the increase of phishing activity which is not only in quantity, but also in the variety of techniques. The loss caused by phishing attacks is quite big. There are a lot of applications for preventing phishing attacks, but most of them are still not accurate enough. Several studies show that ensemble learning algorithm has a good capability in detecting phishing website.

In this research a chrome extension which uses a Random Forest model to detect phishing websites has been developed. Random Forest is one of the most well-known ensemble learning algorithm. The most important hyperparameters which would be experimented with are n_estimators, min_samples_leaf, min_samples_split, max_features, and max_depth. Features used are Lexical features which are based on references from other researches, and Domain-based features which are the newly proposed ones, comprised of Global Page Rank, Average Daily Time, Sites Linking In, Domain Age, and Registration Period. All features are obtained only from the URL.

This research shows that dataset quality is the most impacting factor in making a good model. Hyperparameter tuning is also an important part but is only limited to certain scenario. The newly proposed features could make an improvement to the model's performance. From several experiments, the usage of Lexical and Domain-based features has successfully achieved the best accuracy of 98.28%.

Keywords: *Phishing Website, Ensemble Learning, Flask, Chrome Extension.*

1. LATAR BELAKANG

Phishing merupakan tindakan penipuan yang biasanya dilakukan melalui surat elektronik atau *email* untuk mencuri data pribadi seseorang. Biasanya *email phishing* terlihat berasal dari organisasi resmi yang cukup dikenal. Seringkali dalam email tersebut disertakan sebuah URL yang mengarahkan korban ke *phishing website*. *Website* tersebut terlihat sangat mirip dengan aslinya dan akan meminta korban untuk memasukkan informasi pribadinya yang kemudian akan dimanfaatkan secara ilegal oleh pelaku. Bentuk *phishing* lain yaitu *typosquatting* yaitu memanfaatkan kecenderungan *user* untuk melakukan *typo* atau salah ketik dalam mencari suatu *website* [5] misalnya *goggle.com*.

Terkadang sebuah *phishing website* dapat dengan mudah diidentifikasi, namun tidak jarang juga ada yang tertipu karena *website* tersebut sangat meyakinkan. Berdasarkan penelitian oleh Kaspersky Lab, jumlah serangan *phishing* pada kuartal ketiga tahun 2018 meningkat sebanyak 27.5% dan mencapai jumlah 137 juta [10]. Serangan *phishing* juga mengakibatkan kerugian secara finansial yang tidak sedikit jumlahnya. Menurut Internet Crime Complaint Center FBI, kerugian yang diakibatkan oleh *business email compromise* (BEC) melebihi 12 miliar US *dollar* secara global pada 2018 [1].

Terdapat berbagai metode dalam mengenali *phishing website* namun metode-metode tersebut memiliki akurasi yang rendah dan kekurangan lainnya, seperti yang tertera pada hasil penelitian [6]. Penerapan *blacklist* oleh Google dan Microsoft memiliki kelemahan pada panjangnya waktu yang dibutuhkan dalam

melakukan pembaharuan daftar *blacklist. Web crawler* milik McAfee (Site Advisor) dan VeriSign memiliki kelemahan terhadap serangan-serangan baru atau *zero-day attacks.* Dalam upaya memperoleh akurasi yang tinggi dalam mendeteksi *phishing website*, telah dilakukan banyak penelitian yang menggunakan berbagai metode *machine learning* seperti model *ensemble learning* [11], *hybrid k-Nearest Neighbor* (KNN) dan *Support Vector Machine* (SVM) [2], dan perbandingan 7 algoritma *machine learning* yaitu *Naive Bayes*, *Random Forest*, KNN (n = 3), *Adaboost*, *K-star*, *SMO* dan *Decision Tree* [9].

Pada penelitian ini akan digunakan algoritma Random Forest yang akan diimplementasikan dalam sebuah *chrome extension* untuk mempermudah penggunaan. Google Chrome merupakan web browser yang paling populer dengan *market share* sebesar 57% pada Oktober 2019 [12], sehingga *chrome extension* dapat mencakup banyak *user* potensial. Algoritma Random Forest dipilih karena memiliki keunggulan seperti proses *training* yang cepat dan menghindari *overfitting* [4]. Dalam penelitian [9], fitur yang digunakan hanya *Lexical* dan *Word Vector features* sehingga mengalami kesulitan dalam mengidentifikasi URL pendek. Oleh karena itu dalam penelitian ini fitur yang digunakan adalah *Lexical* dan *Domain-based features* untuk diuji apakah penambahan fitur tersebut meningkatkan performa dari algoritma. *Dataset* yang digunakan didapat dari hasil kompilasi oleh [9] yang berjumlah 73,575 URL yang terdiri dari 36,400 *legitimate* URL dan 37,175 *phishing* URL.

2. TINJAUAN STUDI

2.1 Penelitian yang Berhubungan

Penggunaan *machine learning* dalam mendeteksi *phishing website* memiliki akurasi yang tinggi. Peneliti di [2] melakukan penelitian dengan menggunakan metode *hybrid* antara KNN dan SVM. Model yang dibuatnya memperoleh akurasi sebesar 90,04%.

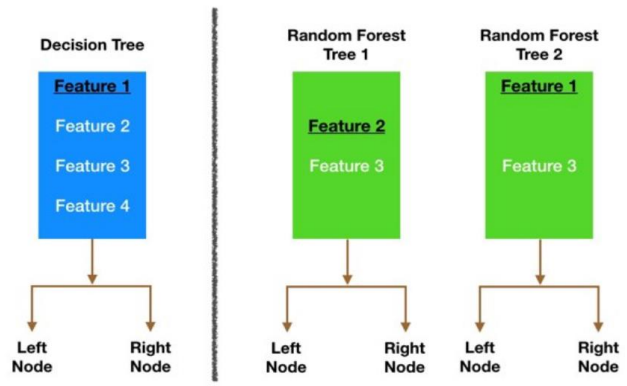
Dari sekian banyak metode *machine learning* yang digunakan dalam mendeteksi *phishing website*, metode *ensemble* dianggap sebagai metode yang paling efektif dan akurat. Hal ini terbukti dari penelitian yang dilakukan oleh [9]. yang membandingkan 7 algoritma *machine learning* yaitu *Naive Bayes*, *Random Forest*, KNN (n = 3), *Adaboost*, *K-star*, SMO dan *Decision Tree*. Fitur yang digunakan oleh peneliti terbagi menjadi 2 kategori, yaitu NLP (*Lexical*) dan *Word Vector* features. Kemudian peneliti membandingkan akurasi dari masing-masing algoritma dimana setiap algoritma menggunakan fitur NLP saja, Word Vector saja, dan *hybrid* NLP dan Word Vector. Hasil penelitian menunjukkan *Random Forest* dengan menggunakan hanya NLP *features* berhasil memperoleh akurasi sebesar 97,98%. Dari hasil penelitian tersebut juga dapat disimpulkan bahwa penggunaan Word Vector *features* tidak dapat menghasilkan akurasi yang sebanding dengan NLP *features*.

Peneliti [11] pada penelitiannya yang menggunakan metode *ensemble* buatannya berhasil memperoleh akurasi 95%. Peneliti juga menuturkan bahwa aplikasi seperti Spoofguard dan Netcraft memiliki akurasi hingga 85%, sedangkan PhishAri yang menggunakan *machine learning* memiliki akurasi sebesar 92,25%. Dari sana juga dapat disimpulkan bahwa metode *ensemble machine learning* lebih unggul dari metode *machine learning* lainnya.

2.2 Random Forest

Random Forest merupakan algoritma *machine learning* yang terdiri dari banyak *decision tree* yang tidak terkorelasi satu sama lain dan beroperasi secara *ensemble* [13]. Algoritma *Random Forest* memiliki kemampuan yang baik dalam menghindari *overfitting* yang terjadi pada *Decision Tree* kompleks. Sebelumnya perlu diketahui permasalahan dalam membuat model *machine learning* ideal yang disebut *Bias-Variance Tradeoff*. Idealnya, sebuah model yang baik memiliki *bias* dan *variance* serendah mungkin [8]. Namun hal ini mustahil dicapai jika model hanya terdiri dari sebuah *Decision Tree* saja. Jika sebuah *Decision Tree* terlalu pendek atau sederhana maka model disebut mengalami *underfitting*, yaitu keadaan dimana model tidak memperoleh informasi yang cukup sehingga menggunakan asumsi dalam menentukan prediksi. Dalam kasus ini, model memiliki *bias* yang tinggi dan *variance* rendah. Sedangkan jika sebuah *Decision Tree* terlalu kompleks maka model disebut mengalami *overfitting*, dimana model memiliki generalisasi yang rendah. Dengan kata lain model hanya mengenali dengan baik data yang sudah pernah dilihat dalam proses *training*, namun ketika berhadapan dengan data baru performa model tidak akan sebaik sebelumnya.

Random Forest dapat mengatasi kelemahan *overfitting* dengan 2 prinsip utama yang dimiliki. Pertama, *bagging* (*Bootstrap Aggregation*) dimana setiap *tree* akan di-*training* menggunakan *subset* dari *dataset* yang dipilih secara acak dan terdapat pengulangan data. Setiap sampel tersebut berukuran sama dengan *dataset* aslinya. Sebagai contoh *training dataset* asli berisi [1, 2, 5, 7], pada salah satu *tree* akan diberi data [1, 2, 2, 5] dan pada *tree* lainnya [2, 5, 7, 7]. Dari contoh dapat dilihat bahwa "7" tidak terdapat dalam *bootstrapped dataset* pada *tree* pertama, demikian pula "1" pada *tree* kedua. Kemudian, *aggregation* merupakan proses dalam menentukan prediksi akhir dari suatu sampel dengan mengambil nilai modus (klasifikasi) atau mengambil nilai rata-rata (regresi) dari keseluruhan prediksi. Kedua, *feature randomness* dimana setiap *tree* mempertimbangkan *subset* fitur pada setiap *split*. Berbeda dengan *decision tree* biasa yang mempertimbangkan semua fitur yang ada dalam menentukan *split*. Hal ini menyebabkan korelasi dari setiap *tree* semakin rendah. Visualisasi dari perbedaan *node splitting* dapat dilihat pada Gambar 1.



Gambar 1. *Node Splitting* dalam *Decision Tree* vs. *Random Forest* [13]

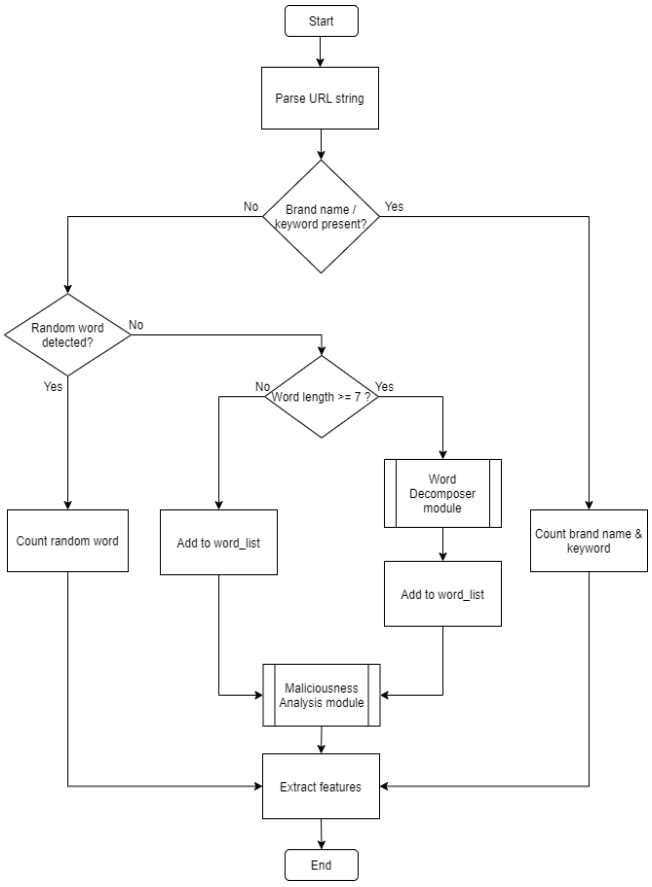
3. DESAIN SISTEM

Proses meliput *Preprocessing*, Membuat model *Random Forest*, sistem untuk *Training* dan *Testing*, serta program *chrome extension* yang terintegrasi dengan model yang sudah dibuat.

3.1 *Dataset* dan *Preprocessing*

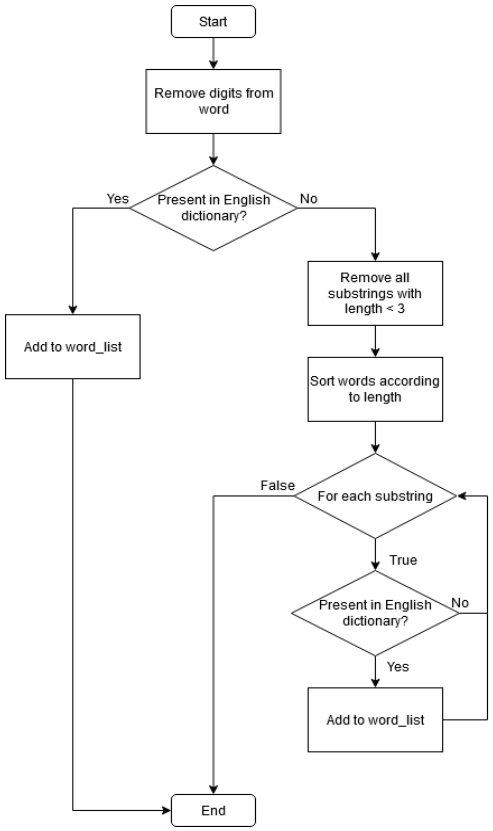
Dataset diperoleh dari hasil kompilasi oleh peneliti [9] yang berjumlah 73,575 URL, dengan pembagian 36,400 *legitimate*

URL dan 37,175 *phishing* URL. Distribusi setiap *class* dalam *dataset* tersebut sudah seimbang. Seluruh tahapan dalam *preprocessing* juga dibuat semirip mungkin berdasarkan referensi dari [9]. Pertama, *string* URL dipecah per bagian (*domain, subdomain, second-level domain*, TLD, dan *file* path) dan per kata. Kemudian, setiap kata yang terdapat didalam *list* diperiksa berdasarkan "allbrands.txt" dan "keywords.txt" untuk mendeteksi keberadaan nama merk terkenal atau kata kunci pada URL. Kata yang tidak terdapat dalam kedua *dataset* tersebut selanjutnya diperiksa oleh modul *Random Word Detector* dari [7] yang menggunakan *Markov Chain* untuk mendeteksi kata acak. Lalu, kata yang bukan merupakan kata acak dan memiliki panjang lebih dari atau sama dengan 7 karakter diperiksa melalui modul *Word Decomposer* untuk mencari setiap kata dari suatu kata majemuk. Terakhir, hasil yang diperoleh dari modul *Word Decomposer* dan kata lain yang panjangnya kurang dari 7 diperiksa oleh modul *Maliciousness Analysis* untuk memeriksa adanya nama *brand* atau kata kunci, dan *typosquatting.* Setelah pemrosesan URL selesai, dilakukan pengambilan fitur. Gambar 2 merupakan *flowchart* keseluruhan proses *preprocessing.*



Gambar 2. ***Preprocessing***

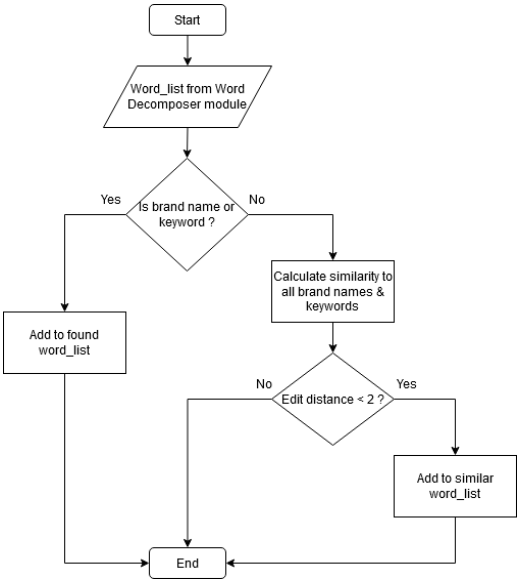
Modul *Word Decomposer* digunakan untuk memecah kata menjadi *substring* jika terdapat lebih dari satu kata didalamnya. Tujuan dari proses ini adalah untuk menemukan nama merk dan kata kunci yang terdapat dalam suatu kata majemuk [9]. Dalam proses ini, setiap *substring* akan diperiksa menggunakan kamus yang terdiri dari kamus Bahasa Inggris, *file* "allbrands.txt", dan *file* "keywords.txt". Misalnya, dari kata "securelogin" akan diperoleh *substring* "secure" dan "login" yang merupakan kata kunci dalam *dataset*. Pertama, setiap angka pada kata dihilangkan, karena pelaku dapat memanfaatkannya untuk membuat deteksi semakin sulit. Kemudian, setiap kata diperiksa keberadaannya dalam kamus. Jika kata terdapat dalam kamus, maka kata langsung dimasukkan ke *word_list* untuk dianalisis lebih lanjut. Jika tidak, maka kata tersebut masuk ke proses selanjutnya untuk diekstrak semua *substring* yang ada. Setelah semua *substring* diperoleh, setiap *substring* yang panjangnya kurang dari atau sama dengan 3 akan dihapus. Kemudian pengecekan dimulai dari *substring* terpanjang. Jika kata tersebut ada dalam kamus, maka kata tersebut dimasukkan ke dalam *word_list.* Jika tidak, pengecekan berlanjut ke *substring* terpanjang berikutnya hingga selesai. Terakhir, dilakukan penghapusan kata yang terdapat atau merupakan bagian dari kata lain yang lebih panjang. Misalnya, terdapat kata "secure" dan "cure", Dalam kasus ini, kata "cure" akan dihapus. Hal ini ditujukan untuk menghilangkan *false positive* [9]. Gambar 3 merupakan *flowchart* dari modul ini.



Gambar 3. ***Word Decomposer***

Modul *Maliciousness Analysis* digunakan untuk mendeteksi *typosquatting. Typosquatting* sendiri merupakan bentuk penyalahgunaan kecenderungan *user* dalam melakukan salah ketik dan mengarahkannya ke *website* berbahaya. Dari pemeriksaan sebelumnya, kata yang lolos akan diperiksa kembali berdasarkan "allbrands.txt" dan "keywords.txt". Jika kata tidak terdapat di kedua *dataset* tersebut, maka dilakukan pengecekan kemiripan antara setiap kata terhadap kedua *dataset* tersebut dengan menggunakan algoritma Edit Distance. Dalam algoritma Edit Distance diperhitungkan penambahan, pengurangan, dan substitusi karakter sehingga memungkinkan perbandingan antara 2 kata yang memiliki panjang berbeda. Telah ditentukan pula *threshold* dari Edit Distance yaitu 2, dimana kata yang memiliki

nilai kurang dari 2 disimpan ke dalam *word_list*. Gambar 4 merupakan *flowchart* dari modul ini.



Gambar 4. ***Maliciousness Analysis***

3.2 *Features*

Fitur yang digunakan pada penelitian ini terdiri dari *Lexical* dan *Domain-based features*, dengan total sebanyak 44 fitur. Fitur *Lexical* disusun berdasarkan gabungan fitur-fitur yang telah digunakan oleh penelitian [3] [9]. Fitur ini diperoleh dari *string* URL seperti URL *length, longest word, shortest word, known* TLD, dan sebagainya. Jumlah dari fitur ini adalah 39 buah. Tabel 1 merupakan daftar dari fitur *Lexical.*

Tabel 1. ***Lexical Features***

URL Length	Panjang dari URL.
Word Count	Jumlah kata yang diperoleh setelah menghapus semua karakter spesial dari URL (Protokol dan TLD tidak termasuk).
Longest Word Length	Panjang kata paling panjang.
Shortest Word Length	Panjang kata paling pendek.
Average Word Length	Rata-rata panjang semua kata.
Standard Deviation of Word Length	Standar deviasi dari panjang semua kata.
Brand Check for Domain	Keberadaan nama *brand* pada *domain.*
Keyword Count	Jumlah kata kunci yang terdapat pada URL.
Brand Name Count	Jumlah nama *brand* yang terdapat pada URL.
Similar Keyword Count	Jumlah kata pada URL yang mirip dengan nama suatu *kata kunci.*
Similar Brand Name Count	Jumlah kata pada URL yang mirip dengan nama suatu *brand.*
Target Keyword Count	Jumlah kata kunci yang menjadi target *typosquatting.*
Target Brand Name Count	Jumlah nama *brand* yang menjadi target *typosqsuatting.*
Random Word Count	Jumlah kata acak atau *gibberish.*
Other Word Count	Jumlah kata yang tidak terdapat pada semua *list* diatas namun terdapat dalam kamus Bahasa Inggris.
Digit Count (Domain)	Jumlah angka yang terdapat pada *domain.*
Digit Count (Subdomain)	Jumlah angka yang terdapat pada *subdomain.*
Digit Count (File Path)	Jumlah angka yang terdapat pada *file path.*
Subdomain Count	Jumlah *subdomain* pada URL.
Random Domain	Apakah *domain* dari URL merupakan kata acak atau bukan.
Length (Domain)	Panjang dari *domain.*
Length (Subdomain)	Panjang dari *subdomain.*
Length (File Path)	Panjang dari *file path.*
Known TLD	Apakah *top-level domain* yang terdapat pada URL termasuk yang valid berdasarkan daftar oleh IANA.
“www” Count in URL	Jumlah “www” dalam *raw words* dari URL.
“com” Count in URL	Jumlah “com” dalam *raw words* dari URL.
Email Presence in URL	Keberadaan *e-mail* dalam URL.
Puny Code	Adanya penggunaan *puny code* atau tidak.
‘-‘ Count	Jumlah karakter ‘-‘ pada URL.
‘.‘ Count	Jumlah karakter ‘.‘ pada URL.
‘/‘ Count	Jumlah karakter ‘/‘ pada URL.
‘@‘ Count	Jumlah karakter ‘@‘ pada URL.
‘?‘ Count	Jumlah karakter ‘?‘ pada URL.
‘=‘ Count	Jumlah karakter ‘=‘ pada URL.
‘&‘ Count	Jumlah karakter ‘?’ pada URL.
‘_‘ Count	Jumlah karakter ‘_‘ pada URL.
IP Address Presence	Keberadaan IP *address* pada URL.
Shortening Services	Adanya penggunaan *shortening service* pada URL.
Valid SSL certificate	Valid atau tidaknya sertifikat SSL yang digunakan *website* terkait.

Domain-*based features* merupakan fitur baru yang diusulkan dalam penelitian ini. Jumlah dari fitur ini adalah 5 buah. Fitur ini terdiri dari informasi berdasarkan *domain* seperti umur *domain*, periode registrasi, peringkat global, dan sebagainya. Fitur-fitur ini diambil dari *website* WHOIS dan Alexa dengan menggunakan *web scraping*. Penambahan *Domain-based features* sendiri bertujuan untuk mengatasi kelemahan dari *Lexical features* yang tidak dapat mendeteksi *phishing website* yang memiliki URL pendek [9].

Tabel 2. *Domain-based Features*

Global Page Rank (Alexa)	Peringkat dari *website* secara global. *Website* berperingkat rendah tidak akan terindeks oleh Alexa.
Average Daily Time on Site (Alexa)	Rata-rata waktu (dalam detik) yang dipakai *user* pada *website* terkait.
Sites Linking In (Alexa)	Jumlah *website* lain yang memiliki tautan ke *website* terkait.
Domain Age (WHOIS)	Usia *domain* (dalam hari) saat pengecekan.
Registration Period (WHOIS)	Rentang waktu (dalam hari) antara aktivasi dan ekspirasi domain.

3.3 Model dan *Hyperparameter Tuning*

Pada proses ini terjadi proses pembuatan model dan juga *hyperparameter tuning*. Dalam penelitian ini akan dilakukan beberapa eksperimen dalam menentukan model akhir yang akan dibandingkan dengan model dari hasil penelitian pada [9]. Pengujian dibagi menjadi 2 kategori besar. Pertama, keseluruhan *dataset* digunakan baik dalam *training* maupun *testing* model. Kedua, *dataset* dibagi menjadi *training* dan *testing dataset.* Kemudian untuk kedua eksperimen tersebut akan dilakukan 3 kali menggunakan model dengan *default hyperparameter*, model dengan *default hyperparameter* dan *bootstrap = False*, dan model dengan *hyperparameter* yang sudah di-*tuning*. Tujuan dari pembagian kategori seperti diatas adalah untuk melihat pengaruh jumlah *training dataset* terhadap performa model, pengaruh dari *bootstrapping*, serta untuk melihat seberapa besar improvisasi yang dihasilkan oleh proses *hyperparameter tuning.* Kemudian semua model dari eksperimen A akan divalidasi kembali dengan menggunakan *Stratified K-Fold Cross Validation* untuk menghitung *metric* dari setiap model ketika di-*testing* dengan data yang belum pernah dilihat. *Hyperparameter* yang digunakan dalam proses *tuning* adalah *n_estimators, max_features, max_depth, min_samples_split,* dan *min_samples_leaf* menggunakan *Randomized Search Cross Validation.*

3.4 *Training* dan *Testing*

Dalam melakukan *training*, pertama dataset dibagi menjadi *training* dan *testing dataset* dengan perbandingan 2:1. Model di-*training* dan di-*testing* menggunakan *dataset* bersangkutan, atau menggunakan keseluruhan *dataset* tergantung dari eksperimen. Setelah itu model dievaluasi berdasarkan beberapa *metric* yaitu *confusion matrix, accuracy, precision,* dan *recall.* Waktu *training* dan *testing* juga direkam.

3.5 Program *Chrome Extension*

Pengaplikasian model *Random Forest* yang telah dibuat dilakukan dalam bentuk *chrome extension.* Model akan disimpan sebagai API menggunakan Flask di sebuah *server*. Kemudian API menerima dan memproses URL yang dikirim dari *extension*. Hasil prediksi dikirim kembali ke *extension* dan aksi selanjutnya ditentukan berdasarkan prediksi. Jika prediksi *legitimate,* maka *user* akan langsung diarahkan menuju *website* tersebut. Sebaliknya, *extension* akan memblokir akses dan menampilkan halaman yang berisi peringatan.

4. PENGUJIAN SISTEM

Seperti yang telah dijelaskan sebelumnya, akan dilakukan dua eksperimen dimana model di-*training* dengan keseluruhan *dataset* dan terdapat pembagian *dataset training* dan *testing*. Masing-masing eksperimen dilakukan dengan 3 model berbeda, total ada 6 model yang dihasilkan.

4.1 Pengujian Model Tanpa Pembagian *Dataset*

Pada bagian ini dicantumkan perbandingan hasil pengujian model-model dengan menggunakan keseluruhan *dataset* baik untuk *training* maupun *testing*. Jumlah *dataset* yaitu 73,575 baris dan 44 kolom. Model A.1 menggunakan *hyperparameter default*, model A.2 menggunakan *hyperparameter default* serta *bootstrap = False*, sedangkan model A.3 menggunakan *hyperparameter* yang sudah di-*tuning.* Tabel 3 merupakan daftar *hyperparameter* setiap model.

Tabel 3. *Hyperparameter* Model Eksperimen A

Hyperparameter	Model A.1	Model A.2	Model A.3
n_estimators	100	100	400
min_samples_split	2	2	2
min_samples_leaf	1	1	1
max_features	auto	auto	12
max_depth	None	None	60

Tabel 4 merupakan *confusion matrix* setiap model eksperimen A. Dari sini dapat diperoleh nilai *Accuracy, Precision,* dan *Recall*.

Tabel 4. *Confusion Matrix* Model Eksperimen A

	True Positive (TP)	False Positive (FP)	False Negative (FN)	True Negative (TN)
Model A.1	37160	13	15	36387
Model A.2	37155	7	20	36393
Model A.3	37164	16	11	36384

Tabel 5 berisi *evaluation metrics* setiap model. Dapat dilihat bahwa jika model di-*train* dan *test* dengan menggunakan *dataset* yang sama, akan menghasilkan *metric* yang nyaris sempurna. Waktu *training* dan *testing* model A.3 jauh lebih lama dikarenakan memiliki jumlah *tree* yang lebih banyak.

Tabel 5. *Evaluation Metrics* Model Eksperimen A

	Train Acc. (%)	Test Acc. (%)	Precision (%)	Recall (%)	Train time (s)	Test time (s)
Model A.1	99.96	99.96	99.97	99.96	8.602	1.063
Model A.2	99.96	99.96	99.98	99.95	12.56	1.122
Model A.3	99.96	99.96	99.96	99.97	62.32	3.901

Untuk mendapat *metric* sesungguhnya, perlu dilakukan validasi dengan menggunakan data yang belum pernah dilihat oleh model. Dalam penelitian ini akan digunakan *Stratified K-Fold Cross Validation* dengan nilai *k* = 10. Pada setiap *fold*, perbandingan *training* : *testing dataset* adalah 9:1 dimana jumlah *training dataset* yaitu 66,217 baris dan 44 kolom, sedangkan *testing*

dataset yaitu 7,358 baris dan 44 kolom. Tabel 6 berisi *Train* dan *Test Accuracy* dari setiap *fold* beserta rata-ratanya.

Tabel 6. Hasil *Stratified 10-Fold CV* (*Train & Test Accuracy*)

Fold	Model A.1		Model A.2		Model A.3	
	Train Acc. (%)	Test Acc. (%)	Train Acc. (%)	Test Acc. (%)	Train Acc. (%)	Test Acc. (%)
1	99.97	98.30	99.97	98.45	99.97	98.37
2	99.97	98.26	99.97	98.37	99.97	98.29
3	99.96	98.21	99.96	98.31	99.96	98.25
4	99.97	98.23	99.97	98.27	99.97	98.29
5	99.96	98.30	99.96	98.44	99.96	98.45
6	99.96	98.30	99.96	98.46	99.96	98.37
7	99.96	98.33	99.97	98.34	99.97	98.27
8	99.96	98.23	99.96	98.34	99.96	98.23
9	99.97	97.91	99.97	98.03	99.97	98.02
10	99.97	98.30	99.97	98.45	99.97	98.31
Avg.	**99.96**	**98.24**	**99.97**	**98.35**	**99.97**	**98.28**

Tabel 7 berisi *Precision* dan *Recall* dari setiap *fold* beserta rata-ratanya. Dapat dilihat bahwa performa ketiga model cukup identik satu sama lain. Tipisnya perbedaan antara *fold* pada setiap model menunjukkan bahwa setiap model memiliki generalisasi yang baik.

Tabel 7. Hasil *Stratified 10-Fold CV* (*Precision & Recall*)

Fold	Model A.1		Model A.2		Model A.3	
	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)
1	97.82	98.84	98.21	98.74	98.00	98.79
2	97.64	98.95	97.85	98.95	97.69	98.95
3	97.76	98.71	98.07	98.60	97.84	98.71
4	97.39	99.17	97.59	99.03	97.46	99.19
5	97.61	99.06	97.92	99.00	97.85	99.11
6	97.82	98.84	98.00	98.98	97.90	98.90
7	97.77	98.95	97.82	98.92	97.71	98.90
8	97.71	98.82	98.02	98.71	97.84	98.68
9	96.87	99.06	97.05	99.11	96.95	99.19
10	97.34	99.35	97.75	99.22	97.36	99.35
Avg.	**97.57**	**98.98**	**97.83**	**98.93**	**97.66**	**98.98**

4.2 Pengujian Model dengan Pembagian *Dataset*

Pada bagian ini dicantumkan perbandingan hasil pengujian model-model yang di-*training* dan di-*testing* dengan menggunakan *dataset* berbeda. Pembagian dilakukan dengan rasio *training : testing* sebesar 2:1. Jumlah *training dataset* yaitu 49,295 baris dan 44 kolom, sedangkan *testing dataset* yaitu 24,280 baris dan 44 kolom.

Tabel 8. *Hyperparameter* Model Eksperimen B

Hyperparameter	Model B.1	Model B.2	Model B.3
n_estimators	100	100	300
min_samples_split	2	2	2
min_samples_leaf	1	1	1
max_features	auto	auto	10
max_depth	None	None	70

Tabel 9 merupakan *confusion matrix* setiap model eksperimen B. Dari sini dapat diperoleh nilai *Accuracy*, *Precision*, dan *Recall*.

Tabel 9. *Confusion Matrix* Model Eksperimen B

	True Positive (TP)	False Positive (FP)	False Negative (FN)	True Negative (TN)
Model B.1	12002	333	185	11760
Model B.2	11996	295	191	11798
Model B.3	11996	322	191	11771

Tabel 5 berisi *evaluation metrics* setiap model. Dapat dilihat terdapat sedikit penurunan performa secara keseluruhan dibandingkan dengan model eksperimen A. Waktu *training* dan *testing* lebih cepat walau tidak terlalu berbeda dikarenakan *training dataset* yang digunakan ukurannya lebih kecil.

Tabel 10. *Evaluation Metrics* Model Eksperimen B

	Train Acc. (%)	Test Acc. (%)	Precision (%)	Recall (%)	Train time (s)	Test time (s)
Model B.1	99.97	97.87	97.30	98.48	5.558	0.329
Model B.2	99.97	98.00	97.60	98.43	8.159	0.352
Model B.3	99.97	97.89	97.39	98.43	25.31	0.920

4.3 Perbandingan dengan Penelitian yang Berhubungan

Tabel 11 berisi perbandingan antara model yang telah dipilih yaitu model A.3 dengan penelitian oleh [9]. Pemilihan tersebut didasarkan pada *Test Accuracy* sebagai prioritas utama, dan *Recall*. Model yang dibandingkan sama-sama dibuat dengan menggunakan algoritma *Random Forest*, menggunakan *dataset* yang sama, dan dievaluasi dengan menggunakan *10-Fold Cross Validation.*

Tabel 11. Perbandingan dengan Penelitian yang Berkaitan

	Test Accuracy (%)	Precision (%)	Recall (%)
Model oleh Sahingoz, et al. [9]	97.98	97.00	99.00
Model A.3 (Hasil *cross validation*)	98.28	97.66	98.98

Dari tabel diatas dapat disimpulkan bahwa tidak terjadi improvisasi yang cukup signifikan dari model yang telah dibuat. Hal ini kemungkinan besar disebabkan oleh *dataset* yang digunakan berisi URL yang relatif lama, khususnya pada *phishing dataset*. Mengingat karakteristik *phishing website* yang berumur pendek, sebagian besar dari URL dalam *phishing dataset* sudah tidak aktif atau berstatus *offline*. Dengan kata lain, *Domain-based features* dari mayoritas URL *phishing* bernilai 0, sehingga terdapat *gap* yang cukup besar antara *legitimate* dan *phishing* URL untuk fitur ini.

5. KESIMPULAN DAN SARAN

Kesimpulan dari pengujian ini adalah:

- Gabungan *Lexical* dan *Domain-based features* memiliki prospek yang cukup baik dalam membuat model *Random Forest* untuk mendeteksi *phishing website.* Namun model yang telah dibuat tidak dapat menghasilkan improvisasi yang cukup signifikan dari penelitian yang dijadikan referensi utama. Hal ini kemungkinan besar disebabkan oleh *dataset* yang kurang *up-to-date* sehingga sebagian besar *phishing* URL sudah tidak aktif lagi. Kemungkinan kedua yaitu model sudah mencapai batas sehingga performa model tidak dapat ditingkatkan lebih jauh lagi.

- *Bootstrapping* tidak terlalu memberikan dampak dalam mencegah *overfitting*. Hal ini dapat dilihat dari perbandingan performa model A.1 - A.2 dan B.1 – B.2 dimana kedua model sangat identik. Namun model masih tetap mempertahankan keacakannya dengan adanya prinsip *feature randomness.*

- *Hyperparameter tuning* tidak selalu dapat memberikan peningkatan terhadap performa model. Dari hasil penelitian dapat dilihat bahwa model dengan *hyperparameter default* sudah dapat memberikan performa yang baik asal *dataset* yang digunakan berkualitas.

- Lamanya waktu *preprocessing* menyebabkan pengalaman *browsing* menjadi tidak nyaman karena *user* harus menunggu beberapa detik sebelum dapat mengakses situs. Berkaitan dengan hal ini, fitur Safe Browsing dari Google Chrome akan terlebih dahulu memblokir situs *phishing* sebelum program selesai memproses URL. Hal ini menyebabkan program *extension* menjadi kurang berguna dalam mendeteksi *phishing website*.

Saran yang diberikan untuk penyempurnaan dan pengembangan lebih lanjut untuk program ini adalah sebagai berikut:

- Mencari cara bagaimana membuat *dataset* yang dapat terus diperbarui atau bahkan juga ditambah.

- Mencari cara untuk mempercepat *preprocessing.*

- Mengimplementasikan *deep learning* yang secara umum lebih *advanced* dari *machine learning* tradisional dan performanya dapat terus meningkat seiring dengan bertambahnya jumlah data.

6. REFERENSI

[1] Aalto, M. 2018. *Statistics Showing 5 Phishing Trends for 2019 (with Infographic)*. Dipetik May 15, 2019, dari hoxhunt.com: https://www.hoxhunt.com/blog/statistics-showing-5-phishing-trends-for-2019

[2] Altaher, A. 2017. Phishing Websites Classification using Hybrid SVM and KNN Approach. *(IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 8, No. 6, 2017*, 90-95.

[3] Ayres, L. D., Brito, I. V., & Souza, R. R. 2019. Using Machine Learning to Automatically Detect Malicious URLs in Brazil. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2019)*, 972-985.

[4] Donges, N. 2019, June 16. *A Complete Guide to the Random Forest Algorithm*. Dipetik November 20, 2019, dari builtin.com: https://builtin.com/data-science/random-forest-algorithm

[5] McAfee. 2013, July 3. *What is Typosquatting?* Dipetik November 18, 2019, dari mcafee.com: https://www.mcafee.com/blogs/consumer/consumer/family-safety/what-is-typosquatting/

[6] Mohammad, R. M., Thabtah, F., & McCluskey, L. 2015. Tutorial and critical analysis of phishing websites. *Computer Science Review*, 1-24.

[7] Neuhaus, R., & Ruvinskiy, R. 2015, August 25. *Gibberish Detector*. Dipetik May 5, 2020, dari github.com: https://github.com/rrenaud/Gibberish-Detector

[8] Ramchandani, P. 2018, October 10. *Random Forests and the Bias-Variance Tradeoff*. Dipetik September 5, 2020, dari towardsdatascience.com: https://towardsdatascience.com/random-forests-and-the-bias-variance-tradeoff-3b77fee339b4

[9] Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. 2018. Machine learning based phishing detection dari URLs. *Expert Systems With Applications*, 345-357.

[10] Spadafora, A. 2018, November 7. *Phishing attacks see major rise*. Dipetik May 15, 2019, dari techradar.com: https://www.techradar.com/news/phishing-attacks-see-major-rise

[11] Ubing, A. A., Jasmi, S. K., Abdullah, A., Jhanjhi, N., & Supramaniam, M. 2019. Phishing Website Detection: An Improved Accuracy through Feature Selection and Ensemble Learning. *(IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 10 No. 1, 2019*, 252-257.

[12] W3Counter. 2019, October 31. *Browser & Platform Market Share - October 2019*. Dipetik November 21, 2019, dari w3counter.com: https://www.w3counter.com/globalstats.php?year=2019&month=10

[13] Yiu, T. 2019, June 12. *Understanding Random Forest: How the Algorithm Works and Why it Is So Effective*. Dipetik November 24, 2019, dari towardsdatascience.com: https://towardsdatascience.com/