Detecção de ataques de phishing em tempo real utilizando algoritmos de aprendizado de máquina

João A. Souza[1], **Dalbert M. Mascarenhas**[1]

[1]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ)
CEP 25.620-003 – Petrópolis – RJ – Brasil

joao.souza.2@aluno.cefet-rj.br, dalbert.mascarenhas@cefet-rj.br

Abstract. *Phishing is a type of social engineering cyber-attack that aims to steal user's personal information. Due to the COVID-19 pandemic and increase of remote work, the number of cyber-attacks has increased, especially phishing. Although many anti-phishing solutions exist, such as blacklists and heuristics, attacks are constantly adapting. This work proposes a machine learning based model for real time detection of phishing attacks that is focused on performance. Six algorithms were tested and out of witch, SVM obtained the best result with an accuracy of 99,19%.*

Resumo. *Phishing é um tipo de ataque cibernético de engenharia social que visa roubar informações de usuários. Com a pandemia de COVID-19 e popularização do modelo de trabalho remoto, o número de ataques cibernéticos aumentou, especialmente os de phishing. Embora diversas soluções anti-phishing existam, como blacklists e heurísticas, os ataques estão em constante adaptação. Este trabalho propõe um modelo baseado em aprendizado de máquina para detecção em tempo real de ataques de phishing com um enfoque no desempenho. Foram testados seis algoritmos dentre os quais o melhor resultado foi obtido pelo SVM com 99,19% de acurácia.*

1. Introdução

Em razão do confinamento causado pela pandemia de COVID-19, foi observado um aumento significativo no número de atividades realizadas através da Internet. Embora quantidade de trabalhos realizados de maneira remota viesse aumentando em ritmo constante, foi a partir de 2020 que essa modalidade de trabalho sofreu um aumento drástico [Ozimek 2020]. No entanto, muitas empresas não possuíam políticas de segurança adequadas para este cenário, o que levou a um aumento no número de ataques cibernéticos [Pranggono and Arabo 2021].

Segundo [Pranggono and Arabo 2021], o tipo de ataque mais comum nesse período foi o de *phishing*. Neste tipo de ataque, o criminoso se passa por um serviço de confiança como sites de bancos, governos ou lojas virtuais a fim de capturar informações pessoais da vítima. A popularidade desse tipo de ataque se deve ao seu alto percentual de sucesso (maior que 30%), e baixo esforço do atacante.

A alternativa mais moderna para detecção de *phishing* é a utilização de aprendizado de máquina. Dado um conjunto de parâmetros, os algoritmos são capazes de ponderá-los e realizar a classificação de forma a diminuir o erro. Algoritmos de aprendizado de máquina, diferentemente de *blacklists*, são capazes de detectar ataques *zero-day*,

além de possuir uma taxa de falsos positivos mais baixa que as abordagens por heurísticas [Khonji et al. 2013].

Diante disso, este trabalho propõe um modelo de aprendizado de máquina que seja capaz de detectar ataques de *phishing*. O diferencial deste em relação à trabalhos anteriores é o foco em desempenho, visando possibilitar a detecção em tempo real. Uma vez que domínios maliciosos sejam identificados, é possível desenvolver ferramentas que auxiliem o usuário na prevenção deste tipo de ataque, protegendo seus dados pessoais.

O resto do artigo está organizado da seguinte forma. Primeiramente são discutidos os trabalhos relacionados na Seção 2. Em seguida, é apresentada a metodologia da solução, incluindo a descrição da coleta e preparo de dados e seleção de atributos e algoritmos na Seção 3. A avaliação dos resultados é apresentada na Seção 4. Finalmente, é apresentada a conclusão do trabalho, bem como uma perspectiva para trabalhos futuros na Seção 5.

2. Trabalhos Relacionados

Para a elaboração deste trabalho foi realizada uma pesquisa sistemática a fim de encontrar artigos que abordassem a detecção de ataques de *phishing*. Com o objetivo de obter melhores resultados, foram utilizadas *strings* de busca contendo palavras chaves relacionadas ao tema. A pesquisa foi feita pelo portal de periódicos da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) utilizando o acesso de Comunidade Acadêmica Federada (CAFe).

A primeira *string* de busca utilizada foi: *(phishing) AND (detection OR identification OR recognition)*. Desta busca foram retornados 4414 resultados dos quais 1 trabalho foi selecionado. Com a finalidade de refinar a busca para ressaltar artigos que utilizassem técnicas de *machine learning*, a *string* de busca foi alterada para: *(phishing) AND (detection OR identification OR recognition) AND (machine learning OR artificial intelligence OR ai)*. Com isso, 761 resultados foram obtidos e 4 artigos foram selecionados.

Além disso, mais 1 artigo foi escolhido utilizando a técnica de *snowballing*. Esta técnica consiste em encontrar trabalhos relacionados nas referências de trabalhos já pesquisados. O processo é realizado de maneira recursiva até que se encontre um trabalho de interesse.

Para a seleção dos artigos foram utilizados os critérios de inclusão e exclusão descritos na Tabela 1. Com o critério de inclusão CI-1, buscou-se encontrar trabalhos que abordassem quaisquer técnicas para detecção de ataques de *phishing*. Já com o critério CI-2, o objetivo foi encontrar exclusivamente trabalhos que empregassem aprendizado de máquina para o mesmo objetivo. Como foram considerados apenas trabalhos onde o texto completo pudesse ser acessado, o critério de exclusão CE-1 foi empregado. Alguns dos resultados continham as palavras-chave de busca mas não estavam relacionados com o tema, sendo desconsiderados pelo critério de exclusão CE-2.

Em [Khonji et al. 2013], é realizado um estudo da literatura sobre técnicas de detecção e prevenção de *phishing*. As técnicas de detecção são divididas em quatro tipos: *blacklists*, baseadas em *heurísticas*, por semelhança visual e *machine learning*. As *blacklists* são mais efetivas quando uma taxa mínima de falsos positivos é requerida, além disso é a defesa que utiliza menos recursos computacionais. As técnicas baseadas em

Tabela 1. Critérios de inclusão e exclusão.

Código	Descrição
CI-1	Artigos que abordam técnicas de detecção para ataques de *phishing*
CI-2	Artigos que abordam técnicas de detecção para ataques de *phishing* utilizando técnicas de aprendizado de máquina
CE-1	Não foi possível ter acesso ao trabalho completo
CE-2	O artigo não está relacionado à área de computação

heurísticas e semelhança visual possuem desempenho semelhante, sendo capazes de detectar ataques *zero-day* (ataques ainda não catalogados), mas necessitam de mais recursos da máquina, além de apresentar uma maior taxa de falsos positivos. Já os classificadores baseados em *machine learning* possuem como principal vantagem a adaptação a novos tipos de ataque, sendo o único método de classificação a apresentar alta acurácia e ser capaz de detectar ataques de *zero-day*.

Para além das quatro abordagens supracitadas, o trabalho realizado em [Sahingoz et al. 2019] classifica sites de *phishing* com base na URL. Foi desenvolvido um sistema em tempo real que utiliza sete algoritmos diferentes de classificação e atributos baseados em processamento de linguagem natural (NLP). Para a medição do sistema, um conjunto de dados foi construído onde os resultados experimentais foram testados. O melhor desempenho obtido foi com o algoritmo Random Forest utilizando apenas atributos baseados em NLP no qual foi obtida uma acurácia de 97,98%.

O modelo proposto em [Rao and Pais 2019] combina os atributos extraídos da URL com dados do código fonte e informações obtidas a partir de serviços de terceiros a fim de superar as desvantagens de outras técnicas *anti-phishing*. Os serviços de terceiros utilizados foram: a idade do domínio, pontuação em *rankings* de páginas web e resultados em mecanismos de busca. Para os atributos baseados em código fonte, foram analisadas as *tags* HTML e endereços de *hiperlinks* presentes na página. O algoritmo que obteve maior desempenho foi o Random Forest utilizado em conjunto com análise de componentes principais obtendo uma acurácia global de 99,55%.

O trabalho realizado em [Stalmans and Irwin 2011] utiliza métodos estatísticos como *Naive Bayes*, distância da variância total, e distribuição de probabilidades para detectar atividades de *botnets*. Os atributos utilizados para classificação foram divididos em atributos de DNS e textuais. Os atributos de DNS incluem TTL médio dos registros, intervalos de IP e sua distribuição geográfica. Já os textuais se baseiam no fato de, muitas vezes, os nomes de domínios maliciosos serem gerados por algoritmos, possuindo palavras não legíveis. Para isso, métodos estatísticos foram aplicados para analisar a frequência das letras presentes no domínio e comparar com a frequência de letras em palavras da língua inglesa. As técnicas de defesa foram capazes de detectar domínios maliciosos com uma alta acurácia e uma baixa taxa de falsos positivos.

Em [Nagunwa et al. 2022], é proposta uma abordagem baseada em aprendizado de máquina para detecção de sites de *phishing* hospedados em redes *fast-flux*. Ao todo foram utilizados 56 atributos cuja efetividade para previsão foi avaliada no contexto de

classificação binária e multi-classe. A proposta obteve uma acurácia de 98,42% e 97,81% para classificações binárias e multi-classe, respectivamente. No entanto, o tempo necessário para extração dos atributos de uma página foi de 162,64 segundos o que inviabilizaria a detecção em tempo real.

No trabalho proposto em [Shahrivari et al. 2020] é feita uma comparação entre o desempenho de diversos algoritmos de aprendizado de máquina na detecção de *phishing*. Foram utilizados 30 atributos com valores booleanos e quatro métricas de avaliação: acurácia, precisão, *recall* e F1. Ao todo foram testados 9 algoritmos: regressão logística, árvore de decisão, *random forest*, *ada booster*, KNN, redes neurais, SVM, *gradient boosting* e XGBoost. O SVM foi testado com quatro núcleos diferentes que foram avaliados de maneira distinta. Os algoritmos como *Random Forest*, XGBoost e *ada booster* que combinam o resultado de classificadores mais simples foram os que obtiveram melhor resultado.

No presente trabalho o objetivo é a construção de um sistema de detecção em tempo real. Dessa forma, empenhou-se em construir um modelo com capacidade de detecção próxima ao que é encontrado na literatura. No entanto, o principal diferencial deste trabalho é a prioridade em reduzir o tempo de detecção. Desta forma, o impacto na experiência do usuário final seria reduzido.

3. Metodologia de solução

Nesta seção é apresentada a metodologia para a solução proposta. Na Subseção 3.1 é apresentada a metodologia utilizada na coleta dos dados usados para treinamento. Os atributos selecionados podem ser vistos na Subseção 3.2. Por fim, os algoritmos e os parâmetros utilizados são exibidos na Subseção 3.3

3.1. Coleta e preparo de dados

Para a obtenção dos atributos necessários para o treinamento dos algoritmos, foi necessário a obtenção de uma lista de URLs maliciosas e legítimas. As URLs legítimas foram obtidas da lista de *sites* mais populares disponível em MOZ. Já os sites maliciosos foram obtidos a partir de duas *blacklists*: OpenPhish e PhishTank. Como os domínios maliciosos ficam por pouco tempo online, as buscas foram realizadas ao longo de um mês com o propósito de obter uma maior quantidade de dados para treinamento.

Com o objetivo de extrair os atributos da lista de URLs, foi criado um *script* na linguagem Python. Para obter informações relacionadas ao domínio foram utilizadas as bibliotecas python-whois e dnspython. Já para obter dados de *page rank* usou-se a API Open Page Rank. Os atributos relacionados ao código HTML da página foram extraídos com o auxílio da biblioteca Beautiful Soup.

Com os dados obtidos pelo *script* foi gerado um arquivo CSV. No total, foram extraídos dados de 990 URLs sendo 490 legítimas e 500 maliciosas. Sites que estavam presentes nas listas mas não estavam acessíveis foram desconsiderados.

3.2. Seleção de atributos

Os atributos utilizados para detecção de *phishing* foram escolhidos a partir de trabalhos similares. Atributos relacionados ao conteúdo HTML, texto da URL e

informações de serviços externos foram selecionados de outros trabalhos de detecção de *phishing* por heurísticas ou aprendizado de máquina como em [Rao and Pais 2019], [Shahrivari et al. 2020] e [Zhang et al. 2007]. Também foram selecionados atributos utilizados para detecção de redes *fast-flux* como os utilizados em [Stalmans and Irwin 2011] e [Nagunwa et al. 2022].

Como a proposta do trabalho é desenvolver um algoritmo que funcione em tempo real, nem todos os atributos foram utilizados. A seleção de atributos foi feita de maneira manual buscando-se uma alta pontuação e minimizando o tempo de detecção. Procurou-se eleger o mínimo de atributos que dependam de serviços externos, já que a comunicação via Internet aumenta o tempo de detecção e prejudicaria a experiência do usuário. Os atributos inicialmente escolhidos foram:

- **A1 - Comprimento da URL:** Valor numérico indicando o tamanho da URL. Atacantes tendem a usar endereços maiores com o propósito de esconder partes suspeitas da URL.
- **A2 - Contém endereço IP:** Valor booleano que indica se a URL é um endereço IP. A grande maioria dos sites legítimos não utiliza IP na URL, logo a presença do endereço pode indicar um ataque.
- **A3 - URL possui símbolo "@":** Valor booleano que indica se o símbolo "@"está presente no texto da URL. Navegadores de Internet ignoram o texto que precede "@"na URL, logo atacantes podem colocar um domínio legítimo nesta parte com o objetivo de enganar o usuário.
- **A4 - Número de pontos na URL:** Valor numérico indicando a contagem de pontos no texto da URL. Atacantes podem utilizar um endereço aparentemente legítimo como subdomínio para enganar usuários, implicando em um número maior de pontos na URL.
- **A5 - Domínio possui hífen:** Valor booleano que indica a presença de hífen no nome de domínio. Atacantes podem adicionar prefixos ou sufixos separados por hífen ao domínio a fim de enganar as vítimas.
- **A6 - URL possui o texto "https":** Valor boolano que indica a presença da *string* "https"no texto da URL. A presença desse texto no corpo da URL pode confundir usuários mais leigos, fazendo com que pensem estar em uma conexão segura.
- **A7 - Redirecionamento na URL:** Valor booleano que indica a presença da *string* "//"no texto da URL. Este símbolo indica que o usuário será redirecionado para outro site.
- **A8 - Utiliza HTTPS:** Valor booleano que indica conexão com SSL (*Secure Socket Layer*). Atacantes podem falsificar conexões HTTPS mas dificilmente um site legítimo não a possuirá.
- **A9 - *Page Rank*:** Valor numérico representando a pontuação da página em mecanismos de busca. Endereços legítimos possuirão pontuação mais alta que os maliciosos
- **A10 - Idade do domínio:** Valor numérico contendo o a idade do domínio em milisegundos. Como campanhas de *phishing* tem curta duração, a idade dos domínios tende a ser menor que a de sites legítimos.
- **A11 - Possui registro DNS:** Valor booleano que indica a presença de um registro DNS para o domínio.

- **A12 - Número de registros A:** Valor numérico que representa a contagem de registros IPv4 para um determinado domínio. Em redes *fast-flux*, o número de IPs associados tende a ser maior
- **A13 - Número de registros AAAA:** Valor numérico que representa a contagem de registros IPv6 para um determinado domínio.
- **A14 - TTL médio dos registros A:** Valor numérico que representa o tempo de vida de um registro IPv4 em segundos. Em redes *fast-flux*, como a associação entre IP e nome de domínio muda frequentemente, o TTL tende a ser menor.
- **A15 - TTL médio dos registros AAAA:** Valor numérico que representa o tempo de vida de um registro IPv6 em segundos.
- **A16 - Número de ASNs:** Valor numérico que representa o contagem de sistemas autônomos nos endereços IP associados ao domínio. Em redes *fast-flux* as máquinas geralmente estão separadas geograficamente resultando em um número maior de ASNs (*Autonomous System Number*).
- **A17 - Possui *iframe*:** Valor booleano que indica a presença da *tag iframe* no código HTML da página. O *iframe* pode ser utilizado para exibir conteúdo de outra página dentro do site atual.
- **A18 - Domínio mais frequente em *anchor*:** Valor booleano que indica se os links mais frequentes na *tag anchor* são o próprio domínio do site. A fim de economizar trabalho ao criar o site, atacantes podem manter links similares aos da página legítima
- **A19 - Domínio mais frequente em *link*:** Valor booleano que indica se os links mais frequentes na *tag link* são o próprio domínio do site.
- **A20 - Domínio mais frequente em *source*:** Valor booleano que indica se os links mais frequentes na *tag source* são o próprio domínio do site.
- **A21 - Taxa de páginas comuns:** Valor numérico que representa a razão da frequência do link mais comum em relação ao total de links na página. A fim de minimizar o esforço de imitar os links de uma página real, atacantes podem utilizar uma única URL.
- **A22 - Taxa de páginas comuns no *footer*:** Valor numérico que representa a razão da frequência do link mais comum em relação ao total de links dentro da tag *footer*.
- **A23 - Taxa de links nulos:** Valor numérico que indica a representa de links que iniciam com o símbolo "#", ou seja, que apontam para a própria página, em relação ao total de links na página. Atacantes tem a intenção de manter o usuário no site, evitando links que levem para outros sites.
- **A24 - Taxa de links nulos no *footer*:** Valor numérico que representa a taxa de links que iniciam com o símbolo "#"em relação ao total de links dentro da tag *footer*.
- **A25 - Possui *anchor*:** Valor booleano que indica a presença da tag *anchor* no corpo da página. Atacantes podem substituir links da página original por imagens, removendo as tags *anchor* do corpo do documento HTML.

Durante os testes realizados, alguns atributos foram descartados. Foi notado que para o conjunto de dados utilizado os atributos A16 e A20 possuem variância nula, ou seja, o mesmo valor para todas as instâncias e, por esse motivo, não foram utilizados no treinamento. Além disso, foi feita uma análise de correlação entre as variáveis como

mostra a Figura 1. Como pode ser observado as variáveis A2 e A11 possuem uma relação linear negativa perfeita e, por isso, optou-se por descartar o atributo A11. Como resultado, 22 atributos foram utilizados no treinamento.

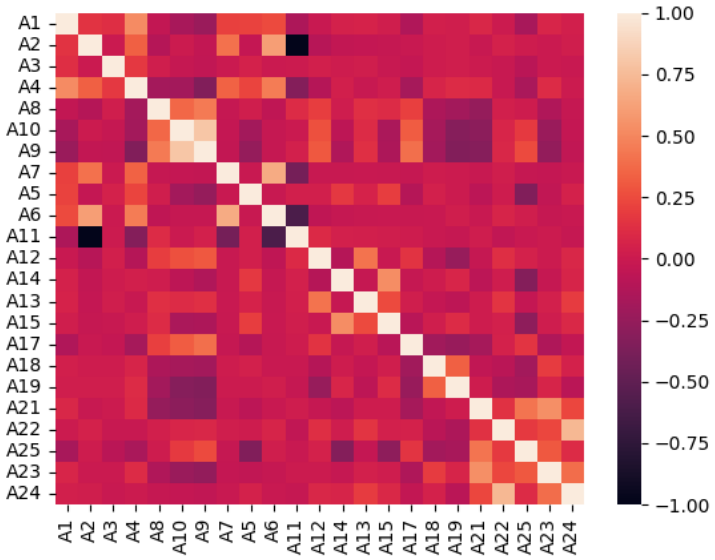

Figura 1. Correlação entre os atributos.

3.3. Escolha dos algoritmos e parâmetros

Para o treinamento, foram utilizados 6 algoritmos de aprendizado supervisionado: MLP, *Naive Bayes*, KNN, Árvore de decisão, *Random Forest* e SVM. Todos os modelos foram implementados através da biblioteca scikit-learn [Pedregosa et al. 2011] em linguagem Python.

Os parâmetros para cada modelo foram escolhidos empiricamente de modo a aumentar o desempenho da classificação. Os valores utilizados para cada um são resumidos na Tabela 2. Os parâmetros com valores nulos ou do tipo booleano falso não são exibidos.

O algoritmo *Naive Bayes* foi utilizada em sua versão gaussiana para variáveis contínuas. Para obter melhor desempenho, foi feita uma redução de dimensão dos atributos através de PCA (*Principal Component Analysis*) para 15 componentes. Mesmo com um número relativamente baixo de variáveis, o desempenho é aumentado pois as variáveis se tornam independentes após realização do PCA, o que se aproxima mais da premissa do *Naive Bayes*. Como não foram obtidos ganhos significativos em outros algoritmos, o PCA foi utilizado apenas no *Naive Bayes*.

O conjunto de dados utilizado apresenta dados com dimensões muito distintas. Atributos como a idade do domínio A10 estão inseridos em um intervalo muito grande enquanto outros como a taxa de links nulos A23 variam apenas de 0 a 1. Como alguns algoritmos como o KNN não apresentam bom funcionamento para escalas muito distintas, foi feita a normalização dos dados, de modo que todos os valores estejam inseridos no intervalo de 0 a 1.

Tabela 2. Parâmetros utilizados no aprendizado dos algoritmos.

Algoritmo	Parâmetros
MLP	Solver: Gradiente descendente estocástico Função de ativação: tangente hiperbólica Taxa de aprendizado: 0,075 Neurônios por camada: 25, 40, 10 Percentual de validação: 20% Máximo de iterações: 500
SVM	Kernel: rbf Coeficiente do Kernel: scale Tolerância para critério de parada: 0,001 Forma da função de decisão: one-vs-rest
KNN	Número de vizinhos: 3 Função de pesos: uniforme Distância: euclidiana Algoritmo: decisão automática
Árvore de decisão	Critério: gini Divisor: *best* Mínimo de amostras para divisão: 2 Mínimo de amostras para nó folha: 1
Random Forest	Número de árvores: 100 Critério: gini Mínimo de amostras para divisão: 2 Mínimo de amostras para nó folha: 1 Número de atributos considerados para divisão: *sqrt*

A quantidade de dados utilizada para treinamento foi relativamente pequena se comparada com outros trabalhos. Em [Shahrivari et al. 2020], por exemplo, foram utilizadas, ao todo, 11055 amostras para treinamento contra 990 utilizadas neste trabalho. Visando contornar possíveis erros de predição causados por esse fato, utilizou-se validação cruzada *k-fold* [Berrar et al. 2019]. Esta técnica consiste em dividir, de maneira aleatória, o conjunto de dados original em k subconjuntos, onde um destes é usado para validação e o restante para treinamento. O processo é repetido k vezes alterando-se o subconjunto utilizado para validação.

4. Resultados

Os resultados obtidos com os experimentos são apresentados nesta Seção. Na Subseção 4.1 são explicados os critérios de avaliação utilizados para aferir o desempenho dos modelos. A pontuação final de cada algoritmo é mostrada na Subseção 4.2.

4.1. Critérios de avaliação

Para a classificação, foi considerado site de *phishing* como valor positivo e site legítimo como negativo. Portanto verdadeiros positivos (VP) indicam sites classificados como *phishing* corretamente e verdadeiros negativos (VN) sites legítimos classificados correta-

mente. Já os falsos positivos (FP) representam sites legítimos classificados erroneamente e falsos negativos (FN), sites maliciosos classificados de maneira incorreta.

Para aferir o desempenho dos modelos na classificação foram utilizadas quatro métricas: acurácia, precisão, *recall* e F1. A acurácia (Equação 1) representa a taxa de classificações corretas em relação ao total. A precisão (Equação 2) indica a habilidade do classificador em não identificar como positiva uma amostra negativa. Já o *recall* (Equação 3) mede a capacidade do modelo em encontrar todas as amostras positivas. Por fim, a pontuação F1 (Equação 4) pode ser interpretada como a média harmônica ponderada entre precisão e *recall*. Todas as medidas variam no intervalo de 0 a 1, sendo 1 o melhor valor e 0 o pior.

$$ac = \frac{VP + VN}{VP + FP + VN + FN} \tag{1}$$

$$p = \frac{VP}{VP + FP} \tag{2}$$

$$r = \frac{VP}{VP + FN} \tag{3}$$

$$F1 = \frac{2pr}{p + r} \tag{4}$$

4.2. Análise dos resultados

Os resultados para cada modelo são resumidos na Tabela 3. Os valores mostrados foram obtidos a partir da média dos resultados gerados por validação cruzada com 5 subconjuntos. Desta forma, 80% dos dados foram utilizados para treinamento e 20% para teste. Como pode ser observado, o algoritmo SVM obteve o melhor resultado em todos os critérios de avaliação. Por outro lado, o Naive Bayes obteve o pior desempenho em quase todos os critérios mesmo com a utilização do PCA.

Os gráficos da Figura 2 mostram os resultados com um intervalo de confiança. Como pode ser observado, ocorreu um maior desvio padrão com o *Naive Bayes*, superando 3 pontos percentuais no caso da precisão. Para os demais algoritmos o intervalo de confiança foi menor, com o desvio padrão não chegando a superar 1,7 pontos percentuais.

Tabela 3. Resultados.

Algoritmo	Acurácia	Precisão	*Recall*	F1
MLP	0,9859	0,9900	0,9820	0,9859
Naive Bayes	0,9414	0,9295	0,9580	0,9431
KNN	0,9697	0,9858	0,9540	0,9695
Árvore de decisão	0,9768	0,9765	0,9780	0,9771
Random Forest	0,9859	0,9884	0,9840	0,9860
SVM	0,9919	0,9980	0,9860	0,9919

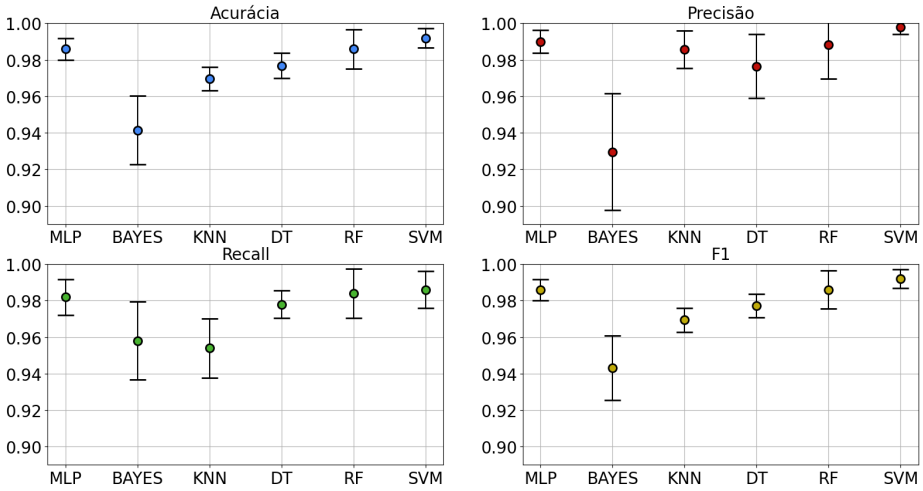

Figura 2. Resultados com intervalo de confiança.

O primeiro teste foi realizado com todos os 22 atributos. No entanto, o tempo médio para obter todos os dados a partir de uma URL foi de 6,13 segundos. Para uma aplicação em tempo real este tempo poderia comprometer a experiência do usuário ao navegar pela Internet. Por esse motivo, foram realizados novos testes considerando apenas atributos que não dependam de serviços externos. Dessa forma, os atributos A9, A10, A12, A13, A14, A15 e A16 não foram utilizados. Os novos resultados são exibidos na Tabela 4

Como pode ser observado nos gráficos da Figura 3, além de um menor desempenho médio, o intervalo de confiança também foi aumentado. A maioria dos desvios superou 2 pontos percentuais, chegando a 7 pontos percentuais para o Recall com o Naive Bayes.

Tabela 4. Resultados para o conjunto de atributos reduzido.

Algoritmo	Acurácia	Precisão	*Recall*	F1
MLP	0,9303	0,9354	0,9260	0,9306
Naive Bayes	0,8758	0,9310	0,8140	0,8672
KNN	0,9182	0,9544	0,8800	0,9149
Árvore de decisão	0,9071	0,9122	0,9060	0,9075
Random Forest	0,9424	0,9550	0,9329	0,9429
SVM	0,9232	0,9241	0,9240	0,9239

O algoritmo *Random Forest* desta vez obteve o melhor desempenho enquanto o *Naive Bayes* se manteve com a pior pontuação. O número de componentes do PCA foi reduzido para 5. Com o novo conjunto de atributos, o tempo para obtenção dos dados foi reduzido para 1,34 segundos em média.

Os experimentos mostraram que os modelos obtiveram bons resultados se comparados com outros trabalhos que também utilizam aprendizado de máquina. Em [Nagunwa et al. 2022], o melhor modelo obteve acurácia de 98.42% enquanto neste tra-

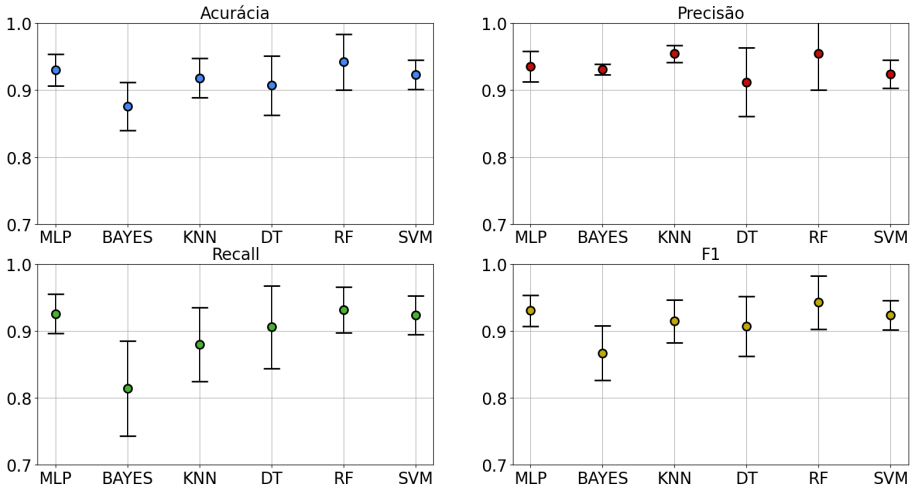

Figura 3. Resultados com intervalo de confiança para atributos reduzidos.

balho a melhor acurácia atingida foi de 99,19%. A pontuação F1 também foi similar com 0,99 em [Nagunwa et al. 2022] contra 0,9919 obtidos no presente trabalho.

Já em [Rao and Pais 2019] o melhor resultado foi obtido através do algoritmo *Random Forest*, porém com uma acurácia mais alta de 99,55%. O mesmo algoritmo obteve uma precisão de 99,18% enquanto que neste trabalho a melhor precisão foi atingida pelo SVM com o valor de 99,80%.

Além disso, o tempo para obtenção dos atributos foi reduzido significativamente. Em [Nagunwa et al. 2022] foram necessários 162,64 segundos contra os 6,13 segundos conseguidos com o modelo proposto. Caso os atributos externos sejam desconsiderados, o tempo é reduzido para 1,34 segundos porém com uma penalização nas métricas de desempenho de classificação.

Em [Nielsen 1994], o autor estabelece 10 segundos como tempo limite de carregamento para que a atenção do usuário seja mantida. No caso da utilização de todos os atributos, cerca de 60% deste tempo estaria comprometido apenas com o algoritmo de classificação. A aplicação em tempo real portanto seria mais factível com o conjunto de atributos reduzido.

5. Conclusão

Neste trabalho, foi desenvolvida uma proposta de solução para detecção de ataques de *phishing*. Os algoritmos empregados atingiram pontuações semelhantes a de outros trabalhos similares com um relativo baixo tempo de detecção.

Em trabalhos futuros, seria válido testar o modelo em conjuntos de dados maiores para verificar a variação na pontuação final. Além disso, seria interessante realizar testes com outros modelos de aprendizado de máquina a fim de obter possíveis melhores resultados.

Com o objetivo de determinar a efetividade da detecção em tempo real proposta, seria importante também utilizar o modelo em uma aplicação real como uma extensão

para navegadores de Internet. Dessa forma, poderia-se aferir como o tempo de detecção e classificação impactaria a experiência do usuário.

No presente trabalho a seleção de atributos foi feita de maneira manual. Por esse motivo não é garantido que a escolha de atributos tenha sido ótima. Em trabalhos futuros, poderia-se implementar uma solução automática e mais eficiente utilizando algoritmos de seleção de atributos.

Referências

Berrar, D. et al. (2019). Cross-validation.

Khonji, M., Iraqi, Y., and Jones, A. (2013). Phishing detection: a literature survey. *IEEE Communications Surveys & Tutorials*, 15(4):2091–2121.

Nagunwa, T., Kearney, P., and Fouad, S. (2022). A machine learning approach for detecting fast flux phishing hostnames. *Journal of Information Security and Applications*, 65:103125.

Nielsen, J. (1994). *Usability engineering*. Morgan Kaufmann.

Ozimek, A. (2020). The future of remote work. *Available at SSRN 3638597*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pranggono, B. and Arabo, A. (2021). Covid-19 pandemic cybersecurity issues. *Internet Technology Letters*, 4(2):e247.

Rao, R. S. and Pais, A. R. (2019). Detection of phishing websites using an efficient feature-based machine learning framework. *Neural Computing and Applications*, 31:3851–3873.

Sahingoz, O. K., Buber, E., Demir, O., and Diri, B. (2019). Machine learning based phishing detection from urls. *Expert Systems with Applications*, 117:345–357.

Shahrivari, V., Darabi, M. M., and Izadi, M. (2020). Phishing detection using machine learning techniques. *arXiv preprint arXiv:2009.11116*.

Stalmans, E. and Irwin, B. (2011). A framework for dns based detection and mitigation of malware infections on a network. In *2011 Information Security for South Africa*, pages 1–8. IEEE.

Zhang, Y., Hong, J. I., and Cranor, L. F. (2007). Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*, pages 639–648.